Package ‘categoryCompare’

November 21, 2016

Version 1.18.0

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, methods, GO.db, KEGG.db, estrogen, org.Hs.eg.db, hgu95av2.db, limma, affy, genefilter

Imports AnnotationDbi, hwriter, GSEABase, Category (>= 2.33.1), GOstats, annotate, colorspace, graph, RCytoscape (>= 1.5.11)

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 (>= 2.8.0) (if used for visualization of results, heavily suggested), CytoscapeRPC plugin (>= 1.8)

TODO Text and HTML output without graphs.

biocViews Annotation, GO, MultipleComparison, Pathways, GeneExpression

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

categoryCompare-package .................................................. 2
breakEdges ................................................................. 4
breakEdges-methods ...................................................... 5
ccCompare-methods ....................................................... 6
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: 0.99.1
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, hgu95av2.db
Imports: Biobase (>= 1.15.29), AnnotationDbi (>= 0.1.15), hwriter, GSEABase, Category (>= 2.21.2), GOstats, annotate, colorspace, graph, RCytoscape (>= 1.5.11)
LazyLoad: yes
biocViews: Bioinformatics, Annotation, GO, MultipleComparisons, Pathways, GeneExpression
SystemRequirements: Cytoscape (>= 2.8.0) (if used for visualization of results, heavily suggested), CytoscapeRPC plugin
TODO: Text and HTML output without graphs.
Built: R 2.15.0; ; 2012-03-15 18:42:40 UTC; windows

Index:

GENccEnrichResult-class Class "GENccEnrichResult"
HyperGParamsCC-class Class "HyperGParamsCC"
HyperGResultCC-class Class "HyperGResultCC"
breakEdges Break Cytoscape (or graphNEL) Network Edges
breakEdges-methods Methods for Function 'breakEdges' in Package 'categoryCompare'
categoryCompare-package Meta-analysis of high-throughput experiments using feature annotations
ccCompare-methods Comparison of enriched annotations
ccCompareCollection-class Class "ccCompareCollection"
ccCompareGeneric-methods Methods for Function 'ccCompareGeneric' in Package 'categoryCompare'
cccCompareResult-class Class "ccCompareResult"
ccData Test data for 'categoryCompare'
ccEnrich-method Perform annotation enrichment for multiple gene lists
ccEnrichCollection-class Class "ccEnrichCollection"
ccEnrichResult-class Class "ccEnrichResult"
ccGeneList-class Class "ccGeneList"
ccOptions-class Class "ccOptions"
ccOutCyt-methods Methods for Function 'ccOutCyt' in Package 'categoryCompare'
ccSigList-class Class "ccSigList"
cwReload-methods Methods for Function 'cwReload' in Package 'categoryCompare'
cytOutData-methods Methods for Function 'cytOutData'
cytOutNodes-methods Methods for Function 'cytOutNodes'
fdr Number of FDR runs to perform
getGeneSymbol Entrez to name, symbol, GO and path conversion, as well as general ID to ID conversion.
graphType-methods graphType
hyperGTestCC Hypergeometric testing with false discovery rate
listNames listNames
mergeLists-methods Function 'mergeLists' in Package 'categoryCompare'
mergedData-class Class "mergedData"
in minCount

**breakEdges**

<table>
<thead>
<tr>
<th>minNodes</th>
<th>Delete nodes with less than a certain number of genes annotated</th>
</tr>
</thead>
<tbody>
<tr>
<td>pvalueType</td>
<td>Type of p-values to return from object</td>
</tr>
<tr>
<td>resetColors-methods</td>
<td>resetColors</td>
</tr>
<tr>
<td>show-methods</td>
<td>Methods for Function show in Package 'categoryCompare'</td>
</tr>
</tbody>
</table>

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**  **Break Cytoscape (or graphNEL) Network Edges**

**Description**

Removes those edges in a graph network with the edge attribute that is "under" or "over" the cutoff value supplied.

**Usage**

```
breakEdges(cwObject, cutoff, edgeAtt='weight', valDir='under', layout='force-directed')
```

**Arguments**

- **cwObject**: The object returned by `ccOutCyt`
- **cutoff**: What is the cutoff to select edges
- **edgeAtt**: Which edge attribute should be used for deciding which edges to select
- **valDir**: Select the edge attribute with values "under" (default) or "over" the input value
- **layout**: What type of layout should be used after the edges are broken

**Details**

When viewing annotations returned by `ccCompare`, unless a hierarchical view is used, the annotations are linked by edges weighted by how many genes are shared between each annotation. The default cutoff is 0.1, any edges above that are kept. This can result in a big hairy mess when viewing the annotation network in Cytoscape. To help clean up the display and encourage finding functionally related sets of annotations, it is very useful to remove edges with low values of overlap, and re-layout the network. `breakEdges` does just that.

The other option is to break the highly overlapping edges prior to sending the network to Cytoscape, thereby speeding up the process, as the fewer edges one has to "push" to Cytoscape the faster it will go.
Value

The first returns nothing, rather it modifies the network in Cytoscape itself. The second returns a new \texttt{ccCompareResult} object for use by \texttt{ccOutCyt}.

Author(s)

Robert M Flight

See Also

\texttt{ccOutCyt breakEdges-methods}

Examples

```r
## Not run:
require(RCytoscape)
g <- makeSimpleGraph()
cw <- CytoscapeWindow('breakEdges',g)
displayGraph(cw)
layout(cw)
redraw(cw)
breakEdges(cw,'score',0,layout=NULL)
## End(Not run)
```

Description

Methods for function \texttt{breakEdges} in package \texttt{categoryCompare}

Methods

\begin{itemize}
  \item signature(cwObject = "ccCompareResult", cutoff = "numeric") Allows one to remove edges in the \texttt{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.
  \item signature(cwObject = "CytoscapeWindowClass", 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 \texttt{ccCompareResult} object.
\end{itemize}

Author(s)

Robert M Flight

See Also

\texttt{CytoscapeWindowClass 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))))})

if hasCy {

cwObj <- ccOutCyt(ccResults$BP,ccOpts)
# now breaking them in the CytoscapeWindow object
breakEdges(cwObj, 0.85)
Sys.sleep(10)
RCytoscape::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:
Annotations arranged as a graph

The tabular results from all enrichment calculations combined into one

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 RCytoscape 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

cCompareResult ccCompareCollection ccOutCyt breakEdges outType ccEnrich

Examples

```r
## 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
```

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


Methods

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

Author(s)

Robert M Flight

See Also

ccCompareResult ccCompare

Examples

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

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")`
**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)
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

```r
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

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

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


**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
Class "ccEnrichResult"

Description
Acts as a container object for multiple \texttt{HyperGResultCC} objects.

Objects from the Class
Objects can be created by calls of the form \texttt{new("ccEnrichResult", \ldots)}.

Extends
Class \texttt{"namedList"}, directly. Class \texttt{"list"}, by class \texttt{"namedList"}, distance 2. Class \texttt{"vector"}, by class \texttt{"namedList"}, distance 3. Class \texttt{"AssayData"}, by class \texttt{"namedList"}, distance 3.

Methods
\begin{description}
\item[fdr] \texttt{signature(object = "ccEnrichResult")}: get the number of runs using random feature lists were performed
\item[pvalueCutoff] \texttt{signature(r = "ccEnrichResult")}: what is the pvalueCutoff to determine significant annotations
\item[pvalueCutoff<-] \texttt{signature(r = "ccEnrichResult")}: change the pvalueCutoff for an annotation to be considered significant
\item[pvalueType<-] \texttt{signature(object = "ccEnrichResult")}: change whether p-values used are from "FDR" or raw p-values
\item[minCount] \texttt{signature(object = "ccEnrichResult")}: how many features need to belong to an annotation to be reported
\item[minCount<-] \texttt{signature(object = "ccEnrichResult")}: adjust the minCount
\item[graphType] \texttt{signature(object = "ccEnrichResult")}: what type of graph should be generated (generally set by the class of object)
\item[graphType<-] \texttt{signature(object = "ccEnrichResult")}: change the type of graph to generate by \texttt{ccCompare}
\end{description}

Author(s)
Robert M Flight
Examples

```r
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
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", "cytoscape" 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)
c1

## Using Hex colors
ccCols <- matrix(c('#FF0000', '#0000FF', '#00FF00'), nrow=3, byrow=TRUE)
c1 <- new("ccOptions", listNames=c('list1','list2','list3'), compareNames=c('list1,list2','list1,list3'), compareColors=c('#FF0000', '#0000FF', '#00FF00'))
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=
ccCols <- matrix(rainbow(4), nrow=1, ncol=4)
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 RCytoscape 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(RCytoscape) directly.
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)
  RCytoscape::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"): ...
cwReload-methods

Author(s)
Robert M Flight

See Also
GENccEnrichResult ccCompareGeneric

Examples
showClass("ccSigList")
## Not run:

```r
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
  RCytoscape::selectNodes(cwObj, c("GO:0007017", "GO:0000226", "GO:0007051", "GO:0007052"))
  savedNodes <- cytOutNodes("random1", cwObj) # save them
  # and selects some other nodes
  RCytoscape::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**

```r
signature(descStr = "character", cwObj = "CytoscapeWindowClass", 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:

```r
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
  RCytoscape::selectNodes(cwObj, c("GO:0007017", "GO:0000226", "GO:0007051", "GO:0007052"))
  savedNodes <- cytOutNodes("random1", cwObj) # save them
  # and selects some other nodes
  RCytoscape::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)
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


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)
gPATH2EG(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
HyperGParamsCC-class

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
HyperGResultCC-class

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 determining 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"
hyperGTestCC

Extends

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

Methods

\textbf{fdr} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{fdrvalues} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{pCC} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{pvalueCutoff<-} \quad \text{signature(r = "HyperGResultCC"): ...}
\textbf{pvalueType} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{pvalueType<-} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{minCount} \quad \text{signature(object = "HyperGResultCC"): ...}
\textbf{minCount<-} \quad \text{signature(object = "HyperGResultCC"): ...}

Author(s)

Robert M Flight

See Also

hyperGTestCC

Examples

\begin{verbatim}
showClass("HyperGResultCC")
\end{verbatim}

\begin{verbatim}
hyperGTestCC
\end{verbatim}

Description

Performs the hypergeometric testing for \texttt{HyperGParamsCC} objects.

Usage

\begin{verbatim}
hyperGTestCC(p)
\end{verbatim}

Arguments

\begin{verbatim}
p \quad \text{A HyperGParamsCC object}
\end{verbatim}

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 \texttt{HyperGResultCC} object
listNames

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)

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
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

```r
showClass("mergedData")
data(ccData)
mergeDat <- mergeLists(geneLists, ccOpt)
```
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

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)
pvalueType

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

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

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
resetColors-methods

Description

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

Methods

signature(cwObj = "CytoscapeWindowClass", 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’)

Note

This is most commonly used with the cwReload function, as the color scheme of the network does not get saved in the CYS file.

Author(s)

Robert M Flight

See Also

CytoscapeWindowClass ccOptions setNodeColorRule cwReload
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

*Topic classes
  ccCompareCollection-class, 7
  ccCompareResult-class, 9
  ccEnrichCollection-class, 12
  ccEnrichResult-class, 13
  ccGeneList-class, 14
  ccOptions-class, 16
  ccSigList-class, 18
  GENccEnrichResult-class, 21
  HyperGParamsCC-class, 24
  HyperGResultCC-class, 25
  mergedData-class, 28

*Topic datasets
  ccData, 10

*Topic methods
  breakEdges-methods, 5
  ccCompareGeneric-methods, 8
  cwReload-methods, 19
  cytOutData-methods, 19
  cytOutNodes-methods, 20
  graphType-methods, 24
  mergeLists-methods, 29
  resetColors-methods, 32
  show-methods, 33

*Topic other possible keyword(s)
  breakEdges-methods, 5
  ccCompareGeneric-methods, 8
  cwReload-methods, 19
  cytOutNodes-methods, 20
  graphType-methods, 24
  resetColors-methods, 32

*Topic package
  categoryCompare-package, 2
  [,GENccEnrichResult,ANY,ANY,ANY-method
    (GENccEnrichResult-class), 21
  [,GENccEnrichResult,ANY,ANY-method
    (GENccEnrichResult-class), 21
  [,ccEnrichResult,ANY,ANY,ANY-method
    (ccEnrichResult-class), 13
  [,ccEnrichResult,ANY,ANY-method
    (ccEnrichResult-class), 13
  allAnnotation,ccCompareResult-method
    (ccCompareResult-class), 9
  annotation,ccSigList-method
    (ccSigList-class), 18
  AssayData, 8, 12, 13, 22
  breakEdges, 4, 5, 7
  breakEdges,ccCompareResult,numERIC-method
    (breakEdges-methods), 5
  breakEdges,CytoscapeWindowClass,numERIC-method
    (breakEdges-methods), 5
  breakEdges-methods, 5
  category,ccSigList-method
    (ccSigList-class), 18
  category,GENccEnrichResult-method
    (GENccEnrichResult-class), 21
  categoryCompare
    (categoryCompare-package), 2
  categoryCompare-package, 2
  ccCompare, 4, 8-10, 12, 13, 18
  ccCompare (ccCompare-methods), 6
  ccCompare,ccEnrichCollection,ccOptions-method
    (ccCompare-methods), 6
  ccCompare,GENccEnrichResult,ccOptions-method
    (ccCompare-methods), 6
  ccCompare,GOccEnrichResult,ccOptions-method
    (ccCompare-methods), 6
  ccCompare,KEGGccEnrichResult,ccOptions-method
    (ccCompare-methods), 6
  ccCompare-methods, 6
  ccCompareCollection, 7, 9, 10
  ccCompareCollection
    (ccCompareCollection-class), 7
  ccCompareCollection-class, 7
  ccCompareGeneric, 19, 22
  ccCompareGeneric
    (ccCompareGeneric-methods), 8
  ccCompareGeneric,GENccEnrichResult,ccOptions-method
    (GENccEnrichResult-class), 21
  ccCompareGeneric-methods, 8
  ccCompareResult, 5, 7, 8, 18
  ccCompareResult
    (ccCompareResult-class), 9