Package: CONCERTDR (via r-universe)

November 25, 2026

Type Package

Title Drug Response Data Analysis Using CMap Database

Version 1.0.0

Description Provides a complete pipeline for drug repurposing analysis using the Connectivity Map (CMap) L1000 database. Functionality includes database subsetting, reference profile extraction from GCTX files, signature matching via Kolmogorov-Smirnov, GSEA-based, cosine similarity, and other methods, result annotation against CMap compound metadata, z-score matrix extraction, and publication-quality clustermap visualization via ComplexHeatmap. Connectivity scoring methods (KS, GSEA, XCos, XSum, Zhang) are implemented internally based on the algorithms described in Lamb et al. (2006), Subramanian et al. (2005), Cheng et al. (2014), and Zhang & Gant (2008), removing the need for external GitHub-only dependencies. CamSum, a correlation-adjusted sum score with an analytic p-value, is also provided.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/DavidsonGroup/CONCERTDR

BugReports https://github.com/DavidsonGroup/CONCERTDR/issues

Depends R (>= 4.5.0)

Imports data.table, matrixStats, rhdf5, stats, utils

Suggests knitr, rmarkdown, testthat, BiocStyle, ComplexHeatmap, circlize, ggplot2, RecordLinkage

VignetteBuilder knitr

Config/testthat/edition 3

biocViews GeneExpression, Pharmacogenomics, Software, Transcriptomics, DataImport

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libssl-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:53 UTC

RemoteUrl https://github.com/bioc/CONCERTDR

RemoteRef RELEASE_3_24

RemoteSha 672802e128ea278eb10004e90a658ceaf65488fb

Contents

annotate_drug_results . . . 2
compute_camsum_rho . . . 4
create_signature_from_gene_lists . . . 5
extract_cmap_data_from_siginfo . . . 6
extract_compound_id . . . 8
extract_signature_zscores . . . 9
fuzzy_drug_match . . . 11
plot.cmap_signature_result . . . 12
plot_signature_direction_tile_barcode . . . 13
print.cmap_signature_result . . . 16
process_combinations_file . . . 17
process_signature_with_df . . . 18
subset_siginfo_beta . . . 21
summary.cmap_signature_result . . . 22

Index **24**

`annotate_drug_results` *Annotate CONCERTDR Results with Drug Information*

Description

Integrate signature matching results with annotations from `siginfo_beta.txt` and `compoundinfo_beta.txt`. It produces three analysis-ready tables and one drug context summary table.

Usage

```
annotate_drug_results(
  results_df,
  sig_info_file,
  comp_info_file,
  output_file = NULL,
  score_col = "Score",
  padj_col = "pAdjValue",
  p_col = "pValue",
  compound_col = "compound",
  keep_dose_in_drug_view = FALSE,
  output_dir = NULL,
```

```
  write_outputs = FALSE,
  verbose = TRUE,
  direction = c("reversal", "mimic")
)
```

Arguments

results_df	Data frame containing signature matching results with a 'compound' column
sig_info_file	Path to siginfo_beta.txt file or data frame with signature information
comp_info_file	Path to compound information file or data frame
output_file	Deprecated single-file output path; when provided, tech_view_all is also written as CSV for compatibility.
score_col	Score column in results_df (default: "Score")
padj_col	Adjusted p-value column name (default: "pAdjValue")
p_col	Raw p-value column name (default: "pValue")
compound_col	Compound/sig-id column in results_df (default: "compound")
keep_dose_in_drug_view	Logical; whether to keep dose_uM in wetlab_drug_view (default: FALSE)
output_dir	Optional directory to write four TSV outputs. If NULL, files are not written unless write_outputs = TRUE.
write_outputs	Logical; write four TSV outputs (default: FALSE)
verbose	Logical; whether to print progress messages (default: TRUE)
direction	"reversal" (default) puts the lowest scores first in the wetlab views and picks each drug's lowest-scoring context as its best; "mimic" uses the highest. It only orders and picks; no row is dropped by the sign of its score.

Details

When compoundinfo contains multiple rows for one pert_id, all distinct nonblank target and MOA values are retained, separated by "; ". This does not duplicate signature results or inflate context counts. Targets and MOAs are independent lists, not positional pairs.

Value

A named list with:

wetlab_drug_view	Drug-focused wetlab view
wetlab_gene_view	Gene-focused wetlab view
tech_view_all	Full technical table with all integrated fields
drug_context_summary	Per-drug context summary
output_files	Named character vector of written files (if any)

Examples

```
ex_results <- data.frame(
  compound = "CVD001_HEPG2_6H:BRD-K03652504-001-01-9:10.0497",
  Score = -0.72,
  pValue = 0.002,
  pAdjValue = 0.02
)
ex_siginfo <- data.frame(
  sig_id = ex_results$compound,
  pert_type = "trt_cp",
  pert_id = "BRD-K03652504-001-01-9",
  pert_iname = "imatinib"
)
ex_compinfo <- data.frame(
  pert_id = "BRD-K03652504-001-01-9",
  pert_name = "imatinib",
  cmap_name = "imatinib"
)
views <- annotate_drug_results(
  results_df = ex_results,
  sig_info_file = ex_siginfo,
  comp_info_file = ex_compinfo,
  write_outputs = FALSE,
  verbose = FALSE
)
names(views)


if (file.exists("sig_match_xsum_results.csv") &&
    file.exists("siginfo_beta.txt") &&
    file.exists("compoundinfo_beta.txt")) {
  results <- read.csv("sig_match_xsum_results.csv")

  views <- annotate_drug_results(
    results_df = results,
    sig_info_file = "siginfo_beta.txt",
    comp_info_file = "compoundinfo_beta.txt",
    output_dir = "results",
    write_outputs = TRUE
  )

  head(views$wetlab_drug_view)
}
```

`compute_camsum_rho` *Mean inter-gene correlation of a query across a reference library*

Description

Computes $\bar{\rho}$, the mean off-diagonal Pearson correlation of the query genes measured across the profiles of `refMatrix`, after dividing each profile by its standard deviation over all genes and negating the down-regulated rows. The $k \times k$ correlation matrix is never formed: two passes over column blocks use the identity $1'R1 = \mathrm{Var}_n(w'(y_n - \bar{y}))$ with $w_g = 1/\mathrm{sd}_g$.

Usage

```
compute_camsum_rho(refMatrix, queryUp, queryDown, chunk = 2000L)
```

Arguments

`refMatrix`	Numeric matrix (or data frame) with genes as rows and profiles as columns, with row and column names.
`queryUp`, `queryDown`	Character vectors of up- and down-regulated gene symbols. Genes present in both are dropped from both.
`chunk`	Number of columns processed at a time (default: 2000).

Details

CamSum's p-values were calibrated with $\bar{\rho}$ estimated on the whole treatment library of one perturbation type (for example all `trt_oe` or all `trt_xpr` profiles). `process_signature_with_df` does this automatically from the GCTX file when the CMap file options are set; this function estimates $\bar{\rho}$ on the matrix it is given.

Value

A single numeric value, $\bar{\rho}$, with attributes `k`, `kU`, `kD` (query genes found in `refMatrix`) and `n_profiles` (profiles used). `NA` when fewer than two query genes vary across the library.

Examples

```
set.seed(1)
ref <- matrix(rnorm(200 * 50), 200, 50,
              dimnames = list(paste0("g", 1:200), paste0("s", 1:50)))
compute_camsum_rho(ref, queryUp = paste0("g", 1:10),
                   queryDown = paste0("g", 11:20))
```

`create_signature_from_gene_lists`

Create a Signature Data Frame from Gene Lists

Description

Converts separate lists of up-regulated and down-regulated genes into a signature data frame compatible with process_signature_with_df and other CONCERTDR functions. This is a convenience wrapper for users who only have gene lists (e.g.\ from a pathway database) rather than continuous fold-change values.

Usage

```
create_signature_from_gene_lists(
  up_genes,
  down_genes,
  up_value = 1,
  down_value = -1
)
```

Arguments

up_genes	Character vector of up-regulated gene symbols.
down_genes	Character vector of down-regulated gene symbols.
up_value	Numeric value to assign to up-regulated genes (default: 1).
down_value	Numeric value to assign to down-regulated genes (default: -1).

Value

A data frame with columns Gene and log2FC, suitable for use as the signature_file argument in process_signature_with_df and extract_signature_zscores.

Examples

```
sig_df <- create_signature_from_gene_lists(
  up_genes   = c("TP53", "MYC", "BRCA1"),
  down_genes = c("EGFR", "VEGFA", "KRAS")
)
head(sig_df)
```

---

extract_cmap_data_from_siginfo

Extract CMap Data Using Siginfo File Directly

Description

Extract expression data from CMap GCTX files using all signatures present in the provided siginfo file. This function processes all signatures found in the siginfo file without requiring a configuration file.

Usage

```
extract_cmap_data_from_siginfo(
  siginfo_file = "siginfo_beta.txt",
  geneinfo_file = "geneinfo_beta.txt",
  gctx_file = "level5_beta_trt_cp_n720216x12328.gctx",
  max_signatures = NULL,
  filter_quality = TRUE,
  verbose = TRUE,
  landmark = TRUE
)
```

Arguments

`siginfo_file`	Path to the signature info file (can be original or pre-filtered)
`geneinfo_file`	Path to the gene info file
`gctx_file`	Path to the GCTX file
`max_signatures`	Integer; maximum number of signatures to process (default: NULL for all)
`filter_quality`	Logical; whether to keep only high-quality signatures (`is_hiq == 1`) (default: TRUE)
`verbose`	Logical; whether to print progress messages (default: TRUE)
`landmark`	Logical; whether to restrict to landmark genes only (default: TRUE)

Details

Duplicate gene symbols are resolved by preferring landmark, then best inferred, then inferred features. Other feature spaces rank last and ties retain the first input row. A warning lists discarded and retained gene IDs. Gene symbols are not renamed or averaged. GCTX data are read in sorted index order and restored to the requested gene and signature order explicitly.

Value

A data frame with expression data for all signatures, with annotation columns indicating sample metadata. Metadata is stored as an attribute.

Examples

```
# Build a tiny GCTX file so the example is self-contained and runnable.
gctx_file <- tempfile(fileext = ".gctx")
rhdf5::h5createFile(gctx_file)
for (group in c("0", "0/DATA", "0/DATA/0", "0/META",
                "0/META/ROW", "0/META/COL")) {
  rhdf5::h5createGroup(gctx_file, group)
}
rhdf5::h5write(matrix(c(1.2, -0.4, 0.7, -1.1), nrow = 2),
               gctx_file, "0/DATA/0/matrix")
rhdf5::h5write(c("1", "2"), gctx_file, "0/META/ROW/id")
rhdf5::h5write(c("SIG_A", "SIG_B"), gctx_file, "0/META/COL/id")
```

```
siginfo <- data.frame(
  sig_id = c("SIG_A", "SIG_B"), is_hiq = 1,
  pert_itime = "24 h", pert_idose = "1 uM", cell_iname = "A375"
)
geneinfo <- data.frame(
  gene_id = c("1", "2"), gene_symbol = c("GENE1", "GENE2"),
  feature_space = "landmark"
)
reference_df <- extract_cmap_data_from_siginfo(
  siginfo_file = siginfo,
  geneinfo_file = geneinfo,
  gctx_file = gctx_file,
  verbose = FALSE
)
reference_df
attr(reference_df, "metadata")
unlink(gctx_file)
```

`extract_compound_id` *Extract Compound ID from Signature Matching Results*

Description

Helper function to extract compound identifiers from complex compound strings in signature matching results. This function can be customized based on the specific format of compound identifiers in your data.

Usage

```
extract_compound_id(
  compound_strings,
  method = "split_colon",
  regex_pattern = NULL,
  part_index = 2
)
```

Arguments

`compound_strings`	Vector of compound identifier strings
`method`	Method for extraction: "split_colon" (default), "split_underscore", or "regex"
`regex_pattern`	Regular expression pattern for extraction (used when method="regex")
`part_index`	Which part to extract when splitting (default: 2)

Value

Vector of extracted compound identifiers

Examples

```
extract_compound_id("A:BRD-K03652504-001-01-9:10")


# Example compound strings
compounds <- c("CVD001_HEPG2_6H:BRD-K03652504-001-01-9:10.0497",
               "CVD001_HEPG2_6H:BRD-A37828317-001-03-0:10")

# Extract using colon splitting (default)
ids <- extract_compound_id(compounds)

# Extract using custom regex
ids <- extract_compound_id(compounds, method = "regex",
                           regex_pattern = "BRD-[A-Z0-9-]+")
```

`extract_signature_zscores` *Extract z-score matrix for barcode heatmap*

Description

Performs all data-loading and GCTX-extraction steps needed for `plot_signature_direction_tile_barcode` without producing any plot. The returned matrix can be inspected, filtered, exported, or passed directly to `plot_signature_direction_tile_barcode(precomputed = )` to avoid re-reading large files on repeated calls.

Usage

```
extract_signature_zscores(
  results_df,
  signature_file,
  reference_df = NULL,
  gctx_file = NULL,
  geneinfo_file = NULL,
  siginfo_file = NULL,
  data_dir = getOption("CONCERTDR.data_dir", NULL),
  selected_drug = NULL,
  selected_drug_col = NULL,
  pert_id_col = "sig_id",
  score_col = "Score",
  max_genes = 100,
  max_perts = 60,
  split_direction = FALSE,
  output_zscores = NULL,
  verbose = TRUE,
  direction = c("reversal", "mimic")
)
```

Arguments

`results_df`	Data frame containing at least perturbation id and score columns.
`signature_file`	Path to a signature file with gene and log2FC columns, or a data frame with `Gene` and `log2FC` columns.
`reference_df`	Optional reference matrix with genes as rows and perturbation ids as columns. If supplied, z-scores are taken directly from this object and no GCTX file is required.
`gctx_file`	Optional path to GCTX expression file.
`geneinfo_file`	Optional path to geneinfo file.
`siginfo_file`	Optional path to siginfo file for readable labels.
`data_dir`	Optional directory containing CMap files.
`selected_drug`	Optional drug identifier to filter `results_df`.
`selected_drug_col`	Column used with `selected_drug`.
`pert_id_col`	Perturbation id column (default: `"sig_id"`).
`score_col`	Score column (default: `"Score"`).
`max_genes`	Maximum number of signature genes to use (default: 100). When `split_direction = FALSE` (default) the first `max_genes` rows of the signature file are taken. When `split_direction = TRUE` the top `max_genes` up-regulated genes (by log2FC) and the top `max_genes` down-regulated genes (by |log2FC|) are selected independently, so the matrix may contain up to `2 * max_genes` gene columns in total.
`max_perts`	Maximum number of perturbations (default: 60).
`split_direction`	Logical; if `TRUE`, apply `max_genes` separately to the up-regulated (log2FC > 0) and down-regulated (log2FC < 0) gene sets rather than to the combined signature. The result contains up to `2 * max_genes` columns ordered down-regulated then up-regulated, matching the layout expected by `plot_signature_direction_tile_barcode` with `split_direction = TRUE`. Default: `FALSE`.
`output_zscores`	Optional TSV path to save the matrix. `NULL` to skip.
`verbose`	Logical; print progress messages.
`direction`	Order in which perturbations are picked by score: `"reversal"` (default) lowest first, `"mimic"` highest first. No perturbation is excluded by the sign of its score. Precomputed matrices are plotted as supplied; choose the direction when extracting them.

Value

A named list with elements:

`z_plot` Numeric matrix — rows = perturbations (labelled `cmap_name | dose | time | cell`), cols = genes (down→up order).

`ordered_genes` Character vector of gene symbols.

logfc_map Named numeric vector of signature log2FC values.

sig_ids Character vector of selected sig_ids.

sig_labels Character vector of human-readable labels.

sig_scores Numeric vector of score_col values in the same order as sig_ids and the rows of z_plot.

Examples

```
is.function(extract_signature_zscores)


# Requires the full CMap GCTX file (downloaded from clue.io)
sig_file <- system.file("extdata", "example_signature.txt",
                        package = "CONCERTDR")
ref_file <- system.file("extdata", "example_reference_df.csv",
                        package = "CONCERTDR")
ref_df <- read.csv(ref_file, row.names = 1, check.names = FALSE)
ref_df$gene_symbol <- rownames(ref_df)
demo_sig <- colnames(ref_df)[1]
zmat <- extract_signature_zscores(
  results_df     = data.frame(sig_id = demo_sig, Score = -0.72),
  signature_file = sig_file,
  reference_df   = ref_df,
  pert_id_col    = "sig_id"
)
```

---

fuzzy_drug_match *Fuzzy String Matching for Drug Names*

Description

Performs fuzzy string matching to find the best matches between query drug names and a reference list of drug names. Uses Levenshtein distance for similarity calculation.

Usage

```
fuzzy_drug_match(
  query_names,
  reference_names,
  method = "levenshtein",
  threshold = 80,
  top_n = 1
)
```

Arguments

query_names	Vector of query drug names
reference_names	Vector of reference drug names to match against
method	Similarity method: "levenshtein" (default), "jaro", or "jarowinkler"
threshold	Minimum similarity threshold (0-100)
top_n	Number of top matches to return for each query (default: 1)

Value

Data frame with query names, matched names, and similarity scores

Examples

```
if (requireNamespace("RecordLinkage", quietly = TRUE)) {
  fuzzy_drug_match(c("asprin"), c("aspirin", "ibuprofen"), threshold = 70)
}


if (requireNamespace("RecordLinkage", quietly = TRUE)) {
  query <- c("aspirin", "ibuprofen", "acetaminophen")
  reference <- c("aspirin", "ibuprofen", "acetaminophen", "naproxen", "diclofenac")
  matches <- fuzzy_drug_match(query, reference, threshold = 80)
  print(matches)
}
```

---

plot.cmap_signature_result

Plot method for cmap_signature_result objects

Description

Plot method for cmap_signature_result objects

Usage

```
## S3 method for class 'cmap_signature_result'
plot(x, method = NULL, plot_type = "scores", top_n = 20, ...)
```

Arguments

x	A cmap_signature_result object
method	Which method to plot (default: first method in results)
plot_type	Type of plot: "scores", "volcano", or "heatmap" (default: "scores")
top_n	Number of compounds to include (default: 20)
...	Additional arguments passed to plotting functions

Value

A ggplot object

Examples

```
mock_res <- structure(
  list(
    results = list(
      ks = data.frame(compound = c("imatinib", "dasatinib", "doxorubicin"),
                      Score = c(-0.72, -0.45, 0.31),
                      pValue = c(0.002, 0.041, 0.280),
                      pAdjValue = c(0.020, 0.205, 0.560),
                      stringsAsFactors = FALSE)
    ),
    summary = data.frame(compound = "imatinib", method = "ks",
                         Score = -0.72, pValue = 0.002, global_rank = 1L,
                         stringsAsFactors = FALSE),
    gene_data    = data.frame(Gene = c("TP53", "MYC"),
                              log2FC = c(1.5, -2.1), stringsAsFactors = FALSE),
    settings     = list(signature_file = "ex.txt",
                        time_completed = Sys.time(), time_taken_mins = 0.1,
                        methods = "ks", permutations = 10),
    common_genes = list(up   = list(found = "TP53", count = 1L, percent = 50),
                        down = list(found = "MYC",  count = 1L, percent = 50))
  ), class = "cmap_signature_result"
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(mock_res, method = "ks", plot_type = "scores")
}
```

---

plot_signature_direction_tile_barcode

Plot 2D Barcode Heatmap from GCTX (Single Drug Context)

Description

Build a 2D barcode heatmap using z-scores extracted from a GCTX file. Genes are ordered by signature direction (down then up), and perturbations are chosen as top `max_perts` rows by best (lowest) `score_col` from the provided technical results table.

Usage

```
plot_signature_direction_tile_barcode(
  results_df = NULL,
  signature_file = NULL,
  reference_df = NULL,
  gctx_file = NULL,
  geneinfo_file = NULL,
```

```
  siginfo_file = NULL,
  data_dir = getOption("CONCERTDR.data_dir", NULL),
  selected_drug = NULL,
  selected_drug_col = NULL,
  pert_id_col = "sig_id",
  score_col = "Score",
  max_genes = 100,
  max_perts = 60,
  cluster_rows = TRUE,
  cluster_method = "complete",
  show_row_dendrogram = TRUE,
  cluster_cols = FALSE,
  cluster_method_cols = "complete",
  show_col_dendrogram = TRUE,
  split_direction = FALSE,
  gap_width = 5,
  precomputed = NULL,
  save_png = FALSE,
  output_png = "barcode_heatmap.png",
  output_zscores = NULL,
  width = NULL,
  height = NULL,
  dpi = 150,
  verbose = TRUE,
  direction = c("reversal", "mimic")
)
```

Arguments

results_df	Data frame (typically tech_view_all) containing at least perturbation id and score columns.
signature_file	Path to a signature file with gene and log2FC columns, or a data frame with Gene and log2FC columns.
reference_df	Optional reference matrix with genes as rows and perturbation ids as columns. If supplied, no GCTX file is needed and the heatmap is built directly from this matrix.
gctx_file	Optional path to GCTX expression file. If NULL, the function tries (in order): option CONCERTDR.gctx_file, env var CONCERTDR_GCTX_FILE, data_dir/level5_beta_all_n120194 then a file with that name in the working directory.
geneinfo_file	Optional path to geneinfo file with gene_symbol and gene_id columns. If NULL, the function tries option CONCERTDR.geneinfo_file, env var CONCERTDR_GENEINFO_FILE, data_dir/geneinfo_beta.txt, then geneinfo_beta.txt in the working directory.
siginfo_file	Optional path to siginfo file for readable perturbation labels. If NULL, the function tries option CONCERTDR.siginfo_file, env var CONCERTDR_SIGINFO_FILE, data_dir/siginfo_beta.txt, then siginfo_beta.txt in the working directory.

data_dir	Optional directory containing CMap files. Useful to avoid repeatedly passing full file paths.
selected_drug	Optional drug identifier to filter results_df before selecting perturbations.
selected_drug_col	Optional column used with selected_drug. If NULL, auto-detect from perturbation_name, display_name, pert_id, cmap_name, pert_name.
pert_id_col	Perturbation id column in results_df (default: "sig_id"; falls back to "compound" for older results).
score_col	Score column in results_df (default: "Score").
max_genes	Maximum number of signature genes to use (default: 100).
max_perts	Maximum number of perturbations to show (default: 60).
cluster_rows	Logical; cluster perturbation rows (default: TRUE).
cluster_method	Clustering linkage method for row clustering (default: "complete").
show_row_dendrogram	Logical; whether to draw a row dendrogram when cluster_rows = TRUE (default: TRUE).
cluster_cols	Logical; cluster gene columns (default: FALSE - genes are kept in signature direction order: down then up).
cluster_method_cols	Clustering linkage method for column clustering (default: "complete").
show_col_dendrogram	Logical; whether to draw a column dendrogram when cluster_cols = TRUE (default: TRUE).
split_direction	Logical; if TRUE, split the heatmap into two side-by-side panels — up-regulated genes (log2FC > 0) on the left and down-regulated genes (log2FC < 0) on the right — with a small gap between them. Row order is determined by clustering the full matrix so both panels stay aligned. Default: FALSE (original single-panel behaviour).
gap_width	Width of the gap between the two panels in mm when split_direction = TRUE. Default: 5.
precomputed	Optional; the output of extract_signature_zscores. If provided, skips all data-loading and GCTX-extraction steps - all data-related arguments (results_df, signature_file, gctx_file, etc.) are ignored.
save_png	Logical; save PNG output (default: FALSE).
output_png	Output PNG path.
output_zscores	Optional TSV path for exported z-score matrix. Default is NULL, so no file is written unless explicitly requested.
width, height	Figure dimensions in inches for PNG. If NULL (default), dimensions are computed automatically from the number of genes and perturbations (~0.22 in/gene width, ~0.28 in/perturbation height).
dpi	PNG resolution.
verbose	Logical; print progress messages.

`direction` Order in which perturbations are picked by score: `"reversal"` (default) lowest first, `"mimic"` highest first. No perturbation is excluded by the sign of its score. Precomputed matrices are plotted as supplied; choose the direction when extracting them.

Value

Invisibly returns a list containing the plotted matrix, selected perturbation ids, and file paths.

Examples

```
is.function(plot_signature_direction_tile_barcode)


if (requireNamespace("ComplexHeatmap", quietly = TRUE) &&
    requireNamespace("circlize", quietly = TRUE)) {
  sig_file <- system.file("extdata", "example_signature.txt",
                          package = "CONCERTDR")
  ref_df <- read.csv(system.file("extdata", "example_reference_df.csv",
                                 package = "CONCERTDR"),
                     row.names = 1, check.names = FALSE)
  ref_df$gene_symbol <- rownames(ref_df)
  demo_sig <- colnames(ref_df)[1]
  plot_signature_direction_tile_barcode(
    results_df    = data.frame(sig_id = demo_sig, Score = -0.72),
    signature_file = sig_file,
    reference_df  = ref_df,
    pert_id_col   = "sig_id"
  )
}
```

---

print.cmap_signature_result

Print method for cmap_signature_result objects

Description

Print method for cmap_signature_result objects

Usage

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

Arguments

`x`	A cmap_signature_result object
`...`	Additional arguments (not used)

Value

x invisibly

Examples

```
mock_res <- structure(
  list(
    results = list(
      ks = data.frame(compound = c("imatinib", "dasatinib"),
                      Score = c(-0.72, -0.45), pValue = c(0.002, 0.041),
                      pAdjValue = c(0.020, 0.205), stringsAsFactors = FALSE)
    ),
    summary = data.frame(compound = "imatinib", method = "ks",
                         Score = -0.72, pValue = 0.002, global_rank = 1L,
                         stringsAsFactors = FALSE),
    gene_data    = data.frame(Gene = c("TP53", "MYC"),
                              log2FC = c(1.5, -2.1), stringsAsFactors = FALSE),
    settings     = list(signature_file = "ex.txt",
                        time_completed = Sys.time(), time_taken_mins = 0.1,
                        methods = "ks", permutations = 10),
    common_genes = list(up   = list(found = "TP53", count = 1L, percent = 50),
                        down = list(found = "MYC",  count = 1L, percent = 50))
  ), class = "cmap_signature_result"
)
print(mock_res)
```

---

process_combinations_file

Process Multiple Combinations of Time, Dose, and Cell Line

Description

Process Multiple Combinations of Time, Dose, and Cell Line

Usage

```
process_combinations_file(
  combinations_file,
  task_id = NULL,
  geneinfo_file = "geneinfo_beta.txt",
  siginfo_file = "siginfo_beta.txt",
  gctx_file = "level5_beta_trt_cp_n720216x12328.gctx",
  output_dir = "."
)
```

Arguments

combinations_file	Path to a file containing combinations to process
task_id	Specific task ID to process (for SLURM array jobs)
geneinfo_file	Path to the gene info file
siginfo_file	Path to the signature info file
gctx_file	Path to the GCTX file
output_dir	Directory to save output files

Value

List of processed files (invisibly)

Examples

```
combinations <- data.frame(
  itime = "24 h",
  idose = "1 uM",
  cell = "K562",
  stringsAsFactors = FALSE
)
combinations_file <- tempfile(fileext = ".tsv")
utils::write.table(
  combinations,
  combinations_file,
  sep = "\t",
  row.names = FALSE,
  quote = FALSE
)
file.exists(combinations_file)

# With real CMap files available locally, run:
# process_combinations_file(
#   combinations_file = combinations_file,
#   geneinfo_file = "path/to/geneinfo_beta.txt",
#   siginfo_file = "path/to/siginfo_beta.txt",
#   gctx_file = "path/to/level5_beta_all_n1201944x12328.gctx",
#   output_dir = tempdir()
# )
```

---

process_signature_with_df

Process drug response signatures against reference data in a dataframe

---

Description

Process drug response signatures against reference data in a dataframe

Usage

```
process_signature_with_df(
  signature_file,
  reference_df,
  output_dir = "results",
  permutations = 100,
  methods = c("ks", "xcos", "xsum", "gsea0", "gsea1", "gsea2", "zhang"),
  topN = 4,
  read_method = "auto",
  save_files = FALSE,
  camsum_alternative = c("two.sided", "greater", "less"),
  camsum_rho_bar = NULL,
  direction = c("reversal", "mimic"),
  vectorized = TRUE
)
```

Arguments

signature_file	Path to a signature gene list file with log2FC values, or a data frame with columns Gene and log2FC. When a data frame is supplied the read_method argument is ignored. You can create a suitable data frame from gene lists using create_signature_from_gene_lists.
reference_df	Dataframe containing reference data (combined across all conditions)
output_dir	Directory for output files (default: "results")
permutations	Number of permutations for statistical testing (default: 100)
methods	Vector of method names to run (default: the seven permutation methods) Options: "ks", "xcos", "xsum", "gsea0", "gsea1", "gsea2", "zhang", "camsum". "camsum" is not run by default; add it explicitly.
topN	Integer; number of top and bottom genes retained from each reference profile for XCos and XSum (default: 4). Does not truncate the query.
read_method	Character; method to use for reading signature file ("auto", "fread", or "read.table") (default: "auto")
save_files	Logical; whether to save results to files (default: FALSE)
camsum_alternative	Alternative hypothesis for the CamSum analytic p-value: "two.sided" (default, same convention as the permutation methods), "greater" (mimic) or "less" (reversal). Ignored by other methods.
camsum_rho_bar	Optional mean inter-gene correlation for CamSum, used for all profiles. If NULL (default), CamSum estimates it on every profile of the same perturbation type in the GCTX file (options CONCERTDR.gctx_file, CONCERTDR.siginfo_file and CONCERTDR.geneinfo_file must be set; the type is read from the "metadata" attribute of reference_df or from siginfo), and otherwise on reference_df with a warning. Ignored by other methods.
direction	Ordering of hits: "reversal" (default) ranks the lowest scores first, "mimic" the highest first. It only orders rank, the summary and plot(); no result is

dropped by the sign of its score, and p-values and `camsum_alternative` are unaffected. Cross-method `global_rank` compares each hit's rank percentile among all profiles of its method, not raw score magnitudes.

`vectorized` Whether the seven permutation methods use the vectorised engine (default: `TRUE`); see `score_ks`. It only changes how the scores are computed, not the formulas, and is skipped automatically when `reference_df` has missing values or duplicated genes. Use `FALSE` to reproduce results of earlier versions exactly.

Details

CamSum uses the analytic null described in `compute_camsum_rho`; `topN` and `permutations` do not apply to it. Its p-values were calibrated with $\bar{\rho}$ estimated on whole LINCS `trt_oe` / `trt_xpr` libraries and are anti-conservative on `trt_cp`. `settings$camsum$rho_source` records where $\bar{\rho}$ came from for each perturbation type.

Value

A structured list containing:

`results`	List containing results for each method
`summary`	Data frame of top hits across all methods
`gene_data`	Original signature gene data
`settings`	List of parameters used for the analysis. When CamSum is run, `settings$camsum` holds k, kU, kD, alternative, and rho_bar, VIF, rho_source and rho_n_profiles per perturbation type.
`common_genes`	Counts of genes found in reference data

Use print() or summary() methods to get a quick overview of results

Examples

```
sig_file <- system.file("extdata", "example_signature.txt",
                        package = "CONCERTDR")
ref_file <- system.file("extdata", "example_reference_df.csv",
                        package = "CONCERTDR")
ref_df <- read.csv(ref_file, row.names = 1, check.names = FALSE)
ref_df$gene_symbol <- rownames(ref_df)
results <- process_signature_with_df(
  signature_file = sig_file,
  reference_df = ref_df,
  output_dir = tempdir(),
  permutations = 10,
  methods = "ks",
  save_files = FALSE
)
summary(results, top_n = 3)

# You can also pass a data.frame directly:
sig_df <- read.delim(sig_file)
results2 <- process_signature_with_df(
```

```
  signature_file = sig_df,
  reference_df = ref_df,
  permutations = 10,
  methods = "ks",
  save_files = FALSE
)
```

`subset_siginfo_beta` *Subset siginfo_beta file interactively with smart preview*

Description

Enhanced interactive filtering that shows what options will be available in subsequent parameters based on your current selection, helping you make informed decisions about the filtering sequence.

Usage

```
subset_siginfo_beta(
  siginfo_file,
  output_file = NULL,
  interactive = TRUE,
  filters = NULL,
  verbose = TRUE,
  show_preview = TRUE
)
```

Arguments

`siginfo_file`	Path to siginfo_beta.txt file
`output_file`	Path to save the filtered siginfo file (optional)
`interactive`	Logical; whether to run in interactive mode (default: TRUE)
`filters`	List of pre-defined filters to apply in non-interactive mode
`verbose`	Logical; whether to print progress messages (default: TRUE)
`show_preview`	Logical; whether to show preview of subsequent options (default: TRUE)

Value

A filtered data frame of the siginfo file

Examples

```
ex_sig <- data.frame(
  pert_type = c("trt_cp", "trt_cp", "trt_sh"),
  pert_itime = c("6 h", "24 h", "24 h"),
  pert_idose = c("10 uM", "10 uM", "5 uM"),
  cell_iname = c("A375", "MCF7", "A375")
```

```
)
ex_sig_file <- tempfile(fileext = ".txt")
write.table(ex_sig, ex_sig_file, sep = "\t", row.names = FALSE, quote = FALSE)
subset_siginfo_beta(
  ex_sig_file,
  interactive = FALSE,
  filters = list(pert_type = "trt_cp"),
  verbose = FALSE,
  show_preview = FALSE
)


ex_file <- system.file("extdata", "example_siginfo.txt", package = "CONCERTDR")

# Interactive mode (run only in an interactive R session)
if (interactive() && nzchar(ex_file)) {
  filtered_siginfo <- subset_siginfo_beta(ex_file)
}

# Non-interactive mode with pre-defined filters
if (nzchar(ex_file)) {
  filtered_siginfo <- subset_siginfo_beta(
    ex_file,
    interactive = FALSE,
    filters = list(
      pert_type = "trt_cp",
      pert_itime = c("6 h", "24 h"),
      pert_idose = "10 uM",
      cell_iname = c("A375", "MCF7")
    ),
    verbose = FALSE,
    show_preview = FALSE
  )
}
```

`summary.cmap_signature_result` *Summary method for cmap_signature_result objects*

Description

Summary method for cmap_signature_result objects

Usage

```
## S3 method for class 'cmap_signature_result'
summary(object, top_n = 10, ...)
```

Arguments

object	A cmap_signature_result object
top_n	Number of top compounds to show (default: 10)
...	Additional arguments (not used)

Value

A data frame with the top compounds

Examples

```
mock_res <- structure(
  list(
    results = list(
      ks = data.frame(compound = c("imatinib", "dasatinib"),
                      Score = c(-0.72, -0.45), pValue = c(0.002, 0.041),
                      pAdjValue = c(0.020, 0.205), stringsAsFactors = FALSE)
    ),
    summary = data.frame(compound = c("imatinib", "dasatinib"),
                         method = c("ks", "ks"),
                         Score = c(-0.72, -0.45), pValue = c(0.002, 0.041),
                         global_rank = 1:2, stringsAsFactors = FALSE),
    gene_data    = data.frame(Gene = c("TP53", "MYC"),
                              log2FC = c(1.5, -2.1), stringsAsFactors = FALSE),
    settings     = list(signature_file = "ex.txt",
                        time_completed = Sys.time(), time_taken_mins = 0.1,
                        methods = "ks", permutations = 10),
    common_genes = list(up   = list(found = "TP53", count = 1L, percent = 50),
                        down = list(found = "MYC",  count = 1L, percent = 50))
  ), class = "cmap_signature_result"
)
summary(mock_res, top_n = 2)
```

Index

annotate_drug_results, 2

compute_camsum_rho, 4, *20*
create_signature_from_gene_lists, 5, *19*

extract_cmap_data_from_siginfo, 6
extract_compound_id, 8
extract_signature_zscores, *6*, 9, *15*

fuzzy_drug_match, 11

plot.cmap_signature_result, 12
plot_signature_direction_tile_barcode, *9, 10*, 13
print.cmap_signature_result, 16
process_combinations_file, 17
process_signature_with_df, *5, 6*, 18

subset_siginfo_beta, 21
summary.cmap_signature_result, 22