Package ‘NetPathMiner’

May 15, 2025

Version 1.44.0

Date 2014 onwards

Title NetPathMiner for Biological Network Construction, Path Mining and Visualization

Description NetPathMiner is a general framework for network path mining using genome-scale networks. It constructs networks from KGML, SBML and BioPAX files, providing three network representations, metabolic, reaction and gene representations. NetPathMiner finds active paths and applies machine learning methods to summarize found paths for easy interpretation. It also provides static and interactive visualizations of networks and paths to aid manual investigation.

Depends R (>= 3.0.2), igraph (>= 1.0)

Suggests rBiopaxParser (>= 2.1), RCurl, graph, knitr, rmarkdown, BiocStyle

VignetteBuilder knitr

License GPL (>= 2)

URL https://github.com/ahmohamed/NetPathMiner

NeedsCompilation yes

SystemRequirements libxml2, libSBML (>= 5.5)

Biarch TRUE

biocViews GraphAndNetwork, Pathways, Network, Clustering, Classification

RoxygenNote 7.2.3

Encoding UTF-8

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

git_branch RELEASE_3_21

git_last_commit b4383b8

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-05-14

Author Ahmed Mohamed [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6507-5300>),
Tim Hancock [aut],
Tim Hancock [aut]

Maintainer Ahmed Mohamed <mohamed@kuicr.kyoto-u.ac.jp>

Contents

`NetPathMiner-package` *General framework for network extraction, path mining.*

Description

NetPathMiner implements a flexible module-based process flow for network path mining and visualization, which can be fully inte-grated with user-customized functions. NetPathMiner supports construction of various types of genome scale networks from KGML, SBML and BioPAX formats, enabling its utility to most common pathway databases. NetPathMiner also provides different visualization techniques to facilitate the analysis of even thousands of output paths.

Author(s)

Ahmed Mohamed <mohamed@kuicr.kyoto-u.ac.jp>

`assignEdgeWeights` *Assigning weights to network edges*

Description

This function computes edge weights based on a gene expression profile.

Usage

```
assignEdgeWeights(
  microarray,
  graph,
  use.attr,
  y,
  weight.method = "cor",
  complex.method = "max",
  missing.method = "median",
  same.gene.penalty = "median",
  bootstrap = 100,
  verbose = TRUE
)
```

Arguments

microarray	Microarray should be a Dataframe or a matrix, with genes as rownames, and samples as columns.
graph	An annotated igraph object.
use.attr	An attribute name to map microarray rows (genes) to graph vertices. The attribute must be annotated in graph, and the values correspond to rownames of microarray. You can check the coverage and if there are complex vertices using getAttrStatus. You can eliminate complexes using expandComplexes.
y	Sample labels, given as a factor or a character vector. This must be the same size as the columns of microarray
weight.method	A function, or a string indicating the name of the function to be used to compute the edge weights. The function is provided with 2 numerical verctors (2 rows from microarray), and it should return a single numerical value (or NA). The default computes Pearson's correlation.
complex.method	A function, or a string indicating the name of the function to be used in weighting edges connecting complexes. If a vertex has >1 attribute value, all possible pairwise weights are first computed, and given to complex.method. The default function is max.
missing.method	A function, or a string indicating the name of the function to be used in weighting edges when one of the vertices lack expression data. The function is passed all edge weights on the graph. Default is median.
same.gene.penalty	A numerical value to be assigned when 2 adjacent vertices have the same attribute value, since correlation and similarity measure will give perfect scores. Alternatively, same.gene.penalty can be a function, computing the penalty from all edge weights on the graph (excluding same-gene and missing values). The default is to take the median
bootstrap	An integer n, where the weight.method is perfomed on n permutations of the gene profiles, and taking the median value. Set it to NA to disable bootstrapping.
verbose	Print the progress of the function.

Value

The input graph with edge.weight as an edge attribute. The attribute can be a list of weights if y labels were provided.

Author(s)

Ahmed Mohamed

Examples

```
## Convert a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)
```

```
## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

# Using Spearman correlation, assigning missing edges to -1
## Not run:
  assignEdgeWeights(microarray, graph, use.attr="miriam.affy.probeset",
      y=factor(colnames(microarray)),
      weight.method = function(x1,x2) cor(x1,x2, method="spearman"),
      missing.method = -1)

## End(Not run)
```

`biopax2igraph` *Processes BioPAX objects into igraph objects*

Description

This function takes BioPAX objects (level 2 or 3) as input, and returns either a metabolic or a signaling network as output.

Usage

```
biopax2igraph(
  biopax,
  parse.as = c("metabolic", "signaling"),
  expand.complexes = FALSE,
  inc.sm.molecules = FALSE,
  verbose = TRUE
)
```

Arguments

`biopax` BioPAX object generated by `readBiopax`.

`parse.as` Whether to process file into a metabolic or a signaling network.

`expand.complexes`
Split protein complexes into individual gene nodes. Ignored if `parse.as="metabolic"`.

`inc.sm.molecules`
Include small molecules that are participating in signaling events. Ignored if `parse.as="metabolic"`.

`verbose` Whether to display the progress of the function.

Details

This function requires rBiopaxParser installed.

Users can specify whether files are processes as metabolic or signaling networks.

Metabolic networks are given as bipartite graphs, where metabolites and reactions represent vertex types. Reactions are constructed from Conversion classes, connecting them to their corresponding Lefts and Rights. Each reaction vertex has genes attribute, listing all Catalysis relationships of this reaction. As a general rule, reactions inherit all annotation attributes of its catalyzig genes.

Signaling network have genes as vertices and edges represent interactions, such as activiation / inhibition. Genes participating in successive reactions are also connected. Signaling interactions are constructed from Control classes, where edges are drawn from controller to controlled.

All annotation attributes are exracted from XRefs associated with the vertices, and are stored according to MIRIAM guidelines (miraim.db, where db is the database name).

Value

An igraph object, representing a metbolic or a signaling network.

Author(s)

Ahmed Mohamed

See Also

Other Database extraction methods: KGML2igraph(), SBML2igraph()

Examples

```
if(requireNamespace("rBiopaxParser")){
    data(ex_biopax)
    # Process biopax as a metabolic network
    g <- biopax2igraph(ex_biopax)
    plotNetwork(g)

    # Process SBML file as a signaling network
    g <- biopax2igraph(ex_biopax, parse.as="signaling", expand.complexes=TRUE)
}
```

---

colorVertexByAttr *Computes colors for vertices according to their attributes.*

Description

This function returns a list of colors for vertices, assigned similar colors if they share a common attribute (ex: in the same pathway, etc).

Usage

```
colorVertexByAttr(graph, attr.name, col.palette = palette())
```

Arguments

graph	An annotated igraph object.
attr.name	The attribute name (ex: "pathway") by which vertices will be colored. Complex attributes, where a vertex belongs to more than one group, are supported.
col.palette	A color palette, or a palette generating function (ex: `col.palette=rainbow`).

Value

A list of colors (in HEX format) for vertices.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotCytoscapeGML()`, `plotNetwork()`, `plotPathClassifier()`, `plotPaths()`

Examples

```
data("ex_kgml_sig")
v.colors <- colorVertexByAttr(ex_kgml_sig, "pathway")
plotNetwork(ex_kgml_sig, vertex.color=v.colors)
```

---

expandComplexes *Expand reactions / complexes into their gene constituents.*

Description

These are general functions to expand vertices by their attributes, i.e. create a separate vertex for each attribute value.

Usage

```
expandComplexes(
  graph,
  v.attr,
  keep.parent.attr = "^pathway",
  expansion.method = c("normal", "duplicate"),
  missing.method = c("keep", "remove", "reconnect")
)

makeGeneNetwork(
  graph,
  v.attr = "genes",
  keep.parent.attr = "^pathway",
  expansion.method = "duplicate",
  missing.method = "remove"
)
```

Arguments

`graph` An annotated igraph object.

`v.attr` Name of the attribute which vertices are expanded to.

`keep.parent.attr` A (List of) `regex` experssions representing attributes to be inherited by daughter vertices. If `"all"` is passed, all parent attributes are inherited.

`expansion.method` If `"duplicate"`, attribute values sharing more than one parent vertex are duplicated for each vertex they participate in. For exmaple, if one gene G1 catalyzes reactions R1, R2; then G1##R1, and G1##R2 vertices are created. If `"normal"` only one vertex (G1) is created, and inherit all R1 and R2 connections and attributes.

`missing.method` How to deal with vertices with no attribute values. `"keep"` retains the parent node, `"remove"` simply deletes the vertex, and `"reconnect"` removes the vertex and connect its neighbours to each other (to prevent graph cuts).

Details

These functions can be very useful when merging networks constructed from different databases. For example, to match a network created from Reactome to a KEGG network, you can expand metabolite vertices by "miriam.kegg.compound" attribute.

Value

A new graph with vertices expanded.

`makeGeneNetwork` returns a graph, where nodes are genes, and edges represent participation in succesive reactions.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: makeMetaboliteNetwork(), makeReactionNetwork(), reindexNetwork(), rmSmallCompounds(), simplifyReactionNetwork(), vertexDeleteReconnect()

Examples

```
## Make a gene network from a reaction network.
data(ex_sbml) # A bipartite metbaolic network.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)
ggraph <- makeGeneNetwork(rgraph)

## Expand vertices into their contituent genes.
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
ggraph <- expandComplexes(ex_kgml_sig, v.attr = "miriam.ncbigene",
     keep.parent.attr= c("^pathway", "^compartment"))

## Create a separate vertex for each compartment. This is useful in duplicating
##  metabolite vertices in a network.
## Not run:
graph <- expandComplexes(graph, v.attr = "compartment",
       keep.parent.attr = "all",
       expansion.method = "duplicate",
       missing.method = "keep")

## End(Not run)
```

---

extractPathNetwork — *Creates a subnetwork from a ranked path list*

Description

Creates a subnetwork from a ranked path list generated by pathRanker.

Usage

```
extractPathNetwork(paths, graph)
```

Arguments

paths	The paths extracted by pathRanker.
graph	A annotated igraph object.

Value

A subnetwork from all paths provided. If paths are computed for several labels (sample categories), a subnetwork is returned for each label.

Author(s)

Ahmed Mohamed

See Also

Other Path ranking methods: getPathsAsEIDs(), pathRanker()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Get the subnetwork of paths in reaction graph.
reaction.sub <- getPathsAsEIDs(ranked.p, rgraph)

## Get the subnetwork of paths in the original metabolic graph.
metabolic.sub <- getPathsAsEIDs(ranked.p, ex_sbml)
```

---

ex_biopax *Biopax example data*

Description

A dataset containing Porphyrin metabolism pathway in Biopax Level 3 and parsed with readBiopax.

Examples

```
data(ex_biopax)
ex_biopax
```

ex_kgml_sig *Singaling network from KGML example*

Description

An example igraph object representing Ras and chemokine signaling pathways in human extracted from KGML files.

Examples

```
data(ex_kgml_sig)
plotNetwork(ex_kgml_sig, vertex.color="pathway")
```

ex_microarray *An microarray data example.*

Description

An microarray data example. This is part of the ALL dataset, for demonstration purposes.

Examples

```
data(ex_microarray)
```

ex_sbml *Metabolic network from SBML example*

Description

An example igraph object representing bipartite metabolic network of Carbohydrate metabolism extracted from SBML file from Reactome database.

Examples

```
data(ex_sbml)
plotNetwork(ex_sbml, vertex.color="compartment.name")
```

`getAttrStatus` *Get / Set vertex attribute names and coverage*

Description

These functions report the annotation status of the vertices of a given network, modify or remove certain annotations.

Usage

```
getAttrStatus(graph, pattern = "^miriam.")

getAttrNames(graph, pattern = "")

getAttribute(graph, attr.name)

setAttribute(graph, attr.name, attr.value)

rmAttribute(graph, attr.name)
```

Arguments

`graph`	An annotated igraph object.
`pattern`	A `regex` experssion representing attribute name pattern.
`attr.name`	The attribute name
`attr.value`	A list of attribute values. This must be the same size as the number of vertices.

Details

NetPathMiner stores all its vertex annotation attributes in a list, and stores them collectively as a single `attr`. This is not to interfer with `graph_attr_names` from `igraph` package. All functions here target NetPathMiner annotations only.

Value

For `getAttrStatus`, a dataframe summarizing the number of vertices with no (`missing`), one (`single`) or more than one (`complex`) attribute value. The coverage

For `getAttrNames`, a character vector of attribute names matching the pattern.

For `getAttribute`, a list of vertex annotation values for the query attribute.

For `setAttribute`, a graph with the new attribute set.

For `rmAttrNames`, a new igraph object with the attibute removed.

Author(s)

Ahmed Mohamed

See Also

Other Attribute handling methods: `stdAttrNames()`

Examples

```
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human

# Get status of attribute "pathway" only
getAttrStatus(ex_kgml_sig, "^pathway$")

# Get status of all attributes  starting with "pathway" and "miriam" keywords
getAttrStatus(ex_kgml_sig, "(^miriam)|(^pathway)")

# Get all attribute names containing "miriam"
getAttrNames(ex_kgml_sig, "miriam")
# Get all attribute names containing "miriam"
getAttribute(ex_kgml_sig, "miriam.ncbigene")

# Remove an attribute from graph
graph <- rmAttribute(ex_kgml_sig, "miriam.ncbigene")
```

---

getGeneSetNetworks *Generate geneset networks from an annotated network.*

Description

This function generates geneset networks based on a given netowrk, by grouping vertices sharing common attributes (in the same pathway or compartment).

Usage

```
getGeneSetNetworks(
  graph,
  use.attr = "pathway",
  format = c("list", "pathway-class")
)
```

Arguments

`graph`	An annotated igraph object..
`use.attr`	The attribute by which vertices are grouped (tepically pathway, or GO)
`format`	The output format. If "list" is specified, a list of subgraphs are returned (default). If "pathway-class" is specified, a list of pathway-class objects are returned. Pathway-class is used by graphite package to run several methods of topology-based enrichment analyses.

Value

A list of geneset networks as igraph or Pathway-class objects.

Author(s)

Ahmed Mohamed

See Also

getGeneSets

Examples

```
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
genesetnets <- getGeneSetNetworks(ex_kgml_sig, use.attr="pathway")

# Integration with graphite package
## Not run:
if(requireNamespace("graphite") & requireNamespace("clipper") & requireNamespace("ALL")){
genesetnets <- getGeneSetNetworks(ex_kgml_sig,
     use.attr="pathway", format="pathway-class")
path <- convertIdentifiers(genesetnets$`Chemokine signaling pathway`,
     "entrez")
genes <- nodes(path)
data(ALL)
all <- as.matrix(exprs(ALL[1:length(genes),1:20]))
classes <- c(rep(1,10), rep(2,10))
rownames(all) <- genes

runClipper(path, all, classes, "mean", pathThr=0.1)
}

## End(Not run)
```

---

getGeneSets *Generate genesets from an annotated network.*

Description

This function generates genesets based on a given netowrk, by grouping vertices sharing common attributes (in the same pathway or compartment). Genes associated with each vertex can be specified through gene.attr argument.

Usage

```
getGeneSets(graph, use.attr = "pathway", gene.attr = "genes", gmt.file)
```

Arguments

graph	An annotated igraph object..
use.attr	The attribute by which vertices are grouped (tepically pathway, or GO)
gene.attr	The attribute listing genes annotated with each vertex (ex: miriam.ncbigene, miriam.uniprot, ...)
gmt.file	Optinal. If provided, Results are exported to a GMT file. GMT files are readily used by most gene set analysis packages.

Value

A list of genesets or written to gmt file if provided.

Author(s)

Ahmed Mohamed

See Also

getGeneSetNetworks

Examples

```
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
genesets <- getGeneSets(ex_kgml_sig, use.attr="pathway", gene.attr="miriam.ncbigene")


# Write the genesets in a GMT file, and read it using GSEABase package.
getGeneSets(ex_kgml_sig, use.attr="pathway", gene.attr="miriam.ncbigene", gmt.file="kgml.gmt")
## Not run:
if(requireNamespace("GSEABase"))
 toGmt("kgml.gmt")

## End(Not run)

# Create genesets using compartment information
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
genesets <- getGeneSets(ex_sbml, use.attr="compartment.name", gene.attr="miriam.uniprot")
```

---

getPathsAsEIDs *Convert a ranked path list to edge ids of a graph*

Description

Convert a ranked path list to Edge ids of a graph, where paths can come from a different representation (for example matching path from a reaction network to edges on a metabolic network).

Usage

```
getPathsAsEIDs(paths, graph)
```

Arguments

`paths`	The paths extracted by `pathRanker`.
`graph`	A annotated igraph object.

Value

A list of edge ids on the provided graph.

Author(s)

Ahmed Mohamed

See Also

Other Path ranking methods: `extractPathNetwork()`, `pathRanker()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Get the edge ids along paths in the reaction graph.
path.eids <- getPathsAsEIDs(ranked.p, rgraph)

## Get the edge ids along paths in the original metabolic graph.
path.eids <- getPathsAsEIDs(ranked.p, ex_sbml)
```

KGML2igraph *Processes KGML files into igraph objects*

Description

This function takes KGML files as input, and returns either a metabolic or a signaling network as output.

Usage

```
KGML2igraph(
  filename,
  parse.as = c("metabolic", "signaling"),
  expand.complexes = FALSE,
  verbose = TRUE
)
```

Arguments

filename	A character vector containing the KGML files to be processed. If a directory path is provided, all *.xml files in it and its subdirectories are included.
parse.as	Whether to process file into a metabolic or a signaling network.
expand.complexes	Split protein complexes into individual gene nodes. This argument is ignored if `parse.as="metabolic"`
verbose	Whether to display the progress of the function.

Details

Users can specify whether files are processes as metabolic or signaling networks.

Metabolic networks are given as bipartite graphs, where metabolites and reactions represent vertex types. This is constructed from <reaction> xml node in KGML file, connecting them to their corresponding substrates and products. Each reaction vertex has `genes` attribute, listing all genes associated with the reaction. As a general rule, reactions inherit all annotation attributes of its catalyzig genes.

Signaling network have genes as vertices and edges represent interactions, such as activiation / inhibition. Genes participating in successive reactions are also connected. Signaling parsing method processes <ECrel>, <PPrel> and <PCrel> interactions from KGML files.

To generate a genome scale network, simply provide a list of files to be parsed, or put all file in a directory, as pass the directory path as `filename`

Value

An igraph object, representing a metbolic or a signaling network.

Author(s)

Ahmed Mohamed

See Also

Other Database extraction methods: SBML2igraph(), biopax2igraph()

Examples

```
if(is.loaded("readkgmlfile")){ # This is false if libxml2 wasn't available at installation.
    filename <- system.file("extdata", "hsa00860.xml", package="NetPathMiner")

    # Process KGML file as a metabolic network
    g <- KGML2igraph(filename)
    plotNetwork(g)

    # Process KGML file as a signaling network
    g <- KGML2igraph(filename, parse.as="signaling", expand.complexes=TRUE)
    plotNetwork(g)
}
```

---

layoutVertexByAttr *A graph layout function, which groups vertices by attribute.*

Description

This function generates a layout for igraph objects, keeping vertices with the same attribute (ex: in the same pathway, etc) close to each other.

Usage

```
layoutVertexByAttr(
  graph,
  attr.name,
  cluster.strength = 1,
  layout = layout.auto
)
```

Arguments

graph	An annotated igraph object.
attr.name	The attribute name by which vertices are laid out.
cluster.strength	A number indicating tie strengths between vertices with the same attribute. The larger it is, the closer the vertices will be.
layout	A layout function, ideally a force-directed layout fuction, such as layout_with_fr and layout_with_kk.

Value

A two-column matrix indicating the x and y postions of vertices.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: colorVertexByAttr(), plotAllNetworks(), plotClassifierROC(), plotClusterMatrix(), plotCytoscapeGML(), plotNetwork(), plotPathClassifier(), plotPaths()

Examples

```
data("ex_kgml_sig")
v.layout <- layoutVertexByAttr(ex_kgml_sig, "pathway")
plotNetwork(ex_kgml_sig, vertex.color="pathway", layout=v.layout)

v.layout <- layoutVertexByAttr(ex_kgml_sig, "pathway", cluster.strength=5)
plotNetwork(ex_kgml_sig, vertex.color="pathway", layout=v.layout)
```

---

makeMetaboliteNetwork *Convert metabolic network to metabolite network.*

Description

This function removes reaction nodes keeping them as edge attributes. The resulting network contains metabolite nodes only, where edges indicate that reaction conversions.

Usage

```
makeMetaboliteNetwork(graph)
```

Arguments

graph	A metabolic network.

Value

A reaction network.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: expandComplexes(), makeReactionNetwork(), reindexNetwork(), rmSmallCompounds(), simplifyReactionNetwork(), vertexDeleteReconnect()

Examples

```
## Conver a metabolic network to a metbolite network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
mgraph <- makeMetaboliteNetwork(ex_sbml)
```

---

makeReactionNetwork *Convert metabolic network to reaction network.*

Description

This function removes metabolite nodes keeping them as edge attributes. The resulting network contains reaction nodes only, where edges indicate that a metabolite produced by one reaction is consumed by the other.

Usage

```
makeReactionNetwork(graph, simplify = FALSE)
```

Arguments

graph	A metabolic network.
simplify	An option to remove translocation and spontaneous reactions that require no catalyzing genes. Translocation reactions are detected from reaction name (SBML, BioPAX), or by having identical substrates and products.

Value

A reaction network.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: expandComplexes(), makeMetaboliteNetwork(), reindexNetwork(), rmSmallCompounds(), simplifyReactionNetwork(), vertexDeleteReconnect()

Examples

```
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)
```

`NPMdefaults` *Default values for NetPathMiner*

Description

This function gets a NetPathMiner default value for a variable.

Usage

```
NPMdefaults(value)
```

Arguments

`value`	a character string indicating the variable name.

Details

NetPathMiner defines the following defaults:

- small.comp.ls Dataframe of ubiquitous metabolites. Used by `rmSmallCompounds`.
- bridge Dataframe of attributes supported by Brigde Database. Used by `fetchAttribute`.
- bridge.organisms A list of bridge supported organisms. Used by `fetchAttribute`.
- bridge.web The base URL for Brigde Database webservices. Used by `fetchAttribute`.

Value

The defuult value for the given variable.

Author(s)

Ahmed Mohamed

Examples

```
# Get the default list of small compounds (uniqitous metabolites).
NPMdefaults("small.comp.ls")
```

pathClassifier *HME3M Markov pathway classifier.*

Description

HME3M Markov pathway classifier.

Usage

```
pathClassifier(
  paths,
  target.class,
  M,
  alpha = 1,
  lambda = 2,
  hme3miter = 100,
  plriter = 1,
  init = "random"
)
```

Arguments

`paths`	The training paths computed by `pathsToBinary`
`target.class`	he label of the targe class to be classified. This label must be present as a label within the `paths\$y` object
`M`	Number of components within the paths to be extracted.
`alpha`	The PLR learning rate. (between 0 and 1).
`lambda`	The PLR regularization parameter. (between 0 and 2)
`hme3miter`	Maximum number of HME3M iterations. It will stop when likelihood change is < 0.001.
`plriter`	Maximum number of PLR iteractions. It will stop when likelihood change is < 0.001.
`init`	Specify whether to initialize the HME3M responsibilities with the 3M model - random is recommended.

Details

Take care with selection of lambda and alpha - make sure you check that the likelihood is always increasing.

Value

A list with the following elements. A list with the following values

`h`	A dataframe with the EM responsibilities.

`theta`	A dataframe with the Markov parameters for each component.
`beta`	A dataframe with the PLR coefficients for each component.
`proportions`	The probability of each HME3M component.
`posterior.probs`	The HME3M posterior probability.
`likelihood`	The likelihood convergence history.
`plrplr`	The posterior predictions from each components PLR model.
`path.probabilities`	The 3M probabilities for each path belonging to each component.
`params`	The parameters used to build the model.
`y`	The binary response variable used by HME3M. A 1 indicates the location of the target.class labels in `paths\$y`
`perf`	The training set ROC curve AUC.
`label`	The HME3M predicted label for each path.
`component`	The HME3M component assignment for each path.

Author(s)

Timothy Hancock and Ichigaku Takigawa

References

Hancock, Timothy, and Mamitsuka, Hiroshi: A Markov Classification Model for Metabolic Pathways, Workshop on Algorithms in Bioinformatics (WABI) , 2009

Hancock, Timothy, and Mamitsuka, Hiroshi: A Markov Classification Model for Metabolic Pathways, Algorithms for Molecular Biology 2010

See Also

Other Path clustering & classification methods: `pathCluster()`, `pathsToBinary()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotPathClassifier()`, `plotPathCluster()`, `predictPathClassifier()`, `predictPathCluster()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)
```

```
## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.class <- pathClassifier(ybinpaths, target.class = "BCR/ABL", M = 3)

## Contingency table of classification performance
table(ybinpaths$y,p.class$label)

## Plotting the classifier results.
plotClassifierROC(p.class)
plotClusters(ybinpaths, p.class)
```

`pathCluster` *3M Markov mixture model for clustering pathways*

Description

3M Markov mixture model for clustering pathways

Usage

```
pathCluster(ybinpaths, M, iter = 1000)
```

Arguments

`ybinpaths`	The training paths computed by `pathsToBinary`.
`M`	The number of clusters.
`iter`	The maximum number of EM iterations.

Value

A list with the following items:

`h`	The posterior probabilities that each path belongs to each cluster.
`labels`	The cluster membership labels.
`theta`	The probabilities of each gene for each cluster.
`proportions`	The mixing proportions of each path.
`likelihood`	The likelihood convergence history.
`params`	The specific parameters used.

Author(s)

Ichigaku Takigawa

Timothy Hancock

References

Mamitsuka, H., Okuno, Y., and Yamaguchi, A. 2003. Mining biologically active patterns in metabolic pathways using microarray expression profiles. SIGKDD Explor. News l. 5, 2 (Dec. 2003), 113-121.

See Also

Other Path clustering & classification methods: `pathClassifier()`, `pathsToBinary()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotPathClassifier()`, `plotPathCluster()`, `predictPathClassifier()`, `predictPathCluster()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot", bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=8)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.cluster <- pathCluster(ybinpaths, M=2)
plotClusters(ybinpaths, p.cluster)
```

`pathRanker` *Extracting and ranking paths from a network*

Description

Given a weighted igraph object, path ranking finds a set of node/edge sequences (paths) to maximize the sum of edge weights. `pathRanker(method="prob.shortest.path")` extracts the K most probable paths within a weighted network. `pathRanker(method="pvalue")` extracts a list of paths whose sum of edge weights are significantly higher than random paths of the same length.

Usage

```
pathRanker(
  graph,
```

```
    method = "prob.shortest.path",
    start,
    end,
    verbose = TRUE,
    ...
)
```

Arguments

graph	A weighted igraph object. Weights must be in `edge.weights` or `weight` edge attributes.
method	Which path ranking method to use.
start	A list of start vertices, given by their vertex id.
end	A list of terminal vertices, given by their vertex id.
verbose	Whether to display the progress of the function.
...	Method-specific parameters. See Details section.

Details

The input here is `graph`. A weight must be assigned to each edge. Bootstrapped Pearson correlation edge weights can be assigned to each edge by `assignEdgeWeights`. However the specification of the edge weight is flexible with the condition that increasing values indicate stronger relationships between vertices.

Probabilistic Shortest Paths: `pathRanker(method="prob.shortest.path")` finds the K most probable loopless paths given a weighted network. Before the paths are ranked the edge weights are converted into probabilistic edge weights using the Empirical Cumulative Distribution (ECDF) over all edge weights. This is called ECDF edge weight. The ECDF edge weight serves as a probabilistic rank of the most important gene-gene interactions. The probabilistic nature of the ECDF edge weights allow for a significance test to determine if a path contains any functional structure or is simply a random walk. The probability of a path is simily the product of all ECDF weights along the path. This is computed as a sum of the logs of the ECDF edge weights.

The follwing arguments can be passed to `pathRanker(method="prob.shortest.path")`:

`K` Maximum number of paths to extract. Defaults to 10.

`minPathSize` The minimum number of edges for each extracted path. Defualts to 1.

`normalize` Specify if you want to normalize the probabilistic edge weights (across different labels) before extracting the paths. Defaults to TRUE.

P-value method: `pathRanker(method="pvalue")` is deprecated. Please use `prob.shortest.path` instead.

Value

A list of paths where each path has the following items:

gene	The ordered sequence of genes visited along the path.
compounds	The ordered sequence of compounds visited along the path.

`weights`	The ordered sequence of the log(ECDF edge weights) along the path.
`distance`	The sum of the log(ECDF edge weights) along each path. (a sum of logs is a product)

Author(s)

Timothy Hancock, Ichigaku Takigawa, Nicolas Wicker and Ahmed Mohamed

See Also

getPathsAsEIDs, extractPathNetwork

Other Path ranking methods: `extractPathNetwork()`, `getPathsAsEIDs()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)
```

`pathsToBinary` *Converts the result from pathRanker into something suitable for pathClassifier or pathCluster.*

Description

Converts the result from pathRanker into something suitable for pathClassifier or pathCluster.

Usage

```
pathsToBinary(ypaths)
```

Arguments

`ypaths`	The result of `pathRanker`.

Details

Converts a set of pathways from pathRanker into a list of binary pathway matrices. If the pathways are grouped by a response label then the *pathsToBinary* returns a list labeled by response class where each element is the binary pathway matrix for each class. If the pathways are from pathRanker then a list wiht a single element containing the binary pathway matrix is returned. To look up the structure of a specific binary path in the corresponding ypaths object simply use matrix index by calling ypaths[[ybinpaths\$pidx[i,]]], where i is the row in the binary paths object you wish to reference.

Value

A list with the following elements.

paths	All paths within ypaths converted to a binary string and concatenated into the one matrix.
y	The response variable.
pidx	An matrix where each row specifies the location of that path within the ypaths object.

Author(s)

Timothy Hancock and Ichigaku Takigawa

See Also

Other Path clustering & classification methods: pathClassifier(), pathCluster(), plotClassifierROC(), plotClusterMatrix(), plotPathClassifier(), plotPathCluster(), predictPathClassifier(), predictPathCluster()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.cluster <- pathCluster(ybinpaths, M=3)
```

```
plotClusters(ybinpaths, p.cluster, col=c("red", "green", "blue") )
```

`plotAllNetworks` *Higlighting ranked paths over multiple network representations.*

Description

This function highlighting ranked paths over different network representations, metabolic, reaction and gene networks. The functions finds equivalent paths across different networks and marks them.

Usage

```
plotAllNetworks(
  paths,
  metabolic.net = NULL,
  reaction.net = NULL,
  gene.net = NULL,
  path.clusters = NULL,
  plot.clusters = TRUE,
  col.palette = palette(),
  layout = layout.auto,
  ...
)
```

Arguments

`paths`	The result of `pathRanker`.
`metabolic.net`	A bipartite metabolic network.
`reaction.net`	A reaction network, resulting from `makeReactionNetwork`.
`gene.net`	A gene network, resulting from `makeGeneNetwork`.
`path.clusters`	The result from `pathCluster` or `pathClassifier`.
`plot.clusters`	Whether to plot clustering information, as generated by `plotClusters`
`col.palette`	A color palette, or a palette generating function (ex: `col.palette=rainbow`).
`layout`	Either a graph layout function, or a two-column matrix specifiying vertex coordinates.
`...`	Additional arguments passed to `plotNetwork`.

Value

Highlights the path list over all provided networks.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: colorVertexByAttr(), layoutVertexByAttr(), plotClassifierROC(), plotClusterMatrix(), plotCytoscapeGML(), plotNetwork(), plotPathClassifier(), plotPaths()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

plotAllNetworks(ranked.p, metabolic.net = ex_sbml, reaction.net = rgraph,
     vertex.label = "", vertex.size = 4)
```

---

plotClassifierROC *Diagnostic plots for pathClassifier.*

Description

Diagnostic plots for pathClassifier.

Usage

```
plotClassifierROC(mix)
```

Arguments

mix	The result from pathClassifier.

Value

Diagnostic plots of the result from pathClassifier. itemTopROC curves for the posterior probabilities (`mix\$posterior.probs`) and for each HME3M component (`mix\$h`). This gives information about what response label each relates to. A ROC curve with an AUC < 0.5 relates to y = 0. Conversely ROC curves with AUC > 0.5 relate to y = 1. itemBottomThe likelihood convergence history for the HME3M model. If the parameters `alpha` or `lambda` are set too large then the likelihood may decrease.

Author(s)

Timothy Hancock and Ichigaku Takigawa

See Also

Other Path clustering & classification methods: `pathClassifier()`, `pathCluster()`, `pathsToBinary()`, `plotClusterMatrix()`, `plotPathClassifier()`, `plotPathCluster()`, `predictPathClassifier()`, `predictPathCluster()`

Other Plotting methods: `colorVertexByAttr()`, `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClusterMatrix()`, `plotCytoscapeGML()`, `plotNetwork()`, `plotPathClassifier()`, `plotPaths()`

`plotClusterMatrix` *Plots the structure of all path clusters*

Description

Plots the structure of all path clusters

Usage

```
plotClusterMatrix(
  ybinpaths,
  clusters,
  col = rainbow(clusters$params$M),
  grid = TRUE
)

plotClusterProbs(clusters, col = rainbow(clusters$params$M))

plotClusters(ybinpaths, clusters, col, ...)
```

Arguments

`ybinpaths`	The training paths computed by `pathsToBinary`.
`clusters`	The pathway cluster model trained by `pathCluster` or `pathClassifier`.
`col`	Colors for each path cluster.
`grid`	A logical, whether to add a `grid` to the plot
`...`	Extra paramaters passed to `plotClusterMatrix`

Value

plotClusterMatrix plots an image of all paths the training dataset. Rows are the paths and columns are the genes (features) included within each path. Paths are colored according to cluster membership.

plotClusterProbs The training set posterior probabilities for each path belonging to a 3M component.

plotClusters: combines the two plots produced by plotClusterProbs and plotClusterMatrix.

Author(s)

Ahmed Mohamed

See Also

Other Path clustering & classification methods: pathClassifier(), pathCluster(), pathsToBinary(), plotClassifierROC(), plotPathClassifier(), plotPathCluster(), predictPathClassifier(), predictPathCluster()

Other Plotting methods: colorVertexByAttr(), layoutVertexByAttr(), plotAllNetworks(), plotClassifierROC(), plotCytoscapeGML(), plotNetwork(), plotPathClassifier(), plotPaths()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=8)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.cluster <- pathCluster(ybinpaths, M=2)
plotClusters(ybinpaths, p.cluster, col=c("red", "blue") )
```

plotCytoscapeGML *Plots an annotated igraph object in Cytoscape.*

Description

plotCytoscape function has been removed because RCytoscape is no longer prensent in Bioconductor. Future plans will use RCy3 for Cytoscape plotting, once RCy3 is supported on MacOS and Windows. plotCytoscapeGML exports the network plot in GML format, that can be later imported into Cytoscape (using "import network from file" option). This fuction is compatible with all Cytoscape versions.

Usage

```
plotCytoscapeGML(
  graph,
  file,
  layout = layout.auto,
  vertex.size,
  vertex.label,
  vertex.shape,
  vertex.color,
  edge.color
)
```

Arguments

graph	An annotated igraph object.
file	Output GML file name to which the network plot is exported.
layout	Either a graph layout function, or a two-column matrix specifiying vertex coordinates.
vertex.size	Vertex size. If missing, the vertex attribute "size" (`V(g)$size`) will be used.
vertex.label	Vertex labels. If missing, the vertex attribute "label" (`V(g)$label`) will be used. If missing, vertices are labeled by their name.
vertex.shape	Vertex shape in one of igraph shapes. If missing, the vertex attribute "shape" (`V(g)$shape`) will be used. Shapes are converted from igraph convention to Cytoscape convention. "square","rectangle" and "vrectangle" are converted to "RECT", "csquare" and "crectangle" are converted to "ROUND_RECT", all other shapes are considered "ELLIPSE"

`vertex.color` A color or a list of colors for vertices. Vetices with multiple colors are not supported. If missing, the vertex attribute "color" (

`V(g)$color)`

) will be used.

`edge.color` A color or a list of colors for edges. If missing, the edge attribute "color" (

`E(g)$color)`

) will be used.

Value

For `plotCytoscapeGML`, results are written to file.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: `colorVertexByAttr()`, `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotNetwork()`, `plotPathClassifier()`, `plotPaths()`

Examples

```
data("ex_sbml")
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)
v.layout <- layoutVertexByAttr(rgraph, "compartment")
v.color <- colorVertexByAttr(rgraph, "compartment")

# Export network plot to GML file
plotCytoscapeGML(rgraph, file="example.gml", layout=v.layout,
   vertex.color=v.color, vertex.size=10)
```

`plotNetwork` *Plots an annotated igraph object.*

Description

This function is a wrapper function for `plot.igraph`, with 2 main additions. 1. Add the ability to color vertices by their attributes (see examples), accompanied by an inofrmative legend. 2. Resize vertex.size, edge.arrow.size, label.cex according to the plot size and the size of the network.

Usage

```
plotNetwork(
  graph,
  vertex.color,
  col.palette = palette(),
  layout = layout.auto,
  legend = TRUE,
  ...
)
```

Arguments

`graph`	An annotated igraph object.
`vertex.color`	A list of colors for vertices, or an attribute names (ex: "pathway") by which vertices will be colored. Complex attributes, where a vertex belongs to more than one group, are supported. This can also be the output of `colorVertexByAttr`.
`col.palette`	A color palette, or a palette generating function (ex: `col.palette=rainbow`).
`layout`	Either a graph layout function, or a two-column matrix specifiying vertex coordinates.
`legend`	Wheter to plot a legend. The legend is only plotted if vertices are colored by attribute values.
`...`	Additional arguments passed to `plot.igraph`.

Value

Produces a plot of the network.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: `colorVertexByAttr()`, `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotCytoscapeGML()`, `plotPathClassifier()`, `plotPaths()`

Examples

```
data("ex_kgml_sig")
plotNetwork(ex_kgml_sig, vertex.color="pathway")
plotNetwork(ex_kgml_sig, vertex.color="pathway", col.palette=heat.colors)
plotNetwork(ex_kgml_sig, vertex.color="pathway",
           col.palette=c("red", "green","blue","grey"))
```

`plotPathClassifier` *Plots the structure of specified path found by pathClassifier.*

Description

Plots the structure of specified path found by pathClassifier.

Usage

```
plotPathClassifier(ybinpaths, obj, m, tol = NULL)
```

Arguments

`ybinpaths`	The training paths computed by `pathsToBinary`
`obj`	The pathClassifier `pathClassifier`.
`m`	The path component to view.
`tol`	A tolerance for 3M parameter `theta` which is the probability for each edge within each cluster. If the tolerance is set all edges with a `theta` below that tolerance will be removed from the plot.

Value

Produces a plot of the paths with the path probabilities and prediction probabilities and ROC curve overlayed.

`Center Plot`	An image of all paths the training dataset. Rows are the paths and columns are the genes (vertices) included within each pathway. A colour within image indicates if a particular gene (vertex) is included within a specific path. Colours flag whether a path belongs to the current HME3M component (P > 0.5).
`Center Right`	The training set posterior probabilities for each path belonging to the current 3M component.
`Center Top`	The ROC curve for this HME3M component.
`Top Bar Plots`	`Theta`: The 3M component probabilities - indicates the importance of each edge is to a path. `Beta`: The PLR coefficient - the magnitude indicates the importance of the edge to the classify the response.

Author(s)

Timothy Hancock and Ichigaku Takigawa

See Also

Other Path clustering & classification methods: `pathClassifier()`, `pathCluster()`, `pathsToBinary()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotPathCluster()`, `predictPathClassifier()`, `predictPathCluster()`

Other Plotting methods: `colorVertexByAttr()`, `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotCytoscapeGML()`, `plotNetwork()`, `plotPaths()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.class <- pathClassifier(ybinpaths, target.class = "BCR/ABL", M = 3)

## Plotting the classifier results.
plotClassifierROC(p.class)
plotClusters(ybinpaths, p.class)
```

---

plotPathCluster — *Plots the structure of specified path cluster*

Description

Plots the structure of specified path found by pathCluster.

Usage

```
plotPathCluster(ybinpaths, clusters, m, tol = NULL)
```

Arguments

ybinpaths	The training paths computed by pathsToBinary.
clusters	The pathway cluster model trained by pathCluster or pathClassifier.
m	The path cluster to view.
tol	A tolerance for 3M parameter theta which is the probability for each edge within each cluster. If the tolerance is set all edges with a theta below that tolerance will be removed from the plot.

Value

Produces a plot of the paths with the path probabilities and cluster membership probabilities.

`Center Plot`	An image of all paths the training dataset. Rows are the paths and columns are the genes (features) included within each path.
`Right`	The training set posterior probabilities for each path belonging to the current 3M component.
`Top Bar Plots`	`Theta`, The 3M component probabilities - indicates the importance of each edge to a pathway.

Author(s)

Timothy Hancock and Ichigaku Takigawa

See Also

Other Path clustering & classification methods: `pathClassifier()`, `pathCluster()`, `pathsToBinary()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotPathClassifier()`, `predictPathClassifier()`, `predictPathCluster()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot", bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=8)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.cluster <- pathCluster(ybinpaths, M=2)
plotPathCluster(ybinpaths, p.cluster, m=2, tol=0.05)
```

`plotPaths` *Plots an annotated igraph object higlighting ranked paths.*

Description

This function plots a network highlighting ranked paths. If `path.clusters` are provided, paths in the same cluster are assigned similar colors.

Usage

```
plotPaths(
  paths,
  graph,
  path.clusters = NULL,
  col.palette = palette(),
  layout = layout.auto,
  ...
)
```

Arguments

`paths`	The result of `pathRanker`.
`graph`	An annotated igraph object.
`path.clusters`	The result from `pathCluster` or `pathClassifier`.
`col.palette`	A color palette, or a palette generating function (ex: `col.palette=rainbow`).
`layout`	Either a graph layout function, or a two-column matrix specifiying vertex coordinates.
`...`	Additional arguments passed to `plotNetwork`.

Value

Produces a plot of the network with paths highlighted. If paths are computed for several labels (sample categories), a plot is created for each label.

Author(s)

Ahmed Mohamed

See Also

Other Plotting methods: `colorVertexByAttr()`, `layoutVertexByAttr()`, `plotAllNetworks()`, `plotClassifierROC()`, `plotClusterMatrix()`, `plotCytoscapeGML()`, `plotNetwork()`, `plotPathClassifier()`

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Plot paths.
plotPaths(ranked.p, rgraph)

## Convert paths to binary matrix, build a classifier.
ybinpaths <- pathsToBinary(ranked.p)
p.class <- pathClassifier(ybinpaths, target.class = "BCR/ABL", M = 3)

## Plotting with clusters, on a metabolic graph.
plotPaths(ranked.p, ex_sbml, path.clusters=p.class)
```

`predictPathClassifier` *Predicts new paths given a pathClassifier model.*

Description

Predicts new paths given a pathClassifier model.

Usage

```
predictPathClassifier(mix, newdata)
```

Arguments

`mix`	The result from `pathClassifier`.
`newdata`	A data.frame containing the new paths to be classified.

Value

A list with the following elements.

`h`	The posterior probabilities for each HME3M component.

posterior.probs
: The posterior probabilities for HME3M model to classify the response.

label
: A vector indicating the HME3M cluster membership.

component
: The HME3M component membership for each pathway.

path.probabilities
: The 3M path probabilities.

plr.probabilities
: The PLR predictions for each component.

Author(s)

Timothy Hancock and Ichigaku Takigawa

See Also

Other Path clustering & classification methods: pathClassifier(), pathCluster(), pathsToBinary(), plotClassifierROC(), plotClusterMatrix(), plotPathClassifier(), plotPathCluster(), predictPathCluster()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot",
 y=factor(colnames(ex_microarray)), bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=6)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.class <- pathClassifier(ybinpaths, target.class = "BCR/ABL", M = 3)

## Just an example of how to predict cluster membership
pclass.pred <- predictPathCluster(p.class, ybinpaths$paths)
```

predictPathCluster *Predicts new paths given a pathCluster model*

Description

Predicts new paths given a pathCluster model.

Usage

```
predictPathCluster(pfit, newdata)
```

Arguments

pfit	The pathway cluster model trained by pathCluster or pathClassifier.
newdata	The binary pathway dataset to be assigned a cluster label.

Value

A list with the following elements:

labels	a vector indicating the 3M cluster membership.
posterior.probs	a matrix of posterior probabilities for each path belonging to each cluster.

Author(s)

Ichigaku Takigawa

Timothy Hancock

See Also

Other Path clustering & classification methods: pathClassifier(), pathCluster(), pathsToBinary(), plotClassifierROC(), plotClusterMatrix(), plotPathClassifier(), plotPathCluster(), predictPathClassifier()

Examples

```
## Prepare a weighted reaction network.
## Conver a metabolic network to a reaction network.
data(ex_sbml) # bipartite metabolic network of Carbohydrate metabolism.
rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)

## Assign edge weights based on Affymetrix attributes and microarray dataset.
# Calculate Pearson's correlation.
data(ex_microarray) # Part of ALL dataset.
rgraph <- assignEdgeWeights(microarray = ex_microarray, graph = rgraph,
 weight.method = "cor", use.attr="miriam.uniprot", bootstrap = FALSE)

## Get ranked paths using probabilistic shortest paths.
```

```
ranked.p <- pathRanker(rgraph, method="prob.shortest.path",
    K=20, minPathSize=8)

## Convert paths to binary matrix.
ybinpaths <- pathsToBinary(ranked.p)
p.cluster <- pathCluster(ybinpaths, M=2)

## just an example of how to predict cluster membership.
pclust.pred <- predictPathCluster(p.cluster,ybinpaths$paths)
```

`registerMemoryErr` *Internal method to register memery errors.*

Description

Internal method to register memery errors, caused by compiled code. This method is used only by the package, and should not be invoked by users.

Usage

```
registerMemoryErr(method)
```

Arguments

`method`	The mathod which generated the error.

Author(s)

Ahmed Mohamed

`reindexNetwork` *Replaces current vertex ids with chosen attribute.*

Description

This function allows users to replace vertex ids with another attribute, calculating connectivities based on the new attribute.

Usage

```
reindexNetwork(graph, v.attr)
```

Arguments

`graph`	An annotated igraph object.
`v.attr`	Name of the attribute to use as vertex ids.

Details

This functions can be very useful when merging networks constructed from different databases. For example, to match a network created from Reactome to a KEGG network, you can reindex the vertices by "miriam.kegg.compound" attribute. Another usage is to remove duplicated vertices (in case of different subcellular compartments, for example). if a network has ATP_membrane & ATP_cytoplasm vertices, reindexing by chemical name will collapse them into one 'ATP' vertex.

Value

A new graph with vertices expanded.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: `expandComplexes()`, `makeMetaboliteNetwork()`, `makeReactionNetwork()`, `rmSmallCompounds()`, `simplifyReactionNetwork()`, `vertexDeleteReconnect()`

Examples

```
 ## Make a gene network from a reaction network.
 data(ex_sbml) # A bipartite metbaolic network.
 rgraph <- makeReactionNetwork(ex_sbml, simplify=TRUE)
 ggraph <- makeGeneNetwork(rgraph)

 ## Expand vertices into their contituent genes.
 data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
 ggraph <- expandComplexes(ex_kgml_sig, v.attr = "miriam.ncbigene",
      keep.parent.attr= c("^pathway", "^compartment"))

 ## Create a separate vertex for each compartment. This is useful in duplicating
 ##  metabolite vertices in a network.
## Not run:
 graph <- expandComplexes(graph, v.attr = "compartment",
        keep.parent.attr = "all",
        expansion.method = "duplicate",
        missing.method = "keep")

## End(Not run)
```

`rmSmallCompounds` *Remove uniquitous compounds from a metabolic network*

Description

This function removes uniquitous compounds (metabolites connected to numerous reactions) from a metabolic network.These compounds are reaction cofactors and currency compounds, such as ATP, CO2, etc. A path through these metabolites may not be bioloigcally meaningful. The defualt small compound list is derived from Reactome, containing keeg.compound, pubchem.compound, ChEBI and CAS identifiers.

Usage

```
rmSmallCompounds(
  graph,
  method = c("remove", "duplicate"),
  small.comp.ls = NPMdefaults("small.comp.ls")
)
```

Arguments

graph	A metabolic network.
method	How to handle small compounds. Either simply delete these vertices "remove" (default), or make a separate vertex for each reaction they participate in "duplicate".
small.comp.ls	A list of small compounds to be used.

Value

A modified graph, with the small compounds removed or duplicated.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: expandComplexes(), makeMetaboliteNetwork(), makeReactionNetwork(), reindexNetwork(), simplifyReactionNetwork(), vertexDeleteReconnect()

Examples

```
data(ex_sbml)


sbml.removed <- rmSmallCompounds(ex_sbml, method="remove")
```

SBML2igraph *Processes SBML files into igraph objects*

Description

This function takes SBML files as input, and returns either a metabolic or a signaling network as output.

Usage

```
SBML2igraph(
  filename,
  parse.as = c("metabolic", "signaling"),
  miriam.attr = "all",
  gene.attr,
  expand.complexes,
  verbose = TRUE
)
```

Arguments

`filename`	A character vector containing the SBML files to be processed. If a directory path is provided, all *.xml and *.sbml files in it and its subdirectories are included.
`parse.as`	Whether to process file into a metabolic or a signaling network.
`miriam.attr`	A list of annotation attributes to be extracted. If `"all"`, then all attibutes written in MIRIAM guidelines (see Details) are extracted (Default). If `"none"`, then no attributes are extracted. Otherwise, only attributes matching those specified are extracted.
`gene.attr`	An attribute to distinguish `species` representing genes from those representing small molecules (see Details). Ignored if `parse.as="metabolic"`.
`expand.complexes`	Split protein complexes into individual gene nodes. Ignored if `parse.as="metabolic"`, or when `gene.attr` is not provided.
`verbose`	Whether to display the progress of the function.

Details

Users can specify whether files are processes as metabolic or signaling networks.

Metabolic networks are given as bipartite graphs, where metabolites and reactions represent vertex types. This is constructed from `ListOfReactions` in SBML file, connecting them to their corresponding substrates and products (`ListOfSpecies`). Each reaction vertex has `genes` attribute, listing all `modifiers` of this reaction. As a general rule, reactions inherit all annotation attributes of its catalyzig genes.

Signaling network have genes as vertices and edges represent interactions. Since SBML format may represent singling events as `reaction`, all species are assumed to be genes (rather than small molecules). For a simple path `S0` -> `R1` -> `S1`, in signaling network, the path will be `S0` -> `M(R1)` -> `S1` where `M(R1)` is R1 modifier(s). To ditiguish gene species from small molecules, user can provide `gene.attr` (for example: `miriam.uniprot` or `miriam.ncbigene`) where only annotated species are considered genes.

All annotation attributes written according to MIRIAM guidlines (either `urn:miriam:xxx:xxx` or `http://identifiers.org/xxx/xxx`) are etxracted by default. Non-conforming attributes can be extracted by specifying `miriam.attr`.

To generate a genome scale network, simply provide a list of files to be parsed, or put all file in a directory, as pass the directory path as `filename`

Note: This function requires libSBML installed (Please see the installation instructions in the Vignette). Some SBML level-3 files may requires additional libraries also (An infomative error will be displayed when parsing such files). Please visit http://sbml.org/Documents/Specifications/SBML_Level_3/Packages for more information.

Value

An igraph object, representing a metbolic or a signaling network.

Author(s)

Ahmed Mohamed

See Also

Other Database extraction methods: `KGML2igraph()`, `biopax2igraph()`

Examples

```
if(is.loaded("readsbmlfile")){ # This is false if libSBML wasn't available at installation.
    filename <- system.file("extdata", "porphyrin.sbml", package="NetPathMiner")

    # Process SBML file as a metabolic network
    g <- SBML2igraph(filename)
    plotNetwork(g)

    # Process SBML file as a signaling network
    g <- SBML2igraph(filename, parse.as="signaling",
                    gene.attr="miriam.uniprot",expand.complexes=TRUE)
    dev.new()
    plotNetwork(g)
}
```

simplifyReactionNetwork *Removes reactions with no gene annotations*

Description

This function removes reaction vertices with no gene annotations as indicated by the parameter gene.attr, and connect their neighbour vertices to preserve graph connectivity. This is particularly meaningful when reactions are translocation or spontaneous reactions, which are not catalysed by genes.

Usage

```
simplifyReactionNetwork(
  reaction.graph,
  gene.attr = "genes",
  remove.missing.genes = TRUE,
  reconnect.threshold = vcount(reaction.graph)
)
```

Arguments

reaction.graph A reaction network.

gene.attr The attribute to be considered as "genes". Reactions missing this annotation, will be removed.

remove.missing.genes
If FALSE, only tranlocation and spontaneous reactions are removed, otherwise all rections with no gene annotations are removed.

reconnect.threshold
An argument passed to vertexDeleteReconnect

Value

A simplified reaction network.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: expandComplexes(), makeMetaboliteNetwork(), makeReactionNetwork(), reindexNetwork(), rmSmallCompounds(), vertexDeleteReconnect()

Examples

```
data(ex_sbml)
rgraph <- makeReactionNetwork(ex_sbml, simplify=FALSE)

## Removes all reaction nodes with no annotated genes.
rgraph <- simplifyReactionNetwork(rgraph, remove.missing.genes=TRUE)
```

---

stdAttrNames *MIRIAM annotation attributes*

Description

These functions deals with conforming with MIRIAM annotation guidelines, conversion and mapping between MIRIAM identifiers.

Usage

```
stdAttrNames(graph, return.value = c("matches", "graph"))

fetchAttribute(
  graph,
  organism = "Homo sapiens",
  target.attr,
  source.attr,
  bridge.web = NPMdefaults("bridge.web")
)
```

Arguments

`graph`	An annotated igraph object.
`return.value`	Specify whether to return the names of matched standard annotations, or modify the graph attribute names to match the standards.
`organism`	The latin name of the organism (Case-sensitive).
`target.attr`	The target annotation, given as MIRIAM standard in the format `miriam.xxx`
`source.attr`	The source annotation attribute from `graph`
`bridge.web`	The base URL for Brigde Database webservices.

Value

For `stdAttrNames`, `matches` gives the original attribute names and their MIRIAM version. Since this is done by simple text matching, mismatches may occur for ambiguous annotations (such as GO, EC number). `graph` returns the input graph with attribute names standardized.

For `fetchAttribute`, the input `graph` with the fetched attribute mapped to vertices.

Author(s)

Ahmed Mohamed

See Also

Other Attribute handling methods: getAttrStatus()

Examples

```
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
## Modify attribute names to match MIRIAM standard annotations.
graph <- stdAttrNames(ex_kgml_sig, "graph")

# Use Attribute fetcher to get affymetrix probeset IDs for network vertices.
## Not run:
  graph <- fetchAttribute(graph, organism="Homo sapiens",
                  target.attr="miriam.affy.probeset")

## End(Not run)
```

---

toGraphNEL   *Converts an annotated igraph object to graphNEL*

Description

Converts an annotated igraph object to graphNEL

Usage

```
toGraphNEL(graph, export.attr = "")
```

Arguments

graph	An annotated igraph object..
export.attr	A regex experssion representing vertex attributes to be exported to the new graphNEL object. Supplying an empty string "" (default) will export all attributes.

Value

A graphNEL object.

Author(s)

Ahmed Mohamed

Examples

```
data(ex_kgml_sig) # Ras and chemokine signaling pathways in human
graphNEL <- toGraphNEL(ex_kgml_sig, export.attr="^miriam.")
```

`vertexDeleteReconnect` *Network editing: removing vertices and connecting their neighbours*

Description

This function removes vertices given as `vids` and connects their neighbours as long as the shortest path beween the neighbours are below the `reconnect.threshold`.

Usage

```
vertexDeleteReconnect(
  graph,
  vids,
  reconnect.threshold = vcount(graph),
  copy.attr = NULL
)
```

Arguments

`graph`	A reaction network.
`vids`	Vertex ids to be removed.
`reconnect.threshold`	If the shortest path between vertices is larger than this threshold, they are not reconnected.
`copy.attr`	A function, or a list of functions, combine edge attributes. Edge attributes of new edges (between reconnected neighbours) are obtained by combining original edges attributes along the shortest path between reconnected neighbors.

Value

A modified graph.

Author(s)

Ahmed Mohamed

See Also

Other Network processing methods: `expandComplexes()`, `makeMetaboliteNetwork()`, `makeReactionNetwork()`, `reindexNetwork()`, `rmSmallCompounds()`, `simplifyReactionNetwork()`

Examples

```
## Remove all reaction vertices from a bipartite metabolic network
##  keeping only metabolite vertices.
data(ex_sbml)
graph <- vertexDeleteReconnect(ex_sbml, vids=which(V(ex_sbml)$reactions))
```

Index