Package 'enrichit'

October 28, 2026

Title 'C++' Implementations of Functional Enrichment Analysis

Version 0.2.7

Maintainer Guangchuang Yu <guangchuangyu@gmail.com>

Description Fast implementations of functional enrichment analysis methods using 'C++' via 'Rcpp'. Currently provides Over-Representation Analysis (ORA), Gene Set Enrichment Analysis (GSEA), design-aware GSEA on RNA-seq counts with per-sample gene set scoring, Weighted Enrichment Analysis for ORA and GSEA, Network-based Set Enrichment Analysis (NSEA), multi-layer network-based enrichment, and multi-omics integration workflows. Additional features include early fusion at the feature level, late fusion at the pathway level, multi-omics contribution tracing, topology-aware explanation helpers, Bayesian term selection, and extremely fast Random Walk with Restart (RWR) using 'RcppEigen'. The enrichment methods build on GSEA by Subramanian et al. (2005) <doi:10.1073/pnas.0506580102>, the multilevel strategy derived from 'fgsea' by Korotkevich et al. (2021) <doi:10.1101/060012>, and network-based enrichment ideas described by Glaab et al. (2012) <doi:10.1093/bioinformatics/bts389>.

License Artistic-2.0

Depends R (>= 3.5.0)

Imports Matrix, methods, Rcpp (>= 1.0.10), rlang, S4Vectors, stats, SummarizedExperiment, yulab.utils (> 0.2.1)

LinkingTo Rcpp, RcppEigen

Suggests AnnotationDbi, BiasedUrn, clusterProfiler, DESeq2, DOSE, fgsea, gson, qvalue, testthat, jsonlite

Encoding UTF-8

URL https://yulab-smu.top/biomedical-knowledge-mining-book/

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Guangchuang Yu [aut, cre]

Repository CRAN

Date/Publication 2026-10-28 05:50:02 UTC

Contents

aggregate_enrichment . . . 3
aggregate_omics . . . 3
as_enrichResult . . . 4
as_genes . . . 6
as_gseaResult . . . 7
as_ranked . . . 9
bayes_enrich . . . 10
bayes_summary . . . 11
classify_omics_pattern . . . 12
collapse_multilayer_scores . . . 13
compareClusterResult-class . . . 13
dsea . . . 14
enrichResult-class . . . 16
EXTID2NAME . . . 17
extract_mnsea_subnetwork . . . 18
geneID . . . 18
geneInCategory . . . 19
get_mnsea_contribution . . . 20
get_omics_contribution . . . 20
gsea . . . 21
gseaResult-class . . . 23
gseaScores . . . 24
gsea_gson . . . 24
gsfilter . . . 26
harmonize_ids . . . 27
mnsea . . . 27
mnseaResult-class . . . 29
mnsea_gson . . . 30
nsea . . . 31
nseaResult-class . . . 33
nsea_gson . . . 34
ora . . . 35
ora_gson . . . 36
prepare_multilayer_network . . . 37
prepare_network . . . 38
propagate_multilayer . . . 39
select_features_for_ora . . . 40
setReadable . . . 40
show . . . 41
ssgsea_scores . . . 42
summary . . . 43

Index **44**

aggregate_enrichment *Aggregate multiple enrichment results (Late Fusion)*

Description

Combine pathway-level enrichment results from multiple omics or independent analyses. P-values of identical pathways are merged using statistical methods (e.g., Brown's method).

Usage

```
aggregate_enrichment(res_list, method = c("brown", "fisher", "stouffer"), ...)
```

Arguments

res_list	A named list of enrichment result objects (e.g., enrichResult, gseaResult, nseaResult).
method	Character, aggregation method for p-values. One of "brown", "fisher", or "stouffer".
...	Additional arguments passed to aggregate_omics (e.g., cov_matrix for Brown's method).

Value

An enrichResult object containing the aggregated p-values, FDR, and combined gene lists.

aggregate_omics *Aggregate multi-omics gene/protein-level statistics*

Description

Aggregate multi-omics or multi-source statistics into a unified object for downstream enrichment analysis.

Usage

```
aggregate_omics(
  x,
  method = c("fisher", "stouffer", "brown", "mean", "weighted_mean", "max_abs"),
  input = c("pvalue", "signed_score"),
  feature_type = "gene",
  conflict_policy = c("keep_all", "strict", "penalty"),
  ...
)
```

Arguments

x	A list of named numeric vectors, a data.frame, or a matrix. Row names (or names for vectors) must represent feature IDs.
method	Character, aggregation method. One of "fisher", "stouffer", "mean", or "max_abs".
input	Character, input type. One of "pvalue" or "signed_score".
feature_type	Character, type of the features (e.g., "gene", "protein"). Default is "gene".
conflict_policy	Character, strategy to handle directional conflicts when input is "signed_score". One of "keep_all" (default, ignore conflicts), "strict" (set to NA if any signs conflict), or "penalty" (divide final score by 2 if signs conflict).
...	Additional arguments.

Value

An object of class `omics_aggregated` containing `score`, `pvalue` (if input is "pvalue"), `input_type`, `feature_type`, and `feature_id`.

---

as_enrichResult *as_enrichResult*

Description

Convert a result table to an `enrichResult` object

Usage

```
as_enrichResult(x, ...)

## Default S3 method:
as_enrichResult(
  x,
  geneSets = NULL,
  gene = NULL,
  universe = NULL,
  ontology = "UNKNOWN",
  organism = "UNKNOWN",
  keytype = "UNKNOWN",
  pAdjustMethod = "BH",
  ...
)
```

Arguments

x	result table (data.frame) with the canonical ORA columns, or a table with common aliases (e.g. PValue, term_id); tool-specific naming should be mapped to the canonical schema by the per-tool importers in 'enrichplot'
...	additional arguments passed to methods
geneSets	gene sets as a named list, a two-column data.frame (term, gene), or a GSON object. If NULL, gene sets are rebuilt from the geneID column (overlap genes only, sufficient for cnetplot()/heatplot()).
gene	query gene vector used in the analysis. If NULL, inferred as the union of geneID entries.
universe	background gene vector. If NULL, inferred as the union of geneSets when available.
ontology, organism, keytype	metadata stored in the object slots.
pAdjustMethod	method passed to stats::p.adjust when p.adjust is missing.

Details

Generic constructor for over-representation analysis results produced by other tools. The input must follow the canonical column schema documented in enrichplot::fortify(); a minimal set of columns (ID, pvalue and at least one of geneID, Count or GeneRatio) is required, the remaining canonical columns are derived when possible.

Value

An enrichResult object. The significance columns are always laid out in the same three-column convention:

pvalue Raw nominal p-value (no multiplicity correction) supplied by the source analysis. For native **enrichit** ORA this is typically Fisher's exact; for imported external results it reflects the method used by the source tool.

p.adjust stats::p.adjust(pvalue, method = pAdjustMethod); default "BH" for Benjamini & Hochberg (1995) step-up false discovery rate control. Because no π_0 is estimated, BH-adjusted p-values are typically **>=** the Storey q-values on the same data.

qvalue Storey & Tibshirani (2003) positive FDR estimate computed through the **qvalue** package. π_0 (the proportion of tests drawn from the null) is estimated from the empirical p-value distribution, so q-values are typically smaller than BH-p.adjust under a non-null majority. If estimation fails (e.g. qvalue is not installed, π_0 estimation fails, or too few p-values are provided), qvalue is left as NA. Use p.adjust when you need a complete significance column for downstream work.

as_genes *Extract significant genes from DESeq2 results for ORA*

Description

Select the genes passing an adjusted-p-value cut-off (optionally split by direction of change) from a **DESeq2** analysis, as a character vector of gene identifiers for `ora`.

Usage

```
as_genes(
  x,
  cutoff = 0.05,
  direction = c("both", "up", "down"),
  minAbsLFC = 0,
  ...
)

## S3 method for class 'DESeqResults'
as_genes(
  x,
  cutoff = 0.05,
  direction = c("both", "up", "down"),
  minAbsLFC = 0,
  ...
)

## S3 method for class 'DESeqDataSet'
as_genes(
  x,
  cutoff = 0.05,
  direction = c("both", "up", "down"),
  minAbsLFC = 0,
  ...
)
```

Arguments

`x`	a `DESeqResults` or `DESeqDataSet` object. For a `DESeqDataSet`, `...` is forwarded to `DESeq2::results()`.
`cutoff`	numeric, adjusted p-value cut-off (default 0.05); genes with `padj` < `cutoff` are kept.
`direction`	character, `"both"` (default), `"up"` or `"down"`; selects genes with positive or negative `log2FoldChange` among the significant ones.
`minAbsLFC`	numeric, keep only significant genes with `abs(log2FoldChange)` > `minAbsLFC` (default 0, no filtering).
`...`	additional arguments; for a `DESeqDataSet` these are passed to `DESeq2::results()`.

Details

Selection uses `padj < cutoff` on the `padj` column as produced by `DESeq2::results()` (which includes independent filtering). `minAbsLFC` is a post-hoc filter applied to the reported `log2FoldChange`; it does not change the underlying test. To test against a minimum effect size, set `lfcThreshold` in `results()` / `DESeq2::results()` instead.

Value

A character vector of gene identifiers (empty if no gene passes, with a warning).

Examples

```
library(DESeq2)
dds <- makeExampleDESeqDataSet(n = 500)
isB <- colData(dds)$condition == "B"
ct <- counts(dds)
ct[1:60, isB] <- ct[1:60, isB] * 5 + 500
storage.mode(ct) <- "integer"
counts(dds) <- ct
dds <- DESeq(dds, quiet = TRUE)
up_genes <- as_genes(dds, contrast = c("condition", "B", "A"), direction = "up")
gene_sets <- list(
    SET1 = paste0("gene", 1:50),
    SET2 = paste0("gene", 201:260)
)
res <- ora(gene = up_genes, gene_sets = gene_sets, universe = rownames(dds))
```

`as_gseaResult` *as_gseaResult*

Description

Convert a result table to a `gseaResult` object

Usage

```
as_gseaResult(x, ...)

## Default S3 method:
as_gseaResult(
  x,
  geneList,
  geneSets = NULL,
  setType = "UNKNOWN",
  organism = "UNKNOWN",
  keytype = "UNKNOWN",
  exponent = 1,
```

```
  scoreType = "std",
  pAdjustMethod = "BH",
  ...
)
```

Arguments

x	result table (data.frame) with GSEA columns (ID, enrichmentScore, pvalue; commonly also NES, p.adjust, setSize, core_enrichment; fgsea-style pathway/ES/pval/padj/size are recognized)
...	additional arguments passed to methods
geneList	named numeric vector of ranked statistics, sorted in descending order (sorted automatically with a warning if not).
geneSets	gene sets as a named list, a two-column data.frame (term, gene), or a GSON object. If NULL, gene sets are rebuilt from core_enrichment (leading edge genes only) and running-score plots will only be approximate.
setType, organism, keytype	metadata stored in the object slots.
exponent, scoreType	parameters used to recompute missing rank/leading_edge/core_enrichment columns.
pAdjustMethod	method passed to stats::p.adjust when p.adjust is missing.

Details

Generic constructor for pre-ranked GSEA results produced by other tools (e.g. 'fgsea' or GSEA reports). The ranked gene statistics must be supplied separately as geneList; the result table alone cannot be converted because most GSEA visualizations consume the ranked list.

Value

A gseaResult object. The result slot carries the same three-column significance layout documented in the as_enrichResult @return section:

pvalue Raw nominal p-value from the chosen GSEA permutation or multilevel null.

p.adjust stats::p.adjust(pvalue, method = pAdjustMethod) (default "BH"). BH-adjusted p-values are typically **>=** the Storey q-value on the same data.

qvalue Storey & Tibshirani (2003) pFDR estimate computed through the **qvalue** package. π_0 is estimated from the empirical p-value distribution, so q-values are typically smaller than BH-p.adjust under a non-null majority. If estimation fails, qvalue is left as NA; use p.adjust when you need a complete significance column.

See the as_enrichResult man page for the full explanation.

as_ranked *Convert DESeq2 results to a ranked gene list for GSEA*

Description

Extract a per-gene statistic from a **DESeq2** analysis as a named numeric vector sorted in descending order, ready to be passed to gsea. This uses the full ranked list rather than a significance cut-off, so no information is discarded by dichotomising the genes.

Usage

```
as_ranked(x, statistic = c("stat", "log2FoldChange"), ...)

## S3 method for class 'DESeqResults'
as_ranked(x, statistic = c("stat", "log2FoldChange"), ...)

## S3 method for class 'DESeqDataSet'
as_ranked(x, statistic = c("stat", "log2FoldChange"), ...)
```

Arguments

x	a DESeqResults or DESeqDataSet object. For a DESeqDataSet, ... is forwarded to DESeq2::results() (e.g. contrast, name, alpha, lfcThreshold).
statistic	character, the column to rank by: "stat" (the Wald statistic, default) or "log2FoldChange". The Wald statistic is generally preferred for ranking as it incorporates the per-gene standard error.
...	additional arguments; for a DESeqDataSet these are passed to DESeq2::results().

Details

Non-finite statistics (genes with all-zero counts, genes flagged by outlier replacement, or failed fits) are dropped with a message. Duplicated gene identifiers are resolved by keeping the entry with the largest absolute statistic, with a warning. Note that lfcShrink() with type = "apeglm" does not return a stat column; use statistic = "log2FoldChange" for shrunk results, and keep in mind that shrunk fold changes are intended for ranking and visualization, not for statistical testing.

Value

A named numeric vector sorted in descending order, suitable for the geneList argument of gsea.

Examples

```
library(DESeq2)
dds <- makeExampleDESeqDataSet(n = 500)
## strengthen a subset of genes in condition B (keep the rest near-null
## so the size factors have a stable reference)
isB <- colData(dds)$condition == "B"
```

```
ct <- counts(dds)
ct[1:60, isB] <- ct[1:60, isB] * 5 + 500
storage.mode(ct) <- "integer"
counts(dds) <- ct
dds <- DESeq(dds, quiet = TRUE)
ranked <- as_ranked(dds, contrast = c("condition", "B", "A"))
gene_sets <- list(
    SET1 = paste0("gene", 1:50),
    SET2 = paste0("gene", 201:260)
)
res <- gsea(ranked, gene_sets, nPerm = 1000)
```

`bayes_enrich` *Bayesian term selection for enrichment results*

Description

`bayes_enrich()` adds a model-based selection layer on top of ORA results. It estimates the posterior probability that each candidate term is an active biological program explaining the observed input genes.

Usage

```
bayes_enrich(
  x,
  candidate = c("top", "significant", "all"),
  n_terms = 200,
  by = "p.adjust",
  prior = 0.1,
  false_positive = 0.01,
  false_negative = 0.9,
  n_iter = 5000,
  burnin = 1000,
  thin = 1,
  posterior_cutoff = 0.5,
  seed = NULL,
  verbose = FALSE
)
```

Arguments

`x`	An `enrichResult` object, typically from `ora_gson()` or a package that builds on enrichit, such as clusterProfiler.
`candidate`	Candidate terms to include. `"significant"` uses `as.data.frame(x)`, `"all"` uses `x@result`, `"top"` uses the top `n_terms` rows from `x@result` ordered by `by`; or provide a character vector of term IDs.

n_terms	Maximum number of candidate terms when candidate = "top" or when more candidates are supplied than this value. Use Inf to disable.
by	Column used to order candidate terms.
prior	Prior probability that a term is active.
false_positive	Probability of observing a gene not covered by active terms.
false_negative	Probability of missing a gene covered by active terms. ORA hit lists contain only a fraction of the genes of each active program, so a realistic scale is close to 1 (default 0.9). The previous default (0.1) assumed the hits cover most genes of the active programs, which pathway-sized gene sets never satisfy: it pushed the posterior of every candidate to 0 on realistic KEGG/Reactome data.
n_iter	Total number of MCMC iterations.
burnin	Number of initial iterations discarded.
thin	Keep one sample every thin iterations after burn-in.
posterior_cutoff	Terms with posterior greater than or equal to this value are marked active.
seed	Optional random seed.
verbose	Print sampler progress.

Details

The implementation uses a lightweight Metropolis-Hastings sampler over binary latent term states. Given active terms, each gene is modeled as observed with probability 1 - false_negative if covered by at least one active term, and with probability false_positive otherwise. The prior probability that a candidate term is active is prior.

This is intended as a result-compression and interpretation layer, not as a replacement for ORA p-values.

Value

The input enrichResult object with additional columns in @result: posterior, posterior_odds, bayes_rank, bayes_active, bayes_covered_gene, and bayes_covered_count.

---

bayes_summary *Summarize Bayesian enrichment results*

Description

Return a data frame sorted by posterior probability from a result processed by bayes_enrich(). This is a convenience wrapper around sorting as.data.frame(x) by decreasing posterior.

Usage

```
bayes_summary(x, active = FALSE, n = Inf)
```

Arguments

x	An enrichResult object processed by bayes_enrich().
active	Logical. If TRUE, keep only rows with bayes_active = TRUE.
n	Number of rows to return. Use Inf to return all rows.

Value

A data frame ordered by decreasing posterior.

---

classify_omics_pattern

Classify pathway-level multi-omics patterns

Description

Compare merged enrichment results with single-omics enrichment results to classify the contribution pattern of each pathway.

Usage

```
classify_omics_pattern(
  merged_res,
  single_res,
  p_cutoff = 0.05,
  by = "p.adjust"
)
```

Arguments

merged_res	An enrichResult or gseaResult object from the merged multi-omics analysis.
single_res	A named list of enrichResult or gseaResult objects from single-omics analyses.
p_cutoff	Numeric, the significance cutoff. Default is 0.05.
by	Character, the column to use for significance threshold. Default is "p.adjust".

Value

The merged_res object with an additional column Omics_Pattern in its result data.frame.

collapse_multilayer_scores *Collapse multi-layer diffusion scores*

Description

Collapse multi-layer diffusion scores

Usage

```
collapse_multilayer_scores(
  x,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  layer_weights = NULL,
  output_space = c("union", "gene"),
  mapping = NULL,
  target_layer = NULL
)
```

Arguments

x	result from propagate_multilayer().
collapse	one of "weighted_mean", "sum", "mean", or "max_abs".
layer_weights	optional named numeric vector used when collapse = "weighted_mean".
output_space	one of "union" or "gene".
mapping	optional mapping data.frame with source_id, target_id, and optional layer columns.
target_layer	optional layer name to extract before collapsing.

Value

A multilayer_collapsed object with a score vector.

compareClusterResult-class *Class "compareClusterResult" This class represents the comparison result of gene clusters by GO categories at specific level or GO enrichment analysis.*

Description

Class "compareClusterResult" This class represents the comparison result of gene clusters by GO categories at specific level or GO enrichment analysis.

Slots

`compareClusterResult` cluster comparing result

`geneClusters` a list of genes

`fun` one of groupGO, enrichGO and enrichKEGG

`gene2Symbol` gene ID to Symbol

`keytype` Gene ID type

`readable` logical flag of gene ID in symbol or not.

`.call` function call

`termsim` Similarity between term

`method` method of calculating the similarity between nodes

`dr` dimension reduction result

`organism` organism

Author(s)

Guangchuang Yu https://yulab-smu.top

See Also

`enrichResult`

`dsea` *Differential set enrichment from a design*

Description

`dsea()` takes counts and a design and returns GSEA results for a contrast: it transforms the counts, reduces the design to a per-gene statistic, ranks the genes and runs the GSEA engine. `dsea_scores()` stops one step earlier and returns the gene set by sample score matrix instead.

Usage

```
dsea(
  x,
  gene_sets,
  design,
  contrast = NULL,
  transform = c("vst", "rlog"),
  min_size = 10L,
  max_size = 500L,
  ...
)

dsea_scores(
```

```
  x,
  gene_sets,
  transform = c("vst", "rlog", "logcpm"),
  min_size = 10L,
  max_size = 500L,
  exponent = 1,
  ...
)
```

Arguments

x	a counts matrix, or a DESeqDataSet.
gene_sets	a named list of gene identifier vectors, identifiers matching rownames(x).
design	for dsea(): an n x p design matrix from model.matrix(), row-aligned with the samples. Must be full rank.
contrast	a column name or index of design. Defaults to the only non-intercept column of a two-column design; a wider design must name the contrast explicitly, because the useful column cannot be guessed.
transform	"vst" (default) or "rlog"; dsea_scores() also accepts "logcpm".
min_size, max_size	gene set size limits, applied to the overlap with the genes actually present.
...	for dsea(), passed to gsea().
exponent	for dsea_scores(): rank weighting passed to ssgsea_scores().

Details

Why the ranking is what it is. Genes are ranked by the difference in mean transformed expression between the two sides of the contrast. That looks like the weakest available choice and it is the one the benchmark supported: ranking by a moderated t or by DESeq2's Wald statistic was no better under uniform or heterogeneous effects, and materially worse when the affected genes were the noisier ones (recall 0.826 for Wald against 0.979 for the mean difference under an eightfold dispersion multiplier; across a 1x/2x/4x series the Wald statistic falls 0.995 to 0.950 while the mean difference stays flat). GSEA's statistic already equalises genes by rank, so layering a variance weight on top discards real signal at the extremes rather than adding power.

Which column may be the contrast. A design column encodes a comparison only when it is not constant and not a group indicator under cell-means coding. The intercept is rejected because ranking by it orders genes by mean expression, which turns the analysis into "highly expressed pathways"; a column of ~ 0 + condition is rejected for the same reason, since its coefficient is one group's mean rather than a difference. A continuous column (a dose slope, say) is allowed.

The transform uses design(x), the ranking uses design**.** When x is a DESeqDataSet, vst(blind = FALSE) reads the design stored in the object, while the ranking reads the design argument. If the two differ the call warns rather than blending two models: refit the object with the intended design, or pass a counts matrix so the transform can use design.

What the design is used for. For a two-group contrast the ranking is the group mean difference. For a design with nuisance terms it is the contrast coefficient from the fitted linear model, so batch and other covariates are removed before ranking rather than ignored.

What this is not. It is a design-aware front end to GSEA, not a new set test. Directions are relative to the ranking: a set scoring high sits toward the top, which is the direction the contrast was oriented in.

Value

`dsea()` returns the data.frame from `gsea()`, sorted by p-value, with two columns added: `padj`, the Benjamini-Hochberg adjusted p-value, and `contrast`, the design column that was tested. Everything else is GSEA's own output, so the table can be filtered and plotted like any other GSEA result.

`dsea_scores()` returns a list with `scores` (gene sets by samples) and `meta` recording the transform and scorer settings. `meta$set_sizes` reports the number of members actually scored, after intersecting with the data, and `meta$raw_set_sizes` the input size, so the two can be compared.

See Also

`gsea`, `ssgsea_scores`

Examples

```
## Not run:
dds <- DESeq2::DESeq(DESeq2::DESeqDataSetFromMatrix(counts, coldata, ~ condition))
design <- model.matrix(~ condition, coldata)
res <- dsea(dds, gene_sets, design = design, contrast = "conditionB")
head(res[order(res$pvalue), ])

sc <- dsea_scores(dds, gene_sets)
dim(sc$scores)

## End(Not run)
```

`enrichResult-class` *Class "enrichResult" This class represents the result of enrichment analysis.*

Description

Class "enrichResult" This class represents the result of enrichment analysis.

Slots

`result` enrichment analysis

`pvalueCutoff` pvalueCutoff

`pAdjustMethod` pvalue adjust method

`qvalueCutoff` qvalueCutoff

`organism` only "human" supported

`ontology` biological ontology

`gene` Gene IDs

`keytype` Gene ID type

`universe` background gene

`gene2Symbol` mapping gene to Symbol

`geneSets` gene sets

`readable` logical flag of gene ID in symbol or not.

`termsim` Similarity between term

`method` method of calculating the similarity between nodes

`dr` dimension reduction result

Author(s)

Guangchuang Yu https://yulab-smu.top

`EXTID2NAME` *EXTID2NAME*

Description

mapping gene ID to gene Symbol

Usage

```
EXTID2NAME(OrgDb, geneID, keytype, toType = "SYMBOL")
```

Arguments

`OrgDb`	OrgDb
`geneID`	entrez gene ID
`keytype`	keytype
`toType`	ID type of the output

Value

gene symbol

Author(s)

Guangchuang Yu https://yulab-smu.top

extract_mnsea_subnetwork *Extract pathway subnetwork data from a* mnseaResult

Description

Extract pathway subnetwork data from a mnseaResult

Usage

```
extract_mnsea_subnetwork(
  res,
  pathway_id = NULL,
  include_couplings = TRUE,
  include_isolated = TRUE
)
```

Arguments

res	A mnseaResult object.
pathway_id	Optional pathway ID. If NULL, the top pathway is used.
include_couplings	Logical, whether to include inter-layer coupling edges. Default is TRUE.
include_isolated	Logical, whether to keep nodes without retained edges. Default is TRUE.

Value

A list with pathway, layer_contribution, nodes, and edges.

geneID *geneID generic*

Description

geneID generic

Usage

```
geneID(x)
```

Arguments

x	enrichResult object

Value

'geneID' return the 'geneID' column of the enriched result which can be converted to data.frame via 'as.data.frame'

Examples

```
## Not run:
data(geneList, package="DOSE")
de <- names(geneList)[1:100]
x <- DOSE::enrichDO(de)
geneID(x)

## End(Not run)
```

---

geneInCategory *geneInCategory generic*

Description

geneInCategory generic

Usage

```
geneInCategory(x)
```

Arguments

x	enrichResult

Value

'geneInCategory' return a list of genes, by spliting the input gene vector to enriched functional categories

Examples

```
## Not run:
data(geneList, package="DOSE")
de <- names(geneList)[1:100]
x <- DOSE::enrichDO(de)
geneInCategory(x)

## End(Not run)
```

get_mnsea_contribution

Get cached contribution tables from a `mnseaResult`

Description

Get cached contribution tables from a `mnseaResult`

Usage

```
get_mnsea_contribution(res, pathway_id = NULL, level = c("pathway", "feature"))
```

Arguments

`res`	A `mnseaResult` object.
`pathway_id`	Optional pathway ID. If `NULL`, returns all pathways for `level = "pathway"` and uses the top pathway for `level = "feature"`.
`level`	One of `"pathway"` or `"feature"`.

Value

A data.frame containing cached contribution information.

get_omics_contribution

Get gene-level omics contribution for a specific pathway

Description

Extract the original multi-omics statistics for genes in a specific enriched pathway.

Usage

```
get_omics_contribution(res, agg, pathway_id = NULL)
```

Arguments

`res`	An `enrichResult` or `gseaResult` object.
`agg`	An `omics_aggregated` object from `aggregate_omics()`.
`pathway_id`	Character, the ID of the pathway to extract. If NULL, the top pathway is used.

Value

A data.frame containing the genes, their original omics statistics, the aggregated score, and whether they belong to the core enrichment.

gsea *Gene Set Enrichment Analysis (GSEA)*

Description

Perform Gene Set Enrichment Analysis (GSEA) using a ranked gene list.

Usage

```
gsea(
  geneList,
  gene_sets,
  weight = NULL,
  minGSSize = 10,
  maxGSSize = 500,
  nPerm = 1000,
  exponent = 1,
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  eps = 1e-10,
  sampleSize = 101,
  seed = FALSE,
  nPermSimple = 1000,
  scoreType = "std",
  verbose = TRUE
)
```

Arguments

geneList	A named numeric vector of gene statistics (e.g., log fold change), ranked in descending order.
gene_sets	A named list of gene sets. Each element is a character vector of genes.
weight	A named numeric vector of weights for genes. The names should match the names of geneList. If provided, the geneList will be multiplied by the weight and resorted before GSEA (default: NULL).
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of each geneSet for analyzing
nPerm	Number of permutations for p-value calculation (default: 1000).
exponent	Weighting exponent for enrichment score (default: 1.0).
method	Permutation method.
adaptive	Logical. Use adaptive permutation.

minPerm	Minimum permutations for adaptive mode.
maxPerm	Maximum permutations for adaptive mode.
pvalThreshold	P-value threshold for early stopping.
eps	Epsilon for multilevel methods (default: 1e-10). Sets the smallest p-value that can be estimated.
sampleSize	Sample size for multilevel methods (default: 101).
seed	Random seed for reproducibility (default: FALSE). Set to a number to make the result reproducible; set to TRUE to use a fixed default seed. If FALSE (default), a seed is drawn from R's random number generator on each run, so results may vary between runs. Note that the C++ engine uses its own RNG seeded with this value: passing an explicit numeric seed (or calling `set.seed()` before `gsea()` when `seed = FALSE`) makes the result reproducible.
nPermSimple	Number of permutations for the simple method (default: 1000).
scoreType	Type of enrichment score calculation: "std", "pos", "neg" (default: "std"). Only honoured by `method = "multilevel"`; fixed and adaptive permutation paths currently use the standard two-sided score.
verbose	Logical. Print progress messages.

Value

A data.frame with columns:

- **ID**: Gene set name
- **enrichmentScore**: Enrichment Score
- **NES**: Normalized Enrichment Score
- **pvalue**: Empirical p-value from the permutation test. With the default two-sided `scoreType = "std"`, the denominator counts only permutations on the same side of zero as the observed ES (fgsea convention). The `"pos"`/`"neg"` score types are supported by the multilevel engine; fixed and adaptive permutation paths currently use the two-sided ES calculation. For fixed or adaptive permutation runs, the smallest estimable p-value is determined by the number of same-side null permutations, not by the total number of permutations.
- **setSize**: Size of the gene set (number of genes found in geneList)
- **nPerm**: (adaptive mode only) Actual number of permutations used
- **rank**: Rank at which the maximum enrichment score is attained
- **leading_edge**: Leading edge statistics (tags, list, signal)
- **core_enrichment**: Genes in the leading edge, separated by '/'

Examples

```
# Example data
stats <- rnorm(1000)
names(stats) <- paste0("Gene", 1:1000)
stats <- sort(stats, decreasing = TRUE)

gs1 <- paste0("Gene", 1:50)
```

```
gs2 <- paste0("Gene", 500:550)
gene_sets <- list(Pathway1 = gs1, Pathway2 = gs2)

# Use default fixed permutation method
result <- gsea(geneList=stats, gene_sets=gene_sets, nPerm=100)

# Use adaptive permutation for more accurate p-values

result_adaptive <- gsea(geneList=stats, gene_sets=gene_sets, adaptive=TRUE)
```

`gseaResult-class` *Class "gseaResult" This class represents the result of GSEA analysis*

Description

Class "gseaResult" This class represents the result of GSEA analysis

Slots

`result` GSEA anaysis

`organism` organism

`setType` setType

`geneSets` geneSets

`geneList` order rank geneList

`keytype` ID type of gene

`permScores` permutation scores

`params` parameters

`gene2Symbol` gene ID to Symbol

`readable` whether convert gene ID to symbol

`dr` dimension reduction result

Author(s)

Guangchuang Yu https://yulab-smu.top

gseaScores *Calculate GSEA Running Enrichment Scores*

Description

Calculate GSEA Running Enrichment Scores

Usage

```
gseaScores(geneList, geneSet, exponent = 1, fortify = FALSE)
```

Arguments

geneList	a named numeric vector of gene statistics (e.g., t-statistics or log-fold changes), sorted in decreasing order.
geneSet	a character vector of gene IDs belonging to the gene set.
exponent	a numeric value defining the weight of the running enrichment score. Default is 1.
fortify	logical. If TRUE, returns a data frame with columns x, runningScore, and position. If FALSE (default), returns the enrichment score (ES).

Value

If fortify = TRUE, a data frame containing the running enrichment scores and positions. If fortify = FALSE, a numeric value representing the Enrichment Score (ES).

Author(s)

Guangchuang Yu

gsea_gson *gsea_gson*

Description

generic function for gene set enrichment analysis

Usage

```
gsea_gson(
  geneList,
  gson,
  weight = NULL,
  nPerm = 1000,
  exponent = 1,
  minGSSize = 10,
  maxGSSize = 500,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  method = "multilevel",
  adaptive = FALSE,
  minPerm = 101,
  maxPerm = 1e+05,
  pvalThreshold = 0.1,
  seed = FALSE,
  verbose = TRUE,
  reportNA = FALSE,
  ...
)
```

Arguments

geneList	A named numeric vector of gene statistics (e.g., log fold change), ranked in descending order.
gson	A GSON object containing gene set information.
weight	A named numeric vector of weights for genes.
nPerm	Number of permutations for p-value calculation (default: 1000).
exponent	Weighting exponent for enrichment score (default: 1.0).
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of each geneSet for analyzing
pvalueCutoff	P-value cutoff applied to both the raw p-value and the adjusted p-value (`p.adjust`), consistent with the historical clusterProfiler/DOSE behavior (default: 0.05).
pAdjustMethod	P-value adjustment method (e.g., "BH").
method	Permutation method.
adaptive	Logical. Use adaptive permutation.
minPerm	Minimum permutations for adaptive mode.
maxPerm	Maximum permutations for adaptive mode.
pvalThreshold	P-value threshold for early stopping.
seed	Random seed for reproducibility. See `gsea()` for details (default: FALSE).
verbose	Logical. Print progress messages.

reportNA	If TRUE, pathways with NA p-values (produced by the multilevel method when gene-level statistics are unbalanced) are retained in the result. Useful with pvalueCutoff = 1 to inspect the complete set of tested pathways. Default: FALSE, which excludes NA rows from the reported result.
...	Additional parameters passed to gsea()

Value

A gseaResult object. The result slot carries the three significance columns pvalue, p.adjust, and qvalue: raw nominal p-values, BH-style multiple-testing adjusted p-values (via stats::p.adjust(); default pAdjustMethod = "BH"), and Storey q-values (via qvalue::qvalue()). If q-value estimation fails, qvalue remains NA. See ?as_gseaResult for the full column semantics.

Author(s)

Guangchuang Yu

---

gsfilter *gsfilter*

Description

filter enriched result by gene set size or gene count

Usage

```
gsfilter(x, by = "GSSize", min = NA, max = NA)
```

Arguments

x	instance of enrichResult or compareClusterResult
by	one of 'GSSize' or 'Count'
min	minimal size
max	maximal size

Value

update object

Author(s)

Guangchuang Yu

harmonize_ids *Harmonize feature IDs to a target space*

Description

Map protein-level or other feature-level statistics to a unified gene-level space.

Usage

```
harmonize_ids(
  x,
  mapping,
  from = "protein",
  to = "gene",
  collapse = c("max_abs", "mean", "min_p")
)
```

Arguments

x	A structured result from aggregate_omics().
mapping	A data.frame with source_id and target_id columns.
from	Character, source feature type. Default is "protein".
to	Character, target feature type. Default is "gene".
collapse	Character, method to collapse multiple source IDs mapped to a single target ID. One of "max_abs", "mean", or "min_p".

Value

A harmonized omics_aggregated object.

mnsea *Multi-layer Network-based Gene Set Enrichment Analysis*

Description

Multi-layer Network-based Gene Set Enrichment Analysis

Usage

```
mnsea(
  seed_list,
  networks,
  couplings,
  gene_sets,
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

seed_list	named list of named numeric vectors, one per layer.
networks	named list of layer-specific networks.
couplings	data.frame of inter-layer edges.
gene_sets	list of gene sets.
mode	one of "evidence" or "signed".
layer_weights	optional named numeric vector.
collapse	one of "weighted_mean", "sum", "mean", or "max_abs".
target_layer	optional layer name to export scores from.
output_space	one of "union" or "gene".
p	restart probability.
interlayer_strength	global scaling factor for coupling edges.
specific_weight	logical.
minGSSize	minimal size of each gene set.
maxGSSize	maximal size of genes annotated for testing.
threshold	convergence threshold.
maxIter	maximal number of iterations.
verbose	logical.
...	additional arguments passed to gsea().

Value

A `mnseaResult` object. The `result` slot carries the three significance columns `pvalue`, `p.adjust`, and `qvalue`: raw nominal p-values, BH-style multiple-testing adjusted p-values (via `stats::p.adjust()`; default `pAdjustMethod = "BH"` when applicable), and Storey q-values (via `qvalue::qvalue()`). If q-value estimation fails, `qvalue` remains `NA`. See `?as_gseaResult` for the full column semantics.

`mnseaResult-class` *Class "mnseaResult" This class represents the result of multi-layer Network-based Set Enrichment Analysis.*

Description

Class "mnseaResult" This class represents the result of multi-layer Network-based Set Enrichment Analysis.

Slots

`result` enrichment analysis

`organism` organism label for the enrichment result

`setType` gene set collection type

`geneSets` gene sets

`geneList` order rank geneList

`keytype` ID type of gene

`permScores` permutation score matrix inherited from `gseaResult`

`gene2Symbol` gene ID to symbol mapping

`readable` logical flag of gene ID in symbol or not.

`termsim` Calculation matrix of termsim.

`method` Method of termsim.

`params` parameters

`dr` dimension reduction result

`multilayer_network` prepared multi-layer network object.

`layer_scores` list of layer-specific diffusion score vectors.

`collapsed_scores` numeric vector used for downstream enrichment.

`layer_weights` numeric vector of layer weights.

`coupling_table` data.frame of inter-layer couplings.

`mode` character, "evidence" or "signed".

`iterations` integer, the actual number of iterations RWR took to converge.

`restart_prob` numeric, the restart probability used in RWR.

`collapse_method` character collapse method used on layer scores.

`target_layer` optional layer name used for downstream export.

`output_space` character output space of collapsed scores.

`pathway_contribution` pathway-by-layer contribution table precomputed for explanation.

`feature_contribution` feature-by-layer contribution table precomputed for explanation.

Author(s)

Guangchuang Yu https://yulab-smu.top

`mnsea_gson` *Multi-layer NSEA using a GSON object*

Description

Multi-layer NSEA using a GSON object

Usage

```
mnsea_gson(
  seed_list,
  networks,
  couplings,
  gson,
  mode = c("evidence", "signed"),
  layer_weights = NULL,
  collapse = c("weighted_mean", "sum", "mean", "max_abs"),
  target_layer = NULL,
  output_space = c("union", "gene"),
  p = 0.5,
  interlayer_strength = 1,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  verbose = TRUE,
  ...
)
```

Arguments

`seed_list`	named list of named numeric vectors, one per layer.
`networks`	named list of layer-specific networks.
`couplings`	data.frame of inter-layer edges.
`gson`	a GSON object.
`mode`	one of "evidence" or "signed".
`layer_weights`	optional named numeric vector.
`collapse`	one of "weighted_mean", "sum", "mean", or "max_abs".
`target_layer`	optional layer name to export scores from.
`output_space`	one of "union" or "gene".

p	restart probability.
interlayer_strength	global scaling factor for coupling edges.
specific_weight	logical.
minGSSize	minimal size of each gene set.
maxGSSize	maximal size of genes annotated for testing.
threshold	convergence threshold.
maxIter	maximal number of iterations.
verbose	logical.
...	additional arguments passed to gsea_gson().

Value

A mnseaResult object. The result slot carries the three significance columns pvalue, p.adjust, and qvalue: raw nominal p-values, BH-style multiple-testing adjusted p-values (via stats::p.adjust(); default pAdjustMethod = "BH" when applicable), and Storey q-values (via qvalue::qvalue()). If q-value estimation fails, qvalue remains NA. See ?as_gseaResult for the full column semantics.

---

nsea *Network-based Gene Set Enrichment Analysis*

Description

Network-based Gene Set Enrichment Analysis

Usage

```
nsea(
  geneList,
  network,
  gene_sets,
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  significance = c("whole_pipeline", "internal"),
  nPerm = 1000,
  seed = NULL,
  exponent = 1,
  verbose = TRUE,
  ...
)
```

Arguments

geneList	named numeric vector. In "evidence" mode, must be non-negative. In "signed" mode, can contain both positive and negative values.
network	edge list (data.frame) or sparse matrix.
gene_sets	list of gene sets.
mode	character, either "evidence" (default) or "signed". If "signed", the network propagation runs separately for positive and negative values.
p	restart probability for RWR (default is 0.5).
specific_weight	logical, whether to apply gene specificity weighting (TF-IDF style) based on gene frequencies in gene_sets. Default is FALSE.
minGSSize	minimal size of each gene set for analyzing. default here is 10.
maxGSSize	maximal size of genes annotated for testing. default here is 500.
threshold	convergence threshold for RWR (default is 1e-9).
maxIter	maximal number of RWR iterations (default is 100).
significance	character, one of "whole_pipeline" (default) or "internal". "whole_pipeline" computes p-values and NES from a permutation null that re-runs the full pipeline (permute gene labels -> re-diffuse -> recompute enrichment scores) and is statistically calibrated. "internal" runs GSEA's internal label-permutation test on the diffused scores and does not model the dependence introduced by network diffusion; it is retained as the earlier formulation.
nPerm	number of whole-pipeline permutations used to estimate p-values and NES (default: 1000). The smallest estimable p-value is 1/(nPerm + 1).
seed	random seed for the permutation null (default: NULL, use the current R RNG state). Set a numeric seed for reproducible results.
exponent	weight of each step in the enrichment score (default: 1).
verbose	logical, print messages.
...	for significance = "internal", other arguments passed to gsea() (e.g. nPerm, method, nPermSimple). For significance = "whole_pipeline" additional arguments are ignored.

Value

A nseaResult object of NSEA results. The result slot carries the three significance columns pvalue, p.adjust, and qvalue: raw nominal p-values, BH-style multiple-testing adjusted p-values (via stats::p.adjust(); default pAdjustMethod = "BH"), and Storey q-values (via qvalue::qvalue()). If q-value estimation fails, qvalue remains NA. See ?as_gseaResult for the full column semantics.

`nseaResult-class` *Class "nseaResult" This class represents the result of Network-based Set Enrichment Analysis (NSEA).*

Description

Class "nseaResult" This class represents the result of Network-based Set Enrichment Analysis (NSEA).

Slots

`result` enrichment analysis

`organism` organism label for the enrichment result

`setType` gene set collection type

`geneSets` gene sets

`geneList` order rank geneList

`keytype` ID type of gene

`permScores` permutation score matrix inherited from `gseaResult`

`gene2Symbol` gene ID to symbol mapping

`readable` logical flag of gene ID in symbol or not.

`termsim` Calculation matrix of termsim.

`method` Method of termsim.

`params` parameters

`dr` dimension reduction result

`network` sparse matrix or data.frame representing the underlying network.

`diffusion_scores` numeric vector of RWR diffusion scores for each node.

`mode` character, "evidence" or "signed", describing the RWR propagation mode.

`iterations` integer, the actual number of iterations RWR took to converge.

`restart_prob` numeric, the restart probability used in RWR.

Author(s)

Guangchuang Yu https://yulab-smu.top

nsea_gson *Network-based GSEA using a GSON object*

Description

Network-based GSEA using a GSON object

Usage

```
nsea_gson(
  geneList,
  network,
  gson,
  mode = c("evidence", "signed"),
  p = 0.5,
  specific_weight = FALSE,
  minGSSize = 10,
  maxGSSize = 500,
  threshold = 1e-09,
  maxIter = 100,
  significance = c("whole_pipeline", "internal"),
  nPerm = 1000,
  seed = NULL,
  exponent = 1,
  pvalueCutoff = 0.05,
  pAdjustMethod = "BH",
  verbose = TRUE,
  ...
)
```

Arguments

geneList	named numeric vector. In "evidence" mode, must be non-negative. In "signed" mode, can contain both positive and negative values.
network	edge list (data.frame) or sparse matrix.
gson	a GSON object.
mode	character, either "evidence" (default) or "signed".
p	restart probability for RWR (default is 0.5).
specific_weight	logical, whether to apply gene specificity weighting (TF-IDF style) based on gene frequencies in the GSON object. Default is FALSE.
minGSSize	minimal size of each gene set for analyzing. default here is 10.
maxGSSize	maximal size of genes annotated for testing. default here is 500.
threshold	convergence threshold for RWR (default is 1e-9).

maxIter	maximal number of RWR iterations (default is 100).
significance	character, one of "whole_pipeline" (default) or "internal". "whole_pipeline" computes p-values and NES from a permutation null that re-runs the full pipeline (permute gene labels -> re-diffuse -> recompute enrichment scores) and is statistically calibrated. "internal" runs GSEA's internal label-permutation test on the diffused scores and does not model the dependence introduced by network diffusion; it is retained as the earlier formulation.
nPerm	number of whole-pipeline permutations used to estimate p-values and NES (default: 1000). The smallest estimable p-value is 1/(nPerm + 1).
seed	random seed for the permutation null (default: NULL, use the current R RNG state). Set a numeric seed for reproducible results.
exponent	weight of each step in the enrichment score (default: 1).
pvalueCutoff	p-value cutoff applied to both raw and adjusted p-values (default: 0.05). Set to 1 to retain all tested gene sets.
pAdjustMethod	p-value adjustment method (default: "BH").
verbose	logical, print messages.
...	for `significance = "internal"`, other arguments passed to `gsea_gson()`; ignored for `significance = "whole_pipeline"`.

Value

A `nseaResult` object. The `result` slot carries the three significance columns `pvalue`, `p.adjust`, and `qvalue`: raw nominal p-values, BH-style multiple-testing adjusted p-values (via `stats::p.adjust()`; default `pAdjustMethod = "BH"`), and Storey q-values (via `qvalue::qvalue()`). If q-value estimation fails, `qvalue` remains `NA`. See `?as_gseaResult` for the full column semantics.

`ora` *Over-Representation Analysis (ORA)*

Description

Perform over-representation analysis using hypergeometric test (Fisher's exact test).

Usage

```
ora(gene, gene_sets, universe, weight = NULL)
```

Arguments

gene	Character vector of differentially expressed genes (or gene list of interest).
gene_sets	A named list of gene sets. Each element is a character vector of genes.
universe	Character vector of background genes (e.g., all genes in the platform).
weight	A named numeric vector of weights for background genes. If provided, Weighted ORA will be performed using Wallenius' noncentral hypergeometric distribution (requires 'BiasedUrn' package). The names should match the universe genes.

Value

A data.frame with columns:

GeneSet	Gene set name
SetSize	Number of genes in the gene set (intersected with universe)
DEInSet	Number of differentially expressed genes in the gene set
DESize	Total number of differentially expressed genes in universe
PValue	Raw p-value from hypergeometric test

Examples

```
# Example data
de_genes <- c("Gene1", "Gene2", "Gene3", "Gene4", "Gene5")
all_genes <- paste0("Gene", 1:1000)

gs1 <- paste0("Gene", 1:50)
gs2 <- paste0("Gene", 51:150)
gs3 <- paste0("Gene", 151:300)
gene_sets <- list(Pathway1 = gs1, Pathway2 = gs2, Pathway3 = gs3)

result <- ora(gene=de_genes, gene_sets=gene_sets, universe=all_genes)
head(result)
```

---

ora_gson *ora-gson*

Description

interal method for enrichment analysis

Usage

```
ora_gson(
  gene,
  pvalueCutoff,
  pAdjustMethod = "BH",
  universe = NULL,
  weight = NULL,
  minGSSize = 10,
  maxGSSize = 500,
  qvalueCutoff = 0.2,
  gson
)
```

Arguments

gene	a vector of entrez gene id.
pvalueCutoff	P-value cutoff.
pAdjustMethod	P-value adjustment method (e.g., "BH").
universe	background genes, default is the intersection of the 'universe' with genes that have annotations. Users can set `options(enrichment_force_universe = TRUE)` to force the 'universe' untouched.
weight	A named numeric vector of weights for background genes. If provided, Weighted ORA will be performed.
minGSSize	minimal size of each geneSet for analyzing
maxGSSize	maximal size of each geneSet for analyzing
qvalueCutoff	cutoff of qvalue
gson	A GSON object containing gene set information.

Details

using the hypergeometric model

Value

A `enrichResult` instance. The `result` slot carries the three significance columns `pvalue`, `p.adjust`, and `qvalue`: raw nominal p-values, BH-style multiple-testing adjusted p-values (via `stats::p.adjust()`; default `pAdjustMethod = "BH"`), and Storey q-values (via `qvalue::qvalue()`). If q-value estimation fails, `qvalue` remains `NA`. See `?as_enrichResult` for the full column semantics.

Author(s)

Guangchuang Yu https://yulab-smu.top

---

prepare_multilayer_network

Prepare multi-layer network for repeated propagation

Description

Prepare multi-layer network for repeated propagation

Usage

```
prepare_multilayer_network(
  networks,
  couplings,
  directed = FALSE,
  intra_normalize = "column",
```

```
  inter_normalize = "column",
  interlayer_strength = 1,
  layer_order = names(networks)
)
```

Arguments

networks	named list of layer-specific networks.
couplings	data.frame of inter-layer edges with columns from_layer, from_id, to_layer, to_id, and optional weight.
directed	logical, whether the multi-layer graph is directed.
intra_normalize	one of "column", "row", or "none".
inter_normalize	one of "column", "row", or "none".
interlayer_strength	numeric scalar used to scale all coupling edges.
layer_order	explicit layer order. Defaults to names(networks).

Value

A multilayer_network object.

---

prepare_network *Prepare network for repeated NSEA runs*

Description

Prepare network for repeated NSEA runs

Usage

```
prepare_network(network, directed = FALSE, normalize = "column")
```

Arguments

network	edge list (data.frame with 2 or 3 columns), a sparse matrix, or a mechgraph object. For mechgraph, its edges table must have from/to columns; a numeric score or weight column, if present, is used as the edge weight, otherwise unit weights are used.
directed	logical, whether the network is directed. Default is FALSE.
normalize	one of "column", "row", or "none". Default is "column".

Value

A sparse matrix (dgCMatrix) that has been properly formatted and normalized.

`propagate_multilayer` *Propagate signals on a multi-layer network*

Description

Propagate signals on a multi-layer network

Usage

```
propagate_multilayer(
  seed_list,
  network,
  mode = c("evidence", "signed"),
  p = 0.5,
  threshold = 1e-09,
  maxIter = 100,
  layer_weights = NULL,
  target_layer = NULL
)
```

Arguments

`seed_list`	named list of named numeric vectors, one per layer.
`network`	a prepared `multilayer_network` object.
`mode`	one of "evidence" or "signed".
`p`	restart probability.
`threshold`	convergence threshold.
`maxIter`	maximum number of iterations.
`layer_weights`	optional named numeric vector of layer weights.
`target_layer`	optional layer name to focus on downstream.

Value

A `multilayer_propagation` object.

select_features_for_ora *Select features for ORA*

Description

Convert continuous aggregated statistics into a discrete list of genes and a universe for Over-Representation Analysis.

Usage

```
select_features_for_ora(x, cutoff = 0.05, by = c("pvalue", "score"), ...)
```

Arguments

x	A structured result from aggregate_omics() or harmonize_ids().
cutoff	Numeric, the threshold to apply.
by	Character, metric to apply the threshold on. One of "pvalue" or "score".
...	Additional arguments.

Value

A list containing gene (the selected feature IDs) and universe (all feature IDs).

setReadable *setReadable*

Description

mapping geneID to gene Symbol

Usage

```
setReadable(x, OrgDb, keyType = "auto", toType = "SYMBOL")
```

Arguments

x	enrichResult Object
OrgDb	OrgDb
keyType	keyType of gene
toType	ID type of the output

Value

enrichResult Object

Author(s)

Guangchuang Yu

---

show *show method*

Description

show method for gseaResult instance

show method for nseaResult instance

show method for mnseaResult instance

show method for enrichResult instance

Usage

```
show(object)

show(object)

show(object)

show(object)
```

Arguments

object	A enrichResult instance.

Value

message

message

message

message

Author(s)

Guangchuang Yu https://yulab-smu.top

`ssgsea_scores` *Single-sample GSEA scores*

Description

Score each sample independently against every gene set, giving a gene set by sample matrix that can be clustered, plotted or tested.

Usage

```
ssgsea_scores(E, gene_sets, exponent = 1, min_size = 10L, max_size = 500L)
```

Arguments

`E`	a genes by samples matrix of expression values, normalised and on a log scale. Row names are gene identifiers, column names samples.
`gene_sets`	a named list of gene identifier vectors.
`exponent`	weight applied to the rank statistic; 1 is the GSEA default, 0 gives the classic unweighted Kolmogorov-Smirnov form.
`min_size`, `max_size`	gene sets outside this range, after intersecting with `rownames(E)`, are dropped.

Details

For sample j the genes are ranked by expression and the running enrichment score of a set is accumulated over that ranking, with the same weighting as `gseaScores()`. A set's score is the integral of its running score over the ranking, normalised by the number of ranked genes. That is the ssGSEA definition, and it is computed here rather than imported so that the package needs no additional dependency for it.

Scores are comparable ACROSS SAMPLES but not across gene sets, because the scale is set by each set's own membership. Scale within gene set before any cross-set comparison.

The caller supplies already-transformed values; `dsea_scores()` does that step and records what it did.

Value

A gene set by sample matrix of scores.

See Also

`dsea_scores`, `gseaScores`

summary *summary method*

Description

summary method for gseaResult instance

summary method for enrichResult instance

Usage

```
summary(object, ...)

summary(object, ...)
```

Arguments

object	A enrichResult instance.
...	additional parameter

Value

A data frame

A data frame

Author(s)

Guangchuang Yu https://yulab-smu.top

Index

∗ **classes**
  compareClusterResult-class, 13
  enrichResult-class, 16
  gseaResult-class, 23
∗ **manip**
  ora_gson, 36

aggregate_enrichment, 3
aggregate_omics, 3
as_enrichResult, 4
as_genes, 6
as_gseaResult, 7
as_ranked, 9

bayes_enrich, 10
bayes_summary, 11

classify_omics_pattern, 12
collapse_multilayer_scores, 13
compareClusterResult-class, 13

DESeq2::results(), *7*, *9*
DESeqDataSet, *6*, *9*
DESeqResults, *6*, *9*
dsea, 14
dsea_scores, *42*
dsea_scores *(*dsea*)*, 14

enrichResult, *14*
enrichResult-class, 16
EXTID2NAME, 17
extract_mnsea_subnetwork, 18

geneID, 18
geneInCategory, 19
get_mnsea_contribution, 20
get_omics_contribution, 20
gsea, *9*, *16*, 21
gsea(), *15*, *16*
gsea_gson, 24
gseahResult-class *(*gseaResult-class*)*, 23
gseaResult-class, 23
gseaScores, 24, *42*
gseaScores(), *42*
gsfilter, 26

harmonize_ids, 27

mnsea, 27
mnsea_gson, 30
mnseaResult-class, 29

nsea, 31
nsea_gson, 34
nseaResult-class, 33

ora, *6*, 35
ora_gson, 36

plot,compareClusterResult-method
  *(*compareClusterResult-class*)*, 13
prepare_multilayer_network, 37
prepare_network, 38
propagate_multilayer, 39

select_features_for_ora, 40
setReadable, 40
show, 41
show,compareClusterResult-method
  *(*compareClusterResult-class*)*, 13
show,enrichResult-method
  *(*enrichResult-class*)*, 16
show,gseaResult-method
  *(*gseaResult-class*)*, 23
show,mnseaResult-method *(*show*)*, 41
show,nseaResult-method *(*show*)*, 41
ssgsea_scores, *16*, 42
ssgsea_scores(), *15*
summary, 43

summary,compareClusterResult-method
(compareClusterResult-class), 13
summary,enrichResult-method
(enrichResult-class), 16
summary,gseaResult-method
(gseaResult-class), 23