

topGO

November 11, 2009

R topics documented:

annFUN	1
classicCount-class	3
classicExpr-class	4
classicScore-class	6
Determines the levels of a Directed Acyclic Graph (DAG)	7
elimCount-class	8
elimExpr-class	9
elimScore-class	10
geneList	11
getPvalues	12
getSigGroups	13
GOdata	14
Gene set tests statistics	14
groupGOTerms	15
groupStats-class	16
inducedGraph	17
parentChild-class	18
printGenes-methods	19
printGraph-methods	19
topGOdata-class	20
topGO-package	23
topGOresult-class	24
weightCount-class	25
Index	26

annFUN	<i>functions to map gene IDs to GO terms</i>
--------	--

Description

These functions are used to compile a list of GO terms and their mappings to gene identifiers.

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(l)

```

Arguments

whichOnto	character string specifying one of the three GO ontologies: "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 all gene identifiers will be used.
affyLib	character string containing the name of the Affymetrix chip.
gene2GO	named list of character vectors. The list names are genes identifiers. For each gene the character vector contains the GO terms IDs it maps to. Only the most specific annotations are required.
GO2genes	named list of character vectors. The list names are GO terms IDs. For each GO the character vector contains the genes identifiers which are mapped to it. Only the most specific annotations are required.
mapping
ID
file
...	other parameters
sep
IDsep
l

Details

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.

All these function restrict the GO terms to the ones belonging to the specified ontology.

Value

A named(GO terms IDs) list of character vectors.

Author(s)

Adrian Alexa

See Also[topGOdata-class](#)**Examples**

```

library(hgu133a.db)
set.seed(111)

## generate a gene list and the GO annotations
numGenes <- 50
selGenes <- sample(ls(hgu133aGO), numGenes)
gene2GO <- lapply(mget(selGenes, envir = hgu133aGO), names)
gene2GO[sapply(gene2GO, is.null)] <- NA

## the annotation for the first three genes
gene2GO[1:3]

## inverting the annotations
go2genes <- annFUN.gene2GO(whichOnto = "CC", gene2GO = gene2GO)

## generate a GO list with the genes annotations
numGO <- 30
selGO <- sample(ls(hgu133aGO2PROBE), numGO)
GO2gene <- lapply(mget(selGO, envir = hgu133aGO2PROBE), as.character)

GO2gene[1:3]

## select only the GO terms for a specific ontology
go2gene <- annFUN.GO2genes(whichOnto = "CC", GO2gene = GO2gene)

```

classicCount-class *Class "classicCount"*

Description

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

Details

TODO: Some details here.....

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.

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

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

Objects from the Class

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

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" ~~

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"): ...

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

classicScore-class *Class "classicScore"*

Description

TODO: A class that extends the virtual class groupStats by adding a slot representing the score of each gene. (used for KS test)

Objects from the Class

Objects can be created by calls of the form `new("classicScore", testStatistic, name, allMembers, groupMembers, score, decreasing)`. *~~ describe objects here ~~*

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" *~~*

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"): ...

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")
```

Determines the levels of a Directed Acyclic Graph (DAG)
Utility functions for Directed Acyclic Graphs (DAG)

Description

Determines the levels of a Directed Acyclic Graph (DAG)

TODO: This function take the a directed graph and constructs a named vector which contain the level on which a node is. The root has level 1.

TODO: Find the root(roots) of the DAG

TODO: Simple function to invert the direction of edges in an directed graph. The returned graph is of class graphNEL. It can use either simple matrices or sparse matrices (SparseM library)

Usage

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

Arguments

dag	~~Describe dag here~~
root	~~Describe root here~~
leafs2root	The leafs2root parameter tell if the graph has edges directed from the leaves to the root, or vice-versa
graphLevels	~~Describe graphLevels here~~
dirGraph	The graph to be transformed
useAlgo	"sparse" or "normal"
useWeights	If weights should be used (if useAlgo = 'normal' that the weigths are used anyway)

Details

.....

Value

it 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.

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.
```

elimCount-class *Classes "elimCount" and "weight01Count"*

Description

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

Details

TODO: Some details here....

Objects from the Class

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

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](#)**Examples**

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

elimExpr-class *Class "elimExpr" ~~~*

Description

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

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(.Object = "elimExpr"):...

```

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

elimScore-class *Classes "elimScore" and "weightOIScore"*

Description

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

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, ...)`.~~
describe objects here ~~

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.

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

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 correspondent 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.

Examples

```
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 *Function to compute p-values of a t-test for a gene expression matrix.*

Description

Warping function for computing the p-values for a gene expression matrix.

Usage

```
getPvalues(edata, classlabel, test = "t", alternative = c("greater", "two.sided", "less"),
genesID = NULL, correction = c("none", "Bonferroni", "Holm", "Hochberg", "Sidak",
"BH", "BY") [8])
```

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

Details

~~ If necessary, more details than the description above ~~

Value

An named vector of p-values is returned.

Author(s)

Adrian Alexa

See Also

[GOKSTest](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

```
library(ALL)
data(ALL)

## discriminate B-cell from T-cell
classLabel <- as.integer(sapply(ALL$BT, function(x) return(substr(x, 1, 1) == 'T')))

## Differentially expressed genes
geneList <- getPvalues(exprs(ALL), classlabel = classLabel,
                       alternative = "greater", correction = "BY")

hist(geneList, 50)
```

 getSigGroups

Algorithms for scoring GO terms

Description

TODO: This function is use for dispatching each algorithm

Usage

```
getSigGroups(object, test.stat, ...)
```

Arguments

```
object      ~~Describe object here~~
test.stat   ~~Describe test.stat here~~
...         ~~Describe ... here~~
```

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

```
comp1      Description of 'comp1'
comp2      Description of 'comp2'
...
```

Author(s)

Adrian Alexa

See Also

[topGOdata-class](#), [classicCount-class](#), [classicScore-class](#)

Examples

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

## The function is currently defined as
function(object, test.stat, ...) standardGeneric("getSigGroups")
```

`GOdata`*Example of a topGO data object*

Description

This data set contains an instance of a `topGOdata` object. It can be used to run an enrichment analysis directly.

Usage

```
data(GOdata)
```

Source

Generated using the ALL gene expression data. See [topGOdata-class](#) for code examples on how-to generate this object.

Examples

```
data(GOdata)

## print the object
GOdata
```

`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)
GOTest(object)
GOglobalTest(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.

GOTest: 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.

Author(s)

Adrian Alexa

See Also

[groupStats-class](#), [getSigGroups-methods](#)

`groupGOTerms` *~~function to do ... ~~*

Description

TODO: Function that split GOTERM in different ontologies. Every new environment contain only the terms from one of the ontologies 'BP', 'CC', 'MF'

Usage

```
groupGOTerms (where)
```

Arguments

`where` The the environment where you wantto bind the results

Details

~~ If necessary, more details than the description above ~~

Value

~Describe the value returned If it is a LIST, use

`comp1` Description of 'comp1'

`comp2` Description of 'comp2'

...

Author(s)

Adrian Alexa

See Also[topGOdata-class](#), [GOTerm](#)**Examples**`groupGOTerms ()`

```
groupStats-class  Class "groupStats"
```

Description

A virtual class containing basic group (GO term) data: gene names, genes scores, 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" ~~

Methods

allMembers<- signature(object = "groupStats"): ...
allMembers signature(object = "groupStats"): ...
initialize signature(.Object = "groupStats"): ...
members<- signature(object = "groupStats"): ...
members signature(object = "groupStats"): ...
Name<- signature(object = "groupStats"): ...
Name signature(object = "groupStats"): ...
numAllMembers signature(object = "groupStats"): ...
numMembers signature(object = "groupStats"): ...
runTest signature(object = "groupStats"): ...
testStatistic signature(object = "groupStats"): ...

Author(s)

Adrian Alexa

See Also

[classicCount-class](#), [getSigGroups-methods](#)

Examples

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

```
inducedGraph      ~~function to do ... ~~
```

Description

TODO: Given a GO term (or a list of GO terms) this function is returning the subgraph induced by node.

Usage

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

Arguments

```
dag           ~~Describe dag here~~
startNodes    ~~Describe startNodes here~~
```

Details

~~ If necessary, more details than the description above ~~

Value

An object of class [graphNEL-class](#) is returned.

Author(s)

Adrian Alexa

See Also

[topGOdata-class](#), [reverseArch](#),

Examples

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

parentChild-class *Classes "parentChild" and "pC"*

Description

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

Objects from the Class

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

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", name = "missing", members = "character"): ...

Author(s)

Adrian Alexa

See Also

[classicCount-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

printGenes-methods *Summary for genes annotated to a GO term*

Description

Function to print summary for the top genes annotated to the specified GO term.

Methods

~~describe this method here

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

printGraph-methods *~~ Methods for Function printGraph in Package 'topGO' ~~*

Description

~~ Methods for function printGraph in Package 'topGO' ~~

Methods

~~describe this method here

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

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" ~
graph: Object of class "graphNEL" ~

Methods

allGenes signature(object = "topGOdata"): ...
attrInTerm signature(object = "topGOdata", attr = "character", whichGO = "character"): ...
attrInTerm signature(object = "topGOdata", attr = "character", whichGO = "missing"): ...
countGenesInTerm signature(object = "topGOdata", whichGO = "character"): ...
 ...
countGenesInTerm signature(object = "topGOdata", whichGO = "missing"): ...
 ...
description<- signature(object = "topGOdata"): ...
description signature(object = "topGOdata"): ...
feasible<- signature(object = "topGOdata"): ...
feasible signature(object = "topGOdata"): ...
geneScore signature(object = "topGOdata"): ...

```

geneSelectionFun<- signature(object = "topGOdata"):...
geneSelectionFun signature(object = "topGOdata"):...
genes signature(object = "topGOdata"): A method for obtaining the list of genes, as
  a character 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: length(genes(object)).
sigGenes signature(object = "topGOdata"): A method for obtaining the list of signif-
  icant genes, as a character vector.
genesInTerm signature(object = "topGOdata", whichGO = "character"):...
genesInTerm signature(object = "topGOdata", whichGO = "missing"):...
genTable signature(object = "topGOdata", resList = "list"):...
GenTable signature(object = "topGOdata", ...):...
getSigGroups signature(object = "topGOdata", test.stat = "classicCount"):
  ...
getSigGroups signature(object = "topGOdata", test.stat = "classicScore"):
  ...
graph<- signature(object = "topGOdata"):...
graph signature(object = "topGOdata"):...
initialize signature(.Object = "topGOdata"):...
ontology<- signature(object = "topGOdata"):...
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"):...
usedGO signature(object = "topGOdata"):...

```

Author(s)

Adrian Alexa

See Also

[buildLevels](#), [annFUN](#)

Examples

```

## load the ALL dataset and the annotation library
library(ALL); data(ALL)
affyLib <- paste(annotation(ALL), "db", sep = ".")
library(package = affyLib, character.only = TRUE)

library(genefilter)
f1 <- pOverA(0.25, log2(100))

```

```

f2 <- function(x) (IQR(x) > 0.5)
ff <- filterfun(f1, f2)
ALL <- ALL[genefilter(ALL, ff), ]

## obtain the list of differentially expressed genes
## discriminate B-cell from T-cell
classLabel <- as.integer(sapply(ALL$BT, function(x) return(substr(x, 1, 1) == 'T')))

## over-expressed genes for T-cell samples
geneList <- getPvalues(exprs(ALL), classlabel = classLabel)

## the distribution of the adjusted p-values
hist(geneList, 100)
hist(geneList[geneList < 1], 100)

## define a function to select the "significant" genes
topDiffGenes <- function(allScore) {
  return(allScore < 0.01)
}

## how many differentially expressed genes are:
sum(topDiffGenes(geneList))

## build the topGodata class
Godata <- new("topGodata",
             ontology = "BP",
             allGenes = geneList,
             geneSel = topDiffGenes,
             description = "GO analysis of ALL data: Differential Expression between B- and T-cells",
             annot = annFUN.db,
             affyLib = affyLib)

## display the Godata object
Godata

#####
## 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)
str(a)
numGenes(Godata)

## obtain the score (p-value) of the genes
selGenes <- names(geneList)[sample(1:length(geneList), 10)]
gs <- geneScore(Godata, whichGenes = selGenes)
print(gs)

## if we want an unnamed vector containing all the feasible genes
gs <- geneScore(Godata, use.names = FALSE)
str(gs)

## the list of significant genes
sg <- sigGenes(Godata)

```

```
str(sg)
numSigGenes(GOdata)

## to update the gene list
.geneList <- geneScore(GOdata, use.names = TRUE)
GOdata ## more available genes
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)
length(go)

## to list the genes annotated to a set of specified GO terms
sel.terms <- sample(go, 10)
ann.genes <- genesInTerm(GOdata, sel.terms)
str(ann.genes)

## the score for these genes
ann.score <- scoresInTerm(GOdata, sel.terms)
str(ann.score)

## to see the number of annotated genes
num.ann.genes <- countGenesInTerm(GOdata)
str(num.ann.genes)

## to summarise the statistics
termStat(GOdata, sel.terms)
```

topGO-package

Enrichment analysis for Gene Ontology

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.

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, Jörg Rahnenführer

Maintainer: Adrian Alexa <alex@mpi-inf.mpg.de>

References

Alexa A., Rahnenführer 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](#)

topGOresult-class *Class "topGOresult"*

Description

Class instance created by [getSigGroups-methods](#)

Objects from the Class

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

Slots

description: Object of class "character" ~~

score: Object of class "numeric" ~~

testName: Object of class "character" ~~

testClass: Object of class "character" ~~

Methods

score: ~~describe this method here

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

weightCount-class *Class "weightCount"*

Description

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

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, ...)`. ~~ describe objects here ~~

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" ~~
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 "weightCount" in the signature.

Author(s)

Adrian Alexa

See Also

[classicScore-class](#), [groupStats-class](#), [getSigGroups-methods](#)

Examples

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

Index

*Topic **classes**

- classicCount-class, 3
- classicExpr-class, 4
- classicScore-class, 5
- elimCount-class, 7
- elimExpr-class, 8
- elimScore-class, 9
- groupStats-class, 15
- parentChild-class, 17
- topGOdata-class, 19
- topGOresult-class, 23
- weightCount-class, 24

*Topic **datasets**

- geneList, 10
- GOdata, 13

*Topic **graphs**

- Determines the levels of a Directed Acyclic Graph (DAG), 6
- getPvalues, 11
- inducedGraph, 16
- topGOdata-class, 19

*Topic **methods**

- getSigGroups, 12
- printGenes-methods, 18
- printGraph-methods, 18

*Topic **misc**

- annFUN, 1
- Gene set tests statistics, 13
- groupGOterms, 14

*Topic **package**

- topGO-package, 22

- algorithm(*topGOresult-class*), 23
- algorithm, *topGOresult*-method (*topGOresult-class*), 23
- algorithm<- (*topGOresult-class*), 23
- algorithm<-, *topGOresult*-method (*topGOresult-class*), 23
- allGenes (*topGOdata-class*), 19
- allGenes, *topGOdata*-method (*topGOdata-class*), 19
- allMembers (*groupStats-class*), 15

- allMembers, *elimScore*-method (*elimScore-class*), 9
- allMembers, *groupStats*-method (*groupStats-class*), 15
- allMembers, *parentChild*-method (*parentChild-class*), 17
- allMembers, *weight01Expr*-method (*elimExpr-class*), 8
- allMembers, *weight01Score*-method (*elimScore-class*), 9
- allMembers, *weightCount*-method (*weightCount-class*), 24
- allMembers<- (*groupStats-class*), 15
- allMembers<-, *classicExpr*-method (*classicExpr-class*), 4
- allMembers<-, *groupStats*-method (*groupStats-class*), 15
- allMembers<-, *parentChild*-method (*parentChild-class*), 17
- allMembers<-, *pC*-method (*parentChild-class*), 17
- allParents (*parentChild-class*), 17
- allParents, *parentChild*-method (*parentChild-class*), 17
- allScore (*classicScore-class*), 5
- allScore, *classicScore*, *logical*-method (*classicScore-class*), 5
- allScore, *classicScore*, *missing*-method (*classicScore-class*), 5
- allScore, *elimScore*, *logical*-method (*elimScore-class*), 9
- allScore, *elimScore*, *missing*-method (*elimScore-class*), 9
- allScore, *weight01Score*, *logical*-method (*elimScore-class*), 9
- allScore, *weight01Score*, *missing*-method (*elimScore-class*), 9
- alternative, *elimScore*-method (*elimScore-class*), 9
- annFUN, 1, 20
- attrInTerm (*topGOdata-class*), 19
- attrInTerm, *topGOdata*, *character*, *character*-method

- (*topGOdata-class*), 19
- attrInTerm, *topGOdata*, character, missing-method (*topGOdata-class*), 19
- buildLevels, 20
- buildLevels (Determines the levels of a Directed Acyclic Graph (DAG)), 6
- classicCount, 7, 17, 24
- classicCount-class, 5, 12, 16, 18
- classicCount-class, 3
- classicExpr, 8
- classicExpr-class, 4
- classicScore, 10
- classicScore-class, 4, 5, 8–10, 12, 18, 23, 24
- classicScore-class, 5
- contTable (*classicCount-class*), 3
- contTable, *classicCount*-method (*classicCount-class*), 3
- contTable, *elimCount*-method (*elimCount-class*), 7
- countGenesInTerm (*topGOdata-class*), 19
- countGenesInTerm, *topGOdata*, character-method (*topGOdata-class*), 19
- countGenesInTerm, *topGOdata*, missing-method (*topGOdata-class*), 19
- cutOff (*elimCount-class*), 7
- cutOff, *elimCount*-method (*elimCount-class*), 7
- cutOff, *elimExpr*-method (*elimExpr-class*), 8
- cutOff, *elimScore*-method (*elimScore-class*), 9
- cutOff<- (*elimCount-class*), 7
- cutOff<-, *elimCount*-method (*elimCount-class*), 7
- cutOff<-, *elimExpr*-method (*elimExpr-class*), 8
- cutOff<-, *elimScore*-method (*elimScore-class*), 9
- description (*topGOdata-class*), 19
- description, *topGOdata*-method (*topGOdata-class*), 19
- description, *topGOresult*-method (*topGOresult-class*), 23
- description<- (*topGOdata-class*), 19
- description<-, *topGOdata*, ANY-method (*topGOdata-class*), 19
- description<-, *topGOresult*, ANY-method (*topGOresult-class*), 23
- Determines the levels of a Directed Acyclic Graph (DAG), 6
- elim (*elimCount-class*), 7
- elim, *elimCount*-method (*elimCount-class*), 7
- elim, *elimScore*-method (*elimScore-class*), 9
- elim, *weight01Count*-method (*elimCount-class*), 7
- elim, *weight01Expr*-method (*elimExpr-class*), 8
- elim, *weight01Score*-method (*elimScore-class*), 9
- elim<- (*elimCount-class*), 7
- elim<-, *elimCount*-method (*elimCount-class*), 7
- elim<-, *elimScore*-method (*elimScore-class*), 9
- elim<-, *weight01Count*-method (*elimCount-class*), 7
- elim<-, *weight01Expr*-method (*elimExpr-class*), 8
- elim<-, *weight01Score*-method (*elimScore-class*), 9
- elimCount-class, 7
- elimExpr-class, 8
- elimScore-class, 9
- emptyExpr, *classicExpr*-method (*classicExpr-class*), 4
- expressionMatrix (*topGOdata-class*), 19
- expressionMatrix, *topGOdata*-method (*topGOdata-class*), 19
- feasible (*topGOdata-class*), 19
- feasible, *topGOdata*-method (*topGOdata-class*), 19
- feasible<- (*topGOdata-class*), 19
- feasible<-, *topGOdata*-method (*topGOdata-class*), 19
- Gene set tests statistics, 13
- geneData (*topGOresult-class*), 23
- geneData, *topGOresult*-method (*topGOresult-class*), 23
- geneData<- (*topGOresult-class*), 23
- geneData<-, *topGOresult*-method (*topGOresult-class*), 23
- geneList, 10

- genes (*topGOdata-class*), 19
- genes, *topGOdata*-method (*topGOdata-class*), 19
- geneScore (*topGOdata-class*), 19
- geneScore, *topGOdata*, character-method (*topGOdata-class*), 19
- geneScore, *topGOdata*, missing-method (*topGOdata-class*), 19
- geneScore, *topGOdata*-method (*topGOdata-class*), 19
- geneSelectionFun (*topGOdata-class*), 19
- geneSelectionFun, *topGOdata*-method (*topGOdata-class*), 19
- geneSelectionFun<- (*topGOdata-class*), 19
- geneSelectionFun<-, *topGOdata*-method (*topGOdata-class*), 19
- genesInTerm (*topGOdata-class*), 19
- genesInTerm, *topGOdata*, character-method (*topGOdata-class*), 19
- genesInTerm, *topGOdata*, missing-method (*topGOdata-class*), 19
- GenTable (*topGOdata-class*), 19
- genTable (*topGOdata-class*), 19
- genTable, *topGOdata*, list-method (*topGOdata-class*), 19
- GenTable, *topGOdata*-method (*topGOdata-class*), 19
- getGraphRoot (*Determines the levels of a Directed Acyclic Graph (DAG)*), 6
- getNoOfLevels (*Determines the levels of a Directed Acyclic Graph (DAG)*), 6
- getPvalues, 11
- getSigGroups, 12
- getSigGroups, *topGOdata*, classicCount-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, classicExpr-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, classicScore-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, elimCount-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, elimExpr-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, elimScore-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, parentChild-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, pC-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, weight01Count-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, weight01Expr-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, weight01Score-method (*getSigGroups*), 12
- getSigGroups, *topGOdata*, weightCount-method (*getSigGroups*), 12
- getSigGroups-methods, 4, 5, 8–11, 14, 16, 18, 23, 24
- getSigGroups-methods (*getSigGroups*), 12
- getSigRatio (*weightCount-class*), 24
- getSigRatio, *weightCount*-method (*weightCount-class*), 24
- GOBPterm (*groupGOTerms*), 14
- GOCCterm (*groupGOTerms*), 14
- GOdata, 13
- GOFisherTest (*Gene set tests statistics*), 13
- GOFisherTest, classicCount-method (*classicCount-class*), 3
- GOFisherTest, elimCount-method (*elimCount-class*), 7
- GOGlobalTest (*Gene set tests statistics*), 13
- GOGlobalTest, classicExpr-method (*classicExpr-class*), 4
- GOKSTest, 11
- GOKSTest (*Gene set tests statistics*), 13
- GOKSTest, classicScore-method (*classicScore-class*), 5
- GOKSTest, elimScore-method (*elimScore-class*), 9
- GOPTerm (*groupGOTerms*), 14
- GOpLot (*printGraph-methods*), 18
- GOTerm, 15
- GOTest (*Gene set tests statistics*), 13
- GOTest, classicScore-method (*classicScore-class*), 5
- graph (*topGOdata-class*), 19
- graph, *topGOdata*-method (*topGOdata-class*), 19
- graph<- (*topGOdata-class*), 19
- graph<-, *topGOdata*-method (*topGOdata-class*), 19
- graphNEL-class, 7, 16
- groupGOTerms, 14

- groupStats, 4, 7, 8, 10, 17, 24
 groupStats-class, 4, 5, 8–11, 14, 18, 23, 24
 groupStats-class, 15
 inducedGraph, 7, 16
 initialize, classicCount-method (classicCount-class), 3
 initialize, classicExpr-method (classicExpr-class), 4
 initialize, classicScore-method (classicScore-class), 5
 initialize, elimCount-method (elimCount-class), 7
 initialize, elimExpr-method (elimExpr-class), 8
 initialize, elimScore-method (elimScore-class), 9
 initialize, groupStats-method (groupStats-class), 15
 initialize, parentChild-method (parentChild-class), 17
 initialize, pC-method (parentChild-class), 17
 initialize, topGOdata-method (topGOdata-class), 19
 initialize, topGOresult-method (topGOresult-class), 23
 initialize, weight01Count-method (elimCount-class), 7
 initialize, weight01Expr-method (elimExpr-class), 8
 initialize, weight01Score-method (elimScore-class), 9
 initialize, weightCount-method (weightCount-class), 24
 inverseList (annFUN), 1
 joinFun (parentChild-class), 17
 joinFun, parentChild-method (parentChild-class), 17
 members (groupStats-class), 15
 members, elimScore-method (elimScore-class), 9
 members, groupStats-method (groupStats-class), 15
 members, weight01Expr-method (elimExpr-class), 8
 members, weight01Score-method (elimScore-class), 9
 members, weightCount-method (weightCount-class), 24
 members<- (groupStats-class), 15
 members<-, groupStats-method (groupStats-class), 15
 membersExpr (classicExpr-class), 4
 membersExpr, classicExpr-method (classicExpr-class), 4
 membersScore (classicScore-class), 5
 membersScore, classicScore-method (classicScore-class), 5
 membersScore, elimScore-method (elimScore-class), 9
 membersScore, weight01Score-method (elimScore-class), 9
 Name (groupStats-class), 15
 Name, groupStats-method (groupStats-class), 15
 Name, weightCount-method (weightCount-class), 24
 Name<- (groupStats-class), 15
 Name<-, groupStats-method (groupStats-class), 15
 nodesInInducedGraph (inducedGraph), 16
 numAllMembers (groupStats-class), 15
 numAllMembers, elimCount-method (elimCount-class), 7
 numAllMembers, elimScore-method (elimScore-class), 9
 numAllMembers, groupStats-method (groupStats-class), 15
 numAllMembers, parentChild-method (parentChild-class), 17
 numAllMembers, weight01Count-method (elimCount-class), 7
 numAllMembers, weight01Expr-method (elimExpr-class), 8
 numAllMembers, weight01Score-method (elimScore-class), 9
 numAllMembers, weightCount-method (weightCount-class), 24
 numGenes (topGOdata-class), 19
 numGenes, topGOdata-method (topGOdata-class), 19
 numMembers (groupStats-class), 15
 numMembers, elimCount-method (elimCount-class), 7
 numMembers, elimScore-method (elimScore-class), 9
 numMembers, groupStats-method (groupStats-class), 15

- numMembers, weight01Count-method
 (*elimCount-class*), 7
 numMembers, weight01Expr-method
 (*elimExpr-class*), 8
 numMembers, weight01Score-method
 (*elimScore-class*), 9
 numMembers, weightCount-method
 (*weightCount-class*), 24
 numSigAll (*classicCount-class*), 3
 numSigAll, classicCount-method
 (*classicCount-class*), 3
 numSigAll, elimCount-method
 (*elimCount-class*), 7
 numSigAll, parentChild-method
 (*parentChild-class*), 17
 numSigAll, weight01Count-method
 (*elimCount-class*), 7
 numSigAll, weightCount-method
 (*weightCount-class*), 24
 numSigGenes (*topGOdata-class*), 19
 numSigGenes, topGOdata-method
 (*topGOdata-class*), 19
 numSigMembers
 (*classicCount-class*), 3
 numSigMembers, classicCount-method
 (*classicCount-class*), 3
 numSigMembers, elimCount-method
 (*elimCount-class*), 7
 numSigMembers, weight01Count-method
 (*elimCount-class*), 7
 numSigMembers, weightCount-method
 (*weightCount-class*), 24

 ontology (*topGOdata-class*), 19
 ontology, topGOdata-method
 (*topGOdata-class*), 19
 ontology<- (*topGOdata-class*), 19
 ontology<-, topGOdata-method
 (*topGOdata-class*), 19

 parentChild-class, 17
 pC-class (*parentChild-class*), 17
 penalise (*weightCount-class*), 24
 penalise, weightCount, numeric, numeric-method
 (*weightCount-class*), 24
 phenotype (*topGOdata-class*), 19
 phenotype, topGOdata-method
 (*topGOdata-class*), 19
 print, topGOdata-method
 (*topGOdata-class*), 19
 print, topGOresult-method
 (*topGOresult-class*), 23

 printGenes (*printGenes-methods*),
 18
 printGenes, topGOdata, character, character-method
 (*printGenes-methods*), 18
 printGenes, topGOdata, character, missing-method
 (*printGenes-methods*), 18
 printGenes-methods, 18
 printGraph (*printGraph-methods*),
 18
 printGraph, topGOdata, topGOresult, numeric, missi
 (*printGraph-methods*), 18
 printGraph, topGOdata, topGOresult, numeric, topGO
 (*printGraph-methods*), 18
 printGraph-methods, 18
 pType (*classicExpr-class*), 4
 pType, classicExpr-method
 (*classicExpr-class*), 4
 pType<- (*classicExpr-class*), 4
 pType<-, classicExpr-method
 (*classicExpr-class*), 4

 rankMembers (*classicScore-class*),
 5
 rankMembers, classicScore-method
 (*classicScore-class*), 5
 rankMembers, elimScore-method
 (*elimScore-class*), 9
 rankMembers, weight01Score-method
 (*elimScore-class*), 9
 readMappings (*annFUN*), 1
 reverseArch, 16
 reverseArch (*Determines the
 levels of a Directed
 Acyclic Graph (DAG)*), 6
 roundFun (*weightCount-class*), 24
 roundFun, weightCount-method
 (*weightCount-class*), 24
 runTest (*getSigGroups*), 12
 runTest, groupStats, missing, missing-method
 (*groupStats-class*), 15
 runTest, groupStats-method
 (*groupStats-class*), 15
 runTest, topGOdata, character, character-method
 (*getSigGroups*), 12
 runTest, topGOdata, missing, character-method
 (*getSigGroups*), 12

 score (*topGOresult-class*), 23
 score, topGOresult, character-method
 (*topGOresult-class*), 23
 score, topGOresult, missing-method
 (*topGOresult-class*), 23
 score<- (*classicScore-class*), 5

- score<- , classicScore-method
 (classicScore-class), 5
 score<- , elimScore-method
 (elimScore-class), 9
 score<- , topGOresult-method
 (topGOresult-class), 23
 scoreOrder (classicScore-class), 5
 scoreOrder, classicScore-method
 (classicScore-class), 5
 scoresInTerm (topGOdata-class), 19
 scoresInTerm, topGOdata, character-method
 (topGOdata-class), 19
 scoresInTerm, topGOdata, missing-method
 (topGOdata-class), 19
 show, topGOdata-method
 (topGOdata-class), 19
 show, topGOresult-method
 (topGOresult-class), 23
 showGroupDensity
 (printGraph-methods), 18
 showSigOfNodes
 (printGraph-methods), 18
 sigAllMembers
 (classicCount-class), 3
 sigAllMembers, classicCount-method
 (classicCount-class), 3
 sigAllMembers, elimCount-method
 (elimCount-class), 7
 sigAllMembers, parentChild-method
 (parentChild-class), 17
 sigAllMembers, weight01Count-method
 (elimCount-class), 7
 sigGenes (topGOdata-class), 19
 sigGenes, topGOdata-method
 (topGOdata-class), 19
 sigMembers (classicCount-class), 3
 sigMembers, classicCount-method
 (classicCount-class), 3
 sigMembers, elimCount-method
 (elimCount-class), 7
 sigMembers, weight01Count-method
 (elimCount-class), 7
 sigMembers<-
 (classicCount-class), 3
 sigMembers<- , classicCount-method
 (classicCount-class), 3
 sigMembers<- , elimCount-method
 (elimCount-class), 7
 sigMembers<- , parentChild-method
 (parentChild-class), 17
 sigMembers<- , pC-method
 (parentChild-class), 17
 significant (weightCount-class),
 24
 significant, weightCount-method
 (weightCount-class), 24
 sigRatio (weightCount-class), 24
 sigRatio, weightCount-method
 (weightCount-class), 24
 sigRatio<- (weightCount-class), 24
 sigRatio<- , weightCount-method
 (weightCount-class), 24
 termStat (topGOdata-class), 19
 termStat, topGOdata, character-method
 (topGOdata-class), 19
 termStat, topGOdata, missing-method
 (topGOdata-class), 19
 testName (topGOresult-class), 23
 testName, topGOresult-method
 (topGOresult-class), 23
 testName<- (topGOresult-class), 23
 testName<- , topGOresult-method
 (topGOresult-class), 23
 testStatistic (groupStats-class),
 15
 testStatistic, groupStats-method
 (groupStats-class), 15
 testStatistic, weightCount-method
 (weightCount-class), 24
 testStatPar (groupStats-class), 15
 testStatPar, groupStats-method
 (groupStats-class), 15
 testStatPar, weightCount-method
 (weightCount-class), 24
 topDiffGenes (geneList), 10
 topGO (topGO-package), 22
 topGO-package, 22
 topGOdata-class, 2, 7, 12, 13, 15, 16, 23
 topGOdata-class, 19
 topGOresult-class, 23
 updateGenes (topGOdata-class), 19
 updateGenes, topGOdata, factor, missing-method
 (topGOdata-class), 19
 updateGenes, topGOdata, numeric, function-method
 (topGOdata-class), 19
 updateGroup (groupStats-class), 15
 updateGroup, groupStats, character, character-met
 (groupStats-class), 15
 updateGroup, parentChild, missing, character-met
 (parentChild-class), 17
 updateGroup, pC, missing, character-method
 (parentChild-class), 17

updateGroup, pC, missing, missing-method
 (*parentChild-class*), 17

updateGroup, weightCount, character, character-method
 (*weightCount-class*), 24

updateTerm<- (*topGOdata-class*), 19

updateTerm<-, topGOdata, character-method
 (*topGOdata-class*), 19

usedGO (*topGOdata-class*), 19

usedGO, topGOdata-method
 (*topGOdata-class*), 19

weight01Count-class
 (*elimCount-class*), 7

weight01Expr, 8

weight01Expr-class
 (*elimExpr-class*), 8

weight01Score-class
 (*elimScore-class*), 9

weightCount-class, 24

Weights (*weightCount-class*), 24

Weights, weightCount, logical-method
 (*weightCount-class*), 24

Weights, weightCount, missing-method
 (*weightCount-class*), 24

Weights, weightCount-method
 (*weightCount-class*), 24

Weights<- (*weightCount-class*), 24

Weights<- , weightCount-method
 (*weightCount-class*), 24

whichAlgorithms (*getSigGroups*), 12

whichTests (*getSigGroups*), 12