Structural Equation Modeling with lavaan

Yves Rosseel
Department of Data Analysis
Ghent University

Gent 9–10 January 2020
# Contents

1 **Introduction to SEM**
   1.1 What is SEM? .......................................................... 5
   1.2 How does SEM work? .................................................. 12
   1.3 A first example: a CFA with three factors ....................... 26
   1.4 The matrix representation of a CFA model ....................... 29
   1.5 A second example: the Political Democracy dataset ............ 36
   1.6 Model estimation ..................................................... 46
   1.7 Model evaluation ..................................................... 47
   1.8 Model respecification ............................................... 50
   1.9 Reporting your results ............................................. 51
   1.10 Further reading .................................................... 52

2 **Introduction to lavaan**
   2.1 Software for SEM ................................................... 54
   2.2 The R package ‘lavaan’ ............................................. 55
   2.3 The lavaan model syntax ........................................... 60
   2.4 lavaan: a brief user’s guide ....................................... 80
3 Multiple groups and measurement invariance 106
  3.1 Mean structures .................................................. 106
  3.2 Multiple groups .................................................. 111
  3.3 Measurement invariance ......................................... 113
  3.4 What if measurement invariance can not be established? (optional) 126
  3.5 Measurement invariance: recent developments and references ... 130

4 Missing data and non-normal (continuous) data 133
  4.1 Missing data ...................................................... 133
  4.2 Nonnormal data and alternative estimators .................... 140

5 Categorical data 151
  5.1 Handling categorical endogenous variables .................... 151
  5.2 Two approaches for handling categorical data in a SEM framework 152
  5.3 A limited information approach: the WLSMV estimator .... 155
  5.4 Using categorical variables in lavaan ........................ 166
  5.5 SEM vs IRT ....................................................... 184

6 Longitudinal Structural Equation Modeling 186
6.1 Repeated measures ANOVA in SEM ................................. 190
6.2 Panel models for longitudinal data ................................. 213
6.3 Growth curve models ............................................... 235
6.4 Alternative models .................................................. 248
1 Introduction to SEM

1.1 What is SEM?

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

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

- strict distinction between ‘dependent’ variables and ‘independent’ variables
path analysis

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

\[ y_5 = \text{reading motivation} \]
\[ y_6 = \text{reading frequency} \]
\[ y_7 = \text{reading ability} \]
confirmatory factor analysis (CFA)

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

\[ y_1, y_2, y_3, y_4, y_5, y_6 \]

\[ \eta_1, \eta_2 \]

\[ \eta_1 = \text{depression} \]

\[ \eta_2 = \text{neuroticism} \]
structural equation modeling (SEM)

- path analysis with latent variables
who is using SEM?

• it is widely used in the social sciences

• it is increasingly ‘discovered’ by other fields:
  – medical sciences
  – neuroimaging
  – biology, ecology (climate change!)
  – ...

• SEM software is also used to perform standard analyses (eg., regression), but where there is need for:
  – dealing with missing data, clustered data, categorical data
  – robust standard errors, goodness-of-fit measures
  – (in)equality constraints
  – …
1.2 How does SEM work?

• we describe here the traditional SEM approach for a set of $P$ observed (continuous) variables

• the starting point of a (traditional) SEM analysis is the observed variance-covariance matrix $S$ of the data
  
  – a $P \times P$ symmetric matrix
  
  – $P$ variances are on the diagonal
  
  – $P(P - 1)/2$ covariances are below the diagonal (and again above the diagonal)
  
  – a covariance represents the strength of a linear relationship between two variables (and can be positive or negative)
  
  – if all variances would be 1, the covariances become correlations

• later, we will also include the means ($\bar{y}$) of the observed variables

• if the data is multivariate normally distributed, then $S$ and $\bar{y}$ contain all information about the data
a dataset: the Holzinger & Swineford dataset

- this is a ‘classic’ dataset, based on data collected by Holzinger & Swineford (1939)

- scores on 26 ‘Mental Ability tests’ of seventh- and eighth-grade children from two different schools (Pasteur and Grant-White)

- the dataset was used in a seminal paper about CFA (Jöreskog, 1969)

- just like Jöreskog (1969), we will use a subset of 9 scores: \( x_1 = \) Visual perception, \( x_2 = \) Cubes, \( x_3 = \) Lozenges, \( x_4 = \) Paragraph comprehension, \( x_5 = \) Sentence completion, \( x_6 = \) Word meaning, \( x_7 = \) Speeded addition, \( x_8 = \) Speeded counting of dots, \( x_9 = \) Speeded discrimination

- these 9 scores are often regarded as indicators of 3 latent variables: ‘visual intelligence’ \((x_1, x_2, x_3)\), ‘textual intelligence’ \((x_4, x_5, x_6)\), en ‘speed’ \((x_7, x_8, x_9)\)

- we will investigate this later using CFA
reading in data + descriptives

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

[1] 301 15

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

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.6667</td>
<td>2.250</td>
<td>0.250</td>
<td>0.000</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>4.1667</td>
<td>5.250</td>
<td>1.375</td>
<td>2.333</td>
</tr>
<tr>
<td>Median</td>
<td>5.000</td>
<td>6.000</td>
<td>2.125</td>
<td>3.000</td>
</tr>
<tr>
<td>Mean</td>
<td>4.9358</td>
<td>6.088</td>
<td>2.250</td>
<td>3.061</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>5.6667</td>
<td>6.750</td>
<td>3.125</td>
<td>3.667</td>
</tr>
<tr>
<td>Max</td>
<td>8.5000</td>
<td>9.250</td>
<td>4.500</td>
<td>6.333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1.000</td>
<td>0.1429</td>
<td>1.304</td>
<td>3.050</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>3.500</td>
<td>1.4286</td>
<td>3.478</td>
<td>4.850</td>
</tr>
<tr>
<td>Median</td>
<td>4.500</td>
<td>2.0000</td>
<td>4.087</td>
<td>5.500</td>
</tr>
<tr>
<td>Mean</td>
<td>4.341</td>
<td>2.1856</td>
<td>4.186</td>
<td>5.527</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>5.250</td>
<td>2.7143</td>
<td>4.913</td>
<td>6.100</td>
</tr>
<tr>
<td>Max</td>
<td>7.000</td>
<td>6.1429</td>
<td>7.435</td>
<td>10.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>2.778</td>
</tr>
</tbody>
</table>
```
computing the variance-covariance matrix for $P = 9$ variables

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

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.358</td>
<td>0.407</td>
<td>0.580</td>
<td>0.505</td>
<td>0.441</td>
<td>0.455</td>
<td>0.085</td>
<td>0.264</td>
<td>0.458</td>
</tr>
<tr>
<td>x2</td>
<td>0.407</td>
<td>1.382</td>
<td>0.451</td>
<td>0.209</td>
<td>0.211</td>
<td>0.248</td>
<td>−0.097</td>
<td>0.110</td>
<td>0.244</td>
</tr>
<tr>
<td>x3</td>
<td>0.580</td>
<td>0.451</td>
<td>1.275</td>
<td>0.208</td>
<td>0.112</td>
<td>0.244</td>
<td>0.088</td>
<td>0.212</td>
<td>0.374</td>
</tr>
<tr>
<td>x4</td>
<td>0.505</td>
<td>0.209</td>
<td>0.208</td>
<td>1.351</td>
<td>1.098</td>
<td>0.896</td>
<td>0.220</td>
<td>0.126</td>
<td>0.243</td>
</tr>
<tr>
<td>x5</td>
<td>0.441</td>
<td>0.211</td>
<td>0.112</td>
<td>1.098</td>
<td>1.660</td>
<td>1.015</td>
<td>0.143</td>
<td>0.181</td>
<td>0.295</td>
</tr>
<tr>
<td>x6</td>
<td>0.455</td>
<td>0.248</td>
<td>0.244</td>
<td>0.896</td>
<td>1.015</td>
<td>1.196</td>
<td>0.144</td>
<td>0.165</td>
<td>0.236</td>
</tr>
<tr>
<td>x7</td>
<td>0.085</td>
<td>−0.097</td>
<td>0.088</td>
<td>0.220</td>
<td>0.143</td>
<td>0.144</td>
<td>1.183</td>
<td>0.535</td>
<td>0.373</td>
</tr>
<tr>
<td>x8</td>
<td>0.264</td>
<td>0.110</td>
<td>0.212</td>
<td>0.126</td>
<td>0.181</td>
<td>0.165</td>
<td>0.535</td>
<td>1.022</td>
<td>0.457</td>
</tr>
<tr>
<td>x9</td>
<td>0.458</td>
<td>0.244</td>
<td>0.374</td>
<td>0.243</td>
<td>0.295</td>
<td>0.236</td>
<td>0.373</td>
<td>0.457</td>
<td>1.015</td>
</tr>
</tbody>
</table>
interludium: regression based on the variance-covariance matrix

• suppose we would like to fit a multiple regression model:

\[ x_1 = \beta_0 + \beta_1 x_2 + \beta_2 x_3 + \beta_3 x_4 + \epsilon \]

• the classical formula to estimate the regression coefficients is based on all data:

\[ \hat{\beta} = (X'X)^{-1}X'y \]

where \( X \) is a matrix with 4 columns: \((1, x_2, x_3, x_4)\), en \( y = x_1 \)

• we can obtain the same solution using the variance-covariance matrix only:

\[ \hat{\beta}_p = (S_{234,234})^{-1}S_{234,1} \]

and if we need the intercept: \( \hat{\beta}_0 = \bar{x}_1 - \hat{\beta}' p \bar{x}_{234} \)
interludium: regression based on the variance-covariance matrix (2)

```r
> fit <- lm(x1 ~ 1 + x2 + x3 + x4, data = HolzingerSwineford1939)
> coef(fit)

(Intercept)   x2   x3   x4
 2.4087782 0.1324665 0.3593608 0.2978916

> SX <- cov( HolzingerSwineford1939[, c("x1", "x2", "x3", "x4")])
> MX <- colMeans( HolzingerSwineford1939[, c("x1", "x2", "x3", "x4")])
> beta.p <- solve(SX[2:4, 2:4]) %*% SX[2:4, 1]
> beta.p

[,1]
x2  0.1324665
x3  0.3593608
x4  0.2978916

> beta.0

[,1]
[1,] 2.408778
```
the model-implied variance-covariance matrix

- the goal of SEM is to test an a priori specified theory/model, based on empirical data; we would like to know if our model ‘fits’ the data (or not)

- each model can be depicted by a path diagram (we may have several alternative models, each one with its own path diagram)

- each path diagram can be converted to a SEM

- SEM will tell us what the implications are for the data if (assumption!) our model is correct: how ‘should’ the data look like, which patterns should we observe?

- in practice, SEM will tell us how the variance-covariance matrix of the data should look like; we call this the ‘model-implied’ variance-covariance matrix ($\hat{\Sigma}$)

- different models $\rightarrow$ different path diagrams $\rightarrow$ different $\hat{\Sigma}$ matrices

- if $\hat{\Sigma}$ is close to $S$, the model fits well
example model-implied covariance matrix (1)

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

\[
\begin{align*}
y_2 &= ay_1 + \epsilon_2 \\
y_3 &= by_1 + \epsilon_3
\end{align*}
\]

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

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

• how can we obtain this matrix $\hat{\Sigma}$?
rules about variances and covariances

• we will apply the following rules about variances:

\[
\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2 \text{Cov}(X, Y)
\]

\[
\text{Var}(aX) = a^2 \text{Var}(X)
\]

• rules about covariances:

\[
\text{Cov}(A + B, C + D) = \text{Cov}(A, C) + \text{Cov}(A, D) + \text{Cov}(B, C) + \text{Cov}(B, D)
\]

\[
\text{Cov}(aX, bY) = ab \text{Cov}(X, Y)
\]

• in a regression setting, we usually assume that the predictors and the error terms are uncorrelated: \(
\text{Cov}(y_1, \epsilon_2) = 0 \text{ and } \text{Cov}(y_1, \epsilon_3) = 0
\)

• for ease of computation, we will also assume that \(\text{Cov}(\epsilon_2, \epsilon_3) = 0\), although we could easily relax this assumption
computing the model-implied variances

- it is given that $\text{Var}(y_1) = 10$

- the ‘model-implied’ variance of $y_2$ is obtained as

\[
\text{Var}(y_2) = \text{Var}(ay_1 + \epsilon_2)
\]
\[
= \text{Var}(ay_1) + \text{Var}(\epsilon_2) + 2\text{Cov}(ay_1, \epsilon_2)
\]
\[
= \text{Var}(ay_1) + \text{Var}(\epsilon_2) + 2a\text{Cov}(y_1, \epsilon_2)
\]
\[
= \text{Var}(ay_1) + \text{Var}(\epsilon_2) + 0
\]
\[
= a^2\text{Var}(y_1) + \text{Var}(\epsilon_2)
\]
\[
= (3)^2 \cdot 10 + 20 = 110
\]

- similarly, the ‘model-implied’ variance of $y_3$ is given by

\[
\text{Var}(y_3) = (5)^2 \cdot 10 + 30 = 280
\]
computing the model-implied covariances

• the covariance between $y_1$ and $y_2$:

$$\text{Cov}(y_1, y_2) = \text{Cov}(y_1, ay_1 + \epsilon_2)$$
$$= \text{Cov}(y_1, ay_1) + \text{Cov}(y_1, \epsilon_2)$$
$$= a \text{Cov}(y_1, y_1) + 0$$
$$= a \text{Var}(y_1)$$
$$= 3 \cdot 10 = 30$$

• similarly, $\text{Cov}(y_1, y_3) = 5 \cdot 10 = 50$

• the covariance between $y_2$ and $y_3$:

$$\text{Cov}(y_2, y_3) = \text{Cov}(ay_1 + \epsilon_2, by_1 + \epsilon_3)$$
$$= \text{Cov}(ay_1, by_1) + \text{Cov}(ay_1, \epsilon_3) + \text{Cov}(\epsilon_2, by_1) + \text{Cov}(\epsilon_2, \epsilon_3)$$
$$= ab \text{Var}(y_1) + a \text{Cov}(y_1, \epsilon_3) + b \text{Cov}(\epsilon_2, y_1) + \text{Cov}(\epsilon_2, \epsilon_3)$$
$$= 3 \cdot 5 \cdot 10 + 3 \cdot 0 + 5 \cdot 0 + 0 = 150$$
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:

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

we find

- 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’:

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

• we can compare all three \(\hat{\Sigma}\) matrices to \(S\) to find out which model fits best
the essence of SEM

- we start with a theory/model/hypothesis that we can represent as a diagram

- we collect data for the $P$ observed variables in the diagram; we summarize the data by computing the sample variance-covariance matrix $S$ (and maybe the means $\overline{y}$)

- a SEM analysis:
  - the user communicates the model to the SEM software
  - using the data ($S$), we ‘fit’ the model, and obtain the model-implied variance-covariance matrix $\hat{\Sigma}$
  - we ‘evaluate’ the model by comparing $\hat{\Sigma}$ to $S$
  - if the fit is good, we can interpret the parameters; if the fit is not good, we may need to respecify our model

- we need a more convenient way to compute $\hat{\Sigma}$
1.3 A first example: a CFA with three factors

• for this example, we use the Holzinger & Swineford (1939) data

• we postulate a CFA with three latent variables (‘factors’):
  – a ‘visual’ factor measured by x1, x2 and x3
  – a ‘textual’ factor measured by x4, x5 and x6
  – a ‘speed’ factor measured by x7, x8 and x9

• we assume the three factors are correlated

• the next slide shows a path diagram of this model

• we will discuss later how we can ‘fit’ this model using SEM software

• in the next subsection, we introduce the matrix representation of a CFA model, in order to have a convenient way to compute the model-implied variance-covariance matrix
• ‘free’ parameters: factor loadings, variances for the factors, covariances between the factors, and residual variances for the indicators
observed (sample-based) variance-covariance matrix $S$

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

model-implied variance-covariance matrix $\hat{\Sigma}$

$$
\begin{array}{cccccccccc}
  & x1 & x2 & x3 & x4 & x5 & x6 & x7 & x8 & x9 \\
 x1 & 1.358 &   &   &   &   &   &   &   &   \\
x2 & 0.448 & 1.382 &   &   &   &   &   &   &   \\
x3 & 0.590 & 0.327 & 1.275 &   &   &   &   &   &   \\
x4 & 0.408 & 0.226 & 0.298 & 1.351 &   &   &   &   &   \\
x5 & 0.454 & 0.252 & 0.331 & 1.090 & 1.660 &   &   &   &   \\
x6 & 0.378 & 0.209 & 0.276 & 0.907 & 1.010 & 1.196 &   &   &   \\
x7 & 0.262 & 0.145 & 0.191 & 0.173 & 0.193 & 0.161 & 1.183 &   &   \\
x8 & 0.309 & 0.171 & 0.226 & 0.205 & 0.228 & 0.190 & 0.453 & 1.022 &   \\
x9 & 0.284 & 0.157 & 0.207 & 0.188 & 0.209 & 0.174 & 0.415 & 0.490 & 1.015 \\
\end{array}
$$
1.4 The matrix representation of a CFA model

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

\[ \Lambda = \begin{bmatrix} x & 0 & 0 \\ x & 0 & 0 \\ x & 0 & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & x & 0 \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} \]

- the variances/covariances of the latent variables are summarized in the PSI matrix:
\[ \Psi = \begin{bmatrix} x \\ x \\ x \\ x \end{bmatrix} \]

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

\[ \Theta = \begin{bmatrix} x & & & \\ & x & & \\ & & x & \\ & & & x \end{bmatrix} \]

- 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:

\[ \Sigma = \Lambda \Psi \Lambda' + \Theta \]

• all parameters are included in three model matrices

• simple matrix multiplication (and addition) gives us the model implied covariance matrix

• for identification purposes, some parameters need to be fixed to a constant (see next slide)

• estimation problem: choose the ‘free’ parameters, so that the estimated implied covariance matrix (\( \hat{\Sigma} \)) is ‘as close as possible’ to the observed covariance matrix \( S \)
  
  – 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
the three ‘estimated’ model matrices

$\lambda$

<table>
<thead>
<tr>
<th></th>
<th>visual</th>
<th>textul</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>x2</td>
<td>0.554</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>x3</td>
<td>0.729</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>x4</td>
<td>0.000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>x5</td>
<td>0.000</td>
<td>1.113</td>
<td>0.000</td>
</tr>
<tr>
<td>x6</td>
<td>0.000</td>
<td>0.926</td>
<td>0.000</td>
</tr>
<tr>
<td>x7</td>
<td>0.000</td>
<td>0.000</td>
<td>1.000</td>
</tr>
<tr>
<td>x8</td>
<td>0.000</td>
<td>0.000</td>
<td>1.180</td>
</tr>
<tr>
<td>x9</td>
<td>0.000</td>
<td>0.000</td>
<td>1.082</td>
</tr>
</tbody>
</table>

$\theta$

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0.549</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>0.000</td>
<td>1.134</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>0.000</td>
<td>0.000</td>
<td>0.844</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.371</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.446</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.356</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.799</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.488</td>
<td></td>
</tr>
<tr>
<td>x9</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.566</td>
</tr>
</tbody>
</table>

$\psi$

<table>
<thead>
<tr>
<th></th>
<th>visual</th>
<th>textul</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>
visual  0.809  
textual  0.408  0.979  
speed    0.262  0.173  0.384  

manual computation of $\hat{\Sigma}$ (optional)

```r
> attach(lavInspect(fit, "est"))
> Sigma <- lambda %*% psi %*% t(lambda) + theta
> Sigma
```

```
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
```
number of free parameters and degrees of freedom

• in our example, we have used ULI: the first factor loading (of each latent variable) was fixed to 1.0

• therefore, we only have 21 free parameters in our model:
  – 6 factor loadings
  – 3 variances for the factors
  – 3 covariances between the factors
  – 9 residual variances for the indicators

• our sample variance-covariance matrix ($S$) contains $P(P + 1)/2 = 45$ (non-redundant) elements (‘sample statistics’)

• the difference between the number of sample statistics and the number of free parameters is called the ‘degrees of freedom’ of the model; for this model, we have $45 - 21 = 24$ degrees of freedom (df = 24)

• the number of free parameters cannot exceed the number of sample statistics; if df = 0, we say the model is ‘saturated’ because in this case $\hat{\Sigma} = S$
1.5 A second example: the Political Democracy dataset

- data from $N = 75$ developing countries regarding the amount of ‘industrialization’ (in 1960) and the level of ‘political democracy’ (in 1960, and again in 1965)

- this dataset is used throughout Bollen’s 1989 book

- overview of the observed variables (indicators):

  \[
  \begin{align*}
  y_1 &: \text{Expert ratings of the freedom of the press in 1960} \\
  y_2 &: \text{The freedom of political opposition in 1960} \\
  y_3 &: \text{The fairness of elections in 1960} \\
  y_4 &: \text{The effectiveness of the elected legislature in 1960} \\
  y_5 &: \text{Expert ratings of the freedom of the press in 1965} \\
  y_6 &: \text{The freedom of political opposition in 1965} \\
  y_7 &: \text{The fairness of elections in 1965} \\
  y_8 &: \text{The effectiveness of the elected legislature in 1965} \\
  x_1 &: \text{The gross national product (GNP) per capita in 1960} \\
  x_2 &: \text{The inanimate energy consumption per capita in 1960} \\
  x_3 &: \text{The percentage of the labor force in industry in 1960}
  \end{align*}
  \]

- three latent variables: $\text{ind60}$, measured by $x_1$, $x_2$ and $x_3$; $\text{dem60}$, measured by $y_1$, $y_2$, $y_3$ and $y_4$; $\text{dem65}$ measured by $y_5$, $y_6$, $y_7$ en $y_8$
model diagram
### preview of (a selection of) the lavaan output

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

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

• this is an example of a ‘full SEM’: the model contains both a measurement part, and a structural part

• we now need 4 model matrices:

  – LAMBDA: the factor loadings
  – THETA: the residual variances (and covariances) of the observed indicators
  – PSI: the (residual) variances and covariances of the latent variables
  – BETA: the regression coefficients of the structural part

• the formula to obtain the model-implied variance-covariance matrix is now slightly more complex:

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

where \( I \) is the identity matrix
the four (already estimated) model matrices

\[
> \text{lavInspect(fit, "est")}
\]

\[
\begin{align*}
\text{\$lambda} & \\
\text{ind60 dem60 dem65} & \\
x1 & 1.000 0.000 0.000 \\
x2 & 2.180 0.000 0.000 \\
x3 & 1.819 0.000 0.000 \\
y1 & 0.000 1.000 0.000 \\
y2 & 0.000 1.257 0.000 \\
y3 & 0.000 1.058 0.000 \\
y4 & 0.000 1.265 0.000 \\
y5 & 0.000 0.000 1.000 \\
y6 & 0.000 0.000 1.186 \\
y7 & 0.000 0.000 1.280 \\
y8 & 0.000 0.000 1.266
\end{align*}
\]

\[
\begin{align*}
\text{\$theta} & \\
x1 & 0.082 \\
x2 & 0.000 0.120 \\
x3 & 0.000 0.000 0.467 \\
y1 & 0.000 0.000 0.000 1.891 \\
y2 & 0.000 0.000 0.000 0.000 7.373 \\
y3 & 0.000 0.000 0.000 0.000 0.000 5.067 \\
y4 & 0.000 0.000 0.000 0.000 1.313 0.000 3.148 \\
y5 & 0.000 0.000 0.000 0.000 0.624 0.000 0.000 2.351
\end{align*}
\]
\[ \begin{align*}
y6 & 0.000 0.000 0.000 0.000 2.153 0.000 0.000 0.000 4.954 
\end{align*} \]

\[ \begin{align*}
y7 & 0.000 0.000 0.000 0.000 0.000 0.795 0.000 0.000 0.000 3.431 
\end{align*} \]

\[ \begin{align*}
y8 & 0.000 0.000 0.000 0.000 0.000 0.000 0.348 0.000 1.356 0.000 3.254 
\end{align*} \]

$\psi$

\[
\begin{array}{c}
\text{ind60} \quad \text{dem60} \quad \text{dem65} \\
\text{ind60} & 0.448 \\
\text{dem60} & 0.000 3.956 \\
\text{dem65} & 0.000 0.000 0.172 \\
\end{array}
\]

$\beta$

\[
\begin{array}{c}
\text{ind60} \quad \text{dem60} \quad \text{dem65} \\
\text{ind60} & 0.000 0.000 0 \\
\text{dem60} & 1.483 0.000 0 \\
\text{dem65} & 0.572 0.837 0 \\
\end{array}
\]
counting the 31 ‘free’ parameters in the model matrices

> `lavInspect(fit)`

```r
$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  0  0  24
y5  0  0  0  0  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
the 31 ‘free’ parameters as a single vector

\[
\begin{align*}
\text{ind60} &= \sim x_2 & \text{ind60} &= \sim x_3 & \text{dem60} &= \sim y_2 & \text{dem60} &= \sim y_3 & \text{dem60} &= \sim y_4 & \text{dem60} &= \sim y_6 \\
\text{dem65} &= \sim y_7 & \text{dem65} &= \sim y_8 & \text{dem60} &\sim \text{ind60} & \text{dem65} &\sim \text{ind60} & \text{dem65} &\sim \text{dem60} & y_1 &\sim y_5 \\
1.280 & & 1.266 & & 1.483 & & 0.572 & & 0.837 & & 0.624 \\
y_2 &\sim y_4 & y_2 &\sim y_6 & y_3 &\sim y_7 & y_4 &\sim y_8 & y_6 &\sim y_8 & x_1 &\sim x_1 \\
1.313 & & 2.153 & & 0.795 & & 0.348 & & 1.356 & & 0.082 \\
x_2 &\sim x_2 & x_3 &\sim x_3 & y_1 &\sim y_1 & y_2 &\sim y_2 & y_3 &\sim y_3 & y_4 &\sim y_4 \\
0.120 & & 0.467 & & 1.891 & & 7.373 & & 5.067 & & 3.148 \\
y_5 &\sim y_5 & y_6 &\sim y_6 & y_7 &\sim y_7 & y_8 &\sim y_8 & \text{ind60} &\sim \text{ind60} & \text{dem60} &\sim \text{dem60} \\
2.351 & & 4.954 & & 3.431 & & 3.254 & & 0.448 & & 3.956 \\
dem65 &\sim dem65 & & & & & & & & & 0.172 \\
\end{align*}
\]

• we refer to this vector of free parameters as the ‘parameter vector’, denoted by \( \hat{\theta} \)

• if we change some of these values, we get a different model-implied variance-covariance matrix \( \hat{\Sigma} \)

• therefore, we often write that \( \Sigma \) is a function of \( \theta \): \( \Sigma(\theta) \)
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

> round(Sigma.hat, 3)
1.6 Model estimation

- we seek those values for $\theta$ that minimize the difference between what we observe in the data, $S$, and what the model implies, $\Sigma(\theta)$

- the final estimated values are denoted by $\hat{\theta}$, and the estimated model-implied covariance matrix can be written as $\hat{\Sigma} = \Sigma(\hat{\theta})$

- there are many ways to quantify this ‘difference’, leading to different discrepancy measures

- the most used discrepancy measure is based on maximum likelihood:

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

- in practice, we replace $\Sigma$ by $\hat{\Sigma} = \Sigma(\hat{\theta})$

- an alternative is (weighted) least squares, for some weight matrix $W$:

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

where $s$ and $\sigma$ are the unique elements of $S$ and $\Sigma$ 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


• the SRMR is (more or less) the ‘average’ of the (standardized) squared residuals (e.g., between the elements of $S$ and $\Sigma$); 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

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


...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).


SEM in R, using lavaan


2 Introduction to lavaan

2.1 Software for SEM

software for SEM: commercial – closed-source

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

software for SEM: non-commercial – open-source

- outside the R ecosystem: gllamm (Stata), Onyx, …
- R packages: sem, OpenMx, lavaan, lava
2.2 The R package ‘lavaan’

what is lavaan?

• **lavaan** is an R package for latent variable analysis:
  
  – general mean/covariance structure modeling: function `lavaan()`
  – user-friendly interface: function `sem()` or `cfa()`
  – support for continuous, binary and ordinal data
  – support for missing data, multiple groups, clustered data, . . .

• under development, future plans:
  
  – EFA, ESEM, mixture/latent-class SEM, IRT, new engine, . . .

• the long-term goal of **lavaan** is
  
  1. to implement all the state-of-the-art capabilities that are currently available in commercial packages
  2. to provide a modular and extensible platform that allows for easy implementation and testing of new statistical and modeling ideas
installing lavaan, finding documentation

- **lavaan** depends on the R project for statistical computing:
  
  http://www.r-project.org

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

  > install.packages("lavaan")

- more information about **lavaan**:

  http://lavaan.org

- the lavaan paper:


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

  https://groups.google.com/d/forum/lavaan
installing a development version of lavaan

• first install the CRAN version, to make sure you have all the dependencies installed

• 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 (using jags or stan) with a lavaan interface

- **lavaan.survey** (Daniel Oberski)
  survey weights, clustering, strata, and finite sampling corrections in SEM

- **Onyx** (Timo von Oertzen, Andreas M. Brandmaier, Siny Tsang)
  interactive graphical interface for SEM (written in Java)

- **semTools** (Terrence Jorgensen and many others)
  collection of useful functions for SEM

- **simsem** (Terrence Jorgensen and many others)
  simulation of SEM models
the lavaan ecosystem (2)

- **semPlot** (Sacha Epskamp)
  visualizations of SEM models

- **EffectLiteR** (Axel Mayer, Lisa Dietzfelbinger)
  using SEM to estimate average and conditional effects

- **MIIVsem** (Zachary Fisher, Kenneth Bollen, and others)
  Functions for estimating structural equation models using instrumental variables.

- many others

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

using standard R – a simple regression

• using the `lm` function in R:

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

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

# show results
summary(fit)
```

• the standard linear model:

\[
y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + \epsilon_i \quad (i = 1, 2, \ldots, n)
\]
lm() output artificial data (N=100)

> summary(fit)

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

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

Coefficients:
                     Estimate Std. Error  t value  Pr(>|t|)
(Intercept)       97.7210    4.7200    20.704  <2e-16 ***
x1                 5.7733    0.5238     11.022  <2e-16 ***
x2                -1.3214    0.4917     -2.688  0.0085 **
x3                 1.1350    0.4575      2.481  0.0149 *
x4                 0.2707    0.4779      0.566  0.5724

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 46.74 on 95 degrees of freedom
Multiple R-squared:  0.5911,  Adjusted R-squared:  0.5738
F-statistic: 34.33 on 4 and 95 DF,  p-value: < 2.2e-16
the lavaan model syntax – a simple regression

• using lavaan’s `sem` function:

```r
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

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

or include it explicitly in the syntax:

```r
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

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

- 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:

```r
myModel <- ' y1 \sim x1 + x2 + x3 + x4 
               y2 \sim x1 + x2 + x3 + x4 ' 
```

![Diagram showing the model with variables x1, x2, x3, x4 and y1, y2, with arrows indicating the relationships between the variables.](image-url)
the lavaan model syntax – path analysis

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

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

![Diagram showing the relationships between variables x1, x2, x3, x4, x5, x6, and x7 in a path analysis model.](image)
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

```r
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:

<table>
<thead>
<tr>
<th>Information</th>
<th>Observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Errors</td>
<td>Bootstrap</td>
</tr>
</tbody>
</table>

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()

\[
\text{HS.model} \leftarrow \\text{visual} = \sim x_1 + x_2 + x_3 \\
\quad \text{textual} = \sim x_4 + x_5 + x_6 \\
\quad \text{speed} = \sim x_7 + x_8 + x_9 \\
\]

\[
\text{fit} \leftarrow \text{cfa(model = HS.model,} \\
\quad \text{data = HolzingerSwineford1939)} \\
\]

\[
\text{summary(fit, fit.measures = TRUE,} \\
\quad \text{standardized = TRUE)} \\
\]
the lavaan model syntax – using lavaan()

\[
\begin{align*}
\text{HS.model} & \leftarrow '\ \\
& \quad \text{'# latent variables'} \\
& \quad \quad \text{visual} \leftarrow 1 \times x1 + x2 + x3 \\
& \quad \quad \text{textual} \leftarrow 1 \times x4 + x5 + x6 \\
& \quad \quad \text{speed} \leftarrow 1 \times x7 + x8 + x9 \\
& \quad \text{'# factor (co)variances'} \\
& \quad \quad \text{visual} \sim\!\!\sim \text{visual}; \text{visual} \sim\!\!\sim \text{textual} \\
& \quad \quad \text{visual} \sim\!\!\sim \text{speed}; \text{textual} \sim\!\!\sim \text{textual} \\
& \quad \quad \text{textual} \sim\!\!\sim \text{speed}; \text{speed} \sim\!\!\sim \text{speed} \\
& \quad \text{'# residual variances'} \\
& \quad \quad x1 \sim \!\! \times x1; x2 \sim \!\! \times x2; x3 \sim \!\! \times x3 \\
& \quad \quad x4 \sim \!\! \times x4; x5 \sim \!\! \times x5; x6 \sim \!\! \times x6 \\
& \quad \quad x7 \sim \!\! \times x7; x8 \sim \!\! \times x8; x9 \sim \!\! \times x9 \\
\end{align*}
\]

fit <- lavaan(model = HS.model,
              data = HolzingerSwinford1939)

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 Information saturated (h1) model
Expected Structured
Standard Errors Standard

Latent Variables:

visual =~

<p>|     | Estimate | Std.Err | z-value | P(&gt;|z|) | Std.lv | Std.all |
|-----|----------|---------|---------|---------|--------|---------|
| x1  | 1.000    |         |         |         | 0.900  | 0.772   |
| x2  | 0.554    | 0.100   | 5.554   | 0.000   | 0.498  | 0.424   |</p>
<table>
<thead>
<tr>
<th>x3</th>
<th>0.729</th>
<th>0.109</th>
<th>6.685</th>
<th>0.000</th>
<th>0.656</th>
<th>0.581</th>
</tr>
</thead>
<tbody>
<tr>
<td>textual</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x4</th>
<th>1.000</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x5</td>
<td>1.113</td>
<td>0.065</td>
<td>17.014</td>
<td>0.000</td>
<td>1.102</td>
<td>0.855</td>
</tr>
<tr>
<td>x6</td>
<td>0.926</td>
<td>0.055</td>
<td>16.703</td>
<td>0.000</td>
<td>0.917</td>
<td>0.838</td>
</tr>
<tr>
<td>speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x7</th>
<th>1.000</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x8</td>
<td>1.180</td>
<td>0.165</td>
<td>7.152</td>
<td>0.000</td>
<td>0.731</td>
<td>0.723</td>
</tr>
<tr>
<td>x9</td>
<td>1.082</td>
<td>0.151</td>
<td>7.155</td>
<td>0.000</td>
<td>0.670</td>
<td>0.665</td>
</tr>
</tbody>
</table>

Covariances:

<table>
<thead>
<tr>
<th>visual</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>textual</td>
<td>0.408</td>
<td>0.074</td>
<td>5.552</td>
<td>0.000</td>
<td>0.459</td>
<td>0.459</td>
</tr>
<tr>
<td>speed</td>
<td>0.262</td>
<td>0.056</td>
<td>4.660</td>
<td>0.000</td>
<td>0.471</td>
<td>0.471</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>visual</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>textual</td>
<td>0.173</td>
<td>0.049</td>
<td>3.518</td>
<td>0.000</td>
<td>0.283</td>
<td>0.283</td>
</tr>
<tr>
<td>speed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variances:

<table>
<thead>
<tr>
<th>x1</th>
<th>0.549</th>
<th>0.114</th>
<th>4.833</th>
<th>0.000</th>
<th>0.549</th>
<th>0.404</th>
</tr>
</thead>
<tbody>
<tr>
<td>x2</td>
<td>1.134</td>
<td>0.102</td>
<td>11.146</td>
<td>0.000</td>
<td>1.134</td>
<td>0.821</td>
</tr>
<tr>
<td>x3</td>
<td>0.844</td>
<td>0.091</td>
<td>9.317</td>
<td>0.000</td>
<td>0.844</td>
<td>0.662</td>
</tr>
<tr>
<td>x4</td>
<td>0.371</td>
<td>0.048</td>
<td>7.779</td>
<td>0.000</td>
<td>0.371</td>
<td>0.275</td>
</tr>
<tr>
<td>x5</td>
<td>0.446</td>
<td>0.058</td>
<td>7.642</td>
<td>0.000</td>
<td>0.446</td>
<td>0.269</td>
</tr>
<tr>
<td>x6</td>
<td>0.356</td>
<td>0.043</td>
<td>8.277</td>
<td>0.000</td>
<td>0.356</td>
<td>0.298</td>
</tr>
<tr>
<td>x7</td>
<td>0.799</td>
<td>0.081</td>
<td>9.823</td>
<td>0.000</td>
<td>0.799</td>
<td>0.676</td>
</tr>
<tr>
<td>x8</td>
<td>0.488</td>
<td>0.074</td>
<td>6.573</td>
<td>0.000</td>
<td>0.488</td>
<td>0.477</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>x9</td>
<td>0.566</td>
<td>0.071</td>
<td>8.003</td>
<td>0.000</td>
<td>0.566</td>
<td>0.558</td>
</tr>
<tr>
<td>visual</td>
<td>0.809</td>
<td>0.145</td>
<td>5.564</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>textual</td>
<td>0.979</td>
<td>0.112</td>
<td>8.737</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>speed</td>
<td>0.384</td>
<td>0.086</td>
<td>4.451</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>
the lavaan model syntax – equality constraints
fitting the model with lavaan

# 1. specifying the model

```r
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

```r
fit <- sem(model, data = PoliticalDemocracy)
```

# 3. display the results

```r
summary(fit, standardized = TRUE)
```
**output**

lavaan 0.6-3 ended normally after 66 iterations

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>NLMINB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of free parameters</td>
<td>31</td>
</tr>
<tr>
<td>Number of equality constraints</td>
<td>3</td>
</tr>
<tr>
<td>Number of observations</td>
<td>75</td>
</tr>
<tr>
<td>Estimator</td>
<td>ML</td>
</tr>
<tr>
<td>Model Fit Test Statistic</td>
<td>40.179</td>
</tr>
<tr>
<td>Degrees of freedom</td>
<td>38</td>
</tr>
<tr>
<td>P-value (Chi-square)</td>
<td>0.374</td>
</tr>
</tbody>
</table>

**Parameter Estimates:**

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Structured</td>
</tr>
<tr>
<td>Standard Errors</td>
<td>Standard</td>
</tr>
</tbody>
</table>

**Latent Variables:**

|           | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|-----------|----------|---------|---------|--------|--------|---------|
| **ind60 =~** |          |         |         |        |        |         |
| x1        | 1.000    | 0.670   | 0.920   |        | 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

given

table

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| y1       | 0.850   | 0.201   | 2.201   | 1.000  | 0.850   |
| y2       | 0.690   | 0.621   | 2.621   | 1.191  | 0.690   |
| y3       | 0.758   | 2.586   | 0.000   | 1.175  | 0.758   |
| y4       | 0.838   | 2.754   | 0.000   | 1.251  | 0.838   |

### dem65 =~

given

table

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| y5       | 0.817   | 2.154   | 0.000   | 1.000  | 0.817   |
| y6       | 0.755   | 2.565   | 0.000   | 1.191  | 0.755   |
| y7       | 0.802   | 2.530   | 0.000   | 1.175  | 0.802   |
| y8       | 0.829   | 2.694   | 0.000   | 1.251  | 0.829   |

### Regressions:

given

table

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| dem60 ~  | ind60   | 0.448   | 1.471   | 0.392  | 0.448   |
| dem65 ~  | ind60   | 0.187   | 0.600   | 0.226  | 0.187   |
|          | dem60   | 0.884   | 0.865   | 0.075  | 0.884   |

### Covariances:

given

table

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| y1 ~ y5  | 0.281   | 0.583   | 0.356   | 1.000  | 0.281   |
| y2 ~ y4  | 0.291   | 1.440   | 0.689   | 2.092  | 0.291   |
| y3 ~ y7  | 0.169   | 0.712   | 0.611   | 1.165  | 0.169   |
| y4 ~ y8  | 0.356   | 2.183   | 0.737   | 2.960  | 0.356   |
## Variances:

|   | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|---|----------|---------|---------|--------|--------|---------|
| .y8 | 0.363    | 0.444   | 0.817   | 0.414  | 0.363  | 0.111   |
| .y6 | 1.372    | 0.577   | 2.378   | 0.017  | 1.372  | 0.338   |
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:

  ```r
  someVar ~ otherVar
  ```

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

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

  is identical to

  ```r
  Y ~ A + B + C
  ```
### overview operators in the lavaan model syntax

<table>
<thead>
<tr>
<th>formula type</th>
<th>operator</th>
<th>mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>latent variable</td>
<td>=~</td>
<td>is manifested by</td>
</tr>
<tr>
<td>regression</td>
<td>~</td>
<td>is regressed on</td>
</tr>
<tr>
<td>(residual) (co)variance</td>
<td>~~</td>
<td>is correlated with</td>
</tr>
<tr>
<td>intercept</td>
<td>~ 1</td>
<td>intercept</td>
</tr>
<tr>
<td>threshold</td>
<td></td>
<td>t</td>
</tr>
<tr>
<td>scaling factor</td>
<td></td>
<td>*~</td>
</tr>
<tr>
<td>formative latent variable</td>
<td><code>&lt;~</code></td>
<td>is a result of</td>
</tr>
<tr>
<td>defined parameter</td>
<td>:=</td>
<td>is defined as</td>
</tr>
<tr>
<td>equality constraint</td>
<td>==</td>
<td>is equal to</td>
</tr>
<tr>
<td>inequality constraint</td>
<td><code>&lt;</code></td>
<td>is smaller than</td>
</tr>
<tr>
<td>inequality constraint</td>
<td><code>&gt;</code></td>
<td>is larger than</td>
</tr>
</tbody>
</table>
more syntax: modifiers

- each rhs term can be preceded by a ‘modifier’

- fixing parameters, and overriding auto-fixed parameters

```r
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

```r
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)

```r
myModel <- ' y ~ 0.5*x1 + x2 + x3 + b1*x1 '
```
the main fitting function: \texttt{lavaan()}

- the \texttt{lavaan()} function –by default– adds \textit{no} model parameters to the parameter table, nor are any actions taken to identify the model

- nevertheless, as a convenience, several \texttt{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 \texttt{lavaan()} function accepts ‘slots’ (for example, \texttt{slotModel}), perhaps created in a previous run

arguments of the \texttt{lavaan()} fitting function

\begin{verbatim}
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, ...)
\end{verbatim}
example using lavaan with an auto.* argument

```r
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

  ```r
  ?lavOptions
  ```

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$model.type</td>
<td>&quot;sem&quot;</td>
</tr>
<tr>
<td>$mimic</td>
<td>&quot;lavaan&quot;</td>
</tr>
<tr>
<td>$meanstructure</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>$int.ov.free</td>
<td>FALSE</td>
</tr>
<tr>
<td>$int.lv.free</td>
<td>FALSE</td>
</tr>
<tr>
<td>$conditional.x</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>$fixed.x</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>$orthogonal</td>
<td>FALSE</td>
</tr>
<tr>
<td>$std.lv</td>
<td></td>
</tr>
<tr>
<td>$parameterization</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>$auto.fix.first</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.fix.single</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.var</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.cov.lv.x</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.cov.y</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.th</td>
<td>FALSE</td>
</tr>
<tr>
<td>$auto.delta</td>
<td>FALSE</td>
</tr>
<tr>
<td>$std.ov</td>
<td>FALSE</td>
</tr>
<tr>
<td>$missing</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>$sample.cov.rescale</td>
<td>FALSE</td>
</tr>
<tr>
<td>$ridge</td>
<td>FALSE</td>
</tr>
<tr>
<td>$ridge.x</td>
<td>FALSE</td>
</tr>
<tr>
<td>$ridge.constant</td>
<td>1e-05</td>
</tr>
<tr>
<td>$group.label</td>
<td>NULL</td>
</tr>
</tbody>
</table>
|$\text{group.equal}$
[1] ""

|$\text{group.partial}$
[1] ""

|$\text{group.w.free}$
[1] FALSE

|$\text{level.label}$
NULL

|$\text{estimator}$
[1] "default"

|$\text{likelihood}$
[1] "default"

|$\text{link}$
[1] "default"

|$\text{representation}$
[1] "default"

|$\text{do.fit}$
[1] TRUE

|$\text{information}$

[1] "default"

|$\text{h1.information}$
[1] "structured"

|$\text{se}$
[1] "default"

|$\text{test}$
[1] "default"

|$\text{bootstrap}$
[1] 1000

|$\text{observed.information}$
[1] "hessian"

|$\text{gamma.n.minus.one}$
[1] FALSE

|$\text{control}$
list()

|$\text{optim.method}$
[1] "nlminb"

|$\text{optim.force.converged}$
[1] FALSE

|$\text{optim.gradient}$
[1] "analytic"

|$\text{optim.init_nelder_mead}$
[1] FALSE

|$\text{optim.var.transform}$
[1] "none"

|$\text{optim.parscale}$

|$\text{em.iter.max}$
[1] 10000

|$\text{em.fx.tol}$
[1] 1e-08

|$\text{em.dx.tol}$
[1] 1e-04

|$\text{em.zerovar.offset}$
[1] 1e-04
<table>
<thead>
<tr>
<th>Option</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$integration.ngh</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$parallel</code></td>
<td>[1] &quot;no&quot;</td>
</tr>
<tr>
<td><code>$ncpus</code></td>
<td>[1] 1</td>
</tr>
<tr>
<td><code>$cl</code></td>
<td>NULL</td>
</tr>
<tr>
<td><code>$iseed</code></td>
<td>NULL</td>
</tr>
<tr>
<td><code>$zero.add</code></td>
<td>[1] &quot;default&quot;</td>
</tr>
<tr>
<td><code>$zero.keep.margins</code></td>
<td>[1] &quot;default&quot;</td>
</tr>
<tr>
<td><code>$zero.cell.warn</code></td>
<td>[1] FALSE</td>
</tr>
<tr>
<td><code>$start</code></td>
<td>[1] &quot;default&quot;</td>
</tr>
<tr>
<td><code>$check.start</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$check.post</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$check.gradient</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$check.vcov</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$h1</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$baseline</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$baseline.conditional.x.free.slopes</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$implied</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$loglik</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$verbose</code></td>
<td>[1] FALSE</td>
</tr>
<tr>
<td><code>$warn</code></td>
<td>[1] TRUE</td>
</tr>
<tr>
<td><code>$debug</code></td>
<td>[1] FALSE</td>
</tr>
</tbody>
</table>
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

```r
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

<table>
<thead>
<tr>
<th>keyword</th>
<th>operator</th>
<th>parameter set</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto.var</td>
<td>~~</td>
<td>(residual) variances observed and latent variables</td>
</tr>
<tr>
<td>auto.cov.y</td>
<td>~~</td>
<td>(residual) covariances observed and latent endogenous variables</td>
</tr>
<tr>
<td>auto.cov.lv.x</td>
<td>~~</td>
<td>covariances among exogenous latent variables</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>keyword</th>
<th>default</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto.fix.first</td>
<td>TRUE</td>
<td>fix the factor loading of the first indicator to 1</td>
</tr>
<tr>
<td>auto.fix.single</td>
<td>TRUE</td>
<td>fix the residual variance of a single indicator to 1</td>
</tr>
<tr>
<td>int.ov.free</td>
<td>TRUE</td>
<td>freely estimate the intercepts of the observed variables (only if a mean structure is included)</td>
</tr>
<tr>
<td>int.lv.free</td>
<td>FALSE</td>
<td>freely estimate the intercepts of the latent variables (only if a mean structure is included)</td>
</tr>
</tbody>
</table>
## standard R extractor functions

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lavInspect()</td>
<td>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</td>
</tr>
<tr>
<td>inspect()</td>
<td>wrapper around the inspect() with some default options</td>
</tr>
<tr>
<td>lavTech()</td>
<td>same as lavInspect() but without pretty printing; use this within scripts or external packages</td>
</tr>
</tbody>
</table>

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

  `?lavInspect`
## other functions (1)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lavaanify()</code></td>
<td>converts a lavaan model syntax to a parameter table</td>
</tr>
<tr>
<td><code>parameterTable()</code></td>
<td>returns the parameter table</td>
</tr>
<tr>
<td><code>parameterEstimates()</code></td>
<td>returns the parameter estimates, including confidence intervals, as a data frame</td>
</tr>
<tr>
<td><code>standardizedSolution()</code></td>
<td>returns one of three types of standardized parameter estimates, as a data frame</td>
</tr>
<tr>
<td><code>modindices()</code></td>
<td>computes modification indices and expected parameter changes</td>
</tr>
<tr>
<td><code>varTable</code></td>
<td>return information about the observed variables in the model</td>
</tr>
<tr>
<td><code>fitMeasures()</code></td>
<td>return all (=default) or a few selected fit measures</td>
</tr>
<tr>
<td><code>lavNames()</code></td>
<td>extract variables names from a fitted lavaan object</td>
</tr>
</tbody>
</table>
### other functions (2)

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

```r
fit <- cfa(HS.model, data = HolzingerSwinford1939)
fitted(fit)
```

```r
$ 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)

$\lambda$

<table>
<thead>
<tr>
<th></th>
<th>visual</th>
<th>text</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x5</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>x6</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>x7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x8</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>x9</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

$\theta$

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>0</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>x9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>
$\psi$

<table>
<thead>
<tr>
<th>visual</th>
<th>textual</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>19</td>
<td>17</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>18</td>
</tr>
</tbody>
</table>

> `lavInspect(fit, "sampstat")`

$\text{cov}$

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.358</td>
<td>0.407</td>
<td>0.580</td>
<td>0.505</td>
<td>0.441</td>
<td>0.455</td>
<td>0.085</td>
<td>0.264</td>
<td>0.458</td>
</tr>
<tr>
<td>1.382</td>
<td>0.451</td>
<td>0.209</td>
<td>0.208</td>
<td>0.211</td>
<td>0.248</td>
<td>0.097</td>
<td>0.110</td>
<td>0.244</td>
</tr>
<tr>
<td>1.275</td>
<td>1.351</td>
<td>1.015</td>
<td>0.384</td>
<td>1.196</td>
<td>1.183</td>
<td>0.535</td>
<td>1.022</td>
<td>1.015</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>visual</th>
<th>textual</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.809</td>
<td>0.408</td>
<td>0.262</td>
</tr>
<tr>
<td>0.979</td>
<td>0.173</td>
<td>0.384</td>
</tr>
</tbody>
</table>
> 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()`

```r
> fitMeasures(fit)
```

<table>
<thead>
<tr>
<th></th>
<th>fmin</th>
<th>chisq</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>npar</td>
<td>0.142</td>
<td>85.306</td>
<td>24.000</td>
</tr>
<tr>
<td>pvalue</td>
<td>baseline.chisq</td>
<td>36.000</td>
<td>0.000</td>
</tr>
<tr>
<td>cfi</td>
<td>0.896</td>
<td>0.896</td>
<td>0.861</td>
</tr>
<tr>
<td>nfi</td>
<td>0.605</td>
<td>0.931</td>
<td>0.931</td>
</tr>
<tr>
<td>logl</td>
<td>-3695.092</td>
<td>7517.490</td>
<td>7595.339</td>
</tr>
<tr>
<td>ntotal</td>
<td>7528.739</td>
<td>0.092</td>
<td>0.071</td>
</tr>
<tr>
<td>rmsea</td>
<td>0.011</td>
<td>0.082</td>
<td>0.082</td>
</tr>
<tr>
<td>crmr</td>
<td>0.065</td>
<td>0.065</td>
<td>0.065</td>
</tr>
<tr>
<td>mfi</td>
<td>0.943</td>
<td>0.894</td>
<td>0.503</td>
</tr>
<tr>
<td>ecvi</td>
<td>0.423</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
example: parameterTable()

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

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

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

     lhs op rhs est   se z pvalue  ci.lower  ci.upper
 1  visual =~   x1  1.000 0.000 NA  NA  1.0000    1.0000
 2  visual =~   x2  0.554 0.100 5.554  0   0.3580    0.7490
 3  visual =~   x3  0.729 0.109 6.685  0   0.5160    0.9430
 4   textual =~   x4  1.000 0.000 NA  NA  1.0000    1.0000
 5   textual =~   x5  1.113 0.065 17.014 0   0.9850    1.2410
 6   textual =~   x6  0.926 0.055 16.703 0   0.8170    1.0350
 7    speed =~   x7  1.000 0.000 NA  NA  1.0000    1.0000
 8    speed =~   x8  1.180 0.165  7.152 0   0.8570    1.5030
 9    speed =~   x9  1.082 0.151  7.155 0   0.7850    1.3780
 10     x1 =~   x1  0.549 0.114  4.833 0   0.3260    0.7720
 11     x2 =~   x2  1.134 0.102 11.146 0   0.9340    1.3330
 12     x3 =~   x3  0.844 0.091  9.317 0   0.6670    1.0220
 13     x4 =~   x4  0.371 0.048  7.779 0   0.2780    0.4650
 14     x5 =~   x5  0.446 0.058  7.642 0   0.3320    0.5610
 15     x6 =~   x6  0.356 0.043  8.277 0   0.2720    0.4410
 16     x7 =~   x7  0.799 0.081  9.823 0   0.6400    0.9590
 17     x8 =~   x8  0.488 0.074  6.573 0   0.3420    0.6330
 18     x9 =~   x9  0.566 0.071  8.003 0   0.4270    0.7050
 19   visual =~  visual 0.809 0.145  5.564 0   0.5240    1.0940
 20   textual =~ textual 0.979 0.112  8.737 0   0.7600    1.1990
 21    speed =~    speed 0.384 0.086  4.451 0   0.2150    0.5530
```
example: modindices()

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

<table>
<thead>
<tr>
<th>lhs</th>
<th>op</th>
<th>rhs</th>
<th>mi</th>
<th>epc</th>
<th>sepc.lv</th>
<th>sepc.all</th>
<th>sepc.nox</th>
</tr>
</thead>
<tbody>
<tr>
<td>visual =~ x9</td>
<td>36.411</td>
<td>0.577</td>
<td>0.519</td>
<td>0.515</td>
<td>0.515</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7 ~ x8</td>
<td>34.145</td>
<td>0.536</td>
<td>0.536</td>
<td>0.859</td>
<td>0.859</td>
<td></td>
<td></td>
</tr>
<tr>
<td>visual =~ x7</td>
<td>18.631</td>
<td>-0.422</td>
<td>-0.380</td>
<td>-0.349</td>
<td>-0.349</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8 ~ x9</td>
<td>14.946</td>
<td>-0.423</td>
<td>-0.423</td>
<td>-0.805</td>
<td>-0.805</td>
<td></td>
<td></td>
</tr>
<tr>
<td>textual =~ x3</td>
<td>9.151</td>
<td>-0.272</td>
<td>-0.269</td>
<td>-0.238</td>
<td>-0.238</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2 ~ x7</td>
<td>8.918</td>
<td>-0.183</td>
<td>-0.183</td>
<td>-0.192</td>
<td>-0.192</td>
<td></td>
<td></td>
</tr>
<tr>
<td>textual =~ x1</td>
<td>8.903</td>
<td>0.350</td>
<td>0.347</td>
<td>0.297</td>
<td>0.297</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2 ~ x3</td>
<td>8.532</td>
<td>0.218</td>
<td>0.218</td>
<td>0.223</td>
<td>0.223</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3 ~ x5</td>
<td>7.858</td>
<td>-0.130</td>
<td>-0.130</td>
<td>-0.212</td>
<td>-0.212</td>
<td></td>
<td></td>
</tr>
<tr>
<td>visual =~ x5</td>
<td>7.441</td>
<td>-0.210</td>
<td>-0.189</td>
<td>-0.147</td>
<td>-0.147</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1 ~ x9</td>
<td>7.335</td>
<td>0.138</td>
<td>0.138</td>
<td>0.247</td>
<td>0.247</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4 ~ x6</td>
<td>6.220</td>
<td>-0.235</td>
<td>-0.235</td>
<td>-0.646</td>
<td>-0.646</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4 ~ x7</td>
<td>5.920</td>
<td>0.098</td>
<td>0.098</td>
<td>0.180</td>
<td>0.180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1 ~ x7</td>
<td>5.420</td>
<td>-0.129</td>
<td>-0.129</td>
<td>-0.195</td>
<td>-0.195</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7 ~ x9</td>
<td>5.183</td>
<td>-0.187</td>
<td>-0.187</td>
<td>-0.278</td>
<td>-0.278</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
example: `lavTestScore()`

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

$test

**total score test:**

<table>
<thead>
<tr>
<th>test</th>
<th>X2</th>
<th>df</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>score</td>
<td>36.411</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$uni

**univariate score tests:**

<table>
<thead>
<tr>
<th>lhs</th>
<th>op</th>
<th>rhs</th>
<th>X2</th>
<th>df</th>
<th>p.value</th>
</tr>
</thead>
<tbody>
<tr>
<td>visual=〜x9</td>
<td>==</td>
<td>0</td>
<td>36.411</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
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

Yves Rosseel
Structural Equation Modeling with lavaan
\begin{verbatim}
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

$s\text{ummary}$

\begin{verbatim}
srmr  srmr.se  srmr.z  srmr.pvalue  usrmr  usrmr.se  
\text{cov}  0.065  0.006  6.063  0  0.058  0.01
\end{verbatim}
\end{verbatim}
**example: lavTestLRT()**

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

**Chi Square Difference Test**

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>Chisq</th>
<th>Chisq diff</th>
<th>Df diff</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit</td>
<td>24</td>
<td>7517.5</td>
<td>7595.3</td>
<td>85.305</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit0</td>
<td>27</td>
<td>7579.7</td>
<td>7646.4</td>
<td>153.527</td>
<td>68.222</td>
<td>3</td>
<td>1.026e-14</td>
</tr>
</tbody>
</table>

---

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

```r
> 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 0.6-4 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:

| Latent Variable | 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    |         |         |         |
speed = ~

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x5</td>
<td>1.113</td>
<td>0.065</td>
<td>17.014</td>
<td>0.000</td>
</tr>
<tr>
<td>x6</td>
<td>0.926</td>
<td>0.055</td>
<td>16.703</td>
<td>0.000</td>
</tr>
<tr>
<td>x7</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>1.180</td>
<td>0.165</td>
<td>7.152</td>
<td>0.000</td>
</tr>
<tr>
<td>x9</td>
<td>1.082</td>
<td>0.151</td>
<td>7.155</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Covariances:

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

Intercepts:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>.x1</td>
<td>4.936</td>
<td>0.067</td>
<td>73.473</td>
<td>0.000</td>
</tr>
<tr>
<td>.x2</td>
<td>6.088</td>
<td>0.068</td>
<td>89.855</td>
<td>0.000</td>
</tr>
<tr>
<td>.x3</td>
<td>2.250</td>
<td>0.065</td>
<td>34.579</td>
<td>0.000</td>
</tr>
<tr>
<td>.x4</td>
<td>3.061</td>
<td>0.067</td>
<td>45.694</td>
<td>0.000</td>
</tr>
<tr>
<td>.x5</td>
<td>4.341</td>
<td>0.074</td>
<td>58.452</td>
<td>0.000</td>
</tr>
<tr>
<td>.x6</td>
<td>2.186</td>
<td>0.063</td>
<td>34.667</td>
<td>0.000</td>
</tr>
<tr>
<td>.x7</td>
<td>4.186</td>
<td>0.063</td>
<td>66.766</td>
<td>0.000</td>
</tr>
<tr>
<td>.x8</td>
<td>5.527</td>
<td>0.058</td>
<td>94.854</td>
<td>0.000</td>
</tr>
<tr>
<td>.x9</td>
<td>5.374</td>
<td>0.058</td>
<td>92.546</td>
<td>0.000</td>
</tr>
<tr>
<td>visual</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>textual</td>
<td>0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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)

GROUP 1

\[ y_1 \rightarrow f_1 \rightarrow y_4 \rightarrow y_2 \rightarrow y_3 \rightarrow y_5 \rightarrow y_6 \]

GROUP 2

\[ y_1 \rightarrow f_1 \rightarrow y_4 \rightarrow y_2 \rightarrow y_3 \rightarrow y_5 \rightarrow y_6 \]

- 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
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

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>Chisq</th>
<th>Chisq diff</th>
<th>Df</th>
<th>diff</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>48</td>
<td>7484.4</td>
<td>7706.8</td>
<td>115.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit2</td>
<td>54</td>
<td>7480.6</td>
<td>7680.8</td>
<td>124.04</td>
<td>8.1922</td>
<td>6</td>
<td>0.2244</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>chisq</th>
<th>df</th>
<th>pvalue</th>
<th>cfi</th>
<th>rmsea</th>
<th>srmr</th>
</tr>
</thead>
<tbody>
<tr>
<td>164.103</td>
<td>60</td>
<td>0.000</td>
<td>0.882</td>
<td>0.107</td>
<td>0.082</td>
</tr>
</tbody>
</table>
```

• step 3a: compare with weak invariance model

```r
> anova(fit2, fit3)

Chi Square Difference Test

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>Chisq</th>
<th>Chisq diff</th>
<th>Df</th>
<th>diff</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit2</td>
<td>54</td>
<td>7480.6</td>
<td>7680.8</td>
<td>124.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit3</td>
<td>60</td>
<td>7508.6</td>
<td>7686.6</td>
<td>164.10</td>
<td>40.059</td>
<td>6</td>
<td>4.435e-07</td>
<td>***</td>
</tr>
</tbody>
</table>
```

---
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(12.1, 12.2) *x2 + c(13.1, 13.2) *x3
+   textual =~ c(1,1) *x4 + c(15.1, 15.2) *x5 + c(16.1, 16.2) *x6
+   speed =~ c(1,1) *x7 + c(18.1, 18.2) *x8 + c(19.1, 19.2) *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 = HolzingerSwinford1939,
+    group = "school")
> # weak invariance model (manual)
> # equal factor loadings
> HS.model.weak <- ' + visual =~ c(1,1) *x1 + c(12, 12)*x2 + c(13, 13)*x3 + textual =~ c(1,1) *x4 + c(15, 15)*x5 + c(16, 16)*x6 + speed =~ c(1,1) *x7 + c(18, 18)*x8 + c(19, 19)*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 <- '
\[
\begin{align*}
\text{visual} & \sim c(1, 1) x_1 + c(12, 12) x_2 + c(13, 13) x_3 \\
\text{textual} & \sim c(1, 1) x_4 + c(15, 15) x_5 + c(16, 16) x_6 \\
\text{speed} & \sim c(1, 1) x_7 + c(18, 18) x_8 + c(19, 19) x_9 \\
\end{align*}
\]

+ # ov intercepts
+ \( x_1 \sim c(i1, i1) * 1 \)
+ \( x_2 \sim c(i2, i2) * 1 \)
+ \( x_3 \sim c(i3, i3) * 1 \)
+ \( x_4 \sim c(i4, i4) * 1 \)
+ \( x_5 \sim c(i5, i5) * 1 \)
+ \( x_6 \sim c(i6, i6) * 1 \)
+ \( x_7 \sim c(i7, i7) * 1 \)
+ \( x_8 \sim c(i8, i8) * 1 \)
+ \( x_9 \sim c(i9, i9) * 1 \)
+
+ # lv means
+ \( \text{visual} \sim c(0, NA) * 1 \)
+ \( \text{textual} \sim c(0, NA) * 1 \)
+ \( \text{speed} \sim c(0, NA) * 1 \)
+
> fit3b <- cfa(HS.model.strong, data = HolzingerSwinford1939, 
+ group = "school")
output strong invariance model

lavaan 0.6-4 ended normally after 61 iterations

Optimization method  NLMINB
Number of free parameters  63
Number of equality constraints  15
Row rank of the constraints matrix  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:

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Structured</td>
</tr>
<tr>
<td>Standard Errors</td>
<td>Standard</td>
</tr>
</tbody>
</table>
Group 1 [Pasteur]:

Latent Variables:

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

| Latent Variable | 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:

| Covariance     | 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:
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>.x1</td>
<td>(i1)</td>
<td>5.001</td>
<td>0.090</td>
<td>55.760</td>
</tr>
<tr>
<td>.x2</td>
<td>(i2)</td>
<td>6.151</td>
<td>0.077</td>
<td>79.905</td>
</tr>
<tr>
<td>.x3</td>
<td>(i3)</td>
<td>2.271</td>
<td>0.083</td>
<td>27.387</td>
</tr>
<tr>
<td>.x4</td>
<td>(i4)</td>
<td>2.778</td>
<td>0.087</td>
<td>31.954</td>
</tr>
<tr>
<td>.x5</td>
<td>(i5)</td>
<td>4.035</td>
<td>0.096</td>
<td>41.858</td>
</tr>
<tr>
<td>.x6</td>
<td>(i6)</td>
<td>1.926</td>
<td>0.079</td>
<td>24.426</td>
</tr>
<tr>
<td>.x7</td>
<td>(i7)</td>
<td>4.242</td>
<td>0.073</td>
<td>57.975</td>
</tr>
<tr>
<td>.x8</td>
<td>(i8)</td>
<td>5.630</td>
<td>0.072</td>
<td>78.531</td>
</tr>
<tr>
<td>.x9</td>
<td>(i9)</td>
<td>5.465</td>
<td>0.069</td>
<td>79.016</td>
</tr>
<tr>
<td>visual</td>
<td></td>
<td>-0.148</td>
<td>0.122</td>
<td>-1.211</td>
</tr>
<tr>
<td>textual</td>
<td></td>
<td>0.576</td>
<td>0.117</td>
<td>4.918</td>
</tr>
<tr>
<td>speed</td>
<td></td>
<td>-0.177</td>
<td>0.090</td>
<td>-1.968</td>
</tr>
</tbody>
</table>

Variances:

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>.x1</td>
<td>0.654</td>
<td>0.128</td>
<td>5.094</td>
<td>0.000</td>
</tr>
<tr>
<td>.x2</td>
<td>0.964</td>
<td>0.123</td>
<td>7.812</td>
<td>0.000</td>
</tr>
<tr>
<td>.x3</td>
<td>0.641</td>
<td>0.101</td>
<td>6.316</td>
<td>0.000</td>
</tr>
<tr>
<td>.x4</td>
<td>0.343</td>
<td>0.062</td>
<td>5.534</td>
<td>0.000</td>
</tr>
<tr>
<td>.x5</td>
<td>0.376</td>
<td>0.073</td>
<td>5.133</td>
<td>0.000</td>
</tr>
<tr>
<td>.x6</td>
<td>0.437</td>
<td>0.067</td>
<td>6.559</td>
<td>0.000</td>
</tr>
<tr>
<td>.x7</td>
<td>0.625</td>
<td>0.095</td>
<td>6.574</td>
<td>0.000</td>
</tr>
<tr>
<td>.x8</td>
<td>0.434</td>
<td>0.088</td>
<td>4.914</td>
<td>0.000</td>
</tr>
<tr>
<td>.x9</td>
<td>0.522</td>
<td>0.086</td>
<td>6.102</td>
<td>0.000</td>
</tr>
<tr>
<td>visual</td>
<td>0.708</td>
<td>0.160</td>
<td>4.417</td>
<td>0.000</td>
</tr>
<tr>
<td>textual</td>
<td>0.870</td>
<td>0.131</td>
<td>6.659</td>
<td>0.000</td>
</tr>
<tr>
<td>speed</td>
<td>0.505</td>
<td>0.115</td>
<td>4.379</td>
<td>0.000</td>
</tr>
</tbody>
</table>
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 practice
  
  - 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

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

- estimated latent means:

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

<table>
<thead>
<tr>
<th>lhs</th>
<th>op</th>
<th>rhs</th>
<th>block</th>
<th>group</th>
<th>label</th>
<th>est</th>
<th>se</th>
<th>z</th>
<th>pvalue</th>
<th>ci.lower</th>
<th>ci.upper</th>
</tr>
</thead>
<tbody>
<tr>
<td>visual</td>
<td>~1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-0.148</td>
<td>0.122</td>
<td>-1.211</td>
<td>0.226</td>
<td>-0.387</td>
<td></td>
<td></td>
</tr>
<tr>
<td>textual</td>
<td>~1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.576</td>
<td>0.117</td>
<td>4.918</td>
<td>0.000</td>
<td>0.347</td>
<td></td>
<td></td>
</tr>
<tr>
<td>speed</td>
<td>~1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-0.177</td>
<td>0.090</td>
<td>-1.968</td>
<td>0.049</td>
<td>-0.354</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Yves Rosseel  Structural Equation Modeling with lavaan
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:

```r
> 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 block group free label plabel est   epc   epv  sepc.lv
70  visual  ~1   2   2   61        .p70. -0.148 0.015 -0.133 0.018
71  textual  ~1   2   2   62        .p71.  0.576 0.019  0.596 0.021
72   speed  ~1   2   2   63        .p72. -0.177 0.001 -0.176 0.011

   sepc.all sepc.nox
70 0.018   0.018
71 0.021   0.021
72 0.001   0.001
```
3.5 Measurement invariance: recent developments and references

- exploratory 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:

• general references:

• reviews:

• testing strategies:


• partial invariance:

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)

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

<table>
<thead>
<tr>
<th>Number of observations</th>
<th>Used</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>156</td>
<td>301</td>
</tr>
</tbody>
</table>

– 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

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

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>0.967</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>0.970</td>
<td>0.967</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>0.970</td>
<td>0.967</td>
<td>0.967</td>
<td>0.970</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.970</td>
<td>0.967</td>
<td>0.967</td>
<td>0.983</td>
<td></td>
</tr>
<tr>
<td>x9</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.967</td>
<td>0.970</td>
<td>0.967</td>
<td>0.983</td>
</tr>
</tbody>
</table>

> # sample statistics unrestricted (h1) model
> lavInspect(fit, "sampstat.h1")
\begin{verbatim}
$ 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 
\end{verbatim}

- 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

```r
> 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:

\[ F_{WLS} = (s - \hat{\sigma})^\top W^{-1} (s - \hat{\sigma}) \]

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

- the weight matrix \( W \) 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:

```r
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

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

- Satorra-Bentler scaled test statistic

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

- robust standard errors + scaled test statistic

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

- estimator MLM = robust standard errors + scaled test statistic

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

- alternative: estimator MLR (also for missing data)

```r
fit <- cfa(HS.model, data = HolzingerSwineford1939, estimator = "MLR", missing = "ml")
```
output: robust standard errors and scaled test statistic

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

lavaan 0.6-3 ended normally after 35 iterations

<table>
<thead>
<tr>
<th>Optimization method</th>
<th>NLMINB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of free parameters</td>
<td>21</td>
</tr>
<tr>
<td>Number of observations</td>
<td>301</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimator</th>
<th>ML</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Fit Test Statistic</td>
<td>85.306</td>
<td>80.872</td>
</tr>
<tr>
<td>Degrees of freedom</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>P-value (Chi-square)</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Scaling correction factor</td>
<td>1.055</td>
<td></td>
</tr>
</tbody>
</table>

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:

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

- bootstrapping the test statistic

  ```r
  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)

```r
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 $y^*$; e.g. ordinal variable with $K = 4$ response categories:
stage 1 – estimating the thresholds in R

- if no exogenous variables, this is just

```r
> 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
0.19 0.29 0.23 0.29

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

1 2 3 4
0.00 0.19 0.48 0.71 1.00

> # convert quantiles to z-scores
> th <- qnorm(cprop)
> th
```
• in the presence of exogenous covariates, this is just ordered probit regression

```r
> 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
-Inf -0.87789630 -0.05015358 0.55338472 Inf
```

```
  1|2    2|3    3|4
-0.86055660 -0.02851101 0.58377987
```
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 $y^*$ (eg bivariate probit regression)
stage 2 – tetrachoric, polychoric, . . ., correlations in R

• lavaan provides the \texttt{lavCor()} function to compute the tetrachoric, polychoric, polyserial, . . . correlations

• example using two binary variables:

```r
> 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, \( W \) 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:

```r
> 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:

\[ F_{WLS} = (s - \hat{\sigma})^\top W^{-1} (s - \hat{\sigma}) \]

where \( s \) and \( \hat{\sigma} \) are vectors containing all relevant sample-based and model-based statistics respectively.

- \( s \) contains: thresholds, correlations, optionally regression slopes of exogenous covariates, optionally variances and means of continuous variables.

- the weight matrix \( W \) is (a consistent estimator of) the asymptotic covariance matrix of the sample statistics (\( s \)).

- robust version: WLSMV
  - use the diagonal of \( W \) 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 choices for $W$, leading to three different estimators:
  – estimator WLS: the full weight matrix $W$ is used during estimation
  – estimator DWLS: only the diagonal of $W$ is used during estimation
  – estimator ULS: $W$ is replaced by the identity matrix ($I$)

• 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

  ```r
  > 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:

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

<table>
<thead>
<tr>
<th>name</th>
<th>idx</th>
<th>nosb</th>
<th>type</th>
<th>exo</th>
<th>user</th>
<th>mean</th>
<th>var</th>
<th>nlev</th>
<th>lnam</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>1</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>176.555</td>
<td>11222.961</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sex</td>
<td>2</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>1.515</td>
<td>0.251</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ageyr</td>
<td>3</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>12.997</td>
<td>1.103</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>agemo</td>
<td>4</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>5.375</td>
<td>11.915</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>school</td>
<td>5</td>
<td>301</td>
<td>factor</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>2</td>
<td>Grant-White</td>
</tr>
<tr>
<td>grade</td>
<td>6</td>
<td>300</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>7.477</td>
<td>0.250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x1</td>
<td>7</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>4.936</td>
<td>1.363</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td>8</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>6.088</td>
<td>1.386</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>9</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>2.250</td>
<td>1.279</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>10</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>3.061</td>
<td>1.355</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>11</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>4.341</td>
<td>1.663</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>12</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>2.186</td>
<td>1.200</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>13</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>4.186</td>
<td>1.187</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td>14</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>5.527</td>
<td>1.025</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>x9</td>
<td>15</td>
<td>301</td>
<td>numeric</td>
<td>0</td>
<td>0</td>
<td>5.374</td>
<td>1.018</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
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, item4) as ordinal in your data.frame (called ‘Data’), you can use something like:

     ```r
     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:

     ```r
     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 0.6-6.1508 ended normally after 35 iterations

<table>
<thead>
<tr>
<th>Model Test千伏 Model:</th>
<th>Standard</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Statistic</td>
<td>30.918</td>
<td>38.427</td>
</tr>
<tr>
<td>Degrees of freedom</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>P-value (Chi-square)</td>
<td>0.156</td>
<td>0.031</td>
</tr>
<tr>
<td>Scaling correction factor</td>
<td>0.869</td>
<td></td>
</tr>
<tr>
<td>Shift parameter</td>
<td>2.861</td>
<td></td>
</tr>
</tbody>
</table>

for the simple second-order correction

<table>
<thead>
<tr>
<th>Model Test Baseline Model:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test statistic</td>
</tr>
<tr>
<td>Degrees of freedom</td>
</tr>
<tr>
<td>P-value</td>
</tr>
<tr>
<td>Scaling correction factor</td>
</tr>
</tbody>
</table>
### User Model versus Baseline Model:

<table>
<thead>
<tr>
<th>Fit Measure</th>
<th>User Model</th>
<th>Baseline Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparative Fit Index (CFI)</td>
<td>0.987</td>
<td>0.967</td>
</tr>
<tr>
<td>Tucker–Lewis Index (TLI)</td>
<td>0.981</td>
<td>0.950</td>
</tr>
<tr>
<td>Robust Comparative Fit Index (CFI)</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>Robust Tucker–Lewis Index (TLI)</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

### Root Mean Square Error of Approximation:

<table>
<thead>
<tr>
<th>Error Measure</th>
<th>User Model</th>
<th>Baseline Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMSEA</td>
<td>0.031</td>
<td>0.045</td>
</tr>
<tr>
<td>90 Percent confidence interval - lower</td>
<td>0.000</td>
<td>0.014</td>
</tr>
<tr>
<td>90 Percent confidence interval - upper</td>
<td>0.059</td>
<td>0.070</td>
</tr>
<tr>
<td>P-value RMSEA &lt;= 0.05</td>
<td>0.847</td>
<td>0.600</td>
</tr>
<tr>
<td>Robust RMSEA</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>90 Percent confidence interval - lower</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>90 Percent confidence interval - upper</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>

### Standardized Root Mean Square Residual:

<table>
<thead>
<tr>
<th>Residual</th>
<th>User Model</th>
<th>Baseline Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRMR</td>
<td>0.083</td>
<td>0.083</td>
</tr>
</tbody>
</table>

### Parameter Estimates:

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Unstructured</td>
</tr>
<tr>
<td>Standard errors</td>
<td>Robust.sem</td>
</tr>
</tbody>
</table>
### Latent Variables:

|         | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|---------|----------|---------|---------|---------|--------|---------|
| visual  |          |         |         |         |        |         |
| x1      | 1.000    | 0.639   | 0.639   | 0.000   | 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   | 0.000   | 0.815  | 0.815   |
| 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   | 0.000   | 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   |
### Thresholds:

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

|       | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|-------|----------|---------|---------|---------|--------|---------|
| 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   |
estimated thresholds and tetrachoric correlations

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

```r
$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)

$\lambda$

<table>
<thead>
<tr>
<th></th>
<th>visual</th>
<th>textul</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x5</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>x6</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>x7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x8</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>x9</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

$\theta$

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
<th>x9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Yves Rosseel Structural Equation Modeling with lavaan
$\psi$

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
\texttt{x3|t1} 9
\texttt{x4|t1} 10
\texttt{x5|t1} 11
\texttt{x6|t1} 12
\texttt{x7|t1} 13
\texttt{x8|t1} 14
\texttt{x9|t1} 15

$\delta$
\begin{tabular}{ll}
\texttt{scales} \\
\hline
\texttt{x1} & 0 \\
\texttt{x2} & 0 \\
\texttt{x3} & 0 \\
\texttt{x4} & 0 \\
\texttt{x5} & 0 \\
\texttt{x6} & 0 \\
\texttt{x7} & 0 \\
\texttt{x8} & 0 \\
\texttt{x9} & 0 \\
\end{tabular}
### tables: univariate

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

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

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

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

the connection with IRT

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


• 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: $\lambda_i$, $\tau_i$ and $\theta_i$ are the factor loadings, the thresholds, and the residual variances)
  – in IRT: $\alpha_i$ and $\beta_i$ 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
Books

6.1 Repeated measures ANOVA in 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 \texttt{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

```r
> 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
```

<table>
<thead>
<tr>
<th></th>
<th>Glad1</th>
<th>Cheer1</th>
<th>Happy1</th>
<th>Glad2</th>
<th>Cheer2</th>
<th>Happy2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glad1</td>
<td>0.7088630</td>
<td>0.4135162</td>
<td>0.3953488</td>
<td>0.2257459</td>
<td>0.1863819</td>
<td>0.2278048</td>
</tr>
<tr>
<td>Cheer1</td>
<td>0.4135162</td>
<td>0.7909256</td>
<td>0.4476782</td>
<td>0.2684330</td>
<td>0.2892800</td>
<td>0.2717608</td>
</tr>
<tr>
<td>Happy1</td>
<td>0.3953488</td>
<td>0.4476782</td>
<td>0.6967241</td>
<td>0.1948053</td>
<td>0.2418595</td>
<td>0.2248294</td>
</tr>
<tr>
<td>Glad2</td>
<td>0.2257459</td>
<td>0.2684330</td>
<td>0.1948053</td>
<td>0.7069615</td>
<td>0.4279434</td>
<td>0.3965998</td>
</tr>
<tr>
<td>Cheer2</td>
<td>0.1863819</td>
<td>0.2892800</td>
<td>0.2418595</td>
<td>0.4279434</td>
<td>0.8256266</td>
<td>0.4558680</td>
</tr>
<tr>
<td>Happy2</td>
<td>0.2278048</td>
<td>0.2717608</td>
<td>0.2248294</td>
<td>0.3965998</td>
<td>0.4558680</td>
<td>0.7053312</td>
</tr>
</tbody>
</table>
R code: fitting the ‘configural’ longitudinal CFA model

```r
> 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)
```
lavaan 0.6–6.1508 ended normally after 28 iterations

Estimator  
ML
Optimization method  
NLMINB
Number of free parameters  
22

Number of observations  
823

Model Test User Model:

Test statistic  
18.432
Degrees of freedom  
5
P-value (Chi-square)  
0.002

Parameter Estimates:

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

Latent Variables:

| Latent Variable | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|-----------------|----------|---------|---------|---------|--------|---------|
| posAffect1 =~   |          |         |         |         |        |         |
| Glad1           | 1.000    |         |         |         | 0.599  | 0.712   |
| Cheer1          | 1.162    | 0.064   | 18.240  | 0.000   | 0.697  | 0.783   |
| Happy1          | 1.080    | 0.060   | 18.055  | 0.000   | 0.647  | 0.776   |
| posAffect2 =~   |          |         |         |         |        |         |
| Glad2           | 1.000    |         |         |         | 0.610  | 0.725   |
## 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 | 0.031 | 0.015 | 2.055 | 0.040 | 0.031 | 0.092 |
| .Cheer1 ~ .Cheer2 | 0.018 | 0.016 | 1.076 | 0.282 | 0.018 | 0.055 |
| .Happy1 ~ .Happy2 | -0.011 | 0.014 | -0.767 | 0.443 | -0.011 | -0.039 |

## Intercepts:

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| .Glad1 | 3.069 | 0.029 | 104.641 | 0.000 | 3.069 | 3.648 |
| .Glad2 | 3.026 | 0.029 | 103.189 | 0.000 | 3.026 | 3.597 |
| .Cheer1 | 2.926 | 0.031 | 94.378 | 0.000 | 2.926 | 3.290 |
| .Cheer2 | 2.857 | 0.032 | 90.334 | 0.000 | 2.857 | 3.149 |
| .Happy1 | 3.110 | 0.029 | 106.913 | 0.000 | 3.110 | 3.727 |
| .Happy2 | 3.093 | 0.029 | 105.772 | 0.000 | 3.093 | 3.687 |
| posAffect1 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| posAffect2 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

## Variances:

| Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|----------|---------|---------|---------|--------|---------|
| .Glad1 | 0.349 | 0.023 | 15.214 | 0.000 | 0.349 | 0.492 |

---

Yves Rosseel

Structural Equation Modeling with lavaan
<table>
<thead>
<tr>
<th>Variable</th>
<th>M</th>
<th>SD</th>
<th>t</th>
<th>p</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheer1</td>
<td>0.306</td>
<td>0.025</td>
<td>12.199</td>
<td>0.000</td>
<td>0.306</td>
<td>0.387</td>
</tr>
<tr>
<td>Happy1</td>
<td>0.278</td>
<td>0.022</td>
<td>12.455</td>
<td>0.000</td>
<td>0.278</td>
<td>0.398</td>
</tr>
<tr>
<td>Glad2</td>
<td>0.335</td>
<td>0.023</td>
<td>14.746</td>
<td>0.000</td>
<td>0.335</td>
<td>0.474</td>
</tr>
<tr>
<td>Cheer2</td>
<td>0.343</td>
<td>0.026</td>
<td>13.084</td>
<td>0.000</td>
<td>0.343</td>
<td>0.416</td>
</tr>
<tr>
<td>Happy2</td>
<td>0.274</td>
<td>0.023</td>
<td>12.153</td>
<td>0.000</td>
<td>0.274</td>
<td>0.389</td>
</tr>
<tr>
<td>posAffect1</td>
<td>0.359</td>
<td>0.034</td>
<td>10.604</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>posAffect2</td>
<td>0.372</td>
<td>0.034</td>
<td>10.819</td>
<td>0.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>
R code: fitting the ‘weak invariance’ longitudinal CFA model

```r
> 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)
```
lavaan 0.6-6.1508 ended normally after 27 iterations

   Estimator                   ML
Optimization method       NLMINB
Number of free parameters   22
Number of equality constraints  2
Row rank of the constraints matrix   2

Number of observations     823

Model Test User Model:

  Test statistic                 18.534
  Degrees of freedom            7
  P-value (Chi-square)           0.010

Parameter Estimates:

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

Latent Variables:

| Latent Variable | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|-----------------|----------|---------|---------|---------|--------|---------|
| posAffect1 =~   |          |         |         |         |        |         |
| Glad1           | 1.000    |         |         |         | 0.602  | 0.715   |
| Cheer1 (ch)     | 1.149    | 0.046   | 25.036  | 0.000   | 0.692  | 0.780   |
| Happy1 (ha)     | 1.078    | 0.043   | 25.189  | 0.000   | 0.649  | 0.777   |
```
posAffect2 =~
  Glad2 1.000 0.607 0.723
  Cheer2 (ch) 1.149 0.046 25.036 0.000 0.697 0.767
  Happy2 (ha) 1.078 0.043 25.189 0.000 0.654 0.781

Covariances:

|                  | Estimate | Std.Err | z-value | P(>|z|) | Std.lv   | Std.all |
|------------------|----------|---------|---------|---------|----------|---------|
| posAffect1 ~ posAffect2 | 0.202    | 0.021   | 9.834   | 0.000   | 0.552    | 0.552   |
| .Glad1 ~ .Glad2   | 0.032    | 0.015   | 2.076   | 0.038   | 0.032    | 0.092   |
| .Cheer1 ~ .Cheer2 | 0.018    | 0.016   | 1.072   | 0.284   | 0.018    | 0.054   |
| .Happy1 ~ .Happy2 | -0.011   | 0.014   | -0.793  | 0.428   | -0.011   | -0.041  |

Intercepts:

|                  | Estimate | Std.Err | z-value | P(>|z|) | Std.lv   | Std.all |
|------------------|----------|---------|---------|---------|----------|---------|
| .Glad1           | 3.069    | 0.029   | 104.457 | 0.000   | 3.069    | 3.641   |
| .Glad2           | 3.026    | 0.029   | 103.362 | 0.000   | 3.026    | 3.603   |
| .Cheer1          | 2.926    | 0.031   | 94.586  | 0.000   | 2.926    | 3.297   |
| .Cheer2          | 2.857    | 0.032   | 90.126  | 0.000   | 2.857    | 3.142   |
| .Happy1          | 3.110    | 0.029   | 106.821 | 0.000   | 3.110    | 3.724   |
| .Happy2          | 3.093    | 0.029   | 105.859 | 0.000   | 3.093    | 3.690   |
| posAffect1       | 0.000    |         |         |         | 0.000    | 0.000   |
| posAffect2       | 0.000    |         |         |         | 0.000    | 0.000   |

Variances:
```

Yves Rosseel
|      | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|------|----------|---------|---------|---------|--------|---------|
| Glad1| 0.347    | 0.022   | 15.605  | 0.000   | 0.347  | 0.489   |
| Cheer1| 0.308    | 0.024   | 13.050  | 0.000   | 0.308  | 0.392   |
| Happy1| 0.276    | 0.021   | 13.094  | 0.000   | 0.276  | 0.396   |
| Glad2| 0.337    | 0.022   | 15.318  | 0.000   | 0.337  | 0.477   |
| Cheer2| 0.340    | 0.025   | 13.597  | 0.000   | 0.340  | 0.412   |
| Happy2| 0.275    | 0.021   | 12.929  | 0.000   | 0.275  | 0.391   |
| posAffect1| 0.363 | 0.029   | 12.394  | 0.000   | 1.000  | 1.000   |
| posAffect2| 0.369 | 0.030   | 12.471  | 0.000   | 1.000  | 1.000   |
R code: testing for weak invariance

• compare the configural and the weak invariance models:

```r
> anova(fit1, fit2)

Chi-Squared 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

```r
> 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 0.6-6.1508 ended normally after 36 iterations

Estimator ML
Optimization method NLMINB
Number of free parameters 23
Number of equality constraints 5
Row rank of the constraints matrix 5

Number of observations 823

Model Test User Model:

Test statistic 20.279
Degrees of freedom 9
P-value (Chi-square) 0.016

Parameter Estimates:

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

Latent Variables:

| posAffect1 =~ | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|--------------|---------|---------|---------|--------|--------|---------|
| 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   |
\[
\text{posAffect2} = \sim \\
\begin{align*}
\text{Glad2} & \quad 1.000 & \quad 0.607 & \quad 0.723 \\
\text{Cheer2} \quad (\text{ch}) & \quad 1.150 & \quad 0.698 & \quad 0.768 \\
\text{Happy2} \quad (\text{ha}) & \quad 1.076 & \quad 0.653 & \quad 0.780 \\
\end{align*}
\]

**Covariances:**

|                     | Estimate | Std.Err | z-value | P(>|z|) | Std.lv | Std.all |
|---------------------|----------|---------|---------|---------|--------|---------|
| \text{posAffect1} \sim \text{posAffect2} | 0.202    | 0.021   | 9.840   | 0.000   | 0.553  | 0.553   |
| \text{Glad1} \sim \text{Glad2}          | 0.032    | 0.015   | 2.074   | 0.038   | 0.032  | 0.092   |
| \text{Cheer1} \sim \text{Cheer2}        | 0.017    | 0.016   | 1.047   | 0.295   | 0.017  | 0.053   |
| \text{Happy1} \sim \text{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 |
|---------------------|----------|---------|---------|---------|--------|---------|
| \text{Glad1} \quad (\text{igl}) | 3.067   | 0.027   | 114.088 | 0.000   | 3.067  | 3.639   |
| \text{Glad2} \quad (\text{igl}) | 3.067   | 0.027   | 114.088 | 0.000   | 3.067  | 3.652   |
| \text{Cheer1} \quad (\text{ich}) | 2.915   | 0.029   | 99.814  | 0.000   | 2.915  | 3.283   |
| \text{Cheer2} \quad (\text{ich}) | 2.915   | 0.029   | 99.814  | 0.000   | 2.915  | 3.204   |
| \text{Happy1} \quad (\text{iha}) | 3.123   | 0.027   | 115.351 | 0.000   | 3.123  | 3.740   |
| \text{Happy2} \quad (\text{iha}) | 3.123   | 0.027   | 115.351 | 0.000   | 3.123  | 3.726   |
| \text{psAffect1}    | 0.000    |         |         |         |        |         |
| \text{psAffect2}    | -0.040   | 0.025   | -1.617  | 0.106   | -0.066 | -0.066  |

**Variances:**

[Table continues with variances for each variable]
|        | 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)
```

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>Chisq</th>
<th>Chisq diff</th>
<th>Df diff</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit2</td>
<td>7</td>
<td>10800</td>
<td>10894</td>
<td>18.534</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit3</td>
<td>9</td>
<td>10798</td>
<td>10883</td>
<td>20.279</td>
<td>1.7451</td>
<td>2</td>
<td>0.4179</td>
</tr>
</tbody>
</table>

- 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 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

```r
> 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:

```r
> anova(fit3, fit4)
```

**Chi-Squared Difference Test**

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>AIC</th>
<th>BIC</th>
<th>Chisq</th>
<th>Chisq diff</th>
<th>Df</th>
<th>diff</th>
<th>Pr(&gt;Chisq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit3</td>
<td>9</td>
<td>10798</td>
<td>10883</td>
<td>20.279</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fit4</td>
<td>10</td>
<td>10799</td>
<td>10879</td>
<td>22.889</td>
<td>2.6099</td>
<td>1</td>
<td>0.1062</td>
<td></td>
</tr>
</tbody>
</table>

• 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:

```
\[ Y_{11} \quad Y_{21} \quad Y_{31} \quad Y_{12} \quad Y_{22} \quad Y_{32} \]
```

```
\[ \epsilon_{11} \quad \epsilon_{21} \quad \epsilon_{31} \quad \epsilon_{12} \quad \epsilon_{22} \quad \epsilon_{32} \]
```

```
\[ f_1 \quad f_2 \]
```

time 1          time 2
**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?
test of equality of means over time: antisocial behavior

• model with 4 means, unequal variances, fully correlated

```r
> model <- ' + # four repeated means + anti1 ~ i1*I; anti2 ~ i2*I; anti3 ~ i3*I; anti4 ~ i4*I + + # four unequal variances + anti1 ~~ anti1; anti2 ~~ anti2; anti3 ~~ anti3; anti4 ~~ anti4 + + # fully correlated + anti1 ~~ anti2 + anti3 + anti4 + anti2 ~~ anti3 + anti4 + anti3 ~~ anti4 +'

> fit <- lavaan(model, sample.cov=COV, sample.mean=MEANS, sample.nobs=180, + sample.cov.rescale=FALSE, mimic="EQS")
> summary(fit)

lavaan 0.6-6.1508 ended normally after 40 iterations

| Estimator | ML |
| Optimization method | NLMINB |
| Number of free parameters | 14 |
| Number of observations | 180 |
Model Test User Model:

Test statistic 0.000
Degrees of freedom 0

Parameter Estimates:

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Structured</td>
</tr>
<tr>
<td>Standard errors</td>
<td>Standard</td>
</tr>
</tbody>
</table>

Covariances:

|                  | Estimate | Std.Err | z-value | P(>|z|) |
|------------------|----------|---------|---------|--------|
| anti1 ~ anti2    | 1.390    | 0.284   | 4.903   | 0.000  |
| anti1 ~ anti3    | 1.698    | 0.300   | 5.652   | 0.000  |
| anti1 ~ anti4    | 1.628    | 0.326   | 4.990   | 0.000  |
| anti2 ~ anti3    | 2.781    | 0.389   | 7.155   | 0.000  |
| anti2 ~ anti4    | 2.437    | 0.408   | 5.973   | 0.000  |
| anti3 ~ anti4    | 2.979    | 0.438   | 6.805   | 0.000  |

Intercepts:

|                  | Estimate | Std.Err | z-value | P(>|z|) |
|------------------|----------|---------|---------|--------|
| anti1 (i1)       | 1.750    | 0.128   | 13.688  | 0.000  |
| anti2 (i2)       | 1.928    | 0.154   | 12.502  | 0.000  |
anti3  (i3)  1.978  0.159  12.426  0.000
anti4  (i4)  2.322  0.177  13.122  0.000

Variances:

|     | Estimate | Std.Err | z-value | P(>|z|) |
|-----|----------|---------|---------|---------|
| anti1 | 2.926    | 0.309   | 9.460   | 0.000   |
| anti2 | 4.257    | 0.450   | 9.460   | 0.000   |
| anti3 | 4.536    | 0.479   | 9.460   | 0.000   |
| anti4 | 5.605    | 0.592   | 9.460   | 0.000   |

• note that the means (and the variances!) seem to increase as a function of time

• but we can test this more formally by specifying a model with equal means, and compare the two models:

```r
> model.equal <- ' + # four repeated means + anti1 ~ i1*1; anti2 ~ i1*1; anti3 ~ i1*1; anti4 ~ i1*1 + + # four unequal variances + anti1 ~~ anti1; anti2 ~~ anti2; anti3 ~~ anti3; anti4 ~~ anti4 + + # fully correlated + anti1 ~~ anti2 + anti3 + anti4
```
+ anti2 \sim anti3 + anti4
+ anti3 \sim anti4
+
> fit.equal <- lavaan(model.equal, sample.cov=COV, sample.mean=MEANS,
+ sample.nobs=180, sample.cov.rescale=FALSE, mimic="EQS")
> anova(fit, fit.equal)

Chi-Squared Difference Test


Df  AIC  BIC  Chisq  Chisq diff Df diff Pr(>Chisq)
fit  0 2878.8 2923.5  0.000
fit.equal  3 2884.0 2919.1 11.089 11.089    3  0.01125 *

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
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 (unequal) 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 0.6-6.1508 ended normally after 23 iterations

<table>
<thead>
<tr>
<th>Estimator</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization method</td>
<td>NLMINB</td>
</tr>
<tr>
<td>Number of free parameters</td>
<td>7</td>
</tr>
<tr>
<td>Number of observations</td>
<td>180</td>
</tr>
</tbody>
</table>
Model Test User Model:

Test statistic: 227.209
Degrees of freedom: 7
P-value (Chi-square): 0.000

Parameter Estimates:

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Structured</td>
</tr>
<tr>
<td>Standard errors</td>
<td>Standard</td>
</tr>
</tbody>
</table>

Regressions:

| antil ~ ant2 | Estimate | Std.Err | z-value | P(>|z|) |
|--------------|----------|---------|---------|--------|
| antil        |          |         |         |        |
| ant1         | 0.796    | 0.062   | 12.733  | 0.000  |
| ant2         |          |         |         |        |
| ant3 ~ ant2  | 0.827    | 0.046   | 18.055  | 0.000  |
| ant1         |          |         |         |        |
| ant3 ~ ant3  | 0.896    | 0.053   | 16.986  | 0.000  |
| ant1         |          |         |         |        |

Intercepts:

| antil        | Estimate | Std.Err | z-value | P(>|z|) |
|--------------|----------|---------|---------|--------|
| antil        | 0.000    |         |         |        |
| antil        | 0.000    |         |         |        |
| antil        | 0.000    |         |         |        |
| antil        | 0.000    |         |         |        |
### Variances:

|   | Estimate | Std.Err | z-value | P(>|z|) |
|---|----------|---------|---------|---------|
| antil | 5.988    | 0.633   | 9.460   | 0.000   |
| anti2 | 4.184    | 0.442   | 9.460   | 0.000   |
| anti3 | 2.995    | 0.317   | 9.460   | 0.000   |
| anti4 | 4.210    | 0.445   | 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

```r
> 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
+ ant1 ~<sup>2</sup> dep1; ant2 ~<sup>2</sup> c2<sup>*</sup>dep2; ant3 ~<sup>2</sup> c2<sup>*</sup>dep3; ant4 ~<sup>2</sup> c2<sup>*</sup>dep4
+
> fit <- lavaan(model, sample.cov=COV, sample.mean=MEANS, sample.nobs=180,
+ sample.cov.rescale=FALSE, mimic="EQS")
> summary(fit)

lavaan 0.6-6.1508 ended normally after 38 iterations

<table>
<thead>
<tr>
<th>Estimator</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization method</td>
<td>NLMINB</td>
</tr>
<tr>
<td>Number of free parameters</td>
<td>24</td>
</tr>
<tr>
<td>Number of equality constraints</td>
<td>14</td>
</tr>
<tr>
<td>Row rank of the constraints matrix</td>
<td>14</td>
</tr>
<tr>
<td>Number of observations</td>
<td>180</td>
</tr>
</tbody>
</table>

Model Test User Model:

| Test statistic     | 391.642 |
| Degrees of freedom | 34      |
| P-value (Chi-square)| 0.000   |

Parameter Estimates:

<table>
<thead>
<tr>
<th>Information</th>
<th>Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information saturated (h1) model</td>
<td>Structured</td>
</tr>
</tbody>
</table>
## Standard errors

### Regressions:

| Regression       | Estimate | Std.Err | z-value | P(>|z|) |
|------------------|----------|---------|---------|---------|
| anti2 \~ anti1   | 0.692    | 0.044   | 15.843  | 0.000   |
| anti3 \~ anti2   | 0.692    | 0.044   | 15.843  | 0.000   |
| anti4 \~ anti3   | 0.692    | 0.044   | 15.843  | 0.000   |
| dep2 \~ dep1     | 0.582    | 0.042   | 13.899  | 0.000   |
| dep3 \~ dep2     | 0.582    | 0.042   | 13.899  | 0.000   |
| dep4 \~ dep3     | 0.582    | 0.042   | 13.899  | 0.000   |
| anti2 \~ dep1    | 0.188    | 0.042   | 4.473   | 0.000   |
| anti3 \~ dep2    | 0.188    | 0.042   | 4.473   | 0.000   |
| anti4 \~ dep3    | 0.188    | 0.042   | 4.473   | 0.000   |
| dep2 \~ anti1    | 0.276    | 0.044   | 6.346   | 0.000   |
| dep3 \~ anti2    | 0.276    | 0.044   | 6.346   | 0.000   |
| dep4 \~ anti3    | 0.276    | 0.044   | 6.346   | 0.000   |
### Covariances:

|       | Estimate | Std.Err | z-value | P(>|z|) |
|-------|----------|---------|---------|---------|
| anti1 ˜˜ dep1 | 5.051 | 0.639 | 7.903 | 0.000 |
| anti1 ˜˜ anti2 ˜˜ dep2 (c2) | 1.732 | 0.174 | 9.930 | 0.000 |
| anti1 ˜˜ anti3 ˜˜ dep3 (c2) | 1.732 | 0.174 | 9.930 | 0.000 |
| anti1 ˜˜ anti4 ˜˜ dep4 (c2) | 1.732 | 0.174 | 9.930 | 0.000 |

### Intercepts:

|       | Estimate | Std.Err | z-value | P(>|z|) |
|-------|----------|---------|---------|---------|
| anti2 | 0.000   | 0.000   | 0.000   | 0.000   |
| anti3 | 0.000   | 0.000   | 0.000   | 0.000   |
| anti4 | 0.000   | 0.000   | 0.000   | 0.000   |
| dep2  | 0.000   | 0.000   | 0.000   | 0.000   |
| dep3  | 0.000   | 0.000   | 0.000   | 0.000   |
| dep4  | 0.000   | 0.000   | 0.000   | 0.000   |
| anti1 | 0.000   | 0.000   | 0.000   | 0.000   |
| dep1  | 0.000   | 0.000   | 0.000   | 0.000   |

### Variances:

|       | Estimate | Std.Err | z-value | P(>|z|) |
|-------|----------|---------|---------|---------|
| anti1 | 5.988    | 0.633   | 9.460   | 0.000   |
| anti2 (ra) | 3.665 | 0.224   | 16.386  | 0.000   |
| anti3 (ra) | 3.665 | 0.224   | 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


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:

```R
> 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):

```R
> 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:

```r
> 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

\[ y_t = (\text{initial time at time 1}) + (\text{growth per unit time}) \times \text{time} + \text{error} \]

\[ y_t = \text{intercept} + \text{slope} \times \text{time} + \text{error} \]
real-world example revisited: antisocial behavioral

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

```r
> 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 0.6–6.1508 ended normally after 25 iterations
Estimator: ML
Optimization method: NLMINB
Number of free parameters: 8
Number of observations: 180

Model Test User Model:

Test statistic: 14.810
Degrees of freedom: 6
P-value (Chi-square): 0.022

Model Test Baseline Model:

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
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 - lower 0.032
90 Percent confidence interval - upper 0.150
P-value RMSEA <= 0.05 0.108

Standardized Root Mean Square Residual:

SRMR 0.065

Parameter Estimates:

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

Latent Variables:

| i   | Estimate | Std.Err | z-value | P(>|z|) |
|-----|----------|---------|---------|---------|
| anti1 | 1.000    |         |         |         |
| anti2 | 1.000    |         |         |         |
| anti3 | 1.000    |         |         |         |
| anti4 | 1.000    |         |         |         |
\begin{align*}
\text{s} & = 0.000 \\
\text{anti1} & = 0.000 \\
\text{anti2} & = 1.000 \\
\text{anti3} & = 2.000 \\
\text{anti4} & = 3.000 \\
\end{align*}

|         | 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    |         |         |         |

|         | 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!)
6.4 Alternative models

- autoregressive latent trajectory (ALT) models
- latent change score models
- ...
autoregressive latent trajectory (ALT) models

• Bollen & Curran, in a series of papers (1999, 2000, 2001, 2004) proposed a hybrid model they called the ‘autoregressive latent trajectory’ (ALT) model; best reference:


• it is a growth curve model, combined with an autoregressive structure for the repeated measures

• can be used when there is interest in both continuous underlying trajectories and time-specific influences across constructs

• the authors described this approach as ‘The Best of Both Worlds’

• nevertheless, the approach has been criticized because the interpretation is not always clear; perhaps we should have the autoregressive structure at the level of the residuals errors?
example ALT model

• combination of a growth curve model with a random intercept, a random slope, and an autoregressive structure for the repeated measures
example: antisocial behavior

- we freely estimate the factor loading of both the intercept and slope

```r
> model <- ' 
  + # growth curve
  + i =~ anti1 + 1*anti2 + 1*anti3 + 1*anti4
  + i ~ 1; i ~~ i
  + s =~ anti1 + 1*anti2 + 2*anti3 + 3*anti4
  + s ~ 1; s ~~ s
  +
  + # unequal residual variances
  + anti1 ~~ anti1
  + anti2 ~~ anti2
  + anti3 ~~ anti3
  + anti4 ~~ anti4
  +
  + # autoregressive components
  + anti2 ~ a1*anti1
  + anti3 ~ a2*anti2
  + anti4 ~ a3*anti3
  + '

> fit <- lavaan(model, sample.cov = COV, sample.mean = MEANS,
  + sample.nobs = 180, mimic = "EQS")
> summary(fit, fit.measures = TRUE)

lavaan 0.6-6.1508 ended normally after 61 iterations
```
<table>
<thead>
<tr>
<th>Estimator</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization method</td>
<td>NLMINB</td>
</tr>
<tr>
<td>Number of free parameters</td>
<td>13</td>
</tr>
<tr>
<td>Number of observations</td>
<td>180</td>
</tr>
</tbody>
</table>

**Model Test User Model:**

| Test statistic          | 2.362 |
| Degrees of freedom      | 1     |
| P-value (Chi-square)    | 0.124 |

**Model Test Baseline Model:**

| Test statistic          | 227.618 |
| Degrees of freedom      | 6      |
| P-value                 | 0.000  |

**User Model versus Baseline Model:**

| Comparative Fit Index (CFI) | 0.994 |
| Tucker-Lewis Index (TLI)    | 0.963 |

**Loglikelihood and Information Criteria:**

| Loglikelihood user model (H0) | -1426.590 |
| Loglikelihood unrestricted model (H1) | -1425.402 |
Akaike (AIC) 2879.179
Bayesian (BIC) 2920.688
Sample-size adjusted Bayesian (BIC) 2879.517

Root Mean Square Error of Approximation:

RMSEA 0.087
90 Percent confidence interval – lower 0.000
90 Percent confidence interval – upper 0.238
P-value RMSEA <= 0.05 0.206

Standardized Root Mean Square Residual:

SRMR 0.026

Parameter Estimates:

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

Latent Variables:

Estimate Std.Err  z-value  P(>|z|)
i =~
anti1 0.967 0.116 8.311 0.000
anti2 1.000
anti3 1.000
anti4 1.000
\[ s = \]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>antil</td>
<td>1.050</td>
<td>0.475</td>
<td>2.210</td>
<td>0.027</td>
</tr>
<tr>
<td>antil2</td>
<td>1.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>antil3</td>
<td>2.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>antil4</td>
<td>3.000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Regressions:**

|      | Estimate | Std.Err | z-value | P(>|z|) |
|------|----------|---------|---------|--------|
| antil2 ~ antil1 (a1) | 0.020 | 0.160 | 0.127 | 0.899 |
| antil3 ~ antil2 (a2) | 0.351 | 0.148 | 2.375 | 0.018 |
| antil4 ~ antil3 (a3) | 0.677 | 0.221 | 3.066 | 0.002 |

**Intercepts:**

|      | Estimate | Std.Err | z-value | P(>|z|) |
|------|----------|---------|---------|--------|
| i    | 2.321    | 0.392   | 5.917   | 0.000  |
| s    | -0.471   | 0.184   | -2.566  | 0.010  |
| antil | 0.000    |         |         |        |
| antil2 | 0.000   |         |         |        |
| antil3 | 0.000   |         |         |        |
| antil4 | 0.000   |         |         |        |

**Variances:**

|      | Estimate | Std.Err | z-value | P(>|z|) |
|------|----------|---------|---------|--------|
| i    | 1.825    | 0.700   | 2.609   | 0.009  |
| s    | -0.350   | 0.133   | -2.635  | 0.008  |
• note the negative variance for the slope factor!
Thank you for attending this workshop!