Package 'CBM'

March 7, 2017

Type Package

Title Cross-platform Bayesian model of RNA-seq and microarray

Version 2.0

Date 2017-02-04

Author Tianzhou Ma, Fang Zhou

Maintainer Tianzhou Ma <tianzhou.ma0105@gmail.com>

Description This packages performs a joint Bayesian modeling for integrating microarray and RNA-seq transcriptomic data.

License GPL (>= 2)

Imports Rcpp (>= 0.12.9), RcppArmadillo, RcppGSL, BH, BayesLogit, snowfall, cvTools

LinkingTo Rcpp, RcppArmadillo, RcppGSL, BH, BayesLogit

RoxygenNote 6.0.1

R topics documented:

GetBayesianQ 1
NormDiff 2
parMCMC 4
RealSubset 5
RunMCMC 5
SimData 6

Index 7

`GetBayesianQ` *Function to get the bayesian q-value*

Description

Function to get the bayesian q-value The `GetBayesianQ` is a function to get the bayesian q-value

Usage

```
GetBayesianQ(Delta, G, K, burnin)
```

Arguments

Delta	is a matrix outputted from the MCMC
G	is the number of matched genes
K	is the number of studies
burnin	is the number of iterations in burnin period (i.e.those chains you wish to discard)

Value

a vector of length G of Bayesian q-values

Examples

```
## Not run:
delta <- MCMC.out[["Delta"]]
q_real <- GetBayesianQ(Delta = delta,G=G,K=K,burnin=3000)

## End(Not run)
```

NormDiff *Function to calculate the normalization factors*

Description

Function to calculate the normalization factors The NormDiff is a function to calculate the normalization factors

Usage

```
NormDiff(test.es, ref.es, cutoff)
```

Arguments

test.es	is a matrix of logFC from the test data (i.e. other than the reference data)
ref.es	is a vector of logFC from the reference data
cutoff	is a logFC threshold to select the gene set for calculating the normalization factors

Value

a vector of length K-1 of normalization factors for the test data, the normalization factor for the reference data is just zero.

Examples

```
## Not run:
data(RealSubset)
G <- nrow(Data.list[[1]])
K <- 4
index_seq <- 1
index_array <- 2:4
count <- Data.list[index_seq]
intensity <- Data.list[index_array]
X_seq <- X.list[index_seq]
X_array <- X.list[index_array]
es_seq=function (x,lib,k) {
  x=array(x)
  v1=log((x[which(X_seq[[k]]==1)]+1)/(lib[which(X_seq[[k]]==1)]))
  v2=log((x[which(X_seq[[k]]==0)]+1)/(lib[which(X_seq[[k]]==0)]))
  u=mean(v1) - mean(v2)
  return(u)
}
beta.obs<-matrix(,nrow=G,ncol=length(index_seq))
for (k in 1:length(index_seq)){
 lib<-colSums(count[[k]])
 for (i in 1:G){
   beta.obs[i,k]<-es_seq(x=count[[k]][i,],lib=lib,k=k)
 }
}
es_array=function (x,k) {
 x <- as.numeric(x)
 v1 <- x[which(X_array[[k]]==1)]
 v2 <- x[which(X_array[[k]]==0)]
 u=mean(v1)- mean(v2)
 return(u)
}
b.obs<-matrix(,nrow=G,ncol=length(index_array))
for (k in 1:length(index_array)){
 for (i in 1:G){
   b.obs[i,k]<-es_array(x=intensity[[k]][i,],k=k)
 }
}
es.obs <- cbind(beta.obs,b.obs)
rownames(es.obs) <- rownames(Data.list[[1]])
up.index <- which(es.obs[,1]>0  &  es.obs[,2]>0 & es.obs[,3]>0 &
es.obs[,4]>0)
down.index <- which(es.obs[,1]<0 &  es.obs[,2]<0 &  es.obs[,3]<0 &
es.obs[,4]<0)
test.es <- abs(es.obs[c(up.index,down.index),2:4])
ref.es <- abs(es.obs[c(up.index,down.index),1])
cutoff <- 0.2
norm.factor <- NormDiff(test.es=test.es,ref.es=ref.es,cutoff=cutoff)
## Further make into a matrix to input into MCMC
norm.mat <- matrix(0,nrow=G,ncol=K)
norm.mat[which(es.obs[,2]>0),2] <- -norm.factor[1]
norm.mat[which(es.obs[,3]>0),3] <- -norm.factor[2]
norm.mat[which(es.obs[,4]>0),4] <- -norm.factor[3]
norm.mat[which(es.obs[,2]<0),2] <- norm.factor[1]
norm.mat[which(es.obs[,3]<0),3] <- norm.factor[2]
norm.mat[which(es.obs[,4]<0),4] <- norm.factor[3]
```

```
## End(Not run)
```

---

parMCMC *Function to run MCMC chain*

Description

Function to run parallel MCMC chain The parMCMC is a function to run the MCMC chain

Usage

```
parMCMC(Data.list, X.list, norm.mat, index.seq, index.array, iteration, chunks,
  seed = 15213)
```

Arguments

Data.list	is a list of K elements, where K is the number of studies, each element is a microarray or RNAseq expression matrix with G rows and N columns, where G is number of matched genes and N is the sample size.
X.list	is a list of K elements, each element includes is a phenotypic condition of the corresponding samples, with case=1, control=0.
norm.mat	is a matrix of normalization factors, with K columns and G rows.
index.seq	index for RNA-seq studies
index.array	index for microarray studies
iteration	is the number of MCMC chains wish to run
chunks	is the number of cpu's called
seed	is a initial seed for random number generator.

Value

a list of MCMC output matrices for three key parameters of interest: the DE indicator "Delta", the study-specific effect size "ES", and the grand mean effect size "Lambda". For details, please refer to the paper "A joint Bayesian modeling for integrating microarray and RNA-seq transcriptomic data"

Examples

```
## Not run:
data(SimData)
G <- nrow(Data.list[[1]])
adjust.seq1 <-  adjust.seq2 <- adjust.array1 <-  adjust.array2 <- rep(0,G)
## For simulation, we already know the normalization factor:
adjust.array1[1:200] <- adjust.array2[1:200] <- -0.25
adjust.array1[201:400] <- adjust.array2[201:400] <- 0.25
norm.mat <- cbind(adjust.seq1,adjust.seq2,adjust.array1,
adjust.array2)
index.seq <- 1:2
index.array <- 3:4
iteration <- 2000
MCMC.out <- parMCMC(Data.list, X.list, norm.mat, index.seq,
```

```
                    index.array, iteration, chunks=2)

## End(Not run)
```

RealSubset *Test data*

Description

Subset of 2000 genes of 4 studies from ILC PR used in the paper; includes a list of dataframes "Data.list" and a list of phenotypic conditions "X.list"

Usage

```
data("RealSubset")
```

Examples

```
data(RealSubset)
```

RunMCMC *Function to run MCMC chain*

Description

Function to run MCMC chain The RunMCMC is a function to run the MCMC chain

Usage

```
RunMCMC(Data.list, X.list, norm.mat, index.seq, index.array, iteration,
  seed = 15213)
```

Arguments

Data.list	is a list of K elements, where K is the number of studies, each element is a microarray or RNAseq expression matrix with G rows and N columns, where G is number of matched genes and N is the sample size.
X.list	is a list of K elements, each element includes is a phenotypic condition of the corresponding samples, with case=1, control=0.
norm.mat	is a matrix of normalization factors, with K columns and G rows.
index.seq	index for RNA-seq studies
index.array	index for microarray studies
iteration	is the number of MCMC chains wish to run
seed	is a initial seed for random number generator.

Value

a list of MCMC output matrices for three key parameters of interest: the DE indicator "Delta", the study-specific effect size "ES", and the grand mean effect size "Lambda". For details, please refer to the paper "A joint Bayesian modeling for integrating microarray and RNA-seq transcriptomic data"

Examples

```
## Not run:
data(SimData)
G <- nrow(Data.list[[1]])
adjust.seq1 <-  adjust.seq2 <- adjust.array1 <-  adjust.array2 <- rep(0,G)
## For simulation, we already know the normalization factor:
adjust.array1[1:200] <- adjust.array2[1:200] <- -0.25
adjust.array1[201:400] <- adjust.array2[201:400] <- 0.25
norm.mat <- cbind(adjust.seq1,adjust.seq2,adjust.array1,
adjust.array2)
index.seq <- 1:2
index.array <- 3:4
iteration <- 2000
MCMC.out <- RunMCMC(Data.list, X.list, norm.mat, index.seq,
                    index.array, iteration)

## End(Not run)
```

SimData *Test data*

Description

Test data of 4 studies and 2000 genes used in the simulation; includes a list of dataframes "Data.list" and a list of phenotypic conditions "X.list"

Usage

```
data("SimData")
```

Examples

```
data(SimData)
```

Index

∗Topic **datasets**
RealSubset, 5
SimData, 6

GetBayesianQ, 1

NormDiff, 2

parMCMC, 4

RealSubset, 5
RunMCMC, 5

SimData, 6