

Package: GSEAlens (via r-universe)

September 18, 2026

Type Package

Title Gene Set Enrichment Analysis Interactive Explorer

Version 0.99.35

Description GSEAlens provides an interactive exploration layer on top of standard Bioconductor RNA-seq workflows. It consumes fitted model objects from limma (MArrayLM) or DESeq2 (DESeqDataSet) as input and accepts expression matrices and sample metadata as SummarizedExperiment objects, ensuring interoperability with core Bioconductor data containers. For core computation, GSEAlens wraps clusterProfiler::GSEA() as its statistical engine (thereby inheriting the fgsea fast GSEA methodology) and draws on MSigDB gene set collections via the msigdb package from CRAN; multi-contrast parallel computation is handled by future (future::multisession). Visualization output relies on Bioconductor graphics packages including enrichplot, ComplexHeatmap, and circlize, producing figures suitable for publication pipelines. The package also includes a built-in Shiny application for interactive exploration of enrichment results after DEG analysis, with the ability to export self-contained reproducible R scripts.

License MIT + file LICENSE

URL <https://github.com/DDI095/GSEAlens>

BugReports <https://github.com/DDI095/GSEAlens/issues>

Encoding UTF-8

RoxygenNote 8.0.0

biocViews GeneSetEnrichment, Visualization, ShinyApps, Software

Imports stats, utils, enrichplot, enrichit, grDevices, graphics, methods, shiny, shinycssloaders, DT, plotly, ggplot2, rlang, igraph, dplyr, tidyr, tibble, stringr, patchwork, ComplexHeatmap, circlize, grid, clusterProfiler, limma, edgeR, DESeq2, SummarizedExperiment, S4Vectors, msigdb, future, future.apply, htmltools, htmlwidgets, jsonlite, clipr, progressr, visNetwork, shinyjs, withr

Suggests testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown, airway, ggrepel

Depends R (>= 4.5.0)

VignetteBuilder knitr

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev perl libx11-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-08-19 04:12:16 UTC

RemoteUrl <https://github.com/bioc/GSEAlens>

RemoteRef HEAD

RemoteSha d1807abe23fdacd7d9ebaf212e545abf0b471fd3

Contents

GSEAlens-package	3
batch_calc_gsea	4
build_edge_list_safely	6
build_gsea_pathways	7
build_hubgene_network	8
calculate_overlap_ratio	9
color_by_direction	10
creat_addition_data_rdsfile	11
create_addition_template	12
create_gsea_env	13
create_gsea_res	14
create_gsea_task	15
extract_gsea_task	15
extract_hub_genes	16
generate_boxplot_data_code	17
generate_boxplot_image_code	18
generate_combined_plot_code	20
generate_de_volcano_code	21
generate_dotplot_code	22
generate_gsea_html_report	24
generate_hubgene_code	25
generate_hubgene_hover_text	26
generate_joint_canvas_code	27
generate_network_code	28
generate_pathway_plot_code	29
generate_volcano_code	30
get_contrast_registry	32
get_core_genes_for_pathway	32
get_core_genes_list	33
get_de_table	34
get_expr_matrix	34

get_geneset_info	35
get_hubgene_legend	36
get_sample_meta	37
get_term_genes	37
gsea_pathwaysets_toy	38
gsea_pathwaysets_toy_hallmark	38
import_gsea_capsule	39
inspect_gsea_env	40
inspect_gsea_res	40
launch_gsea_app	41
mod_ai_abs_page_server	42
plot_directional_gsea	43
plot_gsea_memory	44
precomputed_gseares	45
prepare_hubgene_nodes	45
preprocessed_dds	46
preprocessed_dds_se	47
preprocessed_limma	47
read_addition_data	48
setup_gsea_env	49
validate_param	50
visualization	51
Index	52

Description

GSEAlens provides an interactive exploration layer on top of standard Bioconductor RNA-seq workflows. It consumes fitted model objects from limma (MArrayLM) or DESeq2 (DESeqDataSet) as input and accepts expression matrices and sample metadata as SummarizedExperiment objects, ensuring interoperability with core Bioconductor data containers.

For core computation, GSEAlens wraps [clusterProfiler::GSEA()] as its statistical engine (thereby inheriting the fgsea fast GSEA methodology) and draws on MSigDB gene set collections via the msgdbr package from CRAN. Multi-contrast parallel computation is handled by future (future::multisession).

Visualization output relies on Bioconductor graphics packages including enrichplot, ComplexHeatmap, and circlize, producing figures suitable for publication pipelines. The package also includes a built-in Shiny application for interactive exploration of enrichment results after DEG analysis, with the ability to export self-contained reproducible R scripts.

Details

****Main entry points:****

- [setup_gsea_env()]: Initialize a GSEA analysis environment from limma or DESeq2 fitted objects.
 - [batch_calc_gsea()]: Run GSEA across multiple contrasts in parallel. - [build_gsea_pathways()]: Build MSigDB pathway database (interactive or automated selection). - [extract_gsea_task()]: Extract a single contrast for downstream visualization and reporting.

****Built-in Shiny application:****

Launch the interactive explorer via [launch_gsea_app()] after DEG analysis. The app provides pathway-level drill-down, multi-contrast joint canvases, quadrant plots, hub-gene networks, and one-click export of self-contained reproducible R scripts.

****Vignettes:****

- 'vignette("GSEAlens", package = "GSEAlens")': Main workflow - 'vignette("GSEAlens-preprocessing", package = "GSEAlens")': Data preprocessing

Author(s)

Shenhui Xu <sealgod@qq.com>

References

<<https://github.com/DDI095/GSEAlens>>

See Also

Useful links:

- <https://github.com/DDI095/GSEAlens>
- Report bugs at <https://github.com/DDI095/GSEAlens/issues>

Examples

```
library(GSEAlens)
packageDescription("GSEAlens")$Version
```

batch_calc_gsea

Batch Parallel GSEA Calculation

Description

Consumes a standard GseaEnv object and performs efficient parallel GSEA calculation.

Usage

```
batch_calc_gsea(
  gsea_env,
  custom_series_name = "Auto_Analysis",
  output_dir = tempdir(),
  workers = 2,
  bidirectional = TRUE,
  minGSSize = 10,
  maxGSSize = 500,
  pvalueCutoff = 1,
  seed = 123,
  force = FALSE,
  use_progress = FALSE,
  chunk_size = NULL
)
```

Arguments

<code>gsea_env</code>	GseaEnv object
<code>custom_series_name</code>	String. Analysis series name
<code>output_dir</code>	String. Output directory for results, default <code>"/GSEA_Output"</code>
<code>workers</code>	Number of parallel cores. Default 4
<code>bidirectional</code>	Logical. Whether to automatically generate reverse contrasts, default TRUE
<code>minGSSize</code>	Minimum gene set size, default 10
<code>maxGSSize</code>	Maximum gene set size, default 500
<code>pvalueCutoff</code>	P-value threshold, default 1
<code>seed</code>	Integer or NULL. Base random seed for per-contrast GSEA permutation, default 123. Each contrast derives its own deterministic seed from the base seed and the contrast id, so results are reproducible across runs and independent of <code>'workers'</code> / <code>'chunk_size'</code> . On <code>clusterProfiler <= 4.20.x</code> (where the <code>'seed'</code> argument of <code>'GSEA()'</code> is silently dropped) reproducibility is guaranteed by seeding the worker R RNG via <code>'withr::local_seed()'</code> ; on <code>clusterProfiler >= 4.21.x</code> the derived seed is also forwarded to <code>'clusterProfiler::GSEA(seed = ...)'</code> . Set to NULL to restore native (non-reproducible) engine sampling.
<code>force</code>	Logical. Whether to force recalculation, default FALSE
<code>use_progress</code>	Logical. Whether to show progress bar, default TRUE
<code>chunk_size</code>	Integer. Number of tasks per worker per chunk, default NULL (auto)

Value

GseaRes object

Examples

```
# Build a GseaEnv from pre-computed inputs shipped with the package
data(preprocessed_limma, package = "GSEAlens")

precomp <- preprocessed_limma
data(gsea_pathwaysets_toy, package = "GSEAlens")

pathways <- gsea_pathwaysets_toy
gsea_env <- setup_gsea_env(
  fit = precomp$fit,
  pathway_obj = pathways,
  expr_data = precomp$gsea_limma_voom_data
)
# Run batch GSEA (writes to a temporary directory; ~20s on the toy set)
gsea_res <- batch_calc_gsea(gsea_env, workers = 2, output_dir = tempdir())
```

```
build_edge_list_safely
```

Build Pathway Similarity Network Edges (Safe Version)

Description

Safely construct edge list for pathway network based on Jaccard similarity of core genes. Includes defensive checks for edge cases. Now includes Overlap Coefficient and Dice Coefficient for comprehensive similarity assessment.

Usage

```
build_edge_list_safely(core_genes_list, min_shared_genes = 2)
```

Arguments

```
core_genes_list
  Named list: pathway_id -> core_gene_vector
min_shared_genes
  Minimum number of shared genes between pathways to form an edge
```

Value

Data frame with columns: from, to, shared, weight (Jaccard index), overlap_coef, dice_coef, shared_genes

Examples

```
# Build edges from two pathways sharing 2 core genes
core_genes <- list(
  PATHWAY_A = c("GENE_A", "GENE_B", "GENE_C"),
  PATHWAY_B = c("GENE_B", "GENE_C", "GENE_D")
)
edges <- build_edge_list_safely(core_genes, min_shared_genes = 2)
print(edges)
```

build_gsea_pathways *Build GSEA Pathway Database with Interactive Selection*

Description

Provides an interactive or automated gene set selection interface based on msigdb, generating standardized pathway objects with intelligent semantic tagging. Supports both human (*Homo sapiens*) and mouse (*Mus musculus*) species.

Usage

```
build_gsea_pathways(species = "HS", auto_select = NULL)
```

Arguments

- | | |
|-------------|--|
| species | Character string specifying the species. Options: <ul style="list-style-type: none"> • "HS": <i>Homo sapiens</i> (human, default) • "MM": <i>Mus musculus</i> (mouse) |
| auto_select | Optional parameter for automated selection. Can be: <ul style="list-style-type: none"> • NULL: Launch interactive menu (default) • "ALL": Load entire MSigDB database for the specified species • Character vector: Collection names (e.g., c("H", "C5:GO:BP") for human) |

Value

A list object containing:

- TERM2GENE: Data frame mapping pathway IDs to gene symbols
- meta_dict: Metadata dictionary with descriptions, URLs, and collection info
- SuperTag: Intelligent batch identifier for the selected collections
- collections_used: Data frame of selected collection metadata
- species: Species identifier ("HS" or "MM")

Examples

```
# Automated selection by collection name -- safe for non-interactive use
pathways <- build_gsea_pathways(species = "HS",
                               auto_select = c("H", "C5:GO:BP"))

# MM uses M-prefixed collection codes (MH, M5:GO:BP, ...)
pathways_mm <- build_gsea_pathways(species = "MM",
                                   auto_select = c("MH", "M5:GO:BP"))

## Not run:
# Interactive menu -- only runs when user is at a terminal
```

```

if (interactive()) {
  pathways <- build_gsea_pathways(species = "HS")
}

# Load ALL collections -- downloads the entire MSigDB (slow, needs network)
pathways <- build_gsea_pathways(species = "HS", auto_select = "ALL")

## End(Not run)

```

build_hubgene_network *Build HubGene Network Data Structure (Enhanced with Leading Edge)*

Description

Build complete network data with leading edge information for edges. Now properly passes `de_df` to `extract_hub_genes` for stat extraction.

Usage

```

build_hubgene_network(
  gsea_task,
  pathway_ids,
  min_hub_degree = 2,
  de_df = NULL,
  res_df = NULL,
  seed = 123
)

```

Arguments

<code>gsea_task</code>	GseaTask object
<code>pathway_ids</code>	Character vector of selected pathway IDs
<code>min_hub_degree</code>	Minimum degree threshold for hub genes (default: 2)
<code>de_df</code>	Optional DE data frame containing stat column
<code>res_df</code>	Optional pathway result data frame
<code>seed</code>	Random seed for reproducibility (default: 123)

Value

A list with three components:

<code>nodes</code>	A list containing pathway and gene data frames
<code>edges</code>	Edge data frame with source, target, and leading edge info
<code>hub_df</code>	Hub gene data frame from extraction

Examples

```
# Load pre-computed GseaRes shipped with the package and pick a task
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")

# Pick top 3 enriched pathway IDs from the task result
res_df <- as.data.frame(task$gsea_res)
top_ids <- utils::head(res_df$ID, 3)

# Build the hub-gene network from these pathways
network <- build_hubgene_network(task, top_ids)
names(network)
```

```
calculate_overlap_ratio
```

Calculate ORA Ratio (for DotPlot)

Description

Calculate overlap ratio between pathway genes and differentially expressed genes.

Usage

```
calculate_overlap_ratio(
  pathway_genes,
  de_genes,
  ratio_mode = c("ora", "leading")
)
```

Arguments

pathway_genes	Gene set of pathway (defined by TERM2GENE)
de_genes	Differentially expressed genes (filtered by p-value threshold)
ratio_mode	Character, "ora" (intersection/pathway size) or "leading" (intersection/DE size)

Value

Numeric, ratio value

Examples

```
ratio <- calculate_overlap_ratio(c("GAPDH", "TP53", "MYC"), c("TP53", "MYC", "EGFR"))
```

color_by_direction	<i>Color Nodes by Direction</i>
--------------------	---------------------------------

Description

Assign colors based on NES (pathways) or log2FC (genes).

Usage

```
color_by_direction(
  node_data,
  color_mode = "logFC",
  left_group = "A",
  right_group = "B"
)
```

Arguments

node_data	Node data frame with type and direction columns
color_mode	Color mode: "logFC", "pathway", or "uniform"
left_group	Left group name for legend label
right_group	Right group name for legend label

Value

Node data with added color and color_label columns

Examples

```
# Build a minimal node data frame (pathway + gene nodes)
nodes <- data.frame(
  name = c("P_POS", "P_NEG", "G_UP", "G_DOWN"),
  type = c("pathway", "pathway", "gene", "gene"),
  NES = c( 2.5, -2.0, NA, NA),
  log2FC = c( NA, NA, 1.5, -1.2),
  stringsAsFactors = FALSE
)
colored <- color_by_direction(nodes, color_mode = "logFC")
head(colored$color)
```

creat_addition_data_rdsfile
Create Additional Data RDS File

Description

Convert a CSV file to standardized gzip-compressed RDS format with validation. The output file will be saved to the specified directory with a standardized naming convention.

Usage

```
creat_addition_data_rdsfile(  
  csv_path,  
  output_dir = getwd(),  
  output_name = "addition_data_gsealens.rds",  
  overwrite = FALSE  
)
```

Arguments

csv_path	Character string specifying the input CSV file path.
output_dir	Character string specifying the output directory path. Default: current working directory.
output_name	Character string specifying the output filename. Default: "addition_data_gsealens.rds"
overwrite	Logical scalar indicating whether to overwrite an existing file. Default: FALSE.

Value

Invisible TRUE on success. The RDS file path is printed to the console.

Examples

```
# Build a small CSV in a temp directory, then convert to RDS.  
tmp_csv <- file.path(tempdir(), "annotations_demo.csv")  
utils::write.csv(  
  data.frame(  
    ID = c("HALLMARK_HYPOXIA", "HALLMARK_INFLAMMATORY_RESPONSE"),  
    Brief_Description = c("Hypoxia", "Inflammatory response"),  
    stringsAsFactors = FALSE  
  ),  
  tmp_csv, row.names = FALSE  
)  
  
out_rds <- file.path(tempdir(), "annotations_demo.rds")  
creat_addition_data_rdsfile(  
  tmp_csv,  
  output_dir = tempdir(),  
  output_name = "annotations_demo.rds",
```

```

    overwrite = TRUE
  )
  file.exists(out_rds)

```

```
create_addition_template
```

Create Additional Data Template

Description

Generate a template CSV file containing all pathway IDs from the GseaRes object. Users can edit this template to add custom annotations such as Brief_Description, Custom_Category, Custom_Tag, and User_Notes.

Usage

```

create_addition_template(
  gsea_res,
  output_path = "addition_template_gsealens.csv",
  include_cols = c("Brief_Description", "Custom_Category", "Custom_Tag", "User_Notes")
)

```

Arguments

<code>gsea_res</code>	A GseaRes object (generated by <code>batch_calc_gsea</code>).
<code>output_path</code>	Character string specifying the output template file path. Default: "addition_template_gsealens.csv"
<code>include_cols</code>	Character vector specifying additional columns to include besides ID. Default includes standard annotation columns.

Value

Invisible TRUE on success.

Examples

```

# Load pre-computed GseaRes shipped with the package and write a template
# CSV to a temporary file (no side effects on the user's filesystem).
data(precomputed_gseares, package = "GSEALens")

gsea_res <- precomputed_gseares
out_csv <- file.path(tempdir(), "addition_template_demo.csv")
create_addition_template(gsea_res, output_path = out_csv)

# The template contains one row per pathway ID, ready to edit.
template <- utils::read.csv(out_csv)
utils::head(template)

```

create_gsea_env	<i>Define GseaEnv Class</i>
-----------------	-----------------------------

Description

Standardized GSEA input environment object.

Usage

```
create_gsea_env(  
  backend_info,  
  contrast_registry,  
  de_store,  
  expr_bundle,  
  geneset,  
  raw_obj  
)
```

Arguments

backend_info	Backend information object
contrast_registry	Contrast registry
de_store	Differential expression store
expr_bundle	Expression data bundle
geneset	Gene set collection
raw_obj	Raw data object

Value

A GseaEnv object

Examples

```
gsea_env <- create_gsea_env(  
  backend_info = list(),  
  contrast_registry = data.frame(),  
  de_store = data.frame(),  
  expr_bundle = list(),  
  geneset = character(),  
  raw_obj = NULL  
)
```

create_gsea_res	<i>Define GseaRes Class</i>
-----------------	-----------------------------

Description

GSEA computation result capsule.

Usage

```
create_gsea_res(  
  metadata,  
  backend_info,  
  contrast_registry,  
  de_store,  
  expr_bundle,  
  geneset_info,  
  results  
)
```

Arguments

metadata	Metadata information
backend_info	Backend information object
contrast_registry	Contrast registry
de_store	Differential expression store
expr_bundle	Expression data bundle
geneset_info	Gene set information
results	GSEA results

Value

A GseaRes object

Examples

```
gsea_res <- create_gsea_res(  
  metadata = list(),  
  backend_info = list(),  
  contrast_registry = data.frame(),  
  de_store = data.frame(),  
  expr_bundle = list(),  
  geneset_info = list(),  
  results = list()  
)
```

create_gsea_task	<i>Define GseaTask Class</i>
------------------	------------------------------

Description

Single contrast extraction result object.

Usage

```
create_gsea_task(gsea_res, meta)
```

Arguments

gsea_res	GseaRes object
meta	Metadata for the task

Value

A GseaTask object

Examples

```
gsea_task <- create_gsea_task(  
  gsea_res = NULL,  
  meta = data.frame()  
)
```

extract_gsea_task	<i>Extract GSEA Results for Specific Contrast</i>
-------------------	---

Description

Extract results for a specified contrast from a GseaRes capsule. Supports subsetting by gene set collection and automatically recalculates FDR.

Usage

```
extract_gsea_task(gsea_res, contrast_id, target_collection = "ALL")
```

Arguments

gsea_res	GseaRes object (generated by batch_calc_gsea)
contrast_id	String. Contrast group ID (e.g., "A_vs_B").
target_collection	String vector. Gene set collection to extract (e.g., "H", "C2:CP:KEGG_LEGACY"). Default "ALL" means extract all.

Value

GseaTask object.

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")
class(task)
```

extract_hub_genes	<i>Extract Hub Genes from Selected Pathways (Enhanced with Leading Edge)</i>
-------------------	--

Description

Extract hub genes with their leading edge status for each pathway. A gene is considered a hub if it appears in at least min_degree pathways.

Usage

```
extract_hub_genes(gsea_task, pathway_ids, min_degree = 2, de_df = NULL)
```

Arguments

gsea_task	GseaTask object
pathway_ids	Character vector of pathway IDs to analyze
min_degree	Minimum number of pathways a gene must appear in (default: 2)
de_df	Optional differential expression data frame with stat column

Value

A data frame with columns:

gene	Gene symbol
degree	Number of pathways the gene appears in
pathways	Comma-separated list of pathway IDs
is_hub	Logical indicating if gene is a hub (degree >= min_degree)
stat	Statistic value from de_df or geneList
pathway_leading_edges	List column with per-pathway leading edge status

Examples

```
# Load pre-computed GseaRes shipped with the package and pick a task
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")

# Pick top 3 enriched pathway IDs from the task result
res_df <- as.data.frame(task$gsea_res)
top_ids <- utils::head(res_df$ID, 3)

# Extract hub genes appearing in >= 2 of these pathways
hubs <- extract_hub_genes(task, top_ids, min_degree = 2)
utils::head(hubs)
```

```
generate_boxplot_data_code
```

Generate Boxplot Expression-Data Extraction Code

Description

Produce a self-contained, reproducible R script that extracts the per-sample expression values (Sample / Group / Expression) of a single gene from a GseaRes object. The returned data frame matches exactly what is shown in the Quadrant module's expression boxplot (panel 4), so the user can paste it into other software (e.g. GraphPad Prism, Excel, R) for downstream plotting.

Usage

```
generate_boxplot_data_code(
  GSEAlens_res,
  contrast_id,
  gene_symbol,
  expr_type = "logcpm",
  custom_order = NULL
)
```

Arguments

GSEAlens_res	A GseaRes object.
contrast_id	Character. Contrast ID (e.g. "Treat_vs_Control").
gene_symbol	Character. Gene symbol as shown in the boxplot title.
expr_type	Character. Expression matrix type passed to get_expr_matrix() (e.g. "logcpm", "vst", "count"). Default "logcpm".
custom_order	Character or NULL. Custom group ordering string (e.g. "GroupA -> GroupB") or "default" / NULL to use natural order. When non-default, the generated code will re-order the Group factor accordingly.

Value

A character string of executable R code.

Examples

```
data(precomputed_gseares, package = "GSEAlens")
code <- generate_boxplot_data_code(
  GSEAlens_res = precomputed_gseares,
  contrast_id  = "untrt_vs_trt",
  gene_symbol  = "GAPDH",
  expr_type    = "logcpm"
)
cat(code)
```

```
generate_boxplot_image_code
```

Generate Publication Boxplot Image Code

Description

Return an executable ggplot2 script reproducing the Quadrant module's per-gene expression boxplot (panel 4) as a static, publication-ready figure. Mirrors the interactive plotly: boxplot + jittered points, group colors, optional zero baseline, optional custom group order.

Usage

```
generate_boxplot_image_code(
  box_data_long,
  gene_symbol = "Gene",
  expr_type = "Expression",
  left_group = NA_character_,
  right_group = NA_character_,
  use_zero_baseline = TRUE,
  custom_order = "default",
  base_size = 12,
  margin_top = 15,
  margin_bottom = 15,
  margin_left = 18,
  margin_right = 18,
  fig_width = 5,
  fig_height = 5,
  fig_dpi = 300
)
```

Arguments

<code>box_data_long</code>	data.frame with columns: Sample, Group, Expression. (Same long format returned by <code>extract_boxplot_data()</code> \$long.)
<code>gene_symbol</code>	Character. Gene name for the plot title.
<code>expr_type</code>	Character. Y-axis label (e.g. "logcpm", "log2FC").
<code>left_group</code>	Character. First group name (gets red fill).
<code>right_group</code>	Character. Second group name (gets blue fill).
<code>use_zero_baseline</code>	Logical. Add dashed red line at $y = 0$.
<code>custom_order</code>	Character scalar. Custom group ordering (e.g. "A->B,C") or "default".
<code>base_size</code>	Numeric.
<code>margin_top</code>	Numeric. Top plot margin (mm).
<code>margin_bottom</code>	Numeric. Bottom plot margin (mm).
<code>margin_left</code>	Numeric. Left plot margin (mm).
<code>margin_right</code>	Numeric. Right plot margin (mm).
<code>fig_width</code>	Numeric. Figure width in inches for ggsave.
<code>fig_height</code>	Numeric. Figure height in inches for ggsave.
<code>fig_dpi</code>	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```
# Build a minimal long-format expression data frame (self-contained)
box_long <- data.frame(
  Sample = c("S1", "S2", "S3", "S4"),
  Group = factor(c("untrt", "untrt", "trt", "trt"),
    levels = c("untrt", "trt")),
  Expression = c(8.1, 8.3, 9.5, 9.7)
)
code <- generate_boxplot_image_code(
  box_data_long = box_long,
  gene_symbol = "GAPDH",
  expr_type = "logcpm",
  left_group = "untrt",
  right_group = "trt"
)
cat(code)
```

```
generate_combined_plot_code
```

Generate Combined Pathway Plotting Code

Description

Return an executable R script reproducing the Combined Pathway Plotting module (mod_multi_plot / module 12) as a static figure. Mirrors the interactive plot_directional_gsea() call the app performs when the user selects pathways in the main table's "Combined Display" column.

Usage

```
generate_combined_plot_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  target_pathways,
  subPlot = 3,
  curve_colors = NULL,
  left_group = "Left",
  right_group = "Right"
)
```

Arguments

gsea_res_var	Character. Name of the GseaRes variable the generated code should reference (default "gsea_res"). The user replaces the placeholder assignment with their own object.
contrast_id	Character. Contrast ID passed to extract_gsea_task().
target_pathways	Character vector. Pathway IDs to plot.
subPlot	Integer (1-3). plot_directional_gsea subPlot argument (1 = running score, 2 = rank position, 3 = both). Default 3.
curve_colors	Character vector of color hex codes for pathway curves. If NULL, falls back to a sensible default palette.
left_group	Character. Left group label for axis annotation.
right_group	Character. Right group label.

Value

Character scalar of executable R code.

Examples

```
code <- generate_combined_plot_code(
  contrast_id = "untrt_vs_trt",
  target_pathways = c("HALLMARK_TNFA_SIGNALING_VIA_NFKB")
)
cat(code)
```

```
generate_de_volcano_code
```

Generate Publication DE Volcano Code

Description

Return an executable ggplot2 script reproducing the Quadrant module's gene-level Differential Expression Volcano as a static figure. Assumes a GseaRes object is already loaded. Six-color classification (both / user / pathway / up / down / ns) mirrors the interactive view.

Usage

```
generate_de_volcano_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  user_genes = character(0),
  pathway_genes = character(0),
  logfc_thresh = 1,
  pval_thresh = 0.05,
  left_group = "Left",
  right_group = "Right",
  show_annotations = FALSE,
  base_size = 12,
  margin_top = 15,
  margin_bottom = 15,
  margin_left = 18,
  margin_right = 18,
  fig_width = 8,
  fig_height = 6,
  fig_dpi = 300
)
```

Arguments

<code>gsea_res_var</code>	Character. Name of the GseaRes variable.
<code>contrast_id</code>	Character. Contrast ID for DE table extraction.
<code>user_genes</code>	Character vector of user-selected genes (upper-cased).
<code>pathway_genes</code>	Character vector of pathway genes (upper-cased).
<code>logfc_thresh</code>	Numeric. Default 1.

pval_thresh	Numeric. Default 0.05.
left_group	Character.
right_group	Character.
show_annotations	Logical. Add ggrepel labels.
base_size	Numeric.
margin_top	Numeric. Top plot margin (mm).
margin_bottom	Numeric. Bottom plot margin (mm).
margin_left	Numeric. Left plot margin (mm).
margin_right	Numeric. Right plot margin (mm).
fig_width	Numeric. Figure width in inches for ggsave.
fig_height	Numeric. Figure height in inches for ggsave.
fig_dpi	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```
code <- generate_de_volcano_code(
  gsea_res_var = "gsea_res", contrast_id = "untrt_vs_trt",
  left_group = "Ctrl", right_group = "Trt")
cat(code)
```

generate_dotplot_code *Generate Publication DotPlot Code*

Description

Return an executable ggplot2 script reproducing the Pathway Relationship module's dotplot as a static, publication-ready figure. The generated script assumes a GseaRes object is already loaded in the R environment and derives plot data via `extract_gsea_task()` + `get_core_genes_list()`, producing concise reproducible code without embedding raw data.

Usage

```
generate_dotplot_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  pathways,
  color_mode = "padj",
  size_mode = "core_size",
  left_group = "Left",
```

```

    right_group = "Right",
    palette = "D",
    point_range = c(3, 8),
    target_collection = "ALL",
    base_size = 12,
    margin_top = 18,
    margin_bottom = 18,
    margin_left = 18,
    margin_right = 18,
    fig_width = 9,
    fig_height = 7,
    fig_dpi = 300
  )

```

Arguments

<code>gsea_res_var</code>	Character. Name of the GseaRes variable in the user's environment (default "gsea_res").
<code>contrast_id</code>	Character. Contrast ID to extract.
<code>pathways</code>	Character vector. Pathway IDs to display.
<code>color_mode</code>	Character. One of "padj", "pval", "nes".
<code>size_mode</code>	Character. One of "core_size", "setsize".
<code>left_group</code>	Character. Left contrast label.
<code>right_group</code>	Character. Right contrast label.
<code>palette</code>	Character. Viridis option letter ("A"-"H").
<code>point_range</code>	Numeric length 2. Min/max point size in mm.
<code>target_collection</code>	Character. Gene set collection(s) to extract (e.g. "H", c("C2", "C2:CP")). Default "ALL".
<code>base_size</code>	Numeric. theme_bw base font size.
<code>margin_top</code>	Numeric. Top plot margin (mm).
<code>margin_bottom</code>	Numeric. Bottom plot margin (mm).
<code>margin_left</code>	Numeric. Left plot margin (mm).
<code>margin_right</code>	Numeric. Right plot margin (mm).
<code>fig_width</code>	Numeric. Figure width in inches for ggsave.
<code>fig_height</code>	Numeric. Figure height in inches for ggsave.
<code>fig_dpi</code>	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```
code <- generate_dotplot_code(
  gsea_res_var = "gsea_res", contrast_id = "untrt_vs_trt",
  pathways = c("HALLMARK_TNFA_SIGNALING_VIA_NFKB"))
cat(code)
```

```
generate_gsea_html_report
```

Generate GSEA HTML Report

Description

Automatically avoids column name contamination, intelligently extracts expression matrices from objects, and outputs comfortable web reports with dual p-value display and heatmaps. Uses the ComplexHeatmap engine to draw high-quality heatmaps, perfectly replicating pheatmap aesthetics.

Usage

```
generate_gsea_html_report(
  res_obj,
  output_base_dir = NULL,
  p_adjust_cutoff = 1,
  top_plots = c(15, 15),
  dpi = 200
)
```

Arguments

res_obj	GseaTask object
output_base_dir	Output folder path. If left empty (NULL), it will automatically follow the original project location of the calculation capsule.
p_adjust_cutoff	FDR filtering threshold, default 1
top_plots	Number of pathways to plot detailed sub-pages, format c(positive_count, negative_count)
dpi	Resolution for GSEA enrichment plots, default 200

Value

Path to the generated HTML report (invisible)

Examples

```
data(precomputed_gseares, package = "GSEAlens")
task <- extract_gsea_task(precomputed_gseares, contrast_id = "untrt_vs_trt")
report_path <- generate_gsea_html_report(task,
                                         output_base_dir = tempdir())
```

generate_hubgene_code *Generate Publication HubGene Network Code*

Description

Return an executable R script reproducing the HubGene module's visNetwork view as a static ggplot2 figure. Assumes a GseaRes object is already loaded. Node size is controlled by the same three-mode logic (fixed / fdr / setsize) as the interactive view.

Usage

```
generate_hubgene_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  pathways,
  min_hub_degree = 2,
  pw_size_mode = "setsize",
  seed = 42L,
  target_collection = "ALL",
  base_size = 12,
  margin_top = 18,
  margin_bottom = 18,
  margin_left = 18,
  margin_right = 18,
  fig_width = 10,
  fig_height = 8,
  fig_dpi = 300
)
```

Arguments

gsea_res_var	Character. Name of the GseaRes variable.
contrast_id	Character. Contrast ID to extract.
pathways	Character vector. Pathway IDs to include.
min_hub_degree	Integer. Minimum hub degree for gene inclusion.
pw_size_mode	Character. One of "fixed", "fdr", "setsize".
seed	Integer.
target_collection	Character. Gene set collection(s) to extract. Default "ALL".

base_size	Numeric.
margin_top	Numeric. Top plot margin (mm).
margin_bottom	Numeric. Bottom plot margin (mm).
margin_left	Numeric. Left plot margin (mm).
margin_right	Numeric. Right plot margin (mm).
fig_width	Numeric. Figure width in inches for ggsave.
fig_height	Numeric. Figure height in inches for ggsave.
fig_dpi	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```
code <- generate_hubgene_code(
  gsea_res_var = "gsea_res", contrast_id = "untrt_vs_trt",
  pathways = c("HALLMARK_TNFA_SIGNALING_VIA_NFKB"))
cat(code)
```

```
generate_hubgene_hover_text
      Generate HubGene Hover Text
```

Description

Generate HTML text for plotly hover tooltips.

Usage

```
generate_hubgene_hover_text(
  node,
  node_type,
  left_group = "A",
  right_group = "B"
)
```

Arguments

node	Node data row (as list with named elements)
node_type	Node type: "pathway" or "gene"
left_group	Left group name
right_group	Right group name

Value

Character string with HTML content for hover tooltip

Examples

```
# Example with a pathway node
pathway_node <- list(
  id = "HALLMARK_INFLAMMATORY_RESPONSE",
  NES = 2.3, FDR = 0.001, pvalue = 0.0005,
  n_core = 15, n_total = 50
)
hover_pathway <- generate_hubgene_hover_text(pathway_node, "pathway")
cat(hover_pathway)

# Example with a gene node
gene_node <- list(id = "GENE_A", log2FC = 1.8)
hover_gene <- generate_hubgene_hover_text(gene_node, "gene")
cat(hover_gene)
```

```
generate_joint_canvas_code
```

Generate Joint Canvas Code

Description

Generate Joint Canvas Code

Usage

```
generate_joint_canvas_code(
  GSEAlens_res,
  contrast_ids,
  target_pathways,
  ncol = 2
)
```

Arguments

GSEAlens_res	GseaRes object
contrast_ids	Character vector. Multiple contrast IDs
target_pathways	Character vector. Pathway IDs
ncol	Integer. Number of columns

Value

A character string containing executable R code

Examples

```
data(precomputed_gseares, package = "GSEALens")
task <- extract_gsea_task(precomputed_gseares, contrast_id = "untrt_vs_trt")
target <- as.data.frame(task$gsea_res)$ID[1]
code <- generate_joint_canvas_code(GSEALens_res = precomputed_gseares,
                                  contrast_ids = "untrt_vs_trt",
                                  target_pathways = target)

cat(code)
```

generate_network_code *Generate Publication Pathway Network Code*

Description

Return an executable R script reproducing the Pathway Relationship module's network view as a static ggplot2 figure. Assumes a GseaRes object is already loaded. Uses igraph layout + geom_segment edges + geom_point nodes (no ggraph dependency).

Usage

```
generate_network_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  pathways,
  min_shared_genes = 2,
  width_mode = "weight",
  seed = 42L,
  target_collection = "ALL",
  base_size = 12,
  margin_top = 18,
  margin_bottom = 18,
  margin_left = 18,
  margin_right = 18,
  fig_width = 10,
  fig_height = 8,
  fig_dpi = 300
)
```

Arguments

gsea_res_var	Character. Name of the GseaRes variable.
contrast_id	Character. Contrast ID to extract.
pathways	Character vector. Pathway IDs to include in the network.
min_shared_genes	Integer. Minimum shared core genes for an edge.

width_mode	Character. "rank" or "weight".
seed	Integer. Layout seed.
target_collection	Character. Gene set collection(s) to extract. Default "ALL".
base_size	Numeric.
margin_top	Numeric. Top plot margin (mm).
margin_bottom	Numeric. Bottom plot margin (mm).
margin_left	Numeric. Left plot margin (mm).
margin_right	Numeric. Right plot margin (mm).
fig_width	Numeric. Figure width in inches for ggsave.
fig_height	Numeric. Figure height in inches for ggsave.
fig_dpi	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```
code <- generate_network_code(
  gsea_res_var = "gsea_res", contrast_id = "untrt_vs_trt",
  pathways = c("HALLMARK_TNFA_SIGNALING_VIA_NFKB", "HALLMARK_HYPOXIA"))
cat(code)
```

generate_pathway_plot_code

Generate Pathway Plotting Code

Description

Generate Pathway Plotting Code

Usage

```
generate_pathway_plot_code(
  GSEAlens_res,
  contrast_id,
  target_pathways,
  user_genes = character(0),
  pathway_genes = character(0),
  expr_type = "logcpm"
)
```

Arguments

GSEAlens_res	GseaRes object
contrast_id	Character. Contrast ID
target_pathways	Character vector. Pathway IDs
user_genes	Character vector. Genes of interest
pathway_genes	Character vector. Pathway genes (for coloring)
expr_type	Character. Expression type

Value

A character string containing executable R code

Examples

```
# Generate plotting code for the top enriched pathway in a pre-computed task
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")
res_df <- as.data.frame(task$gsea_res)
top_pathway <- res_df$ID[1]

code <- generate_pathway_plot_code(
  GSEAlens_res = gsea_res,
  contrast_id = "untrt_vs_trt",
  target_pathways = top_pathway
)
cat(code)
```

generate_volcano_code *Generate Publication Volcano Code*

Description

Return an executable ggplot2 script reproducing the Quadrant module's pathway volcano plot as a static, publication-ready figure. Assumes a GseaRes object is already loaded in the R environment.

Usage

```
generate_volcano_code(
  gsea_res_var = "gsea_res",
  contrast_id,
  left_group = "Left",
  right_group = "Right",
  n_selected = 0L,
  selected_ids = character(0),
```

```

    show_annotations = FALSE,
    target_collection = "ALL",
    base_size = 12,
    margin_top = 15,
    margin_bottom = 15,
    margin_left = 18,
    margin_right = 18,
    fig_width = 8,
    fig_height = 6,
    fig_dpi = 300
  )

```

Arguments

<code>gsea_res_var</code>	Character. Name of the GseaRes variable.
<code>contrast_id</code>	Character. Contrast ID to extract.
<code>left_group</code>	Character.
<code>right_group</code>	Character.
<code>n_selected</code>	Integer. Number of selected pathways (subtitle).
<code>selected_ids</code>	Character vector. Pathway IDs for ggrepel annotation.
<code>show_annotations</code>	Logical. Add ggrepel labels for selected pathways.
<code>target_collection</code>	Character. Gene set collection(s) to extract. Default "ALL".
<code>base_size</code>	Numeric.
<code>margin_top</code>	Numeric. Top plot margin (mm).
<code>margin_bottom</code>	Numeric. Bottom plot margin (mm).
<code>margin_left</code>	Numeric. Left plot margin (mm).
<code>margin_right</code>	Numeric. Right plot margin (mm).
<code>fig_width</code>	Numeric. Figure width in inches for ggsave.
<code>fig_height</code>	Numeric. Figure height in inches for ggsave.
<code>fig_dpi</code>	Numeric. DPI for PNG raster output (default 300).

Value

Character scalar of executable R code.

Examples

```

code <- generate_volcano_code(
  gsea_res_var = "gsea_res", contrast_id = "untrt_vs_trt",
  left_group = "Ctrl", right_group = "Trt")
cat(code)

```

get_contrast_registry *Get Contrast Registry*

Description

Retrieve the registered contrasts used for differential expression analysis, including contrast definitions and metadata.

Usage

```
get_contrast_registry(obj)
```

Arguments

obj A GseaEnv or GseaRes object.

Value

The contrast registry object stored in the input object.

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
registry <- get_contrast_registry(gsea_res)
print(registry)
```

get_core_genes_for_pathway
Extract Pathway Core Genes (Leading Edge)

Description

Parse the core_enrichment field from GSEA results to extract the gene list. This is the fundamental data interface for all visualization modules in 03/04.

Usage

```
get_core_genes_for_pathway(gsea_res_obj, pathway_id)
```

Arguments

gsea_res_obj GseaRes object or single GSEA result (list(status=, data=, ...))
pathway_id Character, pathway ID

Value

Character vector, core genes (uppercase normalized)

Examples

```
# Load a pre-computed GseaRes and extract a task
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")
res_df <- as.data.frame(task$gsea_res)
pw_id <- res_df$ID[1]
core_genes <- get_core_genes_for_pathway(task, pw_id)
head(core_genes)
```

get_core_genes_list *Batch Extract Core Genes for Multiple Pathways*

Description

Batch data preparation for 03 module Network/UpSet/Chord visualizations.

Usage

```
get_core_genes_list(gsea_task_obj, pathway_ids)
```

Arguments

gsea_task_obj GseaTask object (single contrast)
 pathway_ids Character vector, pathway ID list

Value

Named list, names = pathway_id, values = core_genes vector

Examples

```
# Load a pre-computed GseaRes and extract a task
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")
res_df <- as.data.frame(task$gsea_res)
pw_ids <- head(res_df$ID, 3)
genes_list <- get_core_genes_list(task, pw_ids)
length(genes_list)
```

get_de_table	<i>Get Differential Expression Table</i>
--------------	--

Description

Extract differential expression results for a specific contrast from the stored results. Automatically handles reverse contrasts (e.g., B_vs_A) by retrieving the forward contrast and flipping the sign of logFC.

Usage

```
get_de_table(obj, contrast_id)
```

Arguments

obj	A GseaRes object containing differential expression results.
contrast_id	Character string specifying the contrast ID (e.g., "A_vs_B"). If the reverse contrast exists in storage, it will be retrieved and logFC/stat signs will be inverted.

Value

A data frame containing at minimum: gene_symbol, logFC, and p-value columns. Additional columns (stat, pvalue, padj, t) may be present depending on the analysis backend.

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
de_result <- get_de_table(gsea_res, contrast_id = "untrt_vs_trt")
head(de_result)
```

get_expr_matrix	<i>Get Expression Matrix</i>
-----------------	------------------------------

Description

Extract expression matrix from GseaEnv or GseaRes object, supporting multiple normalization methods including raw counts, CPM, logCPM, VST, FPKM, TPM, and log-normalized counts.

Usage

```
get_expr_matrix(obj, type = "default", ...)
```

Arguments

obj	A GseaEnv or GseaRes object.
type	Character string specifying the type of expression values to return. Options include: "default" (default display matrix), "raw" (raw counts), "cpm" (counts per million), "logcpm" (log2-transformed CPM), "vst" (variance stabilizing transformation, DESeq2 only), "fpkm" (fragments per kilobase per million), "tpm" (transcripts per million), and "lognorm" (log2 of normalized counts).
...	Additional arguments passed to backend-specific methods.

Value

A numeric matrix with genes as rows and samples as columns.

Examples

```
# Load pre-computed inputs shipped with the package
data(preprocessed_limma, package = "GSEAlens")

precomp <- preprocessed_limma
data(gsea_pathwaysets_toy, package = "GSEAlens")

pathways <- gsea_pathwaysets_toy
gsea_env <- setup_gsea_env(
  fit = precomp$fit,
  pathway_obj = pathways,
  expr_data = precomp$gsea_limma_voom_data
)
expr <- get_expr_matrix(gsea_env)
dim(expr)
```

get_geneset_info	<i>Get Gene Set Information</i>
------------------	---------------------------------

Description

Retrieve gene set collection metadata including pathway definitions and annotation dictionary.

Usage

```
get_geneset_info(obj)
```

Arguments

obj	A GseaEnv or GseaRes object.
-----	------------------------------

Value

An object containing geneset_info with pathway definitions and metadata dictionary.

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
gs_info <- get_geneset_info(gsea_res)
names(gs_info)
```

get_hubgene_legend	<i>Get Color Legend Configuration</i>
--------------------	---------------------------------------

Description

Generate color legend data for HubGene network plot.

Usage

```
get_hubgene_legend(left_group = "A", right_group = "B")
```

Arguments

left_group	Left group name (e.g., treatment or case)
right_group	Right group name (e.g., control or reference)

Value

A named list containing:

pathway_up	Color for up-regulated pathways
pathway_down	Color for down-regulated pathways
gene_up	Color for up-regulated genes
gene_down	Color for down-regulated genes
labels	Named list of legend labels

Examples

```
legend <- get_hubgene_legend("Control", "Treatment")
```

get_sample_meta	<i>Get Sample Metadata</i>
-----------------	----------------------------

Description

Extract and process sample metadata, handling DFrame conversion from DESeq2 and unifying group column naming conventions across backends.

Usage

```
get_sample_meta(obj)
```

Arguments

obj	GseaEnv or GseaRes object
-----	---------------------------

Value

A data frame with sample metadata. Includes a standardized 'group' column derived from the target factor (inferred from dds_obj@design for DESeq2 workflows). Sample row names are preserved for downstream matching.

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
meta <- get_sample_meta(gsea_res)
head(meta)
```

get_term_genes	<i>Get Full Pathway Gene Set (TERM2GENE)</i>
----------------	--

Description

Extract complete gene list for a pathway from geneset_info.

Usage

```
get_term_genes(gsea_res, pathway_id)
```

Arguments

gsea_res	GseaRes object
pathway_id	Pathway ID

Value

Character vector

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")
```

```
gsea_res <- precomputed_gseares
pw_id <- gsea_res$geneset_info$term2gene$gs_name[1]
genes <- get_term_genes(gsea_res, pw_id)
head(genes)
```

`gsea_pathwaysets_toy` *Toy MSigDB pathway collection (Hallmark + KEGG_LEGACY)*

Description

A lightweight pathway object for examples and vignettes, containing 236 pathways (Hallmark + KEGG_LEGACY) pre-assembled by [GSEAlens::build_gsea_pathways()]. See ‘inst/scripts/make_gsea_pathwaysets_toy.R’ for regeneration.

Format

A list with elements ‘TERM2GENE’, ‘meta_dict’, ‘SuperTag’, ‘collections_used’, ‘species’.

Source

Assembled from msigdb (Hallmark + KEGG_LEGACY collections).

Examples

```
data(gsea_pathwaysets_toy, package = "GSEAlens")
names(gsea_pathwaysets_toy)
```

`gsea_pathwaysets_toy_hallmark`
Toy MSigDB pathway collection (Hallmark only)

Description

A minimal Hallmark-only pathway object for fast examples and tests. See ‘inst/scripts/make_gsea_pathwaysets_toy.R’ for regeneration.

Format

A list with elements ‘TERM2GENE’, ‘meta_dict’, ‘SuperTag’, ‘collections_used’, ‘species’.

Source

Assembled from msigdb (Hallmark collection).

Examples

```
data(gsea_pathwaysets_toy_hallmark, package = "GSEAlens")
names(gsea_pathwaysets_toy_hallmark)
```

```
import_gsea_capsule    Load and Smart-Relocate GSEA Computation Capsule
```

Description

Safely load a computation capsule. If the file is moved from its original project path (e.g., copied to desktop), the engine will automatically restore the standard folder structure in the current working directory and relocate the capsule. Fully compatible with GseaEnv and GseaRes objects.

Usage

```
import_gsea_capsule(file_path, auto_relocate = TRUE, inspect = TRUE)
```

Arguments

file_path	Character. Absolute or relative path to the capsule rds file.
auto_relocate	Logical. Whether to automatically fix the folder structure and relocate based on the built-in lineage in the current directory? Default TRUE.
inspect	Logical. Whether to automatically print capsule overview after loading? Default TRUE.

Value

Returns the parsed GseaRes or GseaEnv capsule object.

Examples

```
# Load the pre-computed GseaRes capsule shipped with the package, write it
# to a temp file, then import it (auto_relocate = FALSE to avoid side
# effects).
data(precomputed_gseares, package = "GSEAlens")
tmp_capsule <- file.path(tempdir(), "import_demo.rds")
saveRDS(precomputed_gseares, tmp_capsule)

gsea_res <- import_gsea_capsule(
  tmp_capsule,
  auto_relocate = FALSE,
  inspect = FALSE
)
class(gsea_res)
```

inspect_gsea_env	View GSEA Environment Object Overview
------------------	---

Description

Print detailed information about the GseaEnv object to the console, including backend type, contrast list, and gene set status.

Usage

```
inspect_gsea_env(env_obj)
```

Arguments

env_obj	GseaEnv object
---------	----------------

Value

Invisibly returns the input object (for chaining).

Examples

```
# Build a GseaEnv from pre-computed inputs shipped with the package
data(preprocessed_limma, package = "GSEAlens")

precomp <- preprocessed_limma
data(gsea_pathwaysets_toy, package = "GSEAlens")

pathways <- gsea_pathwaysets_toy
env <- setup_gsea_env(
  fit      = precomp$fit,
  pathway_obj = pathways,
  expr_data = precomp$gsea_limma_voom_data
)
inspect_gsea_env(env)
```

inspect_gsea_res	View GSEA Result Overview
------------------	---

Description

Print detailed summary of GseaRes object to console.

Usage

```
inspect_gsea_res(gsea_res)
```

Arguments

gsea_res GseaRes object

Value

Invisibly returns the input object (for chaining).

Examples

```
# Load a pre-computed GseaRes shipped with the package
data(precomputed_gseares, package = "GSEAlens")

gsea_res <- precomputed_gseares
inspect_gsea_res(gsea_res)
```

launch_gsea_app	<i>GSEAlens Interactive Analysis Workstation</i>
-----------------	--

Description

Modular Shiny application based on GseaRes object, supporting Limma and DESeq2 dual back-ends. Provides interactive visualization for GSEA pathway enrichment analysis including volcano plots, heatmaps, expression boxplots, and pathway relationship networks.

Usage

```
launch_gsea_app(gsea_res, addition_data = NULL)
```

Arguments

gsea_res GseaRes object (generated by batch_calc_gsea())

addition_data Optional. A data frame or file path containing pathway annotations to merge into the main table. Can also be:

- A data.frame with 'ID' column as primary key
- A character vector: file path to .csv, .csv.gz, .rds, or .rds.gz
- NULL: automatically searches for "addition_data_gsealens.rds" or "addition_data_gsealens.csv" in the working directory

Additional columns will be merged to the main pathway table.

Value

Shiny application object

Examples

```

if(interactive()){
  # Basic usage
  gsea_res <- batch_calc_gsea(gsea_env)
  launch_gsea_app(gsea_res)

  # With addition data from file path
  launch_gsea_app(gsea_res, "pathway_annotations.csv")

  # With pre-loaded data frame
  add_data <- read.csv("annotations.csv")
  launch_gsea_app(gsea_res, add_data)
}

```

mod_ai_abs_page_server

AI Interpretation Page Server

Description

AI Interpretation Page Server. Called for side effects within a Shiny module.

Usage

```
mod_ai_abs_page_server(id, gsea_res, data_prep_list, table_controller)
```

Arguments

id	Module ID
gsea_res	GseaRes object
data_prep_list	Data preprocessing list
table_controller	Table controller

Value

No return value; called for side effects within a Shiny module. Extract Leading Edge genes and format as comma-separated Determine confidence level

plot_directional_gsea *Plot Directional GSEA*

Description

Professional-grade GSEA plotting engine with seamless support for single or multiple pathway visualization. Automatically generates elegant legends and native red-blue gene distribution track.

Usage

```
plot_directional_gsea(
  directional_gsea_obj,
  target_pathways,
  main_title = NULL,
  subPlot = 3,
  curveCol = NULL,
  add_pval = FALSE,
  show_contrast_in_axis = FALSE,
  ...
)
```

Arguments

directional_gsea_obj	A wrapped GseaTask object (returned by extract_gsea_task)
target_pathways	Vector of pathway IDs to plot (supports single or multiple)
main_title	Main title name
subPlot	Controls the number of subplots generated (1: enrichment plot only; 2: enrichment + heatmap track; 3: full rank track)
curveCol	Custom curve color vector
add_pval	Whether to add statistical annotations on the plot. Default FALSE.
show_contrast_in_axis	Logical. Whether to display contrast group information on the x-axis of the Rank plot. Default FALSE; recommended to set TRUE only in combined canvas mode.
...	Additional parameters passed to internal engine

Value

ggplot2 composite object

Examples

```
# Load pre-computed GseaRes and pick a task + an enriched pathway ID
data(precomputed_gseares, package = "GSEALens")

gsea_res <- precomputed_gseares
task <- extract_gsea_task(gsea_res, contrast_id = "untrt_vs_trt")
res_df <- as.data.frame(task$gsea_res)
top_pathway <- res_df$ID[1]

# Render the directional GSEA plot (enrichment + rank + heatmap tracks)
p <- plot_directional_gsea(task, target_pathways = top_pathway)
class(p)
```

plot_gsea_memory

*Plot GSEA Benchmark Memory Curve***Description**

Generates a standard memory usage curve plot for publication.

Usage

```
plot_gsea_memory(aligned_data, highlight_phases = TRUE)
```

Arguments

```
aligned_data    Aligned data frame returned by align_benchmark_data
highlight_phases Whether to color by phase, default TRUE
```

Value

A ggplot object

Examples

```
# Build a small synthetic aligned-data frame (mimics the output of
# align_benchmark_data() without needing an external CSV or GseaRes object).
syn <- data.frame(
  timestamp_ms = seq(0, 20000, by = 1000),
  rss_mb = c(120, 150, 220, 280, 310, 330, 340, 335, 320, 305,
             290, 270, 255, 245, 240, 235, 230, 225, 220, 215, 210),
  relative_sec = seq(0, 20, by = 1),
  phase = c(rep("startup", 4), rep("core_compute", 12), rep("cleanup", 5))
)
attr(syn, "gsea_info") <- list(duration_sec = 20, workers = 2)
class(syn) <- c("GseaBenchmarkAligned", "data.frame")
plot_gsea_memory(syn)
plot_gsea_memory(syn, highlight_phases = FALSE)
```

```
precomputed_gseares
```

Pre-computed GSEA results from the airway dataset

Description

A pre-computed ‘GseaRes’ object built from the airway dataset (DESeq2 + clusterProfiler::GSEA). Used in examples and vignettes so that users can try visualization and code-generation functions without having to re-run the full GSEA pipeline.

Format

A ‘GseaRes’ object (list-like).

Source

Built from the ‘airway’ Bioconductor dataset via DESeq2 + clusterProfiler::GSEA.

Examples

```
data(precomputed_gseares, package = "GSEAlens")
class(precomputed_gseares)
```

```
prepare_hubgene_nodes
```

Prepare HubGene Nodes for Plotting

Description

Prepare node data with layout positions for plotly visualization.

Usage

```
prepare_hubgene_nodes(network_data, layout = "fr", seed = 42)
```

Arguments

network_data	Output from build_hubgene_network
layout	Layout algorithm: "fr" (Fruchterman-Reingold), "kk" (Kamada-Kawai), or "circle"
seed	Random seed for reproducibility (default: 42)

Value

A named list with three components:

pathway	Pathway node data frame with coordinates
gene	Gene node data frame with coordinates
all	Combined node data frame

Examples

```
# Build a minimal network_data object (2 pathways + 2 shared genes)
network_data <- list(
  nodes = list(
    pathway = data.frame(
      id = c("PATHWAY_A", "PATHWAY_B"), type = "pathway",
      NES = c(2.0, -1.5), direction = c("up", "down"),
      stringsAsFactors = FALSE
    ),
    gene = data.frame(
      id = c("GENE_X", "GENE_Y"), type = "gene",
      stringsAsFactors = FALSE
    )
  ),
  edges = data.frame(
    from = c("PATHWAY_A", "PATHWAY_A", "PATHWAY_B", "PATHWAY_B"),
    to = c("GENE_X", "GENE_Y", "GENE_X", "GENE_Y"),
    stringsAsFactors = FALSE
  ),
  hub_df = data.frame(gene = c("GENE_X", "GENE_Y"), degree = c(2, 2),
    stringsAsFactors = FALSE)
)
nodes <- prepare_hubgene_nodes(network_data, layout = "fr", seed = 42)
if (!is.null(nodes)) head(nodes$all[, c("id", "type")])
```

```
preprocessed_dds
```

```
Pre-computed DESeq2 DESeqDataSet from the airway dataset
```

Description

A slimmed ‘DESeqDataSet’ object fitted with ‘DESeq()’, suitable as input to [GSEAlens::setup_gsea_env()].

Format

A ‘DESeqDataSet’ object.

Source

Built from the ‘airway’ Bioconductor dataset via DESeq2.

Examples

```
data(preprocessed_dds, package = "GSEAlens")
class(preprocessed_dds)
```

preprocessed_dds_se	<i>Pre-computed DESeqDataSet (SummarizedExperiment form)</i>
---------------------	--

Description

A ‘DESeqDataSet’ based on a ‘SummarizedExperiment’ input, suitable for demonstrating the SE-based entry point of [GSEAlens::setup_gsea_env()].

Format

A ‘DESeqDataSet’ object.

Source

Built from the ‘airway’ Bioconductor dataset via DESeq2.

Examples

```
data(preprocessed_dds_se, package = "GSEAlens")
class(preprocessed_dds_se)
```

preprocessed_limma	<i>Pre-computed limma-voom fit and DGEList from the airway dataset</i>
--------------------	--

Description

A list with elements ‘fit’ (‘MArrayLM’) and ‘gsea_limma_voom_data’ (filtered ‘DGEList’), suitable as input to [GSEAlens::setup_gsea_env()] for the limma-voom workflow.

Format

A list with elements ‘fit’ and ‘gsea_limma_voom_data’.

Source

Built from the ‘airway’ Bioconductor dataset via limma-voom.

Examples

```
data(preprocessed_limma, package = "GSEAlens")
names(preprocessed_limma)
```

read_addition_data	<i>Read Additional Data</i>
--------------------	-----------------------------

Description

Read and validate additional data from CSV or RDS file. Supports gzip-compressed formats (.gz). The data frame must contain an 'ID' column as the primary key for joining with pathway results.

Usage

```
read_addition_data(file_path)
```

Arguments

file_path	Character string specifying the file path. Supported formats: • .csv / .csv.gz - Comma-separated values (optionally gzip compressed) • .rds / .rds.gz - R serialized data (optionally gzip compressed)
-----------	---

Value

A data frame with validated ID column. Rows are deduplicated by ID, keeping the first occurrence of each unique ID.

Examples

```
# Build a small CSV in a temp directory and read it back.
tmp_csv <- file.path(tempdir(), "pathway_annotations_demo.csv")
utils::write.csv(
  data.frame(
    ID = c("HALLMARK_HYPOXIA", "HALLMARK_INFLAMMATORY_RESPONSE"),
    Brief_Description = c("Hypoxia", "Inflammatory response"),
    stringsAsFactors = FALSE
  ),
  tmp_csv, row.names = FALSE
)

add_data <- read_addition_data(tmp_csv)
add_data
```

setup_gsea_env	<i>Initialize GSEA Environment Object</i>
----------------	---

Description

Unified entry point supporting limma-voom (no intercept design) and DESeq2 objects. Automatically extracts contrasts, differential analysis results, and expression matrices to generate a standardized GseaEnv object.

Usage

```
setup_gsea_env(
  fit,
  pathway_obj,
  expr_data = NULL,
  target_factor = NULL,
  verbose = TRUE
)
```

Arguments

fit	Differential analysis object. Supports MArrayLM (limma) or DESeqDataSet (DESeq2).
pathway_obj	Gene set object. Typically generated by build_gsea_pathways().
expr_data	Optional. Expression matrix or DGEList. If NULL, extraction from fit object will be attempted (partial support only).
target_factor	Only for DESeq2. Specifies the target factor of interest. If NULL, automatically inferred as the last term in the design formula.
verbose	Logical. If TRUE (default), print progress messages. Set to FALSE for silent batch-mode use.

Value

Returns a standardized object of class GseaEnv.

Examples

```
# Limma-voom workflow using pre-computed inputs shipped with the package
data(preprocessed_limma, package = "GSEAlens")

precomp <- preprocessed_limma
data(gsea_pathwaysets_toy, package = "GSEAlens")

pathways <- gsea_pathwaysets_toy
env <- setup_gsea_env(
  fit      = precomp$fit,
  pathway_obj = pathways,
```

```
    expr_data = precomp$gsea_limma_voom_data
  )
  class(env)
```

validate_param	<i>Safe Parameter Validation with Fallback (Global Strategy)</i>
----------------	--

Description

Unified handling of invalid parameters (e.g., ncol <= 0) to ensure 03/04 module stability.

Usage

```
validate_param(
  value,
  default,
  min_val = 1,
  max_val = NULL,
  param_name = "parameter"
)
```

Arguments

value	Input value
default	Default value
min_val	Minimum value (inclusive)
max_val	Maximum value (inclusive, optional)
param_name	Parameter name (for warning messages)

Value

Validated value

Examples

```
workers <- validate_param(4, default = 2, min_val = 1, max_val = 8, param_name = "workers")
```

visualization*GSEA Visualization and Report Generation*

Description

Provides static plotting functions and interactive HTML report generation capabilities. This is a documentation landing page for the visualization family of functions in this file.

Value

NULL (this is a documentation landing page; individual functions return their own objects, documented separately).

Index

batch_calc_gsea, 4
build_edge_list_safely, 6
build_gsea_pathways, 7
build_hubgene_network, 8, 45

calculate_overlap_ratio, 9
color_by_direction, 10
creat_addition_data_rdsfile, 11
create_addition_template, 12
create_gsea_env, 13
create_gsea_res, 14
create_gsea_task, 15

extract_gsea_task, 15
extract_hub_genes, 16

generate_boxplot_data_code, 17
generate_boxplot_image_code, 18
generate_combined_plot_code, 20
generate_de_volcano_code, 21
generate_dotplot_code, 22
generate_gsea_html_report, 24
generate_hubgene_code, 25
generate_hubgene_hover_text, 26
generate_joint_canvas_code, 27
generate_network_code, 28
generate_pathway_plot_code, 29
generate_volcano_code, 30
get_contrast_registry, 32
get_core_genes_for_pathway, 32
get_core_genes_list, 33
get_de_table, 34
get_expr_matrix, 34
get_geneset_info, 35
get_hubgene_legend, 36
get_sample_meta, 37
get_term_genes, 37
gsea_pathwaysets_toy, 38
gsea_pathwaysets_toy_hallmark, 38
GSEAlens (GSEAlens-package), 3

GSEAlens-package, 3

import_gsea_capsule, 39
inspect_gsea_env, 40
inspect_gsea_res, 40

launch_gsea_app, 41

mod_ai_abs_page_server, 42

plot_directional_gsea, 43
plot_gsea_memory, 44
precomputed_gseares, 45
prepare_hubgene_nodes, 45
preprocessed_dds, 46
preprocessed_dds_se, 47
preprocessed_limma, 47

read_addition_data, 48

setup_gsea_env, 49

validate_param, 50
visualization, 51