Structural Equation Modeling with lavaan

Yves Rosseel

Department of Data Analysis

Ghent University

Summer School – Using R for personality research

August 23–28, 2014

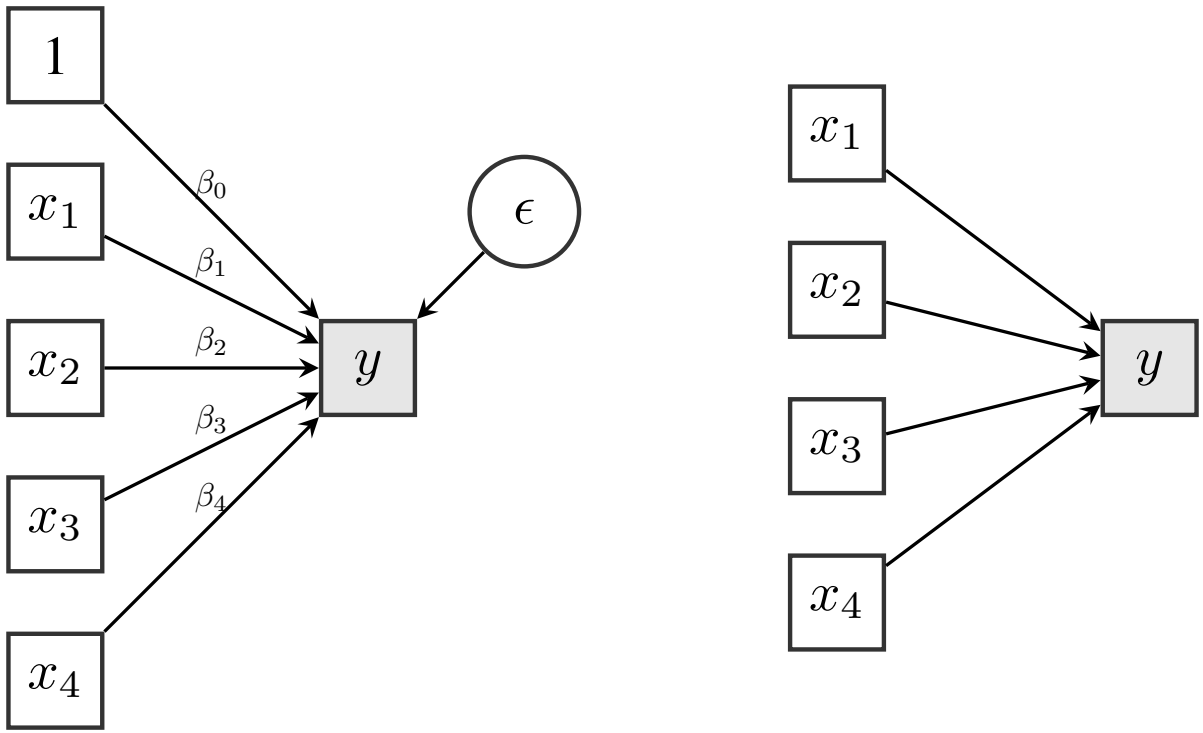
Bertinoro, Italy

Contents

1 Introduction to SEM

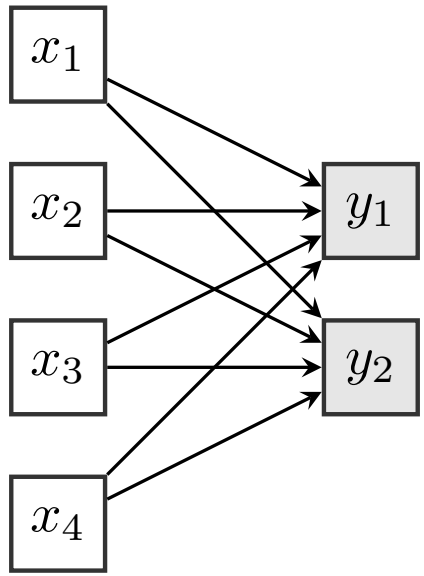
1.1 From regression to structural equation modeling

univariate linear regression



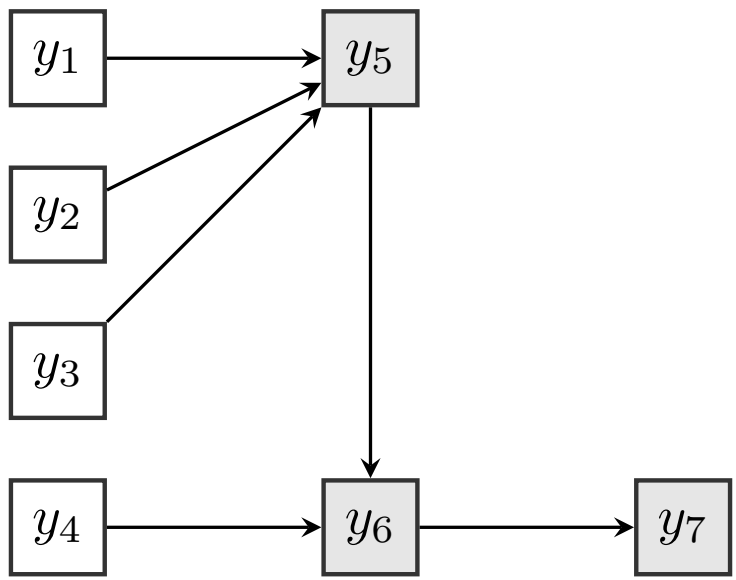
$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$$

multivariate regression



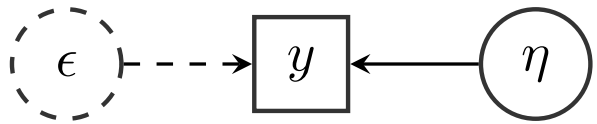
path analysis

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

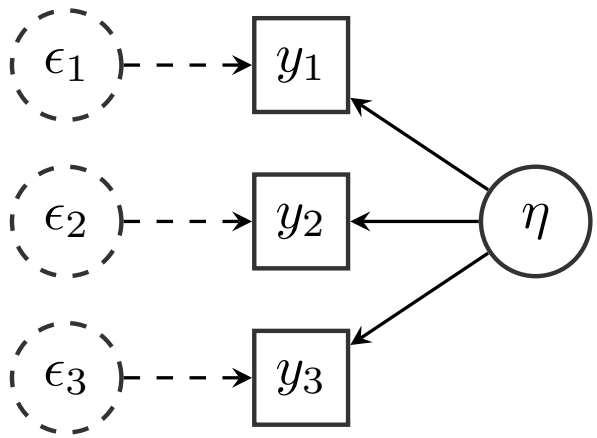


measurement error

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

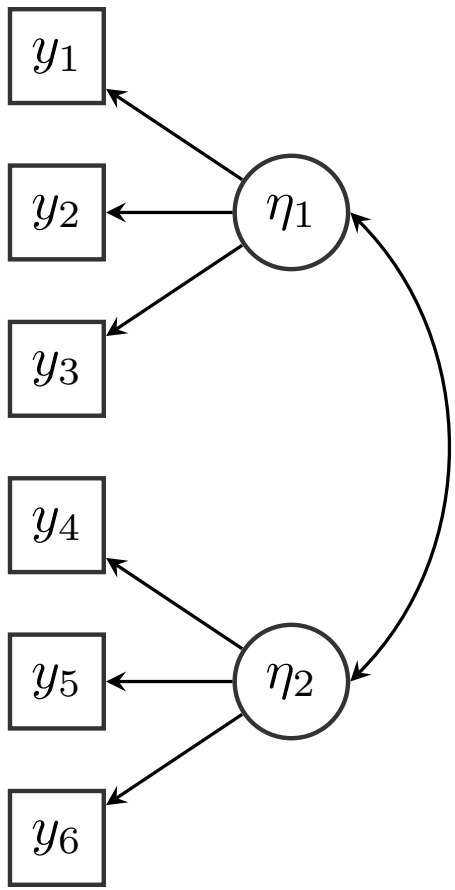


- multiple indicator measurement model



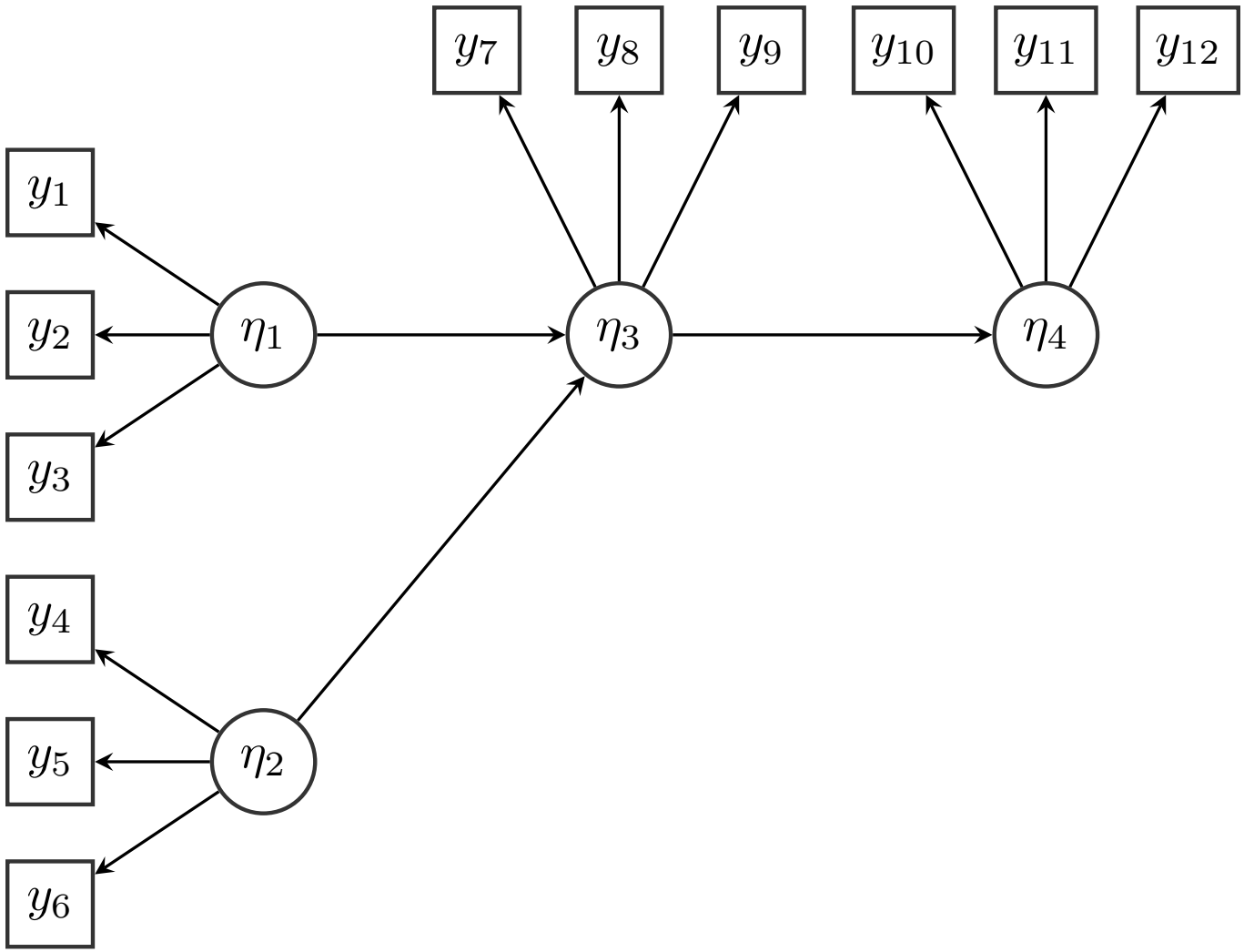
confirmatory factor analysis (CFA)

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



structural equation modeling (SEM)

- path analysis with latent variables

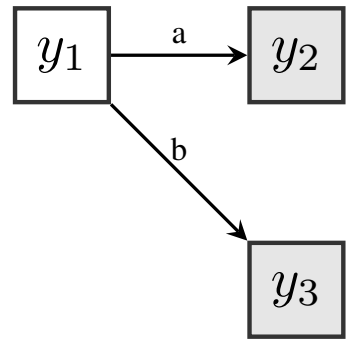


1.2 The model-implied covariance matrix (the essence of SEM)

- 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: $\hat{\Sigma}$
  - for the model-implied mean vector: $\hat{\mu}$
  - ...
- 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 variables, y_1, y_2 and y_3; to explain why they are correlated, we may postulate the following model:

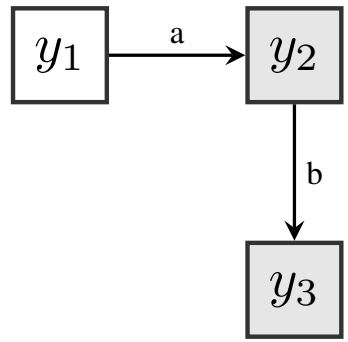


- if we use the following values for the model parameters: $a = 3$ and $b = 5$, $\sigma^2(y_1) = 10$, $\sigma^2(\epsilon_2) = 20$ and $\sigma^2(\epsilon_3) = 30$, then we find (see Appendix):

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

example model-implied covariance matrix (2)

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



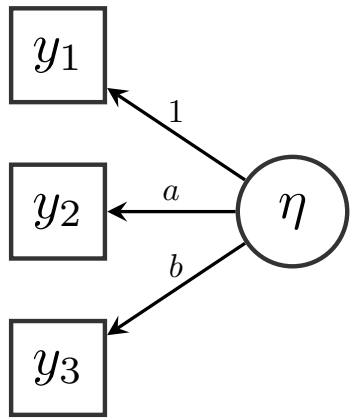
we find

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

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

example model-implied covariance matrix (3)

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



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

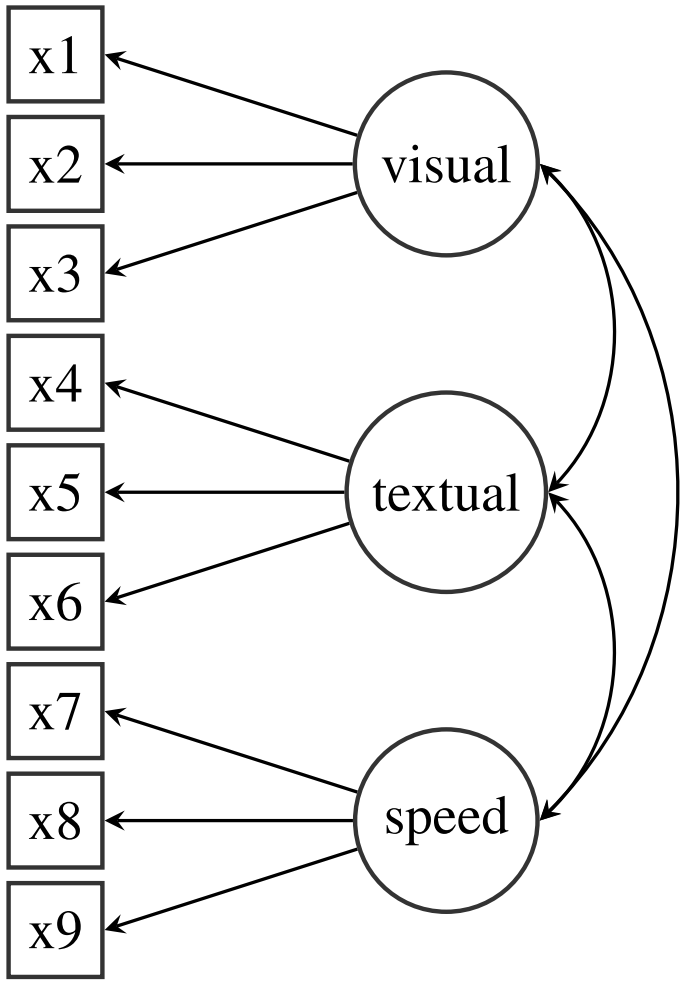
$$\hat{\boldsymbol{\Sigma}} = \begin{bmatrix} 11 & & \\ 4 & 36 & \\ 5 & 20 & 55 \end{bmatrix}$$

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



how does the data look like?

- our dataset contains measures for all *observed variables* in the model
- in many SEM applications, we do not need the full dataset; all we need are summary statistics:
  - the sample-based variance/covariance matrix ($\mathbf{S}$); if there are p variables, the covariance matrix is of size $p \times p$ and contains $p(p+1)/2$ non-duplicated elements
  - the sample size (N)
  - for some SEM applications, we also need the sample-based p-dimensional mean vector
- note that this is true for many statistical techniques (ANOVA, regression, factor analysis, ...)
- but for more advanced applications, we need the full dataset anyway (e.g. if we wish to correct for non-normality, if we have missing values, ...)

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
- the 'covariance matrix' contains all information about the interrelations among the observed variables

observed covariance matrix: 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:

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

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

the standard CFA model: the model implied covariance matrix

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

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

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

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 ($\mathbf{B}$):

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

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

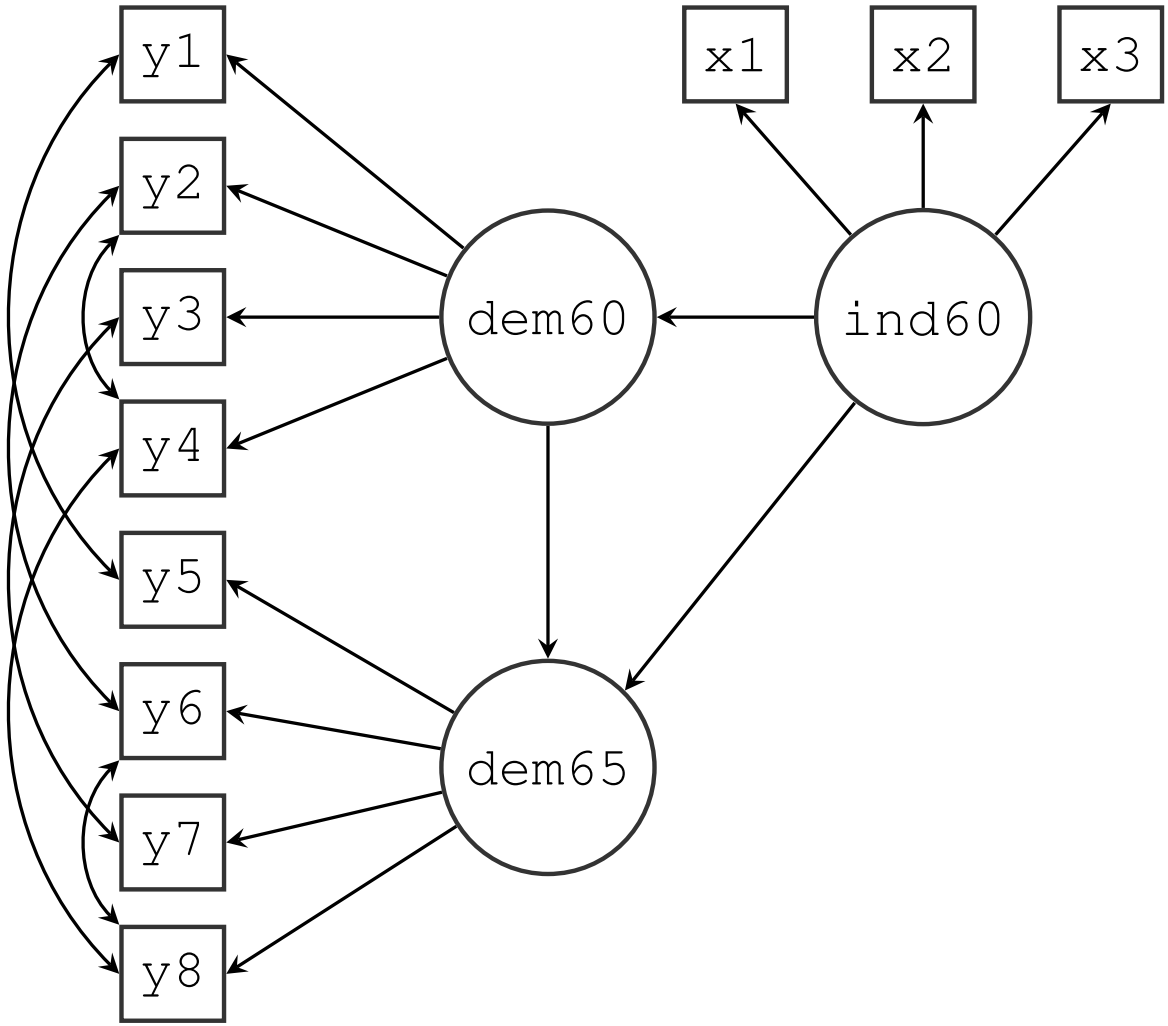
- we need this extended model for
  - second-order CFA
  - MIMIC models
  - SEM models
- in LISREL parlance, this the 'all-y' model

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

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
```

computing the model-implied covariance matrix (optional)

```
# easy way is: fitted(fit)

# manual
attach(inspect(fit, "est"))
IB <- diag(nrow(beta)) - beta
IB.inv <- solve(IB)
Sigma.hat <- lambda %*% IB.inv %*% psi %*% t(IB.inv) %*% t(lambda) + theta
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
attr(,"class")
[1] "lavaan.matrix.symmetric" "matrix"
```

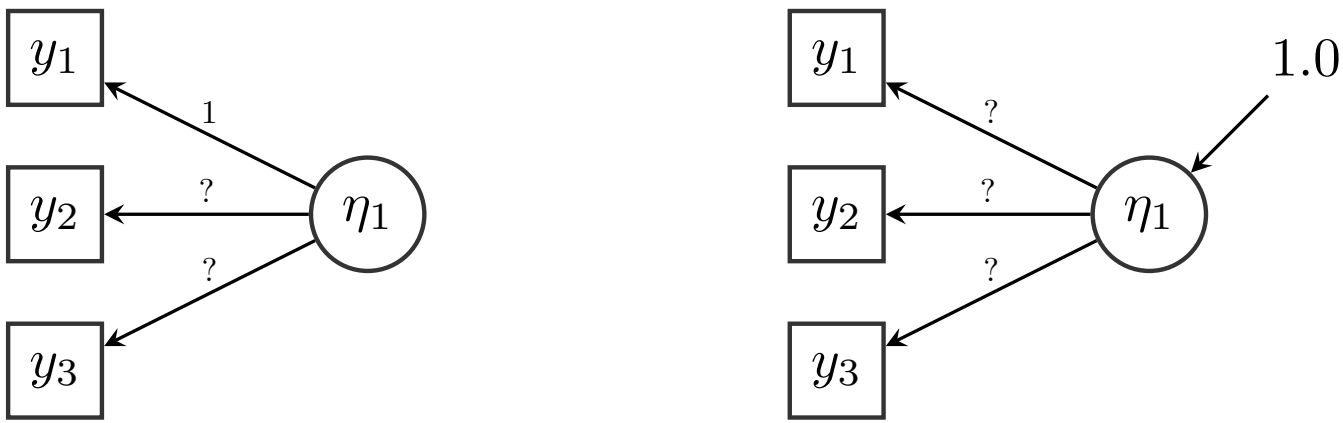
free, fixed and constrained parameters (optional)

- a *free* parameter is free to vary during the iterations of the parameter estimation process until it attains a value that optimizes the fit function (conditional on the other parameters)
- a *fixed* parameter has a prespecified value by hypothesis and remains unchanged during the parameter estimation process
  - e.g. fixing the factor loading of the first indicator to 1.0
  - many elements in the model matrices are fixed to zero
- a *constrained* parameter, is not entirely free, but partially or fully constrained
  - equality constraints: a parameter is (fully) constrained to be equal to another (free) parameter, or a function of (a set of free) parameters
  - inequality constraints: a parameter is partially constrained to be larger than (or smaller than) another parameter, or a function of (a set of free) parameters

setting the metric of the latent variables: UVI of ULI

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

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



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

1.6 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:

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

- 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)

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. (2011). Principles and practice of structural equation modeling (Third Edition). New York: Guilford Press.

Brown, T. A. (2006), Confirmatory Factor Analysis for Applied Research. New York: Guilford Press.

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

Hancock, G. R., & Mueller, R. O. (Eds.). (2006). Structural equation modeling: A second course. Greenwich, CT: Information Age Publishing, Inc.

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

2 Introduction to lavaan

software for SEM: commercial – closed-source

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

software for SEM: non-commercial – open-source

- outside the R ecosystem: gllamm (Stata), ...
- R packages:
  - sem
  - OpenMx
  - lavaan
  - lava

what is lavaan?

- **lavaan** is an R package for latent variable analysis:
  - confirmatory factor analysis: function `cfa()`
  - structural equation modeling: function `sem()`
  - latent curve analysis / growth modeling: function `growth()`
  - general mean/covariance structure modeling: function `lavaan()`
  - support for continuous, binary and ordinal data
- under development, future plans:
  - multilevel SEM, mixture/latent-class SEM, Bayesian SEM
- the long-term goal of **lavaan** is
  1. to implement all the state-of-the-art capabilities that are currently available in commercial packages
  2. to provide a modular and extensible platform that allows for easy implementation and testing of new statistical and modeling ideas

installing lavaan, finding documentation

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

  `http://www.r-project.org`

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

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

- more information about **lavaan**:

  `http://lavaan.org`

- the lavaan paper:

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

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

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

the lavaan ecosystem

- **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

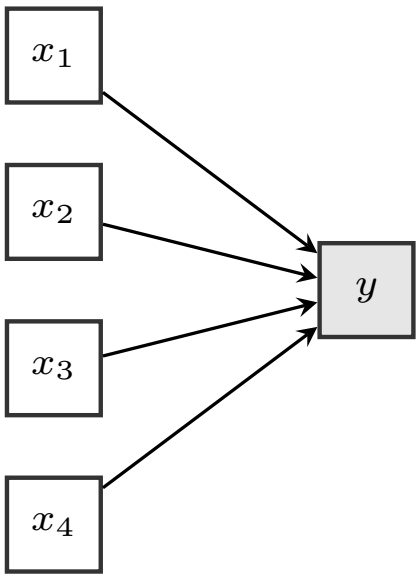
- **simsem** (Sunthud Pornprasertmanit and many others)

  simulation of SEM models

- **semPlot** (Sacha Epskamp)

  visualizations of SEM models

mini intro R: a simple regression




```
# 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:

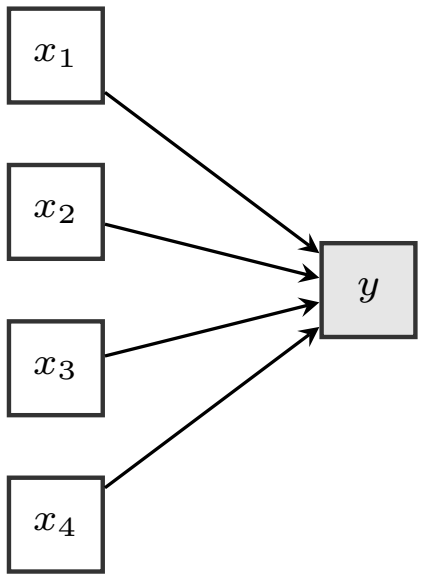
- $y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)$

lm() output artificial data (N=100)

```
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
```

```
# create artificial data
set.seed(1)
x1 <- rnorm(100) * 10; x2 <- rnorm(100) * 10
x3 <- rnorm(100) * 10;  x4 <- rnorm(100) * 10
y <- 100 + 5*x1 + (-2)*x2 + 1*x3 + 0.1*x4 + rnorm(100, sd=40)
myData <- data.frame(y,x1,x2,x3,x4)
```

the lavaan model syntax – a simple regression




```
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)
```

- 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 output

```
lavaan (0.5-13) converged normally after   1 iterations

  Number of observations                           100

  Estimator                                         ML
  Minimum Function Test Statistic                0.000
  Degrees of freedom                                 0
  P-value (Chi-square)                           1.000

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y ~
    x1                5.773    0.511   11.309    0.000
    x2               -1.321    0.479   -2.757    0.006
    x3                1.135    0.446    2.545    0.011
    x4                0.271    0.466    0.581    0.561

Variances:
    y              2075.100  293.463
```

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

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

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

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

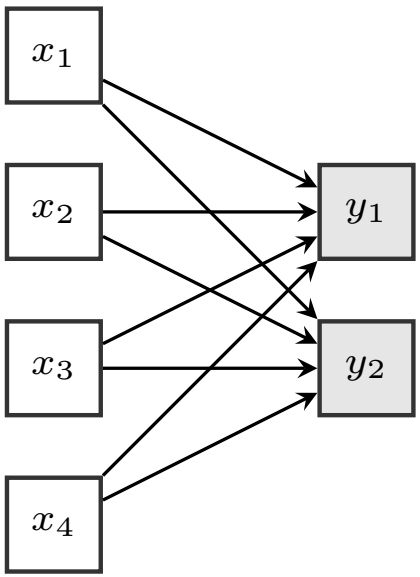
$$\hat{\sigma}_y^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n - (p+1)}$$

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

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

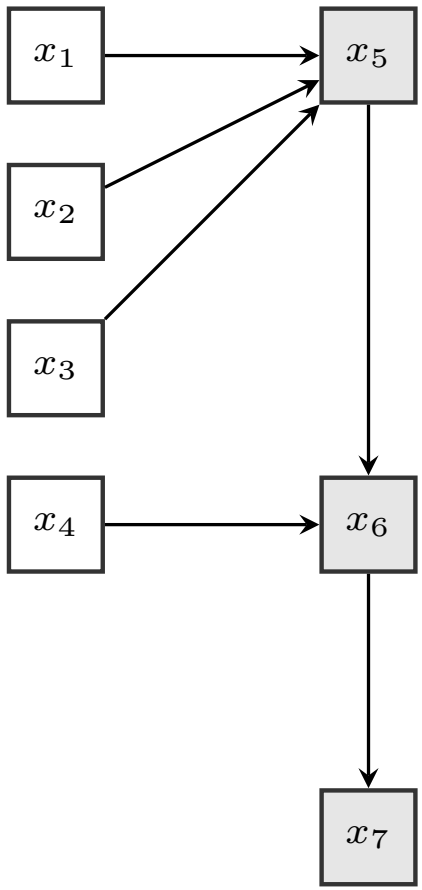
and this affects the standard errors.

the lavaan model syntax – multivariate regression



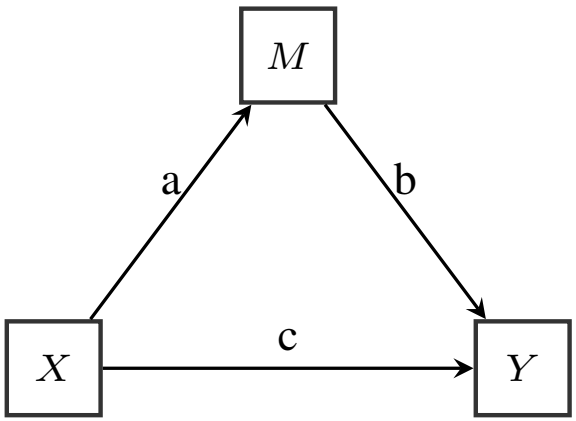

```
myModel <- ' y1 ~ x1 + x2 + x3 + x4
             y2 ~ x1 + x2 + x3 + x4 '
```

the lavaan model syntax – path analysis




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

the lavaan model syntax – mediation analysis

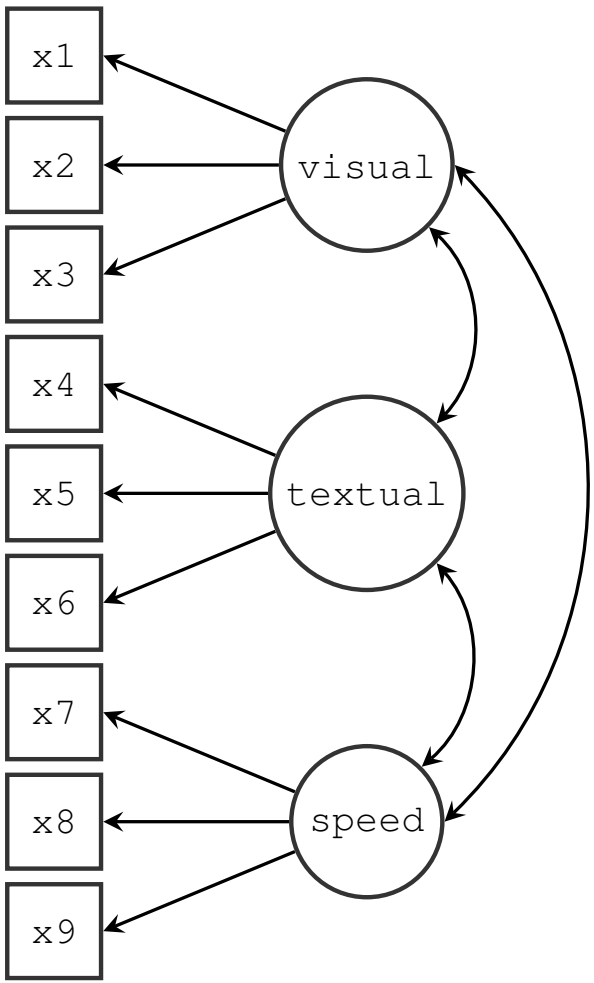



```
myModel <- '
             Y ~ b*M + c*X
             M ~ a*X

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

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

summary(fit)
```

output

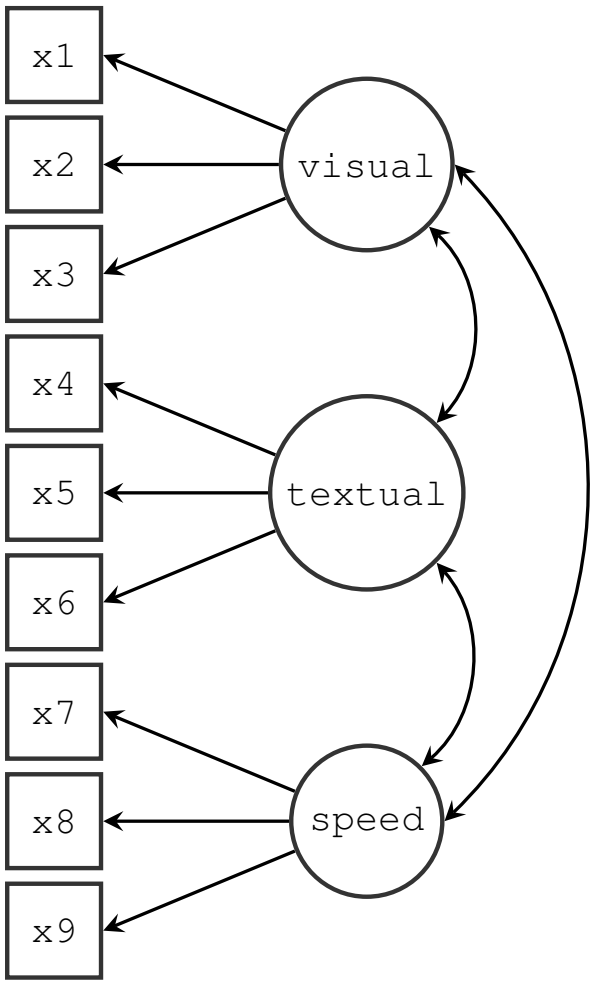
```
...

Parameter estimates:

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

                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  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:
    Y                108.700   17.747
    M                105.408   16.556

Defined parameters:
    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)
```

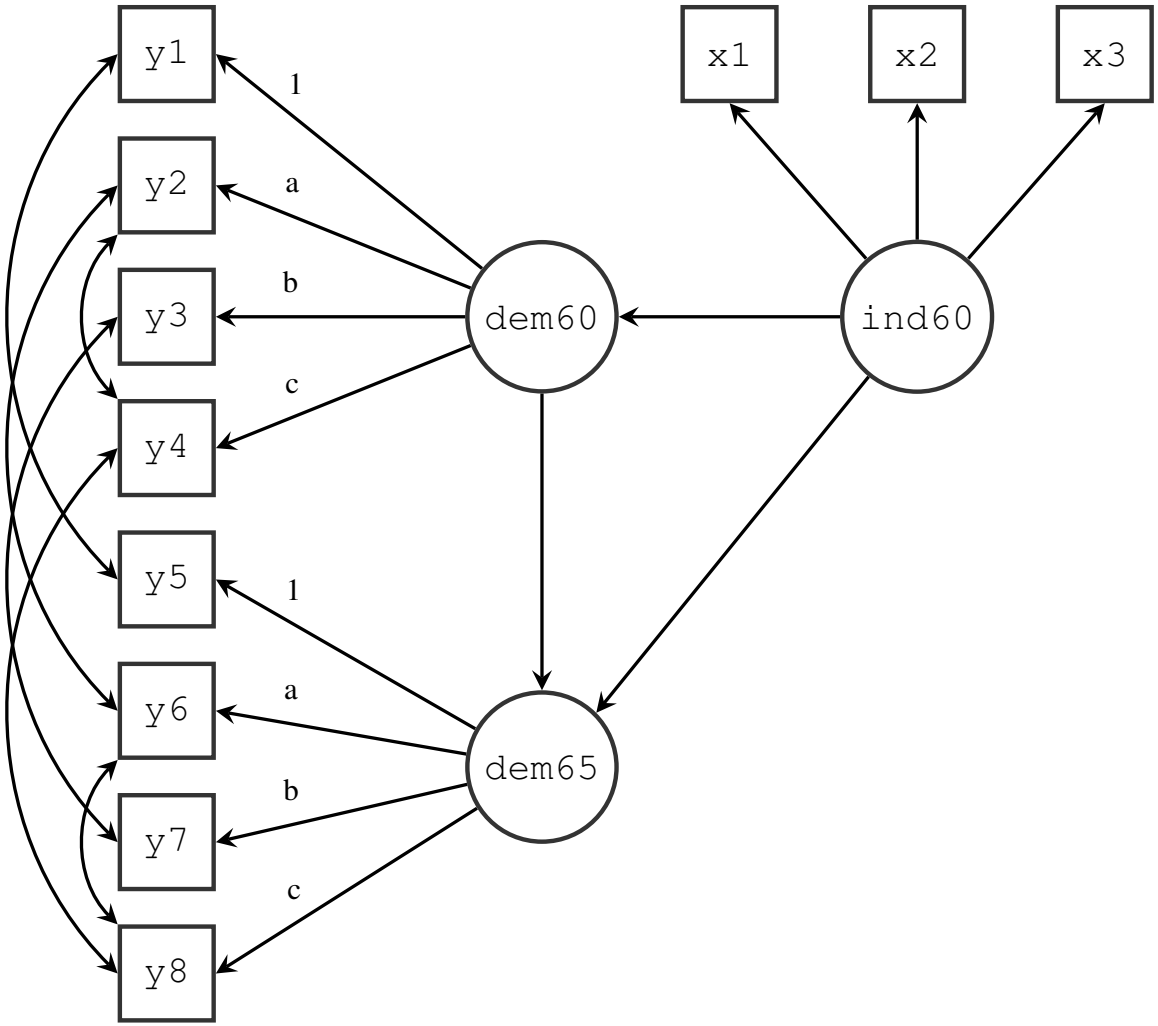
output

```
lavaan (0.5-16) converged normally after 35 iterations

  Number of observations                           301

  Estimator                                         ML
  Minimum Function 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
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  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:
  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

Variances:
    x1                0.549    0.114
    x2                1.134    0.102
    x3                0.844    0.091
    x4                0.371    0.048
    x5                0.446    0.058
    x6                0.356    0.043
    x7                0.799    0.081
    x8                0.488    0.074
    x9                0.566    0.071
    visual            0.809    0.145
    textual           0.979    0.112
    speed             0.384    0.086
```

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 (0.5-16) converged normally after  61 iterations

  Number of observations                            75

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

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)   Std.lv  Std.all
Latent variables:
  ind60 =~
    x1                1.000                               0.670    0.920
    x2                2.180    0.138   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:
  dem60 ~
    ind60             1.471    0.392    3.750    0.000        0.448    0.448
  dem65 ~
    ind60             0.600    0.226    2.660    0.008        0.187    0.187
    dem60             0.865    0.075   11.554    0.000        0.884    0.884

Covariances:
  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:
    x1                0.081    0.019                          0.081    0.154
 ...
```

3 Further topics

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

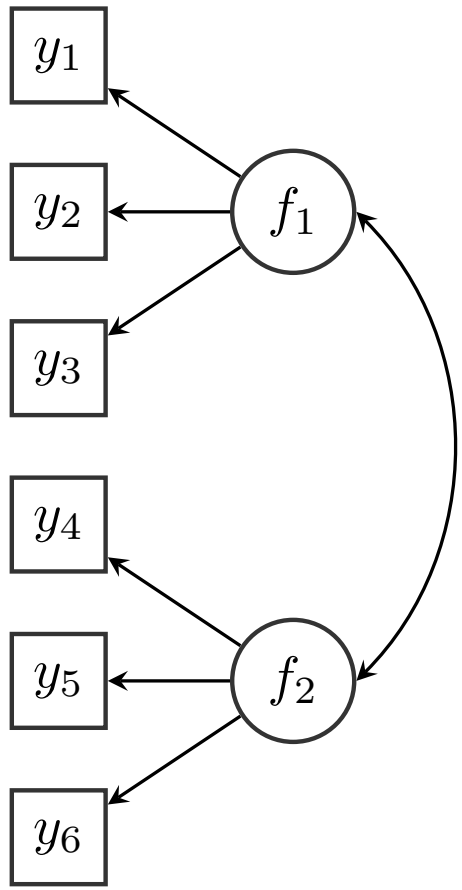
output meanstructure = TRUE

```
lavaan (0.5-16) converged normally after  35 iterations

  Number of observations                           301

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

Parameter estimates:

  Information                                 Expected
  Standard Errors                             Standard

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  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:
  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:
    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:
    x1                0.549    0.114
    x2                1.134    0.102
    ...
```

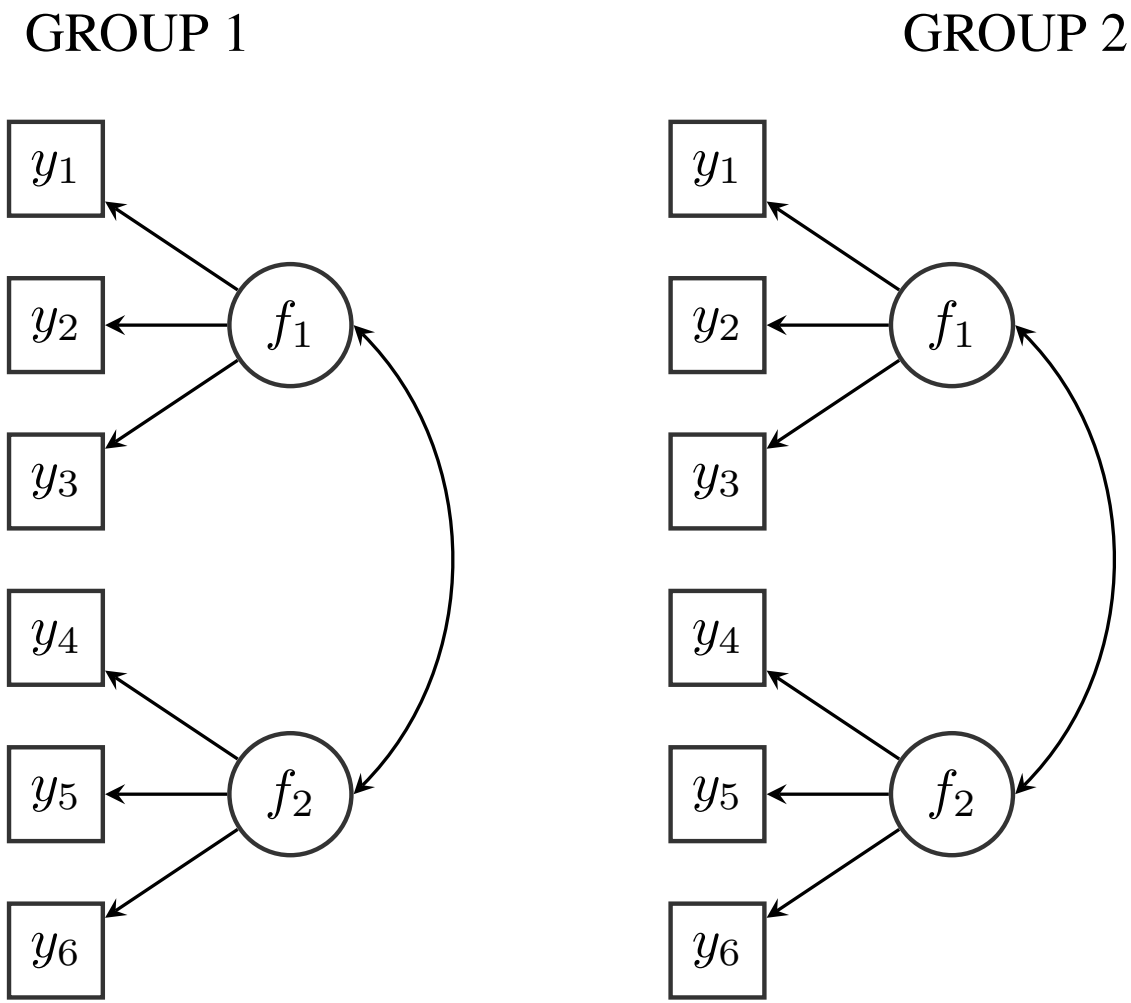
3.2 Multiple groups

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

- model comparison tests: compare the current model with the previous one (either using a chi-square difference test or looking at other fit measures)
- *partial* measurement invariance: most but not all parameters are equivalent across groups
- 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.
  - Cheung, G.W., and Rensvold, R.B. (2000). Evaluating goodness-of-fit indices for testing measurement invariance. *Structural Equation Modeling, 9*, 233-255.
  - 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.

measurement Invariance in lavaan

```
# model 1: configural invariance
fit1 <- cfa(HS.model, data=HolzingerSwineford1939, group="school")

# model 2: weak invariance
fit2 <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
            group.equal="loadings")

# model 3: strong invariance
fit3 <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
            group.equal=c("loadings", "intercepts"))

# model 4: equal loadings + intercepts + means
fit4 <- cfa(HS.model, data=HolzingerSwineford1939, group="school",
            group.equal=c("loadings", "intercepts", "means"))
```

comparing two (nested) models: the anova() function

```
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
```

measurement invariance tests – all together

```
> library(semTools)
> measurementInvariance(HS.model, data=HolzingerSwineford1939,
                        group="school", strict=FALSE)

Measurement invariance tests:

Model 1: configural invariance:
   chisq       df   pvalue      cfi    rmsea      bic
 115.851   48.000    0.000    0.923    0.097 7706.822

Model 2: weak invariance (equal loadings):
   chisq       df   pvalue      cfi    rmsea      bic
 124.044   54.000    0.000    0.921    0.093 7680.771

[Model 1 versus model 2]
  delta.chisq      delta.df delta.p.value     delta.cfi
        8.192         6.000         0.224         0.002

Model 3: strong invariance (equal loadings + intercepts):
   chisq       df   pvalue      cfi    rmsea      bic
 164.103   60.000    0.000    0.882    0.107 7686.588

[Model 1 versus model 3]
  delta.chisq      delta.df delta.p.value     delta.cfi
       48.251        12.000         0.000         0.041
```

```
[Model 2 versus model 3]
  delta.chisq      delta.df delta.p.value     delta.cfi
       40.059         6.000         0.000         0.038

Model 4: equal loadings + intercepts + means:
   chisq       df   pvalue      cfi    rmsea      bic
 204.605   63.000    0.000    0.840    0.122 7709.969

[Model 1 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       88.754        15.000         0.000         0.083

[Model 3 versus model 4]
  delta.chisq      delta.df delta.p.value     delta.cfi
       40.502         3.000         0.000         0.042
```

3.4 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
  - in general, almost all standard statistical methods are no longer valid
  - we can only try to understand the missingness mechanism at hand, and take this into account when modeling the data

missing data in lavaan

- in lavaan 0.5, the default is listwise deletion (but this may change in lavaan 0.6)

```
lavaan (0.5-16) converged normally after  39 iterations

                                                  Used       Total
  Number of observations                           156         301
```

- if the missing mechanism is MCAR or MAR, lavaan provides provides case-wise (or 'full information') maximum likelihood (FIML) estimation:

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

lavaan (0.5-16) converged normally after  51 iterations

  Number of observations                           301

  Number of missing patterns                        38

  Estimator                                         ML
  Minimum Function Test Statistic               69.056
  Degrees of freedom                                24
  P-value (Chi-square)                           0.000
```

3.5 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)
lavaan (0.5-13) converged normally after  35 iterations

  Number of observations                           301

  Estimator                                         ML      Robust
  Minimum Function 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

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.931       0.925
  Tucker-Lewis Index (TLI)                       0.896       0.887
```

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)
```

3.6 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

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

a limited information approach: the WLSMV estimator

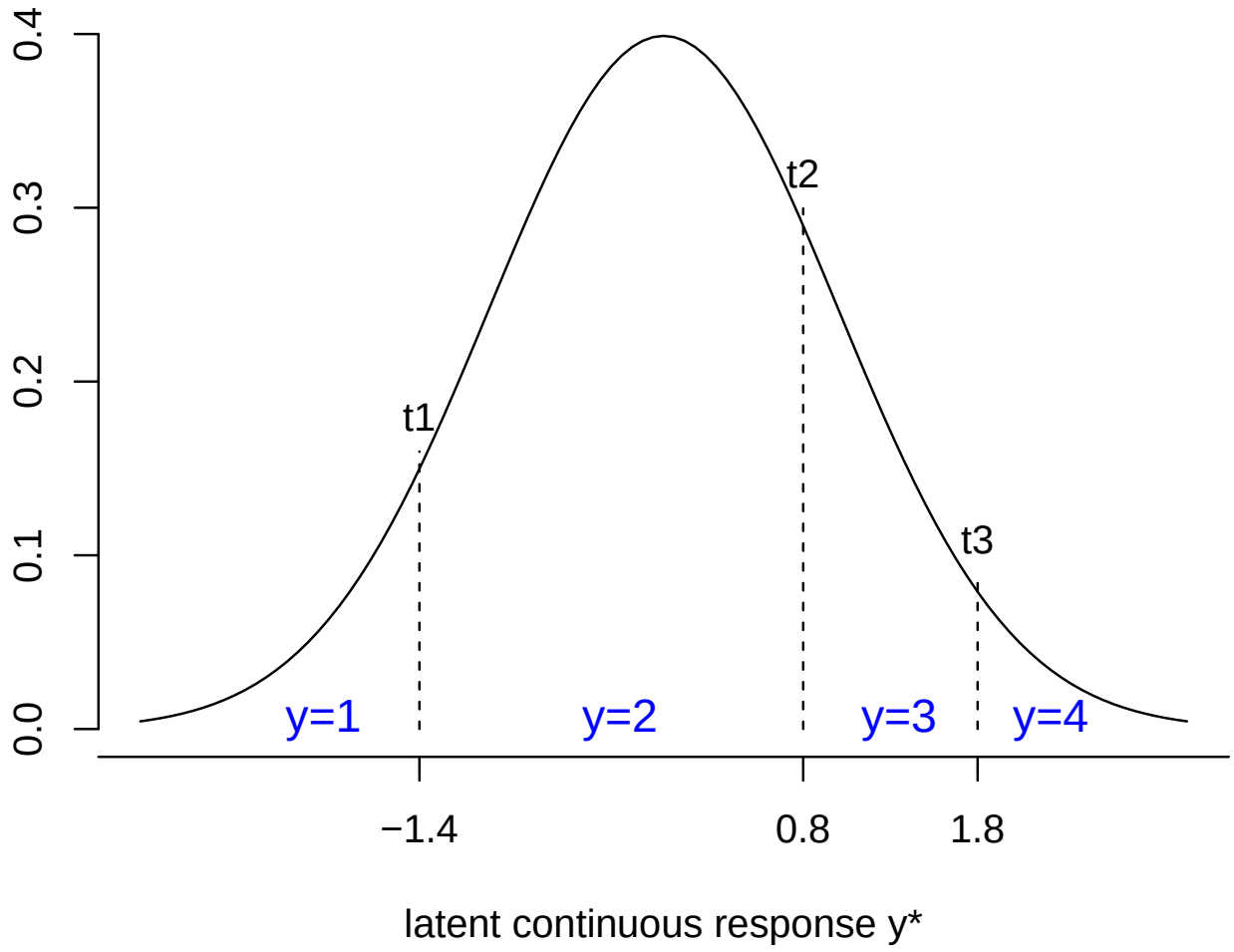
- developed by Bengt Muthén, in a series of papers; the seminal paper is

  > 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 for almost three decades
- 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 7.11
- 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 (1)

- 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 (2)

- estimating the thresholds: maximum likelihood using univariate data
  - if no exogenous variables, this is just

```
# generate 'ordered' data with 4 categories
Y <- sample(1:4, size = 100, replace = TRUE)

prop <- table(Y)/sum(table(Y))
cprop <- c(0, cumsum(prop))

th <- qnorm(cprop)
```

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

```
library(MASS)
X1 <- rnorm(100); X2 <- rnorm(100); X3 <- rnorm(100)

fit <- polr(ordered(Y) ~ X1 + X2 + X3, method = "probit")
fit$zeta
```

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

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

stage 3 – estimating the SEM model

- third stage uses weighted least squares:

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

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

- š contains: thresholds, correlations, optionally regression slopes of exogenous covariates, optionally variances and means of continuous variables

- the weight matrix $\mathbf{W}$ is (a consistent estimator of) the asymptotic covariance matrix of the sample statistics (š)

- robust version: WLSMV
  - use the diagonal of $\mathbf{W}$ only for estimation (DWLS)
  - use the full matrix for inference (standard errors and test statistic)
  - 'MV' stands for the Satterthwaite's mean and variance corrected test statistic

using categorical variables in lavaan

- 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)
```

- 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"))
```

- by default, lavaan will use robust WLS (DWLS + robust standard errors and a scaled-shifted test statistic; this is equivalent to estimator=WLSMV in Mplus)

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))

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

output

```
lavaan (0.5-13) converged normally after  36 iterations

  Number of observations                           301

  Estimator                                       DWLS      Robust
  Minimum Function Test Statistic               30.918      38.546
  Degrees of freedom                                24          24
  P-value (Chi-square)                           0.156       0.030
  Scaling correction factor                                  0.866
  Shift parameter                                            2.861
    for simple second-order correction (Mplus variant)

Model test baseline model:

  Minimum Function Test Statistic              582.533     469.769
  Degrees of freedom                                36          36
  P-value                                        0.000       0.000

Full model versus baseline model:

  Comparative Fit Index (CFI)                    0.987       0.966
  Tucker-Lewis Index (TLI)                       0.981       0.950

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                          0.848       0.598

Parameter estimates:

  Information                                 Expected
  Standard Errors                          Robust.sem

                   Estimate  Std.err  Z-value  P(>|z|)
Latent variables:
  visual =~
    x1                1.000
    x2                0.900    0.188    4.788    0.000
    x3                0.939    0.197    4.766    0.000
  textual =~
   x4                 1.000
    x5                0.976    0.118    8.241    0.000
    x6                1.078    0.125    8.601    0.000
  speed =~
    x7                1.000
    x8                1.569    0.461    3.403    0.001
    x9                1.449    0.409    3.541    0.000

Covariances:
  visual ~~
    textual           0.303    0.061    4.981    0.000
    speed             0.132    0.049    2.700    0.007
  textual ~~
```

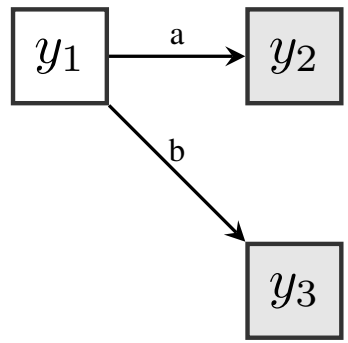
```
    speed             0.076    0.046    1.656    0.098

Intercepts:
    visual            0.000
    textual           0.000
    speed             0.000

Thresholds:
    x1|t1            -0.388    0.074   -5.223    0.000
    x2|t1            -0.054    0.072   -0.748    0.454
    x3|t1             0.318    0.074    4.309    0.000
    x4|t1             0.180    0.073    2.473    0.013
    x5|t1            -0.257    0.073   -3.506    0.000
    x6|t1             1.024    0.088   11.641    0.000
    x7|t1             0.231    0.073    3.162    0.002
    x8|t1             1.128    0.092   12.284    0.000
    x9|t1             0.626    0.078    8.047    0.000

Variances:
    x1                0.592
    x2                0.670
    x3                0.640
    x4                0.303
    x5                0.336
   x6                0.191
    x7                0.778
    x8                0.453
    x9                0.534
```

```
    visual            0.408    0.112
    textual           0.697    0.101
    speed             0.222    0.094

> inspect(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
```

4 Appendices

4.1 Rules about variances and covariances

- consider the following path diagram:



- 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 two equations imply how the three variables are related to each other: there is a relationship between y_2 and y_1, and y_3 and y_1; there is also a relationship between y_2 and y_3 (because, there is a common cause)
- to quantify the (linear) relationship between two variables, we use the *covariance*
- suppose we have already estimated the values of the structural coefficients a and b, can we derive:
  1. the *model-implied covariances* for each pair of variables in the set (y_1, y_2, y_3)?
  2. the *variances* of the (endogenous) variables y_2 and y_3?
  3. what about the variance of the exogenous variable y_1?
- the *model-implied* variance covariance matrix $\hat{\boldsymbol{\Sigma}}$:

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

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:

$$X = a_1X_1 + a_2X_2 + \ldots + a_pX_p \quad \text{and} \quad Y = b_1Y_1 + b_2Y_2 + \ldots + b_qY_q$$

- 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

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

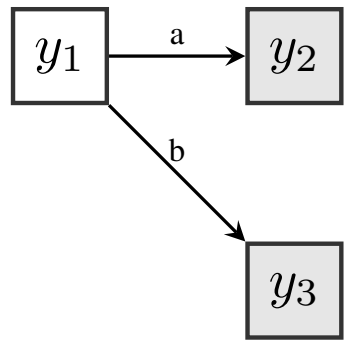
applying the rules

- 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:

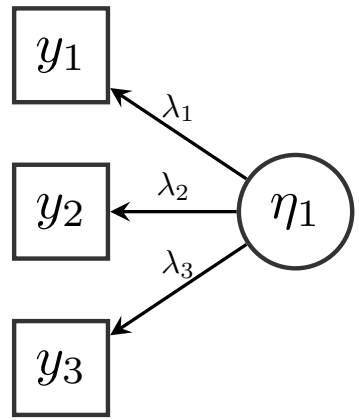


we find

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

the model-implied covariance in a one-factor CFA

- consider this simple one-factor CFA with three indicators:



- 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.

4.2 lavaan: a brief user's guide

syntax: main formula types, and other operators

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

- 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 '
```

user-friendly fitting functions

- `cfa()` or `sem()` (they are currently identical) for confirmatory factor analysis, path analysis and SEM
- `growth()` for (basic) growth curve modeling

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

```
sem(model = NULL, data = NULL, meanstructure = "default",
    fixed.x = "default", orthogonal = FALSE, std.lv = FALSE,
    parameterization = "default", std.ov = FALSE, missing = "default",
    ordered = NULL, sample.cov = NULL, sample.cov.rescale = "default",
    sample.mean = NULL, sample.nobs = NULL, ridge = 1e-05, group = NULL,
    group.label = NULL, group.equal = "", group.partial = "",
    group.w.free = FALSE, cluster = NULL, constraints = "",
    estimator = "default",
    likelihood = "default", link = "default", information = "default",
    se = "default", test = "default", bootstrap = 1000L, mimic = "default",
    representation = "default", do.fit = TRUE, control = list(),
    WLS.V = NULL, NACOV = NULL,
    zero.add = "default", zero.keep.margins = "default",
    start = "default", verbose = FALSE, warn = TRUE, debug = FALSE)
```

power-user fitting functions

- `lavaan()` by default, no model parameters are added automagically to the parameter table
- 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)

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)
```

arguments of the lavaan() fitting function

```
lavaan(model = NULL, data = NULL, model.type = "sem",
    meanstructure = "default",
    int.ov.free = FALSE, int.lv.free = FALSE, fixed.x = "default",
    orthogonal = FALSE, std.lv = FALSE, parameterization = "default",
    auto.fix.first = FALSE, auto.fix.single = FALSE, auto.var = FALSE,
    auto.cov.lv.x = FALSE, auto.cov.y = FALSE, auto.th = FALSE,
    auto.delta = FALSE, std.ov = FALSE, missing = "default",
    ordered = NULL, sample.cov = NULL, sample.cov.rescale = "default",
    sample.mean = NULL, sample.nobs = NULL, ridge = 1e-05, group = NULL,
    group.label = NULL, group.equal = "", group.partial = "",
    group.w.free = FALSE, cluster = NULL, constraints = "",
    estimator = "default",
    likelihood = "default", link = "default", information = "default",
    se = "default", test = "default", bootstrap = 1000L, mimic = "default",
    representation = "default", do.fit = TRUE, control = list(),
    WLS.V = NULL, NACOV = NULL,
    zero.add = "default", zero.keep.margins = "default",
    start = "default", slotOptions = NULL, slotParTable = NULL,
    slotSampleStats = NULL, slotData = NULL, slotModel = NULL,
    verbose = FALSE, warn = TRUE, debug = FALSE)
```

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)

extractor functions: inspecting fitted models

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
`inspect()`	peek into the internal representation of the model; by default, it returns a list of model matrices counting the free parameters in the model; can also be used to extract starting values, gradient values, and much more

other functions

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
`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

extractor examples (1)

```
> 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

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

> predict(fit)
       visual textul  speed
  [1,] -0.818 -0.138  0.062
  [2,]  0.050 -1.013  0.625
  [3,] -0.761 -1.872 -0.841
  [4,]  0.419  0.018 -0.271
  [5,] -0.416 -0.122  0.194
  ...
```

extractor examples (2)

```
> inspect(fit)  # matrix representation
$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

> inspect(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

$mean
   x1    x2    x3    x4    x5    x6    x7    x8    x9
4.936 6.088 2.250 3.061 4.341 2.186 4.186 5.527 5.374
```

fitMeasures

```
> fitMeasures(fit)
                fmin                chisq                   df               pvalue
               0.142               85.306               24.000                0.000
      baseline.chisq          baseline.df      baseline.pvalue                  cfi
             918.852               36.000                0.000                0.931
                 tli                 nnfi                  rfi                  nfi
               0.896                0.896                0.861                0.907
                pnfi                  ifi                  rni                 logl
               0.605                0.931                0.931            -3737.745
  unrestricted.logl                 npar                  aic                  bic
           -3695.092               21.000             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          srmr_bollen
               0.065                0.065                0.065                0.065
srmr_bollen_nomean          srmr_mplus   srmr_mplus_nomean                cn_05
               0.065                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
```

parameterTable

```
> parameterTable(fit)
   id     lhs op     rhs user group free ustart exo label eq.id unco
1   1  visual =~      x1    1     1    0      1   0             0    0
2   2  visual =~      x2    1     1    1     NA   0             0    1
3   3  visual =~      x3    1     1    2     NA   0             0    2
4   4 textual =~      x4    1     1    0      1   0             0    0
5   5 textual =~      x5    1     1    3     NA   0             0    3
6   6 textual =~      x6    1     1    4     NA   0             0    4
7   7   speed =~      x7    1     1    0      1   0             0    0
8   8   speed =~      x8    1     1    5     NA   0             0    5
9   9   speed =~      x9    1     1    6     NA   0             0    6
10 10      x1 ~~      x1    0     1    7     NA   0             0    7
11 11      x2 ~~      x2    0     1    8     NA   0             0    8
12 12      x3 ~~      x3    0     1    9     NA   0             0    9
13 13      x4 ~~      x4    0     1   10     NA   0             0   10
14 14      x5 ~~      x5    0     1   11     NA   0             0   11
15 15      x6 ~~      x6    0     1   12     NA   0             0   12
16 16      x7 ~~      x7    0     1   13     NA   0             0   13
17 17      x8 ~~      x8    0     1   14     NA   0             0   14
18 18      x9 ~~      x9    0     1   15     NA   0             0   15
19 19  visual ~~  visual    0     1   16     NA   0             0   16
20 20 textual ~~ textual    0     1   17     NA   0             0   17
21 21   speed ~~   speed    0     1   18     NA   0             0   18
22 22  visual ~~ textual    0     1   19     NA   0             0   19
23 23  visual ~~   speed    0     1   20     NA   0             0   20
24 24 textual ~~   speed    0     1   21     NA   0             0   21
```

parameterEstimates

```
> parameterEstimates(fit)
      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
22 visual ~~ textual 0.408 0.074  5.552      0    0.264    0.552
23 visual ~~   speed 0.262 0.056  4.660      0    0.152    0.373
24 textual ~~  speed 0.173 0.049  3.518      0    0.077    0.270
```

varTable

```
> varTable(fit)
  name idx nobs    type exo user  mean   var nlev lnam
1   x1   7  301 numeric   0    0 4.936 1.363    0
2   x2   8  301 numeric   0    0 6.088 1.386    0
3   x3   9  301 numeric   0    0 2.250 1.279    0
4   x4  10  301 numeric   0    0 3.061 1.355    0
5   x5  11  301 numeric   0    0 4.341 1.665    0
6   x6  12  301 numeric   0    0 2.186 1.200    0
7   x7  13  301 numeric   0    0 4.186 1.187    0
8   x8  14  301 numeric   0    0 5.527 1.025    0
9   x9  15  301 numeric   0    0 5.374 1.018    0
```

modification indices

```
> subset(modindices(fit), mi > 5)
       lhs op rhs     mi    epc sepc.lv sepc.all sepc.nox
1   visual =~  x5  7.441 -0.210  -0.189   -0.147   -0.147
2   visual =~  x7 18.631 -0.422  -0.380   -0.349   -0.349
3   visual =~  x9 36.411  0.577   0.519    0.515    0.515
4  textual =~  x1  8.903  0.350   0.347    0.297    0.297
5  textual =~  x3  9.151 -0.272  -0.269   -0.238   -0.238
6       x1 ~~  x7  5.420 -0.129  -0.129   -0.102   -0.102
7       x1 ~~  x9  7.335  0.138   0.138    0.117    0.117
8       x2 ~~  x3  8.532  0.218   0.218    0.164    0.164
9       x2 ~~  x7  8.918 -0.183  -0.183   -0.143   -0.143
10      x3 ~~  x5  7.858 -0.130  -0.130   -0.089   -0.089
11      x4 ~~  x6  6.220 -0.235  -0.235   -0.185   -0.185
12      x4 ~~  x7  5.920  0.098   0.098    0.078    0.078
13      x7 ~~  x8 34.145  0.536   0.536    0.488    0.488
14      x7 ~~  x9  5.183 -0.187  -0.187   -0.170   -0.170
15      x8 ~~  x9 14.946 -0.423  -0.423   -0.415   -0.415
```

4.3 Estimator ML, MLM and MLR: robust standard errors and scaled test statistics

estimator ML

- ML is the default estimator in all software packages for SEM
- the likelihood function is derived from the multivariate normal distribution (the 'normal' tradition) or the Wishart distribution (the 'Wishart' tradition)
- standard errors are usually based on the covariance matrix that is obtained by inverting the expected information matrix

$$
\begin{aligned}
n\text{Cov}(\hat{\theta}) &= A^{-1} \\
&= (\Delta' W \Delta)^{-1}
\end{aligned}
$$

  - Δ is a jacobian matrix and W is a function of Σ^{-1}
  - if no meanstructure:

$$
\begin{aligned}
\Delta &= \partial\hat{\Sigma}/\partial\hat{\theta}' \\
W &= 2D'(\hat{\Sigma}^{-1} \otimes \hat{\Sigma}^{-1})D
\end{aligned}
$$

- an alternative is to use the *observed* information matrix

$$
\begin{aligned}
n\text{Cov}(\hat{\theta}) &= A^{-1} \\
&= [-\text{Hessian}]^{-1} \\
&= \left[-\partial F(\hat{\theta})/(\partial\hat{\theta}\partial\hat{\theta}')\right]^{-1}
\end{aligned}
$$

  where $F(\theta)$ is the function that is minimized

- overall model evaluation is based on the likelihood-ratio (LR) statistic (chi-square test): T_{ML}
  - (minus two times the) difference between loglikelihood of user-specified model H_0 and unrestricted model H_1
  - equals (in lavaan) $2 \times n$ times the minimum value of $F(\theta)$
  - T_{ML} follows (under regularity conditions) a chi-square distribution

estimator MLM

- parameter estimates are standard ML estimates
- standard errors are robust to non-normality
  - standard errors are computed using a sandwich-type estimator:

$$
\begin{aligned}
n\text{Cov}(\hat{\theta}) &= A^{-1}BA^{-1} \\
&= (\Delta' W \Delta)^{-1}(\Delta' W \Gamma W \Delta)(\Delta' W \Delta)^{-1}
\end{aligned}
$$

  - A is usually the expected information matrix (but not in Mplus)
  - references: Huber (1967), Browne (1984), Shapiro (1983), Bentler (1983), ...

- chi-square test statistic is robust to non-normality
  - test statistic is 'scaled' by a correction factor

$$T_{SB} = T_{ML}/c$$

  - the scaling factor c is computed by:

$$c = tr\left[U\Gamma\right]/\text{df}$$

    where

$$U = (W^{-1} - W^{-1}\Delta(\Delta' W^{-1}\Delta)^{-1}\Delta' W^{-1})$$

  - correction method described by Satorra & Bentler (1986, 1988, 1994)
- estimator MLM: for complete data only

estimator MLR

- parameter estimates are standard ML estimates
- standard errors are robust to non-normality
  - standard errors are computed using a (different) sandwich approach:

$$n\text{Cov}(\hat{\theta}) = A^{-1}BA^{-1}$$
$$= A_0^{-1}B_0A_0^{-1} = C_0$$

where

$$A_0 = -\sum_{i=1}^{n} \frac{\partial \log L_i}{\partial\hat{\theta}\,\partial\hat{\theta}'} \quad \text{(observed information)}$$

and

$$B_0 = \sum_{i=1}^{n} \left(\frac{\partial \log L_i}{\partial\hat{\theta}}\right) \times \left(\frac{\partial \log L_i}{\partial\hat{\theta}}\right)'$$

  - for both complete and incomplete data

  - Huber (1967), Gourieroux, Monfort & Trognon (1984), Arminger & Schoenberg (1989)

- chi-square test statistic is robust to non-normality
  - test statistic is 'scaled' by a correction factor

$$T_{MLR} = T_{ML}/c$$

  - the scaling factor c is (usually) computed by

$$c = tr\,[M]$$

    where

$$M = C_1(A_1 - A_1\Delta(\Delta' A_1 \Delta)^{-1}\Delta' A_1)$$

  - A_1 and C_1 are computed under the unrestricted (H_1) model
  - correction method described by Yuan & Bentler (2000)

- information matrix (A) can be observed or expected
- for complete data, the MLR and MLM corrections are asymptotically equivalent