Package 'GOSim'

April 12, 2022

Version 1.32.0

Date 2015-04-02

Title Computation of functional similarities between GO terms and gene products; GO enrichment analysis

Author Holger Froehlich <frohlich@bit.uni-bonn.de>

Maintainer Holger Froehlich <frohlich@bit.uni-bonn.de>

Depends GO.db, annotate

Enhances igraph

Imports org.Hs.eg.db, AnnotationDbi, topGO, cluster, flexmix, RBGL, graph, Matrix, corpcor, Rcpp

LinkingTo Rcpp

NeedsCompilation yes

LazyLoad Yes

Description This package implements several functions useful for computing similarities between GO terms and gene products based on their GO annotation. Moreover it allows for computing a GO enrichment analysis

License GPL (>= 2)

biocViews GO, Clustering, Software, Pathways

git_url https://git.bioconductor.org/packages/GOSim

git_branch RELEASE_3_14

git_last_commit 0c5092d

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`calc.diffusion.kernel` *Calculation and loading of diffusion kernel matrices*

Description

Manifold embeddings of gene ontology terms via diffusion kernel techniques. Diffusion kernels are positive semidefinite similarity measures calculated from the graph Laplacian. They are interpreted as the result of a local heat diffusion process along the graph structure.

Usage

```
calc.diffusion.kernel(method="diffKernelLapl", m=7, normalization.method="sqrt", DIR=".")

load.diffusion.kernel(method="diffKernelLapl", DIR=NULL)
```

Arguments

`method`	one of "diffKernelLapl", "diffKernelpower", "diffKernelLLE", "diffKernelexpm"
`m`	(1) Half the power of the transition probability matrix (an integer > 0). (2) an arbitrary positive time constant for the exponential diffusion kernel
`normalization.method`	method to normalize the kernel
`DIR`	directory, where to write ready calculated kernel matrices to and read them from, respectively. If DIR=NULL in function load.diffusion.kernel, the method assumes the kernel matrix to be present in the data directory of GOSim.

Details

The methods argument has to take on one of the following values:

"diffKernelLapl" pseudo inverse of the (unnormalized) graph Laplacian: Takes into account all powers of diffusion and incorporates all paths from one node to another one.

"diffKernelpower" even power of the transition probability matrix: Takes into account local transitions of path length m

"diffKernelLLE" local linear embedding into an Euclidean space: The focus is to preserve local distances to nearest neighbors. The LLE kernel emphasizes short-range interactions between GO terms.

"diffKernelexpm" expm(-mL), where t is a positive constant, L is the (unnormalized) graph Laplacian and expm denotes the matrix exponential. This kernel takes into account all positive integer powers of diffusion, but with an exponential decay of the influence of long-range interactions.

Value

`calc.diffusion.kernel` puts a kernel matrix / similarity matrix named "<method><ontology><organism><evidence levels>.rda" in the defined directoy. It can be used afterwards by calling `load.diffusion.kernel`.

References

Lerman G. & Shaknovich B., Defining Functional Distance using Manifold Embeddings of Gene Ontology Annotations, PNAS, 104(27): 11334 - 11339, 2007

See Also

`load.diffusion.kernel`

`calcICs` *Calculate information contents of GO terms.*

Description

Recalculates the information content of all GO terms.

Usage

```
calcICs(DIR=".")
```

Arguments

`DIR`	directory where to put the resulting files

Details

This functions should only be invoked, if one wants to calculate the information content for GO terms with respect to combinations of evidence codes other than the precomputed ones or, if a new version of the organism annotation packages has been installed. By default the information contents are precomputed using all evidence codes and evidence codes "IMP, IGI, IDA, IEP, IPI" together.

Value

Puts a file named "ICs<ontology><organism><evidence levels>.rda" in directoy DIR. It can be used afterwards by calling `setOntology`.

See Also

`setEvidenceLevel`

Examples

```
setEvidenceLevel("IMP")
setOntology("CC", loadIC=FALSE) # important: setOntology assumes that the IC file already exists. To prevent an err
calcICs()
 # --> this may take some time ...
```

`clusterEvaluation` *Evaluate a given grouping of genes or GO terms.*

Description

Evaluate a given grouping of genes or terms with respect to their GO similarity.

Usage

```
evaluateClustering(clust, Sim)
```

Arguments

`clust`	vector of cluster labels (integer or character) for each gene
`Sim`	similarity matrix

Details

If necessary, more details than the description above

Value

evaluateClustering returns a list with two items:

`clusterstats`	matrix (ncluster x 2) of median within cluster similarities and median absolute deviations
`clustersil`	cluster silhouette values

Author(s)

Holger Froehlich

References

Rousseeuw, P., Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comp. and Applied Mathematics, 1987, 20, 53-6

See Also

getGeneSimPrototypes, getGeneSim, getTermSim, GOenrichment

Examples

```
setOntology("BP")
gomap <- get("gomap",env=GOSimEnv)
allgenes = sample(names(gomap), 1000) # suppose these are all genes
genesOfInterest = sample(allgenes, 20) # suppose these are all genes of interest

sim = getGeneSim(genesOfInterest,verbose=FALSE) # and these are their similarities
hc = hclust(as.dist(1-sim), method="ward") # use them to perform a clustering
plot(hc) # plot the cluster tree
cl = cutree(hc, k=3) # take 3 clusters

if(require(cluster)){
ev = evaluateClustering(cl, sim) # evaluate the clustering
print(ev$clusterstats) # print out some statistics
plot(ev$clustersil,main="") # plot the cluster silhouettes
}

# investigate cluster 1 further
if(require(topGO))
GOenrichment(genesOfInterest[cl == 1], allgenes, cutoff=0.05) # print out what cluster 1 is about
```

---

filterGO *Filter GO.*

Description

Filter out genes from a list not mapping to the actual ontology. Genes not mapping to the currently set ontology ("BP","MF","CC") and not having one of the predefined evidence codes (default is to use all evidence codes) are removed.

Usage

```
filterGO(genelist)
```

Arguments

genelist	character vector of Entrez gene IDs

Value

List with items

"genename"	gene ID
"annotation"	character vector of GO terms mapping to the gene within the actual ontology

Note

The result depends on the currently set ontology. IMPORTANT: The result refers to the GO library that was used to precompute the information contence of GO terms.

Author(s)

Holger Froehlich

See Also

setOntology, setEvidenceLevel, getGOInfo, calcICs

Examples

```
filterGO(c("12345","4559"))
```

---

getAncestors *get list of ALL ancestors associated to each GO term*

Description

Returns the list of all (also indirect) ancestors (= less specific terms) associated to each GO term. The type of relationship between GO terms ("is a" or "part of") is ignored.

Usage

```
getAncestors()
```

Value

List with entry names for each GO term. Each entry contains a character vector with the ancestor GO terms.

Note

The result is computed within the currently set ontology ("BP","MF","CC"). It directly uses the "GO" library to compute the result.

Author(s)

Holger Froehlich

See Also

getOffsprings, getChildren, getParents, setOntology

Examples

```
getAncestors()
```

---

getChildren *Get a list of all direct children of each GO term.*

Description

Returns the list of all direct children (= more specific terms one hierarchy level down) associated to each GO term. The type of relationship between GO terms ("is a" or "part of") is ignored.

Usage

```
getChildren()
```

Value

List with entry names for each GO term. Each entry contains a character vector with the direct children GO terms.

Note

The result is computed within the currently set ontology ("BP","MF","CC"). It directly uses the "GO" library to compute the result.

Author(s)

Holger Froehlich

See Also

getOffsprings, getParents, getAncestors, setOntology

Examples

```
getChildren()
```

`getDisjCommAnc` *Get disjoint common ancestors.*

Description

Returns the GO terms representing the disjoint common ancestors of two GO terms.

Usage

```
getDisjCommAnc(term1, term2)
```

Arguments

`term1`	GO term 1
`term2`	GO term 2

Details

The result is computed within the currently set ontology ("BP","MF","CC").

Value

Character vector of GO terms

Author(s)

Holger Froehlich

References

Couto, F.; Silva, M. & Coutinho, P., Semantic Similarity over the Gene Ontology: Family Correlation and Selecting Disjunctive Ancestors, Conference in Information and Knowledge Management, 2005

See Also

`getTermSim`,`getGOGraph`, `setOntology`

Examples

```
getDisjCommAnc("GO:0006955","GO:0007584")
```

`getGeneFeatures` *Get simple feature vector representation of genes*

Description

Computes feature vectors for list of genes: Each gene is represented by a vector describing the presence/absence of all GO terms. The absence of each GO term is additionally weighted by its information content.

Usage

```
getGeneFeatures(genelist, pca=FALSE, normalization=FALSE, verbose=FALSE)
```

Arguments

`genelist`	character vector of Entrez gene IDs
`pca`	perform PCA on feature vectors to reduce dimensionality
`normalization`	scale the feature vectors to norm 1
`verbose`	print out additional information

Details

The PCA postprocessing determines the principal components that can explain at least 95% of the total variance in the feature space.

Value

matrix with rows being genes and columns being GO terms.

Note

The result depends on the currently set ontology ("BP","MF","CC").

Author(s)

Holger Froehlich

References

M. Mistry, P Pavlidis, Gene Ontology term overlap as a measure of gene functional similarity, BMC Bioinformatics, 9:327, 2008.

See Also

`getGeneSimPrototypes`, `selectPrototypes`, `getGeneSim`, `getTermSim`, `setOntology`

Examples

```
# see selectPrototypes
```

`getGeneFeaturesPrototypes` *Get feature vector representation of genes relative to prototype genes*

Description

Computes feature vectors for list of genes: Each gene is represented by its similarities to predefined prototype genes.

Usage

```
getGeneFeaturesPrototypes(genelist, prototypes = NULL,
                          similarity = "max", similarityTerm = "JiangConrath",
                          pca = TRUE, normalization = TRUE, verbose = FALSE)
```

Arguments

`genelist`	character vector of Entrez gene IDs
`prototypes`	character vector of Entrez gene IDs representing the prototypes
`similarity`	method to calculate the similarity to prototypes
`similarityTerm`	method to compute the GO term similarity
`pca`	perform PCA on feature vectors to reduce dimensionality
`normalization`	scale the feature vectors to norm 1
`verbose`	print out additional information

Details

If no prototypes are passed, the method calls the `selectPrototypes` function with no arguments. Hence, the prototypes in this case are the 250 genes with most known annotations.

The PCA postprocessing determines the principal components that can explain at least 95% of the total variance in the feature space.

The method to calculate the functional similarity of a gene to a certain prototype can be any of those described in `getGeneSim`.

Value

List with items

`"features"`	feature vectors for each gene: n x d data matrix
`"prototypes"`	prototypes (= prinicipal components, if PCA has been performed)

Note

The result depends on the currently set ontology ("BP","MF","CC").

Author(s)

Holger Froehlich

References

[1] H. Froehlich, N. Speer, C. Spieth, and A. Zell, Kernel Based Functional Gene Grouping, Proc. Int. Joint Conf. on Neural Networks (IJCNN), 6886 - 6891, 2006

[2] N. Speer, H. Froehlich, A. Zell, Functional Grouping of Genes Using Spectral Clustering and Gene Ontology, Proc. Int. Joint Conf. on Neural Networks (IJCNN), pp. 298 - 303, 2005

See Also

getGeneSimPrototypes, selectPrototypes, getGeneSim, getTermSim, setOntology

Examples

```
# see selectPrototypes
```

---

getGeneSim *Compute functional similarity for genes*

Description

Calculate the pairwise functional similarities for a list of genes using different strategies.

Usage

```
getGeneSim(genelist1, genelist2=NULL, similarity="funSimMax", similarityTerm="relevance", normalizati
```

Arguments

genelist1	character vector of primary gene IDs according to organism annotation package (see setEvidenceLevel)
genelist2	optional other character vector of primary gene IDs to compare against
similarity	method to calculate the functional similarity between gene products
similarityTerm	method to compute the similarity of GO terms
normalization	normalize similarities yes/no
method	"sqrt": normalize sim(x,y) <- sim(x,y)/sqrt(sim(x,x)*sim(y,y)); "Lin": normalize sim(x,y) <- 2*sim(x,y)/(sim(x,x) + sim(y,y)); "Tanimoto": normalize sim(x,y) <- sim(x,y)/(sim(x,x) + sim(y,y) - sim(x,y)). NOTE: normalization does not have any effect, if term similarity is NOT "relevance" and similarity = "funSimMax", "funSimAvg" or similarity = "OA" and avg=TRUE
avg	standardize the OA kernel by the maximum number of GO terms for both genes
verbose	print out some information

Details

The method to calculate the pairwise functional similarity between gene products can either be:

"max" the maximum similarity between any two GO terms

"mean" the average similarity between any two GO terms

funSimMax the average of best matching GO term similarities. Take the maximum of the scores achieved by assignments of GO terms from gene 1 to gene 2 and vice versa. [2]

funSimAvg the average of best matching GO term similarities. Take the average of the scores achieved by assignments of GO terms from gene 1 to gene 2 and vice versa. [2]

"OA" the optimal assignment (maximally weighted bipartite matching) of GO terms associated to the gene having fewer annotation to the GO terms of the other gene. [1]

"hausdorff" Hausdorff distance between two sets: Let X and Y be the two sets of GO terms associated to two genes. Then $dist(X,Y) = \max\{\sup_{t\in X} \inf_{t'\in Y} d(t,t'), \sup_{t'\in Y} \inf_{t\in X} d(t,t')$ [3]. Since GOSim 1.2.8 we translate the Haussdorff distance into a similarity measure by taking $sim(X,Y) = \exp(-dist(X,Y)$.

"dot" the dot product between feature vectors describing the absence/presence of each GO term. The absence of each GO term is weighted by its information content. Depending on the type of later normalization one can arrive at the cosine similarity (method="sqrt") or at the Tanimoto coefficient (method="Tanimoto").[4]

Value

n x n similarity matrix (n = number of genes)

Note

The result depends on the currently set ontology.

Author(s)

Holger Froehlich

References

[1] H. Froehlich, N. Speer, C. Spieth, A. Zell, Kernel Based Functional Gene Grouping, Proc. Int. Joint Conf. on Neural Networks (IJCNN), 6886 - 6891, 2006.

[2] A. Schlicker, F. Domingues, J. Rahnenfuehrer, T. Lengauer, A new measure for functional similarity of gene products based on Gene Ontology, BMC Bioinformatics, 7, 302, 2006.

[3] A. del Pozo, F. Pazos, A. Valencia, Defining functional distances over Gene Ontology, BMC Bioinformatics, 9:50, 2008.

[4] M. Mistry, P Pavlidis, Gene Ontology term overlap as a measure of gene functional similarity, BMC Bioinformatics, 9:327, 2008.

See Also

getGeneSimPrototypes, getTermSim, setOntology

Examples

```
# see evaluateClustering
```

`getGeneSimPrototypes` *Compute functional similarity of genes with respect to a feature vector representation.*

Description

Computes the pairwise functional similarities for a list of genes: Each gene is represented by a feature vector containing the gene's similarities to predefined prototype genes.

Usage

```
getGeneSimPrototypes(genelist, prototypes = NULL, similarity = "max",
                     similarityTerm = "JiangConrath", pca = TRUE,
                     normalization = TRUE, verbose = FALSE)
```

Arguments

`genelist`	character vector of primary gene IDs according to organism annotation package (see `setEvidenceLevel`)
`prototypes`	character vector of Entrez gene IDs representing the prototypes
`similarity`	method to calculate the similarity to prototypes
`similarityTerm`	method to compute the GO term similarity
`pca`	perform PCA on feature vectors to reduce dimensionality
`normalization`	normalize similarities to [0,1]: sim(x,y) <- 0.5*(sim(x,y)/sqrt(sim(x,x)*sim(y,y)) + 1)
`verbose`	print additional information

Details

The method calls `getGeneFeaturesPrototypes` to calculate the feature vectors. The functional similarity between two genes is essentially given by the dot product between their feature vectors.

Value

List with items

`"similarity"`	n x n similarity matrix (n = number of genes)
`"prototypes"`	prototypes (= prinicipal components, if PCA has been performed)
`"features"`	feature vectors for each gene: n x d data matrix

Note

The result depends on the currently set ontology ("BP","MF","CC").

Author(s)

Holger Froehlich

References

[1] H. Froehlich, N. Speer, C. Spieth, and A. Zell, Kernel Based Functional Gene Grouping, Proc. Int. Joint Conf. on Neural Networks (IJCNN), 6886 - 6891, 2006

[2] N. Speer, H. Froehlich, A. Zell, Functional Grouping of Genes Using Spectral Clustering and Gene Ontology, Proc. Int. Joint Conf. on Neural Networks (IJCNN), pp. 298 - 303, 2005

See Also

getGeneFeaturesPrototypes, selectPrototypes, getGeneSim, getTermSim, setOntology

Examples

```
#\donttest{ may take some time ...
 proto=selectPrototypes(n=5) # --> returns a character vector of 5 genes with the highest number of annotations
 getGeneSimPrototypes(c("207","208"),prototypes=proto, similarityTerm="Resnik")
#}
```

`getGOGraph` *(1) Get GO graph with specified GO terms at its leave; (2) Get GO Graph with GO terms at leaves associated to one or several genes of interest.*

Description

The function getGOGraph returns a graphNEL object representing the GO graph with leaves specified in the argument. The function getGOGraphsGenes returns a set of graphNEL objects. The ith graph object is created by call to getGOGraph with the GO terms associated to gene i. It hence shows for each gene, where its GO terms are located within the GO structure.

Usage

```
getGOGraph(term, prune=Inf)

getGOGraphsGenes(genelist, prune=Inf)
```

Arguments

`term`	character vector of GO terms
`genelist`	character vector of Entrez gene IDs
`prune`	do not show the complete graph, but prune it after the specified number of ancestors

Details

The result is computed within the currently set ontology ("BP","MF","CC").

Value

graphNEL object(s)

Note

directly calls the function `GOGraph` in the "GOstats" library

Author(s)

Holger Froehlich

Examples

```
G=getGOGraph(c("GO:0006955","GO:0007584"))
if(require(igraph)){
g=igraph.from.graphNEL(G)
plot(g, vertex.label=V(g)$name)
Gs = getGOGraphsGenes(c("207","7494"))
g = igraph.from.graphNEL(Gs[[1]])
plot(g, vertex.label=V(g)$name) # plot the first of both GO graphs
}
```

`getGOInfo` *Obtain GO terms and their description for a list of genes.*

Description

Oobtain the GO terms and their description for a list of genes.

Usage

```
getGOInfo(geneIDs)
```

Arguments

`geneIDs`	character vector of primary gene IDs according to organism annotation package (see `setEvidenceLevel`)

Value

List with entry names equal to the gene IDs. Each list contains a sublist with entry names equal to the GO terms associated to the corresponding gene ID. Each entry also contains a description of the GO term, its definition and the ontology ("BP","CC","MF") it belongs to.

Note

The corresponding information is directly extracted from the "GO" library. The result depends on the currently set ontology ("BP","MF","CC"), i.e. only GO terms within the actual ontology are considered. The shown GO information refers to the actually installed GO library.

Author(s)

Holger Froehlich

See Also

setOntology

Examples

```
if(require(annotate)){
 setOntology("BP")
 getGOInfo(c("207","7494"))
}
```

---

getMinimumSubsumer *Compute minimum subsumer of two GO terms.*

Description

Returns the minimum subsumer (i.e. the common ancestor having the maximal information content) of two GO terms

Usage

```
getMinimumSubsumer(term1, term2)
```

Arguments

term1	GO term 1
term2	GO term 2

Details

The result is computed within the currently set ontology ("BP","MF","CC").

Value

GO term representing the minimum subsumer. If there is no minimum subsumer within the currently set GO category (e.g. because one of the GO terms does not exist), the result is the string "NA".

Author(s)

Holger Froehlich

References

P. Resnik, Using Information Content to evaluate semantic similarity in a taxonomy, Proc. 14th Int. Conf. Artificial Intel., 1995

See Also

getTermSim, getGOGraph, setOntology

Examples

```
setOntology("BP")
getMinimumSubsumer("GO:0006955","GO:0007584")
# returns GO:0050896
```

---

getOffsprings *Get all offspring associated with one or more GO term*

Description

Returns the list of all (also indirect) offspring (= more specific terms) associated to each GO term. The type of relationship between GO terms ("is a" or "part of") is ignored.

Usage

```
getOffsprings()
```

Value

List with entry names for each GO term. Each entry contains a character vector with the offspring GO terms.

Note

The result is computed within the currently set ontology ("BP","MF","CC"). It directly uses the "GO" library to compute the result.

Author(s)

Holger Froehlich

See Also

getChildren, getParents, getAncestors, setOntology

Examples

```
getOffsprings()
```

`getParents` *Get direct parents for each GO term.*

Description

Returns the list of all direct parents (= less specific terms one hiearchy level up) associated to each GO term. The type of relationship between GO terms ("is a" or "part of") is ignored.

Usage

```
getParents()
```

Value

List with entry names for each GO term. Each entry contains a character vector with the direct parent GO terms.

Note

The result is computed within the currently set ontology ("BP","MF","CC"). It directly uses the "GO" library to compute the result.

Author(s)

Holger Froehlich

See Also

`getOffsprings`, `getChildren`, `getAncestors`, `setOntology`

Examples

```
getParents()
```

`getTermSim` *Get pairwise GO term similarities.*

Description

Returns the pairwise similarities between GO terms. Different calculation method are implemented.

Usage

```
getTermSim(termlist, method = "relevance", verbose = FALSE)
```

Arguments

`termlist`	character vector of GO terms
`method`	one of the supported methods for GO term similarity (see below)
`verbose`	print out various information or not

Details

Currently the following methods for computing GO term similarities are implemented:

"Resnik" information content of minimum subsumer (ICms) [1], here additionally divided by the maximum information content of all GO terms

"JiangConrath" $1 - \min(1, IC(term1) - 2ICms + IC(term2))$ [2]

"Lin" $\frac{2ICms}{(IC(term1)+IC(term2))}$ [3]

"CoutoEnriched" FuSSiMeg enriched term similarity by Couto et al. [4]. Requires enrichement factors to be set by `setEnrichmentFactors`.

"CoutoResnik" average information content of common disjunctive ancestors of `term1` and `term2` (ICshare) [5]

"CoutoJiangConrath" $1 - \min(1, IC(term1) - 2ICshare + IC(term2))$ [5]

"CoutoLin" $\frac{2ICshare}{(IC(term1)+IC(term2))}$ [5]

"diffKernel" diffusion kernel similarity from a pre-loaded kernel matrix (see `load.diffusion.kernel`). The diffusion kernel is calculated using one of the methods described in [6].

"relevance" sim_Lin * (1 - exp(-ICms)) [7]

"GIC" summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 [8]

Value

n x n matrix (n = number of GO terms) with similarities between GO terms scaled to [0,1]. If a GO term does not exist for the currently set ontology, the similarity is set to "NA".

Note

All calculations use normalized information contents for each GO term. Normalization is achieved by dividing each information content by the maximum information content within the currently set ontology ("BP","MF","CC")

Author(s)

Holger Froehlich

References

[1] P. Resnik, Using Information Content to evaluate semantic similarity in a taxonomy, Proc. 14th Int. Conf. Artificial Intel., 1995

[2] J. Jiang, D. Conrath, Semantic Similarity based on Corpus Statistics and Lexical Taxonomy, Proc. Int. Conf. Research in Comp. Ling., 1998

[3] D. Lin, An Information-Theoretic Definition of Similarity, Proc. 15th Int. Conf. Machine Learning, 1998

[4] F. Couto, M. Silva, P. Coutinho, Implementation of a Functional Semantic Similarity Measure between Gene-Products, DI/FCUL TR 03-29, Department of Informatics, University of Lisbon, 2003

[5] Couto, F.; Silva, M. & Coutinho, P., Semantic Similarity over the Gene Ontology: Family Correlation and Selecting Disjunctive Ancestors, Conference in Information and Knowledge Management, 2005

[6] Lerman G. & Shaknovich B., Defining Functional Distance using Manifold Embeddings of Gene Ontology Annotations, PNAS, 104(27): 11334 - 11339, 2007

[7] A. Schlicker, F. Domingues, J. Rahnenfuehrer, T. Lengauer, A new measure for functional similarity of gene products based on Gene Ontology, BMC Bioinformatics, 7, 302, 2006.

[8] C. Pesquita, D. Faria, H. Bastos, A. Falcao, F. Couto, Evaluating GO-based Semantic Similarity Measures, In: Proc. 10th Annual Bio-Ontologies Meeting 2007, 37 - 40, 2007

See Also

getMinimumSubsumer, getDisjCommAnc, setEnrichmentFactors, setOntology, load.diffusion.kernel

Examples

```
#\donttest{
 setOntology("BP")
 # Lin's method
 getTermSim(c("GO:0006955","GO:0007584"),method="Lin")
# Couto's method combined with Jiang-Conrath distance
getTermSim(c("GO:0006955","GO:0007584"),method="CoutoJiangConrath")

# set enrichment factors
setEnrichmentFactors(alpha=0.1,beta=0.5)
getTermSim(c("GO:0006955","GO:0007584"),method="CoutoEnriched")
#}
```

`GOenrichment` *GO enrichment analysis*

Description

This function performs a GO enrichment analysis using topGO. It combines the two former functions "GOenrichment" and "analyzeCluster".

Usage

```
GOenrichment(genesOfInterest, allgenes, cutoff=0.01, method="elim")
```

Arguments

`genesOfInterest`	character vector of Entrez gene IDs or vector of statistics (p-values, t-statistics, ...) named with entrez gene IDs
`allgenes`	character vector of Entrez gene IDs or vector of statistics named with entrez gene IDs
`cutoff`	significance cutoff for GO enrichment analysis
`method`	topGO method to use

Details

If the parameters 'genesOfInterest' and 'allgenes' are both character vectors of Entrez gene IDs, Fisher's exact test is used. The Kolmogorov-Smirnov test can be used, if a score (e.g. p-value) for each gene is provided. For more details please refer to the topGO vignette.

Value

`GOTerms`	list of significant GO terms and their description
`p.values`	vector of p-values for significant GO terms
`genes`	list of genes associated to each GO term

Author(s)

Holger Froehlich

References

Adrian Alexa, J\"org Rahnenf\"uhrer, Thomas Lengauer: Improved scoring of functional groups from gene expression data by decorrelating GO graph structure, Bioinformatics, 2006, 22(13):1600-1607

See Also

`evaluateClustering`

Examples

```
if(require(org.Hs.eg.db) & require(topGO)){
allgenes = sample(keys(org.Hs.egGO), 1000) # suppose these are all genes
allpvalues = runif(1000) # an these are their pvalues
names(allpvalues) = allgenes
GOenrichment(allpvalues[allpvalues<0.05], allpvalues) # GO enrichment analysis using Kolmogorov-Smirnov test
}
```

IC *Information content of GO terms*

Description

"ICshumanBPall" Information content of GO terms in "biological process" using all evidence codes for human

"ICshumanCCall" Information content of GO terms in "cellular component" using all evidence codes for human

"ICshumanMFall" Information content of GO terms in "molecular function" using all evidence codes for human

Format

A vector of double values

Note

The currently used IC values can be accessed within the GOSimEnv environment.

---

internal *internal functions*

Description

internal functions or data: do not call these functions directly.

Usage

```
various
```

Arguments

various

Value

various

Author(s)

Holger Froehlich

`selectPrototypes` *Heuristic selection of prototypes and dimensionality reduction of feature vectors.*

Description

- Heuristic selection of prototypes
- Dimensionality reduction of feature vectors

Usage

```
selectPrototypes(n = 250, method = "frequency", data = NULL, verbose = FALSE)
```

Arguments

`n`	number of prototypes or maximum number of clusters
`method`	method to select prototypes or to perform subset selection
`data`	data matrix (l x d) of feature vectors (l = number of genes)
`verbose`	print out information

Details

The following heuristics to perform automatic selection of prototypes are implemented:

"frequency" select n genes with highest number of GO annotations in the currently selected ontology

"random" select n genes uniform randomly over all genes with annotations in the currently selected ontology

To perfom dimensionality reduction implemented methods are:

"PCA" dimensionality reduction via principal component analysis; the number of principal components is determined such that at least 95% of total variance in feature space can be explained

"clustering" EM-clustering in feature space

Value

If the function is called to automatically select prototypes, a character vector of gene IDs is returned.

If the function is called to perform dimensionality via PCA, the result is a list with items

If the function is called to perform clustering in feature space, the cluster centers are returned in a l x k matrix (each column is one cluster center). The "flexmix" function in the package "flexmix" is called to perform the clustering. The BIC is used to calculate the optimal number of clusters in the range 2,...,n.

Note

The result depends on the currently set ontology ("BP","MF","CC").

Author(s)

Holger Froehlich

References

[1] H. Froehlich, N. Speer, C. Spieth, and A. Zell, Kernel Based Functional Gene Grouping, Proc. Int. Joint Conf. on Neural Networks (IJCNN), pp. 6886 - 6891, 2006

[2] N. Speer, H. Froehlich, A. Zell, Functional Grouping of Genes Using Spectral Clustering and Gene Ontology, Proc. Int. Joint Conf. on Neural Networks (IJCNN), pp. 298 - 303, 2005

See Also

getGeneFeaturesPrototypes, getGeneSimPrototypes, setOntology

Examples

```
 # takes too much time in the R CMD check
proto=selectPrototypes(n=5) # --> returns a character vector of 5 genes with the highest number of annotations
feat=getGeneFeaturesPrototypes(c("207","7494"),prototypes=proto,pca=FALSE) # --> compute feature vectors
 selectPrototypes(data=feat$features,method="pca") # ... and PCA projection
```

---

setEnrichmentFactors *Set the depth and densitiy enrichment factors for GO term similarity.*

Description

Sets the depth and density enrichment factors for the enriched FuSSiMeg GO term similarity measure by Couto et al.

Usage

```
setEnrichmentFactors(alpha = 0.5, beta = 0.5)
```

Arguments

alpha	depth factor
beta	density factor

Value

none

Note

The enrichment factors are stored internally and are used by the function getTermSim, if one uses the method "CoutoEnriched" to calculate GO term similarities

Author(s)

Holger Froehlich

References

F.Couto,M. Silva, P. Coutinho, Implementation of a Functional Semantic Similarity Measure between Gene-Products, DI/FCUL TR 03-29, Department of Informatics, University of Lisbon, 2003

See Also

getTermSim

Examples

```
#\donttest{
  setEnrichmentFactors(alpha=0.1,beta=0.5)
  getTermSim(c("GO:0006955","GO:0007584"),method="CoutoEnriched")
#}
```

---

setEvidenceLevel *Specifies to use only GO terms with given evidence codes.*

Description

Specifies to use only GO terms with given evidence codes. This, in combination with the specified GO ontology ("BP", "MF", "CC"), influences, how the information content for individual GO terms is calculated.

Usage

```
setEvidenceLevel(evidences = "all", organism=org.Hs.egORGANISM, gomap=org.Hs.egGO)
```

Arguments

evidences	character vector of evidence codes
organism	organism, for which to load a mapping of primary gene IDs to GO terms (see details)
gomap	mapping of primary gene IDs to GO terms to be used (see details)

Details

Each evidence code can be one of:

"IMP" inferred from mutant phenotype

"IGI" inferred from genetic interaction

"IPI" inferred from physical interaction

"ISS" inferred from sequence similarity

"IDA" inferred from direct assay

"IEP" inferred from expression pattern

"IEA" inferred from electronic annotation

"TAS" traceable author statement

"NAS" non-traceable author statement

"ND" no biological data available

"IC" inferred by curator

Gene ids for which no GO associations exist are left out of the environment.

The method retrieves a mapping of primary gene IDs (usually Entrez) to GO terms, restricted by the given evidence codes. This mapping is based on the respective organism annotation packages (e.g. org.Dm.eg.db for fly, org.Hs.eg.db for human, etc.). The user passes the GO mapping and the organism name to the function. Please refer to the annotation packages for further information.

In case there does not exist an annotation package so far, the user can optionally provide its own mapping of primary gene IDs to GO terms instead of using one of the packages mentioned before. The mapping should come in form of a nested list having a format as in the following example (no NAs are allowed):

\$‘11305‘

\$‘11305‘\$‘GO:0006810‘

\$‘11305‘\$‘GO:0006810‘\$GOID \[1\] "GO:0006810"

\$‘11305‘\$‘GO:0006810‘\$Evidence \[1\] "IEA"

\$‘11305‘\$‘GO:0006810‘\$Ontology \[1\] "BP"

\$‘11305‘\$‘GO:0008203‘

\$‘11305‘\$‘GO:0008203‘\$GOID

\[1\] "GO:0008203"

\$‘11305‘\$‘GO:0008203‘\$Evidence \[1\] "ISS"

\$‘11305‘\$‘GO:0008203‘\$Ontology \[1\] "BP"

\$‘11306‘

\$‘11306‘\$‘GO:0006810‘

\$‘11306‘\$‘GO:0006810‘\$GOID \[1\] "GO:0006810"

\$‘11306‘\$‘GO:0006810‘\$Evidence \[1\] "IEA"

\$‘11306‘\$‘GO:0006810‘\$Ontology \[1\] "BP"

\$‘11306‘\$‘GO:0006879‘

\$‘11306‘\$‘GO:0006879‘\$GOID

\[1\] "GO:0006879"

\$‘11306‘\$‘GO:0006879‘\$Evidence \[1\] "IMP"

\$‘11306‘\$‘GO:0006879‘\$Ontology \[1\] "BP"

Value

The mapping is stored in the GOSimEnv environment.

Note

By default all evidence codes are used. If another behavior is wanted, one has to recalculate the information content of all GO terms via `calcICs`. The evidence level influences the behavior of all other functions, especially `filterGO` and `getGOInfo`.

Author(s)

Holger Froehlich

References

<www.geneontology.org>

See Also

`setOntology`, `calcICs`, `filterGO`, `getGOInfo`

Examples

```
setEvidenceLevel("all")
 # the default behavior
```

`setOntology` *Set an ontology as base for subsequent computations.*

Description

Sets the ontology that all subsequent computations are based on and loads the information content of all GO terms within this ontology. At load time of the library the default ontology is "BP". Furtherm, on running this function the environment GOSimEnv is reinitialized, i.e. all global settings or parameters used in the library are reset to their default values.

Usage

```
setOntology(ont = "BP", loadIC=TRUE, DIR=NULL)
```

Arguments

`ont`	the ontology to use ("BP","MF","CC")
`loadIC`	Should the corresponding file with precomputed IC-values be loaded? Default: TRUE. WARNING: If the file is not loaded, no calculations can be performed! This might only be useful, if you want to recalculate IC values.
`DIR`	If not null, load file from this directory. Otherwise the version installed in GOSim's data directory is used.

Details

The following ontologies can be used:

"BP" biological process

"MF" molecular function

"CC" cellular component

Value

none.

Author(s)

Holger Froehlich

Examples

```
# set ontology to "molecular function"

setOntology("MF")
# calculate Resnik similarity of two GO terms within this ontology
getTermSim(c("GO:0004060","GO:0003867"),method="Resnik")
```

Index