CFA_CorrelatedTraits

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/CorrelatedTraits/'
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){
```

```r
            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
```

```r
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+1)
for(vi in 1:pp){    Z[vi,c(j[vi],k[vi])] = NA   }
Z[,p+1] = NA
# Diagonal covariance matrix of study-specific model parameters (factor loadings)
# Random factor loadings are assumed to be uncorrelated
V.theta = matrix(0,11,11)
diag(V.theta) = NA
```

3.2 Model fitting

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

initsl <- list(list(mu.L=rep(.6,p),mu.rho = 0,sd.L = rep(0.1,p),sd.rho = 0.2,
  tau.R=mu.N*3,vR.psi = matrix(0,Nstudy,pp),vR.rep = vR))# Initial values


prm = c('mu.L','sd.L','mu.rho','sd.rho','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 mu.rho ... 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
## Abstracting sd.rho ... 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 -3.38e+03 35.88545 2.07e-01       0.295598
## mu.L[1]   7.27e-01  0.01870 1.08e-04       0.000153
## mu.L[2]   6.51e-01  0.01633 9.43e-05       0.000147
## mu.L[3]   5.69e-01  0.01689 9.75e-05       0.000148
## mu.L[4]   6.58e-01  0.02272 1.31e-04       0.000172
## mu.L[5]   7.84e-01  0.01327 7.66e-05       0.000139
## mu.L[6]   7.45e-01  0.01302 7.52e-05       0.000131
## mu.L[7]   6.51e-01  0.01881 1.09e-04       0.000149
## mu.L[8]   7.35e-01  0.00846 4.89e-05       0.000140
## mu.L[9]   5.33e-01  0.03856 2.23e-04       0.000255
## mu.L[10]  7.45e-01  0.01557 8.99e-05       0.000141
## mu.rho    7.19e-01  0.02447 1.41e-04       0.000184
## ppp       6.91e-01  0.46219 2.67e-03       0.002768
## sd.L[1]   9.18e-02  0.01629 9.41e-05       0.000172
## sd.L[2]   7.65e-02  0.01307 7.55e-05       0.000132
## sd.L[3]   7.99e-02  0.01293 7.47e-05       0.000119
## sd.L[4]   1.17e-01  0.01809 1.04e-04       0.000160
## sd.L[5]   5.53e-02  0.01144 6.61e-05       0.000146
## sd.L[6]   5.49e-02  0.01122 6.48e-05       0.000144
```

```
## sd.L[7]   9.34e-02  0.01527 8.82e-05       0.000143
## sd.L[8]   2.42e-02  0.00971 5.61e-05       0.000261
## sd.L[9]   2.19e-01  0.03102 1.79e-04       0.000239
## sd.L[10]  7.12e-02  0.01345 7.77e-05       0.000154
## sd.rho    1.31e-01  0.02106 1.22e-04       0.000210
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance -3.45e+03 -3.40e+03 -3.38e+03 -3.36e+03 -3.31e+03
## mu.L[1]   6.90e-01  7.15e-01  7.28e-01  7.40e-01  7.64e-01
## mu.L[2]   6.20e-01  6.40e-01  6.51e-01  6.62e-01  6.84e-01
## mu.L[3]   5.36e-01  5.57e-01  5.69e-01  5.80e-01  6.02e-01
## mu.L[4]   6.13e-01  6.43e-01  6.57e-01  6.73e-01  7.03e-01
## mu.L[5]   7.57e-01  7.75e-01  7.84e-01  7.92e-01  8.10e-01
## mu.L[6]   7.20e-01  7.37e-01  7.45e-01  7.54e-01  7.71e-01
## mu.L[7]   6.14e-01  6.38e-01  6.51e-01  6.63e-01  6.88e-01
## mu.L[8]   7.18e-01  7.29e-01  7.35e-01  7.41e-01  7.51e-01
## mu.L[9]   4.56e-01  5.07e-01  5.33e-01  5.58e-01  6.09e-01
## mu.L[10]  7.15e-01  7.35e-01  7.45e-01  7.56e-01  7.76e-01
## mu.rho    6.71e-01  7.03e-01  7.19e-01  7.36e-01  7.67e-01
## ppp       0.00e+00  0.00e+00  1.00e+00  1.00e+00  1.00e+00
## sd.L[1]   6.45e-02  8.02e-02  9.01e-02  1.02e-01  1.28e-01
## sd.L[2]   5.46e-02  6.73e-02  7.52e-02  8.43e-02  1.06e-01
## sd.L[3]   5.85e-02  7.07e-02  7.85e-02  8.77e-02  1.09e-01
## sd.L[4]   8.69e-02  1.05e-01  1.16e-01  1.28e-01  1.57e-01
## sd.L[5]   3.57e-02  4.73e-02  5.42e-02  6.21e-02  8.07e-02
## sd.L[6]   3.60e-02  4.69e-02  5.38e-02  6.17e-02  7.97e-02
## sd.L[7]   6.78e-02  8.25e-02  9.20e-02  1.02e-01  1.27e-01
## sd.L[8]   7.93e-03  1.74e-02  2.34e-02  3.00e-02  4.58e-02
## sd.L[9]   1.67e-01  1.97e-01  2.16e-01  2.38e-01  2.88e-01
## sd.L[10]  4.79e-02  6.18e-02  7.00e-02  7.94e-02  1.01e-01
## sd.rho    9.60e-02  1.17e-01  1.29e-01  1.44e-01  1.78e-01
##
##    deviance     mu.L[1]     mu.L[2]     mu.L[3]     mu.L[4]     mu.L[5]
##  0.65462737  0.95108081 -0.27493111  0.26089803  0.61115845  1.63136142
##     mu.L[6]     mu.L[7]     mu.L[8]     mu.L[9]    mu.L[10]      mu.rho
##  0.24071136 -1.17214759  1.03859492 -0.31095562  0.38038844 -1.14199583
##     sd.L[1]     sd.L[2]     sd.L[3]     sd.L[4]     sd.L[5]     sd.L[6]
##  0.62285366  0.63866368  0.44044018 -0.90861237  0.99506666 -1.70221223
##     sd.L[7]     sd.L[8]     sd.L[9]    sd.L[10]      sd.rho
##  1.09842682  0.01692457  0.57984555 -0.29841563  0.87676937
##
## 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 -3.38e+03 35.88545 2.07e-01       0.295598
## mu.L[1]   7.27e-01  0.01870 1.08e-04       0.000153
```

```
## mu.L[2]   6.51e-01  0.01633 9.43e-05       0.000147
## mu.L[3]   5.69e-01  0.01689 9.75e-05       0.000148
## mu.L[4]   6.58e-01  0.02272 1.31e-04       0.000172
## mu.L[5]   7.84e-01  0.01327 7.66e-05       0.000139
## mu.L[6]   7.45e-01  0.01302 7.52e-05       0.000131
## mu.L[7]   6.51e-01  0.01881 1.09e-04       0.000149
## mu.L[8]   7.35e-01  0.00846 4.89e-05       0.000140
## mu.L[9]   5.33e-01  0.03856 2.23e-04       0.000255
## mu.L[10]  7.45e-01  0.01557 8.99e-05       0.000141
## mu.rho    7.19e-01  0.02447 1.41e-04       0.000184
## ppp       6.91e-01  0.46219 2.67e-03       0.002768
## sd.L[1]   9.18e-02  0.01629 9.41e-05       0.000172
## sd.L[2]   7.65e-02  0.01307 7.55e-05       0.000132
## sd.L[3]   7.99e-02  0.01293 7.47e-05       0.000119
## sd.L[4]   1.17e-01  0.01809 1.04e-04       0.000160
## sd.L[5]   5.53e-02  0.01144 6.61e-05       0.000146
## sd.L[6]   5.49e-02  0.01122 6.48e-05       0.000144
## sd.L[7]   9.34e-02  0.01527 8.82e-05       0.000143
## sd.L[8]   2.42e-02  0.00971 5.61e-05       0.000261
## sd.L[9]   2.19e-01  0.03102 1.79e-04       0.000239
## sd.L[10]  7.12e-02  0.01345 7.77e-05       0.000154
## sd.rho    1.31e-01  0.02106 1.22e-04       0.000210
##
## 2. Quantiles for each variable:
##
##                 2.5%       25%       50%       75%     97.5%
## deviance -3.45e+03 -3.40e+03 -3.38e+03 -3.36e+03 -3.31e+03
## mu.L[1]   6.90e-01  7.15e-01  7.28e-01  7.40e-01  7.64e-01
## mu.L[2]   6.20e-01  6.40e-01  6.51e-01  6.62e-01  6.84e-01
## mu.L[3]   5.36e-01  5.57e-01  5.69e-01  5.80e-01  6.02e-01
## mu.L[4]   6.13e-01  6.43e-01  6.57e-01  6.73e-01  7.03e-01
## mu.L[5]   7.57e-01  7.75e-01  7.84e-01  7.92e-01  8.10e-01
## mu.L[6]   7.20e-01  7.37e-01  7.45e-01  7.54e-01  7.71e-01
## mu.L[7]   6.14e-01  6.38e-01  6.51e-01  6.63e-01  6.88e-01
## mu.L[8]   7.18e-01  7.29e-01  7.35e-01  7.41e-01  7.51e-01
## mu.L[9]   4.56e-01  5.07e-01  5.33e-01  5.58e-01  6.09e-01
## mu.L[10]  7.15e-01  7.35e-01  7.45e-01  7.56e-01  7.76e-01
## mu.rho    6.71e-01  7.03e-01  7.19e-01  7.36e-01  7.67e-01
## ppp       0.00e+00  0.00e+00  1.00e+00  1.00e+00  1.00e+00
## sd.L[1]   6.45e-02  8.02e-02  9.01e-02  1.02e-01  1.28e-01
## sd.L[2]   5.46e-02  6.73e-02  7.52e-02  8.43e-02  1.06e-01
## sd.L[3]   5.85e-02  7.07e-02  7.85e-02  8.77e-02  1.09e-01
## sd.L[4]   8.69e-02  1.05e-01  1.16e-01  1.28e-01  1.57e-01
## sd.L[5]   3.57e-02  4.73e-02  5.42e-02  6.21e-02  8.07e-02
## sd.L[6]   3.60e-02  4.69e-02  5.38e-02  6.17e-02  7.97e-02
## sd.L[7]   6.78e-02  8.25e-02  9.20e-02  1.02e-01  1.27e-01
## sd.L[8]   7.93e-03  1.74e-02  2.34e-02  3.00e-02  4.58e-02
## sd.L[9]   1.67e-01  1.97e-01  2.16e-01  2.38e-01  2.88e-01
## sd.L[10]  4.79e-02  6.18e-02  7.00e-02  7.94e-02  1.01e-01
## sd.rho    9.60e-02  1.17e-01  1.29e-01  1.44e-01  1.78e-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]
```

```
##    0.728    0.651    0.569    0.657    0.784    0.745    0.651    0.735
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.533    0.745    0.090    0.075    0.078    0.116    0.054    0.054
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]   mu.rho   sd.rho
##    0.092    0.023    0.216    0.070    0.719    0.129    0.691
##
## $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]
##    0.019    0.016    0.017    0.023    0.013    0.013    0.019    0.008
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.039    0.016    0.016    0.013    0.013    0.018    0.011    0.011
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]   mu.rho   sd.rho
##    0.015    0.010    0.031    0.013    0.024    0.021
##
## $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]
##    0.690    0.619    0.535    0.611    0.757    0.719    0.613    0.718
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.457    0.715    0.062    0.053    0.057    0.084    0.034    0.035
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]   mu.rho   sd.rho
##    0.066    0.007    0.163    0.046    0.671    0.093
##
## $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]
##    0.763    0.683    0.601    0.701    0.809    0.770    0.686    0.752
##  mu.L[9] mu.L[10]  sd.L[1]  sd.L[2]  sd.L[3]  sd.L[4]  sd.L[5]  sd.L[6]
##    0.610    0.776    0.125    0.103    0.107    0.153    0.078    0.077
##  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]   mu.rho   sd.rho
##    0.125    0.044    0.281    0.098    0.767    0.174
##
## $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]
##    0.613    0.555    0.469    0.509    0.715    0.676    0.533    0.706
##  mu.L[9] mu.L[10]
##    0.257    0.655
##
## $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]
##    0.843    0.747    0.669    0.805    0.853    0.814    0.769    0.764
##  mu.L[9] mu.L[10]
##    0.809    0.835
##
## $conv
##           deviance  mu.L[1]  mu.L[2]  mu.L[3]  mu.L[4]  mu.L[5]  mu.L[6]
##    2.000    0.655    0.951   -0.275    0.261    0.611    1.631    0.241
##  mu.L[7]  mu.L[8]  mu.L[9] mu.L[10]   mu.rho  sd.L[1]  sd.L[2]  sd.L[3]
##   -1.172    1.039   -0.311    0.380   -1.142    0.623    0.639    0.440
##  sd.L[4]  sd.L[5]  sd.L[6]  sd.L[7]  sd.L[8]  sd.L[9] sd.L[10]   sd.rho
##   -0.909    0.995   -1.702    1.098    0.017    0.580   -0.298    0.877
##
## $DIC
## [1] -3064
```

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 <- "POS =~ p1*I1 + p3*I3 + p4*I4 + p7*I7 + p10*I10
          NEG =~ n2*I2 + n5*I5 + n6*I6 + n8*I8 + n9*I9
          POS~~NEG"

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)

## 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 POS  0.7379976 0.011078455    66.61557        0
## 2          p3      A0  I3 POS  0.6058427 0.011417240    53.06385        0
## 3          p4      A0  I4 POS  0.5368963 0.010542498    50.92686        0
## 4          p7      A0  I7 POS  0.6345811 0.012536151    50.62009        0
## 5         p10      A0 I10 POS  0.7871185 0.010513362    74.86840        0
```

```
## 6          n2      A0  I2 NEG  0.7207413 0.011264928    63.98100         0
## 7          n5      A0  I5 NEG  0.6600429 0.010036719    65.76282         0
## 8          n6      A0  I6 NEG  0.6972316 0.010024660    69.55165         0
## 9          n8      A0  I8 NEG  0.5379421 0.013485946    39.88909         0
## 10         n9      A0  I9 NEG  0.7604895 0.009866174    77.08049         0
## 11 POSWITHNEG      S0 NEG POS  0.7446150 0.009986957    74.55874         0
## 12     Tau1_1 vecTau1   1   1 -4.7211056 0.253403879   -18.63076         0
## 13     Tau1_2 vecTau1   2   1 -4.9716802 0.261663648   -19.00027         0
## 14     Tau1_3 vecTau1   3   1 -5.0522066 0.258575995   -19.53858         0
## 15     Tau1_4 vecTau1   4   1 -4.4499369 0.251021427   -17.72732         0
## 16     Tau1_5 vecTau1   5   1 -5.3988416 0.267764591   -20.16264         0
## 17     Tau1_6 vecTau1   6   1 -4.3899799 0.251867082   -17.42975         0
## 18     Tau1_7 vecTau1   7   1 -3.7329335 0.243368749   -15.33859         0
## 19     Tau1_8 vecTau1   8   1 -4.9014580 0.253827160   -19.31022         0
## 20     Tau1_9 vecTau1   9   1 -4.2934450 0.253518198   -16.93545         0
## 21    Tau1_10 vecTau1  10   1 -5.0637752 0.261461945   -19.36716         0
## 22    Tau1_11 vecTau1  11   1 -5.3320672 0.261129245   -20.41926         0
## 23    Tau1_12 vecTau1  12   1 -5.1798836 0.263896242   -19.62849         0
## 24    Tau1_13 vecTau1  13   1 -3.5015316 0.265583267   -13.18431         0
## 25    Tau1_14 vecTau1  14   1 -4.7076158 0.247919426   -18.98849         0
## 26    Tau1_15 vecTau1  15   1 -4.0914071 0.252151450   -16.22599         0
## 27    Tau1_16 vecTau1  16   1 -5.1224116 0.266327392   -19.23351         0
## 28    Tau1_17 vecTau1  17   1 -5.0145319 0.250148240   -20.04624         0
## 29    Tau1_18 vecTau1  18   1 -3.6708123 0.264422660   -13.88237         0
## 30    Tau1_19 vecTau1  19   1 -4.5839111 0.260402532   -17.60317         0
## 31    Tau1_20 vecTau1  20   1 -5.3359474 0.277915930   -19.19986         0
## 32    Tau1_21 vecTau1  21   1 -3.1839408 0.252928679   -12.58830         0
## 33    Tau1_22 vecTau1  22   1 -4.2213271 0.255795900   -16.50272         0
## 34    Tau1_23 vecTau1  23   1 -5.2424964 0.260774307   -20.10358         0
## 35    Tau1_24 vecTau1  24   1 -5.1403856 0.268895992   -19.11663         0
## 36    Tau1_25 vecTau1  25   1 -5.0646797 0.267256509   -18.95063         0
## 37    Tau1_26 vecTau1  26   1 -5.4679785 0.266478053   -20.51943         0
## 38    Tau1_27 vecTau1  27   1 -3.9921905 0.260485088   -15.32598         0
## 39    Tau1_28 vecTau1  28   1 -4.1755151 0.252420768   -16.54188         0
## 40    Tau1_29 vecTau1  29   1 -5.0804458 0.262397821   -19.36162         0
## 41    Tau1_30 vecTau1  30   1 -5.4085655 0.275765394   -19.61292         0
## 42    Tau1_31 vecTau1  31   1 -5.4200061 0.272465531   -19.89245         0
## 43    Tau1_32 vecTau1  32   1 -4.6704905 0.253089726   -18.45389         0
## 44    Tau1_33 vecTau1  33   1 -4.7121002 0.257539037   -18.29664         0
## 45    Tau1_34 vecTau1  34   1 -4.3768061 0.248984509   -17.57863         0
## 46    Tau1_35 vecTau1  35   1 -4.7979216 0.254827770   -18.82810         0
## 47    Tau1_36 vecTau1  36   1 -5.0970432 0.258645108   -19.70671         0
## 48    Tau1_37 vecTau1  37   1 -3.9983037 0.253789000   -15.75444         0
## 49    Tau1_38 vecTau1  38   1 -5.3963456 0.262192744   -20.58160         0
## 50    Tau1_39 vecTau1  39   1 -5.4825283 0.259706568   -21.11047         0
## 51    Tau1_40 vecTau1  40   1 -4.0717447 0.248314013   -16.39756         0
## 52    Tau1_41 vecTau1  41   1 -4.8051095 0.248380578   -19.34575         0
## 53    Tau1_42 vecTau1  42   1 -4.7844965 0.252082777   -18.97986         0
## 54    Tau1_43 vecTau1  43   1 -4.1003734 0.247290665   -16.58119         0
## 55    Tau1_44 vecTau1  44   1 -3.5236223 0.242888977   -14.50713         0
## 56    Tau1_45 vecTau1  45   1 -4.8169738 0.256623276   -18.77060         0
##
## To obtain confidence intervals re-run with intervals=TRUE
##
```

```
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             56                   1564          -2561.185
##    Saturated:             90                   1530          -2777.449
## Independence:             45                   1575           1549.587
## Number of observations/statistics: 109988/1620
##
## chi-square:   ² ( df=34 ) = 216.2637,  p = 2.139516e-28
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -5689.185             -2449.185               -2449.127
## BIC:     -20716.295             -1911.130               -2089.100
## CFI: 0.9574353
## TLI: 0.9436643   (also known as NNFI)
## RMSEA:  0.006981327  [95% CI (0.005936391, 0.008054096)]
## Prob(RMSEA <= 0.05): 1
## timestamp: 2023-12-12 18:27:19
## Wall clock time: 72.79894 secs
## optimizer:  SLSQP
## OpenMx version number: 2.21.8
## Need help?  See help(mxSummary)
```

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

```
## [1] 0.04508931
```