Package: MetMashR (via r-universe)

November 25, 2026

Type Package

Title Metabolite Mashing with R

Version 1.8.0

Description A package to merge, filter sort, organise and otherwise mash together metabolite annotation tables. Metabolite annotations can be imported from multiple sources (software) and combined using workflow steps based on S4 class templates derived from the `struct` package. Other modular workflow steps such as filtering, merging, splitting, normalisation and rest-api queries are included.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 4.5.0), struct

Imports dplyr, methods, httr, httr2, xml2, plyr, scales, ggthemes, utils, rlang, stats, ggplot2, jsonlite

Collate 'generics.R' 'zzz.R' 'annotation_source_class.R' 'annotation_database_class.R' 'AnnotationDb_database.R' 'AnnotationDb_select_class.R' 'BiocFileCache_database_helpers.R' 'BiocFileCache_database_class.R' 'CompoundDb_source_class.R' 'MTox700plus_database_class.R' 'MetMashR-package.R' 'PathBank_metabolite_database_class.R' 'add_columns_class.R' 'add_labels_class.R' 'annotation_bar_chart.R' 'annotation_histogram_class.R' 'annotation_histogram2d_class.R' 'annotation_pie_chart.R' 'annotation_table_class.R' 'annotation_venn_chart.R' 'annotation_upset_chart_class.R' 'calc_ppm_diff_class.R' 'calc_rt_diff_class.R' 'lcms_table_class.R' 'cd_source_class.R' 'chart_plot_doc.R' 'chebi_lookup_parsers.R' 'rest_api_parsers.R' 'rest_api_class.R' 'chebi_lookup_class.R' 'classyfire_batch_lookup_class.R' 'classyfire_lookup_class.R' 'combine_columns_class.R' 'combine_records_class.R'

'combine_sources.R' 'compute_column_class.R'
'compute_record_class.R' 'cts_lite_lookup_class.R'
'database_lookup_class.R' 'dictionaries.R'
'eutils_lookup_class.R' 'excel_database_class.R'
'expand_records_class.R' 'filter_labels_class.R'
'filter_na_class.R' 'filter_range_class.R'
'filter_records_class.R' 'filter_venn_class.R'
'github_file_class.R' 'go_database_class.R'
'hmdb_lookup_class.R' 'id_count_class.R'
'import_source_class.R' 'kegg_lookup_class.R'
'lipidmaps_database_class.R' 'lipidmaps_lookup_class.R'
'ls_source_class.R' 'model_apply_doc.R'
'mspurity_source_class.R' 'mwb_compound_lookup_class.R'
'mwb_refmet_database_class.R' 'mwb_refmet_lookup_class.R'
'mwb_structure_chart_class.R' 'mwb_study_source_class.R'
'mz_match_class.R' 'mzrt_match_class.R'
'normalise_lipids_class.R' 'normalise_strings_class.R'
'opsin_lookup_class.R' 'pivot_columns_class.R'
'prioritise_columns_class.R' 'pubchem_compound_lookup_class.R'
'pubchem_id_exchange_class.R' 'pubchem_property_lookup_class.R'
'pubchem_structure_chart_class.R'
'pubchem_structure_lookup_class.R' 'pubchem_widget.R'
'rdata_database_class.R' 'rds_database_class.R'
'rds_cache_class.R' 'remove_columns_class.R'
'rename_columns_class.R' 'rt_match_class.R'
'select_columns_class.R' 'split_column_class.R'
'sqlite_database_class.R' 'trim_whitespace_class.R'
'unique_records_class.R' 'zenodo_file_class.R'

Suggests covr, httptest, knitr, rmarkdown, testthat (>= 3.0.0), rgoslin, DT, RSQLite, CompoundDb, BiocStyle, BiocFileCache, msPurity, rsvg, metabolomicsWorkbenchR, KEGGREST, magick, structToolbox, ggVennDiagram, patchwork, XML, GO.db, tidytext, tidyr, tidyselect, ComplexUpset, openxlsx, ggplotify, cowplot

Config/testthat/edition 3

VignetteBuilder knitr

URL https://computational-metabolomics.github.io/MetMashR/

Roxygen list(markdown = TRUE)

biocViews WorkflowStep, Metabolomics, KEGG

BugReports https://github.com/computational-metabolomics/MetMashR/issues

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:53:14 UTC

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

RemoteRef RELEASE_3_24

RemoteSha 8f27a047edd5a80dca0865b538bfddeeaf2f8fd3

Contents

add_columns . . . 5
add_labels . . . 6
annotation_bar_chart . . . 7
annotation_database . . . 9
annotation_histogram . . . 10
annotation_histogram2d . . . 11
annotation_pie_chart . . . 13
annotation_source . . . 15
annotation_table . . . 16
annotation_upset_chart . . . 17
annotation_venn_chart . . . 21
AnnotationDb_database . . . 22
AnnotationDb_select . . . 24
BiocFileCache_database . . . 25
cache_as_is . . . 27
calc_ppm_diff . . . 28
calc_rt_diff . . . 29
cd_source . . . 30
chart_plot,annotation_bar_chart,annotation_source-method . . . 32
chebi_lookup . . . 32
check_for_columns . . . 34
classyfire_batch_lookup . . . 35
classyfire_lookup . . . 38
combine_columns . . . 39
combine_records . . . 41
combine_records_helper_functions . . . 42
combine_sources . . . 45
CompoundDb_source . . . 47
compute_column . . . 48
compute_record . . . 49
cts_lite_lookup . . . 50
database_lookup . . . 52
eutils_lookup . . . 54
excel_database . . . 55
filter_labels . . . 57
filter_na . . . 58
filter_range . . . 59
filter_records . . . 60
filter_venn . . . 62
github_file . . . 63
GO_database . . . 65
greek_dictionary . . . 66

hmdb_lookup . . . 67
id_counts . . . 68
import_source . . . 69
is_writable . . . 70
kegg_lookup . . . 70
lcms_table . . . 73
lipidmaps_database . . . 74
lipidmaps_lookup . . . 75
ls_source . . . 77
model_apply,model,annotation_source-method . . . 78
mspurity_source . . . 79
MTox700plus_database . . . 80
mwb_compound_lookup . . . 82
mwb_refmet_database . . . 83
mwb_refmet_lookup . . . 85
mwb_structure . . . 86
mwb_study_source . . . 87
mz_match . . . 89
mzrt_match . . . 90
normalise_lipids . . . 91
normalise_strings . . . 93
opsin_lookup . . . 94
PathBank_metabolite_database . . . 95
pivot_columns . . . 97
prioritise_columns . . . 98
pubchem_compound_lookup . . . 99
pubchem_id_exchange . . . 101
pubchem_property_lookup . . . 103
pubchem_structure . . . 105
pubchem_widget . . . 106
racemic_dictionary . . . 108
rdata_database . . . 108
rds_cache . . . 109
rds_database . . . 110
read_database . . . 111
read_source . . . 112
remove_columns . . . 113
rename_columns . . . 115
required_cols . . . 116
rest_api . . . 117
rt_match . . . 118
select_columns . . . 119
split_column . . . 121
split_records . . . 122
sqlite_database . . . 123
trim_whitespace . . . 124
tripeptide_dictionary . . . 125
unique_records . . . 126

unzip_before_cache 126
upset_min_size 127
vertical_join 129
wherever 130
write_database 131
zenodo_file 132

Index **134**

`add_columns` *Add columns*

Description

A wrapper around `dplyr::left_join`. Adds columns to an annotation table by performing a left-join with an input data.frame (annotations on the left of the join).

Usage

```
add_columns(new_columns, by, ...)
```

Arguments

`new_columns`	(data.frame, annotation_database) A data.frame to be left-joined to the annotation table. Can also be an annotation_database.
`by`	(character) A (named) character vector of column names to join by e.g. `c("A" = "B")` (see `dplyr::left_join` for details).
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `dplyr`

Value

A `add_columns` object with the following `output` slots:

`updated`	(annotation_source) The updated annotations as an `annotation_source` object.

Inheritance

A `add_columns` object inherits the following `struct` classes:

`[add_columns]` -> `[model]` -> `[struct_class]`

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

See Also

dplyr::left_join()

Examples

```
M <- add_columns(
        new_columns = data.frame(),
        by = "id")
```

`add_labels` *Add column of labels*

Description

Adds new columns with the specified labels for each record.

Usage

```
add_labels(labels, replace = FALSE, ...)
```

Arguments

`labels`	(list) A named list of columns and the label to use for all records in that column.
`replace`	(logical) Replace columns. Allowed values are limited to the following: • `"TRUE"`: If present, the new columns will replace existing columns in the source data.frame. • `"FALSE"`: An error will be thrown if the new columns are already in the source data.frame. The default is `FALSE`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `dplyr`

Value

A add_labels object with the following output slots:

updated (annotation_source) The updated annotations as an annotation_source object.

Inheritance

A add_labels object inherits the following struct classes:

[add_labels] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Examples

```
M <- add_labels(
        labels = list(),
        replace = FALSE)
```

---

annotation_bar_chart *Annotation bar chart*

Description

Display a bar chart of labels in the specified column of an annotation_source.

Usage

```
annotation_bar_chart(
  factor_name,
  label_rotation = FALSE,
  label_location = "inside",
  label_type = "percent",
  legend = FALSE,
  ...
)
```

Arguments

factor_name	(character) The name of the column in the annotation_source to generate a chart from.
label_rotation	(logical) Rotate labels. Allowed values are limited to the following: • "TRUE": Rotate labels to match segments. • "FALSE": Do not rotate labels. The default is FALSE.
label_location	(character) Label location. Allowed values are limited to the following: • "inside": Labels are displayed inside the bars. • "outside": Labels are displayed outside the bars. The default is "inside".
label_type	(character) Label type. Allowed values are limited to the following: • "percent": Labels will include the percentage for the segment. • "count": Labels will include the count for the segment. • "none": Labels will not include extra information. The default is "percent".
legend	(logical) Display legend. Allowed values are limited to the following: • "TRUE": Groups are indicated using a legend. • "FALSE": Groups are indicated in the labels. The default is FALSE.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- ggplot2

Value

A annotation_bar_chart object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A annotation_bar_chart object inherits the following struct classes:

[annotation_bar_chart] -> [chart] -> [struct_class]

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. https://ggplot2.tidyverse.org.

Examples

```
M <- annotation_bar_chart(
        factor_name = "V1",
        label_location = "inside",
        label_rotation = FALSE,
        legend = FALSE,
        label_type = "percent")
```

---

annotation_database *An annotation database*

Description

An annotation_database is an annotation_source() where the imported data.frame contains meta data for annotations. For example it might be a table of molecular identifiers, associated pathways etc.

Usage

```
annotation_database(data = data.frame(), tag = "", ...)
```

Arguments

data	(data.frame, NULL) A data.frame of annotation data. The default is data.frame().
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is "".
...	Additional slots and values passed to struct_class.

Value

A annotation_database object. This object has no output slots.

Inheritance

A annotation_database object inherits the following struct classes:

[annotation_database] -> [annotation_source] -> [struct_class]

See Also

Other annotation databases: AnnotationDb_database(), GO_database(), annotation_source(), excel_database(), rdata_database(), rds_cache(), rds_database()

Other annotation sources: annotation_table(), cd_source(), ls_source(), mspurity_source(), mwb_study_source()

Examples

```
M <- annotation_database(
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`annotation_histogram` *Annotation histogram*

Description

Display a histogram of value in the specified column of an annotation_source.

Usage

```
annotation_histogram(
  factor_name,
  bins = 30,
  bin_edge = "grey",
  bin_fill = "lightgrey",
  vline = NULL,
  vline_colour = "red",
  ...
)
```

Arguments

`factor_name`	(character) The name of the column in the `annotation_source` to generate a histogram from.
`bins`	(numeric, integer) The number of bins to use when computing the histogram. The default is `30`.
`bin_edge`	(character) The colour to use when plotting the edges of bins. The default is `"grey"`.
`bin_fill`	(character) The colour to use when plotting the bins. The default is `"lightgrey"`.
`vline`	(numeric, NULL, list) The x-axis location of veritcal lines used to indicate e.g. upper and lower limits. Use NULL if not required. The default is `NULL`.
`vline_colour`	(character) The colour to use when plotting vertical lines. The default is `"red"`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `ggplot2`

Value

A annotation_histogram object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A annotation_histogram object inherits the following struct classes:

[annotation_histogram] -> [chart] -> [struct_class]

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. https://ggplot2.tidyverse.org.

Examples

```
M <- annotation_histogram(
        factor_name = "V1",
        bins = 30,
        bin_edge = "grey",
        bin_fill = "lightgrey",
        vline = NULL,
        vline_colour = "red")
```

---

annotation_histogram2d

Annotation 2D histogram

Description

Display a histogram of value in the specified columns of an annotation_source.

Usage

```
annotation_histogram2d(
  factor_name,
  bins = 30,
  bin_edge = "grey",
  bin_fill = "lightgrey",
  vline = NULL,
  vline_colour = "red",
  ...
)
```

Arguments

factor_name	(character) The names of the two columns in the annotation_source to generate histograms from.
bins	(numeric, integer) The number of bins to use when computing the histograms. The default is 30.
bin_edge	(character) The colour to use when plotting the edges of bins. The default is "grey".
bin_fill	(character) The colour to use when plotting the bins. The default is "lightgrey".
vline	(numeric, NULL, list) The x-axis location of lines used to indicate e.g. upper and lower limits. Use NULL if not required. A 2 element list can be provided to set vlines for each factor_name. The default is NULL.
vline_colour	(character) The colour to use when plotting vertical lines. The default is "red".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- ggplot2
- patchwork

Value

A annotation_histogram2d object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A annotation_histogram2d object inherits the following struct classes:

[annotation_histogram2d] -> [annotation_histogram] -> [chart] -> [struct_class]

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. https://ggplot2.tidyverse.org.

Pedersen T (2025). *patchwork: The Composer of Plots*. doi:10.32614/CRAN.package.patchwork https://doi.org/10.32614/CRAN.package.patchwork. R package version 1.3.2, https://CRAN.R-project.org/package=patchwork.

Examples

```
M <- annotation_histogram2d(
        factor_name = "V1",
        bins = 30,
        bin_edge = "grey",
```

```
          bin_fill = "lightgrey",
          vline = NULL,
          vline_colour = "red")
```

`annotation_pie_chart` *Annotation pie chart*

Description

Display a pie chart of labels in the specified column of an annotation_source.

Usage

```
annotation_pie_chart(
  factor_name,
  label_rotation = FALSE,
  label_location = "inside",
  label_type = "percent",
  legend = FALSE,
  pie_rotation = 0,
  centre_radius = 0,
  centre_label = NULL,
  count_na = FALSE,
  ...
)
```

Arguments

`factor_name`	(character) The name of the column in the `annotation_source` to generate a pie chart from.
`label_rotation`	(logical) Rotate labels. Allowed values are limited to the following: • `"TRUE"`: Rotate labels to match segments. • `"FALSE"`: Do not rotate labels. The default is `FALSE`.
`label_location`	(character) Label location. Allowed values are limited to the following: • `"inside"`: Labels are displayed inside the segments. • `"outside"`: Labels are displayed outside the segments. The default is `"inside"`.
`label_type`	(character) Label type. Allowed values are limited to the following: • `"percent"`: Labels will include the percentage for the segment. • `"count"`: Labels will include the count for the segment. • `"none"`: Labels will not include extra information.

The default is "percent".

legend (logical) Display legend. Allowed values are limited to the following:

- "TRUE": Groups are indicated using a legend.
- "FALSE": Groups are indicated in the labels.

The default is FALSE.

pie_rotation (numeric) The number of degrees to rotate the pie chart by, clockwise. The default is 0.

centre_radius (numeric, integer) The radius of the centre circle. Used to make a "donut" plot. Should be a value between 0 and

1. The default is 0.

centre_label (NULL, character) The text to display in the centre of the pie chart. Mostly used with donut plots where centre_radius is greater than 0. The default is NULL.

count_na (logical) Include the number of missing values in the pie chart. The default is FALSE.

... Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- ggplot2

Value

A annotation_pie_chart object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A annotation_pie_chart object inherits the following struct classes:

[annotation_pie_chart] -> [chart] -> [struct_class]

References

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. https://ggplot2.tidyverse.org.

Examples

```
M <- annotation_pie_chart(
        factor_name = "V1",
        label_location = "inside",
        label_rotation = FALSE,
        legend = FALSE,
        pie_rotation = 0,
        label_type = "percent",
        centre_radius = 0,
        centre_label = NULL,
        count_na = FALSE)
```

---

annotation_source *An annotation source*

Description

A base class defining an annotation source. This object is extended by MetmashR to define other objects.

Usage

```
annotation_source(source = character(0), data = data.frame(), tag = "", ...)
```

Arguments

source	(ANY) The source of annotation data. The default is `character(0)`.
data	(data.frame, NULL) A data.frame of annotation data. The default is `data.frame()`.
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is `""`.
...	Additional slots and values passed to `struct_class`.

Value

A `annotation_source` object. This object has no `output` slots.

Inheritance

A `annotation_source` object inherits the following `struct` classes:

`[annotation_source]` -> `[struct_class]`

See Also

Other annotation databases: `AnnotationDb_database()`, `GO_database()`, `annotation_database()`, `excel_database()`, `rdata_database()`, `rds_cache()`, `rds_database()`

Examples

```
M <- annotation_source(
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`annotation_table` *An annotation table*

Description

An `annotation_table` is an `annotation_source()` where the imported data.frame contains measured experimental data. An `id_column` of values is required to uniquely indentify each record (row) in the table (NB these are NOT molecule identifiers, which may be be present in multiple records).

Usage

```
annotation_table(data = data.frame(), tag = "", id_column = NULL, ...)
```

Arguments

`data`	(data.frame, NULL) A data.frame of annotation data. The default is `data.frame()`.
`tag`	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is `""`.
`id_column`	(character) The column name of the annotation data.frame containing row identifers. If NULL This will be generated automatically. The default is `NULL`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `annotation_table` object. This object has no `output` slots.

Inheritance

A `annotation_table` object inherits the following `struct` classes:

`[annotation_table]` -> `[annotation_source]` -> `[struct_class]`

See Also

Other annotation tables: `cd_source()`, `ls_source()`, `mspurity_source()`

Other annotation sources: `annotation_database()`, `cd_source()`, `ls_source()`, `mspurity_source()`, `mwb_study_source()`

Examples

```
M <- annotation_table(
        id_column = "id",
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`annotation_upset_chart` *Annotation UpSet chart*

Description

Display an UpSet chart of labels in the specified column of an annotation_source.

Usage

```
annotation_upset_chart(
  factor_name,
  group_column = NULL,
  order_intersect_by = "size",
  order_set_by = "name",
  nintersects = NULL,
  filter = NULL,
  relative_width = 0.3,
  relative_height = 3,
  top_bar_color = "grey30",
  top_bar_y_label = NULL,
  top_bar_show_numbers = TRUE,
  top_bar_numbers_size = 3,
  sets_bar_color = "grey30",
  sets_bar_show_numbers = FALSE,
  sets_bar_x_label = "Set Size",
  sets_bar_position = "left",
  intersection_matrix_color = "grey30",
  specific = TRUE,
  ...
)
```

Arguments

`factor_name`	(character) The name of the column(s) in the `annotation_source`(s) to generate an UpSet chart from.
`group_column`	(character, NULL) The name of the column in the `annotation_source` to create groups from in the Venn diagram. This parameter is ignored if there are multiple input tables, as each table is considered to be a group. This parameter

is also ignored if more than one factor_name is provided, as each column is considered a group. The default is NULL.

order_intersect_by
: (character) Order intersect by. Allowed values are limited to the following:
  - "size": Intersections are sorted by size (largest first).
  - "name": Intersections are sorted by name alphabetically.
  - "none": Intersections are not sorted.

  The default is "size".

order_set_by
: (character) Order set by. Allowed values are limited to the following:
  - "size": Sets are sorted by size (largest first).
  - "name": Sets are sorted by name alphabetically.
  - "none": Sets are not sorted.

  The default is "name".

nintersects
: (numeric, integer, NULL) The number of intersections to include in the plot. The default is NULL.

filter
: (function, NULL) A function or list of functions to filter intersections based on their properties. The function(s) should take region_data as input and return a logical vector indicating which intersections to keep. Use upset_min_size(), upset_min_groups(), upset_max_groups(), upset_intersections(), or create custom filter functions. The default is NULL.

relative_width
: (numeric) The relative width of the left panel in the upset plot. The default is 0.3.

relative_height
: (numeric) The relative height of the top panel in the upset plot. The default is 3.

top_bar_color
: (character) The color of the top bar chart showing intersection sizes. The default is "grey30".

top_bar_y_label
: (character, NULL) The label for the Y-axis of the top bar chart. The default is NULL.

top_bar_show_numbers
: (logical) Whether to show numbers on the top bar chart. The default is TRUE.

top_bar_numbers_size
: (numeric) The text size of numbers on the top bar chart. The default is 3.

sets_bar_color
: (character) The color of the sets bar chart. The default is "grey30".

sets_bar_show_numbers
: (logical) Whether to show numbers on the sets bar chart. The default is FALSE.

sets_bar_x_label
: (character) The label for the X-axis of the sets bar chart. The default is "Set Size".

`sets_bar_position`
: (character) Sets bar position. Allowed values are limited to the following:
  - `"left"`: Position the sets bar chart on the left side.
  - `"right"`: Position the sets bar chart on the right side.

  The default is `"left"`.

`intersection_matrix_color`
: (character) The color of the intersection matrix dots and lines. The default is `"grey30"`.

`specific`
: (logical) Whether to include only specific items in subsets (TRUE) or all overlapping items (FALSE). The default is `TRUE`.

`...`
: Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `ggVennDiagram`

The plot object returned is of class 'aplot' which may not be compatible with all plot combination functions. To combine with other ggplot objects using cowplot or patchwork, use ggplotify::as.ggplot() to convert the plot object:

```
library(ggplotify)
g <- chart_plot(C, data)
g_ggplot <- as.ggplot(g)
cowplot::plot_grid(g1, g_ggplot, nrow = 1)
```

Value

A `annotation_upset_chart` object. This object has no `output` slots. See `chart_plot` in the `struct` package to plot this chart object.

Inheritance

A `annotation_upset_chart` object inherits the following `struct` classes:

`[annotation_upset_chart]` -> `[chart]` -> `[struct_class]`

Filtering

Use the `filter` parameter to filter intersections based on their properties:

```
# Filter by minimum size
C <- annotation_upset_chart(factor_name = "V1", filter = upset_min_size(5))

# Filter by minimum number of groups
C <- annotation_upset_chart(factor_name = "V1", filter = upset_min_groups(3))
```

```
# Filter to show only specific combinations
C <- annotation_upset_chart(factor_name = "V1",
                            filter = upset_intersections(c("A/B", "B/C")))

# Custom filter function
custom_filter <- function(region_data) {
  region_data$count >= 3 & grepl("A", region_data$name)
}
C <- annotation_upset_chart(factor_name = "V1", filter = custom_filter)
```

Note

The interface to this class has changed. Some parameters have been renamed:

- 'width_ratio' -> 'relative_width'
- 'xlabel' -> 'top_bar_y_label'
- 'sort_intersections' -> 'order_intersect_by'
- 'intersections' -> 'nintersects'
- 'n_intersections' -> 'nintersects'
- 'queries' -> (removed)
- 'keep_empty_group' -> (removed)
- 'sort_sets' -> 'order_set_by'

Old parameter names will trigger deprecation warnings.

References

Gao C, Dusa A (2026). *ggVennDiagram: A 'ggplot2' Implement of Venn Diagram*. doi:10.32614/CRAN.package.ggVennDiag
https://doi.org/10.32614/CRAN.package.ggVennDiagram. R package version 1.5.7, https:
//CRAN.R-project.org/package=ggVennDiagram.

Examples

```
M <- annotation_upset_chart(
        factor_name = "V1",
        group_column = NULL,
        order_intersect_by = "size",
        order_set_by = "name",
        nintersects = NULL,
        filter = NULL,
        relative_width = 0.3,
        relative_height = 3,
        top_bar_color = "grey30",
        top_bar_y_label = NULL,
        top_bar_show_numbers = FALSE,
        top_bar_numbers_size = 3,
        sets_bar_color = "grey30",
        sets_bar_show_numbers = FALSE,
        sets_bar_x_label = "Set Size",
```

```
          sets_bar_position = "left",
          intersection_matrix_color = "grey30",
          specific = FALSE)
```

`annotation_venn_chart` *Annotation venn chart*

Description

Display a venn diagram of labels present in two annotation_sources.

Usage

```
annotation_venn_chart(
  factor_name,
  group_column = NULL,
  fill_colour = "white",
  line_colour = "black",
  labels = TRUE,
  legend = FALSE,
  ...
)
```

Arguments

`factor_name`	(character) The name of the column(s) in the `annotation_source` to generate a chart from. Up to seven columns can be compared for a single `annotation_source`.
`group_column`	(character, NULL) The name of the column in the `annotation_source` to create groups from in the Venn diagram. This parameter is ignored if there are multiple input tables, as each table is considered to be a group. This parameter is also ignored if more than one `factor_name` is provided, as each column is considered a group. The default is `NULL`.
`fill_colour`	(character) The line colour of the groups in a format compatible with ggplot e.g. "black" or "#000000". Special case ".group" sets the colour based on the group label and "none" will not fill the groups. The default is `"white"`.
`line_colour`	(character) The line colour of the groups in a format compatible with ggplot e.g. "black" or "#000000". Special case ".group" sets the colour based on the group label, and ".none" will not display lines. The default is `"black"`.
`labels`	(logical) Group labels. Allowed values are limited to the following: • `"TRUE"`: Include group labels on the plot. • `"FALSE"`: Do not inlude group labels on the plot. The default is `TRUE`.
`legend`	(logical) Legend. Allowed values are limited to the following:

- "TRUE": Include a legend in the plot.
- "FALSE": Do not inlude a legend in the plot.

The default is FALSE.

... Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- ggVennDiagram

Value

A annotation_venn_chart object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A annotation_venn_chart object inherits the following struct classes:

[annotation_venn_chart] -> [chart] -> [struct_class]

References

Gao C, Dusa A (2026). *ggVennDiagram: A 'ggplot2' Implement of Venn Diagram*. doi:10.32614/CRAN.package.ggVennDiag https://doi.org/10.32614/CRAN.package.ggVennDiagram. R package version 1.5.7, https://CRAN.R-project.org/package=ggVennDiagram.

Examples

```
M <- annotation_venn_chart(
        factor_name = "V1",
        line_colour = ".group",
        fill_colour = ".group",
        labels = FALSE,
        legend = FALSE,
        group_column = NULL)
```

AnnotationDb_database *AnnotationDb database*

Description

Retrieve a table from an AnnotationDb package.

Usage

```
AnnotationDb_database(source, table, ...)
```

Arguments

source	(character) The name of an AnnotationDb package to import the specified table from. Note the package should already be installed.
table	(character) The name of a table to import from the specified source AnnotationDb package.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- AnnotationDbi

Value

A AnnotationDb_database object. This object has no output slots.

Inheritance

A AnnotationDb_database object inherits the following struct classes:

[AnnotationDb_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Pagès H, Carlson M, Falcon S, Li N (2026). *AnnotationDbi: Manipulation of SQLite-based annotations in Bioconductor*. doi:10.18129/B9.bioc.AnnotationDbi https://doi.org/10.18129/B9.bioc.AnnotationDbi. R package version 1.74.0, https://bioconductor.org/packages/AnnotationDbi.

See Also

AnnotationDbi::AnnotationDb

Other annotation databases: GO_database(), annotation_database(), annotation_source(), excel_database(), rdata_database(), rds_cache(), rds_database()

Examples

```
M <- AnnotationDb_database(
        table = character(0),
        tag = character(0),
        data = data.frame(),
        source = character(0))
```

AnnotationDb_select *Select columns from AnnotationDb database*

Description

A wrapper around [annotationDbi::select()] that can be used to import columns from the database where the keys are provided by a column in the annotation table.

Usage

```
AnnotationDb_select(
  database,
  key_column,
  key_type,
  database_columns,
  drop_na = TRUE,
  ...
)
```

Arguments

database	(character) The name of the AnnotationDbi package/object to import.
key_column	(character) The name of a column in the annotation table containing key values used to extract records from the AnnotationDbi database.
key_type	(character) The name of a column in the AnnoationDb database searched for matches to the key values.
database_columns	(character) The name of columns to import from the AnnoationDb database. Special case .all can be used to get all columns.
drop_na	(logical) Drop NA. Allowed values are limited to the following: • "TRUE": Remove rows where all columns of the returned database are NA. • "FALSE": Keep rows where all columns of the returned database are NA. The default is TRUE.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr
- AnnotationDbi

Value

A AnnotationDb_select object with the following output slots:

updated (annotation_source) The updated annotations as an annotation_source object.

Inheritance

A AnnotationDb_select object inherits the following struct classes:

[AnnotationDb_select] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Pagès H, Carlson M, Falcon S, Li N (2026). *AnnotationDbi: Manipulation of SQLite-based annotations in Bioconductor*. doi:10.18129/B9.bioc.AnnotationDbi https://doi.org/10.18129/B9.bioc.AnnotationDbi. R package version 1.74.0, https://bioconductor.org/packages/AnnotationDbi.

See Also

dplyr::left_join()

AnnotationDbi::select()

Examples

```
M <- AnnotationDb_select(
        database = "",
        key_column = "",
        key_type = "",
        database_columns = ".all",
        drop_na = FALSE)
```

BiocFileCache_database

Cached database

Description

A cached resource using BiocFileCache.

Usage

```
BiocFileCache_database(
  source,
  bfc_path = NULL,
  resource_name,
  bfc_fun = cache_as_is,
  import_fun = read.csv,
  offline = FALSE,
  ...
)
```

Arguments

source	(ANY) The source of annotation data.
bfc_path	(character, NULL) BiocFileCache is used to cache the database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache() for details). The default is NULL.
resource_name	(character) The name given to this resource in the cache. (see BiocFileCache::BiocFileCache() for details).
bfc_fun	(function) A function to process the object before storing it in the cache, e.g. to store an unzipped file in the cache instead of the zipped version. This would prevent needing to unzip the resource each time it is retrieved from the cache, but would mean using more space on disk. The default function does nothing to the resource. See BiocFileCache::bfcdownload() for details.
import_fun	(function) A function to process the object after retrieving it from the cache e.g. it might need to be unzipped before importingas a data.frame. This function should take the path to the cached object as the first input and return a data.frame.
offline	(logical) If offline = FALSE then checks to determine if the resource has expired will be skipped, and retrieved directly from the cache. The default is FALSE.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache

Value

A BiocFileCache_database object. This object has no output slots.

Inheritance

A BiocFileCache_database object inherits the following struct classes:

[BiocFileCache_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

See Also

Other database: lipidmaps_database(), sqlite_database()

Examples

```
M <- BiocFileCache_database(
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

cache_as_is *Cache file with no changes using BiocFileCache*

Description

This helper function is for use with BiocFileCache objects. Using it will copy the file directly to the cache without making any changes.

Usage

```
cache_as_is(from, to)
```

Arguments

from	incoming path
to	the outgoing path

Value

TRUE if successful

Examples

```
M <- BiocFileCache_database(
    source = tempfile(),
    resource_name = "example",
    bfc_fun = cache_as_is
)
```

`calc_ppm_diff` *Calculate ppm difference*

Description

Calculate ppm difference between two columns in an `annotation_table`. e.g. for comparing observed m/z to theortical ones.

Usage

```
calc_ppm_diff(
  obs_mz_column,
  ref_mz_column,
  out_column,
  check_names = "unique",
  ...
)
```

Arguments

`obs_mz_column`	(character) Column name in annotation_table containing the observed m/z values.
`ref_mz_column`	(character) Column name in annotation table containing the .
`out_column`	(character) Column name in annotation table to store the computed ppm differences.
`check_names`	(character) Check names. Allowed values are limited to the following: • `"stop"`: If the output column already exists an error will be thrown. • `"unique"`: If the output column already exists a unique column name will be generated. • `"replace"`: If the output column already exists it will be replaced. The default is `"unique"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `calc_ppm_diff` object with the following `output` slots:

`updated` (annotation_table) The input annotation source with the computed ppm diffences in a new column.

Inheritance

A `calc_ppm_diff` object inherits the following `struct` classes:

`[calc_ppm_diff]` -> `[model]` -> `[struct_class]`

Examples

```
M <- calc_ppm_diff(
        obs_mz_column = character(0),
        ref_mz_column = "reference (theoretical) m/z values.",
        out_column = character(0),
        check_names = "unique")
```

---

calc_rt_diff *Calculate RT difference*

Description

Calculate RT difference between two RT values

Usage

```
calc_rt_diff(
  obs_rt_column,
  ref_rt_column,
  out_column,
  check_names = "unique",
  ...
)
```

Arguments

`obs_rt_column`	(character) Column name in annotation table containing the observed (measured) RT values.
`ref_rt_column`	(character) Column name in annotation table containing the reference (theoretical) RT values.
`out_column`	(character) Column name in annotation table to store the computed RT differences.
`check_names`	(character) Check names. Allowed values are limited to the following:

- "stop": If the output column already exists an error will be thrown.
- "unique": If the output column already exists a unique column name will be generated.
- "replace": If the output column already exists it will be replaced.

The default is "unique".

... Additional slots and values passed to struct_class.

Value

A calc_rt_diff object with the following output slots:

updated (annotation_table) The input annotation source with the newly generated column.

Inheritance

A calc_rt_diff object inherits the following struct classes:

[calc_rt_diff] -> [model] -> [struct_class]

Examples

```
M <- calc_rt_diff(
        obs_rt_column = character(0),
        ref_rt_column = character(0),
        out_column = character(0),
        check_names = "unique")
```

---

cd_source *LCMS table*

Description

An LCMS table extends annotation_table() to represent annotation data for an LCMS experiment. Columns representing m/z and retention time are required for an lcms_table.

Usage

```
cd_source(
  source,
  sheets = c(1, 1),
  tag = "CD",
  mz_column = "mz",
  rt_column = "rt",
  id_column = "id",
  data = NULL,
  ...
)
```

Arguments

source	(character) The path to the Compound Discoverer Excel files to import. Both the compounds and isomers file should be included, in that order.
sheets	(character, numeric, integer) The name or index of the sheets to read from the source file(s). A sheet should be provided for each input file. The default is c(1, 1).
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is "CD".
mz_column	(character) The column name of the annotation data.frame containing m/z values. The default is "mz".
rt_column	(character) The column name of the annotation data.frame containing retention time values. The default is "rt".
id_column	(character) The column name of the annotation data.frame containing row identifers. If NULL This will be generated automatically. The default is "id".
data	(data.frame, NULL) A data.frame of annotation data. The default is NULL.
...	Additional slots and values passed to struct_class.

Value

A cd_source object. This object has no output slots.

Inheritance

A cd_source object inherits the following struct classes:

[cd_source] -> [lcms_table] -> [annotation_table] -> [annotation_source] -> [struct_class]

See Also

Other annotation sources: annotation_database(), annotation_table(), ls_source(), mspurity_source(), mwb_study_source()

Other annotation tables: annotation_table(), ls_source(), mspurity_source()

Examples

```
M <- cd_source(
        sheets = c(1, 1),
        mz_column = "mz",
        rt_column = "rt",
        id_column = "id",
        tag = character(0),
        data = data.frame(),
        source = character(0))
```

chart_plot,annotation_bar_chart,annotation_source-method *chart_plot method*

Description

Plots a chart object

Usage

```
chart_plot(obj, dobj, ...)
```

Arguments

obj	a chart object
dobj	a struct object
...	additiional inputs to chart_plot

Value

a plot object

Examples

```
C <- example_chart()
chart_plot(C, example_model())
```

chebi_lookup *ID/synonym lookup via ChEBI*

Description

Uses the ChEBI REST API to look up a ChEBI identifier from a compound name/synonym, or a synonym from a ChEBI identifier, based on the input annotation column.

Usage

```
chebi_lookup(
  query_column,
  search_by = c("name", "chebi_id"),
  suffix = "_chebi",
  records = "best",
  max_records = "50",
  columns = ".all",
  delay = 1,
  ...
)
```

Arguments

query_column	(character) The name of a column in the annotation table containing the search terms. If search_by = "name" this should be a column of compound names/synonyms. If search_by = "chebi_id" this should be a column of ChEBI identifiers (e.g. "CHEBI:27732" or "27732"; any "CHEBI:" prefix is stripped automatically before querying the API).
search_by	(character) Search by. Allowed values are limited to the following: • "name": Search ChEBI by compound name/synonym and return the matching ChEBI ID(s). • "chebi_id": Search ChEBI by ChEBI ID and return the matching synonym(s)/name(s). The default is c("name","chebi_id").
suffix	(character) A suffix appended to all column names in the returned result. The default is "_chebi".
records	(character) Returned record(s). Allowed values are limited to the following: • "": There can be multiple matches for a given search term (multiple ChEBI entities matching a name, or multiple synonyms for a ChEBI ID). • "best": Return only the single best/primary matching record. • "all": Return all matching records. The default is "best".
max_records	(character) The maximum number of hits requested from the ChEBI search API when search_by = "name" (passed as the "size" query parameter). Has no effect when search_by = "chebi_id". The default is "50".
columns	(character) The columns to include in the result. One or more of "chebi_id", "name" (search_by = "name" only), "synonym" (search_by = "chebi_id" only), "inchikey", "smiles", "inchi", "formula", "mass", "monoisotopicmass", "charge", "stars" (ChEBI's own curation-quality rating, 3 = fully manually annotated). Keyword ".all" (the default) returns every column available for the chosen search_by direction. The default is ".all".
delay	(numeric, integer) Delay in seconds between API calls. ChEBI's REST API does not publish a specific rate limit, so this default is a conservative courtesy value; increase it if you see 429/503 responses, or decrease it if you have confirmed a higher rate is acceptable. The default is 1.
...	Additional slots and values passed to struct_class.

Value

A chebi_lookup object with the following output slots:

updated	(annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A chebi_lookup object inherits the following struct classes:

[chebi_lookup] -> [rest_api] -> [model] -> [struct_class]

References

Hastings, Janna, Owen, Gareth, Dekker, Adriano, Ennis, Marcus, Kale, Namrata, Muthukrishnan, Venkatesh, Turner, Steve, Swainston, Neil, Mendes, Pedro, Steinbeck, Christoph (2016). "ChEBI in 2016: Improved services and an expanding collection of metabolites." *Nucleic Acids Research*, *44*(D1), D1214-D1219. doi:10.1093/nar/gkv1031 https://doi.org/10.1093/nar/gkv1031.

Examples

```
M <- chebi_lookup(
        search_by = "name",
        records = "best",
        max_records = "50",
        columns = ".all",
        base_url = "https://www.ebi.ac.uk/chebi/backend/api/public",
        url_template = "<base_url>/es_search?term=<query_column>&size=<max_records>",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 1,
        suffix = "_rest_api")
```

---

check_for_columns *Check for columns in an* annotation_source

---

Description

This method checks for the presence of columns by name in an annotation_source(). It returns TRUE if all are present, or a vector of messages indicating which columns are missing from the data.frame. It is used by MetMashR to ensure validity of certain objects.

Usage

```
check_for_columns(obj, ..., msg = FALSE)

## S4 method for signature 'annotation_source'
check_for_columns(obj, ..., msg = FALSE)
```

Arguments

obj	an annotation_source() object
...	the column names to check for
msg	TRUE/FALSE indicates whether to return a message if some columns are missing. If msg = FALSE then the function returns FALSE if all columns are not present.

Value

logical if all columns are present, or a vector of messages if requested.

Examples

```
# test if column present
AT <- annotation_source(data = data.frame(id = character(0)))
check_for_columns(AT, "id") # TRUE
check_for_columns(AT, "cake") # FALSE

# return a message if missing
check_for_columns(AT, "cake", msg = TRUE)
```

---

classyfire_batch_lookup

ClassyFire batch lookup

Description

Uses ClassyFire's batch submission API to obtain chemical ontology information (kingdom/superclass/class/...) for many SMILES in one or a few requests, rather than one request per compound (see classyfire_lookup()).

Usage

```
classyfire_batch_lookup(
  query_column,
  output_items = c("kingdom", "superclass", "class"),
  output_fields = "name",
  suffix = "_cfb",
  delay = 3,
  max_poll_rounds = 30,
  n_batches = NULL,
  cache = NULL,
  cache_mode = "update",
  verbose = FALSE,
  ...
)
```

Arguments

`query_column`	(character) The name of a column in the annotation table containing SMILES strings to submit to ClassyFire.
`output_items`	(character) The names of the items to return: "kingdom", "superclass", "class", "subclass" and/or "direct_parent" - the taxonomy items ClassyFire returns as a nested name/description/chemont_id/url object per compound. Keyword ".all" returns all of them. Other ClassyFire fields (substituents, ancestors, ...) are array-valued per compound rather than a single nested object and are not supported by this object. The default is `c("kingdom", "superclass", "class")`.
`output_fields`	(character) The fields to return for each output_item, where applicable. Can include "name", "description", "chemont_id" and "url". Keyword ".all" returns all fields. The default is `"name"`.
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_cfb"`.
`delay`	(numeric, integer) Delay in seconds between status polling requests. The default is `3`.
`max_poll_rounds`	(numeric, integer) Maximum number of polling rounds before giving up. The default is `30`.
`n_batches`	(numeric, integer, NULL) Optional number of submissions to split the unique query values across. If NULL, all query values are submitted in a single request. The default is `NULL`.
`cache`	(annotation_database, NULL) A struct cache object (e.g. `rds_cache()`) that stores parsed responses to previous ClassyFire queries, keyed by query value. Values already present in the cache are not resubmitted. Besides speeding up re-runs, this keeps a local, persistent record of what ClassyFire returned - useful given the service can go down or change without notice, as happened to the Chemical Translation Service this package once depended on (see `mwb_refmet_lookup()`). If not using a cache then set to NULL. The default is `NULL`.
`cache_mode`	(character) Cache mode. Allowed values are limited to the following: • `"update"`: The normal mode: values already in `cache` are used as-is, anything missing is submitted to ClassyFire and the result added to the cache. • `"offline"`: Never submit a batch to ClassyFire - only values already present in `cache` are returned, and everything else is left as NA. Useful for continuing to work with a partially-populated cache while ClassyFire is unreachable or down, without waiting on or erroring against the live service. A warning lists how many query values are not covered by the cache when this happens (and, if the cache is entirely empty or unset, that every value will be returned as NA). • `"rebuild"`: Ignore any existing cached value and submit every value to ClassyFire, overwriting the corresponding entry in `cache`. Useful when cached results are known to be stale. The default is `"update"`.

verbose (logical) Whether to print debug information during execution. The default is FALSE.

... Additional slots and values passed to struct_class.

Details

ClassyFire's per-compound endpoint (see classyfire_lookup()) only accepts one InChIKey per request and is aggressively rate-limited, making it impractical for more than a few dozen compounds. ClassyFire also provides a batch submission API (wishartlab/classyfire_api)

- POST .../queries.json with many structures at once, then poll GET .../queries/<id>.json until finished - which this object uses instead: one submission (or a handful, via n_batches) covers the whole input. This batch endpoint takes a **SMILES or InChI** string (not a bare InChIKey) per compound; each is submitted as an <value><TAB><value> pair so ClassyFire echoes the value back as its identifier, letting results be matched back to the original rows by exact equality rather than relying on ClassyFire's undocumented internal ordering.

Value

A classyfire_batch_lookup object with the following output slots:

updated	(annotation_source) The annotation_source after adding data returned by ClassyFire.
query_batches	(list) The query values submitted in each batch during training (excludes any already present in cache).
query_values	(character) Every unique, non-missing value of query_column seen during training (cached or not) - used t
request_ids	(list) ClassyFire query ids for each submitted batch.
trained	(logical) Whether the model has already submitted its ClassyFire jobs.

Inheritance

A classyfire_batch_lookup object inherits the following struct classes:

[classyfire_batch_lookup] -> [model] -> [struct_class]

See Also

Other REST API's: classyfire_lookup(), cts_lite_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- classyfire_batch_lookup(
        query_column = character(0),
        output_items = c("kingdom", "superclass", "class"),
        output_fields = "name",
        suffix = "_cfb",
        delay = 3,
        max_poll_rounds = 30,
        n_batches = NULL,
```

```
        cache = NULL,
        cache_mode = "update",
        verbose = FALSE)
```

`classyfire_lookup` *Query ClassyFire database*

Description

Queries the ClassyFire database by inchikey to obtain chemical ontology information.

Usage

```
classyfire_lookup(
  query_column,
  output_items = "kingdom",
  output_fields = "name",
  suffix = "_cf",
  ...
)
```

Arguments

`query_column`	(character) The name of a column in the annotation table containing values to search in the api call.
`output_items`	(character) The names of the items to return from the results of the search. Can include any number of "kingdom", "superclass", "class", "subclass", "direct_parent", "intermediate_nodes", "substituents", "smiles", "molecular_framework", "description", "ancestors", "predicted_chebi_terms". Keyword ".all" may be used to return all items. The default is `"kingdom"`.
`output_fields`	(character) The names of fields to return for each output_item. Can include any of "name", "description", "chemont_id" and "url". Keyword ".all" may be used to return all fields. Some items do not have fields, so output_category is ignored. The default is `"name"`.
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_cf"`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `dplyr`
- `httr`

Value

A classyfire_lookup object with the following output slots:

updated (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A classyfire_lookup object inherits the following struct classes:

[classyfire_lookup] -> [rest_api] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

See Also

Other REST API's: classyfire_batch_lookup(), cts_lite_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- classyfire_lookup(
        output_items = "kingdom",
        output_fields = "name",
        base_url = "http://classyfire.wishartlab.com/entities",
        url_template = "<base_url>/<query_column>.json",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

combine_columns *Combine columns*

Description

A wrapper for paste() and interaction(). Combines the values in multiple columns row-wise.

Usage

```
combine_columns(
  column_names,
  separator = "_",
  prefix = NULL,
  suffix = NULL,
  output_column = "combined",
  clean = TRUE,
  ...
)
```

Arguments

column_names	(character) The column name(s) in the annotation_source to combine.
separator	(character) A string placed in between the two being joined. The default is "_".
prefix	(character, NULL) A string placed at the start of the combined strings. The default is NULL.
suffix	(character, NULL) A string placed at the end of the combined strings. The default is NULL.
output_column	(character) The name of a column to store the combined values in. The default is "combined".
clean	(logical) Clean old columns. Allowed values are limited to the following: • "TRUE": The named columns are removed after being combined. • "FALSE": The named columns are retained after being combined. The default is TRUE.
...	Additional slots and values passed to struct_class.

Value

A combine_columns object with the following output slots:

updated	(annotation_source) The annotation_source after combining the columns.

Inheritance

A combine_columns object inherits the following struct classes:

[combine_columns] -> [model] -> [struct_class]

Examples

```
M <- combine_columns(
        column_names = "V1",
        separator = "_",
```

```
          output_column = "combined",
          clean = FALSE,
          prefix = NULL,
          suffix = NULL)
```

`combine_records` *Combine annotation records (rows)*

Description

Combine annotation records (rows) based on a key. All records with the same key will be combined. A number of helper functions are provided for common approaches to merging records.

Usage

```
combine_records(
  group_by,
  default_fcn = fuse(separator = " || "),
  fcns = list(),
  ...
)
```

Arguments

`group_by`	(character) The column used as the key for grouping records.
`default_fcn`	(function) The default function to use for summarising columns when combining records and a specific function has not been provided in fcns. The default is `fuse(separator = " || ")`.
`fcns`	(list) A named list of functions to use for summarising named columns when combining records. Names should correspond to the columns in the annotation table. The default is `list()`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `dplyr`

Value

A `combine_records` object with the following `output` slots:

`updated`	(annotation_source) The input annotation source with the newly generated column.

Inheritance

A `combine_records` object inherits the following `struct` classes:

`[combine_records]` -> `[model]` -> `[struct_class]`

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Lloyd GR, Jankevics A, Weber RJM (2020). "struct: an R/Bioconductor-based framework for standardized metabolomics data analysis and beyond." *Bioinformatics*, *36*(22-23), 5551-5552.

Examples

```
M <- combine_records(
        fcns = list(),
        group_by = character(0),
        default_fcn = function(){})
```

---

combine_records_helper_functions

Combine records helper functions

Description

This page documents helper functions for use with `combine_records()`.

Usage

```
compute_mode(ties = FALSE, na.rm = TRUE)

compute_mean(na.rm = TRUE)

compute_median(na.rm = TRUE)

fuse(separator, na_string = "NA")

select_max(max_col, use_abs = FALSE, keep_NA = FALSE, unique = FALSE)

select_min(min_col, use_abs = FALSE, keep_NA = FALSE, unique = FALSE)

select_match(match_col, search_col, separator, na_string = "NA")

select_exact(match_col, match, separator, na_string = "NA")
```

```
fuse_unique(
  separator,
  na_string = "NA",
  digits = 6,
  drop_na = FALSE,
  sort = FALSE
)

prioritise(match_col, priority, separator, no_match = NA, na_string = "NA")

nothing()

count_records()

select_grade(grade_col, keep_NA = FALSE, upper_case = TRUE)
```

Arguments

ties	(logical) If TRUE then all records matching the tied groups are returned. Otherwise the first record is returned.
na.rm	(logical) If TRUE then NA is ignored
separator	(character, NULL) if !NULL this string is used to collapse matches with the same priority
na_string	(character) NA values are replaced with this string
max_col	(character) the column name to search for the maximum value.
use_abs	(logical) If TRUE then the sign of the values is ignored.
keep_NA	(logical) If TRUE keeps records with NA values
unique	(logical) If TRUE then exactly one record is returned even when multiple records tie, by taking the first matching record. This guarantees a single, deterministic and reproducible result per group instead of one row per tied record.
min_col	(character) the column name to search for the minimum value.
match_col	(character) the column with labels to prioritise
search_col	(character) the name of a column to use as a reference for locating values in the matching column.
match	(character) a value to search for in the matching column.
digits	(numeric) the number of digits to use when converting numerical values to characters when determining if values are unique.
drop_na	(logical) exclude NA from the list of unique entires
sort	(logical) sort the values before collapsing.
priority	(character) a list of labels in priority order
no_match	(character, NULL) if !NULL then annotations not matching any of the priority labels are replaced with this value
grade_col	(character) the name of a column containing grades
upper_case	(logical) If TRUE then grades are compared to upper case letters to determine their ordering, otherwise lower case.

Value

A function for use with `combine_records()`

Functions

- `compute_mode()`: returns the most common value, excluding NA. If `ties == TRUE` then all tied values are returned, otherwise the first value in a sorted unique list is returned (equal to min if numeric). If `na.rm = FALSE` then NA are included when searching for the modal value and placed last if `ties = FALSE` (values are returned preferentially over NA).
- `compute_mean()`: calculates the mean value, excluding NA if `na.rm = TRUE`
- `compute_median()`: calculates the median value, excluding NA if `na.rm = TRUE`
- `fuse()`: collapses multiple matching records into a single string using the provided separator.
- `select_max()`: selects a record based on the index of the maximum value in a another column.
- `select_min()`: selects a record based on the index of the minimum in a second column.
- `select_match()`: returns all records based on the indices of identical matches in a second column and collapses them using the provided separator.
- `select_exact()`: returns records based on the index of identical value matching the `match` parameter within the current column, and collapses them using the provided separator if necessary.
- `fuse_unique()`: collapses a set of records to a set of unique values using the provided separator. `digits` can be provided for numeric columns to control the precision used when determining unique values.
- `prioritise()`: reduces a set of annotations by prioritising values according to the input. If there are multiple matches with the same priority then they are collapsed using a separator.
- `nothing()`: a pass-through function to allow some annotation table columns to remain unchanged.
- `count_records()`: adds a new column indicating the number of annotations that match the given grouping variable.
- `select_grade()`: returns records based on the index of the best grade in a second list. The best grade is defined as "A" for `upper_case = TRUE` or "a" for `upper_case = FALSE` and the worst grade is "Z" or "z". Any non-exact matches to a character in `LETTERS` or `letters` are replaced with NA.

Examples

```
# Select matching records
M <- combine_records(
    group_by = "example",
    default_fcn = select_exact(
        match_col = "match_column",
        match = "find_me",
        separator = ", ",
        na_string = "NA"
    )
)
```

```
# Collapse unique values
M <- combine_records(
    group_by = "example",
    default_fcn = fuse_unique(
        digits = 6,
        separator = ", ",
        na_string = "NA",
        sort = FALSE
    )
)

# Prioritise by source
M <- combine_records(
    group_by = "InChiKey",
    default_fcn = prioritise(
        match_col = "source",
        priority = c("CD", "LS"),
        separator = "  || "
    )
)

# Do nothing to all columns
M <- combine_records(
    group_by = "InChiKey",
    default_fcn = nothing()
)

# Add a column with the number of records with a matching inchikey
M <- combine_records(
    group_by = "InChiKey",
    fcns = list(
        count = count_records()
    )
)

# Select annotation with highest (best) grade
M <- combine_records(
    group_by = "InChiKey",
    default_fcn = select_grade(
        grade_col = "grade",
        keep_NA = FALSE,
        upper_case = TRUE
    )
)
```

`combine_sources` *Combine annotation sources (tables)*

Description

Annotation tables are joined and matching columns merged.

Usage

```
combine_sources(
  source_list,
  matching_columns = NULL,
  keep_cols = NULL,
  source_col = "annotation_source",
  exclude_cols = NULL,
  tag = "combined",
  as = annotation_source(name = "combined", description =
    paste0("A source created by combining two or ", "more sources")),
  ...
)
```

Arguments

source_list	(list) A list of annotation sources to be combined.
matching_columns	(character, NULL) A named vector of columns names to be created by merging columns from individual sources. e.g. `c('world'='hello')` will rename the 'hello' column to 'world' if found in any of the tables. The default is `NULL`.
keep_cols	(character, NULL) A list of column names to keep in the combined table (padded with NA) if detected in one of the input tables. Special case ".all" will keep all columns from all tables. The default is `NULL`.
source_col	(character) The column name that will be created to contain a tag to indicate which source the annotation originated from. The default is `"annotation_source"`.
exclude_cols	(NULL, character) Column names to be excluded from the merged annotation table. Note this is applied after `keep_cols`. The default is `NULL`.
tag	(character) The tag given to the newly combined table. The default is `"combined"`.
as	(annotation_source) An annotation_source object to use as the base class for the combined sources. The default is `annotation_source(name = "combined", description = paste0("A source created by combining two or ", "more sources"))`.
...	Additional slots and values passed to `struct_class`.

Value

A `combine_sources` object with the following `output` slots:

combined_table	(annotation_source) The annotation tabel after combining the input tables.

Inheritance

A `combine_sources` object inherits the following `struct` classes:

`[combine_sources]` -> `[model]` -> `[struct_class]`

Examples

```
M <- combine_sources(
        source_list = list(),
        matching_columns = NULL,
        keep_cols = NULL,
        source_col = "annotation_source",
        exclude_cols = NULL,
        tag = "combined",
        as = annotation_source())
```

CompoundDb_source *Import CompDB source*

Description

Imports the compounds table of a CompDB source as an annotation_source.

Usage

```
CompoundDb_source(source, tag = "cdb", ...)
```

Arguments

source	(ANY) The source of annotation data.
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is "cdb".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- CompoundDb

Value

A CompoundDb_source object. This object has no output slots.

Inheritance

A CompoundDb_source object inherits the following struct classes:

[CompoundDb_source] -> [annotation_source] -> [struct_class]

References

Rainer J, Vicini A, Salzer L, Stanstrup J, Badia J, Neumann S, Stravs M, Verri Hernandes V, Gatto L, Gibb S, Witting M (2022). "A Modular and Expandable Ecosystem for Metabolomics Data Annotation in R." *Metabolites*, *12*, 173. doi:10.3390/metabo12020173 https://doi.org/10.3390/metabo12020173. https://www.mdpi.com/2218-1989/12/2/173.

Examples

```
M <- CompoundDb_source(
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`compute_column` *Compute a column*

Description

Compute values for a new column based on an input column.

Usage

```
compute_column(input_columns, output_column, fcn, ...)
```

Arguments

`input_columns`	(character) The name of a column in the input table used to compute a new column.
`output_column`	(character) The name of the newply computed column.
`fcn`	(function) The function used to compute the values for the new column.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `dplyr`

Value

A `compute_column` object with the following `output` slots:

`updated`	(annotation_source) The updated annotations as an `annotation_source` object.

Inheritance

A compute_column object inherits the following struct classes:

[compute_column] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Examples

```
M <- compute_column(
        input_columns = character(0),
        output_column = character(0),
        fcn = function(){})
```

---

compute_record *Compute a value for a record*

Description

Compute values for a record based on other values in a record

Usage

```
compute_record(fcn, ...)
```

Arguments

fcn	(function) The function used to compute the values for the record.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr

Value

A compute_record object with the following output slots:

updated	(annotation_source) The updated annotations as an annotation_source object.

Inheritance

A `compute_record` object inherits the following `struct` classes:

`[compute_record] -> [model] -> [struct_class]`

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Examples

```
M <- compute_record(
        fcn = function(){})
```

`cts_lite_lookup` *Batch lookup via CTS-Lite*

Description

Uses the CTS-Lite batch REST API to match InChIKeys against a curated subset of PubChem, returning the matched PubChem entry plus literature/patent annotation counts.

Usage

```
cts_lite_lookup(
  query_column,
  suffix = "_cts",
  columns = ".all",
  top_hit_only = TRUE,
  first_block_matches = TRUE,
  rdkit_conversion = TRUE,
  cache = NULL,
  cache_mode = "update",
  ...
)
```

Arguments

`query_column`	(character) The name of a column in the annotation table containing InChIKeys (or InChI, SMILES, molecular formula or PubChem CID values - CTS-Lite auto-detects the query type).
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_cts"`.

`columns` (character) The columns to include in the result. One or more of "query_type", "found_match", "match_level", "pubchem_cid", "inchikey", "inchi", "smiles", "compound_name", "molecular_formula", "exact_mass", "literature_count", "patent_count", "annotation_type_count". Keyword ".all" (the default) returns every column. The default is `".all"`.

`top_hit_only` (logical) If TRUE (the default), CTS-Lite returns only the single best-ranked match per query (by literature/patent count). If FALSE, a query may return multiple matches, producing one row per match. The default is `TRUE`.

`first_block_matches`
(logical) If TRUE (the default), an InChIKey (or converted SMILES) query with no exact match may still match on its first 14 characters (the connectivity/skeleton block), reported as match_level = "First Block". If FALSE, only exact matches are returned. The default is `TRUE`.

`rdkit_conversion`
(logical) If TRUE (the default), a SMILES query that fails to match directly is converted to an InChIKey with RDKit and retried. Has no effect for InChIKey queries. The default is `TRUE`.

`cache` (annotation_database, NULL) A struct cache object (e.g. `rds_cache()`) that stores results of previous CTS-Lite queries, keyed by query value. Values already present in the cache are not requeried. If not using a cache then set to NULL. The default is `NULL`.

`cache_mode` (character) Cache mode. Allowed values are limited to the following:

- `"update"`: The normal mode: values already in `cache` are used as-is, anything missing is queried live and the result added to the cache.
- `"offline"`: Never query the live CTS-Lite API - only values already present in `cache` are returned, and everything else is left as NA. A warning lists how many query values are not covered by the cache when this happens.
- `"rebuild"`: Ignore any existing cached value and query the live CTS-Lite API for every value, overwriting the corresponding entry in `cache`.

The default is `"update"`.

`...` Additional slots and values passed to `struct_class`.

Details

CTS-Lite (the Fiehn Lab's successor to the original, now-closed Chemical Translation Service) matches an InChIKey (or InChI, SMILES, molecular formula or PubChem CID - the query type is auto-detected) against a curated subset of PubChem, returning the matched PubChem entry plus literature/patent annotation counts that can be used as a rough confidence signal. It does not support name-based translation - see `mwb_refmet_lookup()` or `chebi_lookup()` for that. Its REST API (`POST .../match`) is batch-based - every query value is submitted in a single request, rather than one request per value - so, unlike MetMashR's other REST API lookups, this object is not built on the `rest_api` base class.

Value

A cts_lite_lookup object with the following output slots:

updated (annotation_source) The annotation_source after adding data returned by CTS-Lite.

Inheritance

A cts_lite_lookup object inherits the following struct classes:

[cts_lite_lookup] -> [model] -> [struct_class]

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- cts_lite_lookup(
        query_column = character(0),
        suffix = "_cts",
        columns = ".all",
        top_hit_only = FALSE,
        first_block_matches = FALSE,
        rdkit_conversion = FALSE,
        base_url = "https://cts-lite.metabolomics.us/match",
        cache = NULL,
        cache_mode = "update")
```

---

database_lookup *ID lookup by database*

Description

Search a database (data.frame) for annotation matches based on values in a specified column.

Usage

```
database_lookup(
  query_column,
  database_column,
  database,
  include = NULL,
  suffix = NULL,
  not_found = NA,
  ...
)
```

Arguments

query_column	(character) The annotation table column name to use as the reference for searching the database e.g. "HMBD_ID".
database_column	(character) The database column to search for matches to the values in annoation_column.
database	(data.frame, annotation_database) A database to be searched. Can be a data.frame or a annotation_database object.
include	(character, NULL) The name of the database columns to be added to the annotations. If NULL, all columns are retained. The default is NULL.
suffix	(character, NULL) A string appended to the column names from the database. Used to distinguish columns from different databases with identical column names.If suffix = NULL then the column names are not changed. The default is NULL.
not_found	(character, numeric, logical, NULL) The returned value when there are no matches. The default is NA.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr

Value

A database_lookup object with the following output slots:

updated	(annotation_source) The input annotation_source is updated with matching columns from the database.

Inheritance

A database_lookup object inherits the following struct classes:

[database_lookup] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Examples

```
M <- database_lookup(
        query_column = "V1",
        database_column = "",
        database = data.frame(),
        include = NULL,
        suffix = NULL,
        not_found = NULL)
```

`eutils_lookup` *NCBI E-utils query*

Description

Submit a query to one of the NCBI E-utils databases. See https://www.ncbi.nlm.nih.gov/books/NBK25501/ for details.

Usage

```
eutils_lookup(query_column, database, term, result_fields = "idlist", ...)
```

Arguments

`query_column`	(character) The column name to use as the reference for searching the database e.g. "HMBD_ID".
`database`	(character) The name of the E-utils database to search. See https://www.ncbi.nlm.nih.gov/books/NBK255 for details.
`term`	(character) A correctly formated search term to use with E-utils. See https://www.ncbi.nlm.nih.gov/books/ for details. When used with the provided url template will automatically include the value from the `query_column` at the beginning of the term.
`result_fields`	(character) The name of the search result field to return. For E-utils this is often "idlist". The default is `"idlist"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `eutils_lookup` object with the following `output` slots:

`updated` (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A `eutils_lookup` object inherits the following `struct` classes:

`[eutils_lookup]` -> `[rest_api]` -> `[model]` -> `[struct_class]`

Examples

```
M <- eutils_lookup(
        database = "gene",
        term = "[pdat]",
        result_fields = "idlist",
        base_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils",
        url_template = paste0(
            "<base_url>/esearch.fcgi?db=<database>&",
            "term=<query_column><term>&retmode=json"
        ),
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

excel_database *Excel database*

Description

A data.frame imported from the sheet of an excel file

Usage

```
excel_database(
  source = character(0),
  sheet = 1,
  rowNames = FALSE,
  colNames = TRUE,
  startRow = 1,
  ...
)
```

Arguments

source	(ANY) The source of annotation data. The default is `character(0)`.
sheet	(character) The name of the sheet to import. The default is `1`.
rowNames	(logical) If TRUE, first column of data will be used as row names. The default is `FALSE`.
colNames	(logical) If TRUE, first row of data will be used as column names. The default is `TRUE`.

startRow	(numeric, integer) First row to begin looking for data. Empty rows at the top of a file are always skipped, regardless of the value of startRow. The default is 1.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- openxlsx

Value

A excel_database object. This object has no output slots.

Inheritance

A excel_database object inherits the following struct classes:

[excel_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Schauberger P, Walker A (2026). *openxlsx: Read, Write and Edit xlsx Files*. doi:10.32614/CRAN.package.openxlsx https://doi.org/10.32614/CRAN.package.openxlsx. R package version 4.2.9, https://CRAN.R-project.org/package=openxlsx.

See Also

Other annotation databases: AnnotationDb_database(), GO_database(), annotation_database(), annotation_source(), rdata_database(), rds_cache(), rds_database()

Examples

```
M <- excel_database(
        sheet = character(0),
        rowNames = FALSE,
        colNames = FALSE,
        startRow = 1,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`filter_labels` *Filter by factor labels*

Description

Removes (or includes) annotations such that the named column excludes (or includes) the specified labels.

Usage

```
filter_labels(
  column_name,
  labels,
  mode = "exclude",
  perl = FALSE,
  fixed = FALSE,
  match_na = FALSE,
  ...
)
```

Arguments

`column_name`	(character) The column name to filter.
`labels`	(character) The labels to filter by. Uses `[grepl()]` so regex is accepted e.g. for partial matching or labels.
`mode`	(character) Filter mode. Allowed values are limited to the following: • `"exclude"`: The specified labels are removed from the annotation table. • `"include"`: Only the specified labels are retained in the annotation table. The default is `"exclude"`.
`perl`	(logical) Use a Perl-compatible regex. The default is `FALSE`.
`fixed`	(logical) Use exact matching. The default is `FALSE`.
`match_na`	(logical) Match NA. Allowed values are limited to the following: • `"TRUE"`: NA values will be treated as if they matched to one of the labels. • `"FALSE"`: NA values will be treated as though they did not match to any of the labels. The default is `FALSE`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `filter_labels` object with the following `output` slots:

`filtered`	(annotation_source) The annotation_source after filtering.
`flags`	(data.frame) A list of flags indicating which annotations had a matching label.

Inheritance

A `filter_labels` object inherits the following `struct` classes:

`[filter_labels]` -> `[model]` -> `[struct_class]`

Examples

```
M <- filter_labels(
        column_name = "V1",
        labels = "",
        mode = "exclude",
        perl = FALSE,
        fixed = FALSE,
        match_na = FALSE)
```

---

filter_na *Filter by missing values*

Description

Filters annotations where the named column is NA

Usage

```
filter_na(column_name, mode = "exclude", ...)
```

Arguments

`column_name`	(character) The column name to use for filtering.
`mode`	(character) Filter mode. Allowed values are limited to the following: • `"include"`: Rows with NA are kept and all others removed. • `"exclude"`: Rows with NA are excluded and all other kept. The default is `"exclude"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `filter_na` object with the following `output` slots:

`filtered`	(annotation_source) Annotation_source after filtering.
`flags`	(data.frame) A list of flags indicating which annotations were removed.

Inheritance

A `filter_na` object inherits the following `struct` classes:

`[filter_na]` -> `[model]` -> `[struct_class]`

Examples

```
M <- filter_na(
        column_name = "V1",
        mode = "exclude")
```

`filter_range` *Filter by range*

Description

Removes annotations where the names column is greater than an upper limit or less than a lower limit.

Usage

```
filter_range(
  column_name,
  upper_limit = Inf,
  lower_limit = -Inf,
  equal_to = TRUE,
  ...
)
```

Arguments

`column_name`	(character) The column name to filter.
`upper_limit`	(numeric, integer, function) The upper limit used for filtering. Can be a value, or a function that computes a value (e.g. mean). The default is `Inf`.
`lower_limit`	(numeric, integer, function) The lower limit used for filtering. Can be a value, or a function that computes a value (e.g. mean). The default is `-Inf`.

equal_to (logical) Equal to limits. Allowed values are limited to the following:
- "TRUE": Greater/less than or equal to the limits are excluded.
- "FALSE": Greater/less than the limits are excluded.

The default is TRUE.

... Additional slots and values passed to struct_class.

Value

A filter_range object with the following output slots:

filtered (annotation_source) Annotation_source after filtering.
flags (data.frame) A list of flags indicating which annotations were removed.

Inheritance

A filter_range object inherits the following struct classes:

[filter_range] -> [model] -> [struct_class]

Examples

```
M <- filter_range(
        column_name = "V1",
        upper_limit = Inf,
        lower_limit = -Inf,
        equal_to = FALSE)
```

---

filter_records *Filter rows*

Description

A wrapper around dplyr::filter. Select rows from an annotation table using tidy grammar.

Usage

```
filter_records(where = wherever(A > 0), ...)
```

Arguments

where (quosures) A list of rlang::quosure for evaluation e.g. A>10 willselect all rows where the values in column A are greater than10. A helper function wherever is provided to generatea suitable list of quosures. The default is wherever(A > 0).

... Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr
- rlang

Value

A filter_records object with the following output slots:

updated (annotation_source) The updated annotations as an annotation_source object.

Inheritance

A filter_records object inherits the following struct classes:

[filter_records] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Henry L, Wickham H (2026). *rlang: Functions for Base Types and Core R and 'Tidyverse' Features*. doi:10.32614/CRAN.package.rlang https://doi.org/10.32614/CRAN.package.rlang. R package version 1.3.0, https://CRAN.R-project.org/package=rlang.

See Also

dplyr::filter()

wherever()

Examples

```
M <- filter_records(
        where = wherever(A>10))
```

`filter_venn` *Filter by factor levels*

Description

Removes (or includes) annotations such that the named column excludes (or includes) the specified intersection levels. Supports any number of groups using intersection-based filtering. If no levels are specified, all available intersection levels will be returned for inspection. If invalid levels are specified, a warning will be shown with the list of valid levels.

Usage

```
filter_venn(
  factor_name,
  group_column = NULL,
  tables = NULL,
  filter = NULL,
  mode = "include",
  ...
)
```

Arguments

`factor_name`	(character) The name of the column(s) in the `annotation_source` to generate intersection groups from. Supports any number of columns for intersection-based filtering.
`group_column`	(character, NULL) The name of the column in the `annotation_source` to create groups from in the Venn diagram. This parameter is ignored if `!is.null(tables)`, as each table is considered to be a group. This parameter is also ignored if more than one `factor_name` is provided, as each column is considered a group. The default is `NULL`.
`tables`	(list, NULL) A list of `annotation_sources` to generate the venn groups from. If the only table of interest is the table coming in from `model_apply` then set `tables = NULL` and use `group_column`. The default is `NULL`.
`filter`	(function, NULL) A function to filter intersections based on their properties. The function should take region_data as input and return a logical vector indicating which intersections to keep. Use upset_intersections(), upset_min_size(), upset_min_groups(), upset_max_groups(), or create custom filter functions. The default is `NULL`.
`mode`	(character) Filter mode. Allowed values are limited to the following: • `"include"`: Only items that appear in the filtered intersections are kept in the output. • `"exclude"`: Items that appear in the filtered intersections are removed from the output. The default is `"include"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `filter_venn` object with the following `output` slots:

`filtered`	(annotation_source) Annotation_source after filtering.
`flags`	(data.frame) A list of flags indicating which annotations were removed.

Inheritance

A `filter_venn` object inherits the following `struct` classes:

`[filter_venn]` -> `[model]` -> `[struct_class]`

Examples

```
M <- filter_venn(
        factor_name = "V1",
        group_column = NULL,
        tables = NULL,
        filter = NULL,
        mode = "include")
```

---

github_file *GitHub file*

Description

Uses the GitHub REST API to retrieve a file from a specifiedGitHub repository.

Usage

```
github_file(
  username,
  repository_name,
  file_path,
  bfc_path = NULL,
  resource_name = paste(username, repository_name, file_path, sep = "_"),
  ...
)
```

Arguments

`username`	(character) The GitHub username to retireve the file from.
`repository_name`	(character) The name of a repository for the specified GitHub usernamethat contains the file to download.

file_path	(character) The path to the file to download within the specified GitHub repository.
bfc_path	(character, NULL) BiocFileCache is used to cache the database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache() for details). The default is NULL.
resource_name	(character) The name given to this resource in the cache. (see BiocFileCache::BiocFileCache() for details). The default is paste(username, repository_name, file_path, sep = "_").
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache
- httr

Value

A github_file object. This object has no output slots.

Inheritance

A github_file object inherits the following struct classes:

[github_file] -> [BiocFileCache_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

Examples

```
M <- github_file(
        username = character(0),
        repository_name = character(0),
        file_path = character(0),
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
```

```
        data = data.frame(),
        source = "ANY")
```

GO_database *GO.db*

Description

Retrieve a table from the Gene Ontology using the GO.db package.

Usage

```
GO_database(source = "GO.db", table = "GOBPOFFSPRING", ...)
```

Arguments

source	(character) The name of an AnnotationDb package to import the specified table from. Note the package should already be installed. The default is "GO.db".
table	(character) The name of a table to import from the GO.db package. Allowed tables include: GOBPANCESTOR,GOBPPARENTS,GOBPCHILDREN,GOBPOFFSPRING (and their CC or MF equivalents), GOTERM, GOSYNONYM, GOOBSOLETE. The default is "GOBPOFFSPRING".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- GO.db

Value

A GO_database object. This object has no output slots.

Inheritance

A GO_database object inherits the following struct classes:

[GO_database] -> [AnnotationDb_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Carlson M (2026). *GO.db: A set of annotation maps describing the entire Gene Ontology*. R package version 3.23.1.

Pagès H, Carlson M, Falcon S, Li N (2026). *AnnotationDbi: Manipulation of SQLite-based annotations in Bioconductor*. doi:10.18129/B9.bioc.AnnotationDbi https://doi.org/10.18129/B9.bioc.AnnotationDbi. R package version 1.74.0, https://bioconductor.org/packages/AnnotationDbi.

See Also

GO.db

Other annotation databases: AnnotationDb_database(), annotation_database(), annotation_source(), excel_database(), rdata_database(), rds_cache(), rds_database()

Examples

```
M <- GO_database(
        table = "GOBPCHILDREN",
        tag = character(0),
        data = data.frame(),
        source = character(0))
```

---

greek_dictionary *Greek dictionary*

Description

A dictionary for converting Greek characters to Romanised names. It is intended for use with the normalise_strings() object.

Usage

```
greek_dictionary
```

Value

A dictionary for use with normalise_strings()

Examples

```
M <- normalise_strings(
    search_column = "example",
    output_column = "result",
    dictionary = greek_dictionary
)
```

`hmdb_lookup` *Compound ID lookup via pubchem*

Description

Requests HMBD records based on HMDB identifiers.

Usage

```
hmdb_lookup(query_column, suffix = "_hmdb", output = "inchikey", ...)
```

Arguments

`query_column`	(character) The name of a column in the annotation table containing values to search in the api call.
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_hmdb"`.
`output`	(character) The value returned from the HMDB xml. The default is `"inchikey"`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `XML`

Value

A `hmdb_lookup` object with the following `output` slots:

`updated`	(annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A `hmdb_lookup` object inherits the following `struct` classes:

`[hmdb_lookup]` -> `[rest_api]` -> `[model]` -> `[struct_class]`

References

Temple Lang D (2026). *XML: Tools for Parsing and Generating XML Within R and S-Plus*. doi:10.32614/CRAN.package.XML https://doi.org/10.32614/CRAN.package.XML. R package version 3.99-0.25, https://CRAN.R-project.org/package=XML.

Examples

```
M <- hmdb_lookup(
        output = "inchikey",
        base_url = "http://www.hmdb.ca/metabolites",
        url_template = "<base_url>/<query_column>.xml",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

id_counts *id counts*

Description

Adds the number of times an identical identifier is present to each record.

Usage

```
id_counts(id_column, count_column = "id_counts", count_na = TRUE, ...)
```

Arguments

id_column	(character) Column name of the variable ids in variable_meta.
count_column	(character) The name of the new column to store the counts in. The default is "id_counts".
count_na	(logical) Count NA. Allowed values are limited to the following: • "TRUE": Report number of NA. • "FALSE": Do not report number of NA. The default is TRUE.
...	Additional slots and values passed to struct_class.

Value

A id_counts object with the following output slots:

updated	(annotation_source) The input annotation source with the newly generated column.

Inheritance

A id_counts object inherits the following struct classes:

[id_counts] -> [model] -> [struct_class]

Examples

```
M <- id_counts(
        id_column = character(0),
        count_column = character(0),
        count_na = FALSE)
```

`import_source` *Import_source*

Description

A wrapper for `read_source()` that can be used in an annotation workflow to import an annotation source.

Usage

```
import_source(...)
```

Arguments

`...`	Additional slots and values passed to `struct_class`.

Value

A `import_source` object with the following `output` slots:

`imported` (annotation_source) The `annotation_source` after importing the data.

Inheritance

A `import_source` object inherits the following `struct` classes:

`[import_source]` -> `[model]` -> `[struct_class]`

Examples

```
M <- import_source()
```

is_writable *Is database writable*

Description

A function that returns TRUE if the database has been designed for use in read and write mode.

Usage

```
is_writable(obj, ...)

## S4 method for signature 'annotation_database'
is_writable(obj)

## S4 method for signature 'rdata_database'
is_writable(obj)
```

Arguments

obj	A annotation_database object
...	additional database specific inputs

Value

TRUE if the database is writable; FALSE otherwise. This method does not check file properties, only the intended usage of the object.

Examples

```
M <- annotation_database()
is_writable(M)
```

kegg_lookup *Convert to or from kegg identifiers*

Description

Searches the Kegg database to obtain external identifiers. KEGG compound, drug and glycan databases can be queried for pubchem and chebi identifiers, and vice-versa.

Usage

```
kegg_lookup(
  get = "pubchem_sid",
  from = "compound",
  query_column,
  suffix = "_kegg",
  cache = NULL,
  cache_mode = "update",
  ...
)
```

Arguments

get	(character) Get identifier. Allowed values are limited to the following: • "compound": KEGG small molecule database. • "glycan": KEGG glycan database. • "drug": KEGG drug database. • "chebi": Chemical Entities of Biological Interest (ChEBI) database. • "pubchem_sid": PubChem Substance Identifier. The default is "pubchem_sid".
from	(character) From identifier. Allowed values are limited to the following: • "compound": KEGG small molecule database. • "glycan": KEGG glycan database. • "drug": KEGG drug database. • "chebi": Chemical Entities of Biological Interest (ChEBI) database. • "pubchem_sid": PubChem Substance Identifier. The default is "compound".
query_column	(character) The name of the column containing identifiers to search the database for. They should be identifiers of the type selected for the "from" slot.
suffix	(character) A suffix appended to all column names in the returned result. The default is "_kegg".
cache	(annotation_database, NULL) A struct cache object (e.g. rds_cache()) that stores results of previous KEGG conversions, keyed by query value. Values already present in the cache are not requeried. If not using a cache then set to NULL. The default is NULL.
cache_mode	(character) Cache mode. Allowed values are limited to the following: • "update": The normal mode: values already in cache are used as-is, anything missing is queried live and the result added to the cache. • "offline": Never query the live KEGG API. Only values already present in cache are returned, and everything else is left as NA. • "rebuild": Ignore any existing cached value and query the live KEGG API for every value, overwriting the corresponding entry in cache. Useful when cached results are known to be stale. The default is "update".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- KEGGREST
- dplyr

Value

A kegg_lookup object with the following output slots:

updated (annotation_source) An annotation_source object with a new column of compound identifiers.

Inheritance

A kegg_lookup object inherits the following struct classes:

[kegg_lookup] -> [model] -> [struct_class]

References

Tenenbaum D, Maintainer B (2026). *KEGGREST: Client-side REST access to the Kyoto Encyclopedia of Genes and Genomes (KEGG)*. doi:10.18129/B9.bioc.KEGGREST https://doi.org/10.18129/B9.bioc.KEGGREST. R package version 1.52.2, https://bioconductor.org/packages/KEGGREST.

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), cts_lite_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- kegg_lookup(
        get = "pubchem_sid",
        from = "compound",
        query_column = "V1",
        suffix = "_kegg",
        cache = NULL,
        cache_mode = "update")
```

`lcms_table` *LCMS table*

Description

An LCMS table extends `annotation_table()` to represent annotation data for an LCMS experiment. Columns representing m/z and retention time are required for an `lcms_table`.

Usage

```
lcms_table(
  data = NULL,
  tag = "",
  id_column = "id",
  mz_column = "mz",
  rt_column = "rt",
  ...
)
```

Arguments

`data`	(data.frame, NULL) A data.frame of annotation data. The default is `NULL`.
`tag`	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is `""`.
`id_column`	(character) The column name of the annotation data.frame containing row identifers. If NULL This will be generated automatically. The default is `"id"`.
`mz_column`	(character) The column name of the annotation data.frame containing m/z values. The default is `"mz"`.
`rt_column`	(character) The column name of the annotation data.frame containing retention time values. The default is `"rt"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `lcms_table` object. This object has no `output` slots.

Inheritance

A `lcms_table` object inherits the following `struct` classes:

`[lcms_table]` -> `[annotation_table]` -> `[annotation_source]` -> `[struct_class]`

Examples

```
M <- lcms_table(
        mz_column = "mz",
        rt_column = "rt",
        id_column = "id",
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`lipidmaps_database` *LIPID MAPS database*

Description

Imports the full LIPID MAPS Structure Database (LMSD) bulk compound export (~50,000 lipids), cached locally with BiocFileCache so it is only downloaded once rather than queried per-row. Columns include lm_id, name, abbrev, core, main_class, sub_class, formula, inchi, inchi_key, kegg_id, hmdb_id, chebi_id, pubchem_cid and smiles – a single database_lookup() against this table can replace many individual lipidmaps_lookup() REST queries.

Usage

```
lipidmaps_database(
  bfc_path = NULL,
  resource_name = "MetMashR_lipidmaps",
 source = paste0("https://www.lipidmaps.org/rest/compound/lm_id/LM/all/download"),
  ...
)
```

Arguments

`bfc_path`	(character, NULL) `BiocFileCache` is used to cache the database locally and prevent unnecessary downloads. If a path is provided then `BiocFileCache` will use this location. If NULL it will use the default location (see `BiocFileCache::BiocFileCache()` for details). The default is `NULL`.
`resource_name`	(character) The name given to this resource in the cache. (see `BiocFileCache::BiocFileCache()` for details). The default is `"MetMashR_lipidmaps"`.
`source`	(ANY) The source of annotation data. The default is `paste0("https://www.lipidmaps.org/rest/com`
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `BiocFileCache`

Value

A `lipidmaps_database` object. This object has no `output` slots.

Inheritance

A `lipidmaps_database` object inherits the following `struct` classes:

`[lipidmaps_database]` -> `[BiocFileCache_database]` -> `[annotation_database]` -> `[annotation_source]` -> `[struct_class]`

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

See Also

Other database: `BiocFileCache_database()`, `sqlite_database()`

Examples

```
M <- lipidmaps_database(
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`lipidmaps_lookup` *LipidMaps api lookup*

Description

Search the LipidMaps database using the API

Usage

```
lipidmaps_lookup(
  query_column,
  context,
  context_item,
  output_item = "all",
  suffix = "_lipidmaps",
  ...
)
```

Arguments

query_column	(character) The name of a column in the annotation table containing values to search in the api call.
context	(character) The search API context. Must be one of "compound", "gene", or "protein".
context_item	(character) The context item being searched. See https://lipidmaps.org/resources/rest for details.
output_item	(character) The names of the columns to return from the results of the search. See https://lipidmaps.org/resources/rest for details. The default is "all".
suffix	(character) A suffix appended to all column names in the returned result. The default is "_lipidmaps".
...	Additional slots and values passed to struct_class.

Value

A lipidmaps_lookup object with the following output slots:

updated	(annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A lipidmaps_lookup object inherits the following struct classes:

[lipidmaps_lookup] -> [rest_api] -> [model] -> [struct_class]

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), cts_lite_lookup(), kegg_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- lipidmaps_lookup(
        query_column = character(0),
        output_item = "input",
        context = "compound",
        context_item = character(0),
        base_url = "https://www.lipidmaps.org/rest",
     url_template = "<base_url>/<context>/<context_item>/<query_column>/<output_item>/json",
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

`ls_source` *LCMS table*

Description

An LCMS table extends `annotation_table()` to represent annotation data for an LCMS experiment. Columns representing m/z and retention time are required for an `lcms_table`.

Usage

```
ls_source(
  source,
  tag = "LS",
  mz_column = "mz",
  rt_column = "rt",
  id_column = "id",
  data = NULL,
  ...
)
```

Arguments

`source`	(ANY) The source of annotation data.
`tag`	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is `"LS"`.
`mz_column`	(character) The column name of the annotation data.frame containing m/z values. The default is `"mz"`.
`rt_column`	(character) The column name of the annotation data.frame containing retention time values. The default is `"rt"`.
`id_column`	(character) The column name of the annotation data.frame containing row identifers. If NULL This will be generated automatically. The default is `"id"`.
`data`	(data.frame, NULL) A data.frame of annotation data. The default is `NULL`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `ls_source` object. This object has no `output` slots.

Inheritance

A `ls_source` object inherits the following `struct` classes:

`[ls_source]` -> `[lcms_table]` -> `[annotation_table]` -> `[annotation_source]` -> `[struct_class]`

See Also

Other annotation sources: annotation_database(), annotation_table(), cd_source(), mspurity_source(), mwb_study_source()

Other annotation tables: annotation_table(), cd_source(), mspurity_source()

Examples

```
M <- ls_source(
        mz_column = "mz",
        rt_column = "rt",
        id_column = "id",
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

model_apply,model,annotation_source-method
Apply method

Description

Applies method to the input DatasetExperiment. Some models also provide separate model_train and model_predict methods.

Usage

```
model_apply(M, D, ...)
model_train(M, D, ...)
model_predict(M, D, ...)
```

Arguments

M	a method object
D	another object used by the first
...	additional inputs (not used)

Value

Returns a modified method object

Examples

```
M <- example_model()
M <- model_apply(M, iris_DatasetExperiment())
```

mspurity_source *msPurity source*

Description

An annotation source for importing an annotation table from the format created by the msPurity package.

Usage

```
mspurity_source(
  source,
  tag = "msPurity",
  mz_column = "mz",
  rt_column = "rt",
  id_column = "id",
  data = NULL,
  ...
)
```

Arguments

source	(ANY) The source of annotation data.
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is "msPurity".
mz_column	(character) The column name of the annotation data.frame containing m/z values. The default is "mz".
rt_column	(character) The column name of the annotation data.frame containing retention time values. The default is "rt".
id_column	(character) The column name of the annotation data.frame containing row identifers. If NULL This will be generated automatically. The default is "id".
data	(data.frame, NULL) A data.frame of annotation data. The default is NULL.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- msPurity

Value

A mspurity_source object. This object has no output slots.

Inheritance

A `mspurity_source` object inherits the following `struct` classes:

`[mspurity_source] -> [lcms_table] -> [annotation_table] -> [annotation_source] -> [struct_class]`

References

Lawson, Nigel T, Weber, M. RJ, Jones, R. M, Chetwynd, J. A, Blanco R, Alejandro G, Guida D, Riccardo, Viant, R. M, Dunn, B W (2017). "msPurity: Automated Evaluation of Precursor Ion Purity for Mass Spectrometry-Based Fragmentation in Metabolomics." *Analytical Chemistry*, *89*, 2432-2439. doi:10.1021/acs.analchem.6b04358 https://doi.org/10.1021/acs.analchem.6b04358.

See Also

Other annotation sources: `annotation_database()`, `annotation_table()`, `cd_source()`, `ls_source()`, `mwb_study_source()`

Other annotation tables: `annotation_table()`, `cd_source()`, `ls_source()`

Examples

```
M <- mspurity_source(
        mz_column = "mz",
        rt_column = "rt",
        id_column = "id",
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

MTox700plus_database *MTox700plus_database*

Description

Imports the MTox700+ database, which is made available under the ODC Attribution License. MTox700+ is a list of toxicologically relevant metabolites derived from publications, public databases and relevant toxicological assays.

Usage

```
MTox700plus_database(
  version = "latest",
  bfc_path = NULL,
  resource_name = "MetMashR_MTox700plus",
  ...
)
```

Arguments

version	(character) The version number of the MTox700+ database to import. Available versions are listed here. version should match the tag of the release e.g. "v1.0". For convenience version = "latest" will always retrieve the most recent release. To prevent unecessary downloads BiocFileCache is used to store a local copy. The default is "latest".
bfc_path	(character, NULL) BiocFileCache is used to cache the database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache() for details). The default is NULL.
resource_name	(character) The name given to this resource in the cache. (see BiocFileCache::BiocFileCache() for details). The default is "MetMashR_MTox700plus".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache
- httr

Value

A MTox700plus_database object. This object has no output slots.

Inheritance

A MTox700plus_database object inherits the following struct classes:

[MTox700plus_database] -> [BiocFileCache_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

Sostare E, Lawson TN, Saunders LR, Colbourne JK, Weber RJM, Sobanski T, Viant MR (2022). "Knowledge-Driven Approaches to Create the MTox700+ Metabolite Panel for Predicting Toxicity." *Toxicological Sciences*, *186*, 208-220. doi:10.1093/toxsci/kfac007 https://doi.org/10.1093/toxsci/kfac007.

Examples

```
M <- MTox700plus_database(
        version = "v1.0",
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

mwb_compound_lookup *Convert to/from kegg identifiers*

Description

Searches MetabolomicsWorkbench for compound identifiers.

Usage

```
mwb_compound_lookup(
  input_item = "inchi_key",
  query_column,
  output_item = "pubchem_id",
  suffix = "_mwb",
  ...
)
```

Arguments

input_item	(character) A valid input item for the compound context (see https://www.metabolomicsworkbench.org/too The values in the query_column should be of this type. The default is "inchi_key".
query_column	(character) The name of a column in the annotation table containing values to search in the api call.
output_item	(character) A comma separated list of Valid output items for the compound context (see https://www.metabolomicsworkbench.org/tools/mw_rest.php). The default is "pubchem_id".
suffix	(character) A suffix appended to all column names in the returned result. The default is "_mwb".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- metabolomicsWorkbenchR
- dplyr

Value

A mwb_compound_lookup object with the following output slots:

updated (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A mwb_compound_lookup object inherits the following struct classes:

[mwb_compound_lookup] -> [rest_api] -> [model] -> [struct_class]

References

Lloyd GR, Weber RJM (2026). *metabolomicsWorkbenchR: Metabolomics Workbench in R*. R package version 1.22.0.

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), cts_lite_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_refmet_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- mwb_compound_lookup(
        input_item = "inchi_key",
        output_item = "inchi_key",
        base_url = "https://www.metabolomicsworkbench.org/rest",
        url_template = "<base_url>/compound/<input_item>/<query_column>/<output_item>",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

mwb_refmet_database *mwb_refmet_database*

Description

Imports the Metabolomics Workbench refmet database.

Usage

```
mwb_refmet_database(bfc = NULL, ...)
```

Arguments

bfc	(character) BiocFileCache is used to cache database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache for details). The default is NULL.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache
- httr
- plyr

Value

A mwb_refmet_database object. This object has no output slots.

Inheritance

A mwb_refmet_database object inherits the following struct classes:

[mwb_refmet_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

Wickham H (2011). "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software*, *40*(1), 1-29. https://www.jstatsoft.org/v40/i01/.

Examples

```
M <- mwb_refmet_database(
        bfc = character(0),
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

`mwb_refmet_lookup` *Metabolomics Workbench RefMet lookup*

Description

Matches a reported compound name (which may be a synonym, conjugate acid/base form, or otherwise non-canonical name) to its RefMet standardised name and identifiers, using the Metabolomics Workbench RefMet REST API. Intended as a substitute for the (now defunct) Chemical Translation Service (CTS): the original 'PubChem Identifier Exchange + CTS' translation route described by some published metabolite-merging methods can no longer be reproduced, since CTS has been shut down and its replacement, CTS-Lite, does not support name-based lookups. RefMet's `match` endpoint plays a similar synonym-resolution role, and its curated picks (e.g. preferring the biologically-relevant stereoisomer/protonation state over a generic entry) often differ usefully from a plain PubChem name search.

Usage

```
mwb_refmet_lookup(query_column, suffix = "_refmet", ...)
```

Arguments

`query_column`	(character) The name of a column in the annotation table containing values to search in the api call.
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_refmet"`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `httr`

Value

A `mwb_refmet_lookup` object with the following `output` slots:

`updated` (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A `mwb_refmet_lookup` object inherits the following `struct` classes:

`[mwb_refmet_lookup]` -> `[rest_api]` -> `[model]` -> `[struct_class]`

References

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), cts_lite_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), pubchem_id_exchange(), rest_api()

Examples

```
M <- mwb_refmet_lookup(
        base_url = "https://www.metabolomicsworkbench.org/rest/refmet",
        url_template = "<base_url>/match/<query_column>/name",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_refmet")
```

---

mwb_structure *MWB molecular structure*

Description

Query the Metabolomic Workbench API and retrieve a display an image of the matching molecular structure.

Usage

```
mwb_structure(query_column, row_index, ...)
```

Arguments

query_column	(character) The name of the annotation_source column with regno compound identifiers.
row_index	(integer, numeric) The row index of the annotation_source to request an image of the molecular structure of.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- cowplot
- metabolomicsWorkbenchR

This object queries the Metabolomics Workbench API for matches to your query without caching the results. It is therefore intended for limited use. If you wish to obtain images for a large number of molecules you should seek an alternative solution.

Value

A mwb_structure object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A mwb_structure object inherits the following struct classes:

[mwb_structure] -> [chart] -> [struct_class]

References

Wilke C (2025). *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*. doi:10.32614/CRAN.package.cowplot https://doi.org/10.32614/CRAN.package.cowplot. R package version 1.2.0, https://CRAN.R-project.org/package=cowplot.

Lloyd GR, Weber RJM (2026). *metabolomicsWorkbenchR: Metabolomics Workbench in R*. R package version 1.22.0.

Examples

```
M <- mwb_structure(
        row_index = 1,
        query_column = "V1")
```

---

mwb_study_source *Metabolomics Workbench study source*

Description

Imports the reported metabolite list for a Metabolomics Workbench study, using the metabolomicsWorkbenchR package. Only the identifiers a study depositor chose to report are returned as-is (e.g. metabolite_name, and where present refmet_name/pubchem_id/other_id) - no translation or enrichment is performed here. To reproduce the scenario of annotation software output that has names but no standardised identifiers (the case MetMashR's mashing steps are designed for), downstream workflow steps should use only the metabolite_name column and treat any pre-existing refmet_name/pubchem_id/other_id values as optional validation data, not as workflow input.

Usage

```
mwb_study_source(source, tag = "MWB", analysis_id = NULL, data = NULL, ...)
```

Arguments

source	(character) A Metabolomics Workbench study identifier e.g. "ST001039".
tag	(character) A (short) character string that is used to represent this source e.g. in column names or source columns when used in a workflow. The default is "MWB".
analysis_id	(character, NULL) Optionally restrict the imported metabolites to one or more specific analysis ids within the study (e.g. a single LC-MS assay). If NULL (the default), metabolites for all analyses in the study are returned. The default is NULL.
data	(data.frame, NULL) A data.frame of annotation data. The default is NULL.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- metabolomicsWorkbenchR

Value

A mwb_study_source object. This object has no output slots.

Inheritance

A mwb_study_source object inherits the following struct classes:

[mwb_study_source] -> [annotation_source] -> [struct_class]

References

Lloyd GR, Weber RJM (2026). *metabolomicsWorkbenchR: Metabolomics Workbench in R*. R package version 1.22.0.

See Also

Other annotation sources: annotation_database(), annotation_table(), cd_source(), ls_source(), mspurity_source()

Examples

```
M <- mwb_study_source(
        analysis_id = NULL,
        tag = character(0),
        data = data.frame(),
        source = character(0))
```

mz_match *mz matching*

Description

Annotations will be matched to the measured data variable meta data.frame by determining which annotations ppm window overlaps with the ppm window from the measured mz.

Usage

```
mz_match(variable_meta, mz_column, ppm_window, id_column, ...)
```

Arguments

variable_meta	(data.frame) A data.frame of variable IDs and their corresponding mz values.
mz_column	(character) Column name of the mz values in variable_meta.
ppm_window	(numeric, integer) Ppm window to use for matching. If a single value is provided then the same ppm is used for both variable meta and the annotations. A named vector can also be provided e.g. c("variable_meta"=5,"annotations"=2) to use ", "different windows for each data table.
id_column	(character) Column name of the variable ids in variable_meta. id_column="rownames" will use the rownames as ids.
...	Additional slots and values passed to struct_class.

Value

A mz_match object with the following output slots:

updated	(annotation_source) The input annotation source with the newly generated column.

Inheritance

A mz_match object inherits the following struct classes:

[mz_match] -> [model] -> [struct_class]

Examples

```
M <- mz_match(
        variable_meta = data.frame(),
        mz_column = character(0),
        ppm_window = 5,
        id_column = character(0))
```

`mzrt_match` *mz matching*

Description

Annotations will be matched to the measured data variable meta data.frame by determining which annotations ppm AND rt windows overlap with the ppm AND rt windows of the measured mz.

Usage

```
mzrt_match(
  variable_meta,
  mz_column,
  rt_column,
  ppm_window,
  rt_window,
  id_column,
  ...
)
```

Arguments

`variable_meta`	(data.frame) A data.frame of variable IDs and their corresponding mz values.
`mz_column`	(character) Column name of the mz values in variable_meta.
`rt_column`	(character) Column name of the rt values in variable_meta.
`ppm_window`	(numeric, integer) Ppm window to use for matching. If a single value is provided then the same ppm is used for both variable meta and the annotations. A named vector can also be provided e.g. c("variable_meta"=5,"annotations"=2) to use different ", "windows for each data table.
`rt_window`	(numeric, integer) Rt window to use for matching. If a single value is provided then the same rt is used for both variable meta and the annotations. A named vector can also be provided e.g. c("variable_meta"=5,"annotations"=2) to use different ", "windows for each data table.
`id_column`	(character) Column name of the variable ids in variable_meta. ", "id_column="rownames" will use the rownames as ids.
`...`	Additional slots and values passed to `struct_class`.

Value

A `mzrt_match` object with the following `output` slots:

`updated`	(annotation_source) The input annotation source with the newly generated column.

Inheritance

A mzrt_match object inherits the following struct classes:

[mzrt_match] -> [model] -> [struct_class]

Examples

```
M <- mzrt_match(
        variable_meta = data.frame(),
        mz_column = character(0),
        ppm_window = 5,
        id_column = character(0),
        rt_column = character(0),
        rt_window = 20)
```

---

normalise_lipids *Normalise Lipids nomenclature*

Description

Normalises differently formated lipid names to a consistent format.

Usage

```
normalise_lipids(
  column_name,
  grammar = ".all",
  columns = ".all",
  suffix = "_goslin",
  batch_size = 10000,
  ...
)
```

Arguments

column_name	(character) The name of the column containing Lipids names to normalise.
grammar	(character) The grammar to use for normalising lipid names. Allowed values are: Shorthand2020, Goslin, FattyAcids, LipidMaps, SwissLipids, HMDB or .all. The default is ".all".
columns	(character) Column names to include from the goslin output. Can be any of "Normalized.Name", "Original.Name", "Grammar", "Adduct", "Adduct.Charge", "Lipid.Maps.Category", "Lipid.Maps.Main.Class", "Species.Name", "Extended.Species.Name", "Molecular.Species.Name", "Sn.Position.Name", "Structure.Defined.Name", "Full.Structure.Name", "Functional.Class.Abbr", "Functional.Class.Synonyms", "Level", "Total.C", "Total.OH", "Total.O", "Total.DB", "Mass", "Sum.Formula"."all" will return all columns. ". The default is ".all".

suffix	(character) A suffic added to the column names of the goslin output. The default is "_goslin".
batch_size	(numeric, integer) The maximum number of annotations to be parsed by rgoslin at a time. If the batch size is less than the total number of records then the records will be split into multiple batches to help prevent crashes. The default is 10000.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- rgoslin

Value

A normalise_lipids object with the following output slots:

updated	(annotation_source) Annotation_source after normalising lipid names.

Inheritance

A normalise_lipids object inherits the following struct classes:

[normalise_lipids] -> [model] -> [struct_class]

References

Kopczynski D, Hoffmann N, Peng B, Ahrends R (2020). "Goslin: A Grammar of Succinct Lipid Nomenclature." *Analytical Chemistry*, *92*(16), 10957-10960. https://pubs.acs.org/doi/10.1021/acs.analchem.0c01690.

Examples

```
M <- normalise_lipids(
        column_name = "V1",
        grammar = ".all",
        columns = ".all",
        suffix = "_goslin",
        batch_size = 10000)
```

normalise_strings *Normalise string*

Description

Replace matching (sub)strings based on a provided dictionary of search terms and their replacements.

Usage

```
normalise_strings(
  search_column,
  output_column = NULL,
  dictionary = list(),
  ...
)
```

Arguments

search_column	(character) The column name of the input annotation_source that will be searched for matching (sub)strings.
output_column	(character, NULL) The name of a new column that the modified strings will be stored in. If NULL the search_column will be replaced. The default is NULL.
dictionary	(list, annotation_database) A list of patterns and functions that take the input pattern and return a replacement string. A annotation_database object containing a suitable list can also be used here. The default is list().
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr

Each item of the dictionary list should #' have at least two fields: "pattern" and "replace". "pattern" is used as inputs to the [grepl()] function to detect matches to the input pattern. Parameters such as perl = TRUE can also be included in the list and these will be passed to [grepl()], otherwise the defaults are used. When a match is detected the function in "replace" is called with the same inputs as [grepl()]. The "replace" function should return a new string. Alternatively replace = NA can be used to return NA for a matching pattern. If a character string is provided then [gsub()] will be used by default.

Value

A normalise_strings object with the following output slots:

updated	(annotation_source) The updated annotations as an annotation_source object.

Inheritance

A `normalise_strings` object inherits the following `struct` classes:

`[normalise_strings] -> [model] -> [struct_class]`

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

See Also

`grepl()`, `gsub()`

Examples

```
M <- normalise_strings(
        search_column = character(0),
        output_column = NULL,
        dictionary = list())
```

`opsin_lookup` *Compound ID lookup via OPSIN*

Description

Uses the OPSIN API to search for identifers based on the input annotation column.

Usage

```
opsin_lookup(query_column, suffix = "_opsin", output = "cids", ...)
```

Arguments

`query_column`	(character) The column name to use as the reference for searching the database e.g. "compound_name". OPSIN expect molecule names as input.
`suffix`	(character) A suffix appended to all column names in the returned result. The default is `"_opsin"`.
`output`	(character) The value returned from the pubchem database. The default is `"cids"`.
`...`	Additional slots and values passed to `struct_class`.

Value

A opsin_lookup object with the following output slots:

updated (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A opsin_lookup object inherits the following struct classes:

[opsin_lookup] -> [rest_api] -> [model] -> [struct_class]

References

Lowe, M. D, Corbett, T. P, Murray-Rust, Peter, Glen, C. R (2011). "Chemical Name to Structure: OPSIN, an Open ", "Source Solution." *Journal of Chemical Information and Modeling*, *51*(3), 793-753. doi:10.1021/ci100384d https://doi.org/10.1021/ci100384d.

Examples

```
M <- opsin_lookup(
        output = "stdinchikey",
        base_url = "https://opsin.ch.cam.ac.uk/opsin",
        url_template = "<base_url>/<query_column>.<output>",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

PathBank_metabolite_database

PathBank_metabolite_database

Description

Imports the PathBank database (https://pathbank.org/) of metabolites linked to pathways.

Usage

```
PathBank_metabolite_database(
  version = "primary",
  bfc_path = NULL,
  resource_name = "MetMashR_PathBank",
  ...
)
```

Arguments

version	(character) PathBank version. Allowed values are limited to the following: • "": The version of the PatchBank database to import. To prevent unecessary downloads BiocFileCache is used to store a local copy. • "complete": The complete PathBank metabolite database. • "primary": The PathBank metabolite database for primary pathways only. The default is "primary".
bfc_path	(character, NULL) BiocFileCache is used to cache the database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache() for details). The default is NULL.
resource_name	(character) The name given to this resource in the cache. (see BiocFileCache::BiocFileCache() for details). The default is "MetMashR_PathBank".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache
- httr

Value

A PathBank_metabolite_database object. This object has no output slots.

Inheritance

A PathBank_metabolite_database object inherits the following struct classes:

[PathBank_metabolite_database] -> [BiocFileCache_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

Wickham H (2026). *httr: Tools for Working with URLs and HTTP*. doi:10.32614/CRAN.package.httr https://doi.org/10.32614/CRAN.package.httr. R package version 1.4.9, https://CRAN.R-project.org/package=httr.

Wishart, S D, Li, Carin, Marcu, Ana, Badran, Hasan, Pon, Allison, Budinski, Zachary, Patron, Jonas, Lipton, Debra, Cao, Xuan, Oler, Eponine, Li, Krissa, Paccoud, Maïlys, Hong, Chelsea, Guo, C A, Chan, Christopher, Wei, William, Ramirez-Gaona, Miguel (2019). "PathBank: a comprehensive pathway database for model organisms." *Nucleic Acids Research*, *48*, D470-D478. doi:10.1093/nar/gkz861 https://doi.org/10.1093/nar/gkz861.

Examples

```
M <- PathBank_metabolite_database(
        version = "primary",
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

pivot_columns *Pivot longer*

Description

Combine multiple groups of columns into a single group of columns with group labels.

Usage

```
pivot_columns(column_groups, group_labels, ...)
```

Arguments

column_groups	(list) A named list of columns to group together into a single group of columns. There should be the same number of columns in each group.
group_labels	(list) A named list of columns and the label to use for all records in that column.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- dplyr

Value

A pivot_columns object with the following output slots:

updated	(annotation_source) The updated annotations as an annotation_source object.

Inheritance

A pivot_columns object inherits the following struct classes:

[pivot_columns] -> [model] -> [struct_class]

References

Wickham H, François R, Henry L, Müller K, Vaughan D (2026). *dplyr: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr https://doi.org/10.32614/CRAN.package.dplyr. R package version 1.2.1, https://CRAN.R-project.org/package=dplyr.

Examples

```
M <- pivot_columns(
        group_labels = list(),
        column_groups = list())
```

`prioritise_columns` *Combine several columns into a single column.*

Description

Several columns are merged into a single column. If multiple columns contain overlapping values then priority can be given columns earlier in the list.

Usage

```
prioritise_columns(
  column_names,
  output_name,
  source_name,
  source_tags = column_names,
  clean = TRUE,
  ...
)
```

Arguments

`column_names`	(character) The name(s) of column(s) to be combined.
`output_name`	(character) The name of the new column.
`source_name`	(character) The column name used to indicate where the merged values originated. The source is NA if all of the columns are NA.
`source_tags`	(character) The tags used to identify the source of each item in the new column. A tag should be provided for each column_name. By default the column name is used.
`clean`	(logical) Clean old columns. Allowed values are limited to the following: • `"TRUE"`: The named columns are removed after being combined. • `"FALSE"`: The named columns are retained after being combined. The default is `TRUE`.
`...`	Additional slots and values passed to `struct_class`.

Value

A `prioritise_columns` object with the following `output` slots:

`updated` (annotation_source) The input annotation source with the newly generated column.

Inheritance

A `prioritise_columns` object inherits the following `struct` classes:

`[prioritise_columns]` -> `[model]` -> `[struct_class]`

Examples

```
M <- prioritise_columns(
        column_names = "V1",
        output_name = "",
        clean = FALSE,
        source_name = "source_name",
        source_tags = "x")
```

---

pubchem_compound_lookup

Compound ID lookup via PubChem

Description

Uses the PubChem API to search for CID based on the input annotation column.

Usage

```
pubchem_compound_lookup(
  query_column,
  search_by,
  suffix = "_pubchem",
  output = "cids",
  records = "best",
  ...
)
```

Arguments

`query_column`	(character) The column name to use as the reference for searching the database e.g. "HMBD_ID".
`search_by`	(character) The PubChem domain to search for matches to the annotation_column.

`suffix` (character) A suffix appended to all column names in the returned result. The default is `"_pubchem"`.

`output` (character) The value returned from the pubchem database. The default is `"cids"`.

`records` (character) Returned record(s). Allowed values are limited to the following:

- `""`: Sometimes there are multiple matches to the PubChem, database especially when searhcing by name.
- `"best"`: Return only the best matching record.
- `"all"`: Return all matching records.

The default is `"best"`.

`...` Additional slots and values passed to `struct_class`.

Value

A `pubchem_compound_lookup` object with the following `output` slots:

`updated` (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A `pubchem_compound_lookup` object inherits the following `struct` classes:

`[pubchem_compound_lookup]` -> `[rest_api]` -> `[model]` -> `[struct_class]`

Examples

```
M <- pubchem_compound_lookup(
        search_by = "cid",
        output = "cids",
        records = "best",
        base_url = "https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound",
        url_template = "<base_url>/<search_by>/<query_column>/<output>/JSON",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

pubchem_id_exchange *PubChem ID Exchange via PUG*

Description

Uses the PubChem PUG API to perform ID exchange operations. Submits a list of identifiers and retrieves corresponding identifiers with the same chemical structure. Prediction data must contain the column specified by query_column. The prediction input need not be the same data used for training, but it must contain compatible query identifiers for joining results.

Usage

```
pubchem_id_exchange(
  query_column,
  output_type = "inchikey",
  suffix = "_pubchem_id_exchange",
  delay = 2,
  max_attempts = 30,
  max_poll_rounds = 10,
  input_type = "synonyms",
  input_source_name = NULL,
  output_source_name = NULL,
  n_batches = NULL,
  cache = NULL,
  cache_mode = "update",
  verbose = FALSE,
  ...
)
```

Arguments

query_column	(character) The name of a column in the annotation table containing values to search in the PUG API call.
output_type	(character) The type of identifiers to return from the PUG API. The default is "inchikey".
suffix	(character) A suffix appended to all column names in the returned result. The default is "_pubchem_id_exchange".
delay	(numeric, integer) Delay in seconds between status polling requests. The default is 2.
max_attempts	(numeric, integer) Maximum number of status polling attempts before giving up. The default is 30.
max_poll_rounds	(numeric, integer) Maximum number of whole-batch polling rounds before giving up when some batches are still unavailable. The default is 10.

`input_type` (character) The type of identifiers being provided as input. Determines which XML structure to use for the query. The default is `"synonyms"`.

`input_source_name` (character, NULL) The name of the external registry source when using Registry IDs as input (input_type = 'source-ids'). Required when input_type is 'source-ids'. The default is `NULL`.

`output_source_name` (character, NULL) The name of the external registry source when requesting Registry IDs as output (output_type = 'regid'). Required when output_type is 'regid'. The default is `NULL`.

`n_batches` (numeric, integer, NULL) Optional number of POST batches to split the unique query values across. If NULL, all query values are submitted in a single request. The default is `NULL`.

`cache` (annotation_database, NULL) A struct cache object (e.g. `rds_cache()`) that stores parsed responses to previous PubChem ID exchange queries, keyed by query value. Values already present in the cache are not resubmitted. Besides speeding up re-runs, this keeps a local, persistent record of what PubChem returned - useful given a translation service can go down or change without notice, as happened to the Chemical Translation Service this package once depended on (see `mwb_refmet_lookup()`). If not using a cache then set to NULL. The default is `NULL`.

`cache_mode` (character) Cache mode. Allowed values are limited to the following:

- `"update"`: The normal mode: values already in `cache` are used as-is, anything missing is submitted to PubChem and the result added to the cache.
- `"offline"`: Never submit a batch to PubChem - only values already present in `cache` are returned, and everything else is left as NA. Useful for continuing to work with a partially-populated cache while PubChem is unreachable or down, without waiting on or erroring against the live service. A warning lists how many query values are not covered by the cache when this happens (and, if the cache is entirely empty or unset, that every value will be returned as NA).
- `"rebuild"`: Ignore any existing cached value and submit every value to PubChem, overwriting the corresponding entry in `cache`. Useful when cached results are known to be stale.

The default is `"update"`.

`verbose` (logical) Whether to print debug information during execution. The default is `FALSE`.

`...` Additional slots and values passed to `struct_class`.

Details

Ported from `structReportsPCB` (same authors), where it was developed to drive PubChem's PUG XML ID Exchange service directly - a batch, asynchronous submit/poll/download workflow, rather than the synchronous per-row PUG REST calls used by `pubchem_compound_lookup()` / `pubchem_property_lookup()`. Prefer this object when translating a large number of identifiers at once, since one request covers many query values instead of one request per row.

Value

A pubchem_id_exchange object with the following output slots:

updated	(annotation_source) The annotation_source after adding data returned by the PUG API.
query_batches	(list) The query values submitted in each batch during training (excludes any already present in cache).
query_values	(character) Every unique, non-missing value of query_column seen during training (cached or not) - used t
request_ids	(list) PubChem PUG request IDs for each submitted batch.
trained	(logical) Whether the model has already submitted its PubChem jobs.

Inheritance

A pubchem_id_exchange object inherits the following struct classes:

[pubchem_id_exchange] -> [model] -> [struct_class]

See Also

Other REST API's: classyfire_batch_lookup(), classyfire_lookup(), cts_lite_lookup(), kegg_lookup(), lipidmaps_lookup(), mwb_compound_lookup(), mwb_refmet_lookup(), rest_api()

Examples

```
M <- pubchem_id_exchange(
        query_column = character(0),
        output_type = "inchikey",
        suffix = "_pubchem_id_exchange",
        delay = 2,
        max_attempts = 30,
        max_poll_rounds = 10,
        input_type = "synonyms",
        input_source_name = NULL,
        output_source_name = NULL,
        n_batches = NULL,
        cache = NULL,
        cache_mode = "update",
        verbose = FALSE)
```

---

pubchem_property_lookup

Compound property lookup via pubchem

Description

Uses the PubChem API to search for CID based onthe input annotation column and returns property information.

Usage

```
pubchem_property_lookup(
  query_column,
  search_by,
  suffix = "_pubchem",
  property = "InChIKey",
  ...
)
```

Arguments

query_column	(character) The column name to use as the reference for searching the database e.g. "HMBD_ID".
search_by	(character) The PubChem domain to search for matches to the annotation_column.
suffix	(character) A suffix appended to all column names in the returned result. The default is "_pubchem".
property	(character) A comma separated list of properties to return from the pubchem database. (see https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest#section=Compound-Property-Tables for details). Keyword ".all" will return all properties. The default is "InChIKey".
...	Additional slots and values passed to struct_class.

Value

A pubchem_property_lookup object with the following output slots:

updated	(annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A pubchem_property_lookup object inherits the following struct classes:

[pubchem_property_lookup] -> [pubchem_compound_lookup] -> [rest_api] -> [model] -> [struct_class]

Examples

```
M <- pubchem_property_lookup(
        search_by = "cid",
        property = "InChIKey",
        output = "cids",
        records = "best",
        base_url = "https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound",
        url_template = "<base_url>/<search_by>/<query_column>/property/<property>/JSON",
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
```

```
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

pubchem_structure *PubChem molecular structure*

Description

Query the PubChem api and retrieve a display an image of the matching molecular structure.

Usage

```
pubchem_structure(
  query_column,
  search_by,
  row_index,
  record_type = "2d",
  image_size = "large",
  ...
)
```

Arguments

query_column	(character) The name of the annotation_source column with compound identifiers of the type specified in the search_by param.
search_by	(character) The PubChem domain to search for matches to the annotation_column.
row_index	(integer, numeric) The row index of the annotation_source to request an image of the molecular structure of.
record_type	(character) The record type to return from the PubChem query. Can be one of "2d" or "3d". The default is "2d".
image_size	(character) The size of the image to return from the PubChem query. Can be one of "large" or "small". For record_type = "2d" an arbitrary image size can be specified e.g. 123x123. The default is "large".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- cowplot

This object queries the PubChem API for matches to your query without caching the results. It is therefore intended for limited use. If you wish to obtain images for a large number of moelucules you should seek an alternative solution.

Value

A pubchem_structure object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A pubchem_structure object inherits the following struct classes:

[pubchem_structure] -> [chart] -> [struct_class]

References

Wilke C (2025). *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*. doi:10.32614/CRAN.package.cowplot https://doi.org/10.32614/CRAN.package.cowplot. R package version 1.2.0, https://CRAN.R-project.org/package=cowplot.

Examples

```
M <- pubchem_structure(
        query_column = "V1",
        search_by = "cid",
        row_index = 1,
        record_type = "2d",
        image_size = "large")
```

---

pubchem_widget *PubChem widget*

Description

Display a PubChem HTML widget for a compound.

Usage

```
pubchem_widget(
  query_column,
  row_index,
  record_type = "2D-Structure",
  hide_title = FALSE,
  width = "600px",
  height = "650px",
  display = TRUE,
  ...
)
```

Arguments

query_column	(character) The name of the annotation_source column with compound identifiers of the type specified in the search_by param.
row_index	(integer, numeric) The row index of the annotation_source to request an image of the molecular structure of.
record_type	(character) The record type for the widget. The default is "2D-Structure".
hide_title	(logical) Hide widget title. Allowed values are limited to the following: • "TRUE": The title is displayed. • "FALSE": The title is not displayed. The default is FALSE.
width	(integer, numeric, character) The width of the widget in a CSS style compatible format. Numerical values will be converted to character. The default is "600px".
height	(integer, numeric, character) The height of the widget in a CSS style compatible format.Numerical values will be converted to character. The default is "650px".
display	(logical) Display widget. Allowed values are limited to the following: • "TRUE": Display the widget. • "FALSE": Do not display the widget and only return the HTML. The default is TRUE.
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- htmltools

Value

A pubchem_widget object. This object has no output slots. See chart_plot in the struct package to plot this chart object.

Inheritance

A pubchem_widget object inherits the following struct classes:

[pubchem_widget] -> [chart] -> [struct_class]

References

Cheng J, Sievert C, Schloerke B, Chang W, Xie Y, Allen J (2025). *htmltools: Tools for HTML*. doi:10.32614/CRAN.package.htmltools https://doi.org/10.32614/CRAN.package.htmltools. R package version 0.5.9, https://CRAN.R-project.org/package=htmltools.

Examples

```
M <- pubchem_widget(
        query_column = "V1",
        row_index = 1,
        record_type = "2D-Structure",
        hide_title = FALSE,
        width = 600,
        height = 400,
        display = FALSE)
```

---

racemic_dictionary *Racemic dictionary*

Description

This dictionary removes racemic properties from molecule names. It is intended for use with the normalise_strings() object.

Usage

```
racemic_dictionary
```

Value

A dictionary for use with normalise_strings()

Examples

```
M <- normalise_strings(
    search_column = "example",
    output_column = "result",
    dictionary = racemic_dictionary
)
```

---

rdata_database *rdata database*

Description

A data.frame stored as an RData file.

Usage

```
rdata_database(source = character(0), variable_name, ...)
```

Arguments

source	(ANY) The source of annotation data. The default is character(0).
variable_name	(character, function) The name of the data.frame in the imported workspace to use as the data.frame for this source. A function can be provided to e.g. extract a data.frame from a list in the imported environment.
...	Additional slots and values passed to struct_class.

Value

A rdata_database object. This object has no output slots.

Inheritance

A rdata_database object inherits the following struct classes:

[rdata_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

See Also

Other annotation databases: AnnotationDb_database(), GO_database(), annotation_database(), annotation_source(), excel_database(), rds_cache(), rds_database()

Examples

```
M <- rdata_database(
        variable_name = "a data frame",
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

rds_cache *rds cache*

Description

A data.frame stored as an RDS file. Intended to be used with rest_api objects as mechanism for caching search results. The data.frame for an rds_cache object must have a column named ".search".

Usage

```
rds_cache(
  source = character(0),
  data = data.frame(.search = character(0)),
  ...
)
```

Arguments

source	(ANY) The source of annotation data. The default is character(0).
data	(data.frame, NULL) A data.frame of annotation data. The default is data.frame(.search = character(0)).
...	Additional slots and values passed to struct_class.

Value

A rds_cache object. This object has no output slots.

Inheritance

A rds_cache object inherits the following struct classes:

[rds_cache] -> [rds_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

See Also

Other annotation databases: AnnotationDb_database(), GO_database(), annotation_database(), annotation_source(), excel_database(), rdata_database(), rds_database()

Examples

```
M <- rds_cache(
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

rds_database *rds database*

Description

A data.frame stored as an RDS file.

Usage

```
rds_database(source = character(0), ...)
```

Arguments

source	(ANY) The source of annotation data. The default is character(0).
...	Additional slots and values passed to struct_class.

Value

A rds_database object. This object has no output slots.

Inheritance

A rds_database object inherits the following struct classes:

[rds_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

See Also

Other annotation databases: AnnotationDb_database(), GO_database(), annotation_database(), annotation_source(), excel_database(), rdata_database(), rds_cache()

Examples

```
M <- rds_database(
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

read_database *Read a database*

Description

Reads an annotation_database and returns the data.frame.

Usage

```
read_database(obj, ...)

## S4 method for signature 'annotation_database'
read_database(obj)

## S4 method for signature 'AnnotationDb_database'
read_database(obj)

## S4 method for signature 'BiocFileCache_database'
read_database(obj)

## S4 method for signature 'MTox700plus_database'
read_database(obj)

## S4 method for signature 'PathBank_metabolite_database'
read_database(obj)

## S4 method for signature 'excel_database'
read_database(obj)
```

```
## S4 method for signature 'github_file'
read_database(obj)

## S4 method for signature 'mwb_refmet_database'
read_database(obj)

## S4 method for signature 'rdata_database'
read_database(obj)

## S4 method for signature 'rds_database'
read_database(obj)

## S4 method for signature 'sqlite_database'
read_database(obj)

## S4 method for signature 'zenodo_file'
read_database(obj)
```

Arguments

obj	An annotation_database object
...	additional database specific inputs

Value

A data.frame

Examples

```
M <- rds_database(tempfile())
df <- read_database(M)
```

---

read_source *Import annotation source*

Description

Import an data from e.g. a raw file and parse it into an annotation_source() object.

Usage

```
read_source(obj, ...)

## S4 method for signature 'annotation_source'
read_source(obj)

## S4 method for signature 'annotation_database'
```

```
read_source(obj)

## S4 method for signature 'cd_source'
read_source(obj)

## S4 method for signature 'ls_source'
read_source(obj)

## S4 method for signature 'mspurity_source'
read_source(obj)

## S4 method for signature 'mwb_study_source'
read_source(obj)
```

Arguments

obj	an annotation_source() object
...	not currently used

Value

an annotation_table() or annotation_database() object

Examples

```
# prepare source
CD <- cd_source(
    source = system.file(
        paste0("extdata/MTox/CD/HILIC_POS.xlsx"),
        package = "MetMashR"
    )
)
```

---

remove_columns *Select columns*

Description

A wrapper around tidyselect::eval_select. Remove columns from an annotation table using tidy grammar.

Usage

```
remove_columns(expression = everything(), ...)
```

Arguments

expression	(call) A valid rlang::expr for tidy evaluation via eval_select. e.g. expression = all_of(c("foo","bar")) will select columns named "foo" and "bar" from the annotation data.frame. . The default is everything().
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- tidyselect
- rlang

Value

A remove_columns object with the following output slots:

updated	(annotation_source) The updated annotations as an annotation_source object.

Inheritance

A remove_columns object inherits the following struct classes:

[remove_columns] -> [model] -> [struct_class]

References

Henry L, Wickham H (2024). *tidyselect: Select from a Set of Strings*. doi:10.32614/CRAN.package.tidyselect https://doi.org/10.32614/CRAN.package.tidyselect. R package version 1.2.1, https://CRAN.R-project.org/package=tidyselect.

Henry L, Wickham H (2026). *rlang: Functions for Base Types and Core R and 'Tidyverse' Features*. doi:10.32614/CRAN.package.rlang https://doi.org/10.32614/CRAN.package.rlang. R package version 1.3.0, https://CRAN.R-project.org/package=rlang.

See Also

dplyr::select()

tidyselect::eval_select()

Examples

```
M <- remove_columns(
        expression = call("example"))
```

rename_columns *Select columns*

Description

A wrapper around dplyr::rename. Rename columns from an annotation table using tidy grammar.

Usage

```
rename_columns(expression, ...)
```

Arguments

expression	(call) A valid rlang::expr for tidy evaluation e.g. expression = all_of(c("foo"="bar")) will rename the column named "bar" and "foo". .
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- tidyselect
- rlang

Value

A rename_columns object with the following output slots:

updated	(annotation_source) The updated annotations as an annotation_source object.

Inheritance

A rename_columns object inherits the following struct classes:

[rename_columns] -> [model] -> [struct_class]

References

Henry L, Wickham H (2024). *tidyselect: Select from a Set of Strings*. doi:10.32614/CRAN.package.tidyselect https://doi.org/10.32614/CRAN.package.tidyselect. R package version 1.2.1, https://CRAN.R-project.org/package=tidyselect.

Henry L, Wickham H (2026). *rlang: Functions for Base Types and Core R and 'Tidyverse' Features*. doi:10.32614/CRAN.package.rlang https://doi.org/10.32614/CRAN.package.rlang. R package version 1.3.0, https://CRAN.R-project.org/package=rlang.

Examples

```
M <- rename_columns(
        expression = call("example"))
```

`required_cols` *Required columns in an annotation source*

Description

Some `annotation_sources`, such as LCMS tables (`lcms_table`), require that certain columns are present in the data.frame. These are defined by slots in the source definition. The name of slots containing the required column names for a source can be retrieved using the `required_cols` function, which will collect and return the names of slots containing required column names for the object and all of its parent objects.

Usage

```
required_cols(obj, ...)

## S4 method for signature 'annotation_source'
required_cols(obj)
```

Arguments

`obj`	an `annotation_source` object
`...`	additional source specific inputs

Value

a character vector of slot names

Examples

```
# prepare object
M <- lcms_table(id_column = "id", mz_column = "mz", rt_column = "rt")

#' # get values for required slots
r <- required_cols(M)

# get slot names for required columns
names(r)
```

rest_api *rest_api*

Description

A base class providing common methods for making REST API calls.

Usage

```
rest_api(
  base_url,
  url_template,
  suffix,
  status_codes,
  delay,
  cache = NULL,
  cache_mode = "update",
  query_column,
  ...
)
```

Arguments

base_url	(character) The base URL of the API.
url_template	(character) A template describing how the URL should be constructed from the base URL and input parameters. e.g. <base_url>//<input_item>/<search_term>/json.The url will be constructed by replacing the values enclosed in <> with the value from corresponding input parameter of the rest_api object.
suffix	(character) A suffix appended to all column names in the returned result.
status_codes	(list) Named list of status codes and function indicating how to respond. Should minimally contain a function to parse a successful response for status code 200. Any codes not provided will be passed to httr::stop_for_status().
delay	(numeric, integer) Delay in seconds between API calls.
cache	(annotation_database, NULL) A struct cache object that contains parsed responses to previous api queries. If not using a cache then set to NULL. The default is NULL.
cache_mode	(character) Cache mode. Allowed values are limited to the following: • "update": The normal mode: values already in cache are used as-is, anything missing is queried live and the result added to the cache. • "offline": Never query the live API - only values already present in cache are returned, and everything else is left as NA. Useful for continuing to work with a partially-populated cache while the API is unreachable or down, without waiting on or erroring against the live service. A warning lists how many query values are not covered by the cache when this happens (and, if the cache is entirely empty or unset, that every value will be returned as NA).

- `"rebuild"`: Ignore any existing cached value and query the live API for every value, overwriting the corresponding entry in `cache`. Useful when cached results are known to be stale.

The default is `"update"`.

`query_column` (character) The name of a column in the annotation table containing values to search in the api call.

`...` Additional slots and values passed to `struct_class`.

Value

A `rest_api` object with the following `output` slots:

`updated` (annotation_source) The annotation_source after adding data returned by the API.

Inheritance

A `rest_api` object inherits the following `struct` classes:

`[rest_api]` -> `[model]` -> `[struct_class]`

See Also

Other REST API's: `classyfire_batch_lookup()`, `classyfire_lookup()`, `cts_lite_lookup()`, `kegg_lookup()`, `lipidmaps_lookup()`, `mwb_compound_lookup()`, `mwb_refmet_lookup()`, `pubchem_id_exchange()`

Examples

```
M <- rest_api(
        base_url = "V1",
        url_template = character(0),
        query_column = character(0),
        cache = NULL,
        cache_mode = "update",
        status_codes = list(),
        delay = 0.5,
        suffix = "_rest_api")
```

---

rt_match *rt matching*

Description

Annotations will be matched to the measured variable meta data.frame by determining which annotations rt window overlaps with the rt window from the measured rt.

Usage

```
rt_match(variable_meta, rt_column, rt_window, id_column, ...)
```

Arguments

variable_meta	(data.frame) A data.frame of variable IDs and their corresponding rt values.
rt_column	(character) Column name of the rt values in variable_meta.
rt_window	(numeric, integer) Rt window to use for matching. If a single value is provided then the same rt is used for both variable meta and the annotations. A named vector can also be provided e.g. c("variable_meta"=5,"annotations"=2) to use different ", "windows for each data table.
id_column	(character) Column name of the variable ids in variable_meta. ", "id_column="rownames" will use the rownames as ids.
...	Additional slots and values passed to struct_class.

Value

A rt_match object with the following output slots:

updated	(annotation_table) The input annotation source with the newly generated column.

Inheritance

A rt_match object inherits the following struct classes:

[rt_match] -> [model] -> [struct_class]

Examples

```
M <- rt_match(
        variable_meta = data.frame(),
        rt_column = character(0),
        rt_window = 20,
        id_column = character(0))
```

---

select_columns *Select columns*

Description

A wrapper around tidyselect::eval_select. Select columns from an annotation table using tidy grammar. This imitates dplyr::select().

Usage

```
select_columns(expression = everything(), ...)
```

Arguments

expression	(call) A valid rlang::expr for tidy evaluation via eval_select. e.g. `expression = all_of(c("foo","bar"))` will select columns named "foo" and "bar" from the annotation data.frame. . The default is `everything()`.
...	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `tidyselect`
- `rlang`

Value

A `select_columns` object with the following `output` slots:

`updated`	(annotation_source) The updated annotations as an `annotation_source` object.

Inheritance

A `select_columns` object inherits the following `struct` classes:

`[select_columns]` -> `[model]` -> `[struct_class]`

References

Henry L, Wickham H (2024). *tidyselect: Select from a Set of Strings*. doi:10.32614/CRAN.package.tidyselect https://doi.org/10.32614/CRAN.package.tidyselect. R package version 1.2.1, https://CRAN.R-project.org/package=tidyselect.

Henry L, Wickham H (2026). *rlang: Functions for Base Types and Core R and 'Tidyverse' Features*. doi:10.32614/CRAN.package.rlang https://doi.org/10.32614/CRAN.package.rlang. R package version 1.3.0, https://CRAN.R-project.org/package=rlang.

See Also

`dplyr::select()`

`tidyselect::eval_select()`

Examples

```
M <- select_columns(
        expression = call("example"))
```

split_column *Split a column*

Description

A wrapper for strsplit. Divides a column into multiple columns by dividing the contents

Usage

```
split_column(
  column_name,
  separator = "_",
  padding = NA,
  keep_indices = NULL,
  clean = TRUE,
  ...
)
```

Arguments

column_name	(character) The column name in the annotation_source split.
separator	(character) A substring to split the column by. The default is "_".
padding	(character, logical) A character string used to represent missing and zero length strings after splitting. The default is NA.
keep_indices	(numeric, integer) The indices of columns to keep after splitting. If NULL then all columns are retained. The default is NULL.
clean	(logical) Clean old columns. Allowed values are limited to the following: • "TRUE": The named columns are removed after being split. • "FALSE": The named columns are retained after being split. The default is TRUE.
...	Additional slots and values passed to struct_class.

Value

A split_column object with the following output slots:

updated	(annotation_source) The annotation_source after splitting the column.

Inheritance

A split_column object inherits the following struct classes:

[split_column] -> [model] -> [struct_class]

Examples

```
M <- split_column(
        column_name = "V1",
        separator = "_",
        clean = FALSE,
        padding = FALSE,
        keep_indices = numeric(0))
```

`split_records` *Expand records*

Description

Expand single records into multiple records by splitting strings in a named column at the chosen separator. For example, if a for a record the column `synonyms = c("glucose,dextrose")` then by splitting at the comma results in two records, one for glucose and one for dextrose with identical values (apart from the column being split). The original record is removed.

Usage

```
split_records(column_name, separator, clean = TRUE, ...)
```

Arguments

`column_name`	(character) The column name of the `annotation_source` to split intomultiple records.
`separator`	(character) The substring used to split the values in column_name into multiple records.
`clean`	(logical) Remove the original column. If FALSE the original column will be retained in the final output with .original appended to the column name. The default is `TRUE`.
`...`	Additional slots and values passed to `struct_class`.

Details

This object makes use of functionality from the following packages:

- `tidytext`

Value

A `split_records` object with the following `output` slots:

`updated`	(annotation_source) The updated annotations as an `annotation_source` object.

Inheritance

A split_records object inherits the following struct classes:

[split_records] -> [model] -> [struct_class]

References

Silge J, Robinson D (2016). "tidytext: Text Mining and Analysis Using Tidy Data Principles in R." *JOSS*, *1*(3). doi:10.21105/joss.00037 https://doi.org/10.21105/joss.00037. http://dx.doi.org/10.21105/joss.00037.

Examples

```
M <- split_records(
        column_name = character(0),
        separator = ",",
        clean = FALSE)
```

---

sqlite_database *SQLite database*

Description

A data.frame stored in an SQLite database.

Usage

```
sqlite_database(source, table = "annotation_database", ...)
```

Arguments

source	(ANY) The source of annotation data.
table	(character) The name of a table in the SQLite database. The default is "annotation_database".
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- RSQLite

Value

A sqlite_database object. This object has no output slots.

Inheritance

A `sqlite_database` object inherits the following `struct` classes:

`[sqlite_database]` -> `[annotation_database]` -> `[annotation_source]` -> `[struct_class]`

References

Müller K, Wickham H, James DA, Falcon S (2026). *RSQLite: SQLite Interface for R*. doi:10.32614/CRAN.package.RSQLite https://doi.org/10.32614/CRAN.package.RSQLite. R package version 3.53.3, https://CRAN.R-project.org/package=RSQLite.

See Also

Other database: `BiocFileCache_database()`, `lipidmaps_database()`

Examples

```
M <- sqlite_database(
        table = character(0),
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

---

trim_whitespace *Trim whitespace*

Description

A wrapper for `trimws()`. Removes leading and/or trailing whitespace from character strings.

Usage

```
trim_whitespace(column_names, which = "both", whitespace = "[ \t\r\n]", ...)
```

Arguments

`column_names`	(character) The column name(s) in the annotation_source to trim white space from. Special case ".all" will apply to all columns.
`which`	(character) Trailing and/or leading whitespace. Allowed values are limited to the following: • `""`: A character string specifying the location of whitespace to remove. • `"left"`: Remove leading whitespace. • `"right"`: Remove trailing whitespace. • `"both"`: Remove both leading and trailing whitespace. The default is `"both"`.

whitespace	(character) A string specifying a regular expression to match (one character of) "white space". See trimws() for details. The default is "[ ]".
...	Additional slots and values passed to struct_class.

Value

A trim_whitespace object with the following output slots:

updated	(annotation_source) The annotation_source after trimming whitespace.

Inheritance

A trim_whitespace object inherits the following struct classes:

[trim_whitespace] -> [model] -> [struct_class]

Examples

```
M <- trim_whitespace(
        column_names = "V1",
        which = "both",
        whitespace = "[
]")
```

---

tripeptide_dictionary *Tripeptide dictionary*

Description

A dictionary for converting tripeptides encoded using single letter IUPAC codes to use three letter codes for amino acids separated by hyphens. e.g. INK becomes Ile-Asn-Lys

Usage

```
tripeptide_dictionary
```

Value

A dictionary for use with normalise_strings()

Examples

```
M <- normalise_strings(
    search_column = "example",
    output_column = "result",
    dictionary = tripeptide_dictionary
)
```

unique_records *Keep unique_records*

Description

reduces an annotation source to unique records only; all duplicates are removed.

Usage

```
unique_records(...)
```

Arguments

...	Additional slots and values passed to struct_class.

Value

A unique_records object with the following output slots:

updated (annotation_source) The updated annotations as an annotation_source object.

Inheritance

A unique_records object inherits the following struct classes:

[unique_records] -> [model] -> [struct_class]

Examples

```
M <- unique_records()
```

unzip_before_cache *Unzip file before caching with BiocFileCache_database*

Description

This helper function is for use with BiocFileCache_database() objects. Using it as the bfc_fun input for this object will unzip a downloaded resource into a temporary folder before storing it in the cache.

Usage

```
unzip_before_cache(from, to)
```

Arguments

from	incoming path
to	the outgoing path

Value

TRUE if successful

Examples

```
M <- BiocFileCache_database(
    source = tempfile(),
    resource_name = "example",
    bfc_fun = unzip_before_cache
)
```

---

upset_min_size *UpSet chart filter helper functions*

Description

These functions create filters for the annotation_upset_chart class to control which intersections are displayed in UpSet plots. Each function returns a filter function that can be used with the filter parameter.

Usage

```
upset_min_size(min_size)

upset_min_groups(min_groups)

upset_max_groups(max_groups)

upset_intersections(combinations)
```

Arguments

min_size	numeric The minimum number of items in an intersection
min_groups	numeric The minimum number of groups in an intersection
max_groups	numeric The maximum number of groups in an intersection
combinations	character Vector of specific intersection combinations to include (e.g., c("A/B", "B/C"))

Details

These filter functions work by analyzing the region data from the Venn diagram to determine which intersections meet the specified criteria:

- `upset_min_size()`: Filters intersections based on the number of items
- `upset_min_groups()`: Filters intersections based on the minimum number of groups involved
- `upset_max_groups()`: Filters intersections based on the maximum number of groups involved
- `upset_intersections()`: Filters to show only specific intersection combinations (e.g., "A/B", "B/C", "A/B/C")

For complex filtering logic, create custom filter functions:

```
# Single filter
filter = upset_min_size(5)

# Specific combinations only
filter = upset_intersections(c("A/B", "B/C"))

# Custom filter function with AND logic
custom_filter <- function(region_data) {
  region_data$count >= 3 & region_data$count <= 10 & grepl("A", region_data$name)
}

# Custom filter function with OR logic
or_filter <- function(region_data) {
  region_data$count >= 5 | grepl("B/C", region_data$name)
}
```

Value

A function that takes `region_data` as input and returns a logical vector indicating which intersections to keep.

Examples

```
# create a filter function that keeps intersections with 5+ items
f <- upset_min_size(5)
is.function(f)


# Filter to show only intersections with 5+ items
C <- annotation_upset_chart(factor_name = "V1", filter = upset_min_size(5))

# Filter to show only intersections involving 3+ groups
C <- annotation_upset_chart(factor_name = "V1", filter = upset_min_groups(3))

# Filter to show only intersections involving 2-4 groups (custom function)
group_range_filter <- function(region_data) {
```

```
  group_counts <- sapply(region_data$name, function(x) {
    if (x == "") return(0)
    groups <- strsplit(x, "/")[[1]]
    length(groups)
  })
  group_counts >= 2 & group_counts <= 4
}
C <- annotation_upset_chart(factor_name = "V1", filter = group_range_filter)

# Filter to show only specific combinations
C <- annotation_upset_chart(factor_name = "V1",
                            filter = upset_intersections(c("A/B", "B/C")))

# Custom filter combining size and group criteria
size_and_group_filter <- function(region_data) {
  region_data$count >= 3 & sapply(region_data$name, function(x) {
    if (x == "") return(FALSE)
    groups <- strsplit(x, "/")[[1]]
    length(groups) >= 2
  })
}
C <- annotation_upset_chart(factor_name = "V1", filter = size_and_group_filter)
```

`vertical_join` *Join sources vertically*

Description

A function to join sources vertically. A vertical join involves matching common columns across source data.frames and padding missing columns to create a single new data.frame with data and records from multiple sources.

Usage

```
vertical_join(x, y, ...)

## S4 method for signature 'annotation_source,annotation_source'
vertical_join(
  x,
  y,
  matching_columns = NULL,
  keep_cols = NULL,
  source_col = "annotation_source",
  exclude_cols = NULL,
  as = annotation_source()
)
```

```
## S4 method for signature 'list,missing'
vertical_join(
  x,
  y,
  matching_columns = NULL,
  keep_cols = NULL,
  source_col = "annotation_source",
  exclude_cols = NULL,
  as = annotation_source()
)
```

Arguments

x	an annotation_source object
y	an second annotation_source object to join with the first
...	additional inputs (not currently used)
matching_columns	(list) a named list of column names that all contain the same information. All columns named in the same list element will be merged into a single column with the same name as the list element.
keep_cols	(character) a list of column names to keep in the final joined table. All other columns will be dropped.
source_col	(character) the name of a new column that will contain the tags of the original source object for each row in the joined table.
exclude_cols	(character) the names of columns to exclude from the joined table.
as	(character) the type of object the joined table should be returned as e.g. "lcms_table".

Value

an annotation_source object

Examples

```
M <- annotation_source(data = data.frame(id = 1, value = "A"))
N <- annotation_source(data = data.frame(id = 2, value = "B"))
O <- vertical_join(M, N, keep_cols = ".all")
```

---

wherever *Filter helper function to select records*

Description

Returns a list of quosures for use with filter_records to allow the use of dplyr-style expressions. See examples.

Usage

```
wherever(...)
```

Arguments

...	Expressions that return a logical value and are defined in terms of the columns in the annotation_source. If multiple conditions are included then they are combined with the & operator. Only records for which all conditions evaluate to TRUE are kept.

Value

a list of quosures for use with filter_records

See Also

filter_records()

Examples

```
# some annotation data
AN <- annotation_source(data = iris)

# filter to setosa where Sepal length is less than 5
M <- filter_records(
    wherever(
        Species == "setosa",
        Sepal.Length < 5
    )
)
M <- model_apply(M, AN)
predicted(M) # 20 rows
```

---

write_database *Write to a database*

Description

Writes a data.frame to a annotation_database.

Usage

```
write_database(obj, ...)

## S4 method for signature 'annotation_database'
write_database(obj, df)
```

```
## S4 method for signature 'rds_database'
write_database(obj, df)

## S4 method for signature 'sqlite_database'
write_database(obj, df)
```

Arguments

obj	A annotation_database object
...	additional database specific inputs
df	(data.frame) the data.frame to store in the database.

Value

Silently returns TRUE if successful, FALSE otherwise

Examples

```
M <- rds_database(tempfile())
write_database(M, data.frame())
```

---

zenodo_file *Zenodo file*

Description

Retrieves a file from a Zenodo record and caches it locally using BiocFileCache.

Usage

```
zenodo_file(
  record_id,
  file_name,
  bfc_path = NULL,
  resource_name = paste("zenodo", record_id, file_name, sep = "_"),
  ...
)
```

Arguments

record_id	(character, numeric) The ID of the Zenodo record containing the file, e.g. 8226097 for doi:10.5281/zenodo.8226097. Each version of a Zenodo record has its own ID.
file_name	(character) The name of the file to download from the Zenodo record.
bfc_path	(character, NULL) BiocFileCache is used to cache the database locally and prevent unnecessary downloads. If a path is provided then BiocFileCache will use this location. If NULL it will use the default location (see BiocFileCache::BiocFileCache() for details). The default is NULL.

resource_name	(character) The name given to this resource in the cache. (see BiocFileCache::BiocFileCache() for details). The default is paste("zenodo", record_id, file_name, sep = "_").
...	Additional slots and values passed to struct_class.

Details

This object makes use of functionality from the following packages:

- BiocFileCache

Value

A zenodo_file object. This object has no output slots.

Inheritance

A zenodo_file object inherits the following struct classes:

[zenodo_file] -> [BiocFileCache_database] -> [annotation_database] -> [annotation_source] -> [struct_class]

References

Shepherd L, Morgan M (2026). *BiocFileCache: Manage Files Across Sessions*. R package version 3.2.0.

Examples

```
M <- zenodo_file(
        record_id = character(0),
        file_name = character(0),
        bfc_path = NULL,
        resource_name = "bfc",
        bfc_fun = function(){},
        import_fun = function(){},
        offline = FALSE,
        tag = character(0),
        data = data.frame(),
        source = "ANY")
```

Index

∗ **REST API's**
classyfire_batch_lookup, 35
classyfire_lookup, 38
cts_lite_lookup, 50
kegg_lookup, 70
lipidmaps_lookup, 75
mwb_compound_lookup, 82
mwb_refmet_lookup, 85
pubchem_id_exchange, 101
rest_api, 117
∗ **annotation databases**
annotation_database, 9
annotation_source, 15
AnnotationDb_database, 22
excel_database, 55
GO_database, 65
rdata_database, 108
rds_cache, 109
rds_database, 110
∗ **annotation sources**
annotation_database, 9
annotation_table, 16
cd_source, 30
ls_source, 77
mspurity_source, 79
mwb_study_source, 87
∗ **annotation tables**
annotation_table, 16
cd_source, 30
ls_source, 77
mspurity_source, 79
∗ **annotation_tables**
lcms_table, 73
∗ **database**
BiocFileCache_database, 25
lipidmaps_database, 74
sqlite_database, 123

add_columns, 5
add_labels, 6
annotation_bar_chart, 7
annotation_database, 9
annotation_database(), *15, 16*, *23*, *31*, *56*, *66*, *78*, *80*, *88*, *109–111*, *113*
annotation_histogram, 10
annotation_histogram2d, 11
annotation_pie_chart, 13
annotation_source, 15
annotation_source(), *9*, *16*, *23*, *34, 35*, *56*, *66*, *109–113*
annotation_table, 16, *28*
annotation_table(), *9*, *30, 31*, *73*, *77, 78*, *80*, *88*, *113*
annotation_upset_chart, 17
annotation_venn_chart, 21
AnnotationDb_database, 22
AnnotationDb_database(), *9*, *15*, *56*, *66*, *109–111*
AnnotationDb_select, 24
AnnotationDbi::AnnotationDb, *23*
AnnotationDbi::select(), *25*

BiocFileCache::bfcdownload(), *26*
BiocFileCache::BiocFileCache, *84*
BiocFileCache::BiocFileCache(), *26*, *64*, *74*, *81*, *96*, *132, 133*
BiocFileCache_database, 25
BiocFileCache_database(), *75*, *124*, *126*

cache_as_is, 27
calc_ppm_diff, 28
calc_rt_diff, 29
cd_source, 30
cd_source(), *9*, *16*, *78*, *80*, *88*
chart_plot, *8*, *11, 12*, *14*, *19*, *22*, *87*, *106, 107*
chart_plot (chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_bar_chart,annotation_source-method, 32

chart_plot,annotation_histogram,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_histogram2d,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_pie_chart,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_upset_chart,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_upset_chart,list-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_venn_chart,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,annotation_venn_chart,list-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,mwb_structure,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,pubchem_structure,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chart_plot,pubchem_widget,annotation_source-method
(chart_plot,annotation_bar_chart,annotation_source-method), 32
chebi_lookup, 32
chebi_lookup(), *51*
check_for_columns, 34
check_for_columns,annotation_source-method
(check_for_columns), 34
classyfire_batch_lookup, 35
classyfire_batch_lookup(), *39*, *52*, *72*, *76*, *83*, *86*, *103*, *118*
classyfire_lookup, 38
classyfire_lookup(), *35*, *37*, *52*, *72*, *76*, *83*, *86*, *103*, *118*
combine_columns, 39
combine_records, 41
combine_records(), *42*, *44*
combine_records_helper_functions, 42
combine_sources, 45
CompoundDb_source, 47
compute_column, 48
compute_method
(combine_records_helper_functions), 42
compute_mode
(combine_records_helper_functions), 42
compute_record, 49
count_records
(combine_records_helper_functions), 42
cts_lite_lookup, 50
cts_lite_lookup(), *37*, *39*, *72*, *76*, *83*, *86*, *103*, *118*
database_lookup, 52
dplyr::filter, *60*
dplyr::filter(), *61*
dplyr::left_join, *5*
dplyr::left_join(), *6*, *25*
dplyr::rename, *115*
dplyr::select(), *114*, *119*, *120*

eutils_lookup, 54
excel_database, 55
excel_database(), *9*, *15*, *23*, *66*, *109–111*

filter_labels, 57
filter_na, 58
filter_range, 59
filter_records, 60
filter_records(), *131*
filter_venn, 62
fuse
(combine_records_helper_functions), 42
fuse_unique
(combine_records_helper_functions), 42

github_file, 63
GO_database, 65
GO_database(), *9*, *15*, *23*, *56*, *109–111*
greek_dictionary, 66
grepl(), *94*
gsub(), *94*

hmdb_lookup, 67

id_counts, 68
import_source, 69
interaction(), *39*
is_writable, 70
is_writable,annotation_database-method
(is_writable), 70
is_writable,rdata_database-method
(is_writable), 70

kegg_lookup, 70
kegg_lookup(), *37*, *39*, *52*, *76*, *83*, *86*, *103*, *118*

lcms_table, 73
lipidmaps_database, 74
lipidmaps_database(), *27*, *124*
lipidmaps_lookup, 75
lipidmaps_lookup(), *37*, *39*, *52*, *72*, *83*, *86*, *103*, *118*
ls_source, 77
ls_source(), *9*, *16*, *31*, *80*, *88*

model_apply
(model_apply,model,annotation_source-method), 78
model_apply,add_columns,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,add_labels,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,AnnotationDb_select,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,calc_ppm_diff,annotation_table-method
(model_apply,model,annotation_source-method), 78
model_apply,calc_rt_diff,annotation_table-method
(model_apply,model,annotation_source-method), 78
model_apply,chebi_lookup,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,classyfire_batch_lookup,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,combine_columns,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,combine_records,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,combine_sources,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,combine_sources,list-method
(model_apply,model,annotation_source-method), 78
model_apply,CompoundDb_source,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,compute_column,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,compute_record,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,cts_lite_lookup,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,database_lookup,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,filter_labels,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,filter_na,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,filter_range,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,filter_records,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,filter_venn,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,id_counts,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,import_source,annotation_source-method
(model_apply,model,annotation_source-method), 78
model_apply,kegg_lookup,annotation_source-method
(model_apply,model,annotation_source-method), 78

model_apply,model,annotation_source-method, 78
model_apply,model,list-method (model_apply,model,annotation_source-method), 78
model_apply,model_seq,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,model_seq,list-method (model_apply,model,annotation_source-method), 78
model_apply,mz_match,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,mzrt_match,lcms_table-method (model_apply,model,annotation_source-method), 78
model_apply,normalise_lipids,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,normalise_strings,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,pivot_columns,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,prioritise_columns,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,pubchem_id_exchange,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,remove_columns,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,rename_columns,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,rest_api,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,rt_match,annotation_table-method (model_apply,model,annotation_source-method), 78
model_apply,select_columns,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,split_column,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,split_records,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,trim_whitespace,annotation_source-method (model_apply,model,annotation_source-method), 78
model_apply,unique_records,annotation_source-method (model_apply,model,annotation_source-method), 78
model_predict (model_apply,model,annotation_source-method), 78
model_predict,classyfire_batch_lookup,annotation_source-method (model_apply,model,annotation_source-method), 78
model_predict,pubchem_id_exchange,annotation_source-method (model_apply,model,annotation_source-method), 78
model_train (model_apply,model,annotation_source-method), 78
model_train,classyfire_batch_lookup,annotation_source-method (model_apply,model,annotation_source-method), 78
model_train,pubchem_id_exchange,annotation_source-method (model_apply,model,annotation_source-method), 78
mspurity_source, 79
mspurity_source(), *9*, *16*, *31*, *78*, *88*
MTox700plus_database, 80
mwb_compound_lookup, 82
mwb_compound_lookup(), *37*, *39*, *52*, *72*, *76*, *86*, *103*, *118*
mwb_refmet_database, 83
mwb_refmet_lookup, 85
mwb_refmet_lookup(), *36*, *37*, *39*, *51*, *52*, *72*, *76*, *83*, *102*, *103*, *118*
mwb_structure, 86
mwb_study_source, 87
mwb_study_source(), *9*, *16*, *31*, *78*, *80*
mz_match, 89
mzrt_match, 90
normalise_lipids, 91
normalise_strings, 93
normalise_strings(), *66*, *108*, *125*

nothing
(combine_records_helper_functions), 42

opsin_lookup, 94

paste(), *39*
PathBank_metabolite_database, 95
pivot_columns, 97
prioritise
(combine_records_helper_functions), 42
prioritise_columns, 98
pubchem_compound_lookup, 99
pubchem_compound_lookup(), *102*
pubchem_id_exchange, 101
pubchem_id_exchange(), *37*, *39*, *52*, *72*, *76*, *83*, *86*, *118*
pubchem_property_lookup, 103
pubchem_property_lookup(), *102*
pubchem_structure, 105
pubchem_widget, 106

racemic_dictionary, 108
rdata_database, 108
rdata_database(), *9*, *15*, *23*, *56*, *66*, *110, 111*
rds_cache, 109
rds_cache(), *9*, *15*, *23*, *56*, *66*, *109*, *111*
rds_database, 110
rds_database(), *9*, *15*, *23*, *56*, *66*, *109, 110*
read_database, 111
read_database,annotation_database-method
(read_database), 111
read_database,AnnotationDb_database-method
(read_database), 111
read_database,BiocFileCache_database-method
(read_database), 111
read_database,excel_database-method
(read_database), 111
read_database,github_file-method
(read_database), 111
read_database,MTox700plus_database-method
(read_database), 111
read_database,mwb_refmet_database-method
(read_database), 111
read_database,PathBank_metabolite_database-method
(read_database), 111
read_database,rdata_database-method
(read_database), 111
read_database,rds_database-method
(read_database), 111
read_database,sqlite_database-method
(read_database), 111
read_database,zenodo_file-method
(read_database), 111
read_source, 112
read_source(), *69*
read_source,annotation_database-method
(read_source), 112
read_source,annotation_source-method
(read_source), 112
read_source,cd_source-method
(read_source), 112
read_source,ls_source-method
(read_source), 112
read_source,mspurity_source-method
(read_source), 112
read_source,mwb_study_source-method
(read_source), 112
remove_columns, 113
rename_columns, 115
required_cols, 116
required_cols,annotation_source-method
(required_cols), 116
rest_api, 117
rest_api(), *37*, *39*, *52*, *72*, *76*, *83*, *86*, *103*
rlang::quosure, *60*
rt_match, 118

select_columns, 119
select_exact
(combine_records_helper_functions), 42
select_grade
(combine_records_helper_functions), 42
select_match
(combine_records_helper_functions), 42
select_max
(combine_records_helper_functions), 42
select_min
(combine_records_helper_functions), 42
split_column, 121
split_records, 122
sqlite_database, 123

sqlite_database(), *27*, *75*
strsplit, *121*

tidyselect::eval_select, *113*, *119*
tidyselect::eval_select(), *114*, *120*
trim_whitespace, 124
trimws(), *124*, *125*
tripeptide_dictionary, 125

unique_records, 126
unzip_before_cache, 126
upset_filters *(*upset_min_size*)*, 127
upset_intersections *(*upset_min_size*)*, 127
upset_max_groups *(*upset_min_size*)*, 127
upset_min_groups *(*upset_min_size*)*, 127
upset_min_size, 127

vertical_join, 129
vertical_join,annotation_source,annotation_source-method *(*vertical_join*)*, 129
vertical_join,list,missing-method *(*vertical_join*)*, 129

wherever, *60*, 130
wherever(), *61*
write_database, 131
write_database,annotation_database-method *(*write_database*)*, 131
write_database,rds_database-method *(*write_database*)*, 131
write_database,sqlite_database-method *(*write_database*)*, 131

zenodo_file, 132