Confirmatory Factor Analysis II

Introduction to Structural Equation Modeling

Utrecht University

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

Outline

Model Estimation

Model-Implied Statistics

Most statistical estimation algorithms operate by minimizing the difference between two key reference points:

1. The *model-implied* statistics/predictions/fitted values
   - The sufficient statistics implied by the structure of your model.
   - Predicted/fitted values produced by your model.
2. The *observed* statistics/values
   - The sufficient statistics calculated from the observed data.
   - The raw outcome values from your dataset.

The predictions/implied statistics produced by a good model must be simpler than the analogous quantities in the observed data.

- A model that exactly replicates the observed data is overfitting.
- The inferences from such models won't generalize to the population.

Model-Implied Statistics

You should already be familiar with this idea from OLS regression.

- The fitted values are the model implied statistics.

$$\hat{Y}_n = \hat{\beta}_0 + \sum_{p=1}^{P} \hat{\beta}_p X_{n,p}$$

- The raw outcome variable, Y, contains the observed values.
- Minimize the difference between $\hat{Y}$ and Y to estimate the model.

$$RSS = \sum_{n=1}^{N} \left(Y_n - \hat{Y}_n \right)^2$$

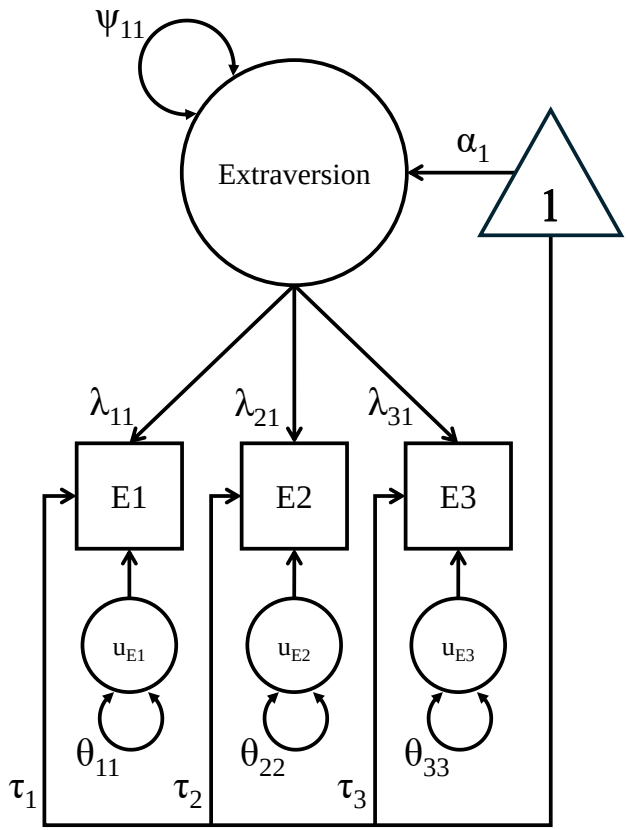
Hypothesized Model

We'll use this toy model to sketch out the intuition for maximum likelihood estimation of CFA models.

$$\alpha = \begin{bmatrix} \alpha_1 \end{bmatrix} \quad \Psi = \begin{bmatrix} \psi_{11} \end{bmatrix}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \quad \Lambda = \begin{bmatrix} \lambda_{11} \\ \lambda_{21} \\ \lambda_{31} \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \theta_{11} & & \\ 0 & \theta_{22} & \\ 0 & 0 & \theta_{33} \end{bmatrix}$$



Parameters → Implied Statistics

The parameter matrices define the model-implied mean vector, $\hat{\mu}$, and covariance matrix, $\hat{\Sigma}$, via the following formulas.

$$\Sigma = \Lambda \Psi \Lambda^T + \Theta$$

$$\mu = \tau + \Lambda \alpha$$

By expanding these formulas, we can see how the model reproduces each mean, variance, and covariance.

$$\hat{\mu} = \begin{bmatrix} \tau_1 + \lambda_{11}\alpha_1 \\ \tau_2 + \lambda_{22}\alpha_1 \\ \tau_3 + \lambda_{33}\alpha_1 \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} \lambda_{11}\psi_{11}\lambda_{11} + \theta_{11} & & \\ \lambda_{11}\psi_{11}\lambda_{21} & \lambda_{21}\psi_{11}\lambda_{21} + \theta_{22} & \\ \lambda_{11}\psi_{11}\lambda_{31} & \lambda_{21}\psi_{11}\lambda_{31} & \lambda_{31}\psi_{11}\lambda_{31} + \theta_{33} \end{bmatrix}$$

Model-Implied Statistics for Estimation

The estimating algorithm chooses values for α, τ, Ψ, and Λ that minimize the differences between the model-implied statistics, $\{\hat{\mu},\ \hat{\Sigma}\}$, and the sufficient statistics calculated from the observed data, $\{\bar{\mathbf{Y}},\ \mathbf{S}\}$.

$$\hat{\mu} = \begin{bmatrix} \tau_1 + \lambda_{11}\alpha_1 \\ \tau_2 + \lambda_{22}\alpha_1 \\ \tau_3 + \lambda_{33}\alpha_1 \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} \lambda_{11}\psi_{11}\lambda_{11} + \theta_{11} & & \\ \lambda_{11}\psi_{11}\lambda_{21} & \lambda_{21}\psi_{11}\lambda_{21} + \theta_{22} & \\ \lambda_{11}\psi_{11}\lambda_{31} & \lambda_{21}\psi_{11}\lambda_{31} & \lambda_{31}\psi_{11}\lambda_{31} + \theta_{33} \end{bmatrix}$$

$$\bar{\mathbf{Y}} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \bar{y}_3 \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} var(y_1) & & \\ cov(y_2, y_1) & var(y_2) & \\ cov(y_3, y_1) & cov(y_3, y_2) & var(y_3) \end{bmatrix}$$

Optimization: Intuition

We can formulate the familiar OLS objective as an abstract optimization problem at follows.

$$f(\beta_0, \beta_1, \ldots, \beta_P) = \underset{\beta_0,\beta_1,\ldots,\beta_P}{\arg\min} \left\| Y - \hat{Y} \right\|$$

Applying the same idea to our CFA optimization problem, we get the following formulation.

$$f(\Lambda, \Psi) = \underset{\Lambda,\Psi}{\arg\min} \left\| \text{Cov}(\mathbf{Y}) - \hat{\Sigma} \right\|$$

$$f(\tau, \alpha) = \underset{\tau,\alpha}{\arg\min} \left\| \bar{\mathbf{Y}} - \hat{\mu} \right\|$$

Optimization: Objective Function

In OLS regression, we minimize the residual sum of squares (RSS).

$$RSS = \sum_{n=1}^{N} \left(Y_n - \hat{Y}_n \right)^2$$

In CFA, we can estimate the model by minimizing the *model deviance*.

$$D_M = -2 \left[\ell \left(\hat{\Sigma}, \hat{\mu} \right) - \ell \left(\mathbf{S}, \bar{\mathbf{Y}} \right) \right]$$

Where, $\ell(\cdot)$ denotes the *loglikelihood* function.

- $\ell \left(\hat{\Sigma}, \hat{\mu} \right)$ returns the loglikelihood value for our estimated model.
- $\ell \left(\mathbf{S}, \bar{\mathbf{Y}} \right)$ returns the loglikelihood value for the *saturated model*.

Optimization: Loglikelihood Functions

The loglikelihood function is a model-based distance metric.

- Derived from the assumed data distribution (i.e., multivariate normal)
- Quantifies the ”chances” of your estimated model having generated the observed data

$$\ell\left(\hat{\Sigma}, \hat{\mu}\right) = -\frac{K}{2}\ln(2\pi) - \frac{1}{2}\ln|\hat{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}\left(Y_n - \hat{\mu}\right)^T \hat{\Sigma}^{-1}\left(Y_n - \hat{\mu}\right)$$

Say we fit two models, M_a and M_b, to the same dataset. If $\ell\left(M_a\right)$ is closer to zero, we know that M_a represents the data better than M_b does.

- M_a’s implied statistics are closer to the observed sufficient statistics.
- M_a’s parameters are more plausible, given the data.

Model Fit

Model Fit: Intuition

We want a plausible, parsimonious, and useful model.

1. Our model should make theoretical sense.
2. Our model's implied statistics should closely align with the observed sufficient statistics.
3. Our model should be no more complex than necessary.

We can use *model fit statistics* to evaluate the second and third criteria.

- Model fit statistics transform $\hat{D}_M$ into interpretable, comparable, scale-free distance measures.
- These indices give us practical tools for evaluating how well our model represents the data.

Different Ways to Quantify Fit

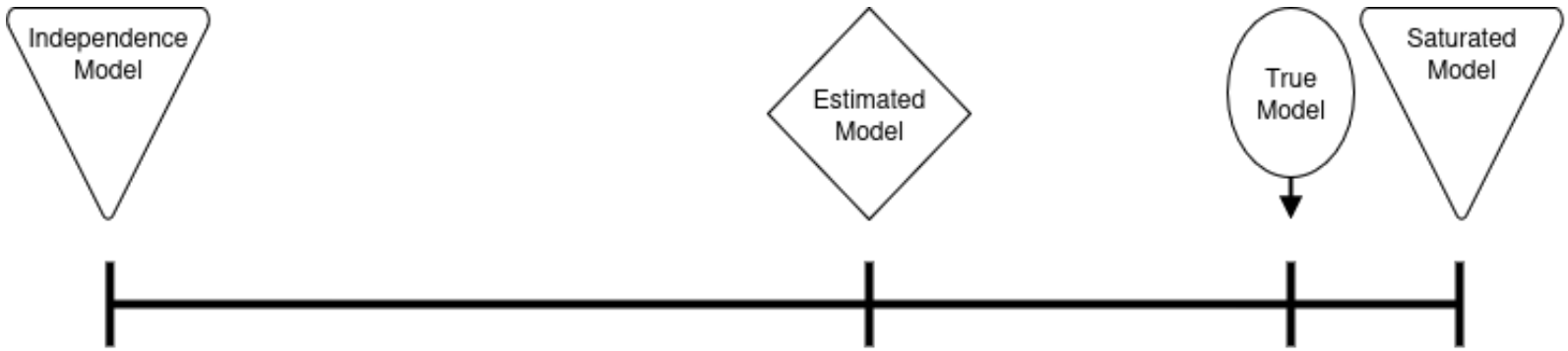
When evaluating model fit, we need to consider four types of model.

1. Our estimated model.
2. The *independence model*
   - The worst possible model
   - All item associations fixed to zero
3. The *saturated model*
   - The ”best” possible model
   - Estimates all possible associations
   - Perfectly reproduces the observed sufficient statistics
4. The *true model*
   - The unknown population model that generates the data
   - Typically simpler than the saturated model

Model Continuum

We can use the *model continuum* diagrammed below to organize the four classes of possible model.

- The left side of the continuum represents bad model fit.
- The right side of the continuum represents good model fit.



Independence Model

```
indMod <- '
A1 ~~ A1
A2 ~~ A2
A3 ~~ A3
O1 ~~ O1
O2 ~~ O2
O3 ~~ O3
'

indOut <- cfa(indMod, data = bfi)
```

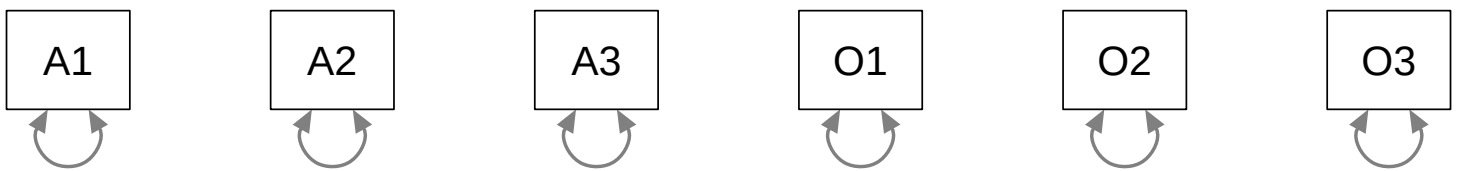
Independence Model

```
partSummary(indOut, 1:4)

lavaan 0.6-19 ended normally after 20 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                         6

                                                  Used       Total
  Number of observations                          2696        2800

Model Test User Model:

  Test statistic                              2053.922
  Degrees of freedom                            15
  P-value (Chi-square)                           0.000
```

Independence Model

```
lavInspect(indOut, "sigma")
```

```
      A1    A2    A3    O1    O2    O3
A1 1.984
A2 0.000 1.377
A3 0.000 0.000 1.708
O1 0.000 0.000 0.000 1.271
O2 0.000 0.000 0.000 0.000 2.431
O3 0.000 0.000 0.000 0.000 0.000 1.485
```



Saturated Model

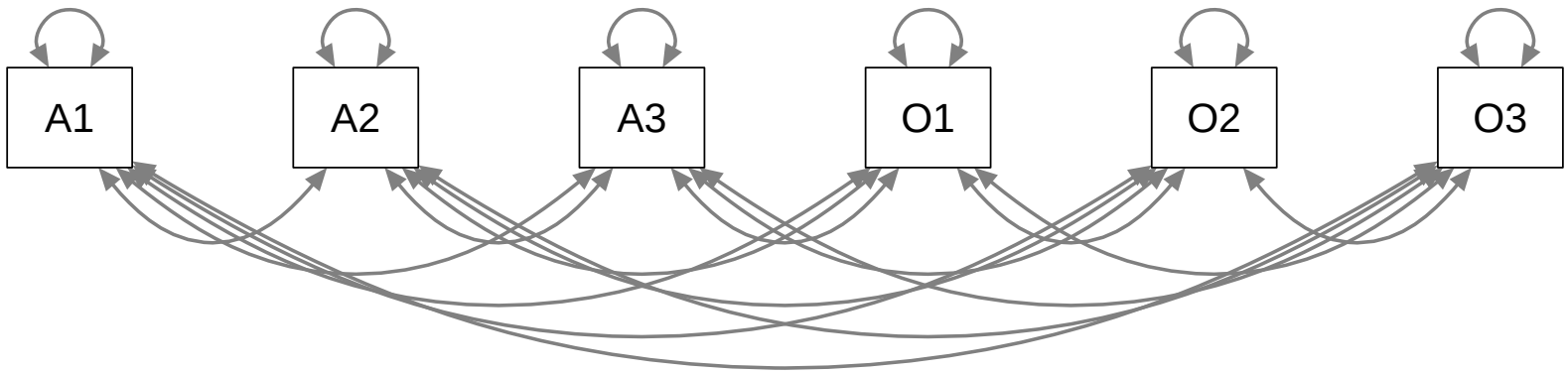
```
satMod <- '
A1 ~~ A1
A2 ~~ A1 + A2
A3 ~~ A1 + A2 + A3
O1 ~~ A1 + A2 + A3 + O1
O2 ~~ A1 + A2 + A3 + O1 + O2
O3 ~~ A1 + A2 + A3 + O1 + O2 + O3
'

satOut <- cfa(satMod, data = bfi)
```

Saturated Model

```
partSummary(satOut, 1:4)

lavaan 0.6-19 ended normally after 36 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        21

                                                  Used       Total
  Number of observations                          2696        2800

Model Test User Model:

  Test statistic                                 0.000
  Degrees of freedom                                 0
```

Saturated Model

```
lavInspect(satOut, "sigma")

       A1     A2     A3     O1     O2     O3
A1  1.984
A2 -0.567  1.377
A3 -0.493  0.751  1.708
O1  0.013  0.174  0.213  1.271
O2  0.172  0.024 -0.004 -0.382  2.431
O3 -0.098  0.234  0.357  0.544 -0.514  1.485
```



Saturated Model

The saturated model reproduces the observed sufficient statistics.

- $\hat{\Sigma} = \mathbf{S}$

```
lavInspect(satOut, "sigma") - lavInspect(satOut, "sampstat")$cov

   A1 A2 A3 O1 O2 O3
A1  0
A2  0  0
A3  0  0  0
O1  0  0  0  0
O2  0  0  0  0  0
O3  0  0  0  0  0  0
```

Measurement Structure

We don't estimate any measurement structure in the independence model or in the saturated model.

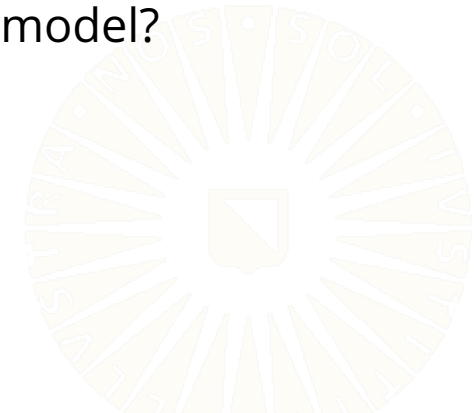
```
lavInspect(indOut, "est")$lambda


A1
A2
A3
O1
O2
O3

lavInspect(indOut, "est")$psi

<0 x 0 matrix>
```

```
lavInspect(satOut, "est")$lambda


A1
A2
A3
O1
O2
O3

lavInspect(satOut, "est")$psi

<0 x 0 matrix>
```

Different Ways to Quantify Fit

For now, we'll conceptualize model fit as comparing different points on the model continuum introduced above.

1. *Absolute Fit*
   - How well does the model reproduce the observed sufficient statistics?
   - How close is our model to some ideal model?
   - Compare our model to the *saturated model*.
2. *Relative/Incremental/Comparative Fit*
   - How much does our model outperform some baseline model?
   - How much improvement do we achieve over some bad model?
   - Compare our model to the *independence model*.



χ^2 Statistic

The χ^2 statistic is the most fundamental model fit statistic.

- Absolute fit statistic
- Derived directly from the optimized value of the objective function
- Allows rigorous hypothesis tests
- Basis for most other fit indices

$$\chi^2 = D_M$$
$$\chi^2 = N \times F_{ML}$$

This statistic follows a χ^2 distribution with $df = Q - P$ (i.e., the same DF as the estimated model).

Example Model

```
library(lavaan)
library(psych)

mod1 <- '
agree =~ A1 + A2 + A3
open  =~ O1 + O2 + O3
'

out1 <- cfa(mod1, data = bfi)
```

Example Model

```
partSummary(out1, 1:4)

lavaan 0.6-19 ended normally after 40 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        13

                                                  Used       Total
  Number of observations                          2696        2800

Model Test User Model:

  Test statistic                               120.283
  Degrees of freedom                             8
  P-value (Chi-square)                         0.000
```

Problems with χ^2

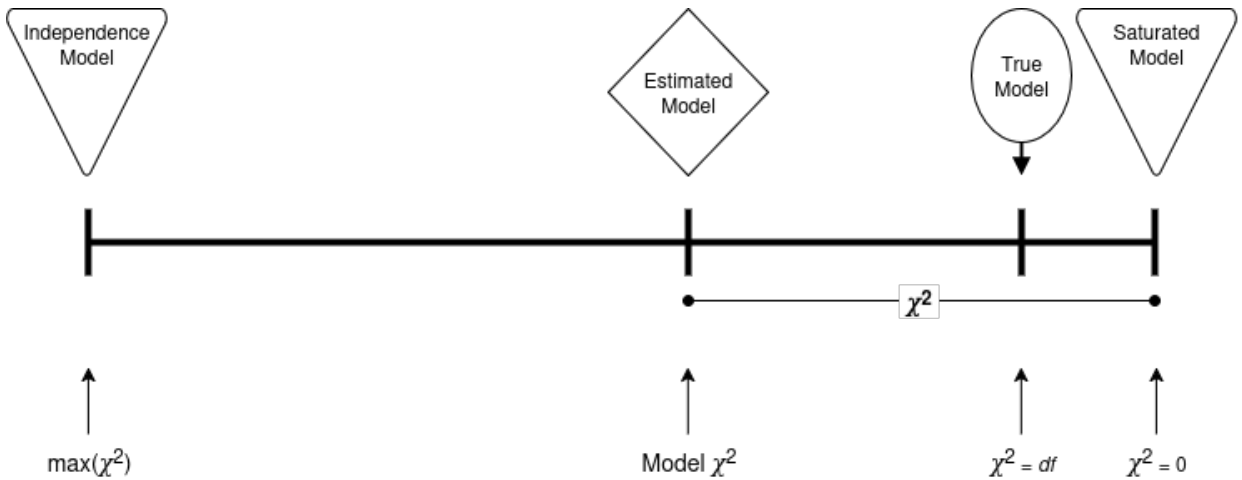
The χ^2 statistic test for *exact fit*:

$$H_0 : \hat{\Sigma} = \mathbf{S}$$

$$H_1 : \hat{\Sigma} \neq \mathbf{S}$$

This hypothesis is perfectly valid, but it isn't a very interesting or useful for real-world data analysis.

- The χ^2 will almost always reject the null hypothesis.
- With a large enough sample, the null hypothesis must be rejected for any model simpler than the exact data generating model.



Approximate Fit Indices

In practice, we won't use the χ^2 test to directly evaluate model fit. We'll use *approximate fit indices* (mostly) derived from the χ^2.

- We have many (maybe too many) options.
- **lavaan** 0.6-19 provides 28 approximate fit measures for a basic CFA.

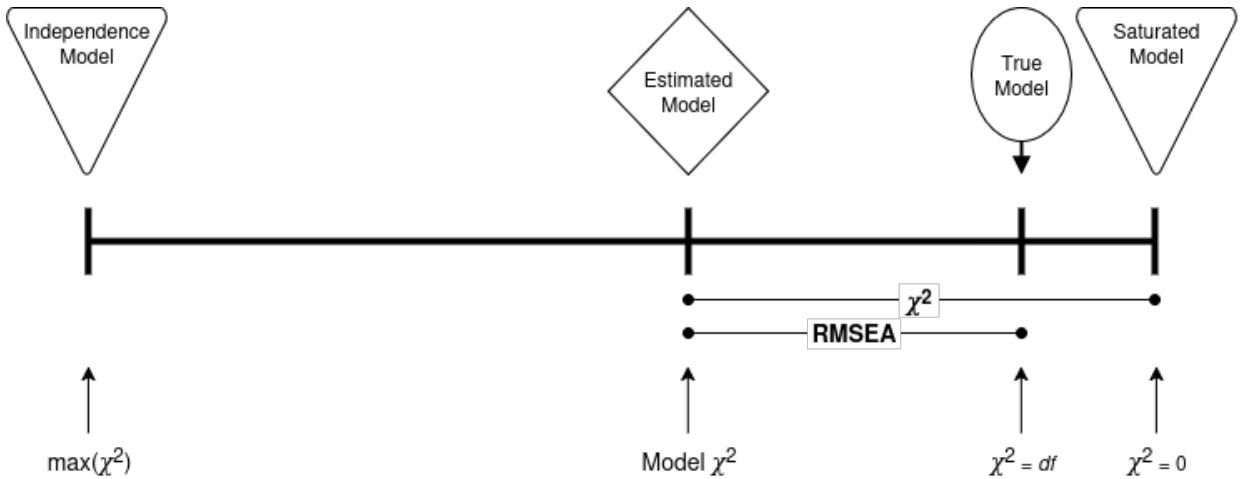
We'll focus on three complementary and well-balanced indices.

- Root Mean Squared Error of Approximation (RMSEA)
- Comparative Fit Index (CFI)
- Standardized Root Mean Squared Residual (SRMR)

RMSEA

The RMSEA is a non-normed *absolute fit index* that evaluates deviations from *close fit*.

$$\text{RMSEA} = \sqrt{\frac{\max\left(\chi^2 - df,\ 0\right)}{N \times df}}$$



RMSEA

The RMSEA must be non-negative, but has no upper bound.

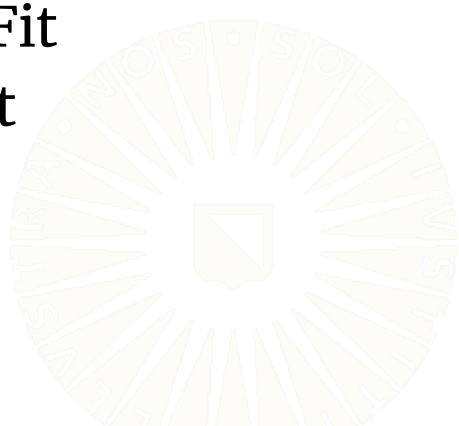
- $\text{RMSEA} \in [0, +\infty)$

The RMSEA will be positive whenever $\chi^2 > df$.

- $\text{RMSEA} = 0$ *iif* $\chi^2 \leq df$

The RMSEA measures deviations from an ideal model, so smaller values represent better fit.

$$\text{RMSEA} \in \begin{cases} [0,\ 0.06] & \Rightarrow \text{Good Fit} \\ (0.06,\ 0.08] & \Rightarrow \text{Acceptable Fit} \\ (0.08,\ 0.1] & \Rightarrow \text{Marginal Fit} \\ (0.1,\ +\infty) & \Rightarrow \text{Bad Fit} \end{cases}$$

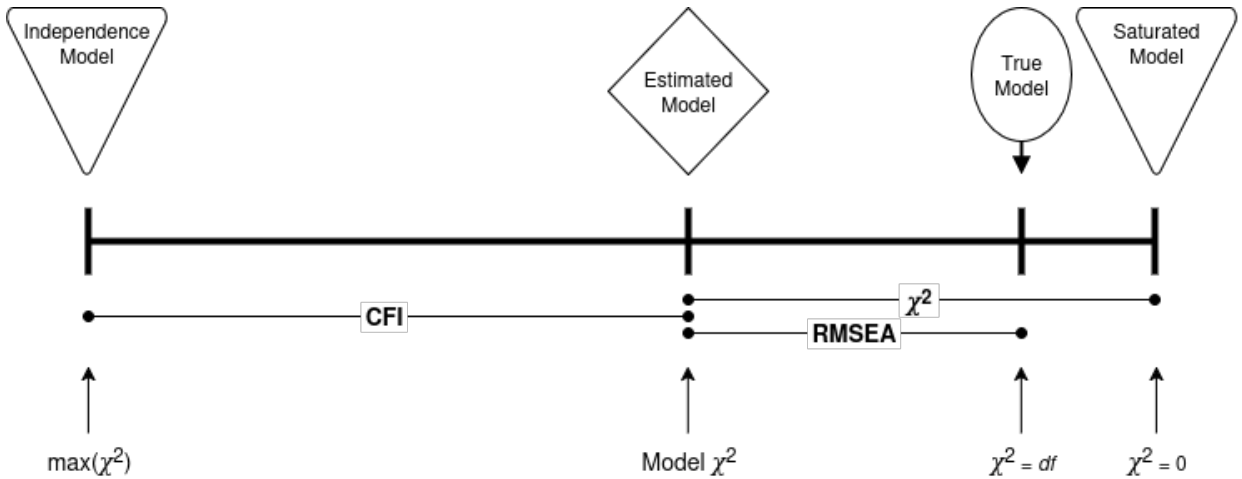


CFI

The CFI is a normed *incremental fit index*.

$$\text{CFI} = 1 - \frac{\max\left(\chi^2 - df,\ 0\right)}{\max\left(\chi_0^2 - df_0,\ \chi^2 - df,\ 0\right)} \overset{\substack{typical\\cases}}{\rightarrow} 1 - \frac{\chi^2 - df}{\chi_0^2 - df_0}$$

We can use the simplified formula on the right in the ordinary situations where $\chi^2 \geq df$, and the baseline model fits worse than our target model.



CFI

The CFI is a normed index, so it is bounded by 0 and 1.

- $\text{CFI} \in [0, 1]$

The CFI = 0 when the estimated and baseline models are equivalent.

- $\text{CFI} = 0$ *iif* $\left[(\chi^2 - df) - (\chi_0^2 - df_0)\right] = 0$

The CFI = 1 for estimated models that are equivalent to the true model or more complex than the true model.

- $\text{CFI} = 1$ *iif* $\chi^2 \leq df$

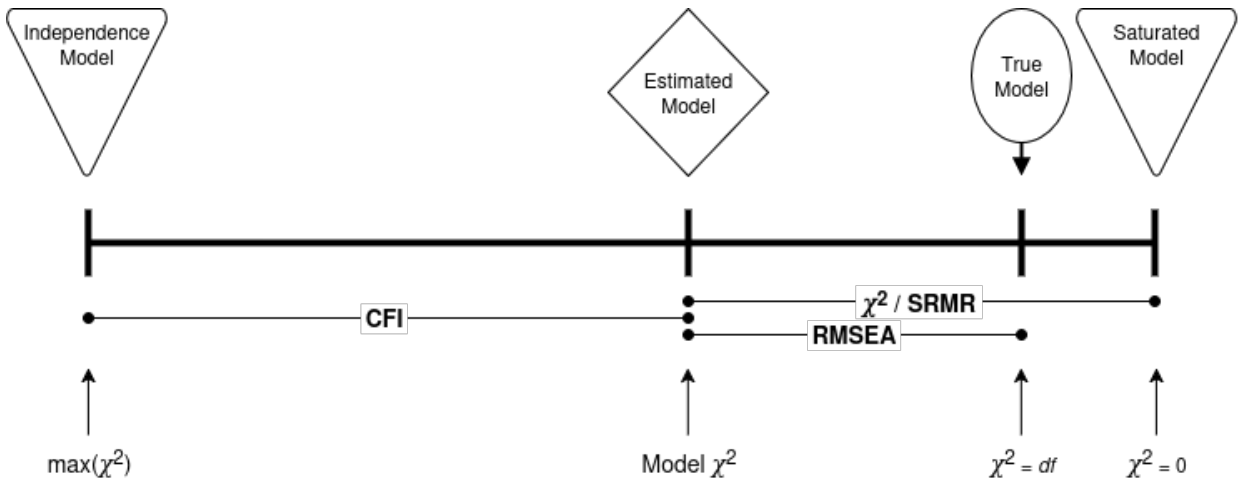
The CFI measures improvement over a poor model, so larger values represent better fit.

$$\text{CFI} \in \begin{cases} [0.95,\ 1] & \Rightarrow \text{Good Fit} \\ [0.90,\ 0.95) & \Rightarrow \text{Acceptable Fit} \\ [0,\ 0.90) & \Rightarrow \text{Bad Fit} \end{cases}$$

SRMR

The SRMR is an *absolute fit index* derived from the standardized model residuals.

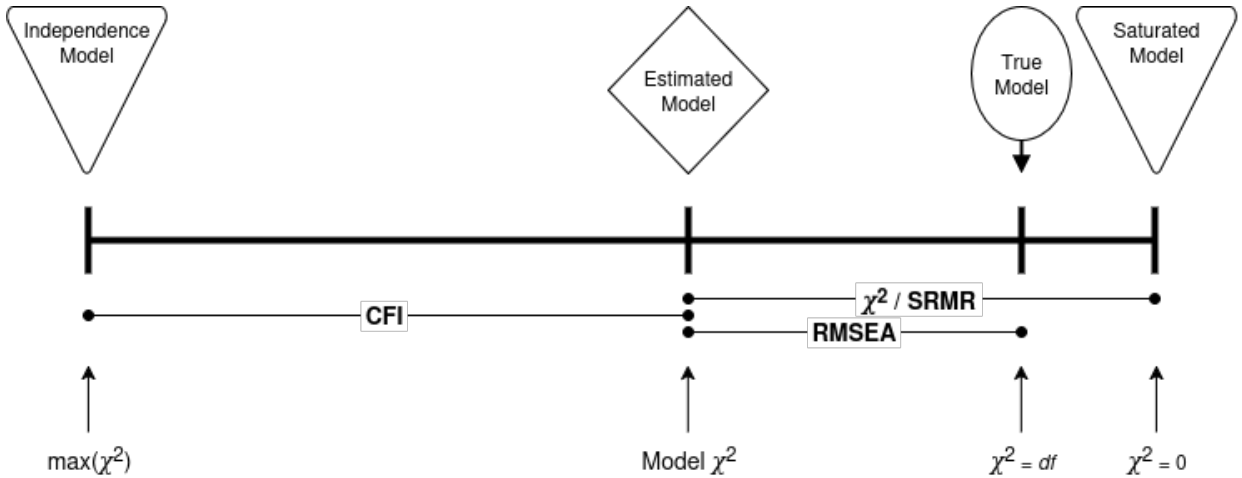
$$\text{SRMR} = \sqrt{\frac{\sum_{i=1}^{M}\sum_{j=1}^{i}\left(\frac{s_{ij}-\hat{\sigma}_{ij}}{s_i s_j}\right)^2}{Q}}$$



SRMR

Basically, the SRMR represents the average magnitude of the standardized model residuals.

$$\text{SRMR} \approx \text{mean}\left[\text{std}\left(\mathbf{S} - \hat{\Sigma}\right)\right]$$



SRMR

Technically, the SRMR is a normed index that is bounded by 0 and 1.

- $\text{SRMR} \in [0, 1]$
- For all real data, $SRMR \ll 1$

The SRMR = 0 when the estimated and saturated models are equivalent.

- $\text{SRMR} = 0$ *iif* $\hat{\Sigma} = \mathbf{S}$

The SRMR measures deviations from a perfectly fitting model, so smaller values represent better fit.

$$\text{SRMR} \in \begin{cases} [0,\ 0.08] & \Rightarrow \text{Good Fit} \\ (0.08,\ 0.1] & \Rightarrow \text{Acceptable Fit} \\ (0.1,\ 1] & \Rightarrow \text{Bad Fit} \end{cases}$$

Example: Target Model

We can use the `fitMeasures()` function to get model fit statistics.

```
fit1 <- fitMeasures(out1)

head(fit1, 22) |> round(3)
```

```
                npar                fmin               chisq                  df
              13.000               0.022             120.283               8.000
              pvalue      baseline.chisq         baseline.df     baseline.pvalue
               0.000            2053.922              15.000               0.000
                 cfi                 tli                nnfi                 rfi
               0.945               0.897               0.897               0.890
                 nfi                pnfi                 ifi                 rni
               0.941               0.502               0.945               0.945
                logl  unrestricted.logl                 aic                 bic
          -26114.771          -26054.629           52255.542           52332.236
              ntotal                bic2
            2696.000           52290.931
```

Example: Target Model

```
tail(fit1, -22) |> round(3)
```

```
                rmsea        rmsea.ci.lower        rmsea.ci.upper
                0.072                 0.061                 0.084
      rmsea.ci.level          rmsea.pvalue        rmsea.close.h0
                0.900                 0.001                 0.050
rmsea.notclose.pvalue     rmsea.notclose.h0                   rmr
                0.138                 0.080                 0.068
           rmr_nomean                  srmr           srmr_bentler
                0.068                 0.039                 0.039
  srmr_bentler_nomean                  crmr           crmr_nomean
                0.039                 0.046                 0.046
           srmr_mplus     srmr_mplus_nomean                cn_05
                0.039                 0.039               348.577
                cn_01                   gfi                  agfi
              451.298                 0.986                 0.963
                pgfi                   mfi                  ecvi
                0.376                 0.979                 0.054
```

Example: Independence Model

We can confirm that the baseline stats provided by lavaan match the versions from our explicitly estimated independence model.

```
indFit <- fitMeasures(indOut) |> as.list()
fit1   <- as.list(fit1)

indFit$chisq

[1] 2053.922

fit1$baseline.chisq

[1] 2053.922

indFit$df

[1] 15

fit1$baseline.df

[1] 15
```

Example: Edge Cases

The saturated model has perfect fit because it perfectly reproduces the observed sufficient statistics.

```
fitMeasures(satOut, c("chisq", "df", "rmsea", "cfi", "srmr"))

chisq    df rmsea   cfi  srmr
    0     0     0     1     0
```

Notice how small the independence model's SRMR value is.

```
indFit$srmr

[1] 0.1989567
```

Manually Calculating χ^2

```
llSat <- logLik(satOut) # saturated model loglikelihood
ll1   <- logLik(out1)   # target model loglikelihood

# Manually calculate the chi-squared stat
-2 * (ll1 - llSat) |> as.numeric()

[1] 120.2832

# Compare to lavaan's version
fit1$chisq

[1] 120.2832
```

Manually Calculating CFI

```
# Target model stats
chisq1 <- fit1$chisq
df1    <- fit1$df

# Independence model stats
chisq0 <- fit1$baseline.chisq
df0    <- fit1$baseline.df

# Manually calculate the CFI
1 - ((chisq1 - df1) / (chisq0 - df0))

[1] 0.9449301

# Compare to lavaan's version
fit1$cfi

[1] 0.9449301
```

Manually Calculating RMSEA

```
# Sample size
n <- fit1$ntotal

# Manually calculate the RMSEA
sqrt((chisq1 - df1) / (n * df1))

[1] 0.07215267

# Compare to lavaan's version
fit1$rmsea

[1] 0.07215267
```

Manually Calculating SRMR

```
M <- 6                  # No. of observed variables
Q <- M * (M + 1) / 2 # No. of unique elements in S

# Model-implied covariance matrix
sHat <- inspect(out1, "sigma")

# Observed covariance matrix
sObs <- inspect(out1, "sampstat")$cov

# Observed standard deviations
se <- sqrt(diag(sObs))
```

Manually Calculating SRMR

```r
# Sum of squared standardized residuals
x <- 0
for(i in 1:M) {
  for(j in 1:i) {
    x <- x + ( (sObs[i, j] - sHat[i, j]) / (se[i] * se[j]) )^2
  }
}

# Root mean of the above = SRMR
sqrt(x / Q) |> as.numeric()

[1] 0.03850419

# Compare to lavaan's version
fit1$srmr

[1] 0.03850419
```

Model Identification

Identifying Constraints

We must fix some parameters to identify the model.

- For each construct, we need $df \geq 0$.
- If the construct has three or more indicators:
  - Fix one parameter in the covariance model.
  - Fix one parameter in the mean model.
- If the construct has two indicators:
  - Fix an additional parameter in the covariance model.
- If the construct has only one indicator:
  - Cannot define a latent factor.

Under-Identified: Unconstrained Model

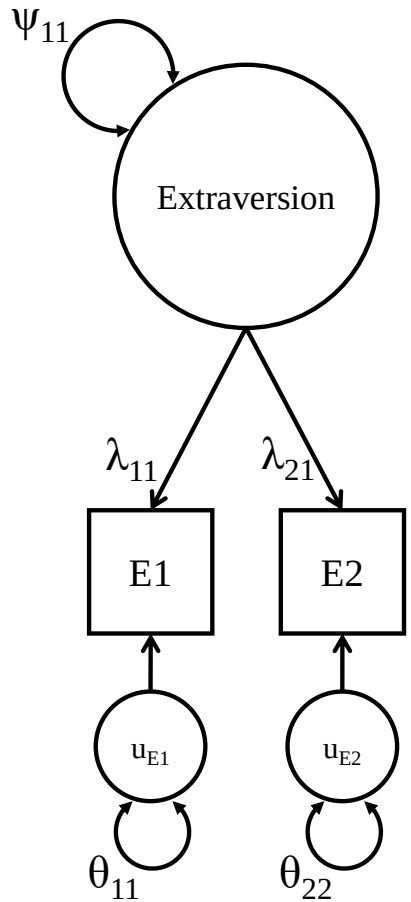
$$\Psi = \begin{bmatrix} \psi_{11} \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \lambda_{11} \\ \lambda_{21} \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \theta_{11} & \\ 0 & \theta_{22} \end{bmatrix}$$

$$Q = \frac{2(2+1)}{2} = 3$$

$$df = Q - P$$
$$= 3 - 5 = -2$$



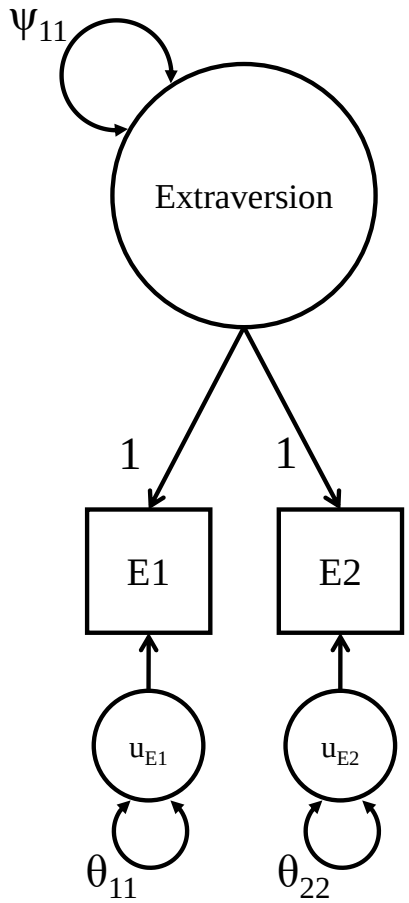
Under-Identified: Marker-Variable

$$\Psi = \begin{bmatrix} \psi_{11} \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \theta_{11} & \\ 0 & \theta_{22} \end{bmatrix}$$

$$df = Q - P$$
$$= 3 - 3 = 0$$



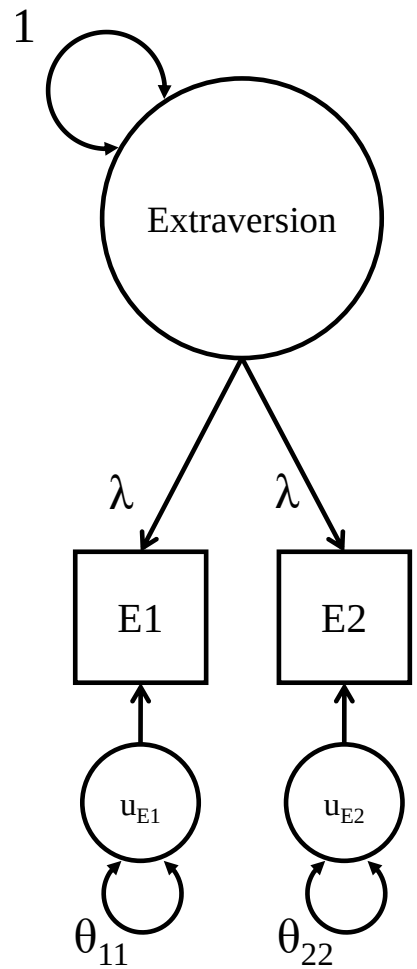
Under-Identified: Fixed-Factor

$$\Psi = \begin{bmatrix} 1 \end{bmatrix}$$

$$\Lambda = \begin{bmatrix} \lambda \\ \lambda \end{bmatrix}$$

$$\Theta = \begin{bmatrix} \theta_{11} & \\ 0 & \theta_{22} \end{bmatrix}$$

$$df = Q - P$$
$$= 3 - 3 = 0$$



Over-Identified Models

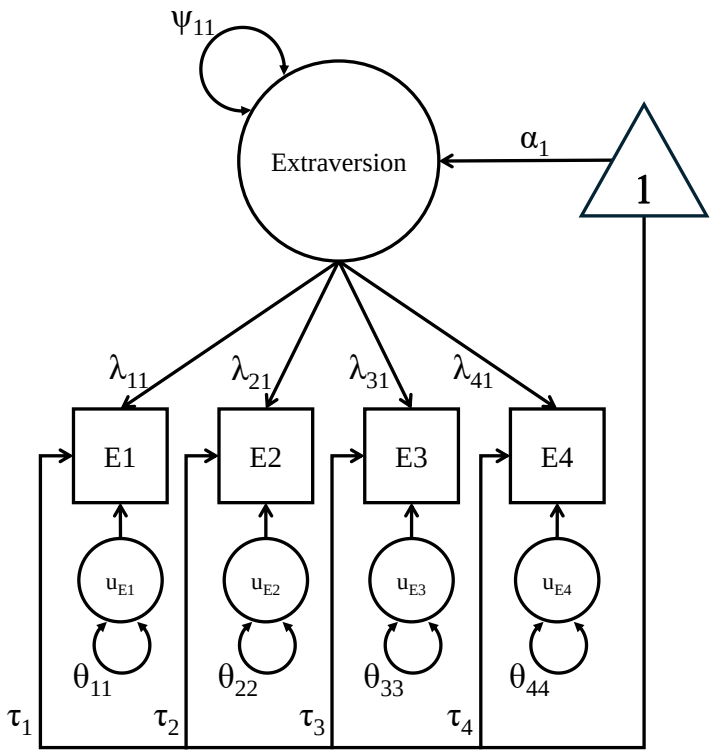
What happens when we have four (or more) indicators?

With 4 indicators, our model contains 14 parameters:

- Four factor loadings
- Four residual variances
- Four item intercepts
- One latent mean
- One latent variance

$$Q = \frac{4(4+1)}{2} + 4 = 14$$

$$df = 14 - 14 = 0$$



Over-Identified Models

With 5 indicators, we have 17 model parameters:

- Five factor loadings
- Five residual variances
- Five item intercepts
- One latent mean
- One latent variance

$$Q = \frac{5(5+1)}{2} + 5 = 20$$

$$df = 20 - 17 = 3$$

With 6 indicators, we have 20 model parameters:

- Six factor loadings
- Six residual variances
- Six item intercepts
- One latent mean
- One latent variance

$$Q = \frac{6(6+1)}{2} + 6 = 27$$

$$df = 27 - 20 = 7$$

Over-Identified Models

With four indicators, we automatically have $df \geq 0$ without imposing any scaling constraints.

- Can we directly estimate the unconstrained model?

```
mod1 <- '
fun =~ Q77 + Q84 + Q170 + Q196
'

fit1 <- lavaan(mod1,
               data          = sapi,
               auto.var      = TRUE,
               meanstructure = TRUE,
               int.ov.free   = TRUE,
               int.lv.free   = TRUE)

Warning:  lavaan->lav_model_vcov():
  Could not compute standard errors!  The information matrix could not be
  inverted.  This may be a symptom that the model is not identified.
```

Hmmm...I guess not.

Over-Identified Models

```
partSummary(fit1, 1:4)

lavaan 0.6-19 ended normally after 13 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        14

                                                  Used       Total
  Number of observations                           970        1000

Model Test User Model:

  Test statistic                                58.017
  Degrees of freedom                                 0
```

Over-Identified Models

```
partSummary(fit1, 7:8)

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fun =~
    Q77               0.824 NA
    Q84               0.578 NA
    Q170              0.479 NA
    Q196              0.619 NA

Intercepts:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Q77               3.574 NA
   .Q84               3.232 NA
   .Q170              3.955 NA
   .Q196              3.803 NA
    fun               0.000 NA
```

Over-Identified Models

```
partSummary(fit1, 9)

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Q77               0.491 NA
   .Q84               0.760 NA
   .Q170              0.703 NA
   .Q196              0.370 NA
    fun               1.012 NA
```

Why $df \geq 0$ is Not Enough

In the above example, the data contain enough information to define our model parameters, but that's not enough.

- We still need scaling constraints.
- The estimation algorithm needs an anchor point from which to extrapolate the relative values of the model parameters.
- Without any scaling constraints, an infinite number of parameter matrices will produce the same $\hat{\mu}$ and $\hat{\Sigma}$.
- An infinite number of solutions are equally good.

```
## Fit a model to get some example parameters:
mod1 <- '
fun   =~ Q77 + Q84 + Q170 + Q196
liked =~ Q44 + Q63 + Q76  + Q98
'

fit1 <- cfa(mod1, data = sapi, effect.coding = TRUE)
```

Why $df \geq 0$ is Not Enough

```
## Extract the estimated factor loadings and latent covariance matrix:
(lambda <- inspect(fit1, "est")$lambda)

       fun liked
Q77  1.254 0.000
Q84  0.954 0.000
Q170 0.795 0.000
Q196 0.997 0.000
Q44  0.000 0.764
Q63  0.000 1.155
Q76  0.000 1.133
Q98  0.000 0.949

(psi <- inspect(fit1, "est")$psi)

        fun liked
fun   0.399
liked 0.241 0.311
```

Why $df \geq 0$ is Not Enough

```
## Extract the estimated residual variances:
(theta <- inspect(fit1, "est")$theta)
```

```
      Q77   Q84  Q170  Q196   Q44   Q63   Q76   Q98
Q77  0.548
Q84  0.000 0.727
Q170 0.000 0.000 0.687
Q196 0.000 0.000 0.000 0.364
Q44  0.000 0.000 0.000 0.000 0.662
Q63  0.000 0.000 0.000 0.000 0.000 0.807
Q76  0.000 0.000 0.000 0.000 0.000 0.000 0.966
Q98  0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.469
```

Why $df \geq 0$ is Not Enough

```
## Manually compute the model-implied covariance matrix:
(sigma  <- lambda %*% psi %*% t(lambda) + theta)

       Q77   Q84  Q170  Q196   Q44   Q63   Q76   Q98
Q77  1.176
Q84  0.477 1.090
Q170 0.398 0.303 0.940
Q196 0.499 0.380 0.316 0.761
Q44  0.231 0.175 0.146 0.183 0.843
Q63  0.349 0.265 0.221 0.277 0.275 1.223
Q76  0.342 0.260 0.217 0.272 0.269 0.407 1.365
Q98  0.287 0.218 0.182 0.228 0.226 0.341 0.335 0.750
```

Why $df \geq 0$ is Not Enough

```
## Randomly sample an arbitrary scaling factor:
(a <- runif(1, 1, 2))

[1] 1.524314

## Rescale all factor loadings by a factor of a:
(lambda2 <- lambda * a)

       fun liked
Q77  1.911 0.000
Q84  1.454 0.000
Q170 1.212 0.000
Q196 1.520 0.000
Q44  0.000 1.164
Q63  0.000 1.760
Q76  0.000 1.727
Q98  0.000 1.447
```

Why $df \geq 0$ is Not Enough

```
## Rescale the latent covariance matrix by a factor of (1 / a^2):
(psi2 <- psi / a^2)

        fun liked
fun   0.172
liked 0.104 0.134

## Compute the model-implied covariance matrix using the rescaled parameters:
(sigma2  <- lambda2 %*% psi2 %*% t(lambda2) + theta)

       Q77   Q84  Q170  Q196   Q44   Q63   Q76   Q98
Q77  1.176
Q84  0.477 1.090
Q170 0.398 0.303 0.940
Q196 0.499 0.380 0.316 0.761
Q44  0.231 0.175 0.146 0.183 0.843
Q63  0.349 0.265 0.221 0.277 0.275 1.223
Q76  0.342 0.260 0.217 0.272 0.269 0.407 1.365
Q98  0.287 0.218 0.182 0.228 0.226 0.341 0.335 0.750
```

Why $df \geq 0$ is Not Enough

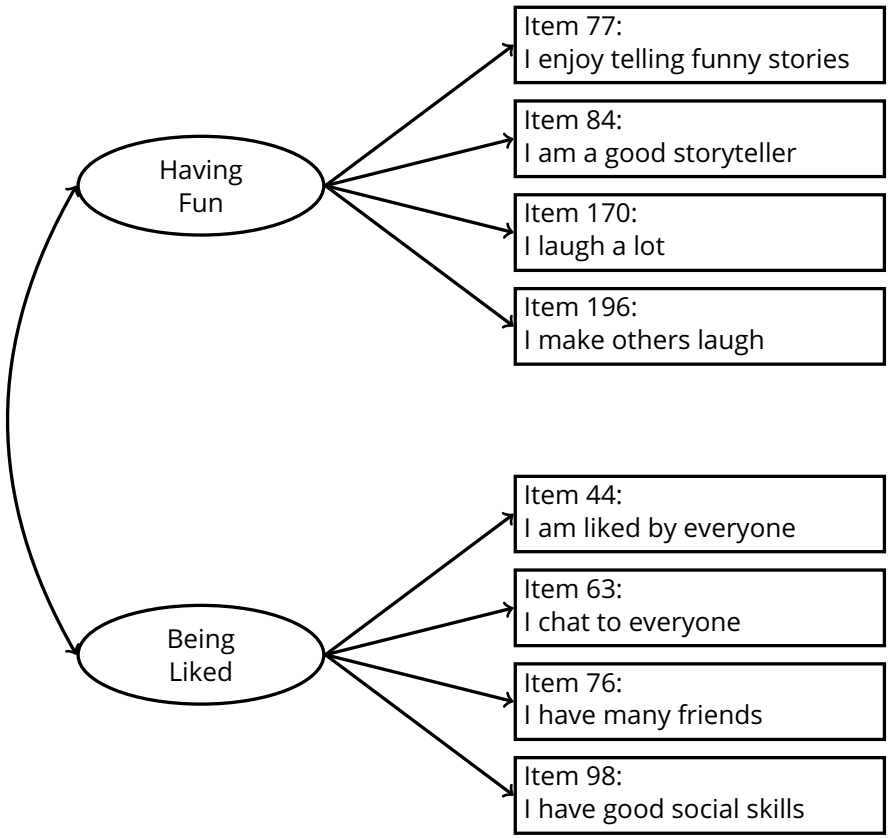
```
## Compare the two model-implied covariance matrices:
all.equal(sigma, sigma2)

[1] TRUE

## Repeat the experiment 100 times:
out <- rep(NA, 100)
for(i in 1:100) {
  a <- runif(1, 1, 2)

  lambda2 <- lambda * a
  psi2    <- psi / a^2
  sigma2  <- lambda2 %*% psi2 %*% t(lambda2) + theta

  out[i] <- all.equal(sigma, sigma2)
}

## Always the same result?
all(out)

[1] TRUE
```

Example

Example: Hypothesized Model

Let's revisit the example SAPI analysis from last week.

- We'll estimate the two-factor CFA shown to the right.



Example: Model Estimation

Load the SAPI data.

```
dataDir <- "data"
sapi <- read.table(here::here(dataDir, "sapi.txt"),
                   header = TRUE,
                   na.strings = "-999")
```

Specify the **lavaan** model syntax for the SAPI extraversion CFA.

```
mod1 <- '
fun   =~ Q77 + Q84 + Q170 + Q196
liked =~ Q44 + Q63 + Q76  + Q98
'
```

Use the `cfa()` function to estimate the model.

```
library(lavaan)
out1 <- cfa(mod1, data = sapi, std.lv = TRUE)
```

Example: Model Evaluation

```
partSummary(out1, 1:4)

lavaan 0.6-19 ended normally after 17 iterations

  Estimator                                         ML
  Optimization method                           NLMINB
  Number of model parameters                        17

                                                  Used       Total
  Number of observations                           959        1000

Model Test User Model:

  Test statistic                               130.193
  Degrees of freedom                            19
  P-value (Chi-square)                          0.000
```

Example: Model Evaluation

```
partSummary(out1, 5:7)

Parameter Estimates:

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

Latent Variables:
                   Estimate  Std.Err  z-value  P(>|z|)
  fun =~
    Q77               0.792    0.035   22.606    0.000
    Q84               0.603    0.035   17.193    0.000
    Q170              0.502    0.033   15.180    0.000
    Q196              0.630    0.028   22.308    0.000
  liked =~
    Q44               0.426    0.034   12.580    0.000
    Q63               0.644    0.040   16.071    0.000
    Q76               0.632    0.043   14.845    0.000
    Q98               0.530    0.031   16.912    0.000
```

Example: Model Evaluation

```
partSummary(out1, 8:9)

Covariances:
                   Estimate  Std.Err  z-value  P(>|z|)
  fun ~~
    liked             0.683    0.033   20.483    0.000

Variances:
                   Estimate  Std.Err  z-value  P(>|z|)
   .Q77               0.548    0.038   14.389    0.000
   .Q84               0.727    0.039   18.703    0.000
   .Q170              0.687    0.035   19.572    0.000
   .Q196              0.364    0.025   14.731    0.000
   .Q44               0.662    0.034   19.291    0.000
   .Q63               0.807    0.048   16.943    0.000
   .Q76               0.966    0.054   17.931    0.000
   .Q98               0.469    0.029   16.121    0.000
    fun               1.000
    liked             1.000
```

Example: Model Evaluation

```
inspect(out1, "r2")

  Q77   Q84  Q170  Q196   Q44   Q63   Q76   Q98
0.534 0.333 0.268 0.521 0.215 0.340 0.293 0.374

fitMeasures(out1) |> head(22) |> round(3)

                npar                fmin               chisq                  df
              17.000               0.068             130.193              19.000
              pvalue     baseline.chisq        baseline.df    baseline.pvalue
               0.000            1574.886              28.000               0.000
                 cfi                 tli                nnfi                 rfi
               0.928               0.894               0.894               0.878
                 nfi                pnfi                 ifi                 rni
               0.917               0.622               0.929               0.928
                logl  unrestricted.logl                 aic                 bic
         -10147.587          -10082.491           20329.175           20411.895
              ntotal                bic2
             959.000           20357.903
```

Example: Model Evaluation

```
fitMeasures(out1) |> tail(-22) |> round(3)
```

```
                  rmsea         rmsea.ci.lower         rmsea.ci.upper
                  0.078                  0.066                  0.091
        rmsea.ci.level           rmsea.pvalue         rmsea.close.h0
                  0.900                  0.000                  0.050
rmsea.notclose.pvalue      rmsea.notclose.h0                    rmr
                  0.421                  0.080                  0.042
            rmr_nomean                   srmr           srmr_bentler
                  0.042                  0.043                  0.043
  srmr_bentler_nomean                   crmr           crmr_nomean
                  0.043                  0.049                  0.049
            srmr_mplus      srmr_mplus_nomean                 cn_05
                  0.043                  0.043                223.037
                  cn_01                    gfi                    agfi
                267.582                  0.968                  0.939
                   pgfi                    mfi                    ecvi
                  0.511                  0.944                  0.171
```

Compare Scaling Methods

```
# Re-estimate the model with marker-variable identification
out2 <- cfa(mod1, data = sapi)

inspect(out1, "est")$lambda - inspect(out2, "est")$lambda

        fun  liked
Q77  -0.208  0.000
Q84  -0.158  0.000
Q170 -0.132  0.000
Q196 -0.165  0.000
Q44   0.000 -0.574
Q63   0.000 -0.868
Q76   0.000 -0.851
Q98   0.000 -0.713

inspect(out1, "est")$psi - inspect(out2, "est")$psi

        fun liked
fun   0.373
liked 0.453 0.818
```

Compare Scaling Methods

Both models partition the indicators' variance in the same way.

```
inspect(out1, "r2") - inspect(out2, "r2")

 Q77  Q84 Q170 Q196  Q44  Q63  Q76  Q98
   0    0    0    0    0    0    0    0

inspect(out1, "est")$theta - inspect(out2, "est")$theta

     Q77 Q84 Q170 Q196 Q44 Q63 Q76 Q98
Q77    0
Q84    0   0
Q170   0   0    0
Q196   0   0    0    0
Q44    0   0    0    0   0
Q63    0   0    0    0   0   0
Q76    0   0    0    0   0   0   0
Q98    0   0    0    0   0   0   0   0
```

Compare Scaling Methods

Both models produce equivalent representations of the data.

```
all.equal(fitMeasures(out1), fitMeasures(out2))

[1] TRUE

inspect(out1, "sigma") - inspect(out2, "sigma")

     Q77 Q84 Q170 Q196 Q44 Q63 Q76 Q98
Q77    0
Q84    0   0
Q170   0   0    0
Q196   0   0    0    0
Q44    0   0    0    0   0
Q63    0   0    0    0   0   0
Q76    0   0    0    0   0   0   0
Q98    0   0    0    0   0   0   0   0
```