BioNet

October 25, 2011

`BioNet-package` *Routines for the functional analysis of biological networks*

Description

This package provides functions for the integrated analysis of biological networks and the detection of functional modules. Different datasets can be integrated into the network by assigning p-values derived from statistical tests to the nodes of the network. E.g. p-values obtained from the differential expression of genes from an Affymetrix array are assigned to the nodes of an protein-protein interaction network. By fitting a beta-uniform mixture model and calculating scores from the p-values, overall scores of network regions can be calculated and an integer linear programming algorithm identifies the maximum scoring subnetwork.

Details

Package:	BioNet
Type:	Package
Version:	1.10.1
Date:	2011-05-03
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Marcus Dittrich, Daniela Beisser

Maintainer: Marcus Dittrich <marcus.dittrich@biozentrum.uni-wuerzburg.de>

References

M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar and T. Mueller (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *(ISMB2008) Bioinformatics* 24: 13. i223-i231 Jul.

D. Beisser, G. W. Klau, T. Dandekar, T. Mueller and M. Dittrich (2010) BioNet: an R-package for the Functional Analysis of Biological Networks. *Bioinformatics* 26:08. 1129-1130 Apr.

aggrPvals *Aggregate several p-values into one p-value*

Description

The function aggregates several p-values into one p-value of p-values based on the order statistics of p-values. An overall p-value is given by the *i*th order statistic.

Usage

```
aggrPvals(pval.matrix, order, plot=TRUE)
```

Arguments

pval.matrix	Numeric matrix of p-values, columns represent different sets of p-values
order	Numeric constant, the order statistic that is used for the aggregation.
plot	Boolean value whether to plot p-value distributions.

Value

Aggregated p-value of the given order.

Author(s)

Daniela Beisser

Examples

```
data(pvaluesExample)
aggrPvals(pval.matrix=pvaluesExample, order=2)
```

bumOptim *Fitting a beta-uniform mixture model to p-value distribution*

Description

The function fits a beta-uniform mixture model to a given p-value distribution.

Usage

```
bumOptim(x, starts=1, labels=NULL)
```

Arguments

x	Numerical vector of p-values, has to be named with the gene names or the gene names can be given in the labels paramater.
starts	Number of start points for the optimization.
labels	Gene names for the p-values.

Value

List of class fb with the following elements:

`lambda`	Fitted parameter *lambda* for the beta-uniform mixture model.
`a`	Fitted parameter *a* for the beta-uniform mixture model.
`negLL`	Negative log-likelihood.
`pvalues`	P-value vector.

Author(s)

Marcus Dittrich and Daniela Beisser

References

M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, T. Mueller (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *(ISMB2008) Bioinformatics*, 24: 13. i223-i231 Jul.

S. Pounds, S.W. Morris (2003) Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values. *Bioinformatics*, 19(10): 1236-1242.

See Also

`fitBumModel`, `plot.bum`, `hist.bum`

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum <- bumOptim(x=pvals, starts=10)
bum
```

`compareNetworks` *Compare parameters of two networks*

Description

The function compares the following parameters of two networks: diameter, average degree, degree exponent, average path length and plots the cumulative degree distributions. The networks have to be connected components.

Usage

```
compareNetworks(network1, network2, plot=TRUE)
```

Arguments

`network1`	Network *graphNEL* or *igraph* format.
`network2`	Second network in *graphNEL* or *igraph* format, or subnetwork drawn from first network.
`plot`	Boolean value, whether to plot the cumulative degree distributions.

Value

A vector of network parameters is returned:

`diam.network1`
: Network diameter

`diam.network2`
: Diameter of the subnetwork

`av.degree.network1`
: Average degree of the network

`av.degree.network2`
: Average degree of the subnetwork

`degree.exponent.network1`
: Degree exponent of the network

`degree.exponent.network2`
: Degree exponent of the subnetwork

`av.path.length.network1`
: Average path lenght of the network

`av.path.length.network2`
: Average path length of the subnetwork

Author(s)

Daniela Beisser

Examples

```
library(DLBCL)
data(interactome)
subnet1 <- largestComp(subNetwork(nodes(interactome)[1:100], interactome))
subnet2 <- largestComp(subNetwork(nodes(interactome)[101:200], interactome))
compareNetworks(network1=subnet1, network2=subnet2)
```

---

fbum *Compute the density of the bum distribution*

Description

Function to compute the density of the beta-uniform mixture model.

Usage

```
fbum(x, lambda, a)
```

Arguments

`x`	A numeric value.
`lambda`	Parameter lambda, mixture parameter, proportion of uniform component
`a`	Parameter a, shape parameter of beta component

Value

Value of the density of the bum distribution for x.

Author(s)

Marcus Dittrich

References

S. Pounds, S.W. Morris (2003) Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values. *Bioinformatics*, 19(10): 1236-1242.

See Also

`bumOptim`, `fitBumModel`

Examples

```
y <- fbum(x=0.5, lambda=0.1, a=0.1)
y
```

---

fbumLL *Calculate log likelihood of BUM model*

Description

The function calculates the log likelihood of the BUM model.

Usage

```
fbumLL(parms, x)
```

Arguments

`parms`	Vector of parameters; lambda and a.
`x`	Numerical vector of p-values.

Value

Log likelihood.

Author(s)

Marcus Dittrich

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum.mle <- fitBumModel(pvals, plot=FALSE)
fbumLL(parms=c(bum.mle$lambda, bum.mle$a), x=pvals)
```

`fdrThreshold` *Calculate p-value threshold for given FDR*

Description

The function calculates the p-value threshold tau for a given false discovery rate. Tau is used for the scoring function.

Usage

```
fdrThreshold(fdr, fb)
```

Arguments

`fdr`	False discovery rate.
`fb`	Model from the beta-uniform mixture fitting.

Value

P-value threshold tau.

Author(s)

Marcus Dittrich

References

S. Pounds, S.W. Morris (2003) Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values. *Bioinformatics*, 19(10): 1236-1242.

See Also

`fbum`, `fitBumModel`

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum.mle <- fitBumModel(pvals, plot=FALSE)
tau <- fdrThreshold(fdr=0.001, fb=bum.mle)
tau
```

`fitBumModel` *Fit beta-uniform mixture model to a p-value distribution*

Description

The function fits a beta-uniform mixture model to a given p-value distribution. The BUM method was introduced by Stan Pounds and Steve Morris to model the p-value distribution as a signal-noise decompostion. The signal component is assumed to be B(a,1)-distributed, whereas the noise component is uniform-distributed under the null hypothesis.

Usage

```
fitBumModel(x, plot = TRUE)
```

Arguments

`x`	Numeric vector of p-values.
`plot`	Boolean value, whether to plot a histogram and qqplot of the p-values with the fitted model.

Value

Maximum likelihood estimator object for the fitted bum model. List of class fb with the following elements:

`lambda`	Fitted parameter *lambda* for the beta-uniform mixture model.
`a`	Fitted parameter *a* for the beta-uniform mixture model.
`negLL`	Negative log-likelihood.
`pvalues`	P-value vector.

Author(s)

Daniela Beisser

References

S. Pounds, S.W. Morris (2003) Estimating the occurrence of false positives and false negatives in microarray studies by approximating and partitioning the empirical distribution of p-values. *Bioinformatics*, 19(10): 1236-1242.

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum.mle <- fitBumModel(pvals, plot=TRUE)
bum.mle
```

getCompScores *Partition scores for subgraphs of the network*

Description

The function partitions the scores into scores for each subgraph of the network.

Usage

```
getCompScores(network, score)
```

Arguments

network	A network in *graphNEL* or *igraph* format.
score	Vector of scores.

Value

A data frame with the components of the network and the score for each PPI identifier.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(interactome)
data(dataLym)
# create random subgraph with 100 nodes and their direct neighbors
nodes <- nodes(interactome)[sample(length(nodes(interactome)), 100)]
subnet <- subNetwork(nodeList=nodes, network=interactome, neighbors="first")
score <- dataLym$score001
names(score) <- dataLym$label
getCompScores(score=score, network=subnet)
```

getEdgeList *Get representation of graph as edgelist*

Description

A network in *graphNEL* or *igraph* format is converted to an edgelist.

Usage

```
getEdgeList(network)
```

Arguments

network	Network in *graphNEL* or *igraph* format.

Value

A matrix whose columns represent the connected edges.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(interactome)
getEdgeList(interactome)[1:10,]
```

---

hist.bum *Histogram of the p-value distribution with the fitted bum model*

Description

The function plots a histogram of the p-values together with the fitted bum-model.

Usage

```
## S3 method for class 'bum'
hist(x, ...)
```

Arguments

x	Maximum likelihood estimator object of the beta-uniform mixture fit.
...	Other graphic parameters for the plot.

Author(s)

Daniela Beisser

See Also

fitBumModel, hist.bum, bumOptim

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
mle <- fitBumModel(pvals, plot=FALSE)
hist(mle)
```

largestComp *Extract largest component of network*

Description

The function extracts the largest component of a network.

Usage

```
largestComp(network)
```

Arguments

network	A graph in *graphNEL* or *igraph* format.

Value

A new graph object that represents the largest component of the given network.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(interactome)
interactome
largestComp(interactome)
```

largestScoreComp *Component with largest score*

Description

The function extracts the component of the network with the largest score. All nodes have to exceed the given level for the score.

Usage

```
largestScoreComp(network, score, level=0)
```

Arguments

network	Network in *graphNEL* or *igraph* format.
score	Vector of scores for the network.
level	Cut-off level for the score for the component.

Value

Subgraph of the network with a score larger than the given level.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(interactome)
data(dataLym)
network <- rmSelfLoops(interactome)
score <- dataLym$score001
names(score) <- dataLym$label
lComp <- largestScoreComp(network=network, score=score, level=1)
## Not run: plotModule(lComp)
```

---

loadNetwork.sif *Load network from Cytoscape sif file*

Description

The function loads a network from a Cytoscape sif file. Edge attributes are provided in the ea.file or vector of ea.files. The node attributes are provided the same way. For other formats see *read.graph* in the igraph package.

Usage

```
loadNetwork.sif(sif.file, na.file, ea.file, format=c("graphNEL", "igraph"), dire
```

Arguments

sif.file	Cytoscape sif file, containing the network.
na.file	File or vector of file with Cytoscape node attibutes.
ea.file	File or vector of file with Cytoscape edge attibutes.
format	Format of output graph, either *graphNEL* or *igraph*.
directed	Boolean value for directed or undirected graph.

Value

Graph with loaded attributes.

Author(s)

Daniela Beisser

Examples

```
## Not run: lib <- file.path(.path.package("BioNet"), "extdata")
# load interaction file, node attribute file with a node weight of 2 for each node and th
network <- loadNetwork.sif(sif.file=file.path(lib,"cytoscape.sif"), na.file=file.path(lib
network;
nodeData(network);
edgeData(network);

## End(Not run)
```

loadNetwork.tab *Load network from tabular format*

Description

The function loads a network from a tabular format.

Usage

```
loadNetwork.tab(file, header=TRUE, directed=FALSE, format=c("graphNEL", "igraph"
```

Arguments

file	File with network to load.
header	Booelan value whether to include header or not.
directed	Booelan value whether the network is to be directed or not.
format	Output format of the network, either *graphNEL* or *igraph*

Author(s)

Marcus Dittrich

See Also

loadNetwork.sif

makeNetwork *Create graph from source and target vectors*

Description

Function to create a graph in *graphNEL* or *igraph* format from a source and a target vector.

Usage

```
makeNetwork(source, target, edgemode="undirected", format=c("graphNEL", "igraph"
```

Arguments

source	Vector of source nodes.
target	Vector of corresponding target nodes.
edgemode	For an "undirected" or "directed" network.
format	Graph format, eiter *graphNEL* or *igraph*.

Value

A graph object.

Author(s)

Marcus Dittrich

See Also

loadNetwork.sif, saveNetwork

Examples

```
source <- c("a", "b", "c", "d")
target <- c("b", "c", "a", "a")
graph <- makeNetwork(source, target, edgemode="undirected")
```

---

mapByVar — *Select probeset by variance and get PPI ID*

Description

The function selects for each gene the probeset with the highest variance and gets the PPI ID for each gene. The PPI identifier is: gene symbol(Entrez ID). Affymetrix identifiers are mapped to the ENTREZ ID.

Usage

```
mapByVar(exprSet, network=NULL, attr="geneID", ignoreAFFX=TRUE)
```

Arguments

exprSet	Affymetrix ExpressionSet.
network	Network that is used to map the Affymetrix identifiers.
attr	The attribute of the network that is used to map the Affymetrix IDs. The IDs are mapped to the unique Entrez gene IDs, which are by default stored in the "geneID" attribute of the network.
ignoreAFFX	Boolean value, whether to ignore or leave AFFX control genes.

Value

Expression matrix with one gene (PPI ID) per probeset.

Author(s)

Daniela Beisser

Examples

```
## Not run: library(ALL);
data(ALL);
mapped.e.set <- mapByVar(ALL);
mapped.e.set[1:10,];
## End(Not run)
```

permutateNodes *Permute node labels*

Description

Function to permutate node labels of a given network.

Usage

```
permutateNodes(network)
```

Arguments

network	Network in *graphNEL* or *igraph* format.

Value

Network with permutated labels.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(interactome)
# remove self-loops before permutating the labels
interactome <- rmSelfLoops(interactome)
perm.net <- permutateNodes(interactome)
perm.net
```

piUpper *Upper bound pi for the fraction of noise*

Description

The function calculates the upper bound pi for the fraction of noise.

Usage

```
piUpper(fb)
```

Arguments

fb	Fitted bum model, list with parameters a and lambda.

Value

Numerical value for the upper bound pi.

Author(s)

Marcus Dittrich

See Also

bumOptim, fitBumModel

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum <- bumOptim(pvals, starts=10)
piUpper(fb=bum)
```

---

plot.bum *Quantile-quantile plot for the beta-uniform mixture model*

Description

The function plot the theoretical quantiles of the fitted bum model against the quantiles of the observed p-value distribution.

Usage

```
## S3 method for class 'bum'
plot(x, ...)
```

Arguments

x	Maximum likelihood estimation object of the fitted bum model.
...	Other graphic parameters for the plot.

Author(s)

Daniela Beisser

See Also

fitBumModel, plot.bum, bumOptim

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
mle <- fitBumModel(pvals, plot=FALSE)
plot(mle)
```

`plot3dModule` *3D plot of the network*

Description

The function plots a network from *graphNEL* or *igraph* format in 3D using a modified function from the package igraph and requires the package rgl which uses openGL. The 3D plot can be zoomed, rotated, shifted on the canvas. This function is just used to visualize the modules. For further plotting options use the rglplot function of the igraph package. If a score attribute is provided in the graph this will be used for the coloring of the nodes. Otherwise a vector of values can be given by the diff.or.score argument. The vector has to contain positive and negative values, either scores or values for differential expression (fold changes). Labels for the nodes can be provided by the labels argument, otherwise it will be automatically looked for a *geneSymbol* attribute of the nodes.

Usage

```
plot3dModule(network, labels=NULL, windowSize = c(100,100,1500,1000), diff.or.sc
```

Arguments

`network`	Network in *graphNEL* or *igraph* format.
`labels`	Labels for the nodes of the network. Otherwise it will be automatically looked for a *geneSymbol* attribute of the nodes.
`windowSize`	Numerical vector of size four to set the size of the rgl device.
`diff.or.scores`	Named numerical vector of differential expression (fold changes) or scores of the nodes in the network. These will be used for node coloring. Otherwise a *score* attribute of the nodes will be automatically used.
`red`	Either "negative" or "positive", to specify which values are to be colored red in the plot.
`...`	Other graphic parameters for the plot.

Author(s)

Daniela Beisser

See Also

`save3dModule`, `plotModule`

Examples

```
library(DLBCL)
data(interactome)
data(dataLym)
interactome <- subNetwork(dataLym$label, interactome)
interactome <- rmSelfLoops(interactome)
fchange <- dataLym$diff
names(fchange) <- dataLym$label
subnet <- largestComp(subNetwork(nodes(interactome)[1:100], interactome))
diff <- fchange[nodes(subnet)]
```

```
## Not run: library(rgl);
plot3dModule(network=subnet, diff.or.score=diff)
## End(Not run)
```

`plotLLSurface` *Log likelihood surface plot*

Description

The function plots the log likelihood surface for all a and lambda parameter of the beta-uniform mixture model.

Usage

```
plotLLSurface(x, opt=NULL, main="Log-Likelihood Surface", color.palette = heat.c
```

Arguments

`x`	Numeric vector of p-values.
`opt`	List of optimal parameters for a and lambda from the beta-uniform mixture model.
`main`	The overall title of the plot.
`color.palette`	Color scheme of the image plot.
`nlevels`	Number of color levels.

Author(s)

Marcus Dittrich

Examples

```
library(DLBCL)
data(dataLym)
pvals <- dataLym$t.pval
names(pvals) <- dataLym$label
mle <- fitBumModel(pvals, plot=FALSE)
plotLLSurface(x=pvals, opt=mle)
```

`plotModule` *Plot of the network*

Description

The function plots a network from *graphNEL* or *igraph* format, adapted from an igraph plotting function. It is just used to visualize the modules. For further plotting options use the plot.igraph function of the igraph package. The shapes of the nodes can be changed according to the scores argument, then negative scores appear squared. The color of the nodes can be changed according to the diff.expr argument. Negative values lead to green nodes, positive values are colored in red. If the vectors are not provided, it will be automatically looked for nodes attributes with the name *score* and *diff.expr*.

Usage

```
plotModule(network, layout=layout.fruchterman.reingold, labels=NULL, diff.expr=N
```

Arguments

`network`	Network in *graphNEL* or *igraph* format.
`layout`	Layout algorithm, e.g. layout.fruchterman.reingold or layout.kamada.kawai.
`labels`	Labels for the nodes of the network.
`diff.expr`	Named numerical vector of differential expression (fold changes) of the nodes in the network. These will be used for coloring of the nodes. It will be automatically looked for nodes attribute with the name *diff.expr*, if the argument is null.
`scores`	Named numerical vector of scores of the nodes in the network. These will be used for the shape of the nodes. It will be automatically looked for nodes attribute with the name *score*, if the argument is null.
`main`	Main title of the plot.
`...`	Other graphic parameters for the plot.

Author(s)

Marcus Dittrich and Daniela Beisser

See Also

`plot3dModule`

Examples

```
library(DLBCL)
data(dataLym)
data(interactome)
interactome <- subNetwork(dataLym$label, interactome)
interactome <- rmSelfLoops(interactome)
fchange <- dataLym$diff
names(fchange) <- dataLym$label
# create random subnetwork
```

```
subnet <- largestComp(subNetwork(nodes(interactome)[1:100], interactome))
fchange <- fchange[nodes(subnet)]

# color random subnetwork by the fold change
## Not run: plotModule(network=subnet, diff.expr=fchange)
```

`print.bum` *Print information about bum model*

Description

The function prints information about the bum model.

Usage

```
## S3 method for class 'bum'
print(x, ...)
```

Arguments

`x`	Maximum likelihood estimator object of the beta-uniform mixture fit.
`...`	Other graphic parameters for print.

Author(s)

Marcus Dittrich

See Also

`fitBumModel`, `summary.bum`

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
mle <- fitBumModel(pvals, plot=FALSE)
print(mle)
```

`pvaluesExample` *Example p-values for aggregation statistics*

Description

Data example consisting of a matrix of p-values. Each gene has two corresponding p-values. These p-values can be aggregated into a p-value of p-values by the method `aggrPvals`.

Usage

```
data(pvaluesExample)
```

Examples

```
data(pvaluesExample)
pvaluesExample[1:10,]
```

readHeinzGraph *Convert HEINZ output to graph*

Description

Function to convert the HEINZ output to a graph object, or if the output is in matrix form, it will create a list of graphs. The function needs the node and the original network, from which the module is calculated.

Usage

```
readHeinzGraph(node.file, network, format=c("graphNEL", "igraph"))
```

Arguments

node.file	Heinz node output file.
network	Original network from which Heinz input was created.
format	Graph format of output, either *igraph* or *graphNEL*.

Value

Graph object.

Author(s)

Daniela Beisser

Examples

```
library(DLBCL)
data(interactome)
# precomputed Heinz output files
## Not run: lib <- file.path(.path.package("BioNet"), "extdata")
module <- readHeinzGraph(node.file=file.path(lib, "lymphoma_nodes_001.txt.0.hnz"), networ
plotModule(module);

## End(Not run)
```

readHeinzTree *Convert HEINZ output to tree*

Description

Converts the HEINZ output to a tree in graph format. If the output is in matrix form, it will create a list of graphs. The function needs the node and edge file and the original network from which the module is calculated.

Usage

```
readHeinzTree(node.file, edge.file, network, format=c("graphNEL", "igraph"))
```

Arguments

node.file	Heinz node output file.
edge.file	Heinz edge output file.
network	Original network from which Heinz input was created.
format	Output format of the graph, either *igraph* or *graphNEL*.

Value

A graph object.

Author(s)

Daniela Beisser

Examples

```
library(DLBCL)
data(interactome)
# precomputed Heinz output files
## Not run: lib <- file.path(.path.package("BioNet"), "extdata")
module <- readHeinzTree(node.file=file.path(lib, "lymphoma_nodes_001.txt.0.hnz"), edge.fi
plotModule(module);

## End(Not run)
```

---

rmSelfLoops *Remove self-loops in a graph*

Description

The function removes self-loops, edges that start and end in the same node, from the network.

Usage

```
rmSelfLoops(network)
```

Arguments

network	A graph object, either in *graphNEL* or *igraph* format.

Value

The graph with the removed edges.

Author(s)

Marcus Dittrich

Examples

```
graph <- makeNetwork(c("a","b","c","d","e","a"), c("b","c","d","e","e","e"))
graph2 <- rmSelfLoops(graph)
edges(graph)
edges(graph2)
```

`runFastHeinz` *Calculate heuristically maximum scoring subnetwork*

Description

The function uses an heuristic approach to calculate the maximum scoring subnetwork. Based on the given network and scores the positive nodes are in the first step aggregated to meta-nodes between which minimum spanning trees are calculated. In regard to this, shortest paths yield the approximated maximum scoring subnetwork. This function can be used if a CPLEX license is not available to calculate the optimal solution.

Usage

```
runFastHeinz(network, scores)
```

Arguments

`network`	A graph in *igraph* or *graphNEL* format.
`scores`	A named vector, containing the scores for the nodes of the network. All nodes need to be scored in order to run the algorithm.

Value

A subnetwork in the input network format.

Author(s)

Daniela Beisser

See Also

`writeHeinzEdges`, `writeHeinzNodes`, `readHeinzTree`, `readHeinzGraph`, `runHeinz`

Examples

```
library(DLBCL)
# load p-values
data(dataLym)
# load graph
data(interactome)
# get induced subnetwork for all genes contained on the chip
interactome <- subNetwork(dataLym$label, interactome)
p.values <- dataLym$t.pval
names(p.values) <- dataLym$label
bum <- fitBumModel(p.values, plot=TRUE)
scores <- scoreNodes(network=interactome, fb=bum, fdr=0.0001)
module <- runFastHeinz(network=interactome, scores=scores)
## Not run: plotModule(module)
```

runHeinz *Start HEINZ*

Description

The function starts HEINZ from command line. The HEINZ folder has to include the heinz.py python script and the dhea file. CPLEX has to be installed and accessible from the computer R runs on.

Usage

```
runHeinz(heinz.folder="", heinz.e.file, heinz.n.file, N=TRUE, E=FALSE, diff=-1,
```

Arguments

heinz.folder	The folder which contains the heinz.py python script and the dhea file.
heinz.e.file	The HEINZ edge input file. See writeHeinzEdges
heinz.n.file	The HEINZ node input file. See writeHeinzNodes
N	Boolean value, whether to run HEINZ on nodes.
E	Boolean value, whether to run HEINZ on edges. HEINZ can run on both with N and E set to TRUE.
diff	Difference of suboptimal solutions to optimal solution in haming distance in percent. Parameter is set to -1 for optimal solution.
n	Number of optimal and suboptimal solutions, the standard n=1 delivers only the optimal solution.

Details

This function starts the integer linear programming algorithm to calculate the optimal scoring subnetwork. The algorithm might be started in the command line when the CPLEX is installed on another machine. To start it from command line use: heinz.py -e edge.file.txt -n node.file.txt -E False/True -N False/True. The results can be loaded with readHeinzTree, readHeinzGraph as a graph object.

Author(s)

Daniela Beisser

References

M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, T. Mueller (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *(ISMB2008) Bioinformatics*, 24: 13. i223-i231 Jul.

See Also

writeHeinzEdges, writeHeinzNodes, readHeinzTree, readHeinzGraph

save3dModule *Save a 3D plot of the network*

Description

The function saves a 3D plot of a network to file, therefore it requires the plot to be open. A screenshot of the 3D plot can be saved in "pdf" format. Background of the device is changed to white for plotting. The screenshot can take several seconds for large plots.

Usage

```
save3dModule(file)
```

Arguments

file	File to save to.

Author(s)

Daniela Beisser

See Also

plot3dModule, plotModule

Examples

```
library(DLBCL)
data(dataLym)
data(interactome)
interactome <- subNetwork(dataLym$label, interactome)
fchange <- dataLym$diff
names(fchange) <- dataLym$label
subnet <- largestComp(subNetwork(nodes(interactome)[1:100], interactome))
diff <- fchange[nodes(subnet)]

## Not run: library(rgl);
plot3dModule(network=subnet, diff.or.score=diff);
save3dModule(file="test")
## End(Not run)
```

saveNetwork *Save undirected network in various formats*

Description

The function saves a graph in a Cytoscape readable format: either in XGMML format, or as two tables, one for the nodes with attributes and one for the edges with attributes, or as .sif file. Or other standard formats like tab separated, .tgf, .net

Usage

```
saveNetwork(network, name="network", file, type=c("table", "XGMML", "sif", "tab"
```

Arguments

network	Network to save.
name	Name of the network, only needed for the XGMML format.
file	File to save to.
type	Type in which graph shall be saved.

Details

The format types are "XGMML", "table", "sif", "tab", "tgf" and "net". XGMML (eXtensible Graph Markup and Modeling Language) is an XML format based on GML which is used for graph description. Edges, nodes and their affiliated attributes are all saved in one file. In the table format two tables are created, one for the nodes with attributes and one for the edges with attributes. The sif format creates a .sif file for the network and an node attribute (.NA) or edge attribute (.EA) for each attribute. The name of the attribute is the filename. Tab writes only the edges of the network in a tabular format. Tgf save the network to simple .tgf format. The net format writes a Pajek readable file of the network and the ET type saves the edge tags to file.

Author(s)

Daniela Beisser and Marcus Dittrich

Examples

```
library(DLBCL)
#create small network
library(igraph)
data(interactome)
interactome <- igraph.from.graphNEL(interactome)
small.net <- subNetwork(V(interactome)[0:15]$name, interactome)
E(small.net)$e.weight <- rep(1,length(E(small.net)))
V(small.net)$n.weight <- rep(2,length(V(small.net)))
summary(small.net)
## Not run: saveNetwork(small.net, file="example_network", name="small.net", type="XGMML"
```

---

scanFDR *Dataframe of scores over a given range of FDRs*

Description

The function generates a dataframe for a given range of FDRs.

Usage

```
scanFDR(fb, fdr, labels=names(fb$pvalues))
```

Arguments

fb	Fitted bum model.
fdr	Vector of FDRs.
labels	Data frame labels.

Value

Dataframe of scores for given p-values and a range of FDRs.

Author(s)

Marcus Dittrich

See Also

bumOptim, fitBumModel

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum <- bumOptim(pvals, starts=10)
scores <- scanFDR(fb=bum, fdr=c(0.1, 0.001, 0.0001))
scores[1:10,]
```

---

scoreFunction *Scoring function for p-values*

Description

The function calculates a score for each gene with a given FDR from the fitted beta-uniform mixture model.

Usage

```
scoreFunction(fb, fdr=0.01)
```

Arguments

fb	Model from the beta-uniform mixture fitting.
fdr	Numeric constant, from the false discovery rate a p-value threshold is calculated. P-values below this threshold are considered to be significant and will score positively, p-values a bove the threshold are supposed to arise from the null model. The FDR can be used to control the size of the maximum scoring subnetwork, by zooming in and out in the same region.

Value

Score vector for the given p-values.

Author(s)

Marcus Dittrich and Daniela Beisser

References

For details on the score calculation see: M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, T. Mueller (2008) Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *(ISMB2008) Bioinformatics*, 24: 13. i223-i231 Jul.

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum.mle <- fitBumModel(pvals, plot=FALSE)
scores <- scoreFunction(fdr=0.1, fb=bum.mle)
scores
```

---

scoreNodes *Score the nodes of a network*

Description

The function derives scores from the p-values of the nodes of a network.

Usage

```
scoreNodes(network, fb, fdr=0.05)
```

Arguments

network	A network in *graphNEL* or *igraph* format.
fb	Fitted bum model.
fdr	False discovery rate.

Value

Ordered score vector for the nodes of the network.

Author(s)

Marcus Dittrich

See Also

bumOptim, fitBumModel

Examples

```
library(DLBCL)
# load p-values
data(dataLym)
# load graph
data(interactome)
# get induced subnetwork for all genes contained on the chip
chipGraph <- subNetwork(dataLym$label, interactome)
p.values <- dataLym$t.pval
names(p.values) <- dataLym$label
bum <- fitBumModel(p.values, plot=TRUE)
scoreNodes(network=chipGraph, fb=bum, fdr=0.001)
```

---

scoreOffset *Change score offset for 2 FDRs*

Description

Function to change score offset from FDR1 to FDR2.

Usage

```
scoreOffset(fb, fdr1, fdr2)
```

Arguments

fb	Model from the beta-uniform mixture fitting.
fdr1	First false discovery rate.
fdr2	Second false discovery rate.

Value

Offset for the score of the second FDR.

Author(s)

Marcus Dittrich

See Also

bumOptim, fitBumModel

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
bum <- bumOptim(pvals, starts=10)
scoreOffset(bum, fdr1=0.001, fdr2=0.000001)
```

subNetwork *Create a subGraph*

Description

The function creates a subgraph with the nodes given in the nodeList or for these nodes including their direct neighbors.

Usage

```
subNetwork(nodeList, network, neighbors=c("none", "first"))
```

Arguments

nodeList	Character vector of nodes, contained in the subgraph.
network	Graph that is used for subgraph extraction.
neighbors	Neighborhood, that is chosen for the subgraph extraction. "none" are only the selected nodes, "first" includes the direct neighbors of the selected nodes.

Value

A graph object.

Author(s)

Marcus Dittrich

Examples

```
library(igraph)
el <- cbind(c("a", "b", "c", "d", "e", "f", "d"), c("b", "c", "d", "e", "f", "a", "b"))
graph <- graph.edgelist(el, directed=TRUE)

node.list <- c("a", "b", "c")
graph2 <- subNetwork(nodeList=node.list, network=graph)
## Not run: par(mfrow=c(1,2));
plotModule(graph);
plotModule(graph2)
## End(Not run)

# or in graphNEL format:
graph3 <- igraph.to.graphNEL(graph)
graph4 <- subNetwork(nodeList=node.list, network=graph3)
graph3
graph4
```

summary.bum *Print summary of informations about bum model*

Description

The function summarizes information about the bum model.

Usage

```
## S3 method for class 'bum'
summary(object, ...)
```

Arguments

object	Maximum likelihood estimator object of the beta-uniform mixture fit.
...	Other graphic parameters for summary.

Author(s)

Daniela Beisser

See Also

fitBumModel, print.bum

Examples

```
data(pvaluesExample)
pvals <- pvaluesExample[,1]
mle <- fitBumModel(pvals, plot=FALSE)
summary(mle)
```

writeHeinz *Write input files for HEINZ*

Description

Function to write the input files with the node and edge scores for HEINZ. These files are used to calculate the maximum scoring subnetwork of the graph. The node scores are matched by their names to the nodes of the network, therefore if nodes.scores are provided as a vector or matrix, the vector has to be named, respectively the matrix has to be provided with rownames. If the network contains more nodes than the score vector, the nodes without a score are scored with the average over all nodes. If the nodes should not be scored and used for the calculation of the maximum scoring subnetwork, draw a subnetwork (subNetwork) first and use this for the argument network. The edge scores can be provided as a vector or matrix as the edge.scores argument. If no scores are provided in the arguments, but the use.node.scores or use.edge.scores argument is set to TRUE, it will be automatically looked for the "score" attribute of the nodes and edges of the network.

Usage

```
writeHeinz(network, file, node.scores=0, edge.scores=0, use.node.score=FALSE, us
```

Arguments

network	Network from which to calculate the maximum scoring subnetwork.
file	File to write to.
node.scores	Numeric vector or matrix of scores for the nodes of the network. Names of the vector or rownames of the matrix have to correspond to the PPI identifiers of the network. The scores can also be used from the node attribute "score", given one score for each node.
edge.scores	Numeric vector of scores for the edges of the network. Edge scores have to be given in the order of the edges in the network. It is better to append the edge scores as the edge attribute "score" to the network: *V(network)\$score* and set use.scores to TRUE.
use.node.score	Boolean value, whether to use the node attribute "score" in the network as node scores.
use.edge.score	Boolean value, whether to use the edge attribute "score" in the network as edge scores.

Author(s)

Daniela Beisser

See Also

writeHeinzNodes and writeHeinzEdges

Examples

```
library(DLBCL)
# use Lymphoma data and graph to find module
data(interactome)
data(dataLym)
# get induced subnetwork for all genes contained on the chip
chipGraph <- subNetwork(dataLym$label, interactome)
score <- dataLym$score001
names(score) <- dataLym$label
## Not run: writeHeinz(network=chipGraph, file="lymphoma_001", node.scores=score, edge.sc
```

---

writeHeinzEdges *Write edge input file for HEINZ*

Description

Function to write an input file for HEINZ with edge scores. If no edge scores are used, they are set to 0. In order to run HEINZ, a node input and edge input file are needed.

Usage

```
writeHeinzEdges(network, file, edge.scores=0, use.score=FALSE)
```

Arguments

network	Network from which to calculate the maximum scoring subnetwork.
file	File to write to.
edge.scores	Numeric vector of scores for the edges of the network. Edge scores have to be given in the order of the edges in the network. It is better to append the edge scores as the edge attribute "score" to the network: *V(network)\$score* and set use.score to TRUE.
use.score	Boolean value, whether to use the edge attribute "score" in the network as edge scores.

Author(s)

Daniela Beisser

See Also

writeHeinzNodes and writeHeinz

Examples

```
library(DLBCL)
# use Lymphoma data and graph to find module
data(interactome)
data(dataLym)
# get induced subnetwork for all genes contained on the chip
chipGraph <- subNetwork(dataLym$label, interactome)
# remove self loops
graph <- rmSelfLoops(chipGraph)
## Not run: writeHeinzEdges(network=graph, file="lymphoma_edges_001", use.score=FALSE)
score <- dataLym$score001
names(score) <- dataLym$label
## Not run: writeHeinzNodes(network=graph, file="lymphoma_nodes_001", node.scores=score)


# write another edge file with edge scores
library(igraph)
data(interactome)
interactome <- igraph.from.graphNEL(interactome)
small.net <- subNetwork(V(interactome)[0:15]$name, interactome)
scores <- c(1:length(E(small.net)))
E(small.net)$score <- scores
## Not run: writeHeinzEdges(network=small.net, file="test_edges", use.score=TRUE)
```

---

writeHeinzNodes *Write node input file for HEINZ*

Description

Function to write an input file with the node scores for HEINZ. This file is used together with the edge input file to calculate the maximum scoring subnetwork of the graph. The scores are matched by their names to the nodes of the network, therefore if nodes.scores are provided as a vector or matrix, the vector has to be named, respectively the matrix has to be provided with rownames.

If the network contains more nodes than the score vector, the nodes without a score are scored with the average over all nodes. If the nodes should not be scored and used for the calculation of the maximum scoring subnetwork, draw a subnetwork `subNetwork` first and use this for the argument network.

Usage

```
writeHeinzNodes(network, file, node.scores=0, use.score=FALSE)
```

Arguments

`network`	Network from which to calculate the maximum scoring subnetwork.
`file`	File to write to.
`node.scores`	Numeric vector or matrix of scores for the nodes of the network. Names of the vector or rownames of the matrix have to correspond to the PPI identifiers of the network. The scores can also be used from the node attribute "score", given one score for each node.
`use.score`	Boolean value, whether to use the node attribute "score" in the network as node scores.

Details

Use `scoreNodes` or `scoreFunction`to derive scores from a vector of p-values.

Author(s)

Daniela Beisser

See Also

`writeHeinzEdges` and `writeHeinz`

Examples

```
#create small network
library(DLBCL)
data(interactome)
small.net <- subNetwork(nodes(interactome)[0:15], interactome)
scores <- c(1:length(nodes(small.net)))
names(scores) <- nodes(small.net)
## Not run: writeHeinzNodes(network=small.net, file="test_nodes", node.scores=scores)

# use Lymphoma data and graph to find module
library(DLBCL)
data(interactome)
data(dataLym)
# get induced subnetwork for all genes contained on the chip
chipGraph <- subNetwork(dataLym$label, interactome)
## Not run: writeHeinzEdges(network=chipGraph, file="lymphoma_edges_001", use.score=FALSE
score <- dataLym$score001
names(score) <- dataLym$label
## Not run: writeHeinzNodes(network=chipGraph, file="lymphoma_nodes_001", node.scores=scc
```

Index