

Package: easybio (via r-universe)

October 7, 2026

Title Comprehensive Single-Cell Annotation and Transcriptomic Analysis Toolkit

Version 1.3.0

Description Provides a comprehensive toolkit for single-cell annotation with the 'CellMarker 3.0' database <<https://bio-bigdata.hrbmu.edu.cn/CellMarker/>>. Streamlines biological label assignment in single-cell RNA-seq data and facilitates transcriptomic analysis, including preparation of TCGA<<https://portal.gdc.cancer.gov/>> and GEO<<https://www.ncbi.nlm.nih.gov/geo/>> datasets, differential expression analysis and visualization of enrichment analysis results. Additional utility functions support various bioinformatics workflows. See Wei Cui (2024) <[doi:10.1101/2024.09.14.609619](https://doi.org/10.1101/2024.09.14.609619)> for more details.

URL <https://github.com/person-c/easybio>

BugReports <https://github.com/person-c/easybio/issues>

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports data.table (>= 1.15.0), checkmate, ggplot2, httr2, lifecycle, R6, xml2

Depends R (>= 4.1.0)

LazyData true

Suggests litedown, patchwork, ggrepel, Seurat, limma, GEOquery, fgsea, BiocParallel, edgeR, testthat (>= 3.0.0),

biocViews limma, GEOquery, edgeR, fgsea

Language en-US

VignetteBuilder litedown

Config/testthat/edition 3

NeedsCompilation no

Author Wei Cui [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0004-8315-5899>)
Maintainer Wei Cui <m2c.w@outlook.com>
Repository <https://cran.r-universe.dev>
Date/Publication 2026-10-07 16:41:31 UTC
RemoteUrl <https://github.com/cran/easybio>
RemoteRef HEAD
RemoteSha 6010abdbbc7ed3a83683413997d24a3794ecda9b

Contents

Artist	3
available_ele	9
available_tissue_class	10
available_tissue_type	11
check_marker	12
CHOL_DEGs	13
dge_list	13
dgeList	14
dprocess_dgeList	15
fininsert	15
get_attr	16
get_marker	17
group_stat	18
group_stat_i	19
groupStat	19
groupStatI	20
limma_fit	20
limmaFit	21
list2dt	21
list2graph	22
match_ref	22
matchCellMarker2	25
pbmc.markers	26
plot_enrichment	26
plot_gsea	27
plot_marker_distribution	27
plot_ora	28
plot_possible_cell	29
plot_rank	30
plot_seurat_dot	30
plot_volcano	31
plotEnrichment2	32
plotGSEA	33
plotMarkerDistribution	33
plotORA	34

plotPossibleCell	34
plotRank	35
plotSeuratDot	35
plotVolcano	36
prepare_geo	36
prepare_tcga	37
process_dge_list	38
set_colnames	38
set_rownames	39
set_savedir	39
setcolnames	40
setrownames	40
setSavedir	41
split_matrix	41
suggest_best_match	42
theme_publication	43
tune_parameters	44
tuneParameters	45
uniprot_id_map	45
work_in	46
workIn	46
Index	48

Artist	<i>Visualization Artist for Custom Plots</i>
--------	--

Description

The Artist class offers a suite of methods designed to create a variety of plots using ggplot2 for data exploration. All methods log their calls and results, allowing you to review all outcomes later via the `get_all_results()` method.

Each `plot_*` method displays the generating command as the plot title. When a `plot_*` method immediately follows a `test_*` method, the test result (p-value) is automatically added as the subtitle.

All methods return `invisible(self)`, enabling fluent method chaining.

Value

The R6 class [Artist](#).

Public fields

- `data` Stores the dataset used for plotting.
- `command` recode the command.
- `result` record the plot.

Methods

Public methods:

- `Artist$new()`
- `Artist$get_all_result()`
- `Artist$test_wilcox()`
- `Artist$test_t()`
- `Artist$plot_scatter()`
- `Artist$plot_box()`
- `Artist$plot_dumbbell()`
- `Artist$plot_bubble()`
- `Artist$plot_barchart_divergence()`
- `Artist$plot_lollipop()`
- `Artist$plot_contour()`
- `Artist$plot_scatter_ellipses()`
- `Artist$plot_donut()`
- `Artist$plot_pie()`
- `Artist$clone()`

Method `new()`: Initializes the Artist class with an optional dataset.

Usage:

```
Artist$new(data = NULL)
```

Arguments:

`data` A data frame containing the dataset to be used for plotting. Default is NULL.

Returns: An instance of the Artist class.

Method `get_all_result()`: Get all history result

Usage:

```
Artist$get_all_result()
```

Returns: a data.table object

Method `test_wilcox()`: Conduct wilcox.test

Usage:

```
Artist$test_wilcox(formula, data = self$data, ...)
```

Arguments:

`formula` `wilcox.test()` formula arguments

`data` A data frame containing the data to be plotted. Default is `self$data`.

`...` Additional aesthetic mappings passed to `wilcox.test()`.

Returns: The Artist object invisibly.

Method `test_t()`: Conduct t.test

Usage:

```
Artist$test_t(formula, data = self$data, ...)
```

Arguments:

formula `t.test()` formula arguments

data A data frame containing the data to be plotted. Default is `self$data`.

... Additional aesthetic mappings passed to `t.test()`.

Returns: The Artist object invisibly.

Method `plot_scatter()`: Creates a scatter plot.

Usage:

```
Artist$plot_scatter(
  data = self$data,
  fun = function(x) x,
  x,
  y,
  ...,
  add = private$test_htest()
)
```

Arguments:

data A data frame containing the data to be plotted. Default is `self$data`.

fun function to process the `self$data`.

x The column name for the x-axis.

y The column name for the y-axis.

... Additional aesthetic mappings passed to `aes()`.

add whether to add the test result as subtitle.

Returns: The Artist object invisibly.

Method `plot_box()`: Creates a box plot.

Usage:

```
Artist$plot_box(
  data = self$data,
  fun = function(x) x,
  x,
  ...,
  add = private$test_htest()
)
```

Arguments:

data A data frame or tibble containing the data to be plotted. Default is `self$data`.

fun function to process the `self$data`.

x The column name for the x-axis.

... Additional aesthetic mappings passed to `aes()`.

add whether to add the test result as subtitle.

Returns: The Artist object invisibly.

Method `plot_dumbbell()`: Creates a dumbbell plot.

This method generates a dumbbell plot using the provided data, mapping the specified columns to the x-axis, y-axis, and color aesthetic.

Usage:

```
Artist$plot_dumbbell(  
  data = self$data,  
  x,  
  y,  
  col,  
  add = private$test_h(),  
  ...  
)
```

Arguments:

data A data frame containing the data to be plotted.

x The column in data to map to the x-axis.

y The column in data to map to the y-axis.

col The column in data to map to the color aesthetic.

add whether to add the test result as subtitle.

... Additional aesthetic mappings or other arguments passed to ggplot.

Returns: The Artist object invisibly.

Method `plot_bubble()`: Creates a bubble plot.

This method generates a bubble plot where points are mapped to the x and y axes, with their size and color representing additional variables.

Usage:

```
Artist$plot_bubble(  
  data = self$data,  
  x,  
  y,  
  size,  
  col,  
  add = private$test_h(),  
  ...  
)
```

Arguments:

data A data frame containing the data to be plotted.

x The column in data to map to the x-axis.

y The column in data to map to the y-axis.

size The column in data to map to the size of the points.

col The column in data to map to the color of the points.

add whether to add the test result as subtitle.

... Additional aesthetic mappings or other arguments passed to ggplot.

Returns: The Artist object invisibly.

Method `plot_barchart_divergence()`: Creates a divergence bar chart.

This method generates a divergence bar chart where bars are colored based on their positive or negative value.

Usage:

```
Artist$plot_barchart_divergence(  
  data = self$data,  
  group,  
  y,  
  add = private$sis_htest(),  
  ...  
)
```

Arguments:

data A data frame containing the data to be plotted.

group The column in data representing the grouping variable.

y The column in data to map to the y-axis.

add whether to add the test result as subtitle.

... Additional aesthetic mappings or other arguments passed to ggplot.

Returns: The Artist object invisibly.

Method `plot_lollipop()`: Creates a lollipop plot.

This method generates a lollipop plot, where points are connected to a baseline by vertical segments.

Usage:

```
Artist$plot_lollipop(data = self$data, x, y, add = private$sis_htest(), ...)
```

Arguments:

data A data frame containing the data to be plotted.

x The column in data to map to the x-axis.

y The column in data to map to the y-axis.

add whether to add the test result as subtitle.

... Additional aesthetic mappings or other arguments passed to ggplot.

Returns: The Artist object invisibly.

Method `plot_contour()`: Creates a contour plot.

This method generates a contour plot that includes filled and outlined density contours, with data points overlaid.

Usage:

```
Artist$plot_contour(data = self$data, x, y, add = private$sis_htest(), ...)
```

Arguments:

data A data frame containing the data to be plotted.

x The column in data to map to the x-axis.

y The column in data to map to the y-axis.

add whether to add the test result as subtitle.

... Additional aesthetic mappings or other arguments passed to ggplot.

Returns: The Artist object invisibly.

Method `plot_scatter_ellipses()`: Creates a scatter plot with ellipses.

This method generates a scatter plot where data points are colored by group, with ellipses representing the confidence intervals for each group.

Usage:

```
Artist$plot_scatter_ellipses(
  data = self$data,
  x,
  y,
  col,
  add = private$sis_htest(),
  ...
)
```

Arguments:

`data` A data frame containing the data to be plotted.

`x` The column in `data` to map to the x-axis.

`y` The column in `data` to map to the y-axis.

`col` The column in `data` to map to the color aesthetic.

`add` whether to add the test result as subtitle.

`...` Additional aesthetic mappings or other arguments passed to `ggplot`.

Returns: The Artist object invisibly.

Method `plot_donut()`: Creates a donut plot.

This method generates a donut plot, which is a variation of a pie chart with a hole in the center. The sections of the donut represent the proportion of categories in the data.

Usage:

```
Artist$plot_donut(data = self$data, x, y, fill, add = private$sis_htest(), ...)
```

Arguments:

`data` A data frame containing the data to be plotted.

`x` The column in `data` to map to the x-axis.

`y` The column in `data` to map to the y-axis.

`fill` The column in `data` to map to the fill color of the sections.

`add` whether to add the test result as subtitle.

`...` Additional aesthetic mappings or other arguments passed to `ggplot`.

Returns: The Artist object invisibly.

Method `plot_pie()`: Creates a pie chart.

This method generates a pie chart where sections represent the proportion of categories in the data.

Usage:

```
Artist$plot_pie(data = self$data, y, fill, add = private$sis_htest(), ...)
```

Arguments:

`data` A data frame containing the data to be plotted.

`y` The column in `data` to map to the y-axis.

fill The column in data to map to the fill color of the sections.
add whether to add the test result as subtitle.
 ... Additional aesthetic mappings or other arguments passed to ggplot.
Returns: The Artist object invisibly.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`Artist$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

Examples

```
library(data.table)
air <- subset(airquality, Month %in% c(5, 6))
setDT(air)
cying <- Artist$new(data = air)
cying$plot_scatter(x = Wind, y = Temp)
cying$test_wilcox(formula = Ozone ~ Month)
cying$plot_scatter(x = Wind, y = Temp)
```

available_ele

Extract Unique Elements from a Column with Optional Filtering

Description

Retrieves the unique, non-missing values from a specified column of a data frame. An optional expression can be provided to filter the rows of the data frame before extracting the values.

Usage

```
available_ele(data, col_name, subset)
```

Arguments

data	A data frame from which to extract values.
col_name	A single string specifying the name of the target column.
subset	An optional logical expression used to subset the data frame. This expression is evaluated in the context of the data, so columns can be referred to by their names directly (e.g., <code>Sepal.Length > 5</code>).

Value

A vector containing the unique, non-NA values from the specified column after the optional filtering has been applied.

Examples

```
# Example 1: Get all unique species from the iris dataset
available_ele(iris, "Species")

# Example 2: Get unique species for flowers with Sepal.Length > 7
available_ele(iris, "Species", subset = Sepal.Length > 7)

# Example 3: Get unique carb values for cars with 6 cylinders
available_ele(mtcars, "carb", subset = cyl == 6)
```

`available_tissue_class`*Retrieve Available Tissue Classes for a Given Species*

Description

This function extracts and returns a unique list of available tissue classes from the CellMarker 3.0 database for a specified species.

Usage

```
available_tissue_class(spc)
```

Arguments

`spc` A character string specifying the species (e.g., "Human" or "Mouse").

Value

A character vector of unique tissue classes available for the given species. If no tissue classes are found, an empty vector is returned.

See Also

[available_tissue_type](#), [get_marker](#)

Examples

```
# Get all tissue classes for Human
available_tissue_class("Human")
```

available_tissue_type *Retrieve Available Tissue Types for a Given Species*

Description

This function extracts and returns a unique list of available tissue types from the CellMarker 3.0 database for a specified species, optionally restricted to some tissue classes.

Usage

```
available_tissue_type(spc, tissue_class = available_tissue_class(spc))
```

Arguments

spc	A character string specifying the species (e.g., "Human" or "Mouse").
tissue_class	A character vector of tissue classes to look in, default <code>available_tissue_class(spc)</code> , i.e. every class the species has. A class the species does not have is an error rather than an empty result.

Details

`tissue_class` and `tissue_type` are two labels recorded for each database entry, not a hierarchy: one tissue type can be listed under several classes. `match_ref()` combines the two with AND, so a pair that never co-occurs selects nothing; this is how to see what a class actually has before passing both.

Value

A character vector of unique tissue types available for the given species. If no tissue types are found, an empty vector is returned.

See Also

[available_tissue_class](#), [match_ref](#)

Examples

```
# Get all tissue types for Human
available_tissue_type("Human")

# The tissue types recorded under one class
available_tissue_type("Human", tissue_class = "Blood")
```

check_marker

Verify and Explore Cell Type Annotations

Description

A post-analysis function that helps to verify and explore the automated cell type annotations generated by `match_ref`. It retrieves marker genes for the top-matching cell types of specified clusters, allowing for deeper inspection of the annotation results.

Usage

```
check_marker(marker, cl = c(), top_cell_n = 2, cis = FALSE)
```

Arguments

<code>marker</code>	A <code>data.table</code> object, which is the result of a call to <code>match_ref()</code> . This object must contain the attributes set by <code>match_ref</code> for the function to work correctly.
<code>cl</code>	A numeric or character vector specifying the cluster IDs to be inspected.
<code>top_cell_n</code>	An integer. For each cluster in <code>cl</code> , the function will retrieve markers for the top <code>top_cell_n</code> cell type annotations. Defaults to 2.
<code>cis</code>	A logical value that switches the function's mode. See Details. Defaults to <code>FALSE</code> .

Details

The function provides two distinct modes for marker retrieval, controlled by the `cis` parameter. This allows the user to answer two different, important questions:

- **`cis = FALSE` (Default): "Is the annotation correct?"** This mode answers the question by fetching the *canonical* markers for the annotated cell type from the reference database (via `get_marker`). It automatically uses the same filtering criteria (species, tissue, etc.) that were used in the original `match_ref` call, ensuring consistency.
- **`cis = TRUE`: "Why was this annotation made?"** This mode answers the question by extracting the *local* markers from the user's own data (i.e., the differentially expressed genes from the marker input) that led to the annotation. This helps understand the evidence behind the match.

Value

A named list. Each name in the list is a cell type, and each element is a character vector of its corresponding marker genes.

See Also

[match_ref](#) to generate the input for this function. [get_marker](#) which is used internally when `cis = FALSE`. [plot_seurat_dot](#) to visualize the results.

Examples

```
## Not run:
library(easybio)
data(pbmc.markers)

# Step 1: Generate cell type annotations
matched_cells <- match_ref(pbmc.markers, n = 50, spc = "Human")

# Step 2: Verify the annotation for cluster 0.
# Let's check the top annotation (top_cell_n = 1).

# Question 1: "Is cluster 0 really a CD4-positive T cell?
# Let's see the canonical markers for it."
# Note: We don't need to pass 'spc' here; it's retrieved from matched_cells.
reference_markers <- check_marker(matched_cells, cl = 0, top_cell_n = 1)
print(reference_markers)
# Now you would typically use these markers in Seurat::DotPlot() or Seurat::FeaturePlot()

# Question 2: "Which of my genes made the algorithm think cluster 0
# is a CD4-positive T cell?"
local_markers <- check_marker(matched_cells, cl = 0, top_cell_n = 1, cis = TRUE)
print(local_markers)

## End(Not run)
```

CHOL_DEGs

Example DEGs data from Limma-Voom workflow for TCGA-CHOL project

Description

The data were obtained by the limma-voom workflow

dge_list

Construct a DGEList Object

Description

This function creates a DGEList object from a count matrix, sample information, and feature information. It is designed to facilitate the analysis of differential gene expression using the edgeR package.

Usage

```
dge_list(count, sample_info, feature_info)
```

Arguments

count	A numeric matrix where rows represent features (e.g., genes) and columns represent samples. Row names should correspond to feature identifiers, and column names should correspond to sample identifiers.
sample_info	A data frame containing information about the samples. The number of rows should match the number of columns in the count matrix.
feature_info	A data frame containing information about the features. The number of rows should match the number of rows in the count matrix.

Value

A DGEList object as defined by the edgeR package, which includes the count data, sample information, and feature information.

dgeList	<i>Construct a DGEList Object (Deprecated)</i>
---------	--

Description**[Deprecated]**

dgeList() was renamed to [dge_list\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
dgeList(...)
```

Arguments

... Arguments passed on to [dge_list\(\)](#).

Value

See [dge_list\(\)](#).

dprocess_dgeList	<i>Filter and Normalize DGEList Data (Deprecated)</i>
------------------	---

Description**[Deprecated]**

dprocess_dgeList() was renamed to [process_dge_list\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
dprocess_dgeList(...)
```

Arguments

... Arguments passed on to [process_dge_list\(\)](#).

Value

See [process_dge_list\(\)](#).

finset	<i>Create a Vector from an Index-to-Label Map</i>
--------	---

Description

Constructs a character vector by mapping labels to specified 0-based numeric indices. This is a utility function often used in single-cell analysis to assign cell type annotations to cluster IDs.

Usage

```
finset(
  x = list(c(0, 1, 3) ~ "Neutrophil", c(2, 4, 8) ~ "Macrophage"),
  len = integer(),
  setname = TRUE,
  na = "Unknown"
)
```

Arguments

x	The mapping of indices to labels. This can be provided in two formats: • A list of formulas, e.g., <code>list(c(0, 1) ~ "LabelA", 2 ~ "LabelB")</code>. • An expression object, e.g., <code>expression(c(0, 1) == "LabelA", 2 == "LabelB")</code>.
len	An optional integer specifying the minimum length of the output vector. If the highest index in x is greater than len, the vector will be automatically extended.

setname	A logical value. If TRUE (the default), the elements of the output vector are named with their corresponding 0-based index (e.g., "0", "1", "2", ...).
na	The character value used to fill positions that are not specified in the mapping. Defaults to "Unknown".

Value

A character vector with the specified labels at the given positions. The vector is named with 0-based indices if setname is TRUE.

Examples

```
# --- Example 1: Using the default formula list format ---
# This is the recommended and default usage.
mapping_formula <- list(
  c(0, 1, 3) ~ "Neutrophil",
  c(2, 4, 8) ~ "Macrophage"
)
fininsert(mapping_formula)

# --- Example 2: Using the expression format for backward compatibility ---
mapping_expr <- expression(
  c(0, 1, 3) == "Neutrophil",
  c(2, 4, 8) == "Macrophage"
)
fininsert(mapping_expr, len = 10, na = "Unassigned")
```

get_attr

Retrieve Attributes from an R Object

Description

This function extracts a specified attribute from an R object.

Usage

```
get_attr(x, attr_name)
```

Arguments

x	An R object that has attributes.
attr_name	The name of the attribute to retrieve.

Value

The value of the attribute with the given name.

`get_marker`*Retrieve Markers for Specific Cells from cellMarker3*

Description

This function extracts a list of markers for one or more cell types from the cellMarker3 dataset. It allows filtering by species, cell type, the number of markers to retrieve, and a minimum count threshold for marker occurrences.

Usage

```
get_marker(  
  spc,  
  cell = character(),  
  tissue_class = available_tissue_class(spc),  
  tissue_type = available_tissue_type(spc),  
  number = 5,  
  min_count = 1  
)
```

Arguments

<code>spc</code>	A character string specifying the species, which can be either 'Human' or 'Mouse'.
<code>cell</code>	A character vector of cell types for which to retrieve markers.
<code>tissue_class</code>	A character specifying the tissue classes, default <code>available_tissue_class(spc)</code> .
<code>tissue_type</code>	A character specifying the tissue types, default <code>available_tissue_type(spc)</code> .
<code>number</code>	An integer specifying the number of top markers to return for each cell type.
<code>min_count</code>	An integer representing the minimum number of times a marker must have been reported to be included in the results.

Details

Cell types absent from the database are skipped. For unknown names, up to three alternative cell types are suggested via [suggest_best_match](#), covering typos (fuzzy matching) as well as partial names.

Value

A named list where each name corresponds to a cell type and each element is a vector of marker names.

See Also

[suggest_best_match](#), [match_ref](#)

Examples

```
# Example usage:
# Retrieve the top 5 markers for 'Macrophage' and 'Monocyte' cell types in humans,
# with a minimum count of 1.
library(easybio)
markers <- get_marker(spc = "Human", cell = c("Macrophage", "Monocyte"))
print(markers)
# Example with a typo in cell name
markers_typo <- get_marker(spc = "Human", cell = c("Macrophae", "Monocyte"))
```

group_stat

Perform Summary Analysis by Group Using Regular Expressions

Description

This function applies a specified function to each group defined by a regular expression pattern applied to the names of a data object. It is useful for summarizing data when groups are defined by a pattern in the names rather than a specific column or index.

Usage

```
group_stat(f, x, xname = colnames(x), patterns)
```

Arguments

f	A function that takes a single argument and returns a summary of the data.
x	A data frame or matrix containing the data to be summarized.
xname	A character vector containing the names of the variables in x.
patterns	A list of regular expressions that define the groups.

Value

A list containing the summary statistics for each group.

Examples

```
library(easybio)
group_stat(f = \(x) x + 1, x = mtcars, patterns = list("mp", "t"))
```

group_stat_i

*Perform Summary Analysis by Group Using an column Index***Description**

This function applies a specified function to each group defined by an column index, and returns a summary of the results. It is useful for summarizing data by group when the groups are defined by an column index.

Usage

```
group_stat_i(f, x, idx)
```

Arguments

f	A function that takes a single argument and returns a summary of the data.
x	A data frame or matrix containing the data to be summarized.
idx	A list of indices or group names that define the column groups.

Value

A list containing the summary statistics for each group.

Examples

```
library(easybio)
group_stat_i(f = \(x) x + 1, x = mtcars, idx = list(c(1, 10), 2))
```

groupStat

*Summarize Data by Group Using Regular Expressions (Deprecated)***Description****[Deprecated]**

groupStat() was renamed to [group_stat\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
groupStat(...)
```

Arguments

...	Arguments passed on to group_stat() .
-----	---

Value

See [group_stat\(\)](#).

groupStatI	<i>Summarize Data by Group Using an Index (Deprecated)</i>
------------	--

Description**[Deprecated]**

groupStatI() was renamed to [group_stat_i\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
groupStatI(...)
```

Arguments

... Arguments passed on to [group_stat_i\(\)](#).

Value

See [group_stat_i\(\)](#).

limma_fit	<i>Fit a Linear Model for RNA-seq data using limma</i>
-----------	--

Description

This function fits a linear model to processed DGEList data using the limma package. It defines contrasts between groups and performs differential expression analysis.

Usage

```
limma_fit(x, group_column)
```

Arguments

x	A processed DGEList object containing normalized count data.
group_column	The name of the column in x\$samples that contains the grouping information for the samples.

Details

limma::makeContrasts() only accepts syntactically valid names, so the group labels are normalised with [make.names\(\)](#) (the design's coefficient names with them). Labels that are not valid R names, such as the TCGA sample types "Primary Tumor" and "Solid Tissue Normal", would otherwise fail: building the contrasts as expressions from those labels either does not parse or silently turns "Non-tumor" - "Tumor" into a subtraction of three symbols. The contrasts keep the original labels in their names, e.g. "Primary TumorvsSolid Tissue Normal".

Value

An eBayes object containing the fitted linear model and results of the differential expression analysis. The design and contrast matrices are attached as the design and contrast attributes.

limmaFit

Fit a Linear Model for RNA-seq Data (Deprecated)

Description**[Deprecated]**

limmaFit() was renamed to [limma_fit\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
limmaFit(...)
```

Arguments

... Arguments passed on to [limma_fit\(\)](#).

Value

See [limma_fit\(\)](#).

list2dt

Convert a List with Vector Values to a Long Data.table

Description

This function converts a named list with vector values in each element to a long data.table. The list is first flattened into a single vector, and then the data.table is created with two columns: one for the name of the original list element and another for the value.

Usage

```
list2dt(x, col_names = c("name", "value"))
```

Arguments

x A named list where each element contains a vector of values.
col_names The colnames of the returned result.

Value

A long data.table with two columns: 'name' and 'value'.

Examples

```
library(easybio)
list2dt(list(a = c(1, 1), b = c(2, 2)))
```

list2graph

Convert a Named List into a Graph Based on Overlap

Description

This function creates a graph from a named list, where the edges are determined by the overlap between the elements of the list. Each node in the graph represents an element of the list, and the weight of the edge between two nodes is the number of overlapping elements between the two corresponding lists.

Usage

```
list2graph(nodes)
```

Arguments

nodes A named list where each element is a vector.

Value

A data.table representing the graph, with columns for the node names (node_1 and node_2) and the weight of the edge (interWeight).

match_ref

Annotate Clusters by Matching Markers with the CellMarker 3.0 Database

Description

This function takes cluster-specific markers, typically from `Seurat::FindAllMarkers`, and annotates each cluster with potential cell types by matching these markers against a reference database. It first filters and selects the top n marker genes for each cluster based on specified thresholds and then compares them to the reference database to find the most likely cell type annotations.

Usage

```
match_ref(
  marker,
  n,
  avg_log2fc_threshold = 0,
  p_val_adj_threshold = 0.05,
  min_pct = NULL,
  spc,
  tissue_class = available_tissue_class(spc),
  tissue_type = available_tissue_type(spc),
  ref = NULL
)
```

Arguments

marker	A data.frame or data.table of markers, usually the output of <code>Seurat::FindAllMarkers</code> . It must contain columns for cluster, gene, avg_log2FC, and p_val_adj. If a pct.1 column is present, the detection rate of each matching marker is reported in the pct_with column of the result.
n	An integer specifying the number of top marker genes to use from each cluster for matching. Genes are ranked by avg_log2FC after filtering.
avg_log2fc_threshold	A numeric value setting the minimum average log2 fold change for a marker to be considered. Defaults to 0.
p_val_adj_threshold	A numeric value setting the maximum adjusted p-value for a marker to be considered. Defaults to 0.05.
min_pct	An optional numeric value between 0 and 1. When given, markers whose detection rate in the cluster they were found for (pct.1) is below min_pct are dropped before matching, so that an annotation cannot rest on genes that are barely detected. NA detection rates are dropped as well. Defaults to NULL (no filtering), which keeps the behaviour of earlier versions. It is ignored, with a message, if marker has no pct.1 column. Note that <code>Seurat::FindAllMarkers(min.pct =)</code> cannot replace this: it gates which genes are tested in either population, so a gene detected in a few cells of the cluster can still end up as a positive marker.
spc	A character string specifying the species, either "Human" or "Mouse". This is used to filter the cellMarker3 database. This parameter is ignored if a custom ref is provided.
tissue_class	A character vector of tissue classes to include from the cellMarker3 database. Defaults to all available tissue classes for the specified species. This parameter is ignored if a custom ref is provided. See <code>available_tissue_class()</code> .
tissue_type	A character vector of tissue types to include from the cellMarker3 database. Defaults to all available tissue types for the specified species. This parameter is ignored if a custom ref is provided. See <code>available_tissue_type()</code> .
ref	An optional long data.frame which must contain 'cell_name' and 'marker' columns to be used as the reference for marker matching. If NULL (the default),

the function uses the built-in cellMarker3 dataset. When a custom ref is provided, the `spc`, `tissue_class`, and `tissue_type` parameters are ignored for the matching process itself, but their original values are saved for provenance.

Details

`tissue_class` and `tissue_type` are the two labels the database records for the sample each entry comes from, and they are combined with AND. A class and a type that never occur together therefore select no reference entry, which warns and returns no candidate; [available_tissue_type\(\)](#) lists the types a class actually has.

Value

A `data.table` where each row represents a potential cell type match for a cluster. The table is keyed by `cluster` and includes columns for `cluster`, `cell_name`, `uniqueN` (number of unique matching markers), `N` (total matches), `ordered_symbol` (matching genes, ordered by frequency), `orderN` (their frequencies), and `pct_with` (the `pct.1` detection rate of each matching gene, aligned with `ordered_symbol`, NA when the input has no `pct.1` column).

Within each cluster, rows are ordered by decreasing `uniqueN`, then by decreasing `N`, so the first row of a cluster is its top candidate.

The returned object also contains important attributes for downstream analysis:

<code>ref</code>	The reference data (either from cellMarker3 or the custom ref) used for the annotation.
<code>is_custom_ref</code>	A logical flag indicating if a custom ref was used.
<code>filter_args</code>	A list containing the filtering parameters used during the annotation, which is essential for the <code>check_marker</code> function.

See Also

[check_marker](#), [plot_possible_cell](#), [available_tissue_class](#), [available_tissue_type](#)

Examples

```
## Not run:
library(easybio)
data(pbmc.markers)

# Basic usage: Annotate clusters using the top 50 markers per cluster
matched_cells <- match_ref(pbmc.markers, n = 50, spc = "Human")
print(matched_cells)

# To see the top annotation for each cluster
top_matches <- matched_cells[, .SD[1], by = cluster]
print(top_matches)

# Advanced usage: Stricter filtering and focus on specific tissues
matched_cells_strict <- match_ref(
  pbmc.markers,
  n = 30,
```

```

    spc = "Human",
    avg_log2fc_threshold = 0.5,
    p_val_adj_threshold = 0.01,
    tissue_type = c("Blood", "Bone marrow")
  )
  print(matched_cells_strict)

# --- Example with a custom reference ---
# Create a custom reference as a named list.
custom_ref_list <- list(
  "T-cell" = c("CD3D", "CD3E"),
  "B-cell" = c("CD79A", "MS4A1"),
  "Myeloid" = "LYZ"
)

# Convert the list to a long data.frame compatible with the 'ref' parameter.
custom_ref_df <- list2dt(custom_ref_list, col_names = c("cell_name", "marker"))

# Run annotation using the custom reference.
# When 'ref' is provided, the internal cellMarker3 database and its filters
# ('spc', 'tissue_class', 'tissue_type') are ignored for matching.
matched_custom <- match_ref(
  pbmc.markers,
  n = 50,
  ref = custom_ref_df
)
print(matched_custom)

## End(Not run)

```

matchCellMarker2

Annotate Clusters by Matching Markers (Deprecated)

Description

[Deprecated]

matchCellMarker2() was renamed to [match_ref\(\)](#) because it supports custom reference datasets as well as the built-in CellMarker 3.0 database. It will be removed in version 1.4.0.

Usage

```
matchCellMarker2(marker, n, ...)
```

Arguments

marker	A data.frame or data.table of markers, usually the output of <code>Seurat::FindAllMarkers</code> . It must contain columns for cluster, gene, avg_log2FC, and p_val_adj. If a pct.1 column is present, the detection rate of each matching marker is reported in the pct_with column of the result.
--------	---

- n An integer specifying the number of top marker genes to use from each cluster for matching. Genes are ranked by avg_log2FC after filtering.
- ... Arguments passed on to `match_ref()`.

Value

See `match_ref()`.

pbmc.markers	<i>Example marker data from Seurat::FindAllMarkers()</i>
--------------	--

Description

The data were obtained by the seurat PBMC workflow. exact script for this data is available as `system.file("example-single-cell.R", package="easybio")`

plot_enrichment	<i>Plot Enrichment for a Specific Pathway in fgsea</i>
-----------------	--

Description

This function creates a plot of enrichment scores for a specified pathway. It provides a visual representation of the enrichment score (ES) along with the ranks and ticks indicating the GSEA walk length.

Usage

```
plot_enrichment(pathways, pwayname, stats, gsea_param = 1, ticks_size = 0.2)
```

Arguments

- pathways A list of pathways.
- pwayname The name of the pathway for which to plot enrichment.
- stats A rank vector obtained from the 'fgsea' package.
- gsea_param The GSEA walk length parameter. Default is 1.
- ticks_size The size of the tick marks. Default is 0.2.

Value

A ggplot object representing the enrichment plot.

plot_gseaVisualization of GSEA Result from `fgsea::fgsea()`

Description

The `plot_gsea` function visualizes the results of a GSEA (Gene Set Enrichment Analysis) using data from the `fgsea` package. It generates a composite plot that includes an enrichment plot and a ranked metric plot.

Usage

```
plot_gsea(fgsea_res, pathways, pwayname, stats, save = FALSE)
```

Arguments

<code>fgsea_res</code>	A data table containing the GSEA results from the <code>fgsea</code> package.
<code>pathways</code>	A list of all pathways used in the GSEA analysis.
<code>pwayname</code>	The name of the pathway to visualize.
<code>stats</code>	A numeric vector representing the ranked statistics.
<code>save</code>	A logical value indicating whether to save the plot as a PDF file. Default is <code>FALSE</code> .

Value

ggplot2 object.

plot_marker_distribution*Plot Distribution of a Marker Across Tissues and Cell Types*

Description

This function creates a dot plot displaying the distribution of a specified marker across different tissues and cell types, based on data from the CellMarker 3.0 database.

Usage

```
plot_marker_distribution(mkr = character())
```

Arguments

<code>mkr</code>	character, the name of the marker to be plotted.
------------------	--

Value

A ggplot2 object representing the distribution of the marker.

Examples

```
## Not run:
plot_marker_distribution("CD14")

## End(Not run)
```

plot_ora	<i>Visualization of ORA Test Results</i>
----------	--

Description

The plot_ora function visualizes the results of an ORA (Over-Representation Analysis) test. It generates a plot with customizable aesthetics for x, y, point size, and fill, with an option to flip the axes.

Usage

```
plot_ora(data, x, y, size, fill, flip = FALSE)
```

Arguments

data	A data frame containing the ORA results to be visualized.
x	The column in data to map to the x-axis.
y	The column in data to map to the y-axis.
size	The column in data to map to the size of the points.
fill	The column in data to map to the fill color of the bars or points. Use a constant value for a single category.
flip	A logical value indicating whether to flip the axes of the plot. Default is FALSE.

Value

ggplot2 object.

plot_possible_cell	<i>Plot Possible Cell Distribution Based on match_ref() Results</i>
--------------------	---

Description

This function creates a plot to visualize the distribution of possible cell types based on the results from the `match_ref()` function, utilizing data from the CellMarker 3.0 database.

Usage

```
plot_possible_cell(
  marker,
  min_unique_n = 2,
  value = c("N", "uniqueN", "pct"),
  min_pct = 0.25
)
```

Arguments

marker	data.table, the result from the <code>match_ref()</code> function.
min_unique_n	integer, the minimum number of unique marker genes that must be matched for a cell type to be included in the plot. Default is 2.
value	character, the measure shown for each candidate cell type: • "N" (the default) the total number of matched reference entries, • "uniqueN" the number of unique matching markers, i.e. the measure <code>match_ref()</code> ranks the candidates by, • "pct" the share of the matching markers that are actually detected in the cluster (see <code>min_pct</code>). This is the evidence check: a candidate can match a dozen markers and still rest on only two of them being expressed. The first two are drawn as points sized and coloured by the measure; "pct" is drawn as tiles filled by the share and labelled with <code>uniqueN</code>. The fill is scaled to the shares in the plot, so the palest tiles are the candidates resting on the fewest detected markers whatever the overall level is; the legend reports the range, and the exact shares are in <code>pct_wi</code>th.
min_pct	numeric between 0 and 1, the detection rate a matching marker has to reach to count as detected when <code>value = "pct"</code> . Defaults to 0.25 and is ignored for the other values. Note that this reads the detection rate recorded in the input of <code>match_ref()</code> (<code>pct.1</code> of <code>Seurat::FindAllMarkers()</code>), not the one <code>Seurat::DotPlot()</code> computes from the Seurat object.

Value

A ggplot2 object representing the distribution of possible cell types.

plot_rank	<i>Visualization of GSEA Rank Statistics</i>
-----------	--

Description

The `plot_rank` function visualizes the ranked statistics of a GSEA (Gene Set Enrichment Analysis) analysis. The function creates a plot where the x-axis represents the rank of each gene, and the y-axis shows the corresponding ranked list metric.

Usage

```
plot_rank(stats)
```

Arguments

`stats` A numeric vector containing the ranked statistics from a GSEA analysis.

Value

ggplot2 object

plot_seurat_dot	<i>Create a Dot Plot to Visualize Marker Gene Expression</i>
-----------------	--

Description

This function generates a `Seurat::DotPlot` to visualize the expression of specified marker genes across different cell clusters or groups. It is designed to work with a list of features, such as the output from the `check_marker` function.

Usage

```
plot_seurat_dot(features, srt, split = FALSE, ...)
```

Arguments

`features` A named list of character vectors. Each name in the list represents a cell type or category, and the corresponding character vector contains the marker genes to be plotted for that category. This is typically the output of `check_marker()`.

`srt` A Seurat object containing the single-cell expression data.

`split` Logical, if TRUE, generates separate dot plots for each cell type in `features`

`...` Additional arguments passed to `Seurat::DotPlot()`, such as `cols`, `dot.scale`, etc.

Value

A ggplot2 object representing the dot plot, which can be further customized.

See Also

[check_marker](#) to generate the features list.

Examples

```
## Not run:
library(easybio)
library(Seurat)
data(pbmc.markers)

# In a real scenario, 'srt' would be your fully processed Seurat object.
# For this example, we create a minimal Seurat object.
# The expression matrix should contain the marker genes for the plot to be meaningful.
marker_genes <- unique(pbmc.markers$gene)
counts <- matrix(
  abs(rnorm(length(marker_genes) * 50, mean = 1, sd = 2)),
  nrow = length(marker_genes),
  ncol = 50
)
rownames(counts) <- marker_genes
colnames(counts) <- paste0("cell_", 1:50)

srt <- Seurat::CreateSeuratObject(counts = counts)
srt$seurat_clusters <- sample(0:3, 50, replace = TRUE)
Idents(srt) <- "seurat_clusters"

# Step 1: Generate cell type annotations
matched_cells <- match_ref(pbmc.markers, n = 50, spc = "Human")

# Step 2: Get canonical markers for cluster 0's top annotation
reference_markers <- check_marker(matched_cells, cl = 0, top_cell_n = 1)

# Step 3: Plot the expression of these markers
if (!is.null(reference_markers) && length(reference_markers) > 0) {
  plot_seurat_dot(features = reference_markers, srt = srt)
}

## End(Not run)
```

plot_volcano

Plot Volcano Plot for Differentially Expressed Genes

Description

This function generates a volcano plot for differentially expressed genes (DEGs) using ggplot2. It allows for customization of the plot with different aesthetic parameters.

Usage

```
plot_volcano(data, data_text, x, y, color, label)
```

Arguments

data	A data frame containing the DEGs result.
data_text	An optional data frame of genes to label. When given, the labels are added to the same plot; x, y and label are mapped for it as well, so the columns must be named like the ones in data.
x	variable representing the aesthetic for the x-axis.
y	variable representing the aesthetic for the y-axis.
color	variable representing the column name for the color aesthetic.
label	variable representing the column name for the text label aesthetic.

Value

A ggplot object representing the volcano plot.

plotEnrichment2	<i>Plot Enrichment for a Pathway (Deprecated)</i>
-----------------	---

Description**[Deprecated]**

plotEnrichment2() was renamed to [plot_enrichment\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotEnrichment2(...)
```

Arguments

... Arguments passed on to [plot_enrichment\(\)](#).

Value

See [plot_enrichment\(\)](#).

`plotGSEA`*Visualize GSEA Results (Deprecated)*

Description**[Deprecated]**

`plotGSEA()` was renamed to `plot_gsea()` to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotGSEA(...)
```

Arguments

... Arguments passed on to `plot_gsea()`.

Value

See `plot_gsea()`.

`plotMarkerDistribution`*Plot Distribution of a Marker (Deprecated)*

Description**[Deprecated]**

`plotMarkerDistribution()` was renamed to `plot_marker_distribution()` to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotMarkerDistribution(...)
```

Arguments

... Arguments passed on to `plot_marker_distribution()`.

Value

See `plot_marker_distribution()`.

plotORA	<i>Visualize ORA Test Results (Deprecated)</i>
---------	--

Description**[Deprecated]**

plotORA() was renamed to [plot_ora\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotORA(...)
```

Arguments

... Arguments passed on to [plot_ora\(\)](#).

Value

See [plot_ora\(\)](#).

plotPossibleCell	<i>Plot Possible Cell Distribution (Deprecated)</i>
------------------	---

Description**[Deprecated]**

plotPossibleCell() was renamed to [plot_possible_cell\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotPossibleCell(...)
```

Arguments

... Arguments passed on to [plot_possible_cell\(\)](#).

Value

See [plot_possible_cell\(\)](#).

plotRank	Visualize GSEA Rank Statistics (Deprecated)
----------	---

Description**[Deprecated]**

plotRank() was renamed to [plot_rank\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotRank(...)
```

Arguments

... Arguments passed on to [plot_rank\(\)](#).

Value

See [plot_rank\(\)](#).

plotSeuratDot	Create a Dot Plot of Marker Gene Expression (Deprecated)
---------------	--

Description**[Deprecated]**

plotSeuratDot() was renamed to [plot_seurat_dot\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
plotSeuratDot(...)
```

Arguments

... Arguments passed on to [plot_seurat_dot\(\)](#).

Value

See [plot_seurat_dot\(\)](#).

plotVolcano	<i>Plot a Volcano Plot (Deprecated)</i>
Description	
[Deprecated]	
plotVolcano() was renamed to plot_volcano() to follow the snake_case naming style. It will be removed in version 1.4.0.	
Usage	
plotVolcano(...)	
Arguments	
...	Arguments passed on to plot_volcano() .
Value	
See plot_volcano() .	

prepare_geo	<i>Download and Process GEO Data</i>
Description	
This function downloads gene expression data from the Gene Expression Omnibus (GEO) database. It retrieves either the expression matrix or the supplementary tabular data if the expression data is not available. The function also allows for the conversion of probe identifiers to gene symbols and can combine multiple probes into a single symbol.	
Usage	
prepare_geo(geo, dir = ".", combine = TRUE, method = "max")	
Arguments	
geo	A character string specifying the GEO Series ID (e.g., "GSE12345").
dir	A character string specifying the directory where files should be downloaded. Default is the current working directory (".").
combine	A logical value indicating whether to combine multiple probes into a single gene symbol. Default is TRUE.
method	A character string specifying the method to use for combining probes into a single gene symbol. Options are "max" (take the maximum value) or "mean" (compute the average). Default is "max".

Value

A list containing:

data	A data frame of the expression matrix, or NULL if not available.
sample	A data frame of the sample metadata.
feature	A data frame of the feature metadata, or NULL if not available.
status	A character string indicating the data source: "expression_matrix", "supplementary_files", or "no_data".
supplementary	Only present when status is "supplementary_files". A named list of data.table objects parsed from supplementary files.

prepare_tcga

Prepare TCGA Data for Analysis

Description

This function prepares TCGA data for downstream analyses such as differential expression analysis with limma or survival analysis. It extracts and processes the necessary information from the TCGA data object, separating tumor and non-tumor samples.

Usage

```
prepare_tcga(data)
```

Arguments

data	A SummarizedExperiment object containing TCGA data, typically obtained from R package TCGABiolinks.
------	---

Details

The two expression tables used to be called exprCount and exprFpkm while the fields next to them were already sample_info and features_info. They are now expr_count and expr_fpkms; reading or assigning an old name still works but warns, and will stop working in 1.4.0.

Value

A list of two tables, all (every sample) and tumor (the tumor samples), each holding the expression matrix (expr_count, or expr_fpkms for the tumor samples), the features_info and the sample_info.

process_dge_list	<i>Filter Low-Expressed Genes and Normalize DGEList Data</i>
------------------	--

Description

This function filters out low-expressed genes from a `DGEList` object and normalizes the count data. It also provides diagnostic plots for raw and filtered data.

Usage

```
process_dge_list(x, group_column, min_count = 10)
```

Arguments

<code>x</code>	A <code>DGEList</code> object containing raw count data.
<code>group_column</code>	The name of the column in <code>x\$samples</code> that contains the grouping information for the samples.
<code>min_count</code>	The minimum number of counts required for a gene to be considered expressed. Genes with counts below this threshold in any group will be filtered out. Defaults to 10.

Details

At most the first 10 samples are shown in the density plots, and each of them keeps its colour, so repeated runs give the same plots.

Value

The function returns a `DGEList` object with low-expressed genes filtered out and normalization factors calculated.

set_colnames	<i>Rename Column Names of a Data Frame or Matrix</i>
--------------	--

Description

This function renames the column names of a data frame or matrix to the specified names.

Usage

```
set_colnames(object, nm)
```

Arguments

<code>object</code>	A data frame or matrix whose column names will be renamed.
<code>nm</code>	A character vector containing the new names for the columns.

Value

A data frame or matrix with the new column names.

set_rownames	<i>Rename Row Names of a Data Frame or Matrix</i>
--------------	---

Description

This function renames the row names of a data frame or matrix to the specified names.

Usage

```
set_rownames(object, nm)
```

Arguments

object	A data frame or matrix whose row names will be renamed.
nm	A character vector containing the new names for the rows.

Value

A data frame or matrix with the new row names.

set_savedir	<i>Set a Directory for Saving Files</i>
-------------	---

Description

This function sets a directory path for saving files, creating the directory if it does not already exist. The directory path is created with the given arguments, which are passed directly to `file.path()`.

Usage

```
set_savedir(...)
```

Arguments

...	Arguments to be passed to <code>file.path()</code> to construct the directory path.
-----	---

Value

The path to the newly created or existing directory.

setcolnames	<i>Rename Column Names (Deprecated)</i>
-------------	---

Description**[Deprecated]**

setcolnames() was renamed to [set_colnames\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
setcolnames(...)
```

Arguments

... Arguments passed on to [set_colnames\(\)](#).

Value

See [set_colnames\(\)](#).

setrownames	<i>Rename Row Names (Deprecated)</i>
-------------	--------------------------------------

Description**[Deprecated]**

setrownames() was renamed to [set_rownames\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
setrownames(...)
```

Arguments

... Arguments passed on to [set_rownames\(\)](#).

Value

See [set_rownames\(\)](#).

setSaveditr	<i>Set a Directory for Saving Files (Deprecated)</i>
-------------	--

Description**[Deprecated]**

setSaveditr() was renamed to [set_saveditr\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
setSaveditr(...)
```

Arguments

... Arguments passed on to [set_saveditr\(\)](#).

Value

See [set_saveditr\(\)](#).

split_matrix	<i>Split a Matrix into Smaller Sub-matrices by Column or Row</i>
--------------	--

Description

This function splits a matrix into multiple smaller matrices by column or row. It is useful for processing large matrices in chunks, such as when performing analysis on a single computer with limited memory.

Usage

```
split_matrix(matrix, chunk_size, column = TRUE)
```

Arguments

matrix	A numeric or logical matrix to be split.
chunk_size	The number of columns or rows to include in each smaller matrix.
column	Divided by column(default is TRUE)

Value

A list of smaller matrices, each with chunk_size columns or rows.

Examples

```
library(easybio)
split_matrix(mtcars, chunk_size = 2)
split_matrix(mtcars, chunk_size = 5, column = FALSE)
```

suggest_best_match	<i>Suggest Best Matches for a String from a Vector of Choices</i>
--------------------	---

Description

This function provides intelligent suggestions for a user's input string by finding the best matches from a given vector of choices. It follows a multi-layered approach:

1. Performs normalization (case-insensitivity, trimming whitespace).
2. Checks for an exact match first for maximum performance and accuracy.
3. If no exact match, it uses a combination of fuzzy string matching (Levenshtein distance via `adist`) to catch typos and partial/substring matching (`grep`) to handle incomplete input.
4. Ranks the potential matches and returns the best suggestion(s). Substring matches are ranked above fuzzy matches, and among themselves choices starting with the input come first, followed by the shortest choices. Fuzzy matches are ranked by increasing distance. Remaining ties keep the order of choices.

Usage

```
suggest_best_match(
  x,
  choices,
  n = 1,
  threshold = 2,
  ignore_case = TRUE,
  return_distance = FALSE
)
```

Arguments

<code>x</code>	A single character string; the user input to find matches for.
<code>choices</code>	A character vector of available, valid options.
<code>n</code>	An integer specifying the maximum number of suggestions to return. Defaults to 1.
<code>threshold</code>	An integer; the maximum Levenshtein distance to consider a choice a "close" match. A lower value is stricter. Defaults to 2.
<code>ignore_case</code>	A logical value. If TRUE, matching is case-insensitive. Defaults to TRUE.
<code>return_distance</code>	A logical value. If TRUE, the output is a data.frame containing the suggestions and their calculated distance/score. Defaults to FALSE.

Value

By default (`return_distance = FALSE`), returns a character vector of the best `n` suggestions. If no suitable match is found, returns `NA`. If `return_distance = TRUE`, returns a `data.frame` with columns `suggestion` and `distance`, or `NULL` if no match is found. The `distance` column holds the Levenshtein distance of a fuzzy match or the fixed score `0.5` of a substring match.

Examples

```
# --- Setup ---
cell_types <- c(
  "B cell", "T cell", "Macrophage", "Monocyte", "Neutrophil",
  "Natural Killer T-cell", "Dendritic cell"
)

# --- Usage ---
# 1. Exact match (after normalization)
suggest_best_match("t cell", cell_types)
#> [1] "T cell"

# 2. Typo correction (fuzzy match)
suggest_best_match("Macrophag", cell_types)
#> [1] "Macrophage"

# 3. Partial input (substring match)
suggest_best_match("Mono", cell_types)
#> [1] "Monocyte"

# 4. Requesting multiple suggestions
suggest_best_match("t", cell_types, n = 3)
#> [1] "T cell" "Monocyte" "Neutrophil"

# 5. No good match found
suggest_best_match("Erythrocyte", cell_types)
#> [1] NA

# 6. Returning suggestions with their distance score
suggest_best_match("t cel", cell_types, n = 3, return_distance = TRUE)
#>  suggestion distance
#> 1      T cell      0.5
#> 2      B cell      2.0
```

 theme_publication

Custom ggplot2 Theme for Academic Publications

Description

`theme_publication` creates a custom `ggplot2` theme designed for academic publications, ensuring clarity, readability, and a professional appearance. It is based on `theme_classic()` and includes additional refinements to axis lines, text, and other plot elements to meet the standards of high-quality academic figures.

Usage

```
theme_publication(base_size = 12, base_family = "sans")
```

Arguments

base_size numeric, the base font size. Default is 12.
base_family character, the base font family. Default is "sans".

Value

A ggplot2 theme object that can be applied to ggplot2 plots.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  theme_publication()
print(p)
```

tune_parameters	<i>Optimize Resolution and Gene Number Parameters for Cell Type Annotation</i>
-----------------	--

Description

This function tunes the resolution parameter in `Seurat::FindClusters()` and the number of top differential genes (`n`) to obtain different cell type annotation results. The function generates UMAP plots for each parameter combination, allowing for a comparison of how different settings affect the clustering and annotation.

Usage

```
tune_parameters(srt, resolution = numeric(), n = integer(), spc)
```

Arguments

srt Seurat object, the input data object to be analyzed.
resolution numeric vector, a vector of resolution values to be tested in `Seurat::FindClusters()`.
n integer vector, a vector of values indicating the number of top differential genes to be used for matching in `match_ref()`.
spc character, the species parameter for the `match_ref()` function, specifying the organism.

Value

A list of ggplot2 objects, each representing a UMAP plot generated with a different combination of resolution and `n` parameters.

tuneParameters	<i>Tune Parameters for Cell Type Annotation (Deprecated)</i>
----------------	--

Description

[Deprecated]

tuneParameters() was renamed to [tune_parameters\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
tuneParameters(...)
```

Arguments

... Arguments passed on to [tune_parameters\(\)](#).

Value

See [tune_parameters\(\)](#).

uniprot_id_map	<i>Map UniProt IDs to Other Identifiers</i>
----------------	---

Description

This function maps UniProt IDs to other identifiers using UniProt's ID mapping service. It sends a request to the UniProt API to perform the mapping and retrieves the results in a tabular format.

Usage

```
uniprot_id_map(..., timeout = 60, interval = 2)
```

Arguments

...	Parameters to be passed in the request body.
timeout	Numeric, how many seconds to wait for the mapping job to finish. Defaults to 60; large submissions can take considerably longer.
interval	Numeric, how many seconds to wait between status requests. Defaults to 2.

Value

A data.table containing the mapped identifiers. NULL, with a warning, if the job did not finish within timeout.

Examples

```
## Not run:
uniprot_id_map(
  ids = "P21802,P12345",
  from = "UniProtKB_AC-ID",
  to = "UniRef90"
)

## End(Not run)
```

work_in	<i>Perform Operations in a Specified Directory and Return to the Original Directory</i>
---------	---

Description

This function allows you to perform operations in a specified directory and then return to the original directory. It is useful when you need to work with files or directories that are located in a specific location, but you want to return to the original working directory after the operation is complete.

Usage

```
work_in(dir, expr)
```

Arguments

dir	The directory path in which to operate. If the directory does not exist, it will be created recursively.
expr	An R expression to be evaluated within the specified directory.

Value

The result of evaluating the expression within the specified directory.

workIn	<i>Perform Operations in a Directory (Deprecated)</i>
--------	---

Description**[Deprecated]**

workIn() was renamed to [work_in\(\)](#) to follow the snake_case naming style. It will be removed in version 1.4.0.

Usage

```
workIn(...)
```

Arguments

... Arguments passed on to [work_in\(\)](#).

Value

See [work_in\(\)](#).

Index

Artist, [3, 3](#)
available_ele, [9](#)
available_tissue_class, [10, 11, 24](#)
available_tissue_type, [10, 11, 24](#)
available_tissue_type(), [24](#)

check_marker, [12, 24, 31](#)
CHOL_DEGs, [13](#)

dge_list, [13](#)
dge_list(), [14](#)
dgeList, [14](#)
dprocess_dgeList, [15](#)

fgsea::fgsea(), [27](#)
fininsert, [15](#)

get_attr, [16](#)
get_marker, [10, 12, 17](#)
group_stat, [18](#)
group_stat(), [19](#)
group_stat_i, [19](#)
group_stat_i(), [20](#)
groupStat, [19](#)
groupStatI, [20](#)

limma_fit, [20](#)
limma_fit(), [21](#)
limmaFit, [21](#)
list2dt, [21](#)
list2graph, [22](#)

make.names(), [20](#)
match_ref, [11, 12, 17, 22](#)
match_ref(), [25, 26](#)
matchCellMarker2, [25](#)

pbmc.markers, [26](#)
plot_enrichment, [26](#)
plot_enrichment(), [32](#)
plot_gsea, [27](#)

plot_gsea(), [33](#)
plot_marker_distribution, [27](#)
plot_marker_distribution(), [33](#)
plot_ora, [28](#)
plot_ora(), [34](#)
plot_possible_cell, [24, 29](#)
plot_possible_cell(), [34](#)
plot_rank, [30](#)
plot_rank(), [35](#)
plot_seurat_dot, [12, 30](#)
plot_seurat_dot(), [35](#)
plot_volcano, [31](#)
plot_volcano(), [36](#)
plotEnrichment2, [32](#)
plotGSEA, [33](#)
plotMarkerDistribution, [33](#)
plotORA, [34](#)
plotPossibleCell, [34](#)
plotRank, [35](#)
plotSeuratDot, [35](#)
plotVolcano, [36](#)
prepare_geo, [36](#)
prepare_tcga, [37](#)
process_dge_list, [38](#)
process_dge_list(), [15](#)

set_colnames, [38](#)
set_colnames(), [40](#)
set_rownames, [39](#)
set_rownames(), [40](#)
set_savedir, [39](#)
set_savedir(), [41](#)
setcolnames, [40](#)
setrownames, [40](#)
setSavedir, [41](#)
split_matrix, [41](#)
suggest_best_match, [17, 42](#)

t.test(), [5](#)
theme_publication, [43](#)

tune_parameters, [44](#)
tune_parameters(), [45](#)
tuneParameters, [45](#)

uniprot_id_map, [45](#)

wilcox.test(), [4](#)
work_in, [46](#)
work_in(), [46](#), [47](#)
workIn, [46](#)