Package 'BayesMetaSeq'

March 19, 2016

Type Package

Title Bayesian hierarchical model for RNA-seq differential meta-analysis

Version 1.0

Date 2016-01-29

Author Tianzhou Ma, George Tseng

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

Depends R (>= 2.10.0), coda (>= 0.11-3), MASS

Description A Bayesian hierarchical model for RNA-seq meta-analysis and biomarkers 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. In addition, the package implements some post-hoc analysis as shown in the paper including power and ROC analysis, heatmap to show clustering results, pathway analysis, etc.

License GPL (>=2)

Imports coda, MASS, gplots, MCMCpack,gplots, gdata, gtools, BayesLogit, mvtnorm

R topics documented:

`BayesMetaSeq-package` *Bayesian hierarchical model for RNA-seq differential meta-analysis*

Description

A Bayesian hierarchical model for RNA-seq meta-analysis and biomarkers 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. In addition, the package implements some post-hoc analysis as shown in the paper including power and ROC analysis, heatmap to show clustering results, pathway analysis, etc.

Details

Package:	BayesMetaSeq
Type:	Package
Version:	1.0
Date:	2016-01-29
License:	What license is it under?

This Bayesian hierarchical model estimate the parameters through Markov Chain Monte Carlo Chain (MCMC) algorithm. "Initialize": first initialize the MCMC chain, "Store": set up an empty list to store estimates of variables in the chain, "MCMCRun": run the MCMC chain.

Author(s)

Tianzhou Ma, George Tseng

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

`DPM` *Run model-based clustering under DPM framework to categorize DE genes (same function already embeded in the MCMC step)*

Description

Input is a list of multi-study count data and the corresponding matrix of differential indicator estimates from the MCMC step. Run DPM to categorize the DE genes. Output the clustering results.

Usage

```
DPM(Data.list, Delta, C_init, pi_alpha, iteration, thin, seed=12345)
```

Arguments

Data.list	Input list of K RNA-seq data matrices with genes on rows and samples on columns, the genes need to be matched in all studies.
Delta	Input matrix of differential indicator estimates with genes on rows and iterations on columns.
C_init	The initial number of clusters to start with in DPM.
pi_alpha	The prior concentration paratmer of Dirichlet distribution.
iteration	Number of MCMC chains to run
thin	Number of MCMC chains to accumulate for clustering in the Dirichelet Process Mixture model
seed	Random seed

Examples

```
    ## Not run:
    ## load the cluster assignment matrix from SimuIII
    data(SimIII_Data)
    Data.list <- count
    Delta <- MCMC.out[['Delta']] ## output from MCMC step (delta matrix)
    C_init <- 10
    pi_alpha <- 2
    iteration <- 10000
    thin <- 20
SimuClusterAssign <- DPM(Data.list=Data.list,Delta=Delta,
C_init=C_init,pi_alpha=pi_alpha, iteration=iteration, thin=thin,seed=12345)

## End(Not run)
```

GetBayesianQ *Compute the Bayesian q-values (BFDR)*

Description

Based on the matrix of estimates of DE indicators from the MCMC output, compute the Bayesian q-value for each gene.

Usage

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

Arguments

Delta	Input matrix of estimates of DE indicators (either 1 or 0) from the MCMC output, with row number equal to GxK (ordered as follows: g1_1,...,g1_K, g2_1,...,g2_K, ...) and column number equal to total number of MCMC chains.
G	Number of genes
K	Number of studies
burnin	The burn-in period one wishes to discard

Value

A vector of Bayesian q-value is returned.

Examples

```
    ## Not run:
data(SimIII_DEOut)
q_real <- GetBayesianQ(Delta = Delta,G=1000,K=3,burnin=3000)

## End(Not run)
```

`GetGOoutput` *GO pathway analysis*

Description

GO pathway analysis for the species "Rattus Rattus" based on the package "topGO".

Usage

```
GetGOoutput(DE.genes, All.genes, nodesize_lower, nodesize_upper, topnodes)
```

Arguments

`DE.genes`	A vector of DE gene symbols
`All.genes`	A vector of all gene symbols
`nodesize_lower`	The minimum node size of an GO term to be returned
`nodesize_upper`	The maximum node size of an GO term to be returned
`topnodes`	The number of top GO terms (sorted by significance level) to be returned

Value

A matrix of GO output, including GO.ID, GO term, Fisher's p-value, Odds ratio, Annotated genes, Significant genes, etc.

Examples

```
## Not run:

## Need the following packages from Bioconductor
source("http://bioconductor.org/biocLite.R")
biocLite("topGO")
biocLite("ALL")
biocLite("org.Rn.eg.db")

## load the top200 genes from each method as well as all background genes
data(Top200Genes)
Allgenes <- Top200Genes[[1]]
Bayes200genes <- Top200Genes[[2]]
edgeR200genes <- Top200Genes[[3]]
DESeq200genes <- Top200Genes[[4]]
```

```
BayesGO <-GetGOoutput(DE.genes=Bayes200genes,All.genes=Allgenes,nodesize_lower=10,
nodesize_upper=200,topnodes=200)
edgeRGO <- GetGOoutput(DE.genes=edgeR200genes,All.genes=Allgenes,nodesize_lower=10,
nodesize_upper=200,topnodes=200)
DESeqGO <- GetGOoutput(DE.genes=DESeq200genes,All.genes=Allgenes,nodesize_lower=10,
nodesize_upper=200,topnodes=200)



## End(Not run)
```

`GetHeatMap` *Plot the heatmap of hierarchical clustering results*

Description

Compute the distance matrix based on the co-occurrence probability for any two genes (the number of times the two genes are assigned to the same cluster divided by the total number of assignments from the MCMC output). Then plot the heatmap of hierarchical clustering results based on the distance matrix

Usage

```
GetHeatMap(cluster.top)
```

Arguments

`cluster.top`	Cluster assignment results from the MCMC output

Examples

```
## Not run:
## install gplots if necessary
install.packages('gplots')
## load the cluster assignment matrix from SimuIII
data(SimIII_ClusterOut)
burnin <- 3000
thin <- 20
DE.index <- 1:300   ## DE gene index
cluster <- SimuClusterAssign[,-c(1:(burnin/thin+1))]
cluster.top <- cluster[DE.index,]
pdf('HeatMapSimCluster.pdf')
GetHeatMap(cluster.top=cluster.top) ## may take some time
dev.off()

## load the cluster assignment matrix from real data (Bayesian 245 DE genes)
data(ClusterBayes245Genes)
cluster.top <- ClusterBayes245Genes ## DE genes only, burnin period already removed
pdf('HeatMapRealCluster.pdf')
GetHeatMap(cluster.top=cluster.top) ## may take some time
dev.off()

## End(Not run)
```

GetPower *Compute the number of true positives among top declared DE genes*

Description

Based on the output from either Bayesian model (the posterior means of DE indicators) or edgeR/DESeq (p-values), compute the number of true positives among top declared DE genes.

Usage

```
GetPower(top, method, data, true.ind)
```

Arguments

top	A vector of top number of DE genes declared
method	Either "Bayesian" or "Fisher"
data	Input data, posterior means of DE indicators for Bayesian method, and p-values for Fisher's method
true.ind	Index of DE genes (truth)

Value

A vector of number of true positives

Examples

```
 ## Not run:
   ## Simulation output from Simu IA K=2
data(SimIA_K2_out)
BayesLowOut <- SimIA_K2_out[c(101:200,601:1000),]
top<-seq(2,100,by=2)
    true.ind <- 1:100
(sapply(top,GetPower,method="Bayes",data=BayesLowOut, true.ind=true.ind))

    BayesHighOut <- SimIA_K2_out[c(1:100,201:600),]
top<-seq(2,100,by=2)
    true.ind <- 1:100
(sapply(top,GetPower,method="Bayes",data=BayesHighOut, true.ind=true.ind))


## End(Not run)
```

GetROC *Compute the sensitivity and specificity*

Description

Based on the output from either Bayesian model (the posterior means of DE indicators) or edgeR/DESeq (p-values), compute the sensitivity and specificity.

Usage

```
GetROC(all, delta.cut, method, data, true.ind, false.ind)
```

Arguments

all	A vector of all gene indices
delta.cut	A vector of posterior mean cutoff for declaring DE or Non-DE, only available when method = "Bayesian"
method	Either "Bayesian" or "Fisher"
data	Input data, posterior means of DE indicators for Bayesian method, and p-values for Fisher's method
true.ind	Index of DE genes (truth)
false.ind	Index of Non-DE genes (truth)

Value

Sensitivity and Specificity returned

Examples

```
   ## Not run:
    ## Simulation output from Simu IA K=2
    data(SimIA_K2_out)
BayesLowOut <- SimIA_K2_out[c(101:200,601:1000),]
    all.ind<-seq(1,500,5)
    delta.cut <- seq(0,1,0.01)
    true.ind <- 1:100
    false.ind <- 101:500
    Bayes.L.Sens <- Bayes.L.Spec<- rep(NA,length(delta.cut))
    for (i in 1:length(delta.cut)){
   out <- GetROC(all=all.ind[i],delta.cut=delta.cut[i],method="Bayes",data=BayesLowOut,   true.ind=true.ind
  Bayes.L.Sens[i] <- out[1]
  Bayes.L.Spec[i] <- out[2]
     }
    library(flux)
    auc(Bayes.L.Sens,Bayes.L.Spec)

    BayesHighOut <- SimIA_K2_out[c(1:100,201:600),]
    all.ind<-seq(1,500,5)
    delta.cut <- seq(0,1,0.01)
    true.ind <- 1:100
    false.ind <- 101:500
    Bayes.H.Sens <- Bayes.H.Spec<- rep(NA,length(delta.cut))
```

```
  for (i in 1:length(delta.cut)){
 out <- GetROC(all=all.ind[i],delta.cut=delta.cut[i],method="Bayes",data=BayesHighOut,   true.ind=true.in
Bayes.H.Sens[i] <- out[1]
Bayes.H.Spec[i] <- out[2]
  }
 library(flux)
 auc(Bayes.H.Sens,Bayes.H.Spec)


## End(Not run)
```

`Initialize` *Initialize the Markov Chain Monte Carlo Chain*

Description

Using the pre-specified hyperparameters to initialize the MCMC chain.

Usage

```
Initialize(Data.list, X.list, lambda_mean, lambda_var, eta_mean, eta_var, m_mean, m_var, IG_alpha
```

Arguments

`Data.list`	Input list of K RNA-seq data matrices with genes on rows and samples on columns, the genes need to be matched in all studies.
`X.list`	Input list of phenotypic data from the K studies, with 1 representing the case and 0 the control.
`lambda_mean`	The prior mean of effect size.
`lambda_var`	The prior variance of effect size.
`eta_mean`	The prior mean of baseline.
`eta_var`	The prior variance of baseline.
`m_mean`	The prior mean of dispersion.
`m_var`	The prior variance of dispersion.
`IG_alpha`	The prior shape parameter of variance.
`IG_beta`	The prior rate parameter of variance.
`C_init`	The initial number of clusters to start with in DPM.
`pi_alpha`	The prior concentration paratmer of Dirichlet distribution.
`epsilon_omega`	Tuning parameter
`epsilon_phi`	Tuning parameter
`epsilon_log_phi`	Tuning parameter
`cov_tune`	Tuning parameter
`seed`	Random seed

Value

A list object that contains the initial values for all the parameters that need to be estimated in MCMC.

Examples

```
  ## Not run:
   ## install the necessary packages
   #install.packages(c('MCMCpack','mvtnorm','BayesLogit','gtools','msm') )

   ## load a 2-study toy example of simulated RNA-seq data to run the Bayesian model
data(Toy2studyEx)
Data.list <- Toy2studyEx
libsize <- lapply(Data.list,colSums) # library size
    logT <- lapply(libsize,log) # log scale library size
G <- sapply(Data.list,nrow)[1]  # number of genes
    S <- sapply(Data.list,ncol)    # number of samples in each study
    K <- length(Data.list) # number of studies
    X.list <- lapply(1:K,FUN=function(k) {
     c(rep(1,S[k]/2),rep(0,S[k]/2)) }   )   ## balanced phenotypic condition

    lambda_mean<-0
    lambda_var<-1
    eta_mean<- -5
    eta_var<- 2
    m_mean <- -2
    m_var <- 0.5
    IG_alpha<- 5
    IG_beta<- 1
    C_init <- 10
    pi_alpha<- 2

    ## Initialize the chain

    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,
                       IG_alpha=IG_alpha,IG_beta=IG_beta,
                       C_init=C_init,pi_alpha=pi_alpha,seed=12345)


## End(Not run)
```

MCMCRun *Run the Markov Chain Monte Carlo Algorithm*

Description

Run the MCMC chain, the posterior estimates will fill up the empty store list.

Usage

```
MCMCRun(Data.list, X.list, Init.value, Store.value, iteration, thin, lambda_mean, lambda_var, eta
```

Arguments

Data.list	Input list of K RNA-seq data matrices with genes on rows and samples on columns, the genes need to be matched in all studies.

X.list	Input list of phenotypic data from the K studies, with 1 representing the case and 0 the control.
Init.value	Initial values of the MCMC chain
Store.value	Empty list that stores the estimates
iteration	Number of MCMC chains to run
thin	Number of MCMC chains to accumulate for clustering in the Dirichelet Process Mixture model
lambda_mean	The prior mean of effect size.
lambda_var	The prior variance of effect size.
eta_mean	The prior mean of baseline.
eta_var	The prior variance of baseline.
m_mean	The prior mean of dispersion.
m_var	The prior variance of dispersion.
IG_alpha	The prior shape parameter of variance.
IG_beta	The prior rate parameter of variance.
pi_alpha	The prior concentration paratmer of Dirichlet distribution.
epsilon_omega	Tuning parameter
epsilon_phi	Tuning parameter
epsilon_log_phi	Tuning parameter
cov_tune	Tuning parameter
seed	Random seed

Value

A list of two elements: the delta matrix ("G*K" rows and "iteration" columns) for differential expression inference and the cluster assignment matrix ("G" rows and "round(iteration/thin)" columns) for clustering analysis.

Examples

```
## Not run:
     MCMC.out <- MCMCRun(Data.list=Data.list,X.list=X.list,Init.value=init,
     Store.value=store,iteration=iteration, thin=thin,
     lambda_mean=lambda_mean, lambda_var=lambda_var, eta_mean=eta_mean, eta_var=eta_var,
    m_mean=m_mean, m_var=m_var, IG_alpha=IG_alpha, IG_beta=IG_beta, pi_alpha=pi_alpha, seed=12345)

## End(Not run)
```

SimIA_K2_Data *Simulation IA data (K=2)*

Description

This dataframe contains a list of 2 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimIA_K2_Data)
```

SimIA_K5_Data *Simulation IA data (K=5)*

Description

This dataframe contains a list of 5 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimIA_K2_Data)
```

SimIB_K2_Data *Simulation IB data (K=2)*

Description

This dataframe contains a list of 2 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimIB_K2_Data)
```

SimIB_K5_Data *Simulation IB data (K=5)*

Description

This dataframe contains a list of 5 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimIB_K5_Data)
```

SimIII_Data *Simulation III data (K=3)*

Description

This dataframe contains a list of 3 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimIII_Data)
```

SimII_K2_Data *Simulation II data (K=2)*

Description

This dataframe contains a list of 2 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimII_K2_Data)
```

SimII_K5_Data *Simulation II data (K=5)*

Description

This dataframe contains a list of 5 count data matrices, each matrix has 1000 rows of genes and 10 columns of samples (G=1000, N=10).

Usage

```
data(SimII_K2_Data)
```

Store *Store estimates of parameters in Markov Chain Monte Carlo Chain*

Description

Set up an empty list to store the estimates of parameters in the MCMC chain.

Usage

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

Arguments

Data.list	Input list of K RNA-seq data matrices with genes on rows and samples on columns, the genes need to be matched in all studies.
X.list	Input list of phenotypic data from the K studies, with 1 representing the case and 0 the control.
iteration	Number of MCMC chains to run
thin	Number of MCMC chains to accumulate for clustering in the Dirichelet Process Mixture model

Value

A empty list object that will store the estimates of parameters in the MCMC chain.

Examples

```
## Not run:
 iteration<- 200
 thin<- 50

 ## Set store list
 store <- Store(Data.list=Data.list,X.list=X.list,
                iteration=iteration,thin=thin)


## End(Not run)
```

Index