Demo of binary PCA methods

YipengSong

July 14, 2017

This document is used to show the binary data simulation based on noise corruption and the data analysis using different PCA methods. The interpretation should be refered to the paper.

1. Binary data simulation

1.1 binary data simulation based on a latent variable model

Binary data with underlying low dimensional structure can be simulated from logistic SVD model. The procedure is the same as section "Simulated binary data based on a latent variable model" in the paper.

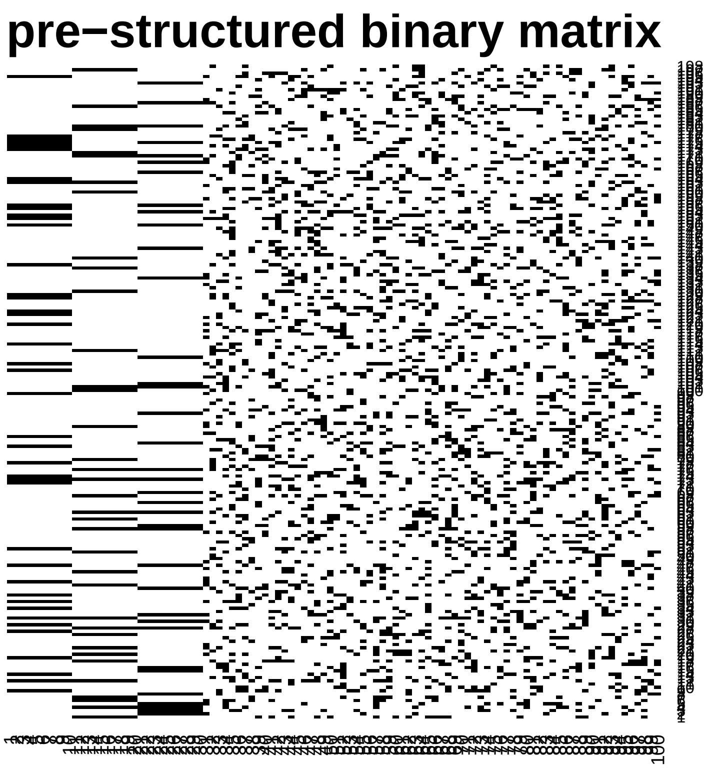
```r
library(MASS)   # for mutivariate gaussian distribution
library(pracma) # for matrix computation

set.seed(123)

## parameters used in the binary simulation
m   = 99; n = 50;         # samples and variables
mu  = matrix(rep(0, n)); # offset term
C   = 10 # constant to adjust the scale of AB^T
R_real = 3  # low rank

## simulation of loading matrix B
B_raw  = matrix(rnorm(n*R_real, sd=1), nrow=n, ncol=R_real)
B_real = gramSchmidt(B_raw)$Q

## simulation of score matrix A
# specify three group means
a1=matrix(c(2,-1,3)); a2=matrix(c(-1,3,-2)); a3=matrix(c(-1,-2,-1))
A1_real = mvrnorm(m/3, mu=a1, Sigma=1*eye(R_real,R_real))
A2_real = mvrnorm(m/3, mu=a2, Sigma=1*eye(R_real,R_real))
A3_real = mvrnorm(m/3, mu=a3, Sigma=1*eye(R_real,R_real))
A_real  = rbind(A1_real,A2_real, A3_real)

## simulation of Theta matrix and probability matrix
Theta      = matrix(rep(1,m))%*%t(mu) + C*A_real%*%t(B_real) # logistic SVD model
Theta_prob = sigmoid(Theta) # transform to 0-1 scale probability

## generating binary data according to corresponding probability
X = matrix(data=0, nrow=m, ncol=n)

# sampling from bernoulli distribution
for(i in 1:m){
  for(j in 1:n){
    X[i,j] = rbinom(1,1,Theta_prob[i,j])
  }
}
```

```
## define class labels
class.lables = vector()
class.lables[1:(m/3)]         = "G1"
class.lables[(m/3+1):(2*m/3)] = "G2"
class.lables[(2*m/3+1):m]     = "G3"
```

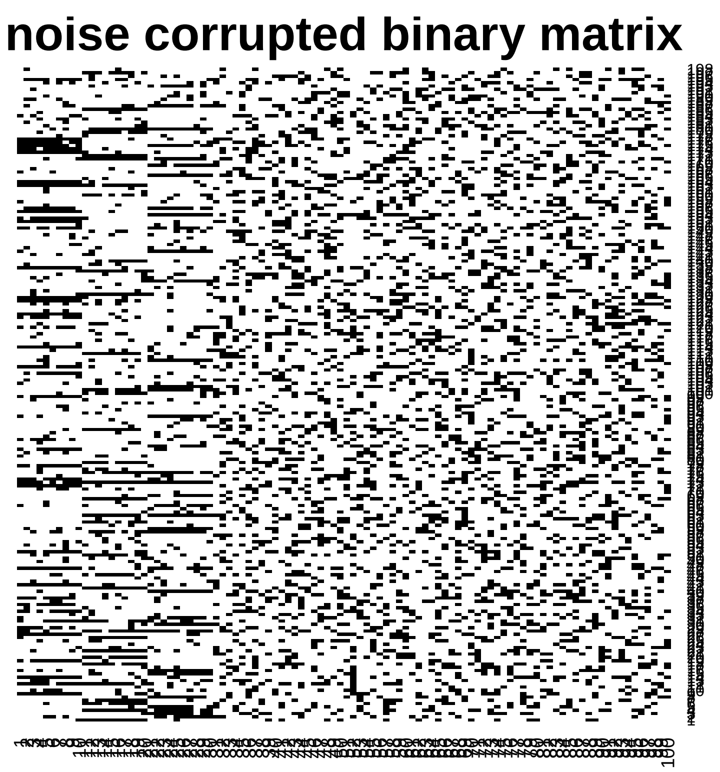
1.2 binary data simulation based on noise corruption of prestructured binary data

The procedure is the same as section “Simulated binary data based on noise corruption of prestructured binary data” in the paper.

```
set.seed(124)

## parameters used during simulation
m = 198; n = 100; R_real = 3  # number of samples, variables and low dimensions
p_str   = 0.2  # the probability for generating structured binary data
p_noise = 0.1  # the probability for fliping structured binary data
ndivd   = 10   # each indepdent variable replicate ndivd times

## simulation of structured binary data
X_str = matrix(data=NA, nrow = m, ncol = R_real*ndivd)

# specify three independent variabels
X_str[,1:ndivd]                = rbinom(m,1,p_str);
X_str[,(ndivd+1):(2*ndivd)]    = rbinom(m,1,p_str);
X_str[,(2*ndivd+1):(3*ndivd)] = rbinom(m,1,p_str);

## simulation unstructured binary data
X_unstr = matrix(data=NA, nrow = m, ncol=(n-R_real*ndivd))
for(i in 1:m){
  for(j in 1:dim(X_unstr)[2]){
    X_unstr[i,j]=rbinom(1,1,p_str)}}

## cancatenate to form pre-structured data
X_pre_struc = cbind(X_str, X_unstr)

# visulize pre-structured data
heatmap(X_pre_struc, scale = "none",
        Colv = NA, Rowv = NA,
        col  = c("white", "black"),
        main ="pre-structured binary matrix")
```



```
## noise corruption pre-strucured binary data
# Bit flipping X_pre_struc
X_corrupted = X_pre_struc
for(i in 1:m){
  for(j in 1:n){
    if(runif(1)<p_noise){
      X_corrupted[i,j]=1-X_pre_struc[i,j]
    }
  }
}

heatmap(X_corrupted, scale = "none",
        Colv = NA, Rowv = NA,
        col  = c("white", "black"),
        main ="noise corrupted binary matrix")
```



```
## noise binary observation is X
X = X_corrupted

## lables for informative features
# define class labels
feature.lables = vector()
feature.lables[1:(n/ndivd)]                  ="v1"
feature.lables[(n/ndivd+1):(2*n/ndivd)]   ="v2"
feature.lables[(2*n/ndivd+1):(3*n/ndivd)]="v3"
feature.lables[(3*n/ndivd+1):n]           ="noise"

## there is no calss labels for samples
# in order to keep consistent with above simulation
class.lables = vector()
class.lables[1:m]         = "G1"
```

2. Visualization of simulated Theta, probability matrix and binary data

Using heatmap to visualize the simulated binary data. Using the score plot to visualize the scores derived from linear PCA on the Theta matrix and probability matrix in above simulation.

```
library(ggplot2)
library(pcaMethods) # for PCA method

## using heatmap to visualize binary data
heatmap(X, scale="none",
```

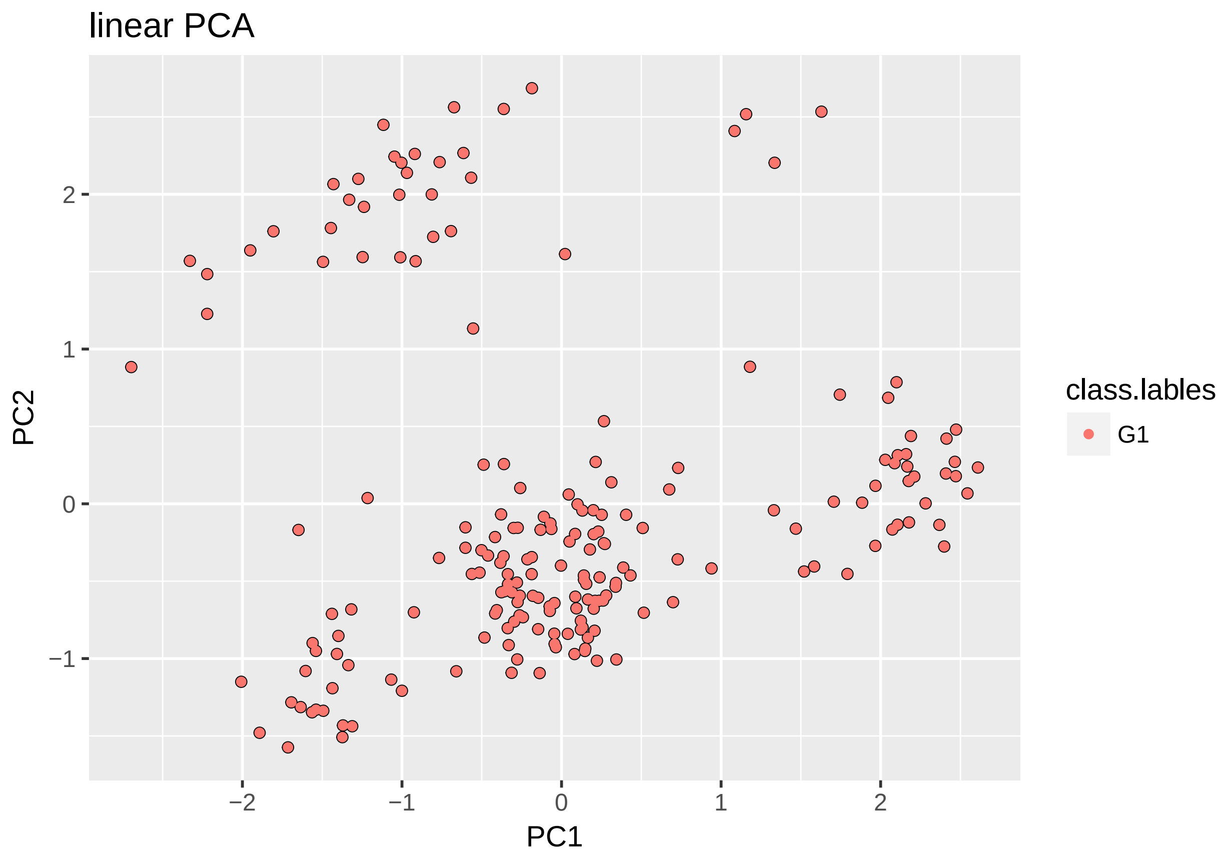
```r
        Colv=NA, Rowv=NA,
        col=c("white", "black"),
        main="binary matrix X")

## using score plot to visualize Theta and probability matrix
K = 2 # specify the number of components

# linear PCA of simulated Theta matrix
pca.Theta = pca(Theta, method="svd", nPcs=K, scale="none", center=TRUE)
pca.scores.Theta = pca.Theta@scores
qplot(pca.scores.Theta[,1], pca.scores.Theta[,2], xlab="PC1", ylab="PC2")+
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("linear PCA of the Theta matrix")

# linear PCA of simulated probability matrix
pca.Theta_prob = pca(Theta_prob, method="svd", nPcs=K, scale="none", center=TRUE)
pca.scores.Theta_prob = pca.Theta_prob@scores
qplot(pca.scores.Theta_prob[,1], pca.scores.Theta_prob[,2], xlab="PC1", ylab="PC2")+
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("linear pca of probability matrix")
```

3. Demo of different PCA methods for binary data

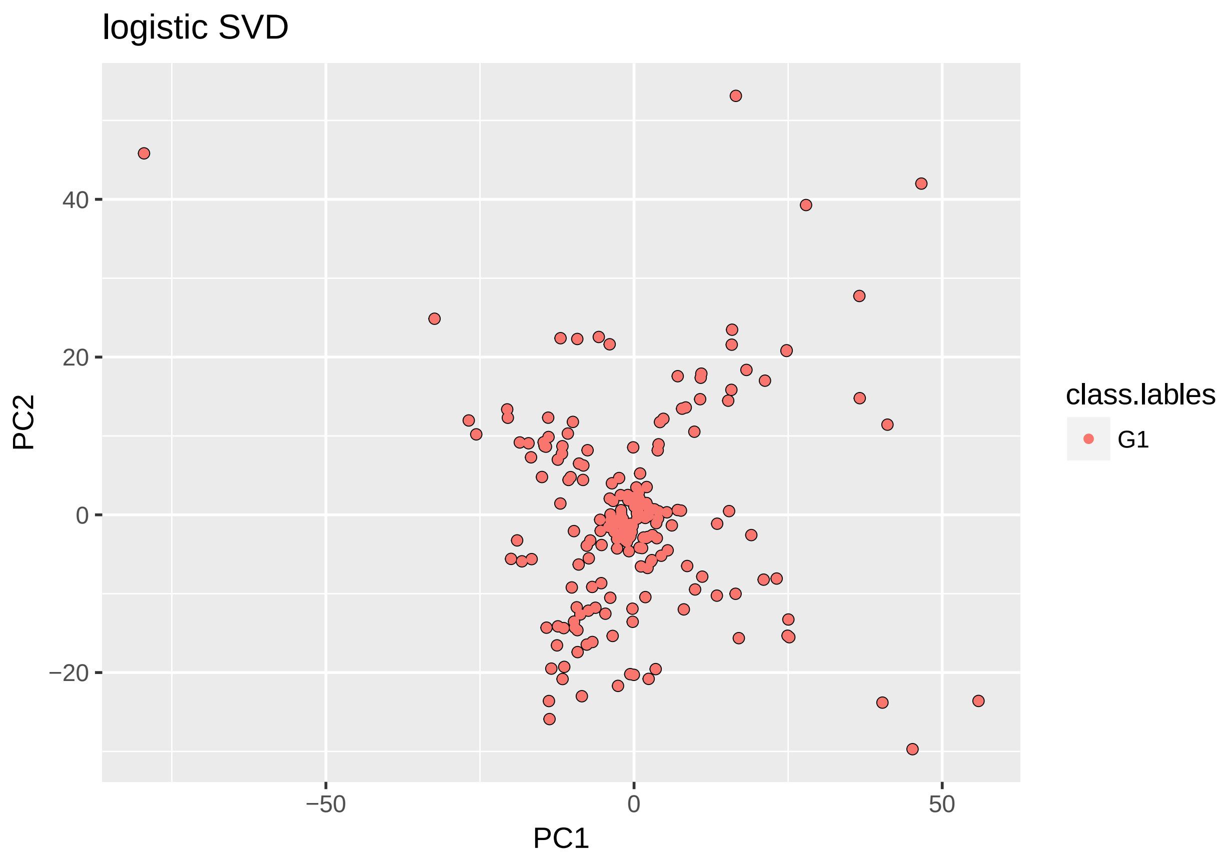
Using different PCA methods to analysis simulated binary data X.

```r
library(homals)      # for gifi method
library(logisticPCA) # for logistic PCA ans logistic SVD
library(pcaMethods)  # for PCA method
library(ggplot2)

K = 2 # specify the number of PCs

## remove the column only has 0 elements
X = X[, colMeans(X)!=0]
m = dim(X)[1]; n = dim(X)[2]

## classicial PCA model
pca.model = pca(X,            # binary data
                method="svd", # using SVD algorithm
                        nPcs=K,       # K PCs
                        scale="none", # no scaling
                        center=TRUE)  # column center
pca.scores   = pca.model@scores   # extract score matrix
pca.loadings = pca.model@loadings # extract loading matrix

## logistic SVD model
logsvd.model = logisticSVD(X,   # binary data
                           k=K, # number of PCs
                           main_effects=TRUE) # including offset term
logsvd.scores   = logsvd.model$A  # extract score matrix
logsvd.loadings = logsvd.model$B  # extract loading matrix

## logistic PCA model
```

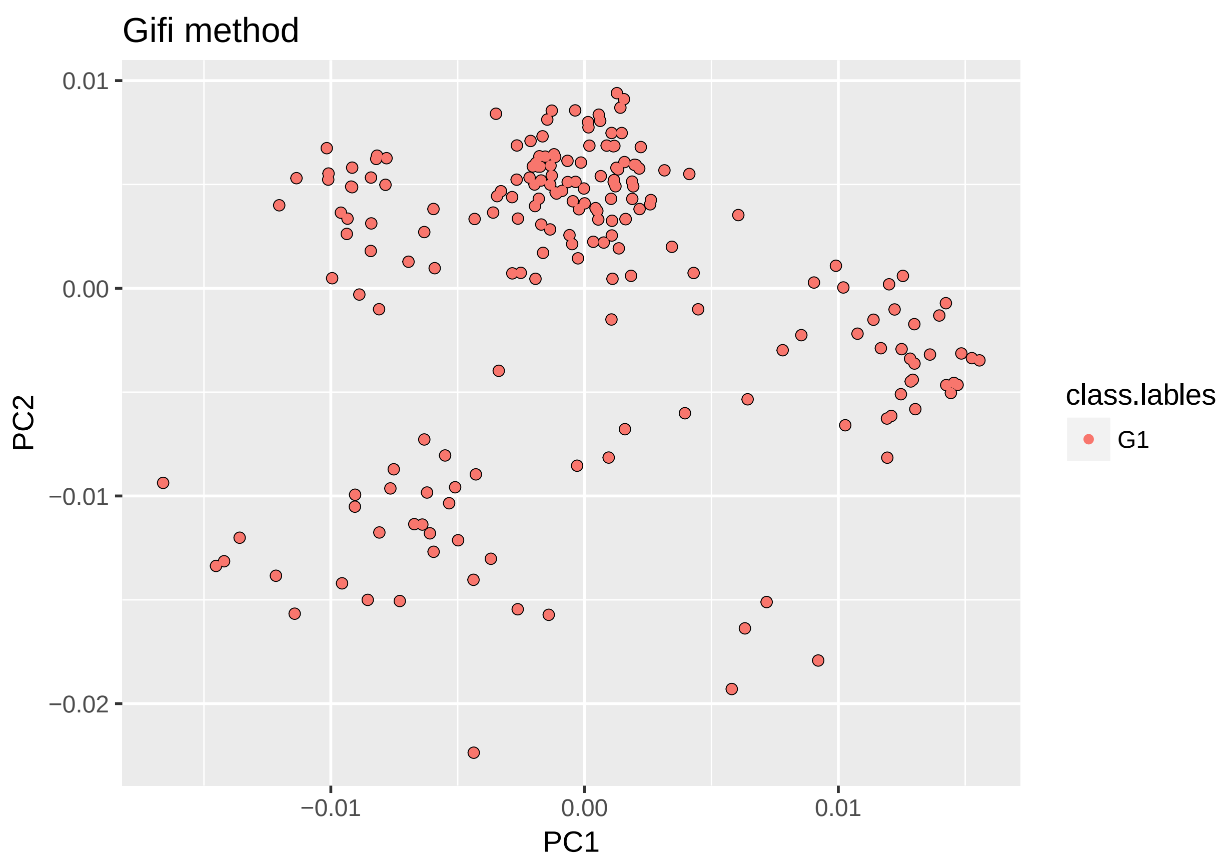
```
logpca.model = logisticPCA(X,       # binary data
                           k=K,     # number of PCs
                           m=2.95, # approximation of natural parameter
                           main_effects = TRUE) # including offset term
logpca.scores   = logpca.model$PCs # extract score matrix
logpca.loadings = logpca.model$U   # extract loading matrix

## Gifi method
gifi.model = homals(as.data.frame(X),  # binary data
                    ndim=K,            # number of PCs
                    rank=1,            # rank constraint for binary PCA soluation
                    level="nominal")   # treat variable as nominal
#plot(gifi.model, plot.type="loadplot")  # loading plot
#plot(gifi.model, plot.type="objplot")   # score plot
gifi.scores = gifi.model$objscores      # extract score matrix
gifi.raw.loadings = gifi.model$loadings # extract raw loading matrix in list form
gifi.loadings=NULL                      # transform to matrix form
for(i in 1:length(gifi.raw.loadings)) gifi.loadings=rbind(gifi.loadings,matrix(gifi.raw.loadings[[i]],n

## visualize above models using score plot
# pca model score plot
qplot(pca.scores[,1], pca.scores[,2], xlab="PC1", ylab="PC2") +
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("linear PCA")
```



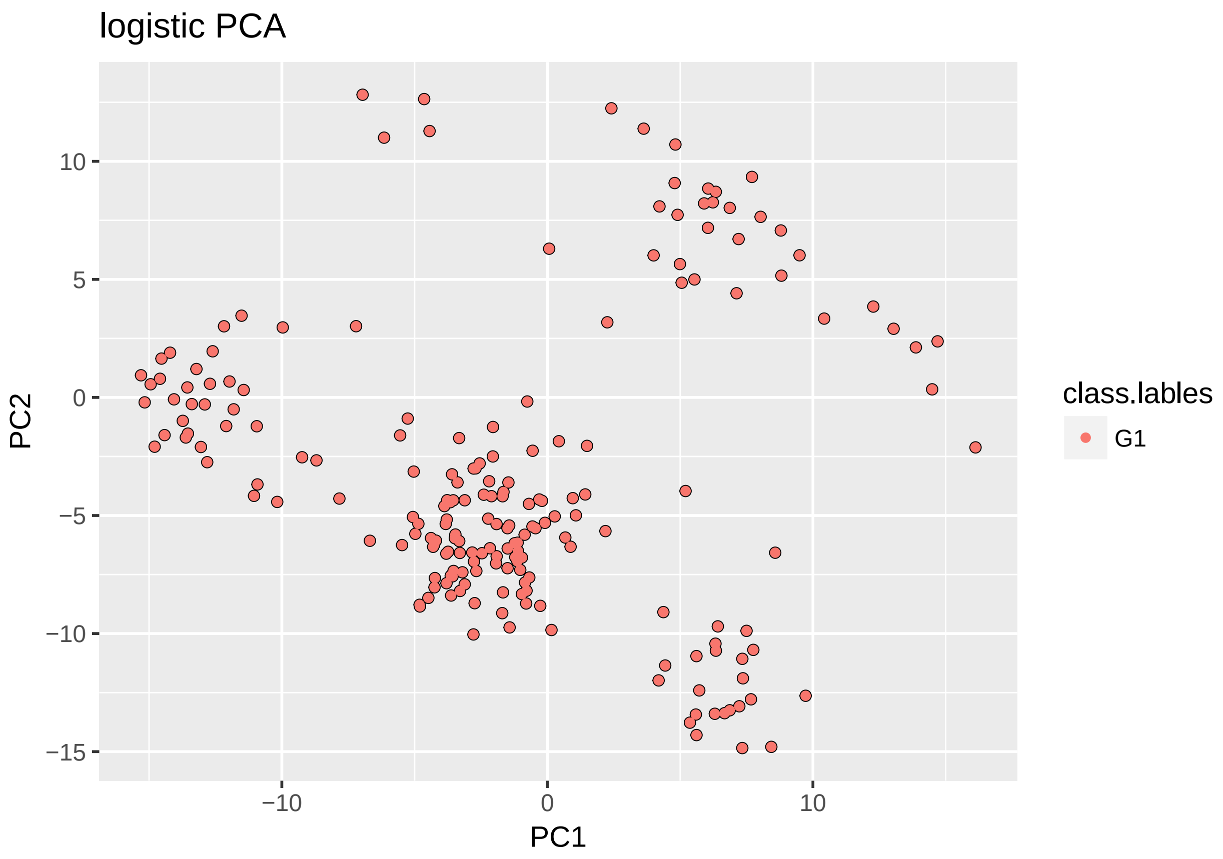
```r
# logistic SVD mdoel score plot
qplot(logsvd.scores[,1], logsvd.scores[,2], xlab="PC1", ylab="PC2") +
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("logistic SVD")
```



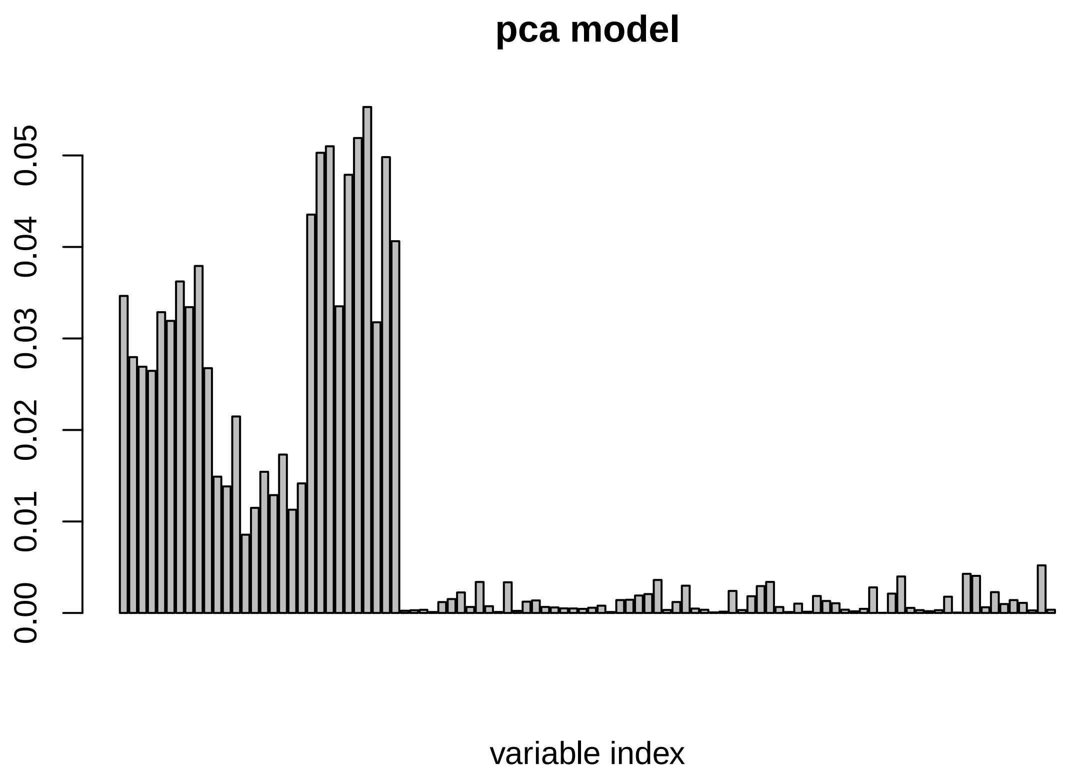
```r
# Gifi model score plot
qplot(gifi.scores[,1], gifi.scores[,2], xlab="PC1", ylab="PC2") +
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("Gifi method")
```



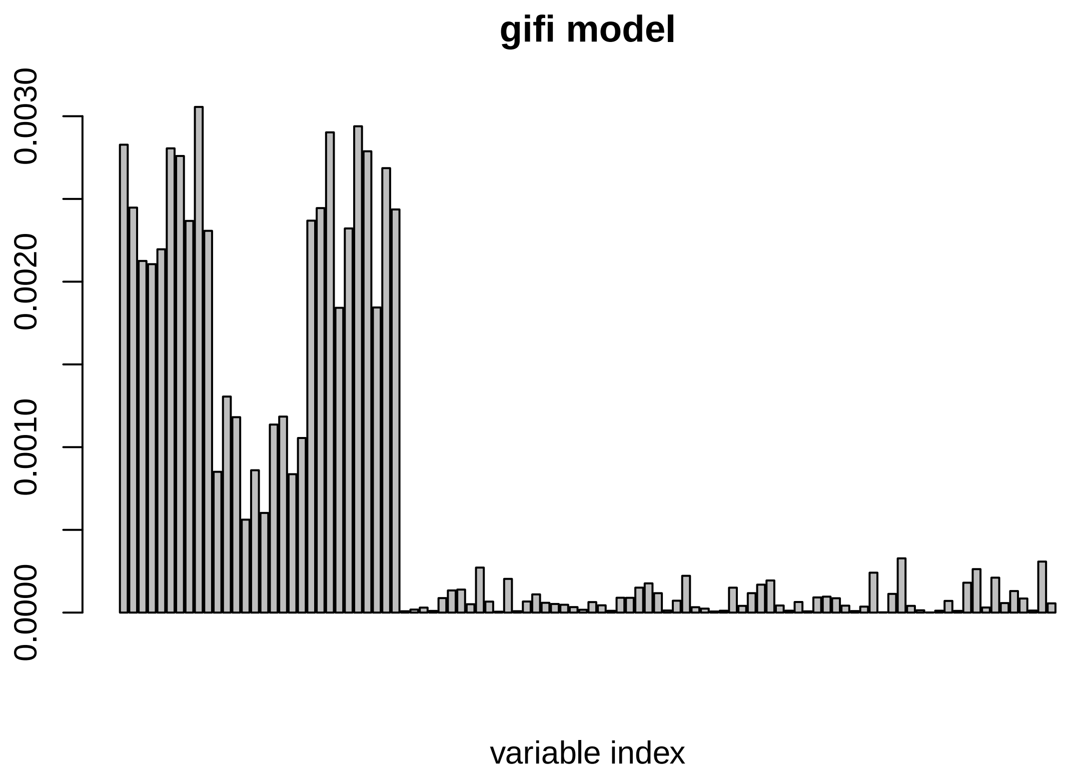
```
# logistic PCA model score plot
qplot(logpca.scores[,1], logpca.scores[,2], xlab="PC1", ylab="PC2") +
  geom_point(aes(shape=class.lables, colour=class.lables), fill="white") +
  ggtitle("logistic PCA")
```



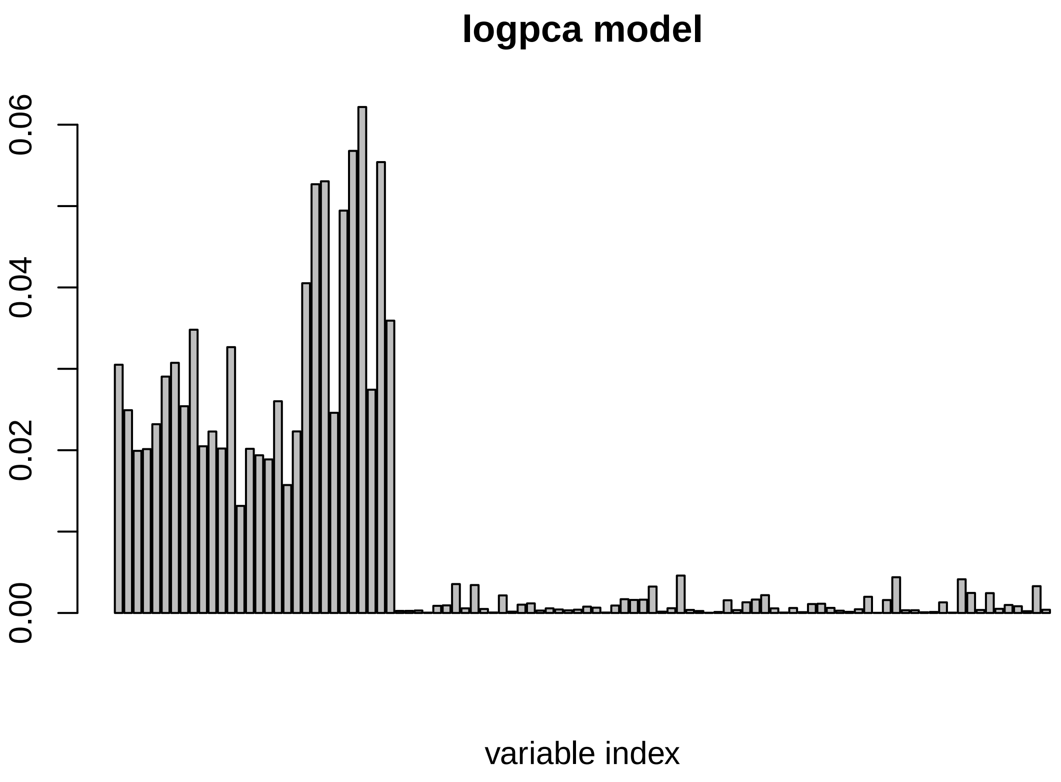
```
## visualize the feature importance as the average square of loadings
barplot(rowMeans((pca.loadings^2)),
        xlab="variable index", main="pca model")
```



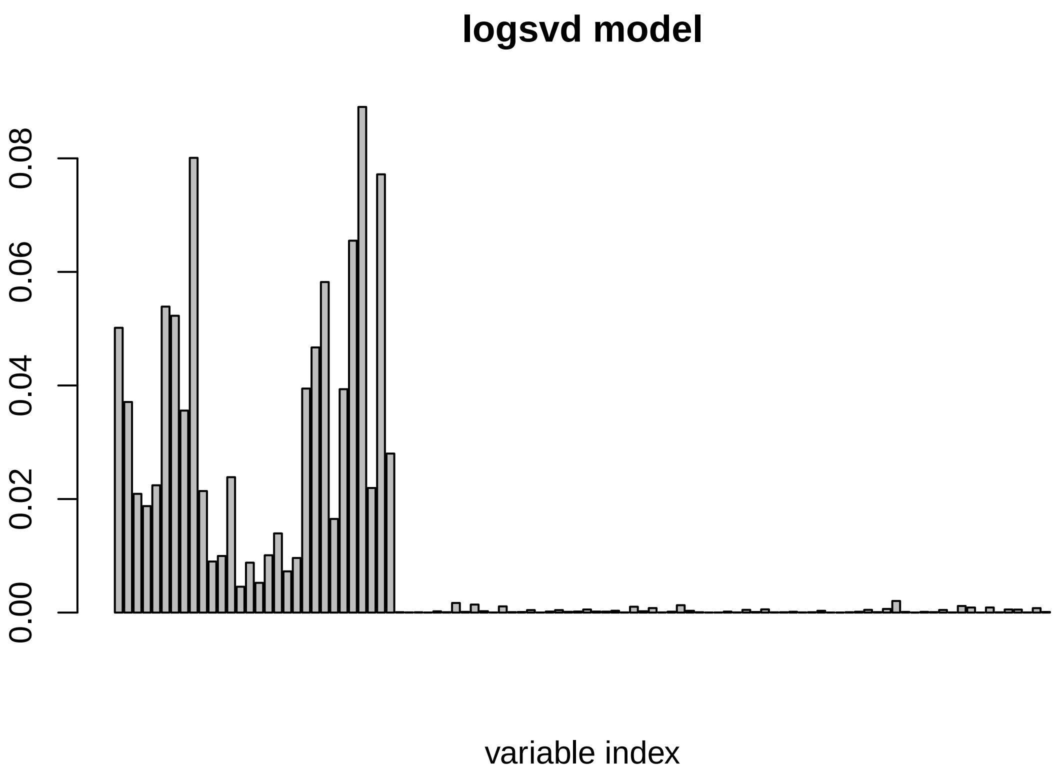
```
barplot(rowMeans((gifi.loadings^2)),
        xlab="variable index", main="gifi model")
```



```
barplot(rowMeans((logpca.loadings^2)),
        xlab="variable index", main="logpca model")
```



```
barplot(rowMeans((logsvd.loadings^2)),
        xlab="variable index", main="logsvd model")
```


logsvd model
0.00
0.02
0.04
0.06
0.08
variable index