Package 'BioNetStat'

October 16, 2020

Type Package

Title Biological Network Analysis

Version 1.8.4

Author Vinícius Jardim, Suzana Santos, André Fujita, and Marcos Buckeridge

Maintainer Vinicius Jardim <viniciusjc@gmail.com>

Description A package to perform differential network analysis, differential node analysis (differential coexpression analysis), network and metabolic pathways view.

License GPL (>= 3)

Depends R (>= 3.5), shiny, igraph, shinyBS, pathview, DT

Imports BiocParallel, RJSONIO, whisker, yaml, pheatmap, ggplot2, plyr, utils, stats, RColorBrewer, Hmisc, psych, knitr

biocViews Network, NetworkInference, Pathways, GraphAndNetwork, Sequencing, Microarray, Metabolomics, Proteomics, GeneExpression, RNASeq, SystemsBiology, DifferentialExpression, GeneSetEnrichment, ImmunoOncology

RoxygenNote 6.1.1

URL http://github.com/jardimViniciusC/BioNetStat

BugReports http://github.com/jardimViniciusC/BioNetStat/issues

Encoding UTF-8

VignetteBuilder knitr

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

git_branch RELEASE_3_11

git_last_commit 37fb4cb

git_last_commit_date 2020-05-31

Date/Publication 2020-10-16

R topics documented:

`adjacencyMatrix` *Adjacency matrix*

Description

creates a function that infers a graph from variables values matrix

Usage

```
adjacencyMatrix(
  method,
  association = "none",
  threshold = "none",
  thr.value = 0.05,
  weighted = TRUE,
  abs.values = TRUE
)
```

Arguments

`method`	a function that measures the association between the variables values.
`association`	a charactere string indicating wich value will be used as association value. The options are "corr" for the correlation value, "pvalue" for nominal pvalue associated to correlation or "fdr" for corrected pvalue for mutiple tests.
`threshold`	a charactere string indicating wich value will be used as threshold value. The options are "corr" for the correlation value, "pvalue" for nominal pvalue associated to correlation or "fdr" for corrected pvalue for mutiple tests. If NULL, no edge is removed.
`thr.value`	a numeric value. The function removes all edges weighted by a value less than or equal to 'thr.value'.
`weighted`	a logical value. If TRUE, then the edges of the graph are weighted by the association degrees between the variables. Otherwise, the edges are are weighted by one.

abs.values a logical value. If TRUE, then the negatives edges of the graph are changed by its absolutes values. Otherwise, the negative edges are kept with negative weights.

Value

a function that creates an adjacency matrix from variable values data.

Examples

```
set.seed(3)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels<-rep(0:3,10)
functionAdjacencyMatrix <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)
```

---

bnsDataTest *Gene expression in gliomas*

Description

Expression of 134 target genes of NFKBIA in four gliomas tissues, astrocytom, oligodendrogliom, oligoastrcytom and Primary Multiform Glioblastom.

Usage

```
bnsDataTest
```

Format

A data frame containing 134 variables (colunms) and 658 observations.

Source

TCGA - The Cancer Genome Atlas

References

KINKER, G. S. et al. Deletion and low expression of NFKBIA are associated with poor prognosis in lower-grade glioma patients. Scientific Reports, v. 6, n. April, p. 24160, 2016.

`centralityPathPlot` *Structural measures of vertices view in metabolic pathways*

Description

Vertices centralities or clustering coefficient view in KEGG metabolic pathways. centralityPathPlot and pathplot are functions based on pathview function of Pathview package. Pathview is a tool set for pathway based data integration and visualization. It maps and renders user data on relevant pathway graphs. All users need is to supply their gene or compound data and specify the target pathway. Pathview automatically downloads the pathway graph data, parses the data file, maps user data to the pathway, and render pathway graph with the mapped data. Pathview generates both native KEGG view and Graphviz views for pathways. keggview.native and keggview.graph are the two viewer functions, and pathview is the main function providing a unified interface to downloader, parser, mapper and viewer functions.

Usage

```
centralityPathPlot(
  gene.data = NULL,
  cpd.data = NULL,
  threshold = NULL,
  thr.value = 0.05,
  species,
  pathway.id,
  kegg.native = TRUE,
  file.name = "path",
  limit = list(gene = NULL, cdp = NULL),
  bins = list(gene = 15, cpd = 15),
  both.dirs = list(gene = FALSE, cpd = FALSE),
  mid = list(gene = "white", cpd = "white"),
  high = list(gene = "red", cpd = "red")
)
```

Arguments

`gene.data`	an output dataframe from diffNetAnalysis function. Data frame structure has genes as rows and statistical test, Nominal p-value, Q-value (p-value FDR adjust for multiple tests)and networks measures, for each network, as columns. Row names should be gene IDs. Here gene ID is a generic concepts, including multiple types of gene, transcript and protein uniquely mappable to KEGG gene IDs. KEGG ortholog IDs are also treated as gene IDs as to handle metagenomic data. Check details for mappable ID types. Default gene.data=NULL. numeric, character, continuous
`cpd.data`	the same as gene.data, excpet named with IDs mappable to KEGG compound IDs. Over 20 types of IDs included in CHEMBL database can be used here. Check details for mappable ID types. Default cpd.data=NULL. Note that gene.data and cpd.data can't be NULL simultaneously.
`threshold`	a character indicating which column has to be used to filter which genes or coumponds will be drawn in metabolic map. The options are "pvalue" or "qvalue" to filter by Nominal p-value or Q-value (p-value FDR adjust for multiple tests), respectively. The default threshold=NULL, do not filter any row of data frame.

`thr.value` a numeric value indicating the upper threshold value to filter data frame rows.

`species` character, either the kegg code, scientific name or the common name of the target species. This applies to both pathway and gene.data or cpd.data. When KEGG ortholog pathway is considered, species="ko". Default species="hsa", it is equivalent to use either "Homo sapiens" (scientific name) or "human" (common name).

`pathway.id` character vector, the KEGG pathway ID(s), usually 5 digit, may also include the 3 letter KEGG species code.

`kegg.native` logical, whether to render pathway graph as native KEGG graph (.png) or using graphviz layout engine (.pdf). Default kegg.native=TRUE.

`file.name` character, the suffix to be added after the pathway name as part of the output graph file. Sample names or column names of the gene.data or cpd.data are also added when there are multiple samples. Default out.suffix="pathview".

`limit` a list of two numeric elements with "gene" and "cpd" as the names. This argument specifies the limit values for gene.data and cpd.data when converting them to pseudo colors. Each element of the list could be of length 1 or 2. Length 1 suggests discrete data or 1 directional (positive-valued) data, or the absolute limit for 2 directional data. Length 2 suggests 2 directional data. Default limit=list(gene=1, cpd=1).

`bins` a list of two integer elements with "gene" and "cpd" as the names. This argument specifies the number of levels or bins for gene.data and cpd.data when converting them to pseudo colors. Default limit=list(gene=10, cpd=10).

`both.dirs` a list of two logical elements with "gene" and "cpd" as the names. This argument specifies whether gene.data and cpd.data are 1 directional or 2 directional data when converting them to pseudo colors. Default limit=list(gene=TRUE, cpd=TRUE).

`mid, high` each is a list of two colors with "gene" and "cpd" as the names. This argument specifies the color spectra to code gene.data and cpd.data. When data are 1 directional (TRUE value in both.dirs), only mid and high are used to specify the color spectra. Default spectra (low-mid-high) "green"-"gray"-"red" and "blue"-"gray"-"yellow" are used for gene.data and cpd.data respectively. The values for 'low, mid, high' can be given as color names ('red'), plot color index (2=red), and HTML-style RGB, ("\#FF0000"=red).

Details

This function uses pathview to visualize the vertex structural measures in metabolic maps. Pathview maps and renders user data on relevant pathway graphs. Pathview is a stand alone program for pathway based data integration and visualization. It also seamlessly integrates with pathway and functional analysis tools for large-scale and fully automated analysis. Pathview provides strong support for data Integration. It works with: 1) essentially all types of biological data mappable to pathways, 2) over 10 types of gene or protein IDs, and 20 types of compound or metabolite IDs, 3) pathways for over 2000 species as well as KEGG orthology, 4) varoius data attributes and formats, i.e. continuous/discrete data, matrices/vectors, single/multiple samples etc. To see mappable external gene/protein IDs do: data(gene.idtype.list), to see mappable external compound related IDs do: data(rn.list); names(rn.list). Pathview generates both native KEGG view and Graphviz views for pathways. Currently only KEGG pathways are implemented. Hopefully, pathways from Reactome, NCI and other databases will be supported in the future.

Value

From viersion 1.9.3, pathview can accept either a single pathway or multiple pathway ids. The result returned by pathview function is a named list corresponding to the input pathway ids. Each element (for each pathway itself is a named list, with 2 elements ("plot.data.gene", "plot.data.cpd"). Both elements are data.frame or NULL depends on the corresponding input data gene.data and cpd.data. These data.frames record the plot data for mapped gene or compound nodes: rows are mapped genes/compounds, columns are: kegg.names standard KEGG IDs/Names for mapped nodes. It's Entrez Gene ID or KEGG Compound Accessions. labels Node labels to be used when needed. all.mapped All molecule (gene or compound) IDs mapped to this node. type node type, currently 4 types are supported: "gene","enzyme", "compound" and "ortholog". x x coordinate in the original KEGG pathway graph. y y coordinate in the original KEGG pathway graph. width node width in the original KEGG pathway graph. height node height in the original KEGG pathway graph. other columns columns of the mapped gene/compound data and corresponding pseudo-color codes for individual vertex measures The results returned by keggview.native and codekeggview.graph are both a list of graph plotting parameters. These are not intended to be used externally.

References

This function is an adaptation of Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

Examples

```
set.seed(5)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels <- rep(0:3,10)
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue", threshold="fdr",
 thr.value=0.05, weighted=FALSE)
vertexCentrality <- degreeCentralityVertexTest(expr, labels, adjacencyMatrix1,numPermutations=10)
vertexCentrality2<-cbind(c(4790, 4791, 4792, 4793, 84807, 4794, 4795, 64332, 595, 898, 23552,
 1017, 8099, 10263, 4609, 23077, 26292, 84073, 4610, 4613, 10408,  80177, 114897, 114898, 114899,
  114900, 114904, 114905, 390664, 338872),vertexCentrality)
centralityPathPlot(gene.data=vertexCentrality2, cpd.data=NULL, threshold="pvalue", thr.value=1,
 species="hsa" , pathway.id="05200", kegg.native=TRUE, file.name="path_example",
limit = list(gene = NULL, cdp = NULL), bins = list(gene = 15,cpd = 15),
both.dirs= list(gene = FALSE,cpd = FALSE), mid =list(gene = "white", cpd = "white"),
high = list(gene = "red",cpd = "red"))
```

`diffNetAnalysis` *Differential network analysis method*

Description

Differential network analysis method

Usage

```
diffNetAnalysis(
  method,
  options = list(bandwidth = "Sturges"),
  varFile,
```

```
  labels,
  varSets = NULL,
  adjacencyMatrix,
  numPermutations = 1000,
  print = TRUE,
  resultsFile = NULL,
  seed = NULL,
  min.vert = 5,
  BPPARAM = NULL,
  na.rm = NULL
)
```

Arguments

method	a function that receives two adjacency matrices and returns a list containing a statistic theta that measures the difference between them, and a p-value for the test H0: theta = 0 against H1: theta > 0.
options	a list contaning parameters used by 'method'. Used only in degreeDistributionTest, spectralEntropyTest and spectralDistributionTest functions. It can be set to either `list(bandwidth="Sturges")` or `list(bandwidth="Silverman")`.
varFile	a numeric matrix contaning variables values data.
labels	a vector of -1s, 0s, and 1s associating each sample with a phenotype. The value 0 corresponds to the first phenotype class of interest, 1 to the second phenotype class of interest, and -1 to the other classes, if there are more than two classes in the gene expression data.
varSets	a list of gene sets. Each element of the list is a character vector v, where v[1] contains the gene set name, v[2] descriptions about the set, v[3..length(v)] the genes that belong to the set.
adjacencyMatrix	a function that receives a numeric matrix containing gene expression data and returns the adjacency matrix of the inferred co-expression graph.
numPermutations	the number of permutations for the permutation test.
print	a logical. If true, it prints execution messages on the screen. resultsFile: path to a file where the partial results of the analysis will be saved. If NULL, then no partial results are saved.
resultsFile	a ".RData" file name to be saved in tha work directory.
seed	the seed for the random number generators. If it is not null then the sample permutations are the same for all the gene sets.
min.vert	lower number of nodes (variables) that has to be to compare the networks.
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to BiocParallel functions. #MulticoreParam()
na.rm	remove the NA values by excluding the rows ("row") or the columns ("col") that contaings it. If NULL (default) the NA values are not removed.

Value

a data frame containing the name, size, test statistic, nominal p-value and adjusted p-value (q-value) associated with each gene set.

Examples

```
# Glioma data
data("varFile")
gliomaData <- system.file("extdata", "bnsData.csv", package = "BioNetStat")
labels<-doLabels(gliomaData)
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue", threshold="fdr",
 thr.value=0.05, weighted=FALSE)
diffNetAnalysis(method=degreeCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)
# The numPermutations number is 1 to do a faster example, but we advise to use unless 1000 permutations in real an

# Random data
set.seed(1)
varFile <- as.data.frame(matrix(rnorm(120),40,30))
labels<-data.frame(code=rep(0:3,10),names=rep(c("A","B","C","D"),10))
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue", threshold="fdr",
 thr.value=0.05, weighted=FALSE)
diffNetAnalysis(method=degreeCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=10, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)
```

---

doLabels *Class vector of data table*

Description

Class vector of data table

Usage

```
doLabels(fileName, factorName = NULL, classes = NULL, dec = ".", sep = ";")
```

Arguments

fileName	the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, getwd().
factorName	string indicating the column name used to determine the labels of each row of matrix data. The NULL (default) indicates that the first column will be used.
classes	a vector of strings indicating which labels of choosed column will be compared, the minimum are two labels. The NULL (default) indicates that all classes will be compared.
dec	the character used in the file for decimal points.
sep	the field separator character. Values on each line of the file are separated by this character. If sep = "" the separator is white space, that is one or more spaces, tabs, newlines or carriage returns, if sep=NULL (default), the function uses tabulation for .txt files or ";" for .csv files.

Value

a vector that identify each row of the readVarFile object as a sample belonging to a state (network).

Examples

```
# Glioma file
gliomaData <- system.file("extdata", "bnsData.csv", package = "BioNetStat")
labels<-doLabels(gliomaData)

# Random file
test1 <- as.data.frame(cbind(rep(LETTERS[1:4],each=10),matrix(rnorm(120),40,30)))
tfl<-tempfile(fileext = ".csv")
write.table(test1, tfl,sep=";",row.names=FALSE)
labels<-doLabels(tfl)
```

---

edgeTest *Edge score equality test*

Description

Nodes scores equality test between network

Usage

```
edgeBetweennessEdgeTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)
```

Arguments

expr	Matrix of variables (columns) vs samples (rows)
labels	a vector in which a position indicates the phenotype of the corresponding sample or state
adjacencyMatrix	a function that returns the adjacency matrix for a given variables values matrix
numPermutations	number of permutations that will be carried out in the permutation test
options	argument non used in this function
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to BiocParallel functions. MulticoreParam()

Value

A table, containing on the columns, the following informations for each variable (rows): "Test Statistic" - difference among the degree centrality of a node in two or more networks associated with each phenotype "Nominal p-value" - the Nominal p-value of the test "Q-value" - the q-value of the test, correction of p-value by FDR to many tests "Factor n" - the node degree centrality in each network compared

Examples

```
set.seed(1)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels<-data.frame(code=rep(0:3,10),names=rep(c("A","B","C","D"),10))
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)
# The numPermutations number is 1 to do a faster example, but we advise to use unless 1000 permutations in real ar

# Edge betweenness centrality test
diffNetAnalysis(method=edgeBetweennessEdgeTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)
```

KLdegree *Kullback-Liebler divergence among the density functions of the degrees of two or more graphs*

Description

'KLdegree' computes the Kullback-Liebler divergence among the density functions of the degrees of two or more graphs

Usage

```
KLdegree(f)
```

Arguments

f	a list containing the components 'x' and 'densities'. The first element is the vector 'x' of 'npoints' coordinates of the points where the density function i estimated, and the second is a vector 'y' of the estimated density values.

Value

returns a list containing the components 'theta' and 'partial'. 'theta' is a value representaing the Kullback-Liebler divergence among the corresponding distributions. 'partial' is a vector of KL divergences between each network distribuiton and the average degree distribution.

See Also

`graph.strength`

`density`

Examples

```
G<-list()
G[[1]]<-erdos.renyi.game(30,0.6)
G[[2]]<-barabasi.game(30,power = 1)
G[[3]]<-watts.strogatz.game(2,30,2,0.3)
f<-nDegreeDensities(G, npoints=1024, bandwidth="Sturges")
KLdegree(f)
```

KLspectrum — *Kullback-Liebler divergence among the spectral density functions of two or more graphs*

Description

'KLspectrum' computes the Kullback-Liebler divergence among the spectral density functions of two or more graphs

Usage

```
KLspectrum(f)
```

Arguments

f	a list containing the components 'x' and 'densities'. The first element is the vector 'x' of 'npoints' coordinates of the points where the density function i estimated, and the second is a vector 'y' of the estimated density values.

Value

returns a list containing the components 'theta' and 'partial'. 'theta' is a value representaing the Kullback-Liebler divergence among the corresponding distributions. 'partial' is a vector of KL divergences between each network distribuiton and the average spectral distribution.

See Also

`graph.strength`

`density`

Examples

```
A<-list()
A[[1]]<-as.matrix(as_adj(erdos.renyi.game(30,0.6,directed = FALSE)))
A[[2]]<-as.matrix(as_adj(barabasi.game(30,power = 1,directed = FALSE)))
A[[3]]<-as.matrix(as_adj(watts.strogatz.game(1,30,2,0.3)))
f<-nSpectralDensities(A, bandwidth="Sturges")
KLspectrum(f)
```

labels — *Labels of glioma tissues in gene expression*

Description

Labels of glioma tissues in gene expression

Usage

```
data(labels)
```

Format

Vector which associates each row of data.frame (varFile) with a state (that will be an network).

Source

TCGA - The Cancer Genome Atlas

References

KINKER, G. S. et al. Deletion and low expression of NFKBIA are associated with poor prognosis in lower-grade glioma patients. Scientific Reports, v. 6, n. April, p. 24160, 2016.

Examples

```
data(labels)
# Run BNS analysis
```

`nDegreeDensities` *Density functions of the degrees of n graphs*

Description

'nDegreeDensities' estimates the density functions of the degrees for n graphs at the same coordinates

Usage

```
nDegreeDensities(
  Gs,
  npoints = 1024,
  bandwidth = "Sturges",
  from = NULL,
  to = NULL
)
```

Arguments

`Gs`	a list of n igraph graphs objects
`npoints`	number of points used in density function estimation
`bandwidth`	a parameters. It can be set to either "Sturges" or "Silverman".
`from`	the lower value used to build the distribution
`to`	the higher value used to build the distribution

Value

a list containing the components 'x' and 'densities'. The first element is the vector 'x' of 'npoints' coordinates of the points where the density function i estimated, and the second is a vector 'y' of the estimated density values.

See Also

graph.strength

density

Examples

```
G<-list()
G[[1]]<-erdos.renyi.game(30,0.6)
G[[2]]<-barabasi.game(30,power = 1)
G[[3]]<-watts.strogatz.game(2,30,2,0.3)
d<-nDegreeDensities(G, npoints=1024, bandwidth="Sturges")
par(mfrow=c(1,3))
plot(d$x,d$densities[,1],main="Erdos-Renyi\n Degree distribution",
xlab="Degree",ylab="Frequency")
plot(d$x,d$densities[,2],main="Barabasi\n Degree distribution",
xlab="Degree",ylab="Frequency")
plot(d$x,d$densities[,3],main="Watts-Strogatz\n Degree distribution",
xlab="Degree",ylab="Frequency")
```

---

networkFeature *Network features*

Description

Network feature average nodes scores (degree, betweenness, closenness, eigenvector centralities or clustering coefficient) or spectral entropies for each network analysed.

Usage

```
averageDegreeCentrality(expr, labels, adjacencyMatrix, options = NULL)

averageBetweennessCentrality(expr, labels, adjacencyMatrix, options = NULL)

averageBetweennessEdgesCentrality(
  expr,
  labels,
  adjacencyMatrix,
  options = NULL
)

averageClosenessCentrality(expr, labels, adjacencyMatrix, options = NULL)

averageEigenvectorCentrality(expr, labels, adjacencyMatrix, options = NULL)

averageClusteringCoefficient(expr, labels, adjacencyMatrix, options = NULL)

averageShortestPath(expr, labels, adjacencyMatrix, options = NULL)

spectralEntropies(
  expr,
  labels,
```

```
  adjacencyMatrix,
  options = list(bandwidth = "Sturges")
)
```

Arguments

expr	Matrix of variables (columns) vs samples (rows)
labels	a vector in which a position indicates the phenotype of the corresponding sample or state
adjacencyMatrix	a function that returns the adjacency matrix for a given variables values matrix
options	a list containing parameters. Used only in spectralEntropies function. It can be set to either `list(bandwidth="Sturges")` or `list(bandwidth="Silverman")`.

Value

a list of values containing the spectral entropie or average node score of each network.

spectralEntropies. A list of values containing the spectral entropy of each network.

Examples

```
set.seed(1)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels<-data.frame(code=rep(0:3,10),names=rep(c("A","B","C","D"),10))
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)

# Average degree centrality
averageDegreeCentrality(expr, labels, adjacencyMatrix1)

# Average betweenness centrality
averageBetweennessCentrality(expr, labels, adjacencyMatrix1)

# Average betweenness centrality
averageBetweennessCentrality(expr, labels, adjacencyMatrix1)

# Average closeness centrality
averageClosenessCentrality(expr, labels, adjacencyMatrix1)

# Average eigenvector centrality
averageEigenvectorCentrality(expr, labels, adjacencyMatrix1)

# Average clustering coefficient
averageClusteringCoefficient(expr, labels, adjacencyMatrix1)

# Average shortest path
averageShortestPath(expr, labels, adjacencyMatrix1)

# Spectral entropies
spectralEntropies(expr, labels, adjacencyMatrix1, options=list(bandwidth="Sturges"))
```

networkTest *Network equality test*

Description

Test of equality between network properties

Usage

```
degreeCentralityTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

betweennessCentralityTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

closenessCentralityTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

eigenvectorCentralityTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

clusteringCoefficientTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
```

```
  BPPARAM = NULL
)

edgeBetweennessTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

degreeDistributionTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = list(bandwidth = "Sturges"),
  BPPARAM = NULL
)

spectralEntropyTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = list(bandwidth = "Sturges"),
  BPPARAM = NULL
)

spectralDistributionTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = list(bandwidth = "Sturges"),
  BPPARAM = NULL
)
```

Arguments

`expr`	Matrix of variables (columns) vs samples (rows)
`labels`	a vector in which a position indicates the phenotype of the corresponding sample or state
`adjacencyMatrix`	a function that returns the adjacency matrix for a given variables values matrix
`numPermutations`	number of permutations that will be carried out in the permutation test
`options`	a list containing parameters. Used only in degreeDistributionTest, spectralEntropyTest and spectralDistributionTest functions. It can be set to either `list(bandwidth="Sturges")` or `list(bandwidth="Silverman")`.

BPPARAM An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to BiocParallel functions. MulticoreParam()

Value

A list containing: "measure" - difference among two or more networks associated with each phenotype. To compare networks by centralities and clustering coefficient, one uses euclidian distance. In spectral entropy comparison, one uses the absolute difference. In distributions (spectral and degree) comparison, one uses Kulback-Liebler divergence. "p.value" - the Nominal p-value of the test. "Partial" - a vector with the weigths of each network in a measure value.

Examples

```
set.seed(1)
data("varFile")
gliomaData <- system.file("extdata", "bnsData.csv", package = "BioNetStat")
labels<-doLabels(gliomaData)
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)
# The numPermutations number is 1 to do a faster example, but we advise to use unless 1000 permutations in real ar

# Degree centrality test
diffNetAnalysis(method=degreeCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Betweenness centrality test
diffNetAnalysis(method=betweennessCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Closeness centrality test
diffNetAnalysis(method=closenessCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Eigenvector centrality test
diffNetAnalysis(method=eigenvectorCentralityTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Clustering coefficient test
diffNetAnalysis(method=clusteringCoefficientTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Edge betweenness centrality test
diffNetAnalysis(method=edgeBetweennessTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Degree distribution test
diffNetAnalysis(method=degreeDistributionTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, options=list(bandwidth="Sturges"))
```

```
# Spectral entropy test
 diffNetAnalysis(method=spectralEntropyTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, options=list(bandwidth="Sturges"))

# Spectral distribution test
 diffNetAnalysis(method=spectralDistributionTest, varFile=varFile, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, options=list(bandwidth="Sturges"))
```

---

nodeScores *Node scores*

Description

Node score (degree, betweenness, closenness, eigenvector centralities or clustering coefficient) for each network analysed.

Usage

```
degreeCentrality(expr, labels, adjacencyMatrix)

betweennessCentrality(expr, labels, adjacencyMatrix)

betweennessEdgesCentrality(expr, labels, adjacencyMatrix)

closenessCentrality(expr, labels, adjacencyMatrix)

eigenvectorCentrality(expr, labels, adjacencyMatrix)

clusteringCoefficient(expr, labels, adjacencyMatrix)
```

Arguments

expr	Matrix of variables (columns) vs samples (rows).
labels	a vector in which a position indicates the phenotype of the corresponding sample or state.
adjacencyMatrix	a function that returns the adjacency matrix for a given variables values matrix.

Value

a list of vector containing the node scores (degree, betweenness, closenness, eigenvector centralities or clustering coefficient) of each network.

Examples

```
set.seed(1)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels<-data.frame(code=rep(0:3,10),names=rep(c("A","B","C","D"),10))
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)
```

```
# Degree centrality
degreeCentrality(expr, labels, adjacencyMatrix1)

# Betweenness Centrality
betweennessCentrality(expr, labels, adjacencyMatrix1)

# Edges Betweenness Centrality
betweennessEdgesCentrality(expr, labels, adjacencyMatrix1)

# Closenness Caentrality
closenessCentrality(expr, labels, adjacencyMatrix1)

# Eigenvector centrality
eigenvectorCentrality(expr, labels, adjacencyMatrix1)

# Clustering coefficient
clusteringCoefficient(expr, labels, adjacencyMatrix1)
```

---

nodeTest *Node score equality test*

Description

Nodes scores equality test between network

Usage

```
degreeCentralityVertexTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

betweennessCentralityVertexTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

closenessCentralityVertexTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
```

```
  BPPARAM = NULL
)

eigenvectorCentralityVertexTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)

clusteringCoefficientVertexTest(
  expr,
  labels,
  adjacencyMatrix,
  numPermutations = 1000,
  options = NULL,
  BPPARAM = NULL
)
```

Arguments

expr	Matrix of variables (columns) vs samples (rows)
labels	a vector in which a position indicates the phenotype of the corresponding sample or state
adjacencyMatrix	a function that returns the adjacency matrix for a given variables values matrix
numPermutations	number of permutations that will be carried out in the permutation test
options	argument non used in this function
BPPARAM	An optional BiocParallelParam instance determining the parallel back-end to be used during evaluation, or a list of BiocParallelParam instances, to be applied in sequence for nested calls to BiocParallel functions. MulticoreParam()

Value

A table, containing on the columns, the following informations for each variable (rows): "Test Statistic" - difference among the degree centrality of a node in two or more networks associated with each phenotype "Nominal p-value" - the Nominal p-value of the test "Q-value" - the q-value of the test, correction of p-value by FDR to many tests "Factor n" - the node degree centrality in each network compared

Examples

```
set.seed(1)
expr <- as.data.frame(matrix(rnorm(120),40,30))
labels<-data.frame(code=rep(0:3,10),names=rep(c("A","B","C","D"),10))
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue",
 threshold="fdr", thr.value=0.05, weighted=FALSE)
# The numPermutations number is 1 to do a faster example, but we advise to use unless 1000 permutations in real ar

# Degree centrality test
```

```
diffNetAnalysis(method=degreeCentralityVertexTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Betweenness centrality test
diffNetAnalysis(method=betweennessCentralityVertexTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Closeness centrality test
diffNetAnalysis(method=closenessCentralityVertexTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Eigenvector centrality test
diffNetAnalysis(method=eigenvectorCentralityVertexTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)

# Clustering coefficient test
diffNetAnalysis(method=clusteringCoefficientVertexTest, varFile=expr, labels=labels, varSets=NULL,
 adjacencyMatrix=adjacencyMatrix1, numPermutations=1, print=TRUE, resultsFile=NULL,
  seed=NULL, min.vert=5, option=NULL)
```

`nSpectralDensities` *Spectral Density functions of n graphs*

Description

Returns the spectral densities for a list of adjacency matrices at the same points

Usage

```
nSpectralDensities(A, from = NULL, to = NULL, bandwidth = "Silverman")
```

Arguments

`A`	a list of adjacency matrices
`from`	the lower value used to build the distribution
`to`	the higher value used to build the distribution
`bandwidth`	a parameters. It can be set to either "Sturges" or "Silverman".

Value

a list containing the components 'x' and 'densities'. The first element is the vector 'x' of 'npoints' coordinates of the points where the density function i estimated, and the second is a vector 'y' of the estimated density values.

See Also

`KLdegree`

`density`

Examples

```
A<-list()
A[[1]]<-as.matrix(as_adj(erdos.renyi.game(30,0.6,directed = FALSE)))
A[[2]]<-as.matrix(as_adj(barabasi.game(30,power = 1,directed = FALSE)))
A[[3]]<-as.matrix(as_adj(watts.strogatz.game(1,30,2,0.3)))
d<-nSpectralDensities(A, bandwidth="Sturges")
par(mfrow=c(1,3))
plot(d$x,d$densities[,1],main="Erdos-Renyi\n Spectral distribution",
xlab="Eigenvalue",ylab="Frequency")
plot(d$x,d$densities[,2],main="Barabasi\n Spectral distribution",
xlab="Eigenvalue",ylab="Frequency")
plot(d$x,d$densities[,3],main="Watts-Strogatz\n Spectral distribution",
xlab="Eigenvalue",ylab="Frequency")
```

---

pathPlot *Variable values view in metabolic pathways*

Description

Variable values view in KEGG metabolic pathways

Usage

```
pathPlot(
  gene.data = NULL,
  cpd.data = NULL,
  labels,
  varr.diff.list = NULL,
  threshold = NULL,
  thr.value = 0.05,
  FUN = median,
  species,
  pathway.id,
  kegg.native = TRUE,
  file.name = "path"
)
```

Arguments

gene.data	either vector (single sample) or a matrix-like data (multiple sample). Vector should be numeric with gene IDs as names or it may also be character of gene IDs. Character vector is treated as discrete or count data. Matrix-like data structure has genes as rows and samples as columns. Row names should be gene IDs. Here, gene ID is a generic concepts, including multiple types of gene, transcript and protein uniquely mappable to KEGG gene IDs. KEGG ortholog IDs are also treated as gene IDs as to handle metagenomic data. Check details for mappable ID types. Default gene.data=NULL. numeric, character, continuous
cpd.data	the same as gene.data, excpet named with IDs mappable to KEGG compound IDs. Over 20 types of IDs included in CHEMBL database can be used here. Check details for mappable ID types. Default cpd.data=NULL. Note that gene.data and cpd.data can't be NULL simultaneously.

labels	a vector of -1s, 0s, and 1s associating each sample with a phenotype. The value 0 corresponds to the first phenotype class of interest, 1 to the second phenotype class of interest, and -1 to the other classes, if there are more than two classes in the gene expression data.
varr.diff.list	an output dataframe from diffNetAnalysis function. Data frame structure has genes as rows and statistical test, Nominal p-value, Q-value (p-value FDR adjust for multiple tests)and networks measures, for each network, as columns. Row names should be gene IDs. Here gene ID is a generic concepts, including multiple types of gene, transcript and protein uniquely mappable to KEGG gene IDs.
threshold	a character indicating which column of "varr.diff.list" has to be used to filter which genes or coumponds will be drawn in metabolic map. The options are "pvalue" or "qvalue" to filter by Nominal p-value or Q-value (p-value FDR adjust for multiple tests), respectively. The default threshold=NULL, do not filter any row of data frame.
thr.value	a numeric value indicating the upper threshold value to filter data frame rows.
FUN	a function to define what value will be used in metabolic map.
species	character, either the kegg code, scientific name or the common name of the target species. This applies to both pathway and gene.data or cpd.data. When KEGG ortholog pathway is considered, species="ko". Default species="hsa", it is equivalent to use either "Homo sapiens" (scientific name) or "human" (common name).
pathway.id	character vector, the KEGG pathway ID(s), usually 5 digit, may also include the 3 letter KEGG species code.
kegg.native	logical, whether to render pathway graph as native KEGG graph (.png) or using graphviz layout engine (.pdf). Default kegg.native=TRUE.
file.name	character, the suffix to be added after the pathway name as part of the output graph file. Sample names or column names of the gene.data or cpd.data are also added when there are multiple samples. Default out.suffix="pathview".

Details

Pathview maps and renders user data on relevant pathway graphs. Pathview is a stand alone program for pathway based data integration and visualization. It also seamlessly integrates with pathway and functional analysis tools for large-scale and fully automated analysis. Pathview provides strong support for data Integration. It works with: 1) essentially all types of biological data mappable to pathways, 2) over 10 types of gene or protein IDs, and 20 types of compound or metabolite IDs, 3) pathways for over 2000 species as well as KEGG orthology, 4) varoius data attributes and formats, i.e. continuous/discrete data, matrices/vectors, single/multiple samples etc. To see mappable external gene/protein IDs do: data(gene.idtype.list), to see mappable external compound related IDs do: data(rn.list); names(rn.list). Pathview generates both native KEGG view and Graphviz views for pathways. Currently only KEGG pathways are implemented. Hopefully, pathways from Reactome, NCI and other databases will be supported in the future.

Value

From viersion 1.9.3, pathview can accept either a single pathway or multiple pathway ids. The result returned by pathview function is a named list corresponding to the input pathway ids. Each element (for each pathway itself is a named list, with 2 elements ("plot.data.gene", "plot.data.cpd"). Both elements are data.frame or NULL depends on the corresponding input data gene.data and cpd.data.

These data.frames record the plot data for mapped gene or compound nodes: rows are mapped genes/compounds, columns are: kegg.names standard KEGG IDs/Names for mapped nodes. It's Entrez Gene ID or KEGG Compound Accessions. labels Node labels to be used when needed. all.mapped All molecule (gene or compound) IDs mapped to this node. type node type, currently 4 types are supported: "gene","enzyme", "compound" and "ortholog". x x coordinate in the original KEGG pathway graph. y y coordinate in the original KEGG pathway graph. width node width in the original KEGG pathway graph. height node height in the original KEGG pathway graph. other columns columns of the mapped gene/compound data and corresponding pseudo-color codes for individual samples The results returned by keggview.native and codekeggview.graph are both a list of graph plotting parameters. These are not intended to be used externally.

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

Examples

```
set.seed(5)
expr <- as.data.frame(matrix(rnorm(120),40,30))
names(expr)<-c(4790, 4791, 4792, 4793, 84807, 4794, 4795, 64332, 595, 898, 23552, 1017, 8099,
 10263, 4609, 23077, 26292, 84073, 4610, 4613, 10408,  80177, 114897, 114898, 114899, 114900,
  114904, 114905, 390664, 338872)
labels <- rep(0:3,10)
adjacencyMatrix1 <- adjacencyMatrix(method="spearman", association="pvalue", threshold="fdr",
 thr.value=0.05, weighted=FALSE)
vertexCentrality <- degreeCentralityVertexTest(expr, labels, adjacencyMatrix1,numPermutations=1) #The numPerr
vertexCentrality2<-cbind(c(4790, 4791, 4792, 4793, 84807, 4794, 4795, 64332, 595, 898, 23552,
 1017, 8099, 10263, 4609, 23077, 26292, 84073, 4610, 4613, 10408,  80177, 114897, 114898, 114899,
  114900, 114904, 114905, 390664, 338872),vertexCentrality)
pathPlot(gene.data=t(expr), cpd.data=NULL, labels=labels, varr.diff.list=vertexCentrality2,
 threshold=NULL, thr.value=1, FUN=median,species="hsa" , pathway.id="05200", kegg.native=TRUE,
  file.name="path")
```

---

readSetFile *Read a collection of variables sets (*.txt)*

Description

'readSetFile' reads a tab-delimited text file containing a collection of gene sets.

Usage

```
readSetFile(fileName)
```

Arguments

fileName	a string containing the file name

Value

a list of gene sets. Each element of the list is a character vector v, where v[1] contains the gene set name, v[2] descriptions about the set, v[3..length(v)] the genes that belong to the set.

Examples

```
# Read example set file
set_fname <- system.file("extdata", "c2.cp.v5.2.symbols.gmt", package = "BioNetStat")
deneSets <- readSetFile(set_fname)
```

---

readVarFile *Read variable values matrix*

Description

Read variable values matrix

Usage

```
readVarFile(fileName, path = NULL, dec = ".", sep = NULL, check.names = TRUE)
```

Arguments

fileName	the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, getwd().
path	the path to the directory that contains the file. Used only by Graphical Interface.
dec	the character used in the file for decimal points.
sep	the field separator character. Values on each line of the file are separated by this character. If sep = "" the separator is white space, that is one or more spaces, tabs, newlines or carriage returns, if sep=NULL (default), the function uses tabulation for .txt files or ";" for .csv files.
check.names	a logical value. If TRUE, the names of the data table kept as they are. Otherwise, the blank space, "-","/" and ",", are replaced by dots.

Value

a dataframe containing only the numeric columns of selected file. Each column is considered as a variable and each row as a sample.

Examples

```
# Glioma file
gliomaData <- system.file("extdata", "bnsData.csv", package = "BioNetStat")
varFile<-readVarFile(gliomaData)

# Random file
test1 <- as.data.frame(cbind(rep(LETTERS[1:4],each=10),matrix(rnorm(120),40,30)))
tf<-tempfile(fileext = ".csv")
write.table(test1, tf,sep=";",row.names=FALSE)
a<-readVarFile(fileName=tf)
```

runBioNetStat *Run BNS*

Description

Run BNS on the browser user interface.

Usage

```
runBioNetStat()
```

Value

open BioNetStat user interface

Examples

```
 # run  runBioNetStat() # to open user interface of BioNetStat
```

varFile *Gene expression in glioma tissues*

Description

Gene expression in glioma tissues

Usage

```
data(varFile)
```

Format

Object of class `data.frame` containing 134 variables (colunms) and 658 observations.

Source

TCGA - The Cancer Genome Atlas

References

KINKER, G. S. et al. Deletion and low expression of NFKBIA are associated with poor prognosis in lower-grade glioma patients. Scientific Reports, v. 6, n. April, p. 24160, 2016.

Examples

```
data(varFile)
# Run BNS analysis
```

Index