Package: coMMpass (via r-universe)

May 13, 2026

Title MMRF CoMMpass Data Analysis Pipeline

Version 0.1.0

Description A reproducible data acquisition and analysis pipeline for the MMRF CoMMpass study using Targets and Nix.

License MIT + file LICENSE

URL https://JohnGavin.github.io/coMMpass-analysis/, https://github.com/JohnGavin/coMMpass-analysis

BugReports https://github.com/JohnGavin/coMMpass-analysis/issues

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder quarto, knitr

Imports TCGAbiolinks, S4Vectors, SummarizedExperiment, edgeR, logger, dplyr, aws.s3, arrow, DBI, duckdb, dbplyr, cli, jsonlite, glue, stringr, tibble

Suggests anndataR, AnnotationDbi, SingleCellExperiment, apeglm, quarto, targets, tarchetypes, crew, DESeq2, fgsea, GenomicDataCommons, ggplot2, limma, msigdbr, org.Hs.eg.db, scales, survival, tidyverse, knitr, rmarkdown, pkgdown, testthat (>= 3.0.0), gert, gh, nanoparquet, DT, htmltools, plotly, plumber, visNetwork, here, dagitty, ggdag, naniar, brms, bayesplot, posterior, splines2

biocViews Software

Additional_repositories https://bioc.r-universe.dev

Config/testthat/edition 3

Config/pak/sysreqs
cmake libicu-dev libpng-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://johngavin.r-universe.dev

Date/Publication 2026-05-13 16:51:00 UTC

RemoteUrl https://github.com/JohnGavin/coMMpass-analysis

RemoteRef HEAD

RemoteSha 78011582e659df92455d7cb1c50c5c329c359f3c

Contents

`acquire_commpass_data` *Main data acquisition function*

Description

Orchestrates the download of CoMMpass data from various sources including RNA-seq data from GDC, clinical data, and optionally AWS data.

Usage

```
acquire_commpass_data(
  download_rnaseq = TRUE,
  download_clinical = TRUE,
  download_aws = FALSE,
  sample_limit = 200,
  random_sample = TRUE,
  seed = 42,
  use_parquet = TRUE
)
```

Arguments

`download_rnaseq`	Whether to download RNA-seq data
`download_clinical`	Whether to download clinical data
`download_aws`	Whether to download from AWS
`sample_limit`	Limit number of samples (NULL for all)
`random_sample`	If TRUE, randomly sample patients
`seed`	Random seed for sampling
`use_parquet`	If TRUE, save parquet files

Value

List of file paths to the downloaded data

See Also

Other data-acquisition: `download_clinical_data()`, `download_gdc_rnaseq()`, `download_s3_subset()`, `get_commpass_clinical()`, `list_s3_commpass()`, `query_commpass_rna()`

Examples

```
## Not run:
# Download only clinical data
results <- acquire_commpass_data(
  download_rnaseq = FALSE,
  download_clinical = TRUE,
  download_aws = FALSE
)

## End(Not run)
```

`annotate_de_results` *Add gene symbol column to DE results table*

Description

Add gene symbol column to DE results table

Usage

```
annotate_de_results(de_results, annotation)
```

Arguments

`de_results`	Data frame with Ensembl IDs as rownames
`annotation`	Data frame from annotate_genes()

Value

DE results with gene_symbol column prepended

See Also

Other pathway: `annotate_genes()`, `plot_enrichment_barplot()`, `plot_enrichment_dotplot()`, `plot_gsea_running_score()`, `run_gsea()`, `run_ora()`, `run_pathway_analysis()`

annotate_genes *Annotate Ensembl gene IDs with symbols and descriptions*

Description

Maps Ensembl gene IDs to HGNC symbols and Entrez IDs. Uses msigdbr gene mapping as the primary source (always available with msigdbr). Falls back to org.Hs.eg.db + AnnotationDbi if msigdbr is not installed.

Usage

```
annotate_genes(ensembl_ids)
```

Arguments

ensembl_ids	Character vector of Ensembl gene IDs (versions stripped)

Value

Data frame with columns: ensembl_gene, gene_symbol, entrez_gene. Unmappable IDs have NA for symbol/entrez.

See Also

Other pathway: annotate_de_results(), plot_enrichment_barplot(), plot_enrichment_dotplot(), plot_gsea_running_score(), run_gsea(), run_ora(), run_pathway_analysis()

api_get_clinical *Get clinical data for API response*

Description

Reads clinical data from the targets store and formats it for API delivery. Optionally filters by patient IDs or variables.

Usage

```
api_get_clinical(patient_ids = NULL, variables = NULL, store = "_targets")
```

Arguments

patient_ids	Optional character vector of patient IDs to filter
variables	Optional character vector of column names to select
store	Path to targets store (default: "_targets")

Value

A data frame of clinical data, or NULL if unavailable

See Also

Other api: api_get_de_results(), api_get_pathways(), api_get_survival(), api_list_datasets(), api_serve(), generate_api_endpoint(), generate_api_index()

`api_get_de_results` *Get DE results for API response*

Description

Reads differential expression results from the targets store. Returns the DESeq2 results table with gene symbols.

Usage

```
api_get_de_results(padj_threshold = 1, lfc_threshold = 0, store = "_targets")
```

Arguments

`padj_threshold`	Filter to genes with padj below this threshold (default: 1, i.e., no filter)
`lfc_threshold`	Filter to genes with absolute log2FC above this threshold (default: 0, i.e., no filter)
`store`	Path to targets store

Value

A data frame of DE results, or NULL if unavailable

See Also

Other api: api_get_clinical(), api_get_pathways(), api_get_survival(), api_list_datasets(), api_serve(), generate_api_endpoint(), generate_api_index()

api_get_pathways *Get pathway analysis results for API response*

Description

Reads GSEA results from the targets store.

Usage

```
api_get_pathways(significant_only = FALSE, store = "_targets")
```

Arguments

significant_only	If TRUE, return only pathways with padj < 0.05
store	Path to targets store

Value

A data frame of pathway results, or NULL if unavailable

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_survival(), api_list_datasets(), api_serve(), generate_api_endpoint(), generate_api_index()

api_get_survival *Get survival data for API response*

Description

Reads prepared survival data from the targets store.

Usage

```
api_get_survival(store = "_targets")
```

Arguments

store	Path to targets store

Value

A data frame of survival data, or NULL if unavailable

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_pathways(), api_list_datasets(), api_serve(), generate_api_endpoint(), generate_api_index()

`api_list_datasets` *List available API datasets*

Description

Returns metadata about all datasets available through the API, including data type, row/column counts, and download formats.

Usage

```
api_list_datasets()
```

Value

A data frame with columns: dataset, description, format, n_rows, n_cols, last_updated

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_pathways(), api_get_survival(), api_serve(), generate_api_endpoint(), generate_api_index()

Examples

```
## Not run:
api_list_datasets()

## End(Not run)
```

`api_serve` *Launch the plumber API*

Description

Starts the plumber API server for programmatic data access. The API reads pre-computed results from the targets store.

Usage

```
api_serve(port = 8080L, host = "127.0.0.1", store = "_targets")
```

Arguments

port	Port number (default: 8080)
host	Host address (default: "127.0.0.1")
store	Path to targets store (default: "_targets")

Value

Invisible NULL (starts server)

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_pathways(), api_get_survival(), api_list_datasets(), generate_api_endpoint(), generate_api_index()

Examples

```
## Not run:
# Start the API server
api_serve(port = 8080)

# Then query from R or curl:
# curl http://localhost:8080/datasets
# curl http://localhost:8080/data/clinical?variables=submitter_id,gender

## End(Not run)
```

---

calculate_cooccurrence

Calculate pairwise co-occurrence of cytogenetic alterations

Description

Computes Fisher's exact test for each pair of cytogenetic markers to identify significant co-occurrence or mutual exclusivity patterns.

Usage

```
calculate_cooccurrence(cyto_data, markers = NULL)
```

Arguments

cyto_data	Data frame from extract_cytogenetic_data()
markers	Character vector of marker columns. Default auto-detects.

Value

Data frame with columns: marker1, marker2, odds_ratio, pvalue, padj (BH-corrected), n_both, n_either, tendency (co-occurrence/exclusive/none)

See Also

Other cytogenetics: compute_riss(), extract_cytogenetic_data(), plot_cooccurrence_heatmap(), plot_cytogenetic_oncoprint(), plot_expression_by_subtype(), summarize_cytogenetics()

Examples

```
## Not run:
cyto <- arrow::read_parquet("data/raw/clinical/cytogenetic_data.parquet")
cooc <- calculate_cooccurrence(cyto)

## End(Not run)
```

---

calculate_qc_metrics *Calculate QC metrics for RNA-seq data*

Description

Calculate QC metrics for RNA-seq data

Usage

```
calculate_qc_metrics(se_data)
```

Arguments

se_data	A SummarizedExperiment object with a "counts" assay

Value

A data frame with QC metrics per sample

---

check_adjustment *Compare model covariates against DAG-implied adjustments*

Description

Checks whether the covariates used in a model match any of the DAG-implied minimal sufficient adjustment sets.

Usage

```
check_adjustment(
  model_covariates,
  exposure = "cytogenetic_risk",
  outcome = "overall_survival",
  dag = NULL
)
```

Arguments

model_covariates	Character vector of covariates used in the model.
exposure	Character. Exposure variable.
outcome	Character. Outcome variable.
dag	A ‘dagitty‘ DAG object.

Value

A list with ‘$sufficient‘ (logical), ‘$model_covariates‘, ‘$adjustment_sets‘, and ‘$recommendation‘.

---

check_dependencies *Check package dependencies*

Description

Check package dependencies

Usage

```
check_dependencies()
```

---

clean_clinical_data *Clean Clinical Data*

Description

Standardizes clinical data column names and formats

Usage

```
clean_clinical_data(clinical_raw)
```

Arguments

clinical_raw	Raw clinical data frame

Value

Cleaned clinical data frame

See Also

Other data-cleaning: clean_expression_data(), clean_treatment_data(), integrate_clinical_expression(), summarize_treatment()

clean_expression_data *Clean Expression Data*

Description

Standardizes expression data format and adds metadata

Usage

```
clean_expression_data(expr_raw)
```

Arguments

expr_raw	Raw expression data (matrix or data frame)

Value

Cleaned expression matrix with gene names as rownames

See Also

Other data-cleaning: clean_clinical_data(), clean_treatment_data(), integrate_clinical_expression(), summarize_treatment()

clean_treatment_data *Clean and standardize treatment data*

Description

Standardizes regimen names, derives regimen class, and creates an ordered response factor from raw CoMMpass treatment response data.

Usage

```
clean_treatment_data(trtresp_raw)
```

Arguments

trtresp_raw	Data frame of raw treatment response data with columns including patient/submitter ID, treatment line, regimen description, best response, and transplant status

Value

Cleaned data frame with standardized columns: ‘patient_id‘, ‘treatment_line‘, ‘regimen_name‘, ‘regimen_class‘, ‘best_response‘ (ordered factor), ‘stem_cell_transplant‘ (logical), ‘treatment_start_days‘ (integer)

See Also

Other data-cleaning: clean_clinical_data(), clean_expression_data(), integrate_clinical_expression(), summarize_treatment()

---

commpass_dag *Define the CoMMpass causal DAG*

Description

Encodes domain-knowledge causal assumptions for the CoMMpass multiple myeloma analysis. Nodes represent measured or latent variables; edges represent assumed causal effects.

Usage

```
commpass_dag()
```

Value

A 'dagitty::dagitty' DAG object

---

compute_pca *Compute PCA from transformed expression data*

Description

Compute PCA from transformed expression data

Usage

```
compute_pca(expr_matrix, n_top = 500L, metadata = NULL)
```

Arguments

expr_matrix	Numeric matrix (genes x samples) of transformed expression values (e.g., VST or logCPM)
n_top	Number of most variable genes to use (default 500)
metadata	Optional data frame of sample annotations to merge with PCA coordinates

Value

List with components: coords (data frame with PC1, PC2, ...), var_explained (numeric vector of variance proportions), n_genes_used

See Also

Other differential-expression: plot_heatmap_de(), plot_ma(), plot_pca(), plot_volcano(), run_deseq2(), run_vst(), summarize_de_methods()

`compute_riss` *Compute Revised International Staging System (R-ISS)*

Description

Classifies patients into R-ISS stages based on ISS stage, cytogenetic risk group, and LDH level (Palumbo et al., JCO 2015).

Usage

```
compute_riss(iss_stage, risk_group, ldh = NA_real_, ldh_uln = 250)
```

Arguments

`iss_stage`	Character vector: "Stage I", "Stage II", "Stage III"
`risk_group`	Character vector: "High", "Standard", "high", "standard"
`ldh`	Numeric vector: LDH value (U/L), or NA
`ldh_uln`	Numeric scalar: upper limit of normal for LDH (default 250)

Value

Character vector: "R-ISS I", "R-ISS II", "R-ISS III", or NA

See Also

Other cytogenetics: `calculate_cooccurrence()`, `extract_cytogenetic_data()`, `plot_cooccurrence_heatmap()`, `plot_cytogenetic_oncoprint()`, `plot_expression_by_subtype()`, `summarize_cytogenetics()`

Examples

```
compute_riss("Stage I", "Standard", ldh = 200)
compute_riss("Stage III", "High", ldh = 300)
compute_riss(
  c("Stage I", "Stage III", "Stage II"),
  c("Standard", "High", "Standard"),
  ldh = c(200, 300, NA)
)
```

correlate_genes *Correlate two genes across samples*

Description

Computes Pearson or Spearman correlation between two genes in an expression matrix.

Usage

```
correlate_genes(expr_matrix, gene_x, gene_y, method = c("pearson", "spearman"))
```

Arguments

expr_matrix	Numeric matrix (genes x samples) of transformed expression values (e.g., VST)
gene_x	Gene name for x-axis
gene_y	Gene name for y-axis
method	Correlation method: "pearson" (default) or "spearman"

Value

List with components: gene_x, gene_y, method, estimate, p_value, n_samples, expr_x (numeric vector), expr_y (numeric vector)

See Also

Other gene-correlation: correlate_genes_batch(), plot_gene_correlation()

correlate_genes_batch *Batch correlation: one gene vs many*

Description

Correlates a target gene against a vector of candidate genes and returns a summary data frame sorted by absolute correlation.

Usage

```
correlate_genes_batch(
  expr_matrix,
  target_gene,
  candidate_genes,
  method = c("pearson", "spearman")
)
```

Arguments

expr_matrix	Numeric matrix (genes x samples)
target_gene	Gene name to correlate against
candidate_genes	Character vector of gene names to test
method	Correlation method: "pearson" or "spearman"

Value

Data frame with columns: gene, estimate, p_value, padj, n_samples, method

See Also

Other gene-correlation: correlate_genes(), plot_gene_correlation()

---

create_project_dirs *Create project directories*

Description

Create project directories

Usage

```
create_project_dirs(base_dir = ".")
```

Arguments

base_dir	Base directory path (default: ".")

---

create_summary_table *Create Summary Statistics Table*

Description

Generate a nicely formatted summary statistics table for numeric variables

Usage

```
create_summary_table(data, vars = NULL)
```

Arguments

data	Data frame
vars	Character vector of variable names (NULL for all numeric)

Value

Data frame with summary statistics

See Also

Other utilities: `example_data()`, `export_h5ad()`, `format_file_size()`, `format_with_commas()`, `gene_report()`, `strip_plotly()`

`download_aws_data` *Download data from AWS S3 open access bucket*

Description

Download data from AWS S3 open access bucket

Usage

```
download_aws_data(
  bucket_name = "gdc-mmrf-commpass-phs000748-2-open",
  prefix = NULL,
  data_dir = "data/raw/aws",
  region = "us-east-1"
)
```

Arguments

`bucket_name`	S3 bucket name
`prefix`	File prefix to filter
`data_dir`	Local directory for downloads
`region`	AWS region

Value

Directory path as character string

`download_clinical_data` *Download clinical data from GDC*

Description

Downloads clinical and biospecimen data from the Genomic Data Commons (GDC) for the specified project. Data is saved in parquet and RDS formats.

Usage

```
download_clinical_data(
  project_id = "MMRF-COMMPASS",
  data_dir = "data/raw/clinical",
  use_parquet = TRUE
)
```

Arguments

`project_id`	Project identifier (default: "MMRF-COMMPASS")
`data_dir`	Directory to save data
`use_parquet`	If TRUE, also save parquet files (default TRUE)

Value

Path to the directory containing the saved data files

See Also

Other data-acquisition: `acquire_commpass_data()`, `download_gdc_rnaseq()`, `download_s3_subset()`, `get_commpass_clinical()`, `list_s3_commpass()`, `query_commpass_rna()`

Examples

```
## Not run:
# Download clinical data
clinical_dir <- download_clinical_data()

# Load from parquet
clinical <- arrow::read_parquet(file.path(clinical_dir, "clinical_data.parquet"))

## End(Not run)
```

download_gdc_rnaseq *Download RNA-seq data from GDC*

Description

Downloads RNA-seq gene expression data from the Genomic Data Commons (GDC) for the specified project. Data is saved as a SummarizedExperiment RDS and as parquet files (counts, sample metadata, gene metadata).

Usage

```
download_gdc_rnaseq(
  project_id = "MMRF-COMMPASS",
  data_dir = "data/raw/gdc",
  sample_limit = 200,
  random_sample = TRUE,
  seed = 42,
  use_parquet = TRUE
)
```

Arguments

project_id	Project identifier (default: "MMRF-COMMPASS")
data_dir	Directory to save data
sample_limit	Maximum number of samples (NULL for all, default 200)
random_sample	If TRUE and sample_limit is set, randomly sample patients using seed for reproducibility (default TRUE)
seed	Random seed for reproducible sampling (default 42)
use_parquet	If TRUE, also save parquet files alongside RDS (default TRUE)

Value

Path to the saved RDS file containing the SummarizedExperiment

See Also

Other data-acquisition: acquire_commpass_data(), download_clinical_data(), download_s3_subset(), get_commpass_clinical(), list_s3_commpass(), query_commpass_rna()

Examples

```
## Not run:
# Download 200 random samples with parquet output
rnaseq_file <- download_gdc_rnaseq(sample_limit = 200)

# Load the SummarizedExperiment
```

```
se_data <- readRDS(rnaseq_file)

# Or load parquet directly
counts <- arrow::read_parquet("data/raw/gdc/rnaseq_counts.parquet")

## End(Not run)
```

`download_s3_subset` *Download a Sample of RNA-seq Files from S3*

Description

Downloads a subset of RNA-seq files from the public MMRF CoMMpass S3 bucket. This function is useful for testing and development with a small sample of data before downloading the full dataset. Files are downloaded using anonymous access to the public bucket.

Usage

```
download_s3_subset(s3_paths, dest_dir = "data/raw/rna_seq", n = 3)
```

Arguments

`s3_paths`	Character vector of S3 object keys (file paths) to download from
`dest_dir`	Destination directory for downloaded files (default: "data/raw/rna_seq")
`n`	Number of files to download (default: 3)

Value

Character vector of successfully downloaded file paths

See Also

Other data-acquisition: `acquire_commpass_data()`, `download_clinical_data()`, `download_gdc_rnaseq()`, `get_commpass_clinical()`, `list_s3_commpass()`, `query_commpass_rna()`

Examples

```
## Not run:
# List available files
s3_files <- list_s3_commpass()

# Download first 3 RNA-seq files
downloaded <- download_s3_subset(s3_files, n = 3)

# Check what was downloaded
basename(downloaded)

## End(Not run)
```

`example_data` *Load example coMMpass datasets*

Description

Returns small **synthetic** datasets (no real patient data) for interactive exploration and testing. The data exercises the full pipeline: QC, cleaning, survival analysis, and cytogenetic classification.

Usage

```
example_data()
```

Details

The SummarizedExperiment is reconstructed at load time from stored plain-R components (matrix + data.frames), so the RDS files have no S4 class dependency.

Value

A named list with three elements:

rnaseq_se A [SummarizedExperiment::SummarizedExperiment] with 50 genes x 20 samples. Assay named '"unstranded"' (matches GDC format). rowData contains 'gene_id' (Ensembl-format with version suffix) and 'gene_type'. colData contains 'submitter_id' and 'sample_type'.

clinical A data.frame (20 patients) with columns 'submitter_id', 'vital_status', 'days_to_death', 'days_to_last_follow_up', 'age_at_diagnosis' (in days), 'gender', 'iss_stage', 'heavy_chain' (myeloma isotype), 'light_chain' (Kappa/Lambda), 'ecog_status' (0-4), 'ldh' (U/L), 'b2m' (beta-2 microglobulin, mg/L), 'albumin' (g/dL), 'flc_kappa' (free kappa light chain, mg/L), 'flc_lambda' (free lambda light chain, mg/L), 'hemoglobin' (g/dL), 'creatinine' (mg/dL), 'calcium' (corrected, mg/dL), 'platelets' (10^9/L).

cytogenetic A data.frame (20 patients) with columns 'patient_id', 't_4_14', 't_11_14', 't_14_16', 'del_17p', 'gain_1q', 'risk_group'.

treatment A data.frame of treatment lines (1-3 per patient) with columns 'patient_id', 'treatment_line', 'regimen_name', 'regimen_class', 'best_response' (ordered factor), 'stem_cell_transplant' (logical), 'treatment_start_days' (integer).

See Also

Other utilities: `create_summary_table()`, `export_h5ad()`, `format_file_size()`, `format_with_commas()`, `gene_report()`, `strip_plotly()`

Examples

```
d <- example_data()
# QC metrics
qc <- calculate_qc_metrics(d$rnaseq_se)
head(qc)
```

```
# Clinical cleaning
clin <- clean_clinical_data(d$clinical)

# Survival data with cytogenetic markers
surv <- prepare_survival_data(d$clinical, cyto_file = d$cytogenetic)
```

`export_h5ad` *Export SummarizedExperiment to H5AD (AnnData) format*

Description

Converts a [SummarizedExperiment::SummarizedExperiment] to AnnData format and writes an H5AD file for Python/scanpy interoperability.

Usage

```
export_h5ad(se, output_path, assay_name = NULL)
```

Arguments

`se`	A [SummarizedExperiment::SummarizedExperiment] object
`output_path`	Path for the output .h5ad file
`assay_name`	Which assay to export (default "unstranded", falls back to "counts" then first available)

Value

The output path (invisibly)

See Also

Other utilities: `create_summary_table()`, `example_data()`, `format_file_size()`, `format_with_commas()`, `gene_report()`, `strip_plotly()`

Examples

```
## Not run:
d <- example_data()
export_h5ad(d$rnaseq_se, "commpass.h5ad")

## End(Not run)
```

extract_cytogenetic_data *Extract cytogenetic markers from clinical data*

Description

Parses clinical data from GDC to extract translocation and copy number alteration status. GDC clinical data includes FISH results for standard myeloma cytogenetic markers.

Usage

```
extract_cytogenetic_data(clinical_dir)
```

Arguments

clinical_dir	Path to clinical data directory (from download_clinical_data)

Value

Path to saved cytogenetic parquet file

See Also

Other cytogenetics: calculate_cooccurrence(), compute_riss(), plot_cooccurrence_heatmap(), plot_cytogenetic_oncoprint(), plot_expression_by_subtype(), summarize_cytogenetics()

Examples

```
## Not run:
cyto_file <- extract_cytogenetic_data("data/raw/clinical")
cyto <- arrow::read_parquet(cyto_file)

## End(Not run)
```

extract_risk_table *Extract risk table from Kaplan-Meier results*

Description

Generates a number-at-risk table at fixed time points from a [run_kaplan_meier()] result. Useful for pre-computing Shiny display data and static API endpoints.

Usage

```
extract_risk_table(km_result, times = c(0, 365, 730, 1095, 1460))
```

Arguments

km_result	List returned by [run_kaplan_meier()]
times	Numeric vector of time points (in days) at which to evaluate the risk table. Defaults to ‘c(0, 365, 730, 1095, 1460)‘ (0-4 years).

Value

Data frame with columns: time, n_risk, n_event, n_censor, survival, and optionally strata (if stratified KM)

See Also

Other survival: plot_forest(), plot_km(), prepare_survival_data(), run_cox_regression(), run_kaplan_meier(), run_km_by_expression(), run_km_by_markers()

---

filter_low_quality *Filter low-quality samples and genes*

Description

Filter low-quality samples and genes

Usage

```
filter_low_quality(se_data, min_counts = 10, min_samples = 3)
```

Arguments

se_data	A SummarizedExperiment object
min_counts	Minimum count threshold per gene
min_samples	Minimum samples with counts above threshold

Value

Filtered SummarizedExperiment

find_consensus_genes *Find consensus DE genes across methods*

Description

Find consensus DE genes across methods

Usage

```
find_consensus_genes(de_results_list, padj_threshold = 0.05, lfc_threshold = 1)
```

Arguments

de_results_list	List of DE result objects from different methods
padj_threshold	Adjusted p-value threshold (default: 0.05)
lfc_threshold	Log2 fold change threshold (default: 1)

Value

List with consensus gene information

format_file_size *Format File Size in Human-Readable Format*

Description

Converts file sizes from bytes to human-readable format with appropriate units

Usage

```
format_file_size(size_bytes, digits = 1)
```

Arguments

size_bytes	Numeric vector of file sizes in bytes
digits	Number of decimal places to show (default: 1)

Value

Character vector with formatted file sizes

See Also

Other utilities: create_summary_table(), example_data(), export_h5ad(), format_with_commas(), gene_report(), strip_plotly()

Examples

```
format_file_size(c(1024, 1048576, 5000877192))
# Returns: "1.0 KB", "1.0 MB", "4.7 GB"
```

---

format_with_commas *Format Number with Thousands Separator*

Description

Adds commas as thousands separators to large numbers

Usage

```
format_with_commas(x)
```

Arguments

x	Numeric value or vector

Value

Character vector with formatted numbers

See Also

Other utilities: create_summary_table(), example_data(), export_h5ad(), format_file_size(), gene_report(), strip_plotly()

Examples

```
format_with_commas(1234567)
# Returns: "1,234,567"
```

---

gene_report *Render a single-gene characterization report*

Description

Renders the parameterized ‘gene-report.qmd‘ vignette for a specific gene, producing a self-contained HTML report with expression distribution, survival stratification, cytogenetic context, and gene-gene correlations.

Usage

```
gene_report(
  gene,
  output_dir = "results/gene_reports",
  vst_matrix = NULL,
  surv_data = NULL,
  cyto_data = NULL
)
```

Arguments

gene	Gene name (e.g., "CD70", "TP53")
output_dir	Directory for the rendered report (default "results/gene_reports/")
vst_matrix	Optional pre-loaded VST matrix (genes x samples). If NULL, the function attempts to load from the targets store.
surv_data	Optional pre-loaded survival data frame. If NULL, attempts to load from the targets store.
cyto_data	Optional pre-loaded cytogenetic data frame

Value

Path to the rendered HTML file (invisibly)

See Also

Other utilities: create_summary_table(), example_data(), export_h5ad(), format_file_size(), format_with_commas(), strip_plotly()

Examples

```
## Not run:
gene_report("CD70")
gene_report("TP53", output_dir = "results/")

## End(Not run)
```

---

generate_api_endpoint *Generate a single API endpoint JSON string*

Description

Wraps a data frame in standard API envelope with metadata and serializes to JSON. Used by [plan_api] targets to produce static JSON files.

Usage

```
generate_api_endpoint(data, dataset_name, description)
```

Arguments

data	A data frame to serialize, or NULL if data is unavailable
dataset_name	Short name for this dataset (e.g. "clinical")
description	Human-readable description

Value

A JSON string (character scalar) with ‘metadata‘ and ‘data‘ fields

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_pathways(), api_get_survival(), api_list_datasets(), api_serve(), generate_api_index()

Examples

```
## Not run:
df <- data.frame(a = 1:3, b = letters[1:3])
json <- generate_api_endpoint(df, "example", "Example data")
cat(json)

## End(Not run)
```

---

generate_api_index *Generate API index metadata*

Description

Creates the index.json content listing all available endpoints, their descriptions, and URLs for the static JSON API.

Usage

```
generate_api_index(
  base_url = "https://JohnGavin.github.io/coMMpass-analysis/api/v1"
)
```

Arguments

base_url	Base URL for the static API. Defaults to the GitHub Pages URL.

Value

A list suitable for JSON serialization containing endpoint catalogue, version, and generated timestamp.

See Also

Other api: api_get_clinical(), api_get_de_results(), api_get_pathways(), api_get_survival(), api_list_datasets(), api_serve(), generate_api_endpoint()

Examples

```
## Not run:
idx <- generate_api_index()
cat(jsonlite::toJSON(idx, pretty = TRUE, auto_unbox = TRUE))

## End(Not run)
```

---

generate_summary_report

Generate summary report

Description

Generate summary report

Usage

```
generate_summary_report(
  qc_metrics,
  de_genes,
  survival,
  pathways,
  output_dir = "results/reports"
)
```

Arguments

qc_metrics	QC metrics data frame
de_genes	DE results with consensus gene information
survival	Survival analysis results
pathways	Pathway analysis results
output_dir	Directory for output report

Value

Path to generated report

get_adjustment_sets *Get adjustment sets for a given analysis*

Description

Uses the DAG to identify the minimal sufficient adjustment set for estimating the causal effect of ‘exposure‘ on ‘outcome‘.

Usage

```
get_adjustment_sets(
  exposure = "cytogenetic_risk",
  outcome = "overall_survival",
  dag = NULL
)
```

Arguments

exposure	Character. The exposure variable name in the DAG.
outcome	Character. The outcome variable name in the DAG.
dag	A ‘dagitty‘ DAG object. Defaults to [commpass_dag()].

Value

A list of character vectors, each a minimal sufficient adjustment set.

get_commpass_clinical *Query GDC for CoMMpass Clinical Data*

Description

Retrieves clinical data for the MMRF CoMMpass study from the Genomic Data Commons (GDC). This includes patient demographics, disease characteristics, treatment information, and outcomes data.

Usage

```
get_commpass_clinical()
```

Value

A data frame containing clinical data for CoMMpass patients

See Also

Other data-acquisition: acquire_commpass_data(), download_clinical_data(), download_gdc_rnaseq(), download_s3_subset(), list_s3_commpass(), query_commpass_rna()

Examples

```
## Not run:
# Get clinical data
clinical <- get_commpass_clinical()

# View first few rows
head(clinical)

# Check available columns
names(clinical)

## End(Not run)
```

---

get_commpass_data_dictionary

Get CoMMpass Data Dictionary

Description

Returns a tibble documenting all known variables in the CoMMpass dataset, including clinical, biospecimen, and RNA-seq data. Each variable includes its category, data type, units, description, typical range, and a link to the GDC data dictionary.

Usage

```
get_commpass_data_dictionary()
```

Value

A tibble with columns: variable, category, data_type, units, description, typical_range, gdc_link

See Also

Other data-dictionary: get_variable_docs()

Examples

```
dd <- get_commpass_data_dictionary()
# Filter to clinical variables
dplyr::filter(dd, category == "clinical")
```

`get_commpass_tbl` *Get Lazy DuckDB Table for CoMMpass Data*

Description

Returns a lazy ‘dplyr::tbl()‘ backed by DuckDB, reading from parquet files. The connection is managed by the caller and must be disconnected when done.

Usage

```
get_commpass_tbl(
 data_type = c("clinical", "biospecimen", "rnaseq_counts", "rnaseq_sample_metadata",
    "rnaseq_gene_metadata"),
  data_dir = "data/raw",
  con = NULL
)
```

Arguments

`data_type`	One of "clinical", "biospecimen", "rnaseq_counts", "rnaseq_sample_metadata", "rnaseq_gene_metadata"
`data_dir`	Base directory containing parquet files
`con`	An existing DuckDB connection. If NULL, a new in-memory connection is created and returned as an attribute of the result.

Value

A lazy dbplyr tbl. If con was NULL, the DuckDB connection is stored as attr(result, "connection") - caller must disconnect it.

See Also

Other storage: `query_commpass_parquet()`

Examples

```
## Not run:
# Create connection and query
con <- DBI::dbConnect(duckdb::duckdb())
clinical_tbl <- get_commpass_tbl("clinical", con = con)

# Chain dplyr operations (lazy - not executed until collect)
result <- clinical_tbl |>
  dplyr::filter(gender == "female") |>
  dplyr::select(submitter_id, age_at_diagnosis, vital_status) |>
  dplyr::collect()

# Clean up
```

```
DBI::dbDisconnect(con, shutdown = TRUE)

## End(Not run)
```

`get_counts_assay` *Get counts assay from SummarizedExperiment*

Description

GDC STAR-Counts uses 'unstranded', but fallback to 'counts' if present

Usage

```
get_counts_assay(se)
```

Arguments

`se`	SummarizedExperiment object

Value

Counts matrix

`get_variable_docs` *Get Extended Documentation for a Variable*

Description

Returns detailed documentation for a specific variable, including scientific context, calculation methods, and usage notes.

Usage

```
get_variable_docs(variable)
```

Arguments

`variable`	Character string naming the variable to document

Value

A list with elements: variable, description, scientific_context, calculation, usage_notes, references

See Also

Other data-dictionary: `get_commpass_data_dictionary()`

Examples

```
docs <- get_variable_docs("age_at_diagnosis")
cat(docs$usage_notes)
```

`integrate_clinical_expression` *Create Integrated Dataset*

Description

Combines clinical and expression data with consistent sample IDs

Usage

```
integrate_clinical_expression(clinical_clean, expr_clean)
```

Arguments

`clinical_clean`	Cleaned clinical data
`expr_clean`	Cleaned expression data

Value

List with matched clinical and expression data

See Also

Other data-cleaning: `clean_clinical_data()`, `clean_expression_data()`, `clean_treatment_data()`, `summarize_treatment()`

`list_s3_commpass` *List AWS S3 CoMMpass Bucket Contents*

Description

Lists files available in the public AWS S3 bucket containing MMRF CoMMpass data. The bucket contains RNA-seq, genomic, and clinical data files. This function uses anonymous access to the public bucket.

Usage

```
list_s3_commpass(prefix = "")
```

Arguments

`prefix`	Optional prefix to filter files (e.g., "RNA-seq/", "clinical/")

Value

Character vector of S3 object keys (file paths)

See Also

Other data-acquisition: acquire_commpass_data(), download_clinical_data(), download_gdc_rnaseq(), download_s3_subset(), get_commpass_clinical(), query_commpass_rna()

Examples

```
## Not run:
# List all available files (limited to first 100)
files <- list_s3_commpass()

# List only RNA-seq files
rna_files <- list_s3_commpass(prefix = "RNA-seq/")

# Count available files
length(files)

## End(Not run)
```

---

normalize_rnaseq *Normalize RNA-seq data*

Description

Normalize RNA-seq data

Usage

```
normalize_rnaseq(se_data, method = "TMM")
```

Arguments

se_data	A SummarizedExperiment object with a "counts" assay
method	Normalization method (default: "TMM")

Value

SummarizedExperiment with added "logCPM" assay

plot_cooccurrence_heatmap

Plot co-occurrence heatmap of cytogenetic alterations

Description

Displays pairwise co-occurrence patterns as a heatmap. Color indicates -log10(p-value) direction: red for co-occurrence, blue for mutual exclusivity.

Usage

```
plot_cooccurrence_heatmap(cooccurrence, title = "Cytogenetic Co-occurrence")
```

Arguments

cooccurrence	Data frame from calculate_cooccurrence()
title	Plot title

Value

A ggplot object

See Also

Other cytogenetics: calculate_cooccurrence(), compute_riss(), extract_cytogenetic_data(), plot_cytogenetic_oncoprint(), plot_expression_by_subtype(), summarize_cytogenetics()

Examples

```
## Not run:
cooc <- calculate_cooccurrence(cyto)
plot_cooccurrence_heatmap(cooc)

## End(Not run)
```

plot_cytogenetic_oncoprint

Plot cytogenetic oncoprint

Description

Creates an oncoprint-style heatmap showing cytogenetic alterations across patients. Rows are alterations, columns are patients. Uses ggplot2 for portability (no ComplexHeatmap dependency).

Usage

```
plot_cytogenetic_oncoprint(
  cyto_data,
  markers = NULL,
  sort_by = c("frequency", "risk"),
  title = "Cytogenetic Landscape"
)
```

Arguments

cyto_data	Data frame from extract_cytogenetic_data() with columns: patient_id, iss_stage, t_4_14, t_11_14, t_14_16, del_17p, gain_1q, etc.
markers	Character vector of marker columns to include. Default uses all available markers.
sort_by	How to sort patients. One of "frequency" (default) or "risk".
title	Plot title.

Value

A ggplot object

See Also

Other cytogenetics: calculate_cooccurrence(), compute_riss(), extract_cytogenetic_data(), plot_cooccurrence_heatmap(), plot_expression_by_subtype(), summarize_cytogenetics()

Examples

```
## Not run:
cyto <- arrow::read_parquet("data/raw/clinical/cytogenetic_data.parquet")
plot_cytogenetic_oncoprint(cyto)

## End(Not run)
```

---

plot_dag *Plot the CoMMpass causal DAG*

Description

Renders the DAG using ggdag with sensible defaults.

Usage

```
plot_dag(dag = NULL, title = "CoMMpass Causal DAG")
```

Arguments

dag	A ‘dagitty‘ DAG object. Defaults to [commpass_dag()].
title	Plot title.

Value

A ggplot object.

`plot_enrichment_barplot` *Bar plot of enrichment results*

Description

Creates a horizontal bar plot of the top N enriched pathways, colored by direction (GSEA) or significance (ORA).

Usage

```
plot_enrichment_barplot(enrich_result, n = 15L, title = "Enrichment Bar Plot")
```

Arguments

enrich_result	Data frame with pathway enrichment results
n	Number of top pathways to show (default 15)
title	Plot title

Value

A ggplot2 object

See Also

Other pathway: `annotate_de_results()`, `annotate_genes()`, `plot_enrichment_dotplot()`, `plot_gsea_running_score()`, `run_gsea()`, `run_ora()`, `run_pathway_analysis()`

plot_enrichment_dotplot

Dot plot of enrichment results

Description

Creates a dot plot of the top N enriched pathways. Dot size represents gene set size, color represents significance (adjusted p-value or NES).

Usage

```
plot_enrichment_dotplot(
  enrich_result,
  n = 15L,
  color_by = "padj",
  title = "Enrichment Dot Plot"
)
```

Arguments

enrich_result	Data frame with pathway enrichment results. Must contain columns: pathway, padj, and one of: NES (GSEA), overlap (ORA).
n	Number of top pathways to show (default 15)
color_by	Which metric to color by: "padj" (default) or "NES"
title	Plot title

Value

A ggplot2 object

See Also

Other pathway: annotate_de_results(), annotate_genes(), plot_enrichment_barplot(), plot_gsea_running_score(), run_gsea(), run_ora(), run_pathway_analysis()

plot_expression_by_subtype

Plot gene expression by cytogenetic subtype

Description

Creates violin + box plots showing gene expression stratified by cytogenetic marker status (positive vs negative). Adds Wilcoxon or t-test p-value annotation.

Usage

```
plot_expression_by_subtype(
  expr_matrix,
  cyto_data,
  gene,
  markers = NULL,
  test = c("wilcox", "t.test")
)
```

Arguments

expr_matrix	Numeric matrix (genes x samples) of transformed expression values (e.g., VST). Column names should match 'cyto_data$patient_id'.
cyto_data	Data frame with 'patient_id' and marker columns (values: "positive"/"negative"/NA)
gene	Gene name (rowname of 'expr_matrix')
markers	Character vector of marker columns. Default auto-detects.
test	Statistical test: "wilcox" (default) or "t.test"

Value

A ggplot object with faceted violin/box plots, one panel per marker

See Also

Other cytogenetics: calculate_cooccurrence(), compute_riss(), extract_cytogenetic_data(), plot_cooccurrence_heatmap(), plot_cytogenetic_oncoprint(), summarize_cytogenetics()

---

plot_forest *Plot forest plot of Cox hazard ratios*

Description

Creates a forest plot from Cox regression results, showing hazard ratios with 95

Usage

```
plot_forest(cox_result, title = "Cox Regression Forest Plot")
```

Arguments

cox_result	List from [run_cox_regression()] with 'hazard_ratios', 'concordance', 'n', 'n_events'
title	Plot title

Value

A ggplot object

See Also

Other survival: extract_risk_table(), plot_km(), prepare_survival_data(), run_cox_regression(), run_kaplan_meier(), run_km_by_expression(), run_km_by_markers()

---

plot_gene_correlation *Scatter plot of gene-gene correlation*

Description

Creates a scatter plot with regression line, correlation coefficient, and p-value annotation from a [correlate_genes()] result.

Usage

```
plot_gene_correlation(cor_result, title = NULL)
```

Arguments

cor_result	List from [correlate_genes()]
title	Plot title (default auto-generated)

Value

A ggplot object

See Also

Other gene-correlation: correlate_genes(), correlate_genes_batch()

---

plot_gsea_running_score *GSEA running enrichment score plot*

Description

Shows the running enrichment score for a specific gene set from fgsea results, along with the ranked gene positions.

Usage

```
plot_gsea_running_score(
  gsea_result,
  gene_set_name,
  gene_sets = NULL,
  title = NULL
)
```

Arguments

gsea_result	List from run_gsea() containing results and ranked_genes
gene_set_name	Name of the gene set to plot
gene_sets	Named list of gene sets (required for tick marks). If NULL, attempts to retrieve from msigdbr using the collection.
title	Plot title (default: gene set name)

Value

A ggplot2 object

See Also

Other pathway: annotate_de_results(), annotate_genes(), plot_enrichment_barplot(), plot_enrichment_dotplot(), run_gsea(), run_ora(), run_pathway_analysis()

---

plot_heatmap_de *Heatmap of top DE genes*

Description

Creates a heatmap of the top differentially expressed genes using z-score scaled expression values.

Usage

```
plot_heatmap_de(
  expr_matrix,
  de_results,
  n_genes = 50L,
  annotation_df = NULL,
  title = "Top DE Genes"
)
```

Arguments

expr_matrix	Numeric matrix of transformed expression (genes x samples)
de_results	Data frame with DE results (rownames = gene IDs)
n_genes	Number of top DE genes to show (default 50)
annotation_df	Optional data frame of sample annotations for column annotation
title	Plot title

Value

A ggplot2 object (tile-based heatmap)

See Also

Other differential-expression: compute_pca(), plot_ma(), plot_pca(), plot_volcano(), run_deseq2(), run_vst(), summarize_de_methods()

---

plot_km *Plot Kaplan-Meier curve*

Description

Creates a ggplot2-based KM survival curve from a survfit object. Supports stratified and unstratified curves with optional log-rank p-value.

Usage

```
plot_km(
  km_result,
  title = "Kaplan-Meier Survival Curve",
  xlab = "Time (days)",
  time_breaks = NULL
)
```

Arguments

km_result	List from [run_kaplan_meier()] with 'fit', 'logrank_p', 'median_survival', 'n_per_group', 'strata'
title	Plot title
xlab	X-axis label
time_breaks	Sequence of time breaks for x-axis (in days). Default marks every 365 days.

Value

A ggplot object

See Also

Other survival: extract_risk_table(), plot_forest(), prepare_survival_data(), run_cox_regression(), run_kaplan_meier(), run_km_by_expression(), run_km_by_markers()

plot_ma *MA plot of DE results*

Description

Plots mean expression (x-axis) vs log2 fold change (y-axis), a standard diagnostic for RNA-seq DE analysis.

Usage

```
plot_ma(
  de_results,
  lfc_threshold = 1,
  padj_threshold = 0.05,
  title = "MA Plot"
)
```

Arguments

de_results	Data frame with DE results
lfc_threshold	Log2 fold-change threshold for coloring (default 1)
padj_threshold	Adjusted p-value threshold (default 0.05)
title	Plot title

Value

A ggplot2 object

See Also

Other differential-expression: compute_pca(), plot_heatmap_de(), plot_pca(), plot_volcano(), run_deseq2(), run_vst(), summarize_de_methods()

plot_pca *PCA plot*

Description

Creates a PCA scatter plot from pre-computed PCA data.

Usage

```
plot_pca(pca_data, color_by = NULL, shape_by = NULL, title = "PCA")
```

Arguments

pca_data	List from compute_pca() with coords and var_explained
color_by	Column name in coords to color by (default NULL)
shape_by	Column name in coords to use for shape (default NULL)
title	Plot title

Value

A ggplot2 object

See Also

Other differential-expression: compute_pca(), plot_heatmap_de(), plot_ma(), plot_volcano(), run_deseq2(), run_vst(), summarize_de_methods()

---

plot_volcano *Volcano plot of DE results*

Description

Creates a ggplot2 volcano plot with significance thresholds and optional gene labels.

Usage

```
plot_volcano(
  de_results,
  lfc_threshold = 1,
  padj_threshold = 0.05,
  n_label = 10L,
  title = "Volcano Plot"
)
```

Arguments

de_results	Data frame with DE results (must have padj and log2FC columns – auto-detected from DESeq2/edgeR/limma conventions)
lfc_threshold	Log2 fold-change threshold for significance (default 1)
padj_threshold	Adjusted p-value threshold (default 0.05)
n_label	Number of top genes to label (default 10)
title	Plot title

Value

A ggplot2 object

See Also

Other differential-expression: compute_pca(), plot_heatmap_de(), plot_ma(), plot_pca(), run_deseq2(), run_vst(), summarize_de_methods()

`prepare_survival_data` *Prepare survival data from clinical and cytogenetic data*

Description

Constructs a data frame suitable for survival analysis from GDC clinical data. Uses 'days_to_death' for deceased patients and 'days_to_last_follow_up' for censored (alive) patients. Merges with cytogenetic risk groups when available.

Usage

```
prepare_survival_data(clinical_data, cyto_file = NULL)
```

Arguments

`clinical_data`	Cleaned clinical data frame (from clean_clinical_data)
`cyto_file`	Path to cytogenetic parquet file (from extract_cytogenetic_data), or NULL to skip cytogenetic integration

Value

Data frame with columns: patient_id, time_days, status (0=censored, 1=dead), age_years, gender, iss_stage, risk_group, plus individual cytogenetic markers

See Also

Other survival: extract_risk_table(), plot_forest(), plot_km(), run_cox_regression(), run_kaplan_meier(), run_km_by_expression(), run_km_by_markers()

`query_commpass_parquet`

Query CoMMpass Parquet Files

Description

Opens an in-memory DuckDB connection, creates a VIEW on the specified parquet file, optionally applies filters, and returns a data frame.

Usage

```
query_commpass_parquet(
 data_type = c("clinical", "biospecimen", "rnaseq_counts", "rnaseq_sample_metadata",
    "rnaseq_gene_metadata"),
  data_dir = "data/raw",
  filters = NULL,
  collect = TRUE
)
```

Arguments

data_type	One of "clinical", "biospecimen", "rnaseq_counts", "rnaseq_sample_metadata", "rnaseq_gene_metadata"
data_dir	Base directory containing parquet files
filters	Optional named list of filters. Names are column names, values are vectors of allowed values (used in WHERE ... IN (...))
collect	If TRUE (default), collect results into a data frame. If FALSE, return a lazy tbl for further dplyr operations.

Value

A data frame (if collect=TRUE) or lazy dbplyr tbl

See Also

Other storage: get_commpass_tbl()

Examples

```
## Not run:
# Read all clinical data
clinical <- query_commpass_parquet("clinical")

# Filter to specific patients
subset <- query_commpass_parquet(
  "clinical",
  filters = list(gender = "female", vital_status = "Alive")
)

# Get lazy tbl for chaining
tbl <- query_commpass_parquet("clinical", collect = FALSE)
result <- tbl |> dplyr::filter(gender == "female") |> dplyr::collect()

## End(Not run)
```

query_commpass_rna *Query GDC for CoMMpass RNA-seq Metadata*

Description

Queries the Genomic Data Commons (GDC) API for Multiple Myeloma Research Foundation (MMRF) CoMMpass study RNA-seq data. This function returns a query object that can be used with other TCGAbiolinks functions to download and prepare the data.

Usage

```
query_commpass_rna()
```

Value

A GDCquery object containing metadata for RNA-seq samples

See Also

Other data-acquisition: `acquire_commpass_data()`, `download_clinical_data()`, `download_gdc_rnaseq()`, `download_s3_subset()`, `get_commpass_clinical()`, `list_s3_commpass()`

Examples

```
## Not run:
# Query for RNA-seq data
query <- query_commpass_rna()

# Use the query to download data
# GDCdownload(query)
# data <- GDCprepare(query)

## End(Not run)
```

render_de_report *Render DE analysis report*

Description

Render DE analysis report

Usage

```
render_de_report(de_results, output_dir = "results/reports")
```

Arguments

de_results	DE results object
output_dir	Directory for output report

Value

Path to generated report

---

run_cox_regression *Run Cox proportional hazards regression*

Description

Fits a Cox PH model with specified covariates. Returns the model object, hazard ratios with confidence intervals, and concordance index.

Usage

```
run_cox_regression(surv_data, covariates = c("age_years", "gender"))
```

Arguments

surv_data	Data frame from [prepare_survival_data()]
covariates	Character vector of covariate column names

Value

List with components: - ‘model‘: coxph object - ‘hazard_ratios‘: data frame with HR, CI, p-values - ‘concordance‘: C-index - ‘n‘: number of patients in model - ‘n_events‘: number of events - ‘ph_test‘: cox.zph result for PH assumption

See Also

Other survival: extract_risk_table(), plot_forest(), plot_km(), prepare_survival_data(), run_kaplan_meier(), run_km_by_expression(), run_km_by_markers()

`run_deseq2` *Run DESeq2 differential expression analysis*

Description

Runs DESeq2 with optional apeglm log-fold-change shrinkage and optional paired longitudinal design for within-patient comparisons.

Usage

```
run_deseq2(
  se_data,
  clinical_data,
  design_formula = ~condition,
  shrink_lfc = TRUE,
  paired = FALSE
)
```

Arguments

`se_data`	A SummarizedExperiment object
`clinical_data`	Clinical data frame
`design_formula`	Design formula for the model
`shrink_lfc`	If TRUE, apply apeglm LFC shrinkage (default TRUE)
`paired`	If TRUE, use paired longitudinal design with patient as blocking factor. Requires 'patient_id' and 'visit' columns in clinical_data. Only patients with >=2 timepoints are included.

Value

List with DE results including raw and shrunken (if requested)

See Also

Other differential-expression: `compute_pca()`, `plot_heatmap_de()`, `plot_ma()`, `plot_pca()`, `plot_volcano()`, `run_vst()`, `summarize_de_methods()`

run_edger *Run edgeR differential expression analysis*

Description

Run edgeR differential expression analysis

Usage

```
run_edger(se_data, clinical_data, design_formula = ~condition)
```

Arguments

se_data	A SummarizedExperiment object
clinical_data	Clinical data frame
design_formula	Design formula for the model

Value

List with DE results

run_gsea *Run Gene Set Enrichment Analysis (GSEA)*

Description

Performs pre-ranked GSEA using fgsea on differential expression results. Genes are ranked by their test statistic (DESeq2 Wald stat or `-log10(pvalue) * sign(log2FC)`). Gene sets come from MSigDB via msigdbr.

Usage

```
run_gsea(
  de_results,
  gene_sets = "hallmark",
  gene_id_type = "ensembl_gene",
  min_size = 15L,
  max_size = 500L
)
```

Arguments

de_results	Data frame of DE results. Must contain gene identifiers (as rownames or in a gene/gene_id column) and at least one of: stat (DESeq2 Wald statistic), or log2FoldChange/logFC + pvalue/PValue.
gene_sets	Character string specifying the MSigDB collection: "hallmark" (default), "kegg", "reactome", "go_bp", "go_mf", "go_cc", "c2", "c7". Alternatively, a named list of character vectors (custom gene sets).
gene_id_type	Type of gene identifiers: "ensembl_gene" (default), "gene_symbol", or "entrez_gene".
min_size	Minimum gene set size (default: 15).
max_size	Maximum gene set size (default: 500).

Value

List with components:

results Data frame with pathway, NES, pval, padj, size, leadingEdge columns.

n_gene_sets Number of gene sets tested.

n_significant Number significant at padj < 0.05.

top_gene_sets Top 20 enriched gene sets as a data frame.

ranked_genes Named numeric vector of gene ranks used.

collection Gene set collection used.

See Also

Other pathway: annotate_de_results(), annotate_genes(), plot_enrichment_barplot(), plot_enrichment_dotplot(), plot_gsea_running_score(), run_ora(), run_pathway_analysis()

---

run_kaplan_meier *Run Kaplan-Meier analysis*

Description

Fits a Kaplan-Meier survival curve, optionally stratified by a grouping variable. Returns the survfit object, log-rank test, and summary statistics.

Usage

```
run_kaplan_meier(surv_data, strata = NULL)
```

Arguments

surv_data	Data frame from [prepare_survival_data()] with columns ‘time_days‘ and ‘status‘
strata	Column name to stratify by (e.g. "risk_group", "iss_stage", "gender"), or NULL for overall curve

Value

List with components: - ‘fit‘: survfit object - ‘logrank‘: survdiff result (NULL if no strata) - ‘logrank_p‘: log-rank p-value (NA if no strata) - ‘median_survival‘: named vector of median survival times - ‘n_per_group‘: sample sizes per stratum - ‘strata‘: the stratification variable used

See Also

Other survival: `extract_risk_table()`, `plot_forest()`, `plot_km()`, `prepare_survival_data()`, `run_cox_regression()`, `run_km_by_expression()`, `run_km_by_markers()`

`run_km_by_expression` *Run KM analysis stratified by gene expression level*

Description

Splits patients into groups based on expression of a single gene (e.g., median split) and runs Kaplan-Meier analysis with log-rank test. Connects differential expression results to clinical outcomes.

Usage

```
run_km_by_expression(
  surv_data,
  expr_matrix,
  gene,
  split = c("median", "tertile", "quartile", "top_bottom_20"),
  min_per_group = 5L
)
```

Arguments

`surv_data`	Data frame from [prepare_survival_data()] with columns ‘patient_id‘, ‘time_days‘, ‘status‘
`expr_matrix`	Numeric matrix (genes x samples) of transformed expression values (e.g., VST). Column names must match ‘surv_data$patient_id‘.
`gene`	Gene name (must be a rowname of ‘expr_matrix‘)
`split`	Split method: "median" (default), "tertile", "quartile", or "top_bottom_20"
`min_per_group`	Minimum patients per group (default 5)

Value

List compatible with [plot_km()]: fit, logrank_p, median_survival, n_per_group, strata, gene, split_method. Returns a list with ‘fit = NULL‘ and a ‘note‘ if requirements not met.

See Also

Other survival: `extract_risk_table()`, `plot_forest()`, `plot_km()`, `prepare_survival_data()`, `run_cox_regression()`, `run_kaplan_meier()`, `run_km_by_markers()`

run_km_by_markers *Run KM analysis for each individual cytogenetic marker*

Description

Iterates over cytogenetic markers and runs stratified KM analysis for each. Markers with fewer than ‘min_positive‘ positive cases are skipped.

Usage

```
run_km_by_markers(surv_data, markers = NULL, min_positive = 3L)
```

Arguments

surv_data	Data frame from [prepare_survival_data()]
markers	Character vector of marker column names. Default auto-detects.
min_positive	Minimum number of positive cases to run analysis (default 3)

Value

Named list of KM results, one per marker

See Also

Other survival: extract_risk_table(), plot_forest(), plot_km(), prepare_survival_data(), run_cox_regression(), run_kaplan_meier(), run_km_by_expression()

run_limma *Run limma differential expression analysis*

Description

Run limma differential expression analysis

Usage

```
run_limma(se_data, clinical_data, design_formula = ~condition)
```

Arguments

se_data	A SummarizedExperiment object
clinical_data	Clinical data frame
design_formula	Design formula for the model

Value

List with DE results

run_ora *Run Over-Representation Analysis (ORA)*

Description

Tests whether a set of significant genes is over-represented in gene set collections using Fisher's exact test (hypergeometric distribution). Gene sets come from MSigDB via msigdbr.

Usage

```
run_ora(
  sig_genes,
  universe,
  gene_sets = "hallmark",
  gene_id_type = "ensembl_gene",
  min_size = 10L,
  max_size = 500L
)
```

Arguments

sig_genes	Character vector of significant gene identifiers.
universe	Character vector of all tested gene identifiers (background).
gene_sets	Character string specifying the MSigDB collection (same options as [run_gsea()]), or a named list of character vectors.
gene_id_type	Type of gene identifiers: "ensembl_gene" (default), "gene_symbol", or "entrez_gene".
min_size	Minimum gene set size (default: 10).
max_size	Maximum gene set size (default: 500).

Value

List with components:

results Data frame with pathway, overlap, gene_set_size, universe_size, p_value, padj, odds_ratio, overlapping_genes.

n_pathways_tested Number of pathways tested.

n_pathways_enriched Number significant at padj < 0.05.

n_sig_genes Number of input significant genes.

top_pathways Top 20 enriched pathways as a data frame.

See Also

Other pathway: annotate_de_results(), annotate_genes(), plot_enrichment_barplot(), plot_enrichment_dotplot(), plot_gsea_running_score(), run_gsea(), run_pathway_analysis()

run_pathway_analysis *Run pathway enrichment analysis*

Description

Wrapper that runs ORA on consensus DE genes. Uses MSigDB Hallmark gene sets by default.

Usage

```
run_pathway_analysis(de_genes, method = "hallmark")
```

Arguments

de_genes	Consensus DE result from [find_consensus_genes()] with consensus_genes character vector and by_method list.
method	Gene set collection: "hallmark" (default), "kegg", "reactome", "go_bp".

Value

List with ORA results (see [run_ora()]).

See Also

Other pathway: annotate_de_results(), annotate_genes(), plot_enrichment_barplot(), plot_enrichment_dotplot(), plot_gsea_running_score(), run_gsea(), run_ora()

run_vst *Run variance stabilizing transformation*

Description

Wrapper around DESeq2::vst() for use in visualization. VST is appropriate for PCA, heatmaps, and clustering – not for DE testing (which uses raw counts).

Usage

```
run_vst(se_data, blind = TRUE)
```

Arguments

se_data	SummarizedExperiment with "counts" assay
blind	Logical, whether VST should be blind to the design (default TRUE for exploratory analysis)

Value

SummarizedExperiment with "vst" assay added, or NULL if DESeq2 is not available

See Also

Other differential-expression: `compute_pca()`, `plot_heatmap_de()`, `plot_ma()`, `plot_pca()`, `plot_volcano()`, `run_deseq2()`, `summarize_de_methods()`

---

save_timestamped *Save results with timestamp*

Description

Save results with timestamp

Usage

```
save_timestamped(object, base_name, dir = "results")
```

Arguments

`object`	R object to save
`base_name`	Base name for the output file
`dir`	Directory to save to (default: "results")

---

setup_logging *Setup logging*

Description

Setup logging

Usage

```
setup_logging(log_file = NULL)
```

Arguments

`log_file`	Optional path to a log file. If NULL, logs to console.

strip_plotly *Strip plotly closure bloat for compact serialization*

Description

Plotly htmlwidgets capture parent environments in closures, causing 60-80x size inflation when serialized via saveRDS/targets. This function runs plotly_build() to resolve lazy data, then replaces closure environments with emptyenv() to eliminate the bloat.

Usage

```
strip_plotly(p)
```

Arguments

p	A plotly htmlwidget object

Value

The same plotly object with closures stripped (much smaller on disk)

See Also

Other utilities: `create_summary_table()`, `example_data()`, `export_h5ad()`, `format_file_size()`, `format_with_commas()`, `gene_report()`

Examples

```
## Not run:
p <- plotly::plot_ly(mtcars, x = ~mpg, type = "histogram")
object.size(p)       # large
p2 <- strip_plotly(p)
object.size(p2)      # small

## End(Not run)
```

summarize_cytogenetics *Summarize cytogenetic alteration frequencies*

Description

Computes frequency and percentage for each marker and risk group.

Usage

```
summarize_cytogenetics(cyto_data)
```

Arguments

cyto_data	Data frame from extract_cytogenetic_data()

Value

Data frame with marker, n_positive, n_tested, pct columns

See Also

Other cytogenetics: calculate_cooccurrence(), compute_riss(), extract_cytogenetic_data(), plot_cooccurrence_heatmap(), plot_cytogenetic_oncoprint(), plot_expression_by_subtype()

Examples

```
## Not run:
cyto <- arrow::read_parquet("data/raw/clinical/cytogenetic_data.parquet")
summarize_cytogenetics(cyto)

## End(Not run)
```

---

summarize_data *Generate summary statistics*

Description

Generate summary statistics

Usage

```
summarize_data(se_data)
```

Arguments

se_data	A SummarizedExperiment object with a "counts" assay

summarize_de_methods *Summarize DE results across methods*

Description

Creates a summary table showing the number of significant genes per DE method.

Usage

```
summarize_de_methods(de_list, padj_threshold = 0.05, lfc_threshold = 1)
```

Arguments

de_list	Named list of DE result lists (each with results_table)
padj_threshold	Adjusted p-value threshold (default 0.05)
lfc_threshold	Log2 fold-change threshold (default 1)

Value

Data frame with method, n_tested, n_sig, n_up, n_down

See Also

Other differential-expression: compute_pca(), plot_heatmap_de(), plot_ma(), plot_pca(), plot_volcano(), run_deseq2(), run_vst()

summarize_treatment *Summarize treatment lines per patient*

Description

Creates a one-row-per-patient summary of treatment history.

Usage

```
summarize_treatment(treatment_clean)
```

Arguments

treatment_clean	Cleaned treatment data from [clean_treatment_data()]

Value

Data frame with columns: patient_id, n_lines, first_regimen, first_regimen_class, had_transplant, best_overall_response

See Also

Other data-cleaning: `clean_clinical_data()`, `clean_expression_data()`, `clean_treatment_data()`, `integrate_clinical_expression()`

Index