SEM 1: Confirmatory Factor Analysis

Week 2 - Fitting CFA models

Sacha Epskamp

10-04-2018

General factor analysis framework:

$$\begin{aligned} \boldsymbol{y}_i &= \boldsymbol{\Lambda}\boldsymbol{\eta}_i + \boldsymbol{\varepsilon}_i \\ \boldsymbol{y} &\sim N(\mathbf{0}, \boldsymbol{\Sigma}) \\ \boldsymbol{\eta} &\sim N(\mathbf{0}, \boldsymbol{\Psi}) \\ \boldsymbol{\varepsilon} &\sim N(\mathbf{0}, \boldsymbol{\Theta}), \end{aligned}$$

in which:

- $\boldsymbol{y}_i$ is a p-length vector of item responses
- $\boldsymbol{\eta}_i$ an m-length vector of latent variables
- $\boldsymbol{\varepsilon}_i$ an p-length vector of residuals
- $\boldsymbol{\Lambda}$ a $p \times m$ matrix of factor loadings
- $\boldsymbol{\Psi}$ an $m \times m$ symmetric variance–covariance matrix (assume always all latent variables are correlated)
- $\boldsymbol{\Theta}$ is a $p \times p$ symmetric variance–covariance matrix, mostly diagonal (unless you explicitly expect violations of local independence)

The general framework:

$$\begin{aligned} \boldsymbol{y}_i &= \boldsymbol{\Lambda}\boldsymbol{\eta}_i + \boldsymbol{\varepsilon}_i \\ \boldsymbol{y} &\sim N(\mathbf{0}, \boldsymbol{\Sigma}) \\ \boldsymbol{\eta} &\sim N(\mathbf{0}, \boldsymbol{\Psi}) \\ \boldsymbol{\varepsilon} &\sim N(\mathbf{0}, \boldsymbol{\Theta}), \end{aligned}$$

Allows you to derive the model-implied variance–covariance matrix:

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Psi}\boldsymbol{\Lambda}^\top + \boldsymbol{\Theta}$$

Covariance modeling

In general, we aim to estimate parameters that lead to a model implied variance–covariance matrix $\boldsymbol{\Sigma}$ by minimizing (note that the Brown book makes a mistake here):

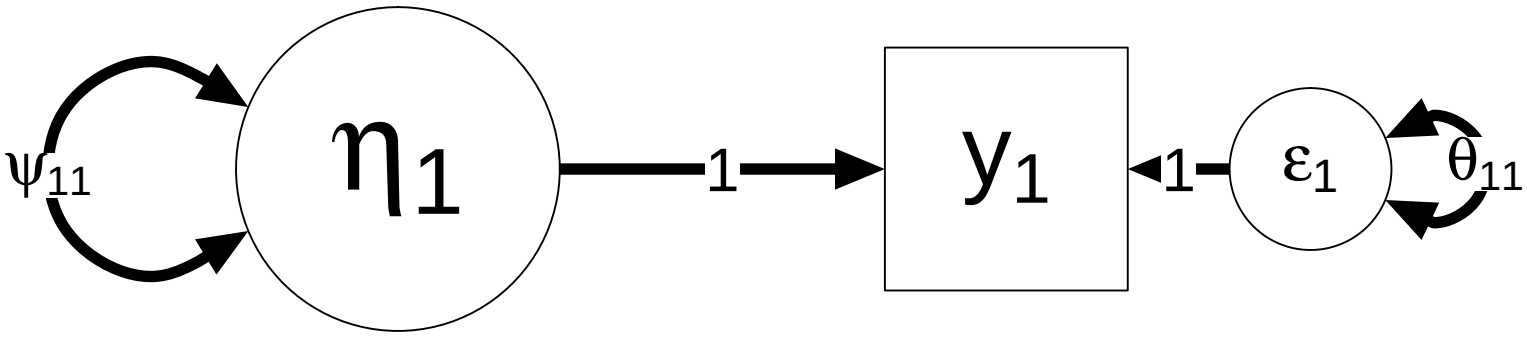
$$F_{\mathrm{ML}} = \mathrm{trace}\left(\boldsymbol{S}\boldsymbol{\Sigma}^{-1}\right) - \ln|\boldsymbol{S}\boldsymbol{\Sigma}^{-1}| - p,$$

in which $\boldsymbol{S}$ is the observed variance–covariance matrix, the trace operator takes the sum of diagonal values, the $|\ldots|$ notation indicates the determinant and p is the number of observed variables. Optimizing this expression is called maximum likelihood estimation. In principle, This expression is optimized if $\boldsymbol{\Sigma}$ resembles $\boldsymbol{S}$ as much as possible! When $\boldsymbol{S} = \boldsymbol{\Sigma}$, $F_{\mathrm{ML}} = 0$.

Identification

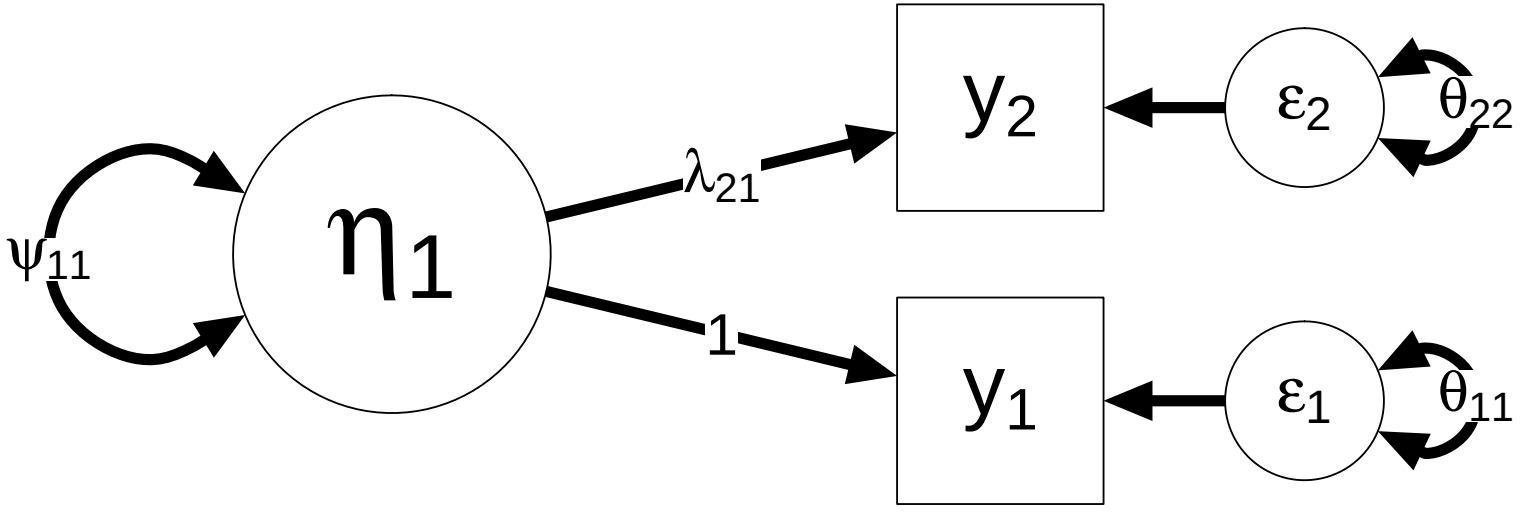
Two main rules:

- Because the unit of the latent variable is unknown, we need to scale the latent variable by fixing its variance to 1 or fixing one (usually the first) factor loading to 1.
- We need at least as many observations (sample variances and covariances) as the number or parameters; we require non-negative degrees of freedom (DF)
  - $\mathrm{DF} = a - b$
  - a: number of observations: $a = p(p+1)/2$ variances and covariances.
  - b: number of parameters we need to estimate (do not count parameters we fixed for scaling)
- In general, we need 3 indicators for a single latent variable model, or 2 per factor for models with multiple (correlated) latent variables.



$$\mathbf{\Lambda} = \left[1\right], \mathbf{\Psi} = \left[\psi_{11}\right], \mathbf{\Theta} = \left[\theta_{11}\right]$$

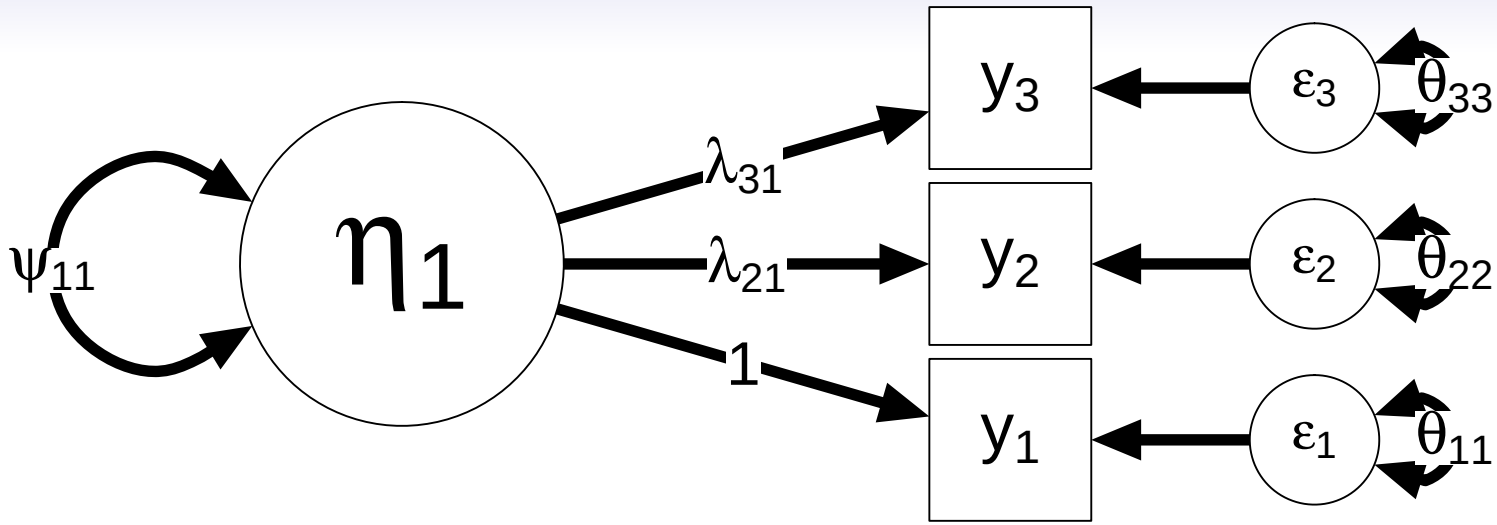
$$\mathbf{\Sigma} = \left[\sigma_{11}\right] = \left[\psi_{11} + \theta_{11}\right]$$



$$\mathbf{\Lambda} = \begin{bmatrix} 1 \\ \lambda_{21} \end{bmatrix}, \mathbf{\Psi} = \begin{bmatrix} \psi_{11} \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \theta_{11} & \\ 0 & \theta_{22} \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{11} & \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} \psi_{11} + \theta_{11} & \\ \psi_{11}\lambda_{21} & \lambda_{21}^2\psi_{11} + \theta_{22} \end{bmatrix}$$

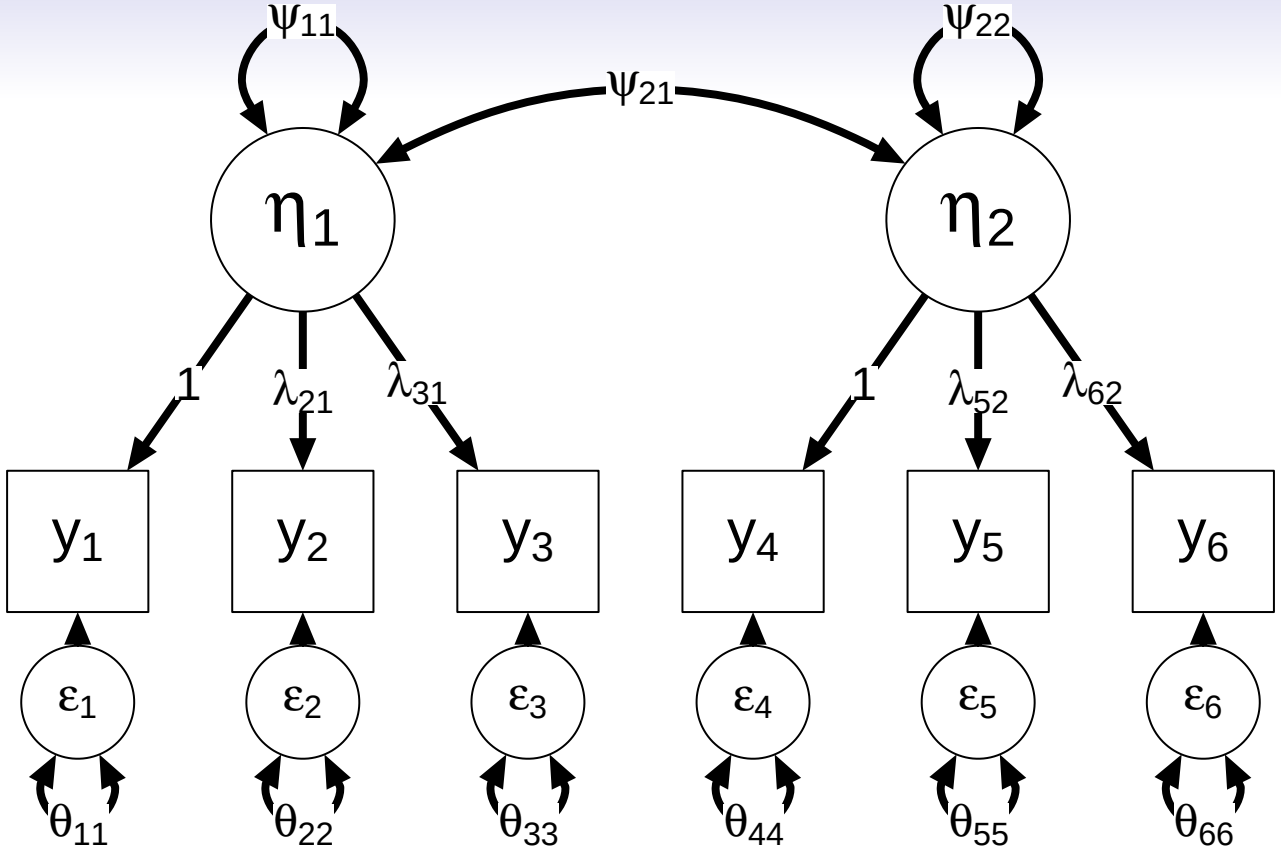
(upper triangular elements in symmetric matrices not shown)



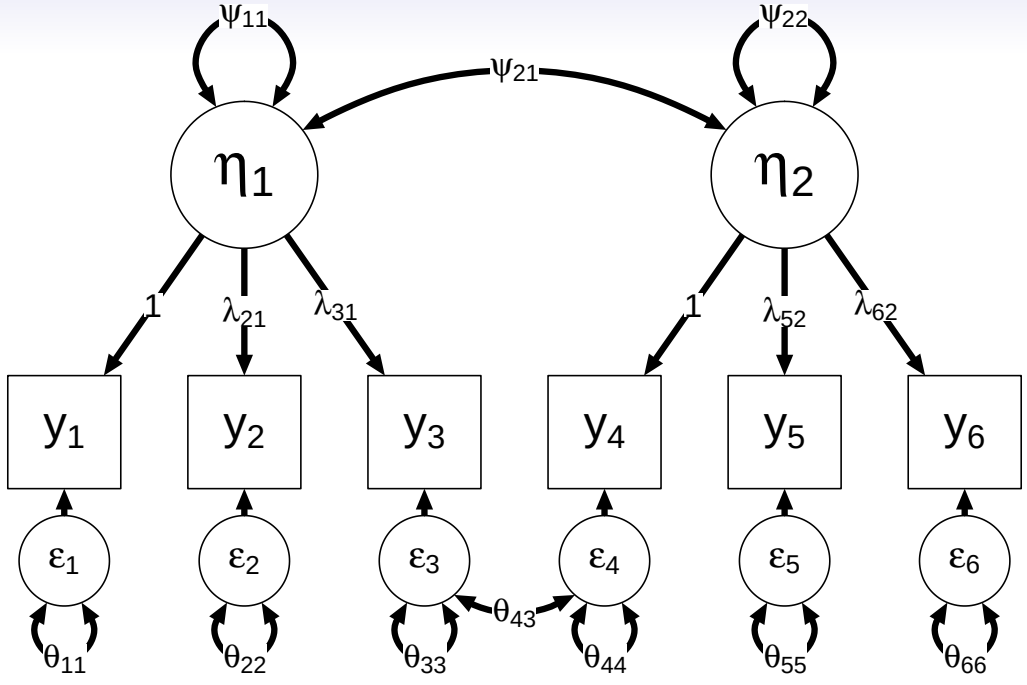
$$\mathbf{\Lambda} = \begin{bmatrix} 1 \\ \lambda_{21} \\ \lambda_{31} \end{bmatrix}, \mathbf{\Psi} = \begin{bmatrix} \psi_{11} \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \theta_{11} & & \\ 0 & \theta_{22} & \\ 0 & 0 & \theta_{33} \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{11} & & \\ \sigma_{21} & \sigma_{22} & \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} = \begin{bmatrix} \psi_{11} + \theta_{11} & & \\ \psi_{11}\lambda_{21} & \lambda_{21}^2\psi_{11} + \theta_{22} & \\ \psi_{11}\lambda_{31} & \psi_{11}\lambda_{21}\lambda_{31} & \lambda_{31}^2\psi_{11} + \theta_{33} \end{bmatrix}$$

$\mathrm{DF} = 0$, just identified (but saturated, will explain the data perfectly)

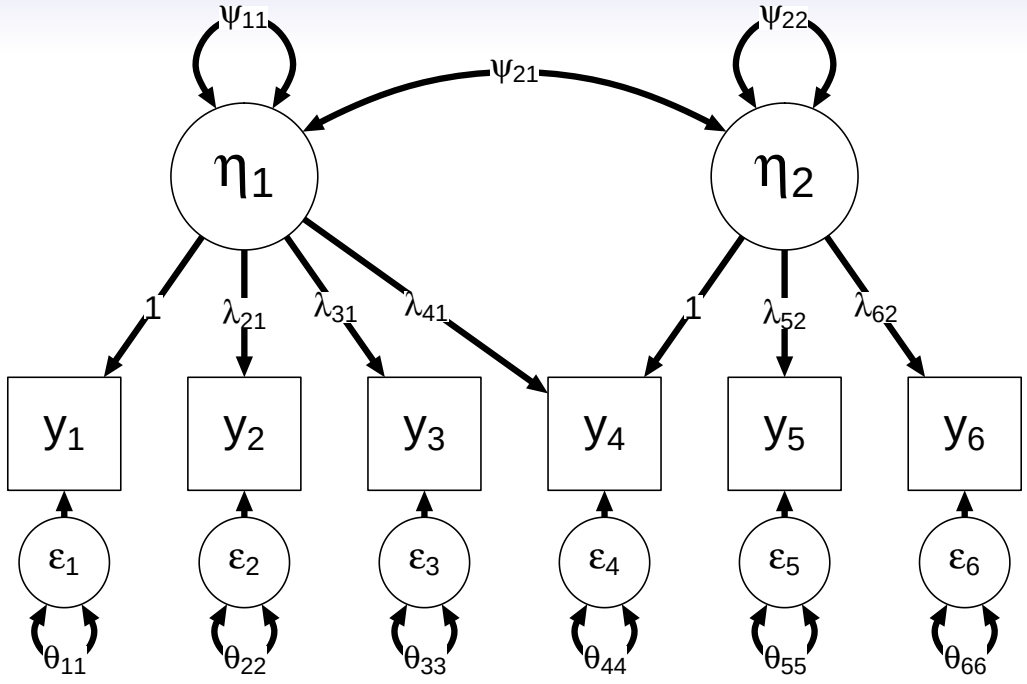


$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ 0 & 1 \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \end{bmatrix}, \mathbf{\Psi} = \begin{bmatrix} \psi_{11} & \\ \psi_{21} & \psi_{22} \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \theta_{11} & & & & & \\ 0 & \theta_{22} & & & & \\ 0 & 0 & \theta_{33} & & & \\ 0 & 0 & 0 & \theta_{44} & & \\ 0 & 0 & 0 & 0 & \theta_{55} & \\ 0 & 0 & 0 & 0 & 0 & \theta_{66} \end{bmatrix}$$



$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ 0 & 1 \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \end{bmatrix}, \mathbf{\Psi} = \begin{bmatrix} \psi_{11} & \\ \psi_{21} & \psi_{22} \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \theta_{11} & & & & & \\ 0 & \theta_{22} & & & & \\ 0 & 0 & \theta_{33} & & & \\ 0 & 0 & \theta_{43} & \theta_{44} & & \\ 0 & 0 & 0 & 0 & \theta_{55} & \\ 0 & 0 & 0 & 0 & 0 & \theta_{66} \end{bmatrix}$$

Residual covariance / correlation



$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 \\ \lambda_{21} & 0 \\ \lambda_{31} & 0 \\ \lambda_{41} & 1 \\ 0 & \lambda_{52} \\ 0 & \lambda_{62} \end{bmatrix}, \mathbf{\Psi} = \begin{bmatrix} \psi_{11} & \\ \psi_{21} & \psi_{22} \end{bmatrix}, \mathbf{\Theta} = \begin{bmatrix} \theta_{11} & & & & & \\ 0 & \theta_{22} & & & & \\ 0 & 0 & \theta_{33} & & & \\ 0 & 0 & 0 & \theta_{44} & & \\ 0 & 0 & 0 & 0 & \theta_{55} & \\ 0 & 0 & 0 & 0 & 0 & \theta_{66} \end{bmatrix}$$

Cross-loading

Lavaan

```r
# Install the package:
install.packages("lavaan")

# Load the package:
library("lavaan")

# Read data into R:
Data <- read.csv("HolzingerSwineford1939.csv")
```

```
# Write a lavaan model:
Model <- '
# Factor loadings:
  visual  =~ 1*x1 + x2 + x3
  textual =~ 1*x4 + x5 + x6
  speed   =~ 1*x7 + x8 + x9
# Variances:
  visual ~~ visual
  textual ~~ textual
  speed ~~ speed
# Covariances:
  visual ~~ textual
  visual ~~ speed
  textual ~~ speed
# Residuals:
  x1 ~~ x1
  x2 ~~ x2
  x3 ~~ x3
  x4 ~~ x4
  x5 ~~ x5
  x6 ~~ x6
  x7 ~~ x7
  x8 ~~ x8
  x9 ~~ x9
'
```

The =~ operator should be read as *is indicated by*

```
visual  =~ 1*x1 + x2 + x3
textual =~ 1*x4 + x5 + x6
speed   =~ 1*x7 + x8 + x9
```

Encodes:

$$\mathbf{\Lambda} = \begin{array}{c} \begin{array}{ccc} \text{visual} & \text{textual} & \text{speed} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ \lambda_{21} & 0 & 0 \\ \lambda_{31} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \lambda_{52} & 0 \\ 0 & \lambda_{62} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & \lambda_{83} \\ 0 & 0 & \lambda_{93} \end{pmatrix} \end{array} \begin{array}{l} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{array}$$

The `~~` operator indicates a *variance* or *covariance*

```
# Variances:
  visual ~~ visual
  textual ~~ textual
  speed ~~ speed
# Covariances:
  visual ~~ textual
  visual ~~ speed
  textual ~~ speed
```

Encodes:

$$\boldsymbol{\Psi} = \begin{array}{c} \begin{array}{ccc} \text{visual} & \text{textual} & \text{speed} \end{array} \\ \begin{pmatrix} \psi_{11} & & \\ \psi_{21} & \psi_{22} & \\ \psi_{31} & \psi_{32} & \psi_{33} \end{pmatrix} \end{array} \begin{array}{c} \text{visual} \\ \text{textual} \\ \text{speed} \end{array}$$

Scaling in latent variance:

```
# Variances:
  visual ~~ 1*visual
  textual ~~ 1*textual
  speed ~~ 1*speed
# Covariances:
  visual ~~ textual
  visual ~~ speed
  textual ~~ speed
```

Encodes:

$$\boldsymbol{\Psi} = \begin{array}{c} \begin{array}{ccc} \text{visual} & \text{textual} & \text{speed} \end{array} \\ \begin{pmatrix} 1 & & \\ \psi_{21} & 1 & \\ \psi_{31} & \psi_{32} & 1 \end{pmatrix} \end{array} \begin{array}{c} \text{visual} \\ \text{textual} \\ \text{speed} \end{array}$$

The `~~` operator indicates a *variance* or *covariance*

```
# Residuals:
  x1 ~~ x1
  x2 ~~ x2
  x3 ~~ x3
  x4 ~~ x4
  x5 ~~ x5
  x6 ~~ x6
  x7 ~~ x7
  x8 ~~ x8
  x9 ~~ x9
```

Encodes diagonal elements of $\boldsymbol{\Theta}$.

```
# Fit in lavaan:
fit <- lavaan(Model, Data)

# Assess fit:
fit
```

```
## lavaan (0.5-23.1097) converged normally after  35 iterations
##
##   Number of observations                           301
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               85.306
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.000
```

```
fitMeasures(fit)
```

```
##                  npar                  fmin                 chisq
##                21.000                 0.142                85.306
##                    df                pvalue        baseline.chisq
##                24.000                 0.000               918.852
##           baseline.df       baseline.pvalue                   cfi
##                36.000                 0.000                 0.931
##                   tli                  nnfi                   rfi
##                 0.896                 0.896                 0.861
##                   nfi                  pnfi                   ifi
##                 0.907                 0.605                 0.931
##                   rni                  logl     unrestricted.logl
##                 0.931             -3737.745             -3695.092
##                   aic                   bic                ntotal
##              7517.490              7595.339               301.000
##                  bic2                 rmsea       rmsea.ci.lower
##              7528.739                 0.092                 0.071
##        rmsea.ci.upper         rmsea.pvalue                   rmr
##                 0.114                 0.001                 0.082
##            rmr_nomean                  srmr          srmr_bentler
##                 0.082                 0.065                 0.065
```

The `cfa()` function makes life easier!

- Automatically fixes first factor loading to 1
- Automatically adds residual variances
- Automatically adds factor (co)variances

```
# lavaan() model:
Model <- '
# Factor loadings:
  visual  =~ 1*x1 + x2 + x3
  textual =~ 1*x4 + x5 + x6
  speed   =~ 1*x7 + x8 + x9
# Variances:
  visual ~~ visual
  textual ~~ textual
  speed ~~ speed
# Covariances:
  visual ~~ textual
  visual ~~ speed
  textual ~~ speed
# Residuals:
  x1 ~~ x1
  x2 ~~ x2
  x3 ~~ x3
  x4 ~~ x4
  x5 ~~ x5
  x6 ~~ x6
  x7 ~~ x7
  x8 ~~ x8
  x9 ~~ x9
'
```

```
# cfa() model:
Model <- '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'
```

```
# Fit in lavaan:
fit <- cfa(Model, Data)

# Assess fit:
fit
```

```
## lavaan (0.5-23.1097) converged normally after  35 iterations
##
##   Number of observations                           301
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               85.306
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.000
```

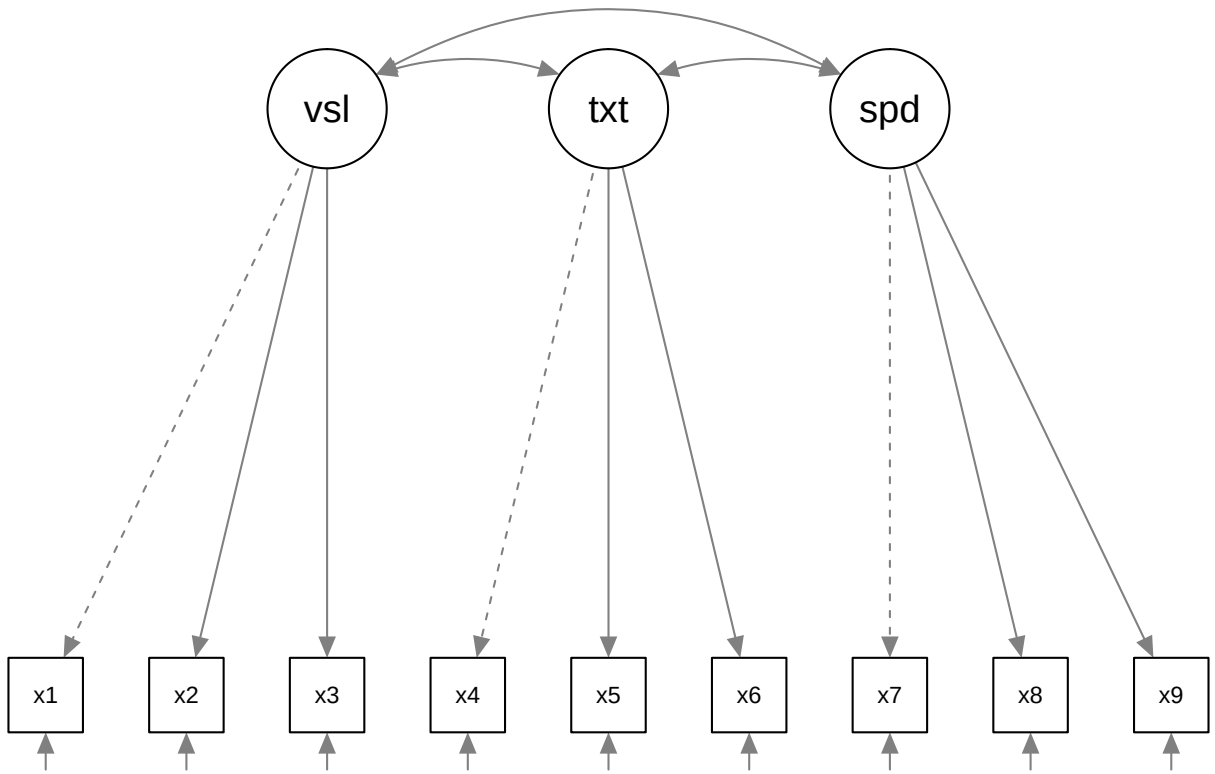
```r
# Or by using a covariance matrix:
covMat <- cov(Data[,c("x1","x2","x3","x4","x5",
                      "x6","x7","x8","x9")])
fit <- cfa(Model,
           sample.cov = covMat,
           sample.nobs = nrow(Data)
)

# Assess fit:
fit
```

```
## lavaan (0.5-23.1097) converged normally after  35 iterations
##
##   Number of observations                           301
##
##   Estimator                                         ML
##   Minimum Function Test Statistic               85.306
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.000
```

```
# Parameter estimates:
parameterEstimates(fit)
```

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

```
library("semPlot")
semPaths(fit, style = "lisrel")
```



Latent variances not drawn and residuals simplified

Jasp

`https://jasp-stats.org/`

- SEM module
- Wrapper around Lavaan
- Just enter a lavaan model, and press command-enter

Ωnyx

Download at: `http://onyx.brandmaier.de/download/`
Manual: `http://onyx.brandmaier.de/userguide.pdf`

Ωnyx

- Free graphical program for SEM modeling
- Fits models and generates lavaan syntax
- Flexible, but takes some time to learn
- Ideal if you do not like R!

Testing for exact fit

Remember the fit function:

$$F_{\mathrm{ML}} = \operatorname{trace}\left(\boldsymbol{S}\boldsymbol{\Sigma}^{-1}\right) - \ln\left|\boldsymbol{S}\boldsymbol{\Sigma}^{-1}\right| - p,$$

Lavaan instead reports `fmin`, which equals $F_{\mathrm{ML}}/2$. If n is the sample size, then we can define:

$$T = nF_{\mathrm{ML}}.$$

If $\operatorname{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}$ (the model is true), then T is χ^2 (chi-square) distributed with the same number of degrees of freedom as the model:

$$T \sim \chi^2(\mathrm{DF}) \iff \operatorname{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}$$

Often (including in the book), T is simply termed χ^2.

```r
# Model matrices:
n <- nrow(Data)
S <- (n-1)/n * cov(Data[,c("x1","x2","x3","x4","x5",
                           "x6","x7","x8","x9")])
Sigma <- lavInspect(fit, "sigma")

# fmin = F_ml / 2:
F_ml <- sum(diag(S %*% solve(Sigma))) -
  log(det(S %*% solve(Sigma))) - ncol(S)
F_ml
```

```
## [1] 0.283407
```

```r
2 * fitMeasures(fit)['fmin']
```

```
##      fmin
## 0.283407
```

```
# Chi-square reported by lavaan and computed:
nrow(Data) * F_ml

## [1] 85.30552

fitMeasures(fit)['chisq']

##    chisq
## 85.30552
```

Testing for exact fit

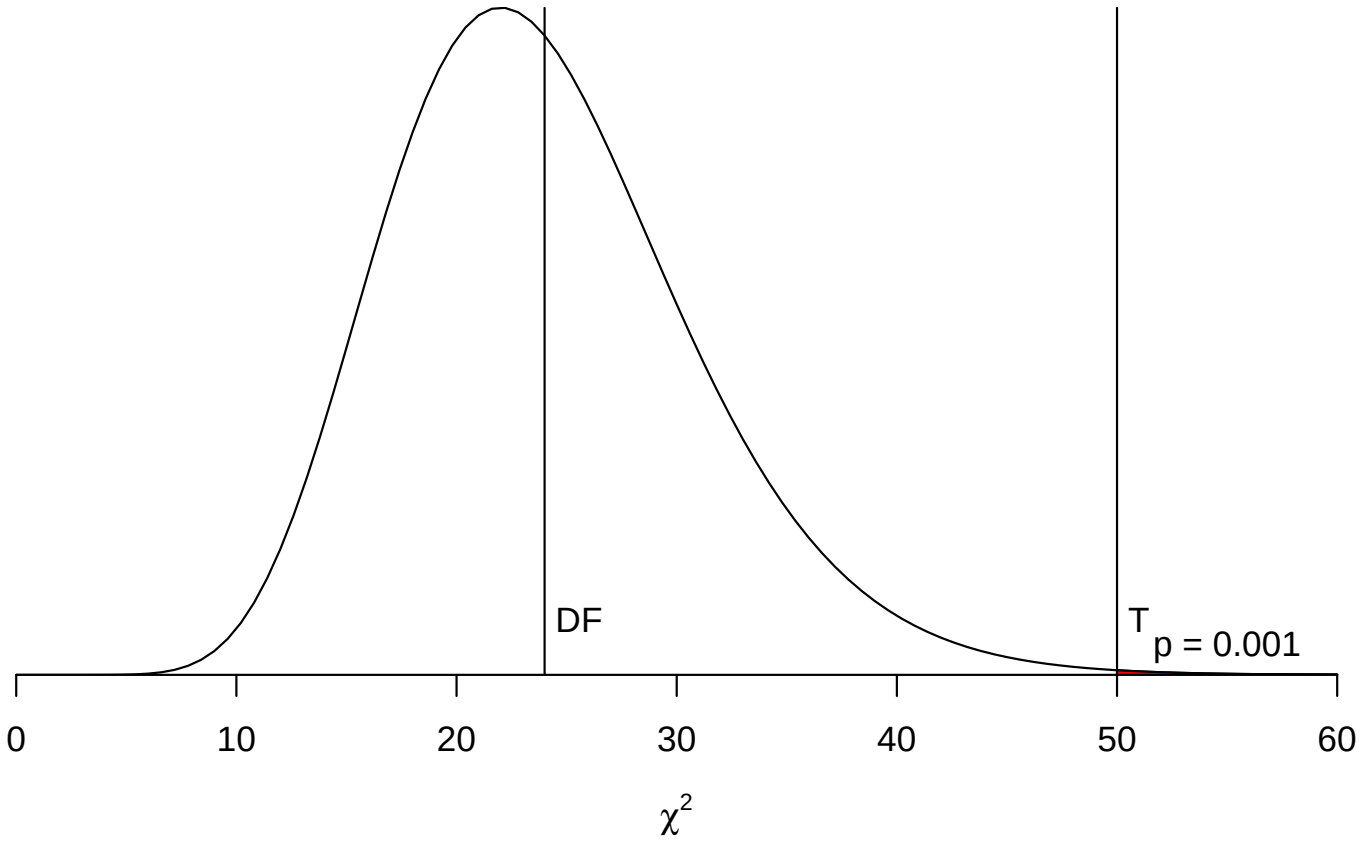
$$T \sim \chi^2(\mathrm{DF}) \iff \mathrm{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}$$

Allows for testing the following hypothesis:

$$H_0 : \mathrm{Var}(\boldsymbol{y}) = \boldsymbol{\Sigma}$$
$$H_1 : \mathrm{Var}(\boldsymbol{y}) \neq \boldsymbol{\Sigma}$$

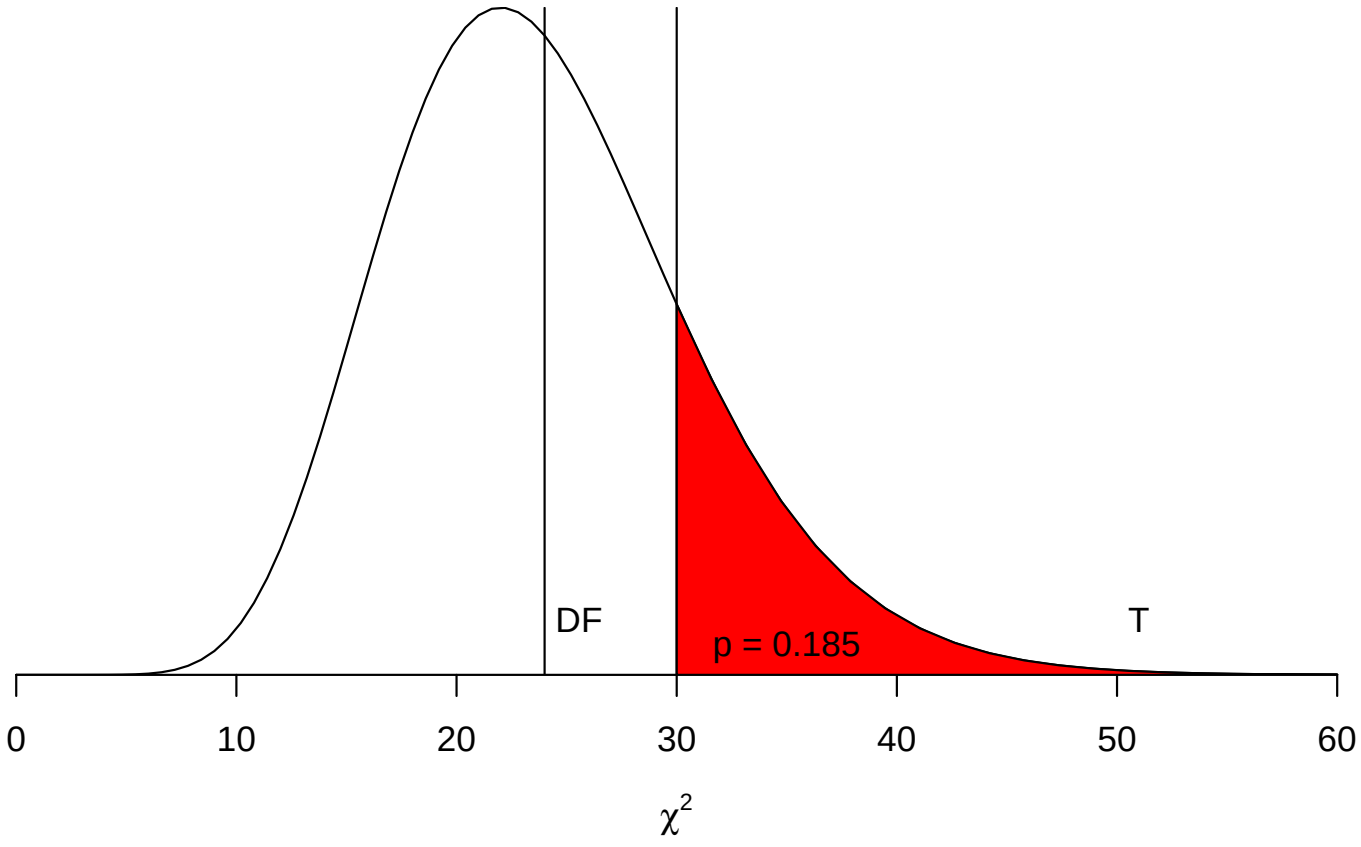
We can reject H_0 if the data is not *likely* under H_0. The $\chi^2(\mathrm{DF})$ distribution computes this likelihood, as an area under the curve right of T. We do not want to reject H_0: p should be above α (typically 0.05).

Degrees of freedom: 24; $T = 50$



$p < 0.05$, model does not fit the data!

Degrees of freedom: 24; $T = 30$



$p > 0.05$, model fits the data!

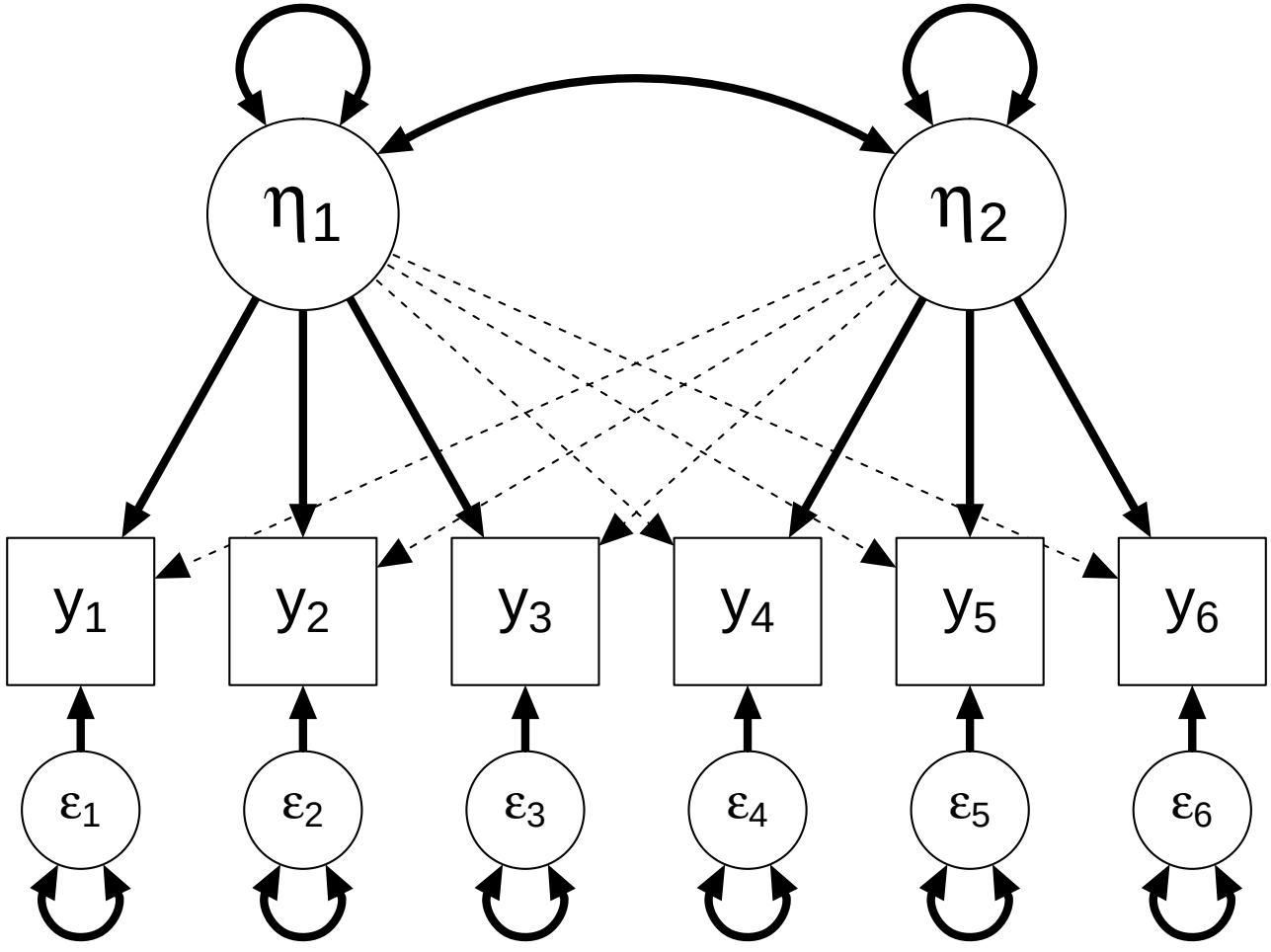
```
fit

## lavaan (0.5-23.1097) converged normally after  35 iterations
##
##   Number of observations                           301
##
##   Estimator                                         ML
##   Minimum Function Test Statistic                85.306
##   Degrees of freedom                                24
##   P-value (Chi-square)                           0.000
```

Model does not fit :(

"All models are wrong but some are useful"

Box, G. E. P. (1979), "Robustness in the strategy of scientific model building", in Launer, R. L.; Wilkinson, G. N., Robustness in Statistics, Academic Press, pp. 201–236.

True model might be not exactly the same, but nearly the same:

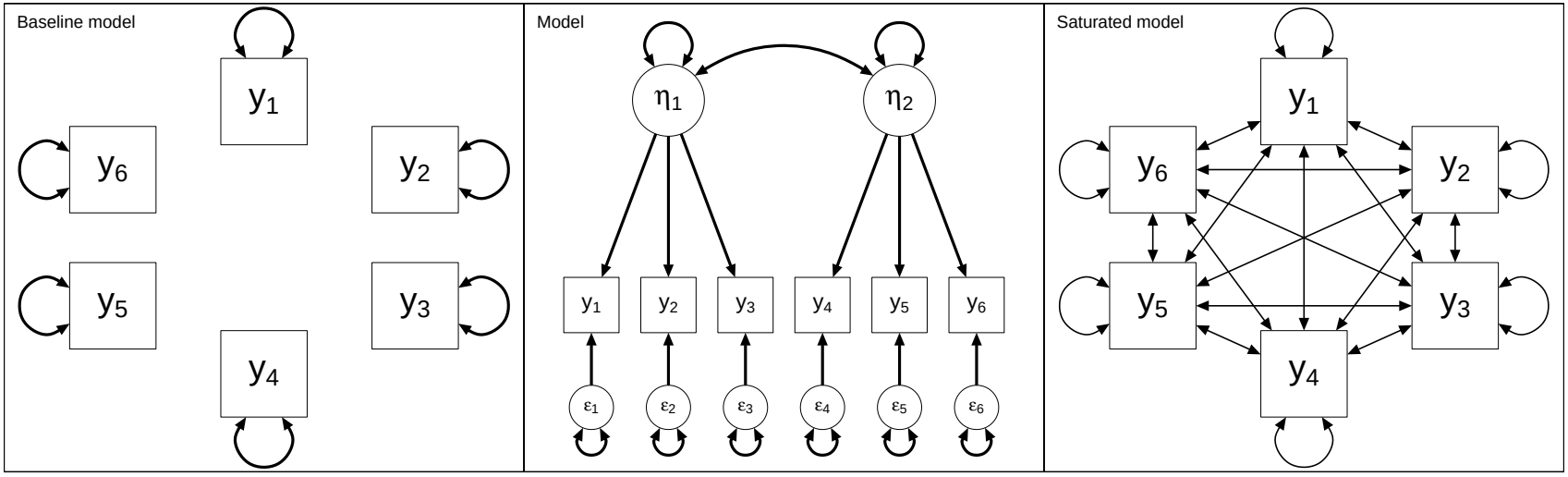


As $N \to \infty$, we will always expect to reject H_0.

Testing for exact fit

- The test of exact fit over-rejects in small samples, because T is only chi-square distributed asymptotically (as n becomes large). When n is small, the chi-square distribution approximation can be poor.
- The chi-square test is often underpowered with small samples, leading to under-rejection. In short, the test of exact fit is unreliable when n is small.
- When n is large, the chi-square test has a lot of power, which leads to rejection of models even when the residuals are very small.

Fit indices



To assess fit, model can be compared to *baseline model* or *saturated model*.

- Baseline model: a model in which no items covary
- Saturated model: A perfectly fitting model ($T = 0$; $DF = 0$)

The χ^2 test compares the model (T_M) to the saturated model (should fit about the same). Many fit indices compare the model to the baseline model instead (T_B; should fit much worse than tested model).

RMSEA

Root Mean Square Error of Approximation

$$RMSEA = \sqrt{\frac{T_M - \mathrm{DF}_M}{(n\mathrm{DF}_M)}}$$

```
Tm <- fitMeasures(fit)[['chisq']]
DFm <- fitMeasures(fit)['df']
sqrt((Tm - DFm)/(n * DFm))

##          df
## 0.09212148

fitMeasures(fit)[['rmsea']]

## [1] 0.09212148
```

RMSEA

RMSEA is a measure of absolute fit (no comparison model). It measures the amount of misfit per degrees of freedom. Smaller values indicate better fit.
Proposed benchmarks from a selection of papers:

- $<$ .05 "very good fit" or "close fit"
- .05 – .08 "good fit" or "fair fit"
- .08 – .1 "mediocre fit" or "good"!
- .05 – .08 "good fit" or "fair fit"
- $>$ .10 "poor or unacceptable"

RMSEA

RMSEA is one of the only fit indices for which the asymptotic sampling distribution is known, so we can make confidence intervals and conduct hypothesis tests about its population value.

- Test of Exact Fit
  - H_0: RMSEA $= 0$ in the population
  - Equivalent to the significance test on the chi-square statistic
- Test of Close Fit (MacCallum et al., 1996)
  - H_0: Null hypothesis: $\mathrm{RMSEA} < \mathrm{RMSEA}_{good}$ in the population.
  - RMSEA_{good} is some acceptable value of RMSEA (in lavaan: 0.05)
- Test of Not-Close Fit (MacCallum et al., 1996)
  - H_0: Null hypothesis: $\mathrm{RMSEA} > \mathrm{RMSEA}_{bad}$ in the population.
  - RMSEA_{bad} is some unacceptable value of RMSEA (e.g., 0.08)

RMSEA test of close fit

Given some boundary RMSEA_{good}, compute non-centrality parameter:

$$\lambda_c = \mathrm{RMSEA}^2_{good} \times n \times \mathrm{DF}_M$$

Compute a one-tailed test with T, now using non-central distribution $\chi_2(\mathrm{DF}_M, \lambda_c)$.

```
lambda_c <- 0.05^2 * n * DFm
pchisq(Tm, DFm, lambda_c, lower.tail = FALSE)

## [1] 0.0006612368

fitMeasures(fit)['rmsea.pvalue']

## rmsea.pvalue
## 0.0006612368
```

Test for close fit is rejected!

RMSEA test of close fit

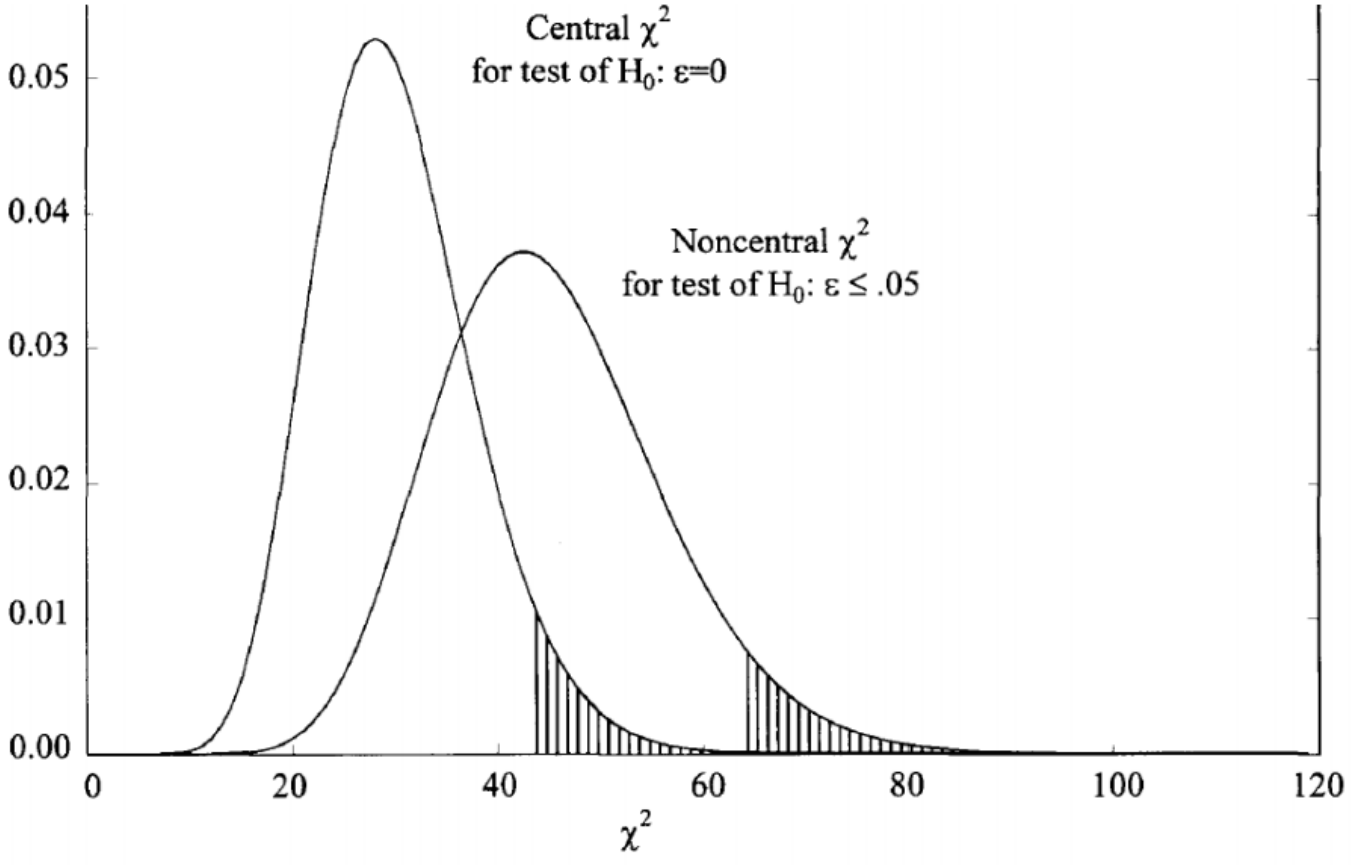


Figure 1. Illustration of difference in critical values between central and noncentral χ^2 distributions.

MacCallum, Browne, & Sugawara (1996)

RMSEA test of not-close fit

Given some boundary RMSEA_{bad}, compute non-centrality parameter:

$$\lambda_c = \mathrm{RMSEA}_{bad}^2 \times n \times \mathrm{DF}_M$$

Compute a one-tailed test with T, using the lower tail of non-central distribution $\chi_2(\mathrm{DF}_M, \lambda_c)$.

```
lambda_c <- 0.08^2 * n * DFm
pchisq(Tm, DFm, lambda_c, lower.tail = TRUE)

## [1] 0.8395529
```

Test for not-close fit can not be rejected! Model does not fit well

RMSEA test of not-close fit

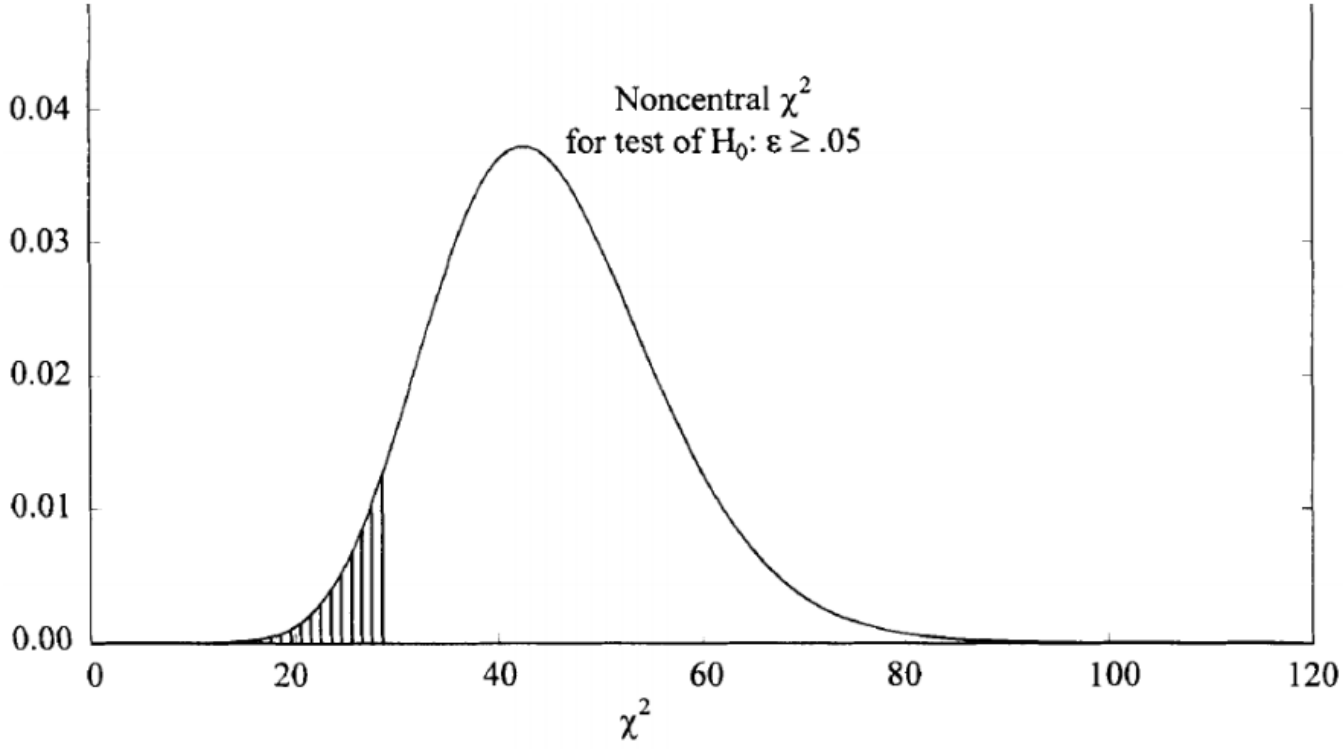
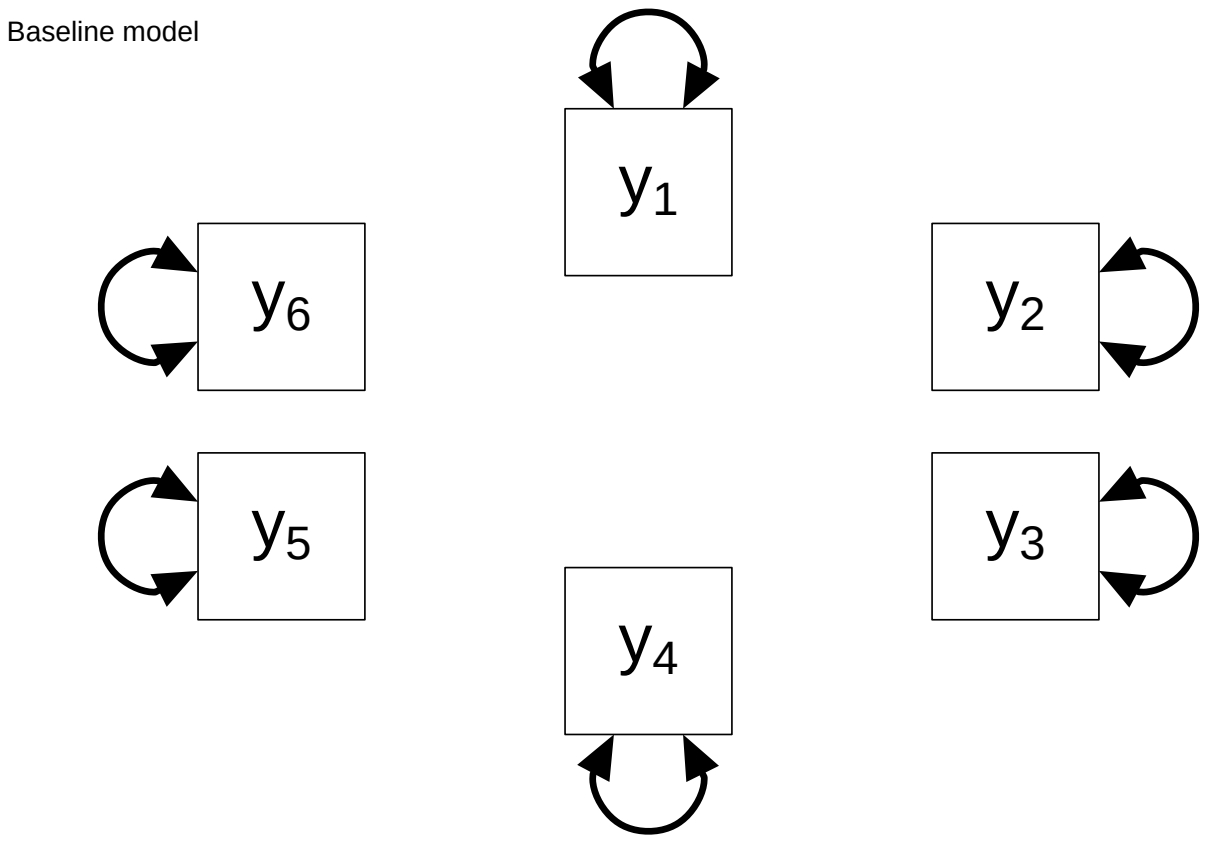


Figure 2. Illustration of critical value of noncentral χ^2 distribution for testing hypothesis of not-close fit.

MacCallum, Browne, & Sugawara (1996)

Incremental fit indices

Incremental fit indices compare the model to the baseline model, which only estimates variances:



Incremental fit indices

```
Tb <- fitMeasures(fit)['baseline.chisq']
Tb

## baseline.chisq
##       918.8516

DFb <- fitMeasures(fit)['baseline.df']
DFb

## baseline.df
##          36
```

How much better does the model fit than the worst possible model?

Incremental fit indices

T_M, df_M: Model test statistic and DF; T_B, df_B: Baseline model test statistic and DF.

$NFI = \frac{T_B - T_M}{T_B}$	% change in the test statistic, ranges from 0-1. Tends to be higher with larger N.	Normed Fit Index, Bentler & Bonnett (1980)
$TLI = \frac{T_B - \frac{df_B}{df_M} T_M}{T_B}$	NFI + reward for parsimonious models. "non-normed" = can take values higher than 1.	Tucker-Lewis Index (TLI), Tucker & Lewis (1973) Non-Normed Fit Index (NNFI), Bentler & Bonnett (1980)
$RFI = \frac{\frac{T_B}{df_B} - \frac{T_M}{df_M}}{T_B / df_B}$	% change in the test statistic relative to its df	Relative Fit Index (Bollen, 1986)
$IFI = \frac{T_B - T_M}{T_B - df_M}$	IFI is less sensitive to sample size than NFI, NNFI, RFI	Incremental Fit Index (Bollen, 1989)

(higher is better)

Incremental fit indices

T_M, df_M: Model test statistic and DF; T_B, df_B: Baseline model test statistic and DF.

$$RNI = \frac{\left(T_B - df_B\right) - \left(T_M - df_M\right)}{T_B - df_B}$$

Nonnormed (can exceed 1)

Relative Noncentrality Index, McDonald & Marsh, 1990

$$CFI = 1 - \frac{T_M - df_M}{T_B - df_B}$$

Constrained to 1 if $df_M > T_M$

Comparative Fit Index, Bentler, 1990

In Practice:
CFI, TLI/NNFI are most commonly reported incremental fit indices. .95 is often used as a cutoff rule-of-thumb for "good fit", and .90 for "acceptable fit" though these cutoffs do not have much empirical support

Goodness of Fit (GFI) and Adjusted GFI

- GFI and AGFI are analogous to R^2 and adjusted R^2 in regression
  - R^2 estimates the proportion of variance in Y that is accounted for by the regression model.
  - GFI estimates the proportion of variance in the sample covariance matrix $\boldsymbol{S}$ that is accounted for by the model structure $\boldsymbol{\Sigma}$
- Both GFI and AGFI can take values from 0 to 1; higher is better.
- Rule of thumb: $>$ .90 is acceptable fit.
- GFI and AGFI tend to be underestimated in small samples.

SRMR

- The largest residual correlation, or list of 5 largest residuals, is very useful for identifying why/how the model does not fit.
- Only look at these if the test of exact fit is significant - if not, the residuals are within the range of sampling error and are most likely noise.
- SRMR is the average of the squared values in the residual correlation matrix. It has been suggested that SRMR should be less than .05 or definitely less than .08. It is less informative than just looking for the biggest residuals!

Residuals

```
residuals(fit)$cov
```

```
##    x1     x2     x3     x4     x5     x6     x7     x8     x9
## x1  0.000
## x2 -0.041  0.000
## x3 -0.010  0.124  0.000
## x4  0.097 -0.017 -0.090  0.000
## x5 -0.014 -0.040 -0.219  0.008  0.000
## x6  0.077  0.038 -0.032 -0.012  0.005  0.000
## x7 -0.177 -0.242 -0.103  0.046 -0.050 -0.017  0.000
## x8 -0.046 -0.062 -0.013 -0.079 -0.047 -0.024  0.082  0.000
## x9  0.175  0.087  0.167  0.056  0.086  0.062 -0.042 -0.032  0.000
```

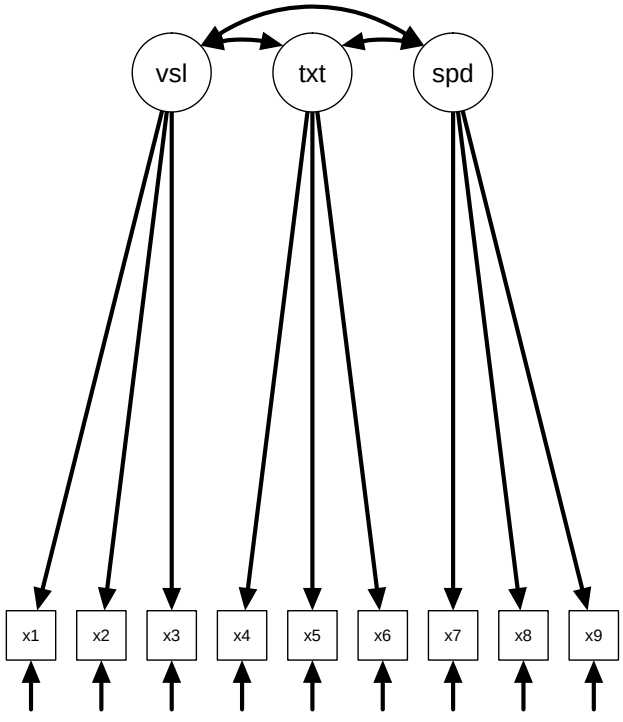
Strategies for Assessing Fit

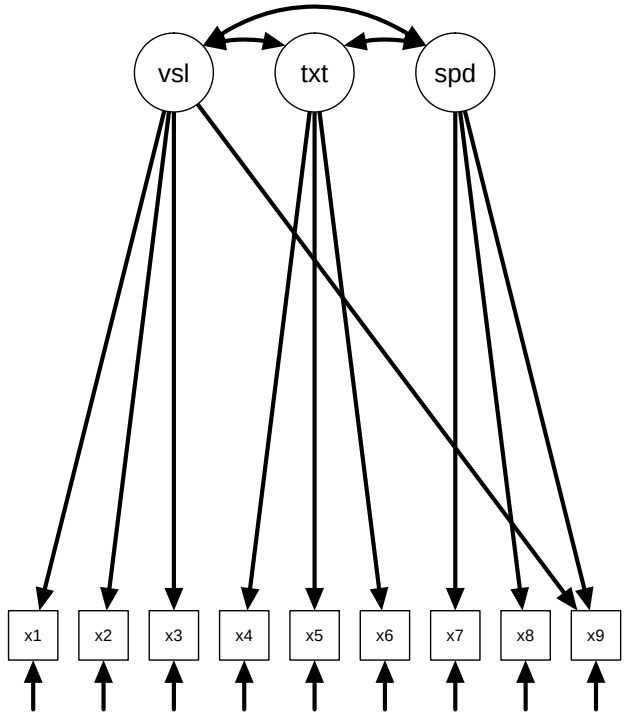
- Always report the chi-square test statistic. You may argue that it is too sensitive to minor misspecifications because you have a large sample size, but report it anyway!
- Report several indices (e.g., RMSEA, CFI, TLI)
- The RMSEA tests of close and not-close fit can be a good index of power (i.e., if neither are significant, you may lack power to detect misspecifications)
- Try to make a holistic judgment based on a set of fit indices
- It goes without saying but... dont cherry pick the indices that make your model look good :)

Another model:

```
Model2 <- '
  visual  =~ x1 + x2 + x3 + x9
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'


# Fit in lavaan:
fit2 <- cfa(Model2, Data)
```

Competing models





A crash course on model selection

Different options:

- Model with a lower Akaike information criterion (AIC) is better
- Model with a lower Bayesian information criterion (BIC) is better
- If models significantly differ, the more complicated model is better
- If models do not significantly differ, the less complicated model is better

Model comparison

Given two models: model A with test statistic T_A and degrees of freedom DF_A, and model B with test statistic T_B and degrees of freedom DF_B. Then we can use the χ^2 distribution to test their difference:

$$T_A - T_B \sim \chi^2(\mathrm{DF}_A - \mathrm{DF}_B)$$

Only if:

- Model B fits the data
- Model A is nested in model B
  - Any $\boldsymbol{\Sigma}$ obtained in model A can be reproduced in model B
  - Often: some parameters in model B can be constrained (e.g., fixed to zero) to obtain model A

This test is called a likelihood ratio test

```
Ta <- fitMeasures(fit)['chisq']
Tb <- fitMeasures(fit2)['chisq']

DFa <- fitMeasures(fit)['df']
DFb <- fitMeasures(fit2)['df']

pchisq(Ta - Tb, DFa - DFb, lower.tail=FALSE)

##        chisq
## 9.586212e-09

# or via lavaan:
anova(fit, fit2)

## Chi Square Difference Test
##
##      Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## fit2 23 7486.6 7568.1 52.382
## fit  24 7517.5 7595.3 85.305     32.923       1  9.586e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

You can always compare information criteria (lower is better), using Akaike's information criterion (AIC):

$$\mathrm{AIC} = -2\ln(L) + 2q$$

or the Bayesian information criterion (BIC):

$$\mathrm{AIC} = -2\ln(L) + q\ln(n)$$

q is the number of parameters and $\ln(L)$ the log-likelihood.

```
anova(fit, fit2)

## Chi Square Difference Test
##
##      Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
## fit2 23 7486.6 7568.1 52.382
## fit  24 7517.5 7595.3 85.305     32.923       1  9.586e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lagrange Multiplier (LM) Tests

- Given a poorly-fitting model, you may want to know what path(s) you could add to make it fit better.
- The LM test gives the expected reduction in the chi-square test statistic that would result if you added particular parameters.
- This is an *exploratory process*. If you make changes based on the LM test results, you must report them as such. Once you've made a post-hoc modification, the p-value is no longer interpretable.

Lagrange Multiplier (LM) Tests

- `modindices(fit)` gives the LM test results for every possible parameter you could add.
- Important: Especially with smaller sample sizes, the LM test results will give parameter suggestions that capitalize on chance. You may be looking at noise.
- `mi` = "modification index" = expected decrease in test statistic
- `epc` = "expected parameter change" = the approximate value that the parameter estimate would take if you added it.

Lagrange Multiplier (LM) Tests

```
mod <- modindices(fit)

library("dplyr")
mod %>% arrange(-mi) %>% head(10)
```

```
##         lhs op rhs        mi        epc    sepc.lv    sepc.all    sep
## 1    visual =~  x9 36.411031  0.5770215  0.5191001  0.51524911  0.515
## 2        x7 ~~  x8 34.145089  0.5364440  0.5364440  0.48784812  0.487
## 3    visual =~  x7 18.630638 -0.4218624 -0.3795158 -0.34890880 -0.348
## 4        x8 ~~  x9 14.946392 -0.4230959 -0.4230959 -0.41541599 -0.415
## 5  textual =~  x3  9.150895 -0.2716376 -0.2688377 -0.23809933 -0.238
## 6        x2 ~~  x7  8.918022 -0.1827254 -0.1827254 -0.14290947 -0.142
## 7  textual =~  x1  8.902732  0.3503311  0.3467201  0.29748838  0.297
## 8        x2 ~~  x3  8.531827  0.2182393  0.2182393  0.16442995  0.164
## 9        x3 ~~  x5  7.858085 -0.1300947 -0.1300947 -0.08943375 -0.089
## 10   visual =~  x5  7.440646 -0.2098980 -0.1888284 -0.14656875 -0.146
```

- Fitting CFA models
  - lavaan (R), Onyx and Jasp
- Testing for exact fit
  - χ^2 test
- Assessing close fit
  - RMSEA (below 0.05 to 0.08)
  - SRMR (below 0.05)
  - CFI, RNI, NFI, TLI, RFI, IFI (above 0.90 to 0.95)
  - (A) GFI (above 0.90)
- Model comparison
  - Likelihood ratio test
  - Information criteria
  - Modefication indices

Thank you for your attention!