Package 'topGO'

November 3, 2025

```
Date 2025-06-23
Description topGO package provides tools for testing GO terms while accounting for the
     topology of the GO graph. Different test statistics and different methods
     for eliminating local similarities and dependencies between GO terms can be
     implemented and applied.
License LGPL
Encoding UTF-8
Depends R (\geq 2.10.0), methods, BiocGenerics (\geq 0.13.6), graph (\geq
     1.14.0), Biobase (>= 2.0.0), GO.db (>= 2.3.0), AnnotationDbi
     (>= 1.7.19), SparseM (>= 0.73)
Imports lattice, matrixStats, DBI
Suggests ALL, hgu95av2.db, hgu133a.db, genefilter, multtest,
     Rgraphviz, globaltest, knitr, BiocStyle, rmarkdown
Collate AllClasses.R topGOmethods.R topGOgraph.R topGOalgo.R
     topGOfunctions.R topGOannotations.R topGOtests.R topGOviz.R
     zzz.R
VignetteBuilder knitr
URL https://github.com/federicomarini/topGO
BugReports https://github.com/federicomarini/topGO/issues
biocViews GeneExpression, Transcriptomics, GeneSetEnrichment, GO,
     Annotation, Pathways, SystemsBiology, Microarray, Sequencing,
     Visualization, Software
git_url https://git.bioconductor.org/packages/topGO
git_branch devel
git_last_commit 35ec429
git_last_commit_date 2025-10-29
Repository Bioconductor 3.23
```

Type Package

Version 2.63.0

Title Enrichment Analysis for Gene Ontology

2 topGO-package

Date/Publication 2025-11-02

Author Adrian Alexa [aut],

Jörg Rahnenführer [aut],

Federico Marini [cre] (ORCID: https://orcid.org/0000-0003-3252-7758)

Maintainer Federico Marini <marinif@uni-mainz.de>

Contents

topGO-package		Enri	chn	nen	t an	aly	sis	for	r C	en	e C	Ont	ole	ogy	,						
Index																					34
	weightCount-class.																				 32
	topGOresult-class .																				 31
	topGOdata-class																				 28
	printGraph-methods																				 26
	parentChild-class .																				 25
	inducedGraph																				
	groupStats-class																				 23
	groupGOTerms																				 22
	GOdata																				 21
	getSigGroups																				 20
	getPvalues																				 19
	geneList																				 18
	Gene set tests statist	ics .																			 17
	elimScore-class																				 16
	elimExpr-class																				 15
	elimCount-class																				 14
	dignostic-methods.					-				-											
	Determines the level																				
	classicScore-class .																				
	classicExpr-class .																				
	classicCount-class.																				
	annFUN																				
	topGO-package																				 2

Description

topGO package provides tools for testing GO terms while accounting for the topology of the GO graph. Different test statistics and different methods for eliminating local similarities and dependencies between GO terms can be implemented and applied.

annFUN 3

Details

Package: topGO Type: Package Version: 1.0

Date: 2006-10-02

License: What license is it under?

TODO: An overview of how to use the package, including the most important functions

Author(s)

Adrian Alexa, J\"org Rahnenf\"uhrer

Maintainer: Adrian Alexa

References

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

See Also

topGOdata-class, groupStats-class, getSigGroups-methods

annFUN

Functions which map gene identifiers to GO terms

Description

These functions are used to compile a list of GO terms such that each element in the list is a character vector containing all the gene identifiers that are mapped to the respective GO term.

Usage

```
annFUN.db(whichOnto, feasibleGenes = NULL, affyLib)
annFUN.org(whichOnto, feasibleGenes = NULL, mapping, ID = "entrez")
annFUN(whichOnto, feasibleGenes = NULL, affyLib)
annFUN.gene2GO(whichOnto, feasibleGenes = NULL, gene2GO)
annFUN.GO2genes(whichOnto, feasibleGenes = NULL, GO2genes)
annFUN.file(whichOnto, feasibleGenes = NULL, file, ...)
readMappings(file, sep = "\t", IDsep = ",")
inverseList(1)
```

4 annFUN

Arguments

whichOnto	character string specifying one of the three GO ontologies, namely: "BP", "MF", "CC" $$
feasibleGenes	character vector containing a subset of gene identifiers. Only these genes will be used to annotate GO terms. Default value is NULL which means that there are no genes filtered.
affyLib	character string containing the name of the Bioconductor annotation package for a specific microarray chip.
gene2G0	named list of character vectors. The list names are genes identifiers. For each gene the character vector contains the GO identifiers it maps to. Only the most specific annotations are required.
G02genes	named list of character vectors. The list names are GO identifiers. For each GO the character vector contains the genes identifiers which are mapped to it. Only the most specific annotations are required.
mapping	character string specifying the name of the Bioconductor package containing the gene mappings for a specific organism. For example: mapping = "org.Hs.eg.db".
ID	character string specifying the gene identifier to use. Currently only the following identifiers can be used: c("entrez", "genbank", "alias", "ensembl", "symbol", "genename", "unigene")
file	character string specifying the file containing the annotations.
	other parameters
sep	the character used to separate the columns in the CSV file
IDsep	the character used to separate the annotated entities
1	a list containing mappings

Details

All these function restrict the GO terms to the ones belonging to the specified ontology and to the genes listed in the feasibleGenes attribute (if not empty).

The function annFUN.db uses the mappings provided in the Bioconductor annotation data packages. For example, if the Affymetrix hgu133a chip it is used, then the user should set affyLib = "hgu133a.db".

The functions annFUN.gene2GO and annFUN.GO2genes are used when the user provide his own annotations either as a gene-to-GOs mapping, either as a GO-to-genes mapping.

The annFUN.org function is using the mappings from the "org.XX.XX" annotation packages. The function supports different gene identifiers.

The annFUN. file function will read the annotations of the type gene2GO or GO2genes from a text file.

Value

A named(GO identifiers) list of character vectors.

annFUN 5

Author(s)

Adrian Alexa

See Also

```
topGOdata-class
```

```
library(hgu133a.db)
set.seed(111)
## generate a gene list and the GO annotations
selGenes <- sample(ls(hgu133aG0), 50)</pre>
gene2GO <- lapply(mget(selGenes, envir = hgu133aGO), names)</pre>
gene2GO[sapply(gene2GO, is.null)] <- NA</pre>
## the annotation for the first three genes
gene2G0[1:3]
## inverting the annotations
G2g <- inverseList(gene2G0)</pre>
## inverting the annotations and selecting an ontology
go2genes <- annFUN.gene2GO(whichOnto = "CC", gene2GO = gene2GO)</pre>
## generate a GO list with the genes annotations
selGO <- sample(ls(hgu133aGO2PROBE), 30)</pre>
GO2gene <- lapply(mget(selGO, envir = hgu133aGO2PROBE), as.character)</pre>
G02gene[1:3]
## select only the GO terms for a specific ontology
go2gene <- annFUN.GO2genes(whichOnto = "CC", GO2gene = GO2gene)</pre>
## Using the org.XX.xx.db annotations
## GO to Symbol mappings (only the BP ontology is used)
xx <- annFUN.org("BP", mapping = "org.Hs.eg.db", ID = "symbol")</pre>
head(xx)
## Not run:
allGenes <- unique(unlist(xx))</pre>
myInterestedGenes <- sample(allGenes, 500)</pre>
geneList <- factor(as.integer(allGenes</pre>
names(geneList) <- allGenes</pre>
```

6 classicCount-class

classicCount-class

Class "classicCount"

Description

This class that extends the virtual class "groupStats" by adding a slot representing the significant members.

Details

This class is used for test statistic based on counts, like Fisher's exact test

Objects from the Class

```
Objects can be created by calls of the form new("classicCount", testStatistic = "function", name = "character", allMembers = "character", groupMembers = "character", sigMembers = "character").
```

Slots

```
significant: Object of class "integer" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
```

Extends

```
Class "groupStats", directly.
```

classicExpr-class 7

Methods

```
contTable signature(object = "classicCount"): ...
initialize signature(.Object = "classicCount"): ...
numSigAll signature(object = "classicCount"): ...
numSigMembers signature(object = "classicCount"): ...
sigAllMembers signature(object = "classicCount"): ...
sigMembers signature(object = "classicCount"): ...
sigMembers signature(object = "classicCount"): ...
```

Author(s)

Adrian Alexa

See Also

classicScore-class, groupStats-class, getSigGroups-methods

Examples

```
##---- Should be DIRECTLY executable !! ----
```

classicExpr-class

Class "classicExpr"

Description

This class that extends the virtual class "groupStats" by adding two slots for accommodating gene expression data.

Objects from the Class

```
Objects can be created by calls of the form new("classicExpr", testStatistic, name, groupMembers, exprDat, pType, ...).
```

Slots

```
eData: Object of class "environment" ~~

pType: Object of class "factor" ~~

name: Object of class "character" ~~

allMembers: Object of class "character" ~~

members: Object of class "character" ~~

testStatistic: Object of class "function" ~~

testStatPar: Object of class "list" ~~
```

8 classicScore-class

Extends

```
Class "groupStats", directly.
```

Methods

```
allMembers<- signature(object = "classicExpr"): ...
emptyExpr signature(object = "classicExpr"): ...
getSigGroups signature(object = "topGOdata", test.stat = "classicExpr"): ...
GOglobalTest signature(object = "classicExpr"): ...
initialize signature(.Object = "classicExpr"): ...
membersExpr signature(object = "classicExpr"): ...
pType<- signature(object = "classicExpr"): ...
pType signature(object = "classicExpr"): ...</pre>
```

Author(s)

Adrian Alexa

See Also

```
classicScore-class, groupStats-class, getSigGroups-methods
```

Examples

```
showClass("classicExpr")
```

classicScore-class

Class "classicScore"

Description

A class that extends the virtual class "groupStats" by adding a slot representing the score of each gene. It is used for tests like Kolmogorov-Smirnov test.

Objects from the Class

Objects can be created by calls of the form new("classicScore", testStatistic, name, allMembers, groupMembers, score, decreasing).

Slots

```
score: Object of class "numeric" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
scoreOrder: Object of class "character" ~~
testStatPar: Object of class "ANY" ~~
```

Extends

Class "groupStats", directly.

Methods

```
allScore Method to obtain the score of all members.
scoreOrder Returns TRUE if the score should be ordered increasing, FALSE otherwise.
membersScore signature(object = "classicScore"): ...
rankMembers signature(object = "classicScore"): ...
score<- signature(object = "classicScore"): ...</pre>
```

Author(s)

Adrian Alexa

See Also

```
classicCount-class, groupStats-class, getSigGroups-methods
```

Examples

```
## define the type of test you want to use
test.stat <- new("classicScore", testStatistic = GOKSTest, name = "KS tests")</pre>
```

```
Determines the levels of a Directed Acyclic Graph (DAG)

*Utility functions to work with Directed Acyclic Graphs (DAG)
```

Description

Basic functions to work with DAGs

Usage

```
buildLevels(dag, root = NULL, leafs2root = TRUE)
getNoOfLevels(graphLevels)
getGraphRoot(dag, leafs2root = TRUE)
reverseArch(dirGraph, useAlgo = "sparse", useWeights = TRUE)
```

Arguments

dag A graphNEL object.

root A character vector specifying the root(s) of the DAG. If not specified the root

node is automatically computed.

leafs2root The leafs2root parameter tell if the graph has edges directed from the leaves to

the root, or vice-versa

graphLevels An object of type list, returned by the buildLevels function.

dirGraph A graphNEL object containing a directed graph.

useAlgo A character string specifying one of the following options c("sparse", "normal").

By default, useAlgo = "sparse", a sparse matrix object is used to transpose the

adjacency matrix. Otherwise a standard R martix is used.

useWeights If weights should be used (if useAlgo = "normal" then the weights are used

anyway)

Details

buildLevels function determines the levels of a Directed Acyclic Graph (DAG). The level of a node is defined as the longest path from the node to the root. The function take constructs a named list containing various information about each nodes level. The root has level 1.

getNoOfLevels - a convenient function to extract the number of levels from the object returned by buildLevels

getGraphRoot finds the root(s) of the DAG

reverseArch - simple function to invert the direction of edges in a DAG. The returned graph is of class graphNEL. It can use either simple matrices or sparse matrices (SparseM library)

Value

buildLevels returns a list containing:

level2nodes Environment where the key is the level number with the value being the nodes

on that level.

nodes2level Environment where the key is the node label (the GO ID) and the value is the

level on which that node lies.

noOfLevels The number of levels noOfNodes The number of nodes

An object of class graphNEL-class is returned.

dignostic-methods 11

Author(s)

Adrian Alexa

See Also

```
topGOdata-class, inducedGraph
```

Examples

```
##--- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##-- or do help(data=index) for the standard data sets.
```

dignostic-methods

Diagnostic functions for topGOdata and topGOresult objects.

Description

The GenTable function generates a summary of the results of the enrichment analysis.

The showGroupDensity function plots the distributions of the gene' scores/ranks inside a GO term.

The printGenes function shows a short summary of the top genes annotated to the specified GO terms.

Usage

```
GenTable(object, ...)
showGroupDensity(object, whichGO, ranks = FALSE, rm.one = TRUE)
printGenes(object, whichTerms, file, ...)
```

Arguments

object an object of class topGOdata.

whichGO terms for which the plot should be generated.

ranks if ranks should be used instead of scores.
rm. one the p-values which are 1 are removed.

whichTerms character vector listing the GO terms for which the summary should be printed.

file character string specifying the file in which the results should be printed.

... Extra arguments for GenTable can be:

... one or more objects of class topGOresult.

12 dignostic-methods

orderBy if more than one topGOresult object is given then orderBy gives the index of which scores will be used to order the resulting table. Can be an integer index or a character vector given the name of the topGOresult object.

ranksOf same as orderBy argument except that this parameter shows the relative ranks of the specified result.

topNodes the number of top GO terms to be included in the table.

numChar the GO term definition will be truncated such that only the first numChar characters are shown.

Extra arguments for printGenes can be:

chip character string containing the name of the Bioconductor annotation package for a microarray chip.

numChar the gene description is trimmed such that it has numChar characters. simplify logical variable affecting how the results are returned. geneCutOff the maximal number of genes shown for each term. pvalCutOff only the genes with a p-value less than pvalCutOff are shown. oneFile if TRUE then a file for each GO term is generated.

Details

GenTable is an easy to use function for summarising the most significant GO terms and the corresponding p-values. The function dispatches for topGOdata and topGOresult objects, and it can take an arbitrary number of the later, making comparison between various results easier.

Note: One needs to type the complete attribute names (the exact name) of this function, like: topNodes = 5, rankOf = "resultFis", etc. This being the price paid for flexibility of specifying different number of topGOdata objects.

The showGroupDensity function analyse the distribution of the gene-wise scores for a specified GO term. The function will show the distribution of the genes in a GO term compared with the complementary set, using a lattice plot.

printGenes The function will generate a table with all the probes annotated to the specified GO term. Various type of identifiers, the gene name and the gene-wise statistics are provided in the table.

One or more GO identifiers can be given to the function using the whichTerms argument. When more than one GO is specified, the function returns a list of data.frames, otherwise only one data.frame is returned.

The function has a argument file which, when specified, will save the results into a file using the CSV format.

For the moment the function will work only when the chip used has an annotation package available in Bioconductor. It will not work with other type of custom annotations.

Value

A data frame or a list of data frames.

dignostic-methods 13

Author(s)

Adrian Alexa

See Also

```
groupStats-class, getSigGroups-methods
```

```
data(GOdata)
## GenTable
## load two topGOresult sample objects: resultFisher and resultKS
data(results.tG0)
## generate the result of Fisher's exact test
sig.tab <- GenTable(GOdata, Fis = resultFisher, topNodes = 20)</pre>
## results of both test
sig.tab <- GenTable(GOdata, resultFisher, resultKS, topNodes = 20)</pre>
## results of both test with specified names
sig.tab <- GenTable(GOdata, Fis = resultFisher, KS = resultKS, topNodes = 20)</pre>
## results of both test with specified names and specified ordering
sig.tab <- GenTable(GOdata, Fis = resultFisher,</pre>
                KS = resultKS, orderBy = "KS",
                ranksOf = "Fis", topNodes = 20)
## showGroupDensity
goID <- "GO:0006091"
print(showGroupDensity(GOdata, goID, ranks = TRUE))
print(showGroupDensity(GOdata, goID, ranks = FALSE, rm.one = FALSE))
## printGenes
## Not run:
library(hgu95av2.db)
goID <- "GO:0006629"
```

14 elimCount-class

```
gt <- printGenes(GOdata, whichTerms = goID, chip = "hgu95av2.db", numChar = 40)
goIDs <- c("GO:0006629", "GO:0007076")
gt <- printGenes(GOdata, whichTerms = goIDs, chip = "hgu95av2.db", pvalCutOff = 0.01)
gt[goIDs[1]]
## End(Not run)</pre>
```

elimCount-class

Classes "elimCount" and "weight01Count"

Description

Classes that extend the "classicCount" class by adding a slot representing the members that need to be removed.

Objects from the Class

Objects can be created by calls of the form new("elimCount", testStatistic, name, allMembers, groupMembers, sigMembers, elim, cutOff, ...).

Slots

```
elim: Object of class "integer" ~~
cutOff: Object of class "numeric" ~~
significant: Object of class "integer" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "list" ~~
```

Extends

```
Class "classicCount", directly. Class "groupStats", by class "classicCount", distance 2.
```

Methods

No methods defined with class "elimCount" in the signature.

Author(s)

Adrian Alexa

See Also

classicScore-class, groupStats-class, getSigGroups-methods

elimExpr-class 15

elimExpr-class

Class "elimExpr"

Description

Classes that extend the "classicExpr" class by adding a slot representing the members that need to be removed.

Details

TODO: Some details here.....

Objects from the Class

Objects can be created by calls of the form new("elimExpr", testStatistic, name, groupMembers, exprDat, pType, elim, cutOff, ...). ~~ describe objects here ~~

Slots

```
cutOff: Object of class "numeric" ~~
elim: Object of class "integer" ~~
eData: Object of class "environment" ~~
pType: Object of class "factor" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "list" ~~
```

Extends

```
Class "weight01Expr", directly. Class "classicExpr", by class "weight01Expr", distance 2. Class "groupStats", by class "weight01Expr", distance 3.
```

Methods

```
cutOff<- signature(object = "elimExpr"): ...
cutOff signature(object = "elimExpr"): ...
getSigGroups signature(object = "topGOdata", test.stat = "elimExpr"): ...
initialize signature(.0bject = "elimExpr"): ...</pre>
```

Author(s)

Adrian Alexa

16 elimScore-class

See Also

```
classicScore-class, groupStats-class, getSigGroups-methods
```

Examples

```
showClass("elimExpr")
```

elimScore-class

Classes "elimScore" and "weight01Score"

Description

Classes that extend the "classicScore" class by adding a slot representing the members that need to be removed.

Details

TODO:

Objects from the Class

Objects can be created by calls of the form new("elimScore", testStatistic, name, allMembers, groupMembers, score, alternative, elim, cutOff, ...). \sim describe objects here \sim

Slots

```
elim: Object of class "integer" ~~
cutOff: Object of class "numeric" ~~
score: Object of class "numeric" ~~
.alternative: Object of class "logical" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "list" ~~
```

Extends

```
Class "classicScore", directly. Class "groupStats", by class "classicScore", distance 2.
```

Methods

No methods defined with class "elimScore" in the signature.

Gene set tests statistics 17

Author(s)

Adrian Alexa

See Also

classicScore-class, groupStats-class, getSigGroups-methods

Examples

```
##---- Should be DIRECTLY executable !! ----
```

Gene set tests statistics

Gene set tests statistics

Description

Methods which implement and run a group test statistic for a class inheriting from groupStats class. See Details section for a description of each method.

Usage

```
GOFisherTest(object)
GOKSTest(object)
GOtTest(object)
GOglobalTest(object)
GOSumTest(object)
GOKSTiesTest(object)
```

Arguments

object

An object of class groupStats or decedent class.

Details

GOFisherTest: implements Fischer's exact test (based on contingency table) for groupStats objects dealing with "counts".

GOKSTest: implements the Kolmogorov-Smirnov test for groupStats objects dealing with gene "scores". This test uses the ks.test function and does not implement the running-sum-statistic test based on permutations.

GOtTest: implements the t-test for groupStats objects dealing with gene "scores". It should be used when the gene scores are t-statistics or any other score following a normal distribution.

GOglobalTest: implement Goeman's globaltest.

Value

All these methods return the p-value computed by the respective test statistic.

18 geneList

Author(s)

Adrian Alexa

See Also

```
groupStats-class, getSigGroups-methods
```

geneList

A toy example of a list of gene identifiers and the respective p-values

Description

The geneList data is compiled from a differential expression analysis of the ALL dataset. It contains just a small number of genes with the corespondent p-values. The information on where to find the GO annotations is stored in the ALL object.

The topDiffGenes function included in this dataset will select the differentially expressed genes, at 0.01 significance level, from geneList.

Usage

```
data(geneList)
```

Source

Generated using the ALL gene expression data. See the "scripts" directory.

```
data(geneList)
## print the object
head(geneList)
length(geneList)
## the number of genes with a p-value less than 0.01
sum(topDiffGenes(geneList))
```

getPvalues 19

getPvalues	Convenient function to compute p-values from a gene expression matrix.

Description

Warping function of "mt.teststat", for computing p-values of a gene expression matrix.

Usage

Arguments

edata Gene expression matrix.
classlabel The phenotype of the data
test Which test statistic to use

alternative The alternative of the test statistic genesID if a subset of genes is provided correction Multiple testing correction procedure

Value

An named numeric vector of p-values.

Author(s)

Adrian Alexa

See Also

```
GOKSTest, groupStats-class, getSigGroups-methods
```

20 getSigGroups

getSigGroups	Interfaces for running the enrichment tests	
--------------	---	--

Description

These function are used for dispatching the specific algorithm for a given topGOdata object and a test statistic.

Usage

```
getSigGroups(object, test.stat, ...)
runTest(object, algorithm, statistic, ...)
whichAlgorithms()
whichTests()
```

Arguments

object	An object of class topGOdata This object contains all data necessary for running the test.
test.stat	An object of class groupStats. This object defines the test statistic.
algorithm	Character string specifying which algorithm to use.
statistic	Character string specifying which test to use.
	Other parameters. In the case of runTest they are used for defining the test statistic

Details

The runTest function can be used only with a predefined set of test statistics and algorithms. The algorithms and the statistical tests which are accessible via the runTest function are shown by the whichAlgorithms() and whichTests() functions.

The runTest function is a warping of the getSigGroups and the initialisation of a groupStats object functions.

...

Value

An object of class topGOresult.

Author(s)

Adrian Alexa

See Also

```
topGOdata-class, groupStats-class, topGOresult-class
```

GOdata 21

Examples

```
## load a sample topGOdata object
data(GOdata)
GOdata
## getSigGroups interface
## define a test statistic
test.stat <- new("classicCount", testStatistic = GOFisherTest, name = "Fisher test")
## perform the test
resultFis <- getSigGroups(GOdata, test.stat)</pre>
resultFis
##################################
## runTest interface
## Enrichment analysis by using the "classic" method and Fisher's exact test
resultFis <- runTest(GOdata, algorithm = "classic", statistic = "fisher")
resultFis
## weight01 is the default algorithm
weight01.fisher <- runTest(GOdata, statistic = "fisher")</pre>
weight01.fisher
## not all combinations are possible!
# weight.ks <- runTest(GOdata, algorithm = "weight", statistic = "t")</pre>
```

GOdata

Sample topGOdata and topGOresult objects

Description

The GOdata contains an instance of a topGOdata object. It can be used to run an enrichment analysis directly.

The resultFisher contains the results of an enrichment analysis.

Usage

data(GOdata)

Source

Generated using the ALL gene expression data. See topGOdata-class for code examples on how-to generate such an object.

22 groupGOTerms

Examples

```
data(GOdata)
## print the object
GOdata

data(results.tGO)
## print the object
resultFisher
```

groupGOTerms

Grouping of GO terms into the three ontologies

Description

This function split the GOTERM environment into three different ontologies. The newly created environments contain each only the terms from one of the following ontologies 'BP', 'CC', 'MF'

Usage

```
groupGOTerms(where)
```

Arguments

where

The the environment where you want to bind the results.

Value

The function returns NULL.

Author(s)

Adrian Alexa

See Also

```
topGOdata-class, GOTERM
```

```
groupGOTerms()
```

groupStats-class 23

groupStats-class

Class "groupStats"

Description

A virtual class containing basic gene set information: the gene universe, the member of the current group, the test statistic defined for this group, etc.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

```
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "ANY" ~~
```

Methods

```
allMembers<- signature(object = "groupStats"): ...
allMembers signature(object = "groupStats"): ...
initialize signature(.0bject = "groupStats"): ...
members<- signature(object = "groupStats"): ...
members signature(object = "groupStats"): ...
Name<- signature(object = "groupStats"): ...
Name signature(object = "groupStats"): ...
numAllMembers signature(object = "groupStats"): ...
runTest signature(object = "groupStats"): ...
testStatistic signature(object = "groupStats"): ...</pre>
```

Author(s)

Adrian Alexa

See Also

classicCount-class, getSigGroups-methods

24 inducedGraph

inducedGraph

The subgraph induced by a set of nodes.

Description

Given a set of nodes (GO terms) this function is returning the subgraph containing these nodes and their ancestors.

Usage

```
inducedGraph(dag, startNodes)
nodesInInducedGraph(dag, startNodes)
```

Arguments

dag An object of class graphNEL containing a directed graph.

startNodes A character vector giving the starting nodes.

Value

An object of class graphNEL-class is returned.

Author(s)

Adrian Alexa

See Also

```
topGOdata-class, reverseArch,
```

```
data(GOdata)
## the GO graph
g <- graph(GOdata)
g
## select 10 random nodes
sn <- sample(nodes(g), 10)
## the subgraph induced by these nodes
sg <- inducedGraph(g, sn)
sg</pre>
```

parentChild-class 25

parentChild-class Classes "parentChild" and "pC"

Description

Classes that extend the "classicCount" class by adding support for the parent-child test.

Objects from the Class

```
Objects can be created by calls of the form new("parentChild", testStatistic, name, groupMembers, parents, sigMembers, joinFun, ...).
```

Slots

```
splitIndex: Object of class "integer" ~~
joinFun: Object of class "character" ~~
significant: Object of class "integer" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
members: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "list" ~~
```

Extends

Class "classicCount", directly. Class "groupStats", by class "classicCount", distance 2.

Methods

```
allMembers<- signature(object = "parentChild"): ...
allMembers signature(object = "parentChild"): ...
allParents signature(object = "parentChild"): ...
getSigGroups signature(object = "topGOdata", test.stat = "parentChild"): ...
initialize signature(.Object = "parentChild"): ...
joinFun signature(object = "parentChild"): ...
numAllMembers signature(object = "parentChild"): ...
numSigAll signature(object = "parentChild"): ...
sigAllMembers signature(object = "parentChild"): ...
sigMembers<- signature(object = "parentChild"): ...
updateGroup signature(object = "parentChild"): ...</pre>
```

26 printGraph-methods

Author(s)

Adrian Alexa

See Also

 ${\tt classicCount-class}, {\tt groupStats-class}, {\tt getSigGroups-methods}$

Examples

```
showClass("parentChild")
showClass("pC")
```

printGraph-methods

Visualisation functions

Description

Functions to plot the subgraphs induced by the most significant GO terms

Usage

Arguments

object an object of class topGOdata.

GOdata an object of class topGOdata.

result an object of class topGOresult.

firstSigNodes the number of top scoring GO terms which

refResult an object of class topGOresult.
termsP.value named vector of p-values.
reverse the direction of the edges.

sigForAll if TRUE the score/p-value of all nodes in the DAG is shown, otherwise only the

score for the sigNodes

wantedNodes the nodes that we want to find, we will plot this nodes with a different color. The

vector contains the names of the nodes

putWN the graph is generated with using the firstSigNodes and the wantedNodes.

printGraph-methods 27

putCL we generate the graph from the nodes given by all previous parameters, plus

their children. if putCL = 1 than only the children are added, if putCL = n we

get the nodes form the next n levels.

type used for plotting pie charts showEdges if TRUE the edge are shown

swPlot if true the graph is plotted, if not no plotting is done.

useInfo additional info to be ploted to each node.

oldSigNodes used to plot the (new) sigNodes in the same color range as the old ones

useFullNames argument for internal use ..
plotFunction argument for internal use ..
.NO.CHAR argument for internal use ..

... Extra arguments for printGraph can be:

fn.prefix character string giving the file name prefix.

useInfo as in showSigOfNodes function.

pdfSW logical attribute switch between PDF or PS formats.

Details

There are two functions available. The showSigOfNodes will plot the induced subgraph to the current graphic device. The printGraph is a warping function for showSigOfNodes and will save the resulting graph into a PDF or PS file.

In the plots, the significant nodes are represented as rectangles. The plotted graph is the upper induced graph generated by these significant nodes.

Author(s)

Adrian Alexa

See Also

```
{\tt groupStats-class}, {\tt getSigGroups-methods}
```

28 topGOdata-class

topGOdata-class

Class "topGOdata"

Description

TODO: The node attributes are environments containing the genes/probes annotated to the respective node

If genes is a numeric vector than this should represent the gene's score. If it is factor it should discriminate the genes in interesting genes and the rest

TODO: it will be a good idea to replace the allGenes and allScore with an ExpressionSet class. In this way we can use tests like global test, globalAncova.... – ALL variables starting with . are just for internal class usage (private)

Objects from the Class

Objects can be created by calls of the form new("topGOdata", ontology, allGenes, geneSelectionFun, description, annotationFun, ...). ~~ describe objects here ~~

Slots

```
description: Object of class "character" ~~
ontology: Object of class "character" ~~
allGenes: Object of class "character" ~~
allScores: Object of class "ANY" ~~
geneSelectionFun: Object of class "function" ~~
feasible: Object of class "logical" ~~
nodeSize: Object of class "integer" ~~
graph: Object of class "graphNEL" ~~
expressionMatrix: Object of class "matrix" ~~
phenotype: Object of class "factor" ~~
```

Methods

topGOdata-class 29

```
feasible<- signature(object = "topGOdata"): ...</pre>
    feasible signature(object = "topGOdata"): ...
    geneScore signature(object = "topGOdata"): ...
    geneSelectionFun<- signature(object = "topGOdata"): ...</pre>
    geneSelectionFun signature(object = "topGOdata"): ...
    genes signature(object = "topGOdata"): A method for obtaining the list of genes, as a charac-
         ter vector, which will be used in the further analysis.
    numGenes signature(object = "topGOdata"): A method for obtaining the number of genes,
         which will be used in the further analysis. It has the same effect as: lenght(genes(object)).
    sigGenes signature(object = "topG0data"): A method for obtaining the list of significant genes,
         as a character vector.
    genesInTerm signature(object = "topGOdata", whichGO = "character"): ...
    genesInTerm signature(object = "topGOdata", whichGO = "missing"): ...
    getSigGroups signature(object = "topGOdata", test.stat = "classicCount"): ...
    getSigGroups signature(object = "topGOdata", test.stat = "classicScore"): ...
    graph<- signature(object = "topGOdata"): ...</pre>
    graph signature(object = "topGOdata"): ...
    initialize signature(.Object = "topGOdata"): ...
    ontology<- signature(object = "topGOdata"): ...</pre>
    ontology signature(object = "topGOdata"): ...
    termStat signature(object = "topGOdata", whichGO = "character"): ...
    termStat signature(object = "topGOdata", whichGO = "missing"): ...
    updateGenes signature(object = "topGOdata", geneList = "numeric", geneSelFun = "function"):
    updateGenes signature(object = "topGOdata", geneList = "factor", geneSelFun = "missing"):
    updateTerm<- signature(object = "topGOdata", attr = "character"): ...</pre>
    usedGO signature(object = "topGOdata"): ...
Author(s)
    Adrian Alexa
```

See Also

buildLevels, annFUN

30 topGOdata-class

```
## load the dataset
data(geneList)
library(package = affyLib, character.only = TRUE)
## the distribution of the adjusted p-values
hist(geneList, 100)
## how many differentially expressed genes are:
sum(topDiffGenes(geneList))
## build the topGOdata class
GOdata <- new("topGOdata",</pre>
             ontology = "BP",
             allGenes = geneList,
             geneSel = topDiffGenes,
             description = "GO analysis of ALL data: DE B-cell vs T-cell",
             annot = annFUN.db,
             affyLib = affyLib)
## display the GOdata object
## Examples on how to use the methods
## description of the experiment
description(GOdata)
## obtain the genes that will be used in the analysis
a <- genes(GOdata)</pre>
str(a)
numGenes(GOdata)
## obtain the score (p-value) of the genes
selGenes <- names(geneList)[sample(1:length(geneList), 10)]</pre>
gs <- geneScore(GOdata, whichGenes = selGenes)</pre>
print(gs)
## if we want an unnamed vector containing all the feasible genes
gs <- geneScore(GOdata, use.names = FALSE)</pre>
str(gs)
## the list of significant genes
sg <- sigGenes(GOdata)</pre>
str(sg)
numSigGenes(GOdata)
## to update the gene list
.geneList <- geneScore(GOdata, use.names = TRUE)</pre>
GOdata ## more available genes
```

topGOresult-class 31

```
GOdata <- updateGenes(GOdata, .geneList, topDiffGenes)
GOdata ## the available genes are now the feasible genes
## the available GO terms (all the nodes in the graph)
go <- usedGO(GOdata)</pre>
length(go)
## to list the genes annotated to a set of specified GO terms
sel.terms <- sample(go, 10)</pre>
ann.genes <- genesInTerm(GOdata, sel.terms)</pre>
str(ann.genes)
## the score for these genes
ann.score <- scoresInTerm(GOdata, sel.terms)</pre>
str(ann.score)
## to see the number of annotated genes
num.ann.genes <- countGenesInTerm(GOdata)</pre>
str(num.ann.genes)
## to summarise the statistics
termStat(GOdata, sel.terms)
```

topGOresult-class

Class "topGOresult"

Description

Class instance created by getSigGroups-methods or by runTest

Objects from the Class

Objects can be created by calls of the form new("topGOresult", description, score, testName, algorithm, geneData).

Slots

description: character string containing a short description on how the object was build. score: named numerical vector containing the p-values or the scores of the tested GO terms. testName: character string containing the name of the test statistic used. algorithm: character string containing the name of the algorithm used. geneData: list containing summary statistics on the genes/gene universe/annotations.

Methods

score: method to access the score slot.
testName: method to access the testName slot.

32 weightCount-class

```
algorithm: method to access the algorithm slot.
geneData: method to access the geneData slot.
show: method to print the object.
combineResults: method to aggregate two or more topGOresult objects. method = c("gmean", "mean", "median", "min", "max") provides the way the object scores (which most of the time are p-values) are combined..
```

Author(s)

Adrian Alexa

See Also

groupStats-class, getSigGroups-methods

Examples

```
data(results.tG0)
s <- score(resultFisher)</pre>
go <- sort(names(s))</pre>
go.sub<- sample(go, 100)</pre>
go.mixed <- c(sample(go, 50), sample(ls(GOCCTerm), 20))</pre>
go.others <- sample(ls(GOCCTerm), 100)</pre>
str(go)
str(go.sub)
str(go.mixed)
str(go.others)
str(score(resultFisher, whichG0 = go))
str(score(resultFisher, whichGO = go.sub))
str(score(resultFisher, whichG0 = go.mixed))
str(score(resultFisher, whichGO = go.others))
avgResult <- combineResults(resultFisher, resultKS)</pre>
avgResult
combineResults(resultFisher, resultKS, method = "min")
```

weightCount-class

Class "weightCount"

Description

~~ A concise (1-5 lines) description of what the class is. ~~

weightCount-class 33

Details

TODO: Some details here.....

Objects from the Class

Objects can be created by calls of the form new("weightCount", testStatistic, name, allMembers, groupMembers, sigMembers, weights, sigRatio, penalise, ...).

Slots

```
weights: Object of class "numeric" ~~
sigRatio: Object of class "function" ~~
penalise: Object of class "function" ~~
roundFun: Object of class "function" ~~
significant: Object of class "integer" ~~
name: Object of class "character" ~~
allMembers: Object of class "character" ~~
testStatistic: Object of class "function" ~~
testStatPar: Object of class "list" ~~
```

Extends

Class "classicCount", directly. Class "groupStats", by class "classicCount", distance 2.

Methods

No methods defined with class "weightCount" in the signature.

Author(s)

Adrian Alexa

See Also

groupStats-class, getSigGroups-methods

Index

-land-Count alone (
classicCount-class, 6 (topGOdata-class), 28	
classicExpr-class, 7 allMembers (groupStats-class), 23	
classicScore-class, 8 allMembers, elimScore-method	
elimCount-class, 14 (elimScore-class), 16	
elimExpr-class, 15 allMembers, groupStats-method	
elimScore-class, 16 (groupStats-class), 23	
groupStats-class, 23 allMembers, parentChild-method	
parentChild-class, 25 (parentChild-class), 25	
topGOdata-class, 28 allMembers, weight01Expr-method	
topGOresult-class, 31 (elimExpr-class), 15	
weightCount-class, 32 allMembers, weight01Score-method	
* datasets (elimScore-class), 16	
geneList, 18 allMembers.weightCount-method	
GOdata, 21 (weightCount-class), 32	
* graphs allMembers<- (groupStats-class), 23	
Determines the levels of a allMembers <classicexpr-method< td=""><td></td></classicexpr-method<>	
Directed Acyclic Graph (DAG), 9 (classicExpr-class). 7	
getPvalues, 19	
inducedGraph, 24 (groupStats-class), 23	
topGOdata-class, 28 allMembers<- parentChild-method	
* methods (narentChild-class) 25	
dignostic-methods, II allMembers<- nC-method	
getSiguroups, 20 (narentChild-class) 25	
printGraph-methods, 26 allParents (parentChild-class) 25	
* MISC	
diffron, 5 (parentChild=class) 25	
Gene Set tests statistics, 1/	
groupGoteriis, 22	nod
* package	
topGO-package, 2 (Classic e-class), 8 allScore, classicScore, missing-met	nod
affyLib (geneList), 18 (classicScore-class), 8	
algorithm (topGOresult-class), 31 allScore, elimScore, logical-method	
algorithm, topGOresult-method (elimScore-class), 16	
(topGOresult-class), 31 allScore, elimScore, missing-method	
algorithm<- (topGOresult-class), 31 (elimScore-class), 16	
algorithm<-,topGOresult-method allScore,weight01Score,logical-me	hod
(topGOresult-class), 31 (elimScore-class), 16	
allGenes (topGOdata-class), 28 allScore, weight01Score, missing-me	hod

(elimScore-class), 16	depth,leaCount-method
alternative, elimScore-method	(elimCount-class), 14
(elimScore-class), 16	<pre>depth,leaExpr-method(elimExpr-class),</pre>
annFUN, 3, 29	15
attrInTerm (topGOdata-class), 28	depth,leaScore-method
attrInTerm, topGOdata, character, character-met	hod (elimScore-class), 16
(topGOdata-class), 28	depth<- (elimCount-class), 14
attrInTerm,topGOdata,character,missing-metho	
(topGOdata-class), 28	(elimCount-class), 14
(depth<-,leaExpr-method
buildLevels, 29	(elimExpr-class), 15
buildLevels (Determines the levels of	depth<-,leaScore-method
a Directed Acyclic Graph	(elimScore-class), 16
(DAG)), 9	description (topGOdata-class), 28
(DAG)), 9	description, topGOdata-method
1 10 11/25 22	(topGOdata-class), 28
classicCount, 14, 25, 33	description, topGOresult-method
classicCount-class, 6	(topGOresult-class), 31
classicExpr, 15	description<- (topGOdata-class), 28
classicExpr-class,7	description<-,topGOdata,ANY-method
classicScore, 16	(topGOdata-class), 28
classicScore-class, 8	description<-,topGOresult,ANY-method
combineResults (topGOresult-class), 31	(topGOresult-class), 31
contTable (classicCount-class), 6	Determines the levels of a Directed
contTable,classicCount-method	Acyclic Graph (DAG), 9
<pre>(classicCount-class), 6</pre>	dignostic-methods, 11
contTable,elimCount-method	arginostic ilictious, ii
(elimCount-class), 14	elim(elimCount-class), 14
<pre>countGenesInTerm(topGOdata-class), 28</pre>	elim,elimCount-method
$\verb countGenesInTerm , topGOdata , character-method $	(elimCount-class), 14
(topGOdata-class), 28	elim,elimScore-method
<pre>countGenesInTerm,topGOdata,missing-method</pre>	(elimScore-class), 16
(topGOdata-class), 28	elim,weight01Count-method
<pre>cutOff(elimCount-class), 14</pre>	(elimCount-class), 14
cutOff,elimCount-method	elim,weight01Expr-method
(elimCount-class), 14	(elimExpr-class), 15
cutOff,elimExpr-method	elim,weight01Score-method
(elimExpr-class), 15	(elimScore-class), 16
cutOff,elimScore-method	elim<- (elimCount-class), 14
(elimScore-class), 16	elim<-,elimCount-method
cutOff<- (elimCount-class), 14	(elimCount-class), 14
cutOff<-,elimCount-method	elim<-,elimScore-method
(elimCount-class), 14	(elimScore-class), 16
cutOff<-,elimExpr-method	elim<-,weight01Count-method
(elimExpr-class), 15	(elimCount-class), 14
cutOff<-,elimScore-method	elim<-,weight01Expr-method
(elimScore-class), 16	(elimExpr-class), 15
	elim<-,weight01Score-method
depth(elimCount-class), 14	(elimScore-class), 16

elimCount-class, 14	<pre>getGraphRoot(Determines the levels of</pre>
elimExpr-class, 15	a Directed Acyclic Graph
elimScore-class, 16	(DAG)), 9
emptyExpr,classicExpr-method	getNoOfLevels(Determines the levels
(classicExpr-class), 7	of a Directed Acyclic Graph
expressionMatrix (topGOdata-class), 28	(DAG)), 9
expressionMatrix,topGOdata-method	getPvalues, 19
(topGOdata-class), 28	getSigGroups, 20
	<pre>getSigGroups,topGOdata,classicCount-method</pre>
feasible (topGOdata-class), 28	(getSigGroups), 20
feasible, topGOdata-method	<pre>getSigGroups,topGOdata,classicExpr-method</pre>
(topGOdata-class), 28	(getSigGroups), 20
feasible<- (topGOdata-class), 28	<pre>getSigGroups,topGOdata,classicScore-method</pre>
feasible<-,topGOdata-method	(getSigGroups), 20
(topGOdata-class), 28	<pre>getSigGroups,topGOdata,elimCount-method</pre>
	(getSigGroups), 20
Gene set tests statistics, 17	<pre>getSigGroups,topGOdata,elimExpr-method</pre>
geneData (topGOresult-class), 31	(getSigGroups), 20
geneData, topGOresult-method	<pre>getSigGroups,topGOdata,elimScore-method</pre>
(topGOresult-class), 31	(getSigGroups), 20
geneData<- (topGOresult-class), 31	<pre>getSigGroups,topGOdata,leaCount-method</pre>
geneData<-,topGOresult-method	(getSigGroups), 20
(topGOresult-class), 31	<pre>getSigGroups,topGOdata,leaExpr-method</pre>
geneList, 18	(getSigGroups), 20
genes (topGOdata-class), 28	<pre>getSigGroups,topGOdata,leaScore-method</pre>
genes, topGOdata-method	(getSigGroups), 20
(topGOdata-class), 28	<pre>getSigGroups,topGOdata,parentChild-method</pre>
geneScore (topGOdata-class), 28	(getSigGroups), 20
geneScore, topGOdata, character-method	getSigGroups,topGOdata,pC-method
(topGOdata-class), 28	(getSigGroups), 20
geneScore, topGOdata, missing-method	getSigGroups,topGOdata,weight01Count-method
(topGOdata-class), 28	(getSigGroups), 20
geneScore, topGOdata-method	<pre>getSigGroups,topGOdata,weight01Expr-method</pre>
(topGOdata-class), 28	(getSigGroups), 20
geneSelectionFun (topGOdata-class), 28	getSigGroups,topGOdata,weight01Score-method
geneSelectionFun, topGOdata-method	(getSigGroups), 20
(topGOdata-class), 28	<pre>getSigGroups,topGOdata,weightCount-method</pre>
geneSelectionFun<- (topGOdata-class), 28	(getSigGroups), 20
geneSelectionFun<-,topGOdata-method	<pre>getSigGroups-methods (getSigGroups), 20</pre>
(topGOdata-class), 28	<pre>getSigRatio (weightCount-class), 32</pre>
genesInTerm (topGOdata-class), 28	getSigRatio,weightCount-method
genesInTerm,topGOdata,character-method	(weightCount-class), 32
(topGOdata-class), 28	GOBPTerm (groupGOTerms), 22
genesInTerm,topGOdata,missing-method	GOCCTerm (groupGOTerms), 22
(topGOdata-class), 28	GOdata, 21
GenTable (dignostic-methods), 11	GOFisherTest (Gene set tests
GenTable, topGOdata-method	statistics), 17
(dignostic-methods), 11	GOFisherTest, classicCount-method

(classicCount-class), 6	initialize,groupStats-method
GOFisherTest,elimCount-method	(groupStats-class), 23
(elimCount-class), 14	initialize,leaCount-method
GOglobalTest(Gene set tests	(elimCount-class), 14
statistics), 17	initialize,leaExpr-method
GOglobalTest,classicExpr-method	(elimExpr-class), 15
(classicExpr-class), 7	initialize,leaScore-method
GOKSTest, 19	(elimScore-class), 16
GOKSTest (Gene set tests statistics), 17	initialize,parentChild-method
GOKSTest,classicScore-method	(parentChild-class), 25
(classicScore-class), 8	initialize,pC-method
GOKSTiesTest(Gene set tests	(parentChild-class), 25
statistics), 17	initialize,topGOdata-method
GOKSTiesTest,classicScore-method	(topGOdata-class), 28
(classicScore-class), 8	initialize,topGOresult-method
GOMFTerm(groupGOTerms), 22	(topGOresult-class), 31
GOplot(printGraph-methods), 26	initialize,weight01Count-method
GOSumTest (Gene set tests statistics),	(elimCount-class), 14
17	initialize,weight01Expr-method
GOSumTest,classicScore-method	(elimExpr-class), 15
(classicScore-class), 8	initialize, weight01Score-method
GOTERM, 22	(elimScore-class), 16
GOtTest (Gene set tests statistics), 17	initialize, weightCount-method
GOtTest,classicScore-method	(weightCount-class), 32
(classicScore-class), 8	inverseList(annFUN), 3
graph(topGOdata-class),28	isinFun (namentChild alasa) 25
graph,topGOdata-method	joinFun parentChild-class), 25
(topGOdata-class), 28	<pre>joinFun,parentChild-method</pre>
graph<-(topGOdata-class), 28	(parefitchillu-class), 23
graph<-,topGOdata-method	leaCount-class (elimCount-class), 14
(topGOdata-class), 28	leaExpr-class (elimExpr-class), 15
groupGOTerms, 22	leaScore-class (elimScore-class), 16
groupStats, <i>8</i> , <i>14–16</i> , <i>25</i> , <i>33</i>	10000010 01000 (01111100010 01000), 10
groupStats-class, 23	members (groupStats-class), 23
	members,elimScore-method
inducedGraph, 11, 24	(elimScore-class), 16
initialize,classicCount-method	members,groupStats,missing-method
<pre>(classicCount-class), 6</pre>	(groupStats-class), 23
initialize,classicExpr-method	members, weight01Expr, missing-method
(classicExpr-class), 7	(elimExpr-class), 15
initialize,classicScore-method	members,weight01Score,missing-method
(classicScore-class), 8	(elimScore-class), 16
initialize,elimCount-method	members,weightCount-method
(elimCount-class), 14	(weightCount-class), 32
initialize,elimExpr-method	<pre>members<-(groupStats-class), 23</pre>
(elimExpr-class), 15	<pre>members<-,groupStats-method</pre>
initialize,elimScore-method	(groupStats-class), 23
(elimScore-class), 16	<pre>membersExpr(classicExpr-class), 7</pre>

membersExpr,classicExpr-method	numMembers,weight01Expr-method
(classicExpr-class), 7	(elimExpr-class), 15
<pre>membersScore(classicScore-class), 8</pre>	numMembers,weight01Score-method
membersScore,classicScore-method	(elimScore-class), 16
(classicScore-class), 8	numMembers,weightCount-method
membersScore,elimScore-method	<pre>(weightCount-class), 32</pre>
(elimScore-class), 16	<pre>numSigAll(classicCount-class), 6</pre>
membersScore,weight01Score-method	numSigAll,classicCount-method
(elimScore-class), 16	(classicCount-class), 6
	numSigAll,elimCount-method
Name (groupStats-class), 23	(elimCount-class), 14
Name, groupStats-method	numSigAll,parentChild-method
(groupStats-class), 23	(parentChild-class), 25
Name, weightCount-method	numSigAll,weight01Count-method
(weightCount-class), 32	(elimCount-class), 14
Name<- (groupStats-class), 23	numSigAll,weightCount-method
Name<-,groupStats-method	<pre>(weightCount-class), 32</pre>
(groupStats-class), 23	numSigGenes (topGOdata-class), 28
nodesInInducedGraph (inducedGraph), 24	numSigGenes,topGOdata-method
numAllMembers (groupStats-class), 23	(topGOdata-class), 28
numAllMembers, elimCount-method	<pre>numSigMembers(classicCount-class), 6</pre>
(elimCount-class), 14	numSigMembers,classicCount-method
numAllMembers, elimScore-method	(classicCount-class), 6
(elimScore-class), 16	numSigMembers,elimCount-method
numAllMembers,groupStats-method	(elimCount-class), 14
(groupStats-class), 23	numSigMembers,weight01Count-method
numAllMembers, parentChild-method	(elimCount-class), 14
(parentChild-class), 25	numSigMembers,weightCount-method
numAllMembers, weight01Count-method	<pre>(weightCount-class), 32</pre>
(elimCount-class), 14	
	ontology (topGOdata-class), 28
<pre>numAllMembers,weight01Expr-method (elimExpr-class), 15</pre>	ontology,topGOdata-method
	(topGOdata-class), 28
numAllMembers, weight01Score-method	ontology<- (topGOdata-class), 28
(elimScore-class), 16	ontology<-,topGOdata-method
numAllMembers, weightCount-method	(topGOdata-class), 28
(weightCount-class), 32	101111111111111111111111111111111111111
numGenes (topGOdata-class), 28	parentChild-class, 25
numGenes, topGOdata-method	pC-class (parentChild-class), 25
(topGOdata-class), 28	penalise (weightCount-class), 32
numMembers (groupStats-class), 23	penalise, weightCount, numeric, numeric-method
numMembers, elimCount-method	(weightCount-class), 32
(elimCount-class), 14	permSumStats (Gene set tests
numMembers, elimScore-method	statistics), 17
(elimScore-class), 16	phenotype (topGOdata-class), 28
numMembers, groupStats-method	phenotype, topGOdata-method
(groupStats-class), 23	(topGOdata-class), 28
numMembers, weight01Count-method	print,topGOdata-method
(elimCount-class), 14	(topGOdata-class), 28

print,topGOresult-method	score (topGOresult-class), 31
(topGOresult-class), 31	score,topGOresult-method
<pre>printGenes (dignostic-methods), 11</pre>	(topGOresult-class), 31
<pre>printGenes, topGOdata, character, character-met</pre>	h sd ore<- (classicScore-class), 8
(dignostic-methods), 11	score<-,classicScore-method
<pre>printGenes, topGOdata, character, missing-method</pre>	d (classicScore-class), 8
(dignostic-methods), 11	score<-,elimScore-method
<pre>printGenes-methods (dignostic-methods),</pre>	(elimScore-class), 16
11	score<-,topGOresult-method
<pre>printGraph (printGraph-methods), 26</pre>	(topGOresult-class), 31
<pre>printGraph,topGOdata,topGOresult,numeric,mis</pre>	• •
(printGraph-methods), 26	scoreOrder,classicScore-method
<pre>printGraph,topGOdata,topGOresult,numeric,top</pre>	
(printGraph-methods), 26	scoresInTerm (topGOdata-class), 28
printGraph-methods, 26	scoresInTerm, topGOdata, character-method
pType (classicExpr-class), 7	(topGOdata-class), 28
pType,classicExpr-method	scoresInTerm, topGOdata, missing-method
(classicExpr-class), 7	(topGOdata-class), 28
pType<- (classicExpr-class), 7	show, topGOdata-method
pType<-,classicExpr-method	(topGOdata-class), 28
(classicExpr-class), 7	show, topGOresult-method
(1.11.1)	(topGOresult-class), 31
rankMembers (classicScore-class), 8	showGroupDensity (dignostic-methods), 11
rankMembers, classicScore-method	showSigOfNodes (printGraph-methods), 26
(classicScore-class), 8	sigAllMembers (classicCount-class), 6
	sigAllMembers, classicCount-method
rankMembers, elimScore-method	(classicCount-class), 6
(elimScore-class), 16	sigAllMembers,elimCount-method
rankMembers, weight01Score-method	(elimCount-class), 14
(elimScore-class), 16	
readMappings (annFUN), 3	sigAllMembers, parentChild-method
resultFisher (GOdata), 21	(parentChild-class), 25
resultKS (GOdata), 21	sigAllMembers, weight01Count-method
reverseArch, 24	(elimCount-class), 14
reverseArch (Determines the levels of	sigGenes (topGOdata-class), 28
a Directed Acyclic Graph	sigGenes, topGOdata-method
(DAG)), 9	(topGOdata-class), 28
roundFun (weightCount-class), 32	sigMembers (classicCount-class), 6
roundFun, weightCount-method	sigMembers, classicCount-method
(weightCount-class), 32	(classicCount-class), 6
runTest (getSigGroups), 20	sigMembers, elimCount-method
runTest,groupStats,missing,missing-method	(elimCount-class), 14
(groupStats-class), 23	sigMembers, weight01Count-method
runTest,groupStats-method	(elimCount-class), 14
(groupStats-class), 23	sigMembers<- (classicCount-class), 6
$\verb"runTest", top GO data", character", character-method$	
(getSigGroups), 20	(classicCount-class), 6
runTest,topGOdata,missing,character-method	sigMembers<-,elimCount-method
(getSigGroups), 20	(elimCount-class), 14

sigMembers<-,parentChild-method	updateGroup,parentChild,missing,character-method
(parentChild-class), 25	(parentChild-class), 25
sigMembers<-,pC-method	updateGroup,pC,missing,character-method
(parentChild-class), 25	(parentChild-class), 25
significant (weightCount-class), 32	updateGroup,pC,missing,missing-method
significant,weightCount-method	(parentChild-class), 25
(weightCount-class), 32	updateGroup,weightCount,character,character-method
sigRatio (weightCount-class), 32	<pre>(weightCount-class), 32</pre>
sigRatio,weightCount-method	updateTerm<- (topGOdata-class), 28
<pre>(weightCount-class), 32</pre>	updateTerm<-,topGOdata,character-method
<pre>sigRatio<- (weightCount-class), 32</pre>	(topGOdata-class), 28
sigRatio<-,weightCount-method	usedGO (topGOdata-class), 28
(weightCount-class), 32	usedGO,topGOdata-method
	(topGOdata-class), 28
termStat(topGOdata-class), 28	
termStat,topGOdata,character-method	<pre>weight01Count-class(elimCount-class),</pre>
(topGOdata-class), 28	14
termStat,topGOdata,missing-method	weight01Expr, 15
(topGOdata-class), 28	weight01Expr-class(elimExpr-class), 15
testName (topGOresult-class), 31	<pre>weight01Score-class(elimScore-class),</pre>
testName,topGOresult-method	16
(topGOresult-class), 31	weightCount-class, 32
testName<- (topGOresult-class), 31	Weights (weightCount-class), 32
testName<-,topGOresult-method	Weights,weightCount,logical-method
(topGOresult-class), 31	<pre>(weightCount-class), 32</pre>
testStatistic (groupStats-class), 23	Weights, weightCount, missing-method
testStatistic,groupStats-method	(weightCount-class), 32
(groupStats-class), 23	Weights, weightCount-method
testStatistic,weightCount-method	(weightCount-class), 32
(weightCount-class), 32	Weights<- (weightCount-class), 32
testStatPar (groupStats-class), 23	<pre>Weights<-,weightCount-method</pre>
testStatPar,groupStats-method	(weightCount-class), 32
(groupStats-class), 23	whichAlgorithms(getSigGroups), 20
testStatPar,weightCount-method	whichTests(getSigGroups), 20
(weightCount-class), 32	
topDiffGenes(geneList), 18	
topGO (topGO-package), 2	
topGO-package, 2	
topGOdata-class, 28	
topGOresult-class, 31	
updateGenes (topGOdata-class), 28	
<pre>updateGenes,topGOdata,factor,missing-method</pre>	
(topGOdata-class), 28	
update Genes, top GO data, numeric, function-methods and the property of the	d
(topGOdata-class), 28	
updateGroup(groupStats-class), 23	
updateGroup,groupStats,character,character-m	ethod
(groupStats-class), 23	