

Package ‘thisutils’

August 21, 2026

Type Package

Title Reliable Utilities for Reusable Research Workflows

Version 0.5.0

Date 2026-08-13

Maintainer Meng Xu <mengxu98@qq.com>

Description Provides reusable building blocks for research packages across matrix representation, numerical computation, neighborhood evaluation, controlled execution, and runtime interoperability. Core tools preserve declared sparse semantics, expose dense-memory and output boundaries, standardize neighborhood and classification results, and align serial and parallel result and error behavior. Additional helpers support structured messages, optional dependency checks, and common statistical workflows.

License MIT + file LICENSE

URL <https://mengxu98.github.io/thisutils/>

BugReports <https://github.com/mengxu98/thisutils/issues>

Depends R (>= 4.1.0)

Imports callr (>= 3.5.0), cli, methods, Matrix, pak, parallel, ps,
Rcpp, rlang, stats, utils

Suggests BiocNeighbors, knitr, remotes, rmarkdown, testthat (>= 3.1.7)

LinkingTo cli, Rcpp

Config/Needs/website mengxu98/thistemplate

Config/testthat/edition 3

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

Language en-US

NeedsCompilation yes

Author Meng Xu [aut, cre] (ORCID: <<https://orcid.org/0000-0002-8300-1054>>),
Haoliang Zhu [aut] (ORCID: <<https://orcid.org/0009-0004-9766-5179>>)

Repository CRAN

Date/Publication 2026-08-21 10:40:02 UTC

Contents

thisutils-package	3
add_pkg_file	4
as_matrix	4
capitalize	5
check_ci_env	6
check_pkg_status	6
check_r	7
check_sparsity	8
classification_metrics_compute	8
collapse_sparse_rows	9
compute_lisi	10
download	11
fast_row_vars	12
figlet	13
figlet_font	14
filter_nonzero_variance_features	14
get_namespace_fun	15
get_verbose	15
invoke_fun	16
is_apple_silicon	17
is_linux	17
is_osx	17
is_outlier	18
is_windows	19
log_message	19
matrix_process	26
matrix_to_table	27
maximump	28
max_depth	29
meanp	29
minimump	30
normalization	30
parallelize_fun	31
parse_inline_expressions	33
pearson_correlation	34
print.thisutils_logo	35
remove_r	35
remove_space	36
run_biocneighbors_knn	37
run_dense_topk	38
run_sparse_stored_topk_by_column	39
run_sparse_topk	39
run_sparse_topk_by_column	40
run_sparse_topk_stored	41
r_square	42
simulate_sparse_matrix	42

thisutils-package3

sparse_cor43

split_indices45

sump46

table_to_matrix47

thisutils_logo48

try_get49

unnest_fun50

votep51

wilkinsonp51

%ss%52

Index53

thisutils-package	<i>Reliable Utilities for Reusable Research Workflows</i>
-------------------	---

Description

Reusable building blocks for matrix representation, numerically stable and resource-controlled computation, neighborhood and classification evaluation, consistent serial/parallel execution, structured messages, and research-package interoperability.

Author(s)

Xu Meng (Maintainer), <mengxu98@qq.com>

Source

<https://mengxu98.github.io/thisutils/>

See Also

Useful links:

- <https://mengxu98.github.io/thisutils/>
- Report bugs at <https://github.com/mengxu98/thisutils/issues>

add_pkg_file	<i>Add a package file and print package information</i>
--------------	---

Description

Add a package file and print package information

Usage

```
add_pkg_file(
  use_figlet = TRUE,
  figlet_font = "Slant",
  colors = c("red", "yellow", "green", "magenta", "cyan", "yellow", "green", "white",
    "magenta", "cyan"),
  verbose = TRUE
)
```

Arguments

use_figlet	Whether to use figlet for ASCII art generation. Default is TRUE. Details see figlet .
figlet_font	Character string, figlet font to use. Default is "Slant".
colors	Character vector, colors to use for the logo elements.
verbose	Whether to print the message. Default is TRUE.

Value

Creates a file named R/<pkg_name>-package.R.

as_matrix	<i>Convert matrix into dense/sparse matrix</i>
-----------	--

Description

Convert matrix into dense/sparse matrix

Usage

```
as_matrix(x, return_sparse = FALSE)
```

Arguments

x	A matrix.
return_sparse	Whether to output a sparse matrix. Default is FALSE.

Value

A dense or sparse matrix.

Examples

```
m <- simulate_sparse_matrix(
  1000, 1000,
  decimal = 3
)

a <- as_matrix(m)
a[1:5, 1:5]

b <- as_matrix(m, return_sparse = TRUE)
b[1:5, 1:5]
```

capitalize

Capitalize the first letter of each word

Description

Capitalize the first letter of each word

Usage

```
capitalize(x, force_tolower = FALSE)
```

Arguments

x A vector of character strings to be capitalized.

force_tolower Whether to force the remaining letters to be lowercase.

Examples

```
x <- c(
  "hello world",
  "hello World"
)
capitalize(x)
```

check_ci_env	<i>Check CI environment</i>
--------------	-----------------------------

Description

Check CI environment

Usage

```
check_ci_env()
```

Value

A logical value.

check_pkg_status	<i>Check if a package is installed with the specified version</i>
------------------	---

Description

Check if a package is installed with the specified version

Usage

```
check_pkg_status(pkg, version = NULL, lib = .libPaths()[1])
```

Arguments

pkg	Package name.
version	Package version to check. If NULL, only checks if the package is installed.
lib	The location of the library directories where to install the packages.

Value

TRUE if the package is installed with the specified version, FALSE otherwise.

Description

Check and install R packages

Usage

```
check_r(  
  packages,  
  lib = .libPaths()[1],  
  dependencies = NA,  
  force = FALSE,  
  install = TRUE,  
  timeout = Inf,  
  load = FALSE,  
  cores = NULL,  
  verbose = TRUE  
)
```

Arguments

packages	Package to be installed. Package source can be <i>CRAN</i> , <i>Bioconductor</i> or <i>Github</i> . By default, the package name is extracted according to the packages parameter.
lib	The location of the library directories where to install the packages.
dependencies	Which dependencies to install. Passed to <code>pak::pkg_install</code> . Default is NA, auto install hard dependencies: <i>Depends</i> , <i>Imports</i> , and <i>LinkingTo</i> , excluding <i>Suggests</i> .
force	Whether to force the installation of packages. Default is FALSE.
install	Whether missing or outdated packages may be installed. Set to FALSE for read-only diagnostics. Default is TRUE for backward compatibility.
timeout	Maximum installation time in seconds. A finite timeout runs installation in a supervised R subprocess and terminates only that process tree on timeout. Default is Inf.
load	Whether to load packages after successful installation. Uses <code>do.call</code> dispatch to avoid CRAN static checks on <code>base::library</code> . Default is FALSE.
cores	Number of workers used by <code>pak::pkg_install()</code> . Use NULL (the default) to let pak select its worker count automatically.
verbose	Whether to print the message. Default is TRUE.

Details

GitHub packages are normally installed with pak. If pak cannot parse a GitHub package's DESCRIPTION file, `check_r()` retries that package with the optional `remotes` package. This preserves the fast dependency resolution path while supporting legacy repositories with malformed metadata. When `PKG_SUBPROCESS_TIMEOUT` is unset, the pak subprocess startup window is temporarily increased to 30 seconds and restored before returning.

Value

Package installation status.

<code>check_sparsity</code>	<i>Check sparsity of matrix</i>
-----------------------------	---------------------------------

Description

Check sparsity of matrix

Usage

```
check_sparsity(x)
```

Arguments

<code>x</code>	A matrix.
----------------	-----------

Value

Sparsity of matrix.

<code>classification_metrics_compute</code>	<i>Compute classification metrics</i>
---	---------------------------------------

Description

Compute multi-class classification metrics from predicted and true labels, including accuracy, macro-F1, purity, NMI, ARI, and rare-class recall.

Usage

```
classification_metrics_compute(predicted, truth, rare_threshold = 0.05)
```


Arguments

- predicted** Character vector of predicted class labels.
- truth** Character vector of true class labels (same length as predicted).
- rare_threshold** Proportion threshold for rare-class recall. Classes with support proportion $\leq$ this value contribute to the `rare_recall` metric. Default is 0.05.

Value

A list with the following components:

- accuracy** Overall accuracy (scalar).
- macro_f1** Macro-averaged F1 score (scalar).
- purity** Cluster purity (scalar).
- nmi** Normalized Mutual Information (scalar).
- ari** Adjusted Rand Index (scalar).
- rare_recall** Mean recall on rare classes (scalar, or NA if no rare classes).
- class_table** A data frame of per-class precision, recall, F1, and support.

Examples

```
predicted <- c("A", "A", "B", "B", "C")
truth <- c("A", "B", "B", "B", "C")
classification_metrics_compute(predicted, truth)
```

`collapse_sparse_rows` *Collapse sparse matrix rows by group*

Description

Collapse sparse matrix rows by group

Usage

```
collapse_sparse_rows(matrix, group)
```

Arguments

- matrix** A sparse matrix.
- group** A vector defining the output row groups.

Value

A sparse matrix with rows collapsed by ‘group’.

Examples

```
mat <- Matrix::Matrix(
  matrix(c(1, 0, 2, 0, 3, 4), nrow = 3, byrow = TRUE),
  sparse = TRUE
)
collapse_sparse_rows(mat, c("g1", "g1", "g2"))
```

compute_lisi

*Compute Local Inverse Simpson's Index (LISI)***Description**

Compute per-cell Local Inverse Simpson's Index (LISI) scores for one or more categorical variables. This is a clean-room reimplementation of the immunogenomics/LISI.

Usage

```
compute_lisi(
  X,
  meta_data,
  label_colnames,
  perplexity = 30,
  tol = 1e-05,
  max_iter = 50,
  knn_algorithm = c("auto", "brute_force", "clustered"),
  n_threads = NULL,
  max_dense_bytes = Inf
)
```

Arguments

X	A matrix-like object with cells in rows and embedding/features in columns.
meta_data	A data frame with one row per cell.
label_colnames	Character vector of column names in meta_data to evaluate.
perplexity	Effective neighborhood size. Defaults to 30.
tol	Tolerance used in the binary search for the target perplexity. Defaults to 1e-5.
max_iter	Maximum number of binary-search iterations. Defaults to 50.
knn_algorithm	Exact nearest-neighbor search strategy. "auto" uses clustered exact search when there are at least 4096 observations and the requested neighborhood is less than one quarter of the data; otherwise it uses brute force. "brute_force" and "clustered" force either exact strategy.
n_threads	Number of C++ worker threads. NULL (the default) preserves the earlier automatic hardware-thread selection. Supply a positive integer to control CPU use explicitly.

`max_dense_bytes`

Maximum estimated bytes allowed for the dense input and its C++ row-major copy. The default, `Inf`, preserves earlier behavior; supply a finite value to enable the guard.

Details

Both nearest-neighbor strategies are exact. Sparse X is converted to a dense matrix before neighbor search; this conversion is included in the `max_dense_bytes` check.

Value

A data frame with one row per cell and one column per label.

References

Korsunsky I, Millard N, Fan J, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature Methods* (2019). <https://www.nature.com/articles/s41592-019-0619-0>

LISI reference implementation: <https://github.com/immunogenomics/LISI>

Examples

```
set.seed(1)
X <- rbind(
  matrix(stats::rnorm(100, mean = -1), ncol = 2),
  matrix(stats::rnorm(100, mean = 1), ncol = 2)
)
meta_data <- data.frame(
  batch = rep(c("A", "B"), each = 50),
  group = sample(c("g1", "g2"), 100, replace = TRUE)
)

res <- compute_lisi(
  X, meta_data,
  c("batch", "group"),
  perplexity = 10
)
head(res)
boxplot(res)
```

download

Download file from the Internet

Description

Download file from the Internet

Usage

```
download(
  url,
  destfile,
  methods = c("auto", "wget", "libcurl", "curl", "wininet", "internal"),
  quiet = FALSE,
  ...,
  max_tries = 2
)
```

Arguments

url	a character string (or longer vector for the "libcurl" method) naming the URL of a resource to be downloaded.
destfile	a character string (or vector, see the url argument) with the file path where the downloaded file is to be saved. Tilde-expansion is performed.
methods	Methods to be used for downloading files. Can be "auto", "wget", "libcurl", "curl", "wininet", "internal". Default is "auto", which means to try different download methods.
quiet	If TRUE, suppress status messages (if any), and the progress bar.
...	Other arguments passed to utils::download.file .
max_tries	Number of tries for each download method. Default is 2.

fast_row_vars

Compute row variances

Description

Compute sample or population variances for each row of a dense or sparse matrix without densifying sparse input.

Usage

```
fast_row_vars(x, unbiased = TRUE)
```

Arguments

x	A matrix or a 'Matrix' sparse matrix.
unbiased	Whether to compute the unbiased sample variance (the default) instead of the population variance.

Value

A numeric vector of row variances. Row names are retained when available. Matrices with zero columns return 'NA' for every row. A one-column matrix returns 'NA' for sample variance and '0' for population variance.

Examples

```
fast_row_vars(matrix(c(1, 2, 3, 4), nrow = 2))
```

figlet	<i>The figlet function</i>
--------	----------------------------

Description

Create ASCII art text using figlet.

Usage

```
figlet(  
  text,  
  font = "Slant",  
  width = getOption("width", 80),  
  justify = "left",  
  absolute = FALSE,  
  strip = TRUE  
)
```

Arguments

text	Text to make bigger.
font	Name of font, path to font, or figlet_font object.
width	Width to use when justifying and breaking lines.
justify	Text justification to use in rendering ("left", "centre", "right").
absolute	Logical, indicating if alignment is absolute.
strip	Logical, indicating if whitespace should be removed.

Value

An object of class figlet_text which is a character vector with a handy print method.

References

<https://github.com/richfitz/rfiglet>, <https://github.com/jbkunst/figletr>, <https://www.figlet.org/>

Examples

```
figlet("thisutils")
```

figlet_font	<i>Get a figlet font</i>
-------------	--------------------------

Description

Get a figlet font

Usage

```
figlet_font(font)
```

Arguments

font	Path or name of the font to load
------	----------------------------------

Value

A 'figlet_font' object for use with [figlet]

filter_nonzero_variance_features	<i>Keep matrix rows with nonzero variance</i>
----------------------------------	---

Description

Subset a matrix to selected rows and discard rows with missing or zero variance. Sparse input remains sparse.

Usage

```
filter_nonzero_variance_features(x, features)
```

Arguments

x	A matrix or a 'Matrix' sparse matrix.
features	Row names or indices to retain before variance filtering.

Value

'x' restricted to selected rows whose variance is positive.

Examples

```
filter_nonzero_variance_features(
  matrix(c(1, 1, 1, 1, 1, 2), nrow = 2),
  features = 1:2
)
```

get_namespace_fun	<i>Get a function from a namespace</i>
-------------------	--

Description

Get a function from a namespace

Usage

```
get_namespace_fun(pkg, fun)
```

Arguments

pkg	The name of the package.
fun	The name of the function.

Value

Function.

get_verbose	<i>Get the verbose option</i>
-------------	-------------------------------

Description

Get the verbose option from the global options or the local argument.

Usage

```
get_verbose(verbose = NULL)
```

Arguments

verbose	The verbose option. Default is 'NULL', which means to get the verbose option from the global options.
---------	---

Value

The verbose option.

Examples

```

get_verbose()
get_verbose(verbose = FALSE)
get_verbose(verbose = TRUE)

options(log_message.verbose = FALSE)
get_verbose()
get_verbose(verbose = TRUE)

options(log_message.verbose = TRUE)
get_verbose()

options(log_message.verbose = NULL)

```

invoke_fun	<i>Invoke a function with a list of arguments</i>
------------	---

Description

Invoke a function with a list of arguments

Usage

```
invoke_fun(.fn, .args = list(), ..., .env = rlang::caller_env())
```

Arguments

<code>.fn</code>	A function, or function name as a string.
<code>.args</code>	A list of arguments.
<code>...</code>	Other arguments passed to the function.
<code>.env</code>	Environment in which to evaluate the call. This will be most useful if <code>.fn</code> is a string, or the function has side-effects.

Examples

```

f <- function(x, y) {
  x + y
}
invoke_fun(f, list(x = 1, y = 2))
invoke_fun("f", list(x = 1, y = 2))
invoke_fun("f", x = 1, y = 2)

```

is_apple_silicon	<i>Check if the system is running on Apple Silicon</i>
------------------	--

Description

Check if the system is running on Apple Silicon

Usage

```
is_apple_silicon()
```

Value

A logical value.

is_linux	<i>Check if the operating system is Linux</i>
----------	---

Description

Check if the operating system is Linux

Usage

```
is_linux()
```

Value

A logical value.

is_osx	<i>Check if the operating system is macOS</i>
--------	---

Description

Check if the operating system is macOS

Usage

```
is_osx()
```

Value

A logical value.

`is_outlier`*Detect outliers using MAD (Median Absolute Deviation)*

Description

Detect outliers using MAD (Median Absolute Deviation)

Usage

```
is_outlier(  
  x,  
  nmads = 2.5,  
  constant = 1.4826,  
  type = c("both", "lower", "higher")  
)
```

Arguments

<code>x</code>	Numeric vector.
<code>nmads</code>	Number of MADs from the median to define the boundaries for outliers. Default is 2.5.
<code>constant</code>	Constant factor to convert the MAD to a standard deviation. Default is 1.4826, which is consistent with the MAD of a normal distribution.
<code>type</code>	Type of outliers to detect. Available options are "both", "lower", or "higher". If type is "both", it detects both lower and higher outliers. If type is "lower", it detects only lower outliers. If type is "higher", it detects only higher outliers.

Value

Numeric vector of indices indicating the positions of outliers in `x`.

Examples

```
x <- c(1, 2, 3, 4, 5, 100)  
is_outlier(x) # returns 6  
  
x <- c(3, 4, 5, NA, 6, 7)  
is_outlier(x, nmads = 1.5, type = "lower") # returns 4  
  
x <- c(10, 20, NA, 15, 35)  
is_outlier(x, nmads = 2, type = "higher") # returns 3, 5
```

is_windows	<i>Check if the operating system is Windows</i>
------------	---

Description

Check if the operating system is Windows

Usage

```
is_windows()
```

Value

A logical value.

log_message	<i>Print formatted message</i>
-------------	--------------------------------

Description

Integrate the message printing function with the [cli](#) package, and the [base::message](#) function. The message could be suppressed by `base::suppressMessages`.

Usage

```
log_message(  
  ...,  
  expr = NULL,  
  verbose = NULL,  
  message_type = c("info", "success", "warning", "error", "running", "ask"),  
  cli_model = TRUE,  
  level = 1,  
  symbol = " ",  
  text_color = NULL,  
  back_color = NULL,  
  text_style = NULL,  
  multiline_indent = FALSE,  
  timestamp = TRUE,  
  timestamp_format = paste0("[", format(Sys.time(), "%Y-%m-%d %H:%M:%S"), "] "),  
  timestamp_style = FALSE,  
  plain_text = FALSE,  
  .envir = parent.frame(),  
  .frame = .envir  
)
```

Arguments

...	The message to print.
expr	An optional expression to evaluate while capturing its standard output, messages, and warnings, then re-printing them with <code>log_message()</code> formatting. The evaluated result is returned invisibly unless it is visible by default.
verbose	Whether to print the message. Default is TRUE.
message_type	Type of message. Could be choose one of "info", "success", "warning", "error", "running", and "ask". When "ask" is used, the function will prompt the user for a Yes/No/Cancel response using <code>utils::askYesNo</code> , and returns TRUE for Yes, FALSE for No, and NA for Cancel. Default is "info".
cli_model	Whether to use the <code>cli</code> package to print the message. Default is TRUE.
level	The level of the message, which affects the indentation. Level 1 has no indentation, higher levels add more indentation. Default is 1.
symbol	The symbol used for indentation. When specified, it ignores the level parameter and uses the symbol directly. Default is " " (two spaces).
text_color	Color for the message text. Supports R color names (e.g., "orange"), hexadecimal colors (e.g., "#000000"), basic colors: "red", "green", "blue", "yellow", "magenta", "cyan", "white", "black", "grey", "silver", "none", and bright colors: "br_red", "br_green", "br_blue", "br_yellow", "br_magenta", "br_cyan", "br_white", "br_black". Default is NULL.
back_color	Background color for the message text. Details see parameter <code>text_color</code> . Default is NULL.
text_style	Text styles to apply. Can be one or more of: "bold", "italic", "underline", "strikethrough", "dim", "inverse". Multiple styles can be combined (e.g., <code>c("bold", "underline")</code>). Default is NULL.
multiline_indent	Whether to apply consistent formatting (timestamp and indentation) to each line in multiline messages. When TRUE, each line gets the full formatting; when FALSE, only the first line gets the timestamp. Default is FALSE.
timestamp	Whether to show the current time in the message. Default is TRUE.
timestamp_format	Format string for timestamp display. Default is "%Y-%m-%d %H:%M:%S".
timestamp_style	Whether to apply the same text styling to the timestamp as the message text. When TRUE, timestamp formatting matches the message; when FALSE, timestamp keeps its default appearance. Default is FALSE.
plain_text	Whether to print only the text content. When TRUE, level, symbol, timestamp, and message type formatting are suppressed, but color and multiline settings still apply.
.envir	The environment to evaluate calls in. Default is <code>parent.frame</code> .
.frame	The frame to use for error reporting. Default is <code>.envir</code> .

Value

Formatted message, a logical value (TRUE/FALSE/NA) if message_type = "ask", or the evaluated result of expr if expr is supplied.

References

<https://cli.r-lib.org/articles/index.html>

Examples

```
# basic usage
log_message("Hello, ", "world!")

log_message("hello, world!")

log_message("Hello, world!", timestamp = FALSE)

log_message(
  "Hello, ", "world!",
  message_type = "success"
)

log_message(
  "Hello, world!",
  message_type = "warning"
)

log_message(
  "Processing data...",
  message_type = "running"
)

log_message(
  "Hello, ", "world!",
  cli_model = FALSE
)

# suppress messages
suppressMessages(log_message("Hello, world!"))
log_message("Hello, world!", verbose = FALSE)
options(log_message.verbose = FALSE)
log_message("Hello, world!")

# for global verbose option
options(log_message.verbose = TRUE)
log_message("Hello, world!", verbose = FALSE)
options(log_message.verbose = NULL)

# cli inline markup
log_message("{.arg abc} is a argument")
```

```

## 'message' can not deal with cli inline markup
message("hello, {.code world}!")

log_message("{.val list('abc')} is a {.cls {class(list('abc'))}}")

log_message("{.code lm(y ~ x)} is a code example")

log_message("{.dt List}list('abc')")

log_message("address: {.email example@example.com}")

log_message("{.emph R} is a programming language")

log_message("{.envvar R_HOME}")

log_message("{.file log_message.R} is a file")

log_message("{.fn lm} is a function")

log_message("{.fun lm} is a function")

log_message("{.help lm} to get help")

log_message("... see {.help [{.fun lm}](stats::lm)} to learn more")

log_message(
  "See the {.href [cli homepage](https://cli.r-lib.org)} for details"
)

log_message("press {.kbd ENTER}")

log_message("press {.key ENTER}")

log_message("URL: {.url https://cli.r-lib.org}")

log_message("Some {.field field}")

log_message("{.path /usr/bin/R} is a path")

log_message("{.pkg cli} is a package")

log_message("{.val object} is a variable")

log_message("{.run Rscript log_message.R} is a runnable file")

log_message("{.str object} is a string")

log_message("{.strong abc} is a strong string")

log_message("{.topic stats::lm} is a topic")

log_message("{.vignette cli} is a vignette")

```

```
# set indentation
log_message("Hello, world!", level = 2)

log_message("Hello, world!", symbol = "->")

log_message(
  "Hello, world!",
  symbol = "#####",
  level = 3
)

# color formatting
log_message(
  "This is a red message",
  text_color = "#ff9900"
)

log_message(
  "This is a message with background",
  back_color = "#EE4000"
)

log_message(
  "This is a message with both text and background",
  text_color = "white",
  back_color = "cyan"
)

log_message(
  "This is a message with background",
  back_color = "#EE4000",
  cli_model = FALSE
)

log_message(
  "This is a message with both text and background",
  text_color = "red",
  back_color = "cyan",
  cli_model = FALSE
)

log_message(
  "Hex color with {.arg cli_model = FALSE}",
  text_color = "#FF5733",
  cli_model = FALSE
)

log_message(
  "Bright red text",
  text_color = "br_red"
)
```

```
log_message(  
    "Bright background",  
    back_color = "br_yellow"  
)  
  
log_message(  
    "Combined grey and style",  
    text_color = "grey",  
    text_style = "bold"  
)  
  
# text style formatting  
log_message(  
    "Bold message",  
    text_style = "bold"  
)  
  
log_message(  
    "Italic message",  
    text_style = "italic"  
)  
  
log_message(  
    "Underlined message",  
    text_style = "underline"  
)  
  
log_message(  
    "Combined styles",  
    text_style = c("bold", "underline")  
)  
  
log_message(  
    "Color and style",  
    text_color = "blue",  
    text_style = c("bold", "italic")  
)  
  
log_message(  
    "Hex color and style",  
    text_color = "#FF5733",  
    text_style = c("bold", "underline")  
)  
  
# multiline message  
log_message(  
    "Line 1\nLine 2\nLine 3",  
    multiline_indent = TRUE,  
    text_style = "italic"  
)
```



```

log_message(
  "Multi-line\ncolored\nmessage",
  text_color = "blue",
  text_style = "italic"
)

log_message(
  "Multi-line\ncolored\nmessage",
  text_color = "blue",
  timestamp = FALSE
)

# timestamp styling
log_message(
  "Multi-line message\nwith timestamp styling",
  text_color = "red",
  text_style = "bold",
  timestamp_style = TRUE
)

log_message(
  "Multi-line message\nwithout timestamp styling",
  text_color = "#669999",
  text_style = c("bold", "italic")
)

# combine cli package and log_message
log_message(
  cli::col_green(
    "I am a green line ",
    cli::col_blue(
      cli::style_underline(
        cli::style_bold("with a blue substring")
      )
    ),
    " that becomes green again!"
  )
)

# cli variables
fun <- function(x = 1) {
  log_message("{.val x}")
  log_message("{.val {x}}")
  log_message("{.val {x + 1}}")
}
fun()

# print objects directly
df <- data.frame(
  x = 1:3,
  y = letters[1:3],

```

```

    z = c(" a", "b  ", "c")
  )
  log_message("Content:\n", df)

  # interactive prompt
  if (interactive()) {
    log_message(
      "Do you want to continue?",
      message_type = "ask"
    )
  }

  # capture output from another expression
  fun <- function() {
    cat("This is standard output\n")
    message("This is a message")
    return(1 + 1)
  }
  fun()

  log_message(
    expr = fun(),
    message_type = "running"
  )

```

matrix_process

*Process matrix***Description**

Process matrix

Usage

```

matrix_process(
  matrix,
  method = c("raw", "zscore", "fc", "log2fc", "log1p"),
  ...
)

```

Arguments

matrix	A matrix.
method	Method to use for processing the matrix.
...	Other arguments passed to the method.

Value

A processed matrix.

Examples

```
m <- simulate_sparse_matrix(10, 10)
matrix_process(m, method = "raw")
matrix_process(m, method = "zscore")
matrix_process(m, method = "fc")
matrix_process(m, method = "log2fc")
matrix_process(m, method = "log1p")
m <- as_matrix(m)
matrix_process(m, method = function(x) x / rowMeans(x))
```

matrix_to_table	<i>Switch matrix to table</i>
-----------------	-------------------------------

Description

Switch matrix to table

Usage

```
matrix_to_table(
  matrix,
  row_names = NULL,
  col_names = NULL,
  threshold = 0,
  keep_zero = TRUE
)
```

Arguments

matrix	A matrix.
row_names	Character vector of row names to filter by.
col_names	Character vector of column names to filter by.
threshold	The threshold for filtering values based on absolute values. Defaults to 0.
keep_zero	Whether to keep zero values in the table. Defaults to true for backward compatibility.

Details

With `keep_zero = FALSE`, sparse input is traversed through its stored entries and implicit zeros are not materialized. The backward-compatible default, `keep_zero = TRUE`, emits every selected matrix position, including implicit sparse zeros, and therefore requires time and output memory proportional to the selected `nrow * ncol`. When zeros are omitted, an all-zero row or column is absent from the table and cannot be reconstructed by `table_to_matrix()` without separate dimension metadata.

Value

A table with three columns: row, col, and value.

See Also

[table_to_matrix](#)

Examples

```
test_matrix <- simulate_sparse_matrix(10, 10)
colnames(test_matrix) <- paste0("c", 1:10)
rownames(test_matrix) <- paste0("r", 1:10)
table <- matrix_to_table(test_matrix)
matrix_new <- table_to_matrix(table)
test_matrix <- test_matrix[rownames(matrix_new), colnames(matrix_new)] |>
  as_matrix()
identical(test_matrix, matrix_new)

matrix_to_table(
  test_matrix,
  threshold = 2
)

matrix_to_table(
  test_matrix,
  row_names = c("r1", "r2"),
  col_names = c("c1", "c2")
)
```

maximump	<i>Maximum P-value</i>
----------	------------------------

Description

Maximum P-value

Usage

```
maximump(p, alpha = 0.05, log.p = FALSE)
```

Arguments

- p A vector of P-values.
- alpha The significance level.
- log.p Whether to return the log of the P-value.

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
maximump(p)
maximump(p, alpha = 0.01)
maximump(p, log.p = TRUE)
```

max_depth	<i>Maximum depth of a list</i>
-----------	--------------------------------

Description

Maximum depth of a list

Usage

```
max_depth(x, depth = 0)
```

Arguments

x	A list.
depth	The depth of the list.

Examples

```
x <- list(
  a = list(b = list(c = 1)),
  d = list(e = list(f = 2))
)
max_depth(x)
```

meanp	<i>Mean P-value</i>
-------	---------------------

Description

Mean P-value

Usage

```
meanp(p)
```

Arguments

p	A vector of P-values.
---	-----------------------

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
meanp(p)
```

minimump	<i>Minimum P-value</i>
----------	------------------------

Description

Minimum P-value

Usage

```
minimump(p, alpha = 0.05, log.p = FALSE)
```

Arguments

p	A vector of P-values.
alpha	The significance level.
log.p	Whether to return the log of the P-value.

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
minimump(p)
minimump(p, alpha = 0.01)
minimump(p, log.p = TRUE)
```

normalization	<i>Normalize numeric vector</i>
---------------	---------------------------------

Description

Normalize numeric vector

Usage

```
normalization(x, method = "max_min", na_rm = TRUE, ...)
```

Arguments

x	Input numeric vector.
method	Method used for normalization.
na_rm	Whether to remove NA values, and if setting TRUE, using 0 instead. Default is TRUE.
...	Parameters for other methods.

Value

Normalized numeric vector.

Examples

```
x <- c(runif(2), NA, -runif(2))
x
normalization(x, method = "max_min")
normalization(x, method = "maximum")
normalization(x, method = "sum")
normalization(x, method = "softmax")
normalization(x, method = "z_score")
normalization(x, method = "mad")
normalization(x, method = "unit_vector")
normalization(x, method = "unit_vector", na_rm = FALSE)
```

parallelize_fun	<i>Parallelize a function</i>
-----------------	-------------------------------

Description

Parallelize a function

Usage

```
parallelize_fun(
  x,
  fun,
  cores = 1,
  export_fun = NULL,
  clean_result = FALSE,
  throw_error = TRUE,
  progress_bar_width = 10L,
  timestamp_format = paste0("[", format(Sys.time(), "%Y-%m-%d %H:%M:%S"), "] "),
  verbose = TRUE,
  backend = c("auto", "fork", "psock"),
  timeout = Inf,
  total_timeout = Inf,
  seed = NULL,
  progress = verbose
)
```

Arguments

x	A vector or list to apply over.
fun	The function to be applied to each element.
cores	The number of worker processes to use for parallelization. Default is 1.
export_fun	Character vector naming functions or other objects from the environment of fun that PSOCK workers need. Objects referenced from a global environment are not discovered automatically; list them here or include them in each element of x.

<code>clean_result</code>	Whether to remove failed results from output. If FALSE, failed results are kept as error objects. Default is FALSE.
<code>throw_error</code>	Whether to print detailed error information for failed results. Default is TRUE.
<code>progress_bar_width</code>	Width of the verbose progress bar in characters. Default is 10L.
<code>timestamp_format</code>	Format string for timestamp display. Default is "%Y-%m-%d %H:%M:%S".
<code>verbose</code>	Whether to print the message. Default is TRUE.
<code>backend</code>	Parallel backend. "auto" uses PSOCK on every platform. Use "fork" to opt in to forked workers on supported systems. "fork" is unavailable on Windows. Prefer PSOCK in long-lived sessions that also use child-process managers such as callr or processx, because they can compete with R's fork-worker signal handler.
<code>timeout</code>	Maximum number of seconds that a parallel worker task may run. Inf disables task timeouts. This is ignored when execution uses one core.
<code>total_timeout</code>	Maximum number of seconds allowed for the complete call. Inf disables the overall deadline. In single-core mode the deadline is checked between inputs but cannot interrupt a function that is already running.
<code>seed</code>	Optional integer seed. When supplied, every input receives a deterministic independent L'Ecuyer-CMRG random-number stream, making results reproducible across worker counts and scheduling order. The caller's random number state is restored when the call finishes.
<code>progress</code>	Whether to draw a dynamic terminal progress bar. The default follows verbose. Set this to FALSE while keeping verbose = TRUE to emit concise lifecycle messages through <code>log_message()</code> without terminal timing output, for example in reports and persistent logs.

Value

A list of computed results. If `clean_result = FALSE`, failed results are included as error objects. If `clean_result = TRUE`, only successful results are returned.

Examples

```
parallelize_fun(1:3, function(x) {
  Sys.sleep(0.2)
  x^2
})

parallelize_fun(list(1, 2, 3), function(x) {
  Sys.sleep(0.2)
  x^2
}, cores = 2)

# Examples with error handling
parallelize_fun(1:5, function(x) {
  if (x == 3) stop("Error on element 3")
  x^2
})
```



```

    }, clean_result = FALSE)

    parallelize_fun(1:5, function(x) {
      if (x == 3) stop("Error on element 3")
      x^2
    }, clean_result = TRUE)

    # Control error printing
    parallelize_fun(1:5, function(x) {
      if (x == 2) stop("Error on element 3")
      if (x == 4) stop("Error on element 4")
      x^2
    })

    parallelize_fun(1:5, function(x) {
      if (x == 3) stop("Error on element 3")
      x^2
    }, throw_error = FALSE)

```

```
parse_inline_expressions
```

Parse inline expressions

Description

Parse “ inline expressions and evaluate them in the current environment, while preserving outer formatting markers like ‘{.val ...}’.

Usage

```
parse_inline_expressions(text, env = parent.frame())
```

Arguments

text	A character string containing inline expressions to parse.
env	Environment in which to evaluate expressions. Defaults to the calling environment.

Value

A character string with expressions evaluated but formatting preserved.

Examples

```

i <- 1
parse_inline_expressions(
  "{.val {i}}"
)

```

```

x <- 5
y <- 10
parse_inline_expressions(
  "{.pkg {x + y}}"
)

name <- "testing"
name <- parse_inline_expressions(
  "{.pkg {name}}"
)
name

log_message(name)

```

pearson_correlation *Correlation and covariance calculation for sparse matrix*

Description

Correlation and covariance calculation for sparse matrix

Usage

```
pearson_correlation(x, y = NULL, max_dense_bytes = Inf)
```

Arguments

x	A numeric matrix or sparse <code>Matrix</code> with observations in rows.
y	An optional numeric matrix or sparse <code>Matrix</code> with the same number of rows as x.
max_dense_bytes	Maximum estimated bytes allowed for the dense covariance, correlation, and cross-product matrices. The default is 'Inf' for backward compatibility. Supply a finite value to enable the guard.

Details

This lower-level helper deliberately returns dense covariance and correlation matrices and therefore requires memory proportional to `'ncol(x) * ncol(y)'`. Prefer `[sparse_cor()]` when a sparse, blockwise result is sufficient.

Value

A list with covariance and correlation matrices.

Examples

```

m1 <- simulate_sparse_matrix(
  100, 100
)
m2 <- simulate_sparse_matrix(
  100, 100,
  sparsity = 0.05
)
a <- pearson_correlation(m1, m2)
a$cov[1:5, 1:5]
a$cor[1:5, 1:5]

```

```
print.thisutils_logo
```

Print logo

Description

Print logo

Usage

```

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

```

Arguments

x	Input information.
...	Other parameters.

Value

Print the ASCII logo

```
remove_r
```

Check and remove R packages

Description

Check and remove R packages

Usage

```
remove_r(packages, lib = .libPaths()[1], verbose = TRUE)
```

Arguments

packages	Package to be removed.
lib	The location of the library directories where to remove the packages.
verbose	Whether to print the message. Default is TRUE.

remove_space	<i>Remove and normalize spaces</i>
--------------	------------------------------------

Description

Remove and normalize spaces

Usage

```
remove_space(
  x,
  trim_start = TRUE,
  trim_end = FALSE,
  collapse_multiple = TRUE,
  preserve_newlines = TRUE
)
```

Arguments

x	A vector of character strings.
trim_start	Whether to remove leading spaces before the first word. Default is TRUE.
trim_end	Whether to remove trailing spaces after the last word. Default is FALSE.
collapse_multiple	Whether to collapse multiple consecutive spaces between words into a single space. Default is TRUE.
preserve_newlines	Whether to preserve newline characters when collapsing spaces. Default is TRUE.

Value

A character vector with spaces normalized according to the specified parameters.

Examples

```
x <- c(
  " hello world ",
  " test case ",
  "no space",
  " multiple spaces "
)
```

```

remove_space(x)
remove_space(x, trim_start = FALSE)
remove_space(x, trim_end = TRUE)
remove_space(x, collapse_multiple = FALSE)
remove_space(
  x,
  trim_start = FALSE,
  trim_end = FALSE,
  collapse_multiple = FALSE
)

# with newlines
multiline <- c(
  "hello\n\n world ",
  " first \n second "
)
remove_space(multiline)
remove_space(multiline, preserve_newlines = FALSE)

```

run_biocneighbors_knn *Find nearest neighbors with BiocNeighbors*

Description

Find nearest neighbors within a reference matrix or from a query matrix to a reference matrix. The wrapper standardizes matrix coercion, distance metric, and self-neighbor removal across consumers.

Usage

```

run_biocneighbors_knn(
  reference,
  query = NULL,
  k,
  metric = c("euclidean", "cosine"),
  exclude_self = FALSE,
  n_threads = 1L
)

```

Arguments

reference	Numeric reference matrix with observations in rows.
query	Optional numeric query matrix with the same number of columns.
k	Number of nearest neighbors to return.
metric	Distance metric: "euclidean" or "cosine".
exclude_self	Whether to remove self-neighbors when 'query' is 'NULL'.
n_threads	Number of BiocNeighbors threads.

Value

A list with integer matrix 'idx' and numeric matrix 'dist'.

Examples

```
if (nzchar(system.file(package = "BiocNeighbors"))) {
  run_biocneighbors_knn(matrix(rnorm(20), ncol = 2), k = 2)
}
```

run_dense_topk

Dense matrix top-k selection

Description

For each column or row of a dense numeric matrix, extract the top k elements and their indices. The default returns the largest values; set decreasing = FALSE to retain the smallest values, as in nearest-neighbor selection from a distance matrix.

Usage

```
run_dense_topk(x, k, by = c("col", "row"), decreasing = TRUE)
```

Arguments

x	A numeric matrix (or something coercible to one).
k	Number of entries to retain per column or row. Must be a positive integer.
by	Direction of selection: "col" ranks row entries independently within each column, while "row" ranks column entries independently within each row.
decreasing	Whether to sort in decreasing order (largest values first). Default is TRUE.

Value

A list with two components:

idx Integer matrix of 1-based indices into the opposite dimension.

value Numeric matrix of corresponding values.

Both matrices have ncol(x) rows when by = "col" and nrow(x) rows when by = "row".

Examples

```
m <- matrix(rnorm(100), nrow = 10)
run_dense_topk(m, k = 3, by = "col")
run_dense_topk(m, k = 3, by = "row")
run_dense_topk(m, k = 3, by = "col", decreasing = FALSE)
```

run_sparse_stored_topk_by_column

Compatibility wrapper for stored column-wise sparse top-k selection

Description

run_sparse_stored_topk_by_column() preserves the stored-entry column-wise interface used in earlier releases. New code should call [run_sparse_topk_stored\(\)](#) with by = "col".

Usage

```
run_sparse_stored_topk_by_column(x, k, decreasing = TRUE)
```

Arguments

x	A dgMatrix (or something coercible to one).
k	Number of top entries to retain per column or row. Must be a positive integer.
decreasing	Whether to sort in decreasing order (largest values first). Default is TRUE.

Value

A list with idx and value matrices as described in [run_sparse_topk_stored\(\)](#).

run_sparse_topk

Sparse matrix top-k selection

Description

For each column or row of a sparse dgMatrix, extract the top k matrix elements and their indices. By default, unstored positions participate with their matrix value of zero. For column-wise selection, this gives the same semantics as [run_dense_topk\(\)](#).

Usage

```
run_sparse_topk(
  x,
  k,
  by = c("col", "row"),
  decreasing = TRUE,
  include_implicit_zeros = TRUE
)
```

Arguments

<code>x</code>	A dgCMatrix (or something coercible to one).
<code>k</code>	Number of top entries to retain per column or row. Must be a positive integer.
<code>by</code>	Direction of selection: "col" ranks row entries independently within each column, while "row" ranks column entries independently within each row.
<code>decreasing</code>	Whether to sort in decreasing order (largest values first). Default is TRUE.
<code>include_implicit_zeros</code>	Whether unstored sparse positions participate as zeros. The default, TRUE, gives ordinary matrix semantics. Set to FALSE to rank only stored entries; for that use case, prefer the more explicit run_sparse_topk_stored() .

Value

A list with two components:

idx Integer matrix of 1-based indices into the opposite dimension.

value Numeric matrix of corresponding values.

Both matrices have `ncol(x)` rows when `by = "col"` and `nrow(x)` rows when `by = "row"`.

Examples

```
m <- Matrix::rsparsematrix(10, 20, density = 0.3)
run_sparse_topk(m, k = 3, by = "col")
run_sparse_topk(m, k = 3, by = "row")
```

```
run_sparse_topk_by_column
```

Compatibility wrapper for column-wise sparse top-k selection

Description

`run_sparse_topk_by_column()` preserves the column-wise interface used in earlier releases: only explicitly stored entries participate, and missing ranks are padded with NA indices and zero values. New code should call [run_sparse_topk\(\)](#) or [run_sparse_topk_stored\(\)](#) to select the intended sparse-zero semantics explicitly.

Usage

```
run_sparse_topk_by_column(x, k, decreasing = TRUE)
```

Arguments

<code>x</code>	A dgCMatrix (or something coercible to one).
<code>k</code>	Number of stored entries to retain per column.
<code>decreasing</code>	Whether to sort in decreasing order. Defaults to TRUE.

Value

A list with `idx` and `value` matrices. Columns with fewer than `k` stored entries use NA indices and zero values for the remaining ranks.

`run_sparse_topk_stored`

Stored sparse entries top-k selection

Description

For each column or row of a sparse `dgMatrix`, rank only explicitly stored entries. Unstored zeros never become candidates. This is useful for sparse adjacency matrices, where stored entries represent graph edges rather than samples from a full numeric matrix.

Usage

```
run_sparse_topk_stored(x, k, by = c("col", "row"), decreasing = TRUE)
```

Arguments

<code>x</code>	A <code>dgMatrix</code> (or something coercible to one).
<code>k</code>	Number of top entries to retain per column or row. Must be a positive integer.
<code>by</code>	Direction of selection: "col" ranks row entries independently within each column, while "row" ranks column entries independently within each row.
<code>decreasing</code>	Whether to sort in decreasing order (largest values first). Default is TRUE.

Value

A list with `idx` and `value` matrices as described in [run_sparse_topk\(\)](#). Groups with fewer than `k` stored entries are padded with NA in both matrices.

Examples

```
graph <- Matrix::sparseMatrix(
  i = c(1, 3, 2), j = c(1, 1, 2), x = c(1, 3, -2),
  dims = c(3, 2)
)
run_sparse_topk_stored(graph, k = 2, by = "col")
```

r_square	<i>Coefficient of determination (R^2)</i>
----------	--

Description

Coefficient of determination (R^2)

Usage

```
r_square(y_true, y_pred)
```

Arguments

y_true	A numeric vector with ground truth values.
y_pred	A numeric vector with predicted values.

Value

The R^2 value.

Examples

```
y <- rnorm(100)
y_pred <- y + rnorm(100, sd = 0.5)
r_square(y, y_pred)
```

simulate_sparse_matrix	<i>Generate a simulated sparse matrix</i>
------------------------	---

Description

This function generates a sparse matrix with a specified number of rows and columns, a given sparsity level, and a distribution function for the non-zero values.

Usage

```
simulate_sparse_matrix(
  nrow,
  ncol,
  sparsity = 0.95,
  distribution_fun = function(n) stats::rpois(n, lambda = 0.5) + 1,
  decimal = 0,
  seed = 1
)
```

Arguments

nrow	Number of rows in the matrix.
ncol	Number of columns in the matrix.
sparsity	Proportion of zero elements (sparsity level). Default is 0.95, meaning 95% of elements are zero (5% are non-zero).
distribution_fun	Function to generate non-zero values.
decimal	Controls the number of decimal places in the generated values. If set to 0, values will be integers. When decimal > 0, random decimal parts are uniformly distributed across the full range. Default is 0.
seed	Random seed for reproducibility.

Value

A sparse matrix of class "dgCMatrix".

Examples

```
simulate_sparse_matrix(1000, 500) |>
  check_sparsity()

simulate_sparse_matrix(10, 10, decimal = 1)
simulate_sparse_matrix(10, 10, decimal = 5)
```

sparse_cor

Resource-controlled sparse correlation

Description

Compute correlations from matrix-like input and return a sparse matrix. Pearson cross-products remain sparse and correlations are evaluated in column blocks so block-local working memory is bounded independently of the full output dimensions.

Usage

```
sparse_cor(
  x,
  y = NULL,
  method = c("pearson", "spearman", "kendall"),
  allow_neg = TRUE,
  remove_na = TRUE,
  remove_inf = TRUE,
  threshold = 0,
  block_size = 256L,
  max_dense_bytes = Inf,
  max_output_entries = Inf,
  ...
)
```

Arguments

<code>x</code>	A numeric matrix or sparse <code>Matrix</code> with observations in rows.
<code>y</code>	An optional numeric matrix or sparse <code>Matrix</code> with the same number of rows as <code>x</code> .
<code>method</code>	Correlation coefficient: "pearson", "spearman", or "kendall".
<code>allow_neg</code>	Logical. Whether to allow negative values or set them to 0.
<code>remove_na</code>	Logical. Whether to replace NA values with 0.
<code>remove_inf</code>	Logical. Whether to replace infinite values with 0.
<code>threshold</code>	Non-negative absolute correlation threshold. Values with absolute magnitude below this threshold are omitted from the sparse output.
<code>block_size</code>	Maximum number of target columns in each Pearson working block. The effective size may be reduced to honor <code>max_dense_bytes</code> .
<code>max_dense_bytes</code>	Maximum estimated bytes allowed for block-local Pearson working arrays or dense rank-correlation arrays. The default is <code>Inf</code> for backward compatibility. Supply a finite value to enable this guard.
<code>max_output_entries</code>	Maximum number of stored values allowed in the sparse result. The default, <code>Inf</code> , preserves the historical unbounded output behavior; set a finite value to fail before returning an output that exceeds the workflow's storage budget.
<code>...</code>	Other arguments passed to <code>stats::cor()</code> for Spearman and Kendall correlations.

Details

Pearson input and cross-products remain sparse. Centering, scaling, and thresholding are fused in a column-block scan rather than materializing a dense correlation block. The final sparse result can nevertheless contain up to $\text{ncol}(x) * \text{ncol}(y)$ stored values when `threshold = 0`; use `threshold` and `max_output_entries` to make this boundary explicit.

Spearman and Kendall correlation require ranking and currently densify the input and full correlation result. Their estimated peak working size is checked against `max_dense_bytes` before conversion.

Value

A sparse correlation matrix.

Examples

```
m1 <- simulate_sparse_matrix(
  500, 100
)
m2 <- simulate_sparse_matrix(
  500, 100,
  seed = 2025
)
```

```
a <- sparse_cor(m1)
b <- sparse_cor(m1, m2)
c <- as_matrix(
  cor(as_matrix(m1)),
  return_sparse = TRUE
)
d <- as_matrix(
  cor(as_matrix(m1), as_matrix(m2)),
  return_sparse = TRUE
)

a[1:5, 1:5]
c[1:5, 1:5]
all.equal(a, c)

b[1:5, 1:5]
d[1:5, 1:5]
all.equal(b, d)

m1[sample(1:500, 10)] <- NA
m2[sample(1:500, 10)] <- NA

sparse_cor(m1, m2)[1:5, 1:5]

system.time(
  sparse_cor(m1)
)
system.time(
  cor(as_matrix(m1))
)

system.time(
  sparse_cor(m1, m2)
)
system.time(
  cor(as_matrix(m1), as_matrix(m2))
)
```

split_indices

Split indices.

Description

An optimised version of split for the special case of splitting row indices into groups.

Usage

```
split_indices(group, n = 0L)
```

Arguments

group	Integer indices
n	The largest integer (may not appear in index). This is hint: if the largest value of group is bigger than n, the output will silently expand.

Value

A list of vectors of indices.

References

<https://github.com/hadley/plyr/blob/d57f9377eb5d56107ba3136775f2f0f005f33aa3/src/split-numeric.cpp#L20>

Examples

```
split_indices(sample(10, 100, rep = TRUE))
split_indices(sample(10, 100, rep = TRUE), 10)
```

sump

Sum P-value

Description

Sum P-value

Usage

```
sump(p)
```

Arguments

p	A vector of P-values.
---	-----------------------

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
sump(p)
```

table_to_matrix	<i>Switch table to matrix</i>
-----------------	-------------------------------

Description

Switch table to matrix

Usage

```
table_to_matrix(  
  table,  
  row_names = NULL,  
  col_names = NULL,  
  threshold = 0,  
  return_sparse = FALSE  
)
```

Arguments

table	A table with three columns: row, col, and value.
row_names	Character vector of row names to filter by.
col_names	Character vector of column names to filter by.
threshold	The threshold for filtering values based on absolute values. Defaults to 0.
return_sparse	Whether to return a sparse matrix. Defaults to false.

Value

A matrix.

See Also

[matrix_to_table](#)

Examples

```
table <- data.frame(  
  row = c("r1", "r2", "r3", "r4", "r5", "r6"),  
  col = c("c4", "c5", "c6", "c1", "c2", "c3"),  
  value = c(0.6, -0.5, -0.4, 0.3, 0.2, 0.1)  
)  
matrix <- table_to_matrix(table)  
table_new <- matrix_to_table(matrix)  
identical(table, table_new)  
  
table_to_matrix(table, threshold = 0.3)  
  
table_to_matrix(  
  table,
```

```

    table,
    row_names = c("r1", "r2"),
    col_names = c("c4", "c5")
  )

  sparse_matrix <- simulate_sparse_matrix(10, 10)
  table_sparse <- matrix_to_table(
    sparse_matrix
  )
  sparse_matrix_new <- table_to_matrix(
    table_sparse,
    return_sparse = TRUE
  )
  identical(sparse_matrix, sparse_matrix_new)

```

 thisutils_logo

The logo of thisutils

Description

The thisutils logo, using ASCII or Unicode characters Use [cli::ansi_strip](#) to get rid of the colors.

Usage

```
thisutils_logo(unicode = cli::is_utf8_output())
```

Arguments

unicode Unicode symbols on UTF-8 platforms. Default is [cli::is_utf8_output](#).

Value

A character vector with class thisutils_logo.

References

<https://github.com/tidyverse/tidyverse/blob/main/R/logo.R>

Examples

```
thisutils_logo()
```

try_get*Try to evaluate an expression a set number of times before failing*

Description

The function is used as a fail-safe if code sometimes works and sometimes doesn't, usually because it depends on a resource that may be temporarily unavailable. It tries to evaluate the expression `max_tries` times. If all the attempts fail, it throws an error; if not, the evaluated expression is returned.

Usage

```
try_get(expr, max_tries = 5, error_message = "", retry_message = "Retrying...")
```

Arguments

<code>expr</code>	The expression to be evaluated.
<code>max_tries</code>	The maximum number of attempts to evaluate the expression before giving up. Default is 5.
<code>error_message</code>	Additional custom error message to be displayed when an error occurs.
<code>retry_message</code>	Message displayed when a new try to evaluate the expression would be attempted.

Value

The evaluated expression if successful, otherwise it throws an error if all attempts are unsuccessful.

Examples

```
f <- function() {  
  value <- runif(1, min = 0, max = 1)  
  if (value > 0.5) {  
    log_message("value is larger than 0.5")  
    return(value)  
  } else {  
    log_message(  
      "value is smaller than 0.5",  
      message_type = "error"  
    )  
  }  
}  
f_evaluated <- try_get(expr = f())  
print(f_evaluated)
```

unnest_fun

*Unnest a list-column***Description**

Implement similar functions to the `tidyr::unnest` function.

Usage

```
unnest_fun(data, cols, keep_empty = FALSE)
```

Arguments

<code>data</code>	A data frame.
<code>cols</code>	Columns to unnest.
<code>keep_empty</code>	By default, you get one row of output for each element of the list your unchopping/unnesting. This means that if there's a size-0 element (like NULL or an empty data frame), that entire row will be dropped from the output. If you want to preserve all rows, use <code>keep_empty = TRUE</code> to replace size-0 elements with a single row of missing values.

Examples

```
data <- data.frame(
  id = 1:3,
  x = c("a", "b", "c"),
  stringsAsFactors = FALSE
)
data$data <- list(
  c(1, 2),
  c(3, 4, 5),
  c(6)
)
unnest_fun(data, cols = "data")

data2 <- data.frame(
  id = 1:3,
  x = c("a", "b", "c"),
  stringsAsFactors = FALSE
)
data2$data <- list(
  c(1, 2),
  numeric(0),
  c(6)
)
unnest_fun(data2, cols = "data")
unnest_fun(data2, cols = "data", keep_empty = TRUE)
```

votep	<i>Vote P-value</i>
-------	---------------------

Description

Vote P-value

Usage

```
votep(p, alpha = 0.5)
```

Arguments

p	A vector of P-values.
alpha	The significance level.

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
votep(p)
votep(p, alpha = 0.01)
```

wilkinsonp	<i>Wilkinson's P-value</i>
------------	----------------------------

Description

Wilkinson's P-value

Usage

```
wilkinsonp(p, r = 1, alpha = 0.05, log.p = FALSE)
```

Arguments

p	A vector of P-values.
r	The number of studies to include in the P-value calculation.
alpha	The significance level.
log.p	Whether to return the log of the P-value.

Examples

```
p <- c(0.01, 0.02, 0.03, 0.04, 0.05)
wilkinsonp(p)
wilkinsonp(p, r = 2)
wilkinsonp(p, alpha = 0.01)
wilkinsonp(p, log.p = TRUE)
```

%ss%

Value selection operator

Description

This operator returns the left side if it's not NULL, otherwise it returns the right side.

Usage

a %ss% b

Arguments

- | | |
|---|---|
| a | The left side value to check. |
| b | The right side value to use if a is NULL. |

Value

a if it is not NULL, otherwise b.

Examples

```
NULL %ss% 10  
5 %ss% 10
```

Index

`%ss%`, 52

`add_pkg_file`, 4

`as_matrix`, 4

`base::library`, 7

`base::message`, 19

`capitalize`, 5

`character`, 12

`check_ci_env`, 6

`check_pkg_status`, 6

`check_r`, 7

`check_sparsity`, 8

`classification_metrics_compute`, 8

`cli::ansi_strip`, 48

`cli::is_utf8_output`, 48

`collapse_sparse_rows`, 9

`compute_lisi`, 10

`do.call`, 7

`download`, 11

`fast_row_vars`, 12

`figlet`, 4, 13

`figlet_font`, 14

`filter_nonzero_variance_features`, 14

`get_namespace_fun`, 15

`get_verbose`, 15

`invoke_fun`, 16

`is_apple_silicon`, 17

`is_linux`, 17

`is_osx`, 17

`is_outlier`, 18

`is_windows`, 19

`log_message`, 19

`log_message()`, 32

`matrix_process`, 26

`matrix_to_table`, 27, 47

`max_depth`, 29

`maximump`, 28

`meanp`, 29

`minimump`, 30

`normalization`, 30

`pak::pkg_install`, 7

`pak::pkg_install()`, 7

`parallelize_fun`, 31

`parent.frame`, 20

`parse_inline_expressions`, 33

`pearson_correlation`, 34

`print.thisutils_logo`, 35

`r_square`, 42

`remove_r`, 35

`remove_space`, 36

`run_biocneighbors_knn`, 37

`run_dense_topk`, 38

`run_dense_topk()`, 39

`run_sparse_stored_topk_by_column`, 39

`run_sparse_topk`, 39

`run_sparse_topk()`, 40, 41

`run_sparse_topk_by_column`, 40

`run_sparse_topk_stored`, 41

`run_sparse_topk_stored()`, 39, 40

`simulate_sparse_matrix`, 42

`sparse_cor`, 43

`split_indices`, 45

`stats::cor()`, 44

`sump`, 46

`table_to_matrix`, 28, 47

`table_to_matrix()`, 27

`thisutils(thisutils-package)`, 3

`thisutils-package`, 3

`thisutils_logo`, 48

`try_get`, 49

`unnest_fun`, [50](#)
`utils::askYesNo`, [20](#)
`utils::download.file`, [12](#)

`votep`, [51](#)

`wilkinsonp`, [51](#)