Package 'evanverse'

October 21, 2025

Type Package

Title Utility Functions for Data Analysis and Visualization

Version 0.3.7

Date 2025-10-16

Author Evan Zhou [aut, cre] (ORCID: <https://orcid.org/0009-0009-4600-8175>)

Maintainer Evan Zhou <evanzhou.bio@gmail.com>

Description A comprehensive collection of utility functions for data analysis and visualization in R. The package provides 55+ functions for data manipulation, file handling, color palette management, bioinformatics workflows, plotting, and package management. Features include void value handling, custom infix operators, flexible file I/O, and publication-ready visualizations with sensible defaults. Implementation follows tidyverse principles (Wickham et al. (2019) <doi:10.21105/joss.01686>) and incorporates best practices from the R community.

License MIT + file LICENSE

License_is_FOSS yes

License_restricts_use no

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

URL https://github.com/evanbio/evanverse,
https://evanbio.github.io/evanverse/

BugReports https://github.com/evanbio/evanverse/issues

VignetteBuilder knitr

Depends R (>= 4.1)

Imports cli, tibble, tidyr, data.table, dplyr, ggplot2, jsonlite, curl, openxlsx, readxl, tictoc, fs, rlang, withr, ggpubr, utils, tools, stats, grDevices

Suggests BiocManager, GSEABase, Biobase, GEOquery, biomaRt, knitr, rmarkdown, testthat (>= 3.0.0), ggvenn, ggVennDiagram, writexl, R.utils, janitor, RColorBrewer, devtools, digest, forestploter, purrr, reactable

NeedsCompilation no

Repository CRAN

Date/Publication 2025-10-21 09:00:02 UTC

Contents

any_void . . . 3
bio_palette_gallery . . . 4
check_pkg . . . 5
cols_with_void . . . 5
comb . . . 6
combine_logic . . . 7
compile_palettes . . . 8
convert_gene_id . . . 8
create_palette . . . 9
df2list . . . 10
df_forest_test . . . 11
download_batch . . . 12
download_gene_ref . . . 13
download_geo_data . . . 13
download_url . . . 15
drop_void . . . 17
file_info . . . 18
file_tree . . . 19
get_ext . . . 20
get_palette . . . 21
gmt2df . . . 22
gmt2list . . . 22
hex2rgb . . . 23
inst_pkg . . . 24
is_void . . . 25
list_palettes . . . 26
map_column . . . 26
perm . . . 27
pkg_functions . . . 28
pkg_version . . . 29
plot_bar . . . 29
plot_density . . . 30
plot_forest . . . 32
plot_pie . . . 33
plot_venn . . . 34
preview_palette . . . 36
read_excel_flex . . . 37

read_table_flex 38
remind 39
remove_palette 40
replace_void 40
rgb2hex 41
rows_with_void 42
safe_execute 43
set_mirror 43
trial 44
update_pkg 45
view 45
with_timer 46
write_xlsx_flex 47
%is% 48
%map% 48
%match% 49
%nin% 50
%p% 51

Index **52**

`any_void` *any_void(): Check if Any Value is Void (NA / NULL / "")*

Description

Test whether any element in a vector or list is considered "void". Void values include `NA`, `NULL`, and empty strings (`""`), and you can customize which ones to consider.

Usage

```
any_void(x, include_na = TRUE, include_null = TRUE, include_empty_str = TRUE)
```

Arguments

`x`	A vector or list to evaluate.
`include_na`	Logical. Consider `NA` as void. Default: TRUE.
`include_null`	Logical. Consider `NULL` as void. Default: TRUE.
`include_empty_str`	Logical. Consider `""` as void. Default: TRUE.

Value

A single logical value:

- `TRUE` if any void values are present.
- `FALSE` otherwise.
- For `NULL` input, returns `TRUE` if `include_null = TRUE`, else `FALSE`.

Examples

```
any_void(c("a", "", NA))                # TRUE
any_void(list("x", NULL, "y"))          # TRUE
any_void(c("a", "b", "c"))              # FALSE
any_void(NULL)                          # TRUE
any_void("", include_empty_str = FALSE) # FALSE
```

`bio_palette_gallery` *bio_palette_gallery(): Visualize All Palettes in a Gallery View*

Description

Display palettes from a compiled RDS in a paged gallery format.

Usage

```
bio_palette_gallery(
  palette_rds = NULL,
  type = c("sequential", "diverging", "qualitative"),
  max_palettes = 30,
  max_row = 12,
  verbose = TRUE
)
```

Arguments

`palette_rds`	Path to compiled RDS. Default: internal palettes.rds from `inst/extdata/`.
`type`	Palette types to include: "sequential", "diverging", "qualitative"
`max_palettes`	Number of palettes per page (default: 30)
`max_row`	Max colors per row (default: 12)
`verbose`	Whether to print summary/logs (default: TRUE)

Value

A named list of ggplot objects (one per page)

check_pkg *check_pkg(): Check if packages are installed and optionally install them*

Description

A utility to check whether CRAN / GitHub / Bioconductor packages are installed, with optional auto-installation via `inst_pkg()`.

Usage

```
check_pkg(
  pkg = NULL,
  source = c("CRAN", "GitHub", "Bioconductor"),
  auto_install = TRUE,
  ...
)
```

Arguments

`pkg`	Character vector of package names or GitHub repos (e.g., "r-lib/devtools").
`source`	Package source: one of "CRAN", "GitHub", "Bioconductor". Case-insensitive.
`auto_install`	Logical. If TRUE (default), install missing packages automatically.
`...`	Additional arguments passed to `inst_pkg()`.

Value

A tibble with columns: `package`, `name`, `installed`, `source`.

Examples

```
check_pkg("ggplot2", source = "CRAN")
check_pkg("r-lib/devtools", source = "GitHub", auto_install = FALSE)
```

cols_with_void *cols_with_void(): Detect Columns Containing Void Values*

Description

Scan a data.frame or tibble and identify columns that contain any "void" values. Void values include `NA`, `NULL`, and `""`, which can be toggled via parameters.

Usage

```
cols_with_void(
  data,
  include_na = TRUE,
  include_null = TRUE,
  include_empty_str = TRUE,
  return_names = TRUE
)
```

Arguments

data	A data.frame or tibble.
include_na	Logical. Detect NA if TRUE. Default: TRUE.
include_null	Logical. Detect NULL if TRUE. Default: TRUE.
include_empty_str	Logical. Detect "" if TRUE. Default: TRUE.
return_names	Logical. If TRUE (default), return column names; else logical vector.

Value

A character vector (column names) or logical vector indicating void presence per column.

Examples

```
df <- data.frame(name = c("A", "", "C"), score = c(1, NA, 3), id = 1:3)
cols_with_void(df)
cols_with_void(df, return_names = FALSE)
cols_with_void(df, include_na = FALSE)
```

---

comb *comb: Calculate Number of Combinations C(n, k)*

Description

Calculates the total number of ways to choose k items from n distinct items (without regard to order), i.e., the number of combinations C(n, k) = n! / (k! * (n - k)!). This function is intended for moderate n and k. For very large values, consider the 'gmp' package.

Usage

```
comb(n, k)
```

Arguments

n	Integer. Total number of items (non-negative integer).
k	Integer. Number of items to choose (non-negative integer, must be \u2264 n).

Value

Numeric. The combination count C(n, k) (returns Inf for very large n).

Examples

```
comb(8, 4)      # 70
comb(5, 2)      # 10
comb(10, 0)     # 1
comb(5, 6)      # 0
```

`combine_logic` *combine_logic: Combine multiple logical vectors with a logical operator*

Description

A utility function to combine two or more logical vectors using logical AND (&) or OR (|) operations. Supports NA handling and checks for consistent vector lengths.

Usage

```
combine_logic(..., op = "&", na.rm = FALSE)
```

Arguments

`...`	Logical vectors to combine.
`op`	Operator to apply: `"&"` (default) or `"|"`.
`na.rm`	Logical. If TRUE, treats NA values as TRUE (default is FALSE).

Value

A single logical vector of the same length as inputs.

Examples

```
x <- 1:5
combine_logic(x > 2, x %% 2 == 1)            # AND by default
combine_logic(x > 2, x %% 2 == 1, op = "|")  # OR logic
combine_logic(c(TRUE, NA), c(TRUE, TRUE), na.rm = TRUE)
```

compile_palettes *compile_palettes(): Compile JSON palettes into RDS*

Description

Read JSON files under `palettes_dir/`, validate content, and compile into a structured RDS file.

Usage

```
compile_palettes(palettes_dir, output_rds, log = TRUE)
```

Arguments

`palettes_dir`	Character. Folder containing subdirs: sequential/, diverging/, qualitative/ (required)
`output_rds`	Character. Path to save compiled RDS file (required). Use tempdir() for examples/tests.
`log`	Logical. Whether to log compilation events. Default: TRUE

Value

Invisibly returns RDS file path (character)

Examples

```
# Compile palettes using temporary directory:
compile_palettes(
  palettes_dir = system.file("extdata", "palettes", package = "evanverse"),
  output_rds = file.path(tempdir(), "palettes.rds")
)
```

convert_gene_id *convert_gene_id(): Convert gene identifiers using a reference table*

Description

Converts between Ensembl, Symbol, and Entrez gene IDs using a reference table. Supports both character vectors and data.frame columns. Automatically loads species-specific reference data from `data/`, or downloads if unavailable.

Usage

```
convert_gene_id(
  query,
  from = "symbol",
  to = c("ensembl_id", "entrez_id"),
  species = c("human", "mouse"),
  query_col = NULL,
  ref_table = NULL,
  keep_na = FALSE,
  preview = TRUE
)
```

Arguments

query	Character vector or data.frame to convert.
from	Source ID type (e.g., "symbol", "ensembl_id", "entrez_id").
to	Target ID type(s). Supports multiple.
species	Either "human" or "mouse". Default "human".
query_col	If query is a data.frame, the column name to convert.
ref_table	Optional reference table.
keep_na	Logical. Whether to keep unmatched rows. Default: FALSE.
preview	Logical. Whether to preview output. Default: TRUE.

Value

A data.frame containing original and converted columns.

---

create_palette *create_palette(): Save Custom Color Palettes as JSON*

Description

Save a named color palette (sequential, diverging, or qualitative) to a JSON file. Used for palette sharing, reuse, and future compilation.

Usage

```
create_palette(
  name,
  type = c("sequential", "diverging", "qualitative"),
  colors,
  color_dir,
  log = TRUE
)
```

Arguments

name	Character. Palette name (e.g., "Blues").
type	Character. One of "sequential", "diverging", or "qualitative".
colors	Character vector of HEX color values (e.g., "#E64B35" or "#E64B35B2").
color_dir	Character. Root folder to store palettes (required). Use tempdir() for examples/tests.
log	Logical. Whether to log palette creation to a temporary log file.

Value

(Invisibly) A list with path and info.

Examples

```
# Create palette in temporary directory:
temp_dir <- file.path(tempdir(), "palettes")
create_palette(
  "blues",
  "sequential",
  c("#deebf7", "#9ecae1", "#3182bd"),
  color_dir = temp_dir
)

create_palette(
  "vividset",
  "qualitative",
  c("#E64B35", "#4DBBD5", "#00A087"),
  color_dir = temp_dir
)

# Clean up
unlink(temp_dir, recursive = TRUE)
```

---

df2list *Convert Data Frame to Named List by Grouping*

Description

Group a data frame by one column and convert to named list. Each key becomes a list name; each value column becomes vector.

Usage

```
df2list(data, key_col, value_col, verbose = TRUE)
```

Arguments

data	A data.frame or tibble to be grouped.
key_col	Character. Column name for list names.
value_col	Character. Column name for list values.
verbose	Logical. Whether to show message. Default = TRUE.

Value

A named list, where each element is a character vector of values.

Examples

```
df <- data.frame(
  cell_type = c("T_cells", "T_cells", "B_cells", "B_cells"),
  marker = c("CD3D", "CD3E", "CD79A", "MS4A1")
)
df2list(df, "cell_type", "marker")
```

---

df_forest_test *Test Dataset for Forest Plots*

Description

A sample dataset used for demonstrating and testing forest plot functionality. Contains example effect sizes, confidence intervals, and study information.

Format

A data frame with 12 rows and 5 columns:

variable Character vector of variable names

estimate Numeric vector of effect estimates

conf.low Numeric vector of lower confidence limits

conf.high Numeric vector of upper confidence limits

p.value Numeric vector of p-values

Source

Created for testing and demonstration purposes.

download_batch *download_batch(): Batch download files using multi_download (parallel with curl)*

Description

A robust batch downloader that supports concurrent downloads with flexible options. Built on top of `curl::multi_download()` for parallelism.

Usage

```
download_batch(
  urls,
  dest_dir,
  overwrite = FALSE,
  unzip = FALSE,
  workers = 4,
  verbose = TRUE,
  timeout = 600,
  resume = FALSE,
  speed_limit = NULL,
  retries = 3
)
```

Arguments

urls	Character vector. List of URLs to download.
dest_dir	Character. Destination directory (required). Use tempdir() for examples/tests.
overwrite	Logical. Whether to overwrite existing files. Default: FALSE.
unzip	Logical. Whether to unzip after download (for supported formats). Default: FALSE.
workers	Integer. Number of parallel workers. Default: 4.
verbose	Logical. Show download progress messages. Default: TRUE.
timeout	Integer. Timeout in seconds for each download. Default: 600.
resume	Logical. Whether to resume interrupted downloads. Default: FALSE.
speed_limit	Numeric. Bandwidth limit in bytes/sec (e.g., 500000 = 500KB/s). Default: NULL.
retries	Integer. Retry attempts if download fails. Default: 3.

Value

Invisibly returns a list of downloaded (and optionally unzipped) file paths.

download_gene_ref *Download gene annotation reference table from Ensembl*

Description

Downloads a standardized gene annotation table for human or mouse using `biomaRt`. Includes Ensembl ID, gene symbol, Entrez ID, gene type, chromosome location, and other metadata.

Usage

```
download_gene_ref(
  species = c("human", "mouse"),
  remove_empty_symbol = FALSE,
  remove_na_entrez = FALSE,
  save = FALSE,
  save_path = NULL
)
```

Arguments

`species`	Organism, either `"human"` or `"mouse"`. Default is `"human"`.
`remove_empty_symbol`	Logical. Remove entries with missing gene symbol. Default: `FALSE`.
`remove_na_entrez`	Logical. Remove entries with missing Entrez ID. Default: `FALSE`.
`save`	Logical. Whether to save the result as `.rds`. Default: `FALSE`.
`save_path`	File path to save (optional). If `NULL`, will use default `gene_ref_<species>_<date>.rds`.

Value

A `data.frame` containing gene annotation.

download_geo_data *Download GEO Data Resources*

Description

Downloads GEO (Gene Expression Omnibus) datasets including expression data, supplemental files, and platform annotations with error handling and logging.

Usage

```
download_geo_data(
  gse_id,
  dest_dir,
  overwrite = FALSE,
  log = TRUE,
  log_file = NULL,
  retries = 2,
  timeout = 300
)
```

Arguments

gse_id	Character. GEO Series accession ID (e.g., "GSE12345").
dest_dir	Character. Destination directory for downloaded files.
overwrite	Logical. Whether to overwrite existing files (default: FALSE).
log	Logical. Whether to create log file (default: TRUE).
log_file	Character or NULL. Log file path (auto-generated if NULL).
retries	Numeric. Number of retry attempts (default: 2).
timeout	Numeric. Timeout in seconds (default: 300).

Details

Downloads GSEMatrix files, supplemental files, and GPL annotations. Includes retry mechanism, timeout control, and logging. Requires: GEOquery, Biobase, withr, cli.

Value

A list with components:

gse_object ExpressionSet object with expression data and annotations

supplemental_files Paths to downloaded supplemental files

platform_info Platform information (platform_id, gpl_files)

meta Download metadata (timing, file counts, etc.)

References

https://www.ncbi.nlm.nih.gov/geo/

Barrett T, Wilhite SE, Ledoux P, Evangelista C, Kim IF, Tomashevsky M, Marshall KA, Phillippy KH, Sherman PM, Holko M, Yefanov A, Lee H, Zhang N, Robertson CL, Serova N, Davis S, Soboleva A. NCBI GEO: archive for functional genomics data sets–update. Nucleic Acids Res. 2013 Jan; 41(Database issue):D991-5.

Examples

```
## Not run:
# Download GEO data (requires network connection):
result <- download_geo_data("GSE12345", dest_dir = tempdir())

# Advanced usage with custom settings:
result <- download_geo_data(
  gse_id = "GSE7305",
  dest_dir = tempdir(),
  log = TRUE,
  retries = 3,
  timeout = 600
)

# Access downloaded data:
expr_data <- Biobase::exprs(result$gse_object)
sample_info <- Biobase::pData(result$gse_object)
feature_info <- Biobase::fData(result$gse_object)

## End(Not run)
```

---

download_url *download_url(): Download File from URL*

Description

Downloads files from URLs (HTTP/HTTPS/FTP/SFTP) with robust error handling, retry mechanisms, and advanced features like resume, bandwidth limiting, and auto-extraction.

Usage

```
download_url(
  url,
  dest,
  overwrite = FALSE,
  unzip = FALSE,
  verbose = TRUE,
  timeout = 600,
  headers = NULL,
  resume = FALSE,
  speed_limit = NULL,
  retries = 3
)
```

Arguments

`url`	Character string. Full URL to the file to download. Supports HTTP, HTTPS, FTP, and SFTP protocols.
`dest`	Character string. Destination file path (required). Use file.path(tempdir(), basename(url)) for examples/tests.
`overwrite`	Logical. Whether to overwrite existing files. Default: FALSE.
`unzip`	Logical. Whether to automatically extract compressed files after download. Supports .zip, .gz, .tar.gz formats. Default: FALSE.
`verbose`	Logical. Whether to show download progress and status messages. Default: TRUE.
`timeout`	Numeric. Download timeout in seconds. Default: 600 (10 minutes).
`headers`	Named list. Custom HTTP headers for the request (e.g., list(Authorization = "Bearer token")). Default: NULL.
`resume`	Logical. Whether to attempt resuming interrupted downloads if a partial file exists. Default: FALSE.
`speed_limit`	Numeric. Bandwidth limit in bytes per second (e.g., 500000 = 500KB/s). Default: NULL (no limit).
`retries`	Integer. Number of retry attempts on download failure. Default: 3.

Details

This function provides a comprehensive solution for downloading files with:

Supported Protocols: Supports HTTP/HTTPS, FTP, and SFTP protocols.

Features: Includes retry mechanism, resume support, bandwidth control, auto-extraction, progress tracking, and custom headers.

Compression Support: Supports .zip, .gz, and .tar.gz formats.

Value

Invisible character string or vector of file paths:

If unzip = FALSE Path to the downloaded file

If unzip = TRUE Vector of paths to extracted files

Dependencies

Required packages: curl, cli, R.utils (automatically checked at runtime).

Examples

```
## Not run:
# Download a CSV file from GitHub:
download_url(
  url = "https://raw.githubusercontent.com/tidyverse/ggplot2/main/README.md",
  dest = file.path(tempdir(), "ggplot2_readme.md"),
  timeout = 30
)

# Download and extract a zip file:
download_url(
  url = "https://cran.r-project.org/src/contrib/Archive/dplyr/dplyr_0.8.0.tar.gz",
  dest = file.path(tempdir(), "dplyr.tar.gz"),
  unzip = TRUE,
  timeout = 60
)

## End(Not run)


# Quick demo with a tiny file:
download_url(
  url = "https://httpbin.org/robots.txt",
  dest = file.path(tempdir(), "robots.txt"),
  timeout = 10,
  verbose = FALSE
)
```

`drop_void` *drop_void: Remove Void Values from a Vector or List*

Description

Removes elements from a vector or list that are considered "void": `NA`, `NULL`, and empty strings (`""`). Each can be toggled via parameters.

Usage

```
drop_void(x, include_na = TRUE, include_null = TRUE, include_empty_str = TRUE)
```

Arguments

`x`	A vector or list.
`include_na`	Logical. Remove `NA` if TRUE. Default: TRUE.
`include_null`	Logical. Remove `NULL` if TRUE. Default: TRUE.
`include_empty_str`	Logical. Remove `""` if TRUE. Default: TRUE.

Value

A cleaned vector or list of the same type as input, with void values removed.

Examples

```
drop_void(c("apple", "", NA, "banana"))
drop_void(list("A", NA, "", NULL, "B"))
drop_void(c("", NA), include_na = FALSE)
```

`file_info` *file_info: Summarise file information*

Description

Given a file or folder path (or vector), returns a data.frame containing file name, size (MB), last modified time, optional line count, and path.

Usage

```
file_info(
  paths,
  recursive = FALSE,
  count_line = TRUE,
  preview = TRUE,
  filter_pattern = NULL,
  full_name = TRUE
)
```

Arguments

`paths`	Character vector of file paths or a folder path.
`recursive`	Logical. If a folder is given, whether to search recursively. Default: FALSE.
`count_line`	Logical. Whether to count lines in each file. Default: TRUE.
`preview`	Logical. Whether to show skipped/missing messages. Default: TRUE.
`filter_pattern`	Optional regex to filter file names (e.g., "\.R$"). Default: NULL.
`full_name`	Logical. Whether to return full file paths. Default: TRUE.

Value

A data.frame with columns: file, size_MB, modified_time, line_count, path.

Examples

```
file_info("R")
file_info(c("README.md", "DESCRIPTION"))
file_info("R", filter_pattern = "\\.R$", recursive = TRUE)
```

file_tree *file_tree: Print and Log Directory Tree Structure*

Description

Print the directory structure of a given path in a tree-like format using ASCII characters for maximum compatibility across different systems. Optionally, save the result to a log file for record keeping or debugging.

Usage

```
file_tree(
  path = ".",
  max_depth = 2,
  verbose = TRUE,
  log = FALSE,
  log_path = NULL,
  file_name = NULL,
  append = FALSE
)
```

Arguments

path	Character. The target root directory path to print. Default is ".".
max_depth	Integer. Maximum depth of recursion into subdirectories. Default is 2.
verbose	Logical. Whether to print the tree to console. Default is TRUE.
log	Logical. Whether to save the tree output as a log file. Default is FALSE.
log_path	Character. Directory path to save the log file if log = TRUE. Default is tempdir().
file_name	Character. Custom file name for the log file. If NULL, a name like "file_tree_YYYYMMDD_HHMMSS.l will be used.
append	Logical. If TRUE, appends to an existing file (if present). If FALSE, overwrites the file. Default is FALSE.

Value

Invisibly returns a character vector containing each line of the file tree.

Examples

```
# Basic usage with current directory:
file_tree()
file_tree(".", max_depth = 3)


# Example with temporary directory and logging:
temp_dir <- tempdir()
```

```
file_tree(temp_dir, max_depth = 2, log = TRUE, log_path = tempdir())
```

---

get_ext *get_ext: Extract File Extension(s)*

Description

Extract file extension(s) from a file name or path. Supports vector input and optionally preserves compound extensions (e.g., .tar.gz) when keep_all = TRUE.

Usage

```
get_ext(paths, keep_all = FALSE, include_dot = FALSE, to_lower = FALSE)
```

Arguments

paths	Character vector of file names or paths.
keep_all	Logical. If TRUE, returns full suffix after first dot in basename. If FALSE, returns only the last extension. Default is FALSE.
include_dot	Logical. If TRUE, includes the leading dot in result. Default is FALSE.
to_lower	Logical. If TRUE, converts extensions to lowercase. Default is FALSE.

Value

Character vector of extensions.

Examples

```
get_ext("data.csv")               # "csv"
get_ext("archive.tar.gz")         # "gz"
get_ext("archive.tar.gz", TRUE)   # "tar.gz"
get_ext(c("a.R", "b.txt", "c"))   # "R" "txt" ""
get_ext("data.CSV", to_lower = TRUE)  # "csv"
```

get_palette *Get Palette: Load Color Palette from RDS*

Description

Load a named palette from data/palettes.rds, returning a vector of HEX colors. Automatically checks for type mismatch and provides smart suggestions.

Usage

```
get_palette(
  name,
  type = c("sequential", "diverging", "qualitative"),
  n = NULL,
  palette_rds = system.file("extdata", "palettes.rds", package = "evanverse")
)
```

Arguments

name	Character. Name of the palette (e.g. "vividset").
type	Character. One of "sequential", "diverging", "qualitative".
n	Integer. Number of colors to return. If NULL, returns all colors. Default is NULL.
palette_rds	Character. Path to RDS file. Default uses system file in package.

Value

Character vector of HEX color codes.

Examples

```
get_palette("vividset", type = "qualitative")
get_palette("softtrio", type = "qualitative", n = 2)
get_palette("blues", type = "sequential", n = 3)
get_palette("contrast_duo", type = "diverging")
```

gmt2df *gmt2df: Convert GMT File to Long-format Data Frame*

Description

Reads a .gmt gene set file and returns a long-format data frame with one row per gene, including the gene set name and optional description.

Usage

```
gmt2df(file, verbose = TRUE)
```

Arguments

file	Character. Path to a .gmt file (supports .gmt or .gmt.gz).
verbose	Logical. Whether to show progress message. Default is TRUE.

Value

A tibble with columns: term, description, and gene.

Examples

```
## Not run:
# Requires a GMT file to run:
gmt_file <- "path/to/geneset.gmt"
result <- gmt2df(gmt_file)
head(result, 10)

## End(Not run)
```

gmt2list *gmt2list: Convert GMT File to Named List*

Description

Reads a .gmt gene set file and returns a named list, where each list element is a gene set.

Usage

```
gmt2list(file, verbose = TRUE)
```

Arguments

file	Character. Path to a .gmt file.
verbose	Logical. Whether to print message. Default is TRUE.

Value

A named list where each element is a character vector of gene symbols.

Examples

```
## Not run:
# Requires a GMT file to run:
gmt_file <- "path/to/geneset.gmt"
gene_sets <- gmt2list(gmt_file)
length(gene_sets)
names(gene_sets)[1:3]

## End(Not run)
```

---

hex2rgb *Convert HEX color(s) to RGB numeric components*

Description

Convert a single HEX color string or a character vector of HEX strings to RGB numeric components. The function accepts values with or without a leading #. Messaging uses `cli` if available and falls back to `message()`.

Usage

```
hex2rgb(hex)
```

Arguments

`hex`	Character. A HEX color string (e.g. `"#FF8000"`) or a character vector of HEX codes. No NA values allowed.

Value

If `hex` has length 1, a named numeric vector with elements `c(r, g, b)`. If `hex` has length > 1, a named list where each element is a named numeric vector for the corresponding input.

Examples

```
hex2rgb("#FF8000")
hex2rgb(c("#FF8000", "#00FF00"))
```

inst_pkg *Install R Packages from Multiple Sources*

Description

A unified installer for R packages from CRAN, GitHub, Bioconductor, or local source.

Usage

```
inst_pkg(
  pkg = NULL,
  source = c("CRAN", "GitHub", "Bioconductor", "Local"),
  path = NULL,
  ...
)
```

Arguments

pkg	Package name(s) or GitHub repo (e.g., "user/repo"). Not required for source = "local".
source	Source of package: "CRAN", "GitHub", "Bioconductor", "local". Case-insensitive, shorthand allowed.
path	Path to local package file (used when source = "local").
...	Additional arguments passed to install.packages(), devtools::install_github(), or BiocManager::install().

Value

NULL (invisibly)

Examples

```
## Not run:
# Install from CRAN:
inst_pkg("dplyr", source = "CRAN")

# Install from GitHub:
inst_pkg("hadley/emo", source = "GitHub")

# Install from Bioconductor:
inst_pkg("scRNAseq", source = "Bioconductor")

# Install from local file:
inst_pkg(source = "local", path = "mypackage.tar.gz")

## End(Not run)
```

```
# Quick demo - try to install a small package (will skip if already installed):
try(inst_pkg("praise", source = "CRAN"))
```

`is_void` *is_void(): Check for Null / NA / Blank ("") Values*

Description

Determine whether input values are considered "void": `NULL`, `NA`, or `""`. Each condition is controlled by a dedicated argument.

Usage

```
is_void(x, include_na = TRUE, include_null = TRUE, include_empty_str = TRUE)
```

Arguments

`x`	A vector or list to evaluate.
`include_na`	Logical. Consider `NA` as void. Default: TRUE.
`include_null`	Logical. Consider `NULL` as void. Default: TRUE.
`include_empty_str`	Logical. Consider `""` as void. Default: TRUE.

Value

A logical vector indicating which elements are void.

- If `x` is `NULL`, returns a single `TRUE` (if include_null=TRUE) or `FALSE`.
- If `x` is an empty vector, returns `logical(0)`.
- If `x` is a list, evaluates each element recursively and returns a flattened logical vector.
- For atomic vectors, returns a logical vector of the same length.

Examples

```
is_void(c(NA, "", "text"))                # TRUE TRUE FALSE
is_void(list(NA, "", NULL, "a"))          # TRUE TRUE TRUE FALSE
is_void("NA", include_na = FALSE)         # FALSE
is_void(NULL)                             # TRUE
```

list_palettes *list_palettes(): List All Color Palettes from RDS*

Description

Load and list all available color palettes compiled into an RDS file.

Usage

```
list_palettes(
  palette_rds = system.file("extdata", "palettes.rds", package = "evanverse"),
  type = c("sequential", "diverging", "qualitative"),
  sort = TRUE,
  verbose = TRUE
)
```

Arguments

palette_rds	Path to the RDS file. Default: "inst/extdata/palettes.rds".
type	Palette type(s) to filter: "sequential", "diverging", "qualitative". Default: all.
sort	Whether to sort by type, n_color, name. Default: TRUE.
verbose	Whether to print listing details to console. Default: TRUE.

Value

A data.frame with columns: name, type, n_color, colors.

Examples

```
list_palettes()
list_palettes(type = "qualitative")
list_palettes(type = c("sequential", "diverging"))
```

map_column *map_column(): Map values in a column using named vector or list*

Description

Maps values in a column of a data.frame (query) to new values using a named vector or list (map), optionally creating a new column or replacing the original.

Usage

```
map_column(
  query,
  by,
  map,
  to = "mapped",
  overwrite = FALSE,
  default = "unknown",
  preview = TRUE
)
```

Arguments

query	A data.frame containing the column to be mapped.
by	A string. Column name in query to be mapped.
map	A named vector or list. Names are original values, values are mapped values.
to	A string. Name of the column to store mapped results (if overwrite = FALSE).
overwrite	Logical. Whether to replace the by column with mapped values. Default: FALSE.
default	Default value to assign if no match is found. Default: "unknown".
preview	Logical. Whether to print preview of result (default TRUE).

Value

A data.frame with a new or modified column based on the mapping (returned invisibly).

Examples

```
df <- data.frame(gene = c("TP53", "BRCA1", "EGFR", "XYZ"))
gene_map <- c("TP53" = "Tumor suppressor", "EGFR" = "Oncogene")
map_column(df, by = "gene", map = gene_map, to = "label")
```

---

perm *Calculate Number of Permutations A(n, k)*

Description

Calculates the total number of ways to arrange k items selected from n distinct items, i.e., the number of permutations A(n, k) = n! / (n - k)!. This function is intended for moderate n and k. For very large numbers, consider supporting the 'gmp' package.

Usage

```
perm(n, k)
```

Arguments

n	Integer. Total number of items (non-negative integer).
k	Integer. Number of items selected for permutation (non-negative integer, must be <= n).

Value

Numeric. The permutation count A(n, k) (returns Inf for very large n).

Examples

```
perm(8, 4)      # 1680
perm(5, 2)      # 20
perm(10, 0)     # 1
perm(5, 6)      # 0
```

---

pkg_functions *pkg_functions: List exported functions from a package*

Description

List exported symbols from a package (via its NAMESPACE). Optionally filter by a case-insensitive keyword. Results are sorted alphabetically.

Usage

```
pkg_functions(pkg, key = NULL)
```

Arguments

pkg	Character scalar. Package name.
key	Optional character scalar. Keyword to filter function names (case-insensitive).

Value

Character vector of exported names (invisibly).

Examples

```
pkg_functions("evanverse")
pkg_functions("evanverse", key = "plot")
```

pkg_version *pkg_version: Check Installed and Latest Versions of R Packages*

Description

This function checks the installed and latest available versions of R packages across CRAN, Bioconductor, and GitHub. It supports case-insensitive matching and smart console previews.

Usage

```
pkg_version(pkg, preview = TRUE)
```

Arguments

pkg	Character vector of package names.
preview	Logical. If TRUE, print the result to console.

Value

A data.frame with columns: package, version (installed), latest (available), and source.

Examples

```
## Not run:
# Check versions of multiple packages (requires network):
pkg_version(c("ggplot2", "dplyr"))

# Check with preview disabled:
result <- pkg_version(c("ggplot2", "limma"), preview = FALSE)

## End(Not run)


# Quick demo with base R package:
try(pkg_version("base", preview = FALSE))
```

plot_bar *Bar plot with optional fill grouping, sorting, and directional layout*

Description

Create a bar chart from a data frame with optional grouping (`fill`), vertical/horizontal orientation, and sorting by values.

Usage

```
plot_bar(
  data,
  x,
  y,
  fill = NULL,
  direction = c("vertical", "horizontal"),
  sort = FALSE,
  sort_by = NULL,
  sort_dir = c("asc", "desc"),
  width = 0.7,
  ...
)
```

Arguments

`data`	A data frame.
`x`	Column name for the x-axis (quoted or unquoted).
`y`	Column name for the y-axis (quoted or unquoted).
`fill`	Optional character scalar. Column name to map to fill (grouping).
`direction`	Plot direction: "vertical" or "horizontal". Default: "vertical".
`sort`	Logical. Whether to sort bars based on y values. Default: FALSE.
`sort_by`	Optional. If `fill` is set and `sort = TRUE`, choose which level of `fill` is used for sorting.
`sort_dir`	Sorting direction: "asc" or "desc". Default: "asc".
`width`	Numeric. Bar width. Default: 0.7.
`...`	Additional args passed to `ggplot2::geom_bar()`, e.g. `alpha`, `color`.

Value

A `ggplot` object.

`plot_density` *plot_density: Univariate Density Plot (Fill Group, Black Outline)*

Description

Create a density plot with group color as fill, and fixed black border for all curves.

Usage

```
plot_density(
  data,
  x,
  group = NULL,
  facet = NULL,
  palette = c("#1b9e77", "#d95f02", "#7570b3"),
  alpha = 0.7,
  base_size = 14,
  xlab = NULL,
  ylab = "Density",
  title = NULL,
  legend_pos = "right",
  adjust = 1,
  show_mean = FALSE,
  mean_line_color = "red",
  add_hist = FALSE,
  hist_bins = NULL,
  add_rug = FALSE,
  theme = "minimal"
)
```

Arguments

data	data.frame. Input dataset.
x	Character. Name of numeric variable to plot.
group	Character. Grouping variable for fill color. (Optional)
facet	Character. Faceting variable. (Optional)
palette	Character vector. Fill color palette, e.g. c("#FF0000","#00FF00","#0000FF"). Will be recycled as needed. Cannot be a palette name. Default: c("#1b9e77", "#d95f02", "#7570b3")
alpha	Numeric. Fill transparency. Default: 0.7.
base_size	Numeric. Theme base font size. Default: 14.
xlab	Character. X-axis label. Default: NULL (uses variable name).
ylab	Character. Y-axis label. Default: "Density".
title	Character. Plot title. Default: NULL.
legend_pos	Character. Legend position. One of "right", "left", "top", "bottom", "none". Default: "right".
adjust	Numeric. Density bandwidth adjust. Default: 1.
show_mean	Logical. Whether to add mean line. Default: FALSE.
mean_line_color	Character. Mean line color. Default: "red".
add_hist	Logical. Whether to add histogram layer. Default: FALSE.
hist_bins	Integer. Number of histogram bins. Default: NULL (auto).

add_rug	Logical. Whether to add rug marks at bottom. Default: FALSE.
theme	Character. ggplot2 theme style. One of "minimal", "classic", "bw", "light", "dark". Default: "minimal".

Value

ggplot object.

`plot_forest` *Draw a forest plot using forestploter with publication-quality styling*

Description

Draw a forest plot using forestploter with publication-quality styling

Usage

```
plot_forest(
  data,
  estimate_col = "estimate",
  lower_col = "conf.low",
  upper_col = "conf.high",
  label_col = "variable",
  p_col = "p.value",
  ref_line = 1,
  sig_level = 0.05,
  bold_sig = TRUE,
  arrow_lab = c("Unfavorable", "Favorable"),
  ticks_at = c(0.5, 1, 1.5, 2),
  xlim = c(0, 3),
  footnote = "P-value < 0.05 was considered statistically significant",
  boxcolor = c("#E64B35", "#4DBBD5", "#00A087", "#3C5488", "#F39B7F", "#8491B4",
    "#91D1C2", "#DC0000", "#7E6148"),
  align_left = 1,
  align_right = NULL,
  align_center = NULL,
  gap_width = 30
)
```

Arguments

data	Data frame with required columns: estimate, lower, upper, label, and p-value.
estimate_col	Name of column containing point estimates.
lower_col	Name of column containing lower CI.
upper_col	Name of column containing upper CI.
label_col	Name of column for variable labels.

p_col	Name of column for p-values.
ref_line	Reference line value, typically 1 for OR/HR.
sig_level	Threshold to bold significant rows (default 0.05).
bold_sig	Whether to bold significant rows.
arrow_lab	Labels at both ends of the forest axis.
ticks_at	Vector of x-axis tick marks.
xlim	Range of x-axis (e.g., c(0, 3)). If NULL, auto-calculated. Default: c(0, 3).
footnote	Caption text below the plot.
boxcolor	Fill colors for CI boxes, will repeat if too short.
align_left	Integer column indices to left-align.
align_right	Integer column indices to right-align.
align_center	Integer column indices to center-align.
gap_width	Number of spaces in the gap column (default = 30).

Value

A forestplot object

---

plot_pie *Plot a Clean Pie Chart with Optional Inner Labels*

Description

Generate a polished pie chart from a vector or a grouped data frame. Labels (optional) are placed inside the pie slices.

Usage

```
plot_pie(
  data,
  group_col = "group",
  count_col = "count",
  label = c("none", "count", "percent", "both"),
  label_size = 4,
  label_color = "black",
  fill = c("#009076", "#C71E1D", "#15607A", "#FA8C00", "#18A1CD"),
  title = "Pie Chart",
  title_size = 14,
  title_color = "black",
  legend.position = "right",
  preview = TRUE,
  save = NULL,
  return_data = FALSE
)
```

Arguments

data	A character/factor vector or data.frame.
group_col	Group column name (for data.frame). Default: "group".
count_col	Count column name (for data.frame). Default: "count".
label	Type of label to display: "none", "count", "percent", or "both". Default: "none".
label_size	Label font size. Default: 4.
label_color	Label font color. Default: "black".
fill	Fill color vector. Default: 5-color palette.
title	Plot title. Default: "Pie Chart".
title_size	Title font size. Default: 14.
title_color	Title color. Default: "black".
legend.position	Legend position. Default: "right".
preview	Whether to print the plot. Default: TRUE.
save	Optional path to save the plot (e.g., "plot.png").
return_data	If TRUE, return list(plot = ..., data = ...). Default: FALSE.

Value

A ggplot object or list(plot, data)

---

plot_venn *Draw Venn Diagrams (2-4 sets, classic or gradient style)*

Description

A flexible and unified Venn diagram plotting function supporting both ggvenn and ggVennDiagram. Automatically handles naming, de-duplication, and visualization.

Usage

```
plot_venn(
  set1,
  set2,
  set3 = NULL,
  set4 = NULL,
  category.names = NULL,
  fill = c("skyblue", "pink", "lightgreen", "lightyellow"),
  label = "count",
  label_geom = "label",
  label_alpha = 0,
  fill_alpha = 0.5,
  label_size = 4,
```

```
  label_color = "black",
  set_color = "black",
  set_size = 5,
  edge_lty = "solid",
  edge_size = 0.8,
  title = "My Venn Diagram",
  title_size = 14,
  title_color = "#F06292",
  legend.position = "none",
  method = c("classic", "gradient"),
  digits = 1,
  label_sep = ",",
  show_outside = "auto",
  auto_scale = FALSE,
  palette = "Spectral",
  direction = 1,
  preview = TRUE,
  return_sets = FALSE,
  ...
)
```

Arguments

set1, set2, set3, set4	Input vectors. At least two sets are required.
category.names	Optional vector of set names. If NULL, variable names are used.
fill	Fill colors (for method = "classic").
label	Label type: "count", "percent", "both", or "none".
label_geom	Label geometry for ggVennDiagram: "label" or "text".
label_alpha	Background transparency for labels (only for gradient).
fill_alpha	Transparency for filled regions (only for classic).
label_size	Size of region labels.
label_color	Color of region labels.
set_color	Color of set labels and borders.
set_size	Font size for set names.
edge_lty	Line type for borders.
edge_size	Border thickness.
title	Plot title.
title_size	Title font size.
title_color	Title font color.
legend.position	Legend position. Default: "none".
method	Drawing method: "classic" (ggvenn) or "gradient" (ggVennDiagram).
digits	Decimal places for percentages (classic only).

label_sep	Separator for overlapping elements (classic only).
show_outside	Show outside elements (classic only).
auto_scale	Whether to auto-scale layout (classic only).
palette	Gradient palette name (gradient only).
direction	Palette direction (gradient only).
preview	Whether to print the plot to screen.
return_sets	If TRUE, returns a list of de-duplicated input sets.
...	Additional arguments passed to the underlying plot function.

Value

A ggplot object (and optionally a list of processed sets if return_sets = TRUE).

Examples

```
set.seed(123)
g1 <- sample(letters, 15)
g2 <- sample(letters, 10)
g3 <- sample(letters, 12)

# Classic 3-set Venn
plot_venn(g1, g2, g3, method = "classic", title = "Classic Venn")

# Gradient 2-set Venn
plot_venn(g1, g2, method = "gradient", title = "Gradient Venn")

# Return sets for downstream use
out <- plot_venn(g1, g2, return_sets = TRUE)
names(out)
```

---

preview_palette *Preview Palette: Visualize a Palette from RDS*

Description

Preview the appearance of a palette from data/palettes.rds using various plot types. This function provides multiple visualization options to help users evaluate color palettes.

Usage

```
preview_palette(
  name,
  type = c("sequential", "diverging", "qualitative"),
  n = NULL,
  plot_type = c("bar", "pie", "point", "rect", "circle"),
```

```
  title = name,
  palette_rds = system.file("extdata", "palettes.rds", package = "evanverse"),
  preview = TRUE
)
```

Arguments

name	Name of the palette.
type	Palette type: "sequential", "diverging", "qualitative".
n	Number of colors to use (default: all).
plot_type	Plot style: "bar", "pie", "point", "rect", "circle".
title	Plot title (default: same as palette name).
palette_rds	Path to RDS file. Default: system.file("extdata", "palettes.rds", package = "evanverse").
preview	Whether to show the plot immediately. Default: TRUE.

Value

NULL (invisible), for plotting side effect.

Examples

```
# Preview sequential palette:
preview_palette("blues", type = "sequential", plot_type = "bar")

# Preview diverging palette:
preview_palette("fire_ice_duo", type = "diverging", plot_type = "pie")

# Preview qualitative palette with custom colors:
preview_palette("balanced_quartet", type = "qualitative", n = 4, plot_type = "circle")
```

---

read_excel_flex *Flexible Excel reader*

Description

Read an Excel sheet via `readxl::read_excel()` with optional column-name cleaning (`janitor::clean_names()`), basic type control, and CLI messages.

Usage

```
read_excel_flex(
  file_path,
  sheet = 1,
  skip = 0,
  header = TRUE,
  range = NULL,
  col_types = NULL,
  clean_names = TRUE,
  guess_max = 1000,
  trim_ws = TRUE,
  na = "",
  verbose = TRUE
)
```

Arguments

file_path	Path to the Excel file (.xlsx or .xls).
sheet	Sheet name or index to read (default: 1).
skip	Number of lines to skip before reading data (default: 0).
header	Logical. Whether the first row contains column names (default: TRUE).
range	Optional cell range (e.g., "B2:D100"). Default: NULL.
col_types	Optional vector specifying column types; passed to readxl.
clean_names	Logical. Clean column names with janitor::clean_names() (default: TRUE).
guess_max	Max rows to guess column types (default: 1000).
trim_ws	Logical. Trim surrounding whitespace in text fields (default: TRUE).
na	Values to interpret as NA (default: "").
verbose	Logical. Show CLI output (default: TRUE).

Value

A tibble (or data.frame) read from the Excel sheet.

---

read_table_flex *Flexible and fast table reader using data.table::fread*

Description

Robust table reader with auto delimiter detection for .csv, .tsv, .txt, and their .gz variants. Uses data.table::fread() and prints CLI messages.

Usage

```
read_table_flex(
  file_path,
  sep = NULL,
  encoding = "UTF-8",
  header = TRUE,
  df = TRUE,
  verbose = FALSE
)
```

Arguments

file_path	Character. Path to the file to be read.
sep	Optional. Field delimiter. If NULL, auto-detected by file extension.
encoding	Character. File encoding accepted by fread: "unknown", "UTF-8", or "Latin-1".
header	Logical. Whether the file contains a header row. Default: TRUE.
df	Logical. Return data.frame instead of data.table. Default: TRUE.
verbose	Logical. Show progress and details. Default: FALSE.

Value

A data.frame (default) or data.table depending on df parameter.

---

remind *Show usage tips for common R commands*

Description

A helper to recall commonly used R functions with short examples.

Usage

```
remind(keyword = NULL)
```

Arguments

keyword	A keyword like "glimpse" or "read_excel". If NULL, show all.

Value

Invisibly returns the matched keywords (character vector).

Examples

```
remind("glimpse")
remind()  # show all keywords
```

remove_palette *Remove a Saved Palette JSON*

Description

Remove a palette file by name, trying across types if necessary.

Usage

```
remove_palette(name, type = NULL, color_dir, log = TRUE)
```

Arguments

name	Character. Palette name (without '.json' suffix).
type	Character. Optional. Preferred type ("sequential", "diverging", or "qualitative").
color_dir	Character. Root folder where palettes are stored (required). Use tempdir() for examples/tests.
log	Logical. Whether to log palette removal to a temporary log file.

Value

Invisibly TRUE if removed successfully, FALSE otherwise.

Examples

```
## Not run:
# Remove a palette (requires write permissions):
remove_palette("blues")

# Remove with specific type:
remove_palette("vividset", type = "qualitative")

## End(Not run)
```

replace_void *Replace void values (NA / NULL / "")*

Description

Replace elements in a vector or list considered "void" with a specified value. Void values include NA, NULL, and empty strings "" (toggle via flags).

Usage

```
replace_void(
  x,
  value = NA,
  include_na = TRUE,
  include_null = TRUE,
  include_empty_str = TRUE
)
```

Arguments

x	A vector or list.
value	The replacement value to use for voids. Default: NA.
include_na	Logical. Replace NA if TRUE. Default: TRUE.
include_null	Logical. Replace NULL if TRUE. Default: TRUE.
include_empty_str	Logical. Replace empty strings "" if TRUE. Default: TRUE.

Value

A cleaned vector or list with void values replaced.

Examples

```
replace_void(c(NA, "", "a"), value = "N/A")
replace_void(list("A", "", NULL, NA), value = "missing")
replace_void(c("", "b"), value = 0, include_empty_str = TRUE)
```

---

rgb2hex *Convert RGB values to HEX color codes*

Description

Convert an RGB triplet (or a list of triplets) to HEX color codes.

Usage

```
rgb2hex(rgb)
```

Arguments

rgb	A numeric vector of length 3 (e.g., c(255, 128, 0)), or a list of such vectors (e.g., list(c(255,128,0), c(0,255,0))).

Value

A HEX color string if a single RGB vector is provided, or a character vector of HEX codes if a list is provided.

Examples

```
rgb2hex(c(255, 128, 0))                          # "#FF8000"
rgb2hex(list(c(255,128,0), c(0,255,0)))          # c("#FF8000", "#00FF00")
```

`rows_with_void` *rows_with_void: Detect rows containing void values (NA / NULL / "")*

Description

Scan a data.frame or tibble and identify rows that contain any "void" values. Void values include `NA`, `NULL`, and empty strings `""` (toggle via flags).

Usage

```
rows_with_void(
  data,
  include_na = TRUE,
  include_null = TRUE,
  include_empty_str = TRUE
)
```

Arguments

`data`	A data.frame or tibble.
`include_na`	Logical. Detect `NA` if TRUE. Default: TRUE.
`include_null`	Logical. Detect `NULL` if TRUE. Default: TRUE.
`include_empty_str`	Logical. Detect empty strings `""` if TRUE. Default: TRUE.

Value

A logical vector of length `nrow(data)` indicating whether each row contains at least one void value.

Examples

```
df <- data.frame(id = 1:3, name = c("A", "", "C"), score = c(10, NA, 20))
rows_with_void(df)
df[rows_with_void(df), ]
```

safe_execute *Safely Execute an Expression*

Description

Evaluate code with unified error handling (and consistent warning reporting). On error, prints a CLI message (unless `quiet = TRUE`) and returns `NULL`.

Usage

```
safe_execute(expr, fail_message = "An error occurred", quiet = FALSE)
```

Arguments

`expr`	Code to evaluate.
`fail_message`	Message to display if an error occurs. Default: "An error occurred".
`quiet`	Logical. If `TRUE`, suppress messages. Default: `FALSE`.

Value

The result of the expression if successful; otherwise `NULL`.

Examples

```
safe_execute(log(1))
safe_execute(log("a"), fail_message = "Failed to compute log")
```

set_mirror *Set CRAN/Bioconductor Mirrors*

Description

Switch CRAN and/or Bioconductor mirrors for faster package installation.

Usage

```
set_mirror(repo = c("all", "cran", "bioc"), mirror = "tuna")
```

Arguments

`repo`	Character. Repository type: "cran", "bioc", or "all" (default: "all").
`mirror`	Character. Predefined mirror name (default: "tuna").

Value

Previous mirror settings (invisibly)

Examples

```
## Not run:
# Set all mirrors to tuna (default):
set_mirror()

# Set only CRAN mirror:
set_mirror("cran", "westlake")

# Set only Bioconductor mirror:
set_mirror("bioc", "ustc")

## End(Not run)


# Quick demo - view current mirror settings:
getOption("repos")
getOption("BioC_mirror")
```

---

trial *Clinical Trial Dataset*

Description

A sample clinical trial dataset used for testing and demonstration of data analysis functions. Contains typical clinical trial variables for testing various statistical and visualization functions.

Format

A data frame with 200 rows and 8 columns:

trt Character vector of treatment assignments

age Numeric vector of patient ages

marker Numeric vector of biomarker levels

stage Factor with tumor stage levels

grade Factor with tumor grade levels

response Integer vector indicating tumor response

death Integer vector indicating patient death

ttdeath Numeric vector of time to death/censoring

Source

Created for testing and demonstration purposes.

update_pkg *Update R Packages from CRAN, GitHub, or Bioconductor*

Description

A unified function to update R packages by source. Supports full updates, source-specific updates, or targeted package updates. Automatically sets mirrors (Tsinghua CRAN, Tsinghua Bioconductor) and handles version compatibility checks. Ensures Bioconductor installations specify the correct version to avoid mismatches.

Usage

```
update_pkg(pkg = NULL, source = NULL)
```

Arguments

pkg	Character vector. Name(s) of package(s) to update. For GitHub, use "user/repo" format. Only required when source is specified.
source	Character. The source of the package(s): "CRAN", "GitHub", or "Bioconductor". Optional if updating all installed CRAN and Bioconductor packages.

Value

Invisible NULL. Outputs update progress and logs via cli.

view *Quick interactive table viewer (reactable)*

Description

Quickly view a data.frame or tibble as an interactive table in the Viewer pane.

Usage

```
view(
  data,
  page_size = 10,
  searchable = TRUE,
  filterable = TRUE,
  striped = TRUE,
  highlight = TRUE,
  compact = FALSE
)
```

Arguments

data	A data.frame or tibble to display.
page_size	Number of rows per page (default = 10).
searchable	Whether to enable search (default = TRUE).
filterable	Whether to enable column filters (default = TRUE).
striped	Whether to show striped rows (default = TRUE).
highlight	Whether to highlight rows on hover (default = TRUE).
compact	Whether to use a compact layout (default = FALSE).

Value

A reactable widget rendered in the Viewer pane.

Examples

```
view(iris)
view(mtcars, page_size = 20, striped = TRUE, filterable = TRUE)
```

---

with_timer *Wrap a function to measure and display execution time*

Description

Wraps a function with CLI-based timing and prints its runtime in seconds. Useful for benchmarking or logging time-consuming tasks.

Usage

```
with_timer(fn, name = "Task")
```

Arguments

fn	A function to be wrapped.
name	A short descriptive name of the task (used in log output).

Details

Requires the `tictoc` package (CLI messages are emitted via `cli`).

Value

A function that executes `fn(...)` and prints timing information (returns invisibly).

Examples

```
slow_fn <- function(n) { Sys.sleep(0.01); n^2 }
timed_fn <- with_timer(slow_fn, name = "Square Task")
timed_fn(5)
```

---

write_xlsx_flex *Flexible Excel writer*

Description

Write a data frame or a **named** list of data frames to an Excel file with optional styling.

Usage

```
write_xlsx_flex(
  data,
  file_path,
  overwrite = TRUE,
  timestamp = FALSE,
  with_style = TRUE,
  auto_col_width = TRUE,
  open_after = FALSE,
  verbose = TRUE
)
```

Arguments

data	A data.frame, or a **named** list of data.frames.
file_path	Output path to a .xlsx file.
overwrite	Whether to overwrite if the file exists. Default: TRUE.
timestamp	Whether to append a date suffix (YYYY-MM-DD) to the filename. Default: FALSE.
with_style	Whether to apply a simple header style (bold, fill, centered). Default: TRUE.
auto_col_width	Whether to auto-adjust column widths. Default: TRUE.
open_after	Whether to open the file after writing (platform-dependent). Default: FALSE.
verbose	Whether to print CLI messages (info/warn/success). Errors are always shown. Default: TRUE.

Value

No return value; writes a file to disk.

%is% *Strict identity comparison with diagnostics*

Description

A semantic operator that checks whether two objects are strictly identical, and prints where they differ if not.

Usage

```
a %is% b
```

Arguments

a	First object (vector, matrix, or data.frame)
b	Second object (vector, matrix, or data.frame)

Value

TRUE if identical, FALSE otherwise (with diagnostics)

Examples

```
1:3 %is% 1:3                  # TRUE
1:3 %is% c(1, 2, 3)           # FALSE, type mismatch (integer vs double)
data.frame(x=1) %is% data.frame(y=1)  # FALSE, column name mismatch
m1 <- matrix(1:4, nrow=2)
m2 <- matrix(c(1,99,3,4), nrow=2)
m1 %is% m2                  # FALSE, value differs at [1,2]
c(a=1, b=2) %is% c(b=2, a=1) # FALSE, names differ
```

%map% *%map%: Case-insensitive mapping returning named vector*

Description

Performs case-insensitive matching between elements in `x` and entries in `table`, returning a named character vector: names are the matched entries from `table`, values are the original elements from `x`. Unmatched values are ignored (not included in the result).

Usage

```
x %map% table
```

Arguments

x	Character vector of input strings.
table	Character vector to match against.

Value

A named character vector. Names are from matched table values, values are from x. If no matches are found, returns a zero-length named character vector.

Examples

```
# Basic matching (case-insensitive)
c("tp53", "brca1", "egfr") %map% c("TP53", "EGFR", "MYC")
# returns: Named vector: TP53 = "tp53", EGFR = "egfr"

# Values not in table are dropped
c("akt1", "tp53") %map% c("TP53", "EGFR")
# returns: TP53 = "tp53"

# All unmatched values returns: empty result
c("none1", "none2") %map% c("TP53", "EGFR")
# returns: character(0)
```

%match% *%match%: Case-insensitive match returning indices*

Description

Performs case-insensitive matching, like base::match(), but ignores letter case.

Usage

```
x %match% table
```

Arguments

x	Character vector to match.
table	Character vector of values to match against.

Value

An integer vector of the positions of matches of x in table, like base::match(). Returns NA for non-matches. Returns an integer(0) if x is length 0.

Examples

```
# Basic matching
c("tp53", "BRCA1", "egfr") %match% c("TP53", "EGFR", "MYC")
# returns: 1 NA 2

# No matches returns: all NA
c("aaa", "bbb") %match% c("xxx", "yyy")

# Empty input
character(0) %match% c("a", "b")

# Order sensitivity (like match): first match is returned
c("x") %match% c("X", "x", "x")
# returns: 1
```

`%nin%` `%nin%`*: Not-in operator (negation of* `%in%`*)*

Description

A binary operator to test whether elements of the left-hand vector are **not** present in the right-hand vector. This is equivalent to `!(x %in% table)`.

Usage

```
x %nin% table
```

Arguments

`x`	vector or NULL: the values to be matched.
`table`	vector or NULL: the values to be matched against.

Value

A logical vector where `TRUE` indicates the corresponding element of `x` is not present in `table`. Results involving `NA` follow base R semantics: e.g., if `x` contains `NA` and `table` does not, the result at that position is `NA` (since `!NA` is `NA`).

Examples

```
c("A", "B", "C") %nin% c("B", "D")   # TRUE FALSE TRUE
1:5 %nin% c(2, 4)                    # TRUE FALSE TRUE FALSE TRUE
NA %nin% c(1, 2)                     # NA (since NA %in% c(1,2) is NA)
NA %nin% c(NA, 1)                    # FALSE (since NA is in table)

# Works with mixed types as `%in%` does:
c(1, "a") %nin% c("a", "b", 2)
```

%p% *%p%: paste two strings with a single space*

Description

An infix operator for string concatenation with one space between `lhs` and `rhs`. Inspired by the readability of `%>%`, intended for expressive text building.

Usage

```
lhs %p% rhs
```

Arguments

`lhs`	A character vector on the left-hand side.
`rhs`	A character vector on the right-hand side.

Value

A character vector, concatenating `lhs` and `rhs` with a single space.

Examples

```
"Hello" %p% "world"
"Good" %p% "job"
c("hello", "good") %p% c("world", "morning")   # vectorized
```

Index

∗ **datasets**
    df_forest_test, 11
    trial, 44
%is%, 48
%map%, 48
%match%, 49
%nin%, 50
%p%, 51

any_void, 3

base::match(), *49*
bio_palette_gallery, 4

check_pkg, 5
cols_with_void, 5
comb, 6
combine_logic, 7
compile_palettes, 8
convert_gene_id, 8
create_palette, 9

df2list, 10
df_forest_test, 11
download_batch, 12
download_gene_ref, 13
download_geo_data, 13
download_url, 15
drop_void, 17

file_info, 18
file_tree, 19

get_ext, 20
get_palette, 21
gmt2df, 22
gmt2list, 22

hex2rgb, 23

inst_pkg, 24

is_void, 25

list_palettes, 26

map_column, 26

perm, 27
pkg_functions, 28
pkg_version, 29
plot_bar, 29
plot_density, 30
plot_forest, 32
plot_pie, 33
plot_venn, 34
preview_palette, 36

read_excel_flex, 37
read_table_flex, 38
remind, 39
remove_palette, 40
replace_void, 40
rgb2hex, 41
rows_with_void, 42

safe_execute, 43
set_mirror, 43

trial, 44

update_pkg, 45

view, 45

with_timer, 46
write_xlsx_flex, 47