

Package ‘GOfuncR’

February 16, 2018

Type Package

Title Gene ontology enrichment using FUNC

Version 0.99.11

Date 2017-11-22

Author Steffi Grote

Maintainer Steffi Grote <steffi_grote@eva.mpg.de>

Description GOfuncR performs a gene ontology enrichment analysis based on the ontology enrichment software FUNC. GO-annotations are obtained from OrganismDb or OrgDb packages ('Homo.sapiens' by default). It provides the standard candidate vs. background enrichment analysis using the hypergeometric test, as well as three additional tests: (i) the Wilcoxon rank-sum test that is used when genes are ranked, (ii) a binomial test that is used when genes are associated with two counts and (iii) a Chi-square or Fisher's exact test that is used in cases when genes are associated with four counts. To correct for multiple testing and interdependency of the tests, family-wise error rates are computed based on random permutations of the gene-associated variables. GOfuncR also provides tools for exploring the ontology graph and the annotations, and options to take gene-length or spatial clustering of genes into account.

License GPL (>= 2)

Imports Rcpp (>= 0.11.5), mapplots (>= 1.5), gtools (>= 3.5.0), GenomicRanges (>= 1.28.4), AnnotationDbi, utils, grDevices, graphics, stats,

Depends R (>= 3.4), vioplot (>= 0.2),

LinkingTo Rcpp

Suggests Homo.sapiens, BiocStyle, knitr, testthat

VignetteBuilder knitr

biocViews GeneSetEnrichment, GO

NeedsCompilation yes

R topics documented:

get_anno_categories	2
get_anno_genes	3

get_child_nodes	4
get_ids	5
get_names	6
get_parent_nodes	7
go_enrich	8
plot_anno_scores	11
Index	13

get_anno_categories	<i>Get all associated ontology categories for the input genes</i>
---------------------	---

Description

Returns all associated GO-categories given a vector of gene-symbols, e.g. `c('SPAG5', 'BTC')`. Besides the default 'Homo.sapiens', also other OrganismDb or OrgDb packages from Bioconductor, like 'Mus.musculus' (mouse) or 'org.Pt.eg.db' (chimp), can be used.

Usage

```
get_anno_categories(genes, database = 'Homo.sapiens')
```

Arguments

genes	a character() vector of gene-symbols, e.g. <code>c('SPAG5', 'BTC')</code> .
database	optional character() defining an OrganismDb or OrgDb annotation package from Bioconductor, like 'Mus.musculus' (mouse) or 'org.Pt.eg.db' (chimp).

Value

a data.frame() with two columns: gene (character()) and GO-ID (character()).

Note

This gives only direct annotations of genes to GO-categories. By definition genes are also indirectly annotated to all parent nodes of those categories. Use [get_parent_nodes](#) to get the higher level categories of the directly annotated GO-categories.

Also note that GO-categories which are not represented or obsolete in the internal GO-graph of GOfuncR are removed to be consistent with the annotations used in [go_enrich](#).

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. Nature Genetics 25, 25-29.

See Also

[get_anno_genes](#)
[get_parent_nodes](#)
[get_names](#)

Examples

```
## get the GO-annotations for two random genes
anno1 = get_anno_categories(c('BTC', 'SPAG5'))
head(anno1)
# add the names and domains of the GO-categories
named_anno1 = merge(anno1, get_names(anno1$go_id))
head(named_anno1)
```

get_anno_genes

Get genes that are annotated to GO-categories

Description

Given a vector of GO-IDs, e.g. `c('GO:0072025','GO:0072221')` the function returns all genes that are annotated to these GO-categories. This includes genes that are annotated to any of the child nodes of a GO-category. Besides the default 'Homo.sapiens', also other `OrganismDb` or `OrgDb` packages from Bioconductor, like 'Mus.musculus' (mouse) or 'org.Pt.eg.db' (chimp), can be used. It is also possible to only consider a user-defined set of genes.

Usage

```
get_anno_genes(go_ids, database = 'Homo.sapiens', genes = NULL)
```

Arguments

go_ids	character() vector of GO-IDs, e.g. <code>c('GO:0051082', 'GO:0042254')</code> .
database	optional character() defining an <code>OrganismDb</code> or <code>OrgDb</code> annotation package from Bioconductor, like 'Mus.musculus' (mouse) or 'org.Pt.eg.db' (chimp).
genes	optional character() vector of gene-symbols. If defined, only annotations of those genes are returned.

Value

A `data.frame()` with two columns: GO-IDs (`character()`) and the annotated genes (`character()`). The output is ordered by GO-ID and gene-symbol.

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics* 25, 25-29.

See Also

[get_anno_categories](#)
[get_ids](#)
[get_names](#)
[get_child_nodes](#)
[get_parent_nodes](#)

Examples

```
## find all genes that are annotated to GO:000109
## ("nucleotide-excision repair complex")
get_anno_genes(go_ids='GO:000109')

## find out wich genes from a set of genes
## are annotated to some GO-categories
genes = c('AGTR1', 'AN01', 'CALB1', 'GYG1', 'PAX2')
gos = c('GO:0001558', 'GO:0005536', 'GO:0072205', 'GO:0006821')
anno_genes = get_anno_genes(go_ids=gos, genes=genes)
# add the names and domains of the GO-categories
cbind(anno_genes ,get_names(anno_genes$go_id)[,2:3])

## find all annotations to GO-categories containing 'serotonin receptor'
sero_ids = get_ids('serotonin receptor')
sero_anno = get_anno_genes(go_ids=sero_ids$go_id)
# merge with names of GO-categories
head(merge(sero_ids, sero_anno))
```

get_child_nodes

Get all child nodes of gene ontology categories

Description

Returns all child nodes (sub-categories) of GO-categories given their GO-IDs, e.g. c('GO:0042254', 'GO:0000109'). The output also states the shortest distance to the child node. Note that a GO-ID itself is also considered as child with distance 0.

Usage

```
get_child_nodes(go_ids)
```

Arguments

go_ids a character() vector of GO-IDs, e.g. c('GO:0051082', 'GO:0042254').

Value

a data.frame() with four columns: parent GO-ID (character()), child GO-ID (character()), child GO-name (character()) and distance (numeric()).

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. Nature Genetics 25, 25-29.

See Also

[get_names](#)
[get_parent_nodes](#)

Examples

```
## get the child nodes (sub-categories) of two random GO-IDs
child_nodes = get_child_nodes(c('GO:0090070', 'GO:0000112'))
child_nodes
```

get_ids

Get the ID of a GO-category given its name

Description

Returns GO-categories given (part of) their name. Matching is not case-sensitive.

Usage

```
get_ids(go_name)
```

Arguments

go_name character(). (partial) name of a GO-category

Value

a data.frame() with three columns: the full names (character()) of the GO-categories that contain go_name; together with the GO-domain ('cellular_component', 'biological_process' or 'molecular_function') and the GO-category IDs (character()).

Note

This is just a `grep(..., ignore.case=TRUE)` on the node names of the ontology. More sophisticated searches, e.g. with regular expressions, could be performed on the table returned by `get_ids('')` which lists all non-obsolete GO-categories.

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. Nature Genetics 25, 25-29.

See Also

[get_names](#)
[get_parent_nodes](#)
[get_child_nodes](#)

Examples

```
## get GO-IDs of categories that contain 'gabaergic' in their names
get_ids('gabaergic')

## get GO-IDs of categories that contain 'blood-brain barrier' in their names
get_ids('blood-brain barrier')

## get all valid GO categories
all_nodes = get_ids('')
head(all_nodes)
```

`get_names`*Get the full names of gene ontology categories given the IDs*

Description

Returns the full names and the domains of GO-categories given the GO-IDs, e.g. 'GO:0042254'.

Usage

```
get_names(go_ids)
```

Arguments

`go_ids` a character() vector of GO-IDs, e.g. `c('GO:0051082', 'GO:0042254')`.

Value

a data.frame() with three columns: `go_id` (character()), `go_name` (character()) and `root_node` (domain, character()).

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. Nature Genetics 25, 25-29.

See Also

[get_ids](#)
[get_child_nodes](#)
[get_parent_nodes](#)

Examples

```
## get the full names of three random GO-IDs  
get_names(c('GO:0051082', 'GO:0042254', 'GO:0000109'))
```

get_parent_nodes	<i>Get all parent nodes of gene ontology categories</i>
------------------	---

Description

Returns all parent nodes (higher level categories) of GO-categories given their GO-IDs, e.g. c('GO:0042254', 'GO:0000109'). The output also states the shortest distance to the parent node. Note that a GO-ID itself is also considered as parent with distance 0.

Usage

```
get_parent_nodes(go_ids)
```

Arguments

go_ids a character() vector of GO-IDs, e.g. c('GO:0051082', 'GO:0042254').

Value

a data.frame() with four columns: child GO-ID (character()), parent GO-ID(character()), parent GO-name(character()) and distance (numeric()).

Author(s)

Steffi Grote

References

[1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. Nature Genetics 25, 25-29.

See Also

[get_names](#)
[get_child_nodes](#)

Examples

```
## get the parent nodes (higher level GO-categories) of two random GO-IDs
parents = get_parent_nodes(c('GO:0051082', 'GO:0042254'))
parents
```

go_enrich

Test gene sets for enrichment in GO-categories

Description

Tests all GO-categories from the three domains 'biological process', 'cellular component' and 'molecular function' for enrichment of user defined gene sets [1]. Human GO-annotations are used by default (from OrganismDb package 'Homo.sapiens'), but also other OrganismDb or OrgDb packages can be used.

The statistical analysis is based on the ontology enrichment software FUNC [2]. go_enrich offers four different statistical tests: (1) the hypergeometric test for a candidate and a background gene set; (2) the wilcoxon rank-sum test for genes that are ranked by scores (e.g. p-value for differential expression); (3) the binomial test for genes that have two associated counts (e.g. amino-acid changes on the human and the chimp lineage); and (4) a 2x2-contingency table test for genes that have four associated counts (e.g. for a McDonald-Kreitman test).

To account for multiple testing family-wise error rates are computed using randomsets. Besides naming candidate genes explicitly, for the hypergeometric test it is also possible to provide entire genomic regions as input. The enrichment analysis is then performed for all genes located in or overlapping these regions and the multiple testing correction accounts for the spatial clustering of genes.

Usage

```
go_enrich(genes, test = 'hyper', n_randsets = 1000, organismDb = 'Homo.sapiens', gene_len = FALSE,
          regions = FALSE, circ_chrom = FALSE, silent = FALSE, domains = NULL, orgDb = NULL, txDb = NULL)
```

Arguments

genes	a data.frame() with gene-symbols (character()) in the first column and test-dependent additional numeric() columns: If test='hyper' (default) a second column with 1 for candidate genes and 0 for background genes. If no background genes are defined, all remaining genes from the internal dataset are used as background. If test='wilcoxon' a second column with the score that is associated with each gene. If test='binomial' two additional columns with two gene-associated integers. If test='contingency' four additional columns with four gene-associated integers. For test='hyper' with regions=TRUE the first column describes chromosomal regions ('chr:start-stop', e.g. '9:0-39200000').
test	character(). 'hyper' (default) for the hypergeometric test, 'wilcoxon' for the Wilcoxon rank test, 'binomial' for the binomial test and 'contingency' for the 2x2-contingency table test (fisher's exact test or chi-square).
n_randsets	integer defining the number of random sets created to compute the FWER.

organismDb	character(). Annotation package for GO-annotations and gene coordinates. Besides the default 'Homo.sapiens' also 'Mus.musculus' and 'Rattus.norvegicus' are currently available on Bioconductor.
gene_len	logical. If test='hyper' the probability of a background gene to be chosen as a candidate gene in a random set is dependent on the gene length.
regions	logical. When TRUE chromosomal regions are analysed, and genes are automatically extracted from the regions defined in genes[,1] as e.g. '9:0-39200000'. Note that this option requires the input of background regions.
circ_chrom	logical. When genes defines chromosomal regions, circ_chrom=TRUE uses background regions from the same chromosome and allows randomly chosen blocks to overlap multiple background regions. Only if test='hyper'.
silent	logical. If TRUE all output to the screen except for warnings and errors is suppressed.
domains	optional character() vector containing one or more of the three GO-domains 'cellular_component', 'biological_process' and 'molecular_function'. If defined, the analysis will be reduced to those domains which saves time.
orgDb	optional character() naming an OrgDb annotation package from Bioconductor. If orgDb is set, organismDb is not used. OrgDb annotations are available for a wider range of species, e.g. 'org.Pt.eg.db' for chimp and 'org.Gg.eg.db' for chicken. Note that options gene_len and regions also need a txDb for the gene coordinates (which are integrated in OrganismDb).
txDb	optional character() naming an TxDb annotation package from Bioconductor (e.g. 'TxDb.Ptroglydytes.UCSC.panTro4.refGene') for the gene coordinates. Only needed when orgDb is specified, and options gene_len or regions are set.

Details

For details please refer to the package's vignette.

Value

A list with components

results	a data.frame() with the FWERs from the enrichment analyses per ontology category, ordered by 'FWER_overrep', 'raw_p_overrep', '-FWER_underrep', '-raw_p_underrep', 'ontology' and 'node_id', or the corresponding columns if another test then the hypegeometric test is used. This table contains the combined results for all three ontology domains. Note that GO-categories without any annotations of candidate or background genes are not listed.
genes	the input genes data.frame(), excluding those genes for which no GO-annotations are available and which therefore were not included in the enrichment analysis. If gene_len=TRUE, also genes without gene coordinates are excluded.
databases	a data.frame() with the annotation packages used.

Author(s)

Steffi Grote

References

- [1] Ashburner, M. et al. (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics* 25: 25-29. doi: [10.1038/75556](https://doi.org/10.1038/75556)
- [2] Pruefer, K. et al. (2007). FUNC: A package for detecting significant associations between gene sets and ontological. *BMC Bioinformatics* 8: 41. doi: [10.1186/14712105841](https://doi.org/10.1186/14712105841)

See Also

[get_parent_nodes](#)
[get_child_nodes](#)
[get_anno_categories](#)
[get_anno_genes](#)
[plot_anno_scores](#)
[get_names](#)
[get_ids](#)

Examples

```
##### Note that argument 'n_randsets' is reduced
##### to lower computational time in the following examples.
##### Using the default value is recommended.

##### Perform a GO-enrichment analysis for some human genes
##### with a defined background set
# create input dataframe that defines the candidate and background genes
candi_gene_ids = c('NCAPG', 'APOL4', 'NGFR', 'NXPH4', 'C21orf59', 'CACNG2',
  'AGTR1', 'ANO1', 'BTBD3', 'MTUS1', 'CALB1', 'GYG1', 'PAX2')
bg_gene_ids = c('FGR', 'NPHP1', 'DRD2', 'ABCC10', 'PTBP2', 'JPH4', 'SMARCC2',
  'FN1', 'NODAL', 'CYP1A2', 'ACSS1', 'CDHR1', 'SLC25A36', 'LEPR', 'PRPS2',
  'TNFAIP3', 'NKX3-1', 'LPAR2', 'PGAM2')
is_candidate = c(rep(1,length(candi_gene_ids)), rep(0,length(bg_gene_ids)))
genes = data.frame(gene_ids=c(candi_gene_ids, bg_gene_ids), is_candidate)
genes

# run enrichment analysis
go_res = go_enrich(genes, n_randset=100)

# go_enrich returns a list with 3 elements:
# 1) results from the analysis
# (ordered by FWER for overrepresentation of candidate genes)
head(go_res[[1]])
# see the top GOs from every GO-domain
by(go_res[[1]], go_res[[1]][,'ontology'], head)
# 2) all valid input genes
go_res[[2]]
# 3) annotation databases used
go_res[[3]]

##### see the package's vignette for more examples
```

plot_anno_scores	<i>Plot distribution of scores of genes annotated to GO-categories</i>
------------------	--

Description

Uses the result of a GO-enrichment analysis performed with `go_enrich` and a vector of GO-IDs and plots for each of these GO-IDs the scores of the annotated genes. This refers to the scores that were provided as user-input in the `go_enrich` analysis.

`plot_anno_scores` works with all four tests implemented in `go_enrich` (hypergeometric, wilcoxon rank-sum, binomial and 2x2 contingency table test), with test-specific output (see details).

Usage

```
plot_anno_scores(res, go_ids)
```

Arguments

<code>res</code>	an object returned from <code>go_enrich</code> (list() of 3 elements).
<code>go_ids</code>	character() vector of GO-IDs, e.g. <code>c('GO:0005737','GO:0071495')</code> . This specifies the GO-categories that are plotted.

Details

The plot depends on the statistical test that was specified in the `go_enrich` call.

For the hypergeometric test pie charts show the amounts of candidate and background genes that are annotated to the GO-categories and the root nodes (candidate genes in the colour of the corresponding root node). The top panel shows the odds-ratio and 95%-CI from fisher's exact test (two-sided) comparing the GO-categories with their root nodes. Note that `go_enrich` reports the the hypergeometric test for over- and under-representation of candidate genes wich correspond to the one-sided fisher's exact tests.

For the wilcoxon rank-sum test violin plots show the distribution of the scores of genes that are annotated to each GO-category and the root nodes. Horizontal lines in the left panel indicate the median of the scores that are annotated to the root nodes. The wilcoxon rank-sum test reported in the `go_enrich` result compares the scores annotated to a GO-category with the scores annotated to the corresponding root node.

For the binomial test pie charts show the amounts of A and B counts associated with each GO-category and root node, (A in the colour of the corresponding root node). The top-panel shows point estimates and the 95%-CI of $p(A)$ in the nodes, as well as horizontal lines that correspond to $p(A)$ in the root nodes. The p-value in the returned object is based on the null hypothesis that $p(A)$ in a node equals $p(A)$ in the corresponding root node. Note that `go_enrich` reports that value for one-sided binomial tests.

For the 2x2 contingency table test pie charts show the proportions of A and B, as well as C and D counts associated with a GO-category. Root nodes are not shown, because this test is independent of the root category. The top panel shows the odds ratio and 95%-CI from Fisher's exact test (two-sided) comparing A/B and C/D inside one node. Note that in `go_enrich`, if all four values are ≥ 10 , a chi-square test is performed instead of fisher's exact test.

Value

For the hypergeometric, binomial and 2x2 contingency table test, a `data.frame()` with the statistics that are used in the plots.

For the wilcoxon rank-sum test no statistical results are plotted, just the distribution of annotated scores. The returned element in this case is a `list()` with three data frames: annotations of genes to the GO-categories, annotations of genes to the root nodes and a table which contains for every GO-ID the corresponding root node.

Author(s)

Steffi Grote

See Also

[go_enrich](#)
[get_anno_genes](#)
[get_names](#)
[vioplot](#)

Examples

```
#### see the package vignette for more examples

#### Note that argument 'n_randsets' is reduced
#### to lower computational time in the example.

## Assign two random counts and to some random genes to create example input
set.seed(123)
high_A_genes = c('G6PD', 'GCK', 'GYS1', 'HK2', 'PYGL', 'SLC2A8',
  'UGP2', 'ZWINT', 'ENGASE')
low_A_genes = c('CACNG2', 'AGTR1', 'ANO1', 'BTBD3', 'MTUS1', 'CALB1',
  'GYG1', 'PAX2')
A_counts = c(sample(15:25, length(high_A_genes)),
  sample(5:15, length(low_A_genes)))
B_counts = c(sample(5:15, length(high_A_genes)),
  sample(15:25, length(low_A_genes)))
genes = data.frame(gene=c(high_A_genes, low_A_genes), A_counts, B_counts)

## perform enrichment analysis to find GO-categories with high fraction of A
go_binom = go_enrich(genes, test='binomial', n_randsets=20)

## plot sums of A and B counts associated with the top GO-categories
top_gos = head(go_binom[[1]]$node_id)
stats = plot_anno_scores(go_binom, go_ids=top_gos)

## look at the results of binomial test used for plotting
## (this is two-sided, go_enrich reports one-sided tests)
head(stats)
```

Index

*Topic **htest**

go_enrich, 8

get_anno_categories, 2, 4, 10

get_anno_genes, 3, 3, 10, 12

get_child_nodes, 4, 4, 6, 7, 10

get_ids, 4, 5, 7, 10

get_names, 3–6, 6, 7, 10, 12

get_parent_nodes, 2–7, 7, 10

go_enrich, 2, 8, 11, 12

plot_anno_scores, 10, 11

vioplot, 12