GPTS Invariance testing Joachim_Kowalski

Jan Szczypiński

08 10 2021

First the MI analysis of GPTS

```
## This is lavaan 0.6-8
## lavaan is FREE software! Please report any bugs.

##

## ################################################################################

## This is semTools 0.5-4

## All users of R (or SEM) are invited to submit functions or ideas for functions.

## ################################################################################
```

recoding variables in dataset 2 and 3

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
d1 = subset(d, badanie ==1)
d2 = subset(d, badanie ==2)
d3 = subset(d, badanie ==3)

d2 = d2 %>%
  mutate_at(vars(1:18),
            ~factor(recode(.,
```

```
                            "1"=0,
                            "2"=1,
                            "3"=2,
                            "4"=3,
                            "5"=4)),ordered=T)

d3 = d3 %>%
  mutate_at(vars(1:18),
            ~factor(recode(.,
                            "1"=0,
                            "2"=1,
                            "3"=2,
                            "4"=3,
                            "5"=4)),ordered=T)

d = rbind(d1,d2)
d = rbind(d,d3)
colnames(d)[1] = 'PartA_gptsa1'
```

We create a matrix to compare fit indices across models with different constraints

```
all.results<-matrix(NA, nrow = 4, ncol = 6)
```

##Configural invariance

We follow the Wu and Estabrook (2016) procedure for caluclating invariance for models with variables on categorical scale We use delta parametrization. Groups are split based on psychiatric diagnosis (yes/no);

First we create a syntax for a configural (baseline) model using semTools package

```
model5 = 'reference =~ PartA_gptsa1 + PartA_gptsa2 +PartA_gptsa3 + PartA_gptsa4 +
          PartA_gptsa5 + PartA_gptsa6 + PartA_gptsa7

          persecutory =~ partB_gptsb1 + partB_gptsb2 + partB_gptsb3 + partB_gptsb4 +
          partB_gptsb5 + partB_gptsb6 + partB_gptsb7 + partB_gptsb8 + partB_gptsb9 +
          partB_gpts10
partB_gptsb2 ~~ partB_gptsb4
partB_gptsb3 ~~ partB_gptsb4
partB_gptsb2 ~~ partB_gptsb3'

MI_syntax_conf = measEq.syntax(configural.model = model5, data = d, ordered = TRUE,
                    ID.cat ='Wu.2016', ID.fac ='unit.variance', parameterization = 'delta',
                    group = "diag", group.equal = 'configural')
```

Next, we fit the configural (baseline model). If the fit indices are ok (Hu and Bentler, 1999) we assume configural invariance

```
cfa.config = cfa(as.character(MI_syntax_conf), data = d, estimator = "WLSMV",
                 group = "diag", missing = 'listwise', ordered = TRUE)
fitmeasures(cfa.config,c('chisq.scaled','df.scaled','cfi.scaled','tli.scaled',
                         'rmsea.scaled','srmr'))
```

```
## chisq.scaled    df.scaled   cfi.scaled   tli.scaled rmsea.scaled         srmr
##      511.573      230.000        0.988        0.986        0.053        0.040
```

We write the fit indices to the matrix created earlier.

```
all.results[1,] =round(data.matrix((fitmeasures(cfa.config,c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr')))), digits = 4)
```

##Treshold invariance First, we create a syntax for a model with thresholds constrain.

```
MI_syntax_thresh = measEq.syntax(configural.model = model5, data = d, ordered = TRUE,
                ID.cat ='Wu.2016', ID.fac ='unit.variance', parameterization = 'delta',
                 group = "diag", group.equal = 'thresholds')
```

Next, we fit the model and save fit indices to the matrix

```
cfa.thresholds <- cfa(as.character(MI_syntax_thresh), data = d, estimator = "WLSMV",
                      group = "diag", missing = 'listwise', ordered = TRUE)
all.results[2,] =round(data.matrix((fitmeasures(cfa.thresholds,c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr')))),digits = 4)
```

Finally we compare baseline model and threshold model using Satorra-Bentler Scaled chiˆ2 difference test

```
robust_diff(cfa.thresholds,cfa.config)
```

```
##      Test statistic   p.value
## [1,]         39.004 0.2548234
```

We see that the models do not differ significantly, thus we can assume threshold invariance.

##Metric invariance

First, we create a syntax for a model with thresholds and loadings constrains.

```
MI_syntax_metric = measEq.syntax(configural.model = model5, data = d, ordered = TRUE,
                                 ID.cat ='Wu.2016', ID.fac ='unit.variance', parameterization = 'delta',
                                  group = "diag", group.equal = c('thresholds','loadings'))
```

Next, we fit the model and save fit indices to the matrix

```
cfa.metric <- cfa(as.character(MI_syntax_metric), data = d, estimator = "WLSMV",
                    group = "diag", missing = 'listwise', ordered = TRUE)

all.results[3,] =round(data.matrix((fitmeasures(cfa.metric,c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr')))),digits = 4)
```

Finally we compare metric model and threshold model using Satorra-Bentler Scaled chiˆ2 difference test

```
robust_diff(cfa.metric,cfa.thresholds)
```

```
##      Test statistic   p.value
## [1,]       9.159807 0.8690144
```

Again the test was not significant so we assume metric invariance and proceed to the last step

##Scalar invariance

First, we create a syntax for a model with thresholds, loadings, and intercepts constrains.

```
MI_syntax_scalar = measEq.syntax(configural.model = model5, data = d, ordered = TRUE,
                                 ID.cat ='Wu.2016', ID.fac ='unit.variance', parameterization = 'delta'
                                 group = "diag", group.equal = c('thresholds','loadings','intercepts'))
```

Next, we fit the model and save fit indices to the matrix

```
cfa.scalar <- cfa(as.character(MI_syntax_scalar), data = d, estimator = "WLSMV",
                  group = "diag", missing = 'listwise', ordered = TRUE)


all.results[4,] = round(data.matrix((fitmeasures(cfa.scalar,c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr')))),digits =4)
```

Finally we compare scalar model and metric model model using Satorra-Bentler Scaled chi^2 difference test

```
robust_diff(cfa.scalar,cfa.metric)
```

```
##      Test statistic    p.value
## [1,]       26.11184 0.03686339
```

```
colnames(all.results) = c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr')
print(all.results)
```

```
##      chisq.scaled df.scaled cfi.scaled tli.scaled rmsea.scaled   srmr
## [1,]     511.5731       230     0.9882     0.9860       0.0533 0.0398
## [2,]     559.2644       264     0.9876     0.9873       0.0509 0.0398
## [3,]     555.3126       279     0.9884     0.9887       0.0479 0.0398
## [4,]     573.5197       294     0.9883     0.9892       0.0469 0.0404
```

The test was significant, thus we can't assume scalar invariance. The next step is to explore which constrained parameter needs to be freed to improve the fit. We use lavTestScore function to print the parameters which have the most influence on the model

```
param = lavTestScore(cfa.scalar)$uni
```

```
## Warning in lavTestScore(cfa.scalar): lavaan WARNING: se is not 'standard'; not
## implemented yet; falling back to ordinary score test
```

```
head(param[order(param$X2,decreasing =T),],10)
```

```
##
## univariate score tests:
##
##      lhs op    rhs     X2 df p.value
```

```
## 2   .p2. == .p146. 16.333  1   0.000
## 1   .p1. == .p145.  9.281  1   0.002
## 22 .p22. == .p166.  8.525  1   0.004
## 23 .p23. == .p167.  5.091  1   0.024
## 19 .p19. == .p163.  4.626  1   0.031
## 9   .p9. == .p153.  4.098  1   0.043
## 14 .p14. == .p158.  3.562  1   0.059
## 50 .p50. == .p194.  3.461  1   0.063
## 20 .p20. == .p164.  3.365  1   0.067
## 10 .p10. == .p154.  3.241  1   0.072
```

We need to use the parTable function to see what parameter numbers mean

```
head(parTable(cfa.scalar),10)
```

```
##    id         lhs op          rhs user block group free ustart exo       label
## 1   1   reference =~ PartA_gptsa1    1     1     1    1     NA   0  lambda.1_1
## 2   2   reference =~ PartA_gptsa2    1     1     1    2     NA   0  lambda.2_1
## 3   3   reference =~ PartA_gptsa3    1     1     1    3     NA   0  lambda.3_1
## 4   4   reference =~ PartA_gptsa4    1     1     1    4     NA   0  lambda.4_1
## 5   5   reference =~ PartA_gptsa5    1     1     1    5     NA   0  lambda.5_1
## 6   6   reference =~ PartA_gptsa6    1     1     1    6     NA   0  lambda.6_1
## 7   7   reference =~ PartA_gptsa7    1     1     1    7     NA   0  lambda.7_1
## 8   8 persecutory =~ partB_gptsb1    1     1     1    8     NA   0  lambda.8_2
## 9   9 persecutory =~ partB_gptsb2    1     1     1    9     NA   0  lambda.9_2
## 10 10 persecutory =~ partB_gptsb3    1     1     1   10     NA   0 lambda.10_2
##    plabel start   est    se
## 1    .p1. 0.835 0.789 0.025
## 2    .p2. 0.841 0.835 0.021
## 3    .p3. 0.711 0.725 0.026
## 4    .p4. 0.945 0.908 0.014
## 5    .p5. 0.797 0.819 0.021
## 6    .p6. 0.838 0.848 0.020
## 7    .p7. 0.679 0.765 0.025
## 8    .p8. 0.881 0.832 0.022
## 9    .p9. 0.889 0.893 0.025
## 10  .p10. 0.845 0.844 0.019
```

We see that freeing the reference =~ PartA_gptsa2 parameter improves the model fit the most

#Partial invariance

We will free the preference =~ PartA_gptsa2 parameter and try to establish a partial scalar invariance

First, we create a syntax for a model with thresholds, loadings, and intercepts constrains.

```
MI_syntax_partial = measEq.syntax(configural.model = model5, data = d, ordered = TRUE,
                                  ID.cat ='Wu.2016', ID.fac ='unit.variance', parameterization = 'delta'
                                  group = "diag", group.equal = c('thresholds','loadings','intercepts'),
                                  group.partial = c('reference =~ PartA_gptsa2'))
```

Next, we fit the model and save fit indices to the matrix

```
cfa.partial <- cfa(as.character(MI_syntax_partial), data = d, estimator = "WLSMV",
                   group = "diag", missing = 'listwise', ordered = TRUE)


all.results = rbind(all.results,round((fitmeasures(cfa.partial,c('chisq.scaled',
                'df.scaled','cfi.scaled','tli.scaled','rmsea.scaled','srmr'))),digits =4))
```

Finally we compare partial scalar model and metric model using Satorra-Bentler Scaled chiˆ2 difference test

```
robust_diff(cfa.partial,cfa.metric)
```

```
##      Test statistic   p.value
## [1,]       18.89233 0.1691045
```

```
print(all.results)
```

```
##      chisq.scaled df.scaled cfi.scaled tli.scaled rmsea.scaled   srmr
## [1,]     511.5731       230     0.9882     0.9860       0.0533 0.0398
## [2,]     559.2644       264     0.9876     0.9873       0.0509 0.0398
## [3,]     555.3126       279     0.9884     0.9887       0.0479 0.0398
## [4,]     573.5197       294     0.9883     0.9892       0.0469 0.0404
## [5,]     556.0204       293     0.9890     0.9898       0.0456 0.0402
```

We now see that there is no difference present between model with released reference =~ PartA_gptsa2 parameter Thus, we can establish partial scalar invariance