MED_NoCovariate

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".
```

```r
library(xlsx)

# Working directory
wd = 'D:/Research/2023/CompareMASEM/MED/'
setwd(paste0(wd,'NoCovariate/'))

# Read in data
dat = read.xlsx(paste0(wd,'data3.xlsx'),1)
head(dat)
```

```
##      AuthorYear                          doi study   N        rXM   rMY
## 1      Wong2018   10.1038/s41598-018-24945-4     1 139         NA    NA
## 2 Vollestad2011   10.1016/j.brat.2011.01.007     2  65  0.4500000 -0.26
## 3    VanSon2013             10.2337/dc12-1477    3 139         NA    NA
## 4    VanSon2013             10.2337/dc12-1477    3 139         NA    NA
## 5    Sevinc2018 10.1097/psy.0000000000000590     4  37 -0.1578195    NA
## 6      Song2015   10.1016/j.nedt.2014.06.010     5  44  0.3202971    NA
##          rXY   AgeM    AgeSD  T1DeprR   T1DeprM  T1DeprSD DeprMeasure
## 1 -0.1823328 52.000  3.09000 2.505803 0.4516041 0.1802233       GCS-D
## 2 -0.5000000 42.500 11.30000 1.965117 0.2682540 0.1365079      BDI-II
## 3 -0.2829384 56.500 13.00000 2.188851 0.3998287 0.1826660      HADS-D
## 4 -0.3345372 56.500 13.00000 4.301732 0.8107914 0.1884802     POMS-D8
## 5         NA 38.292 10.21452       NA        NA        NA        <NA>
## 6 -0.4470000 19.600  1.85000 1.165779 0.2013528 0.1727195      DASS-D
##   FemaleProp Mreliability YReliability AssessTime.day. Quality Noutcome
## 1       1.00         0.93           NA             224      12        3
## 2       0.67         0.90         0.88              56       8        5
## 3       0.50           NA         0.81              56       6        5
## 4       0.50           NA         0.85              56       6        5
## 5       0.64           NA           NA              70       9        1
## 6       0.81         0.93         0.81              70       8        3
```

```r
wd = paste0(wd,'NoCovariate/')
```

2 Functions

```r
# 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){
```

```r
    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[vi
         +(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.',names(est))
    sel.sdL = grep('sd.',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

```
# remove multiple correlations from the same study
sid = dat[,'study']
sel.id = (duplicated(sid)==0)
dat = dat[sel.id,]

vR  = as.matrix(dat[,c('rXM','rXY','rMY')]) # bivariate correlations
N   = dat[,'N'] # individual study sample sizes
Nstudy  = nrow(dat) # number of studies
mu.N    = mean(N) # mean sample size per study

# Coordinations (matrix <-> vector)
p   = 3 # number of observed variables
pp  = p*(p-1)/2 # number of bivariate correlations
index.list = jkvil(p)

# Compute level-1 error covariance matrix
# Or covariance matrix of observed correlation vectors
vR.bar = apply(vR,2,mean,na.rm = TRUE)
vR.impute = Mimpute(vR,N,'MCAR')
Stau.vR <- Vj(vR.bar,N,pp,Nstudy,index.list)
tau.vR <- Stau.vR$tau.vR;

# Hyperparameters for priors (additional error term)
I3 = diag(1,3); u0 = rep(0,3);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)
```

3.2 Model fitting

```
data<-list("Nstudy","N","mu.N",'p',"pp","vR","tau.vR","mu.vR.psi",
    'alpha.prior.vE','beta.prior.vE','u0','I3') #data

vR.inits = vR.impute; vR.inits[which(is.na(vR)==0,arr.ind = TRUE)] = NA
initsl <- list(list(mu.a=0,mu.b=0,mu.cp=0,tau.u=diag(100,3),xi=rep(1,3),tau.R = 100,
    vR.psi = matrix(0,Nstudy,pp),vR = vR.inits)) # initial values

prm = c('mu.a','mu.b','mu.cp','mu.ab','sd.a','sd.b','sd.cp','sd.ab',
```

```r
    'rho.ab','rho.acp','rho.bcp') # Parameters to save;
model.fn = 'Mediation_Random.txt' # Model file name

# stop every 30000 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.a ... 30000 valid values
## Abstracting mu.ab ... 30000 valid values
## Abstracting mu.b ... 30000 valid values
## Abstracting mu.cp ... 30000 valid values
## Abstracting rho.ab ... 30000 valid values
## Abstracting rho.acp ... 30000 valid values
## Abstracting rho.bcp ... 30000 valid values
## Abstracting sd.a ... 30000 valid values
## Abstracting sd.ab ... 30000 valid values
## Abstracting sd.b ... 30000 valid values
## Abstracting sd.cp ... 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 -1.64e+02 31.7746 0.183450        0.89871
## mu.a      3.09e-01  0.0341 0.000197        0.00192
## mu.ab    -8.25e-02  0.0273 0.000158        0.00112
## mu.b     -2.68e-01  0.0844 0.000487        0.00358
## mu.cp    -1.85e-01  0.0419 0.000242        0.00162
## rho.ab    3.86e-03  0.3760 0.002171        0.02037
## rho.acp  -1.25e-01  0.3463 0.001999        0.01651
## rho.bcp  -1.59e-01  0.3667 0.002117        0.01982
## sd.a      1.06e-01  0.0373 0.000215        0.00154
## sd.ab     5.38e-02  0.0278 0.000161        0.00111
## sd.b      1.30e-01  0.0879 0.000507        0.00354
## sd.cp     1.08e-01  0.0453 0.000262        0.00173
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance -224.6000 -186.0000 -1.64e+02 -141.1750 -101.9975
## mu.a        0.2405    0.2862  3.07e-01    0.3306    0.3778
## mu.ab      -0.1391   -0.0978 -8.15e-02   -0.0657   -0.0323
## mu.b       -0.4366   -0.3148 -2.68e-01   -0.2192   -0.1089
## mu.cp      -0.2699   -0.2114 -1.85e-01   -0.1580   -0.1017
## rho.ab     -0.6653   -0.2800 -8.75e-03    0.2767    0.7368
## rho.acp    -0.7490   -0.3794 -1.39e-01    0.1202    0.5687
## rho.bcp    -0.7654   -0.4439 -1.91e-01    0.0973    0.6007
```

```
## sd.a        0.0375    0.0810  1.04e-01    0.1289    0.1856
## sd.ab       0.0175    0.0345  4.81e-02    0.0661    0.1250
## sd.b        0.0194    0.0685  1.12e-01    0.1701    0.3479
## sd.cp       0.0265    0.0764  1.06e-01    0.1369    0.2047
##
##    deviance        mu.a       mu.ab        mu.b       mu.cp      rho.ab
## -0.87470411  0.34816988 -0.10145990  0.44511832  1.42922135 -1.30740370
##     rho.acp     rho.bcp        sd.a       sd.ab        sd.b       sd.cp
##  1.23337168 -0.68372093  1.91384011  0.86445613 -0.21321059  0.03508092
##
## 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 -1.64e+02 31.7746 0.183450        0.89871
## mu.a      3.09e-01  0.0341 0.000197        0.00192
## mu.ab    -8.25e-02  0.0273 0.000158        0.00112
## mu.b     -2.68e-01  0.0844 0.000487        0.00358
## mu.cp    -1.85e-01  0.0419 0.000242        0.00162
## rho.ab    3.86e-03  0.3760 0.002171        0.02037
## rho.acp  -1.25e-01  0.3463 0.001999        0.01651
## rho.bcp  -1.59e-01  0.3667 0.002117        0.01982
## sd.a      1.06e-01  0.0373 0.000215        0.00154
## sd.ab     5.38e-02  0.0278 0.000161        0.00111
## sd.b      1.30e-01  0.0879 0.000507        0.00354
## sd.cp     1.08e-01  0.0453 0.000262        0.00173
##
## 2. Quantiles for each variable:
##
##                2.5%       25%       50%       75%     97.5%
## deviance -224.6000 -186.0000 -1.64e+02 -141.1750 -101.9975
## mu.a        0.2405    0.2862  3.07e-01    0.3306    0.3778
## mu.ab      -0.1391   -0.0978 -8.15e-02   -0.0657   -0.0323
## mu.b       -0.4366   -0.3148 -2.68e-01   -0.2192   -0.1089
## mu.cp      -0.2699   -0.2114 -1.85e-01   -0.1580   -0.1017
## rho.ab     -0.6653   -0.2800 -8.75e-03    0.2767    0.7368
## rho.acp    -0.7490   -0.3794 -1.39e-01    0.1202    0.5687
## rho.bcp    -0.7654   -0.4439 -1.91e-01    0.0973    0.6007
## sd.a        0.0375    0.0810  1.04e-01    0.1289    0.1856
## sd.ab       0.0175    0.0345  4.81e-02    0.0661    0.1250
## sd.b        0.0194    0.0685  1.12e-01    0.1701    0.3479
## sd.cp       0.0265    0.0764  1.06e-01    0.1369    0.2047
```

```
fit[-9]
```

```
## $est
##    mu.a   mu.ab    mu.b   mu.cp    sd.a   sd.ab    sd.b   sd.cp  rho.ab rho.acp
##   0.308  -0.082  -0.268  -0.185   0.104   0.048   0.112   0.106  -0.009  -0.139
## rho.bcp
##  -0.191
```

```
##
## $psd
##    mu.a   mu.ab    mu.b   mu.cp    sd.a   sd.ab    sd.b   sd.cp  rho.ab rho.acp
##   0.034   0.027   0.084   0.042   0.037   0.028   0.088   0.045   0.376   0.346
## rho.bcp
##   0.367
##
## $CIl
##    mu.a   mu.ab    mu.b   mu.cp    sd.a   sd.ab    sd.b   sd.cp  rho.ab rho.acp
##   0.238  -0.138  -0.431  -0.268   0.034   0.012   0.007   0.019  -0.666  -0.781
## rho.bcp
##  -0.818
##
## $CIu
##    mu.a   mu.ab    mu.b   mu.cp    sd.a   sd.ab    sd.b   sd.cp  rho.ab rho.acp
##   0.374  -0.031  -0.105  -0.100   0.181   0.108   0.297   0.194   0.736   0.528
## rho.bcp
##   0.525
##
## $CVl
##   mu.a  mu.ab   mu.b  mu.cp
##  0.175 -0.143 -0.411 -0.321
##
## $CVu
##   mu.a  mu.ab   mu.b  mu.cp
##  0.441 -0.021 -0.125 -0.049
##
## $conv
##          deviance     mu.a    mu.ab     mu.b    mu.cp   rho.ab  rho.acp
##    2.000   -0.875    0.348   -0.101    0.445    1.429   -1.307    1.233
##  rho.bcp     sd.a    sd.ab     sd.b    sd.cp
##   -0.684    1.914    0.864   -0.213    0.035
##
## $DIC
## [1] -104.2
```

4 OSMASEM

4.1 Data preparation

```
MFd = vector('list',Nstudy)
Mat = matrix(0,3,3)
for(studyi in 1:Nstudy){
    Mat[lower.tri(Mat)] = vR[studyi,]
    Mat[upper.tri(Mat)] = vR[studyi,]
    diag(Mat) = 1
    MFd[[studyi]] = Mat
}

## Create a dataframe with the data and the asymptotic variances and covariances (acov)
my.df <- Cor2DataFrame(MFd, N, acov = "weighted")
## Moderator Female proportion (standardized)
my.df$data <- data.frame(my.df$data,check.names=FALSE)
```

```r
summary(my.df)
```

```
##           Length Class      Mode     
## data       9     data.frame list     
## n         38     -none-     numeric  
## obslabels  0     -none-     NULL     
## ylabels    3     -none-     character
## vlabels    6     -none-     character
```

4.2 Model fitting

```r
## Specify the mediation model
model0 <- "Y ~ M + X; M ~ X; X ~~ 1*X"
RAM0 <- lavaan2RAM(model0, obs.variables = c("X","M","Y"))

## 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 
##  
## The Hessian at the solution does not appear to be convex. See ?mxCheckIdentification for possible dia
##  
## free parameters:
##     name  matrix row col    Estimate  Std.Error A   z value     Pr(>|z|)
## 1   MONX      A0   M   X   0.3110244 0.02853032   10.901537 0.000000e+00
## 2   YONX      A0   Y   X  -0.1902570 0.03582811   -5.310273 1.094614e-07
## 3   YONM      A0   Y   M  -0.2576921 0.04461689   -5.775662 7.665103e-09
## 4 Tau1_1 vecTau1   1   1  -4.7146108 0.59486306   -7.925540 2.220446e-15
## 5 Tau1_2 vecTau1   2   1  -4.1931962 0.52442636   -7.995777 1.332268e-15
## 6 Tau1_3 vecTau1   3   1 -31.8035579         NA !        NA           NA
## 
## To obtain confidence intervals re-run with intervals=TRUE
## 
## Model Statistics: 
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:              6                     57              -57.88404
##    Saturated:              6                     57              -57.88404
## Independence:              3                     60               41.89937
## Number of observations/statistics: 2759/63
## 
## 
## ** Information matrix is not positive definite (not at a candidate optimum).
##   Be suspicious of these results. At minimum, do not trust the standard errors.
## 
## chi-square:  ² ( df=0 ) = -5.388046e-11,  p = 1
## Information Criteria: 
##      |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
```

```
## AIC:      -171.8840              -45.88404                 -45.85352
## BIC:      -509.4736              -10.34830                 -29.41227
## CFI: 1
## TLI: 1   (also known as NNFI)
## RMSEA:  0  [95% CI (NA, NA)]
## Prob(RMSEA <= 0.05): NA
## timestamp: 2023-12-13 17:30:26
## Wall clock time: 0.1274171 secs
## optimizer:  SLSQP
## OpenMx version number: 2.21.8
## Need help?  See help(mxSummary)
```

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

```
## [1] 7.799598e-10
```

4.3 Mean indirect effect and its SE

```
# Delta method
parsnv = c('MONX','YONM')
Sigma = vcov(fit0)[parsnv,parsnv]
est = coef(fit0)[parsnv]

ab.est = prod(est)
ab.se = sqrt(t(est[c(2,1)]) %*% Sigma %*% est[c(2,1)] )

c(ab.est,ab.se)
```

```
## [1] -0.08014852  0.01538005
```