MED_WithCovariate

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,'WithCovariate/'))

# 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,'WithCovariate/')
```

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){
    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.',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,]

# remove studies with missing baseline depression
na.id   = which(is.na(dat[,"T1DeprR"])==1)
dat     = dat[-na.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

M          = round(dat[,"T1DeprR"],2) # moderator: baseline depression severity
M     = as.numeric(scale(M))  # standardization
#predM  = as.numeric(summary(M)[c(2,3,5)]) # Low,moderator, and high levels of baseline depression
predM = c(min(M),median(M),max(M))

# 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)
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","pp","vR","tau.vR",'M','predM',
    "mu.vR.psi",'alpha.prior.vE','beta.prior.vE') # data

vR.inits = vR.impute; vR.inits[which(is.na(vR)==0,arr.ind = TRUE)] = NA
initsl <- list(list(b0.a=0,b0.b=0,b0.cp=0,b1.a=0,b1.cp=0,
    sd.ua=0.1,sd.ucp=0.1,tau.R=100,
    vR.psi = matrix(0,Nstudy,pp),vR=vR.inits))# initial values

prm = c(paste0('b0.',c('a','b','cp')),paste0('b1.',c('a','cp')),
    paste0('sd.u',c('a','cp')),'cphat')# Parameters to save

model.fn = paste(wd,'Mediation_Covariate.txt',sep='')# 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 b0.a ... 30000 valid values
## Abstracting b0.b ... 30000 valid values
## Abstracting b0.cp ... 30000 valid values
## Abstracting b1.a ... 30000 valid values
## Abstracting b1.cp ... 30000 valid values
## Abstracting cphat[1] ... 30000 valid values
## Abstracting cphat[2] ... 30000 valid values
## Abstracting cphat[3] ... 30000 valid values
## Abstracting deviance ... 30000 valid values
## Abstracting sd.ua ... 30000 valid values
## Abstracting sd.ucp ... 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
## b0.a        0.2950  0.0457 0.000264        0.00200
## b0.b       -0.2624  0.0519 0.000300        0.00244
## b0.cp      -0.1928  0.0365 0.000211        0.00119
## b1.a       -0.0896  0.0900 0.000520        0.00524
## b1.cp      -0.0800  0.0400 0.000231        0.00152
## cphat[1]   -0.1216  0.0509 0.000294        0.00164
## cphat[2]   -0.1720  0.0379 0.000219        0.00120
## cphat[3]   -0.5511  0.1831 0.001057        0.00700
## deviance -156.6767 28.9413 0.167092        0.66115
## sd.ua       0.0923  0.0374 0.000216        0.00151
## sd.ucp      0.1045  0.0395 0.000228        0.00143
##
## 2. Quantiles for each variable:
```

```
##
##              2.5%      25%      50%      75%     97.5%
## b0.a       0.2089   0.2647   0.2935   0.3243  3.89e-01
## b0.b      -0.3628  -0.2982  -0.2633  -0.2280 -1.58e-01
## b0.cp     -0.2657  -0.2167  -0.1926  -0.1684 -1.22e-01
## b1.a      -0.2641  -0.1501  -0.0926  -0.0324  9.40e-02
## b1.cp     -0.1598  -0.1062  -0.0798  -0.0533 -2.46e-03
## cphat[1]  -0.2216  -0.1554  -0.1220  -0.0878 -2.10e-02
## cphat[2]  -0.2470  -0.1970  -0.1718  -0.1468 -9.90e-02
## cphat[3]  -0.9131  -0.6708  -0.5503  -0.4293 -1.95e-01
## deviance -212.7025 -176.5000 -156.9000 -136.5000 -1.01e+02
## sd.ua      0.0264   0.0659   0.0903   0.1155  1.73e-01
## sd.ucp     0.0306   0.0768   0.1032   0.1301  1.87e-01
##
## [1] "Iteration: 90000"
## Abstracting b0.a ... 30000 valid values
## Abstracting b0.b ... 30000 valid values
## Abstracting b0.cp ... 30000 valid values
## Abstracting b1.a ... 30000 valid values
## Abstracting b1.cp ... 30000 valid values
## Abstracting cphat[1] ... 30000 valid values
## Abstracting cphat[2] ... 30000 valid values
## Abstracting cphat[3] ... 30000 valid values
## Abstracting deviance ... 30000 valid values
## Abstracting sd.ua ... 30000 valid values
## Abstracting sd.ucp ... 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
## b0.a       0.2950  0.0444 0.000257        0.00189
## b0.b      -0.2616  0.0519 0.000299        0.00234
## b0.cp     -0.1924  0.0362 0.000209        0.00117
## b1.a      -0.0861  0.0868 0.000501        0.00449
## b1.cp     -0.0807  0.0391 0.000226        0.00126
## cphat[1]  -0.1207  0.0496 0.000287        0.00170
## cphat[2]  -0.1714  0.0374 0.000216        0.00123
## cphat[3]  -0.5536  0.1795 0.001036        0.00546
## deviance -156.0929 29.0767 0.167874        0.63229
## sd.ua      0.0931  0.0375 0.000216        0.00145
## sd.ucp     0.1047  0.0394 0.000228        0.00153
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%     97.5%
## b0.a       0.2099   0.2652   0.2938   0.3239   0.38480
## b0.b      -0.3659  -0.2958  -0.2606  -0.2262  -0.15990
## b0.cp     -0.2644  -0.2160  -0.1924  -0.1682  -0.12200
## b1.a       -0.2495   -0.1461   -0.0880   -0.0294   0.08983
## b1.cp      -0.1588   -0.1062   -0.0800   -0.0545  -0.00559
## cphat[1]   -0.2176   -0.1533   -0.1212   -0.0886  -0.02070
## cphat[2]   -0.2462   -0.1957   -0.1711   -0.1469  -0.09747
## cphat[3]   -0.9097   -0.6708   -0.5517   -0.4340  -0.20550
## deviance -211.8000 -176.5000 -156.3000 -135.9000 -99.38925
## sd.ua       0.0267    0.0672    0.0905    0.1159   0.17530
## sd.ucp      0.0327    0.0770    0.1030    0.1301   0.18700
##
## [1] "Iteration: 120000"
## Abstracting b0.a ... 30000 valid values
## Abstracting b0.b ... 30000 valid values
## Abstracting b0.cp ... 30000 valid values
## Abstracting b1.a ... 30000 valid values
## Abstracting b1.cp ... 30000 valid values
## Abstracting cphat[1] ... 30000 valid values
## Abstracting cphat[2] ... 30000 valid values
## Abstracting cphat[3] ... 30000 valid values
## Abstracting deviance ... 30000 valid values
## Abstracting sd.ua ... 30000 valid values
## Abstracting sd.ucp ... 30000 valid values
##
## Iterations = 90003:120002
## 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
## b0.a        0.2984  0.0452 0.000261        0.00196
## b0.b       -0.2630  0.0534 0.000308        0.00251
## b0.cp      -0.1908  0.0364 0.000210        0.00128
## b1.a       -0.0786  0.0880 0.000508        0.00480
## b1.cp      -0.0774  0.0395 0.000228        0.00135
## cphat[1]   -0.1220  0.0500 0.000289        0.00168
## cphat[2]   -0.1707  0.0376 0.000217        0.00131
## cphat[3]   -0.5374  0.1815 0.001048        0.00623
## deviance -155.6326 28.9328 0.167044        0.66915
## sd.ua       0.0943  0.0392 0.000226        0.00159
## sd.ucp      0.1003  0.0393 0.000227        0.00145
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## b0.a        0.2087    0.2691    0.2982    0.3269   0.38950
## b0.b       -0.3674   -0.2976   -0.2632   -0.2283  -0.15780
## b0.cp      -0.2640   -0.2147   -0.1905   -0.1669  -0.11900
## b1.a       -0.2469   -0.1364   -0.0808   -0.0217   0.09935
## b1.cp      -0.1580   -0.1036   -0.0766   -0.0506  -0.00154
## cphat[1]   -0.2210   -0.1548   -0.1219   -0.0889  -0.02361
## cphat[2]   -0.2465   -0.1950   -0.1702   -0.1458  -0.09637
## cphat[3]   -0.9012   -0.6573   -0.5343   -0.4148  -0.18630
## deviance -211.7025 -175.9000 -155.7000 -135.4000 -99.62975
## sd.ua       0.0247    0.0666    0.0924    0.1185   0.17900
## sd.ucp      0.0266    0.0731    0.0986    0.1260   0.18140
##
## [1] "Iteration: 150000"
## Abstracting b0.a ... 30000 valid values
## Abstracting b0.b ... 30000 valid values
## Abstracting b0.cp ... 30000 valid values
## Abstracting b1.a ... 30000 valid values
## Abstracting b1.cp ... 30000 valid values
## Abstracting cphat[1] ... 30000 valid values
## Abstracting cphat[2] ... 30000 valid values
## Abstracting cphat[3] ... 30000 valid values
## Abstracting deviance ... 30000 valid values
## Abstracting sd.ua ... 30000 valid values
## Abstracting sd.ucp ... 30000 valid values
##
## Iterations = 120004:150003
## 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
## b0.a        0.2942  0.0454 0.000262        0.00190
## b0.b       -0.2616  0.0515 0.000297        0.00229
## b0.cp      -0.1939  0.0365 0.000210        0.00120
## b1.a       -0.0921  0.0872 0.000504        0.00483
## b1.cp      -0.0816  0.0400 0.000231        0.00128
## cphat[1]   -0.1214  0.0510 0.000294        0.00159
## cphat[2]   -0.1727  0.0379 0.000219        0.00123
## cphat[3]   -0.5590  0.1827 0.001055        0.00590
## deviance -156.3441 28.9386 0.167077        0.63101
## sd.ua       0.0926  0.0386 0.000223        0.00162
## sd.ucp      0.1072  0.0385 0.000222        0.00129
##
## 2. Quantiles for each variable:
##
##              2.5%       25%       50%       75%     97.5%
## b0.a       0.2085    0.2639    0.2926    0.3223   0.38940
## b0.b      -0.3609   -0.2958   -0.2625   -0.2266  -0.15880
## b0.cp     -0.2672   -0.2179   -0.1932   -0.1693  -0.12380
## b1.a      -0.2566   -0.1505   -0.0958   -0.0380   0.09260
## b1.cp     -0.1598   -0.1079   -0.0820   -0.0561  -0.00127
## cphat[1]  -0.2228   -0.1550   -0.1214   -0.0872  -0.02123
## cphat[2]  -0.2496   -0.1972   -0.1719   -0.1472  -0.10020
## cphat[3]  -0.9109   -0.6793   -0.5616   -0.4447  -0.18529
## deviance -212.6000 -176.3000 -156.8000 -136.5000 -99.30950
## sd.ua       0.0200    0.0666    0.0905    0.1158   0.17650
## sd.ucp      0.0366    0.0804    0.1053    0.1318   0.18860
##
##        b0.a        b0.b       b0.cp        b1.a       b1.cp    cphat[1]
## -0.85230254  0.06357431 -0.64265717 -1.01366965 -0.87942357  0.11720276
##    cphat[2]    cphat[3]    deviance       sd.ua      sd.ucp
## -0.36100201 -1.00280127  0.35611010 -0.43161669  0.70984177
##
## Iterations = 120004:150003
## 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
## b0.a        0.2942  0.0454 0.000262        0.00190
## b0.b       -0.2616  0.0515 0.000297        0.00229
## b0.cp      -0.1939  0.0365 0.000210        0.00120
## b1.a       -0.0921  0.0872 0.000504        0.00483
## b1.cp      -0.0816  0.0400 0.000231        0.00128
## cphat[1]   -0.1214  0.0510 0.000294        0.00159
## cphat[2]   -0.1727  0.0379 0.000219        0.00123
## cphat[3]   -0.5590  0.1827 0.001055        0.00590
## deviance -156.3441 28.9386 0.167077        0.63101
## sd.ua       0.0926  0.0386 0.000223        0.00162
## sd.ucp      0.1072  0.0385 0.000222        0.00129
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## b0.a        0.2085    0.2639    0.2926    0.3223   0.38940
## b0.b       -0.3609   -0.2958   -0.2625   -0.2266  -0.15880
## b0.cp      -0.2672   -0.2179   -0.1932   -0.1693  -0.12380
## b1.a       -0.2566   -0.1505   -0.0958   -0.0380   0.09260
## b1.cp      -0.1598   -0.1079   -0.0820   -0.0561  -0.00127
## cphat[1]   -0.2228   -0.1550   -0.1214   -0.0872  -0.02123
## cphat[2]   -0.2496   -0.1972   -0.1719   -0.1472  -0.10020
## cphat[3]   -0.9109   -0.6793   -0.5616   -0.4447  -0.18529
## deviance -212.6000 -176.3000 -156.8000 -136.5000 -99.30950
## sd.ua       0.0200    0.0666    0.0905    0.1158   0.17650
## sd.ucp      0.0366    0.0804    0.1053    0.1318   0.18860
```

```
fit[-9]
```

```
## $est
##     b0.a     b0.b    b0.cp     b1.a    b1.cp    sd.ua   sd.ucp cphat[1]
##    0.293   -0.262   -0.193   -0.096   -0.082    0.091    0.105   -0.121
## cphat[2] cphat[3]
##   -0.172   -0.562
##
## $psd
##     b0.a     b0.b    b0.cp     b1.a    b1.cp    sd.ua   sd.ucp cphat[1]
##    0.045    0.051    0.036    0.087    0.040    0.039    0.039    0.051
## cphat[2] cphat[3]
##    0.038    0.183
##
## $CIl
##     b0.a     b0.b    b0.cp     b1.a    b1.cp    sd.ua   sd.ucp cphat[1]
##    0.206   -0.360   -0.270   -0.262   -0.159    0.016    0.033   -0.225
## cphat[2] cphat[3]
##   -0.250   -0.913
##
## $CIu
##     b0.a     b0.b    b0.cp     b1.a    b1.cp    sd.ua   sd.ucp cphat[1]
##    0.387   -0.158   -0.126    0.085   -0.001    0.168    0.184   -0.023
## cphat[2] cphat[3]
##   -0.101   -0.189
##
## $CVl
## named numeric(0)
##
## $CVu
## named numeric(0)
##
## $conv
##              b0.a     b0.b    b0.cp     b1.a    b1.cp cphat[1] cphat[2]
##    5.000   -0.852    0.064   -0.643   -1.014   -0.879    0.117   -0.361
## cphat[3] deviance    sd.ua   sd.ucp
##   -1.003    0.356   -0.432    0.710
##
## $DIC
## [1] -102.8
```

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,covariate=M,check.names=FALSE)
summary(my.df)
```

```
##           Length Class      Mode
## data      10     data.frame list
## n         33     -none-     numeric
## obslabels  0     -none-     NULL
## ylabels    3     -none-     character
## vlabels    6     -none-     character
```

4.2 Model fitting

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

## Create heterogeneity variances
TOTF = diag(TRUE,3)
TOTF[3,3] = FALSE
T0 <- create.Tau2(RAM=RAM0, RE.type="User", RE.User = TOTF, Transform="expLog", RE.startvalues=0.05)

## Mediation model  with `covariate` as a moderator on the A matrix
Ax1 <- RAM0$A
Ax1[grep("\\*", Ax1)] <- "0*data.covariate"
Ax1[3,2] <- "0"
Ax1
```

```
##   X                  M   Y
## X "0"                "0" "0"
## M "0*data.covariate" "0" "0"
## Y "0*data.covariate" "0" "0"
```

```
## Create matrices with implicit diagonal constraints
M1 <- create.vechsR(A0=RAM0$A, S0=RAM0$S, F0=RAM0$F, Ax=Ax1)

## Fit the bifactor model with One-Stage MASEM
fit1 <- osmasem(model.name="Moderation by covariate",
                Mmatrix=M1, Tmatrix=T0, data=my.df)
summary(fit1, fitIndices= T)
```

```
## Summary of Moderation by covariate
##
## free parameters:
##     name  matrix row col    Estimate  Std.Error A   z value     Pr(>|z|)
## 1   M0NX      A0   M   X  0.29005412 0.03547583    8.176105 2.220446e-16
## 2   Y0NX      A0   Y   X -0.19266487 0.03381242   -5.698051 1.211849e-08
## 3   Y0NM      A0   Y   M -0.26712468 0.04462233   -5.986346 2.146077e-09
## 4 M0NX_1      A1   M   X -0.10406064 0.06929202   -1.501769 1.331567e-01
## 5 Y0NX_1      A1   Y   X -0.08556672 0.03525429   -2.427129 1.521884e-02
## 6 Tau1_1 vecTau1   1   1 -5.35622646 0.85004274   -6.301126 2.954910e-10
## 7 Tau1_2 vecTau1   2   1 -4.35392687 0.54935905   -7.925467 2.220446e-15
##
## To obtain confidence intervals re-run with intervals=TRUE
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             7                     51             -64.32630
##    Saturated:             5                     53             -58.37605
## Independence:             2                     56              80.31656
## Number of observations/statistics: 2420/58
##
## chi-square:   ² ( df=-2 ) = -5.950254,  p = NaN
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -166.3263              -50.32630                -50.27987
## BIC:      -461.6940               -9.78564                -32.02623
## CFI: 1.029112
## TLI: 1   (also known as NNFI)
## RMSEA:  0  [95% CI (NA, NA)]
## Prob(RMSEA <= 0.05): NA
## timestamp: 2023-12-14 20:05:52
## Wall clock time: 0.244334 secs
## optimizer:  SLSQP
## OpenMx version number: 2.21.8
## Need help?  See help(mxSummary)
```

4.3 Prediction

```
# Predicted values
parsnv = c('YONX','YONX_1')
est      = coef(fit1)[parsnv]
pred     = cbind(rep(1),predM)%*%est

# Calculate prediction SE using Delta method
Sigma    = vcov(fit1)[parsnv,parsnv]
se.pred = rep(NA,3)
for(i in 1:3){
    se.pred[i] = sqrt(t(c(1,predM[i]))%*%Sigma%*%c(1,predM[i]))
}
cbind(pred,se.pred)
```

```
##                     se.pred
## [1,] -0.1165951 0.04705699
## [2,] -0.1704049 0.03540305
## [3,] -0.5757063 0.16000705
```