Package: categoryCompare2 (via r-universe)

August 25, 2024

Version 0.100.23

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/MoseleyBioinformaticsLab/categoryCompare2

BugReports https://github.com/MoseleyBioinformaticsLab/categoryCompare2/issues

License MIT + file LICENSE

Depends R (>= 3.5.0)

Suggests knitr, markdown, estrogen, org.Hs.eg.db, hgu95av2.db, limma, affy, genefilter, testthat (>= 3.0.0), visNetwork, Cairo, DiagrammeR, KEGGREST, docopt (>= 0.7.0), RCy3, hgu95av2cdf, rmarkdown

Imports Biobase, methods, AnnotationDbi, colorspace, graph, igraph, jsonlite, base64enc, dplyr, purrr, rlang, GO.db, stats

LazyLoad yes

Description Facilitates comparison of significant annotations (categories) generated on one or more feature lists. Interactive exploration is facilitated through the use of RCytoscape (heavily suggested).

SystemRequirements Cytoscape (>= 3.0) (if used for visualization of results, heavily suggested)

biocViews Annotation, GO, MultipleComparison, Pathways, GeneExpression

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

Repository https://moseleybioinformaticslab.r-universe.dev

RemoteUrl https://github.com/MoseleyBioinformaticsLab/categoryCompare2

RemoteRef v0.100.23

RemoteSha 380e8fc9fe78fe97597a87a9ce97b014f00ff1ab

Contents

`add_data_to_graph` *add table data to graph*

Description

given the annotation_graph and a data.frame, add all of the data in the data.frame to the graph so it is available elsewhere. Note that for NA integer and numerics, the value is modified to -100, and for infinite values, it is modified to 1e100.

Usage

```
add_data_to_graph(graph, data)
```

Arguments

`graph`	the graph to work on
`data`	the data to add to it

Value

graphNEL

add_tooltip *add tooltip*

Description

before passing to Cytoscape, add a tooltip attribute to the graph

Usage

```
add_tooltip(
  in_graph,
  node_data = c("name", "description"),
  description,
  separator = "\n"
)
```

Arguments

in_graph	the graph to work with
node_data	which pieces of node data to use
description	other descriptive text to use
separator	what separator to use for the tooltip

Value

the graph with a new nodeData member "tooltip"

annotation *annotation class*

Description

This class holds an annotation object that defines how annotations relate to features, as well as various pieces about each annotation

Does sensical checks when creating an annotation object.

Usage

```
annotation(
  annotation_features,
  annotation_type = NULL,
  description = character(0),
  links = character(0),
  feature_type = NULL
```

```
)

## S4 method for signature 'annotation'
show(object)

annotation(
  annotation_features,
  annotation_type = NULL,
  description = character(0),
  links = character(0),
  feature_type = NULL
)
```

Arguments

annotation_features	list of annotation to feature relationships
annotation_type	a simple one word description of the annotations
description	character vector providing descriptive text about the annotation
links	character vector defining html links for each annotation (may be empty)
feature_type	one word description of the feature type
object	the annotation object

Details

These objects may be created by hand, or may result from specific functions to create them. Most notably, this package provides functions for creating a Gene Ontology annotation.

See the annotation, each slot is a parameter.

Slots

annotation_features list of annotation to feature relationships

description character vector providing descriptive text about the annotation

counts numeric vector of how many features are in each annotation

links character vector defining html links for each annotation (may be empty)

annotation_type a one word short description of the "type" of annotation

feature_type a one word short description of the "type" of features

annotation_2_json *annotation to json*

Description

Given a ‘categoryCompare2‘ annotation object, generate a JSON representation that can be used with the command line executable

Usage

```
annotation_2_json(annotation_obj, json_file = NULL)
```

Arguments

annotation_obj	the annotation object
json_file	the file to save it to

Value

the json string (invisibly)

annotation_combinations *unique annotation combinations*

Description

determine the unique combinations of annotations that exist in the significant matrix of the cc_graph and assign each node in the graph to a group.

determine the unique combinations of annotations that exist in the significant matrix of the combined_statistics and assign each annotation to a group.

Usage

```
annotation_combinations(object)

## S4 method for signature 'cc_graph'
annotation_combinations(object)

## S4 method for signature 'significant_annotations'
annotation_combinations(object)
```

Arguments

object	the combined_statistics to work on

Value

node_assignment

node_assignment

`annotation_gene_table` *annotation to genes*

Description

Creates a tabular output of annotations to genes providing lookup of which genes are contributing to a particular annotation.

Usage

```
annotation_gene_table(
  combined_enrichment,
  annotations = NULL,
  use_db = NULL,
  input_type = "ENTREZID",
  gene_info = c("SYMBOL", "GENENAME")
)
```

Arguments

`combined_enrichment`	combined enrichment object
`annotations`	which annotations to grab features from
`use_db`	the annotation database
`input_type`	what type of gene id was it?
`gene_info`	what type of info to return for each gene

Value

data.frame

assign_colors *assign colors*

Description

given a node_assign, assign colors to either the independent groups of unique annotations, or to each of the experiments independently.

Usage

```
assign_colors(in_assign, type = "experiment")
```

Arguments

in_assign	the node_assign object generated from a cc_graph
type	either "group" or "experiment"

Value

node_assign with colors

assign_communities *assign communities*

Description

given a cc_graph, find communities of nodes based on their connectivity and weights.

Usage

```
assign_communities(in_graph)
```

Arguments

in_graph	the cc_graph object to use

Value

list

binomial_basic *do binomial test*

Description

does a binomial test

Usage

```
binomial_basic(
  positive_cases,
  total_cases,
  p_expected = 0.5,
  direction = "two.sided",
  conf_level = 0.95
)
```

Arguments

positive_cases	number of positive instances
total_cases	total number of cases observed
p_expected	what is the expected probability
direction	which direction is the test
conf_level	confidence level for the confidence interval

Value

list

binomial_features-class *binomial feature class*

Description

class to hold features undergoing binomail statistical testing

Slots

positivefc	the features with positive fold-changes
negativefc	the features with negative fold-changes
annotation	annotation object

binomial_feature_enrichment *do binomial testing*

Description

do binomial testing

Usage

```
binomial_feature_enrichment(
  binomial_features,
  p_expected = 0.5,
  direction = "two.sided",
  p_adjust = "BH",
  conf_level = 0.95,
  min_features = 1
)
```

Arguments

binomial_features	a binomial_features object
p_expected	the expected probability (default 0.5)
direction	which direction to do the enrichment (two.sided, less, greater)
p_adjust	how to correct the p-values (default is "BH")
conf_level	the confidence level for the confidence interval (default is 0.95)
min_features	a minimum number of features that are annotated to each annotation

Value

enriched_result

binomial_result-class *the binomial results class*

Description

the binomial results class

Slots

positivefc	the positive log-fold-changed genes, a vector of class ANY
negativefc	the negative log-fold-changed genes
annotation	list giving the annotation to feature relationship
statistics	a statistical_results object

categoryCompare2 *categoryCompare2: A package for comparing enrichment results from multiple experiments*

Description

The categoryCompare2 package provides functions for simple enrichment and and comparison of those enrichment results.

cc_graph *cc_graph*

Description

A cc_graph class is a graphNEL with the added slot of significant, a matrix of rows (nodes / annotations) and whether they were found to be significant in a given enrichment (columns). This matrix is used for classifying the annotations into different groups, and generating either pie-charts or coloring the nodes in a visualization.

constructs a *cc_graph* given a graphNEL and a *significant* matrix.

Usage

```
cc_graph(graph, significant)

## S4 method for signature 'cc_graph'
show(object)

cc_graph(graph, significant)
```

Arguments

graph	the graphNEL
significant	a matrix indicating which nodes are significant in which experiment
object	the cc_graph to show

Slots

significant numeric matrix of ones and zeros

combined_coefficient *combined coefficient*

Description

takes an average of the `overlap` and `jaccard` coefficients

Usage

```
combined_coefficient(n1, n2)
```

Arguments

`n1`	group 1
`n2`	group 2

Value

double

combined_enrichment-class *combined enrichments*

Description

The `combined_enrichment` class holds the results of combining several `enriched_result`s together, which includes the original `enriched_result`s, as well as the `cc_graph` and combined `annotation` objects.

Slots

`enriched` list of enriched objects

`annotation` `annotation` where the annotation_features have been combined across the `enriched_result`

`statistics` `combined_statistics` of both

combined_significant_calls *get significant annotations calls*

Description

In the case where we have a combined_enrichment and we want to get all of the significant annotations from each of them, and put them together so we can start doing real meta-analysis.

Usage

```
combined_significant_calls(in_results, queries)
```

Arguments

in_results	a combined_enrichment object
queries	a list of queries that can form a call object

Details

Note that this function returns the original combined_enrichment object with a modified combined_statistics slot where the significant annotations have been added in.

Value

combined_enrichment object

combined_statistics *combined statistics*

Description

holds the results of extracting a bunch of statistics from a combined_enrichment into one entity. This is useful because we want to enable multiple data representations and simple filtering on the actual data.frame of statistics, and this provides flexibility to enable that.

constructor function for the combined_statistics object, makes sure that empty things get initialized correctly

Usage

```
combined_statistics(
  statistic_data,
  which_enrichment,
  which_statistic,
  annotation_id,
  significant = NULL,
  measured = NULL,
  use_names = NULL
)

combined_statistics(
  statistic_data,
  which_enrichment,
  which_statistic,
  annotation_id,
  significant = NULL,
  measured = NULL,
  use_names = NULL
)
```

Arguments

`statistic_data`	the data.frame of statistics
`which_enrichment`	which enrichment gave the results
`which_statistic`	which statistics were calculated in each case
`annotation_id`	the annotations for which we are returning statistics
`significant`	the significant annotations
`measured`	the measured annotations
`use_names`	the order of naming

Value

combined_statistics

Slots

`statistic_data` a `data.frame` of all of the statistics from all of the enrichments

`significant` a `significant_annotations` object, that may be empty

`which_enrichment` a `vector` giving which enrichment each column of the statistics came from

`which_statistic` a `vector` providing which statistic each column contains

combine_annotations *combine annotations*

Description

Takes multiple annotation objects and combines them so that there is a consistent sole set for creating the cc_graph and providing other information about each annotation entry.

Usage

```
combine_annotations(annotation_list)

## S4 method for signature 'list'
combine_annotations(annotation_list)
```

Arguments

annotation_list
: one or more annotation

Value

annotation

---

combine_annotation_features *combine annotation-features*

Description

For the generation of a proper annotation-annotation relationship graph, we need to combine the annotation-feature relationships across multiple annotation objects

Usage

```
combine_annotation_features(annotation_features)
```

Arguments

annotation_features
: list of annotation_features to combine

Value

list of combined annotations

combine_enrichments *combine enrichments*

Description

This is one of the primary workhorse functions behind **categoryCompare2**. The primary function of `categoryCompare` is to enable *comparisons* of different enrichment analyses. To facilitate that, we must first **combine** one (really, we can do this with a single) or more `enriched_result`.

Usage

```
combine_enrichments(...)

## S4 method for signature 'enriched_result'
combine_enrichments(...)

## S4 method for signature 'list'
combine_enrichments(...)
```

Arguments

`...`	list of enriched_result

Value

`combined_enrichment`

combine_text *combine text*

Description

Given lists of named character objects, and a character vector of names to be in the final object, either get the character string from the list that has the names, or check that the character string is the same across all of the lists.

Usage

```
combine_text(list_characters, names_out, text_id)
```

Arguments

`list_characters`	list containing named character strings
`names_out`	the full list of names to use
`text_id`	what is the name for that thing being put out

Value

named character vector

`csv_annotation_table` *print table csv*

Description

print the annotation gene table to a CSV file

Usage

```
csv_annotation_table(annotation_gene_table, out_file = NULL)
```

Arguments

`annotation_gene_table`	list of tables
`out_file`	the file to write to

`enriched_result` *the enriched results class*

Description

given all the slots for an `enriched_result`, checks that all the data is self-consistent, and creates the `enriched_result` object.

Usage

```
enriched_result(features, universe, annotation, statistics)

enriched_result(features, universe, annotation, statistics)
```

Arguments

`features`	the features that were differentially expressed (see details)
`universe`	all of the features that were measured
`annotation`	an `annotation` object
`statistics`	a `statistical_results` object

Value

enriched_result

Slots

`features` the "features" of interest, a vector of class ANY

`universe` all of the "features" in the background

`annotation` list giving the annotation to feature relationship

`statistics` a `statistical_results` object

---

executable_path *executable path*

Description

Show the path to the executables, so the user can add them to whatever they want.

Usage

```
executable_path()
```

---

extract_enrich_stats *extract enrich stats*

Description

Extract statistical table from a single enrichment object.

Usage

```
extract_enrich_stats(enrichment_result)
```

Arguments

`enrichment_result`
: the enrichment result object

Value

data.frame

extract_statistics *get statistics*

Description

extract all statistics for a statistical_results object. These can then be combined into a data.frame that can be returned or used to annotate the graph of annotations.

Usage

```
extract_statistics(in_results)

## S4 method for signature 'statistical_results'
extract_statistics(in_results)
```

Arguments

in_results	the statistical_results object

Value

data.frame

extract_statistics,combined_enrichment-method *extract statistics*

Description

extract all statistics from a combined_enrichment object and create a combined_statistics where each statistic from the underlying statistical_results object in each of the enrichments is named according to which enrichment it was in and what statistic it was.

Usage

```
## S4 method for signature 'combined_enrichment'
extract_statistics(in_results)
```

Arguments

in_results	the combined_enrichment object

Value

combined_statistics

filter_annotation_graph

filter graph by significant entries

Description

If a graph has already been generated, it may be faster to filter a previously generated one than generate a new one from significant data.

Usage

```
filter_annotation_graph(in_graph, comb_enrich)
```

Arguments

in_graph	the cc_graph previously generated
comb_enrich	the combined_enrichment that you want to use to filter with

Value

cc_graph

generate_annotation_graph

generate annotation graph

Description

given a combined_enrichment, generate the annotation similarity graph

Usage

```
generate_annotation_graph(
  comb_enrichment,
  annotation_similarity = "combined",
  low_cut = 5,
  hi_cut = 500
)

## S4 method for signature 'combined_enrichment'
generate_annotation_graph(
  comb_enrichment,
  annotation_similarity = "combined",
  low_cut = 5,
  hi_cut = 500
)
```

Arguments

comb_enrichment	the combined_enrichment object
annotation_similarity	which similarity measure to use
low_cut	keep only those annotations in the graph with at least this many annotated features
hi_cut	keep only those annotations with less than this many annotated features

Value

cc_graph

---

generate_annotation_similarity_graph *annotation similarity graph*

Description

given an annotation-feature list, generate a similarity graph between all of the annotations

Usage

```
generate_annotation_similarity_graph(
  annotation_features,
  similarity_type = "combined"
)
```

Arguments

annotation_features	list where each entry is a set of features to that annotation
similarity_type	which type of overlap coefficient to report

Value

cc_graph

generate_colors *generate colors*

Description

given a bunch of items, generate a set of colors for either single node colorings or pie-chart annotations. Colors are generated using the *hcl* colorspace, and for `n_color >= 5`, the colors are re-ordered in an attempt to create the largest contrasts between colors, as they result from being picked on a circle in *hcl* space.

Usage

```
generate_colors(n_color)
```

Arguments

`n_color`	how many colors to generate

generate_legend *generate a legend*

Description

it often helps to have a legend displayed for reference.

Usage

```
generate_legend(
  in_assign,
  upper_names = TRUE,
  img = FALSE,
  width = 800,
  height = 400,
  pointsize = 70,
  ...
)
```

Arguments

`in_assign`	the assign object from `annotation_combinations`
`upper_names`	whether to make names uppercase for easier viewing
`img`	should a base64 encoded data uri be returned for embedding?
`width`	how wide should the image be if saving to an image
`height`	how high should it be
`pointsize`	the pointsize parameter for Cairo, determines textsize in the image
`...`	any other parameter to `pie`

generate_link *generate link text*

Description

given a named vector of links, generate an actual html link formatted for output in html documents

Usage

```
generate_link(links)
```

Arguments

links	the vector of links

Value

character

generate_piecharts *create piecharts for visualization*

Description

given a group matrix and the colors for each experiment, generate the pie graphs that will be used as glyphs in Cytoscape

Usage

```
generate_piecharts(grp_matrix, use_color)
```

Arguments

grp_matrix	the group matrix
use_color	the colors for each experiment

Details

this should *not be exported in the final version*

Value

list of png files that are pie graphs

generate_table *generate statistical table*

Description

given a combined_enrichment object, get out the data.frame either for investigation or to add data to the cc_graph.

Usage

```
generate_table(comb_enrichment, link_type = "explicit")

## S4 method for signature 'combined_enrichment'
generate_table(comb_enrichment, link_type = "explicit")
```

Arguments

comb_enrichment	the combined_enrichment object
link_type	should their be an "explicit" link (see details)

Details

the link_type controls whether to create an "explicit" link that is actually a column in the data.frame, or create an "implicit" html link that is part of the @name column in the returned data.frame. Useful if you are embedding the data.frame in an html report.

Value

data.frame

get_db_annotation *orgdb annotations*

Description

Generate an annotation object for genes based on an "org.*.db" object, and pulling information from it.

Usage

```
get_db_annotation(
  orgdb = "org.Hs.eg.db",
  features = NULL,
  feature_type = "ENTREZID",
  annotation_type = "GO"
)
```

Arguments

orgdb	the name of the org.*.db object
features	which features to get annotations for
feature_type	which type of IDs to map (see details)
annotation_type	the type of annotation to grab (see details)

Details

This function generates a categoryCompare2 annotation object from a Bioconductor "org.*.db" object. Even though different gene identifiers can be used, almost all of the mappings are via ENTREZID.

The set of feature or gene keys that can be used to create the annotations include:

- ENTREZID: ENTREZ gene ids
- ACCNUM: genbank accession numbers
- SYMBOL: gene symbols, eg ABCA1
- GENENAME: gene names, eg "ATP binding cassette subfamily A member 1"
- ENSEMBL: the ensembl gene ids (all start with ENSG...)
- ENSEMBLPROT: ensembl protein ids (ENSP...)
- ENSEMBLTRANS: ensemlb transcript ids (ENST...)
- REFSEQ: reference sequence IDs, NM, NP, NR, XP, etc
- UNIGENE: gene ids from UNIPROT eg Hs.88556
- UNIPROT: protein ids from UNIPROT eg P80404

The set of annotations that can be mapped to features include:

- GO: annotations from gene ontology
- PATH: KEGG Pathway identifiers (not updated since 2011!)
- CHRLOC: location on the chromosome
- OMIM: mendelian inheritance in man identifiers
- PMID: pubmed identifiers
- PROSITE
- PFAM: protein family identifiers
- IPI: protein-protein interactions

For GO annotations, it is also possible to pass GO to use all 3 sub-ontologies simultaneously, or any combination of BP, MF, and CC.

Value

annotation object

get_significant_annotations *get significant annotations*

Description

given a statistical_results object and some conditional expressions, return the significant annotations

In the case where we have a combined_enrichment and we want to get all of the significant annotations from each of them, and put them together so we can start doing real meta-analysis.

Usage

```
get_significant_annotations(in_results, ...)

## S4 method for signature 'statistical_results'
get_significant_annotations(in_results, ...)

## S4 method for signature 'combined_enrichment'
get_significant_annotations(in_results, ...)
```

Arguments

in_results	a combined_enrichment object
...	conditional expressions

Details

Note that this function returns the original combined_enrichment object with a modified combined_statistics slot where the significant annotations have been added in.

Value

vector of significant annotation_id's

combined_enrichment object

Examples

```
test_stat <- new("statistical_results",
                 annotation_id = c("a1", "a2", "a3"),
                 statistic_data = list(pvalues = c(a1 = 0.01, a2 = 0.5, a3 = 0.0001),
                   counts = c(a1 = 5, a2 = 10, a3 = 1),
                   odds = c(a1 = 20, a2 = 100, a3 = 0)))
get_significant_annotations(test_stat, pvalues < 0.05)
get_significant_annotations(test_stat, odds > 10)
get_significant_annotations(test_stat, pvalues < 0.05, counts >= 1)
```

get_significant_annotations_calls *get significant annotations calls*

Description

In the case where we have a statistical_results and we want to get all of the significant annotations from it

Usage

```
get_significant_annotations_calls(in_results, queries)
```

Arguments

in_results	a statistical_results object
queries	a list of queries that can form a call object

Value

vector of significant annotation_id's

gocats_to_annotation *gocats to annnotations*

Description

Transforms a gocats ancestors JSON list to a GO annotation object.

Usage

```
gocats_to_annotation(
  ancestors_file = "ancestors.json",
  namespace_file = "namespace.json",
  annotation_type = "gocatsGO",
  feature_type = "Uniprot",
  feature_translation = NULL
)
```

Arguments

ancestors_file	the ancestors.json file from gocats (required)
namespace_file	the namespace.json file from gocats (optional)
annotation_type	what annotations are we making? (gocatsGO by default)
feature_type	what type of features are we using (assume Uniprot)
feature_translation	a data.frame used to convert the feature IDs

Value

annotation object

---

graph_to_visnetwork *cc_graph to visnetwork*

Description

takes a cc_graph object and transforms it into something that can be visualized using visNetwork

Usage

```
graph_to_visnetwork(
  in_graph,
  in_assign,
  node_communities = NULL,
  use_nodes = NULL
)
```

Arguments

in_graph	the cc_graph object
in_assign	the colors generated by assign_colors
node_communities	the communities generated by label_communities
use_nodes	the list of nodes to actually use

Value

list

hypergeometric_basic *do hypergeometric test*

Description

does a hypergeometric enrichment test

Usage

```
hypergeometric_basic(
  num_white,
  num_black,
  num_drawn,
  num_white_drawn,
  direction = "over"
)
```

Arguments

num_white	number of white balls in urn
num_black	number of black balls in urn
num_drawn	number of balls taken from urn
num_white_drawn	number of white balls taken from urn
direction	which direction is the test

Value

list

hypergeometric_feature_enrichment *do hypergeometric enrichment*

Description

do hypergeometric enrichment

Usage

```
hypergeometric_feature_enrichment(
  hypergeom_features,
  direction = "over",
  p_adjust = "BH",
  min_features = 1
)
```

Arguments

hypergeom_features	a hypergeometric_features object
direction	which direction to do the enrichment (over or under)
p_adjust	how to correct the p-values (default is "BH")
min_features	how many features should be annotated before testing it?

Details

The `min_features` argument here applies to the minumum number of features an annotation has from the universe of features supplied, **not** the minumum number of features from the differential list. For more about the p-value adjustment, see `stats::p.adjust`

Value

enriched_result

---

hypergeom_features-class *hypergeom feature class*

Description

class to hold features undergoing hypergeometric enrichment

Slots

`significant` the significant features

`universe` all of the features measured

`annotation` annotation object

---

install_executables *install executables*

Description

move executables to user location, default is ~/bin and changes their permissions to make them executable.

Usage

```
install_executables(path = "~/bin")
```

Arguments

path	the path to put the executable scripts

Value

the listing of the files.

---

jaccard_coefficient *jaccard coefficient*

Description

calculates similarity of two groups of objects using "jaccard" coefficient, defined as:

Usage

```
jaccard_coefficient(n1, n2)
```

Arguments

n1	group 1
n2	group 2

Details

length(intersect(n1, n2)) / min(c(length(n1), length(n2)))

Value

double

---

json_2_annotation *json to annotation*

Description

Given a JSON based annotation object, read it in and create the ‘annotation‘ for actually doing enrichment.

Usage

```
json_2_annotation(json_file)
```

Arguments

json_file	the json annotation file

Value

annotation object

`json_annotation_reversal` *annotation reversal*

Description

Given a JSON file of features to annotations, reverse to turn it into annotations to features, and optionally add some meta-information about them.

Usage

```
json_annotation_reversal(
  json_file,
  out_file = "annotations.json",
  feature_type = NULL,
  annotation_type = NULL
)
```

Arguments

`json_file`	the json file to use
`out_file`	the json file to write out to
`feature_type`	the type of features
`annotation_type`	the type of annotations

Value

the json object, invisibly

`kable_annotation_table` *print table kable*

Description

print the annotation gene table in `knitr::kable` format

Usage

```
kable_annotation_table(annotation_gene_table, header_level = 3, cat = TRUE)
```

Arguments

annotation_gene_table	list of tables
header_level	what header level should the labels be done at?
cat	whether to write it directly, or just return the table for later

Value

character

---

label_communities *label communities*

Description

Determine the label of a community based on the most generic member of each community, which is defined as being the one with the most annotations.

Usage

```
label_communities(community_defs, annotation)
```

Arguments

community_defs	the communities from assign_communities
annotation	the annotation object used for enrichment

Value

list

---

multi_query_list *index a list*

Description

Provided a list, and a condition, returns the logical indices into the named part of the list provided. Uses subset like non-standard evaluation so that we can define appropriate expressions.

Usage

```
multi_query_list(list_to_query, ...)
```

Arguments

list_to_query	the list to run the query on
...	the expressions that do the queries

Value

logical "&" of all queries

---

node_assign-class *node_assign*

Description

The node_assign class holds the unique annotation combinations and the assignment of the nodes to those combinations for use in visualization.

Slots

groups the unique groups, as a logical matrix

assignments named character vector providing association with groups

description named character vector providing a description to group

colors named character vector of hex colors for groups or experiments

color_type whether doing group or experiment based colors

pie_locs if doing experiment colors, then pie graphs were generated here

---

overlap_coefficient *overlap coefficient*

Description

calculates the similarity using the "overlap" coefficient, which is

Usage

```
overlap_coefficient(n1, n2)
```

Arguments

n1	group 1 of objects
n2	group 2 of objects

Details

length(intersect(n1, n2)) / length(union(n1, n2))

Value

double

`remove_edges` *remove edges*

Description

given a RCy3 network connection, remove edges according to provided values.

Usage

```
remove_edges(edge_obj, cutoff, edge_attr = "weight", value_direction = "under")

## S4 method for signature 'character,numeric'
remove_edges(edge_obj, cutoff, edge_attr = "weight", value_direction = "under")

## S4 method for signature 'cc_graph,numeric'
remove_edges(edge_obj, cutoff, edge_attr = "weight", value_direction = "under")
```

Arguments

`edge_obj`	cc_graph
`cutoff`	the cutoff to use
`edge_attr`	which attribute to use
`value_direction`	remove edges with value under or over

Value

nothing

cc_graph

`show,binomial_result-method` *show binomial_result*

Description

show binomial_result

Usage

```
## S4 method for signature 'binomial_result'
show(object)
```

Arguments

object	the binomial_result object to show

---

show,combined_statistics-method
show combined_statistics

Description

show combined_statistics

Usage

```
## S4 method for signature 'combined_statistics'
show(object)
```

Arguments

object	combined_statistics

---

show,enriched_result-method
show enriched_result

Description

show enriched_result

Usage

```
## S4 method for signature 'enriched_result'
show(object)
```

Arguments

object	the enriched_result object to show

show,node_assign-method
show node_assign

Description

show node_assign

Usage

```
## S4 method for signature 'node_assign'
show(object)
```

Arguments

object	the node_assign to see

show,significant_annotations-method
show signficant_annotations

Description

show signficant_annotations

Usage

```
## S4 method for signature 'significant_annotations'
show(object)
```

Arguments

object	the significant annotations object to show

significant_annotations *significant annotations*

Description

The significant_annotations class holds which annotations from which enrichment were both **measured** and **significant**. Each of these slots is a *logical matrix* with rows named by *annotation_id* and columns named by the names of the enriched_result that was combined.

Makes a new significant_annotation while checking that everything is valid.

Usage

```
significant_annotations(significant, measured, sig_calls = NULL)

significant_annotations(significant, measured, sig_calls = NULL)
```

Arguments

significant	logical matrix of annotations (rows) and experiments (columns)
measured	logical matrix of annotations (rows) and experiments (columns)
sig_calls	character vector of deparsed calls that resulted in signficant and measured

Slots

significant logical matrix

measured logical matrix

sig_calls character representations of calls used to filter the data

statistical_results-class *statistical results class*

Description

This class holds the part of an enrichment that is the statistical results. It has two pieces, a list of statistics that is a named list with the actual numerical results of applying the statistics. The other piece is the annotation_id vector defining which entry in each vector of the statistics is.

Slots

statistic_data list of numerical statistics

annotation_id vector of ids

method how the statistics were calculated

table_from_graph *table from graph*

Description

Creates a table from the annotation graph, and if provided, adds the community information to the table.

Usage

```
table_from_graph(in_graph, in_assign = NULL, community_info = NULL)
```

Arguments

in_graph	the cc_graph object
in_assign	the node_assign object
community_info	the community_info object

Value

data.frame

vis_in_cytoscape *visualize in cytoscape*

Description

given a graph, and the node assignments, visualize the graph in cytoscape for manipulation

Usage

```
vis_in_cytoscape(in_graph, in_assign, description = "cc2 enrichment")
```

Arguments

in_graph	the cc_graph to visualize
in_assign	the node_assign generated
description	something descriptive about the vis (useful when lots of different visualizations)

Value

something

vis_visnetwork *vis in visNetwork*

Description

Visualize a cc_graph in visNetwork, with selection for communities if that exists.

Usage

```
vis_visnetwork(in_graph_info)
```

Arguments

in_graph_info	the graph structure from graph_to_visnetwork

Index