Package 'signatureSearch'

November 29, 2024

Title Environment for Gene Expression Searching Combined with Functional Enrichment Analysis

Version 1.20.0

Description This package implements algorithms and data structures for performing gene expression signature (GES) searches, and subsequently interpreting the results functionally with specialized enrichment methods.

Depends R(>= 4.2.0), Rcpp, SummarizedExperiment, org.Hs.eg.db

- **Imports** AnnotationDbi, ggplot2, data.table, ExperimentHub, HDF5Array, magrittr, RSQLite, dplyr, fgsea, scales, methods, qvalue, stats, utils, reshape2, visNetwork, BiocParallel, fastmatch, reactome.db, Matrix, clusterProfiler, readr, DOSE, rhdf5, GSEABase, DelayedArray, BiocGenerics, tibble
- Suggests knitr, testthat, rmarkdown, BiocStyle, signatureSearchData, DT

License Artistic-2.0

SystemRequirements C++11

LinkingTo Rcpp

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.2.3

biocViews Software, GeneExpression, GO, KEGG, NetworkEnrichment, Sequencing, Coverage, DifferentialExpression

URL https://github.com/yduan004/signatureSearch/

BugReports https://github.com/yduan004/signatureSearch/issues

LazyData false

 $git_url \ https://git.bioconductor.org/packages/signatureSearch$

git_branch RELEASE_3_20

git_last_commit edcc107

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-28

Author Yuzhu Duan [aut], Brendan Gongol [cre, aut], Thomas Girke [aut]

Maintainer Brendan Gongol <bgong001@ucr.edu>

Contents

signatureSearch-package
addGESSannot
addMOA
add_pcid
append2H5
build_custom_db
calcGseaStatBatchCpp
cellNtestPlot
cell_info
cell_info2
chembl_moa_list
clue_moa_list
comp_fea_res
create_empty_h5
dim
drugs
drugs10
drug_cell_ranks
dsea_GSEA
dtnetplot
enrichGO2
enrichKEGG2
enrichMOA
enrichReactome
feaResult
feaResult-class
GCT object
gctx2h5
gessResult
gessResult-class
gess_cmap
gess_res_vis
getALLEG
getDb
getSig
getTreats
get_targets
gmt2h5
gseGO2
gseKEGG2
gseReactome
head
lincs_expr_inst_info
lincs_pert_info

lincs_pert_info2	49
lincs_sig_info	49
list2df	50
list_rev	50
mabsGO	51
mabsKEGG	52
mabsReactome	53
matrix2h5	54
meanExpr2h5	54
moa_conn	55
parse_gctx	56
qSig	57
qSig-class	58
rand_query_ES	59
read_gmt	60
result	61
runWF	61
set_readable	63
show	64
sim_score_grp	65
tail	66
targetList	66
tarReduce	67
vec_char_redu	67
	69
	02

Index

signatureSearch-package

Environment for Gene Expression Signature Searching Combined with Functional Enrichment Analysis

Description

Welcome to the signatureSearch package! This package implements algorithms and data structures for performing gene expression signature (GES) searches, and subsequently interpreting the results functionally with specialized enrichment methods. These utilities are useful for studying the effects of genetic, chemical and environmental perturbations on biological systems. Specifically, in drug discovery they can be used for identifying novel modes of action (MOA) of bioactive compounds from reference databases such as LINCS containing the genome-wide GESs from tens of thousands of drug and genetic perturbations (Subramanian et al. 2017)

A typical GES search (GESS) workflow can be divided into two major steps. First, GESS methods are used to identify perturbagens such as drugs that induce GESs similar to a query GES of interest. The queries can be drug-, disease- or phenotype-related GESs. Since the MOAs of most drugs in the corresponding reference databases are known, the resulting associations are useful to gain insights into pharmacological and/or disease mechanisms, and to develop novel drug repurposing approaches.

Second, specialized functional enrichment analysis (FEA) methods using annotations systems, such as Gene Ontologies (GO), KEGG and Reactome pathways have been developed and implemented in this package to efficiently interpret GESS results. The latter are usually composed of lists of perturbagens (e.g. drugs) ranked by the similarity metric of the corresponding GESS method.

Finally, network reconstruction functionalities are integrated for visualizing the final results, e.g. in form of drug-target networks.

Details

The GESS methods include CMAP, LINCS, gCMAP, Fisher and Cor. For detailed description, please see help files of each method. Most methods can be easily paralleled for multiple query signatures.

GESS results are lists of perturbagens (here drugs) ranked by their signature similarity to a query signature of interest. Interpreting these search results with respect to the cellular networks and pathways affected by the top ranking drugs is difficult. To overcome this challenge, the knowledge of the target proteins of the top ranking drugs can be used to perform functional enrichment analysis (FEA) based on community annotation systems, such as Gene Ontologies (GO), pathways (e.g. KEGG, Reactome), drug MOAs or Pfam domains. For this, the ranked drug sets are converted into target gene/protein sets to perform Target Set Enrichment Analysis (TSEA) based on a chosen annotation system. Alternatively, the functional annotation categories of the targets can be assigned to the drugs directly to perform Drug Set Enrichment Analysis (DSEA). Although TSEA and DSEA are related, their enrichment results can be distinct. This is mainly due to duplicated targets present in the test sets of the TSEA methods, whereas the drugs in the test sets of DSEA are usually unique. Additional reasons include differences in the universe sizes used for TSEA and DSEA.

Importantly, the duplications in the test sets of the TSEA are due to the fact that many drugs share the same target proteins. Standard enrichment methods would eliminate these duplications since they assume uniqueness in the test sets. Removing duplications in TSEA would be inappropriate since it would erase one of the most important pieces of information of this approach. To solve this problem, we have developed and implemented in this package weighting methods (dup_hyperG, mGSEA and meanAbs) for duplicated targets, where the weighting is proportional to the frequency of the targets in the test set.

Instead of translating ranked lists of drugs into target sets, as for TSEA, the functional annotation categories of the targets can be assigned to the drugs directly to perform DSEA instead. Since the drug lists from GESS results are usually unique, this strategy overcomes the duplication problem of the TSEA approach. This way classical enrichment methods, such as GSEA or tests based on the hypergeometric distribution, can be readily applied without major modifications to the underlying statistical methods. As explained above, TSEA and DSEA performed with the same enrichment statistics are not expected to generate identical results. Rather they often complement each other's strengths and weaknesses.

To perform TSEA and DSEA, drug-target annotations are essential. They can be obtained from several sources, including DrugBank, ChEMBL, STITCH, and the Touchstone dataset from the LINCS project (https://clue.io/). Most drug-target annotations provide UniProt identifiers for the target proteins. They can be mapped, if necessary via their encoding genes, to the chosen functional annotation categories, such as GO or KEGG. To minimize bias in TSEA or DSEA, often caused by promiscuous binders, it can be beneficial to remove drugs or targets that bind to large numbers of distinct proteins or drugs, respectively.

Note, most FEA tests involving proteins in their test sets are performed on the gene level in signatureSearch. This way one can avoid additional duplications due to many-to-one relationships among proteins and their encoding gents. For this, the corresponding functions in signatureSearch will usually translate target protein sets into their encoding gene sets using identifier mapping resources from R/Bioconductor such as the org.Hs.eg.db annotation package. Because of this as well as simplicity, the text in the vignette and help files of this package will refer to the targets of drugs almost interchangeably as proteins or genes, even though the former are the direct targets and the latter only the indirect targets of drugs.

addGESSannot

Terminology

The term Gene Expression Signatures (GESs) can refer to at least four different situations of preprocessed gene expression data: (1) normalized gene expression intensity values (or counts for RNA-Seq); (2) log2 fold changes (LFC), z-scores or p-values obtained from analysis routines of differentially expressed genes (DEGs); (3) rank transformed versions of the expression values obtained under (1) and (2); and (4) gene identifier sets extracted from the top and lowest ranks under (3), such as n top up/down regulated DEGs.

Author(s)

- Yuzhu Duan (yduan004@ucr.edu)
- Brendan Gongol (bgong001@ucr.edu>)
- Thomas Girke (thomas.girke@ucr.edu)

References

Subramanian, Aravind, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, et al. 2017. A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell 171 (6): 1437-1452.e17. http://dx.doi.org/10.1016/j.cell.2017.10.049

Lamb, Justin, Emily D Crawford, David Peck, Joshua W Modell, Irene C Blat, Matthew J Wrobel, Jim Lerner, et al. 2006. The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. Science 313 (5795): 1929-35. http://dx.doi.org/10.1126/science.1132939

Sandmann, Thomas, Sarah K Kummerfeld, Robert Gentleman, and Richard Bourgon. 2014. gCMAP: User-Friendly Connectivity Mapping with R. Bioinformatics 30 (1): 127-28. http://dx.doi.org/10.1093/bioinformatics/btt

Subramanian, Aravind, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, et al. 2005. Gene Set Enrichment Analysis: A Knowledge-Based Approach for Interpreting Genome-Wide Expression Profiles. Proc. Natl. Acad. Sci. U. S. A. 102 (43): 15545-50. http://dx.doi.org/10.1073/pnas.0506580102

See Also

Methods for GESS:

• gess_cmap, gess_lincs, gess_gcmap gess_fisher, gess_cor

Methods for FEA:

- TSEA methods: tsea_dup_hyperG, tsea_mGSEA, tsea_mabs
- DSEA methods: dsea_hyperG, dsea_GSEA

addGESSannot

Add Compound Annotation Info to GESS Result Table

Description

This function supports adding customized compound annotation table to the GESS result table if provided. It then automatically adds the target gene symbols, MOAs and PubChem CID columns (t_gn_sym, MOAss, PCIDss) if the table contains a column that stores compound names.

Usage

```
addGESSannot(
  gess_tb,
  refdb,
  cmp_annot_tb = NULL,
  by = "pert",
   cmp_name_col = "pert"
)
```

Arguments

gess_tb	tibble or data.frame object of GESS result, can be accessed by the result method on the gessResult object from gess_* functions. Or a customized data frame that contains a pert column that stores compound id or name.
refdb	character(1), reference database that can be accessed by the refdb method on the gessResult object. If gess_tb is a customized table, refdb can be just set as 'custom'.
cmp_annot_tb	data.frame or tibble of compound annotation table. This table contains anno- tation information for compounds stored under pert column of gess_tb. Set to NULL if not available. This table should not contain columns with names of "t_gn_sym", "MOAss" or "PCIDss", these three columns will be added inter- nally and thus conserved by the function. If they are contained in cmp_annot_tb, they will be overwritten. If users want to maintain these three columns in the provided annotation table, give them different names.
by	character(1), column name in cmp_annot_tb that can be merged with pert column in gess_tb. If refdb is set as 'lincs2', it will be merged with pert_id column in the GESS result table. If cmp_annot_tb is NULL, by is ignored.
cmp_name_col	character(1), column name in gess_tb or cmp_annot_tb that store compound names. If there is no compound name column, set to NULL. If cmp_name_col is available, three additional columns (t_gn_sym, MOAss, PCIDss) are automatically added by using get_targets, CLUE touchstone compound MOA annotation, and 2017 lincs_pert_info annotation table, respectively as annotation sources. t_gn_sym: target gene symbol, MOAss: MOA annotated from signatureSearch, PCIDss: PubChem CID annotated from signatureSearch.

Value

tibble of gess_tb with target, MOA, PubChem CID annotations and also merged with user provided compound annotation table.

Examples

addMOA

Description

The MOA annotation is a list of MOA name to drug name mappings. This functions add the MOA column to data frame when data frame have a column with compound names

Usage

```
addMOA(df, drug_col, moa_list)
```

Arguments

df	data frame that must contains a column with compound names
drug_col	character (1), name of the column that stores compound names
moa_list	a list object of MOA name (e.g. HDAC inhibitor) to compound name mappings

Value

data frame with an added MOAss column

Examples

add_pcid

Add PCID to drug data frame

Description

This function can be used to add the PCIDss (PubChem CID column added from signatureSearch package) column to a data frame that have a column store compound names. The compound name to PubChem CID annotation is obtained from lincs_pert_info in 2017.

Usage

add_pcid(df, drug_col = "pert")

Arguments

df	data frame or tibble object
drug_col	name of the column that store compound names in df

Value

tibble object with an added PCIDss column

Examples

```
data("lincs_pert_info")
# gess_tb2 <- add_pcid(gess_tb)</pre>
```

append2H5

Append Matrix to HDF5 File

Description

Function to write matrix data to an existing HDF5 file. If the file contains already matrix data then both need to have the same number of rows. The append will be column-wise.

Usage

```
append2H5(x, h5file, name = "assay", printstatus = TRUE)
```

Arguments

X	matrix object to write to an HDF5 file. If the HDF5 file is not empty, the exported matrix data needs to have the same number rows as the matrix stored in the HDF5 file, and will be appended column-wise to the existing one.
h5file	character(1), path to existing HDF5 file that can be empty or contain matrix data
name	The name of the dataset in the HDF5 file.
printstatus	logical, whether to print status

Value

HDF5 file storing exported matrix

Examples

```
mat <- matrix(1:12, nrow=3)
rownames(mat) <- paste0("r", 1:3); colnames(mat) <- paste0("c", 1:4)
tmp_file <- tempfile(fileext=".h5")
create_empty_h5(tmp_file)
append2H5(mat, tmp_file)
rhdf5::h5ls(tmp_file)</pre>
```

build_custom_db build_custom_db

Description

Build custom reference signature database for GESS methods

Usage

```
build_custom_db(df, h5file)
```

Arguments

df	data.frame or matrix containing genome-wide or close to genome-wide GESs of perturbation experiments.
	The row name slots are expected to contain gene or transcript IDs (e.g. Entrez ids), while the column names are expected to have this structure: '(drug)(cell)(factor)', e.g. 'sirolimusMCF7trt_cp'. This format is flexible enough to encode most perturbation types of biological samples. For example, gene knockdown or over expression treatments can be specified by assigning the ID of the affected gene to 'drug', and 'ko' or 'ov' to 'factor', respectively. An example for a knockdown treatment would look like this: 'P53MCF7ko'.
h5file	character vector of length 1 containing the path to the destination hdf5 file

Details

The perturbation-based gene expression data, here provided as data.frame or matrix, will be stored in an HDF5 file. The latter can be used as reference database by compatible GESS methods of signatureSearch. Various types of pre-processed gene expression data can be used here, such as normalized gene expression intensities (or counts for RNA-Seq); log2 fold changes (LFC), Z-scores or p-values obtained from analysis routines of differentially expressed genes (DEGs).

Value

HDF5 file

Examples

calcGseaStatBatchCpp Calculates GSEA statistic valus for all gene sets in 'selectedStats' list.

Description

Takes O(n + mKlogK) time, where n is the number of genes, m is the number of gene sets, and k is the mean gene set size.

Usage

```
calcGseaStatBatchCpp(stats, selectedGenes, geneRanks)
```

cell_info

Arguments

stats	Numeric vector of gene-level statistics sorted in decreasing order
selectedGenes	List of integer vector with integer gene IDs (from 1 to n)
geneRanks	Integer vector of gene ranks

Value

Numeric vector of GSEA statistics of the same length as 'selectedGenes' list

cellNtestPlot Number of Tests in Cell Types

Description

Bar plot of number of perturbations/compounds tested in cell types where cell types are grouped by 'primary site'.

Usage

cellNtestPlot(refdb)

Arguments

```
refdb
```

character(1), one of "lincs", "lincs_expr", "cmap" or "cmap_expr" when using the pre-generated CMAP/LINCS databases or path to the HDF5 file generated with the build_custom_db function. The details is shown in the 'refdb' argument of the qSig function

Value

Faceted bar plot

Examples

```
refdb <- system.file("extdata", "sample_db.h5", package="signatureSearch")
cellNtestPlot(refdb)</pre>
```

cell_info

LINCS 2017 Cell Type Information

Description

It contains cell type (tumor or normal), primary site and subtype annotations of cells in LINCS 2017 database.

Usage

cell_info

cell_info2

Format

A tibble object with 30 rows and 4 columns.

Examples

```
# Load object
data(cell_info)
head(cell_info)
```

cell_info2

LINCS 2020 Cell Type Information

Description

It contains cell type (tumor or normal), primary site, subtype etc. annotations of cells in LINCS 2020 database.

Usage

cell_info2

Format

A tibble object with 240 rows and 21 columns.

Examples

```
# Load object
data(cell_info2)
head(cell_info2)
```

chembl_moa_list MOA to Gene Mappings

Description

It is a list containing MOA terms to gene Entrez id mappings from ChEMBL database

Usage

chembl_moa_list

Format

An object of class list of length 1099.

Examples

```
# Load object
data(chembl_moa_list)
head(chembl_moa_list)
```

clue_moa_list

Description

It is a list containing MOA terms to drug name mappings obtained from Touchstone database at CLUE website (https://clue.io/)

Usage

clue_moa_list

Format

An object of class list of length 701.

Examples

```
# Load object
data(clue_moa_list)
head(clue_moa_list)
```

comp_fea_res Plot for Comparing Ranking Results of FEA Methods

Description

Dot plot for comparing the top ranking functional categories from different functional enrichment analysis (FEA) results. The functional categories are plotted in the order defined by their mean rank across the corresponding FEA results.

Usage

```
comp_fea_res(
  table_list,
  rank_stat = "pvalue",
  Nshow = 20,
  Nchar = 50,
  scien = FALSE,
  ...
)
```

Arguments

table_list	a named list of tibbles extracted from feaResult objects, e.g. generated with different FEA methods.
rank_stat	character(1), column name of the enrichment statisic used for ranking the func- tional categories, e.g. 'pvalue' or 'p.adjust'. Note, the chosen column name needs to be present in each tibble of 'table_list'.

Nshow	integer defining the number of the top functional categories to display in the plot after re-ranking them across FEA methods
Nchar	integer defining number of characters displayed (exceeded characters were replaced by '') in the description of each item
scien	TRUE or FALSE, indicating whether the rank_stat is rounded to the scientific format with 3 digits
	Other arguments passed on to geom_point

Details

The 'comp_fea_res' function computes the mean rank for each functional category across different FEA result instances and then re-ranks them based on that. Since the functional categories are not always present in all enrichment results, the mean rank of a functional category is corrected by an adjustment factor that is the number of enrichment result methods used divided by the number of occurences of a functional category. For instance, if a functional category is only present in the result of one method, its mean rank will be increased accordingly. Subsequently, the re-ranked functional categories are compared in a dot plot where the colors represent the values of the enrichment statistic chosen under the rank_stat argument.

Value

ggplot2 graphics object

Examples

```
method1 <- data.frame("ID"=paste0("GO:", 1:5),
                                  "Description"=paste0("desc", 1:5),
                             "pvalue"=c(0.0001, 0.002, 0.004, 0.01, 0.05))
method2 <- data.frame("ID"=paste0("GO:", c(1,3,5,4,6)),
                          "Description"=paste0("desc", c(1,3,5,4,6)),
                          "pvalue"=c(0.0003, 0.0007, 0.003, 0.006, 0.04))
table_list <- list("method1" = method1, "method2"=method2)
comp_fea_res(table_list, rank_stat="pvalue", Nshow=20)
```

create_empty_h5 Create Empty HDF5 File

Description

This function can be used to create an empty HDF5 file where the user defines the file path and compression level. The empty HDF5 file has under its root group three data slots named 'assay', 'colnames' and 'rownames' for storing a numeric matrix along with its column names (character) and row names (character), respectively.

Usage

```
create_empty_h5(h5file, delete_existing = FALSE, level = 6)
```

Arguments

h5file	character(1), path to the HDF5 file to be created	
delete_existing		
	logical, whether to delete an existing HDF5 file with identical path	
	The compression level used, here given as integer value between 0 (no compression) and 9 (highest and slowest compression).	

Value

empty HDF5 file

Examples

```
tmp_file <- tempfile(fileext=".h5")
create_empty_h5(tmp_file, level=6)</pre>
```

```
dim
```

Dimensions of an Object

Description

Retrieve dimension of the result table in the gessResult, and feaResult objects

Usage

```
## S4 method for signature 'gessResult'
dim(x)
## S4 method for signature 'feaResult'
```

dim(x)

Arguments

x an R object

Value

dim attribute of the result table

Examples

drugs

Description

The drugs generic extracts or assign the drug names/ids stored in the drugs slot of an feaResult object.

Usage

```
drugs(x)
drugs(x) <- value
## S4 method for signature 'feaResult'
drugs(x)
## S4 replacement method for signature 'feaResult'
drugs(x) <- value</pre>
```

Arguments

х	feaResult object
value	A character vector of drug names

Value

character vector

An feaResult object with new assigned drugs slot

Examples

drugs10

Drug Names Used in Examples

Description

A character vector containing the names of the top 10 drugs in the GESS result from the gess_lincs method used in the vignette of signatureSearch.

Usage

drugs10

Format

An object of class character of length 10.

Examples

Load drugs object
data(drugs10)
drugs10

drug_cell_ranks

Summary ranking statistics across cell types

Description

The drug_cell_ranks function returns from a gessResult object the ranks of the perturbagens (e.g. drugs) for each cell type. The results are arranged in separate columns of a data.frame. Additionally, it includes in the last columns summary ranking statistics across all cell types, such as min, mean and max values.

Usage

drug_cell_ranks(gessResult)

Arguments

gessResult 'gessResult' object

Value

data.frame

Examples

dsea_GSEA

Description

The Drug Set Enrichment Analysis (DSEA) with GSEA algorithm (dsea_GSEA function) performs DSEA with the GSEA algorithm from Subramanian et al. (2005). In case of DSEA, drug identifiers combined with their ranking scores of an upstream GESS method are used, such as the NCS values from the LINCS method. To use drug instead of gene labels for GSEA, the former are mapped to functional categories, including GO or KEGG, based on drug-target interaction annotations provided by databases such as DrugBank, ChEMBL, CLUE or STITCH.

The DSEA with Hypergeometric Test (dsea_hyperG) performs DSEA based on the hypergeometric distribution. In case of DSEA, the identifiers of the top ranking drugs from a GESS result table are used. To use drug instead of gene labels for this test, the former are mapped to functional categories, including GO, KEGG or Mode of Action (MOA) categories, based on drug-target interaction annotations provided by databases such as DrugBank, ChEMBL, CLUE or STITCH. Currently, the MOA annotation used by this function are from the CLUE website (https://clue.io).

Compared to the related Target Set Enrichment Analysis (TSEA), the DSEA approach has the advantage that the drugs in the query test sets are usually unique allowing to use them without major modifications to the underlying statistical method(s).

The Target Set Enrichment Analysis (TSEA) with hypergeometric test (tsea_dup_hyperG function) performs TSEA based on a modified hypergeometric test that supports test sets with duplications. This is achieved by maintaining the frequency information of duplicated items in form of weighting values.

The TSEA with mGSEA algorithm (tsea_mGSEA function) performs a Modified Gene Set Enrichment Analysis (mGSEA) that supports test sets (e.g. genes or protein IDs) with duplications. The duplication support is achieved by a weighting method for duplicated items, where the weighting is proportional to the frequency of the items in the test set.

The TSEA with meanAbs (tsea_mabs) method is a simple but effective functional enrichment statistic (Fang et al., 2012). As required for TSEA, it supports query label sets (here for target proteins/genes) with duplications by transforming them to score ranked label lists and then calculating mean absolute scores of labels in label set S.

Usage

```
dsea_GSEA(
    drugList,
    type = "GO",
    ont = "BP",
    exponent = 1,
    nPerm = 1000,
    minGSSize = 10,
    maxGSSize = 500,
    pvalueCutoff = 0.05,
    pAdjustMethod = "BH"
)
dsea_hyperG(
    drugs,
```

dsea_GSEA

```
type = "GO",
  ont = "BP",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  qvalueCutoff = 0.2,
  minGSSize = 10,
  maxGSSize = 500
)
tsea_dup_hyperG(
  drugs,
  universe = "Default",
  type = "GO",
  ont = "MF",
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05,
  minGSSize = 5,
 maxGSSize = 500,
  dt_anno = "all",
  readable = FALSE
)
tsea_mGSEA(
  drugs,
  type = "GO",
  ont = "MF",
  nPerm = 1000,
  exponent = 1,
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  minGSSize = 5,
  maxGSSize = 500,
  verbose = FALSE,
  dt_anno = "all",
  readable = FALSE
)
tsea_mabs(
  drugs,
  type = "GO",
  ont = "MF",
  nPerm = 1000,
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  minGSSize = 5,
  maxGSSize = 500,
  dt_anno = "all",
  readable = FALSE
)
```

Arguments

drugList	named numeric vector, where the names represent drug labels and the numeric component scores. This can be all drugs of a GESS result that are ranked by GESS scores, such as NCS scores from the LINCS method. Note, drugs with scores of zero are ignored by this method.
type	one of 'GO', 'KEGG' or 'Reactome' if TSEA methods. type can also be set as 'MOA' is DSEA methods are used.
ont	character(1). If type is 'GO', assign ont (ontology) one of 'BP', 'MF', 'CC' or 'ALL'. If type is 'KEGG' or 'Reactome', ont is ignored.
exponent	integer value used as exponent in GSEA algorithm. It defines the weight of the items in the item set ${\cal S}.$
nPerm	integer defining the number of permutation iterations for calculating p-values
minGSSize	integer, minimum size of each gene set in annotation system. Annotation cate- gories with less than minGSSize genes/drugs will be ignored by enrichment test. If type is 'MOA', it may be beneficial to set minGSSize to lower values (e.g. 2) than for other functional annotation systems. This is because certain MOA categories contain only 2 drugs.
maxGSSize	integer, maximum size of each gene set in annotation system. Annotation categories with more genes/drugs annotated than maxGSSize will be ignored by enrichment test.
pvalueCutoff	double, p-value cutoff to return only enrichment results for functional categories meeting a user definable confidence threshold
pAdjustMethod	p-value adjustment method, one of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr'
drugs	character vector containing drug identifiers used for functional enrichment test- ing. This can be the top ranking drugs from a GESS result. Internally, drug test sets are translated to the corresponding target protein test sets based on the drug-target annotations provided under the dt_anno argument.
qvalueCutoff	double, qvalue cutoff, similar to pvalueCutoff
universe	character vector defining the universe of genes/proteins. If set as 'Default', it uses all genes/proteins present in the corresponding annotation system (e.g. GO, KEGG or Reactome). If 'type' is 'GO', it can be assigned a custom vector of gene SYMBOL IDs. If 'type' is 'KEGG' or 'Reactome', the vector needs to contain Entrez gene IDs.
dt_anno	drug-target annotation source. It is the same argument as the database argument of the get_targets function. Usually, it is recommended to set the 'dt_anno' to 'all' since it provides the most complete drug-target annotations. Choosing a single annotation source results in sparser drug-target annotations (particularly CLUE), and thus less complete enrichment results.
readable	TRUE or FALSE, it applies when type is 'KEGG' or 'Reactome' indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the result table.
verbose	TRUE or FALSE, print message or not

Details

The classical hypergeometric test assumes uniqueness in its test sets. To maintain the duplication information in the test sets used for TSEA, the values of the total number of genes/proteins in

the test set and the number of genes/proteins in the test set annotated at a functional category are adjusted by maintaining their frequency information in the test set rather than counting each entry only once. Removing duplications in TSEA would be inappropriate since it would erase one of the most important pieces of information of this approach.

The original GSEA method proposed by Subramanian et at., 2005 uses predefined gene sets S defined by functional annotation systems such as GO and KEGG. The goal is to determine whether the genes in S are randomly distributed throughout a ranked test gene list L (e.g. all genes ranked by log2 fold changes) or enriched at the top or bottom of the test list. This is expressed by an Enrichment Score (ES) reflecting the degree to which a set S is overrepresented at the extremes of L.

For TSEA, the query is a target protein set where duplicated entries need to be maintained. To perform GSEA with duplication support, here referred to as mGSEA, the target set is transformed to a score ranked target list $L_t ar$ of all targets provided by the corresponding annotation system. For each target in the query target set, its frequency is divided by the number of targets in the target set, which is the weight of that target. For targets present in the annotation system but absent in the target set, their scores are set to 0. Thus, every target in the annotation system will be assigned a score and then sorted decreasingly to obtain $L_t ar$.

In case of TSEA, the original GSEA method cannot be used directly since a large portion of zeros exists in L_tar . If the scores of the genes in set S are all zeros, N_R (sum of scores of genes in set S) will be zero, which cannot be used as the denominator. In this case, ES is set to -1. If only some genes in set S have scores of zeros then N_R is set to a larger number to decrease the weight of the genes in S that have non-zero scores.

The reason for this modification is that if only one gene in gene set S has a non-zero score and this gene ranks high in $L_t ar$, the weight of this gene will be 1 resulting in an ES(S) close to 1. Thus, the original GSEA method will score the gene set S as significantly enriched. However, this is undesirable because in this example only one gene is shared among the target set and the gene set S. Therefore, giving small weights (lowest non-zero score in $L_t ar$) to genes in S that have zero scores could decrease the weight of the genes in S that have non-zero scores, thereby decreasing the false positive rate. To favor truly enriched functional categories (gene set S) at the top of $L_t ar$, only gene sets with positive ES are selected.

The input for the mabs method is L_tar , the same as for mGSEA. In this enrichment statistic, mabs(S), of a label (e.g. gene/protein) set S is calculated as mean absolute scores of the labels in S. In order to adjust for size variations in label set S, 1000 random permutations of L_tar are performed to determine mabs(S, pi). Subsequently, mabs(S) is normalized by subtracting the median of the mabs(S, pi) and then dividing by the standard deviation of mabs(S, pi) yielding the normalized scores Nmabs(S). Finally, the portion of mabs(S, pi) that is greater than mabs(S) is used as nominal p-value (Fang et al., 2012). The resulting nominal p-values are adjusted for multiple hypothesis testing using the Benjamini-Hochberg method.

Value

feaResult object, the result table contains the enriched functional categories (e.g. GO terms or KEGG pathways) ranked by the corresponding enrichment statistic.

Column description

Descriptions of the columns in FEA result tables stored in the feaResult object that can be accessed with the result method in tabular format, here tibble.

- ont: in case of GO, one of BP, MF, CC, or ALL
- ID: GO or KEGG IDs

- · Description: description of functional category
- GeneRatio: ratio of genes in the test set that are annotated at a specific GO node or KEGG pathway
- BgRatio: ratio of background genes that are annotated at a specific GO node or KEGG pathway
- itemID: IDs of items (genes for TSEA, drugs for DSEA) overlapping among test and annotation sets.
- setSize: size of the functional category
- pvalue from tsea_dup_hyperG: raw p-value of enrichment test
- p.adjust: p-value adjusted for multiple hypothesis testing based on method specified under pAdjustMethod argument
- qvalue: q value calculated with R's qvalue function to control FDR
- enrichmentScore: ES from the GSEA algorithm (Subramanian et al., 2005). The score is calculated by walking down the gene list L, increasing a running-sum statistic when we encounter a gene in S and decreasing when it is not. The magnitude of the increment depends on the gene scores. The ES is the maximum deviation from zero encountered in the random walk. It corresponds to a weighted Kolmogorov-Smirnov-like statistic.
- NES: Normalized enrichment score. The positive and negative enrichment scores are normalized separately by permutating the composition of the gene list L nPerm times, and dividing the enrichment score by the mean of the permutation ES with the same sign.
- pvalue from tsea_mGSEA: The nominal p-value of the ES is calculated using a permutation test. Specifically, the composition of the gene list L is permuted and the ES of the gene set is recomputed for the permutated data generating a null distribution for the ES. The p-value of the observed ES is then calculated relative to this null distribution.
- leadingEdge: Genes in the gene set S (functional category) that appear in the ranked list L at, or before, the point where the running sum reaches its maximum deviation from zero. It can be interpreted as the core of a gene set that accounts for the enrichment signal.
- ledge_rank: Ranks of genes in 'leadingEdge' in gene list L.
- mabs: given a scored ranked gene list L, mabs(S) represents the mean absolute scores of the genes in set S.
- Nmabs: normalized mabs(S)

References

GSEA algorithm: Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Mesirov, J. P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America, 102(43), 15545-15550. URL: https://doi.org/10.1073/pnas.0506580102

MeanAbs algorithm: Fang, Z., Tian, W., & Ji, H. (2012). A network-based gene-weighting approach for pathway analysis. Cell Research, 22(3), 565-580. URL: https://doi.org/10.1038/cr.2011.149

See Also

feaResult, GO_DATA_drug

Examples

```
data(drugs10)
dl <- c(rev(seq(0.1, 0.5, by=0.05)), 0)
names(dl)=drugs10
## KEGG annotation system
# gsea_k_res <- dsea_GSEA(drugList=dl, type="KEGG", exponent=1, nPerm=100,</pre>
                        pvalueCutoff=0.5, minGSSize=2)
#
# result(gsea_k_res)
## GO annotation system
# hyperG_res <- dsea_hyperG(drugs=drugs10, type="G0", ont="MF")</pre>
# result(hyperG_res)
## KEGG annotation system
# hyperG_k_res <- dsea_hyperG(drugs=drugs10, type="KEGG",</pre>
#
                            pvalueCutoff=1, qvalueCutoff=1,
#
                            minGSSize=10, maxGSSize=500)
# result(hyperG_k_res)
## GO annotation system
# res1 <- tsea_dup_hyperG(drugs=drugs10, universe="Default",</pre>
                        type="G0", ont="MF", pvalueCutoff=0.05,
pAdjustMethod="BH", qvalueCutoff=0.1,
#
#
                        minGSSize=5, maxGSSize=500)
#
# result(res1)
#
## KEGG annotation system
# res2 <- tsea_dup_hyperG(drugs=drugs10, type="KEGG",</pre>
                        pvalueCutoff=0.1, qvalueCutoff=0.2,
#
#
                        minGSSize=10, maxGSSize=500)
#
## Reactome annotation system
# res3 <- tsea_dup_hyperG(drugs=drugs10, type="Reactome",</pre>
#
                        pvalueCutoff=1, qvalueCutoff=1)
## GO annotation system
# res1 <- tsea_mGSEA(drugs=drugs10, type="GO", ont="MF", exponent=1,</pre>
#
                   nPerm=1000, pvalueCutoff=1, minGSSize=5)
# result(res1)
# res2 <- tsea_mGSEA(drugs=drugs10, type="KEGG", exponent=1,</pre>
                   nPerm=100, pvalueCutoff=1, minGSSize=5)
#
# result(res2)
## Reactome annotation system
# res3 <- tsea_mGSEA(drugs=drugs10, type="Reactome", pvalueCutoff=1)</pre>
# result(res3)
## GO annotation system
# res1 <- tsea_mabs(drugs=drugs10, type="GO", ont="MF", nPerm=1000,</pre>
#
                  pvalueCutoff=0.05, minGSSize=5)
# result(res1)
## KEGG annotation system
```

dtnetplot

```
# res2 <- tsea_mabs(drugs=drugs10, type="KEGG", nPerm=1000,
# pvalueCutoff=0.05, minGSSize=5)
# result(res2)
## Reactome annotation system
# res3 <- tsea_mabs(drugs=drugs10, type="Reactome", pvalueCutoff=1)
# result(res3)
```

dtnetplot

Drug-Target Network Visualization

Description

Functional modules of GESS and FEA results can be rendered as interactive drug-target networks using the dtnetplot function form signatureSearch. For this, a character vector of drug names along with an identifier of a chosen functional category are passed on to the drugs and set arguments, respectively. The resulting plot depicts the corresponding drug-target interaction network. Its interactive features allow the user to zoom in and out of the network, and to select network components in the drop-down menu located in the upper left corner of the plot.

Usage

dtnetplot(drugs, set, ont = NULL, desc = NULL, verbose = FALSE, ...)

Arguments

drugs	A character vector of drug names
set	character(1) GO term ID, KEGG or Reactome pathway ID. Alternatively, a character vector of gene SYMBOLs can be assigned.
ont	if 'set' is a GO term ID, 'ont' is the corresponding ontology that GO term be- longs to. One of 'BP', 'MF' or 'CC'. If 'set' is anything else, 'ont' is ignored.
desc	character(1), description of the chosen functional category or target set
verbose	TRUE or FALSE, whether to print messages
	Other arguments passed on to visNetwork function.

Value

visNetwork plot and a list of drugs and targets that have interactions

Examples

```
data(drugs10)
dtnetplot(drugs=drugs10,
    set=c("HDAC1", "HDAC2", "HDAC3", "HDAC11", "FOX2"),
    desc="NAD-dependent histone deacetylase activity (H3-K14 specific)")
```

enrichGO2

Description

Given a vector of gene identifiers, this function returns GO term enrichment results based on a hypergeometric test with duplication support in the test set.

Usage

```
enrichGO2(
  gene,
  OrgDb,
  keytype = "SYMBOL",
  ont = "MF",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe,
  qvalueCutoff = 0.2,
  minGSSize = 5,
  maxGSSize = 500,
  pool = FALSE
)
```

Arguments

gene	a vector of gene SYMBOL ids (here the test set)
OrgDb	OrgDb
keytype	Gene ID type of test set
ont	One of "MF", "BP", "CC" or "ALL"
pvalueCutoff	pvalue cutoff
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes
qvalueCutoff	qvalue cutoff
minGSSize	minimum size of each gene set in annotation system
maxGSSize	maximum size of each gene set in annotation system
pool	If ont='ALL', whether 3 GO ontology should be combined

Value

A feaResult instance.

See Also

feaResult-class

enrichKEGG2

Examples

```
# The method supports duplicated elements in 'gene',
# which should be gene SYMBOL ids for GO term enrichment.
gene <- c(rep("HDAC1",4), rep("HDAC3",2), "SOX8", "KLK14")
# data(targetList)
# ego <- enrichGO2(gene = gene, OrgDb="org.Hs.eg.db", ont="MF",
# universe=names(targetList))
```

```
enrichKEGG2
```

KEGG Pathway Enrichment with Hypergeometric Test

Description

Given a vector of gene identifiers, this function returns KEGG pathway enrichment results based on a hypergeometric test with duplication support in the test set.

Usage

```
enrichKEGG2(
  gene,
  organism = "hsa",
  keyType = "kegg",
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  universe,
  minGSSize = 5,
  maxGSSize = 500,
  qvalueCutoff = 0.2,
  readable = FALSE
)
```

Arguments

gene	a vector of entrez gene ids (here the test set)
organism	supported organism are listed in http://www.genome.jp/kegg/catalog/org_list.html
keyType	one of "kegg", 'ncbi-geneid', 'ncbi-proteinid' or 'uniprot'
pvalueCutoff	pvalue cutoff
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
universe	background genes
minGSSize	minimal size of genes annotated by ontology term for testing.
maxGSSize	maximal size of genes annotated for testing
qvalueCutoff	qvalue cutoff
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

Value

A feaResult instance.

Examples

```
# Method supports duplicated elements in "gene", which should be entrez ids
gene <- c(rep("4312",4), rep("8318",2), "991", "10874")
#data(geneList, package="DOSE")
#kk <- enrichKEGG2(gene = gene, universe=names(geneList))
#head(kk)
```

enrichMOA

MOA Category Enrichment with Hypergeometric Test

Description

Given a vector of gene identifiers, this function returns MOA category enrichment results based on a hypergeometric test with duplication support in the test set. The universe for the test is set to the unique genes encoding the target proteins present in the MOA annotation system from the ChEMBL database.

Usage

```
enrichMOA(gene, pvalueCutoff = 0.05, pAdjustMethod = "BH", qvalueCutoff = 0.2)
```

Arguments

gene	a vector of entrez gene ids (here the test set)
pvalueCutoff	pvalue cutoff
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
qvalueCutoff	qvalue cutoff

Value

A feaResult instance.

See Also

feaResult-class

Examples

```
data(geneList, package="DOSE")
emoa <- enrichMOA(gene = names(geneList)[seq(3)])
head(emoa)</pre>
```

enrichReactome

Reactome Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enriched Reactome pathways with FDR control from hypergeometric test.

Description

Reactome Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enriched Reactome pathways with FDR control from hypergeometric test.

Usage

```
enrichReactome(
   gene,
   organism = "human",
   pvalueCutoff = 0.05,
   pAdjustMethod = "BH",
   qvalueCutoff = 0.2,
   universe,
   minGSSize = 5,
   maxGSSize = 500,
   readable = FALSE
)
```

Arguments

gene	a vector of entrez gene id.
organism	one of "human", "rat", "mouse", "celegans", "yeast", "zebrafish", "fly".
pvalueCutoff	Cutoff value of pvalue.
pAdjustMethod	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
qvalueCutoff	Cutoff value of qvalue
universe	background genes
minGSSize	minimal size of genes annotated by functional term for testing.
maxGSSize	maximal size of each gene set for analyzing
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

Value

A feaResult instance.

See Also

feaResult-class

Examples

```
# This method supports duplicated elements in "gene"
gene <- c(rep("4312",4), rep("8318",2), "991", "10874")
#data(geneList, package="DOSE")
#rc <- enrichReactome(gene=gene, universe=names(geneList))
#result(rc)</pre>
```

feaResult

Constructor for feaResult-class

Description

This is a helper function to construct a feaResult object. For detail description, please consult the help file of the feaResult-class.

Usage

```
feaResult(
  result,
  organism = "UNKNOWN",
  ontology = "UNKNOWN",
  drugs = "UNKNOWN",
  targets = "UNKNOWN"
)
```

Arguments

result	tibble object containing the FEA results
organism	character(1), organism information of the annotation system
ontology	character(1), ontology type of the GO annotation system. If the annotation system is KEGG, it will be 'KEGG'
drugs	character vector, input drug names used for the enrichment test
targets	character vector, gene labels of the gene/protein targets for the drugs

Value

feaResult object

Examples

feaResult-class *feaResult object*

Description

The feaResult object stores Functional Enrichment Analysis (FEA) results generated by the corresponding Target and Drug Set Enrichment methods (here TSEA and DSEA) defined by signatureSearch. This includes slots for the FEA results in tabular format, the organism information, and the type of functional annotation used (e.g. GO or KEGG). It also includes the drug information used for the FEA, as well as the corresponding target protein information.

Slots

- result tibble object, this tabular result contains the enriched functional categories (e.g. GO terms or KEGG pathways) ranked by the corresponding enrichment statistic. The result table can be extracted via the result accessor function.
- organism organism information of the annotation system. Currently, limited to 'human', since drug-target annotations are too sparse for other organisms.
- ontology ontology type of the GO annotation system. If the annotation system is KEGG, it will be 'KEGG'
- drugs input drug names used for the enrichment test
- targets target information for the query drugs obtained from the chosen drug-target annotation source.

GCT object

An S4 Class to Represent a GCT Object

Description

The GCT class serves to represent annotated matrices. The mat slot contains the numeric matrix data and the rdesc and cdesc slots contain data frames with annotations about the rows and columns, respectively

Slots

- mat a numeric matrix
- rid a character vector of row ids
- cid a character vector of column ids
- rdesc a data.frame of row descriptors
- ${\tt rdesc}$ a data.frame of column descriptors
- src a character indicating the source (usually file path) of the data

See Also

parse_gctx

gctx2h5

Description

Read matrix-like data from large gctx file in chunks and write result back to an HDF5 file.

Usage

gctx2h5(gctx, cid, new_cid = cid, h5file, by_ncol = 5000, overwrite = TRUE)

Arguments

gctx	character(1), path to gctx file from LINCS
cid	character or integer vector referencing the columns of the matrix to include
new_cid	character vector of the same length as cid, assigning new column names to matrix
h5file	character(1), path of the hdf5 destination file
by_ncol	number of columns to import in each iteration to limit memory usage
overwrite	TRUE or FALSE, whether to overwrite or to append to existing 'h5file'

Value

HDF5 file

Examples

gessResult	Constructor for gessResult-class
------------	----------------------------------

Description

This is a helper function to construct a gessResult object. For detail description, please consult the help file of the gessResult-class.

Usage

```
gessResult(result, query, gess_method, refdb)
```

gessResult-class

Arguments

result	tibble object containing the GESS results
query	list or a matrix, query signature
gess_method	character(1), name of the GESS method
refdb	character(1), path to the reference database

Value

gessResult object

Examples

gessResult-class gessResult object

Description

The gessResult object organizes Gene Expression Signature Search (GESS) results. This includes the main tabular result of a GESS, its query signature, the name of the chosen GESS method and the path to the reference database.

Slots

- result tibble object containing the search results for each perturbagen (e.g. drugs) in the reference database ranked by their signature similarity to the query. The result table can be extracted via the result accessor function.
- query query signature
- gess_method name of the GESS method
- refdb path to the reference database

gess_cmap

GESS Methods

Description

The CMAP search method implements the original Gene Expression Signature Search (GESS) from Lamb et al (2006) known as Connectivity Map (CMap). The method uses as query the two label sets of the most up- and down-regulated genes from a genome-wide expression experiment, while the reference database is composed of rank transformed expression profiles (e.g. ranks of LFC or z-scores).

Correlation-based similarity metrics, such as Spearman or Pearson coefficients, can be used as Gene Expression Signature Search (GESS) methods. As non-set-based methods, they require quantitative gene expression values for both the query and the database entries, such as normalized intensities or read counts from microarrays or RNA-Seq experiments, respectively.

In its iterative form, Fisher's exact test (Upton, 1992) can be used as Gene Expression Signature (GES) Search to scan GES databases for entries that are similar to a query GES.

The gCMAP search method adapts the Gene Expression Signature Search (GESS) method from the gCMAP package (Sandmann et al. 2014) to make it compatible with the database containers and methods defined by signatureSearch. The specific GESS method, called gCMAP, uses as query a rank transformed GES and the reference database is composed of the labels of up and down regulated DEG sets.

LINCS search method implements the Gene Expression Signature Search (GESS) from Subramanian et al, 2017, here referred to as LINCS. The method uses as query the two label sets of the most up- and down-regulated genes from a genome-wide expression experiment, while the reference database is composed of differential gene expression values (e.g. LFC or z-scores). Note, the related CMAP method uses here ranks instead.

Usage

```
gess_cmap(
  qSig,
  chunk_size = 5000,
  ref_trts = NULL,
  workers = 1,
  cmp_annot_tb = NULL,
  by = "pert",
  cmp_name_col = "pert",
  addAnnotations = TRUE
)
gess_cor(
  qSig,
  method = "spearman",
  chunk_size = 5000,
  ref_trts = NULL,
  workers = 1,
  cmp_annot_tb = NULL,
  by = "pert",
  cmp_name_col = "pert",
  addAnnotations = TRUE
)
gess_fisher(
  qSig,
  higher = NULL,
```

gess_cmap

```
lower = NULL,
  padj = NULL,
  chunk_size = 5000,
  ref_trts = NULL,
  workers = 1,
  cmp_annot_tb = NULL,
  by = "pert",
  cmp_name_col = "pert",
  addAnnotations = TRUE
)
gess_gcmap(
  qSig,
  higher = NULL,
  lower = NULL,
  padj = NULL,
  chunk_size = 5000,
  ref_trts = NULL,
  workers = 1,
  cmp_annot_tb = NULL,
  by = "pert",
  cmp_name_col = "pert",
  addAnnotations = TRUE
)
gess_lincs(
  qSig,
  tau = FALSE,
  sortby = "NCS",
  chunk_size = 5000,
  ref_trts = NULL,
  workers = 1,
  cmp_annot_tb = NULL,
  by = "pert",
  cmp_name_col = "pert",
  GeneType = "reference",
  addAnnotations = TRUE
)
```

Arguments

qSig	qSig object defining the query signature including the GESS method (should be 'LINCS') and the path to the reference database. For details see help of qSig and qSig-class.
chunk_size	number of database entries to process per iteration to limit memory usage of search.
ref_trts	character vector. If users want to search against a subset of the reference database, they could set ref_trts as a character vector representing column names (treatments) of the subsetted refdb.
workers	integer(1) number of workers for searching the reference database parallelly, default is 1.

cmp_annot_tb	data.frame or tibble of compound annotation table. This table contains anno- tation information for compounds stored under pert column of gess_tb. Set to NULL if not available. This table should not contain columns with names of "t_gn_sym", "MOAss" or "PCIDss", these three columns will be added inter- nally and thus conserved by the function. If they are contained in cmp_annot_tb, they will be overwritten. If users want to maintain these three columns in the provided annotation table, give them different names.
by	character(1), column name in cmp_annot_tb that can be merged with pert column in gess_tb. If refdb is set as 'lincs2', it will be merged with pert_id column in the GESS result table. If cmp_annot_tb is NULL, by is ignored.
cmp_name_col	character(1), column name in gess_tb or cmp_annot_tb that store compound names. If there is no compound name column, set to NULL. If cmp_name_col is available, three additional columns (t_gn_sym, MOAss, PCIDss) are automatically added by using get_targets, CLUE touchstone compound MOA annotation, and 2017 lincs_pert_info annotation table, respectively as annotation sources. t_gn_sym: target gene symbol, MOAss: MOA annotated from signatureSearch, PCIDss: PubChem CID annotated from signatureSearch.
addAnnotations	Logical value. If TRUE adds drug annotations to results.
method	One of 'spearman' (default), 'kendall', or 'pearson', indicating which correla- tion coefficient to use.
higher	The 'upper' threshold. If not 'NULL', genes with a score larger than or equal to 'higher' will be included in the gene set with sign +1. At least one of 'lower' and 'higher' must be specified.
	higher argument need to be set as 1 if the refdb in qSig is path to the HDF5 file that were converted from the gmt file.
lower	The lower threshold. If not 'NULL', genes with a score smaller than or equal 'lower' will be included in the gene set with sign -1. At least one of 'lower' and 'higher' must be specified.
	lower argument need to be set as NULL if the refdb in qSig is path to the HDF5 file that were converted from the gmt file.
padj	numeric(1), cutoff of adjusted p-value or false discovery rate (FDR) of defining DEGs that is less than or equal to 'padj'. The 'padj' argument is valid only if the reference HDF5 file contains the p-value matrix stored in the dataset named as 'padj'.
tau	TRUE or FALSE, whether to compute the tau score. Note, TRUE is only mean- ingful when the full LINCS database is searched, since accurate Tau score calcu- lation depends on the usage of the exact same database their background values are based on.
sortby	sort the GESS result table based on one of the following statistics: 'WTCS', 'NCS', 'Tau', 'NCSct' or 'NA'
GeneType	A character value of either "reference" or a combination of "best inferred", "landmark" or "inferred" indicating which reference gene set query genes should be filtered against. While "reference" filters query genes against the reference database, "best inferred", "landmark" or "inferred" filter genes against LINCS gene spaces.

Details

Lamb et al. (2006) introduced the gene expression-based search method known as Connectivity Map (CMap) where a GES database is searched with a query GES for similar entries. Specifically,

gess_cmap

this GESS method uses as query the two label sets of the most up- and down-regulated genes from a genome-wide expression experiment, while the reference database is composed of rank transformed expression profiles (e.g.ranks of LFC or z-scores). The actual GESS algorithm is based on a vectorized rank difference calculation. The resulting Connectivity Score expresses to what degree the query up/down gene sets are enriched on the top and bottom of the database entries, respectively. The search results are a list of perturbagens such as drugs that induce similar or opposing GESs as the query. Similar GESs suggest similar physiological effects of the corresponding perturbagens. Although several variants of the CMAP algorithm are available in other software packages including Bioconductor, the implementation provided by signatureSearch follows the original description of the authors as closely as possible.

For correlation searches to work, it is important that both the query and reference database contain the same type of gene identifiers. The expected data structure of the query is a matrix with a single numeric column and the gene labels (e.g. Entrez Gene IDs) in the row name slot. For convenience, the correlation-based searches can either be performed with the full set of genes represented in the database or a subset of them. The latter can be useful to focus the computation for the correlation values on certain genes of interest such as a DEG set or the genes in a pathway of interest. For comparing the performance of different GESS methods, it can also be advantageous to subset the genes used for a correlation-based search to same set used in a set-based search, such as the up/down DEGs used in a LINCS GESS. This way the search results of correlation- and set-based methods can be more comparable because both are provided with equivalent information content.

When using the Fisher's exact test (Upton, 1992) as GES Search (GESS) method, both the query and the database are composed of gene label sets, such as DEG sets.

The Bioconductor gCMAP (Sandmann et al. 2014) package provides access to a related but not identical implementation of the original CMAP algorithm proposed by Lamb et al. (2006). It uses as query a rank transformed GES and the reference database is composed of the labels of up and down regulated DEG sets. This is the opposite situation of the original CMAP method from Lamb et al (2006), where the query is composed of the labels of up and down regulated DEGs and the database contains rank transformed GESs.

Subramanian et al. (2017) introduced a more complex GESS algorithm, here referred to as LINCS. While related to CMAP, there are several important differences among the two approaches. First, LINCS weights the query genes based on the corresponding differential expression scores of the GESs in the reference database (e.g. LFC or z-scores). Thus, the reference database used by LINCS needs to store the actual score values rather than their ranks. Another relevant difference is that the LINCS algorithm uses a bi-directional weighted Kolmogorov-Smirnov enrichment statistic (ES) as similarity metric.

Value

gessResult object, the result table contains the search results for each perturbagen in the reference database ranked by their signature similarity to the query.

Column description

Descriptions of the columns in GESS result tables.

- pert: character, perturbagen (e.g. drugs) in the reference database. The treatment/column names of the reference database are organized as pert_cell_trt_cp format. The pert column in GESS result table contains what stored under the pert slot of the column names.
- cell: character, acronym of cell type
- type: character, perturbation type. In the CMAP/LINCS databases provided by signatureSearchData, the perturbation types are currently treatments with drug-like compounds (trt_cp). If required,

users can build custom signature database with other types of perturbagens (e.g., gene knockdown or over-expression events) with the provided build_custom_db function.

- trend: character, up or down when the reference signature is positively or negatively connected with the query signature, respectively.
- N_upset: integer, number of genes in the query up set
- N_downset: integer, number of genes in the query down set
- WTCS: Weighted Connectivity Score, a bi-directional Enrichment Score for an up/down query set. If the ES values of an up set and a down set are of different signs, then WTCS is (ESup-ESdown)/2, otherwise, it is 0. WTCS values range from -1 to 1. They are positive or negative for signatures that are positively or inversely related, respectively, and close to zero for signatures that are unrelated.
- WTCS_Pval: Nominal p-value of WTCS computed by comparing WTCS against a null distribution of WTCS values obtained from a large number of random queries (e.g. 1000).
- WTCS_FDR: False discovery rate of WTCS_Pval.
- NCS: Normalized Connectivity Score. To make connectivity scores comparable across cell types and perturbation types, the scores are normalized. Given a vector of WTCS values w resulting from a query, the values are normalized within each cell line c and perturbagen type t to obtain NCS by dividing the WTCS value with the signed mean of the WTCS values within the subset of the signatures in the reference database corresponding to c and t.
- Tau: Enrichment score standardized for a given database. The Tau score compares an observed NCS to a large set of NCS values that have been pre-computed for a specific reference database. The query results are scored with Tau as a standardized measure ranging from 100 to -100. A Tau of 90 indicates that only 10 stronger connectivity to the query. This way one can make more meaningful comparisons across query results.

Note, there are NAs in the Tau score column, the reason is that the number of signatures in *Qref* that match the cell line of signature r (the TauRefSize column in the GESS result) is less than 500, Tau will be set as NA since it is redeemed as there are not large enough samples for computing meaningful Tau scores.

- TauRefSize: Size of reference perturbations for computing Tau.
- NCSct: NCS summarized across cell types. Given a vector of NCS values for perturbagen p, relative to query q, across all cell lines c in which p was profiled, a cell-summarized connectivity score is obtained using a maximum quantile statistic. It compares the 67 and 33 quantiles of NCSp,c and retains whichever is of higher absolute magnitude.
- cor_score: Correlation coefficient based on the method defined in the gess_cor function.
- raw_score: bi-directional enrichment score (Kolmogorov-Smirnov statistic) of up and down set in the query signature
- scaled_score: raw_score scaled to values from 1 to -1 by dividing the positive and negative scores with the maximum positive score and the absolute value of the minimum negative score, respectively.
- effect: Scaled bi-directional enrichment score corresponding to the scaled_score under the CMAP result.
- nSet: number of genes in the GES in the reference database (gene sets) after setting the higher and lower cutoff.
- nFound: number of genes in the GESs of the reference database (gene sets) that are also present in the query GES.
- signed: whether gene sets in the reference database have signs, representing up and down regulated genes when computing scores.

- pval: p-value of the Fisher's exact test.
- padj: p-value adjusted for multiple hypothesis testing using R's p.adjust function with the Benjamini & Hochberg (BH) method.
- effect: z-score based on the standard normal distribution.
- LOR: Log Odds Ratio.
- t_gn_sym: character, symbol of the gene encoding the corresponding drug target protein
- MOAss: character, compound MOA annotation from signatureSearch package
- PCIDss: character, compound PubChem CID annotation from signatureSearch package

References

For detailed description of the LINCS method and scores, please refer to: Subramanian, A., Narayan, R., Corsello, S. M., Peck, D. D., Natoli, T. E., Lu, X., Golub, T. R. (2017). A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell, 171 (6), 1437-1452.e17. URL: https://doi.org/10.1016/j.cell.2017.10.049

For detailed description of the CMap method, please refer to: Lamb, J., Crawford, E. D., Peck, D., Modell, J. W., Blat, I. C., Wrobel, M. J., Golub, T. R. (2006). The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. Science, 313 (5795), 1929-1935. URL: https://doi.org/10.1126/science.1132939

Sandmann, T., Kummerfeld, S. K., Gentleman, R., & Bourgon, R. (2014). gCMAP: user-friendly connectivity mapping with R. Bioinformatics , 30 (1), 127-128. URL: https://doi.org/10.1093/bioinformatics/btt592

Graham J. G. Upton. 1992. Fisher's Exact Test. J. R. Stat. Soc. Ser. A Stat. Soc. 155 (3). [Wiley, Royal Statistical Society]: 395-402. URL: http://www.jstor.org/stable/2982890

See Also

qSig, gessResult, addGESSannot

```
db_path <- system.file("extdata", "sample_db.h5",</pre>
                      package = "signatureSearch")
# library(SummarizedExperiment); library(HDF5Array)
# sample_db <- SummarizedExperiment(HDF5Array(db_path, name="assay"))</pre>
# rownames(sample_db) <- HDF5Array(db_path, name="rownames")</pre>
# colnames(sample_db) <- HDF5Array(db_path, name="colnames")</pre>
## get "vorinostat__SKB__trt_cp" signature drawn from sample database
# query_mat <- as.matrix(assay(sample_db[,"vorinostat__SKB__trt_cp"]))</pre>
# qsig_cmap <- qSig(query=list(</pre>
                     upset=c("230", "5357", "2015", "2542", "1759"),
#
                     downset=c("22864", "9338", "54793", "10384", "27000")),
#
                   gess_method="CMAP", refdb=db_path)
#
# cmap <- gess_cmap(qSig=qsig_cmap, chunk_size=5000)</pre>
# result(cmap)
# qsig_sp <- qSig(query=query_mat, gess_method="Cor", refdb=db_path)</pre>
# sp <- gess_cor(qSig=qsig_sp, method="spearman")</pre>
# result(sp)
```

```
# qsig_fisher <- qSig(query=query_mat, gess_method="Fisher", refdb=db_path)</pre>
# fisher <- gess_fisher(qSig=qsig_fisher, higher=1, lower=-1)</pre>
# result(fisher)
# qsig_gcmap <- qSig(query=query_mat, gess_method="gCMAP", refdb=db_path)</pre>
# gcmap <- gess_gcmap(qsig_gcmap, higher=1, lower=-1)</pre>
# result(gcmap)
# qsig_lincs <- qSig(query=list(</pre>
                    upset=c("230", "5357", "2015", "2542", "1759"),
#
                    downset=c("22864", "9338", "54793", "10384", "27000")),
#
#
                  gess_method="LINCS", refdb=db_path)
# lincs <- gess_lincs(qsig_lincs, sortby="NCS", tau=FALSE)</pre>
# result(lincs)
```

gess_res_vis

GESS Result Visualization

Description

The function allows to summarize the ranking scores of selected perturbagens for GESS results across cell types along with cell type classifications, such as normal and tumor cells. In the resulting plot the perturbagens are drugs (along x-axis) and the ranking scores are LINCS' NCS values (y-axis). For each drug the NCS values are plotted for each cell type as differently colored dots, while their shape indicates the cell type class.

Usage

```
gess_res_vis(gess_tb, drugs, col, cell_group = "all", ...)
```

Arguments

gess_tb	tibble in the 'result' slot of the gessResult object, can be extracted via result accessor function
drugs	character vector of selected drugs
col	character(1), name of the score column in 'gess_tb', e.g., "NCS" if the result table is from LINCS method. Can also be set as "rank", this way it will show the ranks of each drug in different cell types.
cell_group	character(1), one of "all", "normal", or "tumor". If "all", it will show scores of each drug in both tumor and normal cell types. If "normal" or "tumor", it will only show normal or tumor cell types.
	Other arguments passed on to geom_point

Value

plot visualizing GESS results

38

getALLEG

References

Subramanian, A., Narayan, R., Corsello, S. M., Peck, D. D., Natoli, T. E., Lu, X., Golub, T. R. (2017). A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell, 171 (6), 1437-1452.e17. URL: https://doi.org/10.1016/j.cell.2017.10.049

Examples

getALLEG

getALLEG

Description

get all entrez gene ID of a specific organism

Usage

getALLEG(organism)

Arguments

organism one of "human", "rat", "mouse", "celegans", "yeast", "zebrafish", "fly".

Value

entrez gene ID vector

Author(s)

Yu Guangchuang

getDb

getDb

Description

mapping organism name to annotationDb package name

Usage

getDb(organism)

40

Arguments

organism one of supported organism

Value

annotationDb name

Author(s)

Yu Guangchuang

getSig

Draw GESs from Reference Database

Description

Functionalities used to draw from reference database (e.g. lincs, lincs_expr) GESs of compound treatment(s) in cell types.

Usage

```
getSig(cmp, cell, refdb)
getDEGSig(
    cmp,
    cell,
    Nup = NULL,
    Ndown = NULL,
    higher = NULL,
    lower = NULL,
    padj = NULL,
    refdb = "lincs"
)
```

getSPsubSig(cmp, cell, Nup = 150, Ndown = 150)

Arguments

cmp	character vector representing a list of compound name available in refdb for getSig function, or character(1) indicating a compound name (e.g. vorinostat) for other functions
cell	character(1) or character vector of the same length as cmp argument. It indicates cell type that the compound treated in
refdb	character(1), one of "lincs", "lincs_expr", "cmap", "cmap_expr", or path to the HDF5 file built from build_custom_db function
Nup	integer(1). Number of most up-regulated genes to be subsetted
Ndown	integer(1). Number of most down-regulated genes to be subsetted
higher	numeric(1), the upper threshold of defining DEGs. At least one of 'lower' and 'higher' must be specified. If Nup or Ndown arguments are defined, it will be ignored.

getTreats

lower	numeric(1), the lower threshold of defining DEGs. At least one of 'lower' and 'higher' must be specified. If Nup or Ndown arguments are defined, it will be ignored.
padj	numeric(1), cutoff of adjusted p-value or false discovery rate (FDR) of defin- ing DEGs if the reference HDF5 database contains the p-value matrix stored in the dataset named as 'padj'. If Nup or Ndown arguments are defined, it will be ignored.

Details

The GES could be genome-wide differential expression profiles (e.g. log2 fold changes or zscores) or normalized gene expression intensity values depending on the data type of refdb or n top up/down regulated DEGs

Value

matrix representing genome-wide GES of the query compound(s) in cell

a list of up- and down-regulated gene label sets

a numeric matrix with one column representing gene expression values drawn from lincs_expr db of the most up- and down-regulated genes. The genes were subsetted according to z-scores drawn from lincs db.

Examples

getTreats

Get Treatment Information

Description

Get treatment information including perturbation name, cell type and perturbation type from the reference database

Usage

getTreats(refdb, sep = TRUE)

Arguments

refdb	character(1), one of "lincs", "lincs_expr", "cmap" or "cmap_expr" when using the pre-generated CMAP/LINCS databases or path to the HDF5 file generated
	with the build_custom_db function. The details is shown in the 'refdb' argument of the qSig function
sep	TRUE or FALSE, whether to separate the treatments or column names of the reference database into 'pert', 'cell' and 'pert_type'.

Value

character vector if sep argument is set as FALSE. Tibble object with 'pert', 'cell', 'pert_type' columns if sep is TRUE

Examples

```
refdb <- system.file("extdata", "sample_db.h5", package="signatureSearch")
treat_info <- getTreats(refdb, sep=TRUE)</pre>
```

get_targets

Target Gene/Protein IDs for Query Drugs

Description

This function returns for a set of query drug names/ids the corresponding target gene/protein ids. The required drug-target annotations are from DrugBank, CLUE and STITCH. An SQLite database storing these drug-target interactions based on the above three annotation resources is available in the signatureSearchData package.

Usage

```
get_targets(drugs, database = "all", verbose = TRUE, output = "df")
```

Arguments

drugs	character vector of drug names
database	drug-target annotation resource; A character vector of any combination of 'Drug- Bank', 'CLUE', STITCH' or 'all'. The target set from the selected resources will be combined. If 'all' is contained in the character vector, target sets from all of the annotation databases (DrugBank, CLUE and STITCH) will be com- bined.
verbose	TRUE or FALSE, whether to print messages
output	one of "df", "list" or "vector". If setting as "df", the result is in a data.frame format containing target gene symbols separated by semicolon for each drug. If setting as "list", the result is a list of targets for each query drug. If setting as "vector", the result is a character vector of the target set that are collapsed with duplications if different drugs have the same targets.

Value

drug-target annotation in a format defined by the output argument.

gmt2h5

See Also

dtlink_db_clue_sti

Examples

data(drugs10)
dt <- get_targets(drugs10)</pre>

gmt2h5

Convert GMT to HDF5 File

Description

Read gene sets from large gmt file in batches, convert the gene sets to 01 matrix and write the result to an HDF5 file.

Usage

```
gmt2h5(gmtfile, dest_h5, by_nset = 5000, overwrite = FALSE)
```

Arguments

gmtfile	character(1), path to gmt file containing gene sets
dest_h5	character(1), path of the hdf5 destination file
by_nset	number of gene sets to import in each iteration to limit memory usage
overwrite	TRUE or FALSE, whether to overwrite or to append to existing 'h5file'

Value

HDF5 file

gseGO2

Description

This modified Gene Set Enrichment Analysis (GSEA) of GO terms supports gene test sets with large numbers of zeros.

Usage

```
gseG02(
  geneList,
  ont = "BP",
  OrgDb,
  keyType = "SYMBOL",
  exponent = 1,
  nproc = 1,
  nPerm = 1000,
  minGSSize = 2,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE
)
```

Arguments

geneList	named numeric vector with gene SYMBOLs in the name slot decreasingly ranked by scores in the data slot.
ont	one of "BP", "MF", "CC" or "ALL"
OrgDb	OrgDb, e.g., "org.Hs.eg.db".
keyType	keytype of gene
exponent	weight of each step
nproc	if not equal to zero, sets BPPARAM to use nproc workers (default = 1)
nPerm	permutation numbers
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
pvalueCutoff	pvalue cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not

Value

feaResult object

gseKEGG2

Examples

```
data(targetList)
# gsego <- gseGO2(geneList=targetList, ont="MF", OrgDb="org.Hs.eg.db",
# pvalueCutoff=1)
# head(gsego)</pre>
```

```
gseKEGG2
```

Modified GSEA with KEGG

Description

This modified Gene Set Enrichment Analysis (GSEA) of KEGG pathways supports gene test sets with large numbers of zeros.

Usage

```
gseKEGG2(
  geneList,
  organism = "hsa",
  keyType = "kegg",
  exponent = 1,
  nproc = 1,
  nPerm = 1000,
  minGSSize = 10,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
  readable = FALSE
)
```

Arguments

geneList	named numeric vector with gene ids in the name slot decreasingly ranked by scores in the data slot.
organism	supported organism listed in URL: http://www.genome.jp/kegg/catalog/org_list.html
keyType	one of "kegg", 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
exponent	weight of each step
nproc	if not equal to zero, sets BPPARAM to use nproc workers (default = 1)
nPerm	permutation numbers
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
pvalueCutoff	pvalue cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

Value

feaResult object

Examples

```
# Gene Entrez id should be used for KEGG enrichment
data(geneList, package="DOSE")
#geneList[100:length(geneList)]=0
#gsekk <- gseKEGG2(geneList=geneList, pvalueCutoff = 1)
#head(gsekk)
```

gseReactome

Modified GSEA with Reactome

Description

This modified Gene Set Enrichment Analysis (GSEA) of Reactome pathways supports gene test sets with large numbers of zeros.

Usage

```
gseReactome(
  geneList,
  organism = "human",
  exponent = 1,
  nPerm = 1000,
  minGSSize = 10,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
  readable = FALSE
)
```

Arguments

geneList	order ranked geneList
organism	one of "human", "rat", "mouse", "celegans", "yeast", "zebrafish", "fly".
exponent	integer value used as exponent in GSEA algorithm.
nPerm	integer defining the number of permutation iterations for calculating p-values
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of each geneSet for analyzing
pvalueCutoff	pvalue Cutoff
pAdjustMethod	pvalue adjustment method
verbose	print message or not TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

46

head

Value

feaResult object

Examples

```
# Gene Entrez id should be used for Reactome enrichment
data(geneList, package="DOSE")
#geneList[100:length(geneList)]=0
#rc <- gseReactome(geneList=geneList, pvalueCutoff=1)</pre>
```

head

Return the First Part of an Object

Description

Return the first part of the result table in the gessResult, and feaResult objects

Usage

```
## S4 method for signature 'gessResult'
head(x, n = 6L, \ldots)
```

```
## S4 method for signature 'feaResult'
head(x, n = 6L, \ldots)
```

Arguments

х	an object
n	a single integer. If positive or zero, size for the resulting object is the number of rows for a data frame. If negative, all but the n last number of rows of x.
	arguments to be passed to or from other methods

Value

data.frame

```
gr <- gessResult(result=dplyr::tibble(pert=letters[seq_len(10)],</pre>
                                             val=seq_len(10)),
                    query=list(up=c("g1","g2"), down=c("g3","g4")),
gess_method="LINCS", refdb="path/to/lincs/db")
head(gr)
fr <- feaResult(result=dplyr::tibble(id=letters[seq_len(10)],</pre>
                                           val=seq_len(10)),
                   organism="human", ontology="MF", drugs=c("d1", "d2"),
                   targets=c("t1","t2"))
```

lincs_expr_inst_info Instance Information of LINCS Expression Database

Description

It is a tibble of 3 columns containing compound treatment information of GEP instances in the LINCS expression database. The columns contain the compound name, cell type and perturbation type (all of them are compound treatment, trt_cp).

Usage

lincs_expr_inst_info

Format

A tibble object with 38,824 rows and 3 columns.

Examples

```
# Load object
data(lincs_expr_inst_info)
head(lincs_expr_inst_info)
```

lincs_pert_info LINCS 2017 Perturbation Information

Description

It is a tibble containing annotation information of compounds in LINCS 2017 database including perturbation name, type, whether in touchstone database, INCHI key, canonical smiles, PubChem CID as well as annotations from ChEMBL database, including ChEMBL ID, DrugBank ID, max FDA phase, therapeutic flag, first approval, indication class, mechanism of action, disease efficacy et al.

Usage

lincs_pert_info

Format

A tibble object with 8,140 rows and 40 columns.

```
# Load object
data(lincs_pert_info)
lincs_pert_info
```

lincs_pert_info2 LINCS 2020 Perturbation Information

Description

It is a tibble containing annotation information of compounds in LINCS 2020 beta database including perturbation id, perturbation name, canonical smiles, Inchi key, compound aliases, target and MOA. The PubChem CID and many other annotations from ChEMBL database were obtained from 2017 LINCS pert info by by left joining with pert_iname.

Usage

lincs_pert_info2

Format

A tibble object with 34419 rows and 48 columns.

Examples

```
# Load object
data(lincs_pert_info2)
lincs_pert_info2
```

lincs_sig_info LINCS Signature Information

Description

It is a tibble of 3 columns containing treatment information of GESs in the LINCS database. The columns contain the perturbation name, cell type and perturbation type (all of them are compound treatment, trt_cp).

Usage

lincs_sig_info

Format

A tibble object with 45,956 rows and 3 columns.

```
# Load object
data(lincs_sig_info)
head(lincs_sig_info)
```

list2df

Description

Convert a list with names that have one to many mapping relationships to a data.frame of two columns, one column is names, the other column is the unlist elements

Usage

```
list2df(list, colnames)
```

Arguments

list	input list with names slot
colnames	character vector of length 2, indicating the column names of the returned data.frame

Value

data.frame

Examples

```
list <- list("n1"=c("e1", "e2", "e4"), "n2"=c("e3", "e5"))
list2df(list, colnames=c("name", "element"))</pre>
```

list_rev Reverse list

Description

Reverse list from list names to elements mapping to elements to names mapping.

Usage

list_rev(list)

Arguments

list input list with names slot

Value

list

```
list <- list("n1"=c("e1", "e2", "e4"), "n2"=c("e1", "e5"))
list_rev(list)</pre>
```

mabsG0

Description

MeanAbs enrichment analysis with GO terms.

Usage

```
mabsGO(
  geneList,
  ont = "BP",
  OrgDb,
  keyType = "SYMBOL",
  nPerm = 1000,
  minGSSize = 5,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH"
)
```

Arguments

geneList	named numeric vector with gene SYMBOLs in the name slot decreasingly ranked by scores in the data slot.
ont	one of "BP", "MF", "CC" or "ALL"
OrgDb	OrgDb
keyType	keytype of gene
nPerm	permutation numbers
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
pvalueCutoff	pvalue cutoff
pAdjustMethod	pvalue adjustment method

Value

feaResult object

Author(s)

Yuzhu Duan

```
data(targetList)
#mg <- mabsGO(geneList=targetList, ont="MF", OrgDb="org.Hs.eg.db",
# pvalueCutoff=1)
#head(mg)</pre>
```

mabsKEGG

Description

MeanAbs enrichment analysis with KEGG pathways.

Usage

```
mabsKEGG(
  geneList,
  organism = "hsa",
  keyType = "kegg",
  nPerm = 1000,
  minGSSize = 5,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  readable = FALSE
)
```

Arguments

geneList	named numeric vector with gene/target ids in the name slot decreasingly ranked by scores in the data slot.
organism	supported organism listed in URL: http://www.genome.jp/kegg/catalog/org_list.html
keyType	one of 'kegg', 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
nPerm	permutation numbers
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
pvalueCutoff	pvalue cutoff
pAdjustMethod	pvalue adjustment method
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

Value

feaResult object

```
# Gene Entrez id should be used for KEGG enrichment
data(geneList, package="DOSE")
#geneList[100:length(geneList)]=0
#mk <- mabsKEGG(geneList=geneList, pvalueCutoff = 1)
#head(mk)
```

mabsReactome

Description

MeanAbs enrichment analysis with Reactome pathways.

Usage

```
mabsReactome(
  geneList,
  organism = "human",
  nPerm = 1000,
  minGSSize = 5,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  readable = FALSE
)
```

Arguments

geneList	named numeric vector with gene/target ids in the name slot decreasingly ranked by scores in the data slot.
organism	one of "human", "rat", "mouse", "celegans", "yeast", "zebrafish", "fly".
nPerm	permutation numbers
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
pvalueCutoff	pvalue cutoff
pAdjustMethod	pvalue adjustment method
readable	TRUE or FALSE indicating whether to convert gene Entrez ids to gene Symbols in the 'itemID' column in the FEA result table.

Value

feaResult object

```
# Gene Entrez id should be used for Reactome enrichment
data(geneList, package="DOSE")
#geneList[100:length(geneList)]=0
#rc <- mabsReactome(geneList=geneList, pvalueCutoff = 1)</pre>
```

matrix2h5

Description

Function writes matrix object to an HDF5 file.

Usage

matrix2h5(matrix, h5file, name = "assay", overwrite = TRUE)

Arguments

matrix	matrix to be written to HDF5 file, row and column name slots need to be populated
h5file	character(1), path to the hdf5 destination file
name	The name of the dataset in the HDF5 file. The default is write the score matrix (e.g. z-score, logFC) to the 'assay' dataset, users could also write the adjusted p-value or FDR matrix to the 'padj' dataset by setting the name as 'padj'.
overwrite	TRUE or FALSE, whether to overwrite or append matrix to an existing 'h5file'

Value

HDF5 file containing exported matrix

Examples

meanExpr2h5

Calculate Mean Expression Values of LINCS Level 3 Data

Description

Function calculates mean expression values for replicated samples of LINCS Level 3 data that have been treated by the same compound in the same cell type at a chosen concentration and treatment time. Usually, the function is used after filtering the Level 3 data with inst_filter. The results (here matrix with mean expression values) are saved to an HDF5 file. The latter is referred to as the 'lincs_expr' database.

Usage

```
meanExpr2h5(gctx, inst, h5file, chunksize = 2000, overwrite = TRUE)
```

moa_conn

Arguments

gctx	character(1), path to the LINCS Level 3 gctx file
inst	tibble, LINCS Level 3 instances after filtering for specific concentrations and times
h5file	character(1), path to the destination HDF5 file
chunksize	number of columns of the matrix to be processed at a time to limit memory usage
overwrite	TRUE or FALSE, whether to overwrite or append data to an existing 'h5file'

Value

HDF5 file, representing the lincs_expr database

Examples

moa_conn

Summarize GESS Results on MOA Level

Description

Function summarizes GESS results on Mode of Action (MOA) level. It returns a tabular representation of MOA categories ranked by their average signature search similarity to a query signature.

Usage

```
moa_conn(gess_tb, moa_cats = "default", cells = "normal")
```

Arguments

gess_tb	tibble in gessResult object
moa_cats	if set as "default", it uses MOA annotations from the CLUE website (https://clue.io). Users can customize it by providing a 'list' of character vectors containing drug names and MOA categories as list component names.
cells	one of "normal", "cancer" or "all", or a character vector containing cell types of interest.
	• "all": all cell types in LINCS database;
	 "normal": normal cell types in LINCS database as one group;
	• "tumor": tumor cell types in LINCS database as one group:

• "tumor": tumor cell types in LINCS database as one group;

Details

Column description of the result table:

moa: Mechanism of Action (MOA)

cells: cell type information

mean_rank: mean rank of drugs in corresponding GESS result for each MOA category

n_drug: number of drugs in each MOA category

Value

data.frame

See Also

gessResult

Examples

parse_gctx

```
Parse GCTX
```

Description

Parse a GCTX file into the R workspace as a GCT object

Usage

```
parse_gctx(
   fname,
   rid = NULL,
   cid = NULL,
   set_annot_rownames = FALSE,
   matrix_only = FALSE
)
```

Arguments

fname	character(1), path to the GCTX file on disk	
rid	either a vector of character or integer row indices or a path to a grp file containing character row indices. Only these indices will be parsed from the file.	
cid	either a vector of character or integer column indices or a path to a grp file containing character column indices. Only these indices will be parsed from the file.	
set_annot_rownames		
matrix_only	boolean indicating whether to set the rownames on the row/column metadata data.frames. Set this to false if the GCTX file has duplicate row/column ids. boolean indicating whether to parse only the matrix (ignoring row and column annotations)	

56

qSig

Value

gct object

Examples

qSig

Helper Function to Construct a qSig Object

Description

It builds a 'qSig' object to store the query signature, reference database and GESS method used for GESS methods.

Usage

qSig(query, gess_method, refdb)

Arguments

query	If 'gess_method' is 'CMAP' or 'LINCS', it should be a list with two charac- ter vectors named upset and downset for up- and down-regulated gene labels, respectively. The labels should be gene Entrez IDs if the reference database is a pre-built CMAP or LINCS database. If a custom database is used, the labels need to be of the same type as those in the reference database.
	If 'gess_method' is 'gCMAP', the query is a matrix with a single column repre- senting gene ranks from a biological state of interest. The corresponding gene labels are stored in the row name slot of the matrix. Instead of ranks one can provide scores (e.g. z-scores). In such a case, the scores will be internally trans- formed to ranks.
	If 'gess_method' is 'Fisher', the query is expected to be a list with two charac- ter vectors named upset and downset for up- and down-regulated gene labels, respectively (same as for 'CMAP' or 'LINCS' method). Internally, the up/down gene labels are combined into a single gene set when querying the reference database with the Fisher's exact test. This means the query is performed with an unsigned set. The query can also be a matrix with a single numeric column and the gene labels (e.g. Entrez gene IDs) in the row name slot. The values in this matrix can be z-scores or LFCs. In this case, the actual query gene set is obtained according to upper and lower cutoffs in the gess_fisher function set by the user.
	If 'gess_method' is 'Cor', the query is a matrix with a single numeric column and the gene labels in the row name slot. The numeric column can contain z-scores, LFCs, (normalized) gene expression intensity values or read counts.
gess_method	one of 'CMAP', 'LINCS', 'gCMAP', 'Fisher' or 'Cor'

refdb	character(1), can be one of "cmap", "cmap_expr", "lincs", "lincs_expr", "lincs2"
	when using the CMAP/LINCS databases from the affiliated signatureSearchData
	package. With 'cmap' the database contains signatures of LFC scores obtained
	from DEG analysis routines; with 'cmap_expr' normalized gene expression val-
	ues; with 'lincs' or 'lincs2' z-scores obtained from the DEG analysis methods
	of the LINCS project; and with 'lincs_expr' normalized expression values.
	To use a custom database, it should be the file path to the HDF5 file gener-
	ated with the build_custom_db function, the HDF5 file needs to have the .h5 extension.
	When the gess_method is set as 'gCMAP' or 'Fisher', it could also be the file path to the HDF5 file converted from the gmt file containing gene sets by us- ing gmt2h5 function. For example, the gmt files could be from the MSigDB https://www.gsea-msigdb.org/gsea/msigdb/index.jsp or GSKB http: //ge-lab.org/#/data.

Value

qSig object

See Also

build_custom_db, signatureSearchData, gmt2h5, qSig-class

Class "qSig"

Examples

qSig-class

Description

S4 object named qSig containing query signature information for Gene Expression Signature (GES) searches. It contains slots for query signature, GESS method and path to the GES reference database.

Slots

query If 'gess_method' is one of 'CMAP' or 'LINCS', this should be a list with two character vectors named upset and downset for up- and down-regulated gene labels (here Entrez IDs), respectively.

If 'gess_method' is 'gCMAP', 'Fisher' or 'Cor', a single column matrix with gene expression values should be assigned. The corresponding gene labels are stored in the row name slot of

the matrix. The expected type of gene expression values is explained in the help files of the corresponding GESS methods.

- gess_method one of 'CMAP', 'LINCS', 'gCMAP', 'Fisher' or 'Cor'
- refdb character(1), can be "cmap", "cmap_expr", "lincs", "lincs_expr", or "lincs2" when using existing CMAP/LINCS databases.

If users want to use a custom database, it should be the file path to the HDF5 file generated with the build_custom_db function. Alternatively, source files of the CMAP/LINCS databases can be used as explained in the vignette of the signatureSearchData package.

db_path character(1), file path to the refdb

rand_query_ES Generate WTCS Null Distribution with Random Queries

Description

Function computes null distribution of Weighted Connectivity Scores (WTCS) used by the LINCS GESS method for computing nominal P-values.

Usage

rand_query_ES(h5file, N_queries = 1000, dest, write = TRUE)

Arguments

h5file	character(1), path to the HDF5 file representing the reference database
N_queries	number of random queries
dest	path to the output file (e.g. "ES_NULL.txt")
write	Logical value indicating if results should be written to dest.

Value

File with path assigned to dest

References

Subramanian, A., Narayan, R., Corsello, S. M., Peck, D. D., Natoli, T. E., Lu, X., Golub, T. R. (2017). A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. Cell, 171 (6), 1437-1452.e17. URL: https://doi.org/10.1016/j.cell.2017.10.049

See Also

gess_lincs

```
db_path = system.file("extdata", "sample_db.h5", package="signatureSearch")
rand <- rand_query_ES(h5file=db_path, N_queries=5, dest="ES_NULL.txt", write=FALSE)
unlink("ES_NULL.txt")</pre>
```

read_gmt

Description

This function reads in and parses information from the MSigDB's .gmt files. Pathway information will be returned as a list of gene sets.

Usage

read_gmt(file, start = 1, end = -1)

Arguments

file	The .gmt file to be read
start	integer(1), read the gmt file from start line
end	integer(1), read the gmt file to the end line, the default -1 means read to the end

Details

The .gmt format is a tab-delimited list of gene sets, where each line is a separate gene set. The first column must specify the name of the gene set, and the second column is used for a short description (which this function discards). For complete details on the .gmt format, refer to the Broad Institute's Data Format's page http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats.

Value

A list, where each index represents a separate gene set.

Warning

The function does not check that the file is correctly formatted, and may return incorrect or partial gene sets, e.g. if the first two columns are omitted. Please make sure that files are correctly formatted before reading them in using this function.

```
gmt_path <- system.file("extdata/test_gene_sets_n4.gmt", package="signatureSearch")
geneSets <- read_gmt(gmt_path)</pre>
```

result

Description

Method extracts tibbles from result slots of feaResult and gessResult objects. They are generated by the GESS and FEA functions defined by signatureSearch, respectively.

Usage

```
result(x)
## S4 method for signature 'feaResult'
result(x)
```

S4 method for signature 'gessResult'
result(x)

Arguments

х

gessResult or feaResult object

Value

tibble

Examples

runWF

Run the Entire GESS/FEA Workflow

Description

This function runs the entire GESS/FEA workflow when providing the query drug and cell type, as well as selecting the reference database (e.g. 'cmap' or 'lincs'), defining the specific GESS and FEA methods. In this case, the query GES is drawn from the reference database. The N (defined by the 'N_gess_drugs' argument) top ranking hits in the GESS tables were then used for FEA where three different annotation systems were used: GO Molecular Function (GO MF), GO Biological Process (GO BP) and KEGG pathways.

The GESS/FEA results will be stored in a list object in R session. A working environment named by the use case will be created under users current working directory or under other directory defined by users. This environment contains a results folder where the GESS/FEA result tables were written to. The working environment also contains a template Rmd vignette as well as a rended HTML report, users could make modifications on the Rmd vignette as they need and re-render it to generate their HTML report.

Usage

```
runWF(
 drug,
 cell,
 refdb,
 gess_method,
 fea_method,
 N_gess_drugs = 100,
 env_dir = ".",
  tau = TRUE,
 Nup = 150,
 Ndown = 150,
 higher = 1,
 lower = -1,
 method = "spearman",
 pvalueCutoff = 1,
 qvalueCutoff = 1,
 minGSSize = 5,
 maxGSSize = 500,
 runFEA = TRUE,
 GenerateReport = TRUE
)
```

Arguments

drug	character(1) representing query drug name (e.g. vorinostat). This query drug should be included in the refdb
cell	character(1) indicating the cell type that the query drug treated in. Details about cell type options in LINCS database can be found in the cell_info table after load the 'signatureSearch' package and running 'data("cell_info")'
refdb	character(1), one of "lincs", "lincs_expr", "cmap", "cmap_expr", or path to the HDF5 file built from build_custom_db function
gess_method	character(1), one of "LINCS", "CORsub", "CORall", "Fisher", "CMAP", "gCMAP". When gess_method is "CORsub" or "CORall", only "lincs_expr" or "cmap_expr" databases are supported.
fea_method	character(1), one of "dup_hyperG", "mGSEA", "mabs", "hyperG", "GSEA"
N_gess_drugs	number of unique drugs in GESS result used as input of FEA
env_dir	character(1), directory under which the result environment located. The default is users current working directory in R session, can be checked via getwd() command in R
tau	TRUE or FALSE indicating whether to compute Tau scores if gess_method is set as 'LINCS'

62

Nup	integer(1). Number of most up-regulated genes to be subsetted for GESS query when gess_method is CMAP, LINCS or CORsub
Ndown	integer(1). Number of most down-regulated genes to be subsetted for GESS query when gess_method is CMAP, LINCS or CORsub
higher	numeric(1), it is defined when gess_method argument is 'gCMAP' or 'Fisher' representing the 'upper' threshold of subsetting genes with a score larger than 'higher'
lower	numeric(1), it is defined when gess_method argument is 'gCMAP' or 'Fisher' representing the 'lower' threshold of subsetting genes
method	One of 'spearman' (default), 'kendall', or 'pearson', indicating which correla- tion coefficient to use
pvalueCutoff	double, p-value cutoff for FEA result
qvalueCutoff	double, qvalue cutoff for FEA result
minGSSize	integer, minimum size of each gene set in annotation system
maxGSSize	integer, maximum size of each gene set in annotation system
runFEA	Logical value indicating if FEA analysis is performed.
GenerateReport	Logical value indicating if a report is generated.

Value

list object containing GESS/FEA result tables

Examples

```
drug <- "vorinostat"; cell <- "SKB"
refdb <- system.file("extdata", "sample_db.h5", package="signatureSearch")
env_dir <- tempdir()
wf_list <- runWF(drug, cell, refdb, gess_method="LINCS",
    fea_method="dup_hyperG", N_gess_drugs=10, env_dir=env_dir, tau=FALSE,
    runFEA=FALSE, GenerateReport= FALSE)</pre>
```

set_readable Set Readable

Description

Mapping 'itemID' column in the FEA enrichment result table from Entrez ID to gene Symbol

Usage

```
set_readable(
   tb,
   OrgDb = "org.Hs.eg.db",
   keyType = "ENTREZID",
   geneCol = "itemID"
)
```

Arguments

tb	tibble object, enrichment result table
OrgDb	character(1), 'org.Hs.eg.db' for human
keyType	character(1), keyType of gene
geneCol	character(1), name of the column in 'tb' containing gene Entrez ids separated by '/' to be converted to gene Symbol

Value

tibble Object

Examples

show

show method

Description

show qSig, gessResult, feaResult objects

Usage

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

show(object)

S4 method for signature 'qSig'
show(object)

Arguments

object object used for show

Value

message

Examples

64

sim_score_grp

gr

sim_score_grp

Summary Scores by Groups of Cell Types

query=list(up=c("g1","g2"), down=c("g3","g4")), gess_method="LINCS", refdb="path/to/lincs/db")

Description

Function appends two columns (score_column_grp1, score_column_grp2) to GESS result tibble. The appended columns contain cell-summarized scores for groups of cell types, such as normal and tumor cells. The cell-summarized score is obtained the same way as the NCSct scores, that is using a maximum quantile statistic. It compares the 67 and 33 quantiles of scores and retains whichever is of higher absolute magnitude.

Usage

sim_score_grp(tib, grp1, grp2, score_column)

Arguments

tib	tibble in gessResult object
grp1	character vector, group 1 of cell types, e.g., tumor cell types
grp2	character vector, group 2 of cell types, e.g., normal cell types
score_column	character, column name of similarity scores to be grouped

Value

tibble

tail

Description

Return the last part of the result table in the gessResult, and feaResult objects

Usage

```
## S4 method for signature 'gessResult'
tail(x, n = 6L, ...)
## S4 method for signature 'feaResult'
tail(x, n = 6L, ...)
```

Arguments

х	an object
n	a single integer. If positive or zero, size for the resulting object is the number of rows for a data frame. If negative, all but the n first number of rows of x.
	arguments to be passed to or from other methods

Value

data.frame

Examples

targetList

Target Sample Data Set

Description

A named numeric vector with Gene Symbols as names. It is the first 1000 elements from the 'targets' slot of the 'mgsea_res' result object introduced in the vignette of this package. The scores represent the weights of the target genes/proteins in the target set of the selected top 10 drugs.

Usage

targetList

tarReduce

Format

An object of class numeric of length 1000.

Examples

```
# Load object
data(targetList)
head(targetList)
tail(targetList)
```

tarReduce

Show Reduced Targets

Description

Reduce number of targets for each element of a character vector by replacting the targets that beyond Ntar to '...'.

Usage

tarReduce(vec, Ntar = 5)

Arguments

vec	character vector, each element composed by a list of targets symbols separated
	by '; '
Ntar	integer, for each element in the vec, number of targets to show

Value

character vector after reducing

Examples

```
vec <- c("t1; t2; t3; t4; t5; t6", "t7; t8")
vec2 <- tarReduce(vec, Ntar=5)</pre>
```

vec_char_redu Reduce Number of Character

Description

Reduce number of characters for each element of a character vector by replacting the part that beyond Nchar (e.g. 50) character to '...'.

Usage

```
vec_char_redu(vec, Nchar = 50)
```

Arguments

vec	character vector to be reduced
Nchar	integer, for each element in the vec, number of characters to remain

Value

character vector after reducing

```
vec <- c(strrep('a', 60), strrep('b', 30))
vec2 <- vec_char_redu(vec, Nchar=50)</pre>
```

Index

```
* GCTX parsing functions
    parse_gctx, 56
* classes
    feaResult-class, 29
    gessResult-class, 31
    qSig-class, 58
* datasets
    cell_info, 10
    cell_info2, 11
    chembl_moa_list, 11
    clue_moa_list, 12
    drugs10, 15
    lincs_expr_inst_info, 48
    lincs_pert_info, 48
    lincs_pert_info2, 49
    lincs_sig_info, 49
    targetList, 66
add_pcid, 7
addGESSannot, 5, 37
addMOA, 7
append2H5, 8
build_custom_db, 8, 10, 36, 40, 42, 58, 59, 62
calcGseaStatBatchCpp, 9
cell_info, 10
cell_info2, 11
cellNtestPlot, 10
chembl_moa_list, 11
clue_moa_list, 12
comp_fea_res, 12
create_empty_h5, 13
dim, 14
dim, feaResult-method (dim), 14
dim, gessResult-method (dim), 14
drug_cell_ranks, 16
drugs, 15
drugs, feaResult, ANY-method (drugs), 15
drugs, feaResult-method (drugs), 15
drugs10, 15
drugs<- (drugs), 15
drugs<-, feaResult-method (drugs), 15
```

dsea_GSEA, 5, 17 dsea_hyperG, 5 dsea_hyperG (dsea_GSEA), 17 dtlink_db_clue_sti, 43 dtnetplot, 23 enrichG02, 24 enrichKEGG2, 25 enrichMOA, 26 enrichReactome, 27 feaResult, 14, 20, 21, 28, 47, 51-53, 64, 66 feaResult-class, 28, 29 GCT object, 29 gctx2h5, 30 geom_point, 13, 38 gess_cmap, 5, 31 gess_cor, 5 gess_cor (gess_cmap), 31 gess_fisher, 5 gess_fisher (gess_cmap), 31 gess_gcmap, 5 gess_gcmap (gess_cmap), 31 gess_lincs, 5, 15, 59 gess_lincs (gess_cmap), 31 gess_res_vis, 38 gessResult, 14, 30, 35, 37, 38, 47, 55, 56, 64, 66 gessResult-class, 30, 31 get_targets, 6, 19, 34, 42 getALLEG, 39 getDb, 39 getDEGSig (getSig), 40 getSig, 40 getSPsubSig (getSig), 40 getTreats, 41 gmt2h5, 43, 58 GO_DATA_drug, 21 gseG02, 44 gseKEGG2, 45 gseReactome, 46

head, 47

```
head, feaResult-method (head), 47
head, gessResult-method (head), 47
lincs_expr_inst_info, 48
lincs_pert_info, 48
lincs_pert_info2, 49
lincs_sig_info, 49
list2df, 50
list_rev, 50
mabsGO, 51
mabsKEGG, 52
mabsReactome, 53
matrix2h5, 54
meanExpr2h5, 54
moa_conn, 55
parse_gctx, 29, 56
qSig, 10, 33, 37, 42, 57, 57, 64
qSig-class, 58
rand_query_ES, 59
read_gmt, 60
result, 29, 31, 38, 61
result, feaResult-method (result), 61
result, gessResult-method (result), 61
runWF, 61
set_readable, 63
show, 64
show, feaResult-method (show), 64
show, gessResult-method (show), 64
show,qSig-method(show),64
signatureSearch
        (signatureSearch-package), 3
signatureSearch-package, 3
signatureSearchData, 42, 58, 59
sim_score_grp, 65
tail, 66
tail, feaResult-method (tail), 66
tail,gessResult-method(tail),66
targetList, 66
tarReduce, 67
tsea_dup_hyperG, 5
tsea_dup_hyperG (dsea_GSEA), 17
tsea_mabs, 5
tsea_mabs (dsea_GSEA), 17
tsea_mGSEA, 5
tsea_mGSEA (dsea_GSEA), 17
```

```
vec_char_redu, 67
visNetwork, 23
```

70