Package 'dnet'

January 19, 2018

Type Package

Title Integrative Analysis of Omics Data in Terms of Network, Evolution and Ontology

Version 1.1.2

Date 2018-1-19

Author Hai Fang and Julian Gough

Maintainer Hai Fang <hfang@well.ox.ac.uk>

Depends R (>= 3.1.0), igraph, supraHex

Imports graph, Rgraphviz, Matrix

Suggests limma, survival, foreach, doParallel, Biobase

Description The focus of the dnet is to make sense of omics data (such as gene expression and muta-
tions) from different angles including: integration with molecular networks, enrichments us-
ing ontologies, and relevance to gene evolutionary ages. Integration is achieved to iden-
tify a gene subnetwork from the whole gene network whose nodes/genes are labelled with infor-
mative data (such as the significant levels of differential expression or sur-
vival risks). To help make sense of identified gene networks, enrichment analysis is also sup-
ported using a wide variety of pre-compiled ontologies and phylostratic gene age informa-
tion in major organisms including: human, mouse, rat, chicken, C.elegans, fruit fly, ze-
brafish and arabidopsis. Add-on functionalities are supports for calculating semantic similar-
ity between ontology terms (and between genes) and for calculating network affin-
ity based on random walk; both can be done via high-performance parallel computing.

URL http://dnet.r-forge.r-project.org, https://github.com/hfang-bristol/dnet

Collate 'dBUMfit.r'
'dBUMscore.r'
'dCheckParallel.r'
'dCommSignif.r'
'dContrast.r'
'dDAGancestor.r'
'dDAGannotate.r'
'dDAGgeneSim.r'
'dDAGinduce.r'
'dDAGlevel.r'
'dDAGreverse.r'
'dDAGroot.r'
'dDAGtermSim.r'
'dDAGtip.r'
'dEnricher.r'

1

'dEnricherView.r'
'dFDRscore.r'
'dFunArgs.r'
'dGSEA.r'
'dGSEAview.r'
'dGSEAwrite.r'
'dNetConfidence.r'
'dNetFind.r'
'dNetInduce.r'
'dNetPipeline.r'
'dNetReorder.r'
'dPvalAggregate.r'
'dRDataLoader.r'
'dRWR.r'
'dRWRcontact.r'
'dRWRpipeline.r'
'dSVDsignif.r'
'visBoxplotAdv.r'
'visDAG.r'
'visGSEA.r'
'visNet.r'
'visNetAnimate.r'
'visNetArc.r'
'visNetCircle.r'
'visNetMul.r'
'visNetReorder.r'

License GPL-2

biocViews Bioinformatics

R **topics documented:**

CLL	*Transcriptome dataset for Chronic Lymphocytic Leukemia patients from Chuang et al. (2012)*

Description

This dataset involves 130 patients with chronic lymphocytic leukemia (CLL). When enrolled in the study, these CLL patients had not received prior therapy for CLL. Additional covariate about the time to treatment (i.e. prognosis) is available. The dataset has been normalised and log2-transformed, and provided as an 'ExpressionSet' object.

Usage

```
CLL <- dRDataLoader(RData='CLL')
```

Value

an object of class "ExpressionSet". It has slots for "assayData", "phenoData", and "featureData":

- assayData: a matrix of 54675 features X 130 samples

- phenoData: variables describing sample phenotypes (i.e. columns in assayData), including information about samples: "Name" for sample names, "Time" for sampling time to first treatment (years) and "Treatment" for treatment event (1:yes, 0:no)

- featureData: variables describing features (i.e. rows in assayData), including information about features/genes: "EntrezID" for gene EntrezID, "Symbol" for gene symbol and "Desc" for gene description

References

Chuang et al. (2012). Subnetwork-based analysis of chronic lymphocytic leukemia identifies pathways that associate with disease progression. *Blood*, 120(13):2639-49.

Examples

```
CLL <- dRDataLoader(RData='CLL')
CLL
library(Biobase)
# extract information about the first 5 samples
pData(CLL)[1:5,]
# extract information about the first 5 features
fData(CLL)[1:5,]
```

---

dBUMfit   *Function to fit a p-value distribution under beta-uniform mixture model*

Description

dBUMfit is supposed to take as input a vector of p-values for deriving their distribution under beta-uniform mixture model (see Note below). The density distribution of input p-values is expressed as a mixture of two components: one for the null hypothesis (the noise component) and the other for the alternative hypothesis (the signal component). The noise component is the uniform density, while the signal component is the remainder of the mixture distribution. It returns an object of class "BUM".

Usage

```
dBUMfit(x, ntry = 1, hist.bum = T, contour.bum = T, verbose = T)
```

Arguments

x	a vector containing input p-values
ntry	an integeter specifying how many trys are used to find the optimised parameters by maximum likelihood estimation
hist.bum	logical to indicate whether the histogram graph should be drawn
contour.bum	logical to indicate whether a contour plot should be drawn to show the log likelihood as a function of two parameters (a and lambda) in the beta-uniform mixture model
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "BUM", a list with following elements:

- lambda: estimated mixture parameter
- a: estimated shape parameter
- NLL: Negative log-likelihood
- pvalues: the input pvalues
- call: the call that produced this result

Note

The probability density function of p-values under the Beta-Uniform Mixture model is formulated as: $f(x|\lambda, a) = \lambda + (1 - \lambda) * a * x^{a-1}$. The model names after mixing two distributions:

- the uniform distribution with the density function as $\frac{1}{b-a}|_{a=0}^{b=1} = 1$
- the beta distribution with the density function as $\frac{\Gamma(a+b)}{\Gamma(a)+\Gamma(b)} * x^{a-1} * (1-x)^{b-1}|_{b=1} = a * x^{a-1}$

Both are mixed via λ. The mixture parameter λ measures the contribution from the uniform distribution. Accordingly, $1 - \lambda$ measures the contribution from the beta distribution. Notably, the probability density function of the beta distribution can be splitted into two parts (rather than the exclusitive signal):

- the constant part as noise: $a * x^{a-1}|_{x=1} = a$
- the rest part as signal: $a * (x^{a-1} - 1)$

In other words, there is no signal at $x = 1$ but all being noise. It is a conservative, upper bound estimation of the noise. Therefore, the probability density function in the model can be decomposed into signal-noise components:

- the signal component: $(1 - \lambda) * a * (x^{a-1} - 1)$
- the noise component: $\lambda + (1 - \lambda) * a$

It is misleading to simply view λ as the noise component and $(1 - \lambda) * a * x^{a-1}$ as the signal component, just as wrongly do in the literatures (e.g. http://www.ncbi.nlm.nih.gov/pubmed/18586718)

See Also

dBUMscore

Examples

```
# 1) generate an vector consisting of random values from beta distribution
x <- rbeta(1000, shape1=0.5, shape2=1)

# 2) fit a p-value distribution under beta-uniform mixture model
fit <- dBUMfit(x)
fit$lambda
fit$a
```

---

dBUMscore	*Function to transform p-values into scores according to the fitted beta-uniform mixture model and/or after controlling false discovery rate*

Description

dBUMscore is supposed to take as input a vector of p-values, which are transformed into scores according to the fitted beta-uniform mixture model. Also if the FDR threshold is given, it is used to make sure that p-values below this are considered significant and thus scored positively. Instead, those p-values above the given FDR are considered insigificant and thus scored negatively.

Usage

```
dBUMscore(fit, method = c("pdf", "cdf"), fdr = NULL, scatter.bum = T)
```

Arguments

fit	an object of class "BUM"
method	the method used for the transformation. It can be either "pdf" for the method based on the probability density function of the fitted model, or "cdf" for the method based on the cumulative distribution function of the fitted model
fdr	the given FDR threshold. By default, it is set to NULL, meaning there is no constraint. If given, those p-values with the FDR below this are considered significant and thus scored positively. Instead, those p-values with the FDR above this given FDR are considered insigificant and thus scored negatively
scatter.bum	logical to indicate whether the scatter graph of scores against p-values should be drawn. Also indicated is the p-value (called tau) corresponding to the given FDR threshold (if any)

Value

- scores: a vector of scores

Note

The transformation from the input p-value x to the score $S(x)$ is based on the fitted beta-uniform mixture model with two parameters λ and a: $f(x|\lambda, a) = \lambda + (1 - \lambda) * a * x^{a-1}$. Specifically, it considers the log-likelyhood ratio between the signal and noise compoment of the model. The probability density function (pdf) of the signal component and the noise component are $(1 - \lambda) * a * (x^{a-1} - 1)$ and $\lambda + (1 - \lambda) * a$, respectively. Accordingly, the cumulative distribution function (cdf) of the signal component and the noise component are $\int_0^x (1 - \lambda) * a * (x^{a-1} - 1) \, dx$ and $\int_0^x \lambda + (1 - \lambda) * a \, dx$. In order to take into account the significance of the p-value, the fdr threshold is also used for down-weighting the score. According to how to measure both components, there are two methods implemented for deriving the score $S(x)$:

- The method "pdf": $S(x) = log_2 \frac{(1-\lambda)*a*(x^{a-1}-1)}{\lambda+(1-\lambda)*a} - log_2 \frac{(1-\lambda)*a*(\tau^{a-1}-1)}{\lambda+(1-\lambda)*a} = log_2 \left(\frac{x^{a-1}-1}{\tau^{a-1}-1} \right)$.
  For the purpose of down-weighting scores, it must ensure $log_2 \frac{(1-\lambda)*a*(\tau^{a-1}-1)}{\lambda+(1-\lambda)*a} \geq 0$, that is, the constraint via $\tau \leq \left(\frac{\lambda+2*a*(1-\lambda)}{a*(1-\lambda)} \right)^{\frac{1}{a-1}}$

- The method "cdf": $S(x) = log_2 \frac{\int_0^x (1-\lambda)*a*(x^{a-1}-1) \, dx}{\int_0^x \lambda+(1-\lambda)*a \, dx} - log_2 \frac{\int_0^\tau (1-\lambda)*a*(\tau^{a-1}-1) \, dx}{\int_0^\tau \lambda+(1-\lambda)*a \, dx} = log_2 \frac{(1-\lambda)*(x^{a-1}-a)}{\lambda+(1-\lambda)*a} -$
  $log_2 \frac{(1-\lambda)*(\tau^{a-1}-a)}{\lambda+(1-\lambda)*a} = log_2 \left(\frac{x^{a-1}-a}{\tau^{a-1}-a} \right)$. For the purpose of down-weighting scores, it must ensure $log_2 \frac{(1-\lambda)*(\tau^{a-1}-a)}{\lambda+(1-\lambda)*a} \geq 0$, that is, the constraint via $\tau \leq \left(\frac{\lambda+2*a*(1-\lambda)}{1-\lambda} \right)^{\frac{1}{a-1}}$

- Where $\tau = \left[\frac{\lambda+(1-\lambda)*a-fdr*\lambda}{fdr*(1-\lambda)} \right]^{\frac{1}{a-1}}$, i.e. the p-value corresponding to the exact fdr threshold.
  It can be deduced from the definition of the false discovery rate: $fdr \doteq \frac{\int_0^\tau \lambda+(1-\lambda)*a \, dx}{\int_0^\tau \lambda+(1-\lambda)*a*x^{a-1} \, dx}$.
  Notably, if the calculated τ exceeds the contraint, it will be reset to the maximum end of that constraint

See Also

[dBUMfit](#)

Examples

```
# 1) generate an vector consisting of random values from beta distribution
x <- rbeta(1000, shape1=0.5, shape2=1)

# 2) fit a p-value distribution under beta-uniform mixture model
fit <- dBUMfit(x)

# 3) calculate the scores according to the fitted BUM and fdr=0.01
# using "pdf" method
scores <- dBUMscore(fit, method="pdf", fdr=0.01)
# using "cdf" method
scores <- dBUMscore(fit, method="cdf", fdr=0.01)
```

---

dCheckParallel   *Function to check whether parallel computing should be used and how*

Description

dCheckParallel is used to check whether parallel computing should be used and how

Usage

```
dCheckParallel(multicores = NULL, verbose = T)
```

Arguments

multicores   an integer to specify how many cores will be registered as the multicore parallel
                backend to the 'foreach' package. If NULL, it will use a half of cores available
                in a user's computer

verbose   logical to indicate whether the messages will be displayed in the screen. By
                default, it sets to true for display

Value

TRUE for using parallel computing; FALSE otherwise

Note

Whether parallel computation with multicores is used is system-specific. Also, it will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel")).

See Also

[dRWR](#), [dRWRcontact](#), [dRWRpipeline](#), [dDAGtermSim](#), [dDAGgeneSim](#)

Examples

```
#dCheckParallel(multicores=2)
```

dCommSignif   *Function to test the significance of communities within a graph*

Description

dCommSignif is supposed to test the significance of communities within a graph. For a community
of the graph, it first calculates two types of degrees for each node: degrees based on parters only
within the community itself, and the degrees based on its parters NOT in the community but in the
graph. Then, it performs two-sample Wilcoxon tests on these two types of degrees to produce the
signficance level (p-value)

Usage

```
dCommSignif(g, comm)
```

Arguments

g   an object of class "igraph" or "graphNEL"

comm   an object of class "communities". Details on this class can be found at http:
                       //igraph.org/r/doc/communities.html

Value

- significance: a vector of p-values (significance)

Note

See Also

dCommSignif

Examples

```
# 1) generate an vector consisting of random values from beta distribution
x <- rbeta(1000, shape1=0.5, shape2=1)

# 2) fit a p-value distribution under beta-uniform mixture model
fit <- dBUMfit(x, ntry=1, hist.bum=FALSE, contour.bum=FALSE)

# 3) calculate the scores according to the fitted BUM and fdr=0.01
# using "pdf" method
scores <- dBUMscore(fit, method="pdf", fdr=0.05, scatter.bum=FALSE)
names(scores) <- as.character(1:length(scores))

# 4) generate a random graph according to the ER model
g <- erdos.renyi.game(1000, 1/100)

# 5) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 6) find the module with the maximum score
```

```
module <- dNetFind(subg, scores)

# 7) find the module and test its signficance
comm <- walktrap.community(module, modularity=TRUE)
significance <- dCommSignif(module, comm)
```

---

dContrast   *Function to help build the contrast matrix*

Description

dContrast is used to help build the contrast matrix

Usage

```
dContrast(level_sorted, contrast.type = c("average", "zero",
"sequential",
"pairwise"))
```

Arguments

level_sorted   a vector of levels (usually sorted) which are contrated to each other

contrast.type   the type of the contrast. It can be one of either 'average' for the contrast against
                the average of all levels, 'zero' for the contrast against the zero, 'sequential' for
                the contrast in a sequential order (it requires the levels being sorted properly), or
                'pairwise' for the pairwise contrast.

Value

a list with following components:

- each: the contrast being specified
- name: the name of the contrast

Note

Examples

```
level_sorted <- c("L1","L2","L3","L4")

# the contrast against the average of all levels
contrasts <- dContrast(level_sorted, contrast.type="average")

# the contrast against the zero
contrasts <- dContrast(level_sorted, contrast.type="zero")

# the contrast in a sequential order
contrasts <- dContrast(level_sorted, contrast.type="sequential")

# the pairwise contrast
contrasts <- dContrast(level_sorted, contrast.type="pairwise")
```

---

dDAGancestor   *Function to find common ancestors of two terms/nodes from a direct acyclic graph (DAG)*

Description

dDAGancestor is supposed to find a list of common ancestors shared by two terms/nodes, given a direct acyclic graph (DAG; an ontology). If two terms are given as NULL, then a sparse matrix of children x ancestors is built for all terms. If one of them is null, then a sparse matrix of children x ancestors is built but only for non-null input terms.

Usage

```
dDAGancestor(g, term1 = NULL, term2 = NULL, verbose = T)
```

Arguments

g   an object of class "igraph" or "graphNEL"

term1   the first term/node as input

term2   the second term/node as input

verbose   logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

- When two terms are given: a list of terms/nodes that are common ancestors for two input terms/nodes
- When two terms are given as NULL: a sparse matrix of children x ancestors is built for all terms, with '1' for the reachable and otherwise '0'.
- When one of terms is given as NULL: a sparse matrix of children x ancestors is built but only for non-null input terms, with '1' for the reachable and otherwise '0'.

Note

See Also

dDAGinduce

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) randomly give two terms
term1 <- sample(V(g)$name,1)
term2 <- sample(V(g)$name,1)

# 3) find common ancestors
dDAGancestor(g, term1, term2)
```

---

dDAGannotate   *Function to generate a subgraph of a direct acyclic graph (DAG) induced by the input annotation data*

Description

dDAGannotate is supposed to produce a subgraph induced by the input annotation data, given a direct acyclic graph (DAG; an ontology). The input is a graph of "igraph" or "graphNET" object, a list of the vertices containing annotation data, and the mode defining the paths to the root of DAG. The induced subgraph contains vertices (with annotation data) and their ancestors along with the defined paths to the root of DAG. The annotations at these vertices (including their ancestors) are also updated according to the true-path rule: a gene annotated to a term should also be annotated by its all ancestor terms.

Usage

```
dDAGannotate(g, annotations, path.mode = c("all_paths",
"shortest_paths",
"all_shortest_paths"), verbose = TRUE)
```

Arguments

g   an object of class "igraph" or "graphNEL"

annotations   the vertices/nodes for which annotation data are provided

path.mode   the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

verbose   logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

- subg: an induced subgraph, an object of class "igraph". In addition to the original attributes to nodes and edges, the return subgraph is also appended by new node attributes: "annotations", which contains a list of genes either as original annotations or inherited annotations; "IC", which stands for information content defined as negative 10-based log-transformed frequency of genes annotated to that term.

Note

For the mode "shortest_paths", the induced subgraph is the most concise, and thus informative for visualisation when there are many nodes in query, while the mode "all_paths" results in the complete subgraph.

See Also

[dDAGinduce](), [dDAGlevel]()

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) load human genes annotated by HPPA
org.Hs.egHPPA <- dRDataLoader(RData='org.Hs.egHPPA')
GS <- org.Hs.egHPPA # as 'GS' object

# 3) prepare for annotation data
# randomly select vertices with annotation data
annotations <- GS$gs[sample(1:length(GS$gs),5)]

# 4) obtain the induced subgraph
# 4a) based on all possible paths (i.e. the complete subgraph induced)
dDAGannotate(g, annotations, path.mode="all_paths", verbose=TRUE)
# 4b) based on shortest paths (i.e. the most concise subgraph induced)
dag <- dDAGannotate(g, annotations, path.mode="shortest_paths",
verbose=TRUE)

# 5) color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$annotations, length)
names(data) <- V(dag)$name
visDAG(g=dag, data=data, node.info="both")
```

---

dDAGgeneSim	*Function to calculate pair-wise semantic similarity between genes based on a direct acyclic graph (DAG) with annotated data*

Description

dDAGgeneSim is supposed to calculate pair-wise semantic similarity between genes based on a direct acyclic graph (DAG) with annotated data. It first calculates semantic similarity between terms and then derives semantic similarity between genes from terms-term semantic similarity. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
dDAGgeneSim(g, genes = NULL, method.gene = c("BM.average", "BM.max",
"BM.complete", "average", "max"), method.term = c("Resnik", "Lin",
"Schlicker", "Jiang", "Pesquita"), force = TRUE, fast = TRUE,
parallel = TRUE, multicores = NULL, verbose = TRUE)
```

Arguments

g	an object of class "igraph" or "graphNEL". It must contain a vertex attribute called 'annotations' for storing annotation data (see example for howto)
genes	the genes between which pair-wise semantic similarity is calculated. If NULL, all genes annotatable in the input dag will be used for calculation, which is very prohibitively expensive!

method.gene   the method used for how to derive semantic similarity between genes from se-
               mantic similarity between terms. It can be "average" for average similarity be-
               tween any two terms (one from gene 1, the other from gene 2), "max" for the
               maximum similarity between any two terms, "BM.average" for best-matching
               (BM) based average similarity (i.e. for each term of either gene, first calcu-
               late maximum similarity to any term in the other gene, then take average of
               maximum similarity; the final BM-based average similiary is the pre-calculated
               average between two genes in pair), "BM.max" for BM based maximum sim-
               ilarity (i.e. the same as "BM.average", but the final BM-based maximum si-
               miliary is the maximum of the pre-calculated average between two genes in
               pair), "BM.complete" for BM-based complete-linkage similarity (inspired by
               complete-linkage concept: the least of any maximum similarity between a term
               of one gene and a term of the other gene). When comparing BM-based simi-
               larity between genes, "BM.average" and "BM.max" are sensitive to the number
               of terms invovled; instead, "BM.complete" is much robust in this aspect. By
               default, it uses "BM.average".

method.term   the method used to measure semantic similarity between terms. It can be "Resnik"
               for information content (IC) of most informative common ancestor (MICA) (see
               http://arxiv.org/pdf/cmp-lg/9511007.pdf), "Lin" for 2*IC at MICA di-
               vided by the sum of IC at pairs of terms (see https://www.cse.iitb.ac.
               in/~cs626-449/Papers/WordSimilarity/3.pdf), "Schlicker" for weighted
               version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/
               pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs
               of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.
               pdf), "Pesquita" for graph information content similarity related to Tanimoto-
               Jacard index (ie. summed information content of common ancestors divided
               by summed information content of all ancestors of term1 and term2 (see http:
               //www.ncbi.nlm.nih.gov/pubmed/18460186))

force   logical to indicate whether the only most specific terms (for each gene) will be
               used. By default, it sets to true. It is always advisable to use this since it is
               computationally fast but without compromising accuracy (considering the fact
               that true-path-rule has been applied when running dDAGannotate)

fast   logical to indicate whether a vectorised fast computation is used. By default, it
               sets to true. It is always advisable to use this vectorised fast computation; since
               the conventional computation is just used for understanding scripts

parallel   logical to indicate whether parallel computation with multicores is used. By de-
               fault, it sets to true, but not necessarily does so. It will depend on whether these
               two packages "foreach" and "doParallel" have been installed. It can be installed
               via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParalle
               If not yet installed, this option will be disabled

multicores   an integer to specify how many cores will be registered as the multicore parallel
               backend to the 'foreach' package. If NULL, it will use a half of cores available in
               a user's computer. This option only works when parallel computation is enabled

verbose   logical to indicate whether the messages will be displayed in the screen. By
               default, it sets to true for display

Value

It returns a sparse matrix containing pair-wise semantic similarity between input genes. This sparse
matrix can be converted to the full matrix via the function as.matrix

Note

For the mode "shortest_paths", the induced subgraph is the most concise, and thus informative for visualisation when there are many nodes in query, while the mode "all_paths" results in the complete subgraph.

See Also

[dDAGtermSim](), [dDAGinduce](), [dDAGtip](), [dCheckParallel]()

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) load human genes annotated by HPPA
org.Hs.egHPPA <- dRDataLoader(RData='org.Hs.egHPPA')

# 3) prepare for ontology and its annotation information
dag <- dDAGannotate(g, annotations=org.Hs.egHPPA,
path.mode="all_paths", verbose=TRUE)

# 4) calculate pair-wise semantic similarity between 5 randomly chosen genes
allgenes <- unique(unlist(V(dag)$annotations))
genes <- sample(allgenes,5)
sim <- dDAGgeneSim(g=dag, genes=genes, method.gene="BM.average",
method.term="Resnik", parallel=FALSE, verbose=TRUE)
sim
```

---

dDAGinduce   *Function to generate a subgraph of a direct acyclic graph (DAG) induced by given vertices*

Description

dDAGinduce is supposed to produce a subgraph induced by given vertices, given a direct acyclic graph (DAG; an ontology). The input is a graph of "igraph" or "graphNET" object, a list of the vertices of the graph, and the mode defining the paths to the root of DAG. The resultant subgraph inherits the class from the input one. The induced subgraph contains exactly the vertices of interest and their defined paths to the root of DAG.

Usage

```
dDAGinduce(g, nodes_query, path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"))
```

Arguments

g	an object of class "igraph" or "graphNEL"
nodes_query	the vertices for which the calculation is performed
path.mode	the mode of paths induced by nodes in query. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

Value

- subg: an induced subgraph, an object of class "igraph" or "graphNEL"

Note

For the mode "shortest_paths", the induced subgraph is the most concise, and thus informative for visualisation when there are many nodes in query, while the mode "all_paths" results in the complete subgraph.

See Also

dDAGroot

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) randomly select vertices as the query nodes
# the query nodes can be igraph vertex sequences
nodes_query <- sample(V(g),5)
# more commonly, the query nodes can be term id
nodes_query <- sample(V(g),5)$name

# 3) obtain the induced subgraph
# 3a) based on all possible paths (i.e. the complete subgraph induced)
subg <- dDAGinduce(g, nodes_query, path.mode="all_paths")
# 3b) based on shortest paths (i.e. the most concise subgraph induced)
subg <- dDAGinduce(g, nodes_query, path.mode="shortest_paths")
```

---

dDAGlevel	*Function to define/calculate the level of nodes in a direct acyclic graph (DAG)*

Description

dDAGlevel is supposed to calculate the level of nodes, given a direct acyclic graph (DAG; an ontology). The input is a graph of "igraph" or "graphNET" object, and the definition of the node level. The return can be the level for each node or the nodes for each level.

Usage

```
dDAGlevel(g, level.mode = c("longest_path", "shortest_path"),
return.mode = c("node2level", "level2node"))
```

Arguments

g   an object of class "igraph" or "graphNEL"

level.mode   the mode of how to define the level of nodes in DAG. It can be "longest_path"
                for defining the node level as the length of the longest path from the node to the
                root, and "shortest_paths" for defining the node level as the length of the shortest
                path from the node to the root

return.mode   the mode of how to return the node level information. It can be "node2level"
                     for returning a named vector (i.e. the level for each node), and "level2node" for
                     returning a named list (i.e. nodes for each level)

Value

When "return.mode" is "node2level", it returns a named vector: for each named node (i.e. Term
ID), it stores its level When "return.mode" is "level2node", it returns a named list: for each named
level, it contains the names (i.e. Term ID) of nodes belonging to this level

Note

The level for the root is 1. The level based on the longest path will ensure that nodes at the same
level will never be reachable (i.e. in the same path), while the level based on the shortest path will
not be necessary. The "longest path" based level can be useful in visiting nodes from the tipmost
level to the root: 1) for the current node, all children have been visited; 2) nodes at the same level
can be looked at independantly. The "shortest path" based level can be useful in deriving nodes
according to their closeness to the root.

See Also

[dDAGroot](), [dDAGreverse]()

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) randomly select vertices as the query nodes
nodes_query <- sample(V(g),5)$name

# 3) obtain the complete subgraph induced
subg <- dDAGinduce(g, nodes_query)

# 4) calculate the node levels
# 4a) definition based on the longest path
dDAGlevel(subg, level.mode="longest_path")
# 4b) definition based on the shortest path
dDAGlevel(subg, level.mode="shortest_path")
# 4c) definition based on the longest path, and return nodes for each level
dDAGlevel(subg, level.mode="longest_path", return.mode="level2node")
```

---

dDAGreverse   *Function to reverse the edge direction of a direct acyclic graph (DAG)*

Description

dDAGreverse is supposed to reverse the edge direction of a direct acyclic graph (DAG; an ontology).
The return graph remains all attributes associated on nodes and edges.

Usage

```
dDAGreverse(g)
```

Arguments

g   an object of class "igraph" or "graphNEL"

Value

- `gr`: a graph being reversed, an object of class "igraph" or "graphNEL"

Note

See Also

[dDAGreverse](#)

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) the graph with reverse edge direction
gr <- dDAGreverse(g)
gr
```

---

dDAGroot   *Function to find the root node of a direct acyclic graph (DAG)*

Description

`dDAGroot` is supposed to find the root node of a direct acyclic graph (DAG; an ontology). It return the name (i.e Term ID) of the root node.

Usage

```
dDAGroot(g)
```

Arguments

g   an object of class "igraph" or "graphNEL"

Value

- `root`: the root name (i.e. Term ID)

Note

See Also

[dDAGroot](#)

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) find the root
root <- dDAGroot(g)
root
```

---

dDAGtermSim   *Function to calculate pair-wise semantic similarity between input*
                                     terms based on a direct acyclic graph (DAG) with annotated data

---

Description

dDAGtermSim is supposed to calculate pair-wise semantic similarity between input terms based on
a direct acyclic graph (DAG) with annotated data. Parallel computing is also supported for Linux
or Mac operating systems.

Usage

```
dDAGtermSim(g, terms = NULL, method = c("Resnik", "Lin", "Schlicker",
"Jiang", "Pesquita"), fast = T, parallel = TRUE, multicores = NULL,
verbose = T)
```

Arguments

g   an object of class "igraph" or "graphNEL". It must contain a vertex attribute
                called 'annotations' for storing annotation data (see example for howto)

terms   the terms/nodes between which pair-wise semantic similarity is calculated. If
                NULL, all terms in the input DAG will be used for calcluation, which is very
                prohibitively expensive!

method   the method used to measure semantic similarity between input terms. It can
                be "Resnik" for information content (IC) of most informative common ancestor
                (MICA) (see http://arxiv.org/pdf/cmp-lg/9511007.pdf), "Lin" for 2*IC
                at MICA divided by the sum of IC at pairs of terms (see https://www.cse.
                iitb.ac.in/~cs626-449/Papers/WordSimilarity/3.pdf), "Schlicker" for
                weighted version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.
                nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC
                at pairs of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/
                9709008.pdf), "Pesquita" for graph information content similarity related to
                Tanimoto-Jacard index (ie. summed information content of common ancestors
                divided by summed information content of all ancestors of term1 and term2
                (see http://www.ncbi.nlm.nih.gov/pubmed/18460186)). By default, it uses
                "Schlicker" method

fast   logical to indicate whether a vectorised fast computation is used. By default, it
                sets to true. It is always advisable to use this vectorised fast computation; since
                the conventional computation is just used for understanding scripts

parallel	logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParalle If not yet installed, this option will be disabled
multicores	an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

It returns a sparse matrix containing pair-wise semantic similarity between input terms. This sparse matrix can be converted to the full matrix via the function as.matrix

Note

See Also

[dDAGinduce](), [dDAGancestor](), [dDAGgeneSim](), [dCheckParallel]()

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) load human genes annotated by HPPA
org.Hs.egHPPA <- dRDataLoader(RData='org.Hs.egHPPA')

# 3) prepare for ontology and its annotation information
dag <- dDAGannotate(g, annotations=org.Hs.egHPPA,
path.mode="all_paths", verbose=TRUE)

# 4) calculate pair-wise semantic similarity between 5 randomly chosen terms
terms <- sample(V(dag)$name, 5)
sim <- dDAGtermSim(g=dag, terms=terms, method="Schlicker",
parallel=FALSE)
sim
```

---

dDAGtip   *Function to find the tip node(s) of a direct acyclic graph (DAG)*

Description

dDAGtip is supposed to find the tip node(s) of a direct acyclic graph (DAG; an ontology). It return the name (i.e Term ID) of the tip node(s).

Usage

```
dDAGtip(g)
```

Arguments

g   an object of class "igraph" or "graphNEL"

Value

- tip: the tip name (i.e. Term ID)

Note

See Also

[dDAGtip](#)

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) find tips
tips <- dDAGtip(g)
tips
```

---

dEnricher   *Function to conduct enrichment analysis given the input data and the ontology in query*

Description

dEnricher is supposed to conduct enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test. The test can respect the hierarchy of the ontology.

Usage

```
dEnricher(data, identity = c("symbol", "entrez"),
check.symbol.identity = FALSE, genome = c("Hs", "Mm", "Rn", "Gg", "Ce",
"Dm", "Da", "At"), ontology = c("GOBP", "GOMF", "GOCC", "PS", "PS2",
"SF",
"DO", "HPPA", "HPMI", "HPCM", "HPMA", "MP", "MsigdbH", "MsigdbC1",
"MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME",
"MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN",
"MsigdbC4CM",
"MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7",
"DGIdb"),
sizeRange = c(10, 1000), min.overlap = 3, which_distance = NULL,
```

```
test = c("HypergeoTest", "FisherTest", "BinomialTest"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/dnet/1.0.7")
```

Arguments

data	an input vector. It contains either Entrez Gene ID or Symbol
identity	the type of gene identity (i.e. row names of input data), either "symbol" for gene symbols (by default) or "entrez" for Entrez Gene ID. The option "symbol" is preferred as it is relatively stable from one update to another; also it is possible to search against synonyms (see the next parameter)
check.symbol.identity	
	logical to indicate whether synonyms will be searched against when gene symbols cannot be matched. By default, it sets to FALSE since it may take a while to do such check using all possible synoyms
genome	the genome identity. It can be one of "Hs" for human, "Mm" for mouse, "Rn" for rat, "Gg" for chicken, "Ce" for c.elegans, "Dm" for fruitfly, "Da" for zebrafish, and "At" for arabidopsis
ontology	the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PS" for phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for domain superfamily assignments, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, and Drug-Gene Interaction database (DGIdb) and the molecular signatures database (Msigdb) only in human (including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7"). Note: These four ("GOBP", "GOMF", "GOCC" and "PS") are availble for all genomes/species; for "Hs" and "Mm", these six ("DO", "HPPA", "HPMI", "HPCM", "HPMA" and "MP") are also supported; all "Msigdb" are only supported in "Hs". For details on the eligibility for pairs of input genome and ontology, please refer to the online Documentations at http://supfam.org/dnet/docs.html
sizeRange	the minimum and maximum size of members of each gene set in consideration. By default, it sets to a minimum of 10 but no more than 1000
min.overlap	the minimum number of overlaps. Only those gene sets that overlap with input data at least min.overlap (3 by default) will be processed
which_distance	which distance of terms in the ontology is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances
test	the statistic test used. It can be "FisherTest" for using fisher's exact test, "HypergeoTest" for using hypergeometric test, or "BinomialTest" for using binomial

test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > binomial test > fisher's exact test. In other words, in terms of the calculated p-value, hypergeometric test < binomial test < fisher's exact test

p.adjust.method
the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

ontology.algorithm
the algorithm used to account for the hierarchy of the ontology. It can be one of "none", "pc", "elim" and "lea". For details, please see 'Note'

elim.pvalue   the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)

lea.depth   the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is uded to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)

verbose   logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

RData.location   the characters to tell the location of built-in RData files. By default, it remotely locates at https://github.com/hfang-bristol/RDataCentre/blob/master/dnet and http://dnet.r-forge.r-project.org/RData. Be aware of several versions and the latest one is matched to the current package version. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget - r - l2 - A" * .RData" - np - nH - -cut - dirs = 0" http : //dnet.r - forge.r - project.org/RData"$

Value

an object of class "eTerm", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene set in consideration, and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

- `data`: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained

- `overlap`: a list of overlapped gene sets, each storing genes overlapped between a gene set and the given input data (i.e. the genes of interest). Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

- `zscore`: a vector containing z-scores

- `pvalue`: a vector containing p-values

- `adjp`: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons

- `call`: the call that produced this result

Note

The interpretation of the algorithms used to account for the hierarchy of the ontology is:

- "none": does not consider the ontology hierarchy at all.

- "lea": computers the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once genes are already annotated to any children terms with a more signfance than itself, then all these genes are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.

- "elim": computers the significance of a term in terms of the significance of its all children. Precisely, once genes are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these genes are eliminated from the ancestors of that term).

- "pc": requires the significance of a term not only using the whole genes as background but also using genes annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.

- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

[dEnricherView](#)

Examples

```
# load data
library(Biobase)
TCGA_mutations <- dRDataLoader(RData='TCGA_mutations')
symbols <- as.character(fData(TCGA_mutations)$Symbol)

# Enrichment analysis using Disease Ontology (DO)
data <- symbols[1:100] # select the first 100 human genes
eTerm <- dEnricher(data, identity="symbol", genome="Hs", ontology="DO")

# visualise the top significant terms in the ontology hierarchy
ig.DO <- dRDataLoader(RData='ig.DO')
```

```
g <- ig.DO
nodes_query <- names(sort(eTerm$adjp)[1:5])
nodes.highlight <- rep("red", length(nodes_query))
names(nodes.highlight) <- nodes_query
subg <- dDAGinduce(g, nodes_query)
# color-code terms according to the adjust p-values (taking the form of 10-based negative logarithm)
visDAG(g=subg, data=-1*log10(eTerm$adjp[V(subg)$name]),
node.info="both", zlim=c(0,2), node.attrs=list(color=nodes.highlight))
# color-code terms according to the z-scores
visDAG(g=subg, data=eTerm$zscore[V(subg)$name], node.info="both",
colormap="darkblue-white-darkorange",
node.attrs=list(color=nodes.highlight))
```

---

dEnricherView   *Function to view enrichment results of dEnricher*

Description

dEnricherView is supposed to view results of enrichment analysis by dEnricher.

Usage

```
dEnricherView(eTerm, top_num = 10, sortBy = c("adjp", "pvalue",
"zscore",
"nAnno", "nOverlap", "none"), decreasing = NULL, details = F)
```

Arguments

eTerm   an object of class "eTerm"

top_num   the maximum number of gene sets (terms) will be viewed

sortBy   which statistics will be used for sorting and viewing gene sets (terms). It can
                be "adjp" for adjusted p value, "pvalue" for p value, "zscore" for enrichment
                z-score, "nAnno" for the number of sets (terms), "nOverlap" for the number in
                overlaps, and "none" for ordering according to ID of gene sets (terms)

decreasing   logical to indicate whether to sort in a decreasing order. If it is null, it would be
                true for "zscore", "nAnno" or "nOverlap"; otherwise it would be false

details   logical to indicate whether the detailed information of gene sets (terms) is also
                viewed. By default, it sets to false for no inclusion

Value

a data frame with following components:

- setID: term ID; as rownames
- name: term name
- nAnno: number in gene members annotated by a term
- nOverlap: number in overlaps
- zscore: enrichment z-score
- pvalue: nominal p value
- adjp: adjusted p value

- namespace: term namespace; optional, it is only appended when "details" is true
- distance: term distance; optional, it is only appended when "details" is true
- members: members (represented as Gene Symbols) in overlaps; optional, it is only appended when "details" is true

Note

See Also

dEnricher

Examples

```
#dEnricherView(eTerm, top_num=10, sortBy="adjp", decreasing=FALSE, details=TRUE)
```

---

dFDRscore   *Function to transform fdr into scores according to log-likelihood ratio between the true positives and the false positivies and/or after controlling false discovery rate*

Description

dFDRscore is supposed to take as input a vector of fdr, which are transformed into scores according to log-likelihood ratio between the true positives and the false positivies. Also if the FDR threshold is given, it is used to make sure that fdr below threshold are considered significant and thus scored positively. Instead, those fdr above the given threshold are considered insigificant and thus scored negatively.

Usage

```
dFDRscore(fdr, fdr.threshold = NULL, scatter = F)
```

Arguments

fdr	a vector containing a list of input fdr
fdr.threshold	the given FDR threshold. By default, it is set to NULL, meaning there is no constraint. If given, those fdr with the FDR below threshold are considered significant and thus scored positively. Instead, those fdr with the FDR above given threshold are considered insigificant and thus scored negatively
scatter	logical to indicate whether the scatter graph of scores against p-values should be drawn. Also indicated is the score corresponding to the given FDR threshold (if any)

Value

- scores: a vector of scores

Note

See Also

[dSVDsignif](), [dNetPipeline]()

Examples

```
# 1) generate data with an iid matrix of 1000 x 9
data <- cbind(matrix(rnorm(1000*3,mean=0,sd=1), nrow=1000, ncol=3),
matrix(rnorm(1000*3,mean=0.5,sd=1), nrow=1000, ncol=3),
matrix(rnorm(1000*3,mean=-0.5,sd=1), nrow=1000, ncol=3))

# 2) calculate the significance according to SVD
# using "fdr" significance
fdr <- dSVDsignif(data, signif="fdr", num.permutation=10)

# 3) calculate the scores according to the fitted BUM and fdr=0.01
# no fdr threshold
scores <- dFDRscore(fdr)
# using fdr threshold of 0.01
scores <- dFDRscore(fdr, fdr.threshold=0.1, scatter=TRUE)
```

---

dFunArgs | *Function to assign (and evaluate) arguments with default values for an input function*

Description

dFunArgs is supposed to assign (and evaluate) arguments with default values for an input function.

Usage

```
dFunArgs(fun, action = F, verbose = T)
```

Arguments

fun	an input function name (character string)
action	logical to indicate whether the function will act as it should be (with assigned values in the current environment). By default, it sets to FALSE
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

a list containing arguments and their default values

Note

This function is potentially useful when debugging. Because the developer does not have to specify default values for all arguments except those arguments are of interest

See Also

[dNetPipeline]()

Examples

```
fun <- "dNetPipeline"
dFunArgs(fun)
```

---

dGSEA	*Function to conduct gene set enrichment analysis given the input data and the ontology in query*

Description

dGSEA is supposed to conduct gene set enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm".

Usage

```
dGSEA(data, identity = c("symbol", "entrez"), check.symbol.identity =
FALSE,
genome = c("Hs", "Mm", "Rn", "Gg", "Ce", "Dm", "Da", "At"),
ontology = c("GOBP", "GOMF", "GOCC", "PS", "PS2", "SF", "DO", "HPPA",
"HPMI", "HPCM", "HPMA", "MP", "MsigdbH", "MsigdbC1", "MsigdbC2CGP",
"MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA",
"MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM",
"MsigdbC5BP",
"MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7", "DGIdb",
"Customised"),
customised.genesets = NULL, sizeRange = c(10, 20000),
which_distance = NULL, weight = 1, nperm = 1000, fast = T,
sigTail = c("two-tails", "one-tail"), p.adjust.method = c("BH", "BY",
"bonferroni", "holm", "hochberg", "hommel"), verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/dnet/1.0.7")
```

Arguments

data	a data frame or matrix of input data. It must have row names, either Entrez Gene ID or Symbol
identity	the type of gene identity (i.e. row names of input data), either "symbol" for gene symbols (by default) or "entrez" for Entrez Gene ID. The option "symbol" is preferred as it is relatively stable from one update to another; also it is possible to search against synonyms (see the next parameter)
check.symbol.identity	logical to indicate whether synonyms will be searched against when gene symbols cannot be matched. By default, it sets to FALSE since it may take a while to do such check using all possible synoyms
genome	the genome identity. It can be one of "Hs" for human, "Mm" for mouse, "Rn" for rat, "Gg" for chicken, "Ce" for c.elegans, "Dm" for fruitfly, "Da" for zebrafish, and "At" for arabidopsis

ontology   the ontology supported currently. It can be "GOBP" for Gene Ontology Bi-
                            ological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC"
                            for Gene Ontology Cellular Component, "PS" for phylostratific age informa-
                            tion, "PS2" for the collapsed PS version (inferred ancestors being collapsed
                            into one with the known taxonomy information), "SF" for domain superfam-
                            ily assignments, "DO" for Disease Ontology, "HPPA" for Human Phenotype
                            Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance,
                            "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Pheno-
                            type Mortality Aging, "MP" for Mammalian Phenotype, and Drug-Gene Inter-
                            action database (DGIdb) and the molecular signatures database (Msigdb) only in
                            human (including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP",
                            "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT",
                            "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF",
                            "MsigdbC5CC", "MsigdbC6", "MsigdbC7"). Note: These four ("GOBP", "GOMF",
                            "GOCC" and "PS") are availble for all genomes/species; for "Hs" and "Mm",
                            these six ("DO", "HPPA", "HPMI", "HPCM", "HPMA" and "MP") are also
                            supported; all "Msigdb" are only supported in "Hs". For details on the eligi-
                            bility for pairs of input genome and ontology, please refer to the online Docu-
                            mentations at http://supfam.org/dnet/docs.html. Also supported are the
                            user-customised gene sets; in doing so, the option "Customised" should be used
                            together with the input of the next parameter "customised.genesets"

customised.genesets
                            an input vector/matrix/list which only works when the user chooses "Customised"
                            in the previous parameter "ontology". It contains either Entrez Gene ID or Sym-
                            bol

sizeRange   the minimum and maximum size of members of each gene set in consideration.
                            By default, it sets to a minimum of 10 but no more than 1000

which_distance   which distance of terms in the ontology is used to restrict terms in consideration.
                            By default, it sets to 'NULL' to consider all distances

weight   type of score weight. It can be "0" for unweighted (an equivalent to Kolmogorov-
                            Smirnov, only considering the rank), "1" for weighted by input gene score (by
                            default), and "2" for over-weighted, and so on

nperm   the number of random permutations. For each permutation, gene-score associa-
                            tions will be permutated so that permutation of gene-term associations is realised

fast   logical to indicate whether to fast calculate expected results from permutated
                            data. By default, it sets to true

sigTail   the tail used to calculate the statistical significance. It can be either "two-tails"
                            for the significance based on two-tails or "one-tail" for the significance based on
                            one tail

p.adjust.method
                            the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni",
                            "holm", "hochberg" and "hommel". The first two methods "BH" (widely used)
                            and "BY" control the false discovery rate (FDR: the expected proportion of false
                            discoveries amongst the rejected hypotheses); the last four methods "bonfer-
                            roni", "holm", "hochberg" and "hommel" are designed to give strong control of
                            the family-wise error rate (FWER). Notes: FDR is a less stringent condition
                            than FWER

verbose   logical to indicate whether the messages will be displayed in the screen. By
                            default, it sets to false for no display

RData.location   the characters to tell the location of built-in RData files. By default, it remotely
locates at https://github.com/hfang-bristol/RDataCentre/blob/master/
dnet and http://dnet.r-forge.r-project.org/RData. Be aware of several
versions and the latest one is matched to the current package version. For the
user equipped with fast internet connection, this option can be just left as de-
fault. But it is always advisable to download these files locally. Especially when
the user needs to run this function many times, there is no need to ask the func-
tion to remotely download every time (also it will unnecessarily increase the
runtime). For examples, these files (as a whole or part of them) can be first
downloaded into your current working directory, and then set this option as:
$RData.location = "."$. Surely, the location can be anywhere as long as the
user provides the correct path pointing to (otherwise, the script will have to re-
motely download each time). Here is the UNIX command for downloading all
RData files (preserving the directory structure): $wget - r - l2 - A" * .RData" -$
$np - nH - -cut - dirs = 0" http : //dnet.r - forge.r - project.org/RData"$

Value

an object of class "eTerm", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of
  gene set in consideration, and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term
  Name"), "namespace" and "distance"

- gs: a list of gene sets, each storing gene members. Always, gene sets are identified by "setID"
  and gene members identified by "Entrez ID"

- data: a matrix of nGene X nSample containing input data in consideration. It is not always
  the same as the input data as only those mappable are retained

- es: a matrix of nSet X nSample containing enrichment score, where nSample is the number
  of samples (i.e. the number of columns in input data

- nes: a matrix of nSet X nSample containing normalised enrichment score. It is the version of
  enrichment score but after being normalised by gene set size

- pvalue: a matrix of nSet X nSample containing nominal p value

- adjp: a matrix of nSet X nSample containing adjusted p value. It is the p value but after being
  adjusted for multiple comparisons

- gadjp: a matrix of nSet X nSample containing globally adjusted p value in terms of all sam-
  ples

- fdr: a matrix of nSet X nSample containing false discovery rate (FDR). It is the estimated
  probability that the normalised enrichment score represents a false positive finding

- qvalue: a matrix of nSet X nSample containing q value. It is the monotunically increasing
  FDR

- weight: the input type of score weight

- call: the call that produced this result

Note

The interpretation of returned components:

- "es": enrichment score for the gene set is the degree to which this gene set is overrepresented
  at the top or bottom of the ranked list of genes in each column of input data;

- "nes": normalised enrichment score for the gene set is enrichment score that has already normalised by gene set size. It is comparable across analysed gene sets;

- "pvalue": nominal p value is the statistical significance of the enrichment score. It is not adjusted for multiple hypothesis testing, and thus is of limited use in comparing gene sets;

- "adjp": adjusted p value by Benjamini & Hochberg method. It is comparable across gene sets;

- "gadjp": globally adjusted p value by Benjamini & Hochberg method. Unlike "adjp", it is adjusted in terms of all samples;

- "fdr": false discovery rate is the estimated probability that the normalised enrichment score represents a false positive finding. Unlike "adjp" or "gadjp" (also aliased as "fdr") that is derived from a list of p values, this version of fdr is directly calculate from the statistic (i.e. normalised enrichment score);

- "qvalue": q value is the monotunically increasing FDR so that the higher "nes", the lower "qvalue".

See Also

dGSEAview, dGSEAwrite, visGSEA

Examples

```
# load data
library(Biobase)
TCGA_mutations <-dRDataLoader(RData='TCGA_mutations')

# gene set enrichment analysis (GSEA) using KEGG pathways
## calculate the total mutations for each gene
tol <- apply(exprs(TCGA_mutations), 1, sum)
data <- data.frame(tol=tol)
eTerm <- dGSEA(data, identity="symbol", genome="Hs",
ontology="MsigdbC2KEGG")
res <- dGSEAview(eTerm, which_sample=1, top_num=5, sortBy="adjp",
decreasing=FALSE, details=TRUE)
visGSEA(eTerm, which_sample=1, which_term=rownames(res)[1])
output <- dGSEAwrite(eTerm, which_content="gadjp", which_score="gadjp",
filename="eTerm.txt")

## based on customised gene sets
eTerm <- dGSEA(data, identity="symbol", genome="Hs",
ontology="Customised", customised.genesets=sample(rownames(data),100))
res <- dGSEAview(eTerm, which_sample=1, top_num=5, sortBy="adjp",
decreasing=FALSE, details=TRUE)
visGSEA(eTerm, which_sample=1, which_term=rownames(res)[1])
```

---

dGSEAview   *Function to view enrichment results in a sample-specific manner*

Description

dGSEAview is supposed to view results of gene set enrichment analysis but for a specific sample.

Usage

```
dGSEAview(eTerm, which_sample = 1, top_num = 10, sortBy = c("adjp",
"gadjp", "ES", "nES", "pvalue", "FWER", "FDR", "qvalue", "none"),
decreasing = NULL, details = F)
```

Arguments

eTerm	an object of class "eTerm"
which_sample	which sample will be viewed
top_num	the maximum number of gene sets will be viewed
sortBy	which statistics will be used for sorting and viewing gene sets. It can be "adjp" for adjusted p value, "gadjp" for globally adjusted p value, "ES" for enrichment score, "nES" for normalised enrichment score, "pvalue" for p value, "FWER" for family-wise error rate, "FDR" for false discovery rate, "qvalue" for q value, "none" for sorting by setID
decreasing	logical to indicate whether to sort in a decreasing order. If it is null, it would be true for "ES" or "nES"; otherwise it would be false
details	logical to indicate whether the detail information of gene sets is also viewed. By default, it sets to false for no inclusion

Value

a data frame with following components:

- setID: term ID
- ES: enrichment score
- nES: normalised enrichment score
- pvalue: nominal p value
- adjp: adjusted p value
- gadjp: globally adjusted p value
- FDR: false discovery rate
- qvalue: q value
- setSize: the number of genes in the set; optional, it is only appended when "details" is true
- name: term name; optional, it is only appended when "details" is true
- namespace: term namespace; optional, it is only appended when "details" is true
- distance: term distance; optional, it is only appended when "details" is true

Note

See Also

[dGSEA](#)

Examples

```
#dGSEAview(eTerm, which_sample=1, top_num=10, sortBy="adjp", decreasing=FALSE, details=TRUE)
```

dGSEAwrite   *Function to write out enrichment results*

Description

dGSEAwrite is supposed to write out enrichment results.

Usage

```
dGSEAwrite(eTerm, which_content = c("gadjp", "adjp", "pvalue", "FWER",
"FDR",
"qvalue", "nES", "ES"), which_score = c("gadjp", "adjp", "FWER", "FDR",
"qvalue", "nES"), cutoff = 0.1, filename = NULL, keep.significance = T)
```

Arguments

eTerm   an object of class "eTerm"

which_content   the content will be written out. It includes two categories: i) based on "adjp"
                 for adjusted p value, "gadjp" for globally adjusted p value, "pvalue" for p value,
                 "FWER" for family-wise error rate, "FDR" for false discovery rate, "qvalue" for
                 q value; ii) based on "ES" for enrichment score, "nES" for normalised enrich-
                 ment score. For the former, the content is : first -1*log10-transformed, and then
                 multiplied by -1 if nES is negative.

which_score   which statistics/score will be used for declaring the significance. It can be "adjp"
                 for adjusted p value, "gadjp" for globally adjusted p value, "FWER" for family-
                 wise error rate, "FDR" for false discovery rate, "qvalue" for q value

cutoff   a cutoff to declare the signficance. It should be used together with 'which_score'

filename   a character string naming a filename

keep.significance
                 logical to indicate whether or not to mask those insignfiicant by NA. By default,
                 it sets to true to mask those insignfiicant by NA

Value

a data frame with following components:

- setID: term ID
- setSize: the number of genes in the set
- name: term name
- namespace: term namespace
- distance: term distance
- sample names: sample names in the next columns

Note

If "filename" is not NULL, a tab-delimited text file will be also written out.

See Also

[dGSEA](#)

Examples

```
#output <- dGSEAwrite(eTerm, which_content="gadjp", which_score="gadjp", filename="eTerm.txt")
```

| dNetConfidence | *Function to append the confidence information from the source graphs into the target graph* |

Description

eConsensusGraph is supposed to append the confidence information (extracted from a list of the source graphs) into the target graph. The confidence information is about how often a node (or an edge) in the target graph that can be found in the input source graphs. The target graph is an object of class "igraph" or "graphNEL", and the source graphs are a list of objects of class "igraph" or "graphNEL". It also returns an object of class "igraph" or "graphNEL"; specifically, the same as the input target graph but appended with the "nodeConfidence" attribute to the nodes and the "edgeConfidence" attribute to the edges.

Usage

```
dNetConfidence(target, sources, plot = F)
```

Arguments

target   the target graph, an object of class "igraph" or "graphNEL"

sources   a list of the source graphs, each with an object of class "igraph" or "graphNEL". These source graphs will be used to calculate how often a node (or an edge) in the target graph that can be found with them.

plot   logical to indicate whether the returned graph (i.e. the target graph plus the confidence information on nodes and edges) should be plotted. If it sets true, the plot will display the returned graph with the size of nodes indicative of the node confidence (the frequency that a node appears in the source graphs), and with the width of edges indicative of the edge confidence (the frequency that an edge appears in the source graphs)

Value

an object of class "igraph" or "graphNEL", which is a target graph but appended with the "nodeConfidence" attribute to the nodes and the "edgeConfidence" attribute to the edges

Note

None

See Also

[visNet](#)

Examples

```
# 1) generate a target graph according to the ER model
g <- erdos.renyi.game(100, 1/100)
target <- dNetInduce(g, V(g), knn=0)

# 2) generate a list source graphs according to the ER model
sources <- lapply(1:100, function(x) erdos.renyi.game(100*runif(1),
1/10))

# 3) append the confidence information from the source graphs into the target graph
g <- dNetConfidence(target=target, sources=sources)

# 4) visualise the confidence target graph
visNet(g, vertex.size=V(g)$nodeConfidence/10,
edge.width=E(g)$edgeConfidence)
```

---

dNetFind   *Function to find heuristically maximum scoring subgraph*

Description

dNetFind is supposed to find the maximum scoring subgraph from an input graph and scores imposed on its nodes. The input graph and the output subgraph are both of "igraph" or "graphNET" object. The input scores imposed on the nodes in the input graph can be divided into two parts: the positve nodes and the negative nodes. The searching for maximum scoring subgraph is deduced to find the connected subgraph containing the positive nodes as many as possible, but the negative nodes as few as possible. To this end, a heuristic search is used (see Note below).

Usage

```
dNetFind(g, scores)
```

Arguments

g   an object of class "igraph" or "graphNEL"

scores   a vector of scores. For each element, it must have the name that could be mapped
                  onto the input graph. Also, the names in input "scores" should contain all those
                  in the input graph "g", but the reverse is not necessary

Value

a subgraph with a maximum score, an object of class "igraph" or "graphNEL"

Note

The search procedure is heuristic to find the subgraph with the maximum score:

- i) transform the input graph into a new graph by collapsing connected positive nodes into a
  meta-node. As such, meta-nodes are isolated to each other but are linked via negative nodes
  (single-nodes). Clearly, meta-nodes have positive scores, and negative scores for the single-
  nodes.

- ii) append the weight attribute to the edges in the transformed graph. There are two types of edges: 1) the single-single edge with two single-nodes as two ends, and 2) single-meta edge with a single-node as one end and a meta-node as the other end. The weight for a single-single edge is the absolute sum of the scores in its two-end single-nodes but normalised by their degrees. The weight for a single-meta edge is simply the absolute score in its single-node end normalised by the degree. As such, weights are all non-negative.

- iii) find minimum spanning tree (MST) in the weighted transformed graph using Prim's greedy algorithm. A spanning tree of the weighted graph is a subgraph that is tree and connects all the node together. The MST is a spanning tree with the sum of its edge weights minimised amongst all possible spanning trees.

- iv) find all shortest paths between any pair of meta-nodes in the MST. Within the weighted transformed graph in ii), a subgraph is induced containing nodes (only occuring in these shortest paths) and all edges between them.

- v) within the induced subgraph, identify single-nodes that are direct neighbors of meta-nodes. For each of these single-nodes, also make sure it has the absolute scores no more than the sum of scores in its neighboring meta-nodes. These single-nodes meeting both criteria are called "linkers".

- vi) still within the induced subgraph in v), find the linker graph that contains only linkers and edges between them. Similarly to iii), find MST of the linker graph, called 'linker MST'. Notably, this linker MST serves as the scaffold, which only contains linkers but has meta-nodes being direcly attached to.

- vii) in linker MST plus its attached meta-nodes, find the optimal path that has the sum of scores of its nodes and attached meta-nodes maximised amongst all possible paths. Nodes along this optimal path plus their attached meta-nodes are called 'subgraph nodes'.

- viii) finally, from the input graph extract a subgraph (called 'subgraph') that only contains subgraph nodes and edges betwen them. This subgraph is the maximum scoring subgraph containing the positive nodes as many as possible, but the negative nodes as few as possible.

See Also

dNetFind

Examples

```
# 1) generate an vector consisting of random values from beta distribution
x <- rbeta(1000, shape1=0.5, shape2=1)

# 2) fit a p-value distribution under beta-uniform mixture model
fit <- dBUMfit(x, ntry=1, hist.bum=FALSE, contour.bum=FALSE)

# 3) calculate the scores according to the fitted BUM and fdr=0.01
# using "pdf" method
scores <- dBUMscore(fit, method="pdf", fdr=0.05, scatter.bum=FALSE)
names(scores) <- as.character(1:length(scores))

# 4) generate a random graph according to the ER model
g <- erdos.renyi.game(1000, 1/100)

# 5) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 6) find the subgraph with the maximum score
subgraph <- dNetFind(subg, scores)
```

dNetInduce | *Function to generate a subgraph induced by given vertices and their k nearest neighbors*

Description

dNetInduce is supposed to produce a subgraph induced by given vertices and its k nearest neighbors. The input is a graph of "igraph" or "graphNET" object, a list of the vertices of the graph, and a k value for finding k nearest neighbors for these vertices. The output is a subgraph induced by given vertices plus their k neighbours. The resultant subgraph inherits the class from the input one. The induced subgraph contains exactly the vertices of interest, and all the edges between them.

Usage

```
dNetInduce(g, nodes_query, knn = 0, remove.loops = F, largest.comp = T,
min.comp.size = 1)
```

Arguments

g
: an object of class "igraph" or "graphNEL"

nodes_query
: the vertices for which the calculation is performed

knn
: an integeter specifying how many k steps are used to find the nearest neighbours of the given vertices. By default, knn is set to zero; it means no neighbors will be considered. When knn is 1, the immediate neighbors of the given vertices will be also considered for inducing the subgraph. The same is true when knn is 2, etc

remove.loops
: logical to indicate whether the loop edges are to be removed. By default, it sets to false

largest.comp
: logical to indicate whether the largest component is only retained. By default, it sets to true for the largest component being left

min.comp.size
: an integer specifying the minimum size of component that will be retained. This parameter only works when setting the false to keep the largest component. By default, it sets to 1 meaning all nodes will be retained

Value

- subg: an induced subgraph, an object of class "igraph" or "graphNEL". Appended with a node attribute 'comp' if multiple components are kept

Note

The given vertices plus their k nearest neighbors will be used to induce the subgraph.

See Also

dNetInduce

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) select the first 10 vertices as the query nodes
nodes_query <- V(g)[1:10]

# 3) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, nodes_query, knn=0)

# 4) produce the induced subgraph based on the nodes in query ane their immediate neighbours
subg <- dNetInduce(g, nodes_query, knn=1)
```

---

dNetPipeline   *Function to setup the pipeline for finding maximum-scoring subgraph*
                            from an input graph and the signficance imposed on its nodes

---

Description

dNetPipeline is supposed to finish ab inito maximum-scoring subgraph identification for the input
graph with the node information on the significance (p-value or fdr). It returns an object of class
"igraph" or "graphNEL".

Usage

```
dNetPipeline(g, pval, method = c("pdf", "cdf", "customised"),
significance.threshold = NULL, nsize = NULL, plot = F, verbose = T)
```

Arguments

g   an object of class "igraph" or "graphNEL"

pval   a vector containing input p-values (or fdr). For each element, it must have the
                            name that could be mapped onto the input graph. Also, the names in input "pval"
                            should contain all those in the input graph "g", but the reverse is not necessary

method   the method used for the transformation. It can be either "pdf" for the method
                            based on the probability density function of the fitted model, or "cdf" for the
                            method based on the cumulative distribution function of the fitted model

significance.threshold
                            the given significance threshold. By default, it is set to NULL, meaning there
                            is no constraint. If given, those p-values below this are considered significant
                            and thus scored positively. Instead, those p-values above this given significance
                            threshold are considered insigificant and thus scored negatively

nsize   the desired number of nodes constrained to the resulting subgraph. It is not
                            nulll, a wide range of significance thresholds will be scanned to find the optimal
                            significance threshold leading to the desired number of nodes in the resulting
                            subgraph. Notably, the given significance threshold will be overwritten by this
                            option.

plot   logical to indicate whether the histogram plot, contour plot and scatter plot
                            should be drawn. By default, it sets to false for no plotting

verbose   logical to indicate whether the messages will be displayed in the screen. By
                            default, it sets to true for display

Value

a subgraph with a maximum score, an object of class "igraph" or "graphNEL"

Note

The pipeline sequentially consists of:

- ia) if the method is either "pdf" or "cdf", [dBUMfit](#) used to fit the p-value distribution under beta-uniform mixture model, and [dBUMscore](#) used to calculate the scores according to the fitted BUM and the significance threshold.

- ib) if the method is either "customised", then the user input list of fdr (or p-values) and the significance threshold will be directly used for score transformation by [dFDRscore](#).

- ii) if there is the desired number of nodes constrained to the resulting subgraph, a wide range of significance thresholds (including rough stage with large intervals, and finetune stage with smaller intervals) will be scanned to find the significance threshold to meet the desired number of nodes.

- iii) [dNetFind](#) used to find maximum-scoring subgraph from the input graph and scores imposed on its nodes.

See Also

[dBUMfit](#), [dBUMscore](#), [dFDRscore](#), [dNetFind](#)

Examples

```
# 1) generate an vector consisting of random values from beta distribution
x <- rbeta(1000, shape1=0.5, shape2=1)
names(x) <- as.character(1:length(x))

# 2) generate a random graph according to the ER model
g <- erdos.renyi.game(1000, 1/100)

# 3) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 4) find maximum-scoring subgraph based on the given significance threshold
# 4a) assume the input is a list of p-values (controlling fdr=0.1)
subgraph <- dNetPipeline(g=subg, pval=x, significance.threshold=0.1)
# 4b) assume the input is a list of customised significance (eg FDR directly)
subgraph <- dNetPipeline(g=subg, pval=x, method="customised",
significance.threshold=0.1)

# 5) find maximum-scoring subgraph with the desired node number nsize=20
subgraph <- dNetPipeline(g=subg, pval=x, nsize=20)
```

---

dNetReorder   *Function to reorder the multiple graph colorings within a sheet-shape rectangle grid*

Description

dNetReorder is reorder the multiple graph colorings within a sheet-shape rectangle grid

Usage

```
dNetReorder(g, data, feature = c("node", "edge"), node.normalise =
c("none",
"degree"), xdim = NULL, ydim = NULL, amplifier = NULL,
metric = c("none", "pearson", "spearman", "kendall", "euclidean",
"manhattan", "cos", "mi"), init = c("linear", "uniform", "sample"),
algorithm = c("sequential", "batch"), alphaType = c("invert", "linear",
"power"), neighKernel = c("gaussian", "bubble", "cutgaussian", "ep",
"gamma"))
```

Arguments

g   an object of class "igraph" or "graphNEL"

data   an input data matrix used to color-code vertices/nodes. One column corresponds to one graph node coloring. The input matrix must have row names, and these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)

feature   the type of the features used. It can be one of either 'edge' for the edge feature or 'node' for the node feature. See 'Note' for explanations.

node.normalise the normalisation of the nodes. It can be one of either 'none' for no normalisation or 'degree' for a node being penalised by its degree.

xdim   an integer specifying x-dimension of the grid

ydim   an integer specifying y-dimension of the grid

amplifier   an integer specifying the amplifier (3 by default) of the number of component planes. The product of the component number and the amplifier constitutes the number of rectangles in the sheet grid

metric   distance metric used to define the similarity between component planes. It can be "none", which means directly using column-wise vectors of codebook/data matrix. Otherwise, first calculate the covariance matrix from the codebook/data matrix. The distance metric used for calculating the covariance matrix between component planes can be: "pearson" for pearson correlation, "spearman" for spearman rank correlation, "kendall" for kendall tau rank correlation, "euclidean" for euclidean distance, "manhattan" for cityblock distance, "cos" for cosine similarity, "mi" for mutual information.

init   an initialisation method. It can be one of "uniform", "sample" and "linear" initialisation methods

algorithm   the training algorithm. Currently, only "sequential" algorithm has been implemented

alphaType   the alpha type. It can be one of "invert", "linear" and "power" alpha types

neighKernel   the training neighbor kernel. It can be one of "gaussian", "bubble", "cutgaussian", "ep" and "gamma" kernels

Value

an object of class "sReorder", a list with following components:

- nHex: the total number of rectangles in the grid

- xdim: x-dimension of the grid
- ydim: y-dimension of the grid
- uOrder: the unique order/placement for each component plane that is reordered to the "sheet"-shape grid with rectangular lattice
- coord: a matrix of nHex x 2, with each row corresponding to the coordinates of each "uOrder" rectangle in the 2D map grid
- call: the call that produced this result

Note

According to which features are used and whether nodes should be penalised by degrees, the feature data are constructed differently from the input data and input graph:

- When the node features are used, the feature data is the input data (or penalised data) with the same dimension.
- When the edge featrues are used, each entry (i.e. given an edge and a sample) in the feature data is the absolute difference between its two-end nodes (or after being penalised).
- After that, the constructed feature are subject to sample correlation analysis by supraHex. That is, a map grid (with sheet shape consisting of a rectangular lattice) is used to train either column-wise vectors of the feature data matrix or the covariance matrix thereof.
- As a result, similar samples are placed closer to each other within this map grid. More precisely, to ensure the unique placement, each sample mapped to the "sheet"-shape grid with rectangular lattice is determinied iteratively in an order from the best matched to the next compromised one. If multiple samples are hit in the same rectangular lattice, the worse one is always sacrificed by moving to the next best one till all samples are placed somewhere exclusively on their own.

The size of "sheet"-shape rectangle grid depends on the input arguments:

- How the input parameters are used to determine nHex is taken priority in the following order: "xdim & ydim" > "nHex" > "data".
- If both of xdim and ydim are given, $nHex = xdim * ydim$.
- If only data is input, $nHex = 5 * sqrt(dlen)$, where dlen is the number of rows of the input data.
- After nHex is determined, xy-dimensions of rectangle grid are then determined according to the square root of the two biggest eigenvalues of the input data.

See Also

[visNetReorder](#)

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) reorder the module with vertices being color-coded by input data
nnodes <- vcount(subg)
nsamples <- 10
```

```
data <- matrix(runif(nnodes*nsamples), nrow=nnodes, ncol=nsamples)
rownames(data) <- V(subg)$name
sReorder <- dNetReorder(g=subg, data, feature="node",
node.normalise="none")
```

---

dPvalAggregate *Function to aggregate p values*

Description

dPvalAggregate is supposed to aggregate a input matrix p-values into a vector of aggregated p-values. The aggregate operation is applied to each row of input matrix, each resulting in an aggregated p-value. The method implemented can be based on the order statistics of p-values or according to Fisher's method or Z-transform method.

Usage

```
dPvalAggregate(pmatrix, method = c("orderStatistic", "fishers",
"Ztransform",
"logistic"), order = ncol(pmatrix), weight = rep(1, ncol(pmatrix)))
```

Arguments

pmatrix	a data frame or matrix of p-values
method	the method used. It can be either "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method (summation of logs), or "Ztransform" for Z-transform test (summation of z values, Stouffer's method) and the weighted Z-test, or "logistic" for summation of logits
order	an integeter specifying the order used for the aggregation according to the order statistics of p-values
weight	a vector specifying the weights used for the aggregation according to Z-transform method

Value

- ap: a vector with the length nrow(pmatrix), containing aggregated p-values

Note

For each row of input matrix with the c columns, there are c p-values that are uniformly independently distributed over [0,1] under the null hypothesis (uniform distribution). According to the order statisitcs, they follow the Beta distribution with the paramters $a = order$ and $b = c - order + 1$. According to the Fisher's method, after transformation by $-2 * \sum^c log(pvalue)$, they follow Chi-Squared distribution. According to the Z-transform method, first converts the one-tailed P-values into standard normal deviates Z, then combines Z via $\frac{\sum^c (w*Z)}{\sqrt{\sum^c (w^2)}}$, where w is the weight (usually square root of the sample size if the weighted Z-test; 1 if Z-transform test), and finally the combined Z follows the standard normal distribution to test the cumulative/aggregated evidence on the common null hypothesis. The logistic method is defined as $\sum^c log(\frac{pvalue}{1-pvalue}) * 1/C$, where $C = sqrt((kpi^2(5k + 2))/(3(5k + 4)))$, following Student's t distribution. Generally speaking, Fisher's method places greater emphasis on small p-values, while the Z-transform method on equal

footings, the logistic method provides a compromise between these two. In other words, the Z-transform method does well in problems where evidence against the combined null is spread more than a small fraction of the individual tests, or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong.

See Also

dPvalAggregate

Examples

```
# 1) generate an iid uniformly-distributed random matrix of 1000x3
pmatrix <- cbind(runif(1000), runif(1000), runif(1000))

# 2) aggregate according to the order statistics
ap <- dPvalAggregate(pmatrix, method="orderStatistic")

# 3) aggregate according to the Fisher's method
ap <- dPvalAggregate(pmatrix, method="fishers")

# 4) aggregate according to the Z-transform method
ap <- dPvalAggregate(pmatrix, method="Ztransform")

# 5) aggregate according to the logistic method
ap <- dPvalAggregate(pmatrix, method="logistic")
```

---

dRDataLoader   *Function to load dnet built-in RData*

Description

dRDataLoader is supposed to load the package built-in RData.

Usage

```
dRDataLoader(RData = c(NA, "TCGA_mutations", "ig.DO", "ig.GOBP",
"ig.GOCC",
"ig.GOMF", "ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA", "ig.MP",
"org.At.eg",
"org.At.egGOBP", "org.At.egGOCC", "org.At.egGOMF", "org.At.egPS",
"org.At.egSF", "org.At.string", "org.Ce.eg", "org.Ce.egGOBP",
"org.Ce.egGOCC",
"org.Ce.egGOMF", "org.Ce.egPS", "org.Ce.egSF", "org.Ce.string",
"org.Da.eg",
"org.Da.egGOBP", "org.Da.egGOCC", "org.Da.egGOMF", "org.Da.egPS",
"org.Da.egSF", "org.Da.string", "org.Dm.eg", "org.Dm.egGOBP",
"org.Dm.egGOCC",
"org.Dm.egGOMF", "org.Dm.egPS", "org.Dm.egSF", "org.Dm.string",
"org.Gg.eg", "org.Gg.egGOBP", "org.Gg.egGOCC", "org.Gg.egGOMF",
"org.Gg.egPS",
"org.Gg.egSF", "org.Gg.string", "org.Hs.eg", "org.Hs.egDGIdb",
"org.Hs.egDO",
```

```
"org.Hs.egGOBP", "org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM",
"org.Hs.egHPMA", "org.Hs.egHPMI", "org.Hs.egHPPA", "org.Hs.egMP",
"org.Hs.egMsigdbC1", "org.Hs.egMsigdbC2BIOCARTA",
"org.Hs.egMsigdbC2CGP",
"org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG",
"org.Hs.egMsigdbC2REACTOME",
"org.Hs.egMsigdbC3MIR", "org.Hs.egMsigdbC3TFT", "org.Hs.egMsigdbC4CGN",
"org.Hs.egMsigdbC4CM", "org.Hs.egMsigdbC5BP", "org.Hs.egMsigdbC5CC",
"org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7",
"org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF", "org.Hs.string",
"org.Mm.eg", "org.Mm.egDO", "org.Mm.egGOBP", "org.Mm.egGOCC",
"org.Mm.egGOMF",
"org.Mm.egHPCM", "org.Mm.egHPMA", "org.Mm.egHPMI", "org.Mm.egHPPA",
"org.Mm.egMP", "org.Mm.egPS", "org.Mm.egSF", "org.Mm.string",
"org.Rn.eg",
"org.Rn.egGOBP", "org.Rn.egGOCC", "org.Rn.egGOMF", "org.Rn.egPS",
"org.Rn.egSF", "CLL", "org.Rn.string"), genome = c(NA, "Hs", "Mm",
"Rn",
"Gg", "Ce", "Dm", "Da", "At"), ontology = c(NA, "GOBP", "GOMF", "GOCC",
"PS", "PS2", "SF", "DO", "HPPA", "HPMI", "HPCM", "HPMA", "MP",
"MsigdbH",
"MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG",
"MsigdbC2REACTOME",
"MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN",
"MsigdbC4CM",
"MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7",
"DGIdb"),
verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/dnet/1.0.7")
```

Arguments

RData   which built-in RData to load. It can be one of "TCGA_mutations", "ig.DO",
"ig.GOBP", "ig.GOCC", "ig.GOMF", "ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA",
"ig.MP", "org.At.eg", "org.At.egGOBP", "org.At.egGOCC", "org.At.egGOMF",
"org.At.egPS", "org.At.egSF", "org.At.string", "org.Ce.eg", "org.Ce.egGOBP",
"org.Ce.egGOCC", "org.Ce.egGOMF", "org.Ce.egPS", "org.Ce.egSF", "org.Ce.string",
"org.Da.eg", "org.Da.egGOBP", "org.Da.egGOCC", "org.Da.egGOMF", "org.Da.egPS",
"org.Da.egSF", "org.Da.string", "org.Dm.eg", "org.Dm.egGOBP", "org.Dm.egGOCC",
"org.Dm.egGOMF", "org.Dm.egPS", "org.Dm.egSF", "org.Dm.string", "org.Gg.eg",
"org.Gg.egGOBP", "org.Gg.egGOCC", "org.Gg.egGOMF", "org.Gg.egPS", "org.Gg.egSF",
"org.Gg.string", "org.Hs.eg", "org.Hs.egDGIdb", "org.Hs.egDO", "org.Hs.egGOBP",
"org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA",
"org.Hs.egHPMI", "org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1", "org.Hs.egMsigdbC2B
"org.Hs.egMsigdbC2CGP", "org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG",
"org.Hs.egMsigdbC2REACTOME", "org.Hs.egMsigdbC3MIR", "org.Hs.egMsigdbC3TFT",
"org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM", "org.Hs.egMsigdbC5BP",
"org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7",
"org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF", "org.Hs.string", "org.Mm.eg",
"org.Mm.egDO", "org.Mm.egGOBP", "org.Mm.egGOCC", "org.Mm.egGOMF",
"org.Mm.egHPCM", "org.Mm.egHPMA", "org.Mm.egHPMI", "org.Mm.egHPPA",
"org.Mm.egMP", "org.Mm.egPS", "org.Mm.egSF", "org.Mm.string", "org.Rn.eg",

"org.Rn.egGOBP", "org.Rn.egGOCC", "org.Rn.egGOMF", "org.Rn.egPS", "org.Rn.egSF", "CLL", "org.Rn.string". On the meanings, please refer to the Documentations at http://supfam.org/dnet/docs.html

genome   the genome identity. It can be one of "Hs" for human, "Mm" for mouse, "Rn" for rat, "Gg" for chicken, "Ce" for c.elegans, "Dm" for fruitfly, "Da" for zebrafish, and "At" for arabidopsis

ontology   the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PS" for phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for domain superfamily assignments, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, and Drug-Gene Interaction database (DGIdb) and the molecular signatures database (Msigdb) only in human (including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7"). Note: These four ("GOBP", "GOMF", "GOCC" and "PS") are availble for all genomes/species; for "Hs" and "Mm", these six ("DO", "HPPA", "HPMI", "HPCM", "HPMA" and "MP") are also supported; all "Msigdb" are only supported in "Hs". For details on the eligibility for pairs of input genome and ontology, please refer to the online Documentations at http://supfam.org/dnet/docs.html

verbose   logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

RData.location   the characters to tell the location of built-in RData files. By default, it remotely locates at https://github.com/hfang-bristol/RDataCentre/blob/master/dnet and http://dnet.r-forge.r-project.org/RData. Be aware of several versions and the latest one is matched to the current package version. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget-r-l2-A"*.RData"-np-nH--cut-dirs = 0"http://dnet.r-forge.r-project.org/RData"$

Value

any use-specified variable that is given on the right side of the assigement sign '<-', which contains the loaded RData.

Note

If there are no use-specified variable that is given on the right side of the assigement sign '<-', then no RData will be loaded onto the working environment.

See Also

[dRDataLoader](#)

Examples

```
org.Hs.egSF <- dRDataLoader(RData='org.Hs.egSF')
org.Hs.eg <- dRDataLoader(RData='org.Hs.eg')
org.Hs.egDGIdb <- dRDataLoader(RData='org.Hs.egDGIdb')
org.Hs.egMsigdbC2KEGG <- dRDataLoader(RData='org.Hs.egMsigdbC2KEGG')
org.Hs.egHPPA <- dRDataLoader(genome='Hs', ontology='HPPA')
ig.MP <- dRDataLoader(RData='ig.MP')
```

---

dRWR   *Function to implement Random Walk with Restart (RWR) on the input*
                              graph

Description

dRWR is supposed to implement Random Walk with Restart (RWR) on the input graph. If the seeds (i.e. a set of starting nodes) are given, it intends to calculate the affinity score of all nodes in the graph to the seeds. If the seeds are not given, it will pre-compute affinity matrix for nodes in the input graph with respect to each starting node (as a seed) by looping over every node in the graph. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
dRWR(g, normalise = c("laplacian", "row", "column", "none"),
setSeeds = NULL, restart = 0.75, normalise.affinity.matrix = c("none",
"quantile"), parallel = TRUE, multicores = NULL, verbose = T)
```

Arguments

g	an object of class "igraph" or "graphNEL"
normalise	the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'
setSeeds	an input matrix used to define sets of starting seeds. One column corresponds to one set of seeds that a walker starts with. The input matrix must have row names, coming from node names of input graph, i.e. V(g)$name, since there is a mapping operation. The non-zero entries mean that the corresonding rows (i.e. the gene/row names) are used as the seeds, and non-zero values can be viewed as how to weight the relative importance of seeds. By default, this option sets to "NULL", suggesting each node in the graph will be used as a set of the seed to pre-compute affinity matrix for the input graph. This default does not scale for large input graphs since it will loop over every node in the graph; however, the pre-computed affinity matrix can be extensively reused for obtaining affinity scores between any combinations of nodes/seeds, allows for some flexibility in the downstream use, in particular when sampling a large number of random node combinations for statistical testing

restart   the restart probability used for RWR. The restart probability takes the value from
                     0 to 1, controlling the range from the starting nodes/seeds that the walker will
                     explore. The higher the value, the more likely the walker is to visit the nodes
                     centered on the starting nodes. At the extreme when the restart probability is
                     zero, the walker moves freely to the neighbors at each step without restarting
                     from seeds, i.e., following a random walk (RW)

normalise.affinity.matrix
                     the way to normalise the output affinity matrix. It can be 'none' for no normali-
                     sation, 'quantile' for quantile normalisation to ensure that columns (if multiple)
                     of the output affinity matrix have the same quantiles

parallel   logical to indicate whether parallel computation with multicores is used. By de-
                     fault, it sets to true, but not necessarily does so. It will depend on whether these
                     two packages "foreach" and "doParallel" have been installed. It can be installed
                     via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParalle
                     If not yet installed, this option will be disabled

multicores   an integer to specify how many cores will be registered as the multicore parallel
                     backend to the 'foreach' package. If NULL, it will use a half of cores available in
                     a user's computer. This option only works when parallel computation is enabled

verbose   logical to indicate whether the messages will be displayed in the screen. By
                     default, it sets to true for display

Value

It returns a sparse matrix, called 'PTmatrix':

- When the seeds are NOT given: a pre-computed affinity matrix with the dimension of n
  X n, where n is the number of nodes in the input graph. Columns stand for starting nodes
  walking from, and rows for ending nodes walking to. Therefore, a column for a starting node
  represents a steady-state affinity vector that the starting node will visit all the ending nodes in
  the graph

- When the seeds are given: an affinity matrix with the dimension of n X nset, where n is
  the number of nodes in the input graph, and nset for the number of the sets of seeds (i.e.
  the number of columns in setSeeds). Each column stands for the steady probability vector,
  storing the affinity score of all nodes in the graph to the starting nodes/seeds. This steady
  probability vector can be viewed as the "influential impact" over the graph imposed by the
  starting nodes/seeds.

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated
with

See Also

dRWRcontact, dRWRpipeline, dCheckParallel

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)
```

```
V(subg)$name <- 1:vcount(subg)

# 3) obtain the pre-computated affinity matrix
PTmatrix <- dRWR(g=subg, normalise="laplacian", restart=0.75,
parallel=FALSE)
# visualise affinity matrix
visHeatmapAdv(PTmatrix, Rowv=FALSE, Colv=FALSE, colormap="wyr",
KeyValueName="Affinity")

# 4) obtain affinity matrix given sets of seeds
# define sets of seeds
# each seed with equal weight (i.e. all non-zero entries are '1')
aSeeds <- c(1,0,1,0,1)
bSeeds <- c(0,0,1,0,1)
setSeeds <- data.frame(aSeeds,bSeeds)
rownames(setSeeds) <- 1:5
# calcualte affinity matrix
PTmatrix <- dRWR(g=subg, normalise="laplacian", setSeeds=setSeeds,
restart=0.75, parallel=FALSE)
PTmatrix
```

dRWRcontact	*Function to estimate RWR-based contact strength between samples from an input gene-sample data matrix, an input graph and its pre-computed affinity matrix*

Description

dRWRcontact is supposed to estimate sample relationships (ie. contact strength between samples) from an input gene-sample matrix, an input graph and its affinity matrix pre-computed according to random walk restart (RWR) of the input graph. It includes: 1) RWR-smoothed columns of input gene-sample matrix based on the pre-computed affinity matrix; 2) calculation of contact strength (inner products of RWR-smooth columns of input gene-sample matrix); 3) estimation of the contact signficance by a randomalisation procedure. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
dRWRcontact(data, g, Amatrix, permutation = c("random", "degree"),
num.permutation = 10, p.adjust.method = c("BH", "BY", "bonferroni",
"holm", "hochberg", "hommel"), adjp.cutoff = 0.05, parallel = TRUE,
multicores = NULL, verbose = T)
```

Arguments

data	an input gene-sample data matrix used for seeds. Each value in input gene-sample matrix does not necessarily have to be binary (non-zeros will be used as a weight, but should be non-negative for easy interpretation).
g	an object of class "igraph" or "graphNEL"
Amatrix	an affinity matrix pre-computed from the input graph. Notes: columns for starting nodes walking from, and rows for ending nodes walking to

permutation	how to do permutation. It can be 'degree' for degree-preserving permutation, 'random' for permutation purely in random
num.permutation	
	the number of permutations used to for generating the distribution of contact strength under randomalisation
p.adjust.method	
	the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER
adjp.cutoff	the cutoff of adjusted pvalue to construct the contact graph
parallel	logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel")). If not yet installed, this option will be disabled
multicores	an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "dContact", a list with following components:

- ratio: a symmetric matrix storing ratio (the observed against the expected) between pairwise samples
- zscore: a symmetric matrix storing zscore between pairwise samples
- pval: a symmetric matrix storing pvalue between pairwise samples
- adjpval: a symmetric matrix storing adjusted pvalue between pairwise samples
- cgraph: the constructed contact graph (as a 'igraph' object) under the cutoff of adjusted value
- call: the call that produced this result

Note

See Also

[dRWR](), [dCheckParallel]()

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
```

```
subg <- dNetInduce(g, V(g), knn=0)
V(subg)$name <- 1:vcount(subg)

# 3) pre-compute affinity matrix from the input graph
Amatrix <- dRWR(g=subg, parallel=FALSE)

# 4) estimate RWR-based sample relationships
# define sets of seeds as data
# each seed with equal weight (i.e. all non-zero entries are '1')
aSeeds <- c(1,0,1,0,1)
bSeeds <- c(0,0,1,0,1)
data <- data.frame(aSeeds,bSeeds)
rownames(data) <- 1:5
# calcualte their two contacts
dContact <- dRWRcontact(data=data, g=subg, Amatrix=Amatrix,
parallel=FALSE)
dContact
```

dRWRpipeline	*Function to setup a pipeine to estimate RWR-based contact strength between samples from an input gene-sample data matrix and an input graph*

Description

dRWRpipeline is supposed to estimate sample relationships (ie. contact strength between samples) from an input gene-sample matrix and an input graph. The pipeline includes: 1) random walk restart (RWR) of the input graph using the input matrix as seeds; 2) calculation of contact strength (inner products of RWR-smoothed columns of input matrix); 3) estimation of the contact signficance by a randomalisation procedure. It supports two methods how to use RWR: 'direct' for directly applying RWR in the given seeds; 'indirectly' for first pre-computing affinity matrix of the input graph, and then deriving the affinity score. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
dRWRpipeline(data, g, method = c("direct", "indirect"),
normalise = c("laplacian", "row", "column", "none"), restart = 0.75,
normalise.affinity.matrix = c("none", "quantile"),
permutation = c("random", "degree"), num.permutation = 10,
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
adjp.cutoff = 0.05, parallel = TRUE, multicores = NULL, verbose = T)
```

Arguments

data	an input gene-sample data matrix used for seeds. Each value in input gene-sample matrix does not necessarily have to be binary (non-zeros will be used as a weight, but should be non-negative for easy interpretation).
g	an object of class "igraph" or "graphNEL"
method	the method used to calculate RWR. It can be 'direct' for directly applying RWR, 'indirect' for indirectly applying RWR (first pre-compute affinity matrix and then derive the affinity score)

normalise   the way to normalise the adjacency matrix of the input graph. It can be 'lapla-
                 cian' for laplacian normalisation, 'row' for row-wise normalisation, 'column'
                 for column-wise normalisation, or 'none'

restart   the restart probability used for RWR. The restart probability takes the value from
                 0 to 1, controlling the range from the starting nodes/seeds that the walker will
                 explore. The higher the value, the more likely the walker is to visit the nodes
                 centered on the starting nodes. At the extreme when the restart probability is
                 zero, the walker moves freely to the neighbors at each step without restarting
                 from seeds, i.e., following a random walk (RW)

normalise.affinity.matrix
                 the way to normalise the output affinity matrix. It can be 'none' for no normali-
                 sation, 'quantile' for quantile normalisation to ensure that columns (if multiple)
                 of the output affinity matrix have the same quantiles

permutation   how to do permutation. It can be 'degree' for degree-preserving permutation,
                 'random' for permutation in random

num.permutation
                 the number of permutations used to for generating the distribution of contact
                 strength under randomalisation

p.adjust.method
                 the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni",
                 "holm", "hochberg" and "hommel". The first two methods "BH" (widely used)
                 and "BY" control the false discovery rate (FDR: the expected proportion of false
                 discoveries amongst the rejected hypotheses); the last four methods "bonfer-
                 roni", "holm", "hochberg" and "hommel" are designed to give strong control of
                 the family-wise error rate (FWER). Notes: FDR is a less stringent condition
                 than FWER

adjp.cutoff   the cutoff of adjusted pvalue to construct the contact graph

parallel   logical to indicate whether parallel computation with multicores is used. By de-
                 fault, it sets to true, but not necessarily does so. It will depend on whether these
                 two packages "foreach" and "doParallel" have been installed. It can be installed
                 via: source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParalle
                 If not yet installed, this option will be disabled

multicores   an integer to specify how many cores will be registered as the multicore parallel
                 backend to the 'foreach' package. If NULL, it will use a half of cores available in
                 a user's computer. This option only works when parallel computation is enabled

verbose   logical to indicate whether the messages will be displayed in the screen. By
                 default, it sets to true for display

Value

an object of class "dContact", a list with following components:

- ratio: a symmetric matrix storing ratio (the observed against the expected) between pairwise
  samples
- zscore: a symmetric matrix storing zscore between pairwise samples
- pval: a symmetric matrix storing pvalue between pairwise samples
- adjpval: a symmetric matrix storing adjusted pvalue between pairwise samples
- cgraph: the constructed contact graph (as a 'igraph' object) under the cutoff of adjusted value
- Amatrix: a pre-computed affinity matrix when using 'inderect' method; NULL otherwise
- call: the call that produced this result

Note

The choice of which method to use RWR depends on the number of seed sets and the number of permutations for statistical test. If the total product of both numbers are huge, it is better to use 'indrect' method (for a single run). However, if the user wants to re-use pre-computed affinity matrix (ie. re-use the input graph a lot), then it is highly recommended to sequentially use dRWR and dRWRcontact instead.

See Also

dRWR, dRWRcontact, dCheckParallel

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)
V(subg)$name <- 1:vcount(subg)

# 3) estimate RWR dating based sample relationships
# define sets of seeds as data
# each seed with equal weight (i.e. all non-zero entries are '1')
aSeeds <- c(1,0,1,0,1)
bSeeds <- c(0,0,1,0,1)
data <- data.frame(aSeeds,bSeeds)
rownames(data) <- 1:5
# calcualte their two contact graph
dContact <- dRWRpipeline(data=data, g=subg, parallel=FALSE)
dContact
```

---

dSVDsignif   *Function to obtain SVD-based gene significance from the input gene-sample matrix*

Description

dSVDsignif is supposed to obtain gene signficance from the given gene-sample matrix according to singular value decomposition (SVD)-based method. The method includes: 1) singular value decomposition of the input matrix; 2) determination of the eigens in consideration (if not given); 3) construction of the gene-specific project vector based on the considered eigens; 4) calculation of the distance statistic from the projection vector to zero point vector; and 5) based on distance statistic to obtain the gene significance.

Usage

```
dSVDsignif(data, num.eigen = NULL, pval.eigen = 0.01, signif = c("fdr",
"pval"), orient.permutation = c("row", "column", "both"),
num.permutation = 100, fdr.procedure = c("stepup", "stepdown"),
verbose = T)
```

Arguments

`data`	an input gene-sample data matrix used for singular value decomposition
`num.eigen`	an integer specifying the number of eigens in consideration. If NULL, this number will be automatically decided on based on the observed relative eigenexpression against randomised relative eigenexpression calculated from a list (here 100) of permutated input matrix
`pval.eigen`	p-value used to call those eigens as dominant. This parameter is used only when parameter 'num.eigen' is NULL. Here, p-value is calcualted to assess how likely the observed relative eigenexpression are more than the maximum relative eigenexpression calculated from permutated matrix
`signif`	the singificance to return. It can be either "pval" for using the p-value as the gene significance, or "fdr" for using the fdr as the gene significance
`orient.permutation`	
	the orientation of matrix being permutated. It can be either "row" to permutate values within each row, or "column" to permutate values within each column, or "both" to permutate values both within rows and columns. Notably, when using the p-value as the gene significance, it is always to permutate values within each row.
`num.permutation`	
	an integer specifying how many permutations are used
`fdr.procedure`	the procedure to adjust the fdr. To ensure that the high distance statistic the more significance, the fdr should be adjusted either using "stepup" for step-up procedure (from the most significant to the least significant) or using "stepdown" for step-down procedure (from the least significant to the most significant)
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a vector storing gene significance

Note

See Also

[dFDRscore](#)

Examples

```
# 1) generate data with an iid matrix of 1000 x 9
data <- cbind(matrix(rnorm(1000*3,mean=0,sd=1), nrow=1000, ncol=3),
matrix(rnorm(1000*3,mean=0.5,sd=1), nrow=1000, ncol=3),
matrix(rnorm(1000*3,mean=-0.5,sd=1), nrow=1000, ncol=3))

# 2) calculate the significance according to SVD
# using "fdr" significance
fdr <- dSVDsignif(data, signif="fdr", num.permutation=10)
# using "pval" significance
pval <- dSVDsignif(data, signif="pval", num.permutation=10)
```

Hiratani_TableS1 *Mouse multilayer omics dataset from Hiratani et al. (2010)*

Description

This multilayer omics dataset involves the information on DNA replication timing, promoter CpG classification and gene expression. It consists of digitised replication timing, promoter CpG status and expression levels of 17,292 genes in a variety of samples.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Datasets/Hiratani_TableS1.RData"))
```

Value

- RT: a replication timing matrix of 17,292 genes X 22 samples. These 22 samples come from 22 cell lines during early mouse embryogenesis, and they can be categorised into: 1) pluripotent cells, including ESCs (ESC_46C, ESC_D3 and ESC_TT2) and iPSCs (iPSC, iPSC_1D4 and iPSC_2D4); 2) partially-reprogrammed iPSCs (piPSC_1A2, piPSC_1B3 and piPSC_V3); 3) early epiblast (EPL and EMB3_D3); 4) late epiblast (EpiSC5 and EpiSC7); 5) Ectoderm (EBM6_D3, EBM9_D3, NPC_46C and NPC_TT2); 6) Mesoderm and Endoderm; and 7) late Mesoderm (Myoblast, MEF_female and MEF_male).
- CpG: a matrix of 17,292 genes X 1 containing gene additional information on promoter CpG classification, with '1' for HCP (high CpG density promoters), '-1' for LCP (low CpG density promoters), '0' for ICP (intermediate CpG density promoters), and 'NA' for unclassified.
- EX: an expression matrix of 17,292 genes X 8 samples, and samples include pluripotent cells (ESC_D3); early epiblast (EMB3_D3); late epiblast (EpiSC7); Ectoderm (EBM6_D3 and EBM9_D3); Mesoderm and Endoderm.

References

Mikkelsen et al. (2007). Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, 448:553-560.
Hiratani et al. (2010). Genome-wide dynamics of replication timing revealed by in vitro models of mouse embryogenesis. *Genome Research*, 20:155-169.

Examples

```
load(url("http://dnet.r-forge.r-project.org/RData/1.0.7/Hiratani_TableS1.RData"))
ls() # you should see three variables: 'RT', 'CpG' and 'EX'
```

ig.DO *Disease Ontology (DO).*

Description

An R object that contains information on Disease Ontology terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://sourceforge.net/p/diseaseontology/code/HEAD/tree/trunk/HumanDO.obo?format=raw.

Usage

```
ig.DO <- dRDataLoader(RData='ig.DO')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- `vertex attributes`: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- `edge attributes`: "relation" (either 'is_a' or 'part_of')

References

Schriml et al. (2012) Disease Ontology: a backbone for disease semantic integration. *Nucleic Acids Res*, 40:D940-946.

Examples

```
ig.DO <- dRDataLoader(RData='ig.DO')
ig.DO
```

---

ig.GOBP   *Gene Ontology Biological Process (GOBP).*

Description

An R object that contains information on Gene Ontology Biological Process terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo.

Usage

```
ig.GOBP <- dRDataLoader(RData='ig.GOBP')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- `vertex attributes`: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- `edge attributes`: "relation" (either 'is_a' or 'part_of')

References

Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
ig.GOBP <- dRDataLoader(RData='ig.GOBP')
ig.GOBP
```

---

ig.GOCC   *Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains information on Gene Ontology Cellular Component terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see <http://igraph.org/r/doc/aaa-igraph-package.html>). This data is prepared based on <http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo>.

Usage

```
ig.GOCC <- dRDataLoader(RData='ig.GOCC')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- `vertex attributes`: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)

- `edge attributes`: "relation" (either 'is_a' or 'part_of')

References

Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
ig.GOCC <- dRDataLoader(RData='ig.GOCC')
ig.GOCC
```

---

ig.GOMF   *Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains information on Gene Ontology Molecular Function terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see <http://igraph.org/r/doc/aaa-igraph-package.html>). This data is prepared based on <http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo>.

Usage

```
ig.GOMF <- dRDataLoader(RData='ig.GOMF')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)

- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
ig.GOMF <- dRDataLoader(RData='ig.GOMF')
ig.GOMF
```

---

ig.HPCM   *Human Phenotype Clinical Modifier (HPCM).*

Description

An R object that contains information on Human Phenotype Clinical Modifier terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://purl.obolibrary.org/obo/hp.obo.

Usage

```
ig.HPCM <- dRDataLoader(RData='ig.HPCM')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)

- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
ig.HPCM <- dRDataLoader(RData='ig.HPCM')
ig.HPCM
```

---

ig.HPMA   *Human Phenotype Mortality Aging (HPMA).*

Description

An R object that contains information on Human Phenotype Mortality Aging terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://purl.obolibrary.org/obo/hp.obo.

Usage

```
data(ig.HPMA)
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
#ig.HPMA <- dRDataLoader(RData='ig.HPMA')
data(ig.HPMA)
ig.HPMA
```

---

ig.HPMI   *Human Phenotype Mode of Inheritance (HPMI).*

Description

An R object that contains information on Human Phenotype Mode of Inheritance terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://purl.obolibrary.org/obo/hp.obo.

Usage

```
ig.HPMI <- dRDataLoader(RData='ig.HPMI')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
ig.HPMI <- dRDataLoader(RData='ig.HPMI')
ig.HPMI
```

---

ig.HPPA *Human Phenotype Phenotypic Abnormality (HPPA).*

Description

An R object that contains information on Human Phenotype Phenotypic Abnormality terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on http://purl.obolibrary.org/obo/hp.obo.

Usage

```
data(ig.HPPA)
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
ig.HPPA <- dRDataLoader(RData='ig.HPPA')
ig.HPPA
```

---

ig.MP   *Mammalian Phenotype (MP).*

Description

An R object that contains information on Mammalian Phenotype terms. These terms are organised as a direct acyclic graph (DAG), which is further stored as an object of the class 'igraph' (see http://igraph.org/r/doc/aaa-igraph-package.html). This data is prepared based on ftp://ftp.informatics.jax.org/pub/reports/MPheno_OBO.ontology.

Usage

```
ig.MP <- dRDataLoader(RData='ig.MP')
```

Value

an object of class "igraph". As a direct graph, it has attributes to vertices/nodes and edges:

- vertex attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
- edge attributes: "relation" (either 'is_a' or 'part_of')

References

Smith et al. (2009) The Mammalian Phenotype Ontology: enabling robust annotation and comparative analysis. *Wiley Interdiscip Rev Syst Biol Med*, 1:390-399.

Examples

```
ig.MP <- dRDataLoader(RData='ig.MP')
ig.MP
```

---

org.At.eg   *Arabidopsis Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the arabidopsis. This data is prepared based on ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.At.eg <- dRDataLoader(RData='org.At.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.At.eg <- dRDataLoader(RData='org.At.eg')
names(org.At.eg)
org.At.eg$gene_info[1:5,]
```

---

org.At.egGOBP	*Annotations of Arabidopsis Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Arabidopsis Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.At.egGOBP <- dRDataLoader(RData='org.At.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.At.egGOBP <- dRDataLoader(RData='org.At.egGOBP')
names(org.At.egGOBP)
```

---

org.At.egGOCC	*Annotations of Arabidopsis Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Arabidopsis Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.At.egGOCC <- dRDataLoader(RData='org.At.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.At.egGOCC <- dRDataLoader(RData='org.At.egGOCC')
names(org.At.egGOCC)
```

---

org.At.egGOMF	*Annotations of Arabidopsis Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Arabidopsis Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.At.egGOMF <- dRDataLoader(RData='org.At.egGOMF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.At.egGOMF <- dRDataLoader(RData='org.At.egGOMF')
names(org.At.egGOMF)
```

---

org.At.egPS	*Annotations of Arabidopsis Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Arabidopsis Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Arabidopsis Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the arabidopsis, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.At.egPS <- dRDataLoader(RData='org.At.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or

left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.

- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.At.egPS <- dRDataLoader(RData='org.At.egPS')
names(org.At.egPS)
```

---

org.At.egSF	*Annotations of Arabidopsis Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Arabidopsis Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Arabidopsis Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.At.egSF <- dRDataLoader(RData='org.At.egSF')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.At.egSF <- dRDataLoader(RData='org.At.egSF')
names(org.At.egSF)
```

---

org.At.string	*Arabidopsis functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in arabidopsis. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.At.string <- dRDataLoader(RData='org.At.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- name: unique id for the vertices
- seqid: protein seqid for the vertices
- geneid: Entrez geneid (if any) for the vertices
- symbol: gene symbol (if any) for the vertices
- description: gene description (if any) for the vertices

Below are attributes for the edges:

- neighborhood_score: predictive score based on neighborhood data
- fusion_score: predictive score based on fusion data
- cooccurence_score: predictive score based on cooccurence data
- coexpression_score: predictive score based on coexpression
- experimental_score: predictive score based on experimental data
- database_score: predictive score based on database
- textmining_score: predictive score based on text mining
- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.At.string <- dRDataLoader(RData='org.At.string')
org.At.string
```

---

org.Ce.eg   *C.elegans Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the c.elegans. This data is prepared based on ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Ce.eg <- dRDataLoader(RData='org.Ce.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.Ce.eg <- dRDataLoader(RData='org.Ce.eg')
names(org.Ce.eg)
org.Ce.eg$gene_info[1:5,]
```

---

org.Ce.egGOBP   *Annotations of C.elegans Entrez Genes (EG) by Gene Ontology Bio-logical Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and C.elegans Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Ce.egGOBP <- dRDataLoader(RData='org.Ce.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Ce.egGOBP <- dRDataLoader(RData='org.Ce.egGOBP')
names(org.Ce.egGOBP)
```

---

org.Ce.egGOCC   *Annotations of C.elegans Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and C.elegans Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Ce.egGOCC <- dRDataLoader(RData='org.Ce.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Ce.egGOCC <- dRDataLoader(RData='org.Ce.egGOCC')
names(org.Ce.egGOCC)
```

---

org.Ce.egGOMF   *Annotations of C.elegans Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and C.elegans Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Ce.egGOMF <- dRDataLoader(RData='org.Ce.egGOMF')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Ce.egGOMF <- dRDataLoader(RData='org.Ce.egGOMF')
names(org.Ce.egGOMF)
```

org.Ce.egPS	*Annotations of C.elegans Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for C.elegans Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a C.elegans Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the c.elegans, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Ce.egPS <- dRDataLoader(RData='org.Ce.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Ce.egPS <- dRDataLoader(RData='org.Ce.egPS')
names(org.Ce.egPS)
```

org.Ce.egSF	*Annotations of C.elegans Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for C.elegans Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for C.elegans Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Ce.egSF <- dRDataLoader(RData='org.Ce.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Ce.egSF <- dRDataLoader(RData='org.Ce.egSF')
names(org.Ce.egSF)
```

org.Ce.string	*C.elegans functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in c.elegans. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Ce.string <- dRDataLoader(RData='org.Ce.string')
```

Value

an object of class "igraph" (see http://igraph.org/r/doc/aaa-igraph-package.html). It has attributes for both vertices and edges. Below are attributes for the vertices:

- name: unique id for the vertices
- seqid: protein seqid for the vertices
- geneid: Entrez geneid (if any) for the vertices
- symbol: gene symbol (if any) for the vertices
- description: gene description (if any) for the vertices

Below are attributes for the edges:

- neighborhood_score: predictive score based on neighborhood data
- fusion_score: predictive score based on fusion data
- cooccurence_score: predictive score based on cooccurence data
- coexpression_score: predictive score based on coexpression
- experimental_score: predictive score based on experimental data
- database_score: predictive score based on database
- textmining_score: predictive score based on text mining
- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Ce.string <- dRDataLoader(RData='org.Ce.string')
org.Ce.string
```

---

org.Da.eg   *Zebrafish Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the zebrafish. This data is prepared based on ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Da.eg <- dRDataLoader(RData='org.Da.eg')
```

Value

an object of class "EG", a list with following components:

- `gene_info`: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.Da.eg <- dRDataLoader(RData='org.Da.eg')
names(org.Da.eg)
org.Da.eg$gene_info[1:5,]
```

---

org.Da.egGOBP	*Annotations of Zebrafish Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Zebrafish Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Da.egGOBP <- dRDataLoader(RData='org.Da.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Da.egGOBP <- dRDataLoader(RData='org.Da.egGOBP')
names(org.Da.egGOBP)
```

org.Da.egGOCC   *Annotations of Zebrafish Entrez Genes (EG) by Gene Ontology Cellu-*
                              lar Component (GOCC).

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Ze-
brafish Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/
obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.
[gz](gz).

Usage

```
org.Da.egGOCC <- dRDataLoader(RData='org.Da.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number
  of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e.
  "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by
  "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-
57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Da.egGOCC <- dRDataLoader(RData='org.Da.egGOCC')
names(org.Da.egGOCC)
```

org.Da.egGOMF   *Annotations of Zebrafish Entrez Genes (EG) by Gene Ontology Molec-*
                              ular Function (GOMF).

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Ze-
brafish Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/
obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.
[gz](gz).

Usage

```
org.Da.egGOMF <- dRDataLoader(RData='org.Da.egGOMF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Da.egGOMF <- dRDataLoader(RData='org.Da.egGOMF')
names(org.Da.egGOMF)
```

---

org.Da.egPS   *Annotations of Zebrafish Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Zebrafish Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Zebrafish Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the zebrafish, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Da.egPS <- dRDataLoader(RData='org.Da.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or

left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Da.egPS <- dRDataLoader(RData='org.Da.egPS')
names(org.Da.egPS)
```

---

org.Da.egSF	*Annotations of Zebrafish Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Zebrafish Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Zebrafish Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Da.egSF <- dRDataLoader(RData='org.Da.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Da.egSF <- dRDataLoader(RData='org.Da.egSF')
names(org.Da.egSF)
```

---

org.Da.string	*Zebrafish functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in zebrafish. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Da.string <- dRDataLoader(RData='org.Da.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- name: unique id for the vertices
- seqid: protein seqid for the vertices
- geneid: Entrez geneid (if any) for the vertices
- symbol: gene symbol (if any) for the vertices
- description: gene description (if any) for the vertices

Below are attributes for the edges:

- neighborhood_score: predictive score based on neighborhood data
- fusion_score: predictive score based on fusion data
- cooccurence_score: predictive score based on cooccurence data
- coexpression_score: predictive score based on coexpression
- experimental_score: predictive score based on experimental data
- database_score: predictive score based on database
- textmining_score: predictive score based on text mining
- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Da.string <- dRDataLoader(RData='org.Da.string')
org.Da.string
```

---

org.Dm.eg   *Fruitfly Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the fruitfly. This data is prepared based on
ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Dm.eg <- dRDataLoader(RData='org.Dm.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number
  of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome",
  "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-
57.

Examples

```
org.Dm.eg <- dRDataLoader(RData='org.Dm.eg')
names(org.Dm.eg)
org.Dm.eg$gene_info[1:5,]
```

---

org.Dm.egGOBP   *Annotations of Fruitfly Entrez Genes (EG) by Gene Ontology Biologi-
                                   cal Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Fruit-
fly Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/
obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.
gz.

Usage

```
org.Dm.egGOBP <- dRDataLoader(RData='org.Dm.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Dm.egGOBP <- dRDataLoader(RData='org.Dm.egGOBP')
names(org.Dm.egGOBP)
```

---

org.Dm.egGOCC	*Annotations of Fruitfly Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Fruit-fly Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Dm.egGOCC <- dRDataLoader(RData='org.Dm.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Dm.egGOCC <- dRDataLoader(RData='org.Dm.egGOCC')
names(org.Dm.egGOCC)
```

---

org.Dm.egGOMF   *Annotations of Fruitfly Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Fruitfly Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Dm.egGOMF <- dRDataLoader(RData='org.Dm.egGOMF')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Dm.egGOMF <- dRDataLoader(RData='org.Dm.egGOMF')
names(org.Dm.egGOMF)
```

org.Dm.egPS	*Annotations of Fruitfly Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Fruitfly Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Fruitfly Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the fruitfly, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Dm.egPS <- dRDataLoader(RData='org.Dm.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Dm.egPS <- dRDataLoader(RData='org.Dm.egPS')
names(org.Dm.egPS)
```

org.Dm.egSF	*Annotations of Fruitfly Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Fruitfly Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Fruitfly Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Dm.egSF <- dRDataLoader(RData='org.Dm.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Dm.egSF <- dRDataLoader(RData='org.Dm.egSF')
names(org.Dm.egSF)
```

org.Dm.string	*Fruitfly functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in fruitfly. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Dm.string <- dRDataLoader(RData='org.Dm.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- name: unique id for the vertices
- seqid: protein seqid for the vertices
- geneid: Entrez geneid (if any) for the vertices
- symbol: gene symbol (if any) for the vertices
- description: gene description (if any) for the vertices

Below are attributes for the edges:

- neighborhood_score: predictive score based on neighborhood data
- fusion_score: predictive score based on fusion data
- cooccurence_score: predictive score based on cooccurence data
- coexpression_score: predictive score based on coexpression
- experimental_score: predictive score based on experimental data
- database_score: predictive score based on database
- textmining_score: predictive score based on text mining
- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Dm.string <- dRDataLoader(RData='org.Dm.string')
org.Dm.string
```

---

org.Gg.eg   *Chicken Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the chicken. This data is prepared based on <ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz>.

Usage

```
org.Gg.eg <- dRDataLoader(RData='org.Gg.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.Gg.eg <- dRDataLoader(RData='org.Gg.eg')
names(org.Gg.eg)
org.Gg.eg$gene_info[1:5,]
```

| org.Gg.egGOBP | *Annotations of Chicken Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).* |

Description

An R object that contains associations between Gene Ontology Biological Process terms and Chicken Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Gg.egGOBP <- dRDataLoader(RData='org.Gg.egGOBP')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Gg.egGOBP <- dRDataLoader(RData='org.Gg.egGOBP')
names(org.Gg.egGOBP)
```

---

org.Gg.egGOCC   *Annotations of Chicken Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Chicken Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Gg.egGOCC <- dRDataLoader(RData='org.Gg.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Gg.egGOCC <- dRDataLoader(RData='org.Gg.egGOCC')
names(org.Gg.egGOCC)
```

---

org.Gg.egGOMF   *Annotations of Chicken Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Chicken Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Gg.egGOMF <- dRDataLoader(RData='org.Gg.egGOMF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Gg.egGOMF <- dRDataLoader(RData='org.Gg.egGOMF')
names(org.Gg.egGOMF)
```

---

org.Gg.egPS   *Annotations of Chicken Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Chicken Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Chicken Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the chicken, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Gg.egPS <- dRDataLoader(RData='org.Gg.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or

left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.

- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Gg.egPS <- dRDataLoader(RData='org.Gg.egPS')
names(org.Gg.egPS)
```

---

org.Gg.egSF	*Annotations of Chicken Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Chicken Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Chicken Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Gg.egSF <- dRDataLoader(RData='org.Gg.egSF')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").

- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Gg.egSF <- dRDataLoader(RData='org.Gg.egSF')
names(org.Gg.egSF)
```

---

org.Gg.string   *Chicken functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in chicken. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Gg.string <- dRDataLoader(RData='org.Gg.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- `name`: unique id for the vertices
- `seqid`: protein seqid for the vertices
- `geneid`: Entrez geneid (if any) for the vertices
- `symbol`: gene symbol (if any) for the vertices
- `description`: gene description (if any) for the vertices

Below are attributes for the edges:

- `neighborhood_score`: predictive score based on neighborhood data
- `fusion_score`: predictive score based on fusion data
- `cooccurence_score`: predictive score based on cooccurence data
- `coexpression_score`: predictive score based on coexpression
- `experimental_score`: predictive score based on experimental data
- `database_score`: predictive score based on database
- `textmining_score`: predictive score based on text mining
- `combined_score`: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Gg.string <- dRDataLoader(RData='org.Gg.string')
org.Gg.string
```

---

org.Hs.eg   *Human Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the human. This data is prepared based on
ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Hs.eg <- dRDataLoader(RData='org.Hs.eg')
```

Value

an object of class "EG", a list with following components:

- `gene_info`: a matrix of nGene X 7 containing gene information, where nGene is the number
  of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome",
  "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-
57.

Examples

```
# not run
org.Hs.eg <- dRDataLoader(RData='org.Hs.eg')
names(org.Hs.eg)
org.Hs.eg$gene_info[1:5,]
```

---

org.Hs.egDGIdb   *Annotations of Human Entrez Genes (EG) by DGIdb categories.*

Description

An R object that contains associations between DGIdb categories and Human Entrez Genes. This
data is prepared based on http://dgidb.genome.wustl.edu/downloads/categories.tsv.

Usage

```
org.Hs.egDGIdb <- dRDataLoader(RData='org.Hs.egDGIdb')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number
  of gene sets (i.e. terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term
  Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by
  "setID" and gene members identified by "Entrez ID"

References

Griffith et al. (2013) DGIdb: mining the druggable genome. *Nature methods*, 10(12):1209-10.

Examples

```
org.Hs.egDGIdb <- dRDataLoader(RData='org.Hs.egDGIdb')
names(org.Hs.egDGIdb)
```

---

org.Hs.egDO   *Annotations of Human Entrez Genes (EG) by Disease Ontology (DO).*

Description

An R object that contains associations between Disease Ontology terms and Human Entrez Genes. This data is first prepared based on http://sourceforge.net/p/diseaseontology/code/HEAD/ tree/trunk/HumanDO.obo?format=raw and http://dga.nubic.northwestern.edu/ajax/Download. ajax.php, which results in annotations of Human Entrez Genes.

Usage

```
org.Hs.egDO <- dRDataLoader(RData='org.Hs.egDO')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. DO terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Schriml et al. (2012) Disease Ontology: a backbone for disease semantic integration. *Nucleic Acids Res*, 40:D940-946.
Peng et al. (2012) The Disease and Gene Annotations (DGA): an annotation resource for human disease. *Nucleic Acids Res*, 41:D553-560.
Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Hs.egDO <- dRDataLoader(RData='org.Hs.egDO')
names(org.Hs.egDO)
```

---

org.Hs.egGOBP | *Annotations of Human Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Human Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Hs.egGOBP <- dRDataLoader(RData='org.Hs.egGOBP')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Hs.egGOBP <- dRDataLoader(RData='org.Hs.egGOBP')
names(org.Hs.egGOBP)
```

---

org.Hs.egGOCC | *Annotations of Human Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Human Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Hs.egGOCC <- dRDataLoader(RData='org.Hs.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Hs.egGOCC <- dRDataLoader(RData='org.Hs.egGOCC')
names(org.Hs.egGOCC)
```

---

org.Hs.egGOMF	*Annotations of Human Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Human Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Hs.egGOMF <- dRDataLoader(RData='org.Hs.egGOMF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Hs.egGOMF <- dRDataLoader(RData='org.Hs.egGOMF')
names(org.Hs.egGOMF)
```

---

org.Hs.egHPCM	*Annotations of Human Entrez Genes (EG) by Human Phenotype Clinical Modifier (HPCM).*

Description

An R object that contains associations between HPCM terms and Human Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

```
org.Hs.egHPCM <- dRDataLoader(RData='org.Hs.egHPCM')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPCM terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
org.Hs.egHPCM <- dRDataLoader(RData='org.Hs.egHPCM')
names(org.Hs.egHPCM)
```

org.Hs.egHPMA	*Annotations of Human Entrez Genes (EG) by Human Phenotype Mortality Aging (HPMA).*

Description

An R object that contains associations between HPMA terms and Human Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

```
data(org.Hs.egHPMA)
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPMA terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
data(org.Hs.egHPMA)
names(org.Hs.egHPMA)
```

org.Hs.egHPMI	*Annotations of Human Entrez Genes (EG) by Human Phenotype Mode of Inheritance (HPMI).*

Description

An R object that contains associations between HPMI terms and Human Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

```
org.Hs.egHPMI <- dRDataLoader(RData='org.Hs.egHPMI')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPMI terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
org.Hs.egHPMI <- dRDataLoader(RData='org.Hs.egHPMI')
names(org.Hs.egHPMI)
```

`org.Hs.egHPPA`	*Annotations of Human Entrez Genes (EG) by Human Phenotype Phenotypic Abnormality (HPPA).*

Description

An R object that contains associations between HPPA terms and Human Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

```
data(org.Hs.egHPPA)
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPPA terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Examples

```
org.Hs.egHPPA <- dRDataLoader(RData='org.Hs.egHPPA')
names(org.Hs.egHPPA)
```

org.Hs.egMP	*Annotations of Human Entrez Genes (EG) by Mammalian Phenotype (MP).*

Description

An R object that contains associations between Mammalian Phenotype terms and Human Entrez Genes. This data is prepared based on ftp://ftp.informatics.jax.org/pub/reports/MPheno_OBO.ontology and ftp://ftp.informatics.jax.org/pub/reports/MGI_PhenoGenoMP.rpt, which results in annotations of Mouse Entrez Genes. Then, these annotations are transferred to Human Entrez Genes based on ftp://anonymous@ftp.ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Hs.egMP <- dRDataLoader(RData='org.Hs.egMP')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. MP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Smith et al. (2009) The Mammalian Phenotype Ontology: enabling robust annotation and comparative analysis. *Wiley Interdiscip Rev Syst Biol Med*, 1:390-399.

Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Hs.egMP <- dRDataLoader(RData='org.Hs.egMP')
names(org.Hs.egMP)
```

org.Hs.egMsigdbC1	*Annotations of Human Entrez Genes (EG) by C1 collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C1 positional gene sets and Human Entrez Genes. C1 collections are about positional gene sets for each human chromosome and cytogenetic band, each gene set corresponding to each human chromosome and each cytogenetic band that has at least one gene. These gene sets are helpful in identifying effects related to chromosomal deletions or amplifications, dosage compensation, epigenetic silencing, and other regional effects.

Usage

```
org.Hs.egMsigdbC1 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC1')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC1 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC1')
names(org.Hs.egMsigdbC1)
```

---

org.Hs.egMsigdbC2BIOCARTA

Annotations of Human Entrez Genes (EG) by C2:BIOCARTA collections.

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C2:BIOCARTA (BioCarta pathways) gene sets and Human Entrez Genes. C2:BIOCARTA gene sets are derived from the BioCarta pathway database http://www.biocarta.com/genes/index.asp.

Usage

```
org.Hs.egMsigdbC2BIOCARTA <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2BIOCARTA')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC2BIOCARTA <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2BIOCARTA')
names(org.Hs.egMsigdbC2BIOCARTA)
```

---

org.Hs.egMsigdbC2CGP   *Annotations of Human Entrez Genes (EG) by C2:CGP collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C2:CGP (chemical and genetic perturbations) gene sets and Human Entrez Genes. C2:CGP gene sets are about expression signatures of genetic and chemical perturbations. A number of these gene sets come in pairs: an xxx_UP (xxx_DN) gene set representing genes induced (repressed) by the perturbation.

Usage

```
org.Hs.egMsigdbC2CGP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2CGP')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC2CGP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2CGP')
names(org.Hs.egMsigdbC2CGP)
```

---

org.Hs.egMsigdbC2CP   *Annotations of Human Entrez Genes (EG) by C2:CP collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C2:CP (Canonical pathways) gene sets and Human Entrez Genes. C2:CP gene sets are from the pathway databases, and usually are canonical representations of a biological process compiled by domain experts.

Usage

```
org.Hs.egMsigdbC2CP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2CP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC2CP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2CP')
names(org.Hs.egMsigdbC2CP)
```

---

org.Hs.egMsigdbC2KEGG   *Annotations of Human Entrez Genes (EG) by C2:KEGG collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C2:KEGG (KEGG pathways) gene sets and Human Entrez Genes. C2:KEGG gene sets are derived from the KEGG pathway database http://www.genome.jp/kegg/pathway.html.

Usage

```
org.Hs.egMsigdbC2KEGG <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2KEGG')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of
  gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset
  brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset
  full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by
  "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for inter-
preting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC2KEGG <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2KEGG')
names(org.Hs.egMsigdbC2KEGG)
```

---

org.Hs.egMsigdbC2REACTOME

*Annotations of Human Entrez Genes (EG) by C2:REACTOME collec-
tions.*

Description

This data is prepared based on the molecular signatures database (Msigdb; [http://www.broadinstitute.](http://www.broadinstitute.org/gsea/msigdb/index.jsp)
[org/gsea/msigdb/index.jsp](http://www.broadinstitute.org/gsea/msigdb/index.jsp)). An R object that contains associations between Msigdb C2:REACTOME
(Reactome pathways) gene sets and Human Entrez Genes. C2:REACTOME gene sets are derived
from the Reactome pathway database http://www.reactome.org/.

Usage

```
org.Hs.egMsigdbC2REACTOME <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2REACTOME')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of
  gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset
  brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset
  full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by
  "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for inter-
preting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC2REACTOME <- dRDataLoader(RData.customised='org.Hs.egMsigdbC2REACTOME')
names(org.Hs.egMsigdbC2REACTOME)
```

---

org.Hs.egMsigdbC3MIR   *Annotations of Human Entrez Genes (EG) by C3:MIR collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C3:MIR (microRNA targets) gene sets and Human Entrez Genes. C3 collections are about motif gene sets that contain genes that share a cis-regulatory motif that is conserved across the human, mouse, rat, and dog genomes, and represent known or likely regulatory elements in promoters and 3'-UTRs. C3:MIR gene sets contain genes that share a 3'-UTR microRNA binding motif.

Usage

```
org.Hs.egMsigdbC3MIR <- dRDataLoader(RData.customised='org.Hs.egMsigdbC3MIR')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC3MIR <- dRDataLoader(RData.customised='org.Hs.egMsigdbC3MIR')
names(org.Hs.egMsigdbC3MIR)
```

org.Hs.egMsigdbC3TFT   *Annotations of Human Entrez Genes (EG) by C3:TFT collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C3:TFT (transcription factor targets) gene sets and Human Entrez Genes. C3 collections are about motif gene sets that contain genes that share a cis-regulatory motif that is conserved across the human, mouse, rat, and dog genomes, and represent known or likely regulatory elements in promoters and 3'-UTRs. C3:TFT gene sets contain genes that share a transcription factor binding site defined in the TRANSFAC (version 7.4, http://www.gene-regulation.com/) database.

Usage

```
org.Hs.egMsigdbC3TFT <- dRDataLoader(RData.customised='org.Hs.egMsigdbC3TFT')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC3TFT <- dRDataLoader(RData.customised='org.Hs.egMsigdbC3TFT')
names(org.Hs.egMsigdbC3TFT)
```

org.Hs.egMsigdbC4CGN   *Annotations of Human Entrez Genes (EG) by C4:CGN collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C4:CGN (cancer gene neighborhoods) gene sets and Human Entrez Genes. C4:CGN gene sets are defined by expression neighborhoods centered on 380 cancer-associated genes (see http://www.ncbi.nlm.nih.gov/pubmed/14593198).

Usage

```
org.Hs.egMsigdbC4CGN <- dRDataLoader(RData.customised='org.Hs.egMsigdbC4CGN')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC4CGN <- dRDataLoader(RData.customised='org.Hs.egMsigdbC4CGN')
names(org.Hs.egMsigdbC4CGN)
```

---

org.Hs.egMsigdbC4CM *Annotations of Human Entrez Genes (EG) by C4:CM collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C4:CM (cancer modules) gene sets and Human Entrez Genes. C4:CM gene sets are defined in http://www.ncbi.nlm.nih.gov/pubmed/15448693; the authors first compiled gene sets ('modules') from a variety of resources such as KEGG, GO, and others, and then by mining a large compendium of cancer-related microarray data, they identified 456 such modules as significantly changed in a variety of cancer conditions.

Usage

```
org.Hs.egMsigdbC4CM <- dRDataLoader(RData.customised='org.Hs.egMsigdbC4CM')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC4CM <- dRDataLoader(RData.customised='org.Hs.egMsigdbC4CM')
names(org.Hs.egMsigdbC4CM)
```

---

org.Hs.egMsigdbC5BP   *Annotations of Human Entrez Genes (EG) by C5:BP collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C5:BP (GO biological process) gene sets and Human Entrez Genes.

Usage

```
org.Hs.egMsigdbC5BP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5BP')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC5BP <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5BP')
names(org.Hs.egMsigdbC5BP)
```

---

org.Hs.egMsigdbC5CC   *Annotations of Human Entrez Genes (EG) by C5:CC collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C5:CC (GO cellular component) gene sets and Human Entrez Genes.

Usage

```
org.Hs.egMsigdbC5CC <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5CC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC5CC <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5CC')
names(org.Hs.egMsigdbC5CC)
```

---

org.Hs.egMsigdbC5MF   *Annotations of Human Entrez Genes (EG) by C5:MF collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C5:MF (GO molecular function) gene sets and Human Entrez Genes.

Usage

```
org.Hs.egMsigdbC5MF <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5MF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC5MF <- dRDataLoader(RData.customised='org.Hs.egMsigdbC5MF')
names(org.Hs.egMsigdbC5MF)
```

---

org.Hs.egMsigdbC6   *Annotations of Human Entrez Genes (EG) by C6 collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C6 oncogenic signature gene sets and Human Entrez Genes. C6 collections contain gene sets that represent signatures of cellular pathways which are often dis-regulated in cancer.

Usage

```
org.Hs.egMsigdbC6 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC6')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC6 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC6')
names(org.Hs.egMsigdbC6)
```

---

org.Hs.egMsigdbC7 *Annotations of Human Entrez Genes (EG) by C7 collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb C7 immunologic signature gene sets and Human Entrez Genes. C7 collections contain gene sets that represent cell states and perturbations within the immune system.

Usage

```
org.Hs.egMsigdbC7 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC7')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbC7 <- dRDataLoader(RData.customised='org.Hs.egMsigdbC7')
names(org.Hs.egMsigdbC7)
```

---

org.Hs.egMsigdbH *Annotations of Human Entrez Genes (EG) by H hallmark collections.*

Description

This data is prepared based on the molecular signatures database (Msigdb; http://www.broadinstitute.org/gsea/msigdb/index.jsp). An R object that contains associations between Msigdb H hallmark gene sets and Human Entrez Genes. This collection is the starting point for the exploration of the MSigDB resource. Hallmark gene sets summarize and represent specific well-defined biological states or processes and display coherent expression. These gene sets were generated by a computational methodology based on identifying gene set overlaps and retaining genes that display coordinate expression. The hallmarks reduce noise and redundancy and provide a better delineated biological space.

Usage

```
org.Hs.egMsigdbH <- dRDataLoader(RData.customised='org.Hs.egMsigdbH')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets, and the 4 columns are "setID" (i.e. "Geneset standard name"), "name" (i.e. "Geneset brief description"), "namespace" (i.e. Geneset category code) and "distance" (i.e. "Geneset full description")

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Subramanian et al. (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc Natl Acad Sci U S A.*, 102(43):15545-50.

Examples

```
org.Hs.egMsigdbH <- dRDataLoader(RData.customised='org.Hs.egMsigdbH')
names(org.Hs.egMsigdbH)
```

---

org.Hs.egPS   *Annotations of Human Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Human Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Human Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the human, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Hs.egPS <- dRDataLoader(RData='org.Hs.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Hs.egPS <- dRDataLoader(RData='org.Hs.egPS')
names(org.Hs.egPS)
```

---

org.Hs.egSF	*Annotations of Human Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Human Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Human Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Hs.egSF <- dRDataLoader(RData='org.Hs.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Hs.egSF <- dRDataLoader(RData='org.Hs.egSF')
names(org.Hs.egSF)
```

---

org.Hs.string	*Human functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in human. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Hs.string <- dRDataLoader(RData='org.Hs.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- name: unique id for the vertices
- seqid: protein seqid for the vertices
- geneid: Entrez geneid (if any) for the vertices
- symbol: gene symbol (if any) for the vertices
- description: gene description (if any) for the vertices

Below are attributes for the edges:

- neighborhood_score: predictive score based on neighborhood data
- fusion_score: predictive score based on fusion data
- cooccurence_score: predictive score based on cooccurence data
- coexpression_score: predictive score based on coexpression
- experimental_score: predictive score based on experimental data
- database_score: predictive score based on database
- textmining_score: predictive score based on text mining
- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Hs.string <- dRDataLoader(RData='org.Hs.string')
org.Hs.string
```

---

org.Mm.eg   *Mouse Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the mouse. This data is prepared based on ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Mm.eg <- dRDataLoader(RData='org.Mm.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.Mm.eg <- dRDataLoader(RData='org.Mm.eg')
names(org.Mm.eg)
org.Mm.eg$gene_info[1:5,]
```

org.Mm.egDO   *Annotations of Mouse Entrez Genes (EG) by Disease Ontology (DO).*

Description

An R object that contains associations between Disease Ontology terms and Mouse Entrez Genes. This data is first prepared based on http://sourceforge.net/p/diseaseontology/code/HEAD/tree/trunk/HumanDO.obo?format=raw and http://dga.nubic.northwestern.edu/ajax/Download.ajax.php, which results in annotations of Human Entrez Genes. Then, these annotations are transferred to Mouse Entrez Genes based on ftp://anonymous@ftp.ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Mm.egDO <- dRDataLoader(RData='org.Mm.egDO')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. DO terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Schriml et al. (2012) Disease Ontology: a backbone for disease semantic integration. *Nucleic Acids Res*, 40:D940-946.
Peng et al. (2012) The Disease and Gene Annotations (DGA): an annotation resource for human disease. *Nucleic Acids Res*, 41:D553-560.
Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Mm.egDO <- dRDataLoader(RData='org.Mm.egDO')
names(org.Mm.egDO)
```

org.Mm.egGOBP   *Annotations of Mouse Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Mouse Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Mm.egGOBP <- dRDataLoader(RData='org.Mm.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Mm.egGOBP <- dRDataLoader(RData='org.Mm.egGOBP')
names(org.Mm.egGOBP)
```

---

org.Mm.egGOCC	*Annotations of Mouse Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Mouse Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Mm.egGOCC <- dRDataLoader(RData='org.Mm.egGOCC')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Mm.egGOCC <- dRDataLoader(RData='org.Mm.egGOCC')
names(org.Mm.egGOCC)
```

---

org.Mm.egGOMF	*Annotations of Mouse Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Mouse Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Mm.egGOMF <- dRDataLoader(RData='org.Mm.egGOMF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Mm.egGOMF <- dRDataLoader(RData='org.Mm.egGOMF')
names(org.Mm.egGOMF)
```

---

org.Mm.egHPCM *Annotations of Mouse Entrez Genes (EG) by Human Phenotype Clinical Modifier (HPCM).*

Description

An R object that contains associations between HPCM terms and Mouse Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite. de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ ALL_FREQUENCIES_genes_to_phenotype.txt, which results in annotations of Human Entrez Genes. Then, these annotations are transferred to Mouse Entrez Genes based on ftp://anonymous@ftp. ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Mm.egHPCM <- dRDataLoader(RData='org.Mm.egHPCM')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPCM terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Mm.egHPCM <- dRDataLoader(RData='org.Mm.egHPCM')
names(org.Mm.egHPCM)
```

---

org.Mm.egHPMA *Annotations of Mouse Entrez Genes (EG) by Human Phenotype Mortality Aging (HPMA).*

Description

An R object that contains associations between HPMA terms and Mouse Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite. de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ ALL_FREQUENCIES_genes_to_phenotype.txt, which results in annotations of Human Entrez Genes. Then, these annotations are transferred to Mouse Entrez Genes based on ftp://anonymous@ftp. ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Mm.egHPMA <- dRDataLoader(RData='org.Mm.egHPMA')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPMA terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Mm.egHPMA <- dRDataLoader(RData='org.Mm.egHPMA')
names(org.Mm.egHPMA)
```

---

org.Mm.egHPMI	*Annotations of Mouse Entrez Genes (EG) by Human Phenotype Mode of Inheritance (HPMI).*

Description

An R object that contains associations between HPMI terms and Mouse Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite. de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ ALL_FREQUENCIES_genes_to_phenotype.txt, which results in annotations of Human Entrez Genes. Then, these annotations are transferred to Mouse Entrez Genes based on ftp://anonymous@ftp. ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Mm.egHPMI <- dRDataLoader(RData='org.Mm.egHPMI')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPMI terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Mm.egHPMI <- dRDataLoader(RData='org.Mm.egHPMI')
names(org.Mm.egHPMI)
```

---

org.Mm.egHPPA	*Annotations of Mouse Entrez Genes (EG) by Human Phenotype Phenotypic Abnormality (HPPA).*

Description

An R object that contains associations between HPPA terms and Mouse Entrez Genes. This data is first prepared based on http://purl.obolibrary.org/obo/hp.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt, which results in annotations of Human Entrez Genes. Then, these annotations are transferred to Mouse Entrez Genes based on ftp://anonymous@ftp.ncbi.nih.gov/pub/HomoloGene/build67/homologene.data.

Usage

```
org.Mm.egHPPA <- dRDataLoader(RData='org.Mm.egHPPA')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. HPPA terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Robinson et al. (2012) The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83:610-615.

Sayers et al. (2011) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 39:D38-51.

Examples

```
org.Mm.egHPPA <- dRDataLoader(RData='org.Mm.egHPPA')
names(org.Mm.egHPPA)
```

org.Mm.egMP	*Annotations of Mouse Entrez Genes (EG) by Mammalian Phenotype (MP).*

Description

An R object that contains associations between Mammalian Phenotype terms and Mouse Entrez Genes. This data is prepared based on `ftp://ftp.informatics.jax.org/pub/reports/MPheno_OBO.ontology` and `ftp://ftp.informatics.jax.org/pub/reports/MGI_PhenoGenoMP.rpt`.

Usage

```
org.Mm.egMP <- dRDataLoader(RData='org.Mm.egMP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. MP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Smith et al. (2009) The Mammalian Phenotype Ontology: enabling robust annotation and comparative analysis. *Wiley Interdiscip Rev Syst Biol Med*, 1:390-399.

Examples

```
org.Mm.egMP <- dRDataLoader(RData='org.Mm.egMP')
names(org.Mm.egMP)
```

org.Mm.egPS	*Annotations of Mouse Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Mouse Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Mouse Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the mouse, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Mm.egPS <- dRDataLoader(RData='org.Mm.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.
Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Mm.egPS <- dRDataLoader(RData='org.Mm.egPS')
names(org.Mm.egPS)
```

---

org.Mm.egSF	*Annotations of Mouse Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Mouse Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Mouse Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Mm.egSF <- dRDataLoader(RData='org.Mm.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Mm.egSF <- dRDataLoader(RData='org.Mm.egSF')
names(org.Mm.egSF)
```

---

org.Mm.string	*Mouse functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in mouse. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Mm.string <- dRDataLoader(RData='org.Mm.string')
```

Value

an object of class "igraph" (see <http://igraph.org/r/doc/aaa-igraph-package.html>). It has attributes for both vertices and edges. Below are attributes for the vertices:

- `name`: unique id for the vertices
- `seqid`: protein seqid for the vertices
- `geneid`: Entrez geneid (if any) for the vertices
- `symbol`: gene symbol (if any) for the vertices
- `description`: gene description (if any) for the vertices

Below are attributes for the edges:

- `neighborhood_score`: predictive score based on neighborhood data
- `fusion_score`: predictive score based on fusion data

- cooccurence_score: predictive score based on cooccurence data

- coexpression_score: predictive score based on coexpression

- experimental_score: predictive score based on experimental data

- database_score: predictive score based on database

- textmining_score: predictive score based on text mining

- combined_score: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Mm.string <- dRDataLoader(RData='org.Mm.string')
org.Mm.string
```

---

org.Rn.eg   *Rat Entrez Genes (EG).*

Description

An R object that contains Entrez Gene information for the rat. This data is prepared based on ftp://ftp.ncbi.nih.gov/gene/DATA/gene_info.gz.

Usage

```
org.Rn.eg <- dRDataLoader(RData='org.Rn.eg')
```

Value

an object of class "EG", a list with following components:

- gene_info: a matrix of nGene X 7 containing gene information, where nGene is the number of Entrez Genes, and the 7 columns are "GeneID", "Symbol", "description", "chromosome", "map_location", "Synonyms" and "dbXrefs"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.

Examples

```
org.Rn.eg <- dRDataLoader(RData='org.Rn.eg')
names(org.Rn.eg)
org.Rn.eg$gene_info[1:5,]
```

---

org.Rn.egGOBP	*Annotations of Rat Entrez Genes (EG) by Gene Ontology Biological Process (GOBP).*

Description

An R object that contains associations between Gene Ontology Biological Process terms and Rat Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Rn.egGOBP <- dRDataLoader(RData='org.Rn.egGOBP')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOBP terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Rn.egGOBP <- dRDataLoader(RData='org.Rn.egGOBP')
names(org.Rn.egGOBP)
```

---

org.Rn.egGOCC	*Annotations of Rat Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component terms and Rat Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Rn.egGOCC <- dRDataLoader(RData='org.Rn.egGOCC')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOCC terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Rn.egGOCC <- dRDataLoader(RData='org.Rn.egGOCC')
names(org.Rn.egGOCC)
```

---

org.Rn.egGOMF	*Annotations of Rat Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function terms and Rat Entrez Genes. This data is prepared based on http://www.geneontology.org/ontology/obo_format_1_2/gene_ontology.1_2.obo and ftp://ftp.ncbi.nih.gov/gene/DATA/gene2go.gz.

Usage

```
org.Rn.egGOMF <- dRDataLoader(RData='org.Rn.egGOMF')
```

Value

an object of class "GS", a list with following components:

- set_info: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. GOMF terms), and the 4 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- gs: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Maglott et al. (2011) Entrez Gene: gene-centered information at NCBI. *Nucleic Acids Res*, 39:D52-57.
Ashburner et al. (2000) Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25-29.

Examples

```
org.Rn.egGOMF <- dRDataLoader(RData='org.Rn.egGOMF')
names(org.Rn.egGOMF)
```

org.Rn.egPS   *Annotations of Rat Entrez Genes (EG) by phylostratific age (PS).*

Description

An R object that contains phylostratific age information for Rat Entrez Genes. This data is prepared based on 1) SUPERFAMILY database which provides domain architecture assignments to all completely sequenced genomes including eukaryotic genomes; 2) ancestral domain architecture repertoires inferred by applying Dollo parsimony to eukaryotic part of species tree of life (sTOL), from which the most recent common ancestor of each domain architecture is determined. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, phylostratific age for a Rat Entrez gene is the first appearance of its domain architecture along the branch from the eukaryotic ancestor to the rat, and thus can be measured by: i) the most recent common ancestor, ii) how many steps it is away starting from the eukaryotic ancestor, and how far it is in the terms of the branch length from the eukaryotic ancestor.

Usage

```
org.Rn.egPS <- dRDataLoader(RData='org.Rn.egPS')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. phylogenetic placement along the branch starting from the eukaryotic ancestor). The 4 columns are "setID" (i.e. "phylogenetic placement ID"), "name" (i.e. name for that placement in the form of "TaxonID:Name"), "namespace" (i.e. Rank for that placement) and "distance" (i.e. the branch length from the eukaryotic ancestor). Notably, since the sTOL is bifurcating with exactly two descendants (unlike the multifurcating nature of the NCBI taxonomy), an internal node in sTOL is either mapped onto a unique taxonomic identifier or left empty (assumedly a hypothetical unknown ancestor). In the latter case, hypothetical unknown ancestor is filled with the information in its nearest descendant with known taxonomic information.

- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Fang et al. (2013) A daily-updated tree of (sequenced) life as a reference for genome research. *Scientific reports*, 3:2015.

Examples

```
org.Rn.egPS <- dRDataLoader(RData='org.Rn.egPS')
names(org.Rn.egPS)
```

---

org.Rn.egSF *Annotations of Rat Entrez Genes (EG) by domain superfamilies (SF).*

Description

An R object that contains domain superfamily information for Rat Entrez Genes. This data is prepared based on SUPERFAMILY database, which provides SCOP domain architecture assignments to all completely sequenced genomes including eukaryotic genomes. The domain architecture for an Entrez gene is the protein product with the longest length of amino acids. Thus, domain superfamily information for Rat Entrez gene is a list of domain superfamilies (excluding unknown gap) appearing in its domain architecture.

Usage

```
org.Rn.egSF <- dRDataLoader(RData='org.Rn.egSF')
```

Value

an object of class "GS", a list with following components:

- `set_info`: a matrix of nSet X 4 containing gene set information, where nSet is the number of gene sets (i.e. SCOP domain superfamilies). The 4 columns are "setID" (i.e. "SCOP domain identifier"), "name" (i.e. "SCOP domain description"), "namespace" (i.e. "SCOP domain level") and "distance" (i.e. "SCOP domain classification").
- `gs`: a list of gene sets, each storing gene members thereof. Always, gene sets are identified by "setID" and gene members identified by "Entrez ID"

References

Morais et al. (2011) SUPERFAMILY 1.75 including a domain-centric gene ontology method. *Nucleic Acids Res*, 39(Database issue):D427-34.

Andreeva et al. (2008) Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res*, 36(Database issue):D419-425

Examples

```
org.Rn.egSF <- dRDataLoader(RData='org.Rn.egSF')
names(org.Rn.egSF)
```

---

org.Rn.string *Rat functional protein association network from STRING (version 10).*

Description

An igraph object that contains a functional protein association network in rat. The network is extracted from the STRING database (version 10). Only those associations with medium confidence (score>=0.4) are retained.

Usage

```
org.Rn.string <- dRDataLoader(RData='org.Rn.string')
```

Value

an object of class "igraph" (see http://igraph.org/r/doc/aaa-igraph-package.html). It has attributes for both vertices and edges. Below are attributes for the vertices:

- `name`: unique id for the vertices
- `seqid`: protein seqid for the vertices
- `geneid`: Entrez geneid (if any) for the vertices
- `symbol`: gene symbol (if any) for the vertices
- `description`: gene description (if any) for the vertices

Below are attributes for the edges:

- `neighborhood_score`: predictive score based on neighborhood data
- `fusion_score`: predictive score based on fusion data
- `cooccurence_score`: predictive score based on cooccurence data
- `coexpression_score`: predictive score based on coexpression
- `experimental_score`: predictive score based on experimental data
- `database_score`: predictive score based on database
- `textmining_score`: predictive score based on text mining
- `combined_score`: combined score from all above predictive scores

References

Szklarczyk et al. (2015) STRING v10: protein-protein interaction networks, integrated over the tree of life. *Nucleic Acids Res*, 43:D447-52.

Examples

```
org.Rn.string <- dRDataLoader(RData='org.Rn.string')
org.Rn.string
```

TCGA_mutations   *TCGA mutational profiles across 12 major cancer types from Kandoth et al. (2013)*

Description

This dataset is available from TCGA, containing somatic mutational profiles for 3096 cancer samples with survival data. These cancer samples belong to one of 12 major cancer types, including breast adenocarcinoma (BRCA), lung adenocarcinoma (LUAD), lung squamous cell carcinoma (LUSC), uterine corpus endometrial carcinoma (UCEC), glioblastoma multiforme (GBM), head and neck squamous cell carcinoma (HNSC), colon and rectal carcinoma (COAD/READ), bladder urothelial carcinoma (BLCA), kidney renal clear cell carcinoma (KIRC), ovarian serous carcinoma (OV) and acute myeloid leukaemia (LAML). For each patient sample, somatic mutations are represented as a profile of states on genes, where non-zero entry indicates a gene for which how many mutations have occurred in the tumor relative to germ line. The dataset is provided as an 'ExpressionSet' object.

Usage

```
data(TCGA_mutations)
```

Value

an object of class "ExpressionSet". It has slots for "assayData", "phenoData", and "featureData":

- `assayData`: a matrix of 19171 genes X 3096 samples
- `phenoData`: variables describing sample phenotypes (i.e. columns in assayData), including clinical/survival information about samples: "time" (i.e. survival time in days), "status" (i.e., survival status: 0=alive; 1=dead), "Age" (the patient age in years), "Gender" (the patient gender: male/female), "TCGA_tumor_type", "Tumor_stage", "Tumor_grade"
- `featureData`: variables describing features (i.e. rows in assayData), including information about features/genes: "EntrezID" for gene EntrezID, "Symbol" for gene symbol, "Desc" for gene description, "Synonyms" for gene symbol alias

References

Kandoth et al. (2013). Mutational landscape and significance across 12 major cancer types. *Nature*, 502(7471):333-9.

Examples

```
#TCGA_mutations <- dRDataLoader(RData='TCGA_mutations')
data(TCGA_mutations)
TCGA_mutations
library(Biobase)
# extract information about the first 5 samples
pData(TCGA_mutations)[1:5,]
# extract information about the first 5 features
fData(TCGA_mutations)[1:5,]
# number of samples for each cancer type
table(pData(TCGA_mutations)$TCGA_tumor_type)
```

---

visBoxplotAdv   *Function to visualise a data frame using advanced boxplot*

Description

visBoxplotAdv is supposed to visualise a data frame using advanced boxplot. In addition to box-plot, a scatter plot is also drawn with various methods to avoid co-incident points so that each point is visible (with fine-controling the color and plotting character). Also, these points can be pies or thermometers, which allows an additional proportation data to be visualised as well.

Usage

```
visBoxplotAdv(formula, data, orientation = c("vertical", "horizontal"),
method = c("center", "hex", "square", "swarm"), corral = c("none",
"gutter", "wrap", "random", "omit"), corralWidth, cex = 1, spacing = 1,
breaks = NULL, labels, at = NULL, add = FALSE, log = FALSE,
xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,
pch = c("circles", "thermometers", "pies")[1], col =
```

```
graphics::par("col"),
bg = NA, pwpch = NULL, pwcol = NULL, pwbg = NULL, pwpie = NULL,
do.plot = TRUE, do.boxplot = TRUE, boxplot.notch = FALSE,
boxplot.border = "#888888C0", boxplot.col = "transparent", ...)
```

Arguments

formula	a formula, such as 'y ~ grp', where 'y' is a numeric vector of data values to be split into groups according to the grouping variable 'grp' (usually a factor)
data	a data.frame (or list) from which the variables in 'formula' should be taken.
orientation	the orientation. It can be one of "vertical" for the vertical orientation, "horizontal" for the horizontal orientation
method	the method for arranging the points. It can be one of "swarm" for arranging points in increasing order (if a point would overlap an existing point, it is shifted sideways (along the group axis) by a minimal amount sufficient to avoid overlap), "center" for first discretizing the values along the data axis (in order to create more efficient packing) and then using a square grid to produce a symmetric swarm, "hex" for first discretization and then arranging points in a hexagonal grid, and "square" for first discretization and then arranging points in a square grid
corral	the method to adjust points that would be placed outside their own group region. It can be one of "none" for not adjusting runaway points, "gutter" for collecting runaway points along the boundary between groups, "wrap" for wrapping runaway points to produce periodic boundaries, "random" for placing runaway points randomly in the region, and "omit" for omitting runaway points
corralWidth	the width of the "corral" in user coordinates
cex	size of points relative to the default. This must be a single value
spacing	relative spacing between points
breaks	breakpoints (optional). If NULL, breakpoints are chosen automatically
labels	labels for each group. Recycled if necessary. By default, these are inferred from the data
at	numeric vector giving the locations where the swarms should be drawn; defaults to '1:n' where n is the number of groups
add	whether to add to an existing plot
log	whether to use a logarithmic scale on the data axis
xlim	limits for x-axis
ylim	limits for y-axis
xlab	labels for x-aixs
ylab	labels for y-aixs
pch	plotting characters, specified by group and recycled if necessary. In additon to the convertional pch values, it can also be "circles", "thermometers", or "pies". For "pies" (or "thermometers"), users can also specify the proportional values (see below "pwpie") to visualise another information in the pie (or themometer) chart
col	plotting colors, specified by group and recycled if necessary
bg	plotting background, specified by group and recycled if necessary
pwpch	point-wise version of pch

pwcol	point-wise version of col
pwbg	point-wise version of bg
pwpie	point-wise proportion used when drawing pies or themometers
do.plot	whether to draw main plot
do.boxplot	whether to draw boxplot. It only works when the main plot is drawn
boxplot.notch	whether to draw a notch in the boxplot. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ
boxplot.border	the color for the outlines of the boxplots
boxplot.col	the color for the bodies of the boxplots
...	additional graphic parameters for the plot

Value

A data frame with plotting information. It has the same row names as the input data

Note

See Also

[visBoxplotAdv](#)

Examples

```
data(TCGA_mutations)
pd <- Biobase::pData(TCGA_mutations)
# only tumor types "LAML" or "BLCA"
data <- pd[pd$TCGA_tumor_type=="LAML" | pd$TCGA_tumor_type=="BLCA",]
labels <- levels(as.factor(data$TCGA_tumor_type))
# colors for gender
pwcol <- as.numeric((data$Gender))
# pie for relative age
pwpie <- data$Age/(max(data$Age))
out <- visBoxplotAdv(formula=time ~ TCGA_tumor_type, data=data,
pch="pies", pwcol=pwcol, pwpie=pwpie)
legend("topright", legend=levels(data$Gender), box.col="transparent",
pch=19, col=unique(pwcol))
```

---

visDAG	*Function to visualise a direct acyclic graph (DAG) with node colorings according to a named input data vector (if provided)*

Description

visDAG is supposed to visualise a direct acyclic graph (DAG) with node colorings according to a named input data vector (if provided)

Usage

```
visDAG(g, data = NULL, height = 7, width = 7, margin = rep(0.1, 4),
colormap = c("yr", "bwr", "jet", "gbr", "wyr", "br", "rainbow", "wb",
"lightyellow-orange"), ncolors = 40, zlim = NULL, colorbar = T,
colorbar.fraction = 0.1, newpage = T,
layout.orientation = c("left_right", "top_bottom", "bottom_top",
"right_left"), node.info = c("none", "term_id", "term_name", "both",
"full_term_name"), numChar = 15, graph.node.attrs = NULL,
graph.edge.attrs = NULL, node.attrs = NULL)
```

Arguments

g   an object of class "igraph"

data   a named input data verctor used to color-code vertices/nodes. The input data
                     vector must have names, and these names should include all node names of
                     input graph, i.e. V(g)$name, since there is a mapping operation. The way of
                     how to color-code is to map values in the data onto the whole colormap (see the
                     next arguments: colormap, ncolors, zlim and colorbar)

height   a numeric value specifying the height of device

width   a numeric value specifying the width of device

margin   margins as units of length 4 or 1

colormap   short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-
                     white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-
                     red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb"
                     (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-
                     green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color
                     names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-
                     white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names
                     can be found in <http://html-color-codes.info/color-names>

ncolors   the number of colors specified over the colormap

zlim   the minimum and maximum z/data values for which colors should be plotted,
                     defaulting to the range of the finite values of z. Each of the given colors will be
                     used to color an equispaced interval of this range. The midpoints of the intervals
                     cover the range, so that values just outside the range will be plotted

colorbar   logical to indicate whether to append a colorbar. If data is null, it always sets to
                     false

colorbar.fraction
                     the relative fraction of colorbar block against the device size

newpage   logical to indicate whether to open a new page. By default, it sets to true for
                     opening a new page

layout.orientation
                     the orientation of the DAG layout. It can be one of "left_right" for the left-
                     right layout (viewed from the DAG root point), "top_bottom" for the top-bottom
                     layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left
                     layout

node.info   tells the ontology term information used to label nodes. It can be one of "none"
                     for no node labeling, "term_id" for using Term ID, "term_name" for using Term
                     Name (the first 15 characters), "both" for using both of Term ID and Name (the
                     first 15 characters), and "full_term_name" for using the full Term Name

numChar   a positive integer specifying wrap width of node labelling

graph.node.attrs

                 a list of global node attributes. These node attributes will be changed globally.
                 See 'Note' below for details on the attributes

graph.edge.attrs

                 a list of global edge attributes. These edge attributes will be changed globally.
                 See 'Note' below for details on the attributes

node.attrs   a list of local edge attributes. These node attributes will be changed locally; as
                 such, for each attribute, the input value must be a named vector (i.e. using Term
                 ID as names). See 'Note' below for details on the attributes

Value

An object of class 'Ragraph'

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings
- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings
- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node
- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings

- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

[dDAGreverse](), [dDAGroot](), [dDAGinduce](), [dDAGlevel]()

Examples

```
# 1) load HPPA as igraph object
ig.HPPA <-dRDataLoader(RData='ig.HPPA')
g <- ig.HPPA

# 2) randomly select vertices as the query nodes
# the more common, the query nodes can be term id
nodes_query <- V(g)[sample(V(g),5)]$name

# 3) obtain the induced subgraph based on all possible paths
subg <- dDAGinduce(g, nodes_query, path.mode="all_paths")

# 4) just visualise the induced subgraph
visDAG(g=subg, node.info="both")

# 5) color-code nodes/terms according to its level
data <- dDAGlevel(subg)
visDAG(g=subg, data=data, node.info="both")
# 5a) globally change the node and edge attributes
visDAG(g=subg, data=data, layout.orientation="top_bottom",
node.info="both",
graph.node.attrs=list(fixedsize=FALSE,shape="box",color="transparent"),
graph.edge.attrs=list(color="black"))
# 5b) locally highlight the root by changing its shape into "box"
root <- dDAGroot(subg)
root.shape <- "box"
names(root.shape) <- V(subg)[root]$name
visDAG(g=subg, data=data, node.info="both",
node.attrs=list(shape=root.shape))
# 5c) further locally remove the root labelling
root.label <- ""
names(root.label) <- V(subg)[root]$name
visDAG(g=subg, data=data, node.info="both",
node.attrs=list(shape=root.shape,label=root.label))
```

---

visGSEA   *Function to visualise running enrichment score for a given sample and*
                                 a gene set

Description

visGSEA is supposed to visualise running enrichment score for a given sample and a gene set. To help understand the underlying running enrichment score, the input gene scores are also displayed. Positions for members in the given gene set are color-coded in both displays (red line for the positive gene scores, and green line for the negative).

Usage

```
visGSEA(eTerm, which_sample = 1, which_term = "GO:0006281", plot = T,
orientation = c("vertical", "horizontal"), hit.linewidth = 0.5,
newpage = T)
```

Arguments

eTerm	an object of class "eTerm"
which_sample	which sample will be used. It can be index or sample names
which_term	which term will be used. It can be index or term ID or term names
plot	logical to indicate whether to plot
orientation	the orientation of the plots. It can be either "vertical" (default) or "horizontal"
hit.linewidth	the line width for the hits (ie genes in the gene set)
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

Value

leading genes (being sorted)

Note

See Also

[dGSEA](), [dGSEAview]()

Examples

```
#visGSEA(eTerm, which_sample=1, which_term=1)
```

---

visNet   *Function to visualise a graph object of class "igraph" or "graphNEL"*

Description

visNet is supposed to visualise a graph object of class "igraph" or "graphNEL". It also allows the color-coding of vertices by providing the input pattern.

Usage

```
visNet(g, pattern = NULL, colormap = c("bwr", "jet", "gbr", "wyr",
"br",
"yr", "rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
newpage = T, glayout = layout.fruchterman.reingold,
vertex.frame.color = NA, vertex.size = NULL, vertex.color = NULL,
vertex.shape = NULL, vertex.label = NULL, vertex.label.cex = NULL,
vertex.label.dist = NULL, vertex.label.color = "black",
vertex.label.family = "sans", ...)
```

Arguments

g	an object of class "igraph" or "graphNEL"
pattern	a numeric vector used to color-code vertices/nodes. Notably, if the input vector contains names, then these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph; otherwise, this input pattern will be ignored. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
glayout	either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout.auto", "layout.random", "layout.circle", "layout.sphere", "layout.fruchterman.reingold", "layout.kamada.kawai", "layout.spring", "layout.reingold.tilford", "layout.fruchterman.reingold.grid", "layout.lgl", "layout.graphopt", "layout.svd" and "layout.norm". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html
vertex.frame.color	
	the color of the frame of the vertices. If it is NA, then there is no frame
vertex.size	the size of each vertex. If it is a vector, each vertex may differ in size
vertex.color	the fill color of the vertices. If it is NA, then there is no fill color. If the pattern is given, this setup will be ignored
vertex.shape	the shape of each vertex. It can be one of "circle", "square", "csquare", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.org/r/doc/vertex.shape.pie.html), "sphere", and "none". If it sets to NULL, these vertices with negative will be "csquare" and the rest "circle".
vertex.label	the label of the vertices. If it is NA, then there is no label. The default vertex labels are the name attribute of the nodes
vertex.label.cex	
	the font size of vertex labels.

vertex.label.dist

> the distance of the label from the center of the vertex. If it is 0 then the label is centered on the vertex. If it is 1 then the label is displayed beside the vertex.

vertex.label.color

> the color of vertex labels.

vertex.label.family

> the font family of vertex labels

...

> additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

invisible

Note

See Also

dNetFind

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) visualise the subg with vertices being color-coded by the pattern
pattern <- runif(vcount(subg))
names(pattern) <- V(subg)$name
visNet(g=subg, pattern=pattern, colormap="bwr", vertex.shape="sphere")
```

---

visNetAnimate	*Function to animate the same graph but with multiple graph node colorings according to input data matrix*

Description

visNetAnimate is supposed to animate the same graph but with multiple colorings according to input data matrix. The output can be a pdf file containing a list of frames/images, a mp4 video file or a gif file. To support video output file, the software 'ffmpeg' must be first installed (also put its path into the system PATH variable; see Note). To support gif output file, the software 'ImageMagick' must be first installed (also put its path into the system PATH variable; see Note).

Usage

```
visNetAnimate(g, data, filename = "visNetAnimate", filetype = c("pdf",
"mp4", "gif"), image.type = c("jpg", "png"), num.frame = ncol(data),
sec_per_frame = 1, height.device = 7, margin = rep(0.1, 4),
border.color = "#EEEEEE", colormap = c("bwr", "jet", "gbr", "wyr",
"br",
"yr", "rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
colorbar.fraction = 0.25, glayout = layout.fruchterman.reingold,
glayout.dynamics = F, mtext.side = 3, mtext.adj = 0, mtext.cex = 1,
mtext.font = 2, mtext.col = "black", ...)
```

Arguments

g
: an object of class "igraph" or "graphNEL"

data
: an input data matrix used to color-code vertices/nodes. One column corresponds to one graph node coloring. The input matrix must have row names, and these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)

filename
: the without-extension part of the name of the output file. By default, it is 'visNetAnimate'

filetype
: the type of the output file, i.e. the extension of the output file name. It can be one of either 'pdf' for the pdf file, 'mp4' for the mp4 video file, 'gif' for the gif file

image.type
: the type of the image files temporarily generated. It can be one of either 'jpg' or 'png'. These temporary image files are used for producing mp4/gif output file. The reason doing so is to accommodate that sometimes only one of image types is supported so that you can choose the right one

num.frame
: a numeric value specifying the number of frames/images. By default, it sets to the number of columns in the input data matrix

sec_per_frame
: a numeric value specifying how long (seconds) it takes to stream a frame/image. This argument only works when producing mp4 video or gif file.

height.device
: a numeric value specifying the height (or width) of device/frame/image.

margin
: margins as units of length 4 or 1

border.color
: the border color of each figure

colormap
: short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

ncolors
: the number of colors specified over the colormap

zlim
: the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted

colorbar   logical to indicate whether to append a colorbar. If pattern is null, it always sets
                     to false

colorbar.fraction
                     the relative fraction of colorbar block against the figure block

glayout   either a function or a numeric matrix configuring how the vertices will be placed
                     on the plot. If layout is a function, this function will be called with the graph
                     as the single parameter to determine the actual coordinates. This function can
                     be one of "layout.auto", "layout.random", "layout.circle", "layout.sphere", "lay-
                     out.fruchterman.reingold", "layout.kamada.kawai", "layout.spring", "layout.reingold.tilford",
                     "layout.fruchterman.reingold.grid", "layout.lgl", "layout.graphopt", "layout.svd"
                     and "layout.norm". A full explanation of these layouts can be found in http:
                     //igraph.org/r/doc/layout_nicely.html

glayout.dynamics
                     logical to indicate whether graph layout should be dynamic. By default, it al-
                     ways sets to false. If YES, the Fruchterman-Reingold layout algorithm http:
                     //igraph.org/r/doc/layout_with_fr.html will be used to stimulate the dy-
                     namic layout

mtext.side   on which side of the mtext plot (1=bottom, 2=left, 3=top, 4=right)

mtext.adj   the adjustment for mtext alignment (0 for left or bottom alignment, 1 for right
                     or top alignment)

mtext.cex   the font size of mtext labels

mtext.font   the font weight of mtext labels

mtext.col   the color of mtext labels

...   additional graphic parameters. See http://igraph.org/r/doc/plot.common.
                     html for the complete list.

Value

If specifying the output file name (see argument 'filename' above), the output file is either 'file-
name.pdf' or 'filename.mp4' or 'filename.gif' in the current working directory. If no output file
name specified, by default the output file is either 'visNetAnimate.pdf' or 'visNetAnimate.mp4' or
'visNetAnimate.gif'

Note

When producing mp4 video, this function requires the installation of the software 'ffmpeg' at
https://www.ffmpeg.org. Shell command lines for ffmpeg installation in Terminal (for both Linux
and Mac) are:

- 1) `wget -O ffmpeg.tar.gz http://www.ffmpeg.org/releases/ffmpeg-2.7.1.tar.gz`

- 2) `mkdir ~/ffmpeg | tar xvfz ffmpeg.tar.gz -C ~/ffmpeg --strip-components=1`

- 3) `cd ffmpeg`

- 4a) # Assuming you want installation with a ROOT (sudo) privilege:
  `./configure --disable-yasm`

- 4b) # Assuming you want local installation without ROOT (sudo) privilege:
  `./configure --disable-yasm --prefix=$HOME/ffmpeg`

- 5) `make`

- 6) `make install`

- 7) # add the system PATH variable to your ~/.bash_profile file if you follow 4b) route:
  `export PATH=$HOME/ffmpeg:$PATH`

- 8) # make sure ffmpeg has been installed successfully:
  `ffmpeg -h`

When producing gif file, this function requires the installation of the software 'ImageMagick' at http://www.imagemagick.org. Shell command lines for ImageMagick installation in Terminal are:

- 1) `wget http://www.imagemagick.org/download/ImageMagick.tar.gz`

- 2) `mkdir ~/ImageMagick | tar xvzf ImageMagick.tar.gz -C ~/ImageMagick --strip-components=1`

- 3) `cd ImageMagick`

- 4) `./configure --prefix=$HOME/ImageMagick`

- 5) `make`

- 6) `make install`

- 7) # add the system PATH variable to your ~/.bash_profile file.
  For Linux:
  `export MAGICK_HOME=$HOME/ImageMagick`
  `export PATH=$MAGICK_HOME/bin:$PATH`
  `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+$LD_LIBRARY_PATH:}$MAGICK_HOME/lib`
  For Mac:
  `export MAGICK_HOME=$HOME/ImageMagick`
  `export PATH=$MAGICK_HOME/bin:$PATH`
  `export DYLD_LIBRARY_PATH=$MAGICK_HOME/lib/`

- 8a) # check configuration:
  `convert -list configure`

- 8b) # check image format supported:
  `identify -list format`

- Tips:
  Prior to 4), please make sure `libjpeg` and `libpng` are installed. If NOT, for Mac try this:
  `brew install libjpeg libpng`
  To check whether ImageMagick does work, please get additional information from:
  `identify -list format`
  `convert -list configure`
  On details, please refer to http://www.imagemagick.org/script/advanced-unix-installation.php

See Also

visNetMul

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) visualise the module with vertices being color-coded by scores
nnodes <- vcount(subg)
nsamples <- 10
```

```
data <- matrix(runif(nnodes*nsamples), nrow=nnodes, ncol=nsamples)
rownames(data) <- V(subg)$name
# output as a <a href="visNetAnimate.pdf">pdf</a> file
visNetAnimate(g=subg, data=data, filetype="pdf")
# output as a <a href="visNetAnimate.mp4">mp4</a> file but with dynamic layout
visNetAnimate(g=subg, data=data, filetype="mp4", glayout.dynamics=TRUE)
# output as a <a href="visNetAnimate.gif">gif</a> file but with dynamic layout
visNetAnimate(g=subg, data=data, filetype="gif", glayout.dynamics=TRUE)
```

---

visNetArc   *Function to visualise an igraph object via arc diagram*

Description

`visNetArc` is supposed to visualise a graph object of class "igraph" via arc diagram in one-dimensional layout. More precisely, it displays vertices (nodes) along an axis, with edges linked by arcs. With proper ordering of vertices (e.g. according to communities and degrees), arc diagram is able to identify clusters and bridges (as effective as two-dimensional layout). One advantage of using arc diagram is to allow for easy annotations along vertices.

Usage

```
visNetArc(g, orientation = c("vertical", "horizontal"), newpage = T,
ordering = NULL, labels = V(g)$name, vertex.label.color = "black",
vertex.label.cex = 1, vertex.color = "transparent",
vertex.frame.color = "black", vertex.size = log(degree(g)) + 0.1,
vertex.pch = 21, vertex.lwd = 1, edge.color = "grey", edge.width = 1,
edge.lty = 1, ...)
```

Arguments

g	an object of class "igraph"
orientation	the orientation of the plots. It can be either "vertical" (default) or "horizontal"
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
ordering	a numeric vector about the ordering of vertices. It is optional. It is highly recommend to order vertices according to communities and degrees
labels	the label of the vertices. The default vertex labels are the name attribute of the nodes
vertex.label.color	the color of vertex labels
vertex.label.cex	the font size of vertex labels
vertex.color	the fill color of the vertices. The default vertex colors are transparent
vertex.frame.color	the color of the frame of the vertices. The default vertex frame colors are black
vertex.size	the size of each vertex. By default, it is decided according to node degrees
vertex.pch	the shape of each vertex. Either an integer specifying a symbol or a single character to be used as the default in plotting points. See http://www.statmethods.net/advgraphs/parameters.html

vertex.lwd	line width for the vertices (default 1)
edge.color	the color of the edges (default "grey")
edge.width	line width for the edges (default 1)
edge.lty	line type for the edges (default 1)
...	additional graphic parameters associated with 'mtext'

Value

invisible

Note

See Also

[visNet](#)

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/80)

# 2) produce the induced subgraph only based on the nodes in query
g <- dNetInduce(g, V(g), knn=0)

# 3) color nodes according to communities identified via a spin-glass model and simulated annealing
com <- spinglass.community(g, spins=4)
vgroups <- com$membership
palette.name <- visColormap(colormap="rainbow")
vcolors <- palette.name(length(com))[vgroups]

# 4) size nodes according to degrees
vdegrees <- igraph::degree(g)

# 5) sort nodes: first by communities and then degrees
tmp <- data.frame(ind=1:vcount(g), vgroups, vdegrees)
ordering <- tmp[order(vgroups,vdegrees),]$ind

# 6) visualise graph using 1-dimensional arc diagram
visNetArc(g, ordering=ordering, labels=V(g)$name,
vertex.label.color=vcolors,
vertex.color=vcolors, vertex.frame.color=vcolors,
vertex.size=log(vdegrees)+0.1)

# 7) as comparison, also visualise graph on 2-dimensional layout
visNet(g, colormap="bwr", layout=layout.kamada.kawai(g),
vertex.label=V(g)$name,
vertex.color=vcolors, vertex.frame.color=vcolors,
vertex.shape="sphere")
```

---

visNetCircle   *Function to visualise an igraph object via circle diagram*

Description

visNetCircle is supposed to visualise a graph object of class "igraph" via circle diagram. For better visualisation, ordering of vertices is determined according to communities and degrees.

Usage

```
visNetCircle(g, com, circles = c("single", "multiple"), newpage = T,
ordering = NULL, colormap = c("rainbow", "bwr", "jet", "gbr", "wyr",
"br",
"yr", "wb"), vertex.label = V(g)$name,
vertex.size = log(igraph::degree(g)) + 2, vertex.label.color = "black",
vertex.label.cex = 0.6, vertex.label.dist = 0.75,
vertex.shape = "sphere", edge.width = 1, edge.lty = 1,
edge.color.within = "grey", edge.color.crossing = "black",
mark.shape = 1, mark.expand = 10, ...)
```

Arguments

g	an object of class "igraph"
com	an object of class "communities" (see http://igraph.org/r/doc/communities.html)
circles	how circles are drawn in the plot. It can be either "single" for all communities being drawn in a single circle (by default) or "multiple" for communities being drawn in the different circles (i.e. one circle per community)
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
ordering	a numeric vector about the ordering of vertices. It is optional. It is highly recommend to order vertices according to communities and degrees
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
vertex.label	the label of the vertices. The default vertex labels are the name attribute of the nodes
vertex.size	the size of each vertex. By default, it is decided according to node degrees
vertex.label.color	
	the color of vertex labels
vertex.label.cex	
	the font size of vertex labels

vertex.label.dist
              the distance of the label from the center of the vertex. If it is 0 then the label is
              centered on the vertex. If it is 1 then the label is displayed beside the vertex.

vertex.shape   the shape of each vertex. It can be one of "circle", "square", "csquare", "rectan-
              gle", "crectangle", "vrectangle", "pie" ([http://igraph.org/r/doc/vertex.](http://igraph.org/r/doc/vertex.shape.pie.html)
              [shape.pie.html](http://igraph.org/r/doc/vertex.shape.pie.html)), "sphere", and "none". If it sets to NULL, these vertices with
              negative will be "csquare" and the rest "circle".

edge.width   line width for the edges (default 1)

edge.lty   line type for the edges (default 1)

edge.color.within
              the color for edges within a community (default "grey")

edge.color.crossing
              the color for edges between communities (default "black")

mark.shape   a numeric scalar or vector controlling the smoothness of the vertex group mark-
              ing polygons. Its possible values are between -1 (fully polygons) and 1 (fully
              smoothness)

mark.expand   a numeric scalar or vector, the size of the border around the marked vertex
              groups

...   additional graphic parameters. See [http://igraph.org/r/doc/plot.common.](http://igraph.org/r/doc/plot.common.html)
              [html](http://igraph.org/r/doc/plot.common.html) for the complete list.

Value

     invisible

Note

See Also

     visNet

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/80)

# 2) produce the induced subgraph only based on the nodes in query
g <- dNetInduce(g, V(g), knn=0)

# 3) color nodes according to communities identified via a spin-glass model and simulated annealing
com <- spinglass.community(g, spins=4)
vgroups <- com$membership
palette.name <- visColormap(colormap="rainbow")
mcolors <- palette.name(length(com))
vcolors <- mcolors[vgroups]

# 4) size nodes according to degrees
vdegrees <- igraph::degree(g)

# 5) sort nodes: first by communities and then degrees
tmp <- data.frame(ind=1:vcount(g), vgroups, vdegrees)
```

```
ordering <- tmp[order(vgroups,vdegrees),]$ind

# 6) visualise graph using circle diagram
# 6a) drawn into a single circle
visNetCircle(g=g, colormap="bwr", com=com, ordering=ordering)

# 6b) drawn into multlpe circles (one circle per community)
visNetCircle(g=g, colormap="bwr", com=com, circles="multiple",
ordering=ordering)

# 7) as comparison, also visualise graph on 2-dimensional layout
mark.groups <- communities(com)
mark.col <- visColoralpha(mcolors, alpha=0.2)
mark.border <- visColoralpha(mcolors, alpha=0.2)
edge.color <- c("grey", "black")[crossing(com,g)+1]
visNet(g, colormap="bwr", glayout=layout.fruchterman.reingold,
vertex.color=vcolors,
vertex.frame.color=vcolors, vertex.shape="sphere",
mark.groups=mark.groups, mark.col=mark.col,
mark.border=mark.border, mark.shape=1, mark.expand=10,
edge.color=edge.color)
```

---

visNetMul	*Function to visualise the same graph but with multiple graph node colorings according to input data matrix*

Description

`visNetMul` is supposed to visualise the same graph but with multiple colorings according to input data matrix

Usage

```
visNetMul(g, data, height = 7, margin = rep(0.1, 4),
border.color = "#EEEEEE", colormap = c("bwr", "jet", "gbr", "wyr",
"br",
"yr", "rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
colorbar.fraction = 0.25, newpage = T,
glayout = layout.fruchterman.reingold, mtext.side = 3, mtext.adj = 0,
mtext.cex = 1, mtext.font = 2, mtext.col = "black", ...)
```

Arguments

g	an object of class "igraph" or "graphNEL"
data	an input data matrix used to color-code vertices/nodes. One column corresponds to one graph node coloring. The input matrix must have row names, and these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
height	a numeric value specifying the height of device

margin	margins as units of length 4 or 1
border.color	the border color of each figure
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false
colorbar.fraction	
	the relative fraction of colorbar block against the figure block
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
glayout	either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout.auto", "layout.random", "layout.circle", "layout.sphere", "layout.fruchterman.reingold", "layout.kamada.kawai", "layout.spring", "layout.reingold.tilford", "layout.fruchterman.reingold.grid", "layout.lgl", "layout.graphopt", "layout.svd" and "layout.norm". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html
mtext.side	on which side of the mtext plot (1=bottom, 2=left, 3=top, 4=right)
mtext.adj	the adjustment for mtext alignment (0 for left or bottom alignment, 1 for right or top alignment)
mtext.cex	the font size of mtext labels
mtext.font	the font weight of mtext labels
mtext.col	the color of mtext labels
...	additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

invisible

Note

See Also

visNet, visNetAnimate

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/80)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) visualise the module with vertices being color-coded by scores
nnodes <- vcount(subg)
nsamples <- 10
data <- matrix(runif(nnodes*nsamples), nrow=nnodes, ncol=nsamples)
rownames(data) <- V(subg)$name
visNetMul(g=subg, colormap="bwr", data=data,
glayout=layout.fruchterman.reingold)
```

---

visNetReorder	*Function to visualise the multiple graph colorings reorded within a sheet-shape rectangle grid*

Description

visNetReorder is supposed to visualise the multiple graph colorings reorded within a sheet-shape rectangle grid

Usage

```
visNetReorder(g, data, sReorder, height = 7, margin = rep(0.1, 4),
border.color = "#EEEEEE", colormap = c("bwr", "jet", "gbr", "wyr",
"br",
"yr", "rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
colorbar.fraction = 0.5, newpage = T,
glayout = layout.fruchterman.reingold, mtext.side = 3, mtext.adj = 0,
mtext.cex = 1, mtext.font = 2, mtext.col = "black", ...)
```

Arguments

g	an object of class "igraph" or "graphNEL"
data	an input data matrix used to color-code vertices/nodes. One column corresponds to one graph node coloring. The input matrix must have row names, and these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the pattern vector should be the same as the number of nodes of input graph. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
sReorder	an object of class "sReorder"
height	a numeric value specifying the height of device
margin	margins as units of length 4 or 1
border.color	the border color of each figure

colormap   short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-
                  white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-
                  red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb"
                  (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-
                  green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color
                  names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-
                  white-darkviolet". A list of standard color names can be found in http://
                  html-color-codes.info/color-names

ncolors   the number of colors specified over the colormap

zlim   the minimum and maximum z/patttern values for which colors should be plotted,
                  defaulting to the range of the finite values of z. Each of the given colors will be
                  used to color an equispaced interval of this range. The midpoints of the intervals
                  cover the range, so that values just outside the range will be plotted

colorbar   logical to indicate whether to append a colorbar. If pattern is null, it always sets
                  to false

colorbar.fraction
                  the relative fraction of colorbar block against the figure block

newpage   logical to indicate whether to open a new page. By default, it sets to true for
                  opening a new page

glayout   either a function or a numeric matrix configuring how the vertices will be placed
                  on the plot. If layout is a function, this function will be called with the graph
                  as the single parameter to determine the actual coordinates. This function can
                  be one of "layout.auto", "layout.random", "layout.circle", "layout.sphere", "lay-
                  out.fruchterman.reingold", "layout.kamada.kawai", "layout.spring", "layout.reingold.tilford",
                  "layout.fruchterman.reingold.grid", "layout.lgl", "layout.graphopt", "layout.svd"
                  and "layout.norm". A full explanation of these layouts can be found in http:
                  //igraph.org/r/doc/layout_nicely.html

mtext.side   on which side of the mtext plot (1=bottom, 2=left, 3=top, 4=right)

mtext.adj   the adjustment for mtext alignment (0 for left or bottom alignment, 1 for right
                  or top alignment)

mtext.cex   the font size of mtext labels

mtext.font   the font weight of mtext labels

mtext.col   the color of mtext labels

...   additional graphic parameters. See http://igraph.org/r/doc/plot.common.
                  html for the complete list.

Value

invisible

Note

See Also

visNet, dNetReorder

Examples

```
# 1) generate a random graph according to the ER model
g <- erdos.renyi.game(100, 1/100)

# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)

# 3) reorder the module with vertices being color-coded by input data
nnodes <- vcount(subg)
nsamples <- 10
data <- matrix(runif(nnodes*nsamples), nrow=nnodes, ncol=nsamples)
rownames(data) <- V(subg)$name
sReorder <- dNetReorder(g=subg, data, feature="node",
node.normalise="none")

# 4) visualise the module with vertices being color-coded by input data
visNetReorder(g=subg, colormap="bwr", data=data, sReorder)
```

Index