Package 'DSGEr'

June 30, 2026

Type Package

Title Transcriptional Disruption Score for Gene Sets

Version 1.3.1

Description Computes pathway-level transcriptional disruption scores from differential expression p-values using permutation tests with Generalized Pareto Distribution (GPD) tail extrapolation and false discovery rate (FDR) correction. Reference: Guo P, Li H (2026) DSGE: Disruption Score of Gene Expression for Gene-Set Enrichment Analysis. <https://github.com/LHJLab/DSGE>.

License MIT + file LICENSE

Encoding UTF-8

Imports data.table, evd, methods, POT, Rcpp

LinkingTo Rcpp

Suggests AnnotationDbi, GO.db

Enhances KEGGREST, org.Hs.eg.db, reactome.db

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation yes

Author Pingwei Guo [aut, cre],
Huijuan Li [aut]

Maintainer Pingwei Guo <1489413126@qq.com>

Repository CRAN

Date/Publication 2026-06-30 11:00:21 UTC

Contents

`calc_dsge` *Compute DSGE (Disruption Score of Gene Expression)*

Description

Computes the mean absolute z-score for all genes passing filters from differential expression analysis results. Higher DSGE indicates stronger global transcriptional perturbation.

Usage

```
calc_dsge(pvalue, base_mean = NULL, base_mean_cutoff = 0.1)
```

Arguments

`pvalue`	Numeric vector of p-values from differential expression analysis (e.g., DESeq2, edgeR, Seurat FindMarkers).
`base_mean`	Numeric vector of mean expression values (same length as pvalue), e.g. DESeq2 baseMean or Seurat avg_log2FC corresponding expression level. If NULL, no expression filtering is applied.
`base_mean_cutoff`	Expression filter threshold, default 0.1. Genes with mean expression at or below this value are excluded as lowly expressed. Ignored when base_mean = NULL.

Value

A list with elements:

`dsge`	scalar, genome-wide DSGE
`n_genes`	integer, number of genes passing filters
`z_scores`	named numeric vector of per-gene raw z-scores

Examples

```
# Generate random p-values (simulating differential expression results)
set.seed(42)
pvals <- runif(1000)
z <- calc_dsge(pvals)
head(z)

# With baseMean filtering
base_mean <- rexp(1000)
z <- calc_dsge(pvals, base_mean)
head(z)
```

`dsge_perm_test` *Permutation test for DSGE enrichment of a single gene set*

Description

Tests whether a target gene set has a significantly higher DSGE than expected by chance. Generates a null distribution via sampling without replacement and computes an empirical right-tail p-value.

Usage

```
dsge_perm_test(
  gene_list,
  pvalue,
  base_mean,
  gene_names,
  base_mean_cutoff = 0.1,
  n_perm = 10000L,
  seed = NULL,
  progress = TRUE,
  heterogeneity = FALSE,
  directional = FALSE,
  direction_vec = NULL,
  use_std = TRUE,
  use_gpd = TRUE,
  gpd_threshold = 0.99,
  gpd_method = "mle",
  safety_margin = 1.6,
  nds_top_frac = 0.25
)
```

Arguments

`gene_list`	Character vector of target gene identifiers (must match values in gene_names).
`pvalue`	Numeric vector of p-values from differential expression analysis (e.g., DESeq2, edgeR, Seurat FindMarkers) for all genes in the background pool.

base_mean Numeric vector of mean expression values, same length as pvalue. Pass NULL to skip expression-level filtering.

gene_names Character vector of gene names, same length as pvalue, must be unique.

base_mean_cutoff
baseMean filter threshold, default 0.1.

n_perm Number of permutations, default 10000.

seed Optional integer random seed for reproducibility.

progress Whether to show a progress bar, default TRUE.

heterogeneity Whether to compute perturbation heterogeneity indices (Gini, CV) and two-sided p-values. Default FALSE. When enabled, also computes Gini and CV null distributions within the permutation loop, increasing runtime by approximately 30%-50%.

directional Whether to compute Normalized Direction Score (NDS) for each pathway. Default FALSE. When TRUE, direction_vec must be provided. NDS ranges from -1 to 1: positive values indicate net up-regulation, negative values net down-regulation. Requires use_gpd = TRUE (default) for reliable extreme p-values alongside directional information.

direction_vec Numeric vector of direction indicators (e.g., log fold changes), same length as pvalue. Used to classify genes as up-regulated (positive direction) or down-regulated (negative direction). Values of exactly 0 are treated as up-regulated. Ignored when directional = FALSE.

use_std Whether to compute and use standardised DSGE. Default TRUE. When enabled:

- The returned list includes dsge_std = (observed - mean(null)) / sd(null).
- p-value is computed from the standardised null distribution (mean(null_std >= dsge_std)).

When FALSE, p-value is computed from the raw null distribution.

use_gpd Whether to use GPD tail extrapolation for extreme p-values. Default TRUE. When TRUE and the observed DSGE exceeds the gpd_threshold quantile of the null, the p-value is extrapolated via a fitted Generalized Pareto Distribution with support-constrained adjustment (Peschel et al. 2025, arXiv:2602.22975) ensuring a non-zero p-value. When FALSE, always uses empirical ECDF (p-value always >= 1/n_perm).

gpd_threshold Tail quantile threshold for GPD fitting, between 0 and 1. Default 0.99. Lower = more tail samples (lower variance, higher bias); higher = fewer samples (higher variance, lower bias).

gpd_method GPD estimation method passed to POT::fitgpd. Default "mle" (maximum likelihood). Other options: "mple", "moments", "pwmu", "pwmb", "mdpd", "med", "pickands", "lme", "mgf".

safety_margin Safety margin for GPD support-constrained adjustment. Default 1.6. Larger values produce larger (more conservative) p-values for extremely extreme observations, avoiding double-precision underflow to zero at the cost of increased bias.

nds_top_frac Fraction of most-perturbed genes retained for NDS calculation. Default 0.25 (top 25%). Only used when directional = TRUE. Lower values focus on the most extreme genes; higher values include more genes.

Details

Algorithm steps:

1. Filter gene pool: baseMean > cutoff, non-missing p-value.
2. Convert p-values to per-gene raw z-scores.
3. Match `gene_list` to the filtered gene pool; compute observed DSGE (mean z-score).
4. Generate null distribution: randomly sample (without replacement) equally-sized gene sets from the pool, computing random DSGE via the same mean z-score formula, repeated n_perm times.
5. Empirical right-tail p-value: count(DSGE_null > DSGE_obs) / n_perm.

Value

A list with elements:

`observed`	observed DSGE value
`n_genes`	number of target genes matched
`null`	permutation null distribution vector
`p_value`	right-tail p-value (GPD tail extrapolation when observed falls above the 'gpd_threshold' quantile (default 99th); empirical ECDF otherwise)
`ecdf`	empirical cumulative distribution function of the null
`dsge_std`	(only when `use_std = TRUE`) standardised DSGE
`nds`	(only when `directional = TRUE`) Normalized Direction Score, ranging from -1 (pure down) to +1 (pure up)
`gini_observed`	(only when `heterogeneity = TRUE`) observed Gini coefficient
`cv_observed`	(only when `heterogeneity = TRUE`) observed CV
`null_gini`	(only when `heterogeneity = TRUE`) Gini null distribution vector
`null_cv`	(only when `heterogeneity = TRUE`) CV null distribution vector
`het_p_value`	(only when `heterogeneity = TRUE`) two-sided permutation p-value based on Gini

Examples

```
# Toy example with simulated data
set.seed(42)
pvals <- runif(500)
base_mean <- rexp(500)
gene_names <- paste0("gene", 1:500)
forebrain_like <- paste0("gene", 1:30)
dsge_perm_test(forebrain_like, pvals, base_mean, gene_names,
               n_perm = 100, seed = 42, progress = FALSE)
```

get_gaf_header *Extract header lines from a GAF file*

Description

Lines starting with ! in GAF files contain metadata (database version, date, column definitions, etc.). This function extracts them for inspecting file version and provenance.

Usage

```
get_gaf_header(file)
```

Arguments

file	Path to the GAF file.

Value

Character vector, one string per header line (with leading ! removed).

Examples

```
# Create a temporary GAF file for demonstration
gaf_file <- tempfile(fileext = ".gaf")
writeLines(c(
  "!gaf-version: 2.2",
  "!generated-by: R example",
  paste0("UniProtKB\tP12345\tGENE_A\t\tGO:0003674\t",
         "PMID:123456\tIDA\t\tF\tGene A\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t"),
  paste0("UniProtKB\tP67890\tGENE_B\t\tGO:0005575\t",
         "PMID:789012\tIBA\t\tC\tGene B\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t")
), gaf_file)
get_gaf_header(gaf_file)
```

get_pathway_genes *Extract gene-pathway (GO term) associations from GAF data*

Description

Splits the parsed GAF data.frame returned by read_gaf() into a named list by GO term. Supports filtering by qualifier, evidence code, ontology aspect, and minimum pathway size. Optionally attaches GO term names from an OBO file.

Usage

```
get_pathway_genes(
  gaf_data,
  genes = c("db_object_id", "db_object_symbol"),
  unique = TRUE,
  min_size = 5L,
  qualifier = NULL,
  evidence = NULL,
  aspect = NULL,
  go_names = NULL
)
```

Arguments

gaf_data	A data.frame returned by read_gaf().
genes	Character vector of column names identifying genes. Default c("db_object_id", "db_object_symbol"). The first is typically a UniProt ID, the second a gene symbol.
unique	Logical. Whether to remove duplicate gene entries within a GO term. Default TRUE (the same gene may be annotated to the same term via multiple evidence lines).
min_size	Minimum gene count threshold; pathways with fewer genes are discarded. Default 5. Set to 5 to ensure sufficient statistical power for the permutation test.
qualifier	Qualifier filter, e.g. "enables", "involved_in", "located_in". NULL (default) returns all. A common combination is c("enables","involved_in") (excludes NOT annotations and located_in annotations).
evidence	Evidence code filter, e.g. "IDA" (direct experimental assay), "IEA" (electronic annotation). NULL (default) returns all. For high-confidence manual annotations only, use e.g. c("IDA", "IPI", "IMP", "IGI","IEP").
aspect	Ontology aspect filter: "F" (Molecular Function), "P" (Biological Process), "C" (Cellular Component). NULL (default) returns all three.
go_names	A data.frame returned by read_obo(). When provided, each pathway data.frame gains go_name and go_namespace columns (placed in the first two positions). go_namespace uses abbreviated forms: "BP", "MF", "CC".

Value

A named list where each element name is a GO term ID (e.g. "GO:0005515") and the value is a data.frame of associated genes. The list is sorted alphabetically by GO ID.

Examples

```
# Create a minimal GAF data.frame for demonstration
toy_gaf <- data.frame(
  db_object_id    = c("1", "2", "3", "1", "2"),
  db_object_symbol = c("GENE_A", "GENE_B", "GENE_C", "GENE_A", "GENE_D"),
  go_id           = c("GO:0003674", "GO:0003674", "GO:0005575",
```

```
                      "GO:0005575", "GO:0005575"),
  aspect          = c("F", "F", "C", "C", "C"),
  evidence_code   = c("IDA", "IBA", "IDA", "IDA", "IEA"),
  stringsAsFactors = FALSE
)
pathway_genes <- get_pathway_genes(toy_gaf, min_size = 1)
pathway_genes[["GO:0003674"]]

# Create a temporary GAF file for demonstration
gaf_file <- tempfile(fileext = ".gaf")
writeLines(c(
  "!gaf-version: 2.2",
  paste0("UniProtKB\tP12345\tGENE_A\t\tGO:0003674\t",
         "PMID:123456\tIDA\t\tF\tGene A\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t"),
  paste0("UniProtKB\tP67890\tGENE_B\t\tGO:0005575\t",
         "PMID:789012\tIBA\t\tC\tGene B\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t")
), gaf_file)
gaf <- read_gaf(gaf_file)

# Create a temporary OBO file
obo_file <- tempfile(fileext = ".obo")
writeLines(c(
  "format-version: 1.2",
  "",
  "[Term]",
  "id: GO:0003674",
  "name: molecular_function",
  "namespace: molecular_function",
  "",
  "[Term]",
  "id: GO:0005575",
  "name: cellular_component",
  "namespace: cellular_component"
), obo_file)
go <- read_obo(obo_file)

# Build pathway-gene mapping with GO names
pathway_genes <- get_pathway_genes(gaf, go_names = go, min_size = 1)
pathway_genes[["GO:0003674"]]
```

`get_pathway_genes_db` *Build pathway-gene mapping from a Bioconductor OrgDb object*

Description

Extracts GO term to gene mappings from an `OrgDb` annotation package (e.g. `org.Hs.eg.db`) or an `AnnotationHub` record. Returns a named list in the same format as `get_pathway_genes()`, ready to pass to `pathway_dsge()`.

Usage

```
get_pathway_genes_db(
  orgdb,
  keytype = "ENTREZID",
  gene_id_col = "db_object_id",
  gene_symbol_col = "db_object_symbol",
  min_size = 5L,
  aspect = NULL,
  evidence = NULL,
  attach_go_names = TRUE,
  use_goall = FALSE
)
```

Arguments

orgdb	An OrgDb object, e.g. org.Hs.eg.db for human, org.Mm.eg.db for mouse, or a similar object retrieved from AnnotationHub.
keytype	Key type used to query the OrgDb. Default "ENTREZID". Must be one of the key types returned by keytypes(orgdb).
gene_id_col	Name for the gene ID column in the output data.frames. Default "db_object_id" (matching the get_pathway_genes() convention; the values are Entrez IDs by default).
gene_symbol_col	Name for the gene symbol column in the output data.frames. Default "db_object_symbol" (matching the get_pathway_genes() convention).
min_size	Minimum gene count per pathway; pathways below this are discarded. Default 5. Pass NULL to keep all.
aspect	Ontology aspect filter. NULL (default) returns all. One or more of "BP" (Biological Process), "MF" (Molecular Function), "CC" (Cellular Component).
evidence	Evidence code filter (e.g. "IDA", "IEA"). NULL (default) keeps all. Pass a character vector to keep only annotations with those evidence codes.
attach_go_names	Logical. Whether to fetch GO term names and ontology classifications via GO.db. Default TRUE. Requires the GO.db package to be installed.
use_goall	Logical. If TRUE, query the GOALL column instead of GO, propagating gene annotations to all ancestor terms along the GO directed acyclic graph. This produces a broader pathway set consistent with clusterProfiler's gseGO/enrichGO default behaviour. Default FALSE uses direct annotations only.

Details

This function provides an alternative to get_pathway_genes() for users who prefer Bioconductor's annotation infrastructure over GAF + OBO files.

Value

A named list where each element is a `data.frame` with columns `go_name` (if attached), `go_namespace` (if attached), `gene_id_col`, and `gene_symbol_col`. The list names are GO term IDs. This matches the output format of `get_pathway_genes()` and can be passed directly to `pathway_dsge()`.

Note

This function requires the `AnnotationDbi` package. The `GO.db` package is required when `attach_go_names = TRUE` (the default). Both are Bioconductor packages; install with `BiocManager::install(c("AnnotationDbi", "GO.db"))`.

See Also

`get_pathway_genes` for the GAF-based equivalent.

Examples

```
# Requires additional Bioconductor database packages

if (require("org.Hs.eg.db", quietly = TRUE)) {
  pw <- get_pathway_genes_db(org.Hs.eg.db)
  str(pw[1:2])
}
```

---

get_pathway_genes_kegg

Build KEGG pathway-gene mapping from a Bioconductor OrgDb object

---

Description

Extracts KEGG pathway to gene mappings from an `OrgDb` annotation package (e.g. `org.Hs.eg.db`) or an `AnnotationHub` record. Returns a named list in the same format as `get_pathway_genes()`, ready to pass to `pathway_dsge()`.

Usage

```
get_pathway_genes_kegg(
  orgdb,
  keytype = "ENTREZID",
  gene_id_col = "kegg_gene_id",
  gene_symbol_col = "kegg_gene_symbol",
  min_size = 5L,
  attach_path_names = TRUE
)
```

Arguments

orgdb	An OrgDb object, e.g. org.Hs.eg.db for human, org.Mm.eg.db for mouse, or a similar object retrieved from AnnotationHub.
keytype	Key type used to query the OrgDb. Default "ENTREZID". Must be one of the key types returned by keytypes(orgdb).
gene_id_col	Name for the gene ID column in the output data.frames. Default "kegg_gene_id".
gene_symbol_col	Name for the gene symbol column in the output data.frames. Default "kegg_gene_symbol".
min_size	Minimum gene count per pathway; pathways below this are discarded. Default 5. Pass NULL to keep all.
attach_path_names	Logical. Whether to fetch pathway names via KEGGREST. Default TRUE. When KEGGREST is not available or network access fails, falls back to NA names.

Details

KEGG pathway IDs are stored in the PATH column of OrgDb objects. IDs include an organism prefix (e.g. "hsa00010" for human, "mmu00010" for mouse). The organism code is auto-detected from the OrgDb species and a built-in lookup table of 15 common model organisms.

Pathway names are fetched online via KEGGREST::keggList(). When the KEGGREST package is not available or network access fails, names are set to NA with a warning.

Value

A named list where each element is a data.frame with columns kegg_name (if attached), organism_code, gene_id_col, and gene_symbol_col. The list names are full KEGG pathway IDs including the organism prefix (e.g. "hsa00010"). S3 class "kegg_pathway" is set on list elements for source auto-detection by pathway_dsge().

Note

Requires the AnnotationDbi package. The KEGGREST package is required only when attach_path_names = TRUE. Install with: BiocManager::install(c("AnnotationDbi", "KEGGREST")).

See Also

get_pathway_genes_db for the GO-based equivalent. get_pathway_genes_reactome for the Reactome equivalent.

Examples

```
# Requires additional Bioconductor database packages

if (require("org.Hs.eg.db", quietly = TRUE)) {
  pw <- get_pathway_genes_kegg(org.Hs.eg.db)
  str(pw[1:2])
}
```

`get_pathway_genes_reactome` *Build Reactome pathway-gene mapping from a Bioconductor OrgDb object*

Description

Extracts Reactome pathway to gene mappings, using `reactome.db` for pathway-to-gene associations and an `OrgDb` for gene symbol resolution. Returns a named list ready to pass to `pathway_dsge()`.

Usage

```
get_pathway_genes_reactome(
  orgdb,
  keytype = "ENTREZID",
  gene_id_col = "reactome_gene_id",
  gene_symbol_col = "reactome_gene_symbol",
  min_size = 5L,
  attach_path_names = TRUE,
  species_prefix = "R-HSA"
)
```

Arguments

`orgdb`	An `OrgDb` object (e.g. `org.Hs.eg.db`) used to resolve Entrez IDs to gene symbols. Only human (R-HSA) pathways are included.
`keytype`	Key type used to query the `OrgDb`. Default `"ENTREZID"`. Must be one of the key types returned by `keytypes(orgdb)`.
`gene_id_col`	Name for the gene ID column in the output data.frames. Default `"reactome_gene_id"`.
`gene_symbol_col`	Name for the gene symbol column in the output data.frames. Default `"reactome_gene_symbol"`.
`min_size`	Minimum gene count per pathway; pathways below this are discarded. Default `5`. Pass `NULL` to keep all.
`attach_path_names`	Logical. Whether to fetch pathway names via `reactome.db`. Default `TRUE`. When `reactome.db` is not installed, falls back to `NA` names.
`species_prefix`	Reactome species prefix for filtering pathways. Default `"R-HSA"` (Homo sapiens). Use `NULL` to keep all species.

Details

Reactome pathway IDs follow the pattern `R-HSA-<number>` (e.g. `R-HSA-177929`). Pathway names are fetched locally from the `reactome.db` Bioconductor package in a single batched query.

Value

A named list where each element is a `data.frame` with columns `reactome_name` (if attached), `gene_id_col`, and `gene_symbol_col`. The list names are Reactome pathway IDs (e.g. `"R-HSA-177929"`). S3 class `"reactome_pathway"` is set on list elements for source auto-detection by `pathway_dsge()`.

Note

Requires `reactome.db` and `AnnotationDbi`. Install with: `BiocManager::install(c("AnnotationDbi", "reactome.db"))`.

See Also

`get_pathway_genes_db` for the GO-based equivalent. `get_pathway_genes_kegg` for the KEGG equivalent.

Examples

```
# Requires additional Bioconductor database packages

if (require("org.Hs.eg.db", quietly = TRUE) &&
    require("reactome.db", quietly = TRUE)) {
  pw <- get_pathway_genes_reactome(org.Hs.eg.db)
  str(pw[1:2])
}
```

`pathway_dsge` *Pathway-level DSGE permutation test with FDR correction*

Description

Computes DSGE for each GO pathway, generates permutation null distributions grouped by number of matched genes, computes p-values using GPD tail extrapolation, and applies Benjamini-Hochberg FDR correction.

Usage

```
pathway_dsge(
  pathway_genes,
  pvalue,
  base_mean = NULL,
  gene_names,
  gene_id_col = NULL,
  source = NULL,
  base_mean_cutoff = 0.1,
  min_size = 5L,
  max_size = 500L,
  n_perm = 10000L,
```

```
  seed = NULL,
  return_null = FALSE,
  progress = TRUE,
  heterogeneity = FALSE,
  directional = FALSE,
  direction_vec = NULL,
  use_std = TRUE,
  use_gpd = TRUE,
  gpd_threshold = 0.99,
  gpd_method = "mle",
  safety_margin = 1.6,
  n_cores = 1L,
  dsge_std = NULL,
  p_adjust_method = "BY",
  nds_top_frac = 0.25
)
```

Arguments

pathway_genes	Named list returned by get_pathway_genes(). Each element is a data.frame containing gene information for one pathway.
pvalue	Numeric vector of p-values from differential expression analysis (e.g., DESeq2, edgeR, Seurat FindMarkers) for all genes in the background pool.
base_mean	Numeric vector of mean expression values, same length as pvalue. Pass NULL to skip expression-level filtering.
gene_names	Character vector of gene names (same length as pvalue), must be unique.
gene_id_col	Column name in pathway_genes data.frames used to match gene names. When NULL (the default), auto-detected from the pathway source: "db_object_symbol" for GO, "kegg_gene_symbol" for KEGG, "reactome_gene_symbol" for REACTOME.
source	Pathway source. One of "GO", "KEGG", or "REACTOME". When NULL (the default), auto-detected from the S3 class of pathway_genes elements ("kegg_pathway" or "reactome_pathway"). Falls back to "GO" for unclassed data.frames.
base_mean_cutoff	baseMean filter threshold, default 0.1.
min_size	Minimum number of matched genes; pathways below this are not tested. Default 5.
max_size	Maximum number of matched genes; pathways above this are excluded. Default 500. Set to Inf to retain all.
n_perm	Number of permutations per size group, default 10000.
seed	Optional integer random seed.
return_null	Whether to include null distribution data in the result (for plotting). Default FALSE. When TRUE, returns a list instead of a data.frame.
progress	Whether to show a progress bar, default TRUE.

`heterogeneity` Whether to compute perturbation heterogeneity indices (Gini, CV) and two-sided p-values. Default `FALSE`. When enabled, additionally computes Gini and CV during null distribution generation, increasing runtime by approximately 30%-50%. The result data.frame will include extra columns `gini`, `cv`, `het_p_value`, `het_p_adj`.

`directional` Whether to compute Normalized Direction Score (NDS) for each pathway. Default `FALSE`. When `TRUE`, `direction_vec` must be provided. NDS ranges from -1 to 1: positive values indicate net up-regulation, negative values net down-regulation. The result data.frame will include an `nds` column.

`direction_vec` Numeric vector of direction indicators (e.g., log fold changes), same length as `pvalue`. Used to classify genes as up-regulated (positive direction) or down-regulated (negative direction). Values of exactly 0 are treated as up-regulated. Ignored when `directional = FALSE`.

`use_std` Whether to compute and use standardised DSGE. Default `TRUE`. When enabled:

- `dsge_std = (observed - mean(null)) / sd(null)` using the size-grouped null distribution is added as a result column.
- p-values are computed from the standardised null distribution (`mean(null_std >= dsge_std)`), without GPD tail extrapolation.

When `FALSE`, p-values are computed from the raw null distribution (using GPD when `use_gpd = TRUE`).

`use_gpd` Whether to use GPD tail extrapolation for extreme p-values. Default `TRUE`. When `TRUE` and the observed DSGE exceeds the `gpd_threshold` quantile of the null, the p-value is extrapolated via a fitted Generalized Pareto Distribution with support-constrained adjustment (Peschel et al. 2025, arXiv:2602.22975) ensuring a non-zero p-value. When `FALSE`, always uses empirical ECDF (p-value always `>= 1/n_perm`).

`gpd_threshold` Tail quantile threshold for GPD fitting, between 0 and 1. Default 0.99. Lower = more tail samples (lower variance, higher bias); higher = fewer samples (higher variance, lower bias).

`gpd_method` GPD estimation method passed to `POT::fitgpd`. Default `"mle"` (maximum likelihood). Other options: `"mple"`, `"moments"`, `"pwmu"`, `"pwmb"`, `"mdpd"`, `"med"`, `"pickands"`, `"lme"`, `"mgf"`.

`safety_margin` Safety margin for GPD support-constrained adjustment. Default `1.6`. Larger values produce larger (more conservative) p-values for extremely extreme observations, avoiding double-precision underflow to zero at the cost of increased bias.

`n_cores` Number of CPU cores for parallel null distribution generation. Default `1` (sequential). Set to `parallel::detectCores()` to use all available cores. Uses `parallel::mclapply` on Unix/macOS (only effective when multiple unique pathway sizes exist). On Windows, falls back to sequential with a message.

`dsge_std` **[Deprecated]**. Use `use_std` instead.

`p_adjust_method`

Multiple testing correction method. Passed to `stats::p.adjust()`. Default `"BY"` (Benjamini-Yekutieli) which controls FDR under arbitrary dependence. Use `"BH"` for Benjamini-Hochberg (controls FDR under positive dependence).

All methods supported by `p.adjust` are valid: `"holm"`, `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"`, `"BY"`, `"fdr"`, `"none"`.

`nds_top_frac` Fraction of most-perturbed genes retained for NDS calculation. Default `0.25` (top 25%). Only used when `directional = TRUE`. Lower values focus on the most extreme genes; higher values include more genes.

Details

Pathways sharing the same number of matched genes reuse the same null distribution, greatly reducing computation.

Algorithm steps (9 steps)

1. **Build gene pool**: filter DESeq2 results (baseMean > cutoff), compute per-gene raw z-scores.
2. **Match pathway genes**: match each pathway's genes to the gene pool via the `gene_id_col` column.
3. **Filter pathways**: retain pathways with `min_size` <= `n_matched` <= `max_size`.
4. **Compute observed DSGE**: per pathway as the mean of member gene z-scores: DSGE = mean(z_i).
5. **Generate size-grouped null distributions**: each unique matched-gene count gets its own permutation run (vectorized batch sampling), with simultaneous GPD tail fitting.
6. **Compute standardised DSGE** (step 6): when `use_std = TRUE`, compute `dsge_std = (observed - mean(null)) / sd(null)` for each pathway, using the size-grouped null distribution.
7. **Compute p-values**: when `use_std = TRUE`, empirical ECDF of the standardised null vs. dsge_std; when `use_std = FALSE`, GPD tail extrapolation (above the `gpd_threshold` quantile, default 99th) + empirical ECDF on the raw null distribution.
8. **BH FDR correction**: Benjamini-Hochberg multiple testing correction on all pathway p-values.
9. **Sort and return**: ordered by p_adj ascending.

Value

By default, a `data.frame` sorted by `p_adj` ascending. The pathway ID and name columns depend on the source:

- For GO: `go_id`, `go_name`, `aspect`
- For KEGG: `kegg_id`, `kegg_name`
- For REACTOME: `reactome_id`, `reactome_name`

All sources include: `n_pathway`, `n_matched`, `dsge`, `p_value`, `p_adj`.

For GO sources, `aspect` is the ontology classification: `"BP"` (Biological Process), `"MF"` (Molecular Function), `"CC"` (Cellular Component).

When `heterogeneity = TRUE`, additionally includes: `gini`, `cv`, `het_p_value`, `het_p_adj`.

When `directional = TRUE`, additionally includes: `nds` (Normalized Direction Score, ranging from -1 to +1).

When `use_std = TRUE` (default), additionally includes: `dsge_std` = (observed DSGE - mean(null)) / sd(null), standardised using the size-grouped null distribution.

When `return_null = TRUE`, returns a `list` with:

table	the data.frame described above
null_raw	named list, keys are pathway sizes (as character), values are DSGE null distribution vectors
null_gpd	named list, keys are pathway sizes, values are GPD fit parameters (or NULL)

When both heterogeneity = TRUE and return_null = TRUE, the list also includes:

null_gini_raw	named list, keys are pathway sizes, values are Gini null distribution vectors
null_cv_raw	named list, keys are pathway sizes, values are CV null distribution vectors

Examples

```
# Toy example with simulated data
set.seed(42)
pvals <- runif(500)
base_mean <- rexp(500)
gene_names <- paste0("gene", 1:500)
pw <- list(
  pathway_A = data.frame(db_object_symbol = paste0("gene", 1:20),
                         go_name = "Pathway A",
                         stringsAsFactors = FALSE),
  pathway_B = data.frame(db_object_symbol = paste0("gene", 21:40),
                         go_name = "Pathway B",
                         stringsAsFactors = FALSE)
)
result <- pathway_dsge(pw, pvals, base_mean, gene_names,
                       min_size = 1, n_perm = 100, seed = 42)
head(result)
```

---

plot_dsge *Plot null distribution vs. observed DSGE for selected pathways*

Description

For pathways in a pathway_dsge() result, plots the density of the permutation null distribution with a dashed red line marking the observed DSGE. If GPD tail fitting is available for the size group, the tail region is highlighted in semi-transparent orange.

Usage

```
plot_dsge(
  result,
  n = 9L,
  pathway_ids = NULL,
  go_ids = NULL,
  col_null = "#2166AC",
  col_tail = "#E41A1C",
  col_obs = "#D73027",
```

```
  col_thr = "#999999",
  safety_margin = NULL,
  cex_main = 0.85,
  use_std = NULL
)
```

Arguments

result	List returned by pathway_dsge() with return_null = TRUE (contains table, null_raw, null_gpd).
n	When pathway_ids (or go_ids) is NULL, the top n most significant pathways (by p_adj ascending) are plotted. Default 9. Ignored when pathway_ids is specified.
pathway_ids	Optional character vector of pathway IDs to plot (e.g. GO IDs, KEGG IDs, or Reactome IDs depending on source). When provided, directly plots these pathways, overriding n. Unmatched IDs are skipped with a warning.
go_ids	**[Deprecated]**. Use pathway_ids instead.
col_null	Color of the null distribution density curve. Default "#2166AC".
col_tail	Color of the GPD tail region highlight. Default "#E41A1C".
col_obs	Color of the observed DSGE vertical line. Default "#D73027".
col_thr	Color of the GPD threshold vertical line. Default "#999999".
safety_margin	Safety margin for visual GPD tail extension in the plot. Defaults to the value from pathway_dsge() result.
cex_main	Title font scaling. Default 0.85.
use_std	Whether to plot standardised DSGE. Defaults to TRUE when the result table contains a dsge_std column (i.e. pathway_dsge() was run with use_std = TRUE), and FALSE otherwise.

Value

No return value; called for its side effect (plotting).

Examples

```
# Build a result object with simulated data for demonstration
set.seed(42)
pvals <- runif(500)
base_mean <- rexp(500)
gene_names <- paste0("gene", 1:500)
pw <- list(
  pw1 = data.frame(db_object_symbol = paste0("gene", 1:30),
                   go_name = "Pathway 1",
                   stringsAsFactors = FALSE),
  pw2 = data.frame(db_object_symbol = paste0("gene", 31:60),
                   go_name = "Pathway 2",
                   stringsAsFactors = FALSE)
)
```

```
result <- pathway_dsge(pw, pvals, base_mean, gene_names,
                       min_size = 1, n_perm = 100, seed = 42,
                       return_null = TRUE)
plot_dsge(result, n = 2)
```

`plot_dsge_volcano` *Gene-level volcano plot for a specific pathway*

Description

Plots a focused volcano plot showing only the genes belonging to a specified GO pathway. Each point is one gene from the pathway, with log2 fold change on the x-axis and statistical significance (-log10 p-value) on the y-axis. This allows visual inspection of the direction, magnitude, and distribution of perturbation within a single pathway.

Usage

```
plot_dsge_volcano(
  de_results,
  dsge_result = NULL,
  pathway_genes,
  go_id,
  logFC_col = "log2FoldChange",
  pval_col = "pvalue",
  padj_col = NULL,
  gene_col = "geneName",
  gene_id_col = "db_object_symbol",
  threshold = 0.05,
  padj_threshold = 0.05,
  lfc_threshold = NULL,
  color_up = "#CC3333",
  color_down = "#3366CC",
  color_nom = "#CC9933",
  color_ns = "#AAAAAA",
  alpha_sig = 0.9,
  alpha_ns = 0.5,
  label = NULL,
  label_genes = NULL,
  label_sig = FALSE,
  cex_label = 0.7,
  cex_point = 1.6,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  go_name = NULL,
  ...
)
```

Arguments

de_results	Data.frame of differential expression results. Must contain at least log2FC, p-value, and gene identifier columns.
dsge_result	Optional result from pathway_dsge(). Pass either the full list (where dsge_result$table is used) or the result data.frame directly. When provided, the pathway's dsge_std and p_adj are shown in the annotation. Default NULL.
pathway_genes	A named list of pathway-gene mappings, as used in pathway_dsge(). Each element is a data.frame with a gene identifier column.
go_id	A single character string specifying which pathway (GO ID) to plot. Must be a name in pathway_genes.
logFC_col	Column name in de_results for log2 fold change. Default "log2FoldChange".
pval_col	Column name in de_results for the p-value (raw or adjusted). Default "pvalue".
padj_col	Optional column name in de_results for adjusted p-values (e.g. "padj"). When provided, points are classified as adjusted-significant using padj_threshold, and an additional "Nominal" category appears in the legend. Default NULL (uses threshold only).
gene_col	Column name in de_results for gene identifiers that match the pathway mapping. Default "geneName".
gene_id_col	Column name in pathway_genes[[go_id]] that holds the gene identifiers. Default "db_object_symbol".
threshold	p-value significance threshold for the horizontal reference line. Default 0.05.
padj_threshold	Adjusted p-value threshold for significance classification. Default 0.05. Only used when padj_col is provided.
lfc_threshold	Numeric vector of logFC thresholds for vertical reference lines (e.g. c(-1, 1)). Default NULL.
color_up	Color for significantly up-regulated genes. Default "#CC3333".
color_down	Color for significantly down-regulated genes. Default "#3366CC".
color_nom	Color for nominally significant genes (raw p-value below threshold but adjusted p-value above padj_threshold). Default "#CC9933". Only used when padj_col is provided.
color_ns	Color for non-significant genes. Default "#AAAAAA".
alpha_sig	Transparency for significant points. Default 0.9.
alpha_ns	Transparency for non-significant points. Default 0.5.
label	Whether to label genes with their names. Auto-set to TRUE when the pathway has ≤ 80 matched genes, FALSE otherwise. Can be forced with TRUE or FALSE.
label_genes	Optional character vector of specific gene names within the pathway to label. Overrides label. Useful for very large pathways where you only want to highlight a few key genes.
label_sig	When TRUE, only label genes that pass the threshold. Default FALSE. Ignored when label_genes is provided.
cex_label	Text size for gene labels. Default 0.70.

cex_point	Point size for significant genes. Non-significant genes are plotted at 0.7 * cex_point. Default 1.6.
xlab, ylab	Axis labels. Auto-generated when NULL.
main	Plot title. Default NULL, auto-generated from GO ID and GO name.
go_name	Optional GO term name for the title. When NULL, the function looks for a go_name column in the DSGE result table or uses only the GO ID.
...	Additional arguments passed to plot().

Value

Invisibly returns the subset of de_results for the pathway genes (data.frame).

Examples

```
# Build input objects with simulated data for demonstration
set.seed(42)
de_results <- data.frame(
  log2FoldChange = rnorm(100, mean = 0, sd = 1.5),
  pvalue = runif(100),
  geneName = paste0("GENE", 1:100),
  stringsAsFactors = FALSE
)
pw <- list(
  "GO:0042101" = data.frame(
    db_object_symbol = paste0("GENE", 1:15),
    go_name = "T cell receptor complex",
    stringsAsFactors = FALSE
  )
)
dsge <- pathway_dsge(pw, de_results$pvalue, de_results$pvalue,
                     de_results$geneName, min_size = 1, n_perm = 100,
                     seed = 42)
plot_dsge_volcano(de_results, dsge, pw, go_id = "GO:0042101",
  logFC_col = "log2FoldChange", pval_col = "pvalue", gene_col = "geneName")
```

---

read_gaf *Read a Gene Ontology Annotation File (GAF)*

Description

Efficiently reads a GAF 2.x format gene ontology annotation file using data.table::fread(). Comment header lines starting with ! are automatically skipped. Returns a data.frame with all 17 standard GAF columns.

Usage

```
read_gaf(file, col_names = GAF_COLUMNS, ...)
```

Arguments

file	Path to the GAF file.
col_names	Character vector of column names. Defaults to the standard 17 GAF 2.2 column names. Pass NULL to keep auto-detected column names (if the file has a header row).
...	Additional arguments passed to data.table::fread.

Value

A data.frame containing the GAF annotation data.

Examples

```
# Create a temporary GAF file for demonstration
gaf_file <- tempfile(fileext = ".gaf")
writeLines(c(
  "!gaf-version: 2.2",
  paste0("UniProtKB\tP12345\tGENE_A\t\tGO:0003674\t",
         "PMID:123456\tIDA\t\tF\tGene A\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t"),
  paste0("UniProtKB\tP67890\tGENE_B\t\tGO:0005575\t",
         "PMID:789012\tIBA\t\tC\tGene B\t\tprotein\t",
         "taxon:9606\t20240101\tGO\t\t")
), gaf_file)
gaf <- read_gaf(gaf_file)
head(gaf)
```

---

read_obo *Read GO term names from an OBO file*

Description

Parses a Gene Ontology OBO format file (version 1.2 / 1.4), extracting the id, name, and namespace from each [Term] stanza. [Typedef] stanzas are skipped.

Usage

```
read_obo(file)
```

Arguments

file	Path to the OBO file (e.g. "go-basic.obo").

Value

A `data.frame` with columns:

`id`	GO identifier (e.g. `"GO:0005515"`)
`name`	Human-readable term name (e.g. `"protein binding"`)
`namespace`	Ontology classification: `"molecular_function"`, `"biological_process"`, or `"cellular_component"`

Examples

```
# Create a temporary OBO file for demonstration
obo_file <- tempfile(fileext = ".obo")
writeLines(c(
  "format-version: 1.2",
  "",
  "[Term]",
  "id: GO:0003674",
  "name: molecular_function",
  "namespace: molecular_function",
  "",
  "[Term]",
  "id: GO:0005575",
  "name: cellular_component",
  "namespace: cellular_component",
  "",
  "[Term]",
  "id: GO:0008150",
  "name: biological_process",
  "namespace: biological_process"
), obo_file)
go_names <- read_obo(obo_file)
head(go_names)
```

Index