Introduction to Structural Equation Modeling using lavaan

Exploratory and Confirmatory Factor Analysis

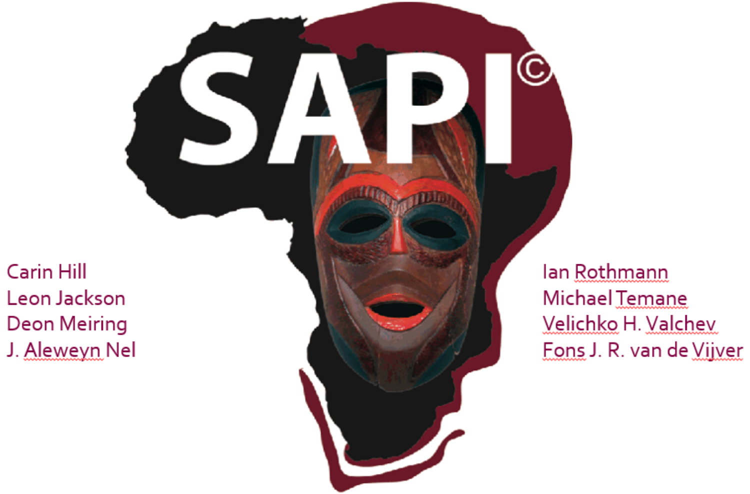
R. M. Kuiper (and many others)

Department of Methodology & Statistics
Utrecht University

Outline of this lecture

Table of Contents

Example: South African Personality Inventory Project (SAPI)



Nel, J. A., Valchev, V. H., Rothmann, S., van de Vijver, F. J. R., Meiring, D., & de Bruin, G. P. (2012). Exploring the personality structure in the 11 languages of South Africa. *Journal of Personality, 80*, 915–948.

SAPI details

- 1216 participants from 11 official language groups
- From about 50,000 descriptive responses to 262 personality items
- Nine personality clusters:
  - Conscientiousness
  - Emotional Stability
  - Extraversion
  - Facilitating
  - Integrity
  - Intellect
  - Openness
  - Relationship Harmony
  - Soft-Heartedness (Ubuntu)
- Our data: selection of 1000 participants

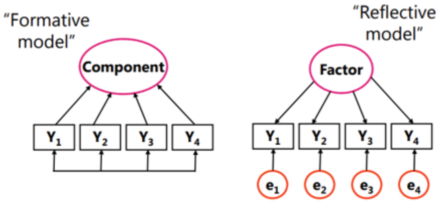
Table of Contents

Factor Analysis

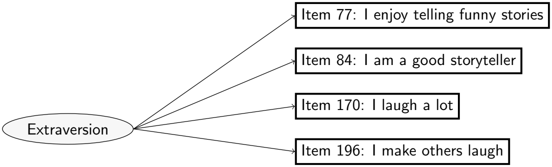
Factor Analysis: Modeling measurement of a latent variable

- EFA: Exploratory Factor Analysis.
- CFA: Confirmatory Factor Analysis.

Both EFA and CFA use a "reflective" measurement model, not a "formative" model.

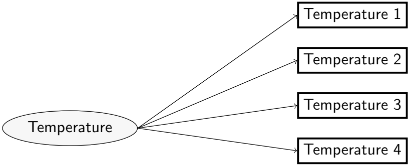


Reflective measurement model



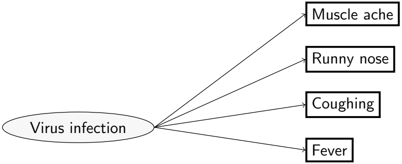
- Items are dependent variables, caused by the factor!
- Latent variable 'extraversion' explains item correlations: The factor is the reason for the covariances/correlations.

Reflective measurement model



Note:
Thermometer readings are the dependent variables, caused by the temperature!

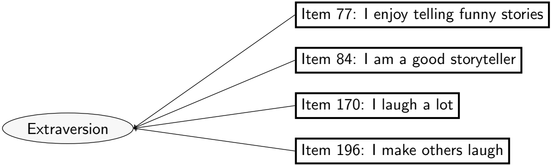
Reflective measurement model



Note:
symptoms are the dependent variables,
caused by the virus infection!

Formative measurement model

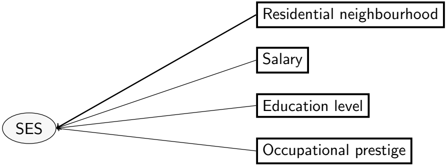
If formative measurement model:



Note:

- Extraversion is the dependent variable, predicted by the items.
- **Extraversion is defined as a (weighted) sum of the items:** This is not a testable measurement model, but a definition.

Formative measurement model



Note:
SES is defined as a (weighted) sum of the items.

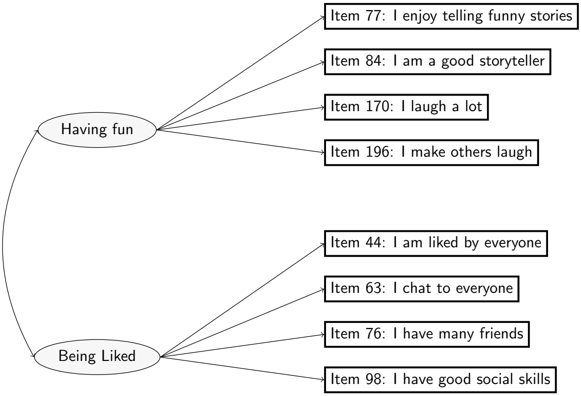
Interesting read

Interesting read on theory & latent variables:

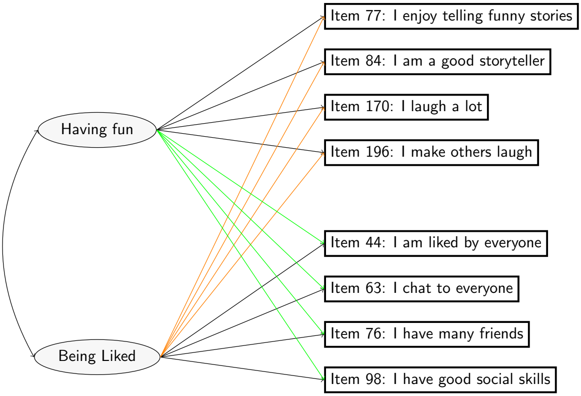
Borsboom, D., Mellenbergh, G.J., & Van Heerden, J. (2003). The theoretical status of latent variables. *Psychological review, 110*(2), 203.

Confirmatory or exploratory?

Two sub-scales of extraversion



EFA: all loadings including cross-loadings



CFA: only hypothesized loadings

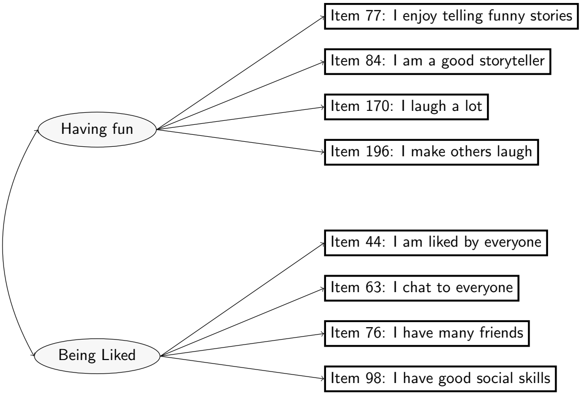
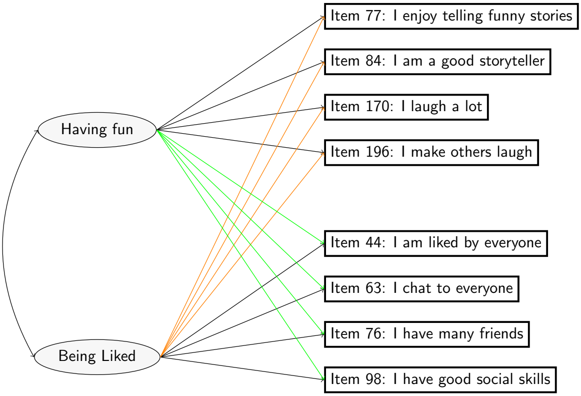


Table of Contents

Step 1: Loading data into R

```
data_sapi <- read.table("Sapi.txt", header = T)

data_sapi[sapply(data_sapi,
    function(x) as.character(x) %in% c("-999") )] <- NA
```

Step 3: Draw your model



Step 4: Specify EFA in lavaan (general case)

In factor analysis (EFA and CFA):

- $\rightarrow$ is latent variable definition ('is measured by'): $=\sim$
- $\leftrightarrow$ is covariance. By default, factors are related.
- In EFA: use *efa*("*efa*")$*$ in front of the latent variable / factor.

```
# 1-factor model
f1 <- '
efa("efa")*f1 =~ y1 + y2 + y3 + ...
'
# 2-factor model
f2 <- '
efa("efa")*f1 +
efa("efa")*f2 =~ y1 + y2 + y3 + ...
'
# 3-factor model
f3 <- '
efa("efa")*f1 +
efa("efa")*f2 +
efa("efa")*f3 =~ y1 + y2 + y3 + ...
```

Step 4: Specify our SAPI EFA model

In factor analysis (EFA and CFA):

- $\rightarrow$ is latent variable definition ('is measured by'): $=\sim$
- $\leftrightarrow$ is covariance. By default, factors related.
- Use *efa*("*efa*")$*$ in front of the latent variable / factor.

```
# two-factor EFA
model.2EFA <- "
 efa('block1')*Havingfun  =~ Q77 + Q84 + Q170 + Q196 +
                              Q44 + Q63 + Q76  + Q98
 efa('block1')*Beingliked =~ Q77 + Q84 + Q170 + Q196 +
                              Q44 + Q63 + Q76  + Q98
"
```

Step 5: Fit the model

Use the cfa() function in lavaan.

```
fit_2EFA <- cfa(model.2EFA, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
# Note: FIML will be discusses in the Missing Data lecture.
```

Step 5. Fit the EFA model: All code

```
# Data
data_sapi <- read.table("Sapi.txt", header = T)
data_sapi[sapply(data_sapi,
    function(x) as.character(x) %in% c("-999") )] <- NA

# Model: two-factor EFA
model.2EFA <- "
 efa('block1')*Havingfun  =~ Q77 + Q84 + Q170 + Q196 +
                              Q44 + Q63 + Q76  + Q98
 efa('block1')*Beingliked =~ Q77 + Q84 + Q170 + Q196 +
                              Q44 + Q63 + Q76  + Q98
"

# Fit model
fit_2EFA <- cfa(model.2EFA, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
```

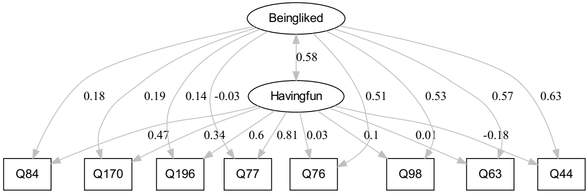
Step 6: Plot the lavaan model in R

```r
if (!require("lavaanPlot")) install.packages("lavaanPlot")
library(lavaanPlot)
```

```r
lavaanPlot(model = fit_2EFA,
           node_options = list(shape = "box",
                               fontname = "Helvetica"),
           edge_options = list(color = "grey"),
           coefs = T,
           stand = T, # standardized
           covs = T)
```

Step 6: Plot the lavaan model in R Ctd.

```
lavaanPlot(model = fit_2EFA,
           node_options = list(shape = "box",
                               fontname = "Helvetica"),
           edge_options = list(color = "grey"),
           coefs = T, stand = T, covs = T)
```



Step 9: Acquiring the summary

```
summary(fit_2EFA)

parameterEstimates(fit_2EFA)

fitMeasures(fit_2EFA, c("chisq", "df", "pvalue",
                        "cfi", "tli",
                        "rmsea","srmr"))
# As an example, there are more.
```

Step 9: Acquiring the summary

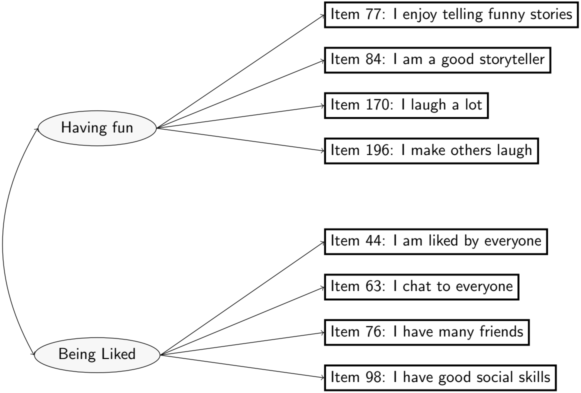
```
parameterEstimates(fit_2EFA)[1:16,1:5][,-4]

##             lhs op  rhs    est
## 1    Havingfun =~  Q77  0.884
## 2    Havingfun =~  Q84  0.493
## 3    Havingfun =~ Q170  0.336
## 4    Havingfun =~ Q196  0.523
## 5    Havingfun =~  Q44 -0.163
## 6    Havingfun =~  Q63  0.014
## 7    Havingfun =~  Q76  0.031
## 8    Havingfun =~  Q98  0.085
## 9   Beingliked =~  Q77 -0.028
## 10  Beingliked =~  Q84  0.186
## 11  Beingliked =~ Q170  0.187
## 12  Beingliked =~ Q196  0.119
## 13  Beingliked =~  Q44  0.576
## 14  Beingliked =~  Q63  0.632
## 15  Beingliked =~  Q76  0.595
```

Table of Contents

Step 1: Loading data into R

```
data_sapi <- read.table("Sapi.txt", header = T)

data_sapi[sapply(data_sapi,
    function(x) as.character(x) %in% c("-999") )] <- NA
```

Step 3: Draw your model



Step 4: Specify CFA in lavaan (general case)

- $\rightarrow$ is latent variable definition ('is measured by'): $=\sim$
- $\leftrightarrow$ is covariance. By default, factors are related.

```
# k-factor model
model.kCFA <- '
latent variable_1 =~ indicator11 + indicator12 + ...
latent variable_2 =~ indicator21 + indicator22 + ...
...
latent variable_k =~ indicatork1 + indicatork2 + ...
'
```

Step 4: Specify our SAPI CFA model

CFA: Only hypothesized loadings.
CFA vs EFA: force cross-loadings to zero.

In lavaan:
one can fix loadings to zero, BUT instead:
let manifest variables only appear in the equation of their factor.

```
# two-factor CFA
model.2CFA <- "
 Havingfun  =~ Q77 + Q84 + Q170 + Q196
 Beingliked =~ Q44 + Q63 + Q76  + Q98
"
```

Step 5: Fit the model

Use the cfa() function in lavaan.

```
fit_2CFA <- cfa(model.2CFA, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
# Note: FIML will be discusses in the Missing Data lecture.
```

5. Fit the CFA model: All code

```
# Data
data_sapi <- read.table("Sapi.txt", header = T)
data_sapi[sapply(data_sapi,
    function(x) as.character(x) %in% c("-999") )] <- NA

# Model: two-factor CFA
model.2CFA <- "
 Havingfun  =~ Q77 + Q84 + Q170 + Q196
 Beingliked =~ Q44 + Q63 + Q76  + Q98
"

# Fit model
fit_2CFA <- cfa(model.2CFA, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
```

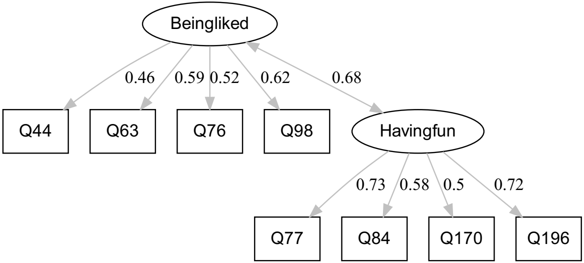
Step 6: Plot the lavaan model in R

```
if (!require("lavaanPlot")) install.packages("lavaanPlot")
library(lavaanPlot)
```

```
lavaanPlot(model = fit_2CFA,
           node_options = list(shape = "box",
                               fontname = "Helvetica"),
           edge_options = list(color = "grey"),
           coefs = T,
           stand = T,
           covs = T)
```

Step 6: Plot the lavaan model in R Ctd.

```
lavaanPlot(model = fit_2CFA,
           node_options = list(shape = "box",
                               fontname = "Helvetica"),
           edge_options = list(color = "grey"), coefs = T,
           stand = T, covs = T)
```



Step 9: Acquiring the summary

```r
summary(fit_2CFA)

parameterEstimates(fit_2CFA)

fitMeasures(fit_2CFA, c("chisq", "df", "pvalue",
                        "cfi", "tli",
                        "rmsea","srmr"))
# As an example, there are more.
```

```r
#The factors scores for each subject can be required via:
predict(fit_2CFA)
```

CFA: modification indices - lavaan commands

Remark: Blending confirmatory and exploratory!
Make sure it makes sense!

In lavaan, modification indices can be requested

- within the summary call:

```
summary(fit_2CFA, modindices = TRUE)
```

- directly:

```
modindices(fit_2CFA, sort = TRUE)
```

- for specific parameters, say, factor loadings:

```
mi <- modindices(fit)
mi[mi$op == "=~",]
```

Also, have a look at the lavTestScore() function.

CFA: modification indices - interpretation

```
#modindices(fit, sort = TRUE, maximum.number = 7) # or:
modindices(fit_2CFA, sort = TRUE)[1:7,] # first 7 rows

##              lhs op  rhs     mi    epc sepc.lv sepc.all sepc.nox
## 51          Q170 ~~ Q196 36.273  0.138   0.138    0.271    0.271
## 38           Q77 ~~  Q84 35.431  0.194   0.194    0.305    0.305
## 46           Q84 ~~ Q196 31.126 -0.143  -0.143   -0.276   -0.276
## 45           Q84 ~~ Q170 20.426 -0.123  -0.123   -0.170   -0.170
## 50           Q84 ~~  Q98 16.530  0.092   0.092    0.156    0.156
## 35 Beingliked =~  Q84 13.085  0.552   0.234    0.222    0.222
## 39           Q77 ~~ Q170 12.876 -0.104  -0.104   -0.166   -0.166
```

- mi: If parameter freely estimated, overall Chi-square statistic could decrease by approximately this amount.
- epc (= expected parameter change): Approximate value that a parameter is expected to attain.

CFA: modification indices - cross-loadings

Cross-loadings:

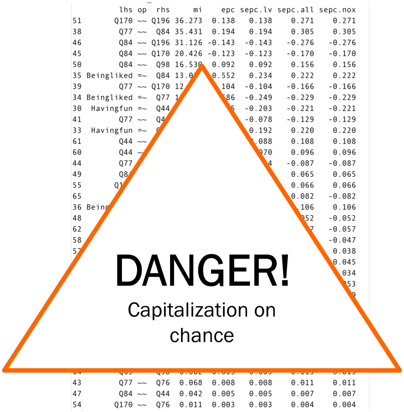
lhs	op	rhs	mi	epc	sepc.lv	sepc.all	sepc.nox
Q170	~~	Q196	36.273	0.138	0.138	0.271	0.271
Q77	~~	Q84	35.431	0.194	0.194	0.305	0.305
Q84	~~	Q196	31.126	-0.143	-0.143	-0.276	-0.276
Q84	~~	Q170	20.426	-0.123	-0.123	-0.170	-0.170
Q84	~~	Q98	16.530	0.092	0.092	0.156	0.156
Beingliked	=~	Q84	13.085	0.552	0.234	0.222	0.222
Q77	~~	Q170	12.876	-0.104	-0.104	-0.166	-0.166
Beingliked	=~	Q77	12.564	-0.586	-0.249	-0.229	-0.229
Havingfun	=~	Q44	11.853	-0.255	-0.203	-0.221	-0.221
Q77	~~	Q44	10.621	-0.078	-0.078	-0.129	-0.129

CFA: modification indices - residual variances

Residual covariances:

lhs	op	rhs	mi	epc	sepc.lv	sepc.all	sepc.nox
Q170	~~	Q196	36.273	0.138	0.138	0.271	0.271
Q77	~~	Q84	35.431	0.194	0.194	0.305	0.305
Q84	~~	Q196	31.126	-0.143	-0.143	-0.276	-0.276
Q84	~~	Q170	20.426	-0.123	-0.123	-0.170	-0.170
Q84	~~	Q98	16.530	0.092	0.092	0.156	0.156
Beingliked	=~	Q84	13.085	0.552	0.234	0.222	0.222
Q77	~~	Q170	12.876	-0.104	-0.104	-0.166	-0.166
Beingliked	=~	Q77	12.564	-0.586	-0.249	-0.229	-0.229
Havingfun	=~	Q44	11.853	-0.255	-0.203	-0.221	-0.221
Q77	~~	Q44	10.621	-0.078	-0.078	-0.129	-0.129

CFA: modification indices - be aware!



CFA: modification indices - modification

```r
modindices(fit_2CFA, sort = TRUE)[1,]

##      lhs op  rhs     mi   epc sepc.lv sepc.all sepc.nox
## 51 Q170 ~~ Q196 36.273 0.138   0.138    0.271    0.271

# Allow residuals of Q170 and Q196 to covary
```

```r
# Modified model:
# two-factor CFA + residual covariance Q170 and Q196
model.2CFA_mod <- "
 Havingfun  =~ Q77 + Q84 + Q170 + Q196
 Beingliked =~ Q44 + Q63 + Q76  + Q98
 Q170 ~~ Q196
"

# Fit model
fit_2CFA_mod <- cfa(model.2CFA_mod, data=data_sapi,
                    missing='fiml', fixed.x=F)  # use FIML
```

CFA: modification indices - test modification

```
anova(fit_2CFA, fit_2CFA_mod)[,-c(2,3)] # without AIC & BIC

##                 Df   Chisq Chisq diff   RMSEA Df diff Pr(>Chisq)
## fit_2CFA_mod 18  88.738
## fit_2CFA     19 124.170     35.432 0.18556       1  2.641e-09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
```

```
modindices(fit_2CFA, sort = TRUE)[1,]

##      lhs op  rhs     mi   epc sepc.lv sepc.all sepc.nox
## 51 Q170 ~~ Q196 36.273 0.138   0.138    0.271    0.271
```

CFA: modification indices - new parameter value

```
parameterEstimates(fit_2CFA_mod)[9,-c(5,6,7)] # no se, z, and p

##     lhs op  rhs   est ci.lower ci.upper
## 9 Q170 ~~ Q196 0.137    0.089    0.184
```

```
modindices(fit_2CFA, sort = TRUE)[1,1:5] # no sepc

##      lhs op  rhs     mi   epc
## 51 Q170 ~~ Q196 36.273 0.138
```

CFA: cross-loadings approximately zero

Problem:

- Restricting cross-loadings to exactly zero can be too strict.
- Consequence: rejection of the model, model modifications that capitalise on chance.

(Possible) solution in Bayesian SEM (BSEM) blavaan:

- Replace exact zero restrictions with approximate ones.
- Using Bayesian small-variance priors.

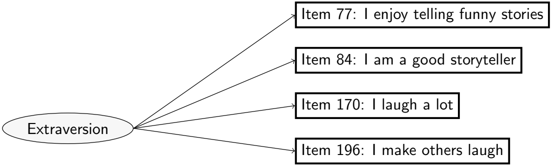
Interesting reading:

– Merkle, E. C., & Rosseel, Y. (2018). blavaan: Bayesian Structural Equation Models via Parameter Expansion. Journal of Statistical Software, 85(4), 1–30. https://doi.org/10.18637/jss.v085.i04
– Muthén, B., & Asparouhov, T. (2012). Bayesian structural equation modelling: A more flexible representation of substantive theory. Psychological Methods, 17(3), 313-335.

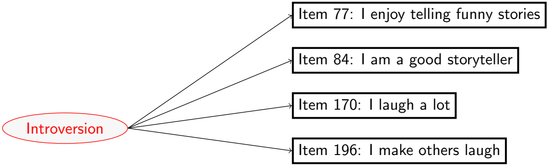
Table of Contents

Latent variable scaling

Latent variables are not observed, thus no inherent scale.



Latent variable scaling Ctd.



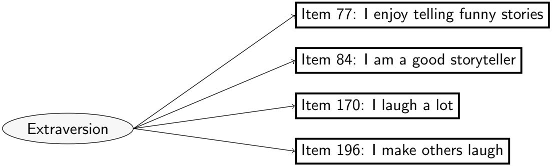
Therefore, set up model such that scale of latent variable is clear.

Three common ways

1. Marker-variable method
Constrain one of the factor loadings (default).

2. Reference group method:
Constrain the factor variance.

3. Effect coding:
Constrain the average of the loadings.



1. Marker-variable method (default)

Default parameterization:

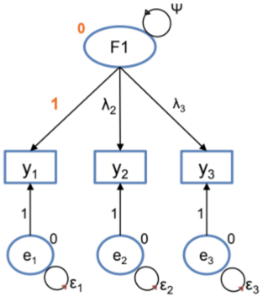
- First factor loading constrained at 1.
- Factor mean constrained at 0.

Other defaults:

- Mean of residuals is by definition 0.
- Residuals have a loading of 1.

Estimated:

- factor variance (Ψ),
- 'other' factor loadings (λ_2, λ_3),
- all item intercepts (ν_1, ν_2, ν_3),
- all residual variances (ϵ_1, ϵ_2, ϵ_3).



1. Default marker-variable method - lavaan

```
# Model
model.1CFA <- '
 Extraversion =~ Q77 + Q84 + Q170 + Q196
'

# Fit model
fit_1CFA <- cfa(model.1CFA, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
```

- First factor loading constrained at 1:

```
Extraversion =~
  Q77                1.000
```

- Factor mean constrained at 0:

```
Extraversion       0.000
```

1. Default marker-variable method - lavaan Ctd

```
parameterEstimates(fit_1CFA)[1:4,-c(5,6,7)]

##            lhs op  rhs   est ci.lower ci.upper
## 1 Extraversion =~  Q77 1.000    1.000    1.000
## 2 Extraversion =~  Q84 0.708    0.616    0.799
## 3 Extraversion =~ Q170 0.567    0.466    0.668
## 4 Extraversion =~ Q196 0.742    0.640    0.845
```

Factor loading of first indicator fixed to 1.
all other loadings are relative to that.

If reference category changed, other loadings also change.

2. Reference-group method

Parameterization:

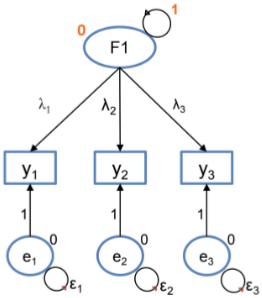
- Factor variance constrained at 1.
- Factor mean constrained at 0.*

Defaults:

- Mean of residuals is by definition 0.
- Residuals have a loading of 1.

Estimated:

- all factor loadings (λ_1, λ_2, λ_3),
- all item intercepts (ν_1, ν_2, ν_3),
- all residual variances (ϵ_1, ϵ_2, ϵ_3).



* In a multi-group model (discussed in another lecture), the mean of the (by default) first group is set to 0 (hence, the name 'reference-group').

2. Reference-group method - lavaan

```
# Model
model.1CFA_RefGr <- '
  # Free first factor loading, using: NA*
  Extraversion =~ NA*Q77 + Q84 + Q170 + Q196

  # Set factor variance to 1, using: 1*
  Extraversion ~~ 1*Extraversion
 '


# Fit model
fit_1CFA_RefGr <- cfa(model.1CFA_RefGr, data=data_sapi,
                missing='fiml', fixed.x=F)  # use FIML
```

Shortcut to fix the variances of (all the) latent variables to 1:

```
fit_1CFA_RefGr2 <- cfa(model.1CFA,    # ! 'original' model !
                       std.lv = TRUE, # fix variances to 1
                data=data_sapi, missing='fiml', fixed.x=F)
```

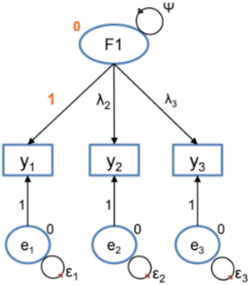
2. Reference-group method - lavaan Ctd

```
parameterEstimates(fit_1CFA_RefGr)[1:4,-c(5,6,7)]

##             lhs op  rhs   est ci.lower ci.upper
## 1 Extraversion =~  Q77 0.835    0.759    0.910
## 2 Extraversion =~  Q84 0.591    0.520    0.662
## 3 Extraversion =~ Q170 0.473    0.404    0.543
## 4 Extraversion =~ Q196 0.619    0.559    0.680

#parameterEstimates(fit_1CFA_RefGr2)[1:4,-c(5,6,7)]
```

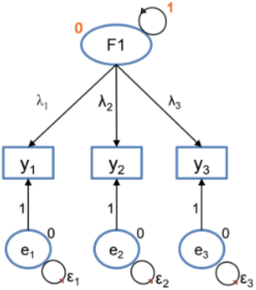
Advantage:
All factor loadings and scores on standardized metric.

Which method to choose?

1. Marker-variable method



2. Reference-group method



Does not matter for substantive conclusions.
Sometimes, pragmatic reasons.

3. Effects-coding method

Parameterization:

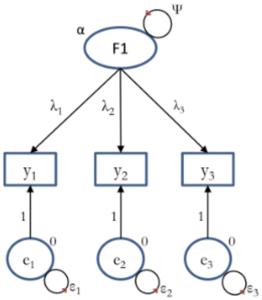
- Constrain the average of the factor loadings to 1: $\frac{1}{3}\sum_{i=1}^{3}\lambda_i = 1$.
- Constrain the average of the item intercepts to 0: $\frac{1}{3}\sum_{i=1}^{3}\nu_i = 0$.

Defaults:

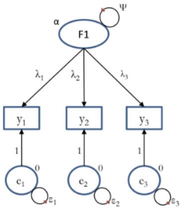
- Mean of residuals is by definition 0.
- Residuals have a loading of 1.

Estimated (subject to the constraints):

- factor variance (Ψ),
- factor mean (α),
- all factor loadings (λ_1, λ_2, λ_3),
- all item intercepts (ν_1, ν_2, ν_3),
- all residual variances (ϵ_1, ϵ_2, ϵ_3).



3. Effects-coding method Ctd



Interpretations can be intuitive:

- Factor on similar scale as the indicators.
- Factor variance (Ψ): average variance of each indicator that can be explained by the factor.
- Factor mean (α): weighted mean of the indicator means

3. Effects-coding method - lavaan model

```
# Model
model.1CFA_EffC <- '
  # Label parameters, such that they can be constrained
  Extraversion =~ lambda1*Q77 + lambda2*Q84 +
                  lambda3*Q170 + lambda4*Q196
  # intercepts
  Q77  ~ nu1*1
  Q84  ~ nu2*1
  Q170 ~ nu3*1
  Q196 ~ nu4*1

  # Constrain average of loadings to 1, i.e., set sum to 4
  lambda1 == 4 - lambda2 - lambda3 - lambda4
  # Constrain average of item intercepts to 0,
  # i.e., set sum to 0
  nu1 == 0 - nu2 - nu3 - nu4
 '
```

3. Effects-coding method - fit lavaan model

Now, use the lavaan() function:

```
# Fit model: Now, use the lavaan() function!
fit_1CFA_EffC <- lavaan(model.1CFA_EffC, data=data_sapi,
                        missing='fiml', fixed.x=F,
                        auto.var = TRUE,
                        auto.fix.first = FALSE,
                        auto.cov.lv.x = TRUE,
                        int.ov.free = TRUE)
```

3. Effects-coding method - lavaan output

- Constrain the average of the factor loadings to 1: $\frac{1}{4}\sum_{i=1}^{4}\lambda_i = 1$.

```
parameterEstimates(fit_1CFA_EffC)[1:4,1:5]
```

```
##             lhs op  rhs   label   est
## 1 Extraversion =~  Q77 lambda1 1.197
## 2 Extraversion =~  Q84 lambda2 1.027
## 3 Extraversion =~ Q170 lambda3 0.879
## 4 Extraversion =~ Q196 lambda4 0.898
```

- Constrain the average of the item intercepts to 0: $\frac{1}{4}\sum_{i=1}^{4}\nu_i = 0$.

```
parameterEstimates(fit_1CFA_EffC)[5:8,1:5]
```

```
##    lhs op rhs label    est
## 5  Q77 ~1         nu1 -0.765
## 6  Q84 ~1         nu2 -0.521
## 7 Q170 ~1         nu3  0.738
## 8 Q196 ~1         nu4  0.548
```

Table of Contents

Summary

- EFA and CFA
- Scaling

Thanks & How to proceed

Thanks for listening!

Are there any questions?

- Ask fellow participant on course platform.
- Ask teacher during Q&A (or via course platform).
- See if making the lab exercises help.
- Check the lavaan tutorial: e.g., `https://lavaan.ugent.be/tutorial/index.html`.
- Do not forget that Google is your best friend :-).

You can start working on the lab exercises.

Table of Contents

Extra:

Sample size

based on https://www.theanalysisfactor.com/sample-size-needed-for-factor-analysis/

Sample Size Rules of Thumb - total sample size

Some authors use a criterion based on the total sample size:

- 100 subjects = sufficient if clear structure; more is better (Kline, 1994).
- 100 subjects = poor; 300 = good; 1000+ = excellent (Comrey & Lee, 1992).
- 300 subjects, though fewer works if correlations are high among variables (Tabachnik & Fidell, 2001).

Sample Size Rules of Thumb - ratio cases vs variables

Others base it on a ratio of the number of cases to the number of variables involved in the factor analysis:

- 10-15 subjects per variable (Pett, Lackey, & Sullivan).
- 10 subjects per variable (Nunnally, 1978).
- 5 subjects per variable or 100 subjects, whichever is larger (Hatcher, 1994).
- 2 subjects per variable (Kline, 1994).

Sample Size Rules of Thumb - ratio cases vs factors

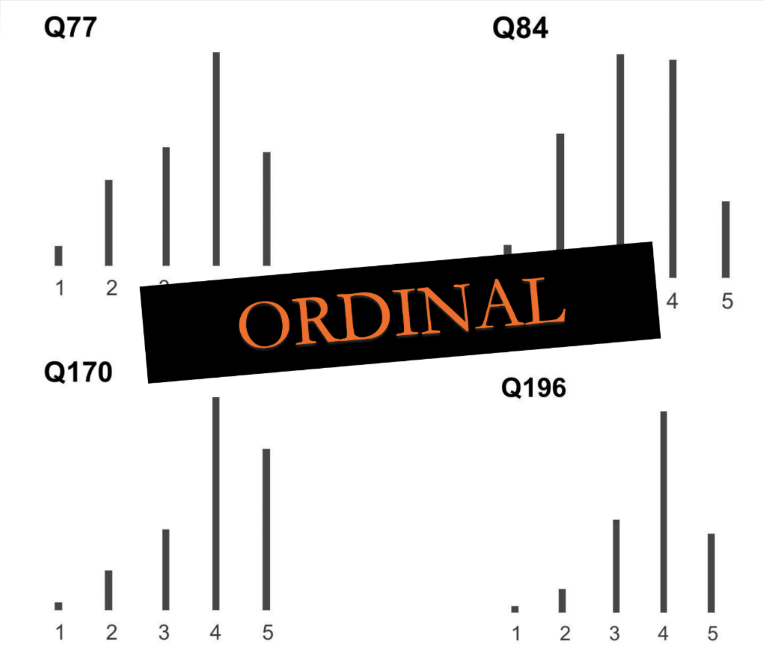
And then others base it on a ratio of cases to the number of factors:

- 20 subjects per factor (Arrindel & van der Ende, 1985).

Extra:
Categorical/Ordinal or continuous indicators?

Note: in cfa() you can, for example use 'ordered = TRUE' for endogenous variable.
Default then: estimator = "WLSMV".

More information on: https://lavaan.ugent.be/tutorial/cat.html


Q77
1 2
Q84
4 5
ORDINAL
Q170
1 2 3 4 5
Q196
1 2 3 4 5

Remark!

Do NOT use a χ^2 test or IC (AIC or BIC)
to compare categorical and continuous models:

- Obviously not nested (so, no χ^2 test anyway).
- AND likelihoods of categorical and continuous indicator models are incomparable!

Note: χ^2 test and IC are based on (log) likelihood ($=$ fit).

Interesting Reading

https://lavaan.ugent.be/tutorial/cat.html

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. Psychometrica, 49(1), 115-132.

Rhemtulla, M., Brosseau-Liard, P.É., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. Psychological Methods, 17(3), 354-373.

Sventina, D., Rutkowski, D. (2020). Multiple group invariance with categorical outcomes using updated guidelines: an illustration using Mplus and the lavaan/semtools packages. Structural Equational Modelling: A Multidisciplinary Journal, 27(1), 111-130