R code for Multiset sparse redundancy analysis

1, Generate three explanatory and one repsonse datasets and run multi-sRDA with 10 fold cross validation

```
# load multi_sRDA functions
sapply(list.files(pattern="[.]R$", path="./Code/multi_sRDA/functions/", full.names=TRUE),
       source)

set.seed(4)

## data generation
N = 50    # number of individuals
k = 4      # number of datasets
m = 2      # number of latent variables (LV's) per dataset
# number of irrelevant variables per dataset
p0=c(1000,500,200,10)
# number of variables associated with the LV's
p1=c(10,8,8,2)

#Function Generate_data
Generate_data <- function(N = 50, k = 4, m = 2,
                                   p0=c(1000,500,200,10),
                                   p1=c(10,8,8,2)){

# input
# size of the data
# N = 50    # number of individuals
# k = 4      # number of datasets
# m = 2      # number of latent variables (LV's) per dataset

# association parameters between the LV's of the k datasets
# e.g. ksi(dataset4) = b1*ksi(dataset3) + b2*ksi(dataset2) + ...
b=array(0,dim=c(m,k,k))
b[1,1,2]=0.8
b[1,2,3]=0.7
b[1,3,4]=0.4
b[2,1,2]=0.6
b[2,2,3]=0.6
b[2,3,4]=0.2


# specify the regression coefficients of the relevant variables per dataset
# on the associated LVs; X = a1*ksi1 + a2*ksi2 + ....
a=array(0,dim=c(max(p1),m,k))
a[1:p1[1],1:2,1]=
  matrix(
    c(.5,0,
      .5,0,
      .5,0,
      .3,0,
      .3,0,
```

```
      .2,0.2,
      0.2,0.3,
      0.2,0.7,
      0.2,0.6,
      0.2,0.6),nrow=p1[1],ncol=m,byrow=TRUE)
a[1:p1[2],1:2,2]=
  matrix(
    c(.5,0,
      .5,0,
      .5,0,
      .4,0,
      0.4,0,
      0.3,0.7,
      0.2,0.6,
      0.2,0.6),nrow=p1[2],ncol=m,byrow=TRUE)
a[1:p1[3],1:2,3]=
  matrix(
    c(.5,0,
      .5,0,
      .5,0,
      .3,0,
      .3,0,
      .2,0.2,
      0.2,0.3,
      0.2,0.6),nrow=p1[3],ncol=m,byrow=TRUE)
a[1:p1[4],1:2,4]=
  matrix(
    c(.5,0,
      0,0.6),nrow=p1[4],ncol=m,byrow=TRUE)


# generate ksi's
ksi=array(NA,dim=c(N,m,k))
for (j in 1:m) {                # loop over the number of LV's
  ksi[,j,1] = rnorm(N,0,1)      # generate values for the LV of the first set of variables
  for (el in 2:k) {              # loop over the other sets of variables
    meanx=0
    sumb2=0
    for (ell in (el-1):1) {            # calculate per person the mean and sd of the LVs
      meanx=meanx+b[j,ell,el]*ksi[,j,ell]
      sumb2=sumb2+b[j,ell,el]
    }
    sdx=max(0.0001,sqrt(1-sumb2))
    ksi[,j,el] = rnorm(N,meanx,sdx)    # sample LV values from the normal distribution
  }
  ksi[,j,]=scale(ksi[,j,])
}

# generate manifest data
X=array(NA,dim=c(N,max(p0+p1),k))
for (el in 1:k) {
  for (j in 1:p0[el]) {         # sample values for the irrelevant manifest variables
    X[,j,el]=rnorm(N,0,1)
```

```
  }           # calculate per person means and sds of the relevant manifest variables
  meanx=ksi[,1:m,el]%*%t(a[1:p1[el],1:m,el])
  suma2=apply(a[,,el],1,sum)
  for (j in (p0[el]+1):(p0[el]+p1[el])) {
    sdx=min(0.0001,sqrt(1-suma2[(j-p0[el])]))
    X[,j,el]=rnorm(N,meanx[,(j-p0[el])],sdx) # sample values for the manifest variables
  }
}

X1=X[,,1]
X2=X[,,2]
X3=X[,,3]
X4=X[,,4]
if (length(which(is.na(apply(X1,2,sd,na.rm=TRUE))))>0)
  {X1=X1[,-which(is.na(apply(X1,2,sd,na.rm=TRUE)))]}
if (length(which(is.na(apply(X2,2,sd,na.rm=TRUE))))>0)
  {X2=X2[,-which(is.na(apply(X2,2,sd,na.rm=TRUE)))]}
if (length(which(is.na(apply(X3,2,sd,na.rm=TRUE))))>0)
  {X3=X3[,-which(is.na(apply(X3,2,sd,na.rm=TRUE)))]}
if (length(which(is.na(apply(X4,2,sd,na.rm=TRUE))))>0)
  {X4=X4[,-which(is.na(apply(X4,2,sd,na.rm=TRUE)))]}

c(dim(X1),dim(X2),dim(X3),dim(X4))

X1=scale(X1)
X2=scale(X2)
X3=scale(X3)
X4=scale(X4)

result <- list(X1,
               X2,
               X3,
               X4
               )

names(result) <- c("X1",
                   "X2",
                   "X3",
                   "X4"
)

result
}

#end of function **********

multiblockdata <- Generate_data(N,
                                k,
                                m,
                                p0,
                                p1)

#X1, X2 and X3 are explanatory sets and X4 is a response dataset
```

```r
  X1 <- multiblockdata$X1
  X2 <- multiblockdata$X2
  X3 <- multiblockdata$X3
  X4 <- multiblockdata$X4

  Data <- cbind(X1,X2,X3,X4)
  EXPL_X = c(0,0,0,0)
  RESP_Y = c(1,0,0,0)
  EXPL_Z = c(0,1,0,0)
  RESP_V = c(0,0,1,0)
  path_matrix = rbind(EXPL_X, RESP_Y,EXPL_Z,RESP_V)

  # blocks of outer model
  blocks = list(1:dim(X1)[2],
                (dim(X1)[2]+1):(dim(X1)[2]+dim(X2)[2]),
                (dim(X1)[2]+dim(X2)[2]+1):(dim(X1)[2]+dim(X2)[2]+dim(X3)[2]),
                (dim(X1)[2]+dim(X2)[2]+dim(X3)[2]+1):
                  (dim(X1)[2]+dim(X2)[2]+dim(X3)[2]+dim(X4)[2]))



modes = c("predictive","predictive", "predictive", "predicted")


time_data <- system.time(
  res_multi_sRDA <- multi_sRDA(Data, path_matrix, blocks, modes,
                               scaled=T, penalization = "ust", nonzero = c(5,10),
                               lambda = Inf, maxiter = 100, cross_validate = T)
)


print.multi_sRDA(res_multi_sRDA)

#latent variable of X1
res_multi_sRDA$scores[,1]

#Non-zero alpha weights associated with X1 and its latent variable
res_multi_sRDA$weights[which(res_multi_sRDA$weights[,2] == "EXPL_X"),]$weight[
  res_multi_sRDA$weights[which(res_multi_sRDA$weights[,2] == "EXPL_X"),]$weight != 0]

#Bete weights of response dataset
res_multi_sRDA$weights[which(res_multi_sRDA$weights[,2] == "RESP_V"),]$weight


# number of iterations
print(res_multi_sRDA$model$iter)

# the selected nonzeros for 10, 5, 5, and 0 for X1, X2, X3 and X4, respectively
# (since the response X4 is not penalized)
print(res_multi_sRDA$nonzero)
```

2, Replicate the simulation study as described in Section 3.2 in the manuscript

```
# load multi_sRDA functions
sapply(list.files(pattern="[.]R$", path="./Code/multi_sRDA/functions/", full.names=TRUE),
       source)

set.seed(4)

## parameters for data generation
N = 100    # number of individuals
k = 4      # number of datasets
m = 2      # number of latent variables (LV's) per dataset
# number of irrelevant variables per dataset
p0=c(1000,500,200,10)
# number of variables associated with the LV's
p1=c(10,8,8,2)


#repeat simulation 100 times#
nr_of_simulations <- 100

sens.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations),ncol = 3)
sens2.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations),ncol = 3)
spec.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations),ncol = 3)
ridge_param <- matrix(c(0,0,0,0),nrow = max(nr_of_simulations),ncol = 4)
lasso_param <- matrix(c(0,0,0,0),nrow = max(nr_of_simulations),ncol = 4)

start_pos <- 1

for (i in start_pos:nr_of_simulations){

  print("nr of simulation")
  print(i)

  # for replication
  nr_seed = runif(1, 10, 10^8)
  set.seed(nr_seed)

  multiblockdata <- multiset_data_generator(N, k,m,
                                            p0,
                                            p1)
  X1 <- multiblockdata$X1
  X2 <- multiblockdata$X2
  X3 <- multiblockdata$X3
  X4 <- multiblockdata$X4

  Data <- cbind(X1,X2,X3,X4)
  EXPL_X = c(0,0,0,0)
  RESP_Y = c(1,0,0,0)
  EXPL_Z = c(0,1,0,0)
  RESP_V = c(0,0,1,0)
  path_matrix = rbind(EXPL_X, RESP_Y,EXPL_Z,RESP_V)
```

```r
# blocks of outer model
blocks = list(1:dim(X1)[2],
              (dim(X1)[2]+1):(dim(X1)[2]+dim(X2)[2]),
              (dim(X1)[2]+dim(X2)[2]+1):(dim(X1)[2]+dim(X2)[2]+dim(X3)[2]),
              (dim(X1)[2]+dim(X2)[2]+dim(X3)[2]+1):
                (dim(X1)[2]+dim(X2)[2]+dim(X3)[2]+dim(X4)[2]))


modes = c("predictive","predictive", "predictive", "predicted")

#if the analysis takes too long, reduce the nonzero grid
time_data <- system.time(
  s_satpls <- multi_sRDA(Data, path_matrix, blocks, modes,
                     scaled=T, penalization = "ust", nonzero =  c(15,10,8,5),
                     lambda = Inf, maxiter = 100, cross_validate = T)
)

#calculate sensitivity and specificity / TPR and TNR

nzero_X_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="EXPL_X",3])>0)
nzero_Y_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="RESP_Y",3])>0)
nzero_Z_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="EXPL_Z",3])>0)

zero_X_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="EXPL_X",3])==0)
zero_Y_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="RESP_Y",3])==0)
zero_Z_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="EXPL_Z",3])==0)

#sensitivity/ TPR and TAVI
sensX = sum(nzero_X_positiong %in% (p0[1]+1):(p1+p0)[1])/min(p1[1],length(nzero_X_positiong))
sensY = sum(nzero_Y_positiong %in% (p0[2]+1):(p1+p0)[2])/min(p1[2],length(nzero_Y_positiong))
sensZ = sum(nzero_Z_positiong %in% (p0[3]+1):(p1+p0)[3])/min(p1[3],length(nzero_Z_positiong))

sens2X = sum(nzero_X_positiong %in% (p0[1]+1):(p1+p0)[1])/p1[1]
sens2Y = sum(nzero_Y_positiong %in% (p0[2]+1):(p1+p0)[2])/p1[2]
sens2Z = sum(nzero_Z_positiong %in% (p0[3]+1):(p1+p0)[3])/p1[3]


#specificiy/TNR
specX = sum(zero_X_positiong %in% 1:p0[1])/p0[1]
specY = sum(zero_Y_positiong %in% 1:p0[2])/p0[2]
specZ = sum(zero_Z_positiong %in% 1:p0[3])/p0[3]


sens.m[i,]  =c(sensX,sensY,sensZ)
sens2.m[i,] =c(sens2X,sens2Y,sens2Z)
spec.m[i,]  =c(specX,specY,specZ)


iter    = s_satpls$model$iter
nonzero = s_satpls$nonzero
lambda  = s_satpls$lambda

#close pdf
```

```
  #dev.off()

  if (i>start_pos){
    print("matrix TAVI:")
    print(sens.m[start_pos:i,])
    print(apply(sens.m[start_pos:i,],2,mean))
    print("matrix TPR:")
    print(sens2.m[start_pos:i,])
    print(apply(sens2.m[start_pos:i,],2,mean))
    print("matrix TNR:")
    print(spec.m[start_pos:i,])
    print(apply(spec.m[start_pos:i,],2,mean))
  }

}

print("Mean TAVI:")
print(apply(sens.m,2,mean))
print("Mean TPR:")
print(apply(sens2.m,2,mean))
print("Mean TNR:")
print(apply(spec.m,2,mean))
```

3, Replicate the simulation study with size of real Marfan data

```
# load multi_sRDA functions
sapply(list.files(pattern="[.]R$", path="./Code/multi_sRDA/functions/", full.names=TRUE),
       source)

set.seed(4)

N = 37    # number of individuals
k = 4      # number of datasets
m = 2      # number of latent variables (LV's) per dataset
# number of irrelevant variables per dataset
p0=c(36000,18000,47,10)
# number of variables associated with the LV's
p1=c(100,100,80,2)


#repeat simulation 100 times####
start_pos <- 1
nr_of_simulations <- 100

sens.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations-start_pos)+1,ncol = 3)
sens2.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations-start_pos)+1,ncol = 3)
spec.m <- matrix(c(0,0,0),nrow = max(nr_of_simulations-start_pos)+1,ncol = 3)
ridge_param <- matrix(c(0,0,0,0),nrow = max(nr_of_simulations-start_pos)+1,ncol = 4)
lasso_param <- matrix(c(0,0,0,0),nrow = max(nr_of_simulations-start_pos)+1,ncol = 4)

for (i in start_pos:nr_of_simulations){
```

```r
print("nr of simulation")
print(i)

# for replication
nr_seed = runif(1, 10, 10^8)
set.seed(nr_seed)

multiblockdata <- multiset_data_generator(N, k,m,
                                          p0,
                                          p1)
X1 <- multiblockdata$X1
X2 <- multiblockdata$X2
X3 <- multiblockdata$X3

Data <- cbind(X1,X2,X3)
EXPL_X = c(0,0,0)
RESP_Y = c(1,0,0)
EXPL_Z = c(1,1,0)
path_matrix = rbind(EXPL_X, RESP_Y,EXPL_Z)

# blocks of outer model
blocks = list(1:dim(X1)[2],
              (dim(X1)[2]+1):(dim(X1)[2]+dim(X2)[2]),
              (dim(X1)[2]+dim(X2)[2]+1):(dim(X1)[2]+dim(X2)[2]+dim(X3)[2]))


modes = c("predictive","predictive", "predicted")

#if the analysis takes too long, reduce the nonzero grid
time_data <- system.time(
  s_satpls <- multi_sRDA(Data, path_matrix, blocks, modes,
                     scaled=T, penalization = "ust", nonzero =  c(150,100,80,50),
                     lambda = Inf, maxiter = 100, cross_validate = T)
)

#calculate sensitivity and specificity / TPR and TNR

nzero_X_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="EXPL_X",3])>0)
nzero_Y_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="RESP_Y",3])>0)
nzero_Z_positiong <- which(abs(s_satpls$weights[s_satpls$weights[,2]=="EXPL_Z",3])>0)

zero_X_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="EXPL_X",3])==0)
zero_Y_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="RESP_Y",3])==0)
zero_Z_positiong <- which((s_satpls$weights[s_satpls$weights[,2]=="EXPL_Z",3])==0)

#sensitivity/ TPR and TAVI
sensX = sum(nzero_X_positiong %in% (p0[1]+1):(p1+p0)[1])/min(p1[1],length(nzero_X_positiong))
sensY = sum(nzero_Y_positiong %in% (p0[2]+1):(p1+p0)[2])/min(p1[2],length(nzero_Y_positiong))
sensZ = sum(nzero_Z_positiong %in% (p0[3]+1):(p1+p0)[3])/min(p1[3],length(nzero_Z_positiong))

sens2X = sum(nzero_X_positiong %in% (p0[1]+1):(p1+p0)[1])/p1[1]
sens2Y = sum(nzero_Y_positiong %in% (p0[2]+1):(p1+p0)[2])/p1[2]
sens2Z = sum(nzero_Z_positiong %in% (p0[3]+1):(p1+p0)[3])/p1[3]
```

```
  #specificiy/TNR
  specX = sum(zero_X_positiong %in% 1:p0[1])/p0[1]
  specY = sum(zero_Y_positiong %in% 1:p0[2])/p0[2]
  specZ = sum(zero_Z_positiong %in% 1:p0[3])/p0[3]


  sens.m[i,]  =c(sensX,sensY,sensZ)
  sens2.m[i,] =c(sens2X,sens2Y,sens2Z)
  spec.m[i,]  =c(specX,specY,specZ)


  iter    = s_satpls$model$iter
  nonzero = s_satpls$nonzero
  lambda  = s_satpls$lambda

  #close pdf
  #dev.off()

  if (i>start_pos){
    print("matrix TAVI:")
    print(sens.m[start_pos:i,])
    print(apply(sens.m[start_pos:i,],2,mean))
    print("matrix TPR:")
    print(sens2.m[start_pos:i,])
    print(apply(sens2.m[start_pos:i,],2,mean))
    print("matrix TNR:")
    print(spec.m[start_pos:i,])
    print(apply(spec.m[start_pos:i,],2,mean))
  }

}

print("Mean TAVI:")
print(apply(sens.m,2,mean))
print("Mean TPR:")
print(apply(sens2.m,2,mean))
print("Mean TNR:")
print(apply(spec.m,2,mean))
```