Package 'sims'

September 13, 2014

Type Package

Title Semantic similarity measures between terms of an arbitrary ontology

Version 1.0

Date 2014-08-16

Author Jose Luis Mosquera and Alex Sanchez

Maintainer Jose Luis Mosquera <jlmosquera@gmail.comt>

Description This package allows the computation of semantic similarity measures, based on different approaches, between terms of an arbitrary ontology

License GPL-2

Depends AnnotationDbi, expm, GOstats, plyr

Imports Matrix, knitr, igraph, methods, plotrix, Rgraphviz, vegan

Suggests org.Hs.eg.db

VignetteBuilder knitr

R topics documented:

`ancestors` *Ancestors for each term of the ontology*

Description

Given the accesibility matrix, generates a list whose elements are the ancestors for each term of the ontology

Usage

```
ancestors(m)
```

Arguments

`m`	`matrix` of accessibility

Value

The resulting object is a `list` whose elements are `character` vectors with the names of the ancestors associated with each term

Author(s)

Jose Luis Mosquera

See Also

`getA`

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)

at <- ancestors(A.mat)
print(at)
```

commonAncestors *Common ancestors for each pair of terms of the ontology*

Description

Generates a list whose elements are the common ancestors for each pair of terms of the ontology

Usage

```
commonAncestors(at)
```

Arguments

at	list of character vectors containing the name of the ancestors of each term of the ontology

Value

Resulting object is a list whose elements are character vectors with the names of the common ancestors between each pair of terms connected

Author(s)

Jose Luis Mosquera

See Also

ancestors, getA, inverseIminusG

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

ca <- commonAncestors(at)
print(at)
```

cosSim *Cosine similarity measure*

Description

This function computes the cosine similarity measure between two columns of a `matrix`

Usage

```
cosSim(x, na.rm = FALSE)
```

Arguments

`x`	`matrix` with two columns
`na.rm`	a `logical` value indicating whether `NA` values should be stripped before the computation proceeds. By default is `FALSE`

Details

Given the two columns of the `matrix`, let A and B be the names of such columns, the cosine similarity, is estimated using a dot product and magnitude as

$$sim_{cos}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Author(s)

Jose Luis Mosquera

Examples

```
a <- sample(c(1:100, NA), 20)
idx.na <- sample(1:20, 4)
a[idx.na] <- NA
b <- sample(1:100, 20)
x <- as.matrix(cbind(a,b))

cos.sim <- cosSim(x, na.rm = TRUE)
print(cos.sim)

## Cosine similarity between semantic similarites of Resnik

data(prostateIds)               # Data set from the package goProfiles

pckg <- "org.Hs.eg.db"          # Organism package of humans

eg.we <- welsh01EntrezIDs[1:10]   # Entrez Gene IDs signature Welsh 01
eg.sg <- singh01EntrezIDs[1:10]   # Entrez Gene IDs signature Singh 01
```

```
WEvsSG.nb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg, ontology = "MF", pckg = pckg, type = "nb", method = "Res")

idx.Inf <- which(WEvsSG.nb == Inf)
WEvsSG.nb[idx.Inf] <-NA

WEvsSG.cos.sim <- cosSim(WEvsSG.nb , na.rm = TRUE)
print(WEvsSG.cos.sim)
```

---

depth *Depth of the ontology*

Description

Computes the depth of the ontology. That is, the longest path from the root term to the farthest refinement term end

Usage

```
depth(x)
```

Arguments

x	can be an OOC object, a matrix of the refinements between terms, or a list of matrices with the number of paths between each pair of terms that are directly connected for each length

Value

numeric value indicating the longest path from the root term to the far refinement term end.

Author(s)

Jose Luis Mosquera

See Also

getGk, getGr, toOOC

Examples

```
data(joslyn)

print(mat.g)
d.G <- depth(mat.g)

print(d.G)

## list of matrices with the number of paths
```

```
## between each pair of terms for each length k

Gk <- getGk(joslyn.OOC)
d.Gk <- depth(Gk)

print(d.Gk)

## matrix with all the number of paths of any length
## between each pair of terms

Gr <- getGr(Gk)
d.Gr <- depth(Gr)

print(d.Gr)
```

`distRada` *Distances of the shortest paths between each pair of terms in the ontology*

Description

This function is a edge-based approach to compute the distance of the shortest between each pair of terms in the ontology.

Usage

```
distRada(sum.paths, at)
```

Arguments

`sum.paths`	`list` of `numeric` vectors with the lengths (in terms of depth) of the number of paths between each pair of terms.
`at`	`list` of ancestors of each term

Details

This function is based on the edge-counting method (i.e. edge-based approach) proposed by Rada *et al.* The measure computes the length of the shortest path (i.e. the number of edges between a pair of terms).

This taxonomical distance was thought to be applied in trees where the shortest path between two terms contains a unique common ancestor, which is known as the Least Common Ancestor (LCA). Note that, in ontologies, this measure is not the most appropriate to be used because DAG structures may show more than one LCA. In any case, the method implemented does not distinguish a unique LCA (see function `LCAs`)

Value

Resulting object is a `matrix` whose elements show the distance (column) between each pair of different terms (rows)

Author(s)

Jose Luis Mosquera

References

Rada, R. et al. Development and application of a metric on semantic nets. Ieee Transactions On Systems Man And Cybernetics, 19(1), pp.17-30, 1989.

See Also

ancestors, LCAs, simRada, simRes.eb, summaryPaths

Examples

```
data(joslyn)

sum.paths <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

dist.rada <- distRada(sum.paths,at)
print(dist.rada)
```

---

getA *Accessibility matrix associated with the DAG structure of the ontology*

Description

Builds the accessibility matrix associated with the DAG structure of the ontology

Usage

```
getA(x)
```

Arguments

x	matrix of the number of paths of any length between each pair of terms.

Value

Resulting object is a matrix whose elements contains a TRUE value, if there exists a path of terms path following the arcs from the term on the rows to the term on the columns.

Depending on the approach applied later on, resulting matrix may be interpreted as the accessibility matrix (i.e. a node-based approach, and an edge-based approach based on distances, or a hybrid-based approach) or the comparability matrix (i.e. an edge-based based on pseudo-distances). In the first case, TRUE values indicate terms that are connected by a path, and FALSE values indicate

which not. In the second case, TRUE values indicate comparable nodes (i.e. terms), but FALSE values indicate non-comparable nodes.

Author(s)

Jose Luis Mosquera

See Also

inverseIminusG, getGr

Examples

```
data(joslyn)

## (I-Gamma)^{-1}
inv.IminusG <- inverseIminusG(joslyn.OOC)
getA(inv.IminusG)

## I + Gamma^1 + Gamma^2 + ... + Gamma^r, where r is the depth of the ontology.
Gr <- getGr(joslyn.OOC)

## (I-Gamma)^{-1} = I + Gamma^1 + Gamma^2 + ... + Gamma^r
Gr==inv.IminusG-diag(nrow(inv.IminusG))

## Accessibility matrix
getA(inv.IminusG-diag(nrow(inv.IminusG)))

getA(Gr)
```

---

getGk *Builds a* list *of the matrices with the number of paths between each pair of terms that are directly connected for each length*

Description

Builds a list whose elements are the matrices (one per each length) of the number of paths between each pair of terms that are directly connected. That is, $\Gamma^1, \Gamma^2, \ldots, \Gamma^r$, where r is the depth of the ontology

Usage

```
getGk(x)
```

Arguments

x	an OOC or a matrix of refinements

Details

Given either an OOC object or a refinement matrix (i.e. G matrix in the OOC object), computes all the number of paths for each length between each pair of terms that are directly connected. Each of these matrices are power matrices of the matrix G. For example, to obtain all the number of paths between each pair of terms connected in the ontology with length 1, the function computes D^1, to obtain all the number of paths with length 2, calculates $G^2 = G * G$, and so on until the length of the largest path, that is, the maximum depth of the ontology

Value

The resulting object is a list of matrices. Each matrix is associated with a length of paths between each pair of connected terms. Elements of these matrices are the number of paths for the same length connecting each pair of terms. For example, let r be the depth of the ontology, then the elements of the resulting list are

G1 matrix with the number of paths between each pair of terms with length 1

G2 matrix with the number of paths between each pair of terms with length 2

... ...

Gr matrix with the number of paths between each pair of terms with length r

Author(s)

Jose Luis Mosquera

See Also

getGr, depth, toOOC

Examples

```
data(joslyn)

Gk <- getGk(joslyn.OOC)  # based on an OOC object
print(Gk)

Gk.2 <- getGk(mat.g)         # based on a matrix
print(Gk.2)
```

---

getGr *Number of paths of any length between each pair of terms that are directly connected*

Description

Computes all the number of paths of any length between each pair of terms that are directly connected.

Usage

```
getGr(x)
```

Arguments

x	can be an object OOC, a matrix of refinement matrix, or a list of matrices with the number of paths of each length between each pair of terms that are directly connected

Details

The computation of the number of paths is based on the formula

$$\mathbf{I} + \Gamma^1 + \ldots + \Gamma^r = (\mathbf{I} - \Gamma)^(- 1)$$

.

There are two ways for performing the computation; first, by calculating the sum of the power matrices of the refinements matrix, and second, by computing the inverse of the matrix $\mathbf{I} - \Gamma$.

Note that depending on the class of the argument x supplied, the computation requires calculating the power matrices of the refinements matrix. That is, $\Gamma^1, \Gamma^2, \ldots, \Gamma^r$, where r is the depth of the ontology

Value

Resulting object is a matrix whose elements are the number of paths of any length between each pair of terms that are directly connected

Author(s)

Jose Luis Mosquera

See Also

getGk, toOOC

Examples

```
data(joslyn)

## Based on the OOC object

Gr <- getGr(joslyn.OOC)
print(Gr)

## Based on the refinement matrix

Gr.2 <- getGr(mat.g)
Gr==Gr.2

## Based on the list of matrices with the number of
```

```
## paths of each length between each pair of terms
## that are directly connected

Gk <- getGk(joslyn.OOC)
Gr.3 <- getGr(Gk)
Gr==Gr.3
```

---

goOOC *Builds an Object-Ontology Complex (OOC) whose slots are associated with GO Identifiers*

Description

Given a list of Entrez Gene Identifiers, the name of an R organism package, and a domain of the GO, the function builds an Object-Ontology Complex where the slots are associated with GO ID's and Entrez Gene ID's. But also, if two list of Entrez Gene Identifiers are provided, then the function builds a list with two elements, each one associated with one of the list of Entrez Gene Identifiers and consisting of an object of class OOC

Usage

```
goOOC(eg1, eg2 = NULL, pckg = "org.Hs.eg.db", ontology = "BP")
```

Arguments

eg1	character vector with the (first list of) Entrez Gene IDs to be interrogated.
eg2	character vector with the second list of Entrez Gene IDs to be interrogated. By default is NULL.
pckg	character with the name of the R organism package. By default is org.Hs.eg.db
ontology	character the ontology from the GO is selected (see *Details*. By default is BP

Details

This function is similar to the function toOOC, but it is particularly adapted to deal with information associated with the Gene Ontology.

If argument eg2 is NULL, then yields an OOC object. Otherwise, the resulting object is a list with two OOC objects. First element is associated with the first list of Entrez Gene Identifiers (provided to the function in argument eg1), and second element is associated with the second list of Entrez Gene Identifiers (provided to the function in argument eg2).

Note that, in case of providing two lists of Entrez Gene Identifiers, both objects OOC from the resulting list have the same refinement matrix (i.e. slot G). This is made in order to compare both lists of Entrez Gene IDs in terms of semantic similarities profiles later on. That is, sometimes the comparison between two list of genes is required. This tasks may be performed by comparing two semantic similarity profiles (i.e. two lists of semantic similarities between each pair of terms and performed with the same measure). Thus, each list of genes yields an induced subgraph from the GO. But, note that both subgraphs might be different. In consequence, no comparison might be

computed because the number of pairs of GO IDs wolud be different. This drawback, can be solved by considering a common subgraph induced by both lists of genes.

Argument ontology has three possibilities

BP Biological Processes

CC Cellular Components

MF Molecular Functions

Value

Resulting object

Author(s)

Jose Luis Mosquera

See Also

toOOC

Examples

```
data(prostateIds)

## Entrez Genes from two different studies
## of prostate cancer
eg.we <- welsh01EntrezIDs[1:10]  # Weslh study
eg.sg <- singh01EntrezIDs[1:10]  # Singh study

## OOC associated with Entrez Gene IDs of Welsh

ooc1 <- goOOC(eg1 = eg.we, eg2 = NULL,
              pckg = "org.Hs.eg.db", ontology = "MF")
class(ooc1)
str(ooc1)
dim(ooc1@G)
dim(ooc1@M)

## OOC associated with Entrez Gene IDs of Singh

ooc2 <- goOOC(eg1 = eg.sg, eg2 = NULL,
              pckg = "org.Hs.eg.db", ontology = "MF")
class(ooc2)
str(ooc2)
dim(ooc2@G)
dim(ooc2@M)

## List of OOC associated with Entrez Gene IDs of Welsh
## and Singh

ooc <- goOOC(eg1 = eg.we, eg2 = eg.sg,
```

```
              pckg = "org.Hs.eg.db", ontology = "MF")
class(ooc)
str(ooc)
dim(ooc[[1]]@G)
dim(ooc[[2]]@G)
table(ooc[[1]]@G==ooc[[2]]@G)
dim(ooc[[1]]@M)
dim(ooc[[2]]@M)
```

`gosims` *Wrapper function that calls different approaches and methods for computing semantic similarities between GO Identifiers given a list of Entrez Gene IDs*

Description

Given a list of Entrez Gene Identifiers, the name of an R organism package, and an ontology domain, the function builds an Object-Ontology Complex particularly adapted for dealing with GO Identifiers. But also, if two list of Entrez Gene Identifiers are provided, then the function builds a list with two elements, each one associated with one of the list of Entrez Gene Identifiers and consisting of an object of class OOC

Usage

```
gosims(eg, ontology = "BP", pckg = "org.Hs.eg.db", type = "nb", method = "Res")
```

Arguments

`eg`	`character` vector with the Entrez Gene IDs to be interrogated.
`ontology`	`character` with the ontology domain of the GO (see *Details*). By default is `BP`
`pckg`	`character` with the name of the R organism package. By default is `org.Hs.eg.db`
`type`	`character` indicating the approach for computing semantic similarities. By default `nb`
`method`	`character` indicating the method used for computing semantic similarities. By default `Res`

Details

Argument `ontology` has three possible domains

`BP` Biological Processes

`CC` Cellular Components

`MF` Molecular Functions

Argument `type` has three possibilities

`nb` node-based approach

`eb` edge-based approach

`eb.pd` edge-based approach based on pseudo-distances

Depending on the approach selected in argument `type`, different `methods` for computing semantic similarities are available.

- Possible methods of *Node-based approach* (i.e. `type = nb`) are

  `Res` semantic similarty of Resnik

  `Lin` semantic similarty of Lin

  `Rel` semantic similarty of Schlicker *et al.*

  `JC` semantic similarty of Jiang and Conrath

  `Nunivers` semantic similarty of Mazandu and Mulder

  `Psec` semantic similarty of Pirro and Seco

  `Faith` semantic similarty of Pirro and Euzenat

  `all` all sementic similarities

- Possible methods of *Edge-based approach* (i.e. `type = eb`) are

  `Rada` semantic similarty of Rada

  `Res.eb` semantic similarty of Resnik based on the shortest path (see function `distRada`)

  `all` all semantic similarities of edge-based approaches implemented in the package

- Possible pseudo-distance methods of *Edge-based approach* (i.e. `type = eb.pd`) are

  `hm` pseudo-distance of the minimum chain length

  `hx` pseudo-distance of the maximum chain length

  `hax` pseudo-distance of the average of extreme chain lengths

  `hap` pseudo-distance of the average of all chain lengths

  `all` all pseudo-distances

Author(s)

Jose Luis Mosquera

See Also

`gosimsAvsB`, `simsBetweenGOIDs`, `summarySims`, `goOOC`, `sims.nb`, `sims.eb`, `pseudoDists`

Examples

```
data(prostateIds)

## Entrez Genes from two different studies
## of prostate cancer
eg.we <- welsh01EntrezIDs[1:10]   # Welsh study
eg.sg <- singh01EntrezIDs[1:10]   # Singh study

## All semantic similarities from node-based approach

nb.all <- gosims(eg = eg.we, ontology = "MF",
                 pckg = "org.Hs.eg.db", type = "nb",
```

```
                method = "all")
head(nb.all, 20)
tail(nb.all, 20)

## All semantic similarities from edge-based approach

eb.all <- gosims(eg = eg.we, ontology = "MF",
                pckg = "org.Hs.eg.db", type = "eb",
                method = "all")
head(eb.all, 20)
tail(eb.all, 20)

## All pseudo-distances from node-based approach

pd.all <- gosims(eg = eg.we, ontology = "MF",
                pckg = "org.Hs.eg.db",
                type = "eb.pd", method = "all")
head(pd.all, 20)
tail(pd.all, 20)
```

`gosimsAvsB` *Wrapper function for computing semantic similarities between GO Identifiers for two lists of Entrez Gene IDs*

Description

Given two lists of Entrez Gene Identifiers computes semantic similarities between GO Identifiers

Usage

```
gosimsAvsB(eg1, eg2, ontology = "BP", pckg = "org.Hs.eg.db", type = "nb", method = "Res")
```

Arguments

`eg1`	`character` vector with the first list of Entrez Gene IDs to be interrogated.
`eg2`	`character` vector with the second list of Entrez Gene IDs to be interrogated.
`ontology`	`character` the ontology from the GO is selected (see *Details*). By default is `BP`
`pckg`	`character` with the name of the `R` organism package. By default is `org.Hs.eg.db`
`type`	`character` indicating the approach for computing semantic similarities. By default `nb`
`method`	`character` indicating the method used for computing semantic similarities. By default `Res`

Details

Argument ontology has three possible domains

BP Biological Processes

CC Cellular Components

MF Molecular Functions

Argument type has three possibilities

nb node-based approach

eb edge-based approach

eb.pd edge-based approach based on pseudo-distances

Depending on the approach selected in argument type, different methods for computing semantic similarities are available.

- Possible methods of *Node-based approach* (i.e. type = nb) are

  Res semantic similarty of Resnik
  Lin semantic similarty of Lin
  Rel semantic similarty of Schlicker *et al.*
  JC semantic similarty of Jiang and Conrath
  Nunivers semantic similarty of Mazandu and Mulder
  Psec semantic similarty of Pirro and Seco
  Faith semantic similarty of Pirro and Euzenat
  all all sementic similarities

- Possible methods of *Edge-based approach* (i.e. type = eb) are

  Rada semantic similarty of Rada
  Res.eb semantic similarty of Resnik based on the shortest path (see function distRada)
  all all semantic similarities of edge-based approaches implemented in the package

- Possible pseudo-distance methods of *Edge-based approach* (i.e. type = eb.pd) are

  hm pseudo-distance of the minimum chain length
  hx pseudo-distance of the maximum chain length
  hax pseudo-distance of the average of extreme chain lengths
  hap pseudo-distance of the average of all chain lengths
  all all pseudo-distances

Value

Resulting object is a matrix semantic similarity values. On the rows there are the pairs of GO ID's compared and on the columns the method(s) used for the computation of the measures. For each measure there are two columns (one per list of genes)

Author(s)

Jose Luis Mosquera

See Also

summarySims, gosims, goOOC, sims.nb, sims.eb, pseudoDists

Examples

```
data(prostateIds)

## R organism package
pckg <- "org.Hs.eg.db"

## Entrez Genes from two different studies
## of prostate cancer
eg.we <- welsh01EntrezIDs[1:10]   # Welsh study
eg.sg <- singh01EntrezIDs[1:10]   # Singh study

## Semantic similarity of Resnik (node-based approach)

WEvsSG.nb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg,
                        ontology = "MF", pckg = pckg,
                        type = "nb", method = "Res")
head(WEvsSG.nb, 20)
tail(WEvsSG.nb, 20)

## Semantic similarity of Resnik (edge-based approach)

WEvsSG.eb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg,
                        ontology = "MF", pckg = pckg,
                        type = "eb", method = "Res.eb")
head(WEvsSG.eb, 20)
tail(WEvsSG.eb, 20)

## Pseudo-distance of the minimum chains length
## (edge-based approach)

WEvsSG.pd <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg,
                        ontology = "MF", pckg = pckg,
                        type = "eb.pd", method = "hm")
head(WEvsSG.pd, 20)
tail(WEvsSG.pd, 20)
```

`gosimsProfiles` *Plots a vertical bar diagram whose bars are associated with the semantic similarities between each pair of terms, and such that bars on the left side of the plot are the corresponding to the first group of objects and on the bars on the right side are the bars corresponding to the second group of objects*

Description

Given a two-columns matrix (or data.frame), this function yields a vertical bar diagram whose bars are associated with the semantic similarities between each pair of terms, and such that bars on the left side of the plot are the corresponding to the first group of objects and on the bars on the right side are the bars corresponding to the second group of objects

Usage

```
gosimsProfiles(x, col = c("tomato", "blue"), cex = 0.4,
                   top.labels = NULL, main = NULL, xlab = "Semantinc Similarity")
```

Arguments

x	matrix (or data.frame) where for each pair terms (rows) contains the values of the semantic similarity measure estimated for each group of objects (columns)
col	character vector with the two elements containing the colors associated with each group of objects. By default are tomato for the left bars on the left side, and blue for the bars on the right side
cex	numeric value with the expansion for the category labels. By default is 0.4
top.labels	character vector indicating the two categories represented on the left and right sides of the plot and a heading for the labels in the center. By default is NULL
main	character with title for the plot. By default NULL
xlab	character with the label for the units of the plot (i.e. labels on the x-axis).

Author(s)

Jose Luis Mosquera

See Also

gosimsAvsB, summarySimsAvsB, plotHistSims, plotGODAG

Examples

```
data(prostateIds)

## Entrez Genes from two different studies
## of prostate cancer
eg.we <- welsh01EntrezIDs[1:10]   # Welsh study
eg.sg <- singh01EntrezIDs[1:10]   # Singh study

WEvsSG.nb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg,
                        ontology = "MF", pckg = "org.Hs.eg.db",
                        type = "nb", method = "Res")

## Plot of the whole profiles of semantic similarities

gosimsProfiles(x = WEvsSG.nb, col = c("tomato", "blue"),
               cex = 0.4, top.labels = c("WE", "SG"),
```

```
                main = "Welsh 01 vs Singh 01", xlab = "Resnik")

## Plot a subset of the whole profiles of semantic similarities

gosimsProfiles(x = WEvsSG.nb[1:200, ], col = c("tomato", "blue"),
                cex = 0.4, top.labels = c("WE", "SG"),
                main = "Welsh 01 vs Singh 01", xlab = "Resnik")
```

ICA *Information Content (IC) of common ancestors*

Description

Builds a `list` of numeric vectors with the Information Content (IC) associated with each common ancestor of each pair of terms

Usage

```
ICA(x, ic)
```

Arguments

`x`	`list` of common ancestors for each pair of terms
`ic`	`numeric` vector with the IC of each term in the ontology

Value

The resulting object is a `list` whose elements are `numeric` vectors with the ICs of the common ancestors for each pair of terms

Author(s)

Jose Luis Mosquera

References

Resnik P. "Using information content to evaluate semantic similarity in a taxonomy". Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 448-453, 1995.

See Also

`ancestors`, `commonAncestors`, `inverseIminusG`, `resnikSummary`

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)     # (I - G)^(-1)
A.mat <- getA(inv.IminusG)                    # accessibility matrix
at <- ancestors(A.mat)                        # ancestors
ca <- commonAncestors(at)                     # common ancestors
resnik.sum <- resnikSummary(x = joslyn.OOC)   # summary of Resnik measures

ica <- ICA(x = ca, ic = resnik.sum[, "ic"])
print(ica)
```

---

inverseIminusG *Computes the number of paths of any length between each pair of terms in the ontology*

Description

Builds a `matrix` with the number of paths of any length between each pair of terms in the ontology

Usage

```
inverseIminusG(x)
```

Arguments

`x`	can be an object of class `OOC` or a `matrix` with the refinements of each term in the ontology

Details

In order to compute faster the matrix, applies the formula `(I - Gamma)^(-1)`, instead of using `I + Gamma^1 + ... + Gamma^r` where `r` is the depth of the ontology.

Value

The resulting object is a `matrix` whose elements are the number of paths

Author(s)

Jose Luis Mosquera

See Also

`toOOC`

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
print(inv.IminusG)
```

`is.OOC` *Tests if its argument is a (strict)* `OOC` *object*

Description

Tests if its argument is a (strict) `OOC` object

Usage

```
is.OOC(x)
```

Arguments

`x`	an R object

Details

`is.OOC` returns `TRUE` if argument `x` is an `OOC` object, and `FALSE` otherwise

Author(s)

Jose Luis Mosquera

See Also

`toOOC`

Examples

```
data(joslyn)

is.OOC(mat.m)            # FALSE
is.OOC(joslyn.OOC)   # TRUE
```

LCAs *Length of the shortest paths containing the Least Common Ancestors (LCA) between each pair of terms*

Description

Computes the Least Common Ancestors between each pair of nodes

Usage

```
LCAs(at, sum.paths)
```

Arguments

at	list of ancestors of each term
sum.paths	list of numeric vectors with the lengths (in terms of depth) of the number of paths between each pair of terms.

Details

The function looks for all the least common ancestor of each pair of terms and computes the shortest path between such terms containing the corresponding LCA.

Value

The resulting object is a list whose elements are numeric vectors with the lengths of the shortest paths associated with each LCA.

Author(s)

jose Luis Mosquera

See Also

ancestors, summaryPaths

Examples

```
data(joslyn)

sum.paths <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

lcas <- LCAs(at, sum.paths)
print(lcas)
```

mapEG2GO *Mapping Entrez Gene IDs to Gene Ontology IDs*

Description

This function maps the Entrez Gene IDs on to the GO IDs provided by an R organism package (e.g. org.Hs.eg.db

Usage

```
mapEG2GO(eg = NULL, pckg = "org.Hs.eg.db")
```

Arguments

eg	character vector with the Entrez Gene IDs. By default is NULL, which means that all the eg from the pckg will be mapped
pckg	character with the name of the package of the organism from where the GO IDs are extracted. By default is org.Hs.eg.db, that is, Homo sapiens specie

Value

The resulting object is a list with three elements, corresponding to an ontology of the the GO (i.e BP, CC and MF). Each of these elements is a data.frame with two columns. First column consists of the list of Entrez Gene IDs mapping the GO IDs placed in the second column. Note that in both columns, the concepts can be repeated. That is, an Entrez Gene ID can map different GO IDs, and a GO ID can be mapped by different Entrez Gene IDs

Author(s)

Jose Luis Mosquera

Examples

```
data(prostateIds)              # Data set from the package goProfiles

pckg <- "org.Hs.eg.db"         # Names of the organism package
eg <- welsh01EntrezIDs[1:10]   # Entrez Gene IDs

eg2go <- mapEG2GO(eg, pckg)
lapply(eg2go, head)
```

mappingMatrix *Mapping matrix from the Entrez Gene IDs to the GO IDs associated with the directed subgraph extracted from GO DAG structure*

Description

Based on the `data.frame` of the maps the Entrez Gene IDs on to the GO IDs and the refinement matrix, this function builds the mapping matrix associated with the GO DAG structure

Usage

```
mappingMatrix(G, df)
```

Arguments

`G`	binary `matrix` indicating the refinements of each GO ID
`df`	`data.frame` with mapping from the Entrez Gene IDs to the GO ID terms

Details

Given the `data.frame` with mapping from the Entrez Gene IDs to the GO IDs, `mappingMatrix` and the refinement matrix, builds a binary `matrix` where on rows are the Entrez Gene IDs and on the columns are all the GO IDs associated with the directed subgraph extracted from the GO DAG.

Value

The resulting object is a binary `matrix`. Each element of this `matrix` is either 1 when an Entrez Gene ID on a row is mapping a GO ID on a column, or 0 otherwise.

Author(s)

Jose Luis Mosquera

See Also

`mapEG2GO`, `refinementMatrix`

Examples

```
data(prostateIds)                    # Data set from the package goProfiles

pckg <- "org.Hs.eg.db"               # Names of the organism package
eg <- welsh01EntrezIDs[1:10]         # Entrez Gene IDs

eg2go <- mapEG2GO(eg, pckg)
eg2go.mf <- eg2go$MF                 # Get mollecular functions

G <- refinementMatrix(df = eg2go.mf, ontology = "MF")
```

```
M <- mappingMatrix(G, df = eg2go.mf)
print(M)
```

`Nt` *Number of times that each term or any of its specializations references to an ancestor*

Description

Calulates the `matrix` of the number of times that each term or any of its specializations references to an ancestor

Usage

```
Nt(x)
```

Arguments

`x`	an object of class `OOC`

Details

The resulting matrix is based on the Neumann series of a matrix. The function calculates the `matrix` based on the formula

$$N_t = M(I - \Gamma)^{-1}$$

Author(s)

Jose Luis Mosquera

See Also

`toOOC`,

Examples

```
data(joslyn)

N.t <- Nt(joslyn.OOC)
print(N.t)
```

OOC — *General container for an Object-Ontology Complex (OOC)*

Description

Class "OOC" is a general container for Object-Ontology Complexes (OOC)

Objects from the Class

Objects can be created by calls of either the form `new("OOC", ...)` or the function `toOOC(T, G, O, M)`

Slots

`T` `character` vector with the names of the terms in the ontology

`G` `matrix` encoding the refinement matrix associated with the DAG of the ontology. Elements of this matrix are 1, when a term on the rows refines a term on the columns, and 0 otherwise

`O` `character` vector with the object identifiers (i.e. features, genes,...) mapping the terms of the ontology

`M` `matrix` encoding the mapping from the set of object identifiers (listed in the slot `O` to the terms listed in the slot `T`

Methods

No methods defined with class `OOC` in the signature

Author(s)

Jose Luis Mosquera

See Also

`toOOC`

pdHap — *Pseudo-distances of the average of all chain lengths between comparable terms of the ontology.*

Description

This function computes pseudo-distances of the average of all chain lengths proposed by Josly *et al.*

Usage

```
pdHap(sum.paths)
```

Arguments

sum.paths	list of vectors with the lengths of the chains of each pair of terms

Details

Pseudo-distance concept is an strategy of the edge-based approach for measuring the distances between terms in an ontology. It was proposed by Joslyn *et al.*

pdHap computes the pseudo-distances proposed of the average of all chain lengths. That is, let t_i and t_j be two comparable terms, then the average of all chain length is defined as

$$\delta_{ap}(t_i, t_j) = \frac{\sum_{h \in \mathbf{h}(t_i, t_j)} h}{|\mathbf{h}|}$$

In case of non-comparable terms, the resulting value is NA.

Author(s)

Jose Luis Mosquera

References

Cliff A. Joslyn, Susan M. Mniszewski Andy W. Fulmer, and Gary G. Heaton. (2004). "The gene ontology categorizer". Bioinformatics, 20(s1):169-77, 2004.

See Also

pdHm, pdHx, pdHax, pseudoDists

Examples

```
data(joslyn)

length.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)   # length of chains
num.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = FALSE)    # number of chains          ##
sum.paths <- lapply(1:nrow(length.chains), function(i, x, y)
                                          {
                                              ch <- rep(x[i, ], y[i, ])
                                              if(sum(ch)==0) ch <- 0
                                              return(ch)
                                          },
                                          x = length.chains,
                                          y = num.chains)
names(sum.paths) <- rownames(length.chains)

pseudodist.Hap <- pdHap(sum.paths)
print(pseudodist.Hap)
```

pdHax *Pseudo-distances of the average of extreme chain lengths between comparable terms of the ontology.*

Description

This function computes pseudo-distances of the average of extreme chain lengths proposed by Josly *et al.*

Usage

```
pdHax(sum.paths)
```

Arguments

sum.paths	list of vectors with the lengths of the chains of each pair of terms

Details

Pseudo-distance concept is an strategy of the edge-based approach for measuring the distances between terms in an ontology. It was proposed by Joslyn *et al.*

pdHax computes the pseudo-distances proposed of the average of extreme chain lengths. That is, let t_i and t_j be two comparable terms, then the average of extreme chain length is defined as

$$\delta_{ax}(t_i, t_j) = \frac{h_*(t_i, t_j) + h^*(t_i, t_j)}{2}$$

In case of non-comparable terms, the resulting value is NA.

Author(s)

Jose Luis Mosquera

References

Cliff A. Joslyn, Susan M. Mniszewski Andy W. Fulmer, and Gary G. Heaton. (2004). "The gene ontology categorizer". Bioinformatics, 20(s1):169-77, 2004.

See Also

pdHm, pdHx, pdHap, pseudoDists

Examples

```
data(joslyn)

length.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)   # length of chains
num.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = FALSE)    # number of chains          ##
sum.paths <- lapply(1:nrow(length.chains), function(i, x, y)
                                          {
                                              ch <- rep(x[i, ], y[i, ])
                                              if(sum(ch)==0) ch <- 0
                                              return(ch)
                                          },
                                          x = length.chains,
                                          y = num.chains)
names(sum.paths) <- rownames(length.chains)

pseudodist.Hax <- pdHax(sum.paths)
print(pseudodist.Hax)
```

---

pdHm *Pseudo-distance of the minimum chain lengths between comparable terms of the ontology.*

Description

This function computes pseudo-distances of the minimum chain lengths proposed by Josly *et al.*

Usage

```
pdHm(sum.paths)
```

Arguments

sum.paths	list of vectors with the lengths of the chains of each pair of terms

Details

Pseudo-distance concept is an strategy of the edge-based approach for measuring the distances between terms in an ontology. It was proposed by Joslyn *et al.*

pdHm computes the pseudo-distances proposed of the minimum chain lengths. That is, let t_i and t_j be two comparable terms, then the minimum chain lengths is defined as

$$\delta_m(t_i, t_j) = h_*(t_i, t_j) = \min_{C \in \mathcal{C}(t_i, t_j)} |C|$$

In case of non-comparable terms, the resulting value is NA.

Author(s)

Jose Luis Mosquera

References

Cliff A. Joslyn, Susan M. Mniszewski Andy W. Fulmer, and Gary G. Heaton. (2004). "The gene ontology categorizer". Bioinformatics, 20(s1):169-77, 2004.

See Also

pdHx, pdHax, pdHap, pseudoDists

Examples

```
data(joslyn)

length.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)   # length of chains
num.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = FALSE)    # number of chains          ##
sum.paths <- lapply(1:nrow(length.chains), function(i, x, y)
                                          {
                                              ch <- rep(x[i, ], y[i, ])
                                              if(sum(ch)==0) ch <- 0
                                              return(ch)
                                          },
                                          x = length.chains,
                                          y = num.chains)
names(sum.paths) <- rownames(length.chains)

pseudodist.Hm <- pdHm(sum.paths)
print(pseudodist.Hm)
```

---

pdHx *Pseudo-distance of the maximum chain lengths between comparable terms of the ontology.*

Description

This function computes pseudo-distances of the maximum chain lengths proposed by Josly *et al.*

Usage

```
pdHx(sum.paths)
```

Arguments

sum.paths	list of vectors with the lengths of the chains of each pair of terms

Details

Pseudo-distance concept is an strategy of the edge-based approach for measuring the distances between terms in an ontology. It was proposed by Joslyn *et al.*

pdHx computes the pseudo-distances proposed of the maximum chain lengths. That is, let t_i and t_j be two comparable terms, then the maximum chain lengths is defined as

$$\delta_x(t_i, t_j) = h^*(t_i, t_j) = \max_{C \in \mathcal{C}(t_i, t_j)} C|C|$$

In case of non-comparable terms, the resulting value is NA.

Author(s)

Jose Luis Mosquera

References

Cliff A. Joslyn, Susan M. Mniszewski Andy W. Fulmer, and Gary G. Heaton. (2004). "The gene ontology categorizer". Bioinformatics, 20(s1):169-77, 2004.

See Also

pdHm, pdHax, pdHap, pseudoDists

Examples

```
data(joslyn)

length.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)   # length of chains
num.chains <- summaryPaths(x = joslyn.OOC, root = "R", len = FALSE)    # number of chains          ##
sum.paths <- lapply(1:nrow(length.chains), function(i, x, y)
                                          {
                                              ch <- rep(x[i, ], y[i, ])
                                              if(sum(ch)==0) ch <- 0
                                              return(ch)
                                          },
                                          x = length.chains,
                                          y = num.chains)
names(sum.paths) <- rownames(length.chains)

pseudodist.Hx <- pdHx(sum.paths)
print(pseudodist.Hx)
```

plotGODAG *Plots a subgraph from the GO associated with one or two lists of Entrez Gene Identifiers*

Description

This function plots a subgraph from the ontology selected from the GO (see *Details*) associated with one or two lists of Entrez Gene IDs

Usage

```
plotGODAG(eg1, eg2 = NULL, pckg = "org.Hs.eg.db", ontology = "MF", verbose = FALSE)
```

Arguments

eg1	character vector with the (first list of) Entrez Gene IDs
eg2	character vector with the second list of Entrez Gene IDs. By default is NULL
pckg	character with the name of the R organism package. By default is org.Hs.eg.db
ontology	character the ontology from the GO is selected (see *Details*. By default is BP
verbose	logical . By default FALSE

Details

There are three possibilities, of course, in argument ontology

- BPBiological Processes
- CCCellular Components
- MFMolecular Functions

The subgraph shows two types of shapes for each node. Circles are nodes not mapped directly and rectangles are nodes mapped directly.

The color of nodes indicate the type of relation with the Entrez Gene IDs. That is, when argument eg2 is NULL, there are two possibilities: nodes mapped directly are shown in red color and their ancestors are shown in yellow color. But, if argument eg2 is not NULL, then there are six different colors. Nodes mapped directly from the first list of Entrez Gene IDs are shown in red color and their ancestors are shown in yellow color. Nodes mapped directly from the second list of Entrez Gene IDs are shown in lightblue color and their ancestors are shown in blue color. Nodes mapped directly from both lists of Entrez Gene IDs are shown in magenta color and their ancestors are shown in violet color.

Author(s)

Jose Luis Mosquera

See Also

gosimsAvsB, summarySimsAvsB, gosimsProfiles

Examples

```
data(prostateIds)               # Data set from the package goProfiles

pckg = "org.Hs.eg.db"           # Organism package of humans

eg.we <- welsh01EntrezIDs[1:10]   # Entrez Gene IDs signature Welsh 01
eg.sg <- singh01EntrezIDs[1:10]   # Entrez Gene IDs signature Singh 01

plotGODAG(eg1 = eg.we, eg2 = NULL, pckg = pckg, ontology = "MF")
plotGODAG(eg1 = eg.sg, eg2 = NULL, pckg = pckg, ontology = "MF")
plotGODAG(eg1 = eg.we, eg2 = eg.sg, pckg = pckg, ontology = "MF")
```

---

plotHistSims *Histogram of two semantic similarity profiles*

Description

This function plots an histogram of two semantic similarity profiles in the same plot

Usage

```
plotHistSims(x, freq = TRUE, main = "Histogram of Semantic Similarities", xlab = "Semantic Similarity")
```

Arguments

x	matrix with measures of semantic similarity measures provided by the function gosimsAvsB
freq	logical value. If TRUE, histogram graphic is a representation of frequencies, the "counts" component of the result. If FALSE, probability densities, component "density", are plotted (so that the histogram has a total area of one). By default is TRUE
main	character to title the graphic
xlab	character to label x-axis

Details

The plot shows two "curves" in the same figure overlapped. Area colored in red is associated with the first column of the matrix provided in argument x, and area colored in blue is associated with the second column of the matrix. Intersected area is colored in *violet*.

Value

Resulting object is of class histogram

Author(s)

Jose Luis Mosquera

See Also

gosimsAvsB, summarySimsAvsB, gosimsProfiles

Examples

```
data(prostateIds)               # Data set from the package goProfiles

pckg = "org.Hs.eg.db"           # Organism package of humans

eg.we <- welsh01EntrezIDs[1:10]   # Entrez Gene IDs signature Welsh 01
eg.sg <- singh01EntrezIDs[1:10]   # Entrez Gene IDs signature Singh 01

WEvsSG.nb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg,
                        ontology = "MF", pckg = pckg,
                        type = "nb", method = "Res")

plotHistSims(x = WEvsSG.nb, freq = TRUE,
             main = "Histogram of Semantic Similarities",
             xlab = "Semantic Similarity")
```

---

pseudoDists *Wrapper function that calls different methods for computing pseudo-distances*

Description

It is a wrapper function that calls different methods for computing pseudo-distances

Usage

```
pseudoDists(x, root = NULL, method = "hm")
```

Arguments

x	OOC object
root	character with the name of the root term of the ontology in the OOC object
method	character indicating the method for computing the pseudo-distance. The options implemented in sims package are • hmpseudo-distance of the minimum chain length • hxpseudo-distance of the maximum chain length • haxpseudo-distance of the average of extreme chain lengths • happseudo-distance of the average of all chain lengths • allall pseudo-distances By default computes the pseudo-distance of the minimum chain length, i.e. hm

Details

In case of non-comparable terms, the resulting value is NA

Value

The resulting object is a `data.frame` where for each pair of different terms (rows) are shown the pseudo-distances proposed by Joslyn *et al.* (columns).

Author(s)

Jose Luis Mosquera

References

Cliff A. Joslyn, Susan M. Mniszewski Andy W. Fulmer, and Gary G. Heaton. (2004). "The gene ontology categorizer". Bioinformatics, 20(s1):169-77, 2004.

See Also

`toOOC`, `pdHm`, `pdHx`, `pdHax`, `pdHap`

Examples

```
data(joslyn)

pseudoDists(x = joslyn.OOC, root = "R", method = "hm")          # minimum chain length
pseudoDists(x = joslyn.OOC, root = "R", method = "hx")          # maximum chain length
pseudoDists(x = joslyn.OOC, root = "R", method = "hax")      # average of extreme chain lengths
pseudoDists(x = joslyn.OOC, root = "R", method = "hap")       # average of all chain lengths

pseudoDists(x = joslyn.OOC, root = "R", method = "all")         # all pseudo-distances
```

---

refinementMatrix *Builds the refinement matrix associated with the DAG structure of Gene Ontology*

Description

Based on the `data.frame` of the maps the Entrez Gene IDs on to the GO IDs, this function builds the refinement matrix (i.e. in terms of the graph theory the accessibility matrix) associated with the GO DAG structure

Usage

```
refinementMatrix(df, ontology = "BP")
```

Arguments

df	data.frame with mapping from the Entrez Gene IDs to the GO ID terms
ontology	character indicating which ontologies is selected (see *Details*). By default is BP (i.e. Biological Processes

Details

Given the data.frame with mapping from the Entrez Gene IDs to the GO IDs, refinementMatrix looks for all the ancestors and builds the matrix with the refinements associated with the directed subgraph extracted from the GO DAG.

ontology argument requires to indicate which of the ontologies in GO is selected for building the refinement matrix. Obviously, there are three possibilities

- BPBiological Processes
- CCCellular Components
- BPMolecular Functions

Value

The resulting object is a binary matrix whose rows and columns are all the GO IDs that make up the subgraph of the GO DAG. Each element of this matrix is either 1 when there exists a refinement from the GO ID on the row to the GO ID on the column, or 0 otherwise.

Author(s)

Jose Luis Mosquera

See Also

mapEG2GO, mappingMatrix

Examples

```
data(prostateIds)              # Data set from the package goProfiles

pckg <- "org.Hs.eg.db"         # Names of the organism package

eg <- welsh01EntrezIDs[1:10]   # Entrez Gene IDs
eg2go <- mapEG2GO(eg, pckg)
eg2go.mf <- eg2go$MF

G <- refinementMatrix(df = eg2go.mf, ontology = "MF")
print(G)
```

`resnikSummary` *Summary table providing with the number of times that each term or any of its refinements appears in the OOC, the probability of finding the term, and the Information Content of the term*

Description

Builds a `data.frame` providing the number of times that each term or any of its refinements appears in the OOC (i.e. `n(t_{i})`), the probability of finding the term (i.e. `p(t_{i})`), and the Information Content of the term (i.e. `IC(t_{i})`)

Usage

```
resnikSummary(x, root = NULL)
```

Arguments

`x`	an object of class `OOC`
`root`	a `character` with the name of the root term in the ontology. If `NULL`, it takes the first element of vocabulary, that is, the first element of the slot `T` from the object `OOC` `x`)

Value

The resulting object is a `data.frame` where on rows are all the terms of the ontology, and on columns are the following measures

`nt`	the number of times that each term or any of its refinements appears in the OOC
`pt`	the probability of finding the term
`ic`	the Information Content of the term

Author(s)

Jose Luis Mosquera

References

Resnik, P. "Using information content to evaluate semantic similarity in a taxonomy". Proceeding of the 14th International Joint Conference on Artificial Intelligence. pp. 448-453, 1995.

See Also

`Nt`, `toOOC`

Examples

```
data(joslyn)

sum.resnik <- resnikSummary(joslyn.OOC)
print(sum.resnik)
```

`simFaith` *Semantic similarity of Pirro and Euzenat for each pair of terms*

Description

Computes the semantic similarity proposed by Pirro and Euzenat for each pair of terms

Usage

```
simFaith(sum.mica)
```

Arguments

`sum.mica`	`data.frame` generated with function `summaryMICA`

Details

Pirro and Euzenat introduced a semantic similarity based on the Most Informative Common Ancestor (MICA). It is a node-based approach, such that

$$sim_{Faith}(t_i, t_j) = \frac{IC(MICA)}{/} IC(t_i + IC(t_j - IC(MICA)$$

Author(s)

Jose Luis Mosquera

References

Pirro, G. and Euzenat, J. "A Feature and Information Theoretic Framework for Semantic Similarity and Relatedness". In Proceedings of the 9th International Semantic Web Conference ISWC. Springer, pp.615-630, 2010.

See Also

`ancestors`, `resnikSummary`, `simJC`, `simLin`, `simNunivers`, `simPsec`, `simRel`, `simRes`, `sims.nb`, `summaryMICA`

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)

sim.Faith <- simFaith(sum.mica)
print(sim.Faith)
```

`simJC` *Semantic similarity of Jiang and Conrath for each pair of terms*

Description

Computes the semantic similarity proposed by Jiang and Conrath for each pair of terms

Usage

```
simJC(sum.mica)
```

Arguments

`sum.mica`	`data.frame` generated with function `summaryMICA`

Details

Jiang and Conrath introduced a semantic similarity based on the Most Informative Common Ancestor (MICA). It is a node-based approach, such that

$$sim_{JC}(t_i, t_j) = \frac{1}{1 + dist_{JC}(t_i, t_j)} = \frac{1}{1 + (IC(t_i) + IC(t_j) - (2MICA))}$$

Author(s)

Jose Luis Mosquera

References

Jiang, J.J. and Conrath, D.W. "Semantic similarity based on corpus statistics and lexical taxonomy". In Proceedings of International Conference Research on Computational Linguistics (ROCLING X), pp.19-33, 1997.

See Also

ancestors, resnikSummary, simFaith, simLin, simNunivers, simPsec, simRel, simRes, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)

sim.JC <- simJC(sum.mica)
print(sim.JC)
```

---

simLin *Semantic similarity of Lin for each pair of terms*

Description

Computes the semantic similarity proposed by Lin for each pair of terms

Usage

```
simLin(sum.mica)
```

Arguments

sum.mica	data.frame generated with function summaryMICA

Details

Lin introduced a semantic similarity based on the Most Informative Common Ancestor (MICA). It is a node-based approach, such that

$$sim_{Lin}(t_i, t_j) = 2IC(MICA)/(IC(t_i) + IC(t_j))$$

Author(s)

Jose Luis Mosquera

References

Lin, D. "An information-theoretic definition of similarity". In Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers, pp. 296<e2><80><93>304, 1998.

See Also

ancestors, resnikSummary, simJC, simFaith, simNunivers, simPsec, simRel, simRes, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)

sim.Lin <- simLin(sum.mica)
print(sim.Lin)
```

---

simNunivers *Semantic similarity of Mazandu and Mulder for each pair of terms*

Description

Computes the semantic similarity proposed by Mazandu and Mulder for each pair of terms

Usage

```
simNunivers(sum.mica)
```

Arguments

sum.mica	data.frame generated with function summaryMICA

Details

Mazandu and Mulder introduced a semantic similarity based on the Most Informative Common Ancestor (MICA). It is a node-based approach, such that

$$sim_{Nunivers}(t_i, t_j) = \frac{IC(MICA)}{\max\{IC(t_i), IC(t_j\}}$$

Author(s)

Jose Luis Mosquera

References

Mazandu, G.K. and Mulder, N.J. "Information content-based Gene Ontology semantic similarity approaches: Toward a unified framework theory". BioMed Research International, 2013.

See Also

ancestors, resnikSummary, simJC, simLin, simFaith, simPsec, simRel, simRes, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)

sim.Nunivers <- simNunivers(sum.mica)
print(sim.Nunivers)
```

---

simPsec *Semantic similarity of Pirro and Seco for each pair of terms*

Description

Computes the semantic similarity proposed by Pirro and Seco for each pair of terms

Usage

```
simPsec(sum.mica)
```

Arguments

sum.mica	data.frame generated with function summaryMICA

Details

Pirro and Seco introduced a semantic similarity based on the Most Informative Common Ancestor (MICA). It is a node-based approach, such that

$$sim_{Psec}(t_i, t_j) = (3IC(MICA)) - IC(t_i - IC(t_j$$

Author(s)

Jose Luis Mosquera

References

Pirro, G. and Seco, N. "Design, Implementation and Evaluation of a New Semantic Similarity Metric Combining Features and Intrinsic Information Content". Meersman, R. and Tari, Z. eds. Lecture Notes in Computer Science, 5332:1271-1288, 2008.

Pirro, G. "A semantic similarity metric combining features and intrinsic information content". Data and Knowledge Engineering, 68(11):1289-1308, 2009.

See Also

ancestors, resnikSummary, simJC, simLin, simNunivers, simFaith, simRel, simRes, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)

sim.Psec <- simPsec(sum.mica)
print(sim.Psec)
```

---

simRada *Semantic similarity measure of Rada* et al. *for each pair of terms*

Description

Computes the semantic similarity, based on the shortest path, proposed by Rada *et al.* between each pair of terms

Usage

```
simRada(sum.paths, at)
```

Arguments

sum.paths	list of numeric vectors with the lengths (in terms of depth) of the number of paths between each pair of terms.
at	list of ancestors of each term

Details

The measure involved in this function is an edge-based approach. It was proposed by Rada *et al.*. The measure is a transformation from the distance of Rada *et al.* (see function distRada) to semantic similarity through the formula $sim = \frac{1}{1+dist}$. That is, the semantic similarity proposed by Rada *et al.* is defined as

$$sim_{Rada}(t_i, t_j) = \frac{1}{1 + d_{Rada}(t_i, t_j)}$$

Value

The resulting object is a matrix where for each pair of different terms (rows) is shown the numeric value of the semantic similarity computed (column).

Author(s)

Jose Luis Mosquera

References

Rada, R. et al. Development and application of a metric on semantic nets. Ieee Transactions On Systems Man And Cybernetics, 19(1), pp.17-30, 1989.

See Also

ancestors, simRes.eb, distRada, LCAs, summaryPaths

Examples

```
data(joslyn)

sum.paths <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

sim.Rada <- simRada(sum.paths, at)
print(sim.Rada)
```

---

simRel *Semantic similarity of Schlicker* et al. *for each pair of terms*

Description

Computes the semantic similarity proposed by Schlicker *et al.* for each pair of terms

Usage

```
simRel(sum.mica, ic)
```

Arguments

sum.mica	data.frame generated with function summaryMICA
ic	numeric vector with the IC of each term in the ontology

Details

Schlicker *et al.* noted that by taking into account specificities of compared terms can lead to high similarities when comparing general terms. For instance, when comparing general terms the semantic similarity of Lin (see function 'simLin'), the maximal similarity will be obtained comparing a (general) term to itself. Actually, the identity of the indiscernible is generally ensured, with the exception of the root which has an $IC = 0$. However, some treatments require this property not to be respected. Therefore, Schlicker *et al.* introduced a semantic similarity based on a modification of the semantic similarity of Lin, such that

$$sim_{Rel}(t_i, t_j) = sim_{Lin}(t_i, t_j)(1 - P(MICA))$$

Value

The resulting object is a matrix where for each pair of different terms (rows) is shown the numeric value of the semantic similarity computed (column)

Author(s)

Jose Luis Mosquera

References

Schlicker, A. *et al.* "A new measure for functional similarity of gene products based on Gene Ontology". BMC Bioinformatics, 7(1):302, 2006.

See Also

ancestors, resnikSummary, simJC, simLin, simNunivers, simPsec, simFaith, simRes, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]
sum.mica <- summaryMICA(at, ic)
```

```
sim.Rel <- simRel(sum.mica, ic)
print(sim.Rel)
```

`simRes` *Semantic similarity of Resnik for each pair of terms*

Description

Computes the semantic similarity proposed by Resnik for each pair of terms

Usage

```
simRes(at, ic, subs = FALSE)
```

Arguments

`at`	list whose elements are the ancestors of each term
`ic`	numeric vector with the IC of each term in the ontology
`subs`	logical value. If TRUE, then the resulting data.frame will show an etra column with the name of the subsumer term associated with MICA value

Details

Given two terms of an ontology the Most Informative Common Ancestor (MICA) is the common ancestor with the highest Information Content (IC). This measure is a semantic similarity, from the node-based approaches, introduced by Resnik and defined as

$$sim_{Res}(t_i, t_j) = IC(MICA) = \max_{t \in S(t_i, tj)}(IC(t)$$

where $S(t_{i,}, tj)$ is the set of terms that subsumes both terms t_i and t_j, and $IC(t)$ is the information content measure of t.

MICA does not take into account the disjoint common ancestors, that is, those common ancestors that do not subsume any other common ancestor.

Value

The resulting object is a `matrix` where for each pair of different terms (rows) is shown the `numeric` value of the semantic similarity computed (column)

Author(s)

Jose Luis Mosquera

References

Resnik P. "Using information content to evaluate semantic similarity in a taxonomy". Proceedings of the 14th International Joint Conference on Artificial Intelligence. 448-453, 1995

See Also

ancestors, commonAncestors, resnikSummary, simJC, simLin, simNunivers, simPsec, simRel, simFaith, sims.nb, summaryMICA

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
ca <- commonAncestors(at)
resnik.sum <- resnikSummary(x = joslyn.OOC)

sim.Res <- simRes(at, ic = resnik.sum[, "ic"])
print(sim.Res)
```

---

simRes.eb *Semantic similarity measure of Resnik* et al. *for each pair of terms, considering the maximal depth of the ontology*

Description

Computes the semantic similarity measure of Resnik between each pair of node, considering the maximal depth of the ontology.

Usage

```
simRes.eb(sum.paths, at, x)
```

Arguments

sum.paths	list of numeric vectors with the lengths (in terms of depth) of the number of paths between each pair of terms.
at	list of ancestors of each term
x	OOC object

Details

This function computes a measure proposed by Resnik in order to normalize the measure of semantic evidences from the graph of the ontology and computed with the the shortest path containing the LCAs of the terms involved int (see function distRada). It is an edge-based approach, and given two terms t_i and t_j, it is defined as

$$latex$$

Value

The resulting object is a `matrix` where for each pair of different terms (rows) is shown the `numeric` value of the semantic similarity computed (column).

Author(s)

Jose Luis Mosquera

References

Resnik P. "Using information content to evaluate semantic similarity in a taxonomy". Proceedings of the 14th International Joint Conference on Artificial Intelligence. 448-453, 1995

Rada, R. et al. Development and application of a metric on semantic nets. Ieee Transactions On Systems Man And Cybernetics, 19(1), pp.17-30, 1989.

See Also

`distRada`, `simRada`

Examples

```
data(joslyn)

sum.paths <- summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

sim.Res.eb <- simRes.eb(sum.paths, at, x = joslyn.OOC)
print(sim.Res.eb)
```

`sims` *Computation of semantic similarity measures between terms of an ontology*

Description

`sims` package is devoted to compute semantic similarity measures between terms of an ontology, and some functions are particularly addressed to manage the Gene Ontology terms.

Details

This package is designed to compute semantic similarities between terms of any ontology. Fourteen measures from different approaches are implemented. Specifically, from node-based approach there are implemented seven semantic similarity measures proposed by Resnik, Lin, Schlicker et al., Jiang and Conrath, Mazandu and Mulder, Pirro and Seco, and Pirro and Euzenat. With regard to edge-based approaches there are implemented: two semantic similarity measures proposed by Resnik,

and Rada et al, one distance measure proposed by Rada and four pseudo-distances proposed by Joslyn et al. The package can manage any ontology, but it is particularly focused on the Gene Ontology. In this regard, there are some functions that allow building the refinements matrix (i.e. the accessibility matrix in terms of graph theory), the mapping matrix (i.e the matrix that maps from Entrez Gene IDs to GO IDs) and plot the DAG structure in order to compare two different list of GO terms. `sims` package can manage Entrez Gene IDs and GO IDs from any organism `R` package.

Package:	sims
Type:	Package
Version:	1.0
Date:	31-08-2014
License:	GPL-2
Depends:	AnnotationDbi, expm, goProfiles, GOstats, igraph, methods, plotrix, Rgraphviz, vegan
Imports:	Matrix, plyr, knitr
Suggests:	org.Hs.eg.db

The most important functions are `sims.nb`, `sims.eb`, `pseudoDists`, `gosims` and `gosimsAvsB`

Author(s)

Jose Luis Mosquera and Alex Sanchez

Maintainer: Jose Luis Mosquera <jlmosquera@gmail.com>

See Also

`sims.nb`, `sims.eb`, `pseudoDists`, `gosims`, `gosimsAvsB`

`sims.eb` *Wrapper function that calls different methods for computing semantic similarities based on edge-based approaches*

Description

It is wrapper function that calls different methods for computing semantic similarities based on edge-based approaches

Usage

```
sims.eb(x, root = NULL, at, method = "Rada")
```

Arguments

`x`	`OOC` object
`root`	`charcter` with the name of the root term of the ontology. By default, it takes the first element of vocabulary (i.e. the first element of the slot `T` from the `OOC` object passed in the argument `x`

at — list of character vectors with the ancestors of each term

method — character indicating the method for computing the semantic similarity. The options implemented in sims package are

- Rada semantic similarty of Rada
- Res.eb semantic similarty of Resnik based on the shortest path (see function distRada)
- all all semantic similarities of edge-based approaches implemented in the package

By default computes the semantic similarity of Rada, that is, Rada

Details

This function computes either all the semantic similarities implemented in the package or the one indicated.

Value

The resulting object is an data.frame where for each pair of different terms (rows) are shown the numeric values of the semantic similarities computed (columns)

Author(s)

Jose Luis Mosquera

See Also

ancestors, simRada, simRes.eb, sims.nb, pseudoDists

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)

sims.eb(x = joslyn.OOC, root = "R", at, method = "Rada")
sims.eb(x = joslyn.OOC, root = "R", at, method = "Res.eb")
sims.eb(x = joslyn.OOC, root = "R", at, method = "all")
```

`sims.nb` *Wrapper function that calls different methods for computing semantic similarities based on node-based approaches*

Description

It is a wrapper function that calls different methods for computing semantic similarities based on node-based approaches

Usage

```
sims.nb(at, ic, method = "Res")
```

Arguments

`at`	`list` of `character` vectors with the ancestors of each term
`ic`	`numeric` vector with the IC of each term in the ontology
`method`	`character` indicating the method for computing the semantic similarity.The options implemented in `sims` package are `Res` semantic similarty of Resnik `Lin` semantic similarty of Lin `Rel` semantic similarty of Schlicker *et al.* `JC` semantic similarty of Jiang and Conrath `Nunivers` semantic similarty of Mazandu and Mulder `Psec` semantic similarty of Pirro and Seco `Faith` semantic similarty of Pirro and Euzenat `all` all sementic similarities By default computes the semantic similarity of Resnik, that is, `Res`

Details

This function computes either all the semantic similarities implemented in the packages or the one indicated.

Value

The resulting object is an `data.frame` where for each pair of different terms (rows) are shown the `numeric` values of the semantic similarities computed (columns)

Author(s)

Jose Luis Mosquera

See Also

ancestors, simRes, simLin, simRel, simJC, simNunivers, simPsec, simFaith, summaryMICA, sims.eb, pseudoDists,resnikSummary

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]

sims.nb(at, ic, method = "Res")     # Renik
sims.nb(at, ic, method = "Lin")     # Lin
sims.nb(at, ic, method = "Rel")     # Schlicker et al.
sims.nb(at, ic, method = "JC")      # Jiang and Conrath
sims.nb(at, ic, method = "Nuviers") # Mazandu and Mulder
sims.nb(at, ic, method = "Psec")    # Pirro & Seco
sims.nb(at, ic, method = "Faith")   # Pirro & Euzenat

sims.nb(at, ic, method = "all")     # All semantic similarities
```

---

simsBetweenGOIDs *Wrapper function that calls different approaches and methods for computing semantic similarities between GO ID ancestors of a list of GO ID's*

Description

Given a list of GO ID's, a GO ontology domain, a type of approach for computing semantic similarities and a specific method, the function calculates the semantic similarities between all the pairs of GO ID ancestors of the induced graph.

Usage

```
simsBetweenGOIDs(goids, ontology = "BP", type = "nb", method = "Res")
```

Arguments

goids	character vector with the GO IDs.
ontology	character with the ontology domain (see *Details*). By default is BP
type	character indicating the approach for computing semantic similarities. By default nb
method	character indicating the method used for computing semantic similarities. By default Res

Details

There are three possibilities, of course, in argument `ontology`

`BP` Biological Processes

`CC` Cellular Components

`MF` Molecular Functions

There are three possibilities in argument `type`

`nb` node-based approach

`eb` edge-based approach

`eb.pd` edge-based approach based on psedudo-distances

According to the approach selected in argument `type`, there are different `methods` for computing semantic similarities.

- Node-based approach (i.e. `type = nb`)
  `Res` semantic similarty of Resnik
  `Lin` semantic similarty of Lin
  `Rel` semantic similarty of Schlicker *et al.*
  `JC` semantic similarty of Jiang and Conrath
  `Nunivers` semantic similarty of Mazandu and Mulder
  `Psec` semantic similarty of Pirro and Seco
  `Faith` semantic similarty of Pirro and Euzenat
  `all` all sementic similarities
- Edge-based approach (i.e. `type = eb`)
  `Rada` semantic similarty of Rada
  `Res.eb` semantic similarty of Resnik based on the shortest path (see function `distRada`
  `all` all semantic similarities of edge-based approaches implemented in the package
- Edge-based approach pseudo-distance (i.e. `type = eb.pd`)
  `hm` pseudo-distance of the minimum chain length
  `hx` pseudo-distance of the maximum chain length
  `hax` pseudo-distance of the average of extreme chain lengths
  `hap` pseudo-distance of the average of all chain lengths
  `all` all pseudo-distances

Author(s)

Jose Luis Mosquera

See Also

`gosims`, `gosimsAvsB`

Examples

```
goids <- c("GO:0004022","GO:0005515")
a <- simsBetweenGOIDs(goids, ontology = "MF", type = "nb", method = "Res")
```

simsMat *Coerces a 1-column* data.frame *resulting from semantic similarity functions to be an object of class* dist.

Description

Given a data.frame with one column resulting from one of the functions for computing semantic similarities, builds a matrix of class dist

Usage

```
simsMat(x)
```

Arguments

x	data.frame with one column resulting from one of the functions for computing semantic similarities

Value

The resulting object is of class dist

Author(s)

Jose Luis Mosquera

See Also

toMat, sims.nb, sims.eb, pseudoDists, gosims

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]

sims.Res <- sims.nb(at, ic, method = "Res")
simsMat(sims.Res)
```

summaryMICA *Computes for each pair of terms the Information Content (IC) of each term the Most Informative Common Ancestor (MICA), and the subsumer associated with the MICA*

Description

Builds a `data.frame` that for each pair of terms gives the Information Content (IC) of each term the Most Informative Common Ancestor (MICA), and the subsumer associated with the MICA

Usage

```
summaryMICA(at, ic)
```

Arguments

`at`	list whose elements are the ancestors of each term
`ic`	numeric vector with the IC of each term in the ontology

Author(s)

Jose Luis Mosquera

See Also

`ancestors`, `commonAncestors`, `inverseIminusG`, `getA`, `resnikSummary`

Examples

```
data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
ca <- commonAncestors(at)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]

sum.mica <- summaryMICA(at, ic)
print(sum.mica)
```

summaryPaths *Lengths of the chains (in terms of depth) or number of paths between each pair of terms.*

Description

Builds a list of numeric vectors with either the lengths of the chains (in terms of depth) or the number of paths between each pair of terms.

Usage

```
summaryPaths(x, root = NULL, len = TRUE)
```

Arguments

x	OOC object
root	character indicating the name of the root term of the ontology in the OOC object
len	logical value. If TRUE, computes the length (depth) of the chain. If FALSE, computes the number of chains. By default is TRUE

Author(s)

Jose Luis Mosquera

See Also

OOC

Examples

```
data(joslyn)

summaryPaths(x = joslyn.OOC, root = "R", len = TRUE)         # Length (depth) of chains
summaryPaths(x = joslyn.OOC, root = "R", len = FALSE)        # Number of chains
```

summarySims *Summary of semantic similarity estimates between each pair of terms and measure*

Description

For estimates of each measure (columns) provided on matrix, this function builds a data.frame with the number of pairs, the number of NA's, the minimum value and the number of minimum values, the maximum value and the number of minimum values, the mean, the standard deviation and the median

Usage

```
summarySims(x)
```

Arguments

x	matrix with the semantic similarity estimates

Details

If any value of the semantic similarities is Inf, then it is converted to an NA value, and it is removed from the computation of the statistics

Value

The resulting object is a data.frame that for each measure (rows) shows the estimates of statistics provided (columns)

Author(s)

Jose Luis Mosquera

See Also

ancestors, inverseIminusG, getA, resnikSummary, sims, sims.eb, pseudoDists, gosims, gosimsAvsB, summarySimsAvsB

Examples

```
## An arbitrary OOC

data(joslyn)

inv.IminusG <- inverseIminusG(joslyn.OOC)
A.mat <- getA(inv.IminusG)
at <- ancestors(A.mat)
resnik.sum <- resnikSummary(x = joslyn.OOC)
ic <- resnik.sum[, "ic"]

sims.all <- sims.nb(at, ic, method = "all")
summarySims(sims.all)

## An OOC associated with the GO

data(prostateIds)               # Data set from the package goProfiles

eg.we <- welsh01EntrezIDs[1:10]   # Entrez Gene IS signature Welsh 01

all.nb <- gosims(eg = eg.we, ontology = "MF", pckg = "org.Hs.eg.db", type = "nb", method = "all")

summarySims(as.matrix(all.nb))
```

summarySimsAvsB *Summary of a two-columns* matrix *with semantic similarity estimates between each pair of terms for the same measure*

Description

This function provides a summary of estimates calculated with the same semantic similarity measure for two lists of objects (i.e. genes). The summary consists of a list with three elements: first, a data.frame with basic statistics for each column, second, the Pearson's Correlation and the correlation test associated with, and three, the Cosine Similarity Measure

Usage

```
summarySimsAvsB(x)
```

Arguments

x	a matrix with two columns where for each pair of terms (rows) contains the estimates of the semantic similarity measure for each list of objects (columns)

Details

If any value of the semantic similarities is Inf, then it is converted to an NA value, and it is removed from the computation of the statistics.

This function can be used with an arbitrary OOC object, but is particularly thought for using Entrex Gene Identifiers (objects) and GO Identifiers (terms).

Author(s)

Jose Luis Mosquera

See Also

summarySims, sims, sims.eb, pseudoDists, gosims, gosimsAvsB, summarySims, cosSim

Examples

```
data(prostateIds)               # Data set from the package goProfiles

eg.we <- welsh01EntrezIDs[1:10]   # Entrez Gene IS signature Welsh 01
eg.sg <- singh01EntrezIDs[1:10]   # Entrez Gene IS signature Singh 01

WEvsSG.nb <- gosimsAvsB(eg1 = eg.we, eg2 = eg.sg, ontology = "MF", pckg = "org.Hs.eg.db", type = "nb", method = "Res'

summarySimsAvsB(WEvsSG.nb)
```

termPairs *Builds the pairs of different terms or characters*

Description

Builds a `character` vector whose elements are the pairs of different terms or characters

Usage

```
termPairs(x)
```

Arguments

`x`	it can be an object of class `OOC` or `character` vector with the names of each term

Details

Given a `character` vector, builds all the pairs of different elements in the vector

Author(s)

Jose Luis Mosquera

See Also

`OOC`

Examples

```
termPairs(letters[1:5])

data(joslyn)
termPairs(joslyn.OOC)
```

toMat *Builds a* `matrix` *of zero and one elements such that zero indicates there is no a relation between row and column, and one there is a relation.*

Description

Given two `character` vectors and a 2-columns `data.frame`, that relates an element of the first `character` vector with an element of the second `character` vector, builds a `matrix` whose `rownames` are the elements of the first `character` vector, `colnames` are the elements of the second `character` vector, and elements are 0 and 1. The values indicate that there is no a relation between a row and a column, when the value is zero, and there is a relation between a row and a column, when the value is one

Usage

```
toMat(df, rnames, cnames)
```

Arguments

df	data.frame with mapping from the Entrez Gene IDs to the GO ID terms
rnames	character indicating which ontologies is selected (see *Details*)
cnames	character indicating which ontologies is selected (see *Details*)

Value

The resulting object is a matrix

Author(s)

Jose Luis Mosquera

See Also

toPairs, mappingMatrix, refinementMatrix

Examples

```
vocabulary <- c("R", "B", "C", "K", "F", "G", "I", "E", "J", "H", "A", "D")
object.ids <- letters[1:10]

origin <- c("B", "C", "K", "F", "G", "I", "I", "E", "J", "E", "J", "A", "A",
            "E", "H", "D", "D", "A")
terminus <- c("R", "R", "R", "B", "B", "B", "C", "C", "C", "K", "K", "F", "G",
              "I", "I", "E", "J", "H")
links <- data.frame(origin, terminus)

mat.g <- toMat(df = links, rnames = vocabulary, cnames = vocabulary)
```

---

toOOC *Builds an Object-Ontology Complex (OOC)*

Description

Builds a object of class OOC that is used as a container of an Object-Ontology Object (OOC)

Usage

```
toOOC(T, G, O, M)
```

Arguments

T	character vector with the names of the terms in the ontology
G	matrix encoding the refinement matrix associated with the DAG of the ontology. Elements of this matrix are 1, when a term on the rows refines a term on the columns, and 0 otherwise
O	character vector with the object identifiers (i.e. features, genes,...) mapping the terms of the ontology
M	matrix encoding the mapping from the set of object identifiers (listed in the argument O to the terms listed in the argument T

Value

The resulting object is an OOC

Author(s)

Jose Luis Mosquera

See Also

OOC

Examples

```
## Terms in the ontology

terms <- c("R", "B", "C", "K", "F", "G", "I", "E", "J", "H", "A", "D")

## Object identifiers

object.ids <- letters[1:10]

## Matrix of refinements

G <- matrix(c(0,0,0,0,0,0,0,0,0,0,0,0,
              1,0,0,0,0,0,0,0,0,0,0,0,
              1,0,0,0,0,0,0,0,0,0,0,0,
              1,0,0,0,0,0,0,0,0,0,0,0,
              0,1,0,0,0,0,0,0,0,0,0,0,
              0,1,0,0,0,0,0,0,0,0,0,0,
              0,1,1,0,0,0,0,0,0,0,0,0,
              0,0,1,1,0,0,1,0,0,0,0,0,
              0,0,1,1,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,1,0,0,0,0,0,
              0,0,0,0,1,1,0,0,0,1,0,0,
              0,0,0,0,0,0,0,1,1,0,0,0),
             nrow = 12, ncol = 12, byrow = TRUE)
colnames(G) <-rownames(G) <- terms

## Mapping matrix
```

```
M <- matrix(c(0,0,0,0,0,0,0,0,0,0,1,0,
              0,0,0,0,1,0,0,1,0,0,1,0,
              0,0,0,0,0,0,0,0,0,0,1,0,
              0,0,0,0,1,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,1,0,0,
              0,0,0,0,0,0,1,0,0,0,0,0,
              0,0,0,0,0,0,0,0,1,0,0,0,
              0,0,0,0,0,0,0,0,1,0,0,0,
              0,0,0,0,0,0,0,0,1,0,0,0,
              0,0,0,0,0,0,0,0,0,0,0,1),
             nrow = 10, ncol = 12, byrow = TRUE)
colnames(M) <- terms
rownames(M) <- object.ids

## Object-Ontology Complex

joslyn.OOC <- toOOC(T = terms, G = G, O = object.ids, M = M)
```

`toPairs` *Builds a 2-columns* `data.frame` *relating elements of a matrix with value one.*

Description

Given a matrix of zero's and one's, builds a 2-columns `data.frame` where on the first columns are the names of the rows and on the second column the names of columns for those elements of the matrix with value one.

Usage

```
toPairs(mat)
```

Arguments

`mat`	`matrix` with 0's and 1's

Value

The resulting object is a 2-columns `data.frame`

Author(s)

Jose Luis Mosquera

See Also

`toMat`, `mappingMatrix`, `refinementMatrix`

Examples

```
a <- sample(0:1, 20, replace = TRUE)
A <- matrix(a, nrow = 5)
colnames(A) <- LETTERS[1:4]
rownames(A) <- letters[1:5]

toPairs(A)
```

Index