Package ‘ViSEAGO’

May 4, 2024

Version 1.18.0

Title ViSEAGO: a Bioconductor package for clustering biological functions using Gene Ontology and semantic similarity

Author Aurelien Brionne [aut, cre], Amelie Juanchich [aut], Christelle hennequet-antier [aut]

Maintainer Aurelien Brionne <aurelien.brionne@inrae.fr>

Imports data.table, AnnotationDbi, AnnotationForge, biomaRt, dendextend, DiagrammeR, DT, dynamicTreeCut, fgsea, GOSemSim, ggplot2, GO.db, grDevices, heatmaply, htmlwidgets, igraph, methods, plotly, processx, topGO, RColorBrewer, R.utils, scales, stats, UpSetR, utils

Depends R (>= 3.6)

Suggests htmltools, org.Mm.eg.db, limma, Rgraphviz, BiocStyle, knitr, rmarkdown, corrplot, remotes, BiocManager

Description The main objective of ViSEAGO package is to carry out a data mining of biological functions and establish links between genes involved in the study. We developed ViSEAGO in R to facilitate functional Gene Ontology (GO) analysis of complex experimental design with multiple comparisons of interest. It allows to study large-scale datasets together and visualize GO profiles to capture biological knowledge. The acronym stands for three major concepts of the analysis: Visualization, Semantic similarity and Enrichment Analysis of Gene Ontology. It provides access to the last current GO annotations, which are retrieved from one of NCBI Entrez-Gene, Ensembl or Uniprot databases for several species. Using available R packages and novel developments, ViSEAGO extends classical functional GO analysis to focus on functional coherence by aggregating closely related biological themes while studying multiple datasets at once. It provides both a synthetic and detailed view using interactive functionalities respecting the GO graph structure and ensuring functional coherence supplied by semantic similarity. ViSEAGO has been successfully applied on several datasets from different species with a variety of biological questions. Results can be easily shared between bioinformaticians and biologists, enhancing reporting capabilities while maintaining reproducibility.

VignetteBuilder knitr

License GPL-3 bioconductor.org
Contents

annotate .......................................................... 3
available_organisms ........................................... 6
Bioconductor2GO ................................................ 7
build_GO_SS ..................................................... 8
clusters_cor .................................................... 9
compare_clusters ............................................... 12
compute_SS_distances ......................................... 15
create_topGOdata ............................................... 17
Custom2GO ....................................................... 19
enrich_GO_terms-class ........................................ 20
Ensembl2GO ..................................................... 21
EntrezGene2GO .................................................. 22
EntrezGene_orthologs .......................................... 23
fgsea-class ...................................................... 24
gene2GO-class .................................................. 24
genomic_ressource-class ..................................... 25
GOclusters_heatmap ............................................ 25
GOcount .......................................................... 28
GOterms_heatmap ................................................ 29
**annotate**

Retrieve GO annotations for a specie from genomic resource database.

**Description**

This method retrieves and stores GO annotations for the organism of interest from one of genomic resource database (Bioconductor, EntrezGene, Ensembl, Uniprot).

**Usage**

```r
annotate(id, object, ortholog = FALSE)
```

## S4 method for signature 'character,genomic_ressource'
annotate(id, object, ortholog = FALSE)

**Arguments**

- **id**
  - identifiant corresponding to the organism of interest. This id name is referenced in the first column of the database used (see `available_organisms`).

- **object**
  - a required `genomic_ressource-class` object created by `Bioconductor2GO`, `EntrezGene2GO`, `Ensembl2GO`, or `Uniprot2GO` methods.

- **ortholog**
  - logical (default to FALSE). Only available for vertebrates organisms and for object created by `EntrezGene2GO` method (see Details).
Details

This method uses a **genomic_ressource-class** object to retrieve GO annotations for the organism of interest. The stored annotations are structured in 3 slots corresponding to the 3 GO categories: MF (Molecular Function), BP (Biological Process), and CC (Cellular Component). Each slot contains GO terms with associated evidence code.

The **genomic_ressource-class** object is created by one of the four available methods: Bioconductor2GO, EntrezGene2GO, Ensembl2GO, or Uniprot2GO.

In the case of vertebrates, setting ortholog argument to TRUE is required if you need to add GO terms with experimental evidence codes from orthologs genes when using EntrezGene2GO method. To display organisms supported by NCBI EntrezGene orthologs pipeline, set the arguments id=NULL and ortholog=TRUE. This approach is highly similar to the strategy developed by Uniprot-GOA consortium for the Electronic Annotation Method using Ensembl Compara.

Value

annotate produces an object of **gene2GO-class** required by build_GO_SS method.

References


See Also

Other genomic_ressource: Bioconductor2GO(), Custom2GO(), Ensembl2GO(), EntrezGene2GO(), Uniprot2GO(), available_organisms(), genomic_ressource-class.taxonomy()

Other GO_terms: GOcount(), GOterms_heatmap(), create_topGOdata(), gene2GO-class, merge_enrich_terms(), runfgsea()

Examples

```r
## Not run:
## load Mus musculus (mouse) GO annotations

# from Bioconductor
Bioconductor<-ViSEAGO::Bioconductor2GO()
```
myGENE2GO<-ViSEAGO::annotate(
    id="org.Mm.eg.db",
    object=Bioconductor
)

# from EntrezGene
EntrezGene2GO<-ViSEAGO::EntrezGene2GO()
myGENE2GO<-ViSEAGO::annotate(
    id="10090",
    object=EntrezGene
)

# from EntrezGene
Ensembl2GO<-ViSEAGO::Ensembl2GO()
myGENE2GO<-ViSEAGO::annotate(
    id="mmusculus_gene_ensembl",
    object=Ensembl
)

# from Uniprot
Uniprot2GO<-ViSEAGO::Uniprot2GO()
myGENE2GO<-ViSEAGO::annotate(
    id="mouse",
    object=Uniprot
)

## from Custom GO annotation file
Custom2GO<-ViSEAGO::Custom2GO(system.file("extdata/customfile.txt",package = "ViSEAGO"))
myGENE2GO<-ViSEAGO::annotate(
    id="myspecies1",
    object=Custom
)

## specific options for EntrezGene database

# Chicken GO annotations without adding orthologs
EntrezGene2GO<-ViSEAGO::EntrezGene2GO()
myGENE2GO<-ViSEAGO::annotate(
    id="9031",
    object=EntrezGene
)

# Chicken GO annotation with the add of orthologs GO annotations
EntrezGene2GO<-ViSEAGO::EntrezGene2GO()
myGENE2GO<-ViSEAGO::annotate(
    id="9031",
    object=EntrezGene,
    ortholog=TRUE
)

# display organisms supported by NCBI EntrezGene orthologs pipeline
EntrezGene2GO<-ViSEAGO::EntrezGene2GO()
ViSEAGO::annotate(
available_organisms

Display available organisms from a specified database.

Description

Display an interactive table with available organisms from a genomic resource database (Bioconductor, EntrezGene, Ensembl, Uniprot).

Usage

available_organisms(object)

## S4 method for signature 'genomic_ressource'
available_organisms(object)

Arguments

object a genomic_ressource-class object created by Bioconductor2GO, EntrezGene2GO, Ensembl2GO, or Uniprot2GO methods.

Details

an interactive datatable.

Value

javascript datatable

References


See Also

Other genomic_ressource: Bioconductor2GO(), Custom2GO(), Ensembl2GO(), EntrezGene2GO(), Uniprot2GO(), annotate(), genomic_ressource-class, taxonomy()

Other visualization: GOclusters_heatmap(), G0count(), G0terms_heatmap(), Upset(), overLapper(), show_heatmap(), show_table()
Examples

# display Bioconductor table
Bioconductor<-ViSEAGO::Bioconductor2GO()
ViSEAGO::available_organisms(Bioconductor)
## Not run:

# display EntrezGene table
EntrezGene<-ViSEAGO::EntrezGene2GO()
ViSEAGO::available_organisms(EntrezGene)

# display Ensembl table
Ensembl<-ViSEAGO::Ensembl2GO()
ViSEAGO::available_organisms(Ensembl)

# display Uniprot table
Uniprot<-ViSEAGO::Uniprot2GO()
ViSEAGO::available_organisms(Uniprot)

## End(Not run)

Bioconductor2GO Check available organisms databases at Bioconductor.

Description

Retrieve the Bioconductor OrgDb available organisms databases packages.

Usage

Bioconductor2GO()

Details

This function gives genome wide annotation for available organisms databases packages from Bioconductor OrgDb. It uses loadAnnDbPkgIndex from AnnotationForge package.

Value

a genomic_ressource-class object required by annotate method.

References


See Also

Other genomic_ressource: Custom2GO(), Ensembl2GO(), EntrezGene2GO(), Uniprot2GO(), annotate(), available_organisms(), genomic_ressource-class, taxonomy()
Examples

```r
# Check Bioconductor OrgDb available organisms
Bioconductor<-ViSEAGO::BioconductorGO()
```

Description

Compute the Information content (IC) on the given ontology, and create a **GO_SS-class** object required by `compute_SS_distances` method to compute GO semantic similarity between enriched GO terms or groups of terms.

Usage

```r
build_GO_SS(gene2GO, enrich_GO_terms)
```  

## S4 method for signature 'gene2GO,enrich_GO_terms'

```r
build_GO_SS(gene2GO, enrich_GO_terms)
```

Arguments

- `gene2GO` a **gene2GO-class** object from `annotate` method.
- `enrich_GO_terms` a **enrich_GO_terms-class** from `merge_enrich_terms` method.

Details

This method use `annotate` and `merge_enrich_terms` output objects (see Arguments), and compute the Information content (IC) using the internal code of `godata` function from **GOSemSim** package.

Value

a **GO_SS-class** object required by `compute_SS_distances`.

References


clusters_cor

See Also

Other GO_semantic_similarity: GO_SS-class, compute_SS_distances()

Examples

```r
## Not run:
# initialise object for compute GO Semantic Similarity
myGOs<-ViSEAGO::build_GO_SS(
   myGENE2GO,
   BP_sResults
)

## End(Not run)
# load data example
utils::data(
   myGOs,
   package="ViSEAGO"
)
```

---

clusters_cor  

*Compute distance matrix between dendrograms partitions.*

Description

Build a distance or correlation matrix between partitions from dendrograms.

Usage

```r
clusters_cor(clusters, method = "adjusted.rand")
```

S4 method for signature 'list,character'

```r
clusters_cor(clusters, method = "adjusted.rand")
```

Arguments

- **clusters**: a list of `GO_clusters-class` objects, from `GOterms_heatmap` or `GOclusters_heatmap`, named as character.
- **method**: available methods ("vi", "nmi", "split.join", "rand", or "adjusted.rand") from `igraph` package `compare` function.

Value

A distance or a correlation matrix.

References

See Also

Other GO_clusters: `GO_clusters-class`, `GOclusters_heatmap()`, `compare_clusters()`, `show_heatmap()`, `show_table()`

Examples

```r
# load example object
data(
  myGOs,
  package="ViSEAGO"
)

## Not run:
# compute Semantic Similarity (SS)
myGOs<-ViSEAGO::compute_SS_distances(
  myGOs,
  distance=c("Resnik","Lin","Rel","Jiang","Wang")
)

# Resnik distance GO terms heatmap
Resnik_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Resnik",
      aggreg.method="ward.D2"
    ),
    cut=list(
      dynamic=list(
        deepSplit=2,
        minClusterSize =2
      )
    ),
  ),
  samples.tree=NULL
)

# Lin distance GO terms heatmap
Lin_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Lin",
      aggreg.method="ward.D2"
    ),
    cut=list(
      dynamic=list(
        deepSplit=2,
      )
    ),
  ),
  samples.tree=NULL
)
```

clusters_cor

    minClusterSize = 2

),
samples.tree = NULL
)

# Resnik distance GO terms heatmap
Rel_clusters_wardD2 <- ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  GO.tree = list(
    tree = list(
      distance = "Rel",
      aggreg.method = "ward.D2"
    ),
    cut = list(
      dynamic = list(
        deepSplit = 2,
        minClusterSize = 2
      )
    ),
    samples.tree = NULL
  )
)

# Resnik distance GO terms heatmap
Jiang_clusters_wardD2 <- ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  GO.tree = list(
    tree = list(
      distance = "Jiang",
      aggreg.method = "ward.D2"
    ),
    cut = list(
      dynamic = list(
        deepSplit = 2,
        minClusterSize = 2
      )
    ),
    samples.tree = NULL
  )
)

# Resnik distance GO terms heatmap
Wang_clusters_wardD2 <- ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  GO.tree = list(

```r
# Build an interactive heatmap of the common GO terms frequency between several partitions.

## Description
Build an interactive heatmap of the common GO terms frequency between several partitions.

## Usage

```r
compare_clusters(clusters)
```

## Arguments

```r
clusters  a list of named GO_clusters-class objects, from GOterms_heatmap or GOclusters_heatmap methods.
```
Details

Build an interactive heatmap of common GO terms frequency between partitions from several `GO_clusters-class` objects.

Value

an interactive javascript heatmap.

References


See Also

Other GO_clusters: `GO_clusters-class, GOclusters_heatmap(), clusters_cor(), show_heatmap(), show_table()`

Examples

```r
# load example object
data(
  myGOs,
  package="ViSEAGO"
)

## Not run:
# compute Semantic Similarity (SS)
myGOs<-ViSEAGO::compute_SS_distances(
  myGOs,
  distance=c("Resnik","Lin","Rel","Jiang","Wang")
)

# Resnik distance GO terms heatmap
Resnik_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Resnik",
      aggreg.method="ward.D2"
    ),
    cut=list(
      dynamic=list(
        deepSplit=2,
        minClusterSize =2
      )
    )
  ),
)
```
samples.tree=NULL)

# Lin distance GO terms heatmap
Lin_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
    myGOs,
    showIC=TRUE,
    showGOlabels=TRUE,
    GO.tree=list(
        tree=list(
            distance="Lin",
            aggreg.method="ward.D2"
        ),
        cut=list(
            dynamic=list(
                deepSplit=2,
                minClusterSize=2
            )
        ),
        samples.tree=NULL
    )
)

# Resnik distance GO terms heatmap
Rel_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
    myGOs,
    showIC=TRUE,
    showGOlabels=TRUE,
    GO.tree=list(
        tree=list(
            distance="Rel",
            aggreg.method="ward.D2"
        ),
        cut=list(
            dynamic=list(
                deepSplit=2,
                minClusterSize=2
            )
        ),
        samples.tree=NULL
    )
)

# Resnik distance GO terms heatmap
Jiang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
    myGOs,
    showIC=TRUE,
    showGOlabels=TRUE,
    GO.tree=list(
        tree=list(
            distance="Jiang",
            aggreg.method="ward.D2"
        ),
compute_SS_distances

Cut

\[ \text{cut} = \text{list(} \]
\[ \quad \text{dynamic} = \text{list(} \]
\[ \quad \quad \text{deepSplit} = 2, \]
\[ \quad \quad \text{minClusterSize} = 2 \]
\[ \quad \) \]
\[ \), \]
\[ \text{samples.tree} = \text{NULL} \]
\)

# Resnik distance GO terms heatmap
Wang_clusters_wardD2 <- ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  GO.tree = list( 
    tree = list( 
      distance = "Wang",
      aggreg.method = "ward.D2"
    ), 
    cut = list( 
      dynamic = list( 
        deepSplit = 2,
        minClusterSize = 2
      )
    ), 
    samples.tree = NULL
  )
)

## End(Not run)

# clusters to compare
clusters <- list( 
  Resnik = "Resnik_clusters_wardD2",
  Lin = "Lin_clusters_wardD2",
  Rel = "Rel_clusters_wardD2",
  Jiang = "Jiang_clusters_wardD2",
  Wang = "Wang_clusters_wardD2"
)

## Not run:
# clusters content comparisons
clusters_comp <- ViSEAGO::compare_clusters(clusters)

## End(Not run)
compute_SS_distances

Description

This method computes distance between GO terms or GO clusters based on semantic similarity.

Usage

compute_SS_distances(object, distance)

## S4 method for signature 'ANY,character'
compute_SS_distances(object, distance)

Arguments

object a GO_SS-class, or GO_clusters-class objects created by build_GO_SS or GOterms_heatmap methods, respectively.
distance The available methods for calculating GO terms Semantic Similarity (SS) are "Resnik", "Rel", "Lin", and "Jiang" which are based on Information Content (IC), and "Wang" which is based on graph topology. The available methods for calculating clusters of GO terms SS are "max", "avg", "rcmax", and "BMA".

Details

This method computes semantic similarity distances between all GO terms provided by GO_SS-class object.
This method also computes semantic similarity distances between all GO clusters provided by GO_clusters-class object.

Semantic Similarity computations are based on mgoSim method from the GoSemSim package.

Value

a GO_SS-class, or a GO_clusters-class object (same class as input object).

References

Guangchuang Yu, Fei Li, Yide Qin, Xiaochen Bo, Yibo Wu and Shengqi Wang. GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. Bioinformatics 2010 26(7):976-978

See Also

Other GO_semantic_similarity: GO_SS-class, build_GO_SS()
create_topGOdata

Create topGOdata object for enrichment test with topGO package.

Examples

# load data example
data(
  myGOs,
  package="ViSEAGO"
)

## Not run:
# compute GO terms Semantic Similarity distances
myGOs<-ViSEAGO::compute_SS_distances(
  myGOs,
  distance=c("Resnik","Lin","Rel","Jiang","Wang")
)

# GOtermsHeatmap with default parameters
Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Wang",
      aggreg.method="ward.D2",
      rotate=NULL
    ),
    cut=list(
      dynamic=list(
        pamStage=TRUE,
        pamRespectsDendro=TRUE,
        deepSplit=2,
        minClusterSize=2
      )
    ),
    samples.tree=NULL
  ),
  samples.tree=NULL
)

# compute clusters of GO terms Semantic Similarity distances
Wang_clusters_wardD2<-ViSEAGO::compute_SS_distances(
  Wang_clusters_wardD2,
  distance=c("max","avg","rcmax","BMA")
)

## End(Not run)
create_topGOdata

Description

This method create a topGOdata-class object required by topGO package in order to perform GO enrichment test.

Usage

create_topGOdata(geneSel, allGenes, geneList = NULL, gene2GO, ont, nodeSize)

## S4 method for signature 'ANY,ANY,ANY,gene2GO,character,numeric'
create_topGOdata(geneSel, allGenes, geneList = NULL, gene2GO, ont, nodeSize)

Arguments

geneSel genes of interest.
allGenes customized background genes.
geneList logical factor (1: genes of interest, 0: genes background, and gene identifiants in names) (default value to NULL).
gene2GO a gene2GO-class object created by annotate method.
ont the ontology used is "MF" (Molecular Function), "BP" (Biological Process), or "CC" (Cellular Component).
nodeSize the minimum number of genes for each GO term.

Details

This method is a convenient wrapper building a topGOdata-class object using a given ontology category (ont argument) in order to perform GO enrichment test. The complete GO annotation is required (gene2GO argument) and also the list of genes of interest (geneSel argument) against the corresponding background (allGenes argument) separately, or grouped together in a factor (geneList argument).

Value

a topGOdata-class object required by runTest from topGO package.

References


See Also

Other GO_terms: GOcount(), GOterms_heatmap(), annotate(), gene2GO-class, merge_enrich_terms(), runfgsea()
Examples

```r
# load genes identifiers (GeneID,ENS...) background (Expressed genes)  
background<-scan(  
  system.file(
    "extdata/data/input",
    "background_L.txt",
    package = "ViSEAGO"
  ),  
  quiet=TRUE,
  what=""
)

# load Differentialy Expressed (DE) gene identifiers from files  
pregnantvslactateDE<-scan(  
  system.file(
    "extdata/data/input",
    "pregnantvslactateDE.txt",
    package = "ViSEAGO"
  ),  
  quiet=TRUE,
  what=""
)

## Not run:  
# create topGOdata for BP for each list of DE genes  
BP_L_pregnantvslactateDE<-ViSEAGO::create_topGOdata(  
  geneSel=pregnantvslactateDE,  
  allGenes=background,  
  gene2GO=myGENE2GO,  
  ont="BP",  
  nodeSize=5
)

## End(Not run)
```

---

**Custom2GO**

*Store organisms GO annotations from custom database file.*

**Description**

Store the available species and current GO annotations from a custom table file.

**Usage**

```r
Custom2GO(file)
```

**Arguments**

- `file` : custom GO annotation file
enrich_GO_terms-class

Details

This function loads a custom GO annotation database table that must contain columns:

- **taxid**: custom taxonomic identifiers
- **gene_id**: custom gene identifiers
- **gene_symbol**: custom gene symbols
- **GOID**: Known GO identifiers (see `select(GO.db, columns=columns(GO.db), keys=keys(GO.db))`)
- **evidence**: Known GO evidence codes

Value

A genomic_ressource-class object required by `annotate`.

References


See Also

Other genomic_ressource: `Bioconductor2GO()`, `Ensembl2GO()`, `EntrezGene2GO()`, `Uniprot2GO()`, `annotate()`, `available_organisms()`, `genomic_ressource-class`, `taxonomy()`

Examples

```r
# Download custom GO annotations
Custom<-ViSEAGO::Custom2GO(
  system.file(
    "extdata/customfile.txt",
    package = "ViSEAGO"
  )
)
```

enrich_GO_terms-class  
enrich_GO_terms class object definition.

Description

This class is invoked by `merge_enrich_terms` method in order to store the merged data.table and associated metadata.

Slots

- **same_genes_background**: logical. object(s) to combine (see examples in `merge_enrich_terms`).
- **ont**: ontology used "MF", "BP", or "CC".
- **method**: enrichment test used "topGO", or "fgsea".
- **summary**: a list with topGO or fgsea object(s) summary informations.
- **data**: a merged data.table of enriched GO terms (p<0.01) in at least once with GO descriptions and statistical values.
Ensembl2GO

See Also

Other enrich_GO_terms: `Upset()`, `overLapper()`, `show_heatmap()`, `show_table()`

Ensembl2GO

Check available organisms datasets at Ensembl.

Description

List Ensembl referenced organisms datasets from the current (NULL) or archive (number in character) annotation version.

Usage

Ensembl2GO(biomart = "genes", GRCh = NULL, version = NULL)

Arguments

biomart the biomart name available with `biomaRt` package `listEnsembl` ("genes", the default) or `listEnsemblGenomes" ("protists_mart", "fungi_mart", "plants_mart").
GRCh GRCh version to connect to if not the current GRCh38, currently this can only be 37
version the annotation version to use (eg. NULL for the default current version, or a version number in character)

Details

This function gives referenced organisms genomes at Ensembl. It uses the `useEnsembl` and `listDatasets` from `biomaRt` package.

Value

a `genomic_ressource-class` object required by `annotate`.

References


See Also

Other genomic_ressource: `Bioconductor2GO()`, `Custom2GO()`, `EntrezGene2GO()`, `Uniprot2GO()`, `annotate()`, `available_organisms()`, `genomic_ressource-class`, `taxonomy()`
## Examples

```r
# Not run:
biomaRt::listEnsembl()

# List Ensembl available organisms
Ensembl<-ViSEAGO::Ensembl2GO(
  biomart="genes",
  GRCh = NULL,
  version=NULL
)
```

## Description

Store available organisms GO annotations at EntrezGene.

### Usage

```r
EntrezGene2GO()
```

### Value

A `genomic_ressource-class` object required by `annotate`.

### References


EntrezGene_orthologs

Download EntrezGene orthologs groups.

Description

Download EntrezGene orthologs groups.

Usage

EntrezGene_orthologs()

Details

Internal function used by annotate in order to download orthologs groups from NCBI Annotation pipeline stored in the gene_group.gz file.

Value

a data.table.

Examples

## Not run:
# Organism taxid, Scientific name and common name
ViSEAGO::EntrezGene_orthologs()

## End(Not run)
fgsea-class  

**Description**

This class is invoked by `runfgsea` method in order to store results.

**Slots**

description a character string with database source, date of stamp, and target species GO annotation.
method fgsea method used.
params a list containing used input parameters for perform `fgseaSimple` or `fgseaMultilevel`.
input a list containing input values.
data a list containing data.table fgsea procedure output.

gene2GO-class  

**Description**

This class is invoked by `annotate` method in order to store GO annotations for each category (MF, BP, CC).

**Slots**

db database source in character.
stamp date of stamp in character.
organism target species GO annotation in character.
MF a list containing GO terms for Molecular Function (MF) category for each gene element.
BP a list containing GO terms for Biological Process (BP) category for each gene element.
CC a list containing GO terms for Cellular Component (CC) category for each gene element.

**See Also**

Other GO_terms: `GOcount()`, `GOTerms_heatmap()`, `annotate()`, `create_topGOdata()`, `merge_enrich_terms()`, `runfgsea()`
**genomic_ressource-class**

*genomic_ressource class object definition.*

**Description**

This class stores the annotations and associated metadata obtained by *Bioconductor2GO*, *EntrezGene2GO*, *Ensembl2GO*, or *Uniprot2GO*.

**Slots**

- `db` name of database used (Bioconductor, EntrezGene, Ensembl, or Uniprot).
- `stamp` date of stamp (for Bioconductor, EntrezGene, and Uniprot), or annotation version for Ensembl database.
- `data` GO annotations from *EntrezGene2GO* method.
- `organisms` informations about species/datasets availables.
- `mart` Ensembl mart from *Ensembl2GO* method.

**See Also**

Other genomic_ressource: *Bioconductor2GO()*,*Custom2GO()*,*Ensembl2GO()*,*EntrezGene2GO()*,*Uniprot2GO()*,*annotate()*,*available_organisms()*,*taxonomy()*

---

**GOclusters_heatmap** *Build a clustering heatmap on GO groups.*

**Description**

This method computes a clustering heatmap based on GO groups semantic similarity.

**Usage**

```
GOclusters_heatmap(
  object,
  tree = list(distance = "BMA", aggreg.method = "ward.D2", rotate = NULL)
)
```

```r
## S4 method for signature 'GO_clusters,list'
GOclusters_heatmap(
  object,
  tree = list(distance = "BMA", aggreg.method = "ward.D2", rotate = NULL)
)
```
Arguments

object a \texttt{GO\_clusters-class} object from \texttt{compute\_SS\_distances}.

tree a named list with:

\textbf{distance} ("BMA" by default) distance computed from the semantic similarity for GO groups which could be "max", "avg", "rcmax", or "BMA".

\textbf{aggreg.method} ("ward.D2" by default) aggregation method criteria from \texttt{hclust} (\texttt{ward.D}, \texttt{ward.D2}, "single", "complete", "average", "mcquitty", "median", or "centroid") to build a dendrogram.

\textbf{rotate} sort the branches of the tree based on a vector - either of labels order or the labels in their new order

Details

This method computes a clustering heatmap based on GO groups semantic similarity (computed with \texttt{compute\_SS\_distances}).

The heatmap color intensity corresponds to the number of GO terms in each GO group.

The GO group description is defined as the first common GO ancestor with the cluster identifiant in brackets.

The dendrogram branches are colored according to GO terms clusters.

Value

a \texttt{GO\_clusters-class} object.

References


See Also

Other GO\_clusters: \texttt{GO\_clusters-class,clusters\_cor()}, \texttt{compare\_clusters()}, \texttt{show\_heatmap()}, \texttt{show\_table()}

Other semantic\_similiarity: \texttt{GOterms\_heatmap()}

Other visualization: \texttt{GOcount()}, \texttt{GOterms\_heatmap()}, \texttt{Upset()}, \texttt{available\_organisms()}, \texttt{overLapper()}, \texttt{show\_heatmap()}, \texttt{show\_table()}
Examples

```r
# load data example
utils::data(
  myGOs,
  package="ViSEAGO"
)

## Not run:
# compute GO terms Semantic Similarity distances
myGOs<-ViSEAGO::compute_SS_distances(
  myGOs,
  distance="Wang"
)

# GOtermsHeatmap with default parameters
Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Wang",
      aggreg.method="ward.D2",
      rotate=NULL
    ),
    cut=list(
      dynamic=list(
        pamStage=TRUE,
        pamRespectsDendro=TRUE,
        deepSplit=2,
        minClusterSize =2
      )
    ),
    samples.tree=NULL
  )
)

# compute clusters of GO terms Semantic Similarity distances
Wang_clusters_wardD2<-ViSEAGO::compute_SS_distances(
  Wang_clusters_wardD2,
  distance="BMA"
)

# GOclusters heatmap
Wang_clusters_wardD2<-ViSEAGO::GOclusters_heatmap(
  Wang_clusters_wardD2,
  tree=list(
    distance="BMA",
    aggreg.method="ward.D2",
    rotate=NULL
  )
)
```
GOcount

Barplot for the count of GO terms.

Description
This method displays in barplot the count of GO terms splitted in two categories (significant or not) for each result of GO enrichment tests.

Usage
GOcount(object, file = NULL)

Arguments
- object: an enrich_GO_terms-class object from merge_enrich_terms method.
- file: the name of the output file (default to NULL for interactive screen display).

Details
This method displays an interactive barplot, using plotly package, from a merge_enrich_terms output object.
A static image (in png) could be printed by setting file argument.

Value
a barplot.

References

See Also
Other GO_terms: GOterms_heatmap(), annotate(), create_topG0data(), gene2GO-class, merge_enrich_terms(), runfgsea()
Other visualization: GOclusters_heatmap(), GOterms_heatmap(), Upset(), available_organisms(), overLapper(), show_heatmap(), show_table()
Examples

# load object
utils::data(
  myGOs,
  package="ViSEAGO"
)

# barplot for the count of GO terms
ViSEAGO::GOcount( myGOs)

GOterms_heatmap Build a clustering heatmap on GO terms.

Description

This method computes a clustering heatmap based on GO terms semantic similarity.

Usage

GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  heatmap_colors = c("#ffffff", "#99000D"),
  GO.tree = list(tree = list(distance = "Wang", aggreg.method = "ward.D2", rotate = NULL), cut = list(dynamic = list(pamStage = TRUE, pamRespectsDendro = TRUE, deepSplit = 2, minClusterSize = 2))),
  samples.tree = NULL
)

## S4 method for signature 'GO_SS'
GOterms_heatmap(
  myGOs,
  showIC = TRUE,
  showGOlabels = TRUE,
  heatmap_colors = c("#ffffff", "#99000D"),
  GO.tree = list(tree = list(distance = "Wang", aggreg.method = "ward.D2", rotate = NULL), cut = list(dynamic = list(pamStage = TRUE, pamRespectsDendro = TRUE, deepSplit = 2, minClusterSize = 2))),
  samples.tree = NULL
)

Arguments

myGOs a GO_SS-class object from compute_SS_distances.
showIC logical (default to TRUE) to display the GO terms Information Content (IC) side bar.
showGOlabels logical (default to TRUE) to display the GO terms ticks on y axis.

heatmap_colors pvalues color range with white to Sangria collors by default (c("#ffffff","#99000D")).

GO.tree a named list of parameters to build and cut the GO terms dendrogram.

tree (a named list with:) distance ("Wang" by default) distance computed from the semantic similarity which could be IC-based ("Resnik", "Rel", "Lin", or "Jiang") or graph-based ("Wang").

aggreg.method ("ward.D2" by default) aggregation method criteria from hclust ("ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", or "centroid") to build a dendrogram.

rotate sort the branches of the tree based on a vector - eithor of labels order or the labels in their new order

cut (a named list with:) static (default to NULL) a numeric value that is the height (between 0 and 1), or the number of clusters (value > 1) to cut the dendrogram.

dynamic (a named list which only contains cutreeDynamic options values below) pamStage (default to TRUE) second (PAM-like) stage will be performed.

pamRespectsDendro (default to TRUE) PAM stage will respect the dendrogram in the sense that objects and small clusters will only be assigned to clusters that belong to the same branch that the objects or small clusters being assigned belong to.

deepSplit (default to 2) provides a rough control over sensitivity for cluster splitting (range 0 to 4). The higher the value (or if TRUE), the more and smaller clusters will be produced.

minClusterSize (default to 2) minimum cluster size.

samples.tree a named list of parameters to build and cut the samples dendrogram (default to NULL).

tree (a named list with:) distance ("pearson" by default) distance computed that could be correlation ("abs.pearson","pearson", "kendall", or "spearman"), or dist method (euclidean", "maximum", "manhattan", "canberra", "binary", or "minkowski").

aggreg.method ("average" by default) same options than for GO.tree argument

cut same options than for GO.tree argument.

Details

This method computes a clustering heatmap based on GO terms semantic similarity (computed with compute_SS_distances).
The dendrogram produced could be cutted in static or dynamic mode.

1. build dendrograms on GO terms and optionally on samples.
2. cut in static or dynamic mode and color the dendrogram branches.
3. build an interactive clustering heatmap based on heatmaply.
**Value**

a `GO_clusters-class` object.

**References**


**See Also**

Other GO terms: `GOcount()`, `annotate()`, `create_topGOdata()`, `gene2GO-class`, `merge_enrich_terms()`, `runfgsea()`

Other semantic similarity: `GOclusters_heatmap()`

Other visualization: `GOclusters_heatmap()`, `GOcount()`, `Upset()`, `available_organisms()`, `overLapper()`, `show_heatmap()`, `show_table()`

**Examples**

```r
# load data example
utils::data(
    myGOs,
    package="ViSEAGO"
)

## Not run:
# compute GO terms Semantic Similarity distances
myGOs<-ViSEAGO::compute_SS_distances(
    myGOs,
    distance="Wang"
)

# GOtermsHeatmap with default parameters
Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
```

myGOs,
showIC=TRUE,
showGOlabels=TRUE,
GO.tree=list(
  tree=list(
    distance="Wang",
    aggreg.method="ward.D2",
    rotate=NULL
  ),
  cut=list(
    dynamic=list(
      pamStage=TRUE,
      pamRespectsDendro=TRUE,
      deepSplit=2,
      minClusterSize =2
    )
  ),
  samples.tree=NULL
)

## End(Not run)

---

**GO_clusters-class**  
*GO_clusters class object*

---

**Description**

This class is invoked by `GOterms_heatmap` and `GOclusters_heatmap` methods to store all results produced.

**Slots**

- **ont**  ontology used "MF", "BP", or "CC".
- **enrich_GOs**  `enrich_GO_terms-class` object.
- **IC**  Information Content (IC).
- **terms_dist**  distance between GO terms based on semantic similarity.
- **clusters_dist**  distance between GO groups based on semantic similarity.
- **hcl_params**  Hierarchical clustering parameters used.
- **dendrograms**  GO terms and samples dendrograms.
- **samples_gp**  samples groups.
- **heatmap**  GO terms and GO groups heatmaps.

**See Also**

Other GO_clusters: `GOclusters_heatmap()`, `clusters_cor()`, `compare_clusters()`, `show_heatmap()`, `show_table()`
**GO_SS-class**

**GO_SS class object definition.**

**Description**

This class is invoked by `build.GO_SS` method in order to store `enrich.GO_terms-class` object, Information Content (IC), and GO terms or groups distances objects based on semantic similarity.

**Slots**

- `ont` ontology used "MF", "BP", or "CC".
- `enrich_GOs` `merge_enrich_terms` output object (`enrich.GO_terms-class` object).
- `IC` Information Content (IC)
- `terms_dist` list of GO terms or groups distances objects based on semantic similarity.

**See Also**

Other GO_semantic_similarity: `build.GO_SS()`, `compute_SS_distances()`

---

**MDSplot**

*Multi Dimensional Scale (MDS) plot*

**Description**

Generate a Multi Dimensional Scale (MDS) plot from distance objects.

**Usage**

```r
MDSplot(object, type = "GOterms", file = NULL)
```

```r
## S4 method for signature 'ANY'
MDSplot(object, type = "GOterms", file = NULL)
```

**Arguments**

- `object` a `GO_SS-class` or `GO_clusters-class` objects from distances computed with `compute_SS_distances`.
- `type` could be "GOterms" to display GOterms MDSplot, or "GOclusters" to display GOclusters MDSplot.
- `file` static image output file name (default to NULL).

**Details**

This method build and display the javascript MDSplot (if file=NULL) from `GO_SS-class` or `GO_clusters-class` objects.

A static png image could be printed by setting file argument.
Value

a MDS plot.

Examples

# load data example
utils::data(
  myGOs,
  package="ViSEAGO"
)

## Not run:
# compute GO terms Semantic Similarity distances
myGOs<-ViSEAGO::compute_SS_distances(
  myGOs,
  distance="Wang"
)

# build MDS plot for a GO_SS-class distance object
ViSEAGO::MDSplot(myGOs,"GOterms")

# GOtermsHeatmap with default parameters
Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Wang",
      aggreg.method="ward.D2",
      rotate=NULL
    ),
    cut=list(
      dynamic=list(
        pamStage=TRUE,
        pamRespectsDendro=TRUE,
        deepSplit=2,
        minClusterSize =2
      )
    )
  ),
  samples.tree=NULL
)

# build MDS plot for a GO_clusters-class distance object, highlighting GO terms clusters.
ViSEAGO::MDSplot(
  Wang_clusters_wardD2,
  "GOterms"
)

# compute clusters of GO terms Semantic Similarity distances
Wang_clusters_wardD2<-ViSEAGO::compute_SS_distances(
  Wang_clusters_wardD2,
merge_enrich_terms

Merge enriched GO terms.

Description

Combine results from GO enrichment tests (obtained with topGO package) or from fgsea (obtained with runfgsea method), for a given ontology (MF, BP, or CC).

Usage

merge_enrich_terms(Input, cutoff = 0.01, envir = .GlobalEnv)

## S4 method for signature 'list'
merge_enrich_terms(Input, cutoff = 0.01, envir = .GlobalEnv)

Arguments

Input

A list containing named elements. Each element must contain the name of:

- **topGO**: topGOdata-class object created by create_topGOdata method and the associated topGOresult-class object.
- **fgsea**: fgsea-class object created by runfgsea method.

cutoff

default pvalue cutoff (default to 0.01). Several cutoff can be used in the same order as list elements.

envir

Objects environment (default to .GlobalEnv).
merge_enrich_terms

Details

This method extracts for each result of GO enrichment test: informations about GO term (identifier, name, and description), gene frequency (number of significant genes / Annotated genes), pvalue, -log10(pvalue), significant genes identifiers (GeneID, or Ensembl ID, or uniprot accession), and gene symbols. At the last, this method builds a merged data.table of enriched GO terms at least once and provides all mentioned columns.

Value

an enrich_GO_terms-class object.

References


See Also

Other GO_terms: GOcount(), GOterms_heatmap(), annotate(), create_topGOdata(), gene2GO-class, runfgea()

Examples

## topGO terms enrichment

# load genes identifiers (GeneID,ENS...) universe/background (Expressed genes)
background_L<-scan(
  system.file(
    "extdata/data/input",
    "background_L.txt",
    package = "ViSEAGO"
  ),
  quiet=TRUE,
  what=""
)

# load Differentially Expressed (DE) gene identifiers from files
PregnantvslactateDE<-scan(
  system.file(
    "extdata/data/input",
    "pregnantvslactateDE.txt",
    package = "ViSEAGO"
  ),
  quiet=TRUE,
  what=""
)

VirginvslactateDE<-scan(
  system.file(
## Not run:
# connect to Bioconductor
Bioconductor<-ViSEAGO::Bioconductor2GO()

# load GO annotations from Bioconductor
myGENE2GO<-ViSEAGO::annotate(
  "org.Mm.eg.db",
  Bioconductor
)

# create topGOdata for BP for each list of DE genes
BP_Pregnantvslactate<-ViSEAGO::create_topGOdata(
  geneSel=PregnantvslactateDE,
  allGenes=background_L,
  gene2GO=myGENE2GO,
  ont="BP",
  nodeSize=5
)

BP_Virginvslactate<-ViSEAGO::create_topGOdata(
  geneSel=VirginvslactateDE,
  allGenes=background_L,
  gene2GO=myGENE2GO,
  ont="BP",
  nodeSize=5
)

BP_Virginvspregnant<-ViSEAGO::create_topGOdata(
  geneSel=VirginvspregnantDE,
  allGenes=background_L,
  gene2GO=myGENE2GO,
  ont="BP",
  nodeSize=5
)
# perform TopGO tests
elim_BP_Pregnantvslactate<-topGO::runTest(  
  BP_L_pregnantvslactate,  
  algorithm ="elim",  
  statistic = "fisher"
)

elim_BP_Virginvslactate<-topGO::runTest(  
  BP_L_virginvslactate,  
  algorithm ="elim",  
  statistic = "fisher"
)

elim_BP_Virginvspregnant<-topGO::runTest(  
  BP_L_virginvspregnant,  
  algorithm ="elim",  
  statistic = "fisher"
)

# merge topGO results
BP_sResults<-ViSEAGO::merge_enrich_terms(  
  Input=list(  
    Pregnantvslactate=c("BP_Pregnantvslactate","elim_BP_Pregnantvslactate"),  
    Virginvslactate=c("BP_Virginvslactate","elim_BP_Virginvslactate"),  
    Virginvspregnant=c("BP_Virginvspregnant","elim_BP_Virginvspregnant")  
  )
)

## End(Not run)

## fgsea analysis

# load gene identifiers and padj test results from Differential Analysis complete tables
Pregnantvslactate<-data.table::fread(  
  system.file(  
    "extdata/data/input",
    "pregnantvslactate.complete.txt",
    package = "ViSEAGO"
  ),
  select = c("Id","padj")
)

Virginvslactate<-data.table::fread(  
  system.file(  
    "extdata/data/input",
    "virginvslactate.complete.txt",
    package = "ViSEAGO"
  ),
  select = c("Id","padj")
)

Virginvspregnant<-data.table::fread(  
  system.file(
merge_enrich_terms

"extdata/data/input",
"virginvspregnant.complete.txt",
package = "ViSEAGO"
),
  select = c("Id","padj")
)

# rank Id based on statistical value (padj)
PregnantvsLactate<-data.table::setorder(PregnantvsLactate,padj)
VirginvsLactate<-data.table::setorder(VirginvsLactate,padj)
VirginvsPregnant<-data.table::setorder(VirginvsPregnant,padj)

## Not run:
# connect to Bioconductor
Bioconductor<-ViSEAGO::Bioconductor2GO()

# load GO annotations from Bioconductor
myGENE2GO<-ViSEAGO::annotate("org.Mm.eg.db",
  Bioconductor)

# perform fgseaMultilevel tests
BP_PregnantvsLactate<-runfgsea(
  geneSel=PregnantvsLactate,
  gene2GO=myGENE2GO,
  ont="BP",
  params = list(
    scoreType = "pos",
    minSize=5
  )
)

BP_VirginvsLactate<-runfgsea(
  geneSel=VirginvsLactate,
  gene2GO=myGENE2GO,
  ont="BP",
  params = list(
    scoreType = "pos",
    minSize=5
  )
)

BP_VirginvsPregnant<-runfgsea(
  geneSel=VirginvsPregnant,
  gene2GO=myGENE2GO,
  ont="BP",
  params = list(
    scoreType = "pos",
    minSize=5
  )
)
# merge fgsea results

```r
BP_sResults<-merge_enrich_terms(
  cutoff=0.01,
  Input=list(
    PregnantvsLactate="BP_PregnantvsLactate",
    VirginvsLactate="BP_VirginvsLactate",
    VirginvsPregnant="BP_VirginvsPregnant"
  )
)
```

## End(Not run)

---

### myGOs

**myGOs dataset**

**Description**

An example of object returned by `build_GO_SS` method from mouse functional analysis of mouse mammary gland RNA-Seq (2_mouse_bioconductor vignette)

**Usage**

```r
data(myGOs,package="ViSEAGO")
```

**Format**

An object of class `GO_SS-class`.

---

### overLapper

**build all sets combinations intersections**

**Description**

build all sets combinations intersections

**Usage**

```r
overLapper(setlist)
```

**Details**

This internal function use build all sets combinations intersections needed for `upset`.

**Value**

A list.
pkgdiagram

See Also

Other enrich_GO_terms: `Upset()`, `enrich_GO_terms-class`, `show_heatmap()`, `show_table()`

Other visualization: `GOclusters_heatmap()`, `GOcount()`, `GOterms_heatmap()`, `Upset()`, `available_organisms()`, `show_heatmap()`, `show_table()`

Examples

```r
## Not run:
# build all intersections combinations
ViSEAGO::overLapper(setlist)

## End(Not run)
```

pkgdiagram

Display the ViSEAGO package diagram.

Description

This function build the ViSEAGO package diagram available displayed in the vignette.

Usage

```r
pkgdiagram(x = c("white", "white", "white", "white", "white"))
```

Arguments

- `x` rectangles colors for the five subgraphs with all white by defaults

Details

This function use `DiagrammeR` package `grViz` to display the ViSEAGO methods diagram with colored or not subgraphs rectangles.

Value

an html diagram

References

Knut Sveidqvist, Mike Bostock, Chris Pettitt, Mike Daines, Andrei Kashcha and Richard Iannone (2017). DiagrammeR: Create Graph Diagrams and Flowcharts Using R. R package version 0.9.0. https://CRAN.R-project.org/package=DiagrammeR.
Examples

# print diagram without focus
ViSEAGO::pkgdiagram(x=c("white","white","white","white","white"))

# print diagram with focus one the gene list
ViSEAGO::pkgdiagram(x=c("black","white","white","white","white"))

# print diagram with a focus on the annotation step
ViSEAGO::pkgdiagram(x=c("white","black","white","white","white"))

# print diagram with a focus on the enrichment step
ViSEAGO::pkgdiagram(x=c("white","white","black","white","white"))

# print diagram with a focus on the Semantic Similarity step
ViSEAGO::pkgdiagram(x=c("white","white","white","black","white"))

# print diagram with focus on the visualization step
ViSEAGO::pkgdiagram(x=c("white","white","white","white","black"))

runfgsea

perform multilevel preranked gene set enrichment analysis.

Description

This method perform fast gene set enrichment analysis (GSEA) using fgsea package.

Usage

runfgsea(
  geneSel,
  gene2GO,
  ont,
  method = c("fgseaSimple", "fgseaMultilevel"),
  params = list(nperm = 10000, sampleSize = 101, minSize = 1, maxSize = Inf, eps = 0,
                scoreType = c("std", "pos", "neg"), nproc = 0, gseaParam = 1, BPPARAM = NULL, absEps = NULL)
)

## S4 method for signature 'ANY,gene2GO,character'
runfgsea(
  geneSel,
  gene2GO,
  ont,
  method = c("fgseaSimple", "fgseaMultilevel"),
  params = list(nperm = 10000, sampleSize = 101, minSize = 1, maxSize = Inf, eps = 0,
                scoreType = c("std", "pos", "neg"), nproc = 0, gseaParam = 1, BPPARAM = NULL, absEps = NULL)
)
Arguments

geneSel a 2 columns data.table with `preranked` gene identifiers (in first column) based on the statistical values (second column).
gene2GO a `gene2GO-class` object created by `annotate` method.
ont the ontology used is "MF" (Molecular Function), "BP" (Biological Process), or "CC" (Cellular Component).
method `fgsea` method to use with `fgseaSimple` or `fgseaMultilevel`.
params a list with `fgseaSimple` or `fgseaMultilevel` parameters.

Details

This method is a convenient wrapper using a given ontology category (ont argument) in order to perform gene set enrichment analysis using `fgseaSimple` or `fgseaMultilevel` algorithm from `fgsea` package.

The complete GO annotation is required (gene2GO argument), and also a 2 columns data.table with `preranked` gene identifiers (in first column) based on statistical values (second column).

Defaults `fgseaSimple` parameters were used for perform test with `nperm` set to 10,000.
Defaults `fgseaMultilevel` parameters were used for perform test except the `eps` arg that was set to 0 for better pvalues estimation.
A gene frequency (%) of leadingEdge/size is added to output data.table.

Value

a `fgsea-class` object.

References


See Also

Other GO_terms: `GOcount()`, `GOTerms_heatmap()`, `annotate()`, `create_topGOdata()`, `gene2GO-class`, `merge_enrich_terms()`

Examples

```r
# gene list
PregnantvsLactate<-data.table::fread(
  system.file(
    "extdata/data/input",
    "pregnantvsLactate.complete.txt",
    package = "ViSEAGO"
  ),
  select = c("Id","padj")
)

# rank Id based on statistical value (padj here)
```
show_heatmap

Display an interactive or static heatmap.

Description

Display a heatmap in interactive or static mode.

Usage

show_heatmap(object, type, file = NULL, plotly_update = FALSE)

## S4 method for signature 'GO_clusters,character'
show_heatmap(object, type, file = NULL, plotly_update = FALSE)

Arguments

object a GO_clusters-class object from GOterms_heatmap or GOclusters_heatmap.
type could be "GOterms" to display GOterms clustering heatmap, or "GOclusters" to display GOclusters heatmap.
file static image output file name (default to NULL).
plotly_update update plotly html dependencies (default to FALSE).
**show_heatmap**

**Details**

This method displays an interactive heatmap (if file=NULL) from `GO_clusters-class` object for "GOterms" or "GOclusters" type.

A static png image could be printed by setting file argument.

Interactive heatmap cannot be displayed between two R versions. Then interactive view (build with previous R version) can be updated to new R version using `plotly_update` argument setting to TRUE.

**Value**

display or print heatmap.

**See Also**

Other enrich_GO_terms: `Upset()`, `enrich_GO_terms-class`, `overLapper()`, `show_table()`

Other GO_clusters: `GO_clusters-class`, `GOclusters_heatmap()`, `clusters_cor()`, `compare_clusters()`, `show_table()`

Other visualization: `GOclusters_heatmap()`, `GOcount()`, `GOterms_heatmap()`, `Upset()`, `available_organisms()`, `overLapper()`, `show_table()`

**Examples**

```r
# load data example
data(myGOs, package="ViSEAGO")

## Not run:
myGOs<-ViSEAGO::compute_SS_distances(
    myGOs, distance="Wang")

ViSEAGO::MDSplot(myGOs)

Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
    myGOs, showIC=TRUE, showGOlabels=TRUE, GO.tree=list(
        tree=list( distance="Wang", aggreg.method="ward.D2", rotate=NULL ),
        cut=list( dynamic=list( )
        )
    )
)

# build MDS plot for a GO_SS-class distance object
ViSEAGO::MDSplot(myGOs)

# GOtermsHeatmap with default parameters
```
show_heatmap

```r
pamStage=TRUE,
pamRespectsDendro=TRUE,
deepSplit=2,
minClusterSize =2
)
)
),
samples.tree=NULL
)

# Display GO terms heatmap
ViSEAGO::show_heatmap(
  Wang_clusters_wardD2,
  "GOterms"
)

# Print GO terms heatmap
ViSEAGO::show_heatmap(
  Wang_clusters_wardD2,
  "GOterms",
  "GOterms_heatmap.png"
)

# compute clusters of GO terms Semantic Similarity distances
Wang_clusters_wardD2<-ViSEAGO::compute_SS_distances(
  Wang_clusters_wardD2,
  distance="BMA"
)

# GOclusters heatmap
Wang_clusters_wardD2<-ViSEAGO::GOclusters_heatmap(
  Wang_clusters_wardD2,
  tree=list(
    distance="BMA",
    aggreg.method="ward.D2",
    rotate=NULL
  )
)

# Display GO clusters heatmap
ViSEAGO::show_heatmap(
  Wang_clusters_wardD2,
  "GOclusters"
)

# Print GO clusters heatmap
ViSEAGO::show_heatmap(
  Wang_clusters_wardD2,
  "GOclusters",
  "GOclusters_heatmap.png"
)

## End(Not run)
```
show_table

Display an interactive or static table.

Description

This method is used to display or print the table for `enrich_GO_terms-class` or `GO_clusters-class` objects.

Usage

```r
show_table(object, file = NULL)

## S4 method for signature 'ANY'
show_table(object, file = NULL)
```

Arguments

- `object`: an `enrich_GO_terms-class` object from `merge_enrich_terms`, or `GO_clusters-class` object from `GOterms_heatmap`.
- `file`: table output file name (default to `NULL`).

Details

This method displays an interactive table (if `file=NULL`) from `enrich_GO_terms-class` or `GO_clusters-class` objects.

The table could be printed by setting `file` argument.

Value

display or print table

References

https://CRAN.R-project.org/package=DT

See Also

- Other `enrich_GO_terms`: `Upset()`, `enrich_GO_terms-class`, `overLapper()`, `show_heatmap()`
- Other `GO_clusters`: `GO_clusters-class`, `GOclusters_heatmap()`, `clusters_cor()`, `compare_clusters()`, `show_heatmap()`
- Other visualization: `GOclusters_heatmap()`, `GOcount()`, `GOterms_heatmap()`, `Upset()`, `available_organisms()`, `overLapper()`, `show_heatmap()`
Examples

```r
# load example object
data(
  myGOs,
  package="ViSEAGO"
)

# display merge_enrich_terms output
ViSEAGO::show_table(myGOs)

# print merge_enrich_terms output
ViSEAGO::show_table(
  myGOs,
  "myGOs.txt"
)

## Not run:
# compute GO terms Semantic Similarity distances
myGOs<-ViSEAGO::compute_SS_distances(
  distance="Wang"
)

# GOtermsHeatmap with default parameters
Wang_clusters_wardD2<-ViSEAGO::GOterms_heatmap(
  myGOs,
  showIC=TRUE,
  showGOlabels=TRUE,
  GO.tree=list(
    tree=list(
      distance="Wang",
      aggreg.method="ward.D2",
      rotate=NULL
    ),
    cut=list(
      dynamic=list(
        pamStage=TRUE,
        pamRespectsDendro=TRUE,
        deepSplit=2,
        minClusterSize =2
      )
    )
  ),
  samples.tree=NULL
)

# display table of GO_clusters-class object
ViSEAGO::show_table(Wang_clusters_wardD2)

# print table of GO_clusters-class object
ViSEAGO::show_table(
  Wang_clusters_wardD2,
  "Wang_clusters_wardD2.txt"
)
```
## taxonomy

Display Organism Scientific and common name from taxid.

### Description
Display Organism Scientific and common name from taxid.

### Usage

```r
taxonomy(...)```

### Details
This internal function uses `E-utils` summary to display in `data.table` Organism Scientific name and common name, forming a `vector` of taxid.

### Value

A `data.table`.

### See Also

Other genomic resource: `Bioconductor2GO()`, `Custom2GO()`, `Ensembl2GO()`, `EntrezGene2GO()`, `Uniprot2GO()`, `annotate()`, `available_organisms()`, `genomic_ressource-class`

### Examples

```r
# Organism Scientific and common name from taxid
Data <- ViSEAGO::taxonomy("9031")
```

---

## Uniprot2GO

Check available organisms databases at Uniprot.

### Description
Check the Uniprot-GOA available organisms.

### Usage

```r
Uniprot2GO()```
Details
This function downloads the current\_release\_numbers file (ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/current\_release\_numbers.txt) from Uniprot-GOA which contains available organisms.

Value
a genomic\_ressource\_class object required by annotate.

References

See Also
Other genomic\_ressource: Bioconductor2GO(), Custom2GO(), Ensembl2GO(), EntrezGene2GO(), annotate(), available\_organisms(), genomic\_ressource\_class, taxonomy()

Examples
## Not run:
# List Uniprot-GOA available organisms
Uniprot<-ViSEAGO::Uniprot2GO()

## End(Not run)

Upset

Enriched GO terms intersections plot.

Description
This method allows to visualize GO terms intersections between results of enrichment tests.

Usage
Upset(object, file = "./upset.xls")

## S4 method for signature 'ANY'
Upset(object, file = "./upset.xls")

Arguments
object an enrich\_GO\_terms\_class or GO\_clusters\_class objects.
file output file name (default to "./upset.xls")
Details

This function displays and print the intersections of enriched GO terms (p<0.01) between all results provided by `enrich_GO_terms-class` or `GO_clusters-class` objects. The intersections are shown in an upset plot and printed in a table.

Value

print table and upset.

See Also

Other enrich_GO_terms: `enrich_GO_terms-class`, `overLapper()`, `show_heatmap()`, `show_table()`

Other visualization: `GOclusters_heatmap()`, `GOcount()`, `GOterms_heatmap()`, `available_organisms()`, `overLapper()`, `show_heatmap()`, `show_table()`

Examples

```r
# load example object
data(
  myGOs,
  package="ViSEAGO"
)

# print upset
ViSEAGO::Upset(myGOs)
```

---

**ViSEAGO**

**ViSEAGO package**

Description

Easier data mining of biological functions organized into clusters using Gene Ontology and semantic.

Details

The main objective of ViSEAGO workflow is to carry out a data mining of biological functions and establish links between genes involved in the study. We developed ViSEAGO in R to facilitate functional Gene Ontology (GO) analysis of complex experimental design with multiple comparisons of interest.

It allows to study large-scale datasets together and visualize GO profiles to capture biological knowledge. The acronym stands for three major concepts of the analysis: Visualization, Semantic similarity and Enrichment Analysis of Gene Ontology (`pkgdiagram`).

It provides access to the last current GO annotations (`annotate`), which are retrieved from one of NCBI EntrezGene (`Bioconductor2GO`, `EntrezGene2GO`), Ensembl (`Ensembl2GO`) or Uniprot (`Uniprot2GO`) databases for available species (`available_organisms`).
ViSEAGO extends classical functional GO analysis (create_topGOdata) to focus on functional coherence by aggregating closely related biological themes while studying multiple datasets at once (merge_enrich_terms).

It provides both a synthetic and detailed view using interactive functionalities respecting the GO graph structure (MDSplot, GOterms_heatmap, GOclusters_heatmap), and ensuring functional coherence supplied by semantic similarity (build_GO_SS, compute_SS_distances).

ViSEAGO has been successfully applied on several datasets from different species with a variety of biological questions. Results can be easily shared between bioinformaticians and biologists, enhancing reporting capabilities while maintaining reproducibility.
Index

* **GO_clusters**
  clusters_cor, 9
  compare_clusters, 12
  GO_clusters-class, 32
  GOclusters_heatmap, 25
  show_heatmap, 44
  show_table, 47
* **GO_semantic_similarity**
  build_GO_SS, 8
  compute_SS_distances, 15
  GO_SS-class, 33
* **GO_terms GO_clusters**
  semantic_similarity visualization
    MDSplot, 33
* **GO_terms**
  annotate, 3
  create_topGOdata, 17
  gene2GO-class, 24
  GOcount, 28
  GOterms_heatmap, 29
  merge_enrich_terms, 35
  runfGSEA, 42
* **datasets**
  myGos, 40
* **enrich GO_terms**
  enrich_GO_terms-class, 20
  overLapper, 40
  show_heatmap, 44
  show_table, 47
  Upset, 50
* **genomic_ressource**
  annotate, 3
  available_organisms, 6
  Bioconductor2GO, 7
  Custom2GO, 19
  Ensembl2GO, 21
  EntrezGene2GO, 22
  genomic_ressource-class, 25
  taxonomy, 49
  Uniprot2GO, 49
* **internal**
  EntrezGene_orthologs, 23
  overLapper, 40
  pkgdiagram, 41
  taxonomy, 49
* **semantic_similarity**
  GOclusters_heatmap, 25
  GOterms_heatmap, 29
* **visualization**
  available_organisms, 6
  GOclusters_heatmap, 25
  GOcount, 28
  GOterms_heatmap, 29
  overLapper, 40
  show_heatmap, 44
  show_table, 47
  Upset, 50
  annotate, 3, 6–8, 18, 20–25, 28, 31, 36, 43, 49–51
  annotate, character, genomic_ressource-method (annotate), 3
  available_organisms, 3, 4, 6, 7, 20, 21, 23, 25, 26, 28, 31, 41, 45, 47, 49–51
  available_organisms, genomic_ressource-method (available_organisms), 6
  Bioconductor2GO, 3, 4, 6, 7, 20, 21, 23, 25, 49–51
  build_GO_SS, 4, 8, 16, 33, 40, 52
  build_GO_SS, gene2GO, enrich_GO_terms-method (build_GO_SS), 8
  clusters_cor, 9, 13, 26, 32, 45, 47
  clusters_cor, list, character-method (clusters_cor), 9
  compare, 9
  compare_clusters, 10, 12, 26, 32, 45, 47
compare_clusters, list-method
(compare_clusters), 12

compute_SS_distances, 8, 9, 15, 26, 29, 30, 33, 52
compute_SS_distances, ANY, character-method
(compute_SS_distances), 15
create_topGOdata, 4, 17, 24, 28, 31, 35, 36, 43, 52
create_topGOdata, ANY, ANY, gene2GO, character, numeric-method
(create_topGOdata), 17
Custom2GO, 4, 6, 7, 19, 21, 23, 25, 49, 50
cutreeDynamic, 30
data.table, 23, 49
enrich_GO_terms-class, 20
Ensembl2GO, 3, 4, 6, 7, 20, 21, 23, 25, 49–51
EntrezGene2GO, 3, 4, 6, 7, 20, 21, 22, 25, 49–51
EntrezGene_orthologs, 23
fgsea-class, 24
fgseaMultilevel, 24, 43
fgseaSimple, 24, 43
gene2GO-class, 24
genomic_ressource-class, 25
GO_clusters-class, 32
GO_SS-class, 33
GOclusters_heatmap, 6, 9, 10, 12, 13, 25, 28, 31, 32, 41, 44, 45, 47, 51, 52
GOclusters_heatmap, GO_clusters, list-method
(GOclusters_heatmap), 25
GOcount, 4, 6, 18, 24, 26, 28, 31, 36, 41, 43, 45, 47, 51
GOcount, ANY-method (GOcount), 28
godata, 8
GOterms_heatmap, 4, 6, 9, 12, 16, 18, 24, 26, 28, 29, 32, 36, 41, 43–45, 47, 51, 52
GOterms_heatmap, GO_SS-method
(GOterms_heatmap), 29
grViz, 41
hclust, 26, 30
heatmaply, 30
list, 40
listDatasets, 21
listEnsembl, 21
listEnsemblGenomes, 21
MDSplot, 33, 52
MDSplot, ANY-method (MDSplot), 33
merge_enrich_terms, 4, 8, 18, 20, 24, 28, 31, 33, 35, 43, 47, 52
merge_enrich_terms, list-method
(merge_enrich_terms), 35
mgoSim, 16
myGOs, 40
pkgdiagram, 41, 51
runfgea, 4, 18, 24, 28, 31, 35, 36, 42
runfgea, ANY, gene2GO, character-method
(runfgea), 42
show_heatmap, 6, 10, 13, 21, 26, 28, 31, 32, 41, 44, 47, 51
show_heatmap, GO_clusters, character-method
(show_heatmap), 44
show_table, 6, 10, 13, 21, 26, 28, 31, 32, 41, 45, 47, 51
show_table, ANY-method (show_table), 47
taxonomy, 4, 6, 7, 20, 21, 23, 25, 49, 50
Uniprot2GO, 3, 4, 6, 7, 20, 21, 23, 25, 49, 49, 51
Upset, 6, 21, 26, 28, 31, 41, 45, 47, 50
upset, 40
Upset, ANY-method (Upset), 50
useEnsembl, 21
vector, 49
ViSEAGO, 51