Package 'RUVnormalize'

October 9, 2015

Title RUV for normalization of expression array data

Version 1.2.0

Date 2013-11-10

Author Laurent Jacob

Maintainer Laurent Jacob <laurent.jacob@univ-lyon1.fr>

Description RUVnormalize is meant to remove unwanted variation from gene expression data when the factor of interest is not defined, e.g., to clean up a dataset for general use or to do any kind of unsupervised analysis.

License GPL-3

LazyLoad yes

Imports RUVnormalizeData, Biobase

Suggests

Enhances spams

Depends R (>= 2.10.0)

NeedsCompilation no

biocViews StatisticalMethod, Normalization

BuildVignettes true

R topics documented:

clScore *Computes a distance between two partitions of the same data*

Description

The function takes as input two partitions of a dataset into clusters, and returns a number which is small if the two partitions are close, large otherwise.

Usage

```
clScore(c1, c2)
```

Arguments

c1	A vector giving the assignment of the samples to cluster for the first partition
c2	A vector giving the assignment of the samples to cluster for the second partition

Value

A number corresponding to the distance between c1 and c2

Examples

```
if(require('RUVnormalizeData')){

    ## Load the data
    data('gender', package='RUVnormalizeData')

    Y <- t(exprs(gender))
    X <- as.numeric(phenoData(gender)$gender == 'M')
    X <- X - mean(X)
    X <- cbind(X/(sqrt(sum(X^2))))
    chip <- annotation(gender)

    ## Extract regions and labs for plotting purposes
    lregions <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][2])
    llabs <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][3])

    ## Dimension of the factors
    m <- nrow(Y)
    n <- ncol(Y)
    p <- ncol(X)

    Y <- scale(Y, scale=FALSE) # Center gene expressions

    cIdx <- which(featureData(gender)$isNegativeControl) # Negative control genes

    ## Prepare plots
    annot <- cbind(as.character(sign(X)))
```

```
colnames(annot) <- 'gender'
plAnnots <- list('gender'='categorical')
lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
gender.col <- c('-1' = "deeppink3", '1' = "blue")

## Remove platform effect by centering.

Y[chip=='hgu95a.db',] <- scale(Y[chip=='hgu95a.db',], scale=FALSE)
Y[chip=='hgu95av2.db',] <- scale(Y[chip=='hgu95av2.db',], scale=FALSE)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

##--------------------------
## Naive RUV-2 no shrinkage
##--------------------------

k <- 20
nu <- 0

## Correction
nsY <- naiveRandRUV(Y, cIdx, nuCoeff=0, k=k)

## Clustering of the corrected data
sdY <- apply(nsY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmres2ns <- kmeans(nsY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
vclust2ns <- kmres2ns$cluster
nsScore <- clScore(vclust2ns, X)

## Plot of the corrected data
svdRes2ns <- NULL
svdRes2ns <- svdPlot(nsY[, ssd[1:nKeep], drop=FALSE],
                     annot=annot,
                     labels=lab.and.region,
                     svdRes=svdRes2ns,
                     plAnnots=plAnnots,
                     kColors=gender.col, file=NULL)

##--------------------------
## Naive RUV-2 + shrinkage
##--------------------------

k <- m
nu.coeff <- 1e-3

## Correction
nY <- naiveRandRUV(Y, cIdx, nuCoeff=nu.coeff, k=k)

## Clustering of the corrected data
sdY <- apply(nY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmres2 <- kmeans(nY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
```

```
    vclust2 <- kmres2$cluster
    nScore <- clScore(vclust2,X)

    ## Plot of the corrected data
    svdRes2 <- NULL
    svdRes2 <- svdPlot(nY[, ssd[1:nKeep], drop=FALSE],
                       annot=annot,
                       labels=lab.and.region,
                       svdRes=svdRes2,
                       plAnnots=plAnnots,
                       kColors=gender.col, file=NULL)
}
```

`iterativeRUV` *Remove unwanted variation from a gene expression matrix using control genes, optionally replicate samples, and iterative estimates of the factor of interest*

Description

The function takes as input a gene expression matrix as well as the index of negative control genes and replicate samples. It estimates and remove unwanted variation from the gene expression. The major difference with naiveRandRUV and naiveReplicateRUV is that iterativeRUV jointly estimates the factor of interest and the unwanted variation term. It does so iteratively, by estimating each term using the current estimate of the other one.

Usage

```
iterativeRUV(Y, cIdx, scIdx=NULL, paramXb, k, nu.coeff=0, cEps=1e-08, maxIter=30,
  Wmethod="svd", Winit=NULL, wUpdate=maxIter + 1)
```

Arguments

`Y`	Expression matrix where the rows are the samples and the columns are the genes.
`cIdx`	Column index of the negative control genes in Y, for estimation of unwanted variation.
`scIdx`	Matrix giving the set of replicates. Each row is a set of arrays corresponding to replicates of the same sample. The number of columns is the size of the largest set of replicates, and the smaller sets are padded with -1 values. For example if the sets of replicates are (1,11,21), (2,3), (4,5), (6,7,8), the scIdx should be 1 11 21 2 3 -1 4 5 -1 6 7 8
`paramXb`	A `list` containing parameters for the estimation of the term of interest: K corresponds to the rank of X. lambda is the regularization parameter. Large values of lambda lead to sparser, more shrunk estimates of beta. D, batch, iter and mode should not be modified unless you are familiar with sparse dictionary learning algorithms.

k	Desired rank for the estimated unwanted variation term. The returned rank may be lower if the replicate arrays and control genes did not contain a signal of rank k.
nu.coeff	Regularization parameter for the unwanted variation.
cEps	tolerance for relative changes of Wa and Xb estimators at each step. When both get smaller than cEps, the iterations stop.
maxIter	Maximum number of iterations.
Wmethod	'svd' or 'rep', depending whether W is estimated from control genes or replicate samples.
Winit	Optionally provides an initial value for W.
wUpdate	Number of iterations between two updates of W. By default, W is never updated. Make sure that enough iterations are done after the last update of W. E.g, setting W to maxIter will only allow for one iteration of estimating alpha given (Xb, W) and no re-estimation of Xb.

Value

A list containing the following terms:

X, b	if p is not NULL, contains an estimate of the factor of interest (X) and its effect (beta) obtained using rank-p restriction of the SVD of Y - W alpha.
W, a	Estimates of the unwanted variation factors (W) and their effect (alpha).
cY	The corrected expression matrix Y - W alpha.

Examples

```
if(require('RUVnormalizeData') && require('spams')){
    ## Load the spams library
    library(spams)

    ## Load the data
    data('gender', package='RUVnormalizeData')

    Y <- t(exprs(gender))
    X <- as.numeric(phenoData(gender)$gender == 'M')
    X <- X - mean(X)
    X <- cbind(X/(sqrt(sum(X^2))))
    chip <- annotation(gender)

    ## Extract regions and labs for plotting purposes
    lregions <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][2])
    llabs <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][3])

    ## Dimension of the factors
    m <- nrow(Y)
    n <- ncol(Y)
    p <- ncol(X)

    Y <- scale(Y, scale=FALSE) # Center gene expressions
```

```
cIdx <- which(featureData(gender)$isNegativeControl) # Negative control genes

## Prepare plots
annot <- cbind(as.character(sign(X)))
colnames(annot) <- 'gender'
plAnnots <- list('gender'='categorical')
lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
gender.col <- c('-1' = "deeppink3", '1' = "blue")

## Remove platform effect by centering.

Y[chip=='hgu95a.db',] <- scale(Y[chip=='hgu95a.db',], scale=FALSE)
Y[chip=='hgu95av2.db',] <- scale(Y[chip=='hgu95av2.db',], scale=FALSE)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

## Prepare control samples

scIdx <- matrix(-1,84,3)
rny <- rownames(Y)
added <- c()
c <- 0

                                    # Replicates by lab
for(r in 1:(length(rny) - 1)){
    if(r %in% added)
        next
    c <- c+1
    scIdx[c,1] <- r
    cc <- 2
    for(rr in seq(along=rny[(r+1):length(rny)])){
        if(all(strsplit(rny[r],'_')[[1]][-3] ==  strsplit(rny[r+rr],'_')[[1]][-3])){
            scIdx[c,cc] <- r+rr
            cc <- cc+1
            added <- c(added,r+rr)
        }
    }
}
scIdxLab <- scIdx

scIdx <- matrix(-1,84,3)
rny <- rownames(Y)
added <- c()
c <- 0

## Replicates by region
for(r in 1:(length(rny) - 1)){
    if(r %in% added)
        next
    c <- c+1
    scIdx[c,1] <- r
```

```
    cc <- 2
    for(rr in seq(along=rny[(r+1):length(rny)])){
        if(all(strsplit(rny[r],'_')[[1]][-2] ==  strsplit(rny[r+rr],'_')[[1]][-2])){
            scIdx[c,cc] <- r+rr
            cc <- cc+1
            added <- c(added,r+rr)
        }
    }
}
scIdx <- rbind(scIdxLab,scIdx)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

## Prepare plots
annot <- cbind(as.character(sign(X)))
colnames(annot) <- 'gender'
plAnnots <- list('gender'='categorical')
lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
gender.col <- c('-1' = "deeppink3", '1' = "blue")

##---------------------------
## Iterative replicate-based
##---------------------------

cEps <- 1e-6
maxIter <- 30
p <- 20

paramXb <- list()
paramXb$K <- p
paramXb$D <- matrix(c(0.),nrow = 0,ncol=0)
paramXb$batch <- TRUE
paramXb$iter <- 1
paramXb$mode <- 'PENALTY'
paramXb$lambda <- 0.25

## Correction
iRes <- iterativeRUV(Y, cIdx, scIdx, paramXb, k=20, nu.coeff=0,
                     cEps, maxIter,
                     Wmethod='rep', wUpdate=11)

ucY <- iRes$cY

## Cluster the corrected data
sdY <- apply(ucY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmresIter <- kmeans(ucY[,ssd[1:nKeep]],centers=2,nstart=200)
vclustIter <- kmresIter$cluster
IterScore <- clScore(vclustIter,X)

## Plot the corrected data
svdResIter <- NULL
```

```
    svdResIter <- svdPlot(ucY[, ssd[1:nKeep], drop=FALSE],
                          annot=annot,
                          labels=lab.and.region,
                          svdRes=svdResIter,
                          plAnnots=plAnnots,
                          kColors=gender.col, file=NULL)

    ##--------------------------
    ## Iterated ridge
    ##--------------------------

    paramXb <- list()
    paramXb$K <- p
    paramXb$D <- matrix(c(0.),nrow = 0,ncol=0)
    paramXb$batch <- TRUE
    paramXb$iter <- 1
    paramXb$mode <- 'PENALTY' #2
    paramXb$lambda <- 1
    paramXb$lambda2 <- 0

    ## Correction
    iRes <- iterativeRUV(Y, cIdx, scIdx=NULL, paramXb, k=nrow(Y), nu.coeff=1e-2/2,
                         cEps, maxIter,
                         Wmethod='svd', wUpdate=11)

    nrcY <- iRes$cY

    ## Cluster the corrected data
    sdY <- apply(nrcY, 2, sd)
    ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
    kmresIter <- kmeans(nrcY[,ssd[1:nKeep]],centers=2,nstart=200)
    vclustIter <- kmresIter$cluster
    IterRandScore <- clScore(vclustIter,X)

    ## Plot the corrected data
    svdResIterRand <- NULL
    svdResIterRand <- svdPlot(nrcY[, ssd[1:nKeep], drop=FALSE],
                              annot=annot,
                              labels=lab.and.region,
                              svdRes=svdResIterRand,
                              plAnnots=plAnnots,
                              kColors=gender.col, file=NULL)
}
```

---

naiveRandRUV *Remove unwanted variation from a gene expression matrix using negative control genes*

Description

The function takes as input a gene expression matrix as well as the index of negative control genes. It estimates unwanted variation from these control genes, and removes them by regression, using ridge and/or rank regularization.

Usage

```
naiveRandRUV(Y, cIdx, nuCoeff=0.001, k=nrow(Y))
```

Arguments

Y	Expression matrix where the rows are the samples and the columns are the genes.
cIdx	Column index of the negative control genes in Y, for estimation of unwanted variation.
nuCoeff	Regularization parameter for the unwanted variation.
k	Desired rank for the estimated unwanted variation term.

Value

A matrix corresponding to the gene expression after substraction of the estimated unwanted variation term.

Examples

```
if(require('RUVnormalizeData')){

    ## Load the data
    data('gender', package='RUVnormalizeData')

    Y <- t(exprs(gender))
    X <- as.numeric(phenoData(gender)$gender == 'M')
    X <- X - mean(X)
    X <- cbind(X/(sqrt(sum(X^2))))
    chip <- annotation(gender)

    ## Extract regions and labs for plotting purposes
    lregions <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][2])
    llabs <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][3])

    ## Dimension of the factors
    m <- nrow(Y)
    n <- ncol(Y)
    p <- ncol(X)

    Y <- scale(Y, scale=FALSE) # Center gene expressions

    cIdx <- which(featureData(gender)$isNegativeControl) # Negative control genes

    ## Prepare plots
```

```
annot <- cbind(as.character(sign(X)))
colnames(annot) <- 'gender'
plAnnots <- list('gender'='categorical')
lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
gender.col <- c('-1' = "deeppink3", '1' = "blue")

## Remove platform effect by centering.

Y[chip=='hgu95a.db',] <- scale(Y[chip=='hgu95a.db',], scale=FALSE)
Y[chip=='hgu95av2.db',] <- scale(Y[chip=='hgu95av2.db',], scale=FALSE)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

##--------------------------
## Naive RUV-2 no shrinkage
##--------------------------

k <- 20
nu <- 0

## Correction
nsY <- naiveRandRUV(Y, cIdx, nuCoeff=0, k=k)

## Clustering of the corrected data
sdY <- apply(nsY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmres2ns <- kmeans(nsY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
vclust2ns <- kmres2ns$cluster
nsScore <- clScore(vclust2ns, X)

## Plot of the corrected data
svdRes2ns <- NULL
svdRes2ns <- svdPlot(nsY[, ssd[1:nKeep], drop=FALSE],
                     annot=annot,
                     labels=lab.and.region,
                     svdRes=svdRes2ns,
                     plAnnots=plAnnots,
                     kColors=gender.col, file=NULL)

##--------------------------
## Naive RUV-2 + shrinkage
##--------------------------

k <- m
nu.coeff <- 1e-2

## Correction
nY <- naiveRandRUV(Y, cIdx, nuCoeff=nu.coeff, k=k)

## Clustering of the corrected data
sdY <- apply(nY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
```

```
    kmres2 <- kmeans(nY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
    vclust2 <- kmres2$cluster
    nScore <- clScore(vclust2,X)

    ## Plot of the corrected data
    svdRes2 <- NULL
    svdRes2 <- svdPlot(nY[, ssd[1:nKeep], drop=FALSE],
                       annot=annot,
                       labels=lab.and.region,
                       svdRes=svdRes2,
                       plAnnots=plAnnots,
                       kColors=gender.col, file=NULL)
}
```

`naiveReplicateRUV` *Remove unwanted variation from a gene expression matrix using replicate samples*

Description

The function takes as input a gene expression matrix as well as the index of negative control genes and replicate samples. It estimates and remove unwanted variation from the gene expression.

Usage

```
naiveReplicateRUV(Y, cIdx, scIdx, k, rrem=NULL, p=NULL, tol=1e-08)
```

Arguments

`Y`	Expression matrix where the rows are the samples and the columns are the genes.
`cIdx`	Column index of the negative control genes in Y, for estimation of unwanted variation.
`scIdx`	Matrix giving the set of replicates. Each row is a set of arrays corresponding to replicates of the same sample. The number of columns is the size of the largest set of replicates, and the smaller sets are padded with -1 values. For example if the sets of replicates are (1,11,21), (2,3), (4,5), (6,7,8), the scIdx should be 1 11 21 2 3 -1 4 5 -1 6 7 8
`k`	Desired rank for the estimated unwanted variation term. The returned rank may be lower if the replicate arrays and control genes did not contain a signal of rank k.
`rrem`	Optional, indicates which arrays should be removed from the returned result. Useful if the replicate arrays were not actual samples but mixtures of RNA which are only useful to estimate UV but which should not be included in the analysis.
`p`	Optional. If given, the function returns an estimate of the term of interest using rank-p restriction of the SVD of the corrected matrix.

tol	Directions of variance lower than this value in the replicate samples are dropped (which may result in an estimated unwanted variation term of rank smaller than k).

Value

A list containing the following terms:

X, b	if p is not NULL, contains an estimate of the factor of interest (X) and its effect (beta) obtained using rank-p restriction of the SVD of Y - W alpha.
W, a	Estimates of the unwanted variation factors (W) and their effect (alpha).
cY	The corrected expression matrix Y - W alpha
Yctls	The differences of replicate arrays which were used to estimate W and alpha.

Examples

```
if(require('RUVnormalizeData')){

    ## Load the data
    data('gender', package='RUVnormalizeData')

    Y <- t(exprs(gender))
    X <- as.numeric(phenoData(gender)$gender == 'M')
    X <- X - mean(X)
    X <- cbind(X/(sqrt(sum(X^2))))
    chip <- annotation(gender)

    ## Extract regions and labs for plotting purposes
    lregions <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][2])
    llabs <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][3])

    ## Dimension of the factors
    m <- nrow(Y)
    n <- ncol(Y)
    p <- ncol(X)

    Y <- scale(Y, scale=FALSE) # Center gene expressions

    cIdx <- which(featureData(gender)$isNegativeControl) # Negative control genes

    ## Prepare plots
    annot <- cbind(as.character(sign(X)))
    colnames(annot) <- 'gender'
    plAnnots <- list('gender'='categorical')
   lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
    gender.col <- c('-1' = "deeppink3", '1' = "blue")

    ## Remove platform effect by centering.

    Y[chip=='hgu95a.db',] <- scale(Y[chip=='hgu95a.db',], scale=FALSE)
    Y[chip=='hgu95av2.db',] <- scale(Y[chip=='hgu95av2.db',], scale=FALSE)
```

```
## Prepare control samples

scIdx <- matrix(-1,84,3)
rny <- rownames(Y)
added <- c()
c <- 0

                                    # Replicates by lab
for(r in 1:(length(rny) - 1)){
    if(r %in% added)
        next
    c <- c+1
    scIdx[c,1] <- r
    cc <- 2
    for(rr in seq(along=rny[(r+1):length(rny)])){
        if(all(strsplit(rny[r],'_')[[1]][-3] ==  strsplit(rny[r+rr],'_')[[1]][-3])){
            scIdx[c,cc] <- r+rr
            cc <- cc+1
            added <- c(added,r+rr)
        }
    }
}
scIdxLab <- scIdx

scIdx <- matrix(-1,84,3)
rny <- rownames(Y)
added <- c()
c <- 0

## Replicates by region
for(r in 1:(length(rny) - 1)){
    if(r %in% added)
        next
    c <- c+1
    scIdx[c,1] <- r
    cc <- 2
    for(rr in seq(along=rny[(r+1):length(rny)])){
        if(all(strsplit(rny[r],'_')[[1]][-2] ==  strsplit(rny[r+rr],'_')[[1]][-2])){
            scIdx[c,cc] <- r+rr
            cc <- cc+1
            added <- c(added,r+rr)
        }
    }
}
scIdx <- rbind(scIdxLab,scIdx)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

## Prepare plots
annot <- cbind(as.character(sign(X)))
colnames(annot) <- 'gender'
```

```
 plAnnots <- list('gender'='categorical')
lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
 gender.col <- c('-1' = "deeppink3", '1' = "blue")

 ## Remove platform effect by centering.

 ## Correction
 sRes <- naiveReplicateRUV(Y, cIdx, scIdx, k=20)

 ## Clustering on the corrected data
 sdY <- apply(sRes$cY, 2, sd)
 ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
 kmresRep <- kmeans(sRes$cY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
 vclustRep <- kmresRep$cluster
 RepScore <- clScore(vclustRep,X)

 ## Plot of the corrected data
 svdResRep <- NULL
 svdResRep <- svdPlot(sRes$cY[, ssd[1:nKeep], drop=FALSE],
                      annot=annot,
                      labels=lab.and.region,
                      svdRes=svdResRep,
                      plAnnots=plAnnots,
                      kColors=gender.col, file=NULL)
}
```

---

svdPlot *Plot the data projected into the space spanned by their first two principal components*

Description

The function takes as input a gene expression matrix and plots the data projected into the space spanned by their first two principal components.

Usage

```
svdPlot(Y, annot=NULL, labels=NULL, svdRes=NULL, plAnnots=NULL, kColors=NULL, file=NULL)
```

Arguments

Y	Expression matrix where the rows are the samples and the columns are the genes.
annot	A matrix containing the annotation to be plotted. Each row must correspond to a sample (row) of argument Y, each column must be a categorical or continuous descriptor for the sample. Optional.
labels	A vector with one element per sample (row) of argument Y. If this argument is specified, each sample is represented by its label. Otherwise, it is represented by a dot (if no annotation is provided) or by the value of the annotation. Optional.

svdRes	A list containing the result of svd(Y), possibly restricted to the first few singular values. Optional: if not provided, the function computes the SVD.
plAnnots	A list specifiying whether each column of the annot argument corresponds to a categorical or continuous factor. Each element of the list is named after a column of annot, and contains a string 'categorical' or 'continuous'. For each element of this list, a plot is produced where the samples are represented by colors corresponding to their annotation. Optional.
kColors	A vector of colors to be used to represent categorical factors. Optional: a default value is provided. If a categorical factors has more levels than the number of colors provided, colors are not used and the factor is represented in black.
file	A string giving the path to a pdf file for the plot. Optional.

Value

A list containing the result of svd(Y, nu=2, nv=0).

Examples

```
if(require('RUVnormalizeData')){

    ## Load the data
    data('gender', package='RUVnormalizeData')

    Y <- t(exprs(gender))
    X <- as.numeric(phenoData(gender)$gender == 'M')
    X <- X - mean(X)
    X <- cbind(X/(sqrt(sum(X^2))))
    chip <- annotation(gender)

    ## Extract regions and labs for plotting purposes
    lregions <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][2])
    llabs <- sapply(rownames(Y),FUN=function(s) strsplit(s,'_')[[1]][3])

    ## Dimension of the factors
    m <- nrow(Y)
    n <- ncol(Y)
    p <- ncol(X)

    Y <- scale(Y, scale=FALSE) # Center gene expressions

    cIdx <- which(featureData(gender)$isNegativeControl) # Negative control genes

    ## Prepare plots
    annot <- cbind(as.character(sign(X)))
    colnames(annot) <- 'gender'
    plAnnots <- list('gender'='categorical')
   lab.and.region <- apply(rbind(lregions, llabs),2,FUN=function(v) paste(v,collapse='_'))
    gender.col <- c('-1' = "deeppink3", '1' = "blue")

    ## Remove platform effect by centering.
```

```
Y[chip=='hgu95a.db',] <- scale(Y[chip=='hgu95a.db',], scale=FALSE)
Y[chip=='hgu95av2.db',] <- scale(Y[chip=='hgu95av2.db',], scale=FALSE)

## Number of genes kept for clustering, based on their variance
nKeep <- 1260

##--------------------------
## Naive RUV-2 no shrinkage
##--------------------------

k <- 20
nu <- 0

## Correction
nsY <- naiveRandRUV(Y, cIdx, nuCoeff=0, k=k)

## Clustering of the corrected data
sdY <- apply(nsY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmres2ns <- kmeans(nsY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
vclust2ns <- kmres2ns$cluster
nsScore <- clScore(vclust2ns, X)

## Plot of the corrected data
svdRes2ns <- NULL
svdRes2ns <- svdPlot(nsY[, ssd[1:nKeep], drop=FALSE],
                     annot=annot,
                     labels=lab.and.region,
                     svdRes=svdRes2ns,
                     plAnnots=plAnnots,
                     kColors=gender.col, file=NULL)

##--------------------------
## Naive RUV-2 + shrinkage
##--------------------------

k <- m
nu.coeff <- 1e-2

## Correction
nY <- naiveRandRUV(Y, cIdx, nuCoeff=nu.coeff, k=k)

## Clustering of the corrected data
sdY <- apply(nY, 2, sd)
ssd <- sort(sdY,decreasing=TRUE,index.return=TRUE)$ix
kmres2 <- kmeans(nY[,ssd[1:nKeep],drop=FALSE],centers=2,nstart=200)
vclust2 <- kmres2$cluster
nScore <- clScore(vclust2,X)

## Plot of the corrected data
svdRes2 <- NULL
svdRes2 <- svdPlot(nY[, ssd[1:nKeep], drop=FALSE],
                   annot=annot,
```

```
                        labels=lab.and.region,
                        svdRes=svdRes2,
                        plAnnots=plAnnots,
                        kColors=gender.col, file=NULL)
    }
```

Index