Package 'BayesMetaSeq'

October 20, 2016

Type Package

Title Bayesian hierarchical model for RNA-seq differential meta-analysis and biomarker categorization

Version 2.0

Date 2016-10-20

Author Tianzhou Ma

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

Depends MCMCpack, mvtnorm, BayesLogit, gtools, gdata, msm, snowfall, cvTools, plyr

Suggests ConsensusClusterPlus

Description A Bayesian hierarchical model for RNA-seq meta-analysis and biomarker categorization by study heterogeneity. BayesMetaSeq models the RNA-seq count data, integrate information across genes and across studies, and modeling homogeneous and heterogeneous differential signals across studies with a DPM model.

License GPL-2

RoxygenNote 5.0.1

R topics documented:

DPM *Function to run DPM part of the model*

Description

Function to run DPM part of the model

Usage

```
DPM(Beta.obs, Delta, iteration, thin, C.init = 10, pi.alpha = 2,
  seed = 12345)
```

Arguments

Beta.obs	A matrix of observed effect size, with G rows and K columns , where K is the number of studies.
Delta	A matrix of delta output from MCMCRun, with G*K rows and T columns , where T is the number of MCMC iterations.
iteration	The number of MCMC chains wish to run
C.init	A hyperparameter
pi.alpha	A hyperparameter
seed	An initial seed for random number generator.

Value

A matrix of clustering output (cluster assignment), with G rows and I columns, where I is equal to iteration/thin.

Examples

```
## Not run:
## After completing the MCMC step
data(test.data)
data(test.label)
Data.list <- test.data
X.list <-  test.label
G <- nrow(Data.list[[1]])
K <- length(Data.list)
libsize <- lapply(Data.list,colSums)
es_seq =function (y,X,lib) {
 y=array(y)
 v1=log((y[which(X==1)]+1)/(lib[which(X==1)]))
 v2=log((y[which(X==0)]+1)/(lib[which(X==0)]))
 u=mean(v1) - mean(v2)
 return(u)
}
## Calculate the observed effect size (get directionality)
beta.obs<-matrix(rep(0,G*K),nrow=G,ncol=K)
for (g in 1:G){
 for (k in 1:K){
   lib<-libsize[[k]]
   beta.obs[g,k]<-es_seq(y=Data.list[[k]][g,],X=X.list[[k]],lib=lib)
```

```
 }
}
## Only interested in the clustering of the first 300 DE genes
beta.obs.de <- beta.obs[1:300,]
delta.de <- MCMC.out[['Delta']][1:(300*3),]
C.init <- 10
pi.alpha<- 2
thin <- 20
## Run the DPM part
cluster.out <- DPM(Beta.obs = beta.obs.de, Delta = delta.de,iteration = iteration,
thin = thin, C.init = C.init, pi.alpha = pi.alpha,  seed=12345)
cluster.top <- cluster.out[,-c(1:(burnin/thin+1))] ## remove the burnin period
rownames(cluster.top) <- paste('gene',1:300,sep='')
## Performs consensus clustering to summarize posterior
library(ConsensusClusterPlus)
copy.upper.lower <- function(m)
{
 m[lower.tri(m)] <- t(m)[lower.tri(m)]
 m
}
mydist <- function(y) {
 diss.mat <- matrix(,nrow=nrow(y),ncol=nrow(y))
 rownames(diss.mat) <- colnames(diss.mat) <- rownames(y)
 diag(diss.mat) <- 1
 for (i in 1:nrow(diss.mat)) {
   for (j in i:ncol(diss.mat)){
     diss.mat[i,j] <- sum(y[i,]==y[j,])/ncol(y)
   }
 }
 diss.mat <- copy.upper.lower(diss.mat)
return(1-diss.mat)
}
results = ConsensusClusterPlus(d=t(cluster.top),maxK=6,reps=50,pItem=0.8,pFeature=1,
    title="Consensus Clustering Result",clusterAlg="hc",innerLinkage="ward.D2",
    finalLinkage="ward.D2", distance="mydist",seed=1262118388.71279,plot="png")

## End(Not run)
```

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

Description

Function to get the bayesian q-value

Usage

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

Arguments

`Delta`	is a matrix outputted from the MCMC
`G`	The number of genes
`K`	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=burnin)

## End(Not run)
```

`Initialize` *Function to initialize MCMC chain*

Description

Function to initialize MCMC chain

Usage

```
Initialize(Data.list, X.list, lambda.mean = 0, lambda.var = 10,
  eta.mean = 0, eta.var = 10, m.mean = 0, m.var = 10, seed = 12345)
```

Arguments

`Data.list`	A list of K elements, where K is the number of studies, each element is a RNAseq data matrix with G rows and N columns, where G is number of genes and N is the sample size. Genes are matched across all studies.
`X.list`	A list of K elements, each element includes is a phenotypic condition of the corresponding samples, with case=1, control=0
`lambda.mean`	A hyperparameter
`lambda.var`	A hyperparameter
`eta.mean`	A hyperparameter
`eta.var`	A hyperparameter
`m.mean`	A hyperparameter
`m.var`	A hyperparameter
`seed`	An initial seed for random number generator.

Value

A list of initial values for each parameter

Examples

```
## Not run:
data(test.data)
data(test.label)
Data.list <- test.data
X.list <- test.label
G <- nrow(Data.list[[1]])
K <- length(Data.list)
lambda.mean<-0
lambda.var<- 10
eta.mean<- 0
eta.var<- 10
m.mean<- 0
m.var<- 10
init <- Initialize(Data.list=Data.list,X.list=X.list,
          lambda.mean=lambda.mean,lambda.var=lambda.var,
          eta.mean=eta.mean,eta.var=eta.var,
          m.mean=m.mean,m.var=m.var,seed=12345)

## End(Not run)
```

MCMCRun *Function to run MCMC chain*

Description

Function to run MCMC chain

Usage

```
MCMCRun(Data.list, X.list, Init.value, Store.value, iteration,
  lambda.mean = 0, lambda.var = 10, eta.mean = 0, eta.var = 10,
  m.mean = 0, m.var = 10, seed = 12345)
```

Arguments

Data.list	A list of K elements, where K is the number of studies, each element is an RNAseq data matrix with G rows and N columns, where G is number of genes and N is the sample size. Genes are matched across all studies.
X.list	A list of K elements, each element includes is a phenotypic condition of the corresponding samples, with case=1, control=0.
Init.value	The output from Initalize
Store.value	The output from Store
iteration	The number of MCMC chains wish to run
lambda.mean	A hyperparameter
lambda.var	A hyperparameter
eta.mean	A hyperparameter
eta.var	A hyperparameter
m.mean	A hyperparameter
m.var	A hyperparameter
seed	An initial seed for random number generator.

Value

A list of MCMC output matrices for each parameter

Examples

```
## Not run:
MCMC.out <- MCMCRun(Data.list=Data.list,X.list=X.list,Init.value=init,
                  Store.value = store,iteration=iteration,
                  lambda.mean=lambda.mean, lambda.var=lambda.var,
                   eta.mean=eta.mean, eta.var=eta.var,
                   m.mean=m.mean, m.var=m.var, seed=12345)

## End(Not run)
```

---

parMCMC *Function to run parallel MCMC chain*

Description

Function to run parallel MCMC chain

Usage

```
parMCMC(Data.list, X.list, iteration, chunks, lambda.mean = 0,
  lambda.var = 10, eta.mean = 0, eta.var = 10, m.mean = 0, m.var = 10,
  seed = 12345)
```

Arguments

Data.list	A list of K elements, where K is the number of studies, each element is an RNAseq data matrix with G rows and N columns, where G is number of genes and N is the sample size. Genes are matched across all studies.
X.list	A list of K elements, each element includes is a phenotypic condition of the corresponding samples, with case=1, control=0.
iteration	The number of MCMC chains wish to run
chunks	The number of data chunks to split into (= number of CPUs called in snowfall)
lambda.mean	A hyperparameter
lambda.var	A hyperparameter
eta.mean	A hyperparameter
eta.var	A hyperparameter
m.mean	A hyperparameter
m.var	A hyperparameter
seed	An initial seed for random number generator.

Value

A list of three components:

- Delta A output matrix of DE indicator estimate delta (Reordered according to gene.name).
- Beta A output matrix of effect size estimate beta (Reordered according to gene.name).
- Gene.Name Reordered gene names according to random split of original dataset; if the original dataset has no gene names, we will simply use gene(RowNumber) to denote the gene name, e.g. gene1 refers to the first gene.

Examples

```
## Not run:
data(test.data)
data(test.label)
Data.list <- test.data
X.list <- test.label
G <- nrow(Data.list[[1]])
K <- length(Data.list)
lambda.mean<-0
lambda.var<- 10
eta.mean<- 0
eta.var<- 10
m.mean<- 0
m.var<- 10
iteration <- 2000
burnin <- 500
chunks <- 2
MCMC.out <- parMCMC(Data.list=Data.list,X.list=X.list,iteration=iteration,
                    chunks = chunks,
                   lambda.mean=lambda.mean, lambda.var=lambda.var,
                    eta.mean=eta.mean, eta.var=eta.var,
                    m.mean=m.mean, m.var=m.var, seed=12345)
delta <- MCMC.out[['Delta']]
gene.name <- MCMC.out[['Gene.Name']]
delta.mean <- apply(delta[,-c(1:burnin)],1,mean)
delta.mat <- matrix(delta.mean,nrow=G,ncol=K,byrow=T)
rownames(delta.mat) <- gene.name
### Don't forget to turn the output back to the original gene order as in Data.list:
delta.mat.order <- delta.mat[rownames(Data.list[[1]]),]

## End(Not run)
```

Store *Function to produce empty matrices to store MCMC results*

Description

Function to produce empty matrices to store MCMC results.

Usage

```
Store(Data.list, iteration)
```

Arguments

Data.list	A list of K elements, where K is the number of studies, each element is an RNAseq data matrix with G rows and N columns, where G is number of genes and N is the sample size. Genes are matched across all studies.
iteration	The number of MCMC chains wish to run

Value

A list of empty store matrices for each parameter

Examples

```
## Not run:
iteration <- 2000
burnin <- 500
store <-Store(Data.list=Data.list, iteration=iteration)

## End(Not run)
```

---

test.data *Test dataset of 3 studies, each study has 1000 genes and 10 samples.*

Usage

```
data("test.data")
```

Examples

```
data(test.data)
## maybe str(test.data) ; plot(test.data) ...
```

---

test.label *A list of 3, corresponding to the phenotypic condition of the test dataset (1=case, 0=control)*

Usage

```
data("test.label")
```

Examples

```
data(test.label)
## maybe str(test.label) ; plot(test.label) ...
```

`Traceplot` *Function to produce traceplots*

Description

Function to produce traceplots

Usage

```
Traceplot(x, name)
```

Arguments

`x`	A vector of estimate of the parameter from MCMC, of length T (iterations)
`name`	The name of the parameter

Value

A traceplot in png format saved in the working directory

Examples

```
## Not run:
beta <- MCMC.out[['Beta']][1,]
name <- expression(beta)
Traceplot(x=beta,name=name)

## End(Not run)
```

Index