Reanalysis of Lv and Maeda (2019) with W-COV GLS with PD and TSSEM

Suzanne Jak and Mike W.-L. Cheung

25 october 2019

Simulate data

We generate data for 10 studies (k = 10), with an average sample size of 200 (n = 200), including two studies with complete data (pc = .20), and 8 studies that missed two variables (pm = .17), and equal factor loadings. According to Table 5 in Lv and Maeda (2019), this specification should lead to biased parameter estimates and biased standard errors for all methods.

```
set.seed(22122018)

library(metaSEM)
library(MASS)
source("Generatedata.R")
source("GLSapproach.R")

###### population model

lambda <- matrix(
  c( .7,0,0,
     .7,0,0,
     .7,0,0,
     .7,0,0,
     0,.7,0,
     0,.7,0,
     0,.7,0,
     0,.7,0,
     0,0,.7,
     0,0,.7,
     0,0,.7,
     0,0,.7),
  nrow = 12,
  ncol = 3,
  byrow=TRUE)

phi <- matrix(c(1,.5,.5,
                .5, 1,.5,
                .5,.5,1),
              nrow = 3,
              ncol = 3,
              byrow=TRUE)

theta <- diag(diag(1-(lambda%*%phi%*%t(lambda))))

# Population correlation  matrix as provided in Table 1 of Lv & Maeda
Rpop <- lambda%*%phi%*%t(lambda) + theta
```

```r
# 10 studies with on average n = 200 (sd = 1)
k <- 10
average.n <- 200


# specify matrices with missing variables and missing correlations
# here pf = .20 and pm = .17, so 8/10 studies miss 2 variables at random

nvar <- 12
varnames <- paste("x",1:nvar,sep="")
missvar <- matrix(FALSE,k,nvar)

# no additional missing at the correlation level

ncor <- nvar*(nvar-1)/2
cornames <- vechs(outer(1:nvar, 1:nvar, function(x, y) paste(varnames[x],varnames[y], sep = "_")))
misscor <- matrix(FALSE,k,ncor)
colnames(misscor) <- cornames

# fitted model

model <-"
f1 =~ x1+x2+x3+x4
f2 =~ x5+x6+x7+x8
f3 =~ x9+x10+x11+x12
f3 ~~ f2
f3 ~~ f1
f2 ~~ f1
"
RAM <- lavaan2RAM(model, obs.variables = paste0("x", 1:12))

# run simulation

run.sim <- function(){

        # generate vector of k sample sizes as Lu and maeda (2019)
        n <- c()
        for (i in 1:k){n[i] <- as.integer(rnorm(1,mean=average.n,sd=1))}

        # specify missing variables for studies 3 to 10
        for (i in 3:10){
          del.var <- sample(1:12,2,replace=FALSE)
          missvar[i,del.var] <- TRUE
        }


        # generate data
        data <- gendat(Rpop,k,n,missvar,misscor)


        # Stage 1 TSSEM
        Stage1.TSSEM <- tssem1FEM(data,n)
```

```
        # Stage 1 GLS
        output.GLS.stage1 <- Stage1.GLS(data,n)


        # Stage 2 TSSEM
        Stage2.TSSEM <- try(wls(Cov = coef.tssem1FEM(Stage1.TSSEM), aCov = vcov.tssem1FEM(Stage1.TSSEM)
                                n = sum(n), Amatrix = RAM$A, Smatrix = RAM$S, Fmatrix=RAM$F),silent=TRU

            if (is.element("try-error", class(Stage2.TSSEM))) {
              resultsStage2.TSSEM <- rep(99,33)
            }   else {
              resultsStage2.TSSEM <-c(
                Stage2.TSSEM$mx.fit$output$fit,
                Stage2.TSSEM$mx.fit$output$estimate,
                Stage2.TSSEM$mx.fit$output$standardErrors,
                Stage1.TSSEM$mx.fit$output$status$code,
                Stage2.TSSEM$mx.fit$output$status$code)
            }

        # Stage 2 GLS

        Stage2.GLS <- try(wls(Cov = output.GLS.stage1$R.GLS, aCov = output.GLS.stage1$acov.GLS,
                              n = sum(n), Amatrix = RAM$A, Smatrix = RAM$S, Fmatrix=RAM$F),silent=TRUE)

            if (is.element("try-error", class(Stage2.GLS))) {
              resultsStage2.GLS <- rep(99,33)
            }   else {
              resultsStage2.GLS <- c(
                Stage2.GLS$mx.fit$output$fit,
                Stage2.GLS$mx.fit$output$estimate,
                Stage2.GLS$mx.fit$output$standardErrors,
                0,
                Stage2.GLS$mx.fit$output$status$code)
            }

        return(data.frame(resultsStage2.GLS,resultsStage2.TSSEM))

}


sim.results<-replicate(2000,run.sim(),simplify=FALSE)

save.image(file="resultsmissing.RData")
```

Calculate bias in parameter estimates and standard errors

```
# view results from first two replications
sim.results[1:2]
```

```
## [[1]]
##    resultsStage2.GLS resultsStage2.TSSEM
## 1        51.16792100         49.87518373
```

```
## 2         0.71393533          0.71488261
## 3         0.70337181          0.70593668
## 4         0.69507859          0.69567327
## 5         0.73900785          0.73903764
## 6         0.71883669          0.72075928
## 7         0.70817954          0.71102273
## 8         0.69697042          0.70076588
## 9         0.70022844          0.70112560
## 10        0.67991983          0.68109248
## 11        0.72210988          0.72533326
## 12        0.70259933          0.70303769
## 13        0.68329098          0.68439976
## 14        0.51080024          0.51183477
## 15        0.51309356          0.51264483
## 16        0.49225696          0.49283676
## 17        0.01671609          0.01671618
## 18        0.01689681          0.01688982
## 19        0.01598761          0.01619705
## 20        0.01518899          0.01534197
## 21        0.01541651          0.01558977
## 22        0.01590957          0.01604648
## 23        0.01762889          0.01786470
## 24        0.01548481          0.01555298
## 25        0.01676071          0.01685342
## 26        0.01524145          0.01544973
## 27        0.01774231          0.01808563
## 28        0.01673715          0.01683162
## 29        0.02367781          0.02342941
## 30        0.02367305          0.02356867
## 31        0.02426051          0.02391612
## 32        0.00000000          0.00000000
## 33        0.00000000          0.00000000
##
## [[2]]
##    resultsStage2.GLS resultsStage2.TSSEM
## 1        44.57370721         44.58746134
## 2         0.67109919          0.67326725
## 3         0.72424374          0.72487770
## 4         0.75941974          0.75829240
## 5         0.71123874          0.71408192
## 6         0.71665958          0.71946358
## 7         0.67206260          0.67329718
## 8         0.70267625          0.70219689
## 9         0.70984858          0.71238489
## 10        0.72091173          0.72184982
## 11        0.68335066          0.68578555
## 12        0.71201385          0.71161295
## 13        0.70825589          0.70897654
## 14        0.53578153          0.53622865
## 15        0.51552751          0.51581004
## 16        0.50983805          0.51002952
## 17        0.01745494          0.01781991
## 18        0.01627941          0.01601780
## 19        0.01707464          0.01702693
```

```
## 20        0.01654862          0.01687507
## 21        0.01528326          0.01539235
## 22        0.01797423          0.01821899
## 23        0.01630116          0.01649413
## 24        0.01848813          0.01830818
## 25        0.01448587          0.01462348
## 26        0.01603087          0.01621317
## 27        0.01488242          0.01499785
## 28        0.01535215          0.01553168
## 29        0.02311360          0.02314454
## 30        0.02320080          0.02310176
## 31        0.02329713          0.02318709
## 32        0.00000000          0.00000000
## 33        0.00000000          0.00000000
```

```
pop.values <- c(rep(.7,12),rep(.5,3))

par.labels <- c(paste0("lambda",1:12),paste0("phi",1:3))
SE.labels <- c(paste0("SE.lambda",1:12),paste0("SE.phi",1:3))

labels <- c("chi2",par.labels,SE.labels,"Stg.error1","Stg.error2")

GLS.results <- matrix(NA,length(sim.results),33)
TSSEM.results <- matrix(NA,length(sim.results),33)
colnames(GLS.results) <- labels
colnames(TSSEM.results) <- labels

for (r in 1:length(sim.results)){
  GLS.results[r,] <- sim.results[[r]][,1]
  TSSEM.results[r,] <- sim.results[[r]][,2]
}

# nonconvergence
sum(TSSEM.results[,32:33])
```

```
## [1] 0
```

```
sum(GLS.results[,32:33])
```

```
## [1] 0
```

```
# check for replications without results
sum(TSSEM.results==99)
```

```
## [1] 0
```

```
sum(GLS.results==99)
```

```
## [1] 0
```

```
# parameter bias
(apply(TSSEM.results[,2:16],2,mean)-pop.values)/pop.values*100
```

```
##   lambda1   lambda2   lambda3   lambda4   lambda5   lambda6   lambda7
## 0.6263341 0.6357878 0.6812354 0.5936272 0.7087313 0.7625331 0.6758261
##   lambda8   lambda9  lambda10  lambda11  lambda12      phi1      phi2
## 0.6473748 0.6579319 0.8306838 0.6966537 0.6157309 1.1769708 1.2065141
##      phi3
```

```
## 1.1485121
```

```
(apply(GLS.results[,2:16],2,mean)-pop.values)/pop.values*100
```

```
##   lambda1   lambda2   lambda3   lambda4   lambda5   lambda6   lambda7
## 0.4327334 0.4450663 0.4973109 0.4000521 0.5186187 0.5722675 0.4874057
##   lambda8   lambda9  lambda10  lambda11  lambda12      phi1      phi2
## 0.4570362 0.4706038 0.6390955 0.5037822 0.4301801 1.2218369 1.2470193
##      phi3
## 1.1890105
```

```
# SE bias
  # TSSEM
par.sd <- apply(TSSEM.results[,2:16],2,sd)
mean.se <- apply(TSSEM.results[,17:31],2,mean)

(mean.se - par.sd)/par.sd * 100
```

```
##  SE.lambda1  SE.lambda2  SE.lambda3  SE.lambda4  SE.lambda5  SE.lambda6
##  -0.7934559  -3.3449241  -1.7771766  -1.0235927  -2.3806739  -2.5627333
##  SE.lambda7  SE.lambda8  SE.lambda9 SE.lambda10 SE.lambda11 SE.lambda12
##  -3.7369958  -2.7209650  -3.4814850  -1.8537086  -1.6557302  -2.5837445
##     SE.phi1     SE.phi2     SE.phi3
##  -1.5554840  -0.7749029  -3.1620334
```

```
  # GLS
par.sd <- apply(GLS.results[,2:16],2,sd)
mean.se <- apply(GLS.results[,17:31],2,mean)

(mean.se - par.sd)/par.sd * 100
```

```
##  SE.lambda1  SE.lambda2  SE.lambda3  SE.lambda4  SE.lambda5  SE.lambda6
##  -1.7768437  -4.0067674  -2.4008884  -1.8714038  -3.2714686  -3.5613561
##  SE.lambda7  SE.lambda8  SE.lambda9 SE.lambda10 SE.lambda11 SE.lambda12
##  -4.6878456  -3.8748046  -4.4845104  -2.8349505  -2.5680496  -3.3407703
##     SE.phi1     SE.phi2     SE.phi3
##  -1.2514793  -0.4663699  -2.8641244
```

Reanalysis of Lv and Maeda (2019) with W-COV GLS with multiple imputation

Suzanne Jak and Mike W.-L. Cheung

25 october 2019

Simulate data

We generate data for 10 studies (k = 10), with an average sample size of 200 (n = 200), including two studies with complete data (pc = .20), and 8 studies that missed two variables (pm = .17), and equal factor loadings. According to Table 5 in Lv and Maeda (2019), this specification should lead to biased parameter estimates and biased standard errors for all methods.

```
set.seed(22122018)

library(metaSEM)
library(MASS)
library(mice)
source("Generatedata.R")
source("GLSapproach.R")

###### population model

lambda <- matrix(
  c( .7,0,0,
     .7,0,0,
     .7,0,0,
     .7,0,0,
     0,.7,0,
     0,.7,0,
     0,.7,0,
     0,.7,0,
     0,0,.7,
     0,0,.7,
     0,0,.7,
     0,0,.7),
  nrow = 12,
  ncol = 3,
  byrow=TRUE)

phi <- matrix(c(1,.5,.5,
                .5, 1,.5,
                .5,.5,1),
              nrow = 3,
              ncol = 3,
              byrow=TRUE)

theta <- diag(diag(1-(lambda%*%phi%*%t(lambda))))

Rpop <- lambda%*%phi%*%t(lambda) + theta
```

```r
# Population correlation  matrix as provided in Table 1 of Lv & Maeda
Rpop

# 10 studies with on average n = 200 (sd = 1)
k <- 10
average.n <- 200


# specify matrices with missing variables and missing correlations
# here pf = .20 and pm = .17, so 8/10 studies miss 2 variables at random

nvar <- 12
varnames <- paste("x",1:nvar,sep="")
missvar <- matrix(FALSE,k,nvar)

# no additional missing at the correlation level

ncor <- nvar*(nvar-1)/2
cornames <- vechs(outer(1:nvar, 1:nvar,
                        function(x, y) paste(varnames[x],varnames[y], sep = "_")))
misscor <- matrix(FALSE,k,ncor)
colnames(misscor) <- cornames

# fitted model

model <-"
f1 =~ x1+x2+x3+x4
f2 =~ x5+x6+x7+x8
f3 =~ x9+x10+x11+x12
f3 ~~ f2
f3 ~~ f1
f2 ~~ f1
"
RAM <- lavaan2RAM(model, obs.variables = paste0("x", 1:12))

# simulation function

run.sim.mi <- function(){

        # sample sizes of primary studies
        n <- c()
        for (i in 1:k){n[i] <- as.integer(rnorm(1,mean=average.n,sd=1))}

        # specify missing variables for studies 3 to 10
        for (i in 3:10){
          del.var <- sample(1:12,2,replace=FALSE)
          missvar[i,del.var] <- TRUE
        }


        # generate data
        data <- gendat(Rpop,k,n,missvar,misscor)
```

```
        r <- t(sapply(data,vechs))

        # use m=40 imputations
        m <- 40

        imp.data <- mice(r,m=m,diagnostics = FALSE,remove_collinear = FALSE, printFlag = FALSE)

        list.imputed.data <- complete(imp.data, action = "all")

        # create list of m imputations containing k correlation  matrices

        datalist <- lapply(list.imputed.data,readstack,no.var = 12, diag = FALSE)

        # fit stage 1 model to each imputed dataset
        stage1 <- lapply(datalist,Stage1.GLS,n=n)

        # fit stage 2 model to each stage 1 pooled matrix
        stage2 <- list()
        for (i in 1:length(stage1)){
          stage2[[i]] <- summary(wls(Cov = stage1[[i]]$R.GLS,
                                     aCov = stage1[[i]]$acov.GLS, n = sum(n),
                                     Amatrix = RAM$A, Smatrix = RAM$S, Fmatrix=RAM$F))$coefficients
        }

        # average parameter estimates across imputations

        estimates <- matrix(NA,m,15)
        for (i in 1:m){
          estimates[i,] <- stage2[[i]][,'Estimate']
        }

        par.est <- apply(estimates,2,mean)

        # use Rubin's rules to calcuate SE's

        stderr <- matrix(NA,m,15)
        for (i in 1:m){
          stderr[i,] <- stage2[[i]][,'Std.Error']
        }

        var.within <- apply(stderr^2,2,mean)

        var.between <- (1/(m-1)) * sum((t(estimates) - par.est)^2)

        var.mi <- var.within + (1 + 1/m) * var.between

        se.mi <- sqrt(var.se)

        return(c(par.est,se.mi))

}
```

```
sim.results<-replicate(200,run.sim.mi(),simplify=FALSE)

save.image(file="resultsMI.RData")
```

Calculate bias in parameter estimates and standard errors

```
pop.values <- c(rep(.7,12),rep(.5,3))

head(sim.results)
```

```
## [[1]]
##  [1] 0.71503058 0.72511402 0.71245165 0.69017948 0.70372350 0.70127790
##  [7] 0.74523120 0.71887279 0.71231506 0.69866182 0.70367837 0.68223294
## [13] 0.50685212 0.51495461 0.48541938 0.02785866 0.02788315 0.02803653
## [19] 0.02828790 0.02798481 0.02798552 0.02758547 0.02783814 0.02791340
## [25] 0.02809693 0.02809798 0.02841184 0.03274829 0.03269013 0.03325668
##
## [[2]]
##  [1] 0.70870979 0.72477441 0.69670760 0.72408940 0.70712328 0.71841110
##  [7] 0.71700098 0.68815392 0.69324599 0.70437828 0.77163766 0.73077010
## [13] 0.57419692 0.51058972 0.50976734 0.02810802 0.02792218 0.02829529
## [19] 0.02792563 0.02816330 0.02802721 0.02805731 0.02829904 0.02830073
## [25] 0.02812482 0.02739824 0.02775249 0.03170147 0.03273080 0.03272222
##
## [[3]]
##  [1] 0.71570635 0.72295143 0.70413986 0.71671833 0.73067074 0.70219740
##  [7] 0.71127155 0.71684321 0.73738750 0.70354094 0.70722501 0.72167413
## [13] 0.53394805 0.51924997 0.53480777 0.02686617 0.02684907 0.02704700
## [19] 0.02689741 0.02668434 0.02709299 0.02699943 0.02687402 0.02666710
## [25] 0.02708925 0.02701768 0.02681029 0.03144683 0.03175133 0.03139772
##
## [[4]]
##  [1] 0.72110026 0.67654067 0.70165895 0.71508568 0.71163136 0.71263035
##  [7] 0.68365817 0.69639660 0.70824299 0.68811331 0.70060839 0.71329282
## [13] 0.50934078 0.54536360 0.51053897 0.02976658 0.03029906 0.02998210
## [19] 0.02974473 0.02981241 0.02969057 0.03008593 0.03006090 0.02995375
## [25] 0.03017406 0.03003051 0.02986273 0.03430856 0.03389037 0.03447071
##
## [[5]]
##  [1] 0.69805391 0.67456715 0.71184838 0.72471763 0.70927903 0.74202004
##  [7] 0.70906318 0.69314612 0.74241656 0.72671282 0.72463064 0.69943015
## [13] 0.52852668 0.53227484 0.53793671 0.02880220 0.02924323 0.02865354
## [19] 0.02858588 0.02859062 0.02819629 0.02865499 0.02875183 0.02816034
## [25] 0.02839207 0.02840287 0.02887812 0.03301470 0.03314853 0.03299153
##
## [[6]]
##  [1] 0.73108551 0.71322019 0.70489185 0.70522072 0.70650354 0.72203146
##  [7] 0.67407293 0.73369767 0.67795883 0.72669594 0.72129542 0.71160935
## [13] 0.45371716 0.53265064 0.51745781 0.02645163 0.02668424 0.02673269
## [19] 0.02672835 0.02679317 0.02650343 0.02714771 0.02637627 0.02711838
## [25] 0.02653206 0.02655756 0.02665583 0.03244558 0.03141849 0.03152225
```

```r
MI.results <- matrix(unlist(sim.results),200,30,byrow = TRUE)

par.labels <- c(paste0("lambda",1:12),paste0("phi",1:3))
SE.labels <- c(paste0("SE.lambda",1:12),paste0("SE.phi",1:3))

labels <- c(par.labels,SE.labels)

colnames(MI.results) <- labels

# check inadmissible values
min(MI.results)
```

```
## [1] 0.02272599
```

```r
max(MI.results)
```

```
## [1] 0.7716377
```

```r
sum(is.na(MI.results))
```

```
## [1] 0
```

```r
# parameter bias
(apply(MI.results[,1:15],2,mean)-pop.values)/pop.values*100
```

```
##   lambda1   lambda2   lambda3   lambda4   lambda5   lambda6   lambda7
## 1.3173741 1.1149674 1.1375910 0.7402200 1.1466359 1.2633714 0.8608357
##   lambda8   lambda9  lambda10  lambda11  lambda12      phi1      phi2
## 0.9054973 1.1772660 0.7608392 1.1052273 0.9613409 1.5140425 1.5263760
##      phi3
## 2.3146646
```

```r
# SE bias
par.sd <- apply(MI.results[,1:15],2,sd)
mean.se <- apply(MI.results[,16:30],2,mean)

(mean.se - par.sd)/par.sd * 100
```

```
##  SE.lambda1  SE.lambda2  SE.lambda3  SE.lambda4  SE.lambda5  SE.lambda6
##    67.11042    52.97960    63.23219    54.15681    54.82657    64.08922
##  SE.lambda7  SE.lambda8  SE.lambda9 SE.lambda10 SE.lambda11 SE.lambda12
##    49.43474    65.61539    46.94457    58.26565    51.62576    39.09454
##     SE.phi1     SE.phi2     SE.phi3
##    35.14733    30.21316    28.42673
```

Functions used to generate data and to apply GLS

```r
# gendat() function to generate data, returns a list with correlation matrices
# removedat() function used in gendat() to impose missing data, as defined by missvar and misscor
# readstack() function read stacked vector, reads in nstudy by ncor dataframe or matrix,
#  outputs list of correlation matrices


# missvar is an nstudy by nvar matrix with TRUE for missing variables
# misscor is an nstudy by ncor matrix with TRUE for missing at correlation level
```

```r
# Rpop is the population correlation matrix
# k is the number of studies
# n is a vector with sample sizes of the studies


# gendat() function to create data

gendat <- function(Rpop, k, n,  missvar, misscor){
  nvar <- nrow(Rpop)
  ncor <- nvar*(nvar-1)/2
  datfull <- matrix(NA,k,nvar+ncor)

  repeat{

    for (i in 1:k){
      datfull[i,] <- vech(cor(mvrnorm(n[i],rep(0,nvar),Rpop)))
    }

    removedata <- removedat(datfull,missvar,misscor,n)

    matrices_miss <- removedata[[1]]
    matcheckpd <- removedata[[2]]

    # check npd, sample again if npd
    checkpd <- is.pd(matcheckpd)
  if(sum(checkpd,na.rm=TRUE)==k) break}


return(matrices_miss)
}


# removedat() function to remove data

removedat <- function(datfull, missvar, misscor,n) {
  # remove correlations as in missvar and misscor
  datmiss <- datfull
  nvar <- ncol(missvar)
  ncor <- ncol(misscor)

  diags <- ifelse(missvar==TRUE,NA,1)
  offdiags <- ifelse(misscor==TRUE,NA,1)

  # create nstudy by ncor+nvar matrix with NA's for missvar en misscor
  pattern.varmiss <- matrix(0,k,ncor+nvar)
  pattern.cormiss <- matrix(0,k,ncor+nvar)

  for (i in 1:nrow(diags)) {
    pattern.varmiss[i,] <- vech(outer(1:nvar, 1:nvar, function(x, y) diags[i,x]+diags[i,y]))
    pattern.cormiss[i,] <- vech(vec2symMat(as.matrix(offdiags[i,]),diag=FALSE))
  }

  # for overall pattern
```

```
  #pattern.miss <- pattern.varmiss + pattern.cormiss

  # replace missing correlations with meanr as in noncor
  datmiss[is.na(pattern.cormiss) == TRUE] <- NA
  datmiss[is.na(pattern.varmiss) == TRUE] <- NA


  # create list of cormatrices
  matrices_miss <- readstack(datmiss, no.var = ncol(diags))

  # create list of cormatrices for pdcheck

  # univariate r
  r <- t(sapply(matrices_miss,vechs))
  n_i <- matrix(n,nrow(r),ncol(r))
  n_i[is.na(r)] <- NA
  meanr <- colSums(r*n_i,na.rm=TRUE)/colSums(n_i,na.rm=TRUE)
  meanr_i <- matrix(meanr,nrow(r),ncol(r),byrow=TRUE)

  datcheckpd <- r

  datcheckpd[is.na(r)] <- meanr_i[is.na(r)]

  matcheckpd <- readstack(datcheckpd, no.var = ncol(diags), diag=FALSE)

  return(list(matrices_miss,matcheckpd))
}


# readstack() function to read stacked vector, output list of correlation matrices
readstack <- function(my.df,no.var,diag=TRUE){
  my.df <- as.matrix(my.df)
  no.groups <- nrow(my.df)
  my.list <- split(my.df, 1:no.groups)
  if(diag==TRUE){
    my.mat <- lapply(my.list, function(x, no.var) {
      mat <- matrix(0, ncol = no.var, nrow = no.var)
      mat[lower.tri(mat, diag = TRUE)] <- x
      mat[upper.tri(mat)] <- t(mat)[upper.tri(mat)]
      mat
    }, no.var)
  }
  if(diag==FALSE){
    my.mat <- lapply(my.list, function(x, no.var) {
      mat <- matrix(0, ncol = no.var, nrow = no.var)
      mat[lower.tri(mat, diag = FALSE)] <- x
      mat[upper.tri(mat)] <- t(mat)[upper.tri(mat)]
      diag(mat)<-1
      mat
    }, no.var)
  }

  var.names <- paste("v", 1:no.var, sep = "")
```

```
  out <- lapply(my.mat, function(x, v.names) {
    dimnames(x) <- list(v.names, v.names)
    x
  }, var.names)
  return(out)
}


# compacov_meanr() returns list of ACOV's for each study
# Stage1.GLS() returns list with pooled R and acov from GLS

# Need to install the metaSEM and Matrix packages first

library(metaSEM)
library(Matrix)


# compacov with weighted mean r's
# arguments are list of correlation matrices and vector of sample sizes
# returns list of V's

compacov_meanr <- function(data,n){

  nvar <- nrow(data[[1]])
  rdat <- t(sapply(data,vechs))
  varnames <- paste("v",1:nvar,sep="")

  # mean weighted by inverse variance
  rdat.v <- (1-rdat^2)^2/n
  weights <- 1/rdat.v
  rpool <- apply(weights*rdat,2,sum,na.rm=TRUE)/apply(weights,2,sum,na.rm=TRUE)


  # Alternative 2: mean weighted by sample size
  #rpool <- apply(rdat*matrix(rep(n/sum(n),ncol(rdat)),nrow(rdat),ncol(rdat)),2,sum,na.rm=TRUE)

  # Alternative 3: simple mean
  #rpool <- apply(rdat,2,mean,na.rm=TRUE)

  r <- vec2symMat(as.matrix(rpool),diag = FALSE)

  rlabels <- outer(1:nvar, 1:nvar,
                   function(x, y) paste(varnames[x],varnames[y], sep = "_"))
  vec <- vechs(rlabels)

  V <- matrix(0,nvar*(nvar-1)/2,nvar*(nvar-1)/2)
  dimnames(V) <- list(vec,vec)

  for (i in 1:nrow(V)){
    for (j in 1:ncol(V)){

      row <- rownames(V)[i]
      col <- rownames(V)[j]
```

```
      s <- max(apply(rlabels,2,function(x) match(row,x)),na.rm=TRUE)
      t <- max(apply(rlabels,1,function(x) match(row,x)),na.rm=TRUE)
      u <- max(apply(rlabels,2,function(x) match(col,x)),na.rm=TRUE)
      v <- max(apply(rlabels,1,function(x) match(col,x)),na.rm=TRUE)

      r_su <- r[s,u]
      r_sv <- r[s,v]
      r_tu <- r[t,u]
      r_tv <- r[t,v]
      r_st <- r[s,t]
      r_uv <- r[u,v]

      V[i,j] <-
        (0.5*r_st*r_uv*(r_su^2+r_sv^2+r_tu^2+r_tv^2)
         + r_su*r_tv+r_sv*r_tu-(r_st*r_su*r_sv
                                + r_st*r_tu*r_tv+r_su*r_tu*r_uv
                                + r_sv*r_tv*r_uv))
    }}

  V.out <- list()
  for (i in 1:length(n)){V.out[[i]] <- V/n[i]}

  return(V.out)
}


####### Stage1.GLS()

Stage1.GLS <- function(data,n){

  nvar <- nrow(data[[1]])
  varnames <- paste("v",1:nvar,sep="")

  ncor <- nvar*(nvar-1)/2
  r <- t(sapply(data,vechs))
  present <- (is.na(r)==FALSE)/1

  rlabels <- vechs(outer(1:nvar, 1:nvar,
                         function(x, y) paste(varnames[x],varnames[y], sep = "_")) )

  # create filter matrices

    Id <- diag(1,ncor)
    createX <- function(present){t(Id[present==TRUE,])}
    d <- apply(present,1,createX)

  X <- matrix(unlist(d),nrow=sum(present),ncol=ncor,byrow=TRUE)

  # vector of present correlations
  rvec <- c(t(r))
  rvec <- rvec[is.na(rvec)==FALSE]

  # V matrices calculated with mean correlations (Becker & Fahrbach (1994), Hafdahl (2007))
```

```
  Vi <- compacov_meanr(data,n)

    # remove missing correlations from Vi's
          for (i in 1:length(Vi)){
                        Vi[[i]] <- Vi[[i]][present[i,]==1,present[i,]==1]}
      V <- as.matrix(bdiag(Vi))

  # calculate mean correlations and acov

  means_GLS_fixed <- solve(t(X)%*%solve(V)%*%X)%*%t(X)%*%solve(V)%*%rvec
  acov.GLS <- solve(t(X)%*%solve(V)%*%X)

  R.GLS <- vec2symMat(means_GLS_fixed,diag=FALSE)

  dimnames(R.GLS) <- list(varnames,varnames)
  dimnames(acov.GLS) <- list(rlabels,rlabels)


  # calculate Qgls

    Q.gls <- t(rvec) %*% (solve(V) - solve(V) %*% X%*%solve(t(X)
             %*% solve(V) %*% X)%*%t(X) %*% solve(V)) %*% rvec
    df.Q <- length(rvec) - ncor

    out <- list(R.GLS = R.GLS,acov.GLS = acov.GLS,
                Q.gls = Q.gls, df.Q = df.Q)


return(out)

}
```

Session information

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Dutch_Netherlands.1252  LC_CTYPE=Dutch_Netherlands.1252
## [3] LC_MONETARY=Dutch_Netherlands.1252 LC_NUMERIC=C
## [5] LC_TIME=Dutch_Netherlands.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
```

```
##  [1] compiler_3.6.1  magrittr_1.5    tools_3.6.1     htmltools_0.3.6
##  [5] yaml_2.2.0      Rcpp_1.0.1      stringi_1.4.3   rmarkdown_1.13
##  [9] knitr_1.23      stringr_1.4.0   xfun_0.7        digest_0.6.19
## [13] evaluate_0.13
```