Simulation results of sparseSCA.R

Introduction

The algorithm of sparseSCA.R .... (to be added)

General conclusions

To be added (once all the simuations are done.)

Situation 1: SimSparse001

2 blocks, 5 components: (note that this is with respect to the P matrix)
0 0 0 1 1
1 1 1 0 0
Furthermore, the distinctive components are sparse.
(note: sparse distinctive component here means some of the loadings in the distictive component are 0's. See the code below.)
The following code is tested under RSCA v0.3.1

```
library(RSCA)
set.seed(112)

I <- 28
J1 <- 144
J2 <-44
Jk <- c(J1, J2)
sumJk <- sum(J1 + J2)
R <- 5


PropNoise <- 0.05
Perc0 <- 0.3

NRSTARTS <- 20
Ndataset <- 50
MAXITER <- 400

LASSO <- .3 #.3
GROUPLASSO <- .1  #.1

Tucker <- array()
ProportionComm <- array()
ProportionDist <- array()
Proportion <- array()

PoutBest <- list()
ToutBest <- list()
TuckerValues <- array()
```

```r
PoutBestPermu <- list()

for (Nd in 1:Ndataset){

  DATA1 <- matrix(rnorm(I*J1, mean = 0, sd = 1), I, J1)
  DATA2 <- matrix(rnorm(I*J2, mean = 0, sd = 1), I, J2)
  DATA <- cbind(DATA1, DATA2)

  svddata <- svd(DATA, R, R)
  Ttrue <- svddata$u
  PTrueC <- as.matrix(svddata$v) %*% diag(svddata$d[1:R])   #note that only the first R eigen values ar

  PTrueCBlock1 <- PTrueC[1:J1,]
  PTrueCBlock2 <- PTrueC[(J1+1):(J1+J2),]

  v1 <- c(1, 2, 3)
  PTrueCBlock1[, v1] <- 0
  v2 <- c(4, 5)
  PTrueCBlock2[, v2] <- 0



  PTrueCBlock1_vec <- as.vector(PTrueCBlock1[, v2])
  v <- sample(1:(J1*2), size = round(Perc0*(J1*2)), replace=F)
  PTrueCBlock1_vec[v] <- 0
  PTrueCBlock1[, v2] <- matrix(PTrueCBlock1_vec, nrow = J1, ncol = 2)

  PTrueCBlock2_vec <- as.vector(PTrueCBlock2[, v1])
  v <- sample(1:(J2*3), size = round(Perc0*(J2*3)), replace=F)
  PTrueCBlock2_vec[v] <- 0
  PTrueCBlock2[, v1] <- matrix(PTrueCBlock2_vec, nrow = J2, ncol = 3)

  PTrueCnew <- rbind(PTrueCBlock1, PTrueCBlock2)

  XTrue <- Ttrue %*% t(PTrueCnew)
  SSXtrue <- sum(XTrue ^ 2)

  Noise <- matrix(rnorm(I*(J1+J2), mean = 0, sd = 1), I, J1+J2)
  SSNoise <- sum(Noise ^ 2)
  g <- sqrt(PropNoise*SSXtrue/(SSNoise-PropNoise*SSNoise))
  NoiseNew <- g*Noise
  SSNoiseNew <- sum(NoiseNew ^ 2)
  Xgenerate <- XTrue + NoiseNew
  SSXgenerate <- sum(Xgenerate ^ 2)
  NoiseVSgenerate <- SSNoiseNew/SSXgenerate

  results <- sparseSCA(Xgenerate, Jk, R, LASSO, GROUPLASSO, NRSTARTS = 50)


  Tout3d <- results$Tmatrix
  PoutBest[[Nd]] <- results$Pmatrix
```

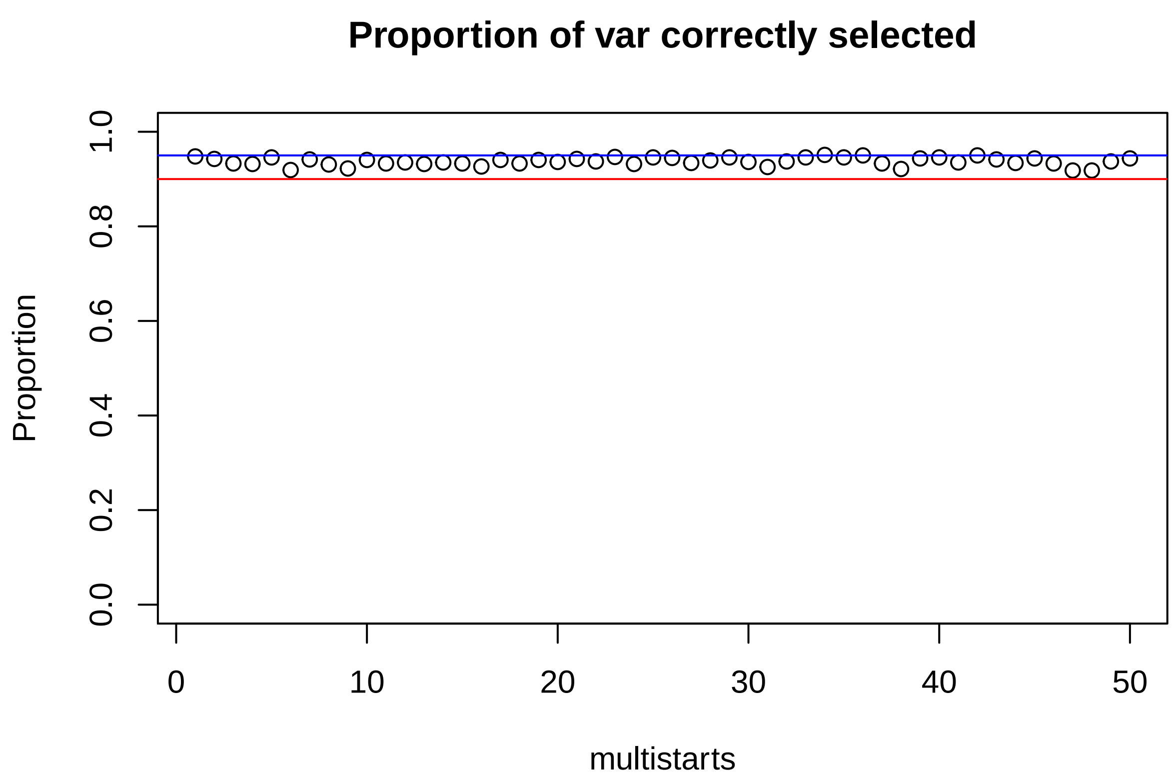
```
  TuckerResults <- TuckerCoef(Ttrue, Tout3d)
  TuckerValues[Nd] <- TuckerResults$tucker_value
  PoutBest[[Nd]] <- PoutBest[[Nd]][, TuckerResults$perm]

  indSelectedC <- which(PoutBest[[Nd]] != 0)
  indDropedC <- which(PoutBest[[Nd]] == 0)
  Proportion[Nd] <- (sum(PTrueCnew[indSelectedC] != 0) + sum(PTrueCnew[indDropedC] == 0))/(sumJk*R)
}

save(Proportion, file = "PropSimSparse001.RData")
save(TuckerValues, file = "TuckerSimSparse001.RData")
```

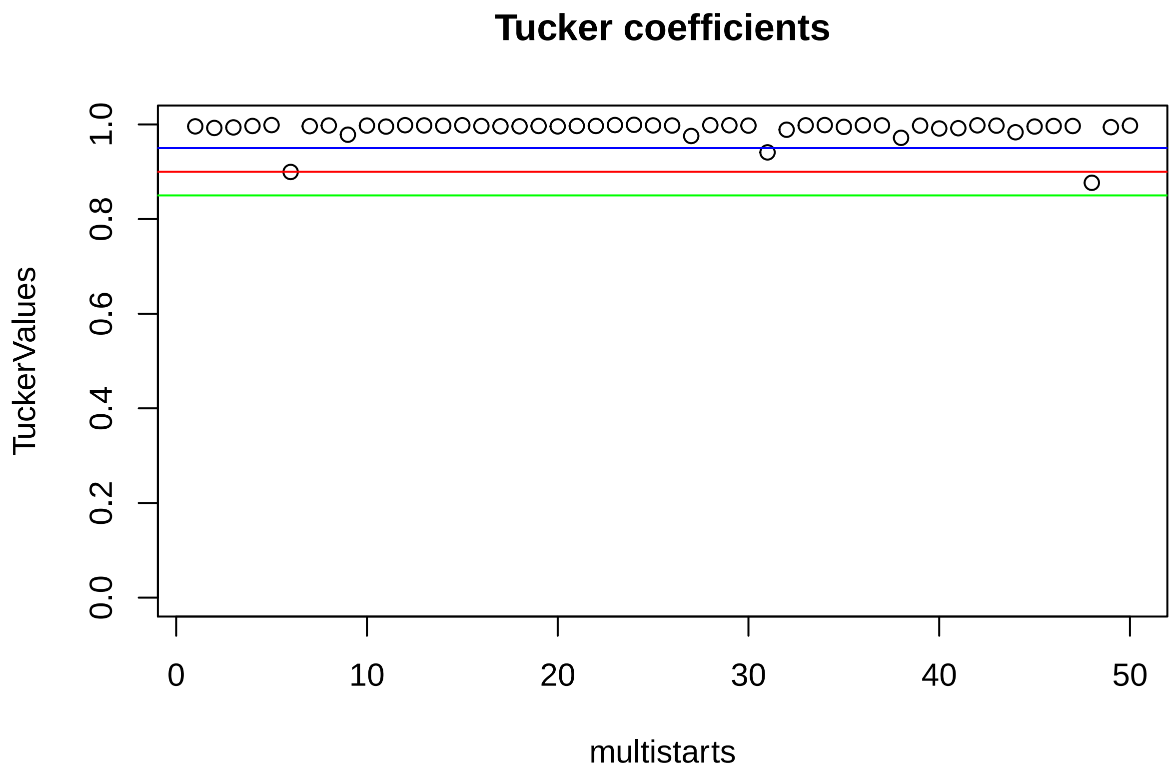
The results are thus saved and plotted (see below).

```
load("PropSimSparse001.RData")
plot(Proportion, ylim=c(0, 1), xlab = "multistarts", main = "Proportion of var correctly selected") +
abline(h = c(.9, .95), col = c("red", "blue"))
```




```
## numeric(0)
```

```
load("TuckerSimSparse001.RData")
plot(TuckerValues, ylim=c(0, 1), xlab = "multistarts", main = "Tucker coefficients") +
abline(h = c(.85, .9, .95), col = c("green", "red", "blue"))
```


Tucker coefficients
TuckerValues
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
multistarts

```
## numeric(0)
```