Package 'dnet'

April 7, 2014

Type Package

Title Dynamic NETworks via integrative analysis of network, expression, evolution and ontology data

Version 1.0.0

Date 2014-4-7

Author Hai Fang and Julian Gough

Maintainer Hai Fang <hfang@cs.bris.ac.uk>

Depends R (>= 3.0.2), igraph, supraHex

Imports ape, graph, Rgraphviz, Matrix

Suggests affy, limma

Description The 'dnet' package is initiated to fill in the need of an open-source tool for analysing biological networks and high-throughput biological data in an integrative manner. More specifically, dnet intends to analyse the biological network whose nodes/genes are associated with digitised information such as expression levels across samples. To help make sense of identified networks, enrichment analysis is also supported using a wide variety of pre-compiled ontologies and phylostratific age information in major organisms including human, mouse, rat, chicken, C.elegans, fruit fly, zebrafish and arabidopsis. In summary, dnet aims to deliver an eye-intuitive tool for integrative analysis of network, expression and ontology data.

URL http://dnet.r-forge.r-project.org

Collate 'dGSEA.r' 'dGSEAview.r' 'dGSEAwrite.r' 'visGSEA.r' 'dPvalAggregate.r' 'dNetInduce.r' 'dBUMfit.r' 'dBUMscore.r' 'dNetFind.r' 'dNetPipeline.r' 'dNetConfidence.r' 'visNet.r' 'visNetMul.r' 'visNetReorder.r' 'dNetReorder.r' 'visNetArc.r' 'visNetCircle.r' 'dRWR.r' 'dContrast.r' 'dCommSignif.r' 'dSVDsignif.r' 'dFDRscore.r' 'visColormap.r' 'visColoralpha.r' 'visHeatmap.r' 'visHeatmapAdv.r' 'visTreeBootstrap.r' 'dDAGinduce.r' 'dDAGreverse.r' 'dDAGroot.r' 'dDAGtip.r' 'dDAGlevel.r' 'dDAGannotate.r' 'visDAG.r' 'dEnricher.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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Datasets/CLL.RData"))
CLL
# 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-likehood 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 $\int_0^x (1-\lambda) * a * (x^{a-1} - 1)\,\mathrm{d}x$ and $\int_0^x \lambda + (1-\lambda) * a\,\mathrm{d}x$. 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": $S(x) = log_2 \frac{(1-\lambda)*a*(x^{a-1}-1)}{\lambda+(1-\lambda)*a} - log_2 \frac{(1-\lambda)*a*(\tau^{a-1}-1)}{\lambda+(1-\lambda)*a} = log_2 \big(\frac{x^{a-1}-1}{\tau^{a-1}-1}\big)$. 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 $\tau \leq \big(\frac{\lambda+2*a*(1-\lambda)}{a*(1-\lambda)}\big)^{\frac{1}{a-1}}$
- The method "cdf": $S(x) = log_2 \frac{\int_0^x (1-\lambda)*a*(x^{a-1}-1)\,\mathrm{d}x}{\int_0^x \lambda+(1-\lambda)*a\,\mathrm{d}x} - log_2 \frac{\int_0^\tau (1-\lambda)*a*(\tau^{a-1}-1)\,\mathrm{d}x}{\int_0^\tau \lambda+(1-\lambda)*a\,\mathrm{d}x} = log_2 \frac{(1-\lambda)*(x^{a-1}-a)}{\lambda+(1-\lambda)*a} - log_2 \frac{(1-\lambda)*(\tau^{a-1}-a)}{\lambda+(1-\lambda)*a} = log_2 \big(\frac{x^{a-1}-a}{\tau^{a-1}-a}\big)$. 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 $\tau \leq \big(\frac{\lambda+2*a*(1-\lambda)}{1-\lambda}\big)^{\frac{1}{a-1}}$
- Where $\tau = \big[\frac{\lambda+(1-\lambda)*a-fdr*\lambda}{fdr*(1-\lambda)}\big]^{\frac{1}{a-1}}$, i.e. the p-value corresponding to the exact fdr threshold. It can be duduced from the definition of the false discovery rate: $fdr \doteq \frac{\int_0^\tau \lambda+(1-\lambda)*a\,\mathrm{d}x}{\int_0^\tau \lambda+(1-\lambda)*a*x^{a-1}\,\mathrm{d}x}$. 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)
```

---

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.sourceforge.net/doc/R/communities.html

Value

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

Note

none

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

none

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")
```

`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 a new node attribute called "annotations", which contains a list of genes either as original annotations or inherited annotations

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 GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

# 2) load human genes annotated by GO Molelular Function terms
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.egGOMF.RData"))
GS <- org.Hs.egGOMF # 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")
```

---

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 GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

# 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 chilren 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 GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

# 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")
# 4b) 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

none

See Also

dDAGreverse

Examples

```
# 1) load GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

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

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

none

See Also

dDAGroot

Examples

```
# 1) load GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

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

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

none

See Also

dDAGtip

Examples

```
# 1) load GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

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

---

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", "DO",
"HPPA",
"HPMI", "HPON", "MP", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP",
"MsigdbC2KEGG",
"MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR",
"MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC",
"MsigdbC6", "MsigdbC7"), sizeRange = c(10, 1000), which_distance =
NULL,
test = c("FisherTest", "HypergeoTest", "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 = "http://dnet.r-forge.r-project.org/data")
```

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, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPON" for Human Phenotype ONset and clinical course, "MP" for Mammalian Phenotype, and the molecular signatures database (Msigdb) in human (including "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 five ("DO", "HPPA", "HPMI", "HPON" 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://dnet.r-forge.r-project.org/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
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 `"http://dnet.r-forge.r-project.org/data"`. 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 remote download each time). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget - r - l2 - A"*.RData" - np - nH - -cut - dirs = 0"http://dnet.r - forge.r - project.org/data"$

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
- `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:

- "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 signficance 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

dEnricher

Examples

```
#\dontrun{
load(url("http://dnet.r-forge.r-project.org/data/Datasets/Hiratani_TableS1.RData"))
data <- rownames(RT)[1:1000]
eTerm <- dEnricher(data, identity="symbol", genome="Mm", ontology="MP",
RData.location="./RData_Rd")
eTerm <- dEnricher(data, identity="symbol", genome="Mm", ontology="MP",
ontology.algorithm="pc", RData.location="./RData_Rd")
eTerm <- dEnricher(data, identity="symbol", genome="Mm", ontology="MP",
ontology.algorithm="elim", RData.location="./RData_Rd")
eTerm <- dEnricher(data, identity="symbol", genome="Mm", ontology="MP",
ontology.algorithm="lea", RData.location="./RData_Rd")
cbind(eTerm$set_info[which(eTerm$pvalue < 1e-3), c(1,2)],
eTerm$pvalue[which(eTerm$pvalue < 1e-3)])

# highlight the top significant terms and also color-code all terms according to the adjust p-values
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.MP.RData"))
g <- ig.MP
nodes_query <- names(sort(eTerm$adjp)[1:5])
nodes.highlight <- rep("red", length(nodes_query))
names(nodes.highlight) <- nodes_query
subg <- dDAGinduce(g, nodes_query)
visDAG(g=subg, data=-1*log10(eTerm$adjp[V(subg)$name]),
node.info="both", zlim=c(0,2), node.attrs=list(color=nodes.highlight))
#}
```

---

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 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

none

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)
```

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", "DO", "HPPA", "HPMI",
"HPON",
"MP", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CP", "MsigdbC2KEGG",
"MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR",
"MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC",
"MsigdbC6", "MsigdbC7"), sizeRange = c(10, 1000), which_distance =
NULL,
weight = 1, nperm = 100, fast = T, sigTail = c("two-tails",
"one-tail"), p.adjust.method = c("BH", "BY", "bonferroni", "holm",
"hochberg", "hommel"), verbose = T,
RData.location = "http://dnet.r-forge.r-project.org/data")
```

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 Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PS" for phylostratific age information, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPON" for Human Phenotype ONset and clinical course, "MP" for Mammalian Phenotype, and the molecular signatures database (Msigdb) in human (including "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 five ("DO", "HPPA", "HPMI", "HPON" 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://dnet.r-forge.r-project.org/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

`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 weigth. 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 associations 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 "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

`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 "http://dnet.r-forge.r-project.org/data". 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 remote download each time). Here is the UNIX command for downloading all RData files (preserving the directory structure): $wget - r - l2 - A"*.RData" - np - nH - -cut - dirs = 0"http://dnet.r - forge.r - project.org/data"$

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 samples
- 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
- 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

```
#\dontrun{
load(url("http://dnet.r-forge.r-project.org/data/Datasets/Hiratani_TableS1.RData"))
data <- RT[1:1000,1:2]
eTerm <- dGSEA(data, identity="symbol", genome="Mm", ontology="MP",
```

```
which_distance=c(1,2))
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")
#}
```

---

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"), 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
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

none

See Also

dGSEA

Examples

```
## Not run:
dGSEAview(eTerm, which_sample=1, top_num=10, sortBy="adjp",
decreasing=FALSE, details=TRUE)

## End(Not run)
```

---

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 enrichment 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

```
## Not run:
output <- dGSEAwrite(eTerm, which_content="gadjp", which_score="gadjp",
filename="eTerm.txt")

## End(Not run)
```

---

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 module*

Description

dNetFind is supposed to find the maximum scoring module from an input graph and scores imposed on its nodes. The input graph and the output module 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 module 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 module with a maximum score, an object of class "igraph" or "graphNEL"

Note

The search procedure is heuristic to find the module 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 among 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 amongest all possible paths. Nodes along this optimal path plus their attached meta-nodes are called 'module nodes'.
- viii) finally, from the input graph extract a subgraph (called 'module') that only contains module nodes and edges betwen them. This module is the maximum scoring module 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 module with the maximum score
module <- 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)
```

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

Value

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

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 module from an input graph and the signficance imposed on its nodes*

Description

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

Usage

```
dNetPipeline(g, pval, method = c("pdf", "cdf", "fdr"), fdr = NULL,
nsize = NULL, plot = F, verbose = T)
```

Arguments

g	an object of class "igraph" or "graphNEL"
pval	a vector containing input p-values. 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

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
nsize	the desired number of nodes constrained to the resulting module. It is not nulll, a wide range of FDR will be scanned to find the FDR threshold leading to the desired number of nodes in the resulting module. Notably, the given FDR threshold will be overwritten.
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 module with a maximum score, an object of class "igraph" or "graphNEL"

Note

The pipeline sequentially consists of:

- i) dBUMfit used to fit the p-value distribution under beta-uniform mixture model.
- ii) if there is the desired number of nodes constrained to the resulting module, a wide range of FDR (including rough stage with large intervals, and finetune stage with smaller intervals) will be scanned to find the FDR threshold to meet the desired number of nodes.
- iii) dBUMscore used to calculate the scores according to the fitted BUM and FDR threshold.
- iv) dNetFind used to find maximum-scoring module from the input graph and scores imposed on its nodes.

See Also

dBUMfit, dBUMscore, 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 module based on fdr=0.1 threshold
module <- dNetPipeline(g=subg, pval=x, fdr=0.1)

## Not run:
# 5) find maximum-scoring module with the desired node number nsize=20
# module <- dNetPipeline(g=subg, pval=x, nsize=20)

## End(Not run)
```

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.
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 rectanges 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). Then, 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.

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.

Usage

```
dPvalAggregate(pmatrix, method = c("orderStatistic", "fishers"),
order = 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
order	an integeter specifying the order used for the aggregation according to on the order statistics of p-values

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 the Fisher's method, after transformation by $-2 * \sum^{c} log(pvalue)$, they follow Chi-Squared distribution.

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 ordre statistics
ap <- dPvalAggregate(pmatrix, method="orderStatistic")

# 3) aggregate according to the Fishers method
ap <- dPvalAggregate(pmatrix, method="fishers")
```

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 give, 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.

Usage

```
dRWR(g, normalise = c("laplacian", "row", "column", "none"),
setSeeds = NULL, restart = 0.75, normalise.affinity.matrix = c("none",
"quantile"), 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 normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

When the seeds are NOT given, it returns:

- `PTmatrix`: pre-computated 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, it returns:

- `PTmatrix`: 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

dNetInduce

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(subg, normalise="laplacian", restart=0.75)
# 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(subg, normalise="laplacian", setSeeds=setSeeds,
restart=0.75)
PTmatrix
```

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

none

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)
## Not run:
# using "pval" significance
pval <- dSVDsignif(data, signif="pval", num.permutation=10)

## End(Not run)
```

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/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/data/Datasets/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.sourceforge.net/doc/R/aaa-igraph-package.html). This data is prepared based on http://sourceforge.net/p/diseaseontology/code/HEAD/tree/trunk/HumanDO.obo.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.DO.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.DO.RData"))
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.sourceforge.net/doc/R/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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOBP.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOBP.RData"))
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.sourceforge.net/doc/R/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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOCC.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOCC.RData"))
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.sourceforge.net/doc/R/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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
ig.GOMF
```

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.sourceforge.net/doc/R/aaa-igraph-package.html). This data is prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPMI.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPMI.RData"))
ig.HPMI
```

ig.HPON — *Human Phenotype ONset and clinical course (HPON).*

Description

An R object that contains information on Human Phenotype ONset and clinical course 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.sourceforge.net/doc/R/aaa-igraph-package.html). This data is prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPON.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPON.RData"))
ig.HPON
```

---

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.sourceforge.net/doc/R/aaa-igraph-package.html). This data is prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPPA.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.HPPA.RData"))
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.sourceforge.net/doc/R/aaa-igraph-package.html). This data is prepared based on http://sourceforge.net/p/diseaseontology/code/HEAD/tree/trunk/HumanMP.obo.

Usage

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.MP.RData"))
```

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

```
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.MP.RData"))
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

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

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

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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/At/org.At.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/At/org.At.egPS.RData"))
names(org.At.egPS)
```

---

org.At.string *Arabidopsis functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in arabidopsis. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Ce/org.Ce.eg.RData"))
names(org.Ce.eg)
org.Ce.eg$gene_info[1:5,]
```

`org.Ce.egGOBP` *Annotations of C.elegans Entrez Genes (EG) by Gene Ontology Biological 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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Ce/org.Ce.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Ce/org.Ce.egPS.RData"))
names(org.Ce.egPS)
```

`org.Ce.string` *C.elegans functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in c.elegans. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Da/org.Da.egGOBP.RData"))
names(org.Da.egGOBP)
```

---

org.Da.egGOCC — *Annotations of Zebrafish Entrez Genes (EG) by Gene Ontology Cellular Component (GOCC).*

Description

An R object that contains associations between Gene Ontology Cellular Component 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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Da/org.Da.egGOCC.RData"))
names(org.Da.egGOCC)
```

`org.Da.egGOMF` *Annotations of Zebrafish Entrez Genes (EG) by Gene Ontology Molecular Function (GOMF).*

Description

An R object that contains associations between Gene Ontology Molecular Function 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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Da/org.Da.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Da/org.Da.egPS.RData"))
names(org.Da.egPS)
```

`org.Da.string` *Zebrafish functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in zebrafish. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Dm/org.Dm.eg.RData"))
names(org.Dm.eg)
org.Dm.eg$gene_info[1:5,]
```

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

Description

An R object that contains associations between Gene Ontology Biological Process 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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Dm/org.Dm.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Dm/org.Dm.egPS.RData"))
names(org.Dm.egPS)
```

`org.Dm.string` *Fruitfly functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in fruitfly. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Gg/org.Gg.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Gg/org.Gg.egPS.RData"))
names(org.Gg.egPS)
```

`org.Gg.string` *Chicken functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in chicken. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.eg.RData"))
names(org.Hs.eg)
org.Hs.eg$gene_info[1:5,]
```

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 and http://dga.nubic.northwestern.edu/ajax/Download.ajax.php, which results in annotations of Human Entrez Genes.

Usage

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

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

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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.egGOMF.RData"))
names(org.Hs.egGOMF)
```

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://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.egHPMI.RData"))
names(org.Hs.egHPMI)
```

---

org.Hs.egHPON — *Annotations of Human Entrez Genes (EG) by Human Phenotype ONset and clinical course (HPON).*

Description

An R object that contains associations between HPON terms and Human Entrez Genes. This data is first prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

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

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. HPON 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

```
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.egHPON.RData"))
names(org.Hs.egHPON)
```

org.Hs.egHPPA *Annotations of Human Entrez Genes (EG) by Human Phenotype Phenotypic Abnormality (HPPA).*

Description

An R object that contains associations between Human Phenotype Phenotypic Abnormality terms and Human Entrez Genes. This data is first prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.obo and http://compbio.charite.de/hudson/job/hpo.annotations.monthly/lastStableBuild/artifact/annotation/ALL_SOURCES_ALL_FREQUENCIES_genes_to_phenotype.txt.

Usage

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC1.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC1.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2BIOCARTA.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2BIOCARTA.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2CGP.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2CGP.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2CP.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2CP.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2KEGG.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2KEGG.RData"))
names(org.Hs.egMsigdbC2KEGG)
```

org.Hs.egMsigdbC2REACTOME *Annotations of Human Entrez Genes (EG) by C2:REACTOME 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: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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2REACTOME.RData"))
```

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" (always empty)
- 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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC2REACTOME.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC3MIR.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC3MIR.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC3TFT.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC3TFT.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC4CGN.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC4CGN.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC4CM.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC4CM.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5BP.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5BP.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5CC.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5CC.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5MF.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC5MF.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC6.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC6.RData"))
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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC7.RData"))
```

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" (always empty)
- `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

```
load(url("http://dnet.r-forge.r-project.org/data/Msigdb/org.Hs.egMsigdbC7.RData"))
names(org.Hs.egMsigdbC7)
```

`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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Hs/org.Hs.egPS.RData"))
names(org.Hs.egPS)
```

`org.Hs.string` *Human functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in human. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Mm/org.Mm.egGOMF.RData"))
names(org.Mm.egGOMF)
```

`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://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Mm/org.Mm.egHPMI.RData"))
names(org.Mm.egHPMI)
```

---

org.Mm.egHPON *Annotations of Mouse Entrez Genes (EG) by Human Phenotype ONset and clinical course (HPON).*

Description

An R object that contains associations between HPON terms and Mouse Entrez Genes. This data is first prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.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

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

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. HPON 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

```
load(url("http://dnet.r-forge.r-project.org/data/Mm/org.Mm.egHPON.RData"))
names(org.Mm.egHPON)
```

`org.Mm.egHPPA` *Annotations of Mouse Entrez Genes (EG) by Human Phenotype Phenotypic Abnormality (HPPA).*

Description

An R object that contains associations between Human Phenotype Phenotypic Abnormality terms and Mouse Entrez Genes. This data is first prepared based on http://compbio.charite.de/svn/hpo/trunk/src/ontology/human-phenotype-ontology.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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Mm/org.Mm.egMP.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Mm/org.Mm.egPS.RData"))
names(org.Mm.egPS)
```

---

org.Mm.string — *Mouse functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in mouse. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

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

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

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

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

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

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

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

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

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

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

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

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

```
load(url("http://dnet.r-forge.r-project.org/data/Rn/org.Rn.egGOMF.RData"))
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 providing 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

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

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 all. (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

```
load(url("http://dnet.r-forge.r-project.org/data/Rn/org.Rn.egPS.RData"))
names(org.Rn.egPS)
```

---

org.Rn.string *Rat functional protein association network from STRING (version 9.1).*

Description

An igraph object that contains a functional protein association network in rat. The network is extracted from the STRING database (version 9.1). Only those associations with medium confidence (score>=0.4) are retained.

Usage

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

Value

an object of class "igraph" (see http://igraph.sourceforge.net/doc/R/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

Franceschini et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808-D815.

Examples

```
load(url("http://dnet.r-forge.r-project.org/data/Rn/org.Rn.string.RData"))
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 3130 cancer samples (i.e. only those having clinical 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 binary (1, 0) states on genes, where '1' indicates a gene for which mutation has occurred in the tumor relative to germ line. The dataset is provided as an 'ExpressionSet' object.

Usage

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

Value

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

- `assayData`: a matrix of 19477 genes X 3130 samples
- `phenoData`: variables describing sample phenotypes (i.e. columns in assayData), including information about samples: "Age", "Days_to_death", "Days_to_last_followup", "Vita_status", "Gender", "Date_diagnosis", "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

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

`visColoralpha` *Function to add transparent (alpha) into colors*

Description

`visColoralpha` is supposed to add transparent (alpha) into colors.

Usage

```
visColoralpha(col, alpha)
```

Arguments

`col`	input colors. It can be vector of R color specifications, such as a color name (as listed by 'colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa"
`alpha`	numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque)

Value

a vector of colors (after transparent being added)

Note

none

See Also

`visColormap`

Examples

```
# 1) define "blue-white-red" colormap
palette.name <- visColormap(colormap="bwr")

# 2) use the return function "palette.name" to generate 10 colors spanning "bwr"
col <- palette.name(10)

# 3) add transparent (alpha=0.5)
cols <- visColoralpha(col, alpha=0.5)
```

visColormap *Function to define a colormap*

Description

visColormap is supposed to define a colormap. It returns a function, which will take an integer argument specifying how many colors interpolate the given colormap.

Usage

```
visColormap(colormap = c("bwr", "jet", "gbr", "wyr", "br", "yr",
"rainbow",
"wb"))
```

Arguments

colormap	short name for the colormap

Value

- palette.name: a function that takes an integer argument for generating that number of colors interpolating the given sequence

Note

The input colormap includes:

- "jet": jet colormap
- "bwr": blue-white-red
- "gbr": green-black-red
- "wyr": white-yellow-red
- "br": black-red
- "yr": yellow-red
- "wb": white-black
- "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", "darkblue-lightblue-lightyellow-darkorange", "darkgreen-white-darkviolet", "darkgreen-lightgreen-lightpink-darkred". A list of standard color names can be found in http://html-color-codes.info/color-names

See Also

visColoralpha

Examples

```
# 1) define "blue-white-red" colormap
palette.name <- visColormap(colormap="bwr")

# 2) use the return function "palette.name" to generate 10 colors spanning "bwr"
palette.name(10)
```

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("lightyellow-orange", "yr", "bwr", "jet", "gbr", "wyr",
"br",
"rainbow", "wb"), 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"), 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

`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 GO Molelular Function as igraph object
load(url("http://dnet.r-forge.r-project.org/data/Obo/ig.GOMF.RData"))
g <- ig.GOMF

# 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", weight = 1,
orientation = c("vertical", "horizontal"), 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
weight	type of score weigth. 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
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

Value

invisible

Note

none

See Also

dGSEA, dGSEAview

Examples

```
## Not run:
visGSEA(eTerm, which_sample=1, which_term=1)

## End(Not run)
```

visHeatmap *Function to visualise input data matrix using heatmap*

Description

visHeatmap is supposed to visualise input data matrix using heatmap. Note: this heatmap displays matrix in a bottom-to-top direction

Usage

```
visHeatmap(data, scale = c("none", "row", "column"), row.metric =
c("none",
"pearson", "spearman", "kendall", "euclidean", "manhattan", "cos",
"mi"),
row.method = c("ward", "single", "complete", "average", "mcquitty",
"median", "centroid"), column.metric = c("none", "pearson", "spearman",
"kendall", "euclidean", "manhattan", "cos", "mi"), column.method =
c("ward",
"single", "complete", "average", "mcquitty", "median", "centroid"),
colormap = c("bwr", "jet", "gbr", "wyr", "br", "yr", "rainbow", "wb"),
ncolors = 64, zlim = NULL, row.cutree = NULL,
row.colormap = c("rainbow"), column.cutree = NULL,
column.colormap = c("rainbow"), ...)
```

Arguments

data	an input gene-sample data matrix used for heatmap
scale	a character indicating when the input matrix should be centered and scaled. It can be one of "none" (no scaling), "row" (being scaled in the row direction), "column" (being scaled in the column direction)
row.metric	distance metric used to calculate the distance metric between rows. It can be one of "none" (i.e. no dendrogram between rows), "pearson", "spearman", "kendall", "euclidean", "manhattan", "cos" and "mi". See details at http://suprahex.r-forge.r-project.org/sDistance.html
row.method	the agglomeration method used to cluster rows. This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". See 'Note' below for details
column.metric	distance metric used to calculate the distance metric between columns. It can be one of "none" (i.e. no dendrogram between rows), "pearson", "spearman", "kendall", "euclidean", "manhattan", "cos" and "mi". See details at http://suprahex.r-forge.r-project.org/sDistance.html
column.method	the agglomeration method used to cluster columns. This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". See 'Note' below for details
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
row.cutree	an integer scalar specifying the desired number of groups being cut from the row dendrogram. Note, this optional is only enabled when the row dengrogram is built
row.colormap	short name for the colormap to color-code the row groups (i.e. sidebar colors used to annotate the rows)
column.cutree	an integer scalar specifying the desired number of groups being cut from the column dendrogram. Note, this optional is only enabled when the column dengrogram is built
column.colormap	short name for the colormap to color-code the column groups (i.e. sidebar colors used to annotate the columns)
...	additional graphic parameters. Type ?heatmap for the complete list.

Value

invisible

Note

The clustering methods are provided:

- "ward": Ward's minimum variance method aims at finding compact, spherical clusters
- "single": The single linkage method (which is closely related to the minimal spanning tree) adopts a 'friends of friends' clustering strategy
- "complete": The complete linkage method finds similar clusters
- "average","mcquitty","median","centroid": These methods can be regarded as aiming for clusters with characteristics somewhere between the single and complete link methods. Two methods "median" and "centroid" are not leading to a monotone distance measure, or equivalently the resulting dendrograms can have so called inversions (which are hard to interpret)

See Also

visHeatmap

Examples

```
# 1) generate data with an iid matrix of 100 x 9
data <- cbind(matrix(rnorm(100*3,mean=0,sd=1), nrow=100, ncol=3),
matrix(rnorm(100*3,mean=0.5,sd=1), nrow=100, ncol=3),
matrix(rnorm(100*3,mean=-0.5,sd=1), nrow=100, ncol=3))
colnames(data) <- c("S1","S1","S1","S2","S2","S2","S3","S3","S3")

# 2) prepare colors for the column sidebar
```

```
lvs <- unique(colnames(data))
lvs_color <- visColormap(colormap="rainbow")(length(lvs))
my_ColSideColors <- sapply(colnames(data), function(x)
lvs_color[x==lvs])

# 3) heatmap with row dendrogram (with 10 color-coded groups)
visHeatmap(data, row.metric="euclidean", row.method="average",
colormap="gbr", zlim=c(-2,2),
ColSideColors=my_ColSideColors, row.cutree=10, row.colormap="jet",
labRow=NA)
```

`visHeatmapAdv` *Function to visualise input data matrix using advanced heatmap*

Description

`visHeatmapAdv` is supposed to visualise input data matrix using advanced heatmap. It allows for adding multiple sidecolors in both columns and rows. Besides, the sidecolor can be automatically added via cutting histogram into groups. Note: this heatmap displays matrix in a top-to-bottom direction

Usage

```
visHeatmapAdv(data, scale = c("none", "row", "column"), Rowv = T,
Colv = T, dendrogram = c("both", "row", "column", "none"),
dist.metric = c("euclidean", "pearson", "spearman", "kendall",
"manhattan",
"cos", "mi"), linkage.method = c("complete", "ward", "single",
"average",
"mcquitty", "median", "centroid"), colormap = c("bwr", "jet", "gbr",
"wyr",
"br", "yr", "rainbow", "wb"), ncolors = 64, zlim = NULL,
RowSideColors = NULL, row.cutree = NULL, row.colormap = c("jet"),
ColSideColors = NULL, column.cutree = NULL, column.colormap = c("jet"),
...)
```

Arguments

`data`	an input gene-sample data matrix used for heatmap
`scale`	a character indicating when the input matrix should be centered and scaled. It can be one of "none" (no scaling), "row" (being scaled in the row direction), "column" (being scaled in the column direction)
`Rowv`	determines if and how the row dendrogram should be reordered. By default, it is TRUE, which implies dendrogram is computed and reordered based on row means. If NULL or FALSE, then no dendrogram is computed and no reordering is done. If a dendrogram, then it is used "as-is", ie without any reordering. If a vector of integers, then dendrogram is computed and reordered based on the order of the vector
`Colv`	determines if and how the column dendrogram should be reordered. Has the options as the Rowv argument above and additionally when x is a square matrix, Colv = "Rowv" means that columns should be treated identically to the rows

`dendrogram` character string indicating whether to draw 'none', 'row', 'column' or 'both' dendrograms. Defaults to 'both'. However, if Rowv (or Colv) is FALSE or NULL and dendrogram is 'both', then a warning is issued and Rowv (or Colv) arguments are honoured

`dist.metric` distance metric used to calculate the distance metric between columns (or rows). It can be one of "none" (i.e. no dendrogram between rows), "pearson", "spearman", "kendall", "euclidean", "manhattan", "cos" and "mi". See details at http://suprahex.r-forge.r-project.org/sDistance.html

`linkage.method` the agglomeration method used to cluster/linkages columns (or rows). This should be one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". See 'Note' below for details

`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

`RowSideColors` NULL or a matrix of "numRowsidebars" X nrow(x), where "numRowsidebars" stands for the number of sidebars annotating rows of x. This matrix contains the color names for vertical sidebars. By default, it sets to NULL. In this case, sidebars in rows can still be enabled by cutting the row dendrogram into several clusters (see the next two parameters)

`row.cutree` an integer scalar specifying the desired number of groups being cut from the row dendrogram. Note, this optional is only enabled when the ColSideColors is NULL

`row.colormap` short name for the colormap to color-code the row groups (i.e. sidebar colors used to annotate the rows)

`ColSideColors` NULL or a matrix of ncol(x) X "numColsidebars", where "numColsidebars" stands for the number of sidebars annotating the columns of x. This matrix contains the color names for horizontal sidebars. By default, it sets to NULL. In this case, sidebars in columns can still be enabled by cutting the column dendrogram into several clusters (see the next two parameters)

`column.cutree` an integer scalar specifying the desired number of groups being cut from the column dendrogram. Note, this optional is only enabled when the column dengrogram is built

`column.colormap` short name for the colormap to color-code the column groups (i.e. sidebar colors used to annotate the columns)

`...` additional graphic parameters. For most parameters, please refer to http://www.inside-r.org/packages/cran/gplots/docs/heatmap.2. For example, the parameters "srtRow" and "srtCol" to control the angle of row/column labels (in degrees from horizontal: 45 degrees for the column, 0 degrees for the row,

by default), i.e. string rotation. The parameters "offsetRow" and "offsetCol" to indicate the number of character-width spaces to place between row/column labels and the edge of the plotting region. Unique to this function, there are two parameters "RowSideWidth" and RowSideLabelLocation, to respectively indicate the fraction of the row side width and the location (either bottom or top) of the row side labelling; the other two parameters "ColSideHeight" and "ColSideLabelLocation" for the column side height and the location (either left or right) of the column side labelling; and two parameters "RowSideBox" and "ColSideBox" to indicate whether there are boxes outside.

Value

invisible

Note

The clustering/linkage methods are provided:

- "ward": Ward's minimum variance method aims at finding compact, spherical clusters
- "single": The single linkage method (which is closely related to the minimal spanning tree) adopts a 'friends of friends' clustering strategy
- "complete": The complete linkage method finds similar clusters
- "average","mcquitty","median","centroid": These methods can be regarded as aiming for clusters with characteristics somewhere between the single and complete link methods. Two methods "median" and "centroid" are not leading to a monotone distance measure, or equivalently the resulting dendrograms can have so called inversions (which are hard to interpret)

See Also

visHeatmapAdv

Examples

```
# 1) generate data with an iid matrix of 100 x 9
data <- cbind(matrix(rnorm(100*3,mean=0,sd=1), nrow=100, ncol=3),
matrix(rnorm(100*3,mean=0.5,sd=1), nrow=100, ncol=3),
matrix(rnorm(100*3,mean=-0.5,sd=1), nrow=100, ncol=3))
colnames(data) <-
c("S1_R1","S1_R2","S1_R3","S2_R1","S2_R2","S2_R3","S3_R1","S3_R2","S3_R3")

# 2) heatmap after clustering both rows and columns
# 2a) shown with row and column dendrograms
visHeatmapAdv(data, dendrogram="both", colormap="gbr", zlim=c(-2,2),
KeyValueName="log2(Ratio)",
add.expr=abline(v=(1:(ncol(data)+1))-0.5,col="white"),
lmat=rbind(c(4,3), c(2,1)), lhei=c(1,5), lwid=c(1,3))
# 2b) shown with row dendrogram only
visHeatmapAdv(data, dendrogram="row", colormap="gbr", zlim=c(-2,2))
# 2c) shown with column dendrogram only
visHeatmapAdv(data, dendrogram="column", colormap="gbr", zlim=c(-2,2))

# 3) heatmap after only clustering rows (with 2 color-coded groups)
visHeatmapAdv(data, Colv=FALSE, colormap="gbr", zlim=c(-2,2),
row.cutree=2, row.colormap="jet", labRow=NA)
```

```
# 4) prepare colors for the column sidebar
# color for stages (S1-S3)
stages <- sub("_.*","",colnames(data))
lvs <- unique(stages)
lvs_color <- visColormap(colormap="rainbow")(length(lvs))
col_stages <- sapply(stages, function(x) lvs_color[x==lvs])
# color for replicates (R1-R3)
replicates <- sub(".*_","",colnames(data))
lvs <- unique(replicates)
lvs_color <- visColormap(colormap="rainbow")(length(lvs))
col_replicates <- sapply(replicates, function(x) lvs_color[x==lvs])
# combine both color vectors
ColSideColors <- cbind(col_stages,col_replicates)
colnames(ColSideColors) <- c("Stages","Replicates")

# 5) heatmap without clustering on rows and columns but with the two sidebars in columns
visHeatmapAdv(data, Rowv=FALSE, Colv=FALSE, colormap="gbr",
zlim=c(-2,2),
density.info="density", tracecol="yellow", ColSideColors=ColSideColors,
ColSideHeight=0.5, ColSideLabelLocation="right")
```

---

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", ...)
```

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.sourceforge.net/doc/R/layout.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.sourceforge.net/doc/R/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.

`...` additional graphic parameters. See http://igraph.sourceforge.net/doc/R/plot.graph.html for the complete list.

Value

invisible

Note

none

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")
```

---

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

none

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.sourceforge.net/doc/R/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", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.sourceforge.net/doc/R/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 marking 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.sourceforge.net/doc/R/plot.graph.html for the complete list.

Value

invisible

Note

none

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,
vertex.label=V(g)$name)

# 6b) drawn into multlpe circles (one circle per community)
visNetCircle(g=g, colormap="bwr", com=com, circles="multiple",
ordering=ordering,
vertex.label=V(g)$name)

# 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.sourceforge.net/doc/R/layout.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.sourceforge.net/doc/R/plot.graph.html for the complete list.

Value

invisible

Note

none

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
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 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
sReorder	an object of class "sReorder"
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.sourceforge.net/doc/R/layout.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.sourceforge.net/doc/R/plot.graph.html for the complete list.

Value

invisible

Note

none

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)
```

---

visTreeBootstrap *Function to build and visualise the bootstrapped tree*

Description

visTreeBootstrap is supposed to build the tree, perform bootstrap analysis and visualise the bootstrapped tree. It returns an object of class "phylo". For easy downstream analysis, the bootstrapped tree is rerooted either at the internal node with the miminum bootstrap/confidence value or at any customised internal node.

Usage

```
visTreeBootstrap(data, algorithm = c("nj", "fastme.ols", "fastme.bal"),
metric = c("euclidean", "pearson", "spearman", "cos", "manhattan",
"kendall", "mi"), num.bootstrap = 100, consensus = FALSE,
consensus.majority = 0.5, reroot = "min.bootstrap",
plot.phylo.arg = NULL, nodelabels.arg = NULL, visTree = T,
verbose = T, ...)
```

Arguments

data	an input data matrix used to build the tree. The built tree describes the relationships between rows of input matrix
algorithm	the tree-building algorithm. It can be one of "nj" for the neighbor-joining tree estimation, "fastme.ols" for the minimum evolution algorithm with ordinary least-squares (OLS) fitting of a metric to a tree structure, and "fastme.bal" for the minimum evolution algorithm under a balanced (BAL) weighting scheme

`metric`	distance metric used to calculate a distance matrix between rows of input matrix. It 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
`num.bootstrap`	an integer specifying the number of bootstrap replicates
`consensus`	logical to indicate whether to return the consensus tree. By default, it sets to false for not doing so. Note: if true, there will be no visualisation of the bootstrapped tree
`consensus.majority`	a numeric value between 0.5 and 1 (or between 50 and 100) giving the proportion for a clade to be represented in the consensus tree
`reroot`	determines if and how the bootstrapped tree should be rerooted. By default, it is "min.bootstrap", which implies that the bootstrapped tree will be rerooted at the internal node with the miminum bootstrap/confidence value. If it is an integer between 1 and the number of internal nodes, the tree will be rerooted at the internal node with this index value
`plot.phylo.arg`	a list of main parameters used in the function "ape::plot.phylo" http://www.inside-r.org/packages/cran/ape/docs/plot.phylo. See 'Note' below for details on the parameters
`nodelabels.arg`	a list of main parameters used in the function "ape::nodelabels" http://www.inside-r.org/packages/cran/ape/docs/nodelabels. See 'Note' below for details on the parameters
`visTree`	logical to indicate whether the bootstrap tree will be visualised. By default, it sets to true for display. Note, the consensus tree can not be enabled for visualisation
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
`...`	additional "ape::plot.phylo" parameters

Value

an object of class "phylo". It can return a bootstrapped tree or a consensus tree (if enabled): When a bootstrapped tree is returned (also visualised by default), the "phylo" object has a list with following components:

- `Nnode`: the number of internal nodes
- `node.label`: the labels for internal nodes. Here, each internal node is associated with the bootstrap value
- `tip.label`: the lables for tip nodes. Tip labels come from the row names of the input matrix, but are not necessarily the same order as they appear in the input matrix
- `edge`: a two-column matrix describing the links between tree nodes (including internal and tip nodes)
- `edge.length`: a vector indicating the edge length in the 'edge'
- Note: the tree structure is indexed with 1:Ntip for tip nodes, and ($Ntip$+1):($Ntip$+$Nnode$) for internal nodes, where $Ntip$ is the number of tip nodes and $Nnode$ for the number of internal nodes. Moreover, $nrow(data) = Ntip = Nnode - 2$.

When a consensus tree is returned (no visualisation), the "phylo" object has a list with following components:

- `Nnode`: the number of internal nodes
- `tip.label`: the lables for tip nodes. Tip labels come from the row names of the input matrix, but are not necessarily the same order as they appear in the input matrix
- `edge`: a two-column matrix describing the links between tree nodes (including internal and tip nodes)

Note

A list of main parameters used in the function "ape::plot.phylo":

- "type": a character string specifying the type of phylogeny to be drawn; it must be one of "phylogram" (the default), "cladogram", "fan", "unrooted", "radial" or any unambiguous abbreviation of these
- "direction": a character string specifying the direction of the tree. Four values are possible: "rightwards" (the default), "leftwards", "upwards", and "downwards"
- "lab4ut": (= labels for unrooted trees) a character string specifying the display of tip labels for unrooted trees: either "horizontal" where all labels are horizontal (the default), or "axial" where the labels are displayed in the axis of the corresponding terminal branches. This option has an effect only if type = "unrooted"
- "edge.color": a vector of mode character giving the colours used to draw the branches of the plotted phylogeny. These are taken to be in the same order than the component edge of phy. If fewer colours are given than the length of edge, then the colours are recycled
- "edge.width": a numeric vector giving the width of the branches of the plotted phylogeny. These are taken to be in the same order than the component edge of phy. If fewer widths are given than the length of edge, then these are recycled
- "edge.lty": same than the previous argument but for line types; 1: plain, 2: dashed, 3: dotted, 4: dotdash, 5: longdash, 6: twodash
- "font": an integer specifying the type of font for the labels: 1 (plain text), 2 (bold), 3 (italic, the default), or 4 (bold italic)
- "cex": a numeric value giving the factor scaling of the tip and node labels (Character EXpansion). The default is to take the current value from the graphical parameters
- "adj": a numeric specifying the justification of the text strings of the labels: 0 (left-justification), 0.5 (centering), or 1 (right-justification). This option has no effect if type="unrooted". If NULL (the default) the value is set with respect of direction (see details)
- "srt": a numeric giving how much the labels are rotated in degrees (negative values are allowed resulting in clock-like rotation); the value has an effect respectively to the value of direction (see Examples). This option has no effect if type="unrooted"
- "no.margin": a logical. If TRUE, the margins are set to zero and the plot uses all the space of the device
- "label.offset": a numeric giving the space between the nodes and the tips of the phylogeny and their corresponding labels. This option has no effect if type="unrooted"
- "rotate.tree": for "fan", "unrooted", or "radial" trees: the rotation of the whole tree in degrees (negative values are accepted

A list of main parameters used in the function "ape::nodelabels":

- "text": a vector of mode character giving the text to be printed. By default, the labels for internal nodes (see "node.label"), that is, the bootstrap values associated with internal nodes

- "node": a vector of mode numeric giving the numbers of the nodes where the text or the symbols are to be printed. By default, indexes for internal nodes, that is, ($Ntip$+1):($Ntip$+$Nnode$), where $Ntip$ is the number of tip nodes and $Nnode$ for the number of internal nodes
- "adj": one or two numeric values specifying the horizontal and vertical, respectively, justification of the text or symbols. By default, the text is centered horizontally and vertically. If a single value is given, this alters only the horizontal position of the text
- "frame": a character string specifying the kind of frame to be printed around the text. This must be one of "rect" (the default), "circle", "none", or any unambiguous abbreviation of these
- "cex": a numeric value giving the factor scaling of the tip and node labels (Character EXpansion). The default is to take the current value from the graphical parameters
- "font": an integer specifying the type of font for the labels: 1 (plain text), 2 (bold), 3 (italic, the default), or 4 (bold italic)
- "col": a character string giving the color to be used for the text or the plotting symbols; this is eventually recycled
- "bg": a character string giving the color to be used for the background of the text frames or of the plotting symbols if it applies; this is eventually recycled. 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

See Also

visTreeBootstrap

Examples

```
# 1) generate an iid normal random matrix of 100x10
data <- matrix( rnorm(100*10,mean=0,sd=1), nrow=100, ncol=10)
colnames(data) <- paste(rep(S,10), seq(1:10), sep="")
data <- t(data)

# 2) build neighbor-joining tree with bootstrap values and visualise it by default
visTreeBootstrap(data)

# 3) only display those internal nodes with bootstrap values > 30
# 3a) generate the bootstrapped tree (without visualisation)
tree_bs <- visTreeBootstrap(data, visTree=FALSE)
# 3b) look at the bootstrap values and ordered row names of input matrix
# the bootstrap values
tree_bs$node.label
# ordered row names of input matrix
tree_bs$tip.label
# 3c) determine internal nodes that should be displayed
Ntip <- length(tree_bs$tip.label) # number of tip nodes
Nnode <- length(tree_bs$node.label) # number of internal nodes
flag <- as.numeric(tree_bs$node.label) > 30
text <- tree_bs$node.label[flag]
node <- Ntip + (1:Nnode)[flag]
visTreeBootstrap(data, nodelabels.arg=list(text=text,node=node))
```

```
# 4) obtain the consensus tree
tree_cons <- visTreeBootstrap(data, consensus=TRUE, num.bootstrap=10)
```

Index