SREE Workshop Data Harmonization Illustrative Example: Continuous Case

2024-09-18

Contents

Install and load packages, prepare data.

```r
# install and load frequently used packages
library(dplyr)
library(lavaan)
library(sirt)
library(here)
# also install packages: haven, numDeriv
```

```r
dir.create("rds", showWarnings = FALSE)
if (!file.exists("rds/dat.rds")) {
  source("code/download_data.R")
  source("code/prepare_data.R")
} else {
  dat <- readRDS("rds/dat.rds")
}
```

```r
m_items <- paste0("i", 1:5)
# get subset of relevant variables
dat <- dat[, c("stu_id", "sample", "sex", "dropout", m_items)]
```

As a baseline for comparison against factor scores, we compute mean scores. We opt for mean scores instead of sum scores due to the missing item in HSLS.

```r
dat$mean_score <- c(rowMeans(dat[dat$sample == "ELS", m_items], na.rm = TRUE),
                    rowMeans(dat[dat$sample == "HSLS", m_items[-3]], na.rm = TRUE))

head(dat[dat$sample == "ELS", c(m_items, "mean_score")])
```

```
##   i1 i2 i3 i4 i5 mean_score
## 1  2  1  2  2  1        1.6
## 2  4  3  4  4  4        3.8
## 3  3  2  2  3  2        2.4
## 4  4  3  3  3  4        3.4
## 5  2  2  3  3  3        2.6
## 6  2  2  3  3 NA        2.5
```

```
head(dat[dat$sample == "HSLS", c(m_items, "mean_score")])
```

```
##       i1 i2 i3 i4 i5 mean_score
## 16198  4  3 NA  4  3       3.50
## 16199  3  3 NA  4  3       3.25
## 16200  4  2 NA  4  3       3.25
## 16201  3  3 NA  3  3       3.00
## 16202  3  3 NA  3  3       3.00
## 16203  4  4 NA  4  3       3.75
```

Obtain factor scores assuming continuous data

Step 1: Determine a configural model.

We declare groups and specify model syntax separately to account for the missing item (i3) in HSLS.

We identify the model by freeing the loading of the first item and constraining the latent variances to one and the latent mean to zero in both groups.

We freely estimate loadings, intercepts, and unique factor variances.

Covariances were added iteratively for items with the largest modification indices. See `determine_configural_model.R` for steps to determining a configural model using `lavaan`. Below is the final configural model chosen based on modification indices.

```
cfa_config <-  'group: ELS
                math =~ NA * i1 + l2_1 * i2 + l3 * i3 + l4_1 * i4 + l5_1 * i5

                i1 ~ nu1_1 * 1
                i2 ~ nu2_1 * 1
                i3 ~ nu3 * 1
                i4 ~ nu4_1 * 1
                i5 ~ nu5_1 * 1

                # variances
                i1 ~~ theta1_1 * i1
                i2 ~~ theta2_1 * i2
                i3 ~~ theta3 * i3
                i4 ~~ theta4_1 * i4
                i5 ~~ theta5_1 * i5

                # covariances
                i1 ~~ i2
                i2 ~~ cov3 * i3
                i2 ~~ i4

                # latent variances
                math ~~ 1 * math
                # latent means
                math ~ 0 * 1
                group: HSLS
                math =~ NA * i1 + l2_2 * i2 + l4_2 * i4 + l5_2 * i5

                i1 ~ nu1_2 * 1
                i2 ~ nu2_2 * 1
                i4 ~ nu4_2 * 1
                i5 ~ nu5_2 * 1

                i1 ~~ theta1_2 * i1
                i2 ~~ theta2_2 * i2
                i4 ~~ theta4_2 * i4
                i5 ~~ theta5_2 * i5
                i1 ~~ i2
                i2 ~~ i4

                math ~~ 1 * math
                math ~ 0 * 1
                '
fit_config  <- cfa(cfa_config, data = dat, group = "sample",
                   estimator = "MLR", missing = "FIML", se = "robust.mlr")
# you can print the s_config object to examine the fit indices and parameter
# estimates
```

The final configural model shows acceptable model fit, with robust scaled CFI = 0.999, robust scaled TLI = 0.991, robust scaled RMSEA = 0.059 with CI = [0.046, 0.072] and SRMR = 0.002.

Step 2: Determine a partial invariance model through traditional MI modeling (continuous case).

MI modeling is performed sequentially and iteratively by setting and releasing increasingly strict constraints on parameters, and comparing model fit with tests such as the likelihood ratio test (LRT). `traditional_mi_testing.R` contains the specific steps to determining the final partial invariance model which involve fitting and comparing the fit of a total of 21 separate models. A summary of the process is provided below.

First, we test for metric invariance after constraining the loadings to be the same across groups. The metric model identified by fixing the latent means to zero in both groups and the latent variance to one in the first group. We compare the fit of the metric model with the fit of the configural model. A significant LRT indicates a significantly worse fit of the more constrained metric model, and suggests that at least one loading is noninvariant across groups. We release one loading (for each possible loading) and compare the fit of the one-loading-released models with the metric model. We choose to release the loading that leads to the greatest improvement in fit, indicated by the largest χ^2 difference, and repeat the process for models with two loadings released. As we have a total of four overlapping items across the two groups, releasing two loadings leaves two loadings that could potentially be tested. However, as a minimum of two items must be invariant, we proceed to testing for scalar invariance.

We constain the intercepts across groups and release the latent mean and variance in group 2 to build our scalar invariance model. The intercepts for items with noninvariant loadings are also freed (Putnick & Bornstein, 2016). As freeing any additional intercepts would lead to a just-identified model in group 2, we move on to testing for strict invariance.

We additionally constrain the unique factor variances across groups to build the strict invariance model, and iteratively release one, two, three, and four loadings following the same procedure as metric invariance but with variances to determine a partial invariance model. Below is the final partial invariance model determined through this procedure.

```r
cfa_partial <-   'group: ELS
                  math =~ NA * i1 + i2 + l3 * i3 + l4 * i4 + l5 * i5

                  i1 ~ NA * 1
                  i2 ~ NA * 1
                  i3 ~ nu3 * 1
                  i4 ~ nu4 * 1
                  i5 ~ nu5 * 1

                  i1 ~~ theta1_1 * i1
                  i2 ~~ theta2_1 * i2
                  i3 ~~ theta3 * i3
                  i4 ~~ theta4_1 * i4
                  i5 ~~ theta5_1 * i5
                  i1 ~~ i2
                  i2 ~~ cov3 * i3
                  i2 ~~ i4

                  math ~~ 1 * math
                  math ~ 0 * 1

                  group: HSLS
                  math =~ NA * i1 + i2 + l4 * i4 + l5 * i5

                  i1 ~ NA * 1
                  i2 ~ NA * 1
                  i4 ~ nu4 * 1
                  i5 ~ nu5 * 1

                  i1 ~~ theta1_2 * i1
                  i2 ~~ theta2_2 * i2
                  i4 ~~ theta4_2 * i4
                  i5 ~~ theta5_2 * i5
                  i1 ~~ i2
                  i2 ~~ i4

                  math ~~ NA * math
                  math ~ NA * 1
                  '
fit_partial  <- cfa(cfa_partial, data = dat, group = "sample",
                    estimator = "MLR", missing = "FIML", se = "robust.mlr")
```

```r
# extract cfa parameter estimates
est_partial <- lavInspect(fit_partial, what = "est")
```

Step 3. Determine an approximate invariance model through alignment optimization (continuous case).

Alignment optimization also begins with determining an appropriate configural model. We extract the measurement intercept and loading estimates from the configural model.

```r
est_config <- lavInspect(fit_config, what = "est")
config_lambda <- merge(est_config$ELS$lambda, est_config$HSLS$lambda,
                       by = "row.names", all = TRUE)
```

```
config_lambda_mat <- t(config_lambda[, -1])
config_nu <- merge(est_config$ELS$nu, est_config$HSLS$nu,
                   by = "row.names", all = TRUE)
config_nu_mat <- t(config_nu[, -1])
```

We next obtain a weight matrix of the sample sizes for each item by sample.

```
m_items <- paste0("i", 1:5)
(wgt_mat <- as.matrix(rbind(
  colSums(!is.na(dat[dat$sample == "ELS", m_items])),
  colSums(!is.na(dat[dat$sample == "HSLS", m_items]))
)))
```

```
##         i1    i2    i3    i4    i5
## [1,] 11391 11432 11047 10823 10661
## [2,] 19049 19006     0 18926 18976
```

We perform alignment using the `invariance.alignment` function from the R package `sirt` with the configural loading and intercept estimates.

```
aligned_par <- invariance.alignment(lambda = config_lambda_mat,
                                    nu = config_nu_mat,
                                    fixed = TRUE, # fix SD of first group to one
                                    wgt = sqrt(wgt_mat))
```

We modify the configural model specification string to freely estimate the latent mean and variance in both groups, and substitute in the aligned estimates from the configural model.

```
cfa_align <-   'group: ELS
                math =~ l1_1 * i1 + l2_1 * i2 + l3 * i3 + l4_1 * i4 + l5_1 * i5

                i1 ~ nu1_1 * 1
                i2 ~ nu2_1 * 1
                i3 ~ nu3 * 1
                i4 ~ nu4_1 * 1
                i5 ~ nu5_1 * 1

                i1 ~~ theta1_1 * i1
                i2 ~~ theta2_1 * i2
                i3 ~~ theta3 * i3
                i4 ~~ theta4_1 * i4
                i5 ~~ theta5_1 * i5
                i1 ~~ i2
                i2 ~~ cov3 * i3
                i2 ~~ i4

                math ~~ NA * math
                math ~ NA * 1

                group: HSLS
                math =~ l1_2 * i1 + l2_2 * i2 + l4_2 * i4 + l5_2 * i5

                i1 ~ nu1_2 * 1
                i2 ~ nu2_2 * 1
                i4 ~ nu4_2 * 1
                i5 ~ nu5_2 * 1
                i1 ~~ theta1_2 * i1
                i2 ~~ theta2_2 * i2
                i4 ~~ theta4_2 * i4
                i5 ~~ theta5_2 * i5
                i1 ~~ i2
                i2 ~~ i4

                math ~~ NA * math
                math ~ NA * 1
                '
```

```
cfa_align <- sub("l1_1", paste0(aligned_par$lambda.aligned[,1][1],
                              collapse = ","), cfa_align)
cfa_align <- sub("l1_2", paste0(aligned_par$lambda.aligned[,1][2],
                              collapse = ","), cfa_align)
cfa_align <- gsub("nu1_1", paste0(aligned_par$nu.aligned[,1][1],
                                collapse = ","), cfa_align)
cfa_align <- gsub("nu1_2", paste0(aligned_par$nu.aligned[,1][2],
                                collapse = ","), cfa_align)
```

We fit the modified model using `cfa()` and obtain parameter estimates.

```
fit_align  <- cfa(model = cfa_align,
                  data = dat,  estimator = "MLR", group = "sample",
                  missing = "FIML", se = "robust.mlr")
```

```
# obtain parameter estimates
est_align <- lavInspect(fit_align, what = "est")
```

Step 4: Compute Bartlett factor scores

We obtain Bartlett scores in the continuous case.

```
fs_partial <- lavPredict(fit_partial, method = "bartlett", se = TRUE)
fs_align <- lavPredict(fit_align, method = "bartlett", se = TRUE)

# extract standard errors of the Bartlett scores
fs_partial_SE <- unlist(attributes(fs_partial)$se)
fs_align_SE <- unlist(attributes(fs_align)$se)

# store FS and FS SE in the score_df data frame
score_df <- as.data.frame(cbind(dat,
  "partial_cont" = as.numeric(unlist(fs_partial)),
  "partial_SE" = fs_partial_SE,
  "approx_cont" = as.numeric(unlist(fs_align)),
  "approx_SE" = fs_align_SE))
head(score_df, 2)
```

```
##      stu_id sample sex dropout i1 i2 i3 i4 i5 mean_score partial_cont
## ELS1 101101    ELS   0       0  2  1  2  2  1        1.6    -1.114292
## ELS2 101102    ELS   0       0  4  3  4  4  4        3.8     1.658808
##      partial_SE approx_cont approx_SE
## ELS1   0.278666   -1.131463 0.2781254
## ELS2   0.278666    1.661305 0.2781254
```

Step 5: Compute reliability

We compute Bartlett scores in the continuous case.

$$\text{Bartlett score reliability} = \frac{\psi}{[\psi + \text{SE}^2]}$$

```r
bartlett_rel <- function(psi, SE) { psi / (psi + SE^2) }

# compute reliability for each individual
rel_partial_ELS <- bartlett_rel(c(est_partial$ELS$psi),
                                score_df[score_df$sample == "ELS",
                                         "partial_SE"])
rel_partial_HSLS <- bartlett_rel(c(est_partial$HSLS$psi),
                                 score_df[score_df$sample == "HSLS",
                                          "partial_SE"])
score_df$partial_rel <- c(rel_partial_ELS, rel_partial_HSLS)

rel_approx_ELS <- bartlett_rel(c(est_align$ELS$psi),
                               score_df[score_df$sample == "ELS",
                                        "approx_SE"])
rel_approx_HSLS <- bartlett_rel(c(est_align$HSLS$psi),
                                score_df[score_df$sample == "HSLS",
                                         "approx_SE"])
score_df$approx_rel <- c(rel_approx_ELS, rel_approx_HSLS)
```

```r
# store error variances
score_df <- as.data.frame(cbind(score_df,
  "partial_ev" = score_df$partial_SE^2,
  "approx_ev" = score_df$approx_SE^2))
```

```r
score_df[, 3:11] <- apply(score_df[, 3:11], FUN = as.numeric, MARGIN = 2)
head(score_df, 2)
```

```
##      stu_id sample sex dropout i1 i2 i3 i4 i5 mean_score partial_cont
## ELS1 101101    ELS   0       0  2  1  2  2  1        1.6    -1.114292
## ELS2 101102    ELS   0       0  4  3  4  4  4        3.8     1.658808
##      partial_SE approx_cont approx_SE partial_rel approx_rel partial_ev
## ELS1   0.278666   -1.131463 0.2781254    0.927941  0.9282002 0.07765475
## ELS2   0.278666    1.661305 0.2781254    0.927941  0.9282002 0.07765475
##       approx_ev
## ELS1 0.07735375
## ELS2 0.07735375
```

```r
saveRDS(score_df, "rds/score_df_continuous.rds")
saveRDS(est_partial, "rds/est_partial.rds")
saveRDS(est_align, "rds/est_align.rds")
saveRDS(fit_partial, "rds/fit_partial.rds")
saveRDS(fit_align, "rds/fit_align.rds")
```