Package ‘MODA’

April 26, 2017

Type Package

Title MODA: MOdule Differential Analysis for weighted gene co-expression network

Version 1.2.0

Date 2016-12-16

Author Dong Li, James B. Brown, Luisa Orsini, Zhisong Pan, Guyu Hu and Shan He

Maintainer Dong Li <dxl466@cs.bham.ac.uk>

Description MODA can be used to estimate and construct condition-specific gene co-expression networks, and identify differentially expressed subnetworks as conserved or condition-specific modules which are potentially associated with relevant biological processes.

License GPL (>= 2)

Depends R (>= 3.3)

Imports grDevices, graphics, stats, utils, WGCNA, dynamicTreeCut, igraph, cluster, AMOUNTAIN, RColorBrewer

RoxygenNote 5.0.1

biocViews GeneExpression, Microarray, DifferentialExpression, Network

Suggests BiocStyle, knitr, rmarkdown

ignoreBuilder knitr

NeedsCompilation no

R topics documented:

CompareAllNets .................................................. 2
comparemodulestwonets ................................. 3
datExpr1 ....................................................... 4
datExpr2 ....................................................... 4
getPartition .................................................. 5
MIcondition ..................................................... 5
ModuleFrequency ............................................. 6
modulesRank .................................................. 7
NMImatrix ..................................................... 7
PartitionDensity ............................................. 8
PartitionModularity ......................................... 9
recursiveigraph .............................. 10
CompareAllNets

Illustration of network comparison

Description

Compare the background network and a set of condition-specific network. Conserved or condition-specific modules are indicated by the plain files, based on the statistics

Usage

CompareAllNets(ResultFolder, intModules, indicator, intconditionModules, conditionNames, specificTheta, conservedTheta)

Arguments

ResultFolder where to store results
intModules how many modules in the background network
indicator identifier of current profile, served as a tag in name
intconditionModules a numeric vector, each of them is the number of modules in each condition-specific network. Or just single number
conditionNames a character vector, each of them is the name of condition. Or just single name
specificTheta the threshold to define min(s)+specificTheta, less than which is considered as condition specific module. s is the sums of rows in Jaccard index matrix. See supplementary file.
conservedTheta The threshold to define max(s)-conservedTheta, greater than which is considered as condition conserved module. s is the sums of rows in Jaccard index matrix. See supplementary file.

Value

None

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

See Also

WeightedModulePartitionHierarchical, comparemodulestwonets
Examples

data(synthetic)
ResultFolder = 'ForSynthetic' # where middle files are stored
CuttingCriterion = 'Density' # could be Density or Modularity
indicator1 = 'X' # indicator for data profile 1
indicator2 = 'Y' # indicator for data profile 2
specificTheta = 0.1 #threshold to define condition specific modules
conservedTheta = 0.1 #threshold to define conserved modules
intModules1 <- WeightedModulePartitionHierarchical(datExpr1, ResultFolder, indicator1, CuttingCriterion)
intModules2 <- WeightedModulePartitionHierarchical(datExpr2, ResultFolder, indicator2, CuttingCriterion)
CompareAllNets(ResultFolder, intModules1, indicator1, intModules2, indicator2, specificTheta, conservedTheta)

Description

Compare the background network and a condition-specific network. A Jaccard index is used to measure the similarity of two sets, which represents the similarity of each module pairs from two networks.

Usage

comparemodulestwonets(sourcehead, nm1, nm2, ind1, ind2)

Arguments

sourcehead prefix of where to store results
nm1 how many modules in the background network
nm2 how many modules in the condition-specific network
ind1 indicator of the background network
ind2 indicator of the condition-specific network

Value

A matrix where each entry is the Jaccard index of corresponding modules from two networks

Author(s)

Dong Li, <dxl1466@cs.bham.ac.uk>
Examples

data(synthetic)
ResultFolder = 'ForSynthetic' # where middle files are stored
CuttingCriterion = 'Density' # could be Density or Modularity
indicator1 = 'X'    # indicator for data profile 1
indicator2 = 'Y'    # indicator for data profile 2
intModules1 <- WeightedModulePartitionHierarchical(datExpr1,ResultFolder,
indicator1,CuttingCriterion)
intModules2 <- WeightedModulePartitionHierarchical(datExpr2,ResultFolder,
indicator2,CuttingCriterion)
JaccardMatrix <- comparemodulestwonets(ResultFolder,intModules1,intModules2,
paste('/DenseModuleGene_',indicator1,sep=''),
paste('/DenseModuleGene_',indicator2,sep=''))

Description
Synthetic gene expression profile with 20 samples and 500 genes.

Format
A matrix with 20 rows and 500 columns.

Author(s)
Dong Li, <dxl466@cs.bham.ac.uk>

Examples
data(synthetic)
### plot the heatmap of the correlation matrix ...
### Not run: heatmap(cor(as.matrix(datExpr1)))

Description
Synthetic gene expression profile with 25 samples and 500 genes.

Format
A matrix with 25 rows and 500 columns.

Author(s)
Dong Li, <dxl466@cs.bham.ac.uk>
getPartition

Examples

data(synthetic)
## plot the heatmap of the correlation matrix ...
## Not run: heatmap(cor(as.matrix(datExpr2)))

getPartition

Get numeric partition from folder

Description

Get identified partitionAssignment, only for synthetic data where gene names are numbers

Usage

getPartition(ResultFolder)

Arguments

ResultFolder  folder used to save modules

Value

Number of partitions

MIcondition

Modules detection by each condition

Description

Module detection on each condition-specific network, which is constructed from all samples but samples belonging to that condition

Usage

MIcondition(datExpr, conditionNames, ResultFolder, GeneNames, maxsize = 100, minsize = 30)

Arguments

datExpr  gene expression profile, rows are samples and columns genes, rowname should contain condition specifier
conditionNames  character vector, each as the condition name
ResultFolder  where to store the clusters
GeneNames  normally the gene official names to replace the colnames of datExpr
maxsize  the maximal nodes allowed in one module
minsize  the minimal nodes allowed in one module
ModuleFrequency

Value

a numeric vector, each entry is the number of modules in condition-specific network

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

Description

Statistics of all conditions. To highlight conserved or condition-specific by counting how frequent each module is labelled as which, and then visualize the frequency by bar plot.

Usage

ModuleFrequency(ResultFolder, intModules, conditionNames, legendNames, indicator)

Arguments

ResultFolder where to store results
intModules how many modules in the background network
conditionNames a character vector, each of them is the name
legendNames a character vector, each of them is the condition name showing up in the frequency barplot of condition. Or just single name
indicator identifier of current profile, served as a tag in name

Value

None

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

See Also

WeightedModulePartitionHierarchical, WeightedModulePartitionLouvain, WeightedModulePartitionSpectral, WeightedModulePartitionAmoutain, CompareAllNets
modulesRank

*Modules rank from recursive communities detection*

**Description**

Assign the module scores by weights, and rank them from highest to lowest.

**Usage**

```r
modulesRank(foldername, indicator, GeneNames)
```

**Arguments**

- `foldername`: folder used to save modules
- `indicator`: normally a specific tag of condition
- `GeneNames`: Gene symbols, sometimes we need them instead of probe ids

**Value**

The number of modules

**Author(s)**

Dong Li, <dxl466@cs.bham.ac.uk>

**See Also**

- `recursiveigraph`

---

NMImatrix

*Illustration of network comparison by NMI*

**Description**

Compare the background network and a set of condition-specific network, returning a pair-wise matrix to show the normalized mutual information between each pair of networks in terms of partitioning.

**Usage**

```r
NMImatrix(ResultFolder, intModules, indicator, intconditionModules, conditionNames, Nsize, legendNames = NULL, plt = FALSE)
```
Arguments

ResultFolder where to store results
intModules how many modules in the background network
indicator identifier of current profile, served as a tag in name
intconditionModules a numeric vector, each of them is the number of modules in each condition-specific network. Or just single number
conditionNames a character vector, each of them is the name of condition. Or just single name
Nsize The number of genes in total
legendNames a character vector, each of them is the condition name showing up in the similarity matrix plot if applicable
plt a boolean value to indicate whether plot the similarity matrix

Value

NMI matrix indicating the similarity between each two networks

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

See Also

CompareAllNets

Description

Calculate the average density of all resulting modules from a partition. The density of each module is defined as the average adjacency of the module genes.

Usage

PartitionDensity(ADJ, PartitionSet)

Arguments

ADJ gene similarity matrix
PartitionSet vector indicates the partition label for genes

Value

partition density, defined as average density of all modules

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>
PartitionModularity

References


Examples

data(synthetic)
ADJ1=abs(cor(datExpr1,use="p"){10
dissADJ=1-ADJ1
hierADJ=hclust(as.dist(dissADJ), method="average" )
groups <- cutree(hierADJ, h = 0.8)
pDensity <- PartitionDensity(ADJ1,groups)

Illustration of modularity density

Description

Calculate the average modularity of a partition. The modularity of each module is defined from a natural generalization of unweighted case.

Usage

PartitionModularity(ADJ, PartitionSet)

Arguments

ADJ gene similarity matrix
PartitionSet vector indicates the partition label for genes

Value

partition modularity, defined as average modularity of all modules

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

References


Examples

data(synthetic)
ADJ1=abs(cor(datExpr1,use="p"){10
dissADJ=1-ADJ1
hierADJ=hclust(as.dist(dissADJ), method="average" )
groups <- cutree(hierADJ, h = 0.8)
pDensity <- PartitionModularity(ADJ1,groups)
### recursiveigraph

*Modules identification by recursive community detection*

**Description**

Modules detection using igraph’s community detection algorithms, when the resulted module is larger than expected, it is further divided by the same program.

**Usage**

```r
recursiveigraph(g, savefile, method = c("fastgreedy", "louvain"),
              maxsize = 200, minsize = 30)
```

**Arguments**

- `g`: igraph object, the network to be partitioned
- `savefile`: plain text, used to store module, each line as a module
- `method`: specify the community detection algorithm
- `maxsize`: maximal module size
- `minsize`: minimal module size

**Value**

None

**Author(s)**

Dong Li, <dxl466@cs.bham.ac.uk>

**References**


---

### WeightedModulePartitionAmoutain

*Modules detection by AMOUNTAIN algorithm*

**Description**

Module detection based on the AMOUNTAIN algorithm, which tries to find the optimal module every time and use a modules extraction way.

**Usage**

```r
WeightedModulePartitionAmoutain(datExpr, Nmodule, foldername, indicatename,
                                GeneNames, maxsize = 200, minsize = 3, power = 6, tao = 0.2)
```
WeightedModulePartitionHierarchical

Arguments

datExpr  gene expression profile, rows are samples and columns genes
Nmodule  the number of clusters(modules)
foldername  where to store the clusters
indicatename  normally a specific tag of condition
GeneNames  normally the gene official names to replace the colnames of datExpr
maxsize  the maximal nodes allowed in one module
minsize  the minimal nodes allowed in one module
power  the power parameter of WGCNA, $W_{ij}=|\text{cor}(x_i,x_j)|^pwr$
tao  the threshold to cut the adjacency matrix

Value

None

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

References


Examples

data(synthetic)
ResultFolder <- 'ForSynthetic' # where middle files are stored
GeneNames <- colnames(datExpr1)
ingtModules1 <- WeightedModulePartitionAmoutain(datExpr1,5,ResultFolder,'X',
GeneNames,maxsize=100,minsize=50)
truemodule <- c(rep(1,100),rep(2,100),rep(3,100),rep(4,100),rep(5,100))
#mymodule <- getPartition(ResultFolder)
#randIndex(table(mymodule,truemodule),adjust=F)

Description

Module detection based on the optimal cutting height of dendrogram, which is selected to make the average density or modularity of resulting partition maximal. The clustering and visualization function are from WGCNA.

Usage

WeightedModulePartitionHierarchical(datExpr, foldername, indicatename,
cutmethod = c("Density", "Modularity"), power = 10)
WeightedModulePartitionLouvain

Arguments

datExpr  gene expression profile, rows are samples and columns genes
foldername  where to store the clusters
indicatename  normally a specific tag of condition
cutmethod  cutting the dendrogram based on maximal average Density or Modularity
power  the power parameter of WGCNA, W_{ij}=\text{cor}(x_i,x_j)^{\text{power}}

Value

The number of clusters

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

References


See Also

PartitionDensity
PartitionModularity

Examples

data(synthetic)
ResultFolder = 'ForSynthetic' # where middle files are stored
CuttingCriterion = 'Density' # could be Density or Modularity
indicator1 = 'X'  # indicator for data profile 1
indicator2 = 'Y'  # indicator for data profile 2
specificTheta = 0.1 #threshold to define condition specific modules
conservedTheta = 0.1 #threshold to define conserved modules
intModules1 <- WeightedModulePartitionHierarchical(datExpr1,ResultFolder, indicator1,CuttingCriterion)
#mymodule <- getPartition(ResultFolder)
#randIndex(table(mymodule,truemodule),adjust=F)

Description

Module detection based on the Louvain algorithm, which tries to maximize overall modularity of resulting partition.

Usage

WeightedModulePartitionLouvain(datExpr, foldername, indicatename, GeneNames, maxsize = 200, minsize = 30, power = 6, tao = 0.2)
WeightedModulePartitionSpectral

Arguments

- **datExpr**
  - gene expression profile, rows are samples and columns genes

- **foldername**
  - where to store the clusters

- **indicatename**
  - normally a specific tag of condition

- **GeneNames**
  - normally the gene official names to replace the colnames of datExpr

- **maxsize**
  - the maximal nodes allowed in one module

- **minsize**
  - the minimal nodes allowed in one module

- **power**
  - the power parameter of WGCNA, \(W_{ij} = |\text{cor}(x_i, x_j)|^\text{power}\)

- **tao**
  - the threshold to cut the adjacency matrix

Value

The number of clusters

Author(s)

Dong Li, <dxl466@cs.bham.ac.uk>

References


Examples

```r
data(synthetic)
ResultFolder <- 'ForSynthetic' # where middle files are stored
indicator <- 'X' # indicator for data profile 1
GeneNames <- colnames(datExpr)
intModules1 <- WeightedModulePartitionLouvain(datExpr, ResultFolder, indicator, GeneNames)
trueModule <- c(rep(1,100),rep(2,100),rep(3,100),rep(4,100),rep(5,100))
#mymodule <- getPartition(ResultFolder)
#randIndex(table(mymodule,truemodule),adjust=F)
```

Description

Module detection based on the spectral clustering algorithm, which mainly solve the eigendecomposition on Laplacian matrix

Usage

```r
WeightedModulePartitionSpectral(datExpr, foldername, indicatename, GeneNames,
    power = 6, nn = 10, k = 2)
```
**WeightedModulePartitionSpectral**

**Arguments**

- `datExpr`: gene expression profile, rows are samples and columns genes
- `foldername`: where to store the clusters
- `indicatename`: normally a specific tag of condition
- `GeneNames`: normally the gene official names to replace the colnames of `datExpr`
- `power`: the power parameter of WGCNA, \( W_{ij}=|\text{cor}(x_i,x_j)|^\text{power} \)
- `nn`: the number of nearest neighbor, used to construct the affinity matrix
- `k`: the number of clusters(modules)

**Value**

None

**Author(s)**

Dong Li, <dxl466@cs.bham.ac.uk>

**References**


**Examples**

data(synthetic)
ResultFolder <- 'ForSynthetic' # where middle files are stored
indicator <- 'X' # indicator for data profile 1
GeneNames <- colnames(datExpr1)
WeightedModulePartitionSpectral(datExpr1,ResultFolder,indicator,
GeneNames,k=5)
truemodule <- c(rep(1,100),rep(2,100),rep(3,100),rep(4,100),rep(5,100))
#mymodule <- getPartition(ResultFolder)
#randIndex(table(mymodule,truemodule),adjust=F)
Index

*Topic NMI
  NMImatrix, 7

*Topic Statistics
  ModuleFrequency, 6

*Topic community
  recursiveigraph, 10

*Topic comparison
  comparemodulestwonets, 3

*Topic cutting
  WeightedModulePartitionHierarchical, 11
  WeightedModulePartitionLouvain, 12
  WeightedModulePartitionSpectral, 13

*Topic data
  datExpr1, 4
  datExpr2, 4

*Topic dendrogram
  WeightedModulePartitionHierarchical, 11
  WeightedModulePartitionLouvain, 12
  WeightedModulePartitionSpectral, 13

*Topic density
  PartitionDensity, 8

*Topic detection
  recursiveigraph, 10

*Topic differential
  CompareAllNets, 2
  ModuleFrequency, 6
  NMImatrix, 7

*Topic modularity
  PartitionModularity, 9

*Topic module
  CompareAllNets, 2
  comparemodulestwonets, 3
  ModuleFrequency, 6
  NMImatrix, 7

*Topic multiplecondition
  MIcondition, 5

*Topic optimization
  WeightedModulePartitionAmoutain, 10

CompareAllNets, 2, 6, 8
comparemodulestwonets, 2, 3
datExpr1, 4
datExpr2, 4
getPartition, 5
MIcondition, 5
ModuleFrequency, 6
modulesRank, 7
NMImatrix, 7
PartitionDensity, 8, 12
PartitionModularity, 9, 12
recursiveigraph, 7, 10
WeightedModulePartitionAmoutain, 6, 10
WeightedModulePartitionHierarchical, 2, 6, 11
WeightedModulePartitionLouvain, 6, 12
WeightedModulePartitionSpectral, 6, 13