CFA_OneFactor

Contents

1 Load packages & set working directory & read in data

```
library(matrixcalc);library(MASS);library(Matrix)
```

```
## Warning:  'matrixcalc' R 4.3.1
```

```
## Warning:  'Matrix' R 4.3.1
```

```
library(coda);library(R2OpenBUGS);library(metaSEM)
```

```
## Warning:  'coda' R 4.3.1
```

```
## Warning:  'R2OpenBUGS' R 4.3.2
```

```
##      OpenMx
```

```
##
##    'OpenMx'
```

```
## The following objects are masked from 'package:Matrix':
##
##     %&%, expm
```

```
## The following object is masked from 'package:matrixcalc':
##
##     vech
```

```
## "SLSQP" is set as the default optimizer in OpenMx.
```

```
## mxOption(NULL, "Gradient algorithm") is set at "central".
```

```
## mxOption(NULL, "Optimality tolerance") is set at "6.3e-14".
```

```
## mxOption(NULL, "Gradient iterations") is set at "2".
```

```
# Working directory
wd = 'D:/Research/2023/CompareMASEM/CFA/OneFactor/'
setwd(wd)
```

2 Functions

```
# vector to matrix
v2m <- function(vec,p,corr= T){
    M = matrix(0,p,p)
    M[lower.tri(M)] = vec
    M = M + t(M)
    if(corr==TRUE){
        diag(M) = 1
    }else{
        diag(M) = diag(M)/2
    }
    return(M)
}

# impute missing values in covariance / correlation matrices of each study
# to obtain a rough estimate of the covariance matrix of covariance / correlation matrix
# weighted average correlation
Mimpute <- function(R,N,missing){
    if(is.null(missing)){
        return(R)
    }else{
        na.pos = which(is.na(R),arr.ind = TRUE)
        mu.N = mean(N)
        Rbar = apply(R,2,mean,na.rm = TRUE)# Becker's mean r

        for(coli in unique(na.pos[,2])){
            id = na.pos[(na.pos[,2] == coli),1]
            R[id,coli] = Rbar[coli]
        }
        return(R)
    }
}

# change the coordinating system of a vectorized matrix to the coordinating system of
# the original matrix
# e.g., from vS to S, the former uses one coordinate (vil), whereas the latter uses two (j,k).
Get.vi2jk <- function(p,diag.incl=FALSE,byrow=FALSE){
    A = matrix(1,p,p)
    if(diag.incl ==FALSE){
        pp = p*(p-1)/2
        vi2jk <- matrix(NA,pp,3)
        vi2jk[,3] <- 1:pp
        if(byrow == FALSE){
            vi2jk[,1:2] <- which(lower.tri(A)==1,arr.ind = TRUE)
        }else{
            vi2jk[,1:2] <- which(upper.tri(A)==1,arr.ind = TRUE)
        }
        colnames(vi2jk) = c('j','k','vi')
    }else{
        pp = p*(p+1)/2
        vi2jk <- matrix(NA,pp,3)
        vi2jk[,3] <- 1:pp
        if(byrow == FALSE){
```

```
            vi2jk[,1:2] <- which(lower.tri(A,diag = TRUE)==1,arr.ind = TRUE)
        }else{
            vi2jk[,1:2] <- which(upper.tri(A,diag = TRUE)==1,arr.ind = TRUE)
        }
        colnames(vi2jk) = c('j','k','vi')
    }
    return(vi2jk)
}

# change the coordinating system of a matrix to the coordinating system of
# the corresponding vectorized matrix
# e.g., from S to vS, the former uses two coordinates (j,k), whereas the latter uses only one (vil).
Get.jk2vi <- function(vi2jk,p,diag.incl=FALSE){
    jk2vi = matrix(0,p,p)
    jk2vi[vi2jk[,1:2]] = vi2jk[,3]
    if(diag.incl){
        jk2vi = jk2vi + t(jk2vi)
        diag(jk2vi) = diag(jk2vi)/2
    }else{
        pp = p*(p-1)/2
        jk2vi = jk2vi + t(jk2vi) + diag(rep(pp+1,p))
    }
    return(jk2vi)
}

jkvil <- function(p){
    vi2jk   = Get.vi2jk(p)
    j   = vi2jk[,1]
    k   = vi2jk[,2]
    vil = Get.jk2vi(vi2jk,p)
    return(list(j=j,k=k,vil=vil))
}

# compute the covariance matrix of correlation matrix
# based on Steiger (1980)
Corr.Cov <- function(vR,N,index.list){
    nvR = length(vR)
    vR  = c(vR,1)
    NvR.cov = matrix(NA,nvR,nvR)
    j = index.list$j
    k = index.list$k
    vil = index.list$vil

    for(vi in 1:nvR){
        NvR.cov[vi,vi] = (1-(vR[vi])^2)^2
    }
    for(vi in 1:(nvR-1)){
    for(vj in (vi+1):nvR){
        NvR.cov[vi,vj] = ((vR[vil[j[vi],j[vj]]]-vR[vi]*vR[vil[k[vi],j[vj]]])*(vR[vil[k[vi],k[vj]]]-vR[v
         +(vR[vil[j[vi],k[vj]]]-vR[vil[j[vi],j[vj]]]*vR[vj])*(vR[vil[k[vi],j[vj]]]-vR[vi]*vR[vil[j[vi],
         +(vR[vil[j[vi],j[vj]]]-vR[vil[j[vi],k[vj]]]*vR[vj])*(vR[vil[k[vi],k[vj]]]-vR[vi]*vR[vil[j[vi],
         +(vR[vil[j[vi],k[vj]]]-vR[vi]*vR[vil[k[vi],k[vj]]])*(vR[vil[j[vj],k[vi]]]-vR[vil[k[vi],k[vj]]]
        NvR.cov[vj,vi] <- NvR.cov[vi,vj]
```

```
    }
    }

    vR.cov = NvR.cov/(N)
    vR.cov = as.matrix(nearPD(vR.cov,posd.tol = 1e-5)$mat)
    return(vR.cov)
}

# Use average correlation vector to compute V_psi
Vj <- function(vR.bar,N,pp,Nstudy,index.list){

    mu.N = mean(N)
    S.vR.bar = Corr.Cov(vR.bar,mu.N,index.list)
    inv.S.vR.bar = solve(S.vR.bar)
    tau.vR = array(NA,dim = c(Nstudy,pp,pp))
    S.vR = array(NA,dim = c(Nstudy,pp,pp))
    for(i in 1:Nstudy){
        S.vR[i,,]<- S.vR.bar/N[i]*mu.N
        tau.vR[i,,] <- inv.S.vR.bar/mu.N*N[i]
    }
    return(list(S.vR = S.vR,tau.vR = tau.vR))
}

# Use individual correlation vectors to compute V_psi
Vj2 <- function(vR.impute,N,pp,Nstudy,index.list){

    tau.vR = array(NA,dim = c(Nstudy,pp,pp))
    S.vR = array(NA,dim = c(Nstudy,pp,pp))
    for(i in 1:Nstudy){
        S.vR[i,,] = Corr.Cov(vR.impute[i,],N[i],index.list)
        tau.vR[i,,] <- solve(S.vR[i,,])
    }
    return(list(S.vR = S.vR,tau.vR = tau.vR))
}

# generate data for meta-analytic CFA
# the two-level model of OSMASEM is used
Gen.CFA.data <- function(Nstudy,mu.N,Model.list,p,missing,N=NULL){

    beta = Model.list$beta
    tau = Model.list$tau
    ind = Model.list$ind
    Z = Model.list$Z
    pp = Model.list$pp
    j = Model.list$j
    j10 = Model.list$j10
    k = Model.list$k
    k10 = Model.list$k10
    vil = Model.list$vil

    # predicted SEM parameters
    coefM <- Z%*%t(beta)
```

```
    # predicted part of the true correlation vector for each study
    vPs = t(apply(coefM,1,function(x,pp,j,k,j10,k10,ind){
        r = rep(NA,pp)
        for(vi in 1:pp){
          r[vi] = x[j[vi]]*x[k[vi]]+x[j10[vi]]*x[k10[vi]]*ind[vi]
        }
        return(r)
    },pp=pp,j=j,k=k,j10=j10,k10=k10,ind=ind) )

    # true correlation vector for each study
    if(tau[1]>0){
       vP = t(apply(vPs,1,function(x,tau,pp){
        r = rep(NA,pp)
        for(vi in 1:pp){ r[vi] = rnorm(1,x[vi],sd=tau[vi]) }
        return(r)
       },tau=tau,pp=pp) )
    }else{ vP=vPs }

    # sample size for each study
    if(is.null(N)){
      N <- rzinb(n =Nstudy, k =0.8, lambda=round(mu.N*0.2), omega = 0)
      N <- N + round(mu.N*0.8)
    }

    # observed correlations
    vR = matrix(NA,Nstudy,pp)
    for(studyi in 1:Nstudy){
        Pm = v2m(vP[studyi,],p,T)
        Pm = nearPD(Pm,corr=T)$mat
        Ri = cor(mvrnorm(N[studyi],rep(0,p),Pm))
        vR[studyi,] = Ri[lower.tri(Ri)]
    }

    #source(paste(wd,'RealData.R',sep=''))
    #vR = Make.Missing2(vR,missing,miss.rate,N) # generate missing values
    return(list(j=j,k=k,vil=vil,pp=pp,N=N,vR=vR,Z=Z))
}

d4osmasem <- function(dsim){
    j = dsim$j
    vR = dsim$vR
    N = dsim$N
    Z = as.matrix(dsim$Z)

    p = max(j)
    R.l = as.list(as.data.frame(t(vR)))
    Mat = lapply(R.l,function(x,p) v2m(x,p,T),p=p)
    my.df = Cor2DataFrame(Mat,N,acov = 'weighted')
    my.df$data = data.frame(my.df$data,covariate=scale(Z[,1]),check.names = FALSE)
    return(my.df)
}

wbugs <-function(data,initsl,prm,mfn,
```

```
    nchains=1,niter=60000,nburnin=30000,nthin=1,wd,
    diagm){
# data: a named list of the data in the likelihood model for OpenBUGS
# initsl: a list with nchains elements; each element is a list of starting values
# prm: vector of names of the parameters to save
# mfn: the file name of the likelihood model for OpenBUGS
# diagm: name of the convergence diagnostic method; either 'Geweke' or 'Gelman'
# The function checks convergence every niter-nburnin iterations

    fit = bugs(data,initsl,prm,mfn,
        n.chains=nchains,n.iter=niter,n.burnin=nburnin,n.thin=1,
        debug=F,saveExec=T,working.directory = wd)

    for(tryi in 2:20){
        print(paste0('Iteration: ',tryi*(niter-nburnin)))
        fit.coda = read.openbugs(stem="",thin = nthin)
        del.id = na.omit(match(c('ppp'),varnames(fit.coda)))
        print(summary(fit.coda),3)
        if(diagm=='Geweke'){
            if(length(del.id)>0){
                tmp.conv = geweke.diag(fit.coda[,-del.id])[[1]]$z
            }else{ tmp.conv = geweke.diag(fit.coda)[[1]]$z }
            crit = (sum((abs(tmp.conv)>1.96),na.rm = T)==0)
        }else if(diagm=='Gelman'){
            if(length(del.id)>0){
                tmp.conv = gelman.diag(fit.coda)$psrf[-del.id,2]
            }else{ tmp.conv = gelman.diag(fit.coda)$psrf[,2] }
            crit = (sum((tmp.conv>1.1),na.rm = T)==0)
        }
        if(crit){
            print(tmp.conv)
            print(summary(fit.coda),3)
            break
        }else{
            fit = bugs(data,initsl,prm,mfn,
            n.chains=nchains,n.iter=niter-nburnin+1,n.burnin=1,n.thin=1,
            restart=T,saveExec=T,working.directory = wd)
        }
    }
    ppp.id = match('ppp',prm)
    sel = NA
    if(is.na(ppp.id)){
        nprm = length(prm)
        for(i in 1:nprm){
            sel = c(sel,grep(prm[i],rownames(summary(fit.coda)$quantiles)))
        }
    }else{
        prm = prm[-ppp.id]
        nprm = length(prm)
        for(i in 1:nprm){
            sel = c(sel,grep(prm[i],rownames(summary(fit.coda)$quantiles)))
        }
    }
```

```
    sel = sel[-1]
    sel = unique(sel)

    if(is.na(ppp.id)){ est = round(summary(fit.coda)$quantiles[sel,'50%'],3)
    }else{
        est = round(c(summary(fit.coda)$quantiles[sel,'50%'],
        summary(fit.coda)$statistics['ppp','Mean']),3)
    }
    psd = round(summary(fit.coda)$statistics[sel,'SD'],3)
    if(diagm=='Geweke'){
        CIl = round(HPDinterval(fit.coda,prob = .95)[[1]][sel,1],3)
        CIu = round(HPDinterval(fit.coda,prob = .95)[[1]][sel,2],3)
    }else if(diagm=='Gelman'){
        fit.coda.l = do.call(rbind,fit.coda)
        HPDCI = HPDinterval(mcmc(fit.coda.l),prob = .95)
        CIl = HPDCI[sel,1]
        CIu = HPDCI[sel,2]
    }
    sel.muL = grep('mu.L',names(est))
    sel.sdL = grep('sd.L',names(est))
    CVl = round(est[sel.muL] - 1.28*est[sel.sdL],3)
    CVu = round(est[sel.muL] + 1.28*est[sel.sdL],3)

    conv = round(c(tryi,tmp.conv),3)
    return(list(est=est,psd=psd,CIl=CIl,CIu=CIu,CVl=CVl,CVu=CVu,conv=conv,
        DIC=fit$DIC,fit.coda=fit.coda))
}
```

3 BMASEM

3.1 Data preparation

```
## Exclude studies that did not report bivariate correlations
index <- Gnambs18$CorMat==1
Gnambs18 <- lapply(Gnambs18, function(x) x[index])

# Convert correlation matrices to correlation vectors
mR = Gnambs18$data
vR = sapply(mR,function(x){ x = x[c(1,3,4,7,10,2,5,6,8,9),c(1,3,4,7,10,2,5,6,8,9)]
    return(x[lower.tri(x)]) })
vR = t(vR)

N      = Gnambs18$n # sample sizes within primary studies
mu.N   = mean(N) # mean sample size
Nstudy = length(Gnambs18$data) # the number of primary studies
Ninv   = 1/N # reciprocals of sample sizes

# Coordinates of correlation matrices and vectors
p  = 10 # number of variables
pp = p*(p-1)/2  # number of bivariate correlations
index.list = jkvil(p)
j = index.list$j
```

```
k = index.list$k
vil = index.list$vil
ind = (j>(p+1)/2)*(k<(p+2)/2)

# Covariance matrices of sample correlation vectors
vR.bar = apply(vR,2,mean,na.rm = TRUE)
Stau.vR = Vj(vR.bar,N,pp,Nstudy,index.list)
tau.vR = Stau.vR$tau.vR

# information for the additional error term
mu.vR.psi = rep(0,pp)
df.prelim = 100*pp/mu.N+pp
alpha.prior.vE = (df.prelim-pp+1)/2
beta.prior.vE = alpha.prior.vE*(0.3/mu.N)

# Matrices for computing ppp
# Compute the between-study covariance matrix of true study-specific correlation vectors
# Z: First derivative of study-specific correlation vectors with respect to model
#    parameters (factor loadings)
# NA: for Openbugs to replace with parameter estimates
# The vi_th element in the vectorized correlation matrix corresponds to the
# correlation between the j_th and the k_th items.
# In the bifactor model, the correlation between the j_th and the_kth items
# equals the product of the j_th and the_kth
# factor loadings plus the product of the (j+10)_th and the (k+10)_th factor
# loadings (the factor loadings of the method factors) if the two items are
# loaded on the same method factor. Therefore, the first derivative of the vi_th
# correlation equals a nonzero value when the derivative is with respect to the
# j(+10)_th or the k(+10)_th factor loading and zero when it is with
# respect to other SEM parameters
Z <- matrix(0,pp,p)
for(vi in 1:pp){    Z[vi,c(j[vi],k[vi])] = NA   }
# Diagonal covariance matrix of study-specific model parameters (factor loadings)
# Random factor loadings are assumed to be uncorrelated
V.theta = matrix(0,10,10)
diag(V.theta) = NA
```

3.2 Model fitting

```
data<-list("Nstudy","N","Ninv","mu.N",'p',"pp","j","k",'V.theta',"vR",
  "tau.vR",'Z',"mu.vR.psi",'alpha.prior.vE','beta.prior.vE')  # data

initsl <- list(list(mu.L=rep(.6,p),sd.L = rep(0.1,p),
  tau.R=mu.N*3,vR.psi = matrix(0,Nstudy,pp),vR.rep = vR))# Initial values

prm = c('mu.L','sd.L','ppp') # parameters to save
model.fn = paste(wd,'CFARandom.txt',sep='') # model file name

# stop every 10000 iterations to check whether convergence is achieved
fit = wbugs(data,initsl,prm,model.fn,
        nchains=1,niter=60000,nburnin=30000,nthin=1,wd,diagm='Geweke')
```

```
## [1] "Iteration: 60000"
```

```
## Abstracting deviance ... 30000 valid values
## Abstracting mu.L[1] ... 30000 valid values
## Abstracting mu.L[2] ... 30000 valid values
## Abstracting mu.L[3] ... 30000 valid values
## Abstracting mu.L[4] ... 30000 valid values
## Abstracting mu.L[5] ... 30000 valid values
## Abstracting mu.L[6] ... 30000 valid values
## Abstracting mu.L[7] ... 30000 valid values
## Abstracting mu.L[8] ... 30000 valid values
## Abstracting mu.L[9] ... 30000 valid values
## Abstracting mu.L[10] ... 30000 valid values
## Abstracting ppp ... 30000 valid values
## Abstracting sd.L[1] ... 30000 valid values
## Abstracting sd.L[2] ... 30000 valid values
## Abstracting sd.L[3] ... 30000 valid values
## Abstracting sd.L[4] ... 30000 valid values
## Abstracting sd.L[5] ... 30000 valid values
## Abstracting sd.L[6] ... 30000 valid values
## Abstracting sd.L[7] ... 30000 valid values
## Abstracting sd.L[8] ... 30000 valid values
## Abstracting sd.L[9] ... 30000 valid values
## Abstracting sd.L[10] ... 30000 valid values
##
## Iterations = 30001:60000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 30000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD Naive SE Time-series SE
## deviance -2.15e+03 30.3561 1.75e-01       0.228323
## mu.L[1]   6.74e-01  0.0209 1.21e-04       0.000189
## mu.L[2]   5.91e-01  0.0189 1.09e-04       0.000177
## mu.L[3]   5.20e-01  0.0187 1.08e-04       0.000174
## mu.L[4]   6.01e-01  0.0244 1.41e-04       0.000199
## mu.L[5]   7.21e-01  0.0172 9.91e-05       0.000173
## mu.L[6]   6.84e-01  0.0170 9.79e-05       0.000175
## mu.L[7]   6.08e-01  0.0210 1.21e-04       0.000183
## mu.L[8]   6.74e-01  0.0136 7.86e-05       0.000179
## mu.L[9]   4.93e-01  0.0401 2.32e-04       0.000269
## mu.L[10]  6.93e-01  0.0179 1.03e-04       0.000173
## ppp       8.50e-01  0.3571 2.06e-03       0.002096
## sd.L[1]   9.65e-02  0.0179 1.03e-04       0.000201
## sd.L[2]   8.40e-02  0.0152 8.77e-05       0.000163
## sd.L[3]   8.45e-02  0.0146 8.43e-05       0.000151
## sd.L[4]   1.20e-01  0.0195 1.12e-04       0.000179
## sd.L[5]   7.16e-02  0.0144 8.32e-05       0.000182
## sd.L[6]   7.03e-02  0.0145 8.40e-05       0.000185
## sd.L[7]   9.86e-02  0.0172 9.93e-05       0.000177
## sd.L[8]   4.65e-02  0.0131 7.54e-05       0.000242
## sd.L[9]   2.20e-01  0.0319 1.84e-04       0.000253
## sd.L[10]  7.71e-02  0.0153 8.81e-05       0.000183
```

```
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance -2.20e+03 -2.17e+03 -2.15e+03 -2.13e+03 -2.09e+03
## mu.L[1]   6.33e-01  6.60e-01  6.74e-01  6.88e-01  7.15e-01
## mu.L[2]   5.54e-01  5.79e-01  5.91e-01  6.04e-01  6.29e-01
## mu.L[3]   4.83e-01  5.08e-01  5.20e-01  5.32e-01  5.57e-01
## mu.L[4]   5.53e-01  5.84e-01  6.00e-01  6.17e-01  6.49e-01
## mu.L[5]   6.87e-01  7.10e-01  7.21e-01  7.33e-01  7.55e-01
## mu.L[6]   6.50e-01  6.72e-01  6.84e-01  6.95e-01  7.17e-01
## mu.L[7]   5.67e-01  5.95e-01  6.09e-01  6.22e-01  6.50e-01
## mu.L[8]   6.46e-01  6.65e-01  6.74e-01  6.83e-01  7.00e-01
## mu.L[9]   4.13e-01  4.66e-01  4.93e-01  5.19e-01  5.72e-01
## mu.L[10]  6.58e-01  6.82e-01  6.93e-01  7.05e-01  7.29e-01
## ppp       0.00e+00  1.00e+00  1.00e+00  1.00e+00  1.00e+00
## sd.L[1]   6.64e-02  8.38e-02  9.48e-02  1.07e-01  1.36e-01
## sd.L[2]   5.88e-02  7.33e-02  8.25e-02  9.31e-02  1.18e-01
## sd.L[3]   5.98e-02  7.43e-02  8.31e-02  9.31e-02  1.17e-01
## sd.L[4]   8.82e-02  1.07e-01  1.18e-01  1.32e-01  1.64e-01
## sd.L[5]   4.72e-02  6.13e-02  7.03e-02  8.02e-02  1.04e-01
## sd.L[6]   4.61e-02  6.01e-02  6.88e-02  7.89e-02  1.03e-01
## sd.L[7]   6.99e-02  8.64e-02  9.69e-02  1.09e-01  1.37e-01
## sd.L[8]   2.37e-02  3.76e-02  4.57e-02  5.46e-02  7.48e-02
## sd.L[9]   1.67e-01  1.98e-01  2.17e-01  2.39e-01  2.92e-01
## sd.L[10]  5.14e-02  6.64e-02  7.55e-02  8.62e-02  1.11e-01
##
## [1] "Iteration: 90000"
## Abstracting deviance ... 30000 valid values
## Abstracting mu.L[1] ... 30000 valid values
## Abstracting mu.L[2] ... 30000 valid values
## Abstracting mu.L[3] ... 30000 valid values
## Abstracting mu.L[4] ... 30000 valid values
## Abstracting mu.L[5] ... 30000 valid values
## Abstracting mu.L[6] ... 30000 valid values
## Abstracting mu.L[7] ... 30000 valid values
## Abstracting mu.L[8] ... 30000 valid values
## Abstracting mu.L[9] ... 30000 valid values
## Abstracting mu.L[10] ... 30000 valid values
## Abstracting ppp ... 30000 valid values
## Abstracting sd.L[1] ... 30000 valid values
## Abstracting sd.L[2] ... 30000 valid values
## Abstracting sd.L[3] ... 30000 valid values
## Abstracting sd.L[4] ... 30000 valid values
## Abstracting sd.L[5] ... 30000 valid values
## Abstracting sd.L[6] ... 30000 valid values
## Abstracting sd.L[7] ... 30000 valid values
## Abstracting sd.L[8] ... 30000 valid values
## Abstracting sd.L[9] ... 30000 valid values
## Abstracting sd.L[10] ... 30000 valid values
##
## Iterations = 60002:90001
## Thinning interval = 1
## Number of chains = 1
```

```
## Sample size per chain = 30000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD Naive SE Time-series SE
## deviance -2.15e+03 30.5030 1.76e-01       0.223145
## mu.L[1]   6.74e-01  0.0206 1.19e-04       0.000181
## mu.L[2]   5.91e-01  0.0189 1.09e-04       0.000176
## mu.L[3]   5.20e-01  0.0191 1.10e-04       0.000179
## mu.L[4]   6.01e-01  0.0246 1.42e-04       0.000204
## mu.L[5]   7.21e-01  0.0170 9.84e-05       0.000172
## mu.L[6]   6.84e-01  0.0167 9.65e-05       0.000171
## mu.L[7]   6.09e-01  0.0211 1.22e-04       0.000182
## mu.L[8]   6.74e-01  0.0135 7.79e-05       0.000181
## mu.L[9]   4.93e-01  0.0399 2.31e-04       0.000275
## mu.L[10]  6.94e-01  0.0179 1.04e-04       0.000175
## ppp       8.54e-01  0.3527 2.04e-03       0.002037
## sd.L[1]   9.65e-02  0.0179 1.03e-04       0.000200
## sd.L[2]   8.42e-02  0.0152 8.76e-05       0.000163
## sd.L[3]   8.47e-02  0.0146 8.42e-05       0.000147
## sd.L[4]   1.21e-01  0.0194 1.12e-04       0.000181
## sd.L[5]   7.15e-02  0.0146 8.42e-05       0.000180
## sd.L[6]   6.97e-02  0.0143 8.28e-05       0.000180
## sd.L[7]   9.85e-02  0.0170 9.83e-05       0.000172
## sd.L[8]   4.63e-02  0.0133 7.67e-05       0.000260
## sd.L[9]   2.20e-01  0.0321 1.85e-04       0.000256
## sd.L[10]  7.72e-02  0.0155 8.95e-05       0.000188
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance -2.20e+03 -2.17e+03 -2.15e+03 -2.12e+03 -2.08e+03
## mu.L[1]   6.33e-01  6.60e-01  6.74e-01  6.87e-01  7.15e-01
## mu.L[2]   5.54e-01  5.79e-01  5.91e-01  6.04e-01  6.29e-01
## mu.L[3]   4.82e-01  5.07e-01  5.20e-01  5.32e-01  5.58e-01
## mu.L[4]   5.53e-01  5.85e-01  6.01e-01  6.17e-01  6.50e-01
## mu.L[5]   6.87e-01  7.10e-01  7.21e-01  7.33e-01  7.54e-01
## mu.L[6]   6.51e-01  6.73e-01  6.84e-01  6.95e-01  7.17e-01
## mu.L[7]   5.67e-01  5.95e-01  6.09e-01  6.22e-01  6.50e-01
## mu.L[8]   6.47e-01  6.65e-01  6.74e-01  6.83e-01  7.00e-01
## mu.L[9]   4.14e-01  4.67e-01  4.93e-01  5.20e-01  5.72e-01
## mu.L[10]  6.58e-01  6.82e-01  6.94e-01  7.06e-01  7.29e-01
## ppp       0.00e+00  1.00e+00  1.00e+00  1.00e+00  1.00e+00
## sd.L[1]   6.65e-02  8.38e-02  9.48e-02  1.07e-01  1.37e-01
## sd.L[2]   5.88e-02  7.35e-02  8.28e-02  9.33e-02  1.18e-01
## sd.L[3]   6.04e-02  7.44e-02  8.33e-02  9.35e-02  1.18e-01
## sd.L[4]   8.80e-02  1.07e-01  1.19e-01  1.32e-01  1.64e-01
## sd.L[5]   4.68e-02  6.12e-02  7.02e-02  8.02e-02  1.04e-01
## sd.L[6]   4.52e-02  5.96e-02  6.84e-02  7.83e-02  1.01e-01
## sd.L[7]   6.98e-02  8.66e-02  9.71e-02  1.09e-01  1.36e-01
## sd.L[8]   2.34e-02  3.71e-02  4.52e-02  5.45e-02  7.54e-02
## sd.L[9]   1.67e-01  1.98e-01  2.17e-01  2.39e-01  2.93e-01
## sd.L[10]  5.12e-02  6.62e-02  7.58e-02  8.67e-02  1.11e-01
```

```
##
##    deviance     mu.L[1]     mu.L[2]     mu.L[3]     mu.L[4]     mu.L[5]
##  1.32571172 -0.65948446  0.04115212 -0.64833428 -0.65084409  1.75573929
##     mu.L[6]     mu.L[7]     mu.L[8]     mu.L[9]    mu.L[10]     sd.L[1]
## -0.96992580 -1.48992522 -0.96562868 -0.34304174 -0.93091594  1.18045801
##     sd.L[2]     sd.L[3]     sd.L[4]     sd.L[5]     sd.L[6]     sd.L[7]
## -1.34253887 -0.59991929 -0.97339531 -0.68671124 -1.12397645  0.44172020
##     sd.L[8]     sd.L[9]    sd.L[10]
##  1.04774141  0.16520020  1.22542406
##
## Iterations = 60002:90001
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 30000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD Naive SE Time-series SE
## deviance -2.15e+03 30.5030 1.76e-01       0.223145
## mu.L[1]   6.74e-01  0.0206 1.19e-04       0.000181
## mu.L[2]   5.91e-01  0.0189 1.09e-04       0.000176
## mu.L[3]   5.20e-01  0.0191 1.10e-04       0.000179
## mu.L[4]   6.01e-01  0.0246 1.42e-04       0.000204
## mu.L[5]   7.21e-01  0.0170 9.84e-05       0.000172
## mu.L[6]   6.84e-01  0.0167 9.65e-05       0.000171
## mu.L[7]   6.09e-01  0.0211 1.22e-04       0.000182
## mu.L[8]   6.74e-01  0.0135 7.79e-05       0.000181
## mu.L[9]   4.93e-01  0.0399 2.31e-04       0.000275
## mu.L[10]  6.94e-01  0.0179 1.04e-04       0.000175
## ppp       8.54e-01  0.3527 2.04e-03       0.002037
## sd.L[1]   9.65e-02  0.0179 1.03e-04       0.000200
## sd.L[2]   8.42e-02  0.0152 8.76e-05       0.000163
## sd.L[3]   8.47e-02  0.0146 8.42e-05       0.000147
## sd.L[4]   1.21e-01  0.0194 1.12e-04       0.000181
## sd.L[5]   7.15e-02  0.0146 8.42e-05       0.000180
## sd.L[6]   6.97e-02  0.0143 8.28e-05       0.000180
## sd.L[7]   9.85e-02  0.0170 9.83e-05       0.000172
## sd.L[8]   4.63e-02  0.0133 7.67e-05       0.000260
## sd.L[9]   2.20e-01  0.0321 1.85e-04       0.000256
## sd.L[10]  7.72e-02  0.0155 8.95e-05       0.000188
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance -2.20e+03 -2.17e+03 -2.15e+03 -2.12e+03 -2.08e+03
## mu.L[1]   6.33e-01  6.60e-01  6.74e-01  6.87e-01  7.15e-01
## mu.L[2]   5.54e-01  5.79e-01  5.91e-01  6.04e-01  6.29e-01
## mu.L[3]   4.82e-01  5.07e-01  5.20e-01  5.32e-01  5.58e-01
## mu.L[4]   5.53e-01  5.85e-01  6.01e-01  6.17e-01  6.50e-01
## mu.L[5]   6.87e-01  7.10e-01  7.21e-01  7.33e-01  7.54e-01
## mu.L[6]   6.51e-01  6.73e-01  6.84e-01  6.95e-01  7.17e-01
## mu.L[7]   5.67e-01  5.95e-01  6.09e-01  6.22e-01  6.50e-01
## mu.L[8]   6.47e-01  6.65e-01  6.74e-01  6.83e-01  7.00e-01
```

```
## mu.L[9]   4.14e-01  4.67e-01  4.93e-01  5.20e-01  5.72e-01
## mu.L[10]  6.58e-01  6.82e-01  6.94e-01  7.06e-01  7.29e-01
## ppp       0.00e+00  1.00e+00  1.00e+00  1.00e+00  1.00e+00
## sd.L[1]   6.65e-02  8.38e-02  9.48e-02  1.07e-01  1.37e-01
## sd.L[2]   5.88e-02  7.35e-02  8.28e-02  9.33e-02  1.18e-01
## sd.L[3]   6.04e-02  7.44e-02  8.33e-02  9.35e-02  1.18e-01
## sd.L[4]   8.80e-02  1.07e-01  1.19e-01  1.32e-01  1.64e-01
## sd.L[5]   4.68e-02  6.12e-02  7.02e-02  8.02e-02  1.04e-01
## sd.L[6]   4.52e-02  5.96e-02  6.84e-02  7.83e-02  1.01e-01
## sd.L[7]   6.98e-02  8.66e-02  9.71e-02  1.09e-01  1.36e-01
## sd.L[8]   2.34e-02  3.71e-02  4.52e-02  5.45e-02  7.54e-02
## sd.L[9]   1.67e-01  1.98e-01  2.17e-01  2.39e-01  2.93e-01
## sd.L[10]  5.12e-02  6.62e-02  7.58e-02  8.67e-02  1.11e-01
```

```
fit[-9]
```

```
## $est
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.674    0.591    0.520    0.601    0.721    0.684    0.609    0.674
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.493    0.694    0.095    0.083    0.083    0.119    0.070    0.068
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]
##    0.097    0.045    0.217    0.076    0.854
##
## $psd
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.021    0.019    0.019    0.025    0.017    0.017    0.021    0.013
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.040    0.018    0.018    0.015    0.015    0.019    0.015    0.014
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]
##    0.017    0.013    0.032    0.016
##
## $CIl
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.632    0.553    0.481    0.552    0.686    0.652    0.569    0.648
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.415    0.657    0.064    0.057    0.057    0.085    0.045    0.044
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]
##    0.068    0.022    0.163    0.049
##
## $CIu
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.714    0.627    0.557    0.649    0.754    0.717    0.651    0.701
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.572    0.728    0.132    0.115    0.113    0.160    0.101    0.099
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]
##    0.133    0.073    0.285    0.108
##
## $CVl
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.552    0.485    0.414    0.449    0.631    0.597    0.485    0.616
##  mu.L[9] mu.L[10]
##    0.215    0.597
##
## $CVu
```

```
##  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]  mu.L[7]  mu.L[8]
##    0.796    0.697    0.626    0.753    0.811    0.771    0.733    0.732
##  mu.L[9] mu.L[10]
##    0.771    0.791
##
## $conv
##          deviance  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]
##    3.000    1.326   -0.659    0.041   -0.648   -0.651    1.756   -0.970
##  mu.L[7]  mu.L[8]  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]
##   -1.490   -0.966   -0.343   -0.931    1.180   -1.343   -0.600   -0.973
##  sd.L[5]  sd.L[6]  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]
##   -0.687   -1.124    0.442    1.048    0.165    1.225
##
## $DIC
## [1] -1900
```

4 OSMASEM

4.1 Data preparation

```
# Modified based on the code from Jak & Cheung (2019)
# Exclude studies that reported CFA results only
index <- Gnambs18$CorMat==1
Gnambs18 <- lapply(Gnambs18, function(x) x[index])

## Create a dataframe with the data and the asymptotic variances and covariances (acov)
my.df <- Cor2DataFrame(Gnambs18$data, Gnambs18$n, acov = "weighted")

## Add the standardized individualism as the moderator
## Standardization of the moderator improves the convergence.
my.df$data <- data.frame(my.df$data,
                         Individualism=scale(Gnambs18$Individualism),
                         check.names=FALSE)
summary(my.df)
```

```
##           Length Class      Mode
## data      1081   data.frame list
## n           36   -none-     numeric
## obslabels   10   -none-     character
## ylabels     45   -none-     character
## vlabels   1035   -none-     character
```

4.2 Model fitting

```
## Specify the bifactor model
model0 <- "SE =~ p1*I1 + p3*I3 + p4*I4 + p7*I7 + p10*I10 +
                 n2*I2 + n5*I5 + n6*I6 + n8*I8 + n9*I9"

RAM0 <- lavaan2RAM(model0, obs.variables = paste0("I", 1:10),std.lv = TRUE)

## Create matrices with implicit diagonal constraints
M0 <- create.vechsR(A0=RAM0$A, S0=RAM0$S, F0=RAM0$F)
```

```r
## Create heterogeneity variances
T0 <- create.Tau2(RAM=RAM0, RE.type="Diag", Transform="expLog", RE.startvalues=0.05)

## Fit the bifactor model with One-Stage MASEM
fit0 <- osmasem(model.name="No moderator", Mmatrix=M0, Tmatrix=T0, data=my.df)
summary(fit0, fitIndices= T)
```

```
## Summary of No moderator
##
## free parameters:
##       name  matrix row col   Estimate  Std.Error A    z value Pr(>|z|)
## 1       p1      A0  I1  SE  0.6913374 0.01521577    45.435579        0
## 2       n2      A0  I2  SE  0.6361038 0.01809109    35.161172        0
## 3       p3      A0  I3  SE  0.5566804 0.01509044    36.889610        0
## 4       p4      A0  I4  SE  0.4968529 0.01271403    39.079109        0
## 5       n5      A0  I5  SE  0.6448447 0.01595716    40.410983        0
## 6       n6      A0  I6  SE  0.5927663 0.01649345    35.939507        0
## 7       p7      A0  I7  SE  0.5803234 0.01426067    40.693961        0
## 8       n8      A0  I8  SE  0.5058110 0.01749385    28.913649        0
## 9       n9      A0  I9  SE  0.6819481 0.01769374    38.541764        0
## 10     p10      A0 I10  SE  0.7434107 0.01639345    45.348036        0
## 11  Tau1_1 vecTau1   1   1 -4.5469002 0.26358475   -17.250240        0
## 12  Tau1_2 vecTau1   2   1 -4.7616980 0.29496824   -16.143087        0
## 13  Tau1_3 vecTau1   3   1 -4.8490362 0.28408749   -17.068813        0
## 14  Tau1_4 vecTau1   4   1 -3.9648819 0.27421236   -14.459165        0
## 15  Tau1_5 vecTau1   5   1 -5.2967090 0.27339319   -19.373961        0
## 16  Tau1_6 vecTau1   6   1 -4.3638690 0.25947337   -16.818177        0
## 17  Tau1_7 vecTau1   7   1 -3.6206351 0.24764682   -14.620156        0
## 18  Tau1_8 vecTau1   8   1 -4.7134679 0.26811984   -17.579706        0
## 19  Tau1_9 vecTau1   9   1 -3.7363214 0.29380031   -12.717214        0
## 20 Tau1_10 vecTau1  10   1 -4.7981796 0.27647069   -17.355111        0
## 21 Tau1_11 vecTau1  11   1 -5.0929861 0.27448574   -18.554647        0
## 22 Tau1_12 vecTau1  12   1 -4.8175132 0.33569027   -14.351066        0
## 23 Tau1_13 vecTau1  13   1 -2.4891070 0.26557503    -9.372519        0
## 24 Tau1_14 vecTau1  14   1 -4.5375724 0.25546961   -17.761691        0
## 25 Tau1_15 vecTau1  15   1 -3.6206888 0.27053148   -13.383614        0
## 26 Tau1_16 vecTau1  16   1 -4.1043839 0.35039340   -11.713645        0
## 27 Tau1_17 vecTau1  17   1 -4.7834807 0.26928975   -17.763323        0
## 28 Tau1_18 vecTau1  18   1 -3.1672770 0.26218904   -12.080127        0
## 29 Tau1_19 vecTau1  19   1 -4.6895427 0.25776227   -18.193286        0
## 30 Tau1_20 vecTau1  20   1 -5.1502339 0.29198525   -17.638678        0
## 31 Tau1_21 vecTau1  21   1 -2.7422955 0.25441363   -10.778886        0
## 32 Tau1_22 vecTau1  22   1 -3.9398756 0.26632011   -14.793759        0
## 33 Tau1_23 vecTau1  23   1 -5.0441111 0.27284327   -18.487211        0
## 34 Tau1_24 vecTau1  24   1 -4.7189190 0.33072847   -14.268258        0
## 35 Tau1_25 vecTau1  25   1 -4.9240723 0.27396808   -17.973161        0
## 36 Tau1_26 vecTau1  26   1 -5.3414065 0.27463369   -19.449204        0
## 37 Tau1_27 vecTau1  27   1 -3.5186753 0.26259511   -13.399622        0
## 38 Tau1_28 vecTau1  28   1 -3.9479970 0.25903199   -15.241349        0
## 39 Tau1_29 vecTau1  29   1 -4.8788274 0.27293846   -17.875192        0
## 40 Tau1_30 vecTau1  30   1 -5.2325609 0.31439923   -16.643046        0
## 41 Tau1_31 vecTau1  31   1 -4.6342740 0.34304086   -13.509394        0
## 42 Tau1_32 vecTau1  32   1 -4.5452675 0.26080400   -17.427905        0
## 43 Tau1_33 vecTau1  33   1 -4.5868601 0.26548061   -17.277571        0
```

```
## 44 Tau1_34 vecTau1  34   1 -4.1485370 0.26972619   -15.380549        0
## 45 Tau1_35 vecTau1  35   1 -4.0319360 0.29423366   -13.703177        0
## 46 Tau1_36 vecTau1  36   1 -4.9585466 0.26590586   -18.647752        0
## 47 Tau1_37 vecTau1  37   1 -3.4594769 0.26584241   -13.013262        0
## 48 Tau1_38 vecTau1  38   1 -3.9094573 0.33606114   -11.633173        0
## 49 Tau1_39 vecTau1  39   1 -5.2975280 0.27643605   -19.163666        0
## 50 Tau1_40 vecTau1  40   1 -3.8573246 0.25589409   -15.073911        0
## 51 Tau1_41 vecTau1  41   1 -4.7249318 0.25106984   -18.819193        0
## 52 Tau1_42 vecTau1  42   1 -4.5814918 0.28391910   -16.136610        0
## 53 Tau1_43 vecTau1  43   1 -3.8846465 0.25950972   -14.969175        0
## 54 Tau1_44 vecTau1  44   1 -3.4415777 0.24603303   -13.988275        0
## 55 Tau1_45 vecTau1  45   1 -4.7024356 0.26539494   -17.718633        0
##
## To obtain confidence intervals re-run with intervals=TRUE
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             55                  1565             -2119.665
##    Saturated:             90                  1530             -2777.449
## Independence:             45                  1575              1549.587
## Number of observations/statistics: 109988/1620
##
## chi-square:  ² ( df=35 ) = 657.7836,  p = 6.096805e-116
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -5249.665              -2009.665               -2009.609
## BIC:     -20286.383              -1481.218               -1656.010
## CFI: 0.854559
## TLI: 0.8130044   (also known as NNFI)
## RMSEA:  0.01271926  [95% CI (0.01171613, 0.01374117)]
## Prob(RMSEA <= 0.05): 1
## timestamp: 2023-12-13 14:03:10
## Wall clock time: 58.75376 secs
## optimizer:  SLSQP
## OpenMx version number: 2.21.8
## Need help?  See help(mxSummary)
```

```
## SRMR
osmasemSRMR(fit0)
```

```
## [1] 0.08474015
```