Package ‘DiseaseCellTypes’

June 5, 2015

Type Package

Title Identify disease-cell type associations using GSC and GSO

Version 0.10.0

Date 5 June 2015

Author Alex J. Cornish

Maintainer Alex J. Cornish `<a.cornish12@imperial.ac.uk>`

Depends R (>= 2.14), igraph

Imports gplots, Matrix, psych, snow, stringr

Suggests BiocGenerics, formatR, knitr, RUnit

Description
This package provides 2 methods for identifying disease-cell type associations: gene set compactness (GSC) and gene set overexpression (GSO). Also provided are functions for building cell type-specific interactomes, cell type-based diseasomes and simulating random walks across networks.

License GPL (>= 2)

LazyLoad yes

VignetteBuilder knitr

R topics documented:

disease.subgraph *Extract and plot a disease subgraph.*

Description

Plot a subgraph containing disease genes and their interacting partners.

Usage

```
disease.subgraph(g, genes, edge.attr="score", vert.attr="expression",
    filename.plot=NULL, filename.legend.vert=NULL, filename.legend.edge=NULL,
   n.bins=500, vert.size.disease=6, vert.size.not.disease=3, edge.width.max=20,
    vert.col.high="black", vert.col.low="lightgrey", edge.col.high="blue",
    edge.col.low="lightgrey", ...)
```

Arguments

`g`	`igraph` object. The network from which the subgraph is extracted.
`genes`	Character vector. The names of the disease-associated genes.
`edge.attr`	Character scalar. The edge attribute under which the edge scores are saved.
`vert.attr`	Character scalar. The vertex attribute under which the expression scores are saved.
`filename.plot`	Character scalar. The name of the file within which the subgraph is plotted. If `NULL`, the subgraph is not plotted.
`filename.legend.vert`	Character scalar. The name of the file within which the vertex color legend is plotted. If `NULL`, the legend is not plotted.
`filename.legend.edge`	Character scalar. The name of the file within which the edge color legend is plotted. If `NULL`, the legend is not plotted.
`n.bins`	Numeric scalar. The number of bins the colors are split into.
`vert.size.disease`	Numeric scaler. The size of the disease vertices.
`vert.size.not.disease`	Numeric scaler. The size of the non-disease vertices.
`edge.width.max`	Numeric scaler. The maximum width of the edges.
`vert.col.high`	Named character vector. The color of high-scoring vertices in any format accepted by `xspline` (colors in RGB, numeric color IDs or symbolic color names). Names must be the same as the rownames of `p.values`. If `NULL`, then all vertices colored using `col.other`.
`vert.col.low`	Named character vector. The color of low-scoring vertices.
`edge.col.high`	Named character vector. The color of high-scoring edges.
`edge.col.low`	Named character vector. The color of low-scoring edges.
`...`	Additional arguments to be passed to `pdf`.

Details

Extract a subgraph from a cell-type specific interactome containing a set of disease genes and their interacting partners.

Disease genes are represented as squares and non-disease genes as circles. Vertices are colored according to the expression of the genes, along a scale from `vert.col.low` for the lowest-weight vertices to `vert.col.high` for the highest. Edges are colored by their score, along a scale from `edge.col.low` for the lowest-weight edges to `edge.col.high` for the highest. Edges are also weighted by their score.

g should be a cell type-specific interactome, created using the `score.edges` function.

Value

The disease subgraph in the form of an `igraph` object.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

See Also

`score.edges`

Examples

```
# create cell type-specific interactome
data(edgelist.string)
data(expression.fantom5)
g <- graph.edgelist(as.matrix(edgelist.string[, c("ID.A", "ID.B")]), directed=FALSE)
expression <- expression.transform(expression.fantom5)
g.myoblast <- score.edges(g, expression[, "myoblast"])

# simulate disease genes
genes.disease <- sample(V(g.myoblast)$name, 5)

# produce cell type-specific interactome
subgraph <- disease.subgraph(g.myoblast, genes.disease)
plot(subgraph)
```

`DiseaseCellTypes.data` *Pre-processed data sets for the DiseaseCellTypes vignette*

Description

Pre-processed text-mining, gene expression, network and disease data and results from the accompanying paper.

Details

disease.classes Classes of diseases represented within `disease.genes`. Classes were found by mapping each disease to a MeSH term and identifying ancestors at the second level of the MeSH ontology. When diseases map to multiple MeSH terms, the most frequently-occuring MeSH term is selected.

disease.genes Disease-associated genes obtained from the DisGeNET database (version 2.1). These associations have been filtered, as described in Cornish et al. (2015). DisGeNET data is available from the DisGeNET website (http://www.disgenet.org/) under the Open Database License.

edgelist.string Protein-protein interactions from the STRING database (version 9.1, downloaded on 2014-09-08). Only physical interactions with a confidence score greater than 0.8 have been included. Protein identifiers have been mapped to Ensembl gene identifiers. STRING data is freely available from the STRING website (http://string-db.org) under the Creative Commons Attribution 3.0 License.

expression.fantom5 Normalized gene-wise expression counts for primary cell types from the FANTOM5 project. Samples have been normalized, grouped and combined, as described in Cornish et al. (2015). FANTOM5 data is made available from the FANTOM5 project website (http://fantom.gsc.riken.jp/) under the Creative Common Attribute 4.0 International license.

pvalues.gsc Disease-cell type association p-values computed using the gene set compactness (GSC) method. 10000 permutations were used.

pvalues.gso Disease-cell type association p-values computed using the gene set overexpression (GSO) method. 10000 permutations were used.

pvalues.text Disease-cell type association p-values computed using the text-mining method. Text mining was completed on 2014-08-31. On some occasions, multiple diseases and cell types are mapped to the same MeSH term, leading to identical rows and columns.

res.gsc Output of the `gene.set.compactness` function for use in vignette example.

References

Our paper in preparation.

Bauer-Mehren, A., Rautschka, M., Sanz, F. et al. (2010). *DisGeNET: a Cytoscape plugin to visualize, integrate, search and analyze gene-disease networks.* Bioinformatics. 26, 2924-6.

Becker, K., Barnes, K., Bright, T. et al. (2004). *The Genetic Association Database.* Nature Genetics. 36, 431-2.

Cheung, W.A, Ouellette, B.F., and Wasserman, W.W. (2012). *Inferring novel gene-disease associations using Medical Subject Heading Over-representation Profiles.* Genome Medicine. 4:75.

Forrest, A., Kawaji,. H, Rehli, M. et al. (2014). *A promoter-level mamalian expression atlas.* Nature. 507, 462-70.

Franceschini, A., Szklarczyk, D., Frankild, S. et al. (2013) *STRING v9.1: protein-protein interaction networks, with increased coverage and integration.* Nucleic Acids Research. 41, D808-15.

Jackson, D.A., Somers, K.M. and Harvey, H.H. (1989) *Similarity measures: Measures of co-occurrence and association or simply measures of co-occurrence?* The American Naturalist. 133, 436-53.

Robinson, M.D., McCarthy, D.J. and Smyth, G.K. (2010) *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.* Bioinformatics. 26, 139:40.

The UniProt Consortium (2010). *The Universal Protein Resource (UniProt) in 2010.* Nucleic Acids Research. 38, D142-8.

Examples

```
data(disease.genes)
```

`distance.rwr` *Compute graph distance using random walks with restart (RWR)*

Description

Use the random walk with restart (RWR) method to compute the distance between vertex pairs on a graph.

Usage

```
distance.rwr(g, v=V(g), edge.attr=NULL, rwr.r=0.7, rwr.cutoff=1e-5, correct.inf=TRUE)
```

Arguments

`g`	`igraph` object. The graph on which to work.
`v`	`igraph` object or numeric vector. The vertices from which each distance is computed.
`edge.attr`	Character scalar. The name of the edge attribute under which edge weights are found. If `NULL`, then all edges are assumed to have equal weight.
`rwr.r`	Numeric scalar. The restart probability.
`rwr.cutoff`	Numeric scalar. Value used to compute the change cutoff that controls when iterations are terminated.
`correct.inf`	Logical. If `TRUE`, then infinite distances produced by the RWR method are converted to the largest finite distance.

Details

Use the iterative random walk with restart (RWR) method described by Kohler et al. to compute the distances between pairs of vertices in a graph. This method is used by the `gene.set.compactness` function to compute the distances between the vertices in the vertex set. Distances are computed between each vertex in `v` and every vertex in `g`.

Let `A` be a column-normalized adjacency matrix for `g` using the edge weights specified in `edge.attr`. Larger edges weights should represent stronger interactions. `p` is a probability matrix with dimensions equal to the number of vertices in `g`. The element `p^{t}(i,j)` is the probability that at time `t`, a random walker starting from vertex `i` is located at vertex `j`. `p^0` is the initial probability distribution and an identity matrix. `r` is the restart probability specified by `rwr.r`. The final distances are computed iteratively using:

`p^(t+1) = (1-r)Ap^t + rp^0`

To save computational time, distances are only computed between vertices in `v` and vertices in `g`. Iterations are stopped when the distance (Manhattan) between `p^(t-1)` and `p^t` falls below `rwr.cutoff` multipled by the number of vertices in `v`.

Distances between vertex pairs are computed by taking the reciprocal of the final probabilities.

Value

Numeric matrix. The distances between the vertices in `v` and the vertices in `g`.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Kohler, S., Bauer, S., Horn, D. et al. (2008) *Walking the interactome for prioritization of candidate disease genes.* The American Journal of Human Genetics. 82, 949-58.

Examples

```
# create a graph and compute the vertex pair distances
g <- barabasi.game(6, directed=FALSE)
distance.rwr(g)
plot(g, layout=layout.fruchterman.reingold)
```

`expression.transform` *Transform raw gene expression values*

Description

Transform a matrix of raw gene expression values to a matrix of percentile-normalized relative gene expression scores.

Usage

```
expression.transform(x, remove.zero=TRUE)
```

Arguments

`x`	Numeric matrix. Absolute expression values. Each row should represent a gene and each column a context (i.e. a cell type).
`remove.zero`	Logical. If `TRUE`, rows in which all values equal zero are removed.

Details

Convert raw gene-wise expression values to relative expression scores for use within the `gene.set.compactness` and `gene.set.overexpression`. First, each gene-wise expression value is divided by the mean expression values across all contexts to produce relative values. These relative values are then percentile-normalized, so that the expression scores of each profile (each column in `x`) range uniformly between 0 and 1. A score of 1 indicates that the gene is the most relatively-overexpressed gene in a profile, while a score of 0 indicates that it is the most relatively-underexpressed.

Value

Numeric matrix. Percentile-normalized relative expression scores.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation.

Examples

```
data(expression.fantom5)
expression.fantom5[1:3, 1:3]

# transform gene expression using the mean method
expression.rel <- expression.transform(expression.fantom5)
expression.rel[1:3, 1:3]
```

`gene.set.compactness` *Run the gene set compactness (GSC) method*

Description

Identify associations between sets of disease-associated genes and cell types using the gene set compactness method.

Usage

```
gene.set.compactness(expression, genes, g, n.perm=10000, expression.miss=median(expression),
    rwr.r=0.7, rwr.cutoff=1e-5, parallel=NULL, verbose=TRUE)
```

Arguments

`expression`	Numeric matrix. Matrix of percentile-transformed relative expression scores. Rows should represent genes and be named with the gene name. Columns should represent contexts (i.e. cell types) and be named with the context name. `expression` should contain at least 2 columns and can be produced using the `expression.transform` function.
`genes`	Character vector. The names of the genes in the gene set to test.
`g`	`igraph` object. The PPI network used to created the observed and permuted interactomes. Gene names should be stored under a vertex attribute named `name`.
`n.perm`	Numeric scalar. Number of permutations to complete.
`expression.miss`	Numeric scalar. The expression score to give to genes not present in `expression`.
`rwr.r`	Numeric scalar. The probability under which random walks restart.
`rwr.cutoff`	Numeric scalar. The value used to determine when the iterative algorithm used to compute the random walk with restart distances terminates.
`parallel`	Numeric scalar or `NULL`. If a numeric scalar and parallel computing is available then this value determines the number of cores that the computation will be split over. See the `snow` package for further details.
`verbose`	Logical. If `TRUE`, then messages about the progress of the function are displayed.

Details

Use a permutation-based approach to identify context-specific interactomes in which a set of genes are significantly more compact. The compactness score is defined by Glaab et al. as the mean distance between pairs of vertices in a set of vertices. Here, we use random walks with restart (RWR) to compute the distance between vertex pairs, similar to the method described by Kohler et al. `rwr.r` determines the probability that a random walker will return to the start node, while `rwr.cutoff` determines when the iterative process will terminate (as described for the `distance.rwr` function).

Context-specific interactomes are created within the function by integrating expression and protein-protein interaction (PPI) data (g). Edges are re-scored using the product of the expression scores of the interactors. Expression data should represent percentile-normalised relative expression scores, created using the `expression.transform` function. The context-specific interactomes are not output by this function, but can be created separately using the `score.edges` function.

Observed interactomes are created for each context in `expression`. `n.perm` permuted interactomes are created by permuting the expression scores, then using these permuted profiles to create permuted interactomes. Empirical p-values, describing whether the gene set is significantly more compact on any of the observed interactomes, are produced for each context by computing the number of permuted compactness scores smaller than each observed compactness score.

Contexts can represent a range of biological entities, including cell types, tissues and disease states. This method is refered to as the 'gene set compactness' method in the associated paper.

Value

A object of class `dct`. This object is a list and contains the following elements

`obs`	The compactness score of the set of genes in `genes` in each of the contexts in `expression`.
`perm`	The compactness score of the set of genes in `genes` in each of the permuted interactomes. `NULL` if no permutations are completed.
`pval`	The compactness-based p-value for each of the contexts. `NA` if no permutations are completed.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation

Glaab, E., Baudot A., Krasnogor N. and Valencia A. (2010). *Extending pathways and processes using molecular interaction networks to analyse cancer genome data.* BMC Bioinformatics. 11(1): 597:607.

Kohler, S., Bauer, S., Horn, D. et al. (2008) *Walking the interactome for prioritization of candidate disease genes.* The American Journal of Human Genetics. 82, 949-958.

Cornish, A.J. and Markowetz, F. (2014) *SANTA: Quantifying the Functional Content of Molecular Networks.*. PLOS Computational Biology. 10:9, e1003808.

See Also

`expression.transform`, `plot.dct`, `score.edges`

Examples

```
# parameters used in the example
n.genes <- 50
n.contexts <- 5
gene.names <- paste("gene", 1:n.genes)
context.names <- paste("context", 1:n.contexts)

# create expression data
expression <- array(runif(n.genes * n.contexts) * 10, dim=c(n.genes, n.contexts),
    dimnames=list(gene.names, context.names))
expression <- expression.transform(expression)

# create the gene set
genes <- sample(gene.names, 4)

# create the PPI data
g <- barabasi.game(n.genes, directed=FALSE)
V(g)$name <- gene.names

# run the function
res <- gene.set.compactness(expression, genes, g, n.perm=20)
res
```

---

gene.set.overexpression *Compute gene set overexpression significance*

Description

Identify associations between sets of disease-associated genes and context (i.e. cell types) using the gene set overexpression method.

Usage

```
gene.set.overexpression(expression, genes, n.perm=10000, expression.miss=median(expression))
```

Arguments

`expression`	Numeric matrix. Matrix of percentile-transformed relative expression scores. Rows should represent genes and be named with the gene name. Columns should represent contexts (i.e. cell types) and be named with the context name. `expression` should contain at least 2 columns and can be produced using the `expression.transform` function.
`genes`	Character vector. The names of the genes in the gene set to test.
`n.perm`	Numeric scalar. Number of permutations to complete.
`expression.miss`	Numeric scalar. The expression score to give to genes not present in `expression`.

Details

Use a permutation-based appraoch to identify contexts (i.e. cell types) in which sets of genes are overexpressed. Permuted expression profiles are created by randomly redistributing the expression scores of each gene between contexts. The mean expression score of the genes in `genes` is computed for each observed expression profile and the permuted expression profiles. An empirical p-value, based on the number of permuted mean scores greater than each observed mean scores, is produced for each context.

Expression data should represent percentile-normalised relative expression scores, possibly created using the `expression.transform` function. Contexts can represent are range of biological entities, including cell types, tissues and disease states. This method is refered to as the 'gene set overexpression' method in the associated paper.

Value

A object of class `dct`. This object is a list and contains the following elements

`obs`	The observed mean expression score for the set of genes in `genes` in each of the contexts in `expression`.
`perm`	The permuted mean expression scores. `NULL` if no permutations are completed.
`pval`	The overexpression-based p-value for each of the contexts. `NA` if no permutations are completed.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation

See Also

`expression.transform`, `plot.dct`

Examples

```
# create data and run expression sig
n.genes <- 50
n.contexts <- 5
gene.names <- paste("gene", 1:n.genes)
context.names <- paste("context", 1:n.contexts)
expression <- array(runif(n.genes * n.contexts) * 10, dim=c(n.genes, n.contexts),
    dimnames=list(gene.names, context.names))
expression <- expression.transform(expression)
genes <- sample(gene.names, 4)
res <- gene.set.overexpression(expression, genes, n.perm=20)
res
```

plot.dct — *Plot the results of the* gene.set.overexpression *and* gene.set.compactness *functions*

Description

Plot the observed scores against the distribution of permuted scores output by the `gene.set.overexpression` or the `gene.set.compactness` functions

Usage

```
## S3 method for class 'dct'
plot(x, contexts=NULL, cutoff=0.05, include.pval=FALSE, ...)
```

Arguments

`x`	Object of class `dct` output by the `gene.set.overexpression` or `gene.set.compactness` functions.
`contexts`	Character vector. If not `NULL`, then these contexts are plotted. This overrides the specified `cutoff`.
`cutoff`	Numeric scalar. The observed scores of context with p-values less than this are plotted.
`include.pval`	Logical. If `TRUE`, then the computed p-value is plotted next to each observed score.
`...`	Additional arguments to be passed to `plot`.

Details

Plot the output of the `gene.set.overexpression` or `gene.set.compactness` functions. If more than 1 permutation is completed, then a histogram showing the distribution of permutation is shown in grey. Observed scores for contexts are shown as red lines. If `contexts` is `NULL`, then the observed scores of contexts with p-values less than `cutoff` are shown. If `contexts` is not `NULL`, then the observed scores of contexts specified by `contexts` are shown.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation.

See Also

`gene.set.compactness`, `gene.set.overexpression`

Examples

```
# create data to be used with gene.set.overexpression
n.genes <- 50
n.contexts <- 20
gene.names <- paste("gene", 1:n.genes)
context.names <- paste("context", 1:n.contexts)
expression <- array(runif(n.genes * n.contexts), dim=c(n.genes, n.contexts),
    dimnames=list(gene.names, context.names))
genes <- sample(gene.names, 4)
res <- gene.set.overexpression(expression, genes, n.perm=1000)

# plot the results
plot(res, cutoff=0.2)
```

---

project.network *Plot a network built from p-value correlations.*

Description

Produce and plot a correlation network from a matrix of p-values, such as the diseasome in the associated paper.

Usage

```
project.network(p.values, col.vert=NULL, col.other="white",
    method.cor=c("pearson", "kendall", "spearman"), rank.max=4,
    vert.size.max=10, vert.label.cex=1, vert.p.cutoff=0.1,
    edge.width.max=10, edge.col.diff="lightgrey", ...)
```

Arguments

`p.values`	Numeric matrix. The p-values. Each row becomes a vertex in the network.
`col.vert`	Named character vector. The color of each vertex in any format accepted by `xspline` (colors in RGB, numeric color IDs or symbolic color names). Names must be the same as the rownames of `p.values`. If `NULL`, then all vertices colored using `col.other`.
`col.other`	Character scalar. If `col.vert` is `NULL`, or the color of a vertex is `NA` or `'other'`, then the vertex is colored this color.
`method.cor`	Character scalar. The method used to compute the correlations between the rows.
`rank.max`	Numeric scalar. The number of edges to add to each vertex.
`vert.size.max`	Numeric scalar. The maximum size of the vertices.
`vert.p.cutoff`	Numeric scaler. The p-value cutoff used identify associations for vertex sizes.
`vert.label.cex`	Numeric scalar. The size of the vertex labels.
`edge.width.max`	Numeric scalar. The maximum width of the edges.
`edge.col.diff`	Character scalar. The color of edges between vertices of different colors.
`...`	Additional arguments to be passed to `plot.igraph`.

Details

Plot a correlation network computed from a matrix of p-values. The -log10 of p-values is first computed, then the correlation between rows computed using the method specified in `method.cor`. A vertex is created for each of the rows in `p.values`. Each vertex is connected to the `rank.max` vertices it correlates most strongly with. Vertices are sized by the number of significant associations they are involved in. Edge widths represent the strength of the correlation.

Vertices are colored according to `col.vert`. If an edge connects two vertices of different color, then it is colored `edge.col.diff`. If an edge connects two vertices of the same color, then it is also colored this color.

The correlation network is returned by the function (invisibly).

This function is used to create the diseasome in the associated paper.

Value

The computed correlation network in the form of an `igraph` object.

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation.

See Also

`project.network.legend`

Examples

```
# simulate p-values and create correlation network
n.diseases <- 5
n.cells <- 6
disease.names <- paste("disease", 1:n.diseases)
p.values <- array(runif(n.diseases * n.cells), dim=c(n.diseases, n.cells),
    dimnames=list(disease.names, NULL))
col.vert <- structure(rainbow(n.diseases), names=disease.names)
project.network(p.values, col.vert, rank.max=2)
```

`project.network.legend` *Plot legend for the* `project.network` *function.*

Description

Plot a legend of classes in a separate plot. To be used with the `project.network` function.

Usage

```
project.network.legend(col.classes, col.other="white", pt.cex=3, ...)
```

Arguments

`col.classes`	Named character vector. The color of each class, in any format accepted by `xspline` (colors in RGB, numeric color IDs or symbolic color names). Vector should be named with the name of each class.
`col.other`	Character scalar. Classes of this color are combined under an 'Other' class.
`pt.cex`	Numeric scalar. The expansion factor for the points.
`...`	Additional arguments to be passed to `legend`.

Details

Create a legend of classes. Classes of the same color can be combined under an 'Other' class if they all have the same color as `col.other`.

Author(s)

Alex J. Cornish <a.cornish12@imperial.ac.uk>

References

Paper in preparation.

See Also

`project.network`

Examples

```
# create classes with 'Others' and plot legend
n.classes <- 5
n.other <- 2
col.other <- "white"
class.names <- paste("Disease", 1:n.classes)
col.classes <- structure(rainbow(n.classes), names=class.names)
col.classes[sample(length(col.classes), n.other)] <- col.other
project.network.legend(col.classes, col.other)
```

`score.edges` *Build a context-specific interactome*

Description

Integrate gene expression data with a protein-protein interaction (PPI) network to build a context-specific interactome.

Usage

```
score.edges(g, expression, edge.attr.type=c("weight", "distance"),
    edge.attr="score", vertex.attr="expression",
    expression.miss=median(expression), correct.inf=TRUE)
```

Arguments

`g`	`igraph` object. The protein-protein interaction (PPI) network. Vertex names should be saved under the vertex attribute named `name`.
`expression`	Named numeric vector. The percentile-normlized relative gene expression scores, possible produced using the `expression.transform` function.
`edge.attr.type`	Character scalar. What the edge scores should represent. If `weight`, then larger scores indicate stronger interactions. If `distance`, then smaller scores indicate stronger interactions.
`edge.attr`	Character scalar. The edge attribute under which the edge scores are saved.
`vertex.attr`	Character scalar. The vertex attribute under which the expression scores are saved.
`expression.miss`	Numeric scalar. The expression score to give to genes not present in `expression`.
`correct.inf`	Logical. If `TRUE`, then expression scores equal to zero are replaced with the smallest non-zero value (to avoid infinite distances).

Details

Create a context-specific interactome by reweighting the edges of a network. These are the interactomes used by the `gene.set.compactness` function and described in the accompanying paper.

If `edge.attr.type == "weight"`, then edges with highly-expressed interacting partners are given larger edge scores, indicating stronger interactions. Weights are computed using:

`w(i,j) = x(i) * x(j)`

where `w(i,j)` is the weight of the edge connecting protein `i` and `j` and `x(i)` is the gene expression score for protein `i`. Pre-existing edge scores are not considered when edges are reweighted.

If `edge.attr.type == "distance"`, then edge scores are computed using:

`d(i,j) = 1 / (x(i) * x(j))`

where `d(i,j)` is the distance along the edge connecting protein `i` and protein `j`.

The names of the genes in `expression` should correspond to the gene names in `g`. I an gene in `g` is not represented in `expression`, then the gene is given an expression score equal to `expression.miss`.

Value

`igraph` object. The new edge scores will be saved under the edge attribute specified by `edge.attr`

Author(s)

Alex J. Cornish <`a.cornish12@imperial.ac.uk`>

References

Paper in preparation.

Examples

```
# create myoblast-specific interactome using FANTOM5 expression data
data(edgelist.string)
data(expression.fantom5)
g <- graph.edgelist(as.matrix(edgelist.string[, c("ID.A", "ID.B")]), directed=FALSE)
expression <- expression.transform(expression.fantom5)
g.myoblast <- score.edges(g, expression[, "myoblast"])
```

Index