```
##############################################################################
##############################################################################
##                                                                          ##
## Simulation code for binary responses for the paper published in JASA    ##
## "Informative estimation and selection of correlation structure for       ##
## longitudinal data" by Jianhui Zhou and Annie Qu                          ##
## (2012), V. 107, p. 701-710.                                              ##
##############################################################################
##############################################################################

library(geepack)
library(mvtnorm)
library(lars)
library(grplasso)
library(mvtBinaryEP)


##################################
#Symmetric square root of matrix x            #
##################################

symsqrt<-function(x)
{
 ee<-eigen(x,symmetric=T)
 ee$vectors%*%diag(sqrt(ee$values),dim(x)[1],dim(x)[1])%*%t(ee$vectors)
}


##################################################
#The weights, which is the derivative of the SCAD                        #
#penalty function as in (4) of Zhou and Qu (2012)                        #
##################################################

dp<-function(theta,lambda,a=3.7){
  p<-length(theta)
  b1<-rep(0,p)
  b1[theta>lambda]<-1
  b2<-rep(0,p)
  b2[theta<(lambda*a)]<-1
  lambda*(1-b1)+((lambda*a)-theta)*b2/(a-1)*b1
}


########################################
#Apply adaptive group lasso             #
#corresponding (4) in Zhou and Qu (2012)#
########################################

aglasso<-function(y,x,lambda,index,r){

   n<-length(y)

   coefs.lm<-lm(y~x-1)$coefficients
   coefs.lm[is.na(coefs.lm)]<-0

   group.id<-unique(index)
   group.num<-length(group.id)
   w<-NULL

    for (i in 1:group.num)
   {
     if (group.id[i]==1)
          w<-c(w,0)
     else
     {
      location<-which(index==group.id[i])
      coefs.group<-coefs.lm[location]
      L1.coefs.group<-sum(abs(coefs.group))
      w<-c(w, length(location)^r*dp(L1.coefs.group, lambda=lambda)*2*n)
     }
   }

# without penalize I
if(sum(w==0)==length(w))
        coefs1<-coefs.lm
    else{
              y.star<-sqrt(2)*(x%*%matrix(coefs.lm,ncol=1))
             x.star<-sqrt(2)*x

              groupU.id<-group.id[which(w==0)]
              groupV.id<-group.id[which(w!=0)]

               U.index<-which(index%in%groupU.id)
               V.index<-which(index%in%groupV.id)

             for(k in V.index)
             {
                 dp.k<-w[which(group.id==index[k])]
                x.star[,k]<-x.star[,k]/dp.k
             }

            if(length(U.index)>0)
            {
              xU.star<-x.star[,U.index]
               HU<-xU.star%*%solve(t(xU.star)%*%xU.star)%*%t(xU.star)
               y.star.star<-y.star-HU%*%matrix(y.star,ncol=1)
               xV.star.star<-x.star[,V.index]-HU%*%x.star[,V.index]

               index.V<-index[V.index]

               run.lasso<-lars(x=xV.star.star,y=y.star.star,type="lasso", normalize=F, intercept=F)

               coefs1V.star<-predict.lars(run.lasso, s=lambda, type="coefficients", mode="lambda")$coef

               coefs1U.star<-solve(t(xU.star)%*%xU.star)%*%t(xU.star)%*%(y.star-x.star[,V.index]%*%matrix(coefs1V.star,ncol=1))
```

```
            coefs1<-coefs.lm
            coefs1[V.index]<-coefs1V.star
            coefs1[U.index]<-coefs1U.star

            for(k in V.index)
            {
                dp.k<-w[which(group.id==index[k])]
               coefs1[k]<-coefs1[k]/dp.k
            }

           }

           else
           {
             y.star.star<-y.star
             xV.star.star<-x.star[,V.index]
             index.V<-index[V.index]

              run.lasso<-lars(x=xV.star.star,y=y.star.star,type="lasso", normalize=F, intercept=F)

              coefs1V.star<-predict.lars(run.lasso, s=lambda, type="coefficients", mode="lambda")$coef

             coefs1<-coefs1V.star
             for(k in V.index)
             {
               dp.k<-w[which(group.id==index[k])]
              coefs1[k]<-coefs1[k]/dp.k
             }

           }

   }
return(list(y=y,x=x,lambda=lambda,index=index, coefs.shrink=coefs1))
}


#####################################
#Select the tuning parameter $\lambda$#
#In the paper, the BIC-type were used #
#####################################

tuning.selection.R.unbalance<-function(R.est, y.scad, x.scad, sd.x, sd.y, N, m, index.gp, lambda.vect, r, l, M){
  aic.vect<-NULL
  bic.vect<-NULL

  for (j in 1:length(lambda.vect))
    {
     aglasso.fit.scad<-aglasso(y.scad, x.scad, lambda.vect[j], index.gp, r)
     coefs.shrink<-(aglasso.fit.scad$coefs.shrink)*(sd.y/sd.x)

     R.inv.est<-matrix(0, m,m)
     for (d in 1:dim(M)[3])
       {
         R.inv.est<-R.inv.est+M[,,d]*coefs.shrink[d]
       }

     eigen.value<-eigen(R.inv.est%*%R.est%*%R.est%*%R.inv.est)$values
     ratio<-eigen.value[1]/eigen.value[m]

     p.lam<-sum(coefs.shrink!=0)

     aic.temp<-N*log((ratio)^l)+2*p.lam
     bic.temp<-N*log((ratio)^l)+log(N)*p.lam


     aic.vect<-c(aic.vect, aic.temp)
     bic.vect<-c(bic.vect, bic.temp)

    }

  aic.lam<-lambda.vect[which(aic.vect==min(aic.vect))[1]]
  bic.lam<-lambda.vect[which(bic.vect==min(bic.vect))[1]]
  aic.min<-aic.vect[which(aic.vect==min(aic.vect))[1]]
  bic.min<-bic.vect[which(bic.vect==min(bic.vect))[1]]

  return(list(lambda.vect=lambda.vect,

              aic.vect=aic.vect,
              bic.vect=bic.vect,

              aic.lam=aic.lam,
              bic.lam=bic.lam,

              aic.min=aic.min,
              bic.min=bic.min

             ))
}


#################################################################
#Specify the 20 basis matrices in Study 2 of Zhou and Qu (2012)#
#################################################################

M.case5<-array(0, c(25, 25, 20))

  M1.temp<-diag(c(1,rep(0,3),1))   ## 1 on the (1,1), (m,m), 0 elsewhere
  M2.temp<-matrix(rep(0,5^2),ncol=5)       ## 1 on two main off-diagonals, 0 elsewhere
   for (i in 1:5)
     for (j in 1:5)
       if (abs(i-j)==1) M2.temp[i,j]<-1

  M3.temp<-matrix(rep(1,5^2),ncol=5)-diag(rep(1,5))  ## off-diagonal 1, diagonal 0


M1<-diag(rep(1, 25))
M2<-diag(c(rep(1,5),rep(0,20)))
```

```
M3<-diag(c(rep(0,5),rep(1,5),rep(0,15)))
M4<-diag(c(rep(0,10), rep(1,5), rep(0,10)))
M5<-diag(c(rep(0,15), rep(1,5), rep(0,5)))

M6<-M7<-M8<-matrix(0, 25, 25)
M6[1:5, 1:5]<-M1.temp
M7[1:5, 1:5]<-M2.temp
M8[1:5, 1:5]<-M3.temp

M9<-M10<-M11<-matrix(0, 25, 25)
M9[6:10, 6:10]<-M1.temp
M10[6:10, 6:10]<-M2.temp
M11[6:10, 6:10]<-M3.temp

M12<-M13<-M14<-matrix(0, 25, 25)
M12[11:15, 11:15]<-M1.temp
M13[11:15, 11:15]<-M2.temp
M14[11:15, 11:15]<-M3.temp

M15<-M16<-M17<-matrix(0, 25, 25)
M15[16:20, 16:20]<-M1.temp
M16[16:20, 16:20]<-M2.temp
M17[16:20, 16:20]<-M3.temp

M18<-M19<-M20<-matrix(0, 25, 25)
M18[21:25, 21:25]<-M1.temp
M19[21:25, 21:25]<-M2.temp
M20[21:25, 21:25]<-M3.temp

M.case5[,,1]<-M1
M.case5[,,2]<-M2
M.case5[,,3]<-M3
M.case5[,,4]<-M4
M.case5[,,5]<-M5
M.case5[,,6]<-M6
M.case5[,,7]<-M7
M.case5[,,8]<-M8
M.case5[,,9]<-M9
M.case5[,,10]<-M10
M.case5[,,11]<-M11
M.case5[,,12]<-M12
M.case5[,,13]<-M13
M.case5[,,14]<-M14
M.case5[,,15]<-M15
M.case5[,,16]<-M16
M.case5[,,17]<-M17
M.case5[,,18]<-M18
M.case5[,,19]<-M19
M.case5[,,20]<-M20


##################################################
#Function to generate data sets in Study 2        #
#Study 2 is the "case 5" in the following function#
##################################################

data.generate.unbalance<-function(case, N, m, p, beta.true, pct.missing, prob.missing){

   x<-rmvnorm(N*m, mean=rep(0,p), diag(rep(1,p)))
   p.temp<-0.5+x%*%matrix(beta.true,ncol=1)
   u.temp<-exp(p.temp)/(1+exp(p.temp))
   y<-NULL


  if (case==1)
    {
      cor.str<-matrix(rep(0,m^2),ncol=m)
      for (i in 1:m)
          cor.str[i,]<-0.7^(abs((1:m)-i))    #AR1

      for(i in 1:N)
      {

        mu.temp<-u.temp[((i-1)*m+1):(i*m)]
        y.temp<-ep(mu=mu.temp, R=cor.str, nRep=1)$y
        y<-c(y, as.vector(y.temp))
      }

    }


    else if (case==2)
     {
       cor.str<-matrix(rep(0.6,m^2),ncol=m)+diag(rep(0.4, m))  #CS

      for(i in 1:N)
      {
        mu.temp<-u.temp[((i-1)*m+1):(i*m)]
        y.temp<-ep(mu=mu.temp, R=cor.str, nRep=1)$y
        y<-c(y, as.vector(y.temp))
      }

    }

    else if (case==3)
    {
      cor.str<-matrix(rep(0,m^2), ncol=m)
      cor.str[1:4,1:4]<-matrix(rep(0.8,16),ncol=4)+diag(rep(0.2,4))
      cor.str[5:9,5:9]<-matrix(rep(0.7,25),ncol=5)+diag(rep(0.3,5))

      for(i in 1:N)
      {

        mu.temp<-u.temp[((i-1)*m+1):(i*m)]
        y.temp<-ep(mu=mu.temp, R=cor.str, nRep=1)$y
        y<-c(y, as.vector(y.temp))
      }
```

```
    }

    else if (case==4)
    {
      cor.str<-diag(rep(1,9))
        for (i in 1:8)
          for( j in (i+1):9)
           {
              cor.str[i,j]<-cor.str[j,i]<-(0.8^(j-i)+0.9)/2
           }
      for(i in 1:N)
      {
        mu.temp<-u.temp[((i-1)*m+1):(i*m)]
        y.temp<-ep(mu=mu.temp, R=cor.str, nRep=1)$y
        y<-c(y, as.vector(y.temp))
      }

    }

     else if (case==5)
    {
      cor.str<-diag(rep(1,m))

      cor.str1<-matrix(rep(0,(m/5)^2),ncol=m/5)
       for (i in 1:(m/5))
         cor.str1[i,]<-0.7^(abs((1:(m/5))-i))

      cor.str3<-matrix(rep(0.8,(m/5)^2),ncol=(m/5))+diag(rep(0.2, m/5))

      cor.str[1:5, 1:5]<-cor.str1
      cor.str[11:15, 11:15]<-cor.str3

      for(i in 1:N)
      {
        mu.temp<-u.temp[((i-1)*m+1):(i*m)]
        y.temp<-ep(mu=mu.temp, R=cor.str, nRep=1)$y
        y<-c(y, as.vector(y.temp))
      }

    }

    else print("case number is wrong")

  id<-rep(1:N,each=m)

    miss.indicator<-rbinom(pct.missing*N*m, 1, 1-prob.missing)
    miss.indicator<-c(miss.indicator, rep(1, (N-pct.missing*N)*m))

    gee.fit.full<-geeglm(y~x, family=binomial(link = "logit"),id=id, corstr="independence")


 return(list(case=case, N=N, m=m, p=p, beta.true=beta.true, gee.fit.full=gee.fit.full,
            x=x, y=y,  cor.str=cor.str, id=id, miss.indicator=miss.indicator, pct.missing=pct.missing))
}


 ##################################################################
 #Main function: to select the correlation structure by QIF          #
 #this function select structure for both balanced and unbalanced data#
 ##################################################################

 qif.cor.unbalance<-function(case, x, y, id, miss.indicator, pct.missing=pct.missing, gee.fit.full, lambda.vect,M,r,l, index.gp)
 {

  ##############  for the balanced data #########################
  p<-dim(x)[2]
  N<-length(unique(id))
  m<-dim(x)[1]/N


   beta.est<-gee.fit.full$coefficients
   eta<-cbind(1,x)%*%matrix(beta.est,ncol=1)
   mu<-as.vector(exp(eta)/(1+exp(eta)))
   residue<-matrix((y-mu)/(mu*(1-mu))^0.5, ncol=m, byrow=T)
   R.mat<-cor(residue)

   R.inv<-solve(R.mat)

   A.vect<-(mu*(1-mu))^(-0.5)
   dmu.vect<-exp(eta)/(1+exp(eta))^2

   x.1<-cbind(rep(1,N*m),x)

   scad.y<-NULL
   scad.x<-NULL

   for (i in 1:N)
    {

     A<-diag(A.vect[((i-1)*m+1):(i*m)])
     dmu<-diag(dmu.vect[((i-1)*m+1):(i*m)])

     scad.y<-c(scad.y, t(dmu%*%x.1[((i-1)*m+1):(i*m),])%*%A%*%R.inv%*%A%*%matrix(y[((i-1)*m+1):(i*m)]-mu[((i-1)*m+1):(i*m)],ncol=1))

     temp<-NULL
     for(k in 1:dim(M)[3])
      {
       temp<-cbind(temp, t(dmu%*%x.1[((i-1)*m+1):(i*m),])%*%A%*%M[,,k]%*%A%*%matrix(y[((i-1)*m+1):(i*m)]-mu[((i-1)*m+1):(i*m)],ncol=1))
      }

     scad.x<-rbind(scad.x, temp)

    }

  sd.x<-apply(scad.x, 2, sd)
```

```
sd.y<-sd(scad.y)

scad.x.stand<-sweep(scad.x, 2, sd.x, "/")
scad.y.stand<-scad.y/sd.y


alpha.lm<-lm(scad.y.stand~scad.x.stand-1)$coefficients


  tuning.R<-tuning.selection.R.unbalance(R.mat, scad.y.stand, scad.x.stand, sd.x, sd.y, N, m, index.gp, lambda.vect, r,l,M)

  aglasso.bic.R<-aglasso(scad.y.stand, scad.x.stand, tuning.R$bic.lam, index.gp,r)
  alpha.bic.R.full<-(aglasso.bic.R)$coefs.shrink*(sd.y/sd.x)

  alpha.bic.R.full<-as.vector(alpha.bic.R.full)

  select.index<-NULL
  uniq.id<-unique(index.gp)

  for( j in 1:length(uniq.id))
 {
    location<-which(index.gp==uniq.id[j])
    if (sum(alpha.bic.R.full[location]!=0)>0)
       {select.index<-c(select.index, location)}

  }

  alpha.temp.full<-lm(scad.y.stand~scad.x.stand[,select.index]-1)$coef*(sd.y/sd.x[select.index])
  alpha.update.full<-rep(0, dim(scad.x.stand)[2])
  alpha.update.full[select.index]<-alpha.temp.full


  ############### for the unbalanced data ########
  x.obs<-as.matrix(x[which(miss.indicator==1),])
  y.obs<-y[which(miss.indicator==1)]
  id.obs<-id[which(miss.indicator==1)]

  gee.fit.obs<-geeglm(y.obs~x.obs, family=binomial(link = "logit"),id=id.obs, corstr="independence")


  ind.complete<-(length(id.obs)-(1-pct.missing)*N*m+1):(length(id.obs))
  y.complete<-y.obs[ind.complete]
  x.complete<-x.obs[ind.complete,]
  id.complete<-id.obs[ind.complete]

  beta.est.obs<-gee.fit.obs$coefficients
  eta.complete<-cbind(1,x.complete)%*%matrix(beta.est.obs,ncol=1)
  mu.complete<-as.vector(exp(eta.complete)/(1+exp(eta.complete)))
  residue.complete<-matrix((y.complete-mu.complete)/(mu.complete*(1-mu.complete))^0.5, ncol=m, byrow=T)
  R.mat.complete<-cor(residue.complete)

  R.inv.complete<-solve(R.mat.complete)

 eta.vect.obs<-cbind(1,x)%*%matrix(beta.est.obs,ncol=1)
 mu.vect.obs<-as.vector(exp(eta.vect.obs)/(1+exp(eta.vect.obs)))
 A.vect.obs<-(mu.vect.obs*(1-mu.vect.obs))^(-0.5)
 dmu.vect.obs<-exp(eta.vect.obs)/(1+exp(eta.vect.obs))^2

  scad.y.obs<-NULL
  scad.x.obs<-NULL
  x.1<-cbind(rep(1,dim(x)[1]),x)

  for (i in 1:N)
  {

    id.temp<-id[which(id==i)]
    miss.temp<-miss.indicator[which(id==i)]

    A.obs<-A.vect.obs[((i-1)*m+1):(i*m)]
    A.obs[which(miss.temp==0)]<-0
    A.obs<-diag(A.obs)

    dmu.obs<-dmu.vect.obs[((i-1)*m+1):(i*m)]
    dmu.obs[which(miss.temp==0)]<-0
    dmu.obs<-diag(dmu.obs)

    y.obs<-y[((i-1)*m+1):(i*m)]
    y.obs[which(miss.temp==0)]<-0

    mu.obs<-mu.vect.obs[((i-1)*m+1):(i*m)]
    mu.obs[which(miss.temp==0)]<-0

    scad.y.obs<-c(scad.y.obs, t(dmu.obs%*%x.1[((i-1)*m+1):(i*m),])%*%A.obs%*%R.inv.complete%*%A.obs%*%matrix(y.obs-mu.obs,ncol=1))

    temp<-NULL
    for(k in 1:dim(M)[3])
     {
       temp<-cbind(temp, t(dmu.obs%*%x.1[((i-1)*m+1):(i*m),])%*%A.obs%*%M[,,k]%*%A.obs%*%matrix(y.obs-mu.obs,ncol=1))
     }

    scad.x.obs<-rbind(scad.x.obs, temp)

  }

  sd.x.obs<-apply(scad.x.obs, 2, sd)
  sd.y.obs<-sd(scad.y.obs)

 scad.x.stand.obs<-sweep(scad.x.obs, 2, sd.x.obs, "/")
 scad.y.stand.obs<-scad.y.obs/sd.y.obs

 alpha.lm.obs<-lm(scad.y.stand.obs~scad.x.stand.obs-1)$coefficients

  tuning.R.obs<-tuning.selection.R.unbalance(R.mat.complete, scad.y.stand.obs, scad.x.stand.obs, sd.x.obs, sd.y.obs, N,m, index.gp, lambda.vect, r,l, M)
  aglasso.bic.R.obs<-aglasso(scad.y.stand.obs, scad.x.stand.obs, tuning.R.obs$bic.lam, index.gp,r)
  alpha.bic.R.obs<-(aglasso.bic.R.obs)$coefs.shrink*(sd.y.obs/sd.x.obs)

  alpha.bic.R.obs<-as.vector(alpha.bic.R.obs)
```

```
    select.index.obs<-NULL
    uniq.id<-unique(index.gp)

    for( j in 1:length(uniq.id))
    {
      location<-which(index.gp==uniq.id[j])
      if (sum(alpha.bic.R.obs[location]!=0)>0)
         {select.index.obs<-c(select.index.obs, location)}

    }

    alpha.temp.obs<-lm(scad.y.stand.obs~scad.x.stand.obs[,select.index.obs]-1)$coef*(sd.y.obs/sd.x.obs[select.index.obs])
    alpha.update.obs<-rep(0, dim(scad.x.stand.obs)[2])
    alpha.update.obs[select.index.obs]<-alpha.temp.obs

return(list(case=case, lambda.vect=lambda.vect, r=r, l=l, index.gp=index.gp,
            lam.bic.R.full=tuning.R$bic.lam,

            alpha.bic.R.full=alpha.bic.R.full,
            alpha.lm.full=(alpha.lm)*(sd.y/sd.x),
            alpha.update.full=alpha.update.full,

            lam.bic.R.obs=tuning.R.obs$bic.lam,

            alpha.bic.R.obs=alpha.bic.R.obs,
            alpha.lm.obs=(alpha.lm.obs)*(sd.y.obs/sd.x.obs),
            alpha.update.obs=alpha.update.obs
           ))

}


##############################################################
#Function to run the simulation using the above main function#
##############################################################

sim.qif.unbalance<-function(run, case, N, m, p, beta.true, lambda.vect,M,r,l, index.gp, pct.missing, prob.missing){

      lam.bic.R.full.vect<-NULL

      alpha.lm.full.mat<-NULL

      alpha.bic.R.full.mat<-NULL
      alpha.update.full.mat<-NULL

      lam.bic.R.obs.vect<-NULL

      alpha.lm.obs.mat<-NULL

      alpha.bic.R.obs.mat<-NULL
      alpha.update.obs.mat<-NULL

      for (i in 1:run)
       {
          print(i)
          mydata<-data.generate.unbalance(case,N, m, p, beta.true, pct.missing, prob.missing)
          result<-qif.cor.unbalance(case, mydata$x, mydata$y, mydata$id, mydata$miss.indicator, pct.missing, mydata$gee.fit.full, lambda.vect, M,r,l, index.gp)


          lam.bic.R.full.vect<-c(lam.bic.R.full.vect,result$lam.bic.R.full)

          alpha.lm.full.mat<-rbind(alpha.lm.full.mat,result$alpha.lm.full)
          alpha.bic.R.full.mat<-rbind(alpha.bic.R.full.mat,result$alpha.bic.R.full)
          alpha.update.full.mat<-rbind(alpha.update.full.mat, result$alpha.update.full)

          lam.bic.R.obs.vect<-c(lam.bic.R.obs.vect,result$lam.bic.R.obs)

          alpha.lm.obs.mat<-rbind(alpha.lm.obs.mat, result$alpha.lm.obs)
          alpha.bic.R.obs.mat<-rbind(alpha.bic.R.obs.mat, result$alpha.bic.R.obs)
          alpha.update.obs.mat<-rbind(alpha.update.obs.mat, result$alpha.update.obs)

       }

return(list(case=case, N=N, m=m,p=p,beta.true=beta.true,lambda.vect=lambda.vect,M=M,r=r,l=l, index.gp=index.gp,
                pct.missing=pct.missing, prob.missing=prob.missing,


                lam.bic.R.full.vect=lam.bic.R.full.vect,

                alpha.bic.R.full.mat=alpha.bic.R.full.mat,

                alpha.lm.full.mat=alpha.lm.full.mat,
                alpha.update.full.mat=alpha.update.full.mat,


                lam.bic.R.obs.vect=lam.bic.R.obs.vect,


                alpha.bic.R.obs.mat=alpha.bic.R.obs.mat,
                alpha.update.obs.mat=alpha.update.obs.mat,

                alpha.lm.obs.mat=alpha.lm.obs.mat
           ))
}


#############################################
#Function to summarize the simulation results#
#############################################

get.result<-function(obj){

    r<-obj$r
    l<-obj$l
    N<-obj$N
    m<-obj$m
```

```
p<-obj$p
beta.true<-obj$beta.true
M<-obj$M

grp.id<-obj$index.gp

run<-dim(obj$alpha.bic.R.full.mat)[1]

working.lambda.full<-c(min(obj$lambda.vect), max(obj$lambda.vect))
check.lambda.full<-max(obj$lam.bic.R.full.vect)

null.lm.exact.full<-rep(0,length(unique(grp.id)))
null.bic.R.exact.full<-rep(0,length(unique(grp.id)))

mean.alpha.lm.full<-NULL
mean.alpha.bic.R.full<-NULL
mean.alpha.update.full<-NULL

sd.alpha.lm.full<-NULL
sd.alpha.bic.R.full<-NULL
sd.alpha.update.full<-NULL

working.lambda.obs<-c(min(obj$lambda.vect), max(obj$lambda.vect))
check.lambda.obs<-max(obj$lam.bic.R.obs.vect)

null.lm.exact.obs<-rep(0,length(unique(grp.id)))
null.bic.R.exact.obs<-rep(0,length(unique(grp.id)))

mean.alpha.lm.obs<-NULL
mean.alpha.bic.R.obs<-NULL
mean.alpha.update.obs<-NULL

sd.alpha.lm.obs<-NULL
sd.alpha.bic.R.obs<-NULL
sd.alpha.update.obs<-NULL

uniq.id<-unique(grp.id)

pd.lm.full<-0
pd.bic.full<-0
pd.update.full<-0

pd.lm.obs<-0
pd.bic.obs<-0
pd.update.obs<-0

for(i in 1:run)
{
  for (j in 1:length(uniq.id))
  {
      location<-which(grp.id==uniq.id[j])
      L<-length(location)

      if(sum(abs(obj$alpha.lm.full.mat[i,location])==0)==L)        null.lm.exact.full[j]<-null.lm.exact.full[j]+1

      if(sum(abs(obj$alpha.bic.R.full.mat[i,location])==0)==L)      null.bic.R.exact.full[j]<-null.bic.R.exact.full[j]+1


      if(sum(abs(obj$alpha.lm.obs.mat[i,location])==0)==L)        null.lm.exact.obs[j]<-null.lm.exact.obs[j]+1

      if(sum(abs(obj$alpha.bic.R.obs.mat[i,location])==0)==L)      null.bic.R.exact.obs[j]<-null.bic.R.exact.obs[j]+1
  }

  temp.alpha.lm.full<-obj$alpha.lm.full.mat[i,]
  temp.alpha.bic.full<-obj$alpha.bic.R.full.mat[i,]
  temp.alpha.update.full<-obj$alpha.update.full.mat[i,]

  temp.alpha.lm.obs<-obj$alpha.lm.obs.mat[i,]
  temp.alpha.bic.obs<-obj$alpha.bic.R.obs.mat[i,]
  temp.alpha.update.obs<-obj$alpha.update.obs.mat[i,]

  est.cor.lm.full<-matrix(0, m,m)
  est.cor.bic.full<-matrix(0, m,m)
  est.cor.update.full<-matrix(0, m,m)

  est.cor.lm.obs<-matrix(0, m,m)
  est.cor.bic.obs<-matrix(0, m,m)
  est.cor.update.obs<-matrix(0, m,m)

  M<-obj$M

  for(k in 1:dim(M)[3])
  {
      est.cor.lm.full<-est.cor.lm.full+temp.alpha.lm.full[k]*M[,,k]
      est.cor.bic.full<-est.cor.bic.full+temp.alpha.bic.full[k]*M[,,k]
      est.cor.update.full<-est.cor.update.full+temp.alpha.update.full[k]*M[,,k]

      est.cor.lm.obs<-est.cor.lm.obs+temp.alpha.lm.obs[k]*M[,,k]
      est.cor.bic.obs<-est.cor.bic.obs+temp.alpha.bic.obs[k]*M[,,k]
      est.cor.update.obs<-est.cor.update.obs+temp.alpha.update.obs[k]*M[,,k]

  }

 if(eigen(est.cor.lm.full)$values[m]>0) pd.lm.full<-pd.lm.full+1
 if(eigen(est.cor.bic.full)$values[m]>0) pd.bic.full<-pd.bic.full+1
 if(eigen(est.cor.update.full)$values[m]>0) pd.update.full<-pd.update.full+1


 if(eigen(est.cor.lm.obs)$values[m]>0) pd.lm.obs<-pd.lm.obs+1
 if(eigen(est.cor.bic.obs)$values[m]>0) pd.bic.obs<-pd.bic.obs+1
 if(eigen(est.cor.update.obs)$values[m]>0) pd.update.obs<-pd.update.obs+1
}

mean.alpha.lm.full<-apply(obj$alpha.lm.full.mat,2,mean)
mean.alpha.bic.R.full<-apply(obj$alpha.bic.R.full.mat,2,mean)
mean.alpha.update.full<-apply(obj$alpha.update.full.mat,2,mean)
```

```
    sd.alpha.lm.full<-apply(obj$alpha.lm.full.mat,2,sd)
    sd.alpha.bic.R.full<-apply(obj$alpha.bic.R.full.mat,2,sd)
    sd.alpha.update.full<-apply(obj$alpha.update.full.mat,2,sd)


    table.exact.full<-rbind(null.lm.exact.full,

                            null.bic.R.exact.full
                        )

    rownames(table.exact.full)<-c("lm","bic.R")

    mean.alpha.lm.obs<-apply(obj$alpha.lm.obs.mat,2,mean)
    mean.alpha.bic.R.obs<-apply(obj$alpha.bic.R.obs.mat,2,mean)
    mean.alpha.update.obs<-apply(obj$alpha.update.obs.mat,2,mean)

    sd.alpha.lm.obs<-apply(obj$alpha.lm.obs.mat,2,sd)
    sd.alpha.bic.R.obs<-apply(obj$alpha.bic.R.obs.mat,2,sd)
    sd.alpha.update.obs<-apply(obj$alpha.update.obs.mat,2,sd)

    table.exact.obs<-rbind(null.lm.exact.obs,
                            null.bic.R.exact.obs
                        )

    rownames(table.exact.obs)<-c("lm","bic.R")

    table.pd<-cbind(c(pd.lm.full, pd.bic.full, pd.update.full),
                    c(pd.lm.obs, pd.bic.obs, pd.update.obs))

return(list(N=N,m=m,p=p, beta.true=beta.true, run=run, r=r,l=l,
            working.lambda.full=working.lambda.full, check.lambda.full=check.lambda.full,
            table.exact.full=table.exact.full,

            mean.alpha.full=rbind(mean.alpha.lm.full, mean.alpha.bic.R.full, mean.alpha.update.full),
            sd.alpha.full=rbind(sd.alpha.lm.full,  sd.alpha.bic.R.full, sd.alpha.update.full),

            working.lambda.obs=working.lambda.obs, check.lambda.obs=check.lambda.obs,
            table.exact.obs=table.exact.obs,

            table.pd=table.pd,

            mean.alpha.obs=rbind(mean.alpha.lm.obs,  mean.alpha.bic.R.obs, mean.alpha.update.obs),
            sd.alpha.obs=rbind(sd.alpha.lm.obs,  sd.alpha.bic.R.obs, sd.alpha.update.obs)
        ))
}


lambda.vect<-seq(0.01,2.0,length=200)
beta.true<-c(1,1,1)
l<-0.25

pct.missing<-0.7
prob.missing<-0.5

index.gp.case5<-c(1,1,1,1,1,2,2,3,4,4,5,6,6,7,8,8,9,10,10,11)
p<-3


set.seed(98394054)
sim5.N100.r0.l25<-sim.qif.unbalance(100, 5, 100, 25, p, beta.true,lambda.vect,M.case5, 0,l,index.gp.case5, pct.missing, prob.missing)
set.seed(98394054)
sim5.N200.r0.l25<-sim.qif.unbalance(100, 5, 200, 25, p, beta.true,lambda.vect,M.case5, 0,l,index.gp.case5, pct.missing, prob.missing)
set.seed(98394054)

sim5.N100.r0.l25.summary<-get.result(sim5.N100.r0.l25)
sim5.N200.r0.l25.summary<-get.result(sim5.N200.r0.l25)

sim5.N100.r0.l25.summary
sim5.N200.r0.l25.summary
```