SREE Workshop Data Harmonization Illustrative Example: Ordinal Case

2024-09-18

Contents

Install and load packages, prepare data.

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

```
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")
}
```

```
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.

```
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))
```

Obtain factor scores assuming ordinal data

Step 1: Determine a partial invariance model through traditional MI modeling (ordinal case).

`traditional_mi_testing_ordinal.R` contains the specific steps we followed to determine the final partial invariance model assuming ordered categorical data. Five models were fit and compared to determine a configural model followed by 24 additional models to determine the final partial invariance model. For additional details, see Tse, W. W. Y., Lai, M. H., & Zhang, Y. (2024). Does strict invariance matter? Valid group mean comparisons with ordered-categorical items. Behavior Research Methods, 56(4), 3117-3139.

Below is the final partial invariance model determined through this procedure.

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

                    i1 ~~ 1 * i1
                    i2 ~~ 1 * i2
                    i3 ~~ 1 * i3
                    i4 ~~ 1 * i4
                    i5 ~~ 1 * i5
                    i1 ~~ i2
                    i2 ~~ i3
                    i2 ~~ i4

                    math ~~ 1 * math
                    math ~ 0 * 1

                    i1 | th1 * t1
                    i1 | th2 * t2
                    i1 | th3 * t3

                    i2 | th4 * t1
                    i2 | th5 * t2
                    i2 | th6 * t3

                    i3 | th7 * t1
                    i3 | th8 * t2
                    i3 | th9 * t3

                    i4 | th10 * t1
                    i4 | th11 * t2
                    i4 | th12 * t3

                    i5 | th13 * t1
                    i5 | th14 * t2
                    i5 | th15 * t3

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

                    i1 ~~ NA * i1
                    i2 ~~ NA * i2
                    i4 ~~ 1 * i4
                    i5 ~~ NA * i5
```

```
                    i1 ~~ i2
                    i2 ~~ i4

                    math ~~ NA * math
                    math ~ NA * 1

                    i1 | th1 * t1
                    i1 | th2a * t2
                    i1 | th3 * t3

                    i2 | th4 * t1
                    i2 | th5a * t2
                    i2 | th6 * t3

                    i4 | th10 * t1
                    i4 | th11a * t2
                    i4 | th12 * t3

                    i5 | th13 * t1
                    i5 | th14a * t2
                    i5 | th15a * t3
                    '

fit_partial_ord  <- cfa(cfa_partial_ord, data = dat, group = "sample",
                        estimator = "WLSMV", ordered = TRUE,
                        parameterization = "theta")
```

```
# extract cfa parameter estimates
est_partial_ord <- lavInspect(fit_partial_ord, what = "est")
```

Step 2. Determine an approximate invariance model through alignment optimization (ordinal case).

```
# fit a graded response model (equivalent to an ordinal cfa).
config_ord <- multipleGroup(data = dat[, m_items], group = dat$sample,
                            itemtype = "graded", invariance = "i3")
```

```
## Warning: data contains response patterns with only NAs

## Iteration: 1, Log-Lik: -148893.119, Max-Change: 2.94493Iteration: 2, Log-Lik: -121681.876, Max-Chang
```

```
# extract coefficients
est_coef <- mirt.wrapper.coef(config_ord)$coef
# Remove coefficients for i3 for HSLS
est_coef[6, c("a1", "d1", "d2", "d3")] <- NA
# est_coef has columns group, item, a1, d1, d2, and d3.
# a1 contains the loadings, d1-d3 are the thresholds.
(els_coef <- est_coef[est_coef$group == "ELS",])
```

```
##   group item       a1       d1         d2        d3
## 1   ELS   i1 3.367244 4.716831 -0.4397776 -3.074938
## 3   ELS   i2 3.483662 3.490577 -0.8780415 -4.181195
## 5   ELS   i3 4.365504 4.500842 -0.4857222 -4.260119
## 7   ELS   i4 4.372924 5.632033  0.2849476 -3.562358
## 9   ELS   i5 4.337701 5.732986  0.4396606 -3.412799
```

```r
(hsls_coef <- est_coef[est_coef$group == "HSLS",])
```

```
##    group item       a1       d1       d2        d3
## 2   HSLS   i1 3.863295 7.498264 3.094064 -2.864859
## 4   HSLS   i2 2.996175 4.982647 1.199976 -3.369710
## 6   HSLS   i3       NA       NA       NA        NA
## 8   HSLS   i4 4.313544 8.691290 4.301483 -2.717268
## 10  HSLS   i5 3.824915 7.657129 3.279324 -3.066883
```

```r
# To compute threshold estimates for polytomous items, we'll treat each
# threshold as an item in sirt::invariance.alignment so we modify the lambda and
# weight matrices to be in compatible dimensions

# prepare lambda and nu matrices with estimates from the configural model
(lambda1 <- rbind(rep(els_coef$a1, 3), rep(hsls_coef$a1, 3)))
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 3.367244 3.483662 4.365504 4.372924 4.337701 3.367244 3.483662 4.365504
## [2,] 3.863295 2.996175       NA 4.313544 3.824915 3.863295 2.996175       NA
##          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]
## [1,] 4.372924 4.337701 3.367244 3.483662 4.365504 4.372924 4.337701
## [2,] 4.313544 3.824915 3.863295 2.996175       NA 4.313544 3.824915
```

```r
(nu1 <- rbind(unlist(els_coef[, c("d1", "d2", "d3")]),
              unlist(hsls_coef[, c("d1", "d2", "d3")])))
```

```
##           d11      d12      d13      d14      d15        d21        d22
## [1,] 4.716831 3.490577 4.500842 5.632033 5.732986 -0.4397776 -0.8780415
## [2,] 7.498264 4.982647       NA 8.691290 7.657129  3.0940640  1.1999761
##             d23       d24       d25       d31       d32       d33       d34
## [1,] -0.4857222 0.2849476 0.4396606 -3.074938 -4.181195 -4.260119 -3.562358
## [2,]         NA 4.3014829 3.2793241 -2.864859 -3.369710        NA -2.717268
##            d35
## [1,] -3.412799
## [2,] -3.066883
```

```r
# weight matrix
(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
```

```r
# perform alignment and obtain aligned parameters of the latent mean and latent
# variance
ord_align <- invariance.alignment(lambda1, nu1,
                                  wgt = sqrt(cbind(wgt_mat, wgt_mat, wgt_mat)))
ord_align$pars
```

```
##      alpha0     psi0
## G1 0.000000 1.000000
## G2 0.699614 0.984914
```

```r
ord_align$pars[2, 2]^2 # square the SD to get the variance, 0.9700555
```

```
## [1] 0.9700555
```

```r
# Constrain latent means and variances to the aligned levels
mirtmodel_al <- "
    F1 = i1, i2, i3, i4, i5
    START [HSLS] = (GROUP, MEAN_1, 0.699614), (GROUP, COV_11, 0.9700555)
  "
# Specify the mirt model explicitly providing the item names
mirtmodel_al <- mirt.model(mirtmodel_al, itemnames = m_items)
# Drop observations where all math items are NAs to avoid issues with `fscores()`
all_na <- rowSums(!is.na(dat[, m_items])) == 0

# Fit the aligned model
mod_aligned <- multipleGroup(dat[!all_na, m_items],
                             model = mirtmodel_al,
                             group = dat$sample[!all_na],
                             itemtype = "graded",
                             invariance = "i3")
```

```
## Iteration: 1, Log-Lik: -149122.928, Max-Change: 2.99206Iteration: 2, Log-Lik: -122268.510, Max-Change
```

```r
est_align_ord <- mirt.wrapper.coef(mod_aligned)$coef
est_align_ord[6, c("a1", "d1", "d2", "d3")] <- NA
```

```r
# Make sure coefficients are comparable to aligned parameters
ord_align$lambda.aligned[, 1:5] - est_align_ord$a1
```

```
##               I1           I2           I3            I4            I5
## G1 -6.425054e-05 2.104800e-04 9.607714e-05 -0.0006018592 -6.585290e-04
## G2  8.397124e-06 5.518844e-05           NA  0.0000146369  2.559408e-05
```

```r
est_align_ord_th <- matrix(unlist(est_align_ord[, c("d1", "d2", "d3")]),
                           nrow = 2)
est_align_ord_lam <- matrix(est_align_ord[, c("a1")], nrow = 2)
```

Step 3: Compute factor scores and obtain standard errors

In the ordinal case, we compute scores from the partial invariance model using the EBM (Empirical Bayes Modal) approach, which is one of the two options available in `lavaan::lavPredict()` for categorical data. We compute EAP (expected a-posteriori) factor scores from the approximate invariance model, which is the default option for 'mirt::fscores()'

```r
fs_partial_ord <- lavPredict(fit_partial_ord, method = "EBM", se = TRUE)
```

```
## Warning in lav_predict_internal(lavmodel = lavmodel, lavdata = lavdata, :
## lavaan WARNING: standard errors not available (yet) for non-normal data
```

```r
fs_align_ord <- as.data.frame(fscores(mod_aligned, mean = c(0, 0),
                                      cov = c(1, 1), full.scores.SE = TRUE,
                                      method = "EAP"))
# store the factor scores
score_df_ord <- cbind(dat, approx_ord = NA, approx_ord_SE = NA)
score_df_ord$approx_ord[!all_na] <- fs_align_ord[, "F1"]
score_df_ord$approx_ord_SE[!all_na] <- fs_align_ord[, "SE_F1"]
```

Note that SE are not available for non-normal data in lavaan so we cannot obtain SE for the factor scores computed using the partial invariance model assuming ordinal data.

Step 4: Compute reliability.

EAP score reliability

$$\text{EAP score reliability} = 1 - \frac{\text{SE}^2}{\psi}$$

```
# obtain latent variances for the two groups
psi_align_ord_ELS <- mirt.wrapper.coef(mod_aligned)$GroupPars$ELS[2]
psi_align_ord_HSLS <- mirt.wrapper.coef(mod_aligned)$GroupPars$HSLS[2]

rel_approx_ord_ELS <- 1 - score_df_ord[score_df_ord$sample == "ELS",
                                       "approx_ord_SE"]^2 /  psi_align_ord_ELS

rel_approx_ord_HSLS <- 1 - score_df_ord[score_df_ord$sample == "HSLS",
                                        "approx_ord_SE"]^2 / psi_align_ord_HSLS
score_df_ord$approx_ord_rel <- c(rel_approx_ord_ELS, rel_approx_ord_HSLS)
score_df_ord[,3:6] <- apply(score_df_ord[,3:6], FUN = as.numeric, MARGIN = 2)
```

```
# store error variances in a data frame
score_df_ord$approx_ord_ev <- score_df_ord$approx_ord_SE^2 * score_df_ord$approx_ord_rel
head(score_df_ord, 2)
```

```
##   stu_id sample sex dropout i1 i2 i3 i4 i5 mean_score approx_ord approx_ord_SE
## 1 101101    ELS   0       0  2  1  2  2  1        1.6  -1.018915     0.2476258
## 2 101102    ELS   0       0  4  3  4  4  4        3.8   1.312507     0.2987173
##   approx_ord_rel approx_ord_ev
## 1      0.9386815    0.05755857
## 2      0.9107680    0.08126969
```

```
saveRDS(score_df_ord, "rds/score_df_ord.rds")
saveRDS(est_partial_ord, "rds/est_partial_ord.rds")
saveRDS(est_align_ord, "rds/est_align_ord.rds")
saveRDS(est_align_ord_th, "rds/est_align_ord_th.rds")
saveRDS(est_align_ord_lam, "rds/est_align_ord_lam.rds")
saveRDS(mod_aligned, "rds/mod_aligned.rds")
saveRDS(fs_partial_ord, "rds/fs_partial_ord.rds")
```