Package ‘BioCor’

April 11, 2018

Title Functional similarities

Version 1.2.1

Author Lluís Revilla Sancho <lluis.revilla@gmail.com>

Maintainer Lluís Revilla Sancho <lluis.revilla@gmail.com>

Description Calculates functional similarities based on the pathways described on KEGG and REACTOME or in gene sets. These similarities can be calculated for pathways or gene sets, genes, or clusters and combined with other similarities. They can be used to improve networks, gene selection, testing relationships...

Depends R (>= 3.4.0)

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData true

biocViews StatisticalMethod, Clustering, GeneExpression, Network, Pathways, NetworkEnrichment, SystemsBiology

Imports BiocParallel, Matrix

Suggests reactome.db, org.Hs.eg.db, WGCNA, GOSemSim, GSEABase, testthat, knitr, rmarkdown, BiocStyle, airway, DESeq2, boot, targetscan.Hs.eg.db, Hmisc,

BugReports https://github.com/llrs/BioCor/issues

URL https://github.com/llrs/BioCor/

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

BioCor-package 2
addSimilarities 2
AintoB 3
clusterGeneSim 4
clusterSim 5
combinadic 6

combineScores 7
combineSources 9
conversions 10
diceSim 10
duplicateIndices 11
geneSim 12
pathSim 13
removeDup 14
seq2mat 15
similarities 16
weighted 17

Index **19**

`BioCor-package` *BioCor: A package to calculate functional similarities*

Description

Calculates a functional similarity measure between gene identifiers based on the pathways described on KEGG and REACTOME.

Important functions

`pathSim` Calculates the similarity between two pathways

`geneSim` Calculates the similarity (based on pathSim) between two genes

`clusterSim` Calculates the similarity between two clusters of genes by joining pathways of each gene.

`clusterGeneSim` Calculates the similarity between two clusters of genes by comparing the similarity between the genes of a cluster

`similarities` Allows to combine the value of matrices of similarities

`conversions` Two functions to convert similarity measures

`weighted` Functions provided to combine similarities

`addSimilarities` *Additive integration of similarities*

Description

Function that use the previously calculated similarities into a single similarity matrix.

Usage

```
addSimilarities(x, bio_mat, weights = c(0.5, 0.18, 0.1, 0.22))
```

Arguments

`x`	A matrix with the similarity of expression
`bio_mat`	A list of matrices of the same dimension as x.
`weights`	A numeric vector of weight to multiply each similarity

Details

The total weight can't be higher than 1 to prevent values above 1 but can be below 1. It uses weighted.sum with abs = TRUE internally.

Value

A square matrix of the same dimensions as the input matrices.

Author(s)

Lluís Revilla

See Also

`similarities`, `weighted`.

Examples

```
set.seed(100)
a <- seq2mat(LETTERS[1:5], rnorm(10))
b <- seq2mat(LETTERS[1:5], seq(from = 0.1, to = 1, by = 0.1))
sim <- list(b)
addSimilarities(a, sim, c(0.5, 0.5))
```

`AintoB` *Insert a matrix into another*

Description

Insert values from a matrix into another matrix baded on the rownames and colnames replacing the values.

Usage

```
AintoB(A, B)
```

Arguments

`A`	A matrix to be inserted.
`B`	A matrix to insert in.

Details

If all the genes with pathway information are already calculated but you would like to use more genes when performing analysis. insert the once you have calculated on the matrix of genes.

Value

A matrix with the values of A in the matrix B.

Author(s)

Lluís Revilla

Examples

```
B <- matrix(ncol = 10, nrow = 10,
    dimnames = list(letters[1:10], letters[1:10]))
A <- matrix(c(1:15), byrow=TRUE, nrow=5,
    dimnames = list(letters[1:5], letters[1:3]))
AintoB(A, B)

# Mixed orders
colnames(A) <- c("c", "h", "e")
rownames(A) <- c("b", "a", "f", "c", "j")
AintoB(A, B)

# Missing colums or rows
colnames(A) <- c("d", "f", "k")
AintoB(A, B)
```

---

clusterGeneSim *Similarity score between clusters of genes based on genes similarity*

Description

Looks for the similarity between genes of a group and then between each group.

Usage

```
clusterGeneSim(cluster1, cluster2, info, method = c("max", "rcmax.avg"), ...)

mclusterGeneSim(clusters, info, method = c("max", "rcmax.avg"), ...)
```

Arguments

`cluster1`	A vector with genes.
`cluster2`	A vector with genes.
`info`	A list of genes and the pathways they are involved.
`method`	A vector with two or one argument to be passed to combineScores the first one is used to summarize the similarities of genes, the second one for clusters.
`...`	Other arguments passed to `combineScoresPar`
`clusters`	A list of clusters of genes to be found in `id`.

Details

Differs with clusterGeneSim that first each combination between genes is calculated, and with this values then the comparison between the two clusters is done. Thus applying combineScores twice, one at gene level and another one at cluster level.

Value

`clusterGeneSim` returns a similarity score of the two clusters or the similarity between the genes of the two clusters.

`mclusterGeneSim` returns a matrix with the similarity scores for each cluster comparison.

Author(s)

Lluís Revilla

See Also

`clusterGeneSim`, `combineScores` and `conversions`

Examples

```
if (require("org.Hs.eg.db")) {
    #Extract the paths of all genes of org.Hs.eg.db from KEGG (last update in
    # data of June 31st 2011)
    genes.kegg <- as.list(org.Hs.egPATH)
    clusterGeneSim(c("18", "81", "10"), c("100", "10", "1"), genes.kegg)
    clusterGeneSim(c("18", "81", "10"), c("100", "10", "1"), genes.kegg,
                   c("avg", "avg"))
    clusterGeneSim(c("18", "81", "10"), c("100", "10", "1"), genes.kegg,
                   c("avg", "rcmax.avg"))
    (clus <- clusterGeneSim(c("18", "81", "10"), c("100", "10", "1"), genes.kegg,
                            "avg"))
    combineScores(clus, "rcmax.avg")
} else {
    warning('You need org.Hs.eg.db package for this example')
}

clusters <- list(cluster1 = c("18", "81", "10"),
                 cluster2 = c("100", "594", "836"),
                 cluster3 = c("18", "10", "83"))
mclusterGeneSim(clusters, genes.kegg)
mclusterGeneSim(clusters, genes.kegg, c("max", "avg"))
mclusterGeneSim(clusters, genes.kegg, c("max", "BMA"))
```

`clusterSim` *Similarity score between clusters of genes based on pathways similarity*

Description

Looks for the similarity between genes in groups

Usage

```
clusterSim(cluster1, cluster2, info, method = "max", ...)

mclusterSim(clusters, info, method = "max", ...)
```

Arguments

`cluster1, cluster2`
A vector with genes.

`info` A list of genes and the pathways they are involved.

`method` To combine the scores of each pathway, one of `c("avg","max", "rcmax", "rcmax.avg", "BMA"),` if NULL returns the matrix of similarities.

`...`	Other arguments passed to `combineScoresPar`
`clusters`	A list of clusters of genes to be found in `id`.

Details

Once the pathways for each cluster are found they are combined using `combineScores`. In `mclusterSim` the function `combineScoresPar` is used instead.

Value

`clusterSim` returns a similarity score of the two clusters

`mclusterSim` returns a matrix with the similarity scores for each cluster comparison.

Author(s)

Lluís Revilla

See Also

For a different approach see `clusterGeneSim`, `combineScores` and `conversions`

Examples

```
if (require("org.Hs.eg.db")) {
#Extract the paths of all genes of org.Hs.eg.db from KEGG (last update in
# data of June 31st 2011)
genes.kegg <- as.list(org.Hs.egPATH)
clusterSim(c("9", "15", "10"), c("33", "19", "20"), genes.kegg)
clusterSim(c("9", "15", "10"), c("33", "19", "20"), genes.kegg, NULL)
clusterSim(c("9", "15", "10"), c("33", "19", "20"), genes.kegg, "avg")
} else {
warning('You need org.Hs.eg.db package for this example')
}

clusters <- list(cluster1 = c("18", "81", "10"),
                 cluster2 = c("100", "10", "1"),
                 cluster3 = c("18", "10", "83"))
mclusterSim(clusters, genes.kegg)
mclusterSim(clusters, genes.kegg, "avg")
```

`combinadic` *i-th combination of n elements taken from r*

Description

Function similar to combn but for larger vectors. To avoid allocating a big vector with all the combinations each one can be computed with this function.

Usage

```
combinadic(n, r, i)
```

Arguments

`n`	Elements to extract the combination from
`r`	Number of elements per combination
`i`	ith combination

Value

The combination ith of the elements

Author(s)

Joshua Ulrich

References

StackOverflow answer 4494469/2886003

See Also

`combn`

Examples

```
#Output of all combinations
combn(LETTERS[1:5], 2)
# Otuput of the second combination
combinadic(LETTERS[1:5], 2, 2)
```

`combineScores` *Combining values*

Description

Combine several values into one by several methods.

`cobmineScoresPar` performs multiple (parallel) combineScores based on a list of elements to combine into one.

Usage

```
combineScores(scores, method, round = FALSE, t = 0)

combineScoresPar(scores, method, subSets = NULL, BPPARAM = NULL, ...)
```

Arguments

`scores` Matrix of scores to be combined

`method` one of `c("avg", "max", "rcmax", "rcmax.avg", "BMA","reciprocal")`, see Details.

`round` Should the resulting value be rounded to the third digit?

`t` Numeric value to filter scores below this value. Only used in the reciprocal method.

`subSets` List of combinations as info in other functions.

`BPPARAM` Determining the parallel back-end. By default a for loop is used.

`...` Other arguments passed to `combineScoresPar`

Details

The input matrix can be a base matrix or a matrix from package Matrix. The methods return:

avg The average or mean value

max The max value

rcmax The max of the column means or row means

rcmax.avg The sum of the max values by rows and columns divided by the number of columns and rows

BMA The same as `rcmax.avg`

reciprocal The double of the sum of the reciprocal maximal similarities (above a threshold) divided by the number of elements. See equation 3 of the Tao *et al* 2007 article

Value

A numeric value as described in details.

Note

`combineScores` is a version of the function of the same name in package GOSemSim (`combineScores`) with optional rounding and some internal differences.

Author(s)

Lluís Revilla based on Guangchuang Yu

References

Ying Tao, Lee Sam, Jianrong Li, Carol Friedman, Yves A. Lussier; Information theory applied to the sparse gene ontology annotation network to predict novel gene function. Bioinformatics 2007; 23 (13): i529-i538. doi: 10.1093/bioinformatics/btm195

See Also

`bpparam`

Examples

```
(d <- structure(c(0.4, 0.6, 0.222222222222222, 0.4, 0.4, 0, 0.25, 0.5,
                 0.285714285714286), .Dim = c(3L, 3L),
                .Dimnames = list(c("a", "b", "c"), c("d", "e", "f"))))
e <- d
sapply(c("avg", "max", "rcmax", "rcmax.avg", "BMA", "reciprocal"),
       combineScores, scores = d)
d[1,2] <- NA
sapply(c("avg", "max", "rcmax", "rcmax.avg", "BMA", "reciprocal"),
       combineScores, scores = d)
colnames(e) <- rownames(e)
combineScoresPar(e, list(a= c("a", "b"), b = c("b", "c")),
                 method = "max")
```

combineSources *Combine different sources of pathways*

Description

Given several sources of pathways with the same for the same id of the genes it merge them.

Usage

```
combineSources(...)
```

Arguments

...	Lists of genes and their pathways.

Details

It assumes that the identifier of the genes are the same for both sources but if many aren't equal it issues a warning. Only unique pathways identifiers are returned.

Value

A single list with the pathways of each source on the same gene.

Examples

```
DB1 <- list(g1 = letters[6:8], g2 = letters[1:5], g3 = letters[4:7])
DB2 <- list(g1 = c("one", "two"), g2 = c("three", "four"),
         g3 = c("another", "two"))
combineSources(DB1, DB2)
combineSources(DB1, DB1)
DB3 <- list(g1 = c("one", "two"), g2 = c("three", "four"),
         g4 = c("five", "six", "seven"), g5 = c("another", "two"))
combineSources(DB1, DB3) # A warning is expected
```

conversions *Convert the similarities formats*

Description

Functions to convert the similarity coefficients between Jaccard and Dice. D2J is the opposite of J2D.

Usage

```
D2J(D)

J2D(J)
```

Arguments

`D`	Dice coefficient, as returned by `diceSim`, `geneSim`, `clusterSim` and `clusterGeneSim`
`J`	Jaccard coefficient

Value

A numeric value.

Author(s)

Lluís Revilla

Examples

```
D2J(0.5)
J2D(0.5)
D2J(J2D(0.5))
```

diceSim *Compare pathways*

Description

Function to estimate how much two list of genes overlap by looking how much of the nodes are shared. Calculates the Dice similarity

Usage

```
diceSim(g1, g2)
```

Arguments

`g1, g2`	A character list with the names of the proteins in each pathway.

Details

It requires a vector of characters otherwise will return an `NA`.

Value

A score between 0 and 1 calculated as the doble of the proteins shared by g1 and g2 divided by the number of genes in both groups.

Author(s)

Lluís Revilla

See Also

Used for `geneSim`, see `conversions` help page to transform Dice score to Jaccard score.

Examples

```
genes.id2 <- c("52", "11342", "80895", "57654", "548953", "11586", "45985")
genes.id1 <- c("52", "11342", "80895", "57654", "58493", "1164", "1163",
"4150", "2130", "159")
diceSim(genes.id1, genes.id2)
diceSim(genes.id2, genes.id2)
```

`duplicateIndices` *Finds the indices of the duplicated events of a vector*

Description

Finds the indices of duplicated elements in the vector given.

Usage

```
duplicateIndices(vec)
```

Arguments

`vec`	Vector of identifiers presumably duplicated

Details

For each duplication it can return a list or if all the duplication events are of the same length it returns a matrix, where each column is duplicated.

Value

The format is determined by the simplify2array

Author(s)

Lluís Revilla

See Also

removeDup

Examples

```
duplicateIndices(c("52", "52", "53", "55")) # One repeated element
duplicateIndices(c("52", "52", "53", "55", "55")) # Repeated elements
duplicateIndices(c("52", "55", "53", "55", "52")) # Mixed repeated elements
```

geneSim *Similarity score genes based on pathways similarity*

Description

Given two genes, calculates the Dice similarity between each pathway which is combined to obtain a similarity between the genes.

Usage

```
geneSim(gene1, gene2, info, method = "max", ...)

mgeneSim(genes, info, method = "max", ...)
```

Arguments

`gene1, gene2`	Ids of the genes to calculate the similarity, to be found in genes.
`info`	A list of genes and the pathways they are involved.
`method`	To combine the scores of each pathway, one of `c("avg","max", "rcmax", "rcmax.avg", "BMA")`, if NULL returns the matrix of similarities.
`...`	Other arguments passed to `combineScoresPar`
`genes`	A vector of genes.

Details

Given the information about the genes and their pathways, uses the ids of the genes to find the Dice similarity score for each pathway comparison between the genes. Later this similarities are combined using `combineScoresPar`.

Value

The highest Dice score of all the combinations of pathways between the two ids compared if a method to combine scores is provided or NA if there isn't information for one gene. If an `NA` is returned this means that there isn't information available for any pathways for one of the genes. Otherwise a number between 0 and 1 (both included) is returned. Note that there isn't a negative value of similarity.

`mgeneSim` returns the matrix of similarities between the genes in the vector

Note

genes accept named characters and the output will use the names of the genes.

Author(s)

Lluis Revilla

See Also

`conversions` help page to transform Dice score to Jaccard score. For the method to combine the scores see `combineScoresPar`.

Examples

```
if (require("org.Hs.eg.db") & require("reactome.db")) {
    # Extract the paths of all genes of org.Hs.eg.db from KEGG
    # (last update in data of June 31st 2011)
    genes.kegg <- as.list(org.Hs.egPATH)
    # Extracts the paths of all genes of org.Hs.eg.db from reactome
    genes.react <- as.list(reactomeEXTID2PATHID)
    geneSim("81", "18", genes.react)
    geneSim("81", "18", genes.kegg)
    geneSim("81", "18", genes.react, NULL)
    geneSim("81", "18", genes.kegg, NULL)
} else {
    warning('You need reactome.db and org.Hs.eg.db package for this example')
}

mgeneSim(c("81", "18", "10"), genes.react)
mgeneSim(c("81", "18", "10"), genes.react, "avg")
named_genes <- structure(c("81", "18", "10"),
.Names = c("ACTN4", "ABAT", "NAT2"))
mgeneSim(named_genes, genes.react, "max")
```

`pathSim` *Calculates the Dice similarity between pathways*

Description

Calculates the similarity between pathways using dice similarity score.

Usage

```
pathSim(pathway1, pathway2, info)

mpathSim(pathways, info, method = NULL, ...)
```

Arguments

`pathway1, pathway2`
A single pathway to calculate the similarity

`info` A list of genes and the pathways they are involved.

`pathways` Pathways to calculate the similarity for

`method` To combine the scores of each pathway, one of `c("avg","max", "rcmax", "rcmax.avg", "BMA")`, if NULL returns the matrix of similarities.

`...` Other arguments passed to `combineScoresPar`

Details

diceSim is used to calculate similarities between the two pathways.

mpathSim compares the similarity between several pathways and can use combineScoresPar to extract the similarity between those pathways. If one needs the matrix of similarities between pathways set the argument methods to NULL.

Value

The similarity between those pathways or all the similarities between each comparison.

Note

pathways accept named characters, and then the output will have the names

Author(s)

Lluís Revilla

See Also

diceSim and combineScores and conversions help page to transform Dice score to Jaccard score.

Examples

```
if (require("reactome.db")){
    # Extracts the paths of all genes of org.Hs.eg.db from reactome
    genes.react <- as.list(reactomeEXTID2PATHID)
    (paths <- sample(unique(unlist(genes.react)), 2))
    pathSim(paths[1], paths[2], genes.react)

    (pathways <- sample(unique(unlist(genes.react)), 10))
    mpathSim(pathways, genes.react, NULL)
    named_paths <- structure(
        c("R-HSA-112310", "R-HSA-112316", "R-HSA-112315"),
        .Names = c("Neurotransmitter Release Cycle",
                   "Neuronal System",
                   "Transmission across Chemical Synapses"))
    mpathSim(named_paths, genes.react, NULL)
} else {
    warning('You need reactome.db package for this example')
}
```

---

removeDup *Remove duplicated rows and columns*

Description

Given the indices of the duplicated entries remove the columns and rows until just one is left, it keeps the duplicated with the highest absolute mean value.

Usage

```
removeDup(cor_mat, dupli)
```

Arguments

`cor_mat`	List of matrices
`dupli`	List of indicies with duplicated entries

Value

A matrix with only one of the columns and rows duplicated

Author(s)

Lluís Revilla

See Also

`duplicateIndices` to obtain the list of indicies with duplicated entries.

Examples

```
a <- seq2mat(c("52", "52", "53", "55"), runif(choose(4, 2)))
b <- seq2mat(c("52", "52", "53", "55"), runif(choose(4, 2)))
mat <- list("kegg" = a, "react" = b)
mat
dupli <- duplicateIndices(rownames(a))
remat <- removeDup(mat, dupli)
remat
```

`seq2mat` *Transforms a vector to a symmetric matrix*

Description

Fils a matrix of `ncol = length(x)` and `nrow = length(x)` with the values in `dat` and setting the diagonal to 1.

Usage

```
seq2mat(x, dat)
```

Arguments

`x`	names of columns and rows, used to define the size of the matrix
`dat`	Data to fill with the matrix with except the diagonal.

Details

`dat` should be at least `choose(length(x), 2)` of length. It assumes that the data provided comes from using the row and column id to obtain it.

Value

A square matrix with the diagonal set to 1 and `dat` on the upper and lower triangle with the columns ids and row ids from x.

Author(s)

Lluís Revilla

See Also

upper.tri and lower.tri

Examples

```
seq2mat(LETTERS[1:5], 1:10)
seq2mat(LETTERS[1:5], seq(from = 0.1, to = 1, by = 0.1))
```

---

similarities *Apply a function to a list of similarities*

Description

Function to join list of similarities by a function provided by the user.

Usage

```
similarities(sim, func, ...)
```

Arguments

sim	list of similarities to be joined. All similarities must have the same dimensions. The genes are assumed to be in the same order for all the matrices.
func	function to perform on those similarities: prod, sum... It should accept as many arguments as similarities matrices are provided, and should use numbers.
...	Other arguments passed to the function func. Usually na.rm or similar.

Value

A matrix of the size of the similarities

Note

It doesn't check that the columns and rows of the matrices are in the same order or are the same.

Author(s)

Lluís Revilla

See Also

weighted for functions that can be used, and addSimilarities for a wrapper to one of them

Examples

```
set.seed(100)
a <- seq2mat(LETTERS[1:5], rnorm(10))
b <- seq2mat(LETTERS[1:5], seq(from = 0.1, to = 1, by = 0.1))
sim <- list(b, a)
similarities(sim, weighted.prod, c(0.5, 0.5))
# Note the differences in the sign of some values
similarities(sim, weighted.sum, c(0.5, 0.5))
```

---

weighted *Weighted operations*

Description

Calculates the weighted sum or product of x. Each values should have its weight, otherwise it will throw an error.

Usage

```
weighted.sum(x, w, abs = TRUE)

weighted.prod(x, w)
```

Arguments

x	an object containing the values whose weighted operations is to be computed
w	a numerical vector of weights the same length as x giving the weights to use for elements of x.
abs	If any x is negative you want the result negative too?

Details

This functions are thought to be used with similarities. As some similarities might be positive and others negative the argument abs is provided for weighted.sum, assuming that only one similarity will be negative (usually the one comming from exprresion correlation).

Value

weighted.sum returns the sum of the product of x*weights removing all NA values. See parameter abs if there are any negative values.

weighted.prod returns the product of product of x*weights removing all NA values.

Author(s)

Lluís Revilla

See Also

weighted.mean, similarities and addSimilarities

Examples

```
expr <- c(-0.2, 0.3, 0.5, 0.8, 0.1)
weighted.sum(expr, c(0.5, 0.2, 0.1, 0.1, 0.1))
weighted.sum(expr, c(0.5, 0.2, 0.1, 0.2, 0.1), FALSE)
weighted.sum(expr, c(0.4, 0.2, 0.1, 0.2, 0.1))
weighted.sum(expr, c(0.4, 0.2, 0.1, 0.2, 0.1), FALSE)
weighted.sum(expr, c(0.4, 0.2, 0, 0.2, 0.1))
weighted.sum(expr, c(0.5, 0.2, 0, 0.2, 0.1))
# Compared to weighted.prod:
weighted.prod(expr, c(0.5, 0.2, 0.1, 0.1, 0.1))
weighted.prod(expr, c(0.4, 0.2, 0.1, 0.2, 0.1))
weighted.prod(expr, c(0.4, 0.2, 0, 0.2, 0.1))
weighted.prod(expr, c(0.5, 0.2, 0, 0.2, 0.1))
```

Index

addSimilarities, 2, *16, 17*
AintoB, 3

BioCor (BioCor-package), 2
BioCor-package, 2
bpparam, *8*

clusterGeneSim, *2*, 4, *5, 6*, *10*
clusterSim, *2*, 5, *10*
combinadic, 6
combineScores, *5, 6*, 7, *8*, *14*
combineScoresPar, *4*, *6*, *8*, *12–14*
combineScoresPar (combineScores), 7
combineSources, 9
combn, *7*
conversions, *2*, *5, 6*, 10, *11*, *13, 14*

D2J (conversions), 10
diceSim, *10*, 10, *14*
duplicateIndices, 11, *15*

geneSim, *2*, *10, 11*, 12

J2D (conversions), 10

lower.tri, *16*

mclusterGeneSim (clusterGeneSim), 4
mclusterSim (clusterSim), 5
mgeneSim (geneSim), 12
mpathSim (pathSim), 13

pathSim, *2*, 13

removeDup, *12*, 14

seq2mat, 15
similarities, *2, 3*, 16, *17*

upper.tri, *16*

weighted, *2, 3*, *16*, 17
weighted.mean, *17*