Computer lab: Experimenting with SAM

0. Introduction

Before we get started, we (install and) load the `lavaan` package

```r
# install.packages{'lavaan'}
library(lavaan)
```

```
## This is lavaan 0.6-20.2278
## lavaan is FREE software! Please report any bugs.
```

to install the latest development version of lavaan

```r
library(remotes)
remotes::install_github("yrosseel/lavaan")
```

and restart your R session. If you want to revert to the official (CRAN) version of lavaan again, simply type `install.packages("lavaan")` in the console and restart your R session.

In this lab, we experiment with the estimation approaches discussed in the workshop. By the end of this lab, the goal is for you to be able to apply one of the Structural After Measurement (SAM) approaches to analyze your own data. We begin by reading and visualizing the data, followed by applying various SAM approaches:

- Using corrected summary statistics for the latent variables
  - local SAM
- Replacing the latent variables by scores
  - uncorrected factor score regression
  - Skrondal and Laake's method
  - correlation-preserving factor scores
  - sum scores
  - instrumental variables
- Other SAM approaches
  - single-indicator approach
  - global SAM

Afterwards, we compare the results of the regression parameters (from the structural model) obtained by the different approaches.

0.1. The European Consumer Satisfaction Index

The European Consumer Satisfaction Index (ECSI) is an economic metric used to assess customer satisfaction. It is based on the Swedish Customer Satisfaction Barometer (Fornell, 1992) and is rooted in well-established theories and frameworks in consumer behavior. The ECSI is designed to be applicable across various industries and has been widely used in research.

For our example, the data is derived from a subset of the ECSI model. The model of interest is presented in Figure 1 and consists of four factors, each measured by four or five observed variables:

- a *costumer expectation* factor measured by: exp1, exp2, exp3, exp4, and exp5
- a *perceived product quality* factor measured by: qual1, qual2, qual3, exp4, and qual5
- a *perceived costumer value* factor measured by: val1, val2, val3, val4
- a *costumer satisfaction* factor measured by: sat1, sat2, sat3, sat4

The full ECSI model also includes variables measuring corporate image, customer complaints, and customer loyalty; however, these are not relevant to our analysis.

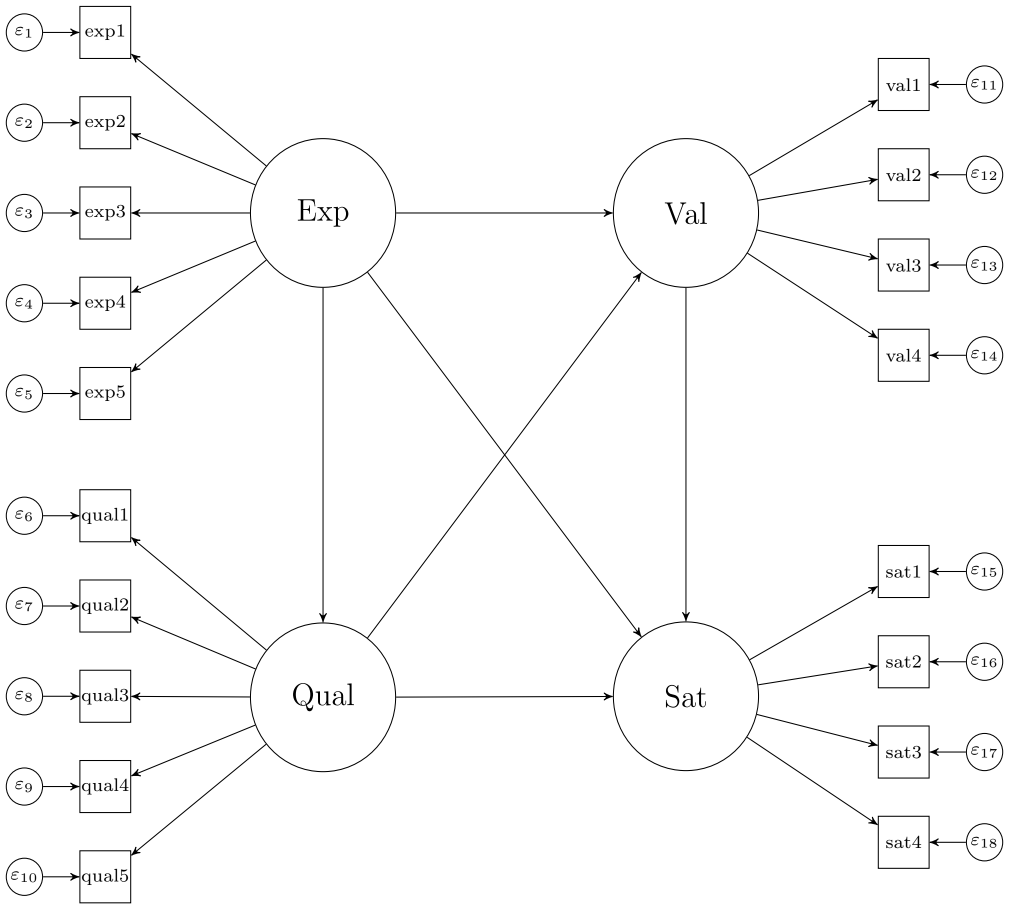

Figure 1: The hypthesized structural equation model.

0.2. Read in and inspect the data

Read in the data

We begin by reading in the csv file and reviewing an overview of the dataset.

```r
dat <- read.csv("ECSI.csv")
str(dat)
```

Visualizing the data (optional)

It is good practice to visualize the data before analyzing it. When fitting a structural equation model, it is important to examine the distribution of the data and the correlations between the observed variables. Although installing the `ggpplot2` package may take some time, the pairs plot it generates is more informative than base R's `pairs` function and provides most of the necessary information.

```r
# install.packages{'ggplot2'}
library(ggplot2)
# install.packages{'GGally'}
library(GGally)
ggpairs(dat[,1:10]) # for expectation and quality
ggpairs(dat[,11:18]) # for value and satisfaction
```

The pairs plots indicate that the observed variables do not exhibit substantial deviations from normality and show medium to high correlations between variables measuring the same factor.

1. Using corrected summary statistics for the latent variables

1.1. Local structural after measurement approach

We start with specifying the model syntax.

```r
# specify the full model
full.model <- '
# measurement model
expectation =~ exp1 + exp2 + exp3 + exp4 + exp5
quality =~ qual1 + qual2 + qual3 + qual4 + qual5
value =~ val1 + val2 + val3 + val4
satisfaction =~ sat1 + sat2 + sat3 + sat4

# structural model
satisfaction ~ value + quality + expectation
value ~ quality + expectation
quality ~ expectation
'
```

After specifying the model, we proceed by fitting the model and exploring the results.

```r
# fit the model
fit.lsam <- sam(full.model, data = dat, sam.method = 'local')

# explore and save the results
summary(fit.lsam)
```

```
## This is lavaan 0.6-20.2278 -- using the SAM approach to SEM
##
##   SAM method                                       LOCAL
##   Mapping matrix M method                             ML
##   Number of measurement blocks                         4
##   Estimator measurement part                          ML
##   Estimator  structural part                          ML
##
##   Number of observations                             150
##
## Summary Information Measurement + Structural:
##
##   Block       Latent Nind Chisq Df
##       1  expectation    5 4.287  5
##       2      quality    5 2.687  5
##       3        value    4 2.383  2
##       4 satisfaction    4 1.378  2
##
```

```
##   Model-based reliability latent variables:
##
##   expectation quality value satisfaction
##         0.908   0.919 0.918        0.795
##
##   Summary Information Structural part:
##
##   chisq df cfi rmsea srmr
##       0  0   1     0    0
##
## Parameter Estimates:
##
##   Standard errors                             Twostep
##   Information                                Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   satisfaction ~
##     value             0.446    0.107    4.178    0.000
##     quality           0.388    0.084    4.616    0.000
##     expectation       0.127    0.084    1.519    0.129
##   value ~
##     quality           0.114    0.073    1.564    0.118
##     expectation       0.443    0.079    5.570    0.000
##   quality ~
##     expectation       0.324    0.087    3.724    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     expectation       1.286    0.209    6.151    0.000
##    .quality           1.125    0.169    6.650    0.000
##    .value             0.686    0.117    5.855    0.000
##    .satisfaction      0.428    0.127    3.368    0.001
```

```r
PE.lsam <- parameterEstimates(fit.lsam)
```

2. Replacing the latent variables by scores

2.1. Uncorrected factor score regression

We have already specified the model, so we can immediately fit the model using the `sam` function, but now using `sam.method = 'fsr'` and `se = 'naive'`.

```r
# fit the model
fit.ufsr <- sam(full.model, data = dat, sam.method = 'fsr', se = 'naive')

# explore and save the results
summary(fit.ufsr)
```

```
## This is lavaan 0.6-20.2278 -- using the SAM approach to SEM
##
##   SAM method                                         FSR
##   Number of measurement blocks                         4
##   Estimator measurement part                          ML
##   Estimator  structural part                          ML
##
##   Number of observations                             150
##
## Summary Information Measurement + Structural:
##
##   Block       Latent Nind Chisq Df
##       1  expectation    5 4.287  5
##       2      quality    5 2.687  5
##       3        value    4 2.383  2
##       4 satisfaction    4 1.378  2
##
## Parameter Estimates:
##
##   Standard errors                                  Naive
##   Information                                   Expected
##   Information saturated (h1) model            Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   satisfaction ~
##     value             0.411    0.078    5.248    0.000
##     quality           0.361    0.063    5.704    0.000
##     expectation       0.138    0.069    2.001    0.045
##   value ~
##     quality           0.118    0.065    1.802    0.072
##     expectation       0.401    0.064    6.240    0.000
##   quality ~
##     expectation       0.294    0.077    3.839    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     expectation       1.416    0.164    8.660    0.000
##    .quality           1.249    0.144    8.660    0.000
##    .value             0.799    0.092    8.660    0.000
##    .satisfaction      0.736    0.085    8.660    0.000
```

```
PE.ufsr <- parameterEstimates(fit.ufsr)
```

2.2. Skrondal and Laake's method

Skrondal and Laake's method may require additional coding as the model becomes more complex. This is because a separate analysis must be conducted for each endogenous latent variable. The process involves computing Bartlett factor scores for the dependent variable(s) and regression factor scores for the independent variable(s). Once these factor scores are obtained, they can be used in a linear regression or path analysis.

We begin with the analysis for `quality`.

```
# independent variable
model.qual.indep <- 'expectation =~ exp1 + exp2 + exp3 + exp4 + exp5'
fit.qual.indep <- cfa(model.qual.indep, data = dat)
SL.qual.indep <- lavPredict(fit.qual.indep, method = "regression")

# dependent variable
model.qual.dep <- 'quality =~ qual1 + qual2 + qual3 + qual4 + qual5'
fit.qual.dep <- cfa(model.qual.dep, data = dat)
SL.qual.dep <- lavPredict(fit.qual.dep, method = "Bartlett")

# save factor scores in data frame
FS.qual <- as.data.frame(cbind(SL.qual.dep, SL.qual.indep))

# subsequent analysis
model.qual <- 'quality ~ expectation'
fit.qual <- sem(model.qual, data = FS.qual)
summary(fit.qual)
```

```
## lavaan 0.6-20.2278 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         2
##
##   Number of observations                           150
```

```
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   quality ~
##     expectation       0.324    0.084    3.839    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .quality           1.249    0.144    8.660    0.000
```

```
PE.qual <- parameterEstimates(fit.qual)

# OR simply with linear regression
fit.qual.lm <- lm(quality ~ expectation, data = FS.qual)
round(coef(fit.qual.lm), 3)
```

```
## (Intercept) expectation
##       0.000       0.324
```

Next, the analysis for `value`.

```
# independent variable
model.val.indep <- 'expectation =~ exp1 + exp2 + exp3 + exp4 + exp5
quality =~ qual1 + qual2 + qual3 + qual4 + qual5'
fit.val.indep <- cfa(model.val.indep, data = dat)
SL.val.indep <- lavPredict(fit.val.indep, method = "regression")

# dependent variable
model.val.dep <- 'value =~ val1 + val2 + val3 + val4'
fit.val.dep <- cfa(model.val.dep, data = dat)
SL.val.dep <- lavPredict(fit.val.dep, method = "Bartlett")

# save factor scores in data frame
FS.val <- as.data.frame(cbind(SL.val.dep, SL.val.indep))
```

```r
# subsequent analysis
model.val <- 'value ~ quality + expectation'
fit.val <- sem(model.val, data = FS.val)
summary(fit.val)
```

```
## lavaan 0.6-20.2278 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         3
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   value ~
##     quality           0.114    0.073    1.576    0.115
##     expectation       0.441    0.072    6.100    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .value             0.800    0.092    8.660    0.000
```

```r
PE.val <- parameterEstimates(fit.val)

# OR simply with linear regression
fit.val <- lm(value ~ quality + expectation, data = FS.val)
round(coef(fit.val), 3)
```

```
## (Intercept)     quality expectation
##       0.000       0.114       0.441
```

Finally, the analysis for `satisfaction`.

```r
model.sat.indep <- 'expectation =~ exp1 + exp2 + exp3 + exp4 + exp5
quality =~ qual1 + qual2 + qual3 + qual4 + qual5
value =~ val1 + val2 + val3 + val4'
fit.sat.indep <- cfa(model.sat.indep, data = dat)
SL.sat.indep <- lavPredict(fit.sat.indep, method = "regression")

# dependent variable
model.sat.dep <- 'satisfaction =~ sat1 + sat2 + sat3 + sat4'
fit.sat.dep <- cfa(model.sat.dep, data = dat)
SL.sat.dep <- lavPredict(fit.sat.dep, method = "Bartlett")

# save factor scores in data frame
FS.sat <- as.data.frame(cbind(SL.sat.dep, SL.sat.indep))

# subsequent analysis
model.sat <- 'satisfaction ~ value + quality + expectation'
fit.sat <- sem(model.sat, data = FS.sat)
summary(fit.sat)
```

```
## lavaan 0.6-20.2278 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         4
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   satisfaction ~
##     value             0.442    0.091    4.844    0.000
##     quality           0.387    0.070    5.508    0.000
##     expectation       0.126    0.083    1.508    0.132
##
## Variances:
##                      Estimate  Std.Err  z-value  P(>|z|)
##    .satisfaction      0.735    0.085    8.660    0.000
```

```r
PE.sat <- parameterEstimates(fit.sat)

# OR simply with linear regression
fit.sat <- lm(satisfaction ~ value + quality + expectation, data = FS.sat)
round(coef(fit.sat), 3)
```

```
## (Intercept)       value     quality expectation
##       0.000       0.442       0.387       0.126
```

We now combine the results to include them in the final comparison.

```r
PE.slm <- rbind(PE.sat, PE.val, PE.qual)
```

2.3. Correlation-preserving factor scores

Using correlation-preserving factor scores also requires additional coding.

First, we specify the measurement model.

```r
meas.model <- '
expectation =~ exp1 + exp2 + exp3 + exp4 + exp5
quality =~ qual1 + qual2 + qual3 + qual4 + qual5
value =~ val1 + val2 + val3 + val4
satisfaction =~ sat1 + sat2 + sat3 + sat4
'
```

Next, we fit we the measurement model and calculate the correlation-preserving factor scores. We save the factor scores in a data frame.

```r
# fit the model and calculate correlation-preserving factor scores
fit.cfa <- sam(meas.model, data = dat, estimator = "ML")
fs.cpreg <- lavPredict(fit.cfa, method = "regression", transform = TRUE)
FS.cpreg <- as.data.frame(fs.cpreg)
```

We then specify the structural model and fit it using the correlation-preserving factor scores

```r
str.model <- '
satisfaction ~ value + quality + expectation
value ~ quality + expectation
quality ~ expectation
'

# fit the model using correlation-preserving factor scores
```

```r
fit.cpfs <- sem(str.model, data = FS.cpreg)

# explore and save the results
summary(fit.cpfs)
```

```
## lavaan 0.6-20.2278 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   satisfaction ~
##     value             0.446    0.064    6.915    0.000
##     quality           0.388    0.051    7.623    0.000
##     expectation       0.127    0.057    2.215    0.027
##   value ~
##     quality           0.114    0.064    1.792    0.073
##     expectation       0.443    0.063    7.016    0.000
##   quality ~
##     expectation       0.324    0.076    4.244    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .satisfaction      0.428    0.049    8.660    0.000
##    .value             0.686    0.079    8.660    0.000
##    .quality           1.125    0.130    8.660    0.000
```

```
PE.cpfs <- parameterEstimates(fit.cpfs)
```

Note that we could also perform a series of linear regressions with the correlation-preserving factor scores and obtain the same estimates for the regression coefficients.

```
# regression model with satisfaction as outcome
fit.sat <- lm(satisfaction ~ value + quality + expectation, data = FS.cpreg)
round(coef(fit.sat), 3)
```

```
## (Intercept)       value     quality expectation
##       0.000       0.446       0.388       0.127
```

```
# regression model with value as outcome
fit.val <- lm(value ~ quality + expectation, data = FS.cpreg)
round(coef(fit.val), 3)
```

```
## (Intercept)     quality expectation
##       0.000       0.114       0.443
```

```
# regression model with quality as outcome
fit.qual <- lm(quality ~ expectation, data = FS.cpreg)
round(coef(fit.qual), 3)
```

```
## (Intercept) expectation
##       0.000       0.324
```

2.4. Sum scores

First, we calculate the sum scores and save them in a data frame.

```
# calculate sum scores for the latent variables
expmean <- rowSums(dat[, c("exp1", "exp2", "exp3" ,"exp4","exp5")])/4
qualmean <- rowSums(dat[, c("qual1", "qual2", "qual3" ,"qual4","qual5")])/4
valmean <- rowSums(dat[, c("val1", "val2", "val3" ,"val4")])/4
satmean <- rowSums(dat[, c("sat1", "sat2", "sat3" ,"sat4")])/4

# save as data frame
dat.sum <- as.data.frame(cbind(expmean, qualmean, valmean, satmean))
```

Next, we specify the structural model (with adjusted names) and fit it using the sum scores.

```
str.model.sum <- '
satmean ~ valmean + qualmean + expmean
valmean ~ qualmean + expmean
qualmean ~ expmean
'

# fit the model using correlation-preserving factor scores
fit.sum <- sem(str.model.sum, data = dat.sum)
```

```r
# explore and save the results
summary(fit.sum)
```

```
## lavaan 0.6-20.2278 ended normally after 1 iteration
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         9
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   satmean ~
##     valmean           0.416    0.077    5.380    0.000
##     qualmean          0.244    0.052    4.664    0.000
##     expmean           0.115    0.056    2.032    0.042
##   valmean ~
##     qualmean          0.087    0.055    1.590    0.112
##     expmean           0.323    0.053    6.049    0.000
##   qualmean ~
##     expmean           0.295    0.076    3.895    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .satmean           0.664    0.077    8.660    0.000
##    .valmean           0.739    0.085    8.660    0.000
##    .qualmean          1.645    0.190    8.660    0.000
```

```r
PE.sum <- parameterEstimates(fit.sum)
```

As before, we could also perform a set of linear regressions.

2.5. Instrumental variables

This can be done using the `MIIVsem` package. First, we install and load the `MIIVsem` package. Next, we use the `miive` function to fit the model using the 2SLS estimator.

```
#install.packages("MIIVsem")
library(MIIVsem)
```

```
## This is MIIVsem 0.5.8

## MIIVsem is BETA software! Please report any bugs.
```

```
# fit the model and check results
fit.miive <- miive(full.model, data = dat)
summary(fit.miive)
```

```
## MIIVsem (0.5.8) results
##
## Number of observations                                           150
## Number of equations                                               17
## Estimator                                                  MIIV-2SLS
## Standard Errors                                             standard
## Missing                                                     listwise
##
##
## Parameter Estimates:
##
##
## STRUCTURAL COEFFICIENTS:
##                    Estimate  Std.Err  z-value  P(>|z|)   Sargan   df   P(Chi)
##   expectation =~
##     exp1              1.000
##     exp2              0.924    0.087   10.645    0.000   11.073   15    0.747
##     exp3              0.808    0.065   12.491    0.000   30.625   15    0.010
##     exp4              0.824    0.096    8.607    0.000   19.885   15    0.176
##     exp5              0.858    0.066   12.995    0.000   13.742   15    0.545
##   quality =~
##     qual1             1.000
##     qual2             0.827    0.082   10.051    0.000   18.005   15    0.262
##     qual3             0.856    0.099    8.653    0.000   18.352   15    0.245
##     qual4             0.854    0.056   15.143    0.000   17.490   15    0.290
##     qual5             0.773    0.061   12.717    0.000   16.292   15    0.363
##   satisfaction =~
##     sat1              1.000
##     sat2              0.724    0.118    6.122    0.000   22.050   15    0.106
##     sat3              0.802    0.104    7.707    0.000   20.215   15    0.164
##     sat4              0.578    0.088    6.555    0.000   21.381   15    0.125
##   value =~
##     val1              1.000
##     val2              0.886    0.062   14.364    0.000   13.425   15    0.569
##     val3              0.820    0.085    9.697    0.000   10.635   15    0.778
##     val4              0.916    0.075   12.136    0.000   14.089   15    0.519
##
##   quality ~
##     expectation       0.306    0.092    3.314    0.001    5.056    3    0.168
##   satisfaction ~
##     expectation       0.203    0.140    1.451    0.147    5.923    8    0.656
##     value             0.430    0.160    2.682    0.007
##     quality           0.365    0.124    2.938    0.003
##   value ~
##     expectation       0.504    0.090    5.631    0.000   12.370    6    0.054
##     quality           0.175    0.091    1.921    0.055
##
## INTERCEPTS:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     exp1              0.000
##     exp2              0.158    0.095    1.662    0.097
##     exp3              0.103    0.069    1.492    0.136
##     exp4             -0.008    0.106   -0.072    0.943
##     exp5              0.025    0.071    0.357    0.721
##     qual1             0.000
##     qual2            -0.034    0.089   -0.384    0.701
##     qual3             0.284    0.106    2.677    0.007
##     qual4             0.018    0.059    0.302    0.763
##     qual5             0.119    0.066    1.810    0.070
##     quality          -0.070    0.099   -0.709    0.478
##     sat1              0.000
##     sat2              0.030    0.117    0.254    0.799
##     sat3             -0.132    0.106   -1.247    0.212
##     sat4             -0.065    0.091   -0.711    0.477
##     satisfaction      0.047    0.123    0.383    0.702
##     val1              0.000
##     val2              0.044    0.057    0.760    0.447
##     val3              0.107    0.084    1.282    0.200
##     val4             -0.073    0.074   -0.991    0.322
##     value            -0.139    0.091   -1.525    0.127
```

3. Other approaches

3.1. Single-indicator approach (fixed reliability)

We start by calculating the sum scores (as done earlier) and set the residual variance using the fixed reliability approach (here we choose 0.8).

```
# calculate sum scores for the latent variables: see above

# fix residual variance
res.exp <- (1 - 0.8)*var(expmean)
res.qual <- (1 - 0.8)*var(qualmean)
res.val <- (1 - 0.8)*var(valmean)
res.sat <- (1 - 0.8)*var(satmean)
```

Next, we specify the measurement and structural models separately, fit the model using the `sem` function, and explore the results.

```
# create model syntax
meas.model.si.fixed <- c("sexp =~ 1*expmean", paste0("expmean ~~ ", res.exp, "*expmean"),
              "squal =~ 1*qualmean", paste0("qualmean ~~ ", res.qual, "*qualmean"),
              "sval =~ 1*valmean", paste0("valmean ~~ ", res.val, "*valmean"),
              "ssat =~ 1*satmean", paste0("satmean ~~ ", res.sat, "*satmean"))
str.model.si.fixed <- '
ssat ~ sval + squal + sexp
sval ~ squal + sexp
squal ~ sexp
'

fit.si.fixed <- sem(c(meas.model.si.fixed, str.model.si.fixed), data = dat.sum)
summary(fit.si.fixed)
```

```
## lavaan 0.6-20.2278 ended normally after 21 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        10
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
```

```
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   sexp =~
##     expmean           1.000
##   squal =~
##     qualmean          1.000
##   sval =~
##     valmean           1.000
##   ssat =~
##     satmean           1.000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   ssat ~
##     sval              0.534    0.120    4.435    0.000
##     squal             0.294    0.071    4.165    0.000
##     sexp              0.074    0.089    0.826    0.409
##   sval ~
##     squal             0.076    0.074    1.028    0.304
##     sexp              0.408    0.072    5.647    0.000
##   squal ~
##     sexp              0.370    0.096    3.870    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .expmean           0.384
##    .qualmean          0.365
##    .valmean           0.198
##    .satmean           0.228
##     sexp              1.522    0.220    6.917    0.000
##    .squal             1.239    0.191    6.478    0.000
##    .sval              0.488    0.087    5.621    0.000
##    .ssat              0.363    0.079    4.616    0.000
```

```
PE.si.fixed <- parameterEstimates(fit.si.fixed)
```

3.2. Single-indicator approach (model-based reliability)

We start by calculating Bartlett factor scores and estimate the reliability.

```r
# fit measurement models separately
fit.exp <- sem('expectation =~ exp1 + exp2 + exp3 + exp4 + exp5', data = dat)
fit.qual <- sem('quality =~ qual1 + qual2 + qual3 + qual4 + qual5',  data = dat)
fit.val <- sem('value =~ val1 + val2 + val3 + val4',  data = dat)
fit.sat <- sem('satisfaction =~ sat1 + sat2 + sat3 + sat4',  data = dat)

# calculate Bartlett factor scores for the latent variables
fs.exp <- lavPredict(fit.exp, method = "Bartlett", fsm = TRUE)
fs.qual <- lavPredict(fit.qual, method = "Bartlett", fsm = TRUE)
fs.val <- lavPredict(fit.val, method = "Bartlett", fsm = TRUE)
fs.sat <- lavPredict(fit.sat, method = "Bartlett", fsm = TRUE)
dat.fs <- data.frame(fs.exp, fs.qual, fs.val, fs.sat)

# estimate residual variance
A <- attr(fs.exp, "fsm")[[1]]; THETA <- lavInspect(fit.exp, "est")$theta
res.exp <- drop(A %*% THETA %*% t(A))
A <- attr(fs.qual, "fsm")[[1]]; THETA <- lavInspect(fit.qual, "est")$theta
res.qual <- drop(A %*% THETA %*% t(A))
A <- attr(fs.val, "fsm")[[1]]; THETA <- lavInspect(fit.val, "est")$theta
res.val <- drop(A %*% THETA %*% t(A))
A <- attr(fs.sat, "fsm")[[1]]; THETA <- lavInspect(fit.sat, "est")$theta
res.sat <- drop(A %*% THETA %*% t(A))
```

Next, we specify the measurement and structural models separately, fit the model using the sem function, and explore the results.

```r
# create model syntax
meas.model.si.modrel <- c("fexp =~ 1*expectation", paste0("expectation ~~ ", res.exp, "*expectation"),
              "fqual =~ 1*quality", paste0("quality ~~ ", res.qual, "*quality"),
              "fval =~ 1*value", paste0("value ~~ ", res.val, "*value"),
              "fsat =~ 1*satisfaction", paste0("satisfaction ~~ ", res.sat, "*satisfaction"))
str.model.si.modrel <- '
fsat ~ fval + fqual + fexp
fval ~ fqual + fexp
fqual ~ fexp
'

fit.si.modrel <- sem(c(meas.model.si.modrel, str.model.si.modrel), data = dat.fs)
summary(fit.si.modrel)
```

```
## lavaan 0.6-20.2278 ended normally after 23 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                        10
##
##   Number of observations                           150
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   fexp =~
##     expectation       1.000
##   fqual =~
##     quality           1.000
##   fval =~
##     value             1.000
##   fsat =~
##     satisfaction      1.000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   fsat ~
##     fval              0.446    0.092    4.867    0.000
##     fqual             0.388    0.070    5.504    0.000
##     fexp              0.127    0.082    1.548    0.122
##   fval ~
##     fqual             0.114    0.073    1.573    0.116
##     fexp              0.443    0.072    6.114    0.000
##   fqual ~
##     fexp              0.324    0.085    3.835    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .expectation       0.130
##    .quality           0.111
##    .value             0.088
##    .satisfaction      0.274
##     fexp              1.286    0.164    7.864    0.000
##    .fqual             1.125    0.144    7.793    0.000
##    .fval              0.686    0.092    7.412    0.000
##    .fsat              0.428    0.085    5.020    0.000
```

```
PE.si.modrel <- parameterEstimates(fit.si.modrel)
```

3.3. Global structural after measurement approach

We already specified the full model syntax earlier. We can use the global SAM by setting `sam.method = 'global'`. This should yield identical results to those of local SAM if the model is correctly specified.

```
# fit the model
fit.gsam <- sam(full.model, data = dat, sam.method = 'global')

# explore and save the results
parameterEstimates(fit.gsam)
```

```
##             lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 19 satisfaction  ~       value 0.446 0.107 4.178  0.000    0.237    0.655
## 20 satisfaction  ~     quality 0.388 0.084 4.616  0.000    0.223    0.553
## 21 satisfaction  ~  expectation 0.127 0.084 1.519  0.129   -0.037    0.291
## 22        value  ~     quality 0.114 0.073 1.564  0.118   -0.029    0.257
## 23        value  ~  expectation 0.443 0.079 5.570  0.000    0.287    0.598
## 24      quality  ~  expectation 0.324 0.087 3.724  0.000    0.154    0.495
## 43  expectation ~~  expectation 1.286 0.209 6.151  0.000    0.876    1.696
## 44      quality ~~     quality 1.125 0.169 6.650  0.000    0.793    1.456
## 45        value ~~       value 0.686 0.117 5.855  0.000    0.456    0.915
## 46 satisfaction ~~ satisfaction 0.428 0.127 3.368  0.001    0.179    0.677
```

```
PE.gsam <- parameterEstimates(fit.gsam)
```

4. Comparison of results

Now that we have used the SAM approaches and saved the results, we can compare the regression parameter estimates of the structural model.

```r
# check and compare results for regression coefficients of structural model
PE.lsam[PE.lsam$op == "~",]
```

```
##             lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 19 satisfaction  ~       value 0.446 0.107 4.178  0.000    0.237    0.655
## 20 satisfaction  ~     quality 0.388 0.084 4.616  0.000    0.223    0.553
## 21 satisfaction  ~ expectation 0.127 0.084 1.519  0.129   -0.037    0.291
## 22        value  ~     quality 0.114 0.073 1.564  0.118   -0.029    0.257
## 23        value  ~ expectation 0.443 0.079 5.570  0.000    0.287    0.598
## 24      quality  ~ expectation 0.324 0.087 3.724  0.000    0.154    0.495
```

```r
PE.ufsr[PE.ufsr$op == "~",]
```

```
##             lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 19 satisfaction  ~       value 0.411 0.078 5.248  0.000    0.258    0.565
## 20 satisfaction  ~     quality 0.361 0.063 5.704  0.000    0.237    0.485
## 21 satisfaction  ~ expectation 0.138 0.069 2.001  0.045    0.003    0.274
## 22        value  ~     quality 0.118 0.065 1.802  0.072   -0.010    0.246
## 23        value  ~ expectation 0.401 0.064 6.240  0.000    0.275    0.527
## 24      quality  ~ expectation 0.294 0.077 3.839  0.000    0.144    0.445
```

```r
PE.slm[PE.slm$op == "~",]
```

```
##             lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 1  satisfaction  ~       value 0.442 0.091 4.844  0.000    0.263    0.620
## 2  satisfaction  ~     quality 0.387 0.070 5.508  0.000    0.249    0.525
## 3  satisfaction  ~ expectation 0.126 0.083 1.508  0.132   -0.038    0.289
## 11        value  ~     quality 0.114 0.073 1.576  0.115   -0.028    0.257
## 12        value  ~ expectation 0.441 0.072 6.100  0.000    0.300    0.583
## 17      quality  ~ expectation 0.324 0.084 3.839  0.000    0.159    0.490
```

```r
PE.cpfs[PE.cpfs$op == "~",]
```

```
##            lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 1 satisfaction  ~       value 0.446 0.064 6.915  0.000    0.320    0.572
## 2 satisfaction  ~     quality 0.388 0.051 7.623  0.000    0.288    0.488
## 3 satisfaction  ~ expectation 0.127 0.057 2.215  0.027    0.015    0.240
## 4        value  ~     quality 0.114 0.064 1.792  0.073   -0.011    0.239
## 5        value  ~ expectation 0.443 0.063 7.016  0.000    0.319    0.566
## 6      quality  ~ expectation 0.324 0.076 4.244  0.000    0.174    0.474
```

```r
PE.sum[PE.sum$op == "~",]
```

```
##        lhs op      rhs   est    se     z pvalue ci.lower ci.upper
## 1  satmean  ~  valmean 0.416 0.077 5.380  0.000    0.265    0.568
## 2  satmean  ~ qualmean 0.244 0.052 4.664  0.000    0.141    0.346
## 3  satmean  ~  expmean 0.115 0.056 2.032  0.042    0.004    0.225
## 4  valmean  ~ qualmean 0.087 0.055 1.590  0.112   -0.020    0.194
## 5  valmean  ~  expmean 0.323 0.053 6.049  0.000    0.218    0.427
## 6 qualmean  ~  expmean 0.295 0.076 3.895  0.000    0.147    0.444
```

```
PE.si.fixed[PE.si.fixed$op == "~",]
```

```
##      lhs op   rhs   est    se     z pvalue ci.lower ci.upper
## 9   ssat  ~  sval 0.534 0.120 4.435  0.000    0.298    0.771
## 10  ssat  ~ squal 0.294 0.071 4.165  0.000    0.155    0.432
## 11  ssat  ~  sexp 0.074 0.089 0.826  0.409   -0.101    0.248
## 12  sval  ~ squal 0.076 0.074 1.028  0.304   -0.069    0.220
## 13  sval  ~  sexp 0.408 0.072 5.647  0.000    0.267    0.550
## 14 squal  ~  sexp 0.370 0.096 3.870  0.000    0.183    0.557
```

```
PE.si.modrel[PE.si.modrel$op == "~",]
```

```
##      lhs op   rhs   est    se     z pvalue ci.lower ci.upper
## 9   fsat  ~  fval 0.446 0.092 4.867  0.000    0.266    0.626
## 10  fsat  ~ fqual 0.388 0.070 5.504  0.000    0.250    0.526
## 11  fsat  ~  fexp 0.127 0.082 1.548  0.122   -0.034    0.288
## 12  fval  ~ fqual 0.114 0.073 1.573  0.116   -0.028    0.257
## 13  fval  ~  fexp 0.443 0.072 6.114  0.000    0.301    0.585
## 14 fqual  ~  fexp 0.324 0.085 3.835  0.000    0.158    0.490
```

```
PE.gsam[PE.gsam$op == "~",]
```

```
##             lhs op         rhs   est    se     z pvalue ci.lower ci.upper
## 19 satisfaction  ~       value 0.446 0.107 4.178  0.000    0.237    0.655
## 20 satisfaction  ~     quality 0.388 0.084 4.616  0.000    0.223    0.553
## 21 satisfaction  ~ expectation 0.127 0.084 1.519  0.129   -0.037    0.291
## 22        value  ~     quality 0.114 0.073 1.564  0.118   -0.029    0.257
## 23        value  ~ expectation 0.443 0.079 5.570  0.000    0.287    0.598
## 24      quality  ~ expectation 0.324 0.087 3.724  0.000    0.154    0.495
```

The employed SAM approaches yield different results for both the coefficient estimates and the significance of some coefficients. Local and global SAM produce identical results. Skrondal and Laake's method yields results that are close to those of local SAM. Correlation-preserving factor scores yield identical results to those of the local SAM regarding the coefficient estimates, but they differ in terms of the significance of the paths. Specifically, the effect from expectation to satisfaction is significant when using correlation-preserving factor scores. This highlights the importance of using a corrected standard error that accounts for measurement error.

Uncorrected factor score regression and sum scoring produce coefficient estimates that deviate more and lead to different conclusions regarding the significance of the effect from expectation to satisfaction, compared to local SAM. This difference is due to the impact of measurement error. The single-indicator with fixed reliability approach also results in different coefficient estimates but does not show any difference in the significance of the examined effects. The single-indicator approach with model-based reliability provides coefficient estimates identical to those of local SAM and even yields nearly identical results for the estimated standard errors.

Reference list

Fornell, C. (1992). A national customer satisfaction barometer: The Swedish experience. *Journal of Marketing, 56*(1), 6–21.