Structural Equation Modeling with lavaan

Yves Rosseel
Department of Data Analysis
Ghent University

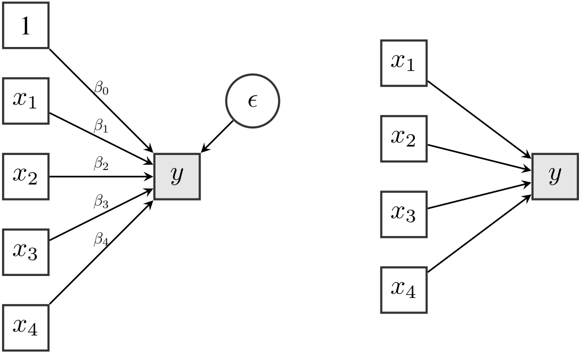
Zürich R Courses
28 + 29 March 2019

Contents

1 Introduction to SEM

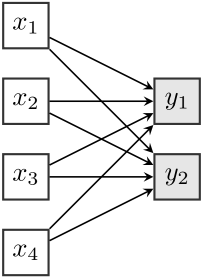
1.1 From regression to structural equation modeling

univariate linear regression



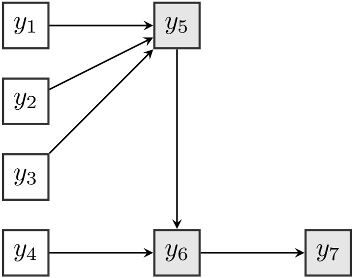
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

multivariate regression



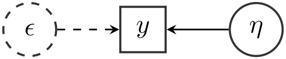
path analysis

- testing models of *causal* relationships among observed variables
- all variables are observed (manifest)
- system of regression equations

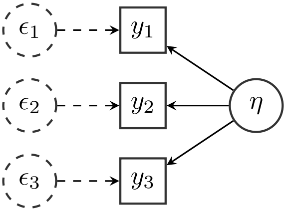


measurement error

- in the social sciences, observed variables are not without measurement error
- single indicator measurement model

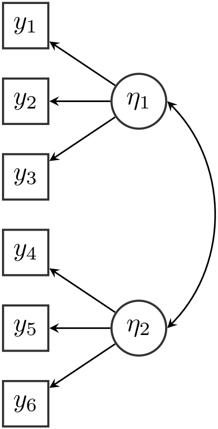


- multiple indicator measurement model



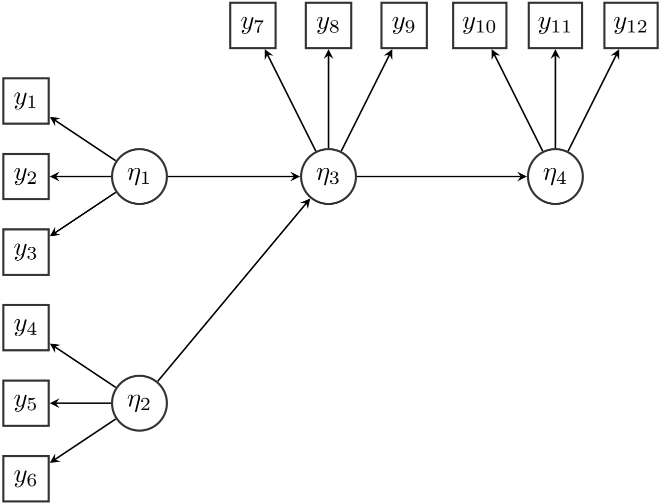
confirmatory factor analysis (CFA)

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



structural equation modeling (SEM)

- path analysis with latent variables

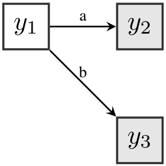


1.2 The model-implied covariance matrix (the essence of SEM)

- the goal of SEM is to test an a priori specified theory (which often can be depicted as 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:
  - measurement model (relationship latent variables and indicators)
  - structural equations (regressions among latent/observed variables)
- each diagram has 'model-based' implications
  - for the model-implied covariance matrix: $\hat{\boldsymbol{\Sigma}}$
  - for the model-implied mean vector: $\hat{\boldsymbol{\mu}}$
  - ...
- different diagrams lead to (potentially) different implications; some implications may not fit with your data

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:



- the two corresponding linear equations are:

$$\begin{cases} y_2 = a\, y_1 + \epsilon_2 \\ y_3 = b\, y_1 + \epsilon_3 \end{cases}$$

- the *model-implied* variance covariance matrix $\hat{\mathbf{\Sigma}}$:

$$\begin{bmatrix} \sigma^2(y_1) & & \\ \sigma(y_2, y_1) & \sigma^2(y_2) & \\ \sigma(y_3, y_1) & \sigma(y_3, y_2) & \sigma^2(y_3) \end{bmatrix}$$

- the five parameters of our model are:
  - the regression coefficients a and b
  - the (plain) variance of $\sigma^2(y_1)$
  - the residual variances $\sigma^2(\epsilon_2)$ and $\sigma^2(\epsilon_3)$
- given specific (estimated) values for these five parameters, how can we construct the model-implied variance/covariance matrix?

rules about variances and covariances (1)

- suppose X and Y are random variables, and a and b are constants.
- some simple rules for variances:
  - $\sigma^2(a) = 0$
  - $\sigma^2(a + X) = \sigma^2(X)$
  - $\sigma^2(aX) = a^2\,\sigma^2(X)$
  - $\sigma^2(X + Y) = \sigma^2(X) + \sigma^2(Y) + 2\,\sigma(X, Y)$
- some simple rules for covariances:
  - $\sigma(a, b) = 0$
  - $\sigma(a, X) = 0$
  - $\sigma(X, Y) = \sigma(Y, X)$
  - $\sigma(X + a, Y + b) = \sigma(X, Y)$
  - $\sigma(aX, bY) = a\,b\,\sigma(X, Y)$

rules about variances and covariances (2)

- given two linear combinations X and Y:

$$X = a_1 X_1 + a_2 X_2 + \ldots + a_p X_p \quad \text{and} \quad Y = b_1 Y_1 + b_2 Y_2 + \ldots + b_q Y_q$$

- the general formula for the variance of a linear combination is given by

$$\begin{aligned} \sigma^2(X) &= \sum_{i=1}^{p} \sum_{j=1}^{p} a_i a_j \sigma(X_i, X_j) \\ &= \sum_{i=1}^{p} a_i^2 \sigma^2(X_i) + \sum_{i=1}^{p} \sum_{j \neq i}^{p} a_i a_j \sigma(X_i, X_j) \end{aligned}$$

- the covariance between these two linear combinations is given by

$$\sigma(X, Y) = \sum_{i=1}^{p} \sum_{j=1}^{q} a_i b_j \sigma(X_i, Y_j)$$

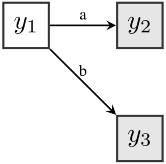
applying the rules

- following the rules for the covariances, we find:
  - $\sigma(y_2, y_3) = a\, b\, \sigma(y_1, y_1) + a\, \sigma(y_1, \epsilon_3) + b\, \sigma(y_1, \epsilon_2) + \sigma(\epsilon_2, \epsilon_3)$
  - $\sigma(y_2, y_1) = a\, \sigma(y_1, y_1) + \sigma(y_1, \epsilon_2)$ and $\sigma(y_3, y_1) = b\, \sigma(y_1, y_1) + \sigma(y_1, \epsilon_3)$
  - but $\sigma(y_1, y_1) = \sigma^2(y_1)$, $\sigma(y_1, \epsilon_3) = 0$, $\sigma(y_1, \epsilon_2) = 0$, and we also assume (here) that $\sigma(\epsilon_2, \epsilon_3) = 0$
- following the rules for variances, we find:
  - $\sigma^2(y_1) = \sigma^2(y_1)$
  - $\sigma^2(y_2) = a^2 \sigma^2(y_1) + \sigma^2(\epsilon_2)$ and $\sigma^2(y_3) = b^2 \sigma^2(y_1) + \sigma^2(\epsilon_3)$
- the *model-implied* variance covariance matrix for our two equations is

$$\begin{bmatrix} \sigma^2(y_1) & & \\ a\, \sigma^2(y_1) & a^2\, \sigma^2(y_1) + \sigma^2(\epsilon_2) & \\ b\, \sigma^2(y_1) & a\, b\, \sigma^2(y_1) & b^2\, \sigma^2(y_1) + \sigma^2(\epsilon_3) \end{bmatrix}$$

the model-implied covariance matrix for our two-equation model

- for example, if $a = 3$ and $b = 5$, $\sigma^2(y_1) = 10$, $\sigma^2(\epsilon_2) = 20$ and $\sigma^2(\epsilon_3) = 30$, then for this model:

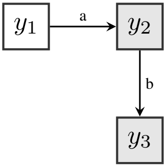


we find

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

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:



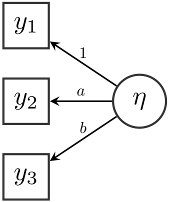
we find

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

- 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':



- the model-implied covariance is again a function of the model parameters:

$$\begin{bmatrix} \lambda_1^2\sigma^2(\eta_1) + \sigma^2(\epsilon_1) & & \\ \lambda_1\lambda_2\sigma^2(\eta_1) & \lambda_2^2\sigma^2(\eta_1) + \sigma^2(\epsilon_2) & \\ \lambda_1\lambda_3\sigma^2(\eta_1) & \lambda_2\lambda_3\sigma^2(\eta_1) & \lambda_3^2\sigma^2(\eta_1) + \sigma^2(\epsilon_3) \end{bmatrix}$$

where we have assumed that the ϵ's are uncorrelated

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

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

summary

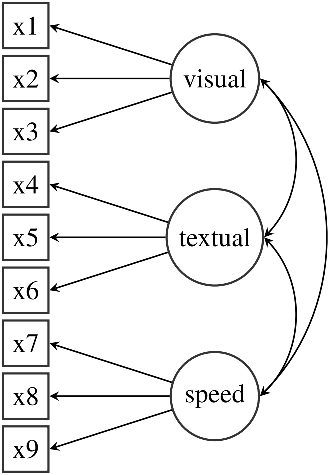
- in general: different models produce different model-implied covariance matrices
- computation of these model-implied variances and covariances is straightforward but tedious
- that is why we will translate our model into a matrix representation

1.3 Matrix representation in a CFA model

classic example CFA

- well-known dataset; based on Holzinger & Swineford (1939) data
- also analyzed by Jöreskog (1969)
- 9 observed ‘indicators’ measuring three ‘latent’ 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
- N=301
- we assume the three factors are correlated

diagram of the model



data

	x1	x2	x3	x4	x5	x6	x7	x8	x9
1	3.3333333	7.75	0.375	2.3333333	5.75	1.2857143	3.391304	5.75	6.361111
2	5.3333333	5.25	2.125	1.6666667	3.00	1.2857143	3.782609	6.25	7.916667
3	4.5000000	5.25	1.875	1.0000000	1.75	0.4285714	3.260870	3.90	4.416667
4	5.3333333	7.75	3.000	2.6666667	4.50	2.4285714	3.000000	5.30	4.861111
5	4.8333333	4.75	0.875	2.6666667	4.00	2.5714286	3.695652	6.30	5.916667
6	5.3333333	5.00	2.250	1.0000000	3.00	0.8571429	4.347826	6.65	7.500000
7	2.8333333	6.00	1.000	3.3333333	6.00	2.8571429	4.695652	6.20	4.861111
8	5.6666667	6.25	1.875	3.6666667	4.25	1.2857143	3.391304	5.15	3.666667
9	4.5000000	5.75	1.500	2.6666667	5.75	2.7142857	4.521739	4.65	7.361111
10	3.5000000	5.25	0.750	2.6666667	5.00	2.5714286	4.130435	4.55	4.361111
11	3.6666667	5.75	2.000	2.0000000	3.50	1.5714286	3.739130	5.70	4.305556
12	5.8333333	6.00	2.875	2.6666667	4.50	2.7142857	3.695652	5.15	4.138889
13	5.6666667	4.50	4.125	2.6666667	4.00	2.2857143	5.869565	5.20	5.861111
14	6.0000000	5.50	1.750	4.6666667	4.00	1.5714286	5.130435	4.70	4.444444
15	5.8333333	5.75	3.625	5.0000000	5.50	3.0000000	4.000000	4.35	5.861111
16	4.6666667	4.75	2.375	2.6666667	4.25	0.7142857	4.086957	3.80	5.138889
...									
301	4.3333333	6.00	3.375	3.6666667	5.75	3.1428571	4.086957	6.95	5.166667

- data is complete
- under normality, the data can be summarized by the covariance matrix $(\mathbf{S})$ and the mean vector $(\mathbf{m})$

observed covariance matrix: $\mathbf{S}$

- p is the number of observed variables: $p = 9$
- observed covariance matrix (elements divided by N-1):

```
     x1    x2   x3   x4   x5   x6    x7   x8   x9
x1 1.36
x2 0.41  1.38
x3 0.58  0.45 1.28
x4 0.51  0.21 0.21 1.35
x5 0.44  0.21 0.11 1.10 1.66
x6 0.46  0.25 0.24 0.90 1.01 1.20
x7 0.09 -0.10 0.09 0.22 0.14 0.14  1.18
x8 0.26  0.11 0.21 0.13 0.18 0.17  0.54 1.02
x9 0.46  0.24 0.37 0.24 0.30 0.24  0.37 0.46 1.02
```

- we want to ‘explain’ the observed correlations/covariances by postulating a number of latent variables (factors) and a corresponding factor structure
- we will ‘rewrite’ the $p(p+1)/2 = 45$ elements in the covariance matrix as a function a smaller number of ‘free parameters’ in the CFA model, summarized in a number of (typically sparse) matrices

the standard CFA model: matrix representation

- the classic LISREL representation uses three matrices (for 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:

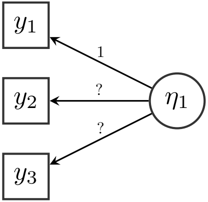
$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Psi}\mathbf{\Lambda}' + \mathbf{\Theta}$$

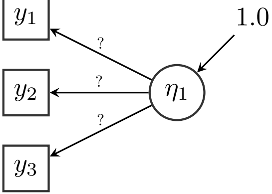
- all parameters are included in three model matrices
- simple matrix multiplication (and addition) gives us the model implied covariance matrix
- for identification purposes, some parameters need to be fixed to a constant
- estimation problem: choose the 'free' parameters, so that the estimated implied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

setting the metric of the latent variables: UVI of ULI

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

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





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

observed covariance matrix

	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

model-implied covariance matrix

	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

1.4 The implied covariance matrix for the full SEM model

- in the LISREL representation, we need an additional matrix ($\mathbf{B}$):

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

  where $\mathbf{B}$ summarizes the regressions among the latent variables

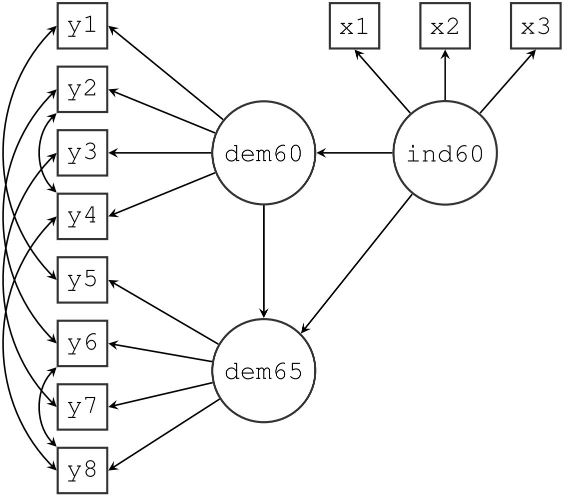
- we need this extended model for
  - second-order CFA
  - MIMIC models
  - SEM models
- in LISREL parlance, this the 'all-y' model

example: Political Democracy

- Industrialization and Political Democracy dataset (N=75)
- This dataset is used throughout Bollen's 1989 book (see pages 12, 17, 36 in chapter 2, pages 228 and following in chapter 7, pages 321 and following in chapter 8).
- The dataset contains various measures of political democracy and industrialization in developing countries:

```
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
```

model diagram



selection of the output

```
                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  ind60 =~
    x1                1.000                               0.670    0.920
    x2                2.180    0.139   15.742    0.000    1.460    0.973
    x3                1.819    0.152   11.967    0.000    1.218    0.872
  dem60 =~
    y1                1.000                               2.223    0.850
    y2                1.257    0.182    6.889    0.000    2.794    0.717
    y3                1.058    0.151    6.987    0.000    2.351    0.722
    y4                1.265    0.145    8.722    0.000    2.812    0.846
  dem65 =~
    y5                1.000                               2.103    0.808
    y6                1.186    0.169    7.024    0.000    2.493    0.746
    y7                1.280    0.160    8.002    0.000    2.691    0.824
    y8                1.266    0.158    8.007    0.000    2.662    0.828

Regressions:
  dem60 ~
    ind60             1.483    0.399    3.715    0.000    0.447    0.447
  dem65 ~
    ind60             0.572    0.221    2.586    0.010    0.182    0.182
    dem60             0.837    0.098    8.514    0.000    0.885    0.885
...
```

1.5 Model parameters and model matrices

31 ‘free’ model parameters

```
> coef(fit)

     ind60=~x2      ind60=~x3      dem60=~y2      dem60=~y3      dem60=~y4      dem65=~y6
         2.180          1.819          1.257          1.058          1.265          1.186
     dem65=~y7      dem65=~y8    dem60~ind60    dem65~ind60    dem65~dem60        y1~~y5
         1.280          1.266          1.483          0.572          0.837          0.624
        y2~~y4         y2~~y6         y3~~y7         y4~~y8         y6~~y8        x1~~x1
         1.313          2.153          0.795          0.348          1.356          0.082
        x2~~x2         x3~~x3         y1~~y1         y2~~y2         y3~~y3        y4~~y4
         0.120          0.467          1.891          7.373          5.067          3.148
        y5~~y5         y6~~y6         y7~~y7         y8~~y8   ind60~~ind60   dem60~~dem60
         2.351          4.954          3.431          3.254          0.448          3.956
  dem65~~dem65
         0.172
```

model matrices: free parameters

```
> inspect(fit)

$lambda
   ind60 dem60 dem65
x1     0     0     0
x2     1     0     0
x3     2     0     0
y1     0     0     0
y2     0     3     0
y3     0     4     0
y4     0     5     0
y5     0     0     0
y6     0     0     6
y7     0     0     7
y8     0     0     8

$theta
   x1 x2 x3 y1 y2 y3 y4 y5 y6 y7 y8
x1 18
x2  0 19
x3  0  0 20
y1  0  0  0 21
y2  0  0  0  0 22
y3  0  0  0  0  0 23
y4  0  0  0  0 13  0 24
y5  0  0  0 12  0  0  0 25
```

```
y6   0   0   0   0 14   0   0   0 26
y7   0   0   0   0   0 15   0   0   0 27
y8   0   0   0   0   0   0 16   0 17   0 28

$psi
      ind60 dem60 dem65
ind60 29
dem60  0    30
dem65  0     0    31

$beta
      ind60 dem60 dem65
ind60     0     0     0
dem60     9     0     0
dem65    10    11     0
```

model matrices: estimated values

```
> inspect(fit, "est")

$lambda
   ind60 dem60 dem65
x1 1.000 0.000 0.000
x2 2.180 0.000 0.000
x3 1.819 0.000 0.000
y1 0.000 1.000 0.000
y2 0.000 1.257 0.000
y3 0.000 1.058 0.000
y4 0.000 1.265 0.000
y5 0.000 0.000 1.000
y6 0.000 0.000 1.186
y7 0.000 0.000 1.280
y8 0.000 0.000 1.266

$theta
   x1    x2    x3    y1    y2    y3    y4    y5    y6    y7    y8
x1 0.082
x2 0.000 0.120
x3 0.000 0.000 0.467
y1 0.000 0.000 0.000 1.891
y2 0.000 0.000 0.000 0.000 7.373
y3 0.000 0.000 0.000 0.000 0.000 5.067
y4 0.000 0.000 0.000 0.000 1.313 0.000 3.148
y5 0.000 0.000 0.000 0.624 0.000 0.000 0.000 2.351
```

```
y6 0.000 0.000 0.000 0.000 2.153 0.000 0.000 0.000 4.954
y7 0.000 0.000 0.000 0.000 0.000 0.795 0.000 0.000 0.000 3.431
y8 0.000 0.000 0.000 0.000 0.000 0.000 0.348 0.000 1.356 0.000 3.254

$psi
      ind60 dem60 dem65
ind60 0.448
dem60 0.000 3.956
dem65 0.000 0.000 0.172

$beta
      ind60 dem60 dem65
ind60 0.000 0.000     0
dem60 1.483 0.000     0
dem65 0.572 0.837     0
```

manually computing the model-implied covariance matrix (optional)

```
# make the model matrices available in R's workspace
attach(inspect(fit, "est"))

# compute (I - B)^(-1)
IB <- diag(nrow(beta)) - beta
IB.inv <- solve(IB)

# compute the model-implied model matrix (using formula on slide 24)
Sigma.hat <- lambda %*% IB.inv %*% psi %*% t(IB.inv) %*% t(lambda) + theta

# print the matrix
round(Sigma.hat, 3)
```

```
      x1    x2    x3    y1     y2     y3     y4    y5     y6     y7     y8
x1 0.530 0.978 0.815 0.665  0.836  0.703  0.841 0.814  0.965  1.041  1.030
x2 0.978 2.252 1.778 1.450  1.822  1.534  1.834 1.774  2.103  2.270  2.245
x3 0.815 1.778 1.950 1.209  1.520  1.279  1.530 1.479  1.754  1.893  1.873
y1 0.665 1.450 1.209 6.834  6.211  5.228  6.251 5.143  5.358  5.782  5.721
y2 0.836 1.822 1.520 6.211 15.179  6.570  9.169 5.679  8.887  7.267  7.190
y3 0.703 1.534 1.279 5.228  6.570 10.597  6.612 4.780  5.667  6.911  6.051
y4 0.841 1.834 1.530 6.251  9.169  6.612 11.054 5.716  6.777  7.313  7.584
y5 0.814 1.774 1.479 5.143  5.679  4.780  5.716 6.773  5.243  5.658  5.598
y6 0.965 2.103 1.754 5.358  8.887  5.667  6.777 5.243 11.171  6.709  7.994
y7 1.041 2.270 1.893 5.782  7.267  6.911  7.313 5.658  6.709 10.671  7.163
y8 1.030 2.245 1.873 5.721  7.190  6.051  7.584 5.598  7.994  7.163 10.341
```

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. John Wiley & Sons.

Beaujean, A. A. (2014). Latent variable modeling using R: A step-by-step guide. 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:
  - confirmatory factor analysis: function `cfa()`
  - structural equation modeling: function `sem()`
  - general mean/covariance structure modeling: function `lavaan()`
  - support for continuous, binary and ordinal data
- under development, future plans:
  - multilevel SEM (0.6), mixture/latent-class SEM (0.7)
- 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

installing a development version of lavaan

- first method: type in R:

```
> install.packages("lavaan", repos = "http://www.da.ugent.be",
                   type = "source")
```

- second method, using the devtools package:

```
> library(devtools)
> install_github("yrosseel/lavaan")
```

- third method: if no internet, but you have a lavaan *.tar.gz file

```
> install.packages("c:/temp/lavaan_0.6-1.tar.gz", NULL, type = "source")
```

where you need to adapt the first string to point to the directory where the lavaan *.tar.gz file is located

the lavaan ecosystem

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

  Bayesian SEM (currently using jags) 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** (Sunthud Pornprasertmanit and many others)

  collection of useful functions for SEM

- **simsem** (Sunthud Pornprasertmanit 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

- **nlsem** (Nora Umbach and many others)

  esfunctiontimation of structural equation models with nonlinear effects and underlying nonnormal distributions

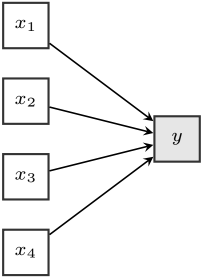
- many others

  bmem, coefficientalpha, eqs2lavaan, fSRM, influence.SEM, MIIVsem, 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:

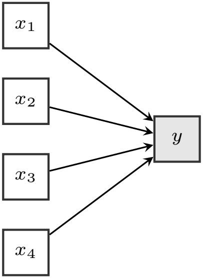
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

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 0.6-3 ended normally after 32 iterations

  Optimization method                           NLMINB
  Number of free parameters                          5

  Number of observations                           100

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0
  Minimum Function Value               0.0000000000000

Parameter Estimates:

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

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

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

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

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

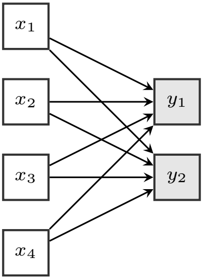
- if maximum likelihood is used, $\hat{\sigma}_y^2$ is computed by:

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

and this affects the standard errors.

the lavaan model syntax – multivariate regression

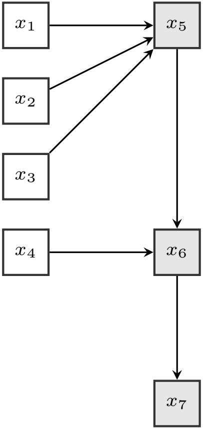
- for each dependent variable, we write a separate regression equation:




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

the lavaan model syntax – path analysis

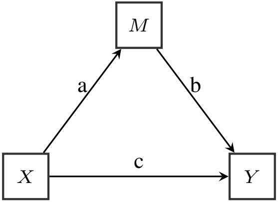
- for each dependent variable, we write a separate regression equation:




```
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

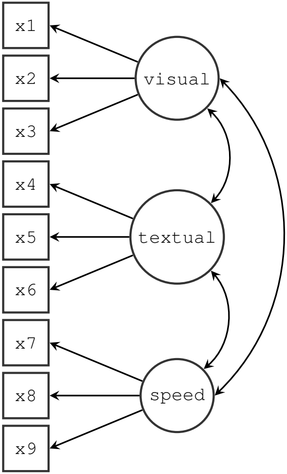



```
myModel <- '
              Y ~ b*M + c*X
              M ~ a*X

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

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

summary(fit)
```

partial output

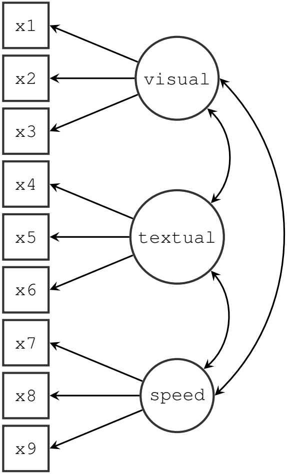
```
Parameter estimates:

  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

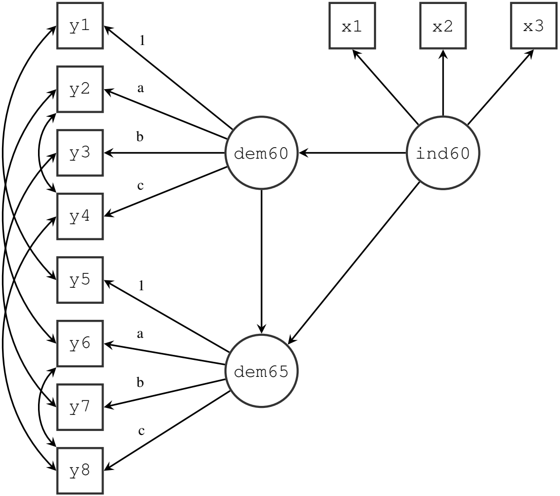
```
lavaan 0.6-3 ended normally after 35 iterations

  Optimization method                           NLMINB
  Number of free parameters                         21

  Number of observations                           301

  Estimator                                         ML
  Model Fit Test Statistic                      85.306
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000

Model test baseline model:

  Minimum Function 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

  Number of free parameters                         21
  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          0.071  0.114
  P-value RMSEA <= 0.05                          0.001

Standardized Root Mean Square Residual:

  SRMR                                           0.065

Parameter Estimates:

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

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 0.6-3 ended normally after 66 iterations

  Optimization method                           NLMINB
  Number of free parameters                         31
  Number of equality constraints                     3

  Number of observations                            75

  Estimator                                         ML
  Model Fit Test Statistic                      40.179
  Degrees of freedom                                38
  P-value (Chi-square)                           0.374

Parameter Estimates:

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

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

syntax: lhs op rhs

- each line in the model syntax is a 'formula' and contains three parts:
  - the left-hand side ('lhs')
  - the operator ('op')
  - the right-hand side ('rhs')
- examples:

```
someVar ~~ otherVar
```

- the '+' operator in a formula allows to collect formulas with the same lhs/rhs in a single formula; therefore

```
Y ~ A
Y ~ B
Y ~ C
```

is identical to

```
Y ~ A + B + C
```

overview operators in the lavaan model syntax

formula type	operator	mnemonic
latent variable	`=~`	is manifested by
regression	`~`	is regressed on
(residual) (co)variance	`~~`	is correlated with
intercept	`~ 1`	intercept
threshold	`| t1`	first threshold
scaling factor	`~*~`	is scaled by
formative latent variable	`<~`	is a result of
defined parameter	`:=`	is defined as
equality constraint	`==`	is equal to
inequality constraint	`<`	is smaller than
inequality constraint	`>`	is larger than

more syntax: modifiers

- each rhs term can be preceded by a 'modifier'
- fixing parameters, and overriding auto-fixed parameters

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

- linear and nonlinear equality and inequality constraints

```
model.constr <- ' # model with labeled parameters
                    y ~ b1*x1 + b2*x2 + b3*x3

                  # constraints
                    b1 == (b2 + b3)^2
                    b1 > exp(b2 + b3) '
```

- several modifiers (eg. fix and label)

```
myModel <- ' y ~ 0.5*x1 + x2 + x3 + b1*x1 '
```

the main fitting function: `lavaan()`

- the `lavaan()` function –by default– adds *no* model parameters to the parameter table, nor are any actions taken to identify the model
- nevertheless, as a convenience, several `auto.*` arguments are available to
  - automatically add a set of parameters (e.g. all (residual) variances)
  - take actions to make the model identifiable (e.g. set the metric of the latent variables)
- the `lavaan()` function accepts 'slots' (for example, `slotModel`), perhaps created in a previous run

arguments of the lavaan() fitting function

```
lavaan(model = NULL, data = NULL, ordered = NULL, sample.cov = NULL,
    sample.mean = NULL, sample.nobs = NULL, group = NULL, cluster = NULL,
    constraints = "", WLS.V = NULL, NACOV = NULL, slotOptions = NULL,
    slotParTable = NULL, slotSampleStats = NULL, slotData = NULL,
    slotModel = NULL, slotCache = NULL, ...)
```

example using lavaan with an auto.* argument

```
HS.model.mixed <- ' # latent variables
                     visual  =~ 1*x1 + x2 + x3
                     textual =~ 1*x4 + x5 + x6
                     speed   =~ 1*x7 + x8 + x9
                   # factor covariances
                     visual  ~~ textual + speed
                     textual ~~ speed
                  '
fit <- lavaan(HS.model.mixed, data = HolzingerSwineford1939,
              auto.var = TRUE)
```

the '...' argument accepts a long list of options

- see the man page of `lavOptions()` to get a complete overview

```
?lavOptions
```

- each of these options can be added as extra arguments to the `lavaan()` function

overview lavOptions()

```
$model.type
[1] "sem"

$mimic
[1] "lavaan"

$meanstructure
[1] "default"

$int.ov.free
[1] FALSE

$int.lv.free
[1] FALSE

$conditional.x
[1] "default"

$fixed.x
[1] "default"

$orthogonal
[1] FALSE

$std.lv
[1] FALSE

$parameterization
[1] "default"

$auto.fix.first
[1] FALSE

$auto.fix.single
[1] FALSE

$auto.var
[1] FALSE

$auto.cov.lv.x
[1] FALSE

$auto.cov.y
[1] FALSE

$auto.th
[1] FALSE

$auto.delta
[1] FALSE

$std.ov
[1] FALSE

$missing
[1] "default"

$sample.cov.rescale
[1] "default"

$ridge
[1] FALSE

$ridge.x
[1] FALSE

$ridge.constant
[1] "default"

$ridge.constant.x
[1] 1e-05

$group.label
NULL
```

```
$group.equal
[1] ""

$group.partial
[1] ""

$group.w.free
[1] FALSE

$level.label
NULL

$estimator
[1] "default"

$likelihood
[1] "default"

$link
[1] "default"

$representation
[1] "default"

$do.fit
[1] TRUE

$information
[1] "default"

$h1.information
[1] "structured"

$se
[1] "default"

$test
[1] "default"

$bootstrap
[1] 1000

$observed.information
[1] "hessian"

$gamma.n.minus.one
[1] FALSE

$control
list()

$optim.method
[1] "nlminb"

$optim.method.cor
[1] "nlminb"

$optim.force.converged
[1] FALSE

$optim.gradient
[1] "analytic"

$optim.init_nelder_mead
[1] FALSE

$optim.var.transform
[1] "none"

$optim.parscale
[1] "none"

$em.iter.max
[1] 10000

$em.fx.tol
[1] 1e-08

$em.dx.tol
[1] 1e-04

$em.zerovar.offset
[1] 1e-04
```

```
$integration.ngh
[1] 21

$parallel
[1] "no"

$ncpus
[1] 1

$cl
NULL

$iseed
NULL

$zero.add
[1] "default"

$zero.keep.margins
[1] "default"

$zero.cell.warn
[1] FALSE

$start
[1] "default"

$check.start
[1] TRUE

$check.post
[1] TRUE

$check.gradient
[1] TRUE

$check.vcov
[1] TRUE

$h1
[1] TRUE

$baseline
[1] TRUE

$baseline.conditional.x.free.slopes
[1] TRUE

$implied
[1] TRUE

$loglik
[1] TRUE

$verbose
[1] FALSE

$warn
[1] TRUE

$debug
[1] FALSE
```

user-friendly fitting functions: `sem()` and `cfa()`

- `sem()` is just a wrapper around the `lavaan()` function where several `auto.*` arguments are set to `TRUE` (see next slide)
- `cfa()` is identical to `sem()`
- the older `growth()` function will be removed, and should not be used anymore

arguments of the cfa() and sem() fitting functions

```
sem(model = NULL, data = NULL, ordered = NULL, sample.cov = NULL,
    sample.mean = NULL, sample.nobs = NULL, group = NULL, cluster = NULL,
    constraints = "", WLS.V = NULL, NACOV = NULL, ...)
```

auto.* elements and other automatic actions

keyword	operator	parameter set
`auto.var`	~~	(residual) variances observed and latent variables
`auto.cov.y`	~~	(residual) covariances observed and latent endogenous variables
`auto.cov.lv.x`	~~	covariances among exogenous latent variables
keyword	**default**	**action**
`auto.fix.first`	TRUE	fix the factor loading of the first indicator to 1
`auto.fix.single`	TRUE	fix the residual variance of a single indicator to 1
`int.ov.free`	TRUE	freely estimate the intercepts of the observed variables (only if a mean structure is included)
`int.lv.free`	FALSE	freely estimate the intercepts of the latent variables (only if a mean structure is included)

standard R extractor functions

Method	Description
`summary()`	print a long summary of the model results
`show()`	print a short summary of the model results
`coef()`	returns the estimates of the free parameters in the model as a named numeric vector
`fitted()`	returns the implied moments (covariance matrix and mean vector) of the model
`resid()`	returns the raw, normalized or standardized residuals (difference between implied and observed moments)
`vcov()`	returns the covariance matrix of the estimated parameters
`predict()`	compute factor scores
`logLik()`	returns the log-likelihood of the fitted model (if maximum likelihood estimation was used)
`AIC()`, `BIC()`	compute information criteria (if maximum likelihood estimation was used)
`update()`	update a fitted lavaan object

lavaan-specific extractor functions

Method	Description
`lavInspect()`	main extractor function to extract information from fitted lavaan object; by default, it returns a list of model matrices counting the free parameters in the model; can also be used to extract starting values, sample statistics, implied statistics and much more
`inspect()`	wrapper around the `inspect()` with some default options
`lavTech()`	same as `lavInspect()` but without pretty printing; use this within scripts or external packages

- see the man page for `lavInspect()` to see all the options:

```
?lavInspect
```

other functions (1)

Function	Description
`lavaanify()`	converts a lavaan model syntax to a parameter table
`parameterTable()`	returns the parameter table
`parameterEstimates()`	returns the parameter estimates, including confidence intervals, as a data frame
`standardizedSolution()`	returns one of three types of standardized parameter estimates, as a data frame
`modindices()`	computes modification indices and expected parameter changes
`varTable`	return information about the observed variables in the model
`fitMeasures()`	return all (=default) or a few selected fit measures
`lavNames()`	extract variables names from a fitted lavaan object

other functions (2)

Function	Description
`lavTables()`	frequency tables for categorical variables and related statistics
`lavCor()`	compute polychoric, polyserial and/or Pearson correlations
`lavTestLRT()`	compare two or more (nested) models using a likelihood ratio test
`lavTestWald()`	Wald test for testing a linear hypothesis about the parameters of fitted lavaan object
`lavTestScore()`	Score test (or Lagrange Multiplier test) for releasing one or more fixed or constrained parameters in model
`bootstrapLavaan()`	bootstrap any arbitrary statistic that can be extracted from a fitted lavaan object
`bootstrapLRT()`	bootstrap a chi-square difference test for comparing to alternative models

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: lavTestScore()

```
> lavTestScore(fit, add = "visual =~ x9")

$test

total score test:

   test     X2 df p.value
1 score 36.411  1       0

$uni

univariate score tests:

         lhs op rhs     X2 df p.value
1 visual=~x9 ==   0 36.411  1       0
```

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
     srmr srmr.se srmr.z srmr.pvalue usrmr usrmr.se
cov 0.065   0.006  6.063           0 0.058     0.01
```

example: lavTestLRT()

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

Chi Square 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

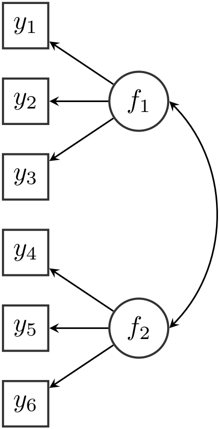
```
lavaan 0.6-3 ended normally after 35 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301

  Estimator                                         ML
  Model Fit 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
```

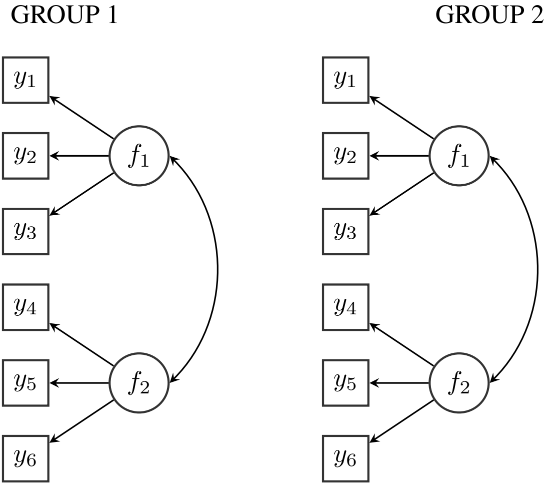
3.2 Multiple groups

single group analysis (CFA)



- factor means typically fixed to zero

multiple group analysis (CFA)

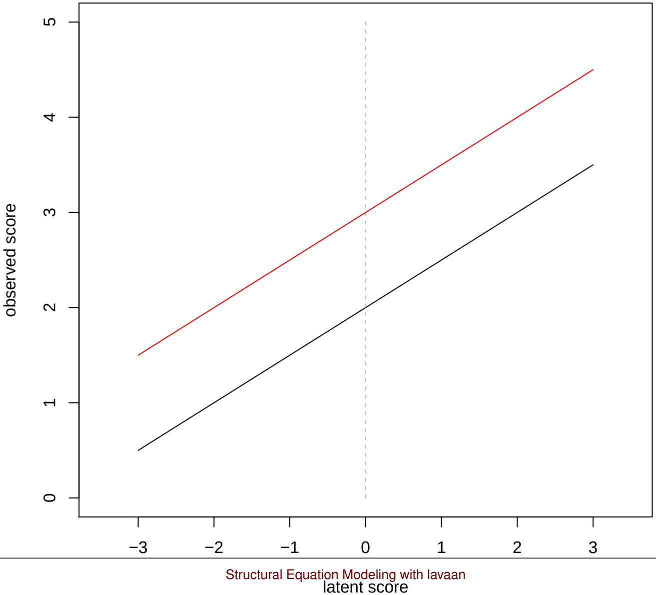


- can we compare the means of the latent variables?

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)



criteria to decide whether the parameter constraints are violated

- formal model comparison tests: compare the current model with the previous one using a chi-square difference test (likelihood ratio test)
  - may be sensitive to large sample sizes (over-powered)
- informal model comparison: compare the difference between fit measures (often CFI or RMSEA) between the current model and the previous model
  - Cheung & Rensvold (2002); Chen (2007)
- look at the overall fit of the current model (either using the chi-squared test, or some fit measures)
- look at parameters of interest
  - Millsap (1997), Millsap and Kwok (2004), Millsap (2007), Meuleman (2012), Oberski (2014)

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 Square 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 Square 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")
```

output strong invariance model

```
lavaan 0.6-3 ended normally after 61 iterations

  Optimization method                           NLMINB
  Number of free parameters                         63
  Number of equality constraints                    15

  Number of observations per group
  Pasteur                                          156
  Grant-White                                      145

  Estimator                                         ML
  Model Fit Test Statistic                     164.103
  Degrees of freedom                                60
  P-value (Chi-square)                           0.000

Chi-square for each group:

  Pasteur                                       90.210
  Grant-White                                   73.892

Parameter Estimates:

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

```
Group 1 [Pasteur]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2        (l2)    0.576    0.101    5.713    0.000
    x3        (l3)    0.798    0.112    7.146    0.000
  textual =~
    x4                1.000
    x5        (l5)    1.120    0.066   16.965    0.000
    x6        (l6)    0.932    0.056   16.608    0.000
  speed =~
    x7                1.000
    x8        (l8)    1.130    0.145    7.786    0.000
    x9        (l9)    1.009    0.132    7.667    0.000

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

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

```
   .x1          (i1)     5.001     0.090   55.760     0.000
   .x2          (i2)     6.151     0.077   79.905     0.000
   .x3          (i3)     2.271     0.083   27.387     0.000
   .x4          (i4)     2.778     0.087   31.954     0.000
   .x5          (i5)     4.035     0.096   41.858     0.000
   .x6          (i6)     1.926     0.079   24.426     0.000
   .x7          (i7)     4.242     0.073   57.975     0.000
   .x8          (i8)     5.630     0.072   78.531     0.000
   .x9          (i9)     5.465     0.069   79.016     0.000
    visual               0.000
    textual              0.000
    speed                0.000

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

```
Group 2 [Grant-White]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  visual =~
    x1                1.000
    x2        (l2)    0.576    0.101    5.713    0.000
    x3        (l3)    0.798    0.112    7.146    0.000
  textual =~
    x4                1.000
    x5        (l5)    1.120    0.066   16.965    0.000
    x6        (l6)    0.932    0.056   16.608    0.000
  speed =~
    x7                1.000
    x8        (l8)    1.130    0.145    7.786    0.000
    x9        (l9)    1.009    0.132    7.667    0.000

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

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

```
   .x1          (i1)     5.001     0.090   55.760     0.000
   .x2          (i2)     6.151     0.077   79.905     0.000
   .x3          (i3)     2.271     0.083   27.387     0.000
   .x4          (i4)     2.778     0.087   31.954     0.000
   .x5          (i5)     4.035     0.096   41.858     0.000
   .x6          (i6)     1.926     0.079   24.426     0.000
   .x7          (i7)     4.242     0.073   57.975     0.000
   .x8          (i8)     5.630     0.072   78.531     0.000
   .x9          (i9)     5.465     0.069   79.016     0.000
    visual              -0.148     0.122   -1.211     0.226
    textual              0.576     0.117    4.918     0.000
    speed               -0.177     0.090   -1.968     0.049

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

3.4 What if measurement invariance can not be established? (optional)

1. remove some groups, and/or use subgroups instead (e.g. only a few countries)
2. when is a violation of invariance large enough to warrant concern?
   - sometimes, we can still retain a ranking among groups
   - the invariance violations may not have a substantive impact on the comparison (see, e.g. Oberski, 2014)
3. invariance may not need to hold for all indicators/items
   - partial invariance (Byrne, Shavelson & Muthén, 1989)
   - delete these items, or not? literature is not conclusive
4. try to explain/understand the reason why we observe non-invariance
   - can we blame one or two items?
   - how to find these items that 'behave' differently

which parameters are responsible?

- if invariance is violated, how can we accurately locate which parameters are responsible? this turns out to be rather tricky
- one approach: use modification indices to relax equality constraints until we reach measurement invariance
  - data-driven respecifications are likely to mislead, especially if many modifications are needed (MacCallum, 1986)
  - unclear how well this works in pratice
  - different ways to define the metric of latent variables may have a huge impact
- fit a MIMIC model: a CFA model where the grouping variable (gender, age, . . .) is included as an exogenous covariate influencing the items directly
- use person/country level predictors to ‘explain’ the differential item functioning (using multilevel CFA)

what is the impact of releasing the equality constraints?

- we wish to compare the latent means across two groups
- the first group is the reference group, and the latent means are fixed to zero; the second group has a free latent means; these are the parameters-of-interest
- we first fit the strong (scalar) invariance model

```
> fit.strong <- cfa(HS.model, data = HolzingerSwineford1939,
+                    group = "school",
+                    group.equal = c("loadings", "intercepts"))
```

- estimated latent means:

```
> PE <- parameterEstimates(fit.strong)
> idx <- with(PE, which(op == "~1" &
+                        lhs %in% c("visual","textual","speed") &
+                        group == 2))
> PE[idx,]

       lhs op rhs block group label    est    se      z pvalue ci.lower
70  visual ~1          2     2       -0.148 0.122 -1.211  0.226   -0.387
71 textual ~1          2     2        0.576 0.117  4.918  0.000    0.347
72   speed ~1          2     2       -0.177 0.090 -1.968  0.049   -0.354
   ci.upper
```

```
70    0.091
71    0.806
72   -0.001
```

- next, we ‘release’ the equality constraints, and observe by how much the parameters of interest change:

```
> EPC <- lavTestScore(fit.strong, epc = TRUE)$epc
> idx <- with(EPC, which(op == "~1" &
+                        lhs %in% c("visual","textual","speed") &
+                        group == 2))
> EPC[idx,]
```

```
expected parameter changes (epc) and expected parameter values (epv):

       lhs op rhs group free label plabel    est    epc    epv
70  visual ~1         2   61        .p70. -0.148 -0.015 -0.163
71 textual ~1         2   62        .p71.  0.576 -0.019  0.557
72   speed ~1         2   63        .p72. -0.177 -0.001 -0.178
```

3.5 Measurement invariance: recent developments and references

- explorarory SEM (ESEM; Asparouhov & Muthen, 2009)
  - cross-loadings can be non-zero
- Bayesian SEM (e.g. Muthen & Asparouhov, 2012, 2013)
  - approximate (instead of strict) measurement invariance
  - these methods allow for some 'wiggle room' across groups
- alignment (Asparouhov & Muthen, 2013)
  - equality constraints are replaced by a procedure similar to rotation in EFA

references

- technical:
  - Meredith, W. (1993). Measurement invariance, factor analysis and factorial invariance. *Psychometrika, 58*, 525–543.
  - Millsap, R.E. (2011). *Statistical approaches to measurement invariance*. Routledge.
- general references:
  - Vandenberg, R.J. and Lance, C.E. (2000). A Review and Synthesis of the Measurement Invariance Literature: Suggestions, Practices, and Recommendations for Organizational Research. *Organizational Research Methods, 3*, 4–69.
  - Horn, J.L., & McArdle, J.J. (1992). A practical and theoretical guide to measurement invariance in aging research. *Experimental Aging Research, 18*, 117–144.
- reviews:
  - Schmitt, N., & Kuljanin, G. (2008). Measurement invariance: Review of practice and implications. *Human Resource Management Review, 18*, 210–222.

- Davidov, E., Meuleman, B., Cieciuch, J., Schmidt, P., & Billiet, J. (2014). Measurement equivalence in cross-national research. *Annual Review of Sociology, 40*, 55–75.

- testing strategies:
  - Cheung, G.W., and Rensvold, R.B. (2000). Evaluating goodness-of-fit indices for testing measurement invariance. *Structural Equation Modeling, 9*, 233–255.

- partial invariance:
  - Byrne, B.M., Shavelson, R.J and Muthén, B. (1989). Testing for the equivalence of factor covariance and mean structures: The issue of partial measurement invariance. *Psychological Bulletin, 105*, 456–466.

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 0.6-3 ended normally after 51 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301
  Number of missing patterns                        13

  Estimator                                         ML
  Model Fit Test Statistic                      85.868
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000


> # missing patterns
> lavInspect(fit, "patterns")

      x1 x2 x3 x4 x5 x6 x7 x8 x9
 [1,]  1  1  1  1  1  1  1  1  1
 [2,]  1  0  1  1  1  1  1  1  1
 [3,]  1  1  0  1  1  1  1  1  1
 [4,]  0  1  1  1  1  1  1  1  1
```

```
 [5,]   1   1   1   0   1   1   1   1   1
 [6,]   1   1   1   1   0   1   1   1   1
 [7,]   1   1   1   1   1   0   1   1   1
 [8,]   1   1   1   1   1   1   0   1   1
 [9,]   1   1   1   1   1   1   1   0   1
[10,]   1   1   1   1   1   1   1   1   0
[11,]   0   1   1   0   1   0   1   1   1
[12,]   1   1   1   1   0   1   1   0   1
[13,]   1   1   1   1   1   1   0   1   0
```

```
> # percentage complete cases per pair
> lavInspect(fit, "coverage")
```

```
   x1    x2    x3    x4    x5    x6    x7    x8    x9
x1 0.983
x2 0.967 0.983
x3 0.967 0.967 0.983
x4 0.970 0.967 0.967 0.983
x5 0.967 0.967 0.967 0.967 0.983
x6 0.970 0.967 0.967 0.970 0.967 0.983
x7 0.967 0.967 0.967 0.967 0.967 0.967 0.983
x8 0.967 0.967 0.967 0.967 0.970 0.967 0.967 0.983
x9 0.967 0.967 0.967 0.967 0.967 0.967 0.970 0.967 0.983
```

```
> # sample statistics unrestricted (h1) model
> lavInspect(fit, "sampstat.h1")
```

```
$cov
   x1     x2     x3     x4     x5     x6     x7     x8     x9
x1  1.367
x2  0.412  1.398
x3  0.590  0.478  1.266
x4  0.503  0.215  0.214  1.357
x5  0.438  0.218  0.119  1.096  1.665
x6  0.431  0.248  0.233  0.875  1.011  1.173
x7  0.074 -0.113  0.045  0.224  0.141  0.137  1.189
x8  0.274  0.109  0.205  0.154  0.212  0.153  0.534  1.002
x9  0.476  0.232  0.367  0.243  0.296  0.225  0.371  0.460  1.023

$mean
   x1    x2    x3    x4    x5    x6    x7    x8    x9
4.937 6.087 2.271 3.062 4.338 2.182 4.186 5.518 5.372
```

- the sample statistics for the unrestricted (h1) model are only needed to get a loglikelihood for h1
- together with the loglikelihood for the user-specified model (h0) we can compute the likelihood ratio test statistic (= the chi-squared test statistic)

example: lavaan + two.stage

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

lavaan 0.6-3 ended normally after 36 iterations

  Optimization method                           NLMINB
  Number of free parameters                         30

  Number of observations                           301
  Number of missing patterns                        13

  Estimator                                         ML      Robust
  Model Fit Test Statistic                      90.130      87.108
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.000       0.000
  Scaling correction factor                                  1.035
    for the 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 (see Appendix)
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 (see Appendix)

robust ML in lavaan

- robust standard errors

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

- Satorra-Bentler scaled test statistic

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

- robust standard errors + scaled test statistic

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

- estimator MLM = robust standard errors + scaled test statistic

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

- alternative: estimator MLR (also for missing data)

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

output: robust standard errors and scaled test statistic

```
> fit <- cfa(HS.model, data = HolzingerSwineford1939,
+            estimator = "MLM")
> summary(fit, fit.measures = TRUE, estimates = FALSE)

lavaan 0.6-3 ended normally after 35 iterations

  Optimization method                           NLMINB
  Number of free parameters                         21

  Number of observations                           301

  Estimator                                         ML       Robust
  Model Fit Test Statistic                      85.306       80.872
  Degrees of freedom                                24           24
  P-value (Chi-square)                           0.000        0.000
  Scaling correction factor                                   1.055
    for the Satorra-Bentler correction

Model test baseline model:

  Minimum Function Test Statistic              918.852      789.298
  Degrees of freedom                                36           36
  P-value                                        0.000        0.000

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

  Number of free parameters                         21          21
  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          0.071  0.114       0.068  0.110
  P-value RMSEA <= 0.05                          0.001       0.001

  Robust RMSEA                                               0.091
  90 Percent Confidence Interval                             0.070  0.113

Standardized Root Mean Square Residual:

  SRMR                                           0.065       0.065
```

mimic option

```
> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "EQS")
...
  Estimator                                         ML       Robust
  Minimum Function Test Statistic               85.022       81.141
...

> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "Mplus")

...
  Estimator                                         ML       Robust
  Minimum Function Test Statistic               85.306       81.908
...

> cfa(HS.model, data = HolzingerSwineford1939,
      estimator = "MLM", mimic = "lavaan")
...
  Estimator                                         ML       Robust
  Minimum Function Test Statistic               85.306       80.872
...
```

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

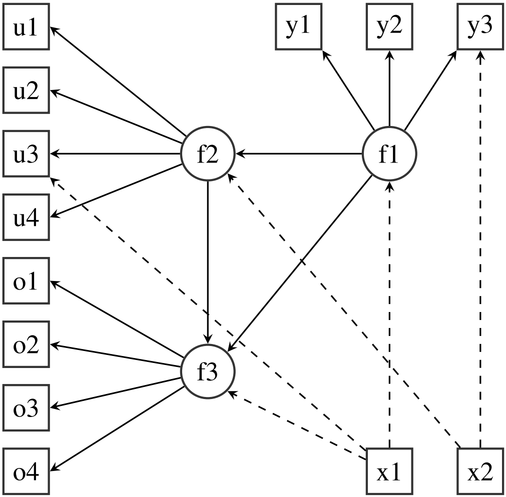
categorical endogenous variables

- 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

example SEM framework: u = binary, o = ordered, y = numeric



full information approach

1. marginal maximum likelihood (MML)
2. latent response approach
3. Bayesian estimation

limited information approaches

1. three stage least squares (Mplus WLSMV)
2. pairwise likelihood estimation

5.3 A limited information approach: the WLSMV estimator

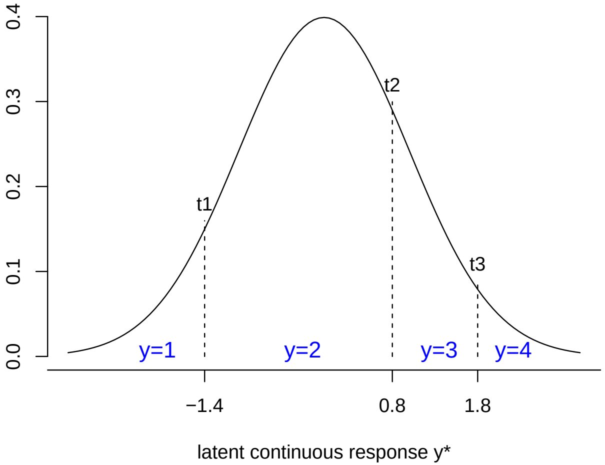
- developed by Bengt Muthén, in a series of papers; the seminal paper is

  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 1 – estimating the thresholds in R

- if no exogenous variables, this is just

```
> set.seed(1234)
> # generate `ordered' data with 4 categories
> Y <- sample(1:4, size = 100, replace = TRUE)
> # construct table of proportions
> prop <- table(Y)/sum(table(Y))
> prop

Y
   1    2    3    4
0.31 0.24 0.26 0.19


> # cumulative proportions
> cprop <- c(0, cumsum(prop))
> cprop

        1    2    3    4
0.00 0.31 0.55 0.81 1.00


> # convert quantiles to z-scores
> th <- qnorm(cprop)
> th
```

```
                1          2          3          4
    -Inf -0.4958503  0.1256613  0.8778963        Inf
```

- in the presence of exogenous covariates, this is just ordered probit regression

```
> library(MASS)
> X1 <- rnorm(100); X2 <- rnorm(100); X3 <- rnorm(100)
> # fit ordered probit regression
> fit <- polr(ordered(Y) ~ X1 + X2 + X3, method = "probit")
> # (residual) thresholds
> fit$zeta

       1|2        2|3        3|4
-0.4947713  0.1264129  0.8784373
```

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 2 – tetrachoric, polychoric, . . . , correlations in R

- lavaan provides the `lavCor()` function to compute the tetrachoric, polychoric, polyserial, . . . correlations
- example using two binary variables:

```
> library(lavaan)
> # create some random correlated data
> set.seed(1234)
> Y12 <- MASS::mvrnorm(n = 100, mu = c(0,0),
+                      Sigma = matrix(c(1,0.5,0.5,1), 2, 2))
> # transform to binary
> y1 <- cut(Y12[,1], breaks = c(-Inf, 0, +Inf), labels = FALSE)
> y2 <- cut(Y12[,2], breaks = c(-Inf, 0, +Inf), labels = FALSE)
> Data <- data.frame(y1 = y1, y2 = y2)

> # compute tetrachoric correlation
> lavCor(Data, ordered = c("y1", "y2"))
```

```
   y1    y2
y1 1.000
y2 0.713 1.000
```

stage 2b – estimating the W matrix

- in the ideal case, $\mathbf{W}$ reflects the (asymptotic) variance matrix of the sample statistics: the thresholds and the correlations
- an estimate of (N times) this variance matrix can be computed as follows:

```
> fit <- sem('y1 ~~ y2', data = Data, ordered = c("y1", "y2"),
+            estimator = "WLS")
> lavInspect(fit, "Gamma")

      [,1]   [,2]   [,3]
[1,]  1.658
[2,]  0.809  1.601
[3,] -0.070 -0.015  0.960
```

- the first two rows/columns correspond to the two thresholds; the last row/column corresponds to the single tetrachoric correlation
- the diagonal elements reflect the variances of these statistics (over repeated sampling)
- th off-diagonal elements reflect the covariances of these statistics (over repeated sampling)

stage 3 – estimating the SEM model

- third stage uses weighted least squares:

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

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

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

alternative estimators, standard errors, and test statistics

- in the weighted least squares framework, we can choose between three different choises for $\mathbf{W}$, leading to three different estimators:
  - estimator WLS: the full weight matrix $\mathbf{W}$ is used during estimation
  - estimator DWLS: only the diagonal of $\mathbf{W}$ is used during estimation
  - estimator ULS: $\mathbf{W}$ is replaced by the identity matrix ($\mathbf{I}$)
- two common types of standard errors:
  - 'classic' standard errors (based on the information matrix only)
  - 'robust' standard errors (using a sandwich type approach)
- four test statistics:
  - uncorrected, standard chi-square test statistic
  - mean adjusted test statistic (Satorra-Bentler type)
  - mean and variance adjusted test statistic (Satterthwaite type)
  - scaled and shifted test statistic (new in Mplus 6)

the Mplus legacy (optional)

- in Mplus, the ‘default’ estimator (for models with endogenous categorical variables) is termed WLSMV
- the term ‘WLSMV’ is widely used in the SEM literature
- in version 1 up to version 5 of Mplus, estimator WLSMV implies:
  - diagonally weighted least-squares estimation (DWLS)
  - robust standard errors
  - a mean and variance adjusted test statistic (hence, the MV extension)
- other available estimators (in Mplus) are
  - WLS (classical WLS, full weight matrix, classic standard errors and test statistic)
  - WLSM (DWLS + robust standard errors + mean-adjusted test statistic)
  - ULS, USLM and ULSMV (the latter two use the full weight matrix for computing standard errors and adjusted test statistics)

- since Mplus 6 (April 2010), the mean and variance adjusted test statistic was replaced by a 'scaled and shifted' test statistic
  - they still call this WLSMV
  - no need to adjust the degrees of freedom, so interpretation is easier
  - to get the 'old' behaviour, you need to set the 'satterthwaite=on' option

5.4 Using categorical variables in lavaan

- before you start, check the 'type' (or class) of the variables you will use in your model: are they numeric, or factor, or ordered, ...?
- in R, you can check the 'type' of a variable by typing

```
> x <- c(3,4,5)
> class(x)

[1] "numeric"


> x <- factor(x)
> class(x)

[1] "factor"


> x <- ordered(x)
> class(x)

[1] "ordered" "factor"
```

varTable

- a convenience function to screen the variables in lavaan is the ‘varTable()’ function:

```
> # library(lavaan)
> varTable(HolzingerSwineford1939)

     name idx nobs    type exo user    mean       var nlev                lna
1      id   1  301 numeric   0    0 176.555 11222.961    0
2     sex   2  301 numeric   0    0   1.515     0.251    0
3   ageyr   3  301 numeric   0    0  12.997     1.103    0
4   agemo   4  301 numeric   0    0   5.375    11.915    0
5  school   5  301  factor   0    0      NA        NA    2 Grant-White|Pasteu
6   grade   6  300 numeric   0    0   7.477     0.250    0
7      x1   7  301 numeric   0    0   4.936     1.363    0
8      x2   8  301 numeric   0    0   6.088     1.386    0
9      x3   9  301 numeric   0    0   2.250     1.279    0
10     x4  10  301 numeric   0    0   3.061     1.355    0
11     x5  11  301 numeric   0    0   4.341     1.665    0
12     x6  12  301 numeric   0    0   2.186     1.200    0
13     x7  13  301 numeric   0    0   4.186     1.187    0
14     x8  14  301 numeric   0    0   5.527     1.025    0
15     x9  15  301 numeric   0    0   5.374     1.018    0
```

using categorical variables in lavaan (2)

- two approaches to deal with ‘ordered’ (including binary) endogenous variables in lavaan:

  1. declare them as ‘ordered’ (using the `ordered()` function, which is part of base R) in your data.frame before you run the analysis;

     for example, if you need to declare four variables (say, item1, item2, item3, item3) as ordinal in your data.frame (called ‘Data’), you can use something like:

     ```
     Data[,c("item1","item2","item3","item4")] <-
         lapply(Data[,c("item1","item2","item3","item4")], ordered)
     ```

  2. use the `ordered=` argument when using one of the fitting functions; for example, if you have four binary or ordinal variables (say, item1, item2, item3, item4), you can use:

     ```
     fit <- cfa(myModel, data=myData, ordered=c("item1","item2",
                                                 "item3","item4"))
     ```

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 0.6-3 ended normally after 35 iterations

  Optimization method                           NLMINB
  Number of free parameters                         21

  Number of observations                           301

  Estimator                                       DWLS      Robust
  Model Fit 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
    for simple second-order correction (Mplus variant)

Model test baseline model:

  Minimum Function Test Statistic              582.533     468.233
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000

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          0.000  0.059       0.014  0.070
  P-value RMSEA <= 0.05                          0.847       0.600

  Robust RMSEA                                                  NA
  90 Percent Confidence Interval                                NA      NA

Standardized Root Mean Square Residual:

  SRMR                                           0.083       0.083

Parameter Estimates:

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

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  visual =~
    x1                1.000                               0.639    0.639
```

```
    x2              0.900    0.188    4.788    0.000    0.575    0.575
    x3              0.939    0.197    4.766    0.000    0.600    0.600
  textual =~
    x4              1.000                               0.835    0.835
    x5              0.976    0.118    8.241    0.000    0.815    0.815
    x6              1.078    0.125    8.601    0.000    0.900    0.900
  speed =~
    x7              1.000                               0.471    0.471
    x8              1.569    0.461    3.403    0.001    0.740    0.740
    x9              1.449    0.409    3.541    0.000    0.683    0.683

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

Intercepts:
                 Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1              0.000                               0.000    0.000
   .x2              0.000                               0.000    0.000
   .x3              0.000                               0.000    0.000
   .x4              0.000                               0.000    0.000
   .x5              0.000                               0.000    0.000
   .x6              0.000                               0.000    0.000
   .x7              0.000                               0.000    0.000
```

```
   .x8               0.000                                   0.000    0.000
   .x9               0.000                                   0.000    0.000
    visual           0.000                                   0.000    0.000
    textual          0.000                                   0.000    0.000
    speed            0.000                                   0.000    0.000

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

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .x1                0.592                               0.592    0.592
   .x2                0.670                               0.670    0.670
   .x3                0.640                               0.640    0.640
   .x4                0.303                               0.303    0.303
   .x5                0.336                               0.336    0.336
   .x6                0.191                               0.191    0.191
   .x7                0.778                               0.778    0.778
   .x8                0.453                               0.453    0.453
```

```
   .x9              0.534                               0.534    0.534
    visual          0.408    0.112    3.651    0.000    1.000    1.000
    textual         0.697    0.101    6.883    0.000    1.000    1.000
    speed           0.222    0.094    2.363    0.018    1.000    1.000

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

estimated thresholds and tetrachoric correlations

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

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

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

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

estimators, standard errors and test statistics in lavaan

- in lavaan, you can set your estimator, type of standard errors, and type of test statistic separately
- estimators (least squares framework):
  - `estimator="WLS"`
  - `estimator="DWLS"`
  - `estimator="ULS"`
- standard errors:
  - `se="standard"`
  - `se="robust"`
  - `se="bootstrap"`
- test statistics:
  - `test="standard"`

- `test="Satorra.Bentler"`
- `test="Satterthwaite"`
- `test="scaled.shifted"`
- `test="bootstrap` or `test="Bollen.Stine"`

- or you can use the Mplus style shortcuts
- `estimator="WLSMV"` implies
  - `estimator="DWLS"`
  - `se="robust"`
  - `test="scaled.shifted"` (following Mplus 6 and higher)
- `estimator="WLSMVS"` implies
  - `estimator="DWLS"`
  - `se="robust"`
  - `test="Satterthwaite"` (following older versions of Mplus)

- alternatives:
  - `estimator="WLSM"`
  - `estimator="ULSMV"`
  - `estimator="ULSM"`

parameter matrices

```
> inspect(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 0
x2 0  0
x3 0  0  0
x4 0  0  0  0
x5 0  0  0  0  0
x6 0  0  0  0  0  0
x7 0  0  0  0  0  0  0
x8 0  0  0  0  0  0  0  0
x9 0  0  0  0  0  0  0  0  0
```

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

$nu
   intrcp
x1      0
x2      0
x3      0
x4      0
x5      0
x6      0
x7      0
x8      0
x9      0

$alpha
        intrcp
visual       0
textual      0
speed        0

$tau
      thrshl
x1|t1      7
x2|t1      8
```

```
x3|t1      9
x4|t1     10
x5|t1     11
x6|t1     12
x7|t1     13
x8|t1     14
x9|t1     15

$delta
   scales
x1      0
x2      0
x3      0
x4      0
x5      0
x6      0
x7      0
x8      0
x9      0
```

tables: univariate

```
> lavTables(fit, dim = 1)

   id lhs rhs nobs obs.freq obs.prop est.prop X2
1   1  x1   1  301      105    0.349    0.349  0
2   1  x1   2  301      196    0.651    0.651  0
3   2  x2   1  301      144    0.478    0.478  0
4   2  x2   2  301      157    0.522    0.522  0
5   3  x3   1  301      188    0.625    0.625  0
6   3  x3   2  301      113    0.375    0.375  0
7   4  x4   1  301      172    0.571    0.571  0
8   4  x4   2  301      129    0.429    0.429  0
9   5  x5   1  301      120    0.399    0.399  0
10  5  x5   2  301      181    0.601    0.601  0
11  6  x6   1  301      255    0.847    0.847  0
12  6  x6   2  301       46    0.153    0.153  0
13  7  x7   1  301      178    0.591    0.591  0
14  7  x7   2  301      123    0.409    0.409  0
15  8  x8   1  301      262    0.870    0.870  0
16  8  x8   2  301       39    0.130    0.130  0
17  9  x9   1  301      221    0.734    0.734  0
18  9  x9   2  301       80    0.266    0.266  0
```

tables: bivariate (only first four)

```
> head( lavTables(fit, dim = 2), 16)

   id lhs rhs nobs row col obs.freq obs.prop est.prop    X2
1   1  x1  x2  301   1   1       63    0.209    0.222 0.228
2   1  x1  x2  301   2   1       81    0.269    0.256 0.198
3   1  x1  x2  301   1   2       42    0.140    0.127 0.400
4   1  x1  x2  301   2   2      115    0.382    0.395 0.128
5   2  x1  x3  301   1   1       83    0.276    0.271 0.022
6   2  x1  x3  301   2   1      105    0.349    0.353 0.017
7   2  x1  x3  301   1   2       22    0.073    0.078 0.078
8   2  x1  x3  301   2   2       91    0.302    0.298 0.020
9   3  x1  x4  301   1   1       76    0.252    0.243 0.101
10  3  x1  x4  301   2   1       96    0.319    0.328 0.075
11  3  x1  x4  301   1   2       29    0.096    0.105 0.233
12  3  x1  x4  301   2   2      100    0.332    0.323 0.076
13  4  x1  x5  301   1   1       56    0.186    0.183 0.020
14  4  x1  x5  301   2   1       64    0.213    0.216 0.017
15  4  x1  x5  301   1   2       49    0.163    0.166 0.022
16  4  x1  x5  301   2   2      132    0.439    0.435 0.009
```

5.5 SEM vs IRT

the connection with IRT

- the theoretical relationship between SEM and IRT has been well documented:

Takane, Y., & De Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. Psychometrika, 52, 393-408.

Kamata, A., & Bauer, D. J. (2008). A note on the relation between factor analytic and item response theory models. Structural Equation Modeling, 15, 136-153.

Jöreskog, K. G., & Moustaki, I. (2001). Factor analysis of ordinal variables: A comparison of three approaches. Multivariate Behavioral Research, 36, 347-387.

- IRT: focus is on the scale and the item characteristics, person scores
- SEM: focus is (often) on the structural relations among either observed or latent variables; with or without exogenous covariates
- in lavaan (since 0.5-16): `estimator="MML"`

when are they equivalent (optional)?

- probit (normal-ogive) versus logit: both metrics are used in practice
- a single-factor CFA on binary items is equivalent to a 2-parameter IRT model (Birnbaum, 1968):
  - in CFA: λ_i, τ_i and θ_i are the factor loadings, the thresholds, and the residual variances)
  - in IRT: α_i and β_i are item discrimination and difficulty respectively
  - for a standardized factor: $\alpha_i = \lambda_i/\sqrt{\theta_i}$ and $\beta_i = \tau_i/\lambda_i$
- a single-factor CFA on polychotomous (ordinal) items is equivalent to the graded response model (Samejima, 1969)
- there is no CFA equivalent for the 3-parameter model (with a guessing parameter)
- the Rasch model is equivalent to a single-factor CFA on binary items, but where all factor loadings are constrained to be equal (and the probit metric is converted to a logit metric)

6 Longitudinal Structural Equation Modeling

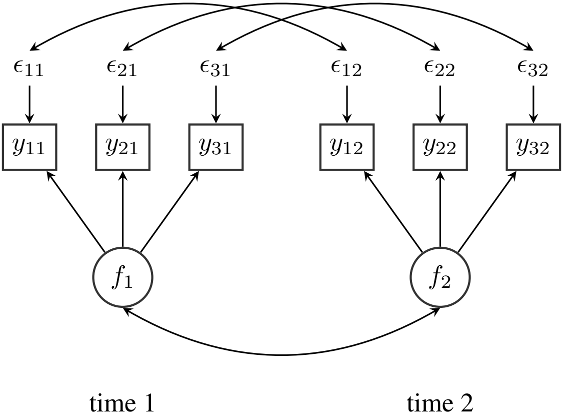
- long history, mostly for ‘balanced data’: same number of time points for each observation
  - repeated measures models
  - panel models, simplex models, autoregressive models
  - growth curve models (random coefficient models)
  - hybrid models (growth curve + autoregressive)
  - latent-state, latent-trait 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:



comments

- first of all, we need to establish measurement invariance across time points
  - it is tempting to do this using a multiple group analysis, using the time points as group levels, but this will not allow us to specify correlated residuals among the corresponding variables (and the time points are not independent)
  - therefore, we need to use labels for the different time points (for factor loadings and intercepts of observed variables), and impose the equality constraints by using the same label for the different time points
- since we wish to compare the latent means, we need 'strong invariance':
  - equal factor loadings
  - equal intercepts/means of the observed variables
- usually, we allow the residuals variances of the corresponding variables across time to be correlated

- if we have more than two time points, we can allow for all possible correlations among the repeated latent variables (this corresponds to the 'unstructured' assumption)
- the latent mean/intercept of the first time point is fixed to zero, while we estimate the latent mean/intercept of the other time points (although alternative coding schemes are possible)

real-world example

- example from Todd Little's book (Longitudinal SEM, 2013): table 3.8 and figure 3.10 (but with equality constraints)
- the latent variable 'positive affect' is measured by three indicators (Glad, Cheerful and Happy): 823 children in grades 7 en 8 responded to questions like "In the past 2 weeks, I have felt ..." (with 4 response categories: almost never, seldom, often, almost always)
- measured at two time points: in the fall of two successive school years
- main question: is there a significant difference in (self-reported) 'positive affect' between the two time points?
- this is the SEM equivalent of the paired t-test

caveats

- we have no access to the full data, but the tables in the book report all the sample statistics we need (means, standard deviations, correlations, sample size)
- we will have to convert the correlations to covariances (using the standard deviations); lavaan has a convenience function `getCov()` for doing just that
- before we attempt to compare two latent means, we must first establish measurement invariance (over time)

R code: reading in the sample statistics

```
> library(lavaan)
> MEAN <- c(3.06893, 2.92590, 3.11013, 3.02577, 2.85656, 3.09346)
> SDS  <- c(0.84194, 0.88934, 0.83470, 0.84081, 0.90864, 0.83984)
> lower <- '
+     1.00000
+     0.55226    1.00000
+     0.56256    0.60307    1.00000
+     0.31889    0.35898    0.27757    1.00000
+     0.24363    0.35798    0.31889    0.56014    1.00000
+     0.32217    0.36385    0.32072    0.56164    0.59738    1.00000 '
> COV <- getCov(lower, sds=SDS, names = c("Glad1", "Cheer1", "Happy1",
+                                          "Glad2", "Cheer2", "Happy2"))
> COV

           Glad1    Cheer1    Happy1     Glad2    Cheer2    Happy2
Glad1  0.7088630 0.4135162 0.3953488 0.2257459 0.1863819 0.2278048
Cheer1 0.4135162 0.7909256 0.4476782 0.2684330 0.2892800 0.2717608
Happy1 0.3953488 0.4476782 0.6967241 0.1948053 0.2418595 0.2248294
Glad2  0.2257459 0.2684330 0.1948053 0.7069615 0.4279434 0.3965998
Cheer2 0.1863819 0.2892800 0.2418595 0.4279434 0.8256266 0.4558680
Happy2 0.2278048 0.2717608 0.2248294 0.3965998 0.4558680 0.7053312
```

R code: fitting the ‘configural’ longitudinal CFA model

```
> model1 <- '
+     posAffect1 =~ 1*Glad1 + Cheer1 + Happy1
+     posAffect2 =~ 1*Glad2 + Cheer2 + Happy2
+     posAffect1 ~~ posAffect2
+
+     # intercepts
+     Glad1  ~ 1
+     Glad2  ~ 1
+     Cheer1 ~ 1
+     Cheer2 ~ 1
+     Happy1 ~ 1
+     Happy2 ~ 1
+
+     # residual covariances
+     Glad1  ~~ Glad2
+     Cheer1 ~~ Cheer2
+     Happy1 ~~ Happy2
+
+     # latent means: fixed to zero
+     posAffect1 ~ 0
+     posAffect2 ~ 0
+ '
> fit1 <- lavaan(model1, sample.cov = COV, sample.mean = MEAN,
+                sample.nobs = 823, auto.var = TRUE)
> # summary(fit1, standardized = TRUE)
```

R code: fitting the ‘weak invariance’ longitudinal CFA model

```
> model2 <- '
+     posAffect1 =~ 1*Glad1 + ch*Cheer1 + ha*Happy1
+     posAffect2 =~ 1*Glad2 + ch*Cheer2 + ha*Happy2
+     posAffect1 ~~ posAffect2
+
+     # intercepts
+     Glad1  ~ 1
+     Glad2  ~ 1
+     Cheer1 ~ 1
+     Cheer2 ~ 1
+     Happy1 ~ 1
+     Happy2 ~ 1
+
+     # residual covariances
+     Glad1  ~~ Glad2
+     Cheer1 ~~ Cheer2
+     Happy1 ~~ Happy2
+
+     # latent means: fixed to zero
+     posAffect1 ~ 0
+     posAffect2 ~ 0
+ '
> fit2 <- lavaan(model2, sample.cov = COV, sample.mean = MEAN,
+                sample.nobs = 823, auto.var = TRUE)
> # summary(fit2, standardized = TRUE)
```

R code: testing for weak invariance

- compare the configural and the weak invariance models:

```
> anova(fit1, fit2)

Chi Square Difference Test

     Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit1  5 10804 10908 18.432
fit2  7 10800 10894 18.534     0.1019       2     0.9503
```

- good, we have weak invariance over time

R code: fitting the ‘strong invariance’ longitudinal CFA model

```
> model3 <- '
+     posAffect1 =~ 1*Glad1 + ch*Cheer1 + ha*Happy1
+     posAffect2 =~ 1*Glad2 + ch*Cheer2 + ha*Happy2
+     posAffect1 ~~ posAffect2
+
+     # intercepts
+     Glad1  ~ igl*1
+     Glad2  ~ igl*1
+     Cheer1 ~ ich*1
+     Cheer2 ~ ich*1
+     Happy1 ~ iha*1
+     Happy2 ~ iha*1
+
+     # residual covariances
+     Glad1  ~~ Glad2
+     Cheer1 ~~ Cheer2
+     Happy1 ~~ Happy2
+
+     # latent means: fixed to zero
+     posAffect1 ~ 0*1 # baseline
+     posAffect2 ~ 1   # difference compared to baseline
+ '
> fit3 <- lavaan(model3, sample.cov = COV, sample.mean = MEAN,
+                sample.nobs = 823, auto.var = TRUE)
> summary(fit3, standardized = TRUE)

lavaan (0.6-1.1133) converged normally after  36 iterations
```

```
  Number of observations                           823

  Estimator                                         ML
  Model Fit Test Statistic                      20.279
  Degrees of freedom                                 9
  P-value (Chi-square)                           0.016

Parameter Estimates:

  Information                                 Expected
  Standard Errors                             Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  posAffect1 =~
    Glad1             1.000                               0.603    0.715
    Cheer1    (ch)    1.150    0.046   25.063    0.000    0.693    0.780
    Happy1    (ha)    1.076    0.043   25.208    0.000    0.648    0.777
  posAffect2 =~
    Glad2             1.000                               0.607    0.723
    Cheer2    (ch)    1.150    0.046   25.063    0.000    0.698    0.768
    Happy2    (ha)    1.076    0.043   25.208    0.000    0.653    0.780

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
  posAffect1 ~~
    posAffect2        0.202    0.021    9.840    0.000    0.553    0.553
 .Glad1 ~~
```

```
   .Glad2            0.032    0.015    2.074    0.038    0.032    0.092
 .Cheer1 ~~
   .Cheer2           0.017    0.016    1.047    0.295    0.017    0.053
 .Happy1 ~~
   .Happy2          -0.011    0.014   -0.800    0.424   -0.011   -0.041

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .Glad1    (igl)    3.067    0.027  114.088    0.000    3.067    3.639
   .Glad2    (igl)    3.067    0.027  114.088    0.000    3.067    3.652
   .Cheer1   (ich)    2.915    0.029   99.814    0.000    2.915    3.283
   .Cheer2   (ich)    2.915    0.029   99.814    0.000    2.915    3.204
   .Happy1   (iha)    3.123    0.027  115.351    0.000    3.123    3.740
   .Happy2   (iha)    3.123    0.027  115.351    0.000    3.123    3.726
    psAffct1          0.000                               0.000    0.000
    psAffct2         -0.040    0.025   -1.617    0.106   -0.066   -0.066

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
   .Glad1             0.347    0.022   15.601    0.000    0.347    0.489
   .Cheer1            0.308    0.024   13.028    0.000    0.308    0.391
   .Happy1            0.277    0.021   13.133    0.000    0.277    0.397
   .Glad2             0.336    0.022   15.312    0.000    0.336    0.477
   .Cheer2            0.340    0.025   13.578    0.000    0.340    0.411
   .Happy2            0.275    0.021   12.968    0.000    0.275    0.392
    posAffect1        0.363    0.029   12.399    0.000    1.000    1.000
    posAffect2        0.369    0.030   12.477    0.000    1.000    1.000
```

R code: testing for strong invariance

- compare the weak and the strong invariance model:

```
> anova(fit2, fit3)

Chi Square Difference Test

     Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit2  7 10800 10894 18.534
fit3  9 10798 10883 20.279     1.7451       2     0.4179
```

- splendid, we have strong invariance over time

Is there a difference between the two latent means?

- because there is only a single (latent) variable, we can immediately see the answer (='no') in the output of model3
- in general, we would need to fit a 'null' model where we constrain the two latent means to be equal; next, we compare these two models using anova()

R code: fitting the 'null' model

```
> model4 <- '
+     posAffect1 =~ 1*Glad1 + ch*Cheer1 + ha*Happy1
+     posAffect2 =~ 1*Glad2 + ch*Cheer2 + ha*Happy2
+     posAffect1 ~~ posAffect2
+
+     # intercepts
+     Glad1   ~ igl*1
+     Glad2   ~ igl*1
+     Cheer1  ~ ich*1
+     Cheer2  ~ ich*1
+     Happy1  ~ iha*1
+     Happy2  ~ iha*1
+
+     # residual covariances
+     Glad1   ~~ Glad2
+     Cheer1  ~~ Cheer2
+     Happy1  ~~ Happy2
+
+     # latent means: fixed to zero
+     posAffect1 ~ 0*1 # baseline
+     posAffect2 ~ 0*1 # equal means, both equal to zero
+ '
> fit4 <- lavaan(model4, sample.cov = COV, sample.mean = MEAN,
+                sample.nobs = 823, auto.var = TRUE)
```

- compare model3 versus model4:

```
> anova(fit3, fit4)

Chi Square Difference Test

     Df   AIC   BIC  Chisq Chisq diff Df diff Pr(>Chisq)
fit3  9 10798 10883 20.279
fit4 10 10799 10879 22.889     2.6099       1     0.1062
```

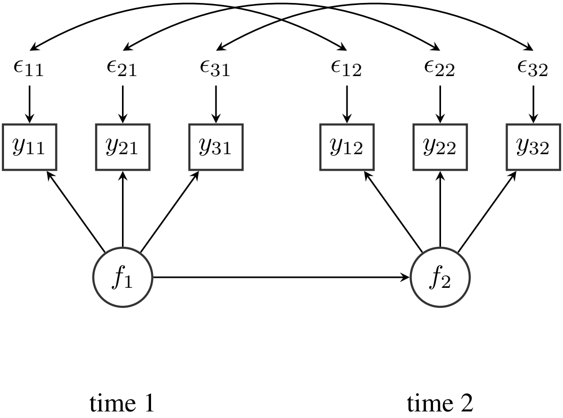
- answer: there is NO difference between the two values of reported ‘positive affect’ as measured over two successive school years

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
  - ...

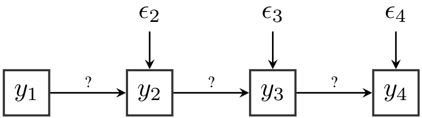
example panel model with a single latent variable

- example with 2 time points:



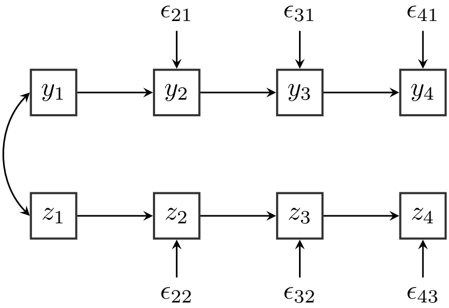
autoregressive models

- 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:



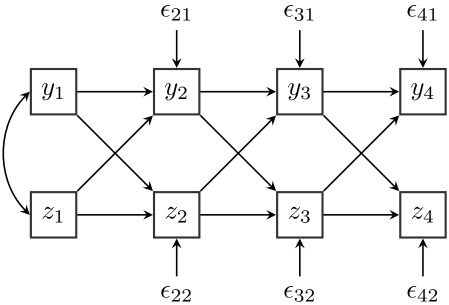
multivariate panel models

- 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)



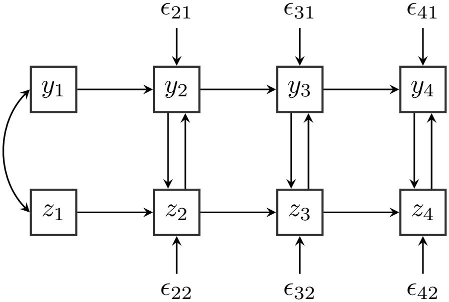
crosslagged effects

- 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?



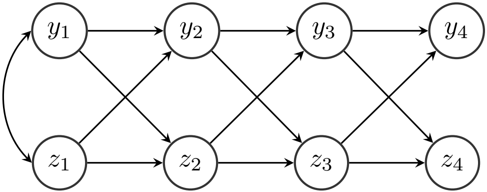
contemporaneous effects

- 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

real-world example

- example from the Curran & Bollen (2001) book chapter 'The Best of Two Worlds'
- topic: developmental relation between antisocial behavior and depressive symptomatology
- original data come from the National Longitudinal Survey of Youth (NLSY); original 1979 panel included a total of 12,686 respondents
- in 1986, the children of the original NLSY female respondents were also included in the survey
- the sample used for this example only includes data of children that were 8 years old at the first wave of measurement, have no missings on all four waves, only biological children, resulting in a final sample of $N = 180$
- we only include the sumscores of two constructs (antisocial behavior and depressive symptomatology), measured at 4 time points

preparing the sample statistics

- creating a covariance matrix for all 8 observed variables:

```
> lower <- '
+  2.926
+  1.390 4.257
+  1.698 2.781 4.536
+  1.628 2.437 2.979 5.605
+  1.240 0.789 0.903 1.278 3.208
+  0.592 1.890 1.419 1.004 1.706 3.994
+  0.929 1.278 1.900 1.000 1.567 1.654 3.583
+  0.659 0.949 1.731 2.420 0.988 1.170 1.146 3.649 '
> COV <- getCov(lower, names=c("anti1", "anti2", "anti3", "anti4",
+                              "dep1", "dep2", "dep3", "dep4"))
> MEANS <- c(1.750, 1.928, 1.978, 2.322, 2.178, 2.489, 2.294, 2.222)
```

questions

- what happens over time? how are the two constructs related to each other?

first-order autoregressive model: antisocial behavior

- unequal autoregressive coefficients, unequal residual variances

```
> model <- '
+   anti2 ~ a21*anti1
+   anti3 ~ a32*anti2
+   anti4 ~ a43*anti3
+
+   # one variance
+   anti1 ~~ anti1
+
+   # three (unqequal) residual variances
+   anti2 ~~ anti2; anti3 ~~ anti3; anti4 ~~ anti4
+ '
> fit <- lavaan(model, sample.cov=COV, sample.mean=MEANS, sample.nobs=180,
+               sample.cov.rescale=FALSE, mimic="EQS")
> summary(fit)

lavaan (0.6-1.1133) converged normally after  17 iterations

  Number of observations                           180

  Estimator                                         ML
  Model Fit Test Statistic                      29.004
  Degrees of freedom                                 3
  P-value (Chi-square)                           0.000
```

```
Parameter Estimates:

  Information                                  Expected
  Standard Errors                              Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  anti2 ~
    anti1    (a21)    0.475    0.083    5.733    0.000
  anti3 ~
    anti2    (a32)    0.653    0.060   10.936    0.000
  anti4 ~
    anti3    (a43)    0.657    0.067    9.797    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
    anti1             2.926    0.309    9.460    0.000
   .anti2             3.597    0.380    9.460    0.000
   .anti3             2.719    0.287    9.460    0.000
   .anti4             3.649    0.386    9.460    0.000
```

- fit not very good; we could impose equality constraints (equal regressions, equal residuals), but the overall impression is that the autoregressive model does not fit the data well

a bivariate crosslagged model

```
> model <- '
+   # antisocial behavior
+   anti2 ~ a*anti1
+   anti3 ~ a*anti2
+   anti4 ~ a*anti3
+
+   # variances + residuals
+   anti1 ~~ anti1; anti2 ~~ ra*anti2; anti3 ~~ ra*anti3; anti4 ~~ ra*anti4
+
+   # depressive symptomatology
+   dep2 ~ d*dep1
+   dep3 ~ d*dep2
+   dep4 ~ d*dep3
+
+   # variances + residuals
+   dep1 ~~ dep1; dep2 ~~ rd*dep2; dep3 ~~ rd*dep3; dep4 ~~ rd*dep4
+
+   # crosslagged effects
+   anti2 ~ ad*dep1
+   anti3 ~ ad*dep2
+   anti4 ~ ad*dep3
+
+   dep2 ~ da*anti1
+   dep3 ~ da*anti2
+   dep4 ~ da*anti3
```

```
+
+   # correlated residuals within time
+   anti1 ~~ dep1; anti2 ~~ c2*dep2; anti3 ~~ c2*dep3; anti4 ~~ c2*dep4
+ '
> fit <- lavaan(model, sample.cov=COV, sample.mean=MEANS, sample.nobs=180,
+               sample.cov.rescale=FALSE, mimic="EQS")
> summary(fit)

lavaan (0.6-1.1133) converged normally after  30 iterations

  Number of observations                           180

  Estimator                                         ML
  Model Fit Test Statistic                      95.092
  Degrees of freedom                                26
  P-value (Chi-square)                           0.000

Parameter Estimates:

  Information                                 Expected
  Standard Errors                             Standard

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  anti2 ~
    anti1      (a)    0.603    0.044   13.760    0.000
  anti3 ~
    anti2      (a)    0.603    0.044   13.760    0.000
```

```
  anti4 ~
    anti3        (a)    0.603    0.044   13.760    0.000
  dep2 ~
    dep1         (d)    0.343    0.045    7.617    0.000
  dep3 ~
    dep2         (d)    0.343    0.045    7.617    0.000
  dep4 ~
    dep3         (d)    0.343    0.045    7.617    0.000
  anti2 ~
    dep1        (ad)    0.016    0.047    0.341    0.733
  anti3 ~
    dep2        (ad)    0.016    0.047    0.341    0.733
  anti4 ~
    dep3        (ad)    0.016    0.047    0.341    0.733
  dep2 ~
    anti1       (da)    0.160    0.042    3.831    0.000
  dep3 ~
    anti2       (da)    0.160    0.042    3.831    0.000
  dep4 ~
    anti3       (da)    0.160    0.042    3.831    0.000

Covariances:
                    Estimate  Std.Err  z-value  P(>|z|)
  anti1 ~~
    dep1               1.240    0.247    5.019    0.000
 .anti2 ~~
   .dep2        (c2)    1.322    0.149    8.880    0.000
 .anti3 ~~
```

```
   .dep3       (c2)    1.322    0.149    8.880    0.000
  .anti4 ~~
   .dep4       (c2)    1.322    0.149    8.880    0.000

 Variances:
                     Estimate  Std.Err  z-value  P(>|z|)
    anti1               2.926    0.309    9.460    0.000
   .anti2      (ra)    3.344    0.204   16.386    0.000
   .anti3      (ra)    3.344    0.204   16.386    0.000
   .anti4      (ra)    3.344    0.204   16.386    0.000
    dep1                3.208    0.339    9.460    0.000
   .dep2       (rd)    3.035    0.185   16.386    0.000
   .dep3       (rd)    3.035    0.185   16.386    0.000
   .dep4       (rd)    3.035    0.185   16.386    0.000
```

- it would seem that earlier antisocial behavior predicts later depressive symptomatology, but not vice versa
- however, we should be careful with these parameters because the model does not fit the data well!

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

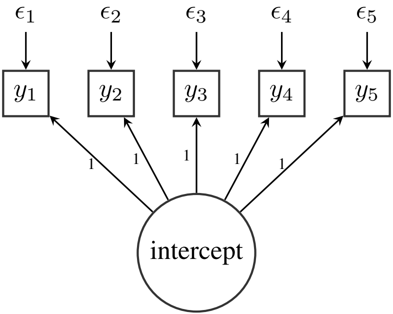
- Bollen, K.A., & Curran, P.J. (2006). *Latent curve models: A structural equation perspective.* John Wiley & Sons.
- Duncan, T.E., Duncan, S.C., & Strycker, L.A. (2006). *An introduction to latent variable growth curve modeling: Concepts, issues, and applications*. Routledge Academic.
- Preacher, K.J.,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

from latent variable to random effect

- a random effect is simply a latent variable with the following properties:
  - the repeated measures are the indicators of the latent variable
  - the factor loadings are fixed to a specific pattern
  - the intercepts of the observed repeated measures are fixed to zero
  - the mean/intercept of the latent variable is freely estimated
  - the (residual) variance of the latent variable is freely estimated
- typical patterns for the factor loadings:
  - by fixing all factor loadings to unity, we obtain a random intercept
  - by fixing all factor loadings to a linear scale (eg. 0, 1, 2, 3, ...) we obtain a random slope
  - by fixing all factor loadings to a quadratic scale (eg. 0, 1, 4, 9, ...), we obtain a random quadratic effect
  - ...

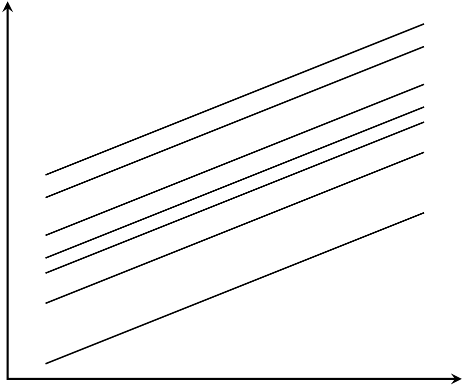
random intercept

- creating a random intercept:



random intercept only, positive linear trend

- a random-intercept-only model assumes that all individuals follow the same trend, but with a different initial point (intercept)



R code

- when using the sem() or cfa() fitting functions, you need to manually set the intercepts of the observed repeated variables to zero, and free the latent intercept:

```
> model <- '
+     # random intercept
+     int =~ 1*y1 + 1*y2 + 1*y3 + 1*y4 + 1*y5
+
+     # zero intercepts
+     y1 + y2 + y3 + y4 + y5 ~ 0*1
+
+     # free latent intercept
+     int ~ 1
+ '
```

- the growth() fitting function does this automatically (for all latent variables):

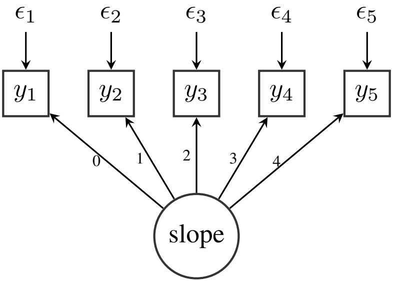
```
> model <- '
+     # random intercept
+     int =~ 1*y1 + 1*y2 + 1*y3 + 1*y4 + 1*y5
+ '
```

- when both ‘regular’ latent variables, and ‘random effects’ are used in the same model, it is perhaps better to use the lavaan() function:

```
> model <- '
+     # random intercept
+     int =~ 1*y1 + 1*y2 + 1*y3 + 1*y4 + 1*y5
+
+     # free latent intercept and variance
+     int ~ 1
+     int ~~ int
+
+     # add residual variances
+     y1 ~~ y1; y2 ~~ y2; y3 ~~ y3; y4 ~~ y4; y5 ~~ y5
+ '
```

random slope

- creating a random slope:

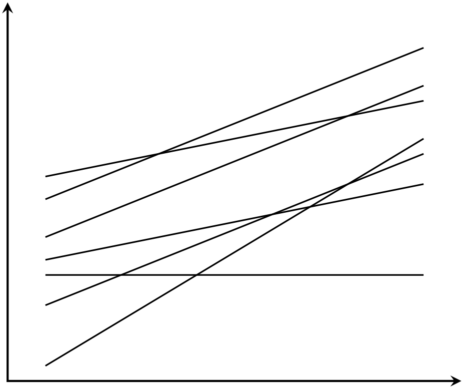


- here, the ‘reference’ point is the first time point; another coding scheme (-4, -3, -2, -1, 0) treats the last time point as the reference point

- this will not affect model fit, but it will change the interpretation of the parameters

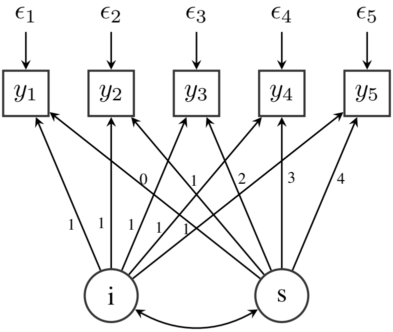
random intercept and random slope

- different intercepts, different slopes



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

real-world example revisted: antisocial behavioral

- two-factor (intercept and slope) growth curve model

```
> model <- '
+   # intercept
+   i =~ 1*anti1 + 1*anti2 + 1*anti3 + 1*anti4
+   i ~ 1  # mean intercept (fixed effect)
+   i ~~ i # variance random intercept
+
+   # slope
+   s= ~ 0*anti1 + 1*anti2 + 2*anti3 + 3*anti4
+   s ~ 1  # mean slope (fixed effect)
+   s ~~ s # variance random slope
+
+   # unequal residual variances
+   anti1 ~~ anti1
+   anti2 ~~ anti2
+   anti3 ~~ anti3
+   anti4 ~~ anti4
+ '
> fit <- lavaan(model, sample.cov = COV, sample.mean = MEANS,
+               sample.nobs = 180, mimic = "EQS")
> summary(fit, fit.measures = TRUE)

lavaan (0.6-1.1133) converged normally after  25 iterations
```

```
  Number of observations                           180

  Estimator                                         ML
  Model Fit Test Statistic                      14.810
  Degrees of freedom                                 6
  P-value (Chi-square)                           0.022

Model test baseline model:

  Minimum Function Test Statistic              227.618
  Degrees of freedom                                 6
  P-value                                        0.000

User model versus baseline model:

  Comparative Fit Index (CFI)                    0.960
  Tucker-Lewis Index (TLI)                       0.960

Loglikelihood and Information Criteria:

  Loglikelihood user model (H0)              -1432.849
  Loglikelihood unrestricted model (H1)      -1425.402

  Number of free parameters                          8
  Akaike (AIC)                                2881.697
  Bayesian (BIC)                              2907.241
  Sample-size adjusted Bayesian (BIC)         2881.905
```

```
Root Mean Square Error of Approximation:

  RMSEA                                              0.091
  90 Percent Confidence Interval          0.032  0.150
  P-value RMSEA <= 0.05                              0.108

Standardized Root Mean Square Residual:

  SRMR                                               0.065

Parameter Estimates:

  Information                                     Expected
  Standard Errors                                 Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  i =~
    anti1             1.000
    anti2             1.000
    anti3             1.000
    anti4             1.000
  s =~
    anti1             0.000
    anti2             1.000
    anti3             2.000
    anti4             3.000
```

```
Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
    i                 1.733    0.126   13.759    0.000
    s                 0.170    0.056    3.026    0.002
   .anti1             0.000
   .anti2             0.000
   .anti3             0.000
   .anti4             0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
    i                 1.670    0.260    6.430    0.000
    s                 0.198    0.057    3.499    0.000
   .anti1             1.526    0.249    6.137    0.000
   .anti2             2.136    0.274    7.802    0.000
   .anti3             1.648    0.246    6.699    0.000
   .anti4             2.264    0.406    5.572    0.000
```

- fairly good (much better than the autoregressive model!)

7 Multilevel SEM

7.1 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

7.2 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):

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

  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; $\bar{\mathbf{y}}_j$ 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

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

  and

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

- 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

7.3 Two-level SEM in lavaan

- multilevel SEM development started around jan 2017
- implemented in lavaan (0.6-3):
  - 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
  - missing data, random slopes
  - gllamm framework (but more user-friendly)
  - case-wise likelihood approach
  - more levels

lavaan syntax setup for two-level SEM

$\mathbf{\Sigma}_B$

Between

Within

$\mathbf{\Sigma}_W$

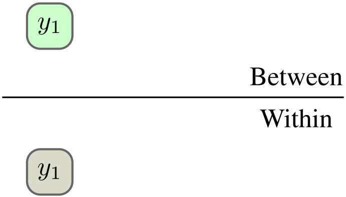
```
model <- '

  level: 1

    # here comes the within level

  level: 2

    # here comes the between level
'

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

example: Demo.twolevel (simulated data)

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

```
> library(lavaan)
> head(round(Demo.twolevel[,c(1:4,7:12)], 3), n = 10)

       y1     y2     y3     y4     x1     x2     x3     w1     w2 cluster
1   0.229  1.356 -0.691  0.803  1.174 -0.623  0.647 -0.248 -0.499       1
2   0.309 -1.862 -2.418  0.766 -1.004 -0.567  0.020 -0.248 -0.499       1
3   0.200 -1.340  0.438  1.197 -0.440 -2.134 -0.459 -0.248 -0.499       1
4   1.045 -0.962 -0.446 -0.203 -0.625 -0.337  1.285 -0.248 -0.499       1
5   0.688 -0.457 -0.642  0.990 -0.845 -0.042  1.560 -0.248 -0.499       1
6  -2.069 -0.600  0.315  0.676 -0.783 -0.224 -0.381 -2.322 -0.691       2
7  -0.787 -0.488  1.132 -0.256 -0.178 -0.583  3.748 -2.322 -0.691       2
8   3.454  1.409  0.930  1.280  0.950  0.259  0.709 -2.322 -0.691       2
9   0.599 -0.291 -1.070  1.930 -1.189  0.815 -0.321 -2.322 -0.691       2
10  1.518 -0.283  0.578  0.851  1.379  0.403  2.190 -2.322 -0.691       2
```

model 1: the empty (univariate) model




```
library(lavaan)

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

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

summary(fit, nd = 4)
```

lavaan output (parameter estimates only)

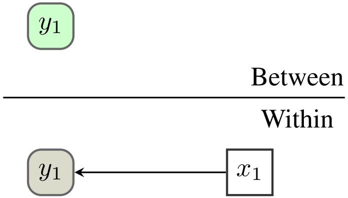
```
Level 1 [within]:

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
```

model 2: simple twolevel regression (predictor within)

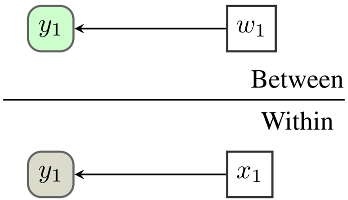



```
model <- '

  level: 1

    y1 ~ x1

  level: 2

    y1 ~~ y1

'



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



summary(fit, nd = 4)
```

lavaan output (parameter estimates only)

```
Level 1 [within]:

Regressions:
                    Estimate   Std.Err   z-value   P(>|z|)
  y1 ~
    x1                0.4944    0.0276   17.8803    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 [cluster]:

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

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

model 3: simple twolevel regression (within + between predictor)



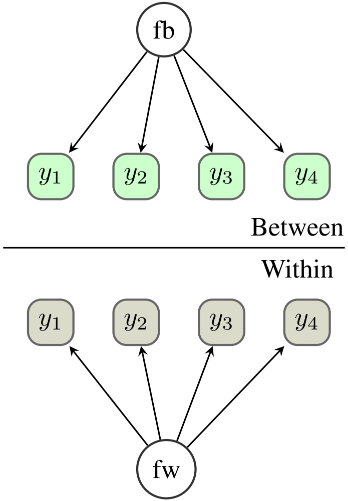

```
model <- '
  level: 1
    y1 ~ x1
  level: 2
    y1 ~ w1
'

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

summary(fit, nd = 4)
```

lavaan output

```
lavaan 0.6-3 ended normally after 21 iterations

  Optimization method                           NLMINB
  Number of free parameters                          5

  Number of observations                          2500
  Number of clusters [cluster]                     200

  Estimator                                         ML
  Model Fit Test Statistic                       0.000
  Degrees of freedom                                 0
  Minimum Function Value               1.3852419442005

Parameter Estimates:

  Information                                 Observed
  Observed information based on                Hessian
  Standard Errors                             Standard


Level 1 [within]:

Regressions:
                   Estimate   Std.Err   z-value   P(>|z|)
  y1 ~
    x1               0.4939    0.0276   17.8658    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 [cluster]:

Regressions:
                   Estimate  Std.Err  z-value  P(>|z|)
  y1 ~
    w1               0.1607   0.0787   2.0416   0.0412

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.0148   0.0738   0.2010   0.8407

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1               0.9128   0.1074   8.5006   0.0000
```

model 4: one-factor model at both levels




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

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

lavaan output

```
> summary(fit)

lavaan 0.6-3 ended normally after 44 iterations

  Optimization method                           NLMINB
  Number of free parameters                         20

  Number of observations                          2500
  Number of clusters [cluster]                     200

  Estimator                                         ML
  Model Fit Test Statistic                       1.274
  Degrees of freedom                                 4
  P-value (Chi-square)                           0.866

Parameter Estimates:

  Information                                 Observed
  Observed information based on                Hessian
  Standard Errors                             Standard


Level 1 [within]:

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

```
  fw =~
    y1                1.000
    y2                0.751    0.042   18.051    0.000
    y3                0.713    0.040   18.034    0.000
    y4                0.315    0.028   11.189    0.000

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

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


Level 2 [cluster]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fb =~
```

```
    y1                 1.000
    y2                 0.714    0.056   12.801    0.000
    y3                 0.579    0.050   11.474    0.000
    y4                 0.057    0.094    0.611    0.541

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

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

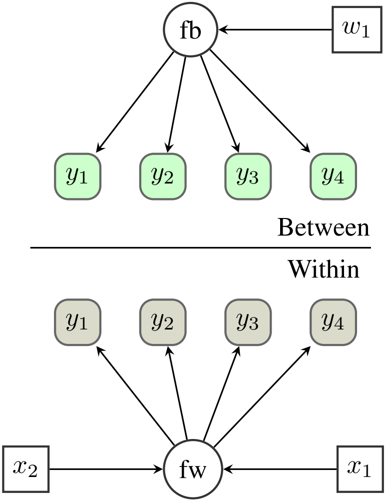
more output

```
> fitMeasures(fit)

           npar                fmin               chisq                  df
         20.000               2.904               1.274               4.000
         pvalue      baseline.chisq         baseline.df     baseline.pvalue
          0.866            1510.108              12.000               0.000
            cfi                 tli                nnfi                 rfi
          1.000               1.005               1.005               0.997
            nfi                pnfi                 ifi                 rni
          0.999               0.333               1.002               1.002
           logl   unrestricted.logl                 aic                 bic
     -16448.595          -16447.958           32937.191           33053.672
         ntotal                bic2               rmsea      rmsea.ci.lower
       2500.000           32990.127               0.000               0.000
 rmsea.ci.upper        rmsea.pvalue                srmr         srmr_within
          0.016               1.000               0.020               0.001
   srmr_between
          0.018

> lavInspect(fit, "h1")

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

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

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


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

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

> lavInspect(fit, "implied")

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

```
y4 0.331 0.248 0.236 1.184

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


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

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

> lavInspect(fit, "icc")

   y1    y2    y3    y4
0.321 0.256 0.223 0.495
```

model 5: adding covariates (no output)




```
model <- '

    level: 1

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

    level: 2

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

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

7.4 Evaluating model fit

- if no random slopes are involved, we can fit an unrestricted (saturated) model: we estimate all the elements of $\boldsymbol{\Sigma}_W$, $\boldsymbol{\Sigma}_B$ 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 model separately for each level, and leave the other level saturated

7.5 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)

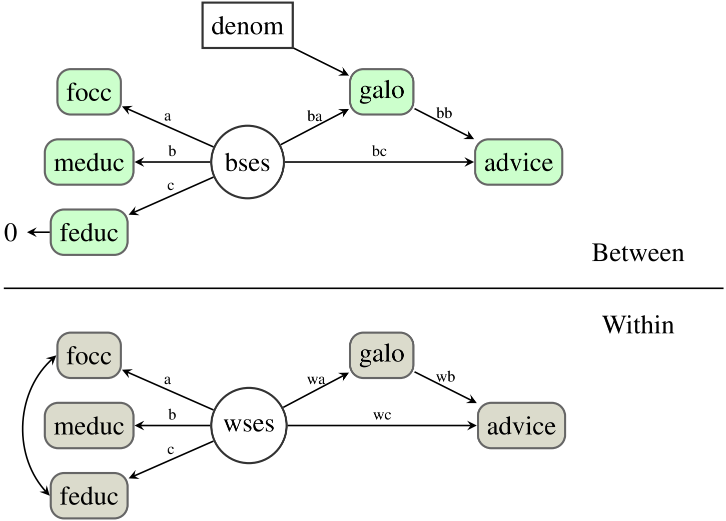
- at the school level, we have one variable: the school's denomination (denom) coded as 1=Protestant, 2=Nondenominational, 3=Catholic

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

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)
```

7.6 Alternative approaches to analyze multilevel data

- 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

the 'wide data' approach

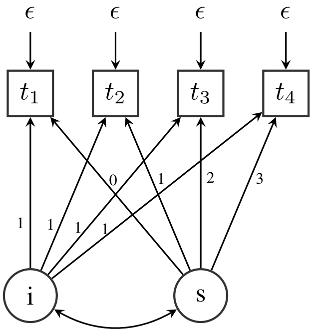
- wonderful paper about this:

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

- main idea: use single-level SEM software to fit multilevel 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
  - typical example: growth curve model
  - the cluster sizes are (very) small
  - the number of variables (per unit) is relatively small

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, header = FALSE, nd = 4)

Parameter Estimates:

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

Latent Variables:
                   Estimate   Std.Err   z-value   P(>|z|)
  int =~
    t1               1.0000
    t2               1.0000
    t3               1.0000
    t4               1.0000
  slope =~
    t1               0.0000
    t2               1.0000
    t3               2.0000
    t4               3.0000

Covariances:
                   Estimate   Std.Err   z-value   P(>|z|)
  int ~~
    slope            0.6267    0.0687    9.1288    0.0000
```

```
Intercepts:
                   Estimate   Std.Err   z-value   P(>|z|)
    int              0.6172    0.0769    8.0286    0.0000
    slope            1.0052    0.0419   24.0128    0.0000
   .t1               0.0000
   .t2               0.0000
   .t3               0.0000
   .t4               0.0000

Variances:
                   Estimate   Std.Err   z-value   P(>|z|)
    int              1.9279    0.1685   11.4388    0.0000
    slope            0.5765    0.0500   11.5402    0.0000
   .t1        (v1)   0.6223    0.0311   20.0000    0.0000
   .t2        (v1)   0.6223    0.0311   20.0000    0.0000
   .t3        (v1)   0.6223    0.0311   20.0000    0.0000
   .t4        (v1)   0.6223    0.0311   20.0000    0.0000
```

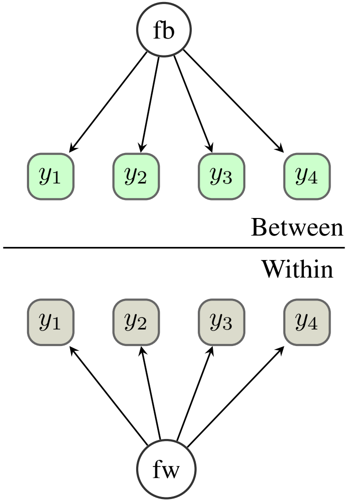
R code: using lmer

```
> # 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: 1-factor model, cluster size = 3




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

fit <- sem(model,
           data = Demo.twolevel3,
           cluster = "cluster")
```

lavaan output

```
> summary(fit)

lavaan 0.6-3 ended normally after 37 iterations

  Optimization method                           NLMINB
  Number of free parameters                         20

  Number of observations                           600
  Number of clusters [cluster]                     200

  Estimator                                         ML
  Model Fit Test Statistic                       3.271
  Degrees of freedom                                 4
  P-value (Chi-square)                           0.514

Parameter Estimates:

  Information                                 Observed
  Observed information based on                Hessian
  Standard Errors                             Standard


Level 1 [within]:

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

```
  fw =~
    y1                1.000
    y2                0.692    0.087    7.922    0.000
    y3                0.599    0.080    7.453    0.000
    y4                0.286    0.056    5.071    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                0.703    0.152    4.627    0.000
   .y2                1.047    0.102   10.309    0.000
   .y3                1.045    0.093   11.264    0.000
   .y4                1.065    0.078   13.666    0.000
    fw                1.292    0.196    6.606    0.000


Level 2 [cluster]:

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fb =~
```

```
    y1                1.000
    y2                0.825    0.142    5.828    0.000
    y3                0.554    0.099    5.613    0.000
    y4                0.219    0.136    1.608    0.108

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.066    0.088    0.748    0.454
   .y2               -0.007    0.077   -0.095    0.924
   .y3               -0.089    0.063   -1.419    0.156
   .y4                0.053    0.088    0.599    0.549
    fb                0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .y1                0.132    0.109    1.208    0.227
   .y2                0.110    0.088    1.257    0.209
   .y3                0.062    0.058    1.068    0.286
   .y4                1.110    0.150    7.403    0.000
    fb                0.752    0.187    4.022    0.000
```

convert data long to wide (easy way)

```
> nvar <- 4
> cluster.size <- 3
> nclusters <- 200
> wideData <- matrix(lav_matrix_vecr(Demo.twolevel3[,1:nvar]),
                     nrow = nclusters,
                     ncol = cluster.size*nvar, byrow = TRUE)
> wideData <- as.data.frame(wideData)
> names(wideData) <- paste(rep(c("y1","y2","y3","y4"), cluster.size),
                           rep(1:cluster.size, each = nvar), sep = ".")
> head(wideData)
```

```
        y1.1       y2.1        y3.1       y4.1       y1.2        y2.2
1  0.2293216  1.3555232 -0.69117022  0.8028079  0.3085801 -1.86243965
2 -2.0687644 -0.5997856  0.31484184  0.6764432 -0.7873959 -0.48754215
3 -2.1695595 -1.9343478 -1.64821625 -0.3379444 -2.8947225 -1.96586346
4 -1.8371725  0.6392386 -1.45754960  1.5419783 -1.4253655 -0.58169082
5  1.6812553 -0.5118063 -0.06997512  2.0456384 -1.0448336  0.02769213
6 -2.0189349 -0.8825048  0.23228017  1.1865589  1.3220591  1.00622305
         y3.2       y4.2       y1.3       y2.3       y3.3      y4.3
1 -2.41797825  0.7659289  0.2004934 -1.3400514  0.4376087  1.197419
2  1.13215273 -0.2564694  3.4544134  1.4087639  0.9297677  1.280142
3 -2.64146609  1.1436848 -2.9211906 -2.0192952 -1.7193465 -1.114822
4 -0.85232621  0.5417006  0.6245559  1.3408396 -0.8370214  1.296544
5  0.03396535  0.9999995  1.1553863 -0.7218829  0.9744591  3.389794
6 -0.64796238  1.0826751  0.6538137 -0.4748541  0.4157787 -1.371790
```

wide-format syntax

```
> 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

    # 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 0.6-3 ended normally after 32 iterations

  Optimization method                           NLMINB
  Number of free parameters                         36
  Number of equality constraints                    16

  Number of observations                           200

  Estimator                                         ML
```

```
  Model Fit Test Statistic                           78.077
  Degrees of freedom                                     70
  P-value (Chi-square)                                0.238

Parameter Estimates:

  Information                                      Observed
  Observed information based on                     Hessian
  Standard Errors                                  Standard

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fw1 =~
    y1.1              1.000
    y2.1     (lw2)    0.692    0.087    7.922    0.000
    y3.1     (lw3)    0.599    0.080    7.453    0.000
    y4.1     (lw4)    0.286    0.056    5.071    0.000
  fw2 =~
    y1.2              1.000
    y2.2     (lw2)    0.692    0.087    7.922    0.000
    y3.2     (lw3)    0.599    0.080    7.453    0.000
    y4.2     (lw4)    0.286    0.056    5.071    0.000
  fw3 =~
    y1.3              1.000
    y2.3     (lw2)    0.692    0.087    7.922    0.000
    y3.3     (lw3)    0.599    0.080    7.453    0.000
    y4.3     (lw4)    0.286    0.056    5.071    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               0.825    0.142    5.828    0.000
    by3               0.554    0.099    5.613    0.000
    by4               0.219    0.136    1.608    0.108

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               0.066    0.088    0.748    0.454
    by2              -0.007    0.077   -0.095    0.924
    by3              -0.089    0.063   -1.419    0.156
    by4               0.053    0.088    0.599    0.549
    fw1               0.000
    fw2               0.000
    fw3               0.000
```

```
    fb                  0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
    fw1       (fvw)    1.292    0.196    6.606    0.000
    fw2       (fvw)    1.292    0.196    6.606    0.000
    fw3       (fvw)    1.292    0.196    6.606    0.000
   .y1.1      (rw1)    0.703    0.152    4.627    0.000
   .y1.2      (rw1)    0.703    0.152    4.627    0.000
   .y1.3      (rw1)    0.703    0.152    4.627    0.000
   .y2.1      (rw2)    1.047    0.102   10.309    0.000
   .y2.2      (rw2)    1.047    0.102   10.309    0.000
   .y2.3      (rw2)    1.047    0.102   10.309    0.000
   .y3.1      (rw3)    1.045    0.093   11.264    0.000
   .y3.2      (rw3)    1.045    0.093   11.264    0.000
   .y3.3      (rw3)    1.045    0.093   11.264    0.000
   .y4.1      (rw4)    1.065    0.078   13.666    0.000
   .y4.2      (rw4)    1.065    0.078   13.666    0.000
   .y4.3      (rw4)    1.065    0.078   13.666    0.000
    by1                0.132    0.109    1.208    0.227
    by2                0.110    0.088    1.257    0.209
    by3                0.062    0.058    1.068    0.286
    by4                1.110    0.150    7.403    0.000
    fb                 0.752    0.187    4.022    0.000
```

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:

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

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
```

7.7 Comments

- 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
  - mediation and moderation
  - random slopes
  - categorical outcomes
  - missing data
  - the gllamm framework

Thank you for attending this workshop!