Package ‘gatom’

February 20, 2024

Title Finding an Active Metabolic Module in Atom Transition Network

Version 1.0.0

Description This package implements a metabolic network analysis pipeline to identify an active metabolic module based on high throughput data. The pipeline takes as input transcriptional and/or metabolic data and finds a metabolic subnetwork (module) most regulated between the two conditions of interest. The package further provides functions for module post-processing, annotation and visualization.

biocViews GeneExpression, DifferentialExpression, Pathways, Network

Depends R (>= 4.3.0)

Imports data.table, igraph, BioNet, plyr, methods, XML, sna, intergraph, network, GGally, grid, ggplot2, mwcsr, pryr, htmlwidgets, htmltools, shinyCyJS (>= 1.0.0)

Suggests testthat, knitr, rmarkdown, KEGGREST, AnnotationDbi, org.Mm.eg.db, reactome.db, fgsea, readr, BiocStyle, R.utils

License MIT + file LICENCE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

VignetteBuilder knitr

URL https://github.com/ctlab/gatom/

BugReports https://github.com/ctlab/gatom/issues

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

git_branch RELEASE_3_18

git_last_commit c6257d1

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-02-20
Abbreviate lipid labels for lipid module

Usage

abbreviateLabels(module, orig.names, abbrev.names)
**addHighlyExpressedEdges**

**Arguments**

- **module**: Module to prepare
- **orig.names**: whether to use original names from the dataset
- **abbrev.names**: whether to use abbreviated names for all lipids

**Value**

module object with abbreviated labels

---

**Description**

Add reactions without highly changing genes but with high average expression

**Usage**

```r
addHighlyExpressedEdges(m, g, top = 3000)
```

**Arguments**

- **m**: Metabolic module
- **g**: Scored graph
- **top**: Maximum rank value for the gene to be considered highly expressed

**Value**

module with added edges that correspond to high average expression

**Examples**

```r
data(mEx)
data(gEx)
m <- addHighlyExpressedEdges(m = mEx, g = gEx)
```
connectAtomsIntoMetabolite

collapseAtomsIntoMetabolites

Description

Collapse atoms belonging to the same metabolite into one vertex

Usage

collapseAtomsIntoMetabolites(m)

Arguments

m Metabolic module

Value

module in which atoms of the same metabolite are collapsed into one

Examples

data(mEx)
m <- collapseAtomsIntoMetabolites(m = mEx)

connectAtomsInsideMetabolite

Description

Connect atoms belonging to the same metabolite with edges

Usage

connectAtomsInsideMetabolite(m)

Arguments

m Metabolic module

Value

module in which atoms of the same metabolite are connected
createShinyCyJSWidget

Examples

```r
data(mEx)
m <- connectAtomsInsideMetabolite(m = mEx)
```

---

**createShinyCyJSWidget**  *Creates shinyCyJS widget from module*

**Description**

Creates shinyCyJS widget from module

**Usage**

```r
createShinyCyJSWidget(
  module,
  layout = list(name = "cose-bilkent", animate = FALSE, randomize = FALSE, 
                nodeDimensionsIncludeLabels = TRUE),
  ...
)
```

**Arguments**

- `module`  Module
- `layout`  Layout for the module
- `...`  Other parameters

**Value**

html widget of input module

**Examples**

```r
data(mEx)
hw <- createShinyCyJSWidget(module = mEx)
```
gene.de.rawEx

---

gatom
gatom: a package for finding an active metabolic module in atom transition network

---

Description

This package implements a metabolic network analysis pipeline to identify an active metabolic module based on high throughput data. The pipeline takes as input transcriptional and/or metabolic data and finds a metabolic subnetwork (module) most regulated between the two conditions of interest. The package further provides functions for module post-processing, annotation and visualization.

Functions

Data preprocessing: `prepareDE`, `getMetDEMeta`, `getGeneDEMeta`

Graph creation: `makeMetabolicGraph`

Graph scoring: `scoreGraph`

Module postprocessing: `collapseAtomsIntoMetabolites`, `connectAtomsInsideMetabolite`, `addHighlyExpressedEdges`, `abbreviateLabels`

Plotting module: `createShinyCyJSWidget`

Exporting module: `saveModuleToHtml`, `saveModuleToDot`, `saveModuleToPdf`, `saveModuleToXgmml`

For detailed pipeline analysis, see gatom vignette: `vignette("gatom-tutorial", package = "gatom")`

Example Data

Example data provided by gatom consists of: metabolite differential abundance data (`met.de.rawEx`), gene differential expression data (`gene.de.rawEx`), KEGG-based network object (`networkEx`), KEGG-based metabolite database object (`met.kegg.dbEx`), Example organism annotation object (`org.Mm.eg.gatom.annoEx`), metabolic graph with atom topology (`gEx`), scored metabolic graph with atom topology (`gsEx`), and metabolic module (`mEx`).

---

gene.de.rawEx

Example gene differential expression data.

---

Description


Format

tibble/data.frame object
**getGeneDEMeta**

Finds columns in gene differential expression table required for gatom analysis

**Description**

Default values for all columns are NULL which mean they are determined automatically.

**Usage**

```r
getGeneDEMeta(
  gene.de.raw,
  org.gatom.anno,
  idColumn = NULL,
  idType = NULL,
  pvalColumn = NULL,
  logPvalColumn = NULL,
  log2FCColumn = NULL,
  baseMeanColumn = NULL,
  signalColumn = NULL,
  signalRankColumn = NULL
)
```

**Arguments**

- **gene.de.raw**: A table with differential expression results, an object convertible to data.frame.
- **org.gatom.anno**: Organism-specific annotation obtained from makeOrgGatomAnnotation function.
- **idColumn**: Specifies column name with gene identifiers.
- **idType**: Specifies type of gene IDs (one of the supported by annotation).
- **pvalColumn**: Specifies column with p-values.
- **logPvalColumn**: Specifies column with log p-values, if there is no such column one will be generated automatically.
- **log2FCColumn**: Specifies column with log2-fold changes.
- **baseMeanColumn**: Specifies column with average expression across samples.
- **signalColumn**: Specifies column with identifier of the measured entity (such as gene ID for RNA-seq and probe ID for microarrays). Could be NULL (automatic, set from based on pval and log2FC columns), character (column name), or function (evaluated in a scope of original data frame).
- **signalRankColumn**: Specifies how the genes are ranked from highly to lowly expressed, used in 'addHighlyExpressedEdgues' function. Could be NULL (automatic), character (column name) function (evaluated in a scope of original data frame).
getMetabolicPathways

Value

object with prepared columns for the analysis for gene data

Examples

data("org.Mm.eg.gatom.annoEx")
data("gene.de.rawEx")
de.meta <- getGeneDEMeta(gene.de.rawEx, org.gatom.anno = org.Mm.eg.gatom.annoEx)

getMetabolicPathways Generate list of metabolic pathways from Reactome and KEGG databases

Description

Generate list of metabolic pathways from Reactome and KEGG databases

Usage

gemetabolicPathways(
  universe,
  metGenes,
  keggOrgCode,
  threshold = 0.01,
  includeReactome = TRUE,
  includeKEGG = TRUE
)

Arguments

  universe       list of genes
  metGenes      list of metabolic genes
  keggOrgCode   KEGG organism code, like mmu or hsa
  threshold      threshold for Fisher test to filter out non-metabolic pathways
  includeReactome whether to include Reactome pathways (only works for Entrez ID universe)
  includeKEGG   whether to include KEGG pathways and modules

Value

list of metabolic pathways for given organism and list of genes
getMetDEMeta

Description
Finds columns in differential expression table for metabolites required for gatom analysis

Usage
getMetDEMeta(
  met.de.raw,  # A table with differential expression results, an object convertible to data.frame.
  met.db,  # Metabolite database
  idColumn = NULL,  # Specifies column name with metabolite identifiers.
  idType = NULL,  # Specifies type of metabolite IDs (one of the supported by annotation).
  pvalColumn = NULL,  # Specifies column with p-values.
  logPvalColumn = NULL,  # Specifies column with log p-values, if there is no such column one will be generated automatically.
  log2FCColumn = NULL,  # Specifies column with log2-fold changes.
  signalColumn = NULL  # Specifies column with identifier of the measured entity. Could be NULL (automatic, set from based on pval and log2FC columns), character (column name), or function (evaluated in a scope of original data frame).
)

Arguments

Value
object with prepared columns for the analysis for metabolite data

Examples

data("met.kegg.dbEx")
data("met.de.rawEx")
de.meta <- getMetDEMeta(met.de.rawEx, met.db = met.kegg.dbEx)
Example metabolic graph with atom topology.

Description

Format
igraph object

Example scored metabolic graph with atom topology.

Description

Format
igraph object

createMetabolicGraph

Description
Creates metabolic graph based on specified data

Usage
makeMetabolicGraph(
  network,
  topology = c("atoms", "metabolites"),
  org.gatom.anno,
  gene.de,
  gene.de.meta = getGeneDEMeta(gene.de, org.gatom.anno),
  gene.keep.top = 12000,
  met.db,
  met.de,
  met.de.meta = getMetDEMeta(met.de, met.db),
  met.to.filter = fread(system.file("extdata","mets2mask.lst", package = "gatom"))$ID,
  gene2reaction.extra = NULL,
  keepReactionsWithoutEnzymes = FALSE,
  largest.component = TRUE
)


\texttt{makeMetabolicGraph}

**Arguments**

- \texttt{network} \hspace{1cm} Network object
- \texttt{topology} \hspace{1cm} Way to determine network vertices
- \texttt{org.gatom.anno} \hspace{1cm} Organism annotation object
- \texttt{gene.de} \hspace{1cm} Table with the differential gene expression, set to NULL if absent
- \texttt{gene.de.meta} \hspace{1cm} Annotation of \texttt{‘gene.de’} table
- \texttt{gene.keep.top} \hspace{1cm} Only the \texttt{‘gene.keep.top’} of the most expressed genes will be kept for the network
- \texttt{met.db} \hspace{1cm} Metabolite database
- \texttt{met.de} \hspace{1cm} Table with the differential expression for metabolites, set to NULL if absent
- \texttt{met.de.meta} \hspace{1cm} Annotation of \texttt{‘met.de’} table
- \texttt{met.to.filter} \hspace{1cm} List of metabolites to filter from the network
- \texttt{gene2reaction.extra} \hspace{1cm} Additional gene to reaction mappings. Should be a data.table with \texttt{‘gene’} and \texttt{‘reaction’} columns
- \texttt{keepReactionsWithoutEnzymes} \hspace{1cm} If TRUE, keep reactions that have no annotated enzymes, thus expanding the network but including some reactions which are not possible in the considered species.
- \texttt{largest.component} \hspace{1cm} If TRUE, only the largest connected component is returned

**Value**

\texttt{igraph} object created from input data

**Examples**

```r
data("gene.de.rawEx")
data("met.de.rawEx")
data("met.kegg.dbEx")
data("networkEx")
data("org.Mm.eg.gatom.annoEx")
g <- makeMetabolicGraph(network = networkEx, topology = "atoms",
                        org.gatom.anno = org.Mm.eg.gatom.annoEx,
                        gene.de = gene.de.rawEx, met.db = met.kegg.dbEx,
                        met.de = met.de.rawEx)
```
makeOrgGatomAnnotation

Create an organism annotation object for network analysis

Description

Create an organism annotation object for network analysis

Usage

makeOrgGatomAnnotation(
  org.db,
  idColumns = c(Entrez = "ENTREZID", RefSeq = "REFSEQ", Ensembl = "ENSEMBL", Symbol = "SYMBOL"),
  nameColumn = "SYMBOL",
  enzymeColumn = "ENZYME",
  appendEnzymesFromKegg = TRUE,
  appendOrthologiesFromKegg = TRUE,
  filterNonSpecificEnzymes = TRUE,
  keggOrgCode = NULL
)

Arguments

org.db Bioconductor org.db object, e.g. org.Mm.eg.db
idColumns vector of column names from `org.db` object to creat ID mappings. First ID will be used as a base identifier, should be compatible with KEGG and Reactome databases.
nameColumn column with a human readable gene symbol. Default to "SYMBOL".
enzymeColumn column with an Enzyme Commission ID. Default to "ENZYME".
appendEnzymesFromKegg
  if TRUE, KEGG databases will be sued to extend gene to enzyme mappings obtained from org.db package.
appendOrthologiesFromKegg
  if TRUE, KEGG database will be sued to extend gene to orthology mappings obtained from org.db package
filterNonSpecificEnzymes
  if TRUE, will filter out non-specific enzymes from gene to enzyme mappings obtained from org.db package
keggOrgCode KEGG organism code, e.g. "mmu". If set to NULL, the code is determined automatically.

Value

organism annotation object that will be used for network analysis
Examples

```r
library(org.Mm.eg.db)
org.Mm.eg.gatom.anno <- makeOrgGatomAnnotation(org.db = org.Mm.eg.db)
```

met.de.rawEx  
Example metabolite differential abundance data.

Description


Format

tibble/data.frame object

met.kegg.dbEx  
Example KEGG-based metabolite database object

Description


Format

list object

mEx  
Example metabolic module.

Description


Format

igraph object
### prepareDE

Makes data.table with differential expression results containing all columns required for gatom and in the expected format based on metadata object

**Usage**

```r
prepareDE(de.raw, de.meta)
```

**Arguments**

- `de.raw`: Table with differential expression results, an object convertable to data.frame
- `de.meta`: Object with differential expression table metadata acquired with `getGeneDEMeta` or `getMetDEMeta` functions
**Value**

data.table object with converted differential expression table

**Examples**

data("org.Mm.eg.gatom.annoEx")
data("gene.de.rawEx")
demeta <- getGeneDEMeta(gene.de.rawEx, org.gatom.anno = org.Mm.eg.gatom.annoEx)
de <- prepareDE(gene.de.rawEx, de.meta)

**saveModuleToDot**  
*Save module to a graphviz dot file*

**Description**

Save module to a graphviz dot file

**Usage**

```r
saveModuleToDot(
  module,  
  file,  
  name = NULL,  
  extra.node.attrs = NULL,  
  extra.edge.attrs = NULL
)
```

**Arguments**

- `module`  
  Module to save
- `file`  
  File to save to
- `name`  
  Name of the module
- `extra.node.attrs`  
  Table with additional node attributes to be written to the dot file as is
- `extra.edge.attrs`  
  Table with additional edge attributes to be written to the dot file as is

**Value**

Returns NULL

**Examples**

data(mEx)
saveModuleToDot(module = mEx, file = "module.dot")
saveModuleToHtml

Save module to a html widget

Description
Save module to a html widget

Usage

```
saveModuleToHtml(
  module,
  file,
  name = "",
  sizingPolicy = htmlwidgets::sizingPolicy(defaultWidth = "100%", defaultHeight = "90vh", padding = 10),
  ...)
```

Arguments

- **module**: Module to save
- **file**: File to save to
- **name**: Name of the module
- **sizingPolicy**: A widget sizing policy
- **...**: Other parameters

Value

Returns NULL

Examples

```
data(mEx)
saveModuleToHtml(module = mEx, file = "module.html")
```

saveModuleToPdf

Save module to a nice pdf file

Description
Save module to a nice pdf file
saveModuleToPdf

Usage

saveModuleToPdf(module, file, name = NULL, n_iter = 100, force = 1e-05)

Arguments

- module: Module to save
- file: File to save to
- name: Name of the module
- n_iter: Number of repel algorithm iterations
- force: Value of repel force

Value

Returns NULL

Examples

data(mEx)
saveModuleToPdf(module = mEx, file = "module.pdf")

saveModuleToXgmml

Description

Save module to an XGMML file

Usage

saveModuleToXgmml(module, file, name = NULL)

Arguments

- module: Module to save
- file: File to save to
- name: Name of the module

Value

Returns NULL

Examples

data(mEx)
saveModuleToXgmml(module = mEx, file = "module.xgmml"
scoreGraph  

Description

Score metabolic graph

Usage

scoreGraph(
  g,
  k.gene,
  k.met,
  vertex.threshold.min = 0.1,
  edge.threshold.min = 0.1,
  met.score.coef = 1,
  show.warnings = TRUE,
  raw = FALSE
)

Arguments

- **g**: Metabolic graph obtained with makeMetabolicGraph function
- **k.gene**: Number of gene signals to be scored positively, the higher is the number, the larger will be the resulting module. If set to NULL, genes will not be used for scoring.
- **k.met**: Number of metabolite signals to be scored positively, the higher is the number, the larger will be the resulting module. If set to NULL, metabolites will not be used for scoring.
- **vertex.threshold.min**: The worst acceptable estimated FDR for vertices. If necessary number of positive metabolite signals will be decreased from `k.met` to reach this threshold. Default value is 0.1.
- **edge.threshold.min**: The worst acceptable estimated FDR for vertices. If necessary number of positive metabolite signals will be decreased from `k.gene` to reach this threshold. Default value is 0.1.
- **met.score.coef**: Coefficient on which all vertex weights are multiplied. Can be used to balance vertex and edge weights. Default values is 1.
- **show.warnings**: whether to show warnings
- **raw**: whether to return raw scored graph, not a SGMWCS instance. Default to FALSE.

Value

SGMWCS instance or scored igraph object
Examples

data("gEx")
gs <- scoreGraph(g = gEx, k.gene = 25, k.met = 25)

Description

code adopted from https://github.com/ramnathv/htmlwidgets/issues/231

Usage

styleWidget(hw, style = "", addl_selector = "", elementId = NULL)

Value

styled html widget
Index

* internal
  styleWidget, 19

abbreviateLabels, 2, 6
addHighlyExpressedEdges, 3, 6
collapseAtomsIntoMetabolites, 4, 6
connectAtomsInsideMetabolite, 4, 6
createShinyCyJSWidget, 5, 6
gatom, 6
gene.de.rawEx, 6, 6
getGeneDEMeta, 6, 7
gemetabolicPathways, 8
getMetDEMeta, 6, 9
gEx, 6, 10
gsEx, 6, 10
makeMetabolicGraph, 6, 10
makeOrgGatomAnnotation, 12
met.de.rawEx, 6, 13
met.kegg.dbEx, 6, 13
mEx, 6, 13

networkEx, 6, 14
org.Mm.eg.gatom.annoEx, 6, 14
prepareDE, 6, 14

saveModuleToDot, 6, 15
saveModuleToHtml, 6, 16
saveModuleToPdf, 6, 16
saveModuleToXgmm1, 6, 17
scoreGraph, 6, 18
styleWidget, 19