Package 'ctdR'

November 12, 2026

Type Package

Title Enrichment Analysis of Chemical-Gene Interactions from the Comparative Toxicogenomics Database

Version 0.99.12

Description Identifies chemicals significantly associated with a set of genes using data from the Comparative Toxicogenomics Database (CTD, <https://ctdbase.org>). Supports four enrichment methods through a unified interface: Over-Representation Analysis (ORA) via a hypergeometric test on stats::phyper(), Gene Set Enrichment Analysis (GSEA) via 'fgsea', the competitive gene-set test CAMERA via 'limma', and per-sample Gene Set Variation Analysis (GSVA) via 'GSVA'. Users download the CTD chemical-gene interactions file once, import it with import_CTD(), and then run enrichment_CTD() on either a gene list (ORA, GSEA) or an expression matrix (CAMERA, GSVA). This package does not bundle or redistribute CTD data: its example interaction file is synthetic, and the tables of the replication vignette hold counts and p-values only. Users must download the required data files directly from <https://ctdbase.org> and comply with the CTD data licensing terms (see <https://ctdbase.org/about/legal.jsp>).

License Apache License 2.0 | file LICENSE

URL https://luigicorsaro.com/ctdR/, https://github.com/drake69/ctdR

BugReports https://github.com/drake69/ctdR/issues

Encoding UTF-8

Depends R (>= 4.6.0)

Imports fgsea, ggplot2, readr, org.Hs.eg.db, AnnotationDbi, limma, GSVA, stats, BiocIO, BiocFileCache, S4Vectors, SummarizedExperiment, methods

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

biocViews GeneSetEnrichment, Pathways, Annotation, GeneExpression, DataImport

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

git_url https://git.bioconductor.org/packages/ctdR

git_branch devel

git_last_commit e1d78d2

git_last_commit_date 2026-11-05

Repository Bioconductor 3.24

Date/Publication 2026-11-12

Author Luigi Corsaro [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1218-230X>)

Maintainer Luigi Corsaro <lcorsaro69@gmail.com>

Contents

ctdR-package . . . 3
.annotate_genes . . . 4
.assay_index . . . 5
.as_expr_matrix . . . 5
.attach_provenance . . . 6
.build_interaction_table . . . 6
.check_organism . . . 7
.ctd_bfc . . . 7
.ctd_cache_dir . . . 8
.ctd_cache_has . . . 8
.ctd_cache_load . . . 9
.ctd_cache_save . . . 9
.ctd_known_fields . . . 10
.ctd_license_reminder . . . 10
.ctd_provenance_cached . . . 10
.ctd_provenance_new . . . 11
.deduplicate_chemicals . . . 11
.detect_id_type . . . 12
.filter_gene_sets . . . 12
.filter_organism . . . 13
.format_enrichment_result . . . 13
.format_organism . . . 14
.gene_symbol_map . . . 15
.has_metadata_slot . . . 15
.is_remote_url . . . 16
.is_se . . . 16
.load_geneset_list . . . 17
.looks_like_pvalues . . . 17
.map_chemical_genes . . . 18
.parse_ctd_header . . . 18
.plot_camera . . . 19
.plot_gsva_heatmap . . . 20
.read_and_validate_ctd . . . 20
.resolve_alpha_column . . . 21
.resolve_ctd_source . . . 21
.run_camera . . . 22
.run_gsea . . . 23
.run_gsva . . . 24
.run_ora . . . 25

.save_ctd_cache . . . 26
.select_se_assay . . . 26
.to_symbols . . . 27
.validate_enrichment_args . . . 27
.validate_expr_matrix . . . 28
.validate_expr_rownames . . . 28
as_genesets_CTD . . . 29
CTDFile . . . 30
ctdR-cache . . . 31
ctdR-source . . . 31
ctd_cache . . . 32
ctd_provenance . . . 33
enrichment_CTD . . . 34
gsea . . . 38
import_CTD . . . 39
ora . . . 40
plot_CTD . . . 42
utils-input . . . 44

Index **45**

`ctdR-package` *ctdR: Enrichment Analysis of Chemical-Gene Interactions from CTD*

Description

ctdR identifies chemicals significantly associated with a set of genes using data from the Comparative Toxicogenomics Database (CTD, `https://ctdbase.org`).

Four enrichment methods are supported through a unified interface:

ORA Over-Representation Analysis via `phyper`.

GSEA Gene Set Enrichment Analysis via `fgsea`.

CAMERA Competitive gene-set test accounting for inter-gene correlation, via `camera`.

GSVA Per-sample Gene Set Variation Analysis via `gsva`.

ORA and GSEA take a gene-level data frame. CAMERA and GSVA take expression data, either as a numeric matrix (genes x samples) or as a `SummarizedExperiment`; GSVA returns its per-sample scores in whichever of the two it was given, so sample annotation stays attached to the results.

Quick Start

1. Download **CTD_chem_gene_ixns.csv.gz** from the CTD downloads page, `https://ctdbase.org/downloads/`, in a browser. There is no need to decompress it.
2. Import the data once: `import_CTD("path/to/CTD_chem_gene_ixns.csv.gz")`
3. Run enrichment analysis: `enrichment_CTD(my_genes, method = "ORA")`

Data Licensing Disclaimer

This package does **not** bundle, redistribute, or embed any data from the Comparative Toxicogenomics Database. CTD data are created and maintained by NC State University and are subject to specific licensing terms and conditions.

Users are solely responsible for:

- Downloading the required data files directly from https://ctdbase.org.
- Reading and accepting the CTD Terms of Service at https://ctdbase.org/about/legal.jsp before using the data.
- Complying with all applicable CTD data licensing requirements, including proper citation of CTD in any publications or derived works.

By using ctdR with CTD data, you acknowledge that you have read, understood, and agreed to the CTD Terms of Service.

Author(s)

Maintainer: Luigi Corsaro <lcorsaro69@gmail.com> (ORCID)

Authors:

- Luigi Corsaro <lcorsaro69@gmail.com> (ORCID)

References

Davis AP, Wiegers TC, Johnson RJ, Sciaky D, Wiegers J, Mattingly CJ (2023). "Comparative Toxicogenomics Database (CTD): update 2023." *Nucleic Acids Research*, 51(D1), D1257-D1262. doi:10.1093/nar/gkac833. https://ctdbase.org

See Also

import_CTD to import CTD data, enrichment_CTD to run enrichment analysis.

.annotate_genes *Annotate GSEA results with enriched gene symbols*

Description

Annotate GSEA results with enriched gene symbols

Usage

```
.annotate_genes(results, gene_table, chemical_sets)
```

Arguments

results	Data frame of enrichment results carrying a ChemicalID column.
gene_table	Data frame with an EntrezID column and a GeneLabel column holding the label to report.
chemical_sets	Named list mapping ChemicalID to the Entrez IDs of that chemical's target genes.

Value

A character vector of comma-separated gene symbols per chemical.

.assay_index *Resolve an assay selector to a positional index*

Description

Resolve an assay selector to a positional index

Usage

```
.assay_index(assays_list, assay)
```

Arguments

assays_list	The assays() list of a SummarizedExperiment.
assay	A single assay name, a positive integer index, or NULL for the first assay.

Value

An integer index into assays_list.

.as_expr_matrix *Coerce an expression input to a plain numeric matrix*

Description

Single coercion point for the matrix-based methods. Accepts either a numeric matrix (returned unchanged) or a SummarizedExperiment, from which the requested assay is extracted.

Usage

```
.as_expr_matrix(x, assay = NULL)
```

Arguments

x	A numeric matrix (genes x samples) or a SummarizedExperiment.
assay	Assay name, index, or NULL for the first. Ignored when x is already a matrix.

Value

A validated numeric matrix with genes in rows.

.attach_provenance *Attach a provenance record to a result*

Description

Uses `metadata()` on a `SummarizedExperiment`, which is the slot the class provides, and an attribute on anything else.

Usage

```
.attach_provenance(x, prov)
```

Arguments

`x`	The object to annotate.
`prov`	A `ctd_provenance` record, or `NULL` to leave `x` untouched.

Value

`x`, annotated.

.build_interaction_table *Build the long-format chemical–gene–action interaction table*

Description

For each (ChemicalID, GeneID) pair, collapses all observed InteractionActions values (pipe-separated) into a single string. This table is cached and used at enrichment time when `interaction_types` is specified.

Usage

```
.build_interaction_table(ctd_data)
```

Arguments

`ctd_data`	Filtered CTD interaction data frame.

Value

A data frame with columns `ChemicalID`, `EntrezID`, and `InteractionActions` (pipe-collapsed per pair).

.check_organism *Validate the organism argument of import_CTD()*

Description

Validate the organism argument of import_CTD()

Usage

```
.check_organism(organism)
```

Arguments

organism	"all", or NCBI Taxonomy IDs as numbers or strings.

Value

"all", or a character vector of unique taxonomy IDs.

.ctd_bfc *Open (creating if needed) the ctdR BiocFileCache*

Description

Open (creating if needed) the ctdR BiocFileCache

Usage

```
.ctd_bfc(cache_dir = .ctd_cache_dir())
```

Arguments

cache_dir	Directory backing the cache.

Value

A BiocFileCache object.

.ctd_cache_dir *Resolve the ctdR cache directory*

Description

Honours `getOption("ctdR.cache")` (used by the test suite to redirect the cache to a temporary location); otherwise the standard per-user package cache from `tools::R_user_dir()`.

Usage

```
.ctd_cache_dir()
```

Value

A single directory path (character).

.ctd_cache_has *Is a resource with this name present in the cache?*

Description

Is a resource with this name present in the cache?

Usage

```
.ctd_cache_has(bfc, rname)
```

Arguments

`bfc`	A `BiocFileCache` object.
`rname`	Resource name.

Value

A logical scalar.

.ctd_cache_load *Load a cached object by resource name*

Description

Load a cached object by resource name

Usage

```
.ctd_cache_load(bfc, rname)
```

Arguments

bfc	A BiocFileCache object.
rname	Resource name.

Value

The deserialized object; stop()s if the resource is absent.

.ctd_cache_save *Save (or overwrite) an object into the cache under a resource name*

Description

Save (or overwrite) an object into the cache under a resource name

Usage

```
.ctd_cache_save(bfc, rname, value)
```

Arguments

bfc	A BiocFileCache object.
rname	Resource name.
value	The object to serialize.

Value

Invisibly, the resource path written.

.ctd_known_fields *Column names a CTD chemical-gene interactions file must provide*

Description

Used both to locate the column-name line inside the file header and to validate the result of reading it.

Usage

```
.ctd_known_fields()
```

Value

Character vector of expected CTD field names.

.ctd_license_reminder *Emit the CTD data-licensing reminder once per session*

Description

Emit the CTD data-licensing reminder once per session

Usage

```
.ctd_license_reminder()
```

Value

Invisibly `TRUE` the first time it prints, `FALSE` after.

.ctd_provenance_cached *Read the cached provenance record, if there is one*

Description

Read the cached provenance record, if there is one

Usage

```
.ctd_provenance_cached(bfc)
```

Arguments

`bfc`	A **BiocFileCache** object.

Value

A `ctd_provenance` record, or `NULL` when the cache predates provenance tracking.

.ctd_provenance_new *Build a provenance record*

Description

Build a provenance record

Usage

```
.ctd_provenance_new(
  report_created,
  source,
  n_chemicals,
  n_interactions,
  organism = "9606"
)
```

Arguments

report_created	CTD's own release string, or NA_character_.
source	The path or URL the user supplied, before resolution.
n_chemicals	Number of chemicals retained after filtering.
n_interactions	Number of chemical-gene pairs retained.
organism	Character vector of the NCBI Taxonomy IDs kept, or "all" when every species was kept.

Value

An object of class ctd_provenance.

.deduplicate_chemicals *Deduplicate the chemical metadata table*

Description

Ensures one row per ChemicalID. Warns if the same ID appears with multiple names (CTD data quality issue — first occurrence is kept). Reports as a message if the same name is shared by multiple IDs (legitimate: parent compound and its derivatives each have distinct MeSH IDs).

Usage

```
.deduplicate_chemicals(chem_df)
```

Arguments

chem_df	Data frame with columns ChemicalID and ChemicalName.

Value

A data frame with one row per unique ChemicalID.

.detect_id_type *Detect identifier type from a character vector*

Description

Detect identifier type from a character vector

Usage

```
.detect_id_type(ids)
```

Arguments

ids	Character vector of identifiers (typically rownames(x)).

Value

Either "entrez" (if every non-empty id is purely numeric) or "symbol" otherwise.

.filter_gene_sets *Filter cached gene sets by interaction type*

Description

Loads ctd_interactions.rda and rebuilds Entrez-ID gene set lists and symbol TERM2GENE tables retaining only interactions whose InteractionActions field matches at least one of the requested types.

Usage

```
.filter_gene_sets(cache_dir, interaction_types)
```

Arguments

cache_dir	Directory holding cached CTD .rda files.
interaction_types	Character vector of interaction types to retain.

Value

A list with entrez (named list: ChemicalID → Entrez IDs) and symbols (data frame with columns term, gene).

.filter_organism *Keep the interactions of the requested species*

Description

Reports how many interactions each retained species contributes, so a multi-species import shows its composition, and stops when the request matches nothing rather than caching empty gene sets.

Usage

```
.filter_organism(ctd, organism)
```

Arguments

ctd	CTD interaction data frame (all columns character).
organism	Character vector of taxonomy IDs, or "all".

Value

The filtered data frame.

.format_enrichment_result *Canonicalize and order an enrichment result data frame*

Description

Shared post-processing for the ORA / GSEA / CAMERA runners. Each runner hands in a data frame that already has the harmonized columns ChemicalID and pvalue; this helper computes padj, joins chemical metadata, applies a canonical leading-column order, sorts by padj ascending, and drops row names. Centralizing this step keeps the runners small and the public output schema consistent across methods.

Usage

```
.format_enrichment_result(
  res,
  chemicals_meta,
  pAdjustMethod,
  method,
  rename = NULL,
  drop = NULL
)
```

Arguments

res	Data frame as returned by an enrichment engine (ora(), gsea(), limma::camera()), still carrying the engine's native column names.
chemicals_meta	Data frame with ChemicalID and ChemicalName columns.
pAdjustMethod	Method passed to p.adjust.
method	Method label (one of "ORA", "GSEA", "CAMERA") stamped into the new Method column so results can be rbind'd across methods without losing provenance.
rename	Named character vector mapping old_engine_colname = "NewCanonicalName". Applied before the metadata merge. Centralizes the engine->canonical-schema mapping so each engine can keep its native column names.
drop	Character vector of engine column names to remove before the merge / sort.

Value

A data frame whose leading columns are ChemicalID, ChemicalName, Method, PValue, PValueAdjusted (whichever of those are present after the rename), followed by the remaining columns in their original order, sorted by PValueAdjusted ascending.

.format_organism *Describe the species recorded in a provenance record*

Description

Describe the species recorded in a provenance record

Usage

```
.format_organism(organism)
```

Arguments

organism	The organism field: taxonomy IDs, "all" for every species, or NULL for a record written before the field existed, when only human interactions could be imported.

Value

A single string.

.gene_symbol_map *Symbol for every gene in a CTD extract*

Description

Without pooling (a human-only import) every gene takes its current HGNC symbol from **org.Hs.eg.db**, and a gene it does not know is dropped: the behaviour of every earlier version. With pooling, a gene known to **org.Hs.eg.db** still takes its HGNC symbol, and any other gene takes the symbol CTD gives it, upper-cased, so that genes of different species sharing a name pool into one symbol, as on the CTD website. The test is on the Entrez ID, not on the organism of the interaction: CTD records some interactions on a gene of another species than the experiment's.

Usage

```
.gene_symbol_map(ctd_data, pool_species = FALSE)
```

Arguments

ctd_data	CTD interaction data frame with GeneID and GeneSymbol.
pool_species	Logical. TRUE when species other than human were imported.

Value

Named character vector: names are Entrez IDs, values symbols.

.has_metadata_slot *Does this object carry a metadata() slot?*

Description

Annotated is the S4Vectors virtual class that provides metadata(). Both SummarizedExperiment and DataFrame extend it, so asking about the slot rather than about a specific class keeps the rule to one line: use the slot where there is one, an attribute where there is not.

Usage

```
.has_metadata_slot(x)
```

Arguments

x	Any object.

Value

TRUE when metadata() applies to x.

.is_remote_url *Is a source string a remote (downloadable) URL?*

Description

Is a source string a remote (downloadable) URL?

Usage

```
.is_remote_url(src)
```

Arguments

src	Character scalar.

Value

TRUE for http(s)/ftp URLs, otherwise FALSE.

.is_se *Test whether an object is a SummarizedExperiment*

Description

Uses methods::is() so the check works without attaching **SummarizedExperiment**, and keeps the dependency confined to the matrix-based branches.

Usage

```
.is_se(x)
```

Arguments

x	Any object.

Value

TRUE if x inherits from SummarizedExperiment.

.load_geneset_list *Load CTD gene-set list from the user cache*

Description

Load CTD gene-set list from the user cache

Usage

```
.load_geneset_list(id_type, cache_dir)
```

Arguments

id_type	Either "entrez" or "symbol".
cache_dir	Directory holding the cached .rda files.

Value

A named list where names are CTD chemical IDs and elements are character vectors of gene IDs (Entrez or HGNC SYMBOL).

.looks_like_pvalues *Do these values look like p-values?*

Description

GSEA ranks the highest value first, so a p-value column used as is turns the ranking upside down. Checking the range is the one test that needs no knowledge of where the column came from.

Usage

```
.looks_like_pvalues(values)
```

Arguments

values	Numeric vector; NA values are ignored.

Value

TRUE when values is numeric and every non-missing value lies in [0, 1], FALSE otherwise.

.map_chemical_genes *Map chemicals to gene IDs and symbols*

Description

Map chemicals to gene IDs and symbols

Usage

```
.map_chemical_genes(
  ctd_data,
  chemicals_ids,
  symbol_map = .gene_symbol_map(ctd_data)
)
```

Arguments

ctd_data	Filtered CTD interaction data frame.
chemicals_ids	Character vector of chemical IDs.
symbol_map	Named character vector, Entrez ID to symbol, as built by .gene_symbol_map().

Value

A list with entrez (named list) and symbols (data frame).

.parse_ctd_header *Parse the comment header of a CTD file*

Description

A CTD download has no header row. The field names sit *inside* the commented preamble, on the line after # Fields:, and the release date sits on the line beginning # Report created:.

Usage

```
.parse_ctd_header(file_path, min_match = 3L, n_peek = 50L)
```

Arguments

file_path	Path to the CTD CSV file.
min_match	Integer. How many known field names a commented line must list to be taken as the column-name line. Three, so that a passing mention of one name in the licence text cannot be mistaken for the real thing.
n_peek	Integer. How many lines to read from the top of the file. The header of a CTD download is 29 lines, so 50 leaves room for it to grow while keeping this a single bounded read of a file that is hundreds of megabytes on disk.

Details

The column-name line is located by looking for the commented line that lists at least `min_match` of the names in `.ctd_known_fields`, rather than by counting header lines. Counting is the more fragile assumption of the two: the package already depends on those names everywhere (`OrganismID`, `ChemicalID` and the rest are referenced throughout), so matching them adds no new dependency, whereas a hard-coded line count adds one that buys nothing. The failure modes differ too. If CTD inserts a comment line, a fixed count shifts every column silently and the analysis proceeds on misaligned data; if CTD renames a column, this search finds nothing and stops before a single record is read.

Value

A list with `fields` (character vector of column names in file order), `report_created` (the CTD release string, or `NA_character_` when the file does not carry one) and `n_comment_lines`.

`.plot_camera` *Plot CAMERA enrichment results*

Description

Plot CAMERA enrichment results

Usage

```
.plot_camera(results, type, n, title)
```

Arguments

`results`	Data frame of CAMERA results, as returned by `enrichment_CTD` with `method = "CAMERA"`.
`type`	Character. Plot shape: `"bar"`, `"dot"` or `"lollipop"`.
`n`	Integer. Number of top chemicals to show, taken after sorting by `PValueAdjusted` ascending.
`title`	Character. Plot title.

Value

A `ggplot` object.

.plot_gsva_heatmap *Plot GSVA scores as a sample x chemical heatmap*

Description

Plot GSVA scores as a sample x chemical heatmap

Usage

```
.plot_gsva_heatmap(scores, n, title)
```

Arguments

scores	Numeric matrix of GSVA scores with chemicals in rows and samples in columns.
n	Integer. Number of chemicals to show, taken by decreasing score variance across samples.
title	Character. Plot title.

Value

A ggplot object.

.read_and_validate_ctd *Read and validate a CTD CSV file*

Description

Read and validate a CTD CSV file

Usage

```
.read_and_validate_ctd(file_path, organism = "9606")
```

Arguments

file_path	Path to the CTD CSV file.
organism	Character vector of NCBI Taxonomy IDs to keep, or "all" for every species (as returned by .check_organism()).

Value

A filtered data frame of CTD interactions, carrying the parsed header in its "ctd_header" attribute.

.resolve_alpha_column *Decide which column a significance threshold applies to*

Description

A real differential-expression table has several numeric columns and the interesting one is rarely the second: `limma::topTable()` puts the log fold change there. Thresholding a position rather than a name would silently filter on the wrong quantity, so the column is named by the caller. The second column remains the fallback for the two-column case, and the choice is reported either way.

Usage

```
.resolve_alpha_column(x, alpha_column)
```

Arguments

`x`	The input data frame.
`alpha_column`	A column name, a positive index, or `NULL` to take the second column.

Value

The resolved column name.

.resolve_ctd_source *Resolve a CTD source to a readable local path*

Description

Accepts a local file path, a `file://` URI, or an `http(s)`/`ftp` URL. Remote URLs are downloaded and cached via **BiocFileCache** (emitting `.ctd_license_reminder` once). No default URL is assumed.

Usage

```
.resolve_ctd_source(src)
```

Arguments

`src`	Character scalar: local path or URL.

Value

A local, readable file path.

`.run_camera` *Competitive gene-set enrichment with CAMERA*

Description

Internal function that runs the CAMERA competitive gene-set test (`camera`) of expression data against the cached CTD chemical gene sets. Unlike ORA/GSEA — which take a single (ranked) gene list — CAMERA tests, for each chemical, whether its target genes show a stronger differential signal than the rest of the transcriptome under a user-supplied design and contrast, while accounting for inter-gene correlation.

Usage

```
.run_camera(
  expr,
  design,
  contrast,
  id_type = NULL,
  pAdjustMethod = "BH",
  chemicals_meta,
  cache_dir,
  interaction_types = NULL,
  assay = NULL,
  ...
)
```

Arguments

`expr`	Numeric expression matrix (genes x samples) or a `SummarizedExperiment`, which is reduced to the requested assay: CAMERA returns one row per chemical, so there is no per-sample output for the container to annotate. `rownames(expr)` must be Entrez IDs or HGNC symbols matching the cached CTD gene sets.
`design`	Design matrix produced e.g. by `model.matrix`.
`contrast`	Either a column number or column name of `design`, or a numeric contrast vector.
`id_type`	Either `"entrez"`, `"symbol"`, or `NULL` for auto-detection from `rownames(expr)`.
`pAdjustMethod`	Method passed to `p.adjust` to compute the `padj` column.
`chemicals_meta`	Data frame with columns `ChemicalID` and `ChemicalName`, used to annotate results.
`cache_dir`	Directory holding the cached CTD `.rda` files.
`interaction_types`	Character vector of CTD `InteractionActions` values to retain when building gene sets, or `NULL` for all.
`assay`	Assay name or index to use when `expr` is a `SummarizedExperiment`; `NULL` (default) takes the first. Ignored for a matrix.
`...`	Forwarded to `camera` (e.g. `inter.gene.cor`, `use.ranks`, `allow.neg.cor`, `trend.var`).

Value

A data frame with columns ChemicalID, ChemicalName, NGenes, Direction, Correlation (if reported), pvalue, padj, sorted by padj ascending.

.run_gsea *Run GSEA branch of enrichment analysis*

Description

Loads the cached chemical->Entrez-ID mapping, prepares the ranked input, calls the gsea engine, and feeds the result through the shared .format_enrichment_result.

Usage

```
.run_gsea(
  x,
  chemicals_meta,
  cache_dir,
  pAdjustMethod,
  interaction_types = NULL,
  gene_id_type = "symbol",
  ...
)
```

Arguments

x	Data frame with column EntrezID and a numeric value column used for ranking.
chemicals_meta	Data frame with ChemicalID and ChemicalName columns.
cache_dir	Directory holding cached CTD .rda files.
pAdjustMethod	Multiple-testing correction name.
interaction_types	Character vector of CTD InteractionActions values to retain when building gene sets, or NULL for all.
gene_id_type	Either "symbol" or "entrez": the identifier reported in the EnrichedGenes column.
...	Forwarded to gsea (e.g. minSize, maxSize).

Value

A data frame of GSEA enrichment results, formatted by .format_enrichment_result.

`.run_gsva` *Per-sample gene-set scoring with GSVA*

Description

Internal function that runs Gene Set Variation Analysis (gsva) on expression data using the cached CTD chemical gene sets. Unlike ORA/GSEA/CAMERA — which return a single p-value per chemical (group-level inference) — GSVA produces **per-sample enrichment scores**: one row per chemical, one column per sample. These scores are suitable for downstream clustering, association tests against phenotypes, survival analysis, or heatmap visualization.

A `SummarizedExperiment` is handed to GSVA unchanged rather than reduced to its assay, because GSVA is itself SE-in/SE-out: the scores then come back in a container that still carries `colData`, which is what makes per-sample scores interpretable.

Usage

```
.run_gsva(
  expr,
  id_type = NULL,
  cache_dir,
  interaction_types = NULL,
  assay = NULL,
  ...
)
```

Arguments

`expr`	Numeric expression matrix (genes x samples) or a `SummarizedExperiment`. `rownames(expr)` must be Entrez IDs or HGNC symbols matching the cached CTD gene sets.
`id_type`	Either `"entrez"`, `"symbol"`, or `NULL` for auto-detection from `rownames(expr)`.
`cache_dir`	Directory holding the cached CTD `.rda` files.
`interaction_types`	Character vector of CTD `InteractionActions` values to retain when building gene sets, or `NULL` for all.
`assay`	Assay name or index to use when `expr` is a `SummarizedExperiment`; `NULL` (default) takes the first. Ignored for a matrix.
`...`	Forwarded to `gsvaParam` (e.g. `kcdf`, `minSize`, `maxSize`, `tau`, `maxDiff`).

Value

GSVA enrichment scores with CTD chemical IDs in rows and samples in columns, in the same container as `expr`: a numeric matrix for a matrix input, a `SummarizedExperiment` (with `colData` preserved) for a `SummarizedExperiment` input.

`.run_ora` *Run ORA branch of enrichment analysis*

Description

Loads the cached chemical->gene-symbol mapping, maps Entrez IDs to symbols, runs ORA, and merges chemical names.

Usage

```
.run_ora(
  x,
  chemicals_meta,
  cache_dir,
  pAdjustMethod,
  interaction_types = NULL,
  gene_id_type = "symbol",
  universe = NULL,
  alpha = NULL,
  alpha_column = NULL,
  ...
)
```

Arguments

`x`	Data frame with column `EntrezID`.
`chemicals_meta`	Data frame with `ChemicalID` and `ChemicalName` columns.
`cache_dir`	Directory holding cached CTD `.rda` files.
`pAdjustMethod`	Multiple-testing correction name.
`interaction_types`	Character vector of CTD `InteractionActions` values to retain when building gene sets, or `NULL` for all.
`gene_id_type`	Either `"symbol"` or `"entrez"`: the identifier used to build the TERM2GENE table and reported in the `EnrichedGenes` column.
`universe`	Background gene identifiers, or `NULL` for all genes in the gene sets. Converted to match `gene_id_type`, so Entrez IDs may be supplied whichever mode is in use.
`alpha`	Significance threshold, or `NULL`. When given, `x` is taken to be the whole result table: the rows below the threshold are tested and every row is the background.
`alpha_column`	Column `alpha` applies to: a name, an index, or `NULL` for the second column.
`...`	Forwarded to `ora` (`minGSSize`, `maxGSSize`).

Value

A data frame of ORA enrichment results.

.save_ctd_cache *Save CTD cache files*

Description

Save CTD cache files

Usage

```
.save_ctd_cache(
  bfc,
  chemicals,
  entrez,
  symbols,
  interactions,
  provenance = NULL,
  symbol_map = NULL
)
```

Arguments

bfc	A BiocFileCache object (from .ctd_bfc()).
chemicals	Data frame of chemical IDs and names.
entrez	Named list of Entrez IDs per chemical.
symbols	Data frame of term-gene symbol mappings.
interactions	Long-format data frame (ChemicalID, EntrezID, InteractionActions).
provenance	A ctd_provenance record, or NULL to skip writing one (which is what a caller that has none should pass).
symbol_map	Named character vector, Entrez ID to symbol, or NULL to skip writing it.

Value

Invisible NULL.

.select_se_assay *Select one assay of a SummarizedExperiment, keeping the container*

Description

Reduces x to the single requested assay, which then becomes the first (and only) one. Methods that consume a SummarizedExperiment directly, such as gsvaParam, pick the first assay by default, so this avoids having to pass an assay name downstream and works whether or not the assays are named.

Usage

```
.select_se_assay(x, assay = NULL)
```

Arguments

x	A SummarizedExperiment.
assay	Assay name, index, or NULL for the first.

Value

x with a single assay, validated.

.to_symbols *Convert gene identifiers to HGNC symbols where possible*

Description

Entrez IDs that do not map keep their original value, matching how the input gene list is handled, so a partially mappable universe narrows the background rather than emptying it. Values that are already symbols pass through: they simply do not map, and are kept.

Usage

```
.to_symbols(ids)
```

Arguments

ids	Vector of gene identifiers, or NULL.

Value

A character vector of symbols, or NULL for NULL input.

.validate_enrichment_args *Validate user inputs for enrichment_CTD()*

Description

Checks pAdjustMethod, CTD cache presence, x shape per method, and CAMERA-specific design/contrast requirements. Centralizing here keeps enrichment_CTD() below the BiocCheck 50-line recommendation.

Usage

```
.validate_enrichment_args(
  x,
  method,
  design,
  contrast,
  pAdjustMethod,
  cache_dir
)
```

Arguments

x	The user input (data.frame, matrix, or SummarizedExperiment).
method	Already normalized via match.arg().
design	Design matrix for CAMERA (or NULL).
contrast	Contrast spec for CAMERA (or NULL).
pAdjustMethod	Multiple-testing correction name.
cache_dir	Directory expected to hold CTD .rda files.

Value

Invisibly TRUE on success; otherwise stop()s with an informative message.

.validate_expr_matrix *Validate that an expression matrix has usable rownames*

Description

Validate that an expression matrix has usable rownames

Usage

```
.validate_expr_matrix(x)
```

Arguments

x	A numeric matrix.

.validate_expr_rownames *Validate gene identifiers taken from expression rownames*

Description

Validate gene identifiers taken from expression rownames

Usage

```
.validate_expr_rownames(rn)
```

Arguments

rn	The rownames() of a matrix or SummarizedExperiment.

as_genesets_CTD *Export CTD Gene Sets for Existing Enrichment Engines*

Description

Returns the cached CTD chemical gene sets as a named list, in the shape expected by third-party Bioconductor enrichment engines — for example the `gs` argument of `EnrichmentBrowser::sbea()`, or the input to `GSEABase::GeneSetCollection()`. This lets you run the CTD gene sets through existing infrastructure instead of, or alongside, `enrichment_CTD`.

Usage

```
as_genesets_CTD(
  id_type = c("entrez", "symbol"),
  interaction_types = NULL,
  cache_dir = .ctd_cache_dir()
)
```

Arguments

`id_type`	Either `"entrez"` (default) or `"symbol"`: the identifier space of the returned genes, to match your expression data or downstream engine.
`interaction_types`	Optional character vector of CTD interaction actions (e.g. `"increases^expression"`) to retain. `NULL` (default) keeps all interactions.
`cache_dir`	Directory holding the cached CTD files. Defaults to the **ctdR** user cache populated by `import_CTD`.

Details

ctdR builds *chemical-centric* gene sets: each CTD chemical becomes a gene set of its interacting genes. `enrichment_CTD` consumes these internally; this helper exposes the same gene sets so they integrate with the wider ecosystem. Call `import_CTD` once first to populate the cache.

The returned representation is deliberately a plain named list (not a bespoke class): it is accepted directly by `EnrichmentBrowser::sbea()` and is a one-line conversion away from a `GeneSetCollection`. **ctdR** does not take a dependency on those engines — you supply them.

Value

A named list: names are CTD `ChemicalIDs`, elements are character vectors of gene identifiers (Entrez IDs or HGNC symbols).

See Also

`import_CTD` to populate the cache; `enrichment_CTD` for the built-in enrichment interface.

Examples

```
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)
gene_sets <- as_genesets_CTD("entrez")
length(gene_sets)
gene_sets[[1]]

## Not run:
# Hand the CTD gene sets to an existing engine (EnrichmentBrowser):
library(EnrichmentBrowser)
res <- sbea(method = "camera", se = my_se, gs = as_genesets_CTD("entrez"))
gsRanking(res)

## End(Not run)
```

CTDFile *CTD Chemical-Gene Interactions File*

Description

`CTDFile` is a `BiocFile` subclass representing a Comparative Toxicogenomics Database (CTD) chemical-gene interactions file (`CTD_chem_gene_ixns.csv` or its gzip-compressed `.csv.gz` form). Pair it with `import` to read the file into R following the Bioconductor **BiocIO** import/export convention, as an alternative to the side-effecting `import_CTD` caching workflow.

In **ctdR**, **CTD** always denotes the *Comparative Toxicogenomics Database* (`https://ctdbase.org`). The class is named `CTDFile` (not `CTD`) to avoid confusion with unrelated software that shares the acronym (the CRAN package **CTD** is a graph algorithm; the `ctd` object in **EWCE** is a cell-type dataset).

Usage

```
CTDFile(resource)

## S4 method for signature 'CTDFile,ANY,ANY'
import(con, format, text, ...)
```

Arguments

`resource`	Character scalar: a local path or a URL to the CTD chemical-gene interactions file.
`con`	A `CTDFile` object to import.
`format`	Ignored; present for compatibility with the **BiocIO** `import` generic.
`text`	Ignored; present for compatibility with the **BiocIO** `import` generic.
`...`	Additional arguments (currently unused).

Value

`CTDFile()` returns a `CTDFile` object. `import()` on a `CTDFile` returns a `DataFrame` of validated human (OrganismID 9606) CTD chemical-gene interactions.

Data Licensing Disclaimer

This package does **not** bundle or redistribute any CTD data. The Comparative Toxicogenomics Database is maintained by NC State University and its data are subject to specific licensing terms. Users are responsible for downloading the data directly from https://ctdbase.org and for complying with the CTD Terms of Service (https://ctdbase.org/about/legal.jsp).

See Also

`import_CTD` for the caching workflow required before `enrichment_CTD`; `BiocFile` for the parent class.

Examples

```
sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
ctd_file <- CTDFile(sample_file)
ctd_file
interactions <- BiocIO::import(ctd_file)
head(interactions)
```

---

ctdR-cache *Internal BiocFileCache-backed store for processed CTD data*

Description

Small internal wrapper around **BiocFileCache**. `import_CTD()` writes the four processed CTD objects (`chemicals`, `ChemicalName_GeneEntrezIds`, `ChemicalName_GeneSymbols`, `ctd_interactions`) as cache resources keyed by name; readers retrieve them by the same name. Replaces the previous hand-managed **rappdirs** directory of `.rda` files.

---

ctdR-source *CTD source resolution (local path or URL)*

Description

Internal helpers that let `import_CTD` and the `CTDFile` `import()` method accept either a local file path or a URL. Remote URLs are downloaded and cached with **BiocFileCache**; a one-time data-licensing reminder is emitted. No default CTD URL is assumed — the caller must supply the source, keeping the user the party who downloads under (and agrees to) the CTD terms.

ctd_cache *Retrieve Cached CTD Data*

Description

Returns a processed CTD table from the **ctdR** cache populated by `import_CTD`, without reaching into the `.rda` cache files by hand. Use this instead of `load(file.path(cache_dir, "...rda"))` when you need the chemical metadata or the raw interaction table in your own code.

For the chemical *gene sets* (the input to enrichment engines) use `as_genesets_CTD` instead.

Usage

```
ctd_cache(what = c("chemicals", "interactions"), cache_dir = .ctd_cache_dir())
```

Arguments

`what`	Which cached table to return: `"chemicals"` a data frame with one row per chemical (`ChemicalID`, `ChemicalName`). `"interactions"` the long-format interaction table (`ChemicalID`, `EntrezID`, `InteractionActions`).
`cache_dir`	Directory holding the cached CTD files. Defaults to the **ctdR** user cache populated by `import_CTD`.

Value

A data frame with the requested cached table.

See Also

`import_CTD` to populate the cache; `as_genesets_CTD` for the chemical gene sets.

Examples

```
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)
chem <- ctd_cache("chemicals")
head(chem)
```

`ctd_provenance` *Provenance of the CTD data an analysis ran on*

Description

Returns the record of which CTD release produced a result: the `Report created` date CTD stamps into its own file header, where the file came from, when it was imported, and how much of it was kept.

An analysis is only reproducible if the version of the data behind it can be named. CTD is re-released continuously and its downloads are not versioned in the filename, so the release date inside the header is the only thing that identifies which snapshot a result came from. This carries it from the file all the way to the object you report on.

Usage

```
ctd_provenance(x)

## S3 method for class 'ctd_provenance'
print(x, ...)
```

Arguments

`x`	An object returned by `enrichment_CTD`, or the `DataFrame` returned by importing a `CTDFile`. Omit it to read the record of the data currently cached, which answers "which release am I about to analyse" before any analysis has been run.
`...`	Ignored, present for compatibility with the generic.

Details

Where the record is stored depends on what the object is, because the methods do not all return the same container. Objects that provide a `metadata()` slot keep it there, which covers the `SummarizedExperiment` returned by GSVA and the `DataFrame` returned by importing a `CTDFile`. The data frames returned by ORA, GSEA and CAMERA, and a plain score matrix, have no such slot, so there it rides on an attribute. Use this accessor rather than reaching for either directly, so that code keeps working whichever method produced the object.

The record survives subsetting, ordering, `head()` and the common **dplyr** verbs. It does not survive `merge()` or `subset()`, which drop attributes; retrieve it before those if you need it afterwards.

Value

An object of class `ctd_provenance`: a list with `report_created` (the CTD release string, `NA` if the file carried none), `source`, `accessed`, `n_chemicals`, `n_interactions`, `organism` (the NCBI Taxonomy IDs kept, or `"all"`) and `ctdR_version`. Returns `NULL`, with a warning, when `x` carries no record.

See Also

`import_CTD`, which reads the record, and `enrichment_CTD`, which attaches it to its results.

Examples

```
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)

genes <- data.frame(
    EntrezID = c("7124", "3569", "7157", "672", "1956"),
    pvalue = c(0.001, 0.003, 0.01, 0.02, 0.05)
)
# Which release is cached, before running anything:
ctd_provenance()

res <- enrichment_CTD(genes, method = "ORA")
ctd_provenance(res)
```

`enrichment_CTD` *Chemical Enrichment Analysis Using CTD*

Description

Identifies chemicals whose known gene targets are significantly enriched in a user-supplied gene list or expression matrix, using data from the Comparative Toxicogenomics Database (CTD).

Four methods are available:

ORA Over-Representation Analysis (default). Tests whether the overlap between your gene list and each chemical's target genes is larger than expected by chance. Uses a hypergeometric test computed directly on `phyper`. Input: data frame with column `EntrezID` (character or numeric Entrez gene IDs) and an optional numeric value column.

GSEA Gene Set Enrichment Analysis. Uses a ranked gene list (ranked by the numeric column in the input, e.g. p-values or fold changes) to detect chemicals whose targets cluster toward the top or bottom of the ranking. Uses `fgsea`.

CAMERA Competitive gene-set test accounting for inter-gene correlation. Uses `camera`. Input: a numeric expression matrix (genes x samples) or a `SummarizedExperiment`, plus a design matrix and a contrast.

GSVA Gene Set Variation Analysis (sample-level scoring). Returns per-sample enrichment scores for each chemical, suitable for downstream clustering or association testing. Uses `gsva`. Input: a numeric expression matrix (genes x samples) or a `SummarizedExperiment`, which is also the container the scores are returned in.

Usage

```
enrichment_CTD(
  x,
```

```
  method = c("ORA", "GSEA", "CAMERA", "GSVA"),
  design = NULL,
  contrast = NULL,
  id_type = NULL,
  pAdjustMethod = "BH",
  interaction_types = NULL,
  gene_id_type = c("symbol", "entrez"),
  universe = NULL,
  alpha = NULL,
  alpha_column = NULL,
  assay = NULL,
  ...
)
```

Arguments

x	The input. Its expected type depends on `method`: • For `"ORA"` and `"GSEA"`: a data frame with at least two columns, `EntrezID` (character or numeric Entrez gene IDs) and a numeric value column (e.g. p-value). For ORA this is either the genes you have already selected, in which case say what the background was with `universe`, or the whole result table with `alpha` to select from it. For GSEA, an optional column named `stat` can be added with a signed ranking statistic (e.g. the moderated t-statistic from `limma::eBayes()`); when present it is used directly for ranking, preserving directionality and avoiding ties. When absent, the second column must hold p-values: it is transformed via `-log10()` with a warning, scored one-sided (`scoreType = "pos"`), and values outside [0, 1] stop the run. The second column is ignored by ORA. • For `"CAMERA"` and `"GSVA"`: either a numeric expression matrix with genes in rows and samples in columns, or a `SummarizedExperiment` whose assay holds that matrix (see `assay` to choose which one). `rownames(x)` must be either Entrez IDs or HGNC SYMBOLs.
method	Character. Enrichment method: `"ORA"` (default), `"GSEA"`, `"CAMERA"`, or `"GSVA"`.
design	Design matrix (required when `method = "CAMERA"`).
contrast	Contrast specification for `camera` (column number, column name, or numeric vector). Required when `method = "CAMERA"`.
id_type	Either `"entrez"`, `"symbol"`, or `NULL` (default) for auto-detection from `rownames(x)`. Only used when `method` is `"CAMERA"` or `"GSVA"`.
pAdjustMethod	Character. Multiple-testing correction applied to the raw p-values, passed to `p.adjust`. Any value in `stats::p.adjust.methods` is accepted: `"holm"`, `"hochberg"`, `"hommel"`, `"bonferroni"`, `"BH"` (Benjamini-Hochberg, the default), `"BY"`, `"fdr"` (alias for `"BH"`), or `"none"`. See `p.adjust` for the meaning of each method. Not used for `method = "GSVA"` (which returns per-sample scores, not p-values).
interaction_types	Character vector of CTD `InteractionActions` values to retain when building gene sets, or `NULL` (default) to use all cached interactions. Values follow the `verb\^{}noun` convention used by CTD, e.g. `"increases\^{}expression"`, `"decreases\^{}expression"`, `"affects\^{}binding"`. A gene is included in a chemical's set if *any* of its recorded interaction actions matches one of the specified types. Requires that `import_CTD()` has been run (the filter is applied to

the cached `ctd_interactions.rda` file). Restricting to expression interactions is recommended for RNA-seq analyses to improve biological specificity.

`gene_id_type` Character. Identifier type used in the `EnrichedGenes` output column: `"symbol"` (default) returns HGNC gene symbols with Entrez ID as fallback for unmapped genes; `"entrez"` skips the symbol lookup and returns Entrez IDs directly. Only applies to `"ORA"` and `"GSEA"`; ignored by `"CAMERA"` and `"GSVA"`.

`universe` Background gene identifiers for `"ORA"`: every gene that entered your differential test, not only the significant ones and not every gene sequenced. `NULL` (default) falls back to every gene in the CTD gene sets, and says so, because the package cannot know what your platform measured.

This is the input that decides whether an ORA result means anything. A background wider than what the experiment could detect fills the urn with genes that could never have been drawn, so the overlap looks more selective than it was and p-values come out too small. The error is anti-conservative. On the RNA-seq example bundled with this package, the fallback background returns 32 chemicals at FDR < 0.05 and the correct one returns 19.

ORA is the only method that needs this argument, and the reason is the shape of its input. A bare gene list carries no record of what was measurable, so the background has to be supplied separately. GSEA ranks the whole list you give it, which is already the background; `"CAMERA"` and `"GSVA"` intersect the gene sets with `rownames(x)`, so theirs is the set of measured genes by construction. Passing `universe` to any of those three raises a warning rather than being quietly dropped.

`alpha` Significance threshold for `"ORA"`, or `NULL` (default). Supplying it says that `x` is the *whole* result table rather than a list already filtered: the rows whose second column falls below `alpha` become the genes to test, and every row becomes the background.

Every row of the table becomes the background, the selected genes included: a hypergeometric test draws n genes from an urn of N, and the drawn ones were in the urn. The background is not the non-significant remainder.

This is the safer way to run ORA, because the background is then derived from the same object as the gene list and cannot disagree with it. Filtering first and passing `universe` separately asks the caller to reconnect two things that were together a moment earlier, and that reconnection is what goes wrong.

Name the column to threshold with `alpha_column`: which p-value to judge on is the researcher's decision, not the package's. The function reports the column it used, how many genes passed and how many form the background.

Mutually exclusive with `universe`, and ignored by the other three methods, which already hold their own background.

`alpha_column` Which column `alpha` applies to: a name, a positive index, or `NULL` (default) for the second column. Naming it matters on a real result table, where the second column is usually a fold change: `limma::topTable()` puts `logFC` there. Pass the whole table and say which p-value to judge on, `alpha_column = "padj"` or `"adj.P.Val"`, rather than cutting the table down to two columns first.

`assay` Which assay to use when `x` is a `SummarizedExperiment`: an assay name, a positive index, or `NULL` (default) for the first assay. Ignored when `x` is a matrix or a data frame.

`...` Additional arguments forwarded to the underlying engine: `ora` for ORA (`universe`, `minGSSize`, `maxGSSize`; both thresholds are chosen for CTD rather than inherited, `minGSSize` defaulting to 2 and `maxGSSize` to `Inf`, see `ora` for the mea-

surements behind them), fgseaMultilevel for GSEA (e.g. minSize, maxSize, nproc), camera for CAMERA, gsva for GSVA (e.g. minSize, maxSize).

Details

Before calling this function you must import the CTD data once with import_CTD. If the cached data is not found, the function stops with an informative error message.

Value

- For "ORA", "GSEA", and "CAMERA": a data frame of enrichment results sorted by PValueAdjusted ascending. All three methods share the leading columns ChemicalID, ChemicalName, Method, PValue, PValueAdjusted; method-specific extras follow (see the package vignette for the full per-method schema).
- For "GSVA": enrichment scores with chemicals (CTD chemical IDs) in rows and samples in columns. The container follows the input: a matrix in returns a numeric matrix, while a SummarizedExperiment in returns a SummarizedExperiment whose assay holds the scores and whose colData is carried over from the input, so sample annotation stays attached to the results.

Data Licensing Disclaimer

This package does **not** bundle or redistribute any CTD data. The Comparative Toxicogenomics Database is maintained by NC State University and its data are subject to specific licensing terms. Users must download the data directly from https://ctdbase.org and comply with the CTD Terms of Service (https://ctdbase.org/about/legal.jsp).

See Also

import_CTD to import and cache the CTD data; plot_CTD to visualize results.

Examples

```
# Import the bundled sample data first:
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)

# ORA / GSEA: prepare a gene list with Entrez IDs and a numeric value
genes <- data.frame(
    EntrezID = c("7124", "3569", "7157", "672", "1956"),
    pvalue = c(0.001, 0.003, 0.01, 0.02, 0.05)
)
ora_results <- enrichment_CTD(genes, method = "ORA")
gsea_results <- enrichment_CTD(genes, method = "GSEA")

# CAMERA / GSVA: expression data plus, for CAMERA, design + contrast.
# Uses the bundled GSE311566 subset, a SummarizedExperiment
```

```
# (Dex vs DMSO, female PBMCs; see inst/extdata/README.md).
se <- readRDS(system.file(
    "extdata", "GSE311566_subset.rds", package = "ctdR"
))
d <- model.matrix(~ se$group)
camera_results <- enrichment_CTD(se, method = "CAMERA",
    design = d, contrast = 2)

# GSVA: SummarizedExperiment in, SummarizedExperiment out,
# so the sample annotation stays attached to the scores.
gsva_scores <- enrichment_CTD(se, method = "GSVA")
SummarizedExperiment::colData(gsva_scores)$group
```

`gsea` *Gene Set Enrichment Analysis (GSEA)*

Description

Internal engine that performs Gene Set Enrichment Analysis via `fgsea`. Detects chemicals whose known target genes cluster toward the extremes of a ranked gene list.

This function returns the raw GSEA result with two GSEA-specific decorations (`Enriched_GENE`) and harmonized column names (`ChemicalID`, `pvalue`). The adjusted-p, chemical-name join, column ordering and sort are applied downstream by `.format_enrichment_result` so that the schema stays consistent across ORA / GSEA / CAMERA.

Usage

```
gsea(ChemicalName_GeneEntrezIds, gene_table, ...)
```

Arguments

`ChemicalName_GeneEntrezIds`
: A named list where each element is a character vector of Entrez gene IDs associated with a CTD chemical. Names are CTD chemical IDs.

`gene_table`
: A data frame with at least two columns:

  `EntrezID` Entrez gene IDs (character).

  (second column) P-values, used as the `-log10(value)` fallback ranking when no `stat` column is present. Values outside [0, 1] stop the run: a fold change there would be ranked upside down, and its negative values dropped.

  `stat` **(optional)** Numeric ranking statistic used directly by `fgsea` (higher = ranked first). Recommended: the moderated t-statistic from `limma::eBayes()` or `-log10(p) * sign(logFC)`. When present, suppresses the directionality warning. A `stat` column whose values all lie in [0, 1] raises a warning, because p-values put there would rank the least significant genes first.

`...`
: Additional arguments forwarded to `fgseaMultilevel` (e.g. `nproc` to set the number of parallel workers, `eps` for p-value precision). With the `-log10(p)` fallback every value is positive, so `scoreType` defaults to `"pos"` unless given here.

Value

A data frame with `fgsea::fgseaMultilevel`'s native columns (`pathway`, `pval`, `ES`, `NES`, `size`, `leadingEdge`) plus two ctdR-added columns (`Enriched_GENE`). Not yet sorted and not yet joined with chemical metadata; pass to `.format_enrichment_result` for the canonical shape.

`import_CTD` *Import CTD Chemical-Gene Interaction Data*

Description

Parses, filters, and caches the CTD chemical-gene interactions file so that it can be used by `enrichment_CTD`. This function must be called **once** before running any enrichment analysis.

The raw data file must be downloaded manually from the CTD website. The required file is **CTD_chem_gene_ixns.csv.gz**, available at `https://ctdbase.org/reports/CTD_chem_gene_ixns.csv.gz`.

Usage

```
import_CTD(file_path, organism = 9606)
```

Arguments

`file_path`	Character. A local path **or** a URL to the CTD chemical-gene interactions file (`CTD_chem_gene_ixns.csv` or `CTD_chem_gene_ixns.csv.gz`). Remote URLs are downloaded and cached with **BiocFileCache**; the package assumes no default URL, so you supply the source and a one-time CTD data-licensing reminder is shown on first remote fetch.
`organism`	NCBI Taxonomy IDs of the species to keep, as numbers or character strings. Defaults to `9606` (*Homo sapiens*). `"all"` keeps every species and forces symbol-keyed gene sets to pool species by gene symbol (see the Species section). The choice is recorded in the provenance of the import (see `ctd_provenance`).

Details

Processing steps performed by `import_CTD`:

1. Reads the file (plain or gzip-compressed), taking the column names from its commented header.
2. Keeps the interactions of the species named in `organism`, **Homo sapiens** only (OrganismID 9606) by default.
3. For each chemical, collects the associated Entrez gene IDs.
4. Maps Entrez IDs to HGNC gene symbols via **org.Hs.eg.db**.
5. Saves four cached objects via **BiocFileCache**: `chemicals`, `ChemicalName_GeneEntrezIds`, `ChemicalName_GeneSymbols`, and `ctd_interactions` (a long-format table of chemical–gene–action triples used by `enrichment_CTD(interaction_types = ...)`).

The cache is managed by **BiocFileCache** under `tools::R_user_dir("ctdR", "cache")`. To re-import (e.g. after downloading a newer CTD release), simply call `import_CTD()` again — existing cache resources are overwritten.

Filtering by interaction type (e.g., to retain only `"increases^expression"` interactions) is done at enrichment time via the `interaction_types` argument of `enrichment_CTD`, not here — so a single import supports any combination of filters without re-running this step.

Value

Invisible NULL. Called for its side effect of caching the processed data.

Data Licensing Disclaimer

This package does **not** bundle or redistribute any CTD data. The Comparative Toxicogenomics Database is maintained by NC State University and its data are subject to specific licensing terms. Users are responsible for downloading the data directly from https://ctdbase.org and for complying with the CTD Terms of Service (https://ctdbase.org/about/legal.jsp). By using this function you acknowledge that you have read and accepted those terms.

Species

By default only human interactions are kept, as in every earlier version. organism accepts other NCBI Taxonomy IDs (for example 10090 for mouse, 10116 for rat), several at once, or "all" for every species in the file, including the interactions CTD records without a species.

When species other than human are imported, and always with organism = "all", the symbol-keyed gene sets are **forced** to pool genes across species **by gene symbol** (orthology by homonymy); there is no option to turn it off: a mouse *Esr1* interaction counts as *ESR1*. This is how the CTD website aggregates genes, and what published analyses based on it did; it is a match by name, not an orthology mapping. Human genes take their symbol from **org.Hs.eg.db** as before; genes of other species take the symbol CTD gives them, in upper case. The Entrez-keyed gene sets are not pooled: each interaction keeps the Entrez ID of the gene CTD curated it on, so analyses on Entrez IDs (gene_id_type = "entrez", and GSEA) match a non-human gene only to an input in the same species.

See Also

enrichment_CTD for running enrichment analysis after import, including the interaction_types filter.

Examples

```
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)
```

---

ora *Over-Representation Analysis (ORA)*

Description

Internal engine that performs Over-Representation Analysis with a hypergeometric test computed directly on `phyper`. Returns a result table with one row per tested chemical.

Column renaming, multiple-testing correction, chemical-name join, canonical ordering and sort are applied downstream by `.format_enrichment_result` so the engine stays close to the vocabulary of the test itself.

Usage

```
ora(
  ChemicalName_GeneSymbols,
  gene_symbols,
  pAdjustMethod = "BH",
  universe = NULL,
  minGSSize = 2,
  maxGSSize = Inf
)
```

Arguments

`ChemicalName_GeneSymbols`
: A data frame with two columns (`term`, `gene`) mapping CTD chemical IDs to HGNC gene symbols. The first two columns are used, whatever their names.

`gene_symbols`
: Character vector of HGNC gene symbols to test for enrichment. Duplicates and `NA` are removed.

`pAdjustMethod`
: Character. Method for multiple testing correction (default `"BH"`). Passed to `p.adjust`.

`universe`
: Vector of background gene identifiers, or `NULL` (default) to use every gene in `ChemicalName_GeneSymbols`. Set it to `rownames(expr)` or to the full tested gene list to restrict the background to measured genes only. Coerced with `as.character()`, so an integer column of Entrez IDs works as it does for `gene_symbols`.

  The default is a fallback, not a recommendation, and the function says so when it uses it. A hypergeometric test asks how many of n genes drawn from N would land in a set. Genes that your experiment could never have detected still sit in N, filling the urn with balls that cannot be drawn, so the draw looks more selective than it was and the p-value comes out too small. The error is anti-conservative: it manufactures significance.

  The right universe is every gene that entered your test, not every gene you sequenced and not only the significant ones: a gene filtered out for low expression could not have been selected, so it does not belong in the urn either. On the RNA-seq example bundled with this package the difference is 32 significant chemicals against 19.

`minGSSize`
: Integer. Minimum gene set size after intersection with the background (default 2). The default is chosen for CTD, where the median chemical has 4 target genes: a one-gene set is degenerate, since its p-value equals the ratio of input genes to background whichever gene it contains, so it measures membership rather than enrichment.

`maxGSSize`
: Maximum gene set size after intersection with the background. The default is `Inf`, that is no upper limit, and that too is a choice made for CTD rather than inherited.

The reasoning mirrors the one for `minGSSize`, and reaches the opposite conclusion. A set of M genes cannot, even when every one of the m input genes falls inside it, produce a p-value below $C(M,m)/C(N,m) \approx (M/N)^m$. That floor rises with M, so a large enough set becomes untestable. Measured on CTD, with 28,571 genes in the background and an input list of 169, the largest still-testable set is about 26,600 genes, 93% of the universe. The largest chemical in CTD has 16,536. No chemical is untestable from above, so an upper cut removes sets that could have been declared significant.

What it removes is not marginal. A cut at 500 excludes 265 chemicals, among them benzo(a)pyrene, valproic acid, sodium arsenite, bisphenol A, aflatoxin B1 and particulate matter. Their sets are large because the literature on them is large: in CTD size tracks how well studied a chemical is, where in GO a large term is one that has stopped meaning anything. Keeping them costs 3% more tests, 8,235 against 7,970.

Set it to a finite value if you have a reason of your own; ctdR does not impose one.

One consequence to be aware of: without a cap the significant results skew towards large sets. This is not the fold enrichment talking, which moves the other way, since M sits in its denominator. It is that a p-value measures how unlikely an excess is, and the excess is counted in genes. With 154 input genes from a background of 27,444, a fold of 1.5 means 0.4 genes above expectation for a set of 100 (p = 0.43) and 45 genes above for a set of 16,000 (p = 1.4e-15). Sort by `p.adjust` to rank by evidence, and read `foldEnrichment` beside it to see how sharp the association is; neither answers the question alone.

Details

Earlier versions delegated this step to the **clusterProfiler** package. That pulled in 59 packages, and a visualization layer this package never used, to reach a single call. The test itself is one line of `stats`, so it is computed here and that dependency is gone. The p-values are unchanged: the two implementations were compared over 24 configurations of input list, minimum set size and background universe, and the largest absolute difference was exactly 0.

The background is every gene appearing in `ChemicalName_GeneSymbols`, optionally narrowed by `universe`. Gene sets are intersected with that background *before* the size filter, so `minGSSize` and `maxGSSize` always refer to the set as actually tested rather than to its nominal size.

Value

A data frame with columns `ChemicalID`, `GeneRatio`, `BgRatio`, `pvalue`, `p.adjust`, `geneID`, `Count` and `foldEnrichment`, sorted by `pvalue` ascending. Returns an empty data frame with the same structure when no gene set can be tested. Emits a message reporting how many chemicals the size filter left untested, since those are absent from the result rather than present with a large p-value.

`plot_CTD` *Plot Chemical Enrichment Results*

Description

Visualizes the output of `enrichment_CTD`. The function auto-detects the enrichment method from the input class and column structure:

- **ORA / GSEA** (data frame with `Count` or `NES`): bar or dot plot of fold enrichment.
- **CAMERA** (data frame with `Direction` and `NGenes`): bar or dot plot of $-\log_{10}(\text{padj})$, colored by direction of enrichment.
- **GSVA** (numeric matrix `chemicals` x `samples`, or a `SummarizedExperiment` wrapping one): heatmap of per-sample enrichment scores for the top-N chemicals selected by score variance across samples.

Usage

```
plot_CTD(results, type = "bar", n = 20, title = NULL)
```

Arguments

`results`	A data frame, a numeric matrix, or a `SummarizedExperiment` returned by `enrichment_CTD`.
`type`	Character. Plot type for tabular results: `"bar"` (default) or `"dot"`. Ignored for GSVA score input.
`n`	Integer. Number of top chemicals to display (default 20). Selection criterion: ascending `padj` for tabular results, descending score variance across samples for GSVA matrices.
`title`	Character. Plot title. If `NULL` (default), a title is generated automatically based on the detected method.

Value

A `ggplot2` object.

Examples

```
# Examples write to a temporary cache, so running them cannot
# disturb CTD data you have already imported. Set the same option
# yourself to keep an analysis isolated from your main cache.
options(ctdR.cache = tempfile())

sample_file <- system.file(
    "extdata", "CTD_chem_gene_ixns_sample.csv",
    package = "ctdR"
)
import_CTD(sample_file)

# ORA / GSEA
genes <- data.frame(
    EntrezID = c("7124", "3569", "7157", "672", "1956"),
    pvalue = c(0.001, 0.003, 0.01, 0.02, 0.05)
)
ora_results <- enrichment_CTD(genes, method = "ORA")
plot_CTD(ora_results, type = "bar")
plot_CTD(ora_results, type = "dot")

# GSVA scores: the heatmap accepts either the score matrix or the
# SummarizedExperiment that enrichment_CTD() returns for an SE input.
se <- readRDS(system.file(
    "extdata", "GSE311566_subset.rds", package = "ctdR"
))
gsva_scores <- enrichment_CTD(se, method = "GSVA")
```

```
plot_CTD(gsva_scores)
```

`utils-input` *Internal input helpers for ctdR*

Description

Shared helpers used by the matrix-based enrichment methods (CAMERA, GSVA): identifier type detection from rownames and gene-set loading from the CTD cache.

Index

∗ **internal**
.annotate_genes, 4
.as_expr_matrix, 5
.assay_index, 5
.attach_provenance, 6
.build_interaction_table, 6
.check_organism, 7
.ctd_bfc, 7
.ctd_cache_dir, 8
.ctd_cache_has, 8
.ctd_cache_load, 9
.ctd_cache_save, 9
.ctd_known_fields, 10
.ctd_license_reminder, 10
.ctd_provenance_cached, 10
.ctd_provenance_new, 11
.deduplicate_chemicals, 11
.detect_id_type, 12
.filter_gene_sets, 12
.filter_organism, 13
.format_enrichment_result, 13
.format_organism, 14
.gene_symbol_map, 15
.has_metadata_slot, 15
.is_remote_url, 16
.is_se, 16
.load_geneset_list, 17
.looks_like_pvalues, 17
.map_chemical_genes, 18
.parse_ctd_header, 18
.plot_camera, 19
.plot_gsva_heatmap, 20
.read_and_validate_ctd, 20
.resolve_alpha_column, 21
.resolve_ctd_source, 21
.run_camera, 22
.run_gsea, 23
.run_gsva, 24
.run_ora, 25
.save_ctd_cache, 26
.select_se_assay, 26
.to_symbols, 27
.validate_enrichment_args, 27
.validate_expr_matrix, 28
.validate_expr_rownames, 28
ctdR-cache, 31
ctdR-source, 31
gsea, 38
ora, 40
utils-input, 44
.CTDFile *(*CTDFile*)*, 30
.annotate_genes, 4
.as_expr_matrix, 5
.assay_index, 5
.attach_provenance, 6
.build_interaction_table, 6
.check_organism, 7
.ctd_bfc, 7
.ctd_cache_dir, 8
.ctd_cache_has, 8
.ctd_cache_load, 9
.ctd_cache_save, 9
.ctd_known_fields, 10, *19*
.ctd_license_reminder, 10, *21*
.ctd_provenance_cached, 10
.ctd_provenance_new, 11
.deduplicate_chemicals, 11
.detect_id_type, 12
.filter_gene_sets, 12
.filter_organism, 13
.format_enrichment_result, 13, *23*, *38, 39*, *41*
.format_organism, 14
.gene_symbol_map, 15
.has_metadata_slot, 15
.is_remote_url, 16
.is_se, 16
.load_geneset_list, 17
.looks_like_pvalues, 17
.map_chemical_genes, 18
.parse_ctd_header, 18
.plot_camera, 19
.plot_gsva_heatmap, 20
.read_and_validate_ctd, 20
.resolve_alpha_column, 21
.resolve_ctd_source, 21

.run_camera, 22
.run_gsea, 23
.run_gsva, 24
.run_ora, 25
.save_ctd_cache, 26
.select_se_assay, 26
.to_symbols, 27
.validate_enrichment_args, 27
.validate_expr_matrix, 28
.validate_expr_rownames, 28

as_genesets_CTD, 29, *32*

BiocFile, *30, 31*

camera, *3*, *22*, *34, 35*, *37*
ctd_cache, 32
ctd_provenance, 33, *39*
CTDFile, 30, *33*
CTDFile-class *(*CTDFile*)*, 30
ctdR *(*ctdR-package*)*, 3
ctdR-cache, 31
ctdR-package, 3
ctdR-source, 31

DataFrame, *31*

enrichment_CTD, *4*, *19*, *29*, *31*, *33*, 34, *39, 40*, *42, 43*

fgsea, *3*, *34*, *38*
fgseaMultilevel, *37, 38*

gsea, *23*, 38
gsva, *3*, *24*, *34*, *37*
gsvaParam, *24*, *26*

import, *30*
import,CTDFile,ANY,ANY-method *(*CTDFile*)*, 30
import_CTD, *4*, *29–33*, *37*, 39

model.matrix, *22*

ora, *25*, *36*, 40

p.adjust, *14*, *22*, *35*, *41*
phyper, *3*, *34*, *41*
plot_CTD, *37*, 42
print.ctd_provenance *(*ctd_provenance*)*, 33

SummarizedExperiment, *3*, *5*, *16*, *22*, *24*, *33–37*, *43*

utils-input, 44