Package 'GNET2'

December 23, 2024

Type Package

Title Constructing gene regulatory networks from expression data through functional module inference

Version 1.22.0

Author Chen Chen, Jie Hou and Jianlin Cheng

Maintainer Chen Chen <ccm3x@mail.missouri.edu>

Description Cluster genes to functional groups with E-M process. Iteratively perform TF assigning and Gene assigning, until the assignment of genes did not change, or max number of iterations is reached.

License Apache License 2.0

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Depends R (>= 3.6)

Imports ggplot2,xgboost,Rcpp,reshape2,grid,DiagrammeR,methods,stats,matrixStats,graphics,SummarizedExperiment,dplyr,i
grDevices, utils

RoxygenNote 7.1.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

biocViews GeneExpression, Regression, Network, NetworkInference, Software

URL https://github.com/chrischen1/GNET2

BugReports https://github.com/chrischen1/GNET2/issues

git_url https://git.bioconductor.org/packages/GNET2

git_branch RELEASE_3_20

git_last_commit 0009f98

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-12-23

Contents

`build_module` *Fit a regression tree.*

Description

Fit a regression tree based on Gaussian Likelihood score. Provided in case the best split is not applicable for R dnorm() function.

Usage

```
build_module(X, Y, max_depth, cor_cutoff, min_divide_size)
```

Arguments

`X`	A n by p matrix as input.
`Y`	A n by q matrix as response.
`max_depth`	Maximum depth of the tree.
`cor_cutoff`	Cutoff for within group Pearson correlation coefficient, if all data belong to a node have average correlation greater or equal to this, the node would not split anymore.
`min_divide_size`	Minimum number of data belong to a node allowed for further split of the node.

Value

A matrix for sample informatrion for each partition level. First column is feature index used by the node and second is the value used to split, the rest of the columns are the split of sample: 0 means less or equal, 1 means greater and -1 means the sample does not belong to this node.

Examples

```
build_module(X = matrix(rnorm(5*10),5,10), Y = matrix(rnorm(5*10),5,10),
                        max_depth=3,cor_cutoff=0.9,min_divide_size=3)
```

build_moduleR *Build regression tree.*

Description

Build regression tree based on Gaussian Likelihood score.

Usage

```
build_moduleR(X, Y, max_depth, cor_cutoff, min_divide_size)
```

Arguments

X	A n by p matrix as input.
Y	A n by q matrix as response.
max_depth	Maximum depth of the tree.
cor_cutoff	Cutoff for within group Pearson correlation coefficient, if all data belong to a node have average correlation greater or equal to this, the node would not split anymore.
min_divide_size	Minimum number of data belong to a node allowed for further split of the node.

Value

A matrix for sample information for each tree level. First column is feature index used by the node and second is the value used to split, the rest of the columns are the split of sample: 0 means less or equal, 1 means greater and -1 means the sample does not belong to this node.

Examples

```
build_moduleR(X = matrix(rnorm(5*10),5,10), Y = matrix(rnorm(5*10),5,10),
                            max_depth=3,cor_cutoff=0.9,min_divide_size=3)
```

build_moduleR_heuristic *Build regression tree with splits are detemined by K-means heuristicly.*

Description

Build regression tree based on Gaussian Likelihood score. The spliting of the tree is determined heuristicly by k_means.

Usage

```
build_moduleR_heuristic(
  X,
  Y,
  max_depth,
  cor_cutoff,
  min_divide_size,
  split_table
)
```

Arguments

X	A n by p matrix as input.
Y	A n by q matrix as response.
max_depth	Maximum depth of the tree.
cor_cutoff	Cutoff for within group Pearson correlation coefficient, if all data belong to a node have average correlation greater or equal to this, the node would not split anymore.
min_divide_size	Minimum number of data belong to a node allowed for further split of the node.
split_table	split table generated by K-means with build_split_table()

Value

A matrix for sample informatrion for each tree level. First column is feature index used by the node and second is the value used to split, the rest of the columns are the split of sample: 0 means less or equal, 1 means greater and -1 means the sample does not belong to this node.

Examples

```
X <- matrix(rnorm(5*10),5,10)
build_moduleR_heuristic(X = X, Y = matrix(rnorm(5*10),5,10),max_depth=3,cor_cutoff=0.9,
                        min_divide_size=3,split_table = build_split_table(X))
```

---

build_split_table *Build split table by K-means heuristicly.*

Description

Build split table by K-means with 3 cluster centers for each column of X

Usage

```
build_split_table(X)
```

Arguments

X	A n by p matrix as input.

Value

A n by p matrix with each column consists of 3 clusters: -1 for low, 0 for mid and 1 for high

Examples

```
split_table <- build_split_table(matrix(rnorm(5*10),5,10))
```

`calc_likelihood_score` *Calculate Gaussian Likelihood score.*

Description

Calculate Gaussian Likelihood score.

Usage

```
calc_likelihood_score(x, group_labels)
```

Arguments

`x`	A n by p matrix.
`group_labels`	A vector of length n, indicating the group of rows.

Value

The sum of log likelihood score of each group on each column.

Examples

```
calc_likelihood_score(x = matrix(rnorm(5*10),5,10), group_labels = c(rep(1,2),rep(2,3)))
```

`extract_edges` *Extract the network from the gnet result*

Description

Extract the network as edge list from the gnet result. For a module, each regulator and downstream gene will form a directed edge.

Usage

```
extract_edges(gnet_result)
```

Arguments

`gnet_result`	Returned results from gnet().

Value

A matrix of scores of for the regulator-target interaction.

Examples

```
set.seed(1)
init_group_num = 8
init_method = 'kmeans'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
edge_list <- extract_edges(gnet_result)
```

---

get_correlation_list *Calculate correlation within each group.*

Description

Calculate Pearson correlation coefficient within each group.

Usage

```
get_correlation_list(x, group_labels)
```

Arguments

x	A n by p matrix.
group_labels	A vector of length n, indicating the group of rows.

Value

An array of Pearson correlation coefficient for each row, rows belong to the same group have same values.

Examples

```
get_correlation_list(x = matrix(rnorm(5*10),5,10), group_labels = c(rep(1,2),rep(2,3)))
```

---

gnet *Run GNET2*

Description

Build regulation modules by iteratively perform regulator assigning and Gene assigning, until the assignment of genes did not change, or max number of iterations reached.

Usage

```
gnet(
  input,
  reg_names,
  init_method = "boosting",
  init_group_num = 4,
  max_depth = 3,
  cor_cutoff = 0.9,
  min_divide_size = 3,
  min_group_size = 2,
  max_iter = 5,
  heuristic = TRUE,
  max_group = 0,
  force_split = 0.5,
  nthread = 4
)
```

Arguments

input	A SummarizedExperiment object, or a p by n matrix of expression data of p genes and n samples, for example log2 RPKM from RNA-Seq.
reg_names	A list of potential upstream regulators names, for example a list of known transcription factors.
init_method	Cluster initialization, can be "boosting" or "kmeans", default is using "boosting".
init_group_num	Initial number of function clusters used by the algorithm.
max_depth	max_depth Maximum depth of the tree.
cor_cutoff	Cutoff for within group Pearson correlation coefficient, if all data belong to a node have average correlation greater or equal to this, the node would not split anymore.
min_divide_size	Minimum number of data belong to a node allowed for further split of the node.
min_group_size	Minimum number of genes allowed in a group.
max_iter	Maxumum number of iterations allowed if not converged.
heuristic	If the splites of the regression tree is determined by k-means heuristicly.
max_group	Max number of group allowed for the first clustering step, default equals init_group_num and is set to 0.
force_split	Force split the largest gene group into smaller groups by kmeans. Default is 0.5(Split if it contains more than half target genes)
nthread	Number of threads to run GBDT based clustering

Value

A list of expression data of genes, expression data of regulators, within group score, table of tree structure and final assigned group of each gene.

Examples

```
set.seed(1)
init_group_num = 8
init_method = 'boosting'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
```

---

kneepointDetection *Knee point detection.*

Description

Detect the knee point of the array.

Usage

```
kneepointDetection(vect)
```

Arguments

vect	A list of sorted numbers.

Value

The index of the data point which is the knee.

Examples

```
kneepointDetection(sort(c(runif(10,1,3),c(runif(10,5,10))),TRUE))
```

---

plot_gene_group *Plot a module*

Description

Plot the regulators module and heatmap of the expression inferred downstream genes for each sample. It can be interpreted as two parts: the bars at the top shows how samples are splited by the regression tree and the heatmap at the bottom shows how downstream genes are regulated by each subgroup determined by the regulators.

Usage

```
plot_gene_group(
  gnet_result,
  group_idx,
  tree_layout = 1,
  max_gene_num = 100,
  plot_leaf_labels = TRUE,
  group_labels = NULL
)
```

Arguments

gnet_result	Results returned by gnet().
group_idx	Index of the module.
tree_layout	zoom ratio for the regulatory tree. Default is 1. Need to be increased for trees with >5 regulators.
max_gene_num	Max size of gene to plot in the heatmap. Only genes with highest n variances will be kept.
plot_leaf_labels	If the plot includes a color bar of leaf labels at the bottom.
group_labels	Labels of experiment conditions,Used for the color bar of experiment conditions. Default is NULL

Value

None

Examples

```
set.seed(1)
init_group_num = 5
init_method = 'boosting'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
plot_gene_group(gnet_result,group_idx=1)
```

---

plot_group_correlation

Plot the correlation of each group

Description

Plot the correlation of each group and auto detected knee point. It can be used to determined which clustered are kept for further analysis.

Usage

```
plot_group_correlation(gnet_result)
```

Arguments

gnet_result	Results returned by gnet().

Value

A list of indices of the data point with correlation higher than the knee point.

Examples

```
set.seed(1)
gnet_result <- list('group_score'=c(runif(10,1,3),c(runif(10,5,3))))
group_keep <- plot_group_correlation(gnet_result)
```

---

plot_tree *Plot the regression tree.*

Description

Plot the regression tree given the index of a module.

Usage

```
plot_tree(gnet_result, group_idx)
```

Arguments

gnet_result	Results returned by gnet().
group_idx	Index of the module.

Value

None

Examples

```
set.seed(1)
init_group_num = 5
init_method = 'boosting'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
plot_tree(gnet_result,group_idx=1)
```

save_gnet *Save the GNET2 results*

Description

Save the edge list, group index of each gene and plot the top groups

Usage

```
save_gnet(gnet_result, save_path = ".", num_module = 10, max_gene_num = 100)
```

Arguments

gnet_result	Results returned by gnet().
save_path	path to save files
num_module	The number of modules with highest score to plot.
max_gene_num	The max number of genes to show in the heatmap.

Value

None

Examples

```
set.seed(1)
init_group_num = 5
init_method = 'boosting'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
save_gnet(gnet_result)
```

similarity_score *Compute the similarity from a predefined condition group*

Description

Compute the similarity between a predefined condition grouping and the sample cluster of each module, which is defined as the Adjusted Rand index between the two vectors, or the inverse of K-L divergence between the upper triangle matrix of the pairwise distance of predefined ranked condition grouping and the pairwise distance of sample cluster of each module.

Usage

```
similarity_score(gnet_result, group, ranked = FALSE)
```

Arguments

gnet_result	Returned results from gnet().
group	predefined condition grouping
ranked	the grouping information is categorical(treatment/control) or ordinal(dosage, time points)?

Value

A list of similarity scores between a predefined condition grouping and the sample cluster of each module, and the p values for the similarity scores based on permutation.

Examples

```
set.seed(1)
init_group_num = 8
init_method = 'kmeans'
exp_data <- matrix(rnorm(50*10),50,10)
reg_names <- paste0('TF',1:5)
rownames(exp_data) <- c(reg_names,paste0('gene',1:(nrow(exp_data)-length(reg_names))))
colnames(exp_data) <- paste0('condition_',1:ncol(exp_data))
se <- SummarizedExperiment::SummarizedExperiment(assays=list(counts=exp_data))
gnet_result <- gnet(se,reg_names,init_method,init_group_num)
s <- similarity_score(gnet_result,rep(1:5,each = 2))
```

Index