Package 'categoryCompare'

May 8, 2026

Version 1.57.0

Title Meta-analysis of high-throughput experiments using feature annotations

Author Robert M. Flight <rflight79@gmail.com>

Maintainer Robert M. Flight <rflight79@gmail.com>

URL https://github.com/rmflight/categoryCompare

BugReports https://github.com/rmflight/categoryCompare/issues

License GPL-2

Depends R (>= 2.10), Biobase, BiocGenerics (>= 0.13.8),

Suggests knitr, GO.db, KEGGREST, estrogen, org.Hs.eg.db, hgu95av2.db, limma, affy, genefilter, rmarkdown

Imports AnnotationDbi, hwriter, GSEABase, Category (>= 2.33.1), GOstats, annotate, colorspace, graph, RCy3 (>= 1.99.29), methods, grDevices, utils

LazyLoad yes

Description Calculates significant annotations (categories) in each of two (or more) feature (i.e. gene) lists, determines the overlap between the annotations, and returns graphical and tabular data about the significant annotations and which combinations of feature lists the annotations were found to be significant. Interactive exploration is facilitated through the use of RCytoscape (heavily suggested).

SystemRequirements Cytoscape (>= 3.6.1) (if used for visualization of results, heavily suggested)

TODO Text and HTML output without graphs.

biocViews Annotation, GO, MultipleComparison, Pathways, GeneExpression

VignetteBuilder knitr

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

git_branch devel

git_last_commit 05d1e21

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Contents

`categoryCompare-package`

Meta-analysis of high-throughput experiments using feature annotations

---

Description

Calculates significant annotations (categories) in each of two (or more) feature (i.e. gene) lists, determines the overlap between the annotations, and returns graphical and tabular data about the significant annotations and which combinations of feature lists the annotations were found to be significant. Interactive exploration is facilitated through the use of RCytoscape (heavily suggested).

Details

Package:	categoryCompare
Version:	1.21.2
License:	GPL-2
Depends:	Biobase (>= 1.15.29), AnnotationDbi (>= 0.1.15), Category
Suggests:	methods, GSEABase, hwriter, colorspace, graph, GO.db, KEGG.db, estrogen, org.Hs.eg.db, hgu9
Imports:	Biobase (>= 1.15.29), AnnotationDbi (>= 0.1.15), hwriter, GSEABase, Category (>= 2.21.2), GO
LazyLoad:	yes
biocViews:	Bioinformatics, Annotation, GO, MultipleComparisons, Pathways, GeneExpression
SystemRequirements:	Cytoscape (>= 2.8.0) (if used for visualization of results, heavily suggested), CytoscapeRPC plug
TODO:	Text and HTML output without graphs.
Built:	R 2.15.0; ; 2012-03-15 18:42:40 UTC; windows

Index:

	rate
listNames	listNames
mergeLists-methods	Function 'mergeLists' in Package 'categoryCompare'
mergedData-class	Class '"mergedData"'
minCount	minCount
minNodes	Delete nodes with less than a certain number of genes annotated
pvalueType	Type of p-values to return from object
resetColors-methods	resetColors
show-methods	Methods for Function show in Package 'categoryCompare'

Further information is available in the following vignettes:

categoryCompare_vignette	categoryCompare: High-throughput data meta-analysis using gene annotations (source)

Author(s)

Robert M. Flight <rflight79@gmail.com>

Maintainer: Robert M. Flight <rflight79@gmail.com>

---

breakEdges-methods *Methods for Function* breakEdges *in Package* **categoryCompare**

Description

Methods for function breakEdges in package **categoryCompare**

Methods

signature(cwObject = "ccCompareResult", cutoff = "numeric") Allows one to remove edges in the ccCompareResult mainGraph slot prior to passing it into Cytoscape for visualization. Given that the number of edges can be rather large (especially for Gene Ontology) this can easily speed up the transfer, without actually losing any information.

signature(cwObject = "numeric", cutoff = "numeric") Once an annotation graph is in Cytoscape, remove edges above or below the cutoff. Note that this does not affect the original graph in the ccCompareResult object.

Author(s)

Robert M Flight

See Also

breakEdges ccCompareResult ccOutCyt

Examples

```
data(ccData)

# breaking the edges in a ccCompareResult
ccResults$BP <- breakEdges(ccResults$BP, 0.8)
## Not run:
hasCy <- (if (.Platform$OS.type %in% "windows") {
(length(grep("Cytoscape", system("tasklist", intern=TRUE))) > 0)})

if hasCy {

 cwObj <- ccOutCyt(ccResults$BP,ccOpts)
 # now breaking them in the CytoscapeWindow object
 breakEdges(cwObj, 0.85)
 Sys.sleep(10)
 RCy3::deleteWindow(cwObj)
 }

## End(Not run)
```

---

ccCompare-methods *Comparison of enriched annotations*

Description

Takes the results from ccEnrich and compares the enriched annotations based on the settings previously set in ccOptions. Returns a ccCompareResult or ccCompareCollection object, see Details.

Usage

```
ccCompare(ccEnrichResult, ccOptions)
```

Arguments

ccEnrichResult	The enriched annotations collection returned from ccEnrich. This can be the ccEnrichCollection, GOccEnrichResult, or KEGGccEnrichResult
ccOptions	A ccOptions object that will determine which lists to actually compare against each other. See details below.

Details

Based on the enrichments found for each gene list, we now want to compare the annotations between lists. ccCompare accesses the annotations for each enrichment performed for each list, and makes the comparisons defined in ccOptions.

Value

ccCompare generates both a graph of the comparisons (to show how the categories are linked to each list and each other) and tabular output. The tabular output is a data frame, with ID for each term that was considered as a candidate annotation for each list, as well as a long description (Desc) of what the term is, and then membership and statistics from each gene list.

For each type of comparison (GO, KEGG, etc) a ccCompareResult is generated, with the following slots:

mainGraph	Annotations arranged as a graph
mainTable	The tabular results from all enrichment calculations combined into one
allAnnotation	A list of lists, where each entry is the annotation identifier, then a list for each comparison, with the genes that are annotated to that term that also belong to each list

The default is to generate an overlap graph for GO and KEGG, where the overlap is a measure of the similarity of the features (genes) annotated to each annotation term (based on a formula from EnrichmentMap). Optionally for GO, one can generate a hierarchical layout where the parent GO terms of the significant terms will also be included in the graph, with term origin saved in the node annotation (see example below to do this).

Only those terms with more than 10 and less than 500 annotated genes (according to the GO annotation file) are included.

When using weighted overlap graphs and **RCy3** for viewing, it is recommended to use breakEdges and minNodes to remove edges with low weights and nodes with only a few genes from the dataset annotated to them.

Author(s)

Robert M Flight

See Also

ccCompareResult ccCompareCollection ccOutCyt breakEdges outType ccEnrich

Examples

```
## Not run:
require(GO.db)
require(KEGG.db)
require(org.Hs.eg.db)

## End(Not run)
data(ccData)

# note that enrichLists is generated from ccEnrich
# ccResults <- ccCompare(enrichLists,ccOpts)
ccResults

# use the GO hierarchy tree
graphType(enrichLists$BP) <- "hierarchical"
# ccResultsBPHier <- ccCompare(enrichLists$BP,ccOpts)
ccResultsBPHier
```

ccCompareCollection-class

Class "ccCompareCollection"

Description

Holds multiple ccCompareResult objects.

Objects from the Class

Objects can be created by calls of the form new("ccCompareCollection", ...).

These are not normally created by the user, but rather by ccCompare while performing the categorical comparisons for each type of category

Slots

.Data: Object of class "list"

names: Object of class "character"

Extends

Class "namedList", directly. Class "list", by class "namedList", distance 2. Class "vector", by class "namedList", distance 3. Class "AssayData", by class "namedList", distance 3.

Methods

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

Author(s)

Robert M Flight

See Also

ccCompareResult ccCompare

Examples

```
showClass("ccCompareCollection")
```

ccCompareGeneric-methods

Methods for Function ccCompareGeneric *in Package* **categoryCompare**

Description

Methods for function ccCompareGeneric in package **categoryCompare**

Methods

signature(gccResult = "GENccEnrichResult", ccOptions = "ccOptions")

ccCompareResult-class *Class* "ccCompareResult"

Description

Holds the results from a single category comparison

Objects from the Class

Objects can be created by calls of the form new("ccCompareResult", ...).

Slots

mainGraph: Object of class "graph". Holds the graph describing the relationships between the annotations

subGraph: Object of class "list". Not currently used

mainTable: Object of class "data.frame". Table of results, with all the various statistics for each annotation in the category

allAnnotation: Object of class "list". For each annotation, which genes from which comparison are annotated to that particular annotation

categoryName: Object of class "character". Which category (e.g. GO, KEGG, etc) was used

ontology: Object of class "character". If GO, which ontology was used

Methods

allAnnotation signature(object = "ccCompareResult"): ...

mainGraph signature(object = "ccCompareResult"): ...

mainTable<- signature(object = "ccCompareResult"): ...

Author(s)

Robert M Flight

See Also

ccCompare ccCompareCollection

Examples

```
showClass("ccCompareResult")
```

ccData *Test data for* categoryCompare

Description

Processed data from the estrogen example data set

Usage

```
data(ccData)
```

Format

table10: Log-ratio output from **limma** for the comparison of presence-absence of estrogen at 10 hours

table48: Log-ratio output from **limma** for the comparison of presence-absence of estrogen at 48 hours

gUniverse: All of the genes measured on the chip

gseaRes: Toy results of GSEA analysis of 3 different tissues

enrichLists: Apply ccEnrich to a ccGeneList from table10 and table48

ccResults: Apply ccCompare to enrichLists

ccResultsBPHier: Modify enrichLists$BP to use a "hierarchical" layout

geneLists: a ccGeneList generated from genes in table10 and table48

ccOpts: a ccOptions object describing what we are going to do as far as feature list comparisons

Author(s)

Robert M Flight

Source

Taken from the **estrogen** package in Bioconductor, and then processed using the normal **affy** and **limma** tools.

See Also

ccGeneList ccEnrichCollection ccCompareCollection ccEnrich ccCompare

Examples

```
data(ccData)
```

ccEnrich-method *Perform annotation enrichment for multiple gene lists*

Description

Takes a ccGeneList object containing all the information needed to perform enrichment calculations for Gene Ontology.

Usage

```
ccEnrich(ccGeneList)
```

Arguments

ccGeneList	A ccGeneList object, which is really just a list of lists, with some extra slots to tell us how to examine results. Each entry in the list should be named to allow identification later on. Each sub list should contain a vector genes denoting the genes of interest, a vector universe denoting the gene background (i.e. all genes on the chip), and an entry annotation denoting an organism database package (such as **org.Hs.eg.db**). See ccGeneList for more details regarding this object.

Details

This function is essentially a wrapper for hyperGTestCC that performs all of the calculations for the many gene lists in one go, returning a list of HyperGResultCC objects, one for each of the ccTypes and each gene list. These various HyperGResultCC objects can then be accessed and results compared among the lists for each of the ontologies

Value

A list of HyperGResultCC objects, one for each ccType and gene list, returned as ccEnrichResult objects for each ccType. This can be passed with a ccOptions object to ccCompare to generate actual annotation comparisons.

Author(s)

Robert M Flight

See Also

ccGeneList, hyperGTestCC, ccEnrichResult

Examples

```
## Not run:
require(GO.db)
require(KEGG.db)
require(org.Hs.eg.db)

## End(Not run)
data(ccData)
```

```
g10 <- (unique(table10$Entrez[1:100]))
g48 <- (unique(table48$Entrez[1:100]))

list10 <- list(genes=g10, universe=gUniverse, annotation="org.Hs.eg.db")
list48 <- list(genes=g48, universe=gUniverse, annotation="org.Hs.eg.db")

geneLists <- list(T10=list10, T48=list48)
geneLists <- new("ccGeneList", geneLists, ccType=c("BP","KEGG"))
geneLists <- new("ccGeneList", geneLists, ccType=c("CC","KEGG"))

# set number of fdr runs to 0 to speed up runtime, not generally recommended.
geneLists <- new("ccGeneList", geneLists, ccType = c('BP','KEGG'), pvalueCutoff=0.01, fdr=0)
# enrichLists <- ccEnrich(geneLists)
```

`ccEnrichCollection-class` *Class "ccEnrichCollection"*

Description

Holds multiple classes of `ccEnrichResult` in one object to allow `ccCompare` to work on only the one object and generate all of the results of a comparison.

Objects from the Class

Objects can be created by calls of the form `new("ccEnrichCollection", ...)`.

Slots

`.Data`: Object of class `"list"`

`names`: Object of class `"character"` The names (generally GO ontologies or KEGG, but can be changed) of each set of results

Extends

Class `"namedList"`, directly. Class `"list"`, by class "namedList", distance 2. Class `"vector"`, by class "namedList", distance 3. Class `"AssayData"`, by class "namedList", distance 3.

Methods

pvalueCutoff<- `signature(r = "ccEnrichCollection")`: Changes the `pvalueCutoff` to be used to decide significant annotations for all of the contained `ccEnrichResult` objects

pvalueType<- `signature(object = "ccEnrichCollection")`: Changes whether to use `p-values` or `fdr` values to determine those annotations that are significant in all of the contained `ccEnrichResult` objects

minCount<- `signature(object = "ccEnrichCollection")`: how many features have to be annotated to a term to be reported as significant

graphType `signature(object = "ccEnrichCollection")`: Gets the type of graph that should be output for this collection

Author(s)

Robert M Flight

See Also

ccEnrich hyperGTestCC ccCompare ccEnrichResult

Examples

```
data(ccData)
enrichLists
```

---

ccEnrichResult-class *Class "ccEnrichResult"*

Description

Acts as a container object for multiple HyperGResultCC objects.

Objects from the Class

Objects can be created by calls of the form `new("ccEnrichResult", ...)`.

Extends

Class "namedList", directly. Class "list", by class "namedList", distance 2. Class "vector", by class "namedList", distance 3. Class "AssayData", by class "namedList", distance 3.

Methods

fdr `signature(object = "ccEnrichResult")`: get the number of runs using random feature lists were performed

pvalueCutoff `signature(r = "ccEnrichResult")`: what is the pvalueCutoff to determine significant annotations

pvalueCutoff<- `signature(r = "ccEnrichResult")`: change the pvalueCutoff for an annotation to be considered significant

pvalueType<- `signature(object = "ccEnrichResult")`: change whether p-values used are from "FDR" or raw p-values

minCount `signature(object = "ccEnrichResult")`: how many features need to belong to an annotation to be reported

minCount<- `signature(object = "ccEnrichResult")`: adjust the minCount

graphType `signature(object = "ccEnrichResult")`: what type of graph should be generated (generally set by the class of object)

graphType<- `signature(object = "ccEnrichResult")`: change the type of graph to generate by ccCompare

Author(s)

Robert M Flight

Examples

```
data(ccData)
enrichRes <- enrichLists[[1]]
fdr(enrichRes)
pvalueType(enrichRes)
enrichRes
pvalueType(enrichRes) <- 'pval'
enrichRes

pvalueCutoff(enrichRes)
pvalueCutoff(enrichRes) <- 0.01
enrichRes
```

---

ccGeneList-class *Class "ccGeneList"*

Description

This stores the actual gene lists and related information that will be used in categoryCompare.

Objects from the Class

Objects can be created by calls of the form `new("ccGeneList", list)`. `ccGeneList` is actually just an extension of R list objects. The input `list` should be a list of lists. See `Details` for more information.

Slots

`fdr`: Object of class `"numeric"` The number of `fdr` runs to perform to account for different list sizes and term dependence

`pvalueCutoff`: Object of class `"numeric"` Value used to determine whether or not a particular term is significant or not

`ccType`: Object of class `"character"` What types of annotations to use. Currently supported ones include "BP", "MF", "CC" (from Gene Ontology) and "KEGG"

`testDirection`: Object of class `"character"` Are you interested in "over" or "under" represented annotations

Methods

fdr `signature(object = "ccGeneList")`: how many random runs to perform

fdr<- `signature(object = "ccGeneList")`: change the number of random runs

pvalueCutoff `signature(object = "ccGeneList")`: what is the pvalue to consider significant

pvalueCutoff<- `signature(object = "ccGeneList")`: change the cutoff for significance

ccType `signature(object = "ccGeneList")`: what type of annotations are going to be examined

ccType<- `signature(object = "ccGeneList")`: change the type of annotations to examine

testDirection `signature(object = "ccGeneList")`: query for "over" or "under" represented annotations

testDirection<- `signature(object = "ccGeneList")`: change the type of representation ("over" or "under")

listNames `signature(object = "ccGeneList")`: what are the names of the lists contained

Details

The input list should be a list of lists, with at least three sub-lists.

`testList <- list(list1=list(genes='...',universe='...',annotation='...'), list2=list(...))`

genes : These are the gene identifiers of the genes that are of interest (differentially expressed genes)

universe : All of the genes that were measured in this particular experiments (i.e. all the genes on the chip)

annotation : What organism or chip do these ID's come from (e.g. "org.Hs.eg.db" for Human Entrez gene ID's, "hgu133a.db" for probe ID's from the Affymetrix U133A chip)

data : A data-frame that contains extra information about the genes of interest. At the very least, the data-frame must have a column `ID` that matches the ID's contained in `genes`

What actually happens when running `ccEnrich` is that the appropriate `HyperGParamsCC` objects are generated for each geneList and each type of annotation (e.g. BP, CC, KEGG), and then the calculations performed on each one.

Note

The `ccGeneList` object is what will undergo all of the enrichment calculations. When the results are combined with the `ccOptions` object, we can get our results of actual comparisons between experiments.

Author(s)

Robert M Flight

See Also

`ccOptions`

Examples

```
data(ccData)
g10 <- (unique(table10$Entrez[1:100]))
g48 <- (unique(table48$Entrez[1:100]))

list10 <- list(genes=g10, universe=gUniverse, annotation="org.Hs.eg.db")
list48 <- list(genes=g48, universe=gUniverse, annotation="org.Hs.eg.db")

geneLists <- list(T10=list10, T48=list48)
geneLists <- new("ccGeneList", geneLists, ccType=c("BP","KEGG"))
geneLists
```

ccOptions-class *Class "ccOptions"*

Description

These objects store the various options required by categoryCompare for actually making comparisons and generating output.

Objects from the Class

Objects can be created by calls of the form `new("ccOptions", listNames=c('list1','list2',etc))`. This is the minimum call required, and will generate a `ccOptions` object where comparisons are assumed between all the lists supplied. See the examples section for more examples of how to initialize new objects.

Slots

`listNames`: Object of class `"character"` The actual names of the various datasets defined in the `ccData` object

`compareNames`: Object of class `"character"` Which lists to compare, each entry should be a comma separated list

`compareIndx`: Object of class `"list"` List indices for each of the comparison, not usually set by the user. Generated automatically.

`compareColors`: Object of class `"character"` For graphical and tabular output each comparison can be colored. Should be one color for each comparison. Can be either an `n` by 3 matrix of rgb triples, or a character vector of hexadecimal color codes, or character vector of color names ('red','green','blue', etc)

`cssClass`: Object of class `"character"` Classnames used when generating HTML tables to color entries. Generated automatically upon initialization, or modifying `compareNames`

`outType`: Object of class `"character"` Sets the type of output generated by `ccTables`. Valid types are "html", "text", "rcy3" or "none", default is "text" when the `ccOptions` object is initialized without an `outType` specified.

Methods

compareColors `signature(object = "ccOptions")`: ...

compareColors<- `signature(object = "ccOptions")`: ...

compareIndx `signature(object = "ccOptions")`: ...

compareNames `signature(object = "ccOptions")`: ...

compareNames<- `signature(object = "ccOptions")`: ...

cssClass `signature(object = "ccOptions")`: ...

listNames `signature(object = "ccOptions")`: ...

listNames<- `signature(object = "ccOptions")`: ...

outType `signature(object = "ccOptions")`: ...

outType<- `signature(object = "ccOptions")`: ...

Author(s)

Robert M Flight

Examples

```
showClass("ccOptions")
## A very basic "ccOptions" for a comparison of two sets of data, "list1" and "list2"
c1 <- new("ccOptions", listNames=c('list1','list2'))
c1

## Now lets get a little more complicated
c1 <- new("ccOptions", listNames=c('list1','list2'),
compareNames=c('list1,list2','list1,list3'), compareColors=c('red','blue'))
c1

# set the type of output you want to eventually produce
c1 <- new("ccOptions", listNames=c('list1','list2'), outType='html')
c1

c1 <- new("ccOptions", listNames=c('list1','list2'), outType=c('html','text','none'))
c1

## Using RGB colors
ccCols <- matrix(c(255,0,0, 0,0,255), nrow=2, ncol=3)
ccCols <- rgb(ccCols, maxColorValue=255)
c1 <- new("ccOptions", listNames=c('list1','list2','list3'),
compareNames=c('list1,list2','list1,list3'), compareColors=ccCols)

## Using Hex colors
c1 <- new("ccOptions", listNames=c('list1','list2','list3'),
compareNames=c('list1,list2','list1,list3'), compareColors=c('#FF0000','#0000FF'))
c1

## or even using a color palette from R.
## Note that you need at least enough colors to cover all of individual and
## possible permutations (n!) if you use compareNames='all'
c1 <- new("ccOptions", listNames=c('list1','list2','list3'),
compareNames=c('list1,list2','list1,list3'), compareColors=rainbow(4))
c1
```

---

ccOutCyt-methods *Methods for Function* ccOutCyt *in Package* **categoryCompare**

Description

Passes a ccCompareResult object to Cytoscape for interactive visualization of ccCompare results.

Details

Note that only some basic, required methods have been imported from RCy3 for use with categoryCompare, and these are hidden in the functions within categoryCompare and are not visible to the user. If

access to all the functionality of RCytoscape is desired (and trust me, there is a lot of useful stuff in there), then the user should use library(RCy3) directly.

It should also be noted that deletion of edges via RCy3 is slow, so some edge filtering should be done by breakEdges prior to using ccOutCyt.

Methods

signature(ccCompRes = "ccCompareResult", ccOpts = "ccOptions", ...) At a minimum, this method requires a ccCompareResult and a ccOptions to work.

... may include:

layout = "character" to override the default layout set by ccCompare, as well as options

postText = "character" to add a user set string to the Cytoscape window

In addition, any of the arguments to CytoscapeWindow may also be set, such as host or port.

See Also

ccCompareResult ccOptions ccCompare CytoscapeWindowClass

Examples

```
## Not run:
hasCy <- (if (.Platform$OS.type %in% "windows") {
(length(grep("Cytoscape", system("tasklist", intern=TRUE))) > 0)})

if hasCy {

 ccResults$BP <- breakEdges(ccResults$BP, 0.8)
 cwObj <- ccOutCyt(ccResults$BP,ccOpts)
 Sys.sleep(10)
 RCy3::deleteWindow(cwObj)
 }

## End(Not run)
```

---

ccSigList-class *Class* "ccSigList"

Description

Holds a generic list of significant annotations. Allows one to use Bioconductor annotation packages, or when combined into a GENccEnrichResult, use custom annotation / gene mappings.

Objects from the Class

Objects can be created by calls of the form new("ccSigList", ...).

Slots

sigID: Object of class "character"

categoryName: Object of class "character"

ontology: Object of class "character"

annotation: Object of class "character"

Methods

annotation signature(object = "ccSigList"): ...

category signature(object = "ccSigList"): ...

ontology signature(object = "ccSigList"): ...

sigID signature(object = "ccSigList"): ...

Author(s)

Robert M Flight

See Also

GENccEnrichResult ccCompareGeneric

Examples

```
showClass("ccSigList")
```

---

cwReload-methods *Methods for Function* cwReload *in Package* **categoryCompare**

Description

Methods for function cwReload in package **categoryCompare**

Methods

signature(oldCW = "numeric", windowName = "character", ccOpts = "ccOptions") This method is now deprecated as the cwObj doesn't store anything, but is merely the network SUID pointing to the network in Cytoscape. See RCy3::getNetworkSuid.

---

cytOutData-methods *Methods for Function* cytOutData

Description

Takes the saveObj generated by cytOutNodes and writes the data to a file

Value

A text file with the annotations previously saved using cytOutNodes

Methods

signature(saveObj = "list", compareResult = "ccCompareResult", mergedData = "mergedData") saveObj is the list object generated by cytOutNodes, compareResult is the object from ccCompare, and mergedData is created using mergeLists, but is optional.

... : optional arguments also include: orgType, default is "header" where each group is seperate, "annotate" pushes all the data into one table with a new column that designates which groups the annotation was found in; fileName, the name of a text file to output the results to; displayFile, whether or not to display the file (default is "FALSE")

Examples

```
## Not run:
hasCy <- (if (.Platform$OS.type %in% "windows") {
(length(grep("Cytoscape", system("tasklist", intern=TRUE))) > 0)})

if hasCy {
 data(ccData)
 ccResults$BP <- breakEdges(ccResults$BP, 0.8)
 cwObj <- ccOutCyt(ccResults$BP,ccOpts)
 # user selects some nodes in Cytoscape
 RCy3::selectNodes(cwObj, c("GO:0007017", "GO:0000226", "GO:0007051", "GO:0007052"))
 savedNodes <- cytOutNodes("random1", cwObj) # save them
 # and selects some other nodes
 RCy3::selectNodes(cwObj,
 c("GO:0071103", "GO:0034728", "GO:0006323", "GO:0030261", "GO:0006334"),
 preserve.current.selection=FALSE)
 savedNodes <- cytOutNodes("random2", cwObj, savedNodes)

 # now spit results out to a file
 cytOutData(savedNodes, ccResults$BP)
}
## End(Not run)
```

`cytOutNodes-methods` *Methods for Function* `cytOutNodes`

Description

Allows export of currently selected nodes in the Cytoscape window for data export

Methods

`signature(descStr = "character", cwObj = "numeric", saveObj = "list")` `descStr` is a string describing the nodes that are currently selected, `cwObj` is the `CytoscapeWindow` that the nodes are in, and then `saveObj` is a previously generated `cytOutNodes` list, and is optional.

Examples

```
## Not run:
hasCy <- (if (.Platform$OS.type %in% "windows") {
(length(grep("Cytoscape", system("tasklist", intern=TRUE))) > 0)})

if hasCy {
 ccResults$BP <- breakEdges(ccResults$BP, 0.8)
 cwObj <- ccOutCyt(ccResults$BP,ccOpts)
 # user selects some nodes in Cytoscape
 RCy3::selectNodes(cwObj, c("GO:0007017", "GO:0000226", "GO:0007051", "GO:0007052"))
 savedNodes <- cytOutNodes("random1", cwObj) # save them
 # and selects some other nodes
 RCy3::selectNodes(cwObj,
 c("GO:0071103", "GO:0034728", "GO:0006323", "GO:0030261", "GO:0006334"),
 preserve.current.selection=FALSE)
 savedNodes <- cytOutNodes("random2", cwObj, savedNodes)
```

```
 }

## End(Not run)
```

`fdr` *Number of FDR runs to perform*

Description

Queries or sets the number of random runs to perform to generate an estimate of the false discovery rate. Defaults to 50

Usage

```
fdr(object)
```

Arguments

`object`	Can be `ccGeneList`, `HyperGParamsCC`, `HyperGResultCC`, `ccEnrichResult` See Details for more information.

Details

`fdr(object)` gets the number of `fdr` runs for `ccGeneList`, `HyperGParamsCC`, `HyperGResultCC`, `ccEnrichResult`

`fdr(object)<-` will set the number of `fdr` runs to be used by `ccEnrich` and `HyperGTestCC` when performing calculations on either a `ccGeneList` or `HyperGParamsCC`, respectively

Author(s)

Robert M Flight

See Also

`HyperGResultCC` `ccEnrichResult` `ccGeneList` `HyperGParamsCC`

`GENccEnrichResult-class`

Class `"GENccEnrichResult"`

Description

Holds generic `ccEnrich` type results

Objects from the Class

Objects can be created by calls of the form `new("GENccEnrichResult", ...)`.

Slots

`.Data`: Object of class "list" The actual list containing the ccEnrichResults

`categoryName`: Object of class "character"

`ontology`: Object of class "character"

`geneAnnMapping`: Object of class "namedList"

`graphType`: Object of class "character"

`names`: Object of class "character"

Extends

Class "namedList", directly. Class "list", by class "namedList", distance 2. Class "vector", by class "namedList", distance 3. Class "AssayData", by class "namedList", distance 3.

Methods

[`signature(x = "GENccEnrichResult", i = "ANY", j = "ANY")`: Subsets the object to just those lists that are desired

categoryName `signature(object = "GENccEnrichResult")`:

ccCompareGeneric `signature(gccResult = "GENccEnrichResult", ccOptions = "ccOptions")`: ...

geneAnnMapping `signature(object = "GENccEnrichResult")`: ...

graphType `signature(object = "GENccEnrichResult")`: ...

graphType<- `signature(object = "GENccEnrichResult")`: ...

ontology `signature(object = "GENccEnrichResult")`: ...

Author(s)

Robert M Flight

See Also

ccCompareGeneric ccSigList

Examples

```
data(ccData)
locA <- grep("A",gseaRes$Tissues)
locL <- grep("L",gseaRes$Tissues)
locM <- grep("M",gseaRes$Tissues)

A <- new("ccSigList",sigID=gseaRes$KEGGID[locA],categoryName="KEGG",annotation="org.Mm.eg")
L <- new("ccSigList",sigID=gseaRes$KEGGID[locL],categoryName="KEGG",annotation="org.Mm.eg")
M <- new("ccSigList",sigID=gseaRes$KEGGID[locM],categoryName="KEGG",annotation="org.Mm.eg")
ccEnrichCol <- list(A=A,L=L,M=M)
ccEnrichCol <- new("GENccEnrichResult", ccEnrichCol, categoryName="KEGG")
```

`getGeneSymbol` *Entrez to name, symbol, GO and path conversion, as well as general ID to ID conversion.*

Description

Get different attributes for the Entrez gene Ids

Usage

```
getGeneSymbol(id, annPackage)
getGeneName(id, annPackage)
getGO2ALLEGS(id, annPackage)
getPATH2EG(id, annPackage)
getAnnotation(id, annPackage, mapID, doUnlist=TRUE)
```

Arguments

`id`	The IDs one wants to get information for.
`annPackage`	Which annotation package to use.
`mapID`	Which mapping to use
`doUnlist`	should the results be unlisted or not?

Details

The type of ID will change depending on the function. For `getGene...` the ID should be Entrez IDs. For `getGO2ALLEGS` Gene Ontology IDs should be used, and for `getPATH2EG` KEGG pathways IDs should be used. For `getAnnotation`, any ID can be used.

Value

Returns the requested information.

Note

These functions are generally called internally for mapping between genes and various objects.

Author(s)

Robert M Flight

graphType-methods *graphType*

Description

Gets and sets the graphType for a couple of different ccEnrichResults objects

Methods

signature(object = "ccEnrichResult")

signature(object = "GENccEnrichResult")

See Also

ccEnrichResult GENccEnrichResult

HyperGParamsCC-class *Class "HyperGParamsCC"*

Description

This class extends the HyperGParams class in Category by providing options for multiple testing and the storing of extra data in addition to the gene list of interest (not currently used, but might be in the future).

Objects from the Class

Objects can be created by calls of the form new("HyperGParamsCC", ...). In general the user will not create these directly, but they are created and used by to carry out the enrichment calculations.

Slots

fdr: Object of class "numeric" The number of FDR runs to perform

data: Object of class "data.frame" Extra data stored in the object

geneIds: Object of class "ANY" The genes of interest

universeGeneIds: Object of class "ANY" The gene universe or background used (all the genes on the chip)

annotation: Object of class "character" The annotation package used to get information about the geneIds

datPkg: Object of class "DatPkg" Generated automatically from the annotation slot

categorySubsetIds: Object of class "ANY" A specific set of category IDs that one wants to restrict the testing to

categoryName: Object of class "character" What type of category to use, currently either "GO" or "KEGG"

pvalueCutoff: Object of class "numeric" What should be the p-value to decide significance

testDirection: Object of class "character" "over" or "under" represented annotation terms

Extends

Class "GOHyperGParams", directly.

Methods

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

Author(s)

Robert M Flight

See Also

HyperGResultCC ccEnrich Category-package

Examples

```
showClass("HyperGParamsCC")
```

HyperGResultCC-class *Class "HyperGResultCC"*

Description

Contains the results of performing a hypergeometric test on a HyperGParams object.

Objects from the Class

Objects can be created by calls of the form `new("HyperGResultCC", ...)`.

Slots

fdr: Object of class "numeric" The number of FDR runs performed

fdrvalues: Object of class "numeric" The FDR values generated

pvalueType: Object of class "character" Whether to use p-values or FDR values in determing the significant terms returned

data: Object of class "data.frame" Extra data

pvalues: Object of class "numeric" P-values calculated for each term

oddsRatios: Object of class "numeric"

expectedCounts: Object of class "numeric"

catToGeneId: Object of class "list"

organism: Object of class "character"

annotation: Object of class "character"

geneIds: Object of class "ANY"

testName: Object of class "character"

pvalueCutoff: Object of class "numeric"

testDirection: Object of class "character"

Extends

Class "HyperGResult", directly. Class "HyperGResultBase", by class "HyperGResult", distance 2.

Methods

fdr `signature(object = "HyperGResultCC")`: ...

fdrvalues `signature(object = "HyperGResultCC")`: ...

pCC `signature(object = "HyperGResultCC")`: ...

pvalueCutoff<- `signature(r = "HyperGResultCC")`: ...

pvalueType `signature(object = "HyperGResultCC")`: ...

pvalueType<- `signature(object = "HyperGResultCC")`: ...

minCount `signature(object = "HyperGResultCC")`: ...

minCount<- `signature(object = "HyperGResultCC")`: ...

Author(s)

Robert M Flight

See Also

hyperGTestCC

Examples

```
showClass("HyperGResultCC")
```

---

hyperGTestCC *Hypergeometric testing with false discovery rate*

Description

Performs the hypergeometric testing for HyperGParamsCC objects.

Usage

```
hyperGTestCC(p)
```

Arguments

p	A HyperGParamsCC object

Details

This is the heart of categoryCompare, the function that calculates the HyperGeometric statistics for the given categories of annotation for each gene list.

Value

Returns a HyperGResultCC object

Author(s)

Robert M Flight

See Also

HyperGParamsCC HyperGResultCC GOHyperGParamsCC KEGGHyperGParamsCC GOHyperGResultCC KEGGHyperGResultCC

Examples

```
require(GO.db)
require(org.Hs.eg.db)
data(ccData)
g10 <- unique(table10$Entrez)
testGO <- new("GOHyperGParamsCC", geneIds=g10, universeGeneIds=gUniverse,
annotation="org.Hs.eg.db", ontology="CC", conditional=FALSE,
testDirection="over",fdr=0, pvalueCutoff = 0.01)
# ccHypRes <- hyperGTestCC(testGO)
# summary(ccHypRes)
```

---

listNames *listNames*

Description

Extracts the listNames from ccGeneList or ccOptions objects.

Usage

```
listNames(object)
```

Arguments

object	This will be either a ccGeneList or ccOptions object

Author(s)

Robert M Flight

See Also

ccGeneList ccOptions

mergedData-class *Class* "mergedData"

Description

Stores merged data tables from the "data" entry in a ccGeneList. This is useful for output later.

Objects from the Class

Objects can be created by calls of the form new("mergedData", ...).

Slots

.Data: Object of class "list"

useIDName: Object of class "character"

names: Object of class "character"

row.names: Object of class "data.frameRowLabels"

.S3Class: Object of class "character"

Extends

Class "data.frame", directly. Class "list", by class "data.frame", distance 2. Class "oldClass", by class "data.frame", distance 2. Class "data.frameOrNULL", by class "data.frame", distance 2. Class "vector", by class "data.frame", distance 3.

Methods

signature(saveObj = "list", compareResult = "ccCompareResult", mergedData = "mergedData")

Author(s)

Robert M. Flight

See Also

mergeLists cytOutData

Examples

```
showClass("mergedData")

data(ccData)
mergeDat <- mergeLists(geneLists, ccOpts)
```

mergeLists-methods *Function* mergeLists *in Package* **categoryCompare**

Description

Merges the gene lists or the data tables from a ccGeneList object, providing a single table with all the input data, that can then be queried later, using cytTableOut

Usage

```
mergeLists(ccGeneList,ccOptions,isGene=TRUE)
```

Arguments

ccGeneList	a ccGeneList object
ccOptions	a ccOptions object
isGene	are the identifiers genes, or something else (metabolites, etc)

Value

A mergedData object which is really just a glorified data frame. If the ccGeneList input had a data list, then these are all merged into a single table. Otherwise, it contains just the gene names and which list they were present in.

Methods

```
signature(ccGeneList = "ccGeneList", ccOptions = "ccOptions")
```

See Also

ccGeneList ccOptions mergedData

Examples

```
data(ccData)
g10 <- (unique(table10$Entrez[1:100]))
g48 <- (unique(table48$Entrez[1:100]))

list10 <- list(genes=g10, universe=gUniverse, annotation="org.Hs.eg.db", data=table10[1:100,])
list48 <- list(genes=g48, universe=gUniverse, annotation="org.Hs.eg.db", data=table48[1:100,])

geneLists <- list(T10=list10, T48=list48)
geneLists <- new("ccGeneList", geneLists, ccType=c("BP","KEGG"))
ccOpts <- new("ccOptions", listNames = names(geneLists))
mergedDat <- mergeLists(geneLists,ccOpts)

list10 <- list(genes=g10, universe=gUniverse, annotation="org.Hs.eg.db")
list48 <- list(genes=g48, universe=gUniverse, annotation="org.Hs.eg.db")
geneLists <- list(T10=list10, T48=list48)
geneLists <- new("ccGeneList", geneLists, ccType=c("BP","KEGG"))
ccOpts <- new("ccOptions", listNames = names(geneLists))
mergedDat <- mergeLists(geneLists,ccOpts)
```

minCount *minCount*

Description

Extracts and sets the minimum number of genes that an annotation must have to be considered in subsequent steps.

Usage

```
minCount(object)
```

Arguments

object	This will be either a HyperGResultCC, ccEnrichResult, or ccEnrichCollection object. See Details for more information.

Details

`minCount(object)` fetches the set `minCount` for `HyperGResultCC` and `ccEnrichResult` objects

`minCount(object)<-` will set the `minCount` for `HyperGResultCC` objects, and when applied to `ccEnrichResult` and `ccEnrichCollection` sets the `minCount` for all of the contained objects, so be careful if you want to use different `minCounts` for different results

Author(s)

Robert M Flight

See Also

HyperGResultCC ccEnrichResult ccEnrichCollection

Examples

```
data(ccData)
enrichLists
minCount(enrichLists) <- 5
enrichLists
```

minNodes *Delete nodes with less than a certain number of genes annotated*

Description

Deletes from the graph those annotations with less than a certain number of genes

Usage

```
minNodes(cwObj, cutoff)
```

Arguments

cwObj	a CytoscapeWindowClass object returned from ccOutCyt
cutoff	the minimum number of genes that an annotation must have

Author(s)

Robert M Flight

See Also

CytoscapeWindowClass ccOutCyt

Examples

```
## Not run:
hasCy <- (if (.Platform$OS.type %in% "windows") {
(length(grep("Cytoscape", system("tasklist", intern=TRUE))) > 0)})

if hasCy {
 data(ccData)
 ccResults$BP <- breakEdges(ccResults$BP, 0.8)
 cwObj <- ccOutCyt(ccResults$BP,ccOpts)

 minNodes(cwObj, 5)

}
## End(Not run)
```

---

pvalueType *Type of p-values to return from object*

Description

Queries or sets the type of p-values to return from objects, either base calculated (pvals) or from fdr calculations (fdr)

Usage

```
pvalueType(object)
```

Arguments

object	Can be HyperGResultCC, ccEnrichResult, ccEnrichCollection. See Details for more information

Details

pvalueType(object) gets the type of p-values to be returned from HyperGResultCC and ccEnrichResult objects

pvalueType(object)<- will set the type of p-values to be returned from HyperGResultCC, ccEnrichResult, ccEnrichCollection. Note that for a ccEnrichCollection, the type is changed for all contained ccEnrichResults

Author(s)

Robert M Flight

See Also

HyperGResultCC ccEnrichResult ccEnrichCollection

Examples

```
# pvalueType-Methods
data(ccData)

## Not run: pvalueType(enrichLists) # this returns an error
pvalueType(enrichLists[[1]])
pvalueType(enrichLists[[1]][[1]])

# change the type for one of the results
pvalueType(enrichLists[[1]]) <- 'pval'  # Not recommended practice
enrichLists

# change for all of the results
pvalueType(enrichLists) <- 'pval'
enrichLists
```

---

resetColors-methods *resetColors*

Description

If the color of particular nodes have been modified from the original color scheme in ccOptions, this will reset them

Methods

signature(cwObj = "numeric", ccOpts = "ccOptions") What CytoscapeWindow to apply this to, and what ccOptions to use for the color scheme.

Optional Arguments: Note that optional arguments include node.attribute.name (default is 'fillcolor') and mode (default is 'lookup')

Author(s)

Robert M Flight

See Also

ccOptions setNodeColorMapping

show-methods *Methods for Function show in Package 'categoryCompare'*

Description

The show and summary methods for HyperGResultCC objects generated using hyperGTestCC

Methods

show, signature(object = "HyperGResultCC")

summary, signature(object = "HyperGResultCC")

Author(s)

Robert M Flight

Examples

```
## Not run:
data(ccData)
show(enrichLists)
summary(enrichLists[[1]][[1]])

## End(Not run)
```

Index