This is one part of the simulation study to assess the performance of NCT: network structure invariance test, density=.1 and equally sized groups.

```
#!/usr/bin/env Rscript
args <- commandArgs(trailingOnly=TRUE)
if (length(args)==0){
  args <- 1
}


library(foreach)
library(parallel)
library(doParallel)
library(psych)
library("MASS")
library(GPArotation)
library("corpcor")
library("qgraph")
library("polycor")
library(pdist)
library(matrixcalc)
library(ggm)
library(sm)
library(lavaan)
library(Matrix)

###CLAUDIAS NCT code######
NCT <- function(data1,
                data2,
                gamma,
                it = 100,
                binary.data=FALSE,
                paired=FALSE,
                weighted=TRUE,
                AND=TRUE,
                test.edges=FALSE,
                edges,
                progressbar=TRUE,
                make.positive.definite=TRUE){

  if (missing(gamma)){
    if (binary.data){
      gamma <- 0.25
    } else {
      gamma <- 0.5
    }
  }

  if (progressbar==TRUE) pb <- txtProgressBar(max=it, style = 3)
  x1 <- data1
  x2 <- data2
  nobs1 <- nrow(x1)
  nobs2 <- nrow(x2)
  dataall <- rbind(x1,x2)
  data.list <- list(x1,x2)
  b <- 1:(nobs1+nobs2)
```

```
  nvars <- ncol(x1)
  nedges <- nvars*(nvars-1)/2

  glstrinv.perm <- glstrinv.real <- nwinv.real <- nwinv.perm <- c()
  diffedges.perm <- matrix(0,it,nedges)
  diffedges.permtemp <- matrix(0, nvars, nvars)
  einv.perm.all <- array(NA,dim=c(nvars, nvars, it))
  edges.pval.HBall <- matrix(NA,nvars,nvars)
  edges.pvalmattemp <- matrix(0,nvars,nvars)

  ####################################
  ### procedure for non-binary data ###
  ####################################
  ## Real data
  if(binary.data==FALSE)
  {

    cor_x1 <- cor(x1)
    cor_x2 <- cor(x2)

    if(make.positive.definite){
      # cor_x1 <- make.positive.definite(cor_x1)
      # cor_x2 <- make.positive.definite(cor_x2)
      cor_x1 <- matrix(nearPD(cor_x1, corr=TRUE)$mat, ncol = nvars)
      cor_x1 <- (cor_x1 + t(cor_x1)) / 2 # make symmetric
      cor_x2 <- matrix(nearPD(cor_x2, corr=TRUE)$mat, ncol = nvars)
      cor_x2 <- (cor_x2 + t(cor_x2)) / 2 # make symmetric
    }

    nw1 <- EBICglasso(cor_x1,nrow(x1),gamma=gamma)
    nw2 <- EBICglasso(cor_x2,nrow(x2),gamma=gamma)
    if(weighted==FALSE){
      nw1=(nw1!=0)*1
      nw2=(nw2!=0)*1
    }

    ##### Invariance measures #####

    ## Global strength invariance
    glstrinv.real <- abs(sum(abs(nw1[upper.tri(nw1)])) -
sum(abs(nw2[upper.tri(nw2)])))
    # global strength of individual networks
    glstrinv.sep <- c(sum(abs(nw1[upper.tri(nw1)])),
sum(abs(nw2[upper.tri(nw2)])))

    ## Individual edge invariance
    diffedges.real <- abs(nw1-nw2)[upper.tri(abs(nw1-nw2))]
    diffedges.realmat <- matrix(diffedges.real,it,nedges,byrow=TRUE)
    diffedges.realoutput <- abs(nw1-nw2)

    ## Network structure invariance
    nwinv.real <- max(diffedges.real)


    ## permuted data
    for (i in 1:it)
    {
```

```
      if(paired==FALSE)
      {
        s <- sample(1:(nobs1+nobs2),nobs1,replace=FALSE)
        x1perm <- dataall[s,]
        x2perm <- dataall[b[-s],]

        cor_x1 <- cor(x1perm)
        cor_x2 <- cor(x2perm)

        if(make.positive.definite){
          # cor_x1 <- make.positive.definite(cor_x1)
          # cor_x2 <- make.positive.definite(cor_x2)
          cor_x1 <- matrix(nearPD(cor_x1, corr=TRUE)$mat, ncol = nvars)
          cor_x1 <- (cor_x1 + t(cor_x1)) / 2 # make symmetric
          cor_x2 <- matrix(nearPD(cor_x2, corr=TRUE)$mat, ncol = nvars)
          cor_x2 <- (cor_x2 + t(cor_x2)) / 2 # make symmetric
        }



        r1perm <- EBICglasso(cor_x1,nrow(x1perm),gamma=gamma)
        r2perm <- EBICglasso(cor_x2,nrow(x2perm),gamma=gamma)
        if(weighted==FALSE){
          r1perm=(r1perm!=0)*1
          r2perm=(r2perm!=0)*1
        }
      }



      if(paired==TRUE)
      {
        s <- sample(c(1,2),nobs1,replace=TRUE)
        x1perm <- x1[s==1,]
        x1perm <- rbind(x1perm,x2[s==2,])
        x2perm <- x2[s==1,]
        x2perm <- rbind(x2perm,x1[s==2,])

        cor_x1 <- cor(x1perm)
        cor_x2 <- cor(x2perm)

        if(make.positive.definite){
          # cor_x1 <- make.positive.definite(cor_x1)
          # cor_x2 <- make.positive.definite(cor_x2)
          cor_x1 <- matrix(nearPD(cor_x1, corr=TRUE)$mat, ncol = nvars)
          cor_x1 <- (cor_x1 + t(cor_x1)) / 2 # make symmetric
          cor_x2 <- matrix(nearPD(cor_x2, corr=TRUE)$mat, ncol = nvars)
          cor_x2 <- (cor_x2 + t(cor_x2)) / 2 # make symmetric
        }


        r1perm <- EBICglasso(cor(x1perm),nrow(x1perm),gamma=gamma)
        r2perm <- EBICglasso(cor(x2perm),nrow(x2perm),gamma=gamma)
        if(weighted==FALSE){
          r1perm=(r1perm!=0)*1
          r2perm=(r2perm!=0)*1
        }
```

```
      }

      ## Invariance measures for permuted data
      glstrinv.perm[i] <- abs(sum(abs(r1perm[upper.tri(r1perm)])) -
sum(abs(r2perm[upper.tri(r2perm)])))
      diffedges.perm[i,] <- abs(r1perm-r2perm)[upper.tri(abs(r1perm -
r2perm))]
      diffedges.permtemp[upper.tri(diffedges.permtemp, diag=FALSE)] <-
diffedges.perm[i,]
      diffedges.permtemp <- diffedges.permtemp + t(diffedges.permtemp)
      einv.perm.all[,,i] <- diffedges.permtemp
      nwinv.perm[i] <- max(diffedges.perm[i,])

      if (progressbar==TRUE) setTxtProgressBar(pb, i)
    }

    if(test.edges==TRUE)
    {
      # vector with uncorrected p values
      edges.pvaltemp <- colSums(diffedges.perm >= diffedges.realmat)/it
      # matrix with uncorrected p values
      edges.pvalmattemp[upper.tri(edges.pvalmattemp,diag=FALSE)] <-
edges.pvaltemp
      edges.pvalmattemp <- edges.pvalmattemp + t(edges.pvalmattemp)

      if(is.character(edges))
      {
        ept.HBall <- p.adjust(edges.pvaltemp, method='holm')
        # matrix with corrected p values
        edges.pval.HBall[upper.tri(edges.pval.HBall,diag=FALSE)] <- ept.HBall
        rownames(edges.pval.HBall) <- colnames(edges.pval.HBall) <-
colnames(data1)
        einv.pvals <- melt(edges.pval.HBall, na.rm=TRUE, value.name = 'p-
value')
        einv.perm <- einv.perm.all
        einv.real <- diffedges.realoutput
      }

      if(is.list(edges))
      {
        einv.perm <- matrix(NA,it,length(edges))
        colnames(einv.perm) <- edges
        uncorrpvals <- einv.real <- c()
        for(j in 1:length(edges))
        {
          uncorrpvals[j] <- edges.pvalmattemp[edges[[j]][1], edges[[j]][2]]
          einv.real[j] <- diffedges.realoutput[edges[[j]][1], edges[[j]][2]]
          for(l in 1:it){
            einv.perm[l,j] <- einv.perm.all[,,l][edges[[j]][1],
edges[[j]][2]]
          }
        }
        HBcorrpvals <- p.adjust(uncorrpvals, method='holm')
        einv.pvals <- HBcorrpvals
      }

      edges.tested <- colnames(einv.perm)
```

```
      res <- list(glstrinv.real = glstrinv.real,
                  glstrinv.sep = glstrinv.sep,
                  glstrinv.pval = sum(glstrinv.perm >= glstrinv.real)/it,
                  glstrinv.perm = glstrinv.perm,
                  nwinv.real = nwinv.real,
                  nwinv.pval = sum(nwinv.perm >= nwinv.real)/it,
                  nwinv.perm = nwinv.perm,
                  edges.tested = edges.tested,
                  einv.real = einv.real,
                  einv.pvals = einv.pvals,
                  einv.perm = einv.perm,
                  nw1 = nw1,
                  nw2 = nw2)
    }

    if (progressbar==TRUE) close(pb)

    if(test.edges==FALSE)
    {
      res <- list(
        glstrinv.real = glstrinv.real,
        glstrinv.sep = glstrinv.sep,
        glstrinv.pval = sum(glstrinv.perm >= glstrinv.real)/it,
        glstrinv.perm = glstrinv.perm,
        nwinv.real = nwinv.real,
        nwinv.pval = sum(nwinv.perm >= nwinv.real)/it,
        nwinv.perm = nwinv.perm,
        nw1 = nw1,
        nw2 = nw2
      )
    }

  }

  ##################################
  ### procedure for binary data ###
  ##################################
  ## Real data
  if(binary.data==TRUE)
  {
    IF1 <- IsingFit(x1, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
    IF2 <- IsingFit(x2, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
    nw1 <- IF1$weiadj
    nw2 <- IF2$weiadj
    if(weighted==FALSE){
      nw1=(nw1!=0)*1
      nw2=(nw2!=0)*1
    }

    ##### Invariance measures #####

    ## Global strength invariance
    glstrinv.real <- abs(sum(abs(nw1[upper.tri(nw1)])) -
sum(abs(nw2[upper.tri(nw2)])))
```

```
    # global strength of individual networks
    glstrinv.sep <- c(sum(abs(nw1[upper.tri(nw1)])),
sum(abs(nw2[upper.tri(nw2)])))

    ## Individual edge invariance
    diffedges.real <- abs(nw1-nw2)[upper.tri(abs(nw1-nw2))]
    diffedges.realmat <- matrix(diffedges.real,it,nedges,byrow=TRUE)
    diffedges.realoutput <- abs(nw1-nw2)

    ## Network structure invariance
    nwinv.real <- max(diffedges.real)

    ## permuted data
    for (i in 1:it)
    {
      if(paired==FALSE)
      {
        # a check to prevent an error of lognet(): a variable with only one 0
or 1 is not allowed
        checkN=0
        while(checkN==0){
          s <- sample(1:(nobs1+nobs2),nobs1,replace=FALSE)
          x1perm <- dataall[s,]
          x2perm <- dataall[b[-s],]

          cm1 <- colMeans(x1perm)
          cm2 <- colMeans(x2perm)
          checkN=ifelse(any(cm1<(1/nobs1)) | any(cm1>(nobs1-1)/nobs1) |
any(cm2<(1/nobs2)) | any(cm2>(nobs2-1)/nobs2),0,1)
        }
        IF1perm <- IsingFit(x1perm, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
        IF2perm <- IsingFit(x2perm, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
        r1perm <- IF1perm$weiadj
        r2perm <- IF2perm$weiadj
        if(weighted==FALSE){
          r1perm=(r1perm!=0)*1
          r2perm=(r2perm!=0)*1
        }
      }

      if(paired==TRUE)
      {
        checkN=0
        while(checkN==0)
        {
          s <- sample(c(1,2),nobs1,replace=TRUE)
          x1perm <- x1[s==1,]
          x1perm <- rbind(x1perm,x2[s==2,])
          x2perm <- x2[s==1,]
          x2perm <- rbind(x2perm,x1[s==2,])

          cm1 <- colMeans(x1perm)
          cm2 <- colMeans(x2perm)
          checkN=ifelse(any(cm1<(1/nobs1)) | any(cm1>(nobs1-1)/nobs1) |
any(cm2<(1/nobs2)) | any(cm2>(nobs2-1)/nobs2),0,1)
```

```
        }

        IF1perm <- IsingFit(x1perm, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
        IF2perm <- IsingFit(x2perm, AND = AND, gamma=gamma, plot=FALSE,
progressbar=FALSE)
        r1perm <- IF1perm$weiadj
        r2perm <- IF2perm$weiadj
        if(weighted==FALSE)
        {
          r1perm=(r1perm!=0)*1
          r2perm=(r2perm!=0)*1
        }
      }

      ## Invariance measures for permuted data
      glstrinv.perm[i] <- abs(sum(abs(r1perm[upper.tri(r1perm)])) -
sum(abs(r2perm[upper.tri(r2perm)])))
      diffedges.perm[i,] <- abs(r1perm - r2perm)[upper.tri(abs(r1perm -
r2perm))]
      diffedges.permtemp[upper.tri(diffedges.permtemp, diag=FALSE)] <-
diffedges.perm[i,]
      diffedges.permtemp <- diffedges.permtemp + t(diffedges.permtemp)
      einv.perm.all[,,i] <- diffedges.permtemp
      nwinv.perm[i] <- max(diffedges.perm[i,])

      if (progressbar==TRUE) setTxtProgressBar(pb, i)
    }

    if(test.edges==TRUE)
    {
      # vector with uncorrected p values
      edges.pvaltemp <- colSums(diffedges.perm >= diffedges.realmat)/it
      # matrix with uncorrected p values
      edges.pvalmattemp[upper.tri(edges.pvalmattemp,diag=FALSE)] <-
edges.pvaltemp
      edges.pvalmattemp <- edges.pvalmattemp + t(edges.pvalmattemp)

      if(is.character(edges))
      {
        ept.HBall <- p.adjust(edges.pvaltemp, method='holm')
        # matrix with corrected p values
        edges.pval.HBall[upper.tri(edges.pval.HBall,diag=FALSE)] <- ept.HBall
        rownames(edges.pval.HBall) <- colnames(edges.pval.HBall) <-
colnames(data1)
        einv.pvals <- melt(edges.pval.HBall, na.rm=TRUE, value.name = 'p-
value')
        einv.perm <- einv.perm.all
        einv.real <- diffedges.realoutput
      }


      if(is.list(edges))
      {
        einv.perm <- matrix(NA,it,length(edges))
        colnames(einv.perm) <- edges
        uncorrpvals <- einv.real <- c()
```

```
        for(j in 1:length(edges))
        {
          uncorrpvals[j] <- edges.pvalmattemp[edges[[j]][1], edges[[j]][2]]
          einv.real[j] <- diffedges.realoutput[edges[[j]][1], edges[[j]][2]]
          for(l in 1:it){
            einv.perm[l,j] <- einv.perm.all[,,l][edges[[j]][1],
edges[[j]][2]]
          }
        }
        HBcorrpvals <- p.adjust(uncorrpvals, method='holm')
        einv.pvals <- HBcorrpvals
      }

      edges.tested <- colnames(einv.perm)

      res <- list(glstrinv.real = glstrinv.real,
                  glstrinv.sep = glstrinv.sep,
                  glstrinv.pval = sum(glstrinv.perm >= glstrinv.real)/it,
                  glstrinv.perm = glstrinv.perm,
                  nwinv.real = nwinv.real,
                  nwinv.pval = sum(nwinv.perm >= nwinv.real)/it,
                  nwinv.perm = nwinv.perm,
                  edges.tested = edges.tested,
                  einv.real = einv.real,
                  einv.pvals = einv.pvals,
                  einv.perm = einv.perm)
    }


    if (progressbar==TRUE) close(pb)

    if(test.edges==FALSE)
    {
      res <- list(
        glstrinv.real = glstrinv.real,
        glstrinv.sep = glstrinv.sep,
        glstrinv.pval = sum(glstrinv.perm >= glstrinv.real)/it,
        glstrinv.perm = glstrinv.perm,
        nwinv.real = nwinv.real,
        nwinv.pval = sum(nwinv.perm >= nwinv.real)/it,
        nwinv.perm = nwinv.perm
      )
    }
  }

  class(res) <- "NCT"
  return(res)
}

## Methods:

summary.NCT <- function(x,...){

  cat("\n NETWORK INVARIANCE TEST
      Test statistic M: ", x$nwinv.real,
      "\n p-value", x$nwinv.pval,
      "\n\n GLOBAL STRENGTH INVARIANCE TEST
```

```
      Global strength per group: ", x$glstrinv.sep,
      "\n Test statistic S: ", x$glstrinv.real,
      "\n p-value", x$glstrinv.pval,
      "\n\n EDGE INVARIANCE TEST
      Edges tested: ", x$edges.tested,
      "\n Test statistic E: ", x$einv.real,
      "\n p-value", x$einv.pvals
  )

}

plot.NCT <- function(x,what = c("strength","network","edge"),...){

  what <- match.arg(what)

  ## Plot results of global strength invariance test (not reliable with only
10 permutations!):
  if (what == "strength"){
    hist(x$glstrinv.perm, main=paste('p =', x$glstrinv.pval), xlab =
'Difference in global strength', xlim = c(0, max(x$glstrinv.real,
x$glstrinv.perm)))
    points(x$glstrinv.real, 0, col='red', pch=17)
  }

  if (what == "network"){

    ## Plot results of the network invariance test (not reliable with only 10
permutations!):
    hist(x$nwinv.perm, main=paste('p =', x$nwinv.pval), xlab='Maximum of
difference', xlim = c(0, max(x$nwinv.real, x$nwinv.perm)))
    points(x$nwinv.real, 0, col='red', pch=17)
  }

  if (what == "edge"){

    ## Plot results of the maximum difference in edge weights (not reliable
with only 10 permutations)
    nedgetests <- ncol(x$einv.perm)
    for(i in 1:nedgetests){
      hist(x$einv.perm[,i], main=paste('p =', x$einv.pval[i]),
xlab=paste('Difference in edge strength ', colnames(x$einv.perm)[i]),
xlim=c(0, max(x$einv.real, x$einv.perm)))
      points(x$einv.real[i], 0, col='red', pch=17)
    }

  } #else stop("Method not implemented yet.")

}

######################## END OF CLAUDIAS NCT CODE
############################

graph.data <- function(d, n, gm=NA, theta=NA, prob, v=0.1, u=0.1,
upper.bound=.9, lower.bound=.3){
  # simulates graph
  count=0
  repeat{
```

```
    if(any(is.na(gm)) & any(is.na(theta))){
      gm.ig <- erdos.renyi.game(d,prob)
      gm <- as.matrix(get.adjacency(gm.ig))
      theta = matrix(runif(d^2, lower.bound, upper.bound), d, d) * gm
      diag(theta) = 0
      omega = theta * v
      diag(omega) = abs(min(Re(eigen(omega)$values))) + 0.1 + u
      omega <- as.matrix(forceSymmetric(omega))
      if(!is.positive.definite(omega)) gm=theta=NA else break
    }
    if(!any(is.na(gm))& !any(is.na(theta))){
      omega = theta * v
      diag(omega) = abs(min(Re(eigen(omega)$values))) + 0.1 + u
      omega <- as.matrix(forceSymmetric(omega))
      if(is.positive.definite(omega)) break
    }
    if(!any(is.na(gm))& any(is.na(theta))){
      theta = matrix(runif(d^2, lower.bound, upper.bound), d, d) * gm
      omega = theta * v
      diag(omega) = abs(min(Re(eigen(omega)$values))) + 0.1 + u
      omega <- as.matrix(forceSymmetric(omega))
      if(!is.positive.definite(omega)) theta=NA else break
    }
    count=count+1
    print(count)
  }
  sigma = cov2cor(solve(omega))
  omega = solve(sigma)
  x = mvrnorm(n, rep(0, d), sigma)
  sigmahat = cor(x)
  results <- list(data=x, sigma=sigma, theta=theta, gm=gm)
  return(results)
}



############ programming the permutation test in parallel test
############################


nCores <- 16
loops<-1000

#conditions
#hypotheses: H0 =H0 H1 =maximum edge split halves, H2=maximum edge is set to
zero
eq<-"equal" #group1 is denser than group2, equal= equal groupsizes, unequal1
= group1 (the denser group) is larger, unequal2= group2 the less dense group
is larger
dens<-0.1
l<-4#samplesizes
threeGammas <- list(gamma0=list(), gamma25=list(),g amma50=list())
  for(gam in 1:3){
    if(gam==1){gamma<-0}else if(gam==2) {gamma<-0.25} else {gamma<-0.5}
threeVariableNumbers < -list(NV10=list(), NV20=list(), NV30=list())
for(nv in 1:3){
  nV<-nv*10
```

```
    threehypotheses <- list(H0=list(), H1=list(), H2=list()) #H0 is equal, H1
is edge split half H2 is edge zero
    for(h in 0:2){
      hypothesis <- h
      allsamplesizes <- list(nobs250=list(), nobs500=list(), nobs750=list(),
nobs1000=list())
      for(n in 1:1){
        if(eq=="equal"){nobs <- n*250; nobs2<-n*250} else if(eq=="unequal1")
{nobs<-n*250; nobs2<-1.5*nobs} else {nobs2<-n*250; nobs<-1.5*nobs2}
        minnobs <- n*250
        cl <- nCores-1 # no of cores
        cl <- makeCluster(cl, outfile="") #registering
        registerDoParallel(cl)
        #saveoutput
        corgr1<-list(I(replicate(loops,matrix(rep(NA, nV*nV), nV))))
        corgr2<-list(I(replicate(loops,matrix(rep(NA, nV*nV), nV))))
        NW1<-list(I(replicate(loops,matrix(rep(NA, nV*nV), nV))))
        NW2<-list(I(replicate(loops,matrix(rep(NA, nV*nV), nV))))
        pvalues<-rep(NA,loops)
        #loop
        output<-foreach(gr=1:loops, .combine='rbind', .multicombine=TRUE,
.init=list(list(), list()), .packages=c("psych", "MASS", "GPArotation",
"corpcor", "qgraph", "polycor", "pdist", "matrixcalc", "ggm", "sm", "lavaan",
"igraph", "Matrix", "matrixcalc", "MASS")) %dopar% {
          graph<-graph.data(nV, nobs, prob=dens, upper.bound=-1,
lower.bound=-5)
          cor1<-graph$sigma
          pcor1<-round(cor2pcor(cor1),4)
          data1<-graph$data #mvrnorm(nobs,mu=rep(0,nV), Sigma=cor1)
          lowerpcor<-pcor1[lower.tri(pcor1)]
          edges<-which(lowerpcor!=0)
          nedges<-length(edges)
          if(hypothesis==0){
            pcor2<-pcor1
            data2<-mvrnorm(nobs2,mu=rep(0,nV),Sigma=cor1)
          }
          if(hypothesis==1){
            row<-as.numeric(which(pcor1 == max(lowerpcor), arr.ind = T)[1,1])
            col<-as.numeric(which(pcor1 == max(lowerpcor), arr.ind = T)[1,2])
            newvalue<-max(lowerpcor)/2
            pcor2<-pcor1
            pcor2[row,col]<-newvalue
            pcor2[col,row]<-newvalue
            cor2<-pcor2cor(pcor2)
            data2<-mvrnorm(nobs2, mu=rep(0,nV), Sigma=cor2)
          }
          if(hypothesis==2){
            row<-as.numeric(which(pcor1 == max(lowerpcor), arr.ind = T)[1,1])
            col<-as.numeric(which(pcor1 == max(lowerpcor), arr.ind = T)[1,2])
            newvalue<-0
            pcor2<-pcor1
            pcor2[row,col]<-newvalue
            pcor2[col,row]<-newvalue
            cor2<-pcor2cor(pcor2)
            data2<-mvrnorm(nobs2, mu=rep(0, nV), Sigma=cor2)
          }
```

```
          pvalues[gr]<-NCT(data1, data2, gamma=gamma,
progressbar=F)$nwinv.pval
          corgr1[[gr]]<-cor(data1)
          corgr2[[gr]]<-cor(data2)
          NW1[[gr]]<-pcor1
          NW2[[gr]]<-pcor2
          rejections<-list("pvalues"=pvalues[gr], "corgr1"=I(corgr1[[gr]]),
"corgr2"=I(corgr2[[gr]]), "NW1"=I(NW1[[gr]]), "NW2"=I(NW2[[gr]]))
          return(rejections)
        }
        stopCluster(cl)
        #str(output)
        output<-output[-1,]
        rownames(output)<-NULL

        #separate vectors from matrices (output1, elements are values,
output2 elements are matrices)
        output1<-output[,1]
        output2<-output[,2:5]

        #when one column in output1:
        dataA<-unlist(output1)
        #when multiple columns in output1:
        #dataA<-unlist(output1[,1])
        #for(i in 2:ncol(output1)){
        #  dataA<-cbind(dataA,unlist(output1[,i]))
        #}


        dataB<-I(output2[,1])
        for(i in 2:ncol(output2)){
          dataB<-data.frame(dataB,I(output2[,i]))
        }

        combinedoutput<-data.frame(dataA,dataB)
        colnames(combinedoutput)<-colnames(output)

        allsamplesizes[[n]]<-combinedoutput


        #sum(pvalues<0.05)/loops
      }
      threehypotheses[[h+1]]<-allsamplesizes
    }
    threeVariableNumbers[[nv]]<-threehypotheses
  }
threeGammas[[gam]]<-threeVariableNumbers
}
saveRDS(threeGammas,file=paste0("output_invariance_dens1_equal.rds"))


#saveRDS(completeoutput,file="bootstrap_5var_positive_2500nobs.rds")

#saveRDS(output_bootstrap,file=paste0("bootstrap_",args,".rds"))
```