ASCI and Structural Equation Model

Yerim Chung and Geonsik Yu

2017 7 25

1. Load libraries and the survey dataset.

- http://www2.gsu.edu/~mkteer/sem2.html#loading
- http://lavaan.ugent.be/index.html

```
if (!require("lavaan", quietly = TRUE)){install.packages("lavaan")}
```

```
## This is lavaan 0.5-23.1097
```

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

```
library(lavaan);
library(readr);
library(psych);
data_directory <- "~/Desktop/r_workspace/data-test5.csv";
data <- suppressMessages( read_csv(data_directory, progress = FALSE)[1:198,] );
data$a10 <- 10 - data$a10;   ## conformation = 10 - disconformation
```

2. Define a structural equation model and conduct fitting.

- Some of resulting standard errors are negaive (Heywood cases).
- In our model, we should check the variance estimate of eta-1 (-0.12).
- http://zencaroline.blogspot.kr/2007/05/heywood-cases-negative-error-variances.html

```
model <- '
# measurement model
  ksi =~ a1 + a2 + a3
  eta1 =~ a4 + a5 + a6
  eta2 =~ a7 + a8
  eta3 =~ a9 + a10 + a11
  eta4 =~ a15
  eta5 =~ a12 + a13 + a14
# regressions
  eta1 ~ ksi
  eta2 ~ ksi + eta1
  eta3 ~ ksi + eta1 + eta2
  eta4 ~ eta3
  eta5 ~ eta3 + eta4
';
fit <- sem(model, data = data);
```

```
## Warning in lav_object_post_check(object): lavaan WARNING: some estimated lv
## variances are negative
```

3. Print output.

```r
summary(fit, standardized=TRUE);
```

```
## lavaan (0.5-23.1097) converged normally after 105 iterations
##
##   Number of observations                           198
##
##   Estimator                                         ML
##   Minimum Function Test Statistic              418.023
##   Degrees of freedom                                82
##   P-value (Chi-square)                           0.000
##
## Parameter Estimates:
##
##   Information                                 Expected
##   Standard Errors                             Standard
##
## Latent Variables:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   ksi =~
##     a1                1.000                               0.278    0.200
##     a2                3.115    1.096    2.842    0.004    0.867    0.766
##     a3                3.007    1.054    2.854    0.004    0.837    0.818
##   eta1 =~
##     a4                1.000                               0.953    0.865
##     a5                0.469    0.075    6.283    0.000    0.447    0.428
##     a6                0.948    0.064   14.903    0.000    0.904    0.814
##   eta2 =~
##     a7                1.000                               0.762    0.756
##     a8                1.236    0.108   11.492    0.000    0.942    0.772
##   eta3 =~
##     a9                1.000                               0.940    0.902
##     a10               0.279    0.076    3.668    0.000    0.262    0.262
##     a11               1.071    0.063   17.019    0.000    1.007    0.852
##   eta4 =~
##     a15               1.000                               1.229    1.000
##   eta5 =~
##     a12               1.000                               0.985    0.881
##     a13               0.701    0.093    7.541    0.000    0.691    0.507
##     a14               1.033    0.097   10.666    0.000    1.018    0.663
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##   eta1 ~
##     ksi               3.644    1.279    2.849    0.004    1.064    1.064
##   eta2 ~
##     ksi              -0.217    0.919   -0.236    0.813   -0.079   -0.079
##     eta1              0.848    0.268    3.165    0.002    1.060    1.060
##   eta3 ~
##     ksi               1.733    1.031    1.681    0.093    0.513    0.513
##     eta1             -0.286    1.021   -0.280    0.779   -0.290   -0.290
##     eta2              0.878    1.330    0.660    0.509    0.712    0.712
##   eta4 ~
```

```
##     eta3              -0.845    0.080  -10.535    0.000   -0.646   -0.646
##   eta5 ~
##     eta3               0.821    0.068   12.029    0.000    0.783    0.783
##     eta4              -0.204    0.045   -4.516    0.000   -0.254   -0.254
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)   Std.lv  Std.all
##    .a1                1.869    0.188    9.953    0.000    1.869    0.960
##    .a2                0.529    0.061    8.661    0.000    0.529    0.413
##    .a3                0.347    0.045    7.754    0.000    0.347    0.331
##    .a4                0.306    0.042    7.327    0.000    0.306    0.252
##    .a5                0.895    0.091    9.848    0.000    0.895    0.817
##    .a6                0.417    0.049    8.478    0.000    0.417    0.338
##    .a7                0.435    0.054    8.078    0.000    0.435    0.428
##    .a8                0.601    0.077    7.784    0.000    0.601    0.404
##    .a9                0.202    0.030    6.819    0.000    0.202    0.186
##    .a10               0.928    0.094    9.900    0.000    0.928    0.931
##    .a11               0.384    0.047    8.104    0.000    0.384    0.275
##    .a15               0.000                               0.000    0.000
##    .a12               0.281    0.052    5.406    0.000    0.281    0.224
##    .a13               1.378    0.143    9.661    0.000    1.378    0.743
##    .a14               1.320    0.143    9.245    0.000    1.320    0.560
##     ksi               0.077    0.054    1.424    0.154    1.000    1.000
##    .eta1             -0.120    0.050   -2.407    0.016   -0.132   -0.132
##    .eta2              0.028    0.044    0.640    0.522    0.048    0.048
##    .eta3              0.088    0.046    1.896    0.058    0.099    0.099
##    .eta4              0.879    0.094    9.376    0.000    0.582    0.582
##    .eta5              0.064    0.045    1.424    0.154    0.066    0.066
```

4. Detailed outputs.

```r
# Unstandardized solution matrix (factor loading, or lambda).
inspect(fit,what="est")$lambda
```

```
##       ksi  eta1  eta2  eta3 eta4  eta5
## a1  1.000 0.000 0.000 0.000    0 0.000
## a2  3.115 0.000 0.000 0.000    0 0.000
## a3  3.007 0.000 0.000 0.000    0 0.000
## a4  0.000 1.000 0.000 0.000    0 0.000
## a5  0.000 0.469 0.000 0.000    0 0.000
## a6  0.000 0.948 0.000 0.000    0 0.000
## a7  0.000 0.000 1.000 0.000    0 0.000
## a8  0.000 0.000 1.236 0.000    0 0.000
## a9  0.000 0.000 0.000 1.000    0 0.000
## a10 0.000 0.000 0.000 0.279    0 0.000
## a11 0.000 0.000 0.000 1.071    0 0.000
## a15 0.000 0.000 0.000 0.000    1 0.000
## a12 0.000 0.000 0.000 0.000    0 1.000
## a13 0.000 0.000 0.000 0.000    0 0.701
## a14 0.000 0.000 0.000 0.000    0 1.033
```

```r
# Standardized solution matrix (factor loading, or lambda).
inspect(fit,what="std")$lambda
```

```
##       ksi  eta1  eta2  eta3 eta4  eta5
## a1  0.200 0.000 0.000 0.000    0 0.000
## a2  0.766 0.000 0.000 0.000    0 0.000
## a3  0.818 0.000 0.000 0.000    0 0.000
## a4  0.000 0.865 0.000 0.000    0 0.000
## a5  0.000 0.428 0.000 0.000    0 0.000
## a6  0.000 0.814 0.000 0.000    0 0.000
## a7  0.000 0.000 0.756 0.000    0 0.000
## a8  0.000 0.000 0.772 0.000    0 0.000
## a9  0.000 0.000 0.000 0.902    0 0.000
## a10 0.000 0.000 0.000 0.262    0 0.000
## a11 0.000 0.000 0.000 0.852    0 0.000
## a15 0.000 0.000 0.000 0.000    1 0.000
## a12 0.000 0.000 0.000 0.000    0 0.881
## a13 0.000 0.000 0.000 0.000    0 0.507
## a14 0.000 0.000 0.000 0.000    0 0.663
```

5. Goodness of fit of the model.

```r
fitmeasures(fit)
```

```
##                npar                fmin               chisq
##              38.000               1.056             418.023
##                  df              pvalue      baseline.chisq
##              82.000               0.000            2180.067
##         baseline.df     baseline.pvalue                 cfi
##             105.000               0.000               0.838
##                 tli                nnfi                 rfi
##               0.793               0.793               0.754
##                 nfi                pnfi                 ifi
##               0.808               0.631               0.840
##                 rni                logl   unrestricted.logl
##               0.838           -3768.288           -3559.276
##                 aic                 bic              ntotal
##            7612.575            7737.529             198.000
##                bic2               rmsea      rmsea.ci.lower
##            7617.145               0.144               0.130
##      rmsea.ci.upper        rmsea.pvalue                 rmr
##               0.158               0.000               0.137
##          rmr_nomean                srmr        srmr_bentler
##               0.137               0.102               0.102
## srmr_bentler_nomean         srmr_bollen  srmr_bollen_nomean
##               0.102               0.102               0.102
##          srmr_mplus   srmr_mplus_nomean               cn_05
##               0.102               0.102              50.326
##               cn_01                 gfi                agfi
##              55.326               0.785               0.685
##                pgfi                 mfi                ecvi
##               0.536               0.428               2.495
```

6. ASCI (simplified fomula).

```r
# Load weighted of eta-3.
numerator <- sum(inspect(fit,what="est")$lambda[9:11,4] * colMeans(data[,11:13])) - sum(inspect(fit,wha
denominator <- sum(inspect(fit,what="est")$lambda[9:11,4]) * 9

ASCI <- (numerator / denominator) *100
ASCI
```

```
## [1] 62.91129
```

7. Cronbach's alpha.

```r
alpha(data[,3:5], check.keys =  TRUE);
```

```
##
## Reliability analysis
## Call: alpha(x = data[, 3:5], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
##       0.49      0.54    0.54      0.28 1.2 0.066  6.5 0.84
##
##  lower alpha upper     95% confidence boundaries
## 0.37 0.49 0.62
##
##  Reliability if an item is dropped:
##    raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## a1      0.78      0.78   0.636     0.636 3.50    0.032
## a2      0.21      0.22   0.126     0.126 0.29    0.107
## a3      0.15      0.16   0.085     0.085 0.18    0.118
##
##  Item statistics
##      n raw.r std.r r.cor r.drop mean  sd
## a1 198  0.64  0.56  0.14   0.11  6.5 1.4
## a2 198  0.75  0.79  0.71   0.42  6.4 1.1
## a3 198  0.76  0.81  0.74   0.48  6.5 1.0
##
## Non missing response frequency for each item
##       3    4    5    6    7    8    9   10 miss
## a1 0.02 0.07 0.17 0.24 0.24 0.23 0.05 0.00    0
## a2 0.02 0.03 0.18 0.26 0.37 0.14 0.01 0.00    0
## a3 0.01 0.02 0.14 0.30 0.40 0.13 0.01 0.01    0
```

```r
alpha(data[,6:8], check.keys =  TRUE);
```

```
##
## Reliability analysis
## Call: alpha(x = data[, 6:8], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
##       0.71      0.71    0.68      0.45 2.5 0.036  6.5 0.87
##
##  lower alpha upper     95% confidence boundaries
```

```
## 0.64 0.71 0.78
##
##  Reliability if an item is dropped:
##    raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## a4      0.49      0.49    0.32      0.32 0.95    0.073
## a5      0.84      0.84    0.72      0.72 5.18    0.023
## a6      0.47      0.47    0.31      0.31 0.89    0.075
##
##  Item statistics
##      n raw.r std.r r.cor r.drop mean  sd
## a4 198  0.86  0.85  0.79   0.64  6.3 1.1
## a5 198  0.67  0.68  0.38   0.34  6.8 1.0
## a6 198  0.86  0.86  0.80   0.65  6.3 1.1
##
## Non missing response frequency for each item
##       3    4    5    6    7    8    9 miss
## a4 0.01 0.03 0.22 0.26 0.35 0.13 0.01    0
## a5 0.00 0.02 0.11 0.23 0.41 0.20 0.04    0
## a6 0.01 0.04 0.20 0.29 0.32 0.13 0.01    0
```

```r
#alpha(data[,9:10], check.keys =  TRUE);
```

```r
alpha(data[,11:13], check.keys =  TRUE);
```

```
##
## Reliability analysis
## Call: alpha(x = data[, 11:13], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean   sd
##       0.66      0.65    0.67      0.38 1.8 0.042  6.7 0.83
##
##  lower alpha upper     95% confidence boundaries
## 0.57 0.66 0.74
##
##  Reliability if an item is dropped:
##     raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## a9       0.31      0.31    0.19      0.19 0.45    0.097
## a10      0.87      0.87    0.77      0.77 6.79    0.018
## a11      0.30      0.30    0.18      0.18 0.44    0.099
##
##  Item statistics
##       n raw.r std.r r.cor r.drop mean  sd
## a9  198  0.86  0.85  0.83   0.65  6.8 1.0
## a10 198  0.56  0.59  0.21   0.19  6.9 1.0
## a11 198  0.87  0.85  0.83   0.63  6.4 1.2
##
## Non missing response frequency for each item
##        3    4    5    6    7    8    9 miss
## a9  0.01 0.01 0.11 0.20 0.40 0.26 0.02    0
## a10 0.00 0.01 0.08 0.20 0.43 0.24 0.04    0
## a11 0.01 0.06 0.14 0.24 0.36 0.18 0.01    0
```

```r
alpha(data[,14:16], check.keys =  TRUE);
```

```
##
## Reliability analysis
```

```
## Call: alpha(x = data[, 14:16], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase mean  sd
##       0.73      0.74    0.66      0.49 2.9 0.032  5.2 1.1
##
##  lower alpha upper     95% confidence boundaries
## 0.67 0.73 0.8
##
##  Reliability if an item is dropped:
##     raw_alpha std.alpha G6(smc) average_r S/N alpha se
## a12      0.62      0.62    0.45      0.45 1.7    0.053
## a13      0.68      0.70    0.54      0.54 2.4    0.042
## a14      0.64      0.65    0.48      0.48 1.9    0.050
##
##  Item statistics
##       n raw.r std.r r.cor r.drop mean  sd
## a12 198  0.80  0.83  0.70   0.60  6.4 1.1
## a13 198  0.79  0.79  0.61   0.53  3.5 1.4
## a14 198  0.84  0.82  0.67   0.57  5.8 1.5
##
## Non missing response frequency for each item
##        1    2    3    4    5    6    7    8    9   10 miss
## a12 0.00 0.00 0.01 0.05 0.16 0.29 0.35 0.14 0.02 0.00    0
## a13 0.07 0.16 0.35 0.16 0.20 0.04 0.02 0.00 0.00 0.00    0
## a14 0.01 0.02 0.04 0.12 0.23 0.21 0.26 0.09 0.02 0.01    0
```