Supplemental Materials #2: Re-analysis of Study 14 (Monnot et al., 2011) in Yu et al. (2016)

Mike W.-L. Cheung

October 20, 2017

Contents

Functions for the analysis

- This function is modified from the one in Yu et al. (2016) Supplemental Materials #1 with three key differences:
  1. The original function incorrectly uses SDs to represent variances to generate the bootstrap correlation matrices. This error is corrected here.
  2. The original function calls build.matrix() recursively. R will throw an error when it goes too deep in generating positive definite matrices. This revised function breaks it down into two functions so that there are no recursive calls.
  3. It adds a `nearPD` argument. If it is `TRUE`, it transforms the non-positive definite matrices into the near positive definite matrices with the `nearPD()` function in the `Matrix` package. If it is `FALSE`, new matrices will be generated until it is positive definite.

```
random.matrices <- function(point.matrix, reps, nearPD=FALSE) {
  ## SDs of the correlations
  SDs <- diag(point.matrix[,,2][lower.tri(point.matrix[,,2])])

  ## Convert the SDs to Variances
  sigma <- SDs^2

  ## Mean of correlation vector
  mu <- point.matrix[,,1][lower.tri(point.matrix[,,1])]

  build.matrix <- function(mu, sigma)  {
    vec <- mvrnorm(1, mu, Sigma=sigma )
    mat <- diag(1, nrow(point.matrix))
    mat[lower.tri(mat)] <- vec
    trans.mat <- t(mat)
    mat[upper.tri(mat)] <- trans.mat[upper.tri(trans.mat)]
    mat
  }
```

```
  ## Counter for NPD matrices
  countNPD <- 0

  ## Setup an array to store the generated correlation matrices
  output <- array(1, dim = c(nrow(point.matrix), ncol(point.matrix), reps))

  for (rep in 1:reps) {
    mat <- build.matrix(mu=mu, sigma=sigma)

    ## Run nearPD==TRUE
    if (nearPD) {
      if (!is.positive.definite(mat)) {
        mat <- as.matrix(nearPD(mat, keepDiag = T)$mat)
        countNPD <- countNPD+1
      }
    } else {
      ## Run rearPD==FALSE
      while(!is.positive.definite(mat)) {
        mat <- build.matrix(mu=mu, sigma=sigma)
        countNPD <- countNPD+1
      }
    }

    output[,,rep] <- mat
  }
  rownames(output) <- colnames(output) <- rownames(point.matrix)
  return(list(countTotal=(reps+countNPD), countNPD=countNPD, data=output))
}


## This function tests the biases of the generated data.
testBias <- function(point.matrix, reps, nearPD=FALSE) {
  ## Generate data
  mat <- random.matrices(point.matrix=point.matrix, reps=reps, nearPD=nearPD)

  ## Convert it into a list for ease of manipulations
  mat.list <- alply(mat$data, 3)

  ## Percentage of matrices is PD
  Percentage.NPD <- mat$countNPD/mat$countTotal*100

  ## Convert the list into a matrix of vectors to calculate mean and SD
  mat.vec <- t(sapply(mat.list, lav_matrix_vech, diagonal=FALSE))

  ## Mean of the correlation matrices
  mat.rc <- lav_matrix_vech_reverse(apply(mat.vec, 2, mean), diagonal = FALSE)
  diag(mat.rc) <- 1

  # Percentage bias of the mean
  Percentage.bias.rc <- (mat.rc-point.matrix[,,1])/point.matrix[,,1]*100
  # Remove Inf as NA
  Percentage.bias.rc[is.infinite(Percentage.bias.rc)] <- NA
```

```
  ## sd of the correlation matrices
  mat.sd <- lav_matrix_vech_reverse(apply(mat.vec, 2, sd), diagonal = FALSE)

  # Percentage bias of the SDs
  Percentage.bias.sd <- (mat.sd-point.matrix[,,2])/point.matrix[,,2]*100

  ## Bias of correlation among the generated correlation vectors
  CorAmongCor <- lav_matrix_vech(cor(mat.vec), diagonal = FALSE)

  list(summary=list(Percentage.NPD=Percentage.NPD,
                    Percentage.bias.rc=Percentage.bias.rc,
                    Percentage.bias.sd=Percentage.bias.sd,
       Summary.percentage.bias.rc=summary(lav_matrix_vech(Percentage.bias.rc, diagonal = FALSE)),
       Summary.percentage.bias.sd=summary(lav_matrix_vech(Percentage.bias.sd, diagonal = FALSE)),
                    CorAmongCor=summary(CorAmongCor)), data=mat$data)
}

## This function is based on the one in Yu et al. Supplemental Materials #1
FIMASEM <- function(model, input.matrices, sample.nobs) {
  coefs.fits <- as.data.frame(t(sapply(1:dim(input.matrices)[3], function(i) {
    temp.sem <- sem(model=model, sample.nobs=sample.nobs,
                    sample.cov=input.matrices[,,i], warn=FALSE)
    c(coef(temp.sem), fitMeasures(temp.sem))})))
}
```

Input data

```
## Required packages
lib2install <- c("plyr", "matrixcalc", "lavaan", "MASS", "psych", "Matrix")

## Install them automatically if they are not available on your computer
for (i in lib2install) {
  if (!(i %in% rownames(installed.packages()))) install.packages(i)
}

## Libraries used in the analysis
library(plyr)
library(matrixcalc)
library(lavaan)
library(MASS)
library(psych)
library(Matrix)

## Set the seed for replication
set.seed(927037462)

## Large bootstrap replications to minimize simulation error
reps <- 10000

## Sample size
sample.nobs <- 200
```

- It is important to note that standard deviations (SDs) were incorrectly used to represent variances to generate the bootstrap correlation matrices in the `random.matrices.univarite()` in Yu's et al. Supplemental Materials #1. This error is corrected here.
  - `inputmatrices`: SD matrix used in Yu et al. (2016)

```
rc <- matrix(c(1,.47,.12,.20,.24,.00,
               .47,1,.11,.18,.11,-.04,
               .12,.11,1,.70,.73,.40,
               .20,.18,.70,1,.56,.22,
               .24,.11,.73,.56,1,.41,
               .00,-.04,.40,.22,.41,1), nr=6)
sd <- matrix(c(.00,.30,.20,.14,.23,.20,
              .30,.00,.22,.17,.22,.18,
              .20,.22,.00,.19,.16,.20,
              .14,.17,.19,.00,.26,.16,
              .23,.22,.16,.26,0,.24,
              .20,.18,.20,.16,.24,0), nr=6)
input.matrices <- array(0, dim=c(nrow(rc), ncol(rc),2))
input.matrices[,,1] <- rc
input.matrices[,,2] <- sd
dimnames(input.matrices) <- list(c('ORGC','JSAT','PROUATT','INSTRU','UCOM','UPART'),
                                 c('ORGC','JSAT','PROUATT','INSTRU','UCOM','UPART'),
                                 c("rc", "sd"))
input.matrices
```

```
## , , rc
##
##         ORGC  JSAT PROUATT INSTRU UCOM UPART
## ORGC    1.00  0.47    0.12   0.20 0.24  0.00
## JSAT    0.47  1.00    0.11   0.18 0.11 -0.04
## PROUATT 0.12  0.11    1.00   0.70 0.73  0.40
## INSTRU  0.20  0.18    0.70   1.00 0.56  0.22
## UCOM    0.24  0.11    0.73   0.56 1.00  0.41
## UPART   0.00 -0.04    0.40   0.22 0.41  1.00
##
## , , sd
##
##         ORGC JSAT PROUATT INSTRU UCOM UPART
## ORGC    0.00 0.30    0.20   0.14 0.23  0.20
## JSAT    0.30 0.00    0.22   0.17 0.22  0.18
## PROUATT 0.20 0.22    0.00   0.19 0.16  0.20
## INSTRU  0.14 0.17    0.19   0.00 0.26  0.16
## UCOM    0.23 0.22    0.16   0.26 0.00  0.24
## UPART   0.20 0.18    0.20   0.16 0.24  0.00
```

Checking the generated correlation matrices

Replace NPD correlation matrices with newly generated matrices (replacement method)

```
replacement.data <- testBias(point.matrix=input.matrices, reps=reps, nearPD=FALSE)
replacement.data$summary
```

```
## $Percentage.NPD
## [1] 74.72066
##
## $Percentage.bias.rc
##                ORGC        JSAT     PROUATT      INSTRU       UCOM
## ORGC      0.0000000 -20.3546341  13.9482931  -0.8136603 -14.486557
## JSAT    -20.3546341   0.0000000   0.6180548  -2.7104838   5.806848
## PROUATT  13.9482931   0.6180548   0.0000000 -11.5333205 -10.058401
## INSTRU   -0.8136603  -2.7104838 -11.5333205   0.0000000  -6.821452
## UCOM    -14.4865571   5.8068476 -10.0584013  -6.8214520   0.000000
## UPART            NA -12.9549677  -5.8898993   5.8309321 -10.331016
##              UPART
## ORGC            NA
## JSAT    -12.954968
## PROUATT  -5.889899
## INSTRU    5.830932
## UCOM    -10.331016
## UPART     0.000000
##
## $Percentage.bias.sd
##                ORGC       JSAT   PROUATT     INSTRU      UCOM      UPART
## ORGC            NaN -18.655245 -13.56228  -5.383758 -15.83886  -5.357794
## JSAT    -18.655245        NaN -15.14714  -6.172967 -13.04984  -5.104401
## PROUATT -13.562278 -15.147142       NaN -22.546825 -19.97234 -13.962333
## INSTRU   -5.383758  -6.172967 -22.54683        NaN -27.39910  -6.051394
## UCOM    -15.838856 -13.049844 -19.97234 -27.399097       NaN -17.534838
## UPART    -5.357794  -5.104401 -13.96233  -6.051394 -17.53484        NaN
##
## $Summary.percentage.bias.rc
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -20.3546 -11.2327  -6.3557  -4.9822   0.2601  13.9483        1
##
## $Summary.percentage.bias.sd
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -27.399 -18.095 -13.962 -13.716  -6.112  -5.104
##
## $CorAmongCor
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0646523 -0.0153979 -0.0006046  0.0120218  0.0200958  0.1617657
```

Replace NPD correlation matrices with nearPD matrices (nearPD method)

```
nearPD.data <- testBias(point.matrix=input.matrices, reps=reps, nearPD=TRUE)
nearPD.data$summary
```

```
## $Percentage.NPD
## [1] 43.20763
##
## $Percentage.bias.rc
##                ORGC         JSAT     PROUATT     INSTRU      UCOM
## ORGC      0.0000000  -4.20810649  8.24636236 -0.6371194 -5.498776
## JSAT     -4.2081065   0.00000000  0.02210231 -1.8611796  2.847520
## PROUATT   8.2463624   0.02210231  0.00000000 -4.3695688 -4.777884
```

```
## INSTRU  -0.6371194  -1.86117956 -4.36956885  0.0000000 -1.369934
## UCOM    -5.4987756   2.84752011 -4.77788410 -1.3699339  0.000000
## UPART           NA -11.32082825 -1.08899926  3.6836698 -2.805620
##              UPART
## ORGC            NA
## JSAT    -11.320828
## PROUATT  -1.088999
## INSTRU    3.683670
## UCOM     -2.805620
## UPART     0.000000
##
## $Percentage.bias.sd
##              ORGC      JSAT    PROUATT     INSTRU       UCOM     UPART
## ORGC          NaN -9.492244  -9.523041  -5.798344  -9.057849 -3.112887
## JSAT    -9.492244       NaN  -9.642144  -6.178509  -8.623723 -4.959597
## PROUATT -9.523041 -9.642144        NaN -16.106709 -18.544962 -8.165233
## INSTRU  -5.798344 -6.178509 -16.106709        NaN -15.341907 -3.775147
## UCOM    -9.057849 -8.623723 -18.544962 -15.341907        NaN -9.140049
## UPART   -3.112887 -4.959597  -8.165233  -3.775147  -9.140049       NaN
##
## $Summary.percentage.bias.rc
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -11.3208  -4.3292  -1.6156  -1.6527  -0.1427   8.2464        1
##
## $Summary.percentage.bias.sd
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -18.545  -9.583  -9.058  -9.164  -5.988  -3.113
##
## $CorAmongCor
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0608334 -0.0113056  0.0008941  0.0104198  0.0237635  0.1451707
```

Re-analysis of FIMASEM

```
model <- "UPART ~ UCOM
          UCOM ~ ORGC + JSAT + PROUATT + INSTRU
          ORGC ~ JSAT
          PROUATT ~ INSTRU"
```

Analysis of the bootstrap data with replacement method

```
fit <- FIMASEM(model, replacement.data$data, sample.nobs=sample.nobs)

cfi.fit <- ifelse(fit$cfi>0.9, yes=1, no=0)
srmr.fit <- ifelse(fit$srmr<0.1, yes=1, no=0)
rmsea.fit <- ifelse(fit$rmsea<0.05, yes=1, no=0)

cat("Percentage of CFA >.9:", mean(cfi.fit)*100)
```

```
## Percentage of CFA >.9: 9.71
```

```
cat("Percentage of SRMR < .1:", mean(srmr.fit)*100)
```

```
## Percentage of SRMR < .1: 20.51
```

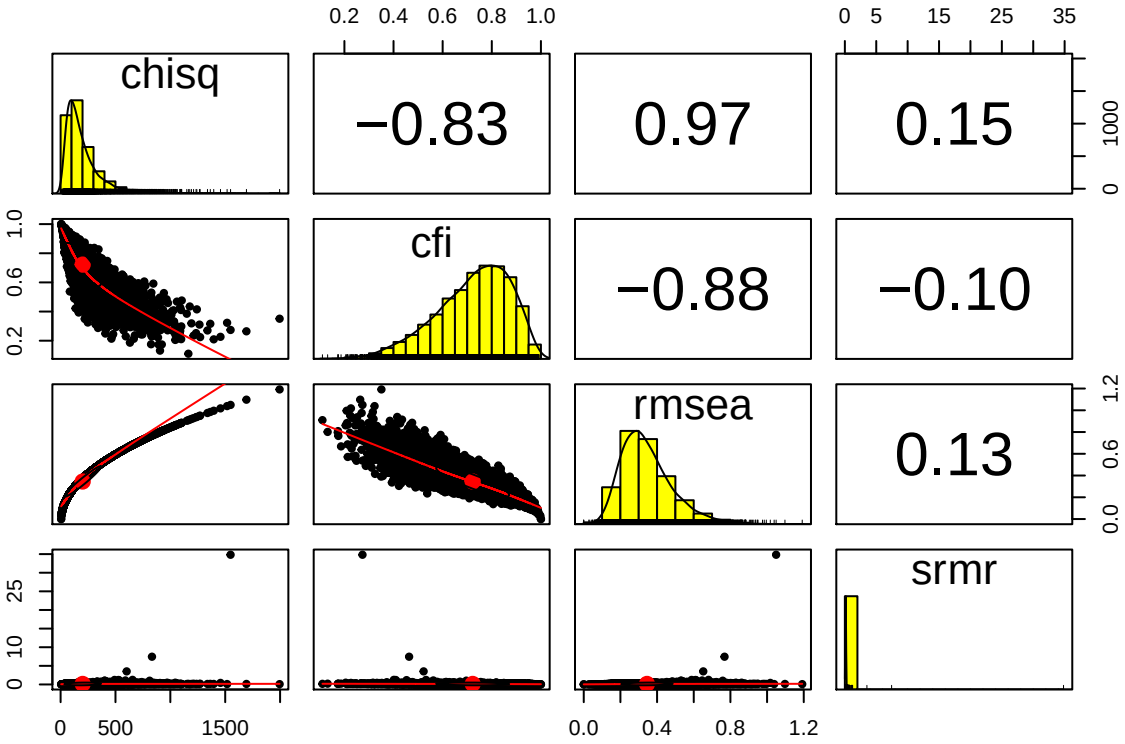
```
cat("Percentage of sample RMSEA < .05:", mean(rmsea.fit)*100)
```

```
## Percentage of sample RMSEA < .05: 0.07
```

```
describe(fit[, c("chisq", "df", "cfi", "rmsea", "srmr")],
         trim=0, quant=c(.10, .25, .50, .75, .90))
```

```
##       vars     n   mean     sd median trimmed    mad  min     max   range
## chisq    1 10000 201.12 167.14 151.99  201.12 108.63 4.83 1995.53 1990.70
## df       2 10000   7.00   0.00   7.00    7.00   0.00 7.00    7.00    0.00
## cfi      3 10000   0.72   0.15   0.74    0.72   0.15 0.11    1.00    0.89
## rmsea    4 10000   0.34   0.14   0.32    0.34   0.13 0.00    1.19    1.19
## srmr     5 10000   0.14   0.36   0.13    0.14   0.04 0.03   34.83   34.79
##         skew kurtosis   se  Q0.1 Q0.25   Q0.5  Q0.75   Q0.9
## chisq  2.31     8.48 1.67 56.78 90.52 151.99 254.54 408.70
## df      NaN      NaN 0.00  7.00  7.00   7.00   7.00   7.00
## cfi   -0.66     0.02 0.00  0.51  0.63   0.74   0.84   0.90
## rmsea  0.96     1.27 0.00  0.19  0.24   0.32   0.42   0.54
## srmr  89.01  8457.13 0.00  0.09  0.10   0.13   0.16   0.19
```

```
pairs.panels(fit[, c("chisq", "cfi", "rmsea", "srmr")], hist.col="yellow")
```




```
save.image("SuppMat2.RData")

sessionInfo()
```

```
## R version 3.4.2 (2017-09-28)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Linux Mint 18.2
```

```
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=en_SG.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_SG.UTF-8        LC_COLLATE=en_SG.UTF-8
##  [5] LC_MONETARY=en_SG.UTF-8    LC_MESSAGES=en_SG.UTF-8
##  [7] LC_PAPER=en_SG.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_SG.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] Matrix_1.2-11      psych_1.7.8        MASS_7.3-47
## [4] lavaan_0.5-23.1097 matrixcalc_1.0-3   plyr_1.8.4
## [7] rmarkdown_1.6
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.13    knitr_1.17      magrittr_1.5    mnormt_1.5-5
##  [5] pbivnorm_0.6.0  lattice_0.20-35 quadprog_1.5-5  stringr_1.2.0
##  [9] tools_3.4.2     parallel_3.4.2  grid_3.4.2      nlme_3.1-131
## [13] htmltools_0.3.6 yaml_2.1.14     rprojroot_1.2   digest_0.6.12
## [17] evaluate_0.10.1 stringi_1.1.5   compiler_3.4.2  backports_1.1.1
## [21] stats4_3.4.2    foreign_0.8-69
```