Package ‘maldipickr’

December 14, 2023

Title Dereplicate and Cherry-Pick Mass Spectrometry Spectra

Version 1.3.0

Description Convenient wrapper functions for the analysis of matrix-assisted laser desorption/ionization-time-of-flight (MALDI-TOF) spectra data in order to select only representative spectra (also called cherry-pick). The package covers the preprocessing and dereplication steps (based on Strejcek, Smrhova, Junkova and Uhlik (2018) <doi:10.3389/fmicb.2018.01294>) needed to cluster MALDI-TOF spectra before the final cherry-picking step. It enables the easy exclusion of spectra and/or clusters to accommodate complex cherry-picking strategies. Alternatively, cherry-picking using taxonomic identification MALDI-TOF data is made easy with functions to import inconsistently formatted reports.

License GPL (>= 3)

URL https://github.com/ClavelLab/maldipickr

BugReports https://github.com/ClavelLab/maldipickr/issues

Depends R (>= 3.2.0)

Imports dplyr, lifecycle, magrittr, MALDIquant, readBrukerFlexData, rlang, stats, tibble, tidyr, tidyselect, tools, utils

Suggests knitr, rmarkdown, spelling, testthat

VignetteBuilder knitr

Config/fusen/version 0.5.2

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

NeedsCompilation no

Author Charlie Pauvert [aut, cre, cph] (<https://orcid.org/0000-0001-9832-2507>), David Wylensek [ctb] (<https://orcid.org/0000-0002-8424-5712>), Selina Nüchtern [ctb], Thomas Clavel [ctb, fnd, cph] (<https://orcid.org/0000-0002-7229-5595>)

Maintainer Charlie Pauvert <cpauvert@ukaachen.de>

Repository CRAN

Date/Publication 2023-12-14 12:30:02 UTC

R topics documented:

`check_spectra` *Evaluate the spectra regularities*

Description

Assess whether all the spectra in the list are not empty, of the same length and correspond to profile data.

Usage

```
check_spectra(spectra_list, tolerance = sqrt(.Machine$double.eps))
```

Arguments

`spectra_list`	A list of MALDIquant::MassSpectrum objects
`tolerance`	A numeric indicating the accepted tolerance to the spectra length. The default value is the machine numerical precision and is close to 1.5e-8.

Value

A list of logical vectors of length `spectra_list` indicating if the spectra are empty (`is_empty`), of an odd length (`is_outlier_length`) or not a profile spectra (`is_not_regular`).

See Also

process_spectra

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import the six spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)
# Display the list of checks, with FALSE where no anomaly is detected
check_spectra(spectra_list)
# The overall sanity can be checked with Reduce
Reduce(any, check_spectra(spectra_list)) # Should be FALSE
```

---

delineate_with_identification
Delineate clusters from taxonomic identifications

Description

From the report of taxonomic identification produced by the Bruker MALDI Biotyper spectra sharing the same identification are labeled in the same cluster. Spectra with unknown identification (e.g., due to database completeness) are set in unique cluster.

Usage

```
delineate_with_identification(tibble_report)
```

Arguments

`tibble_report`	A tibble of *n* rows, with *n* the number of spectra, produced by `read_biotyper_report()` or `read_many_biotyper_reports()`. The long format and the best hits options are expected to be used in these functions to produce a compliant input tibble.

Details

As all unknown identification are considered unique clusters *within one input tibble*, it is important to consider whether the taxonomic identifications come from a single report or multiple reports, depending on the research question. A message is displayed to confirm from which type of reports the delineation was done.

Value

A tibble of n rows for each spectra and 3 columns:

- `name`: the spectra names from the `name` column from the output of either `read_biotyper_report()` or `read_many_biotyper_reports()`.
- `membership`: integers stating the cluster number to which the spectra belong to. It starts from 1 to c, the total number of clusters.
- `cluster_size`: integers indicating the total number of spectra in the corresponding cluster.

See Also

delineate_with_similarity

Examples

```
report_unknown <- read_biotyper_report(
  system.file("biotyper_unknown.csv", package = "maldipickr")
)
delineate_with_identification(report_unknown)
```

---

delineate_with_similarity

Delineate clusters from a similarity matrix

Description

From a matrix of spectra similarity (e.g., with the cosine metric, or Pearson product moment), infer the species clusters based on a threshold **above** (or **equal to**) which spectra are considered alike.

Usage

```
delineate_with_similarity(sim_matrix, threshold, method = "complete")
```

Arguments

`sim_matrix`	A $n \times n$ similarity matrix, with n the number of spectra. Columns should be named as the rows.
`threshold`	A numeric value indicating the minimal similarity between two spectra. Adjust accordingly to the similarity metric used.
`method`	The method of hierarchical clustering to use. The default and recommended method is "complete", but any methods from stats::hclust are valid.

Details

The similarity matrix is converted to a distance matrix by subtracting the value one. This approach works for cosine similarity and positive correlations that have an upper bound of 1. Clusters are then delineated using hierarchical clustering. The default method of hierarchical clustering is the complete linkage (also known as farthest neighbor clustering) to ensure that the within-group minimum similarity of each cluster respects the threshold. See the Details section of stats::hclust for others valid methods to use.

Value

A tibble of n rows for each spectra and 3 columns:

- `name`: the rownames of the similarity matrix indicating the spectra names
- `membership`: integers stating the cluster number to which the spectra belong to. It starts from 1 to c, the total number of clusters.
- `cluster_size`: integers indicating the total number of spectra in the corresponding cluster.

See Also

For similarity metrics: `coop::tcosine`, `stats::cor`, `Hmisc::rcorr`. For using taxonomic identifications for clusters : delineate_with_identification. For further analyses: set_reference_spectra.

Examples

```
# Toy similarity matrix between the six example spectra of
#  three species. The cosine metric is used and a value of
#  zero indicates dissimilar spectra and a value of one
#  indicates identical spectra.
cosine_similarity <- matrix(
  c(
    1, 0.79, 0.77, 0.99, 0.98, 0.98,
    0.79, 1, 0.98, 0.79, 0.8, 0.8,
    0.77, 0.98, 1, 0.77, 0.77, 0.77,
    0.99, 0.79, 0.77, 1, 1, 0.99,
    0.98, 0.8, 0.77, 1, 1, 1,
    0.98, 0.8, 0.77, 0.99, 1, 1
  ),
  nrow = 6,
  dimnames = list(
    c(
      "species1_G2", "species2_E11", "species2_E12",
      "species3_F7", "species3_F8", "species3_F9"
    ),
    c(
      "species1_G2", "species2_E11", "species2_E12",
      "species3_F7", "species3_F8", "species3_F9"
    )
  )
)
# Delineate clusters based on a 0.92 threshold applied
#  to the similarity matrix
```

```
delineate_with_similarity(cosine_similarity, threshold = 0.92)
```

`get_spectra_names` *Extract spectra names and check for uniqueness*

Description

Given the list of raw spectra, `get_spectra_names()` extracts the spectra names using the file metadata, and warns if the associated sanitized names are not unique.

Usage

```
get_spectra_names(spectra_list)
```

Arguments

`spectra_list`	A list of MALDIquant::MassSpectrum objects.

Value

A tibble with four columns

- `sanitized_name`: spectra names based on `fullName` where dots and dashes are converted to underscores
- `name`: spectra name using the `name` label in the spectra metadata
- `fullName`: spectra full name using the `fullName` label in the spectra metadata
- `file`: the path to the raw spectra data

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import the six spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)
# Extract the names
get_spectra_names(spectra_list)

# Artificially create duplicated entries to show the warning
get_spectra_names(spectra_list[c(1,1)])
```

`import_biotyper_spectra` *Importing spectra from the Bruker MALDI Biotyper device*

Description

This function is a wrapper around the `readBrukerFlexData::readBrukerFlexDir()` to read both acqus and acqu MALDI files.

Usage

```
import_biotyper_spectra(
  biotyper_directory,
  remove_calibration = c("BTS", "Autocalibration")
)
```

Arguments

`biotyper_directory`
: A path to the folder tree with the spectra to be imported.

`remove_calibration`
: A vector of characters used as regex to indicate which (calibration) spectra are going to be removed.

Details

When using `readBrukerFlexData::readBrukerFlexDir()` on acqus files (instead of the native acqu files), the function will fail with the following error message:

```
Error in .readAcquFile(fidFile = fidFile, verbose = verbose) :
File '/data/maldi_dir/targetA/0_D10/1/1SLin/acqu' doesn't exists!
```

But it turns out that acqu and acqus files are the same, so the function here create acqu symbolic links that point to acqus files.

Value

A list of MALDIquant::MassSpectrum objects

See Also

check_spectra, process_spectra

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import the six spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)
# Display the list of spectra
spectra_list
```

`import_spede_clusters` *Import clusters results generated by SPeDE*

Description

Reformat the table output from the analysis of raw Bruker MALDI Biotyper spectra by the SPeDE tool from Dumolin et al. (2019) to be consistent with the Strejcek et al. (2018) procedure followed in the maldipickr package.

Usage

```
import_spede_clusters(path)
```

Arguments

`path`	Path to the comma separated table generated by SPeDE

Value

A tibble with the following columns:

- `name`: a character denoting the spectra name (all spaces, dashes and dots are replaced by underscores "_" in SPeDE)
- `membership`: integers stating the cluster number to which the spectra belong to. It starts from 1 to c, the total number of clusters.
- `cluster_size`: integers indicating the total number of spectra in the corresponding cluster.
- `quality`: a character indicating the spectra quality category by SPeDE, out of GREEN, ORANGE and RED.
- `is_reference`: a logical indicating whether the corresponding spectra is a reference spectra of the cluster.

References

Dumolin C, Aerts M, Verheyde B, Schellaert S, Vandamme T, Van Der Jeugt F, De Canck E, Cnockaert M, Wieme AD, Cleenwerck I, Peiren J, Dawyndt P, Vandamme P, & Carlier A. (2019). "Introducing SPeDE: High-Throughput Dereplication and Accurate Determination of Microbial Diversity from Matrix-Assisted Laser Desorption–Ionization Time of Flight Mass Spectrometry Data". *MSystems* 4(5). doi:10.1128/msystems.00437-19.

See Also

https://github.com/LM-UGent/SPeDE

Examples

```
# Reformat the output from SPeDE table
# https://github.com/LM-UGent/SPeDE
import_spede_clusters(
  system.file("spede.csv", package = "maldipickr")
)
```

---

is_well_on_edge *Identify the wells on the plate's edge*

Description

Identify the wells on the plate's edge

Usage

```
is_well_on_edge(
  well_number,
  plate_layout = c(96, 384),
  edges = c("top", "bottom", "left", "right"),
  details = FALSE
)
```

Arguments

well_number	A vector of positive numeric well identifier
plate_layout	An integer indicating the maximum number of well on the plate
edges	A character vector pointing which plate edges should be considered
details	A logical controlling whether a data.frame with more details should be returned

Details

Flag the wells located on the edges of a 96- or 384-well plate, based on the following well numbering:

- Well numbers start at 1
- Well are numbered from left to right and then top to bottom of the plate.

Value

A logical vector, the same length as `well_number` indicating whether the well is on the edge. If `details = TRUE`, the function returns a data.frame that complements the logical vector with the `well_number`, row and column positions.

Examples

```
# Logical vector indicating whether the wells are on the four edges
is_well_on_edge(1:96, plate_layout = 96)
# More details can be obtained to verify the results
well_df <- is_well_on_edge(1:96, plate_layout = 96, details = TRUE)
# And the resulting prediction displayed
matrix(well_df$is_edge, ncol = max(well_df$col), byrow = TRUE)
```

---

merge_processed_spectra

Merge multiple processed spectra and peaks

Description

Aggregate multiple processed spectra, their associated peaks and metadata into a feature matrix and a concatenated metadata table.

Usage

```
merge_processed_spectra(
  processed_spectra,
  remove_peakless_spectra = TRUE,
  interpolate_missing = TRUE
)
```

Arguments

`processed_spectra`
: A list of the processed spectra and associated peaks and metadata in two possible formats:
  - A list of **in-memory objects** (named `spectra`, `peaks`, `metadata`) produced by process_spectra.
  - **[Deprecated]** A list of **paths** to RDS files produced by process_spectra when using the `rds_prefix` option.

`remove_peakless_spectra`
: A logical indicating whether to discard the spectra without detected peaks.

`interpolate_missing`
: A logical indicating if intensity values for missing peaks should be interpolated from the processed spectra signal or left NA which would then be converted to 0.

Value

A $n \times p$ matrix, with n spectra as rows and p features as columns that are the peaks found in all the processed spectra.

See Also

process_spectra, the "Value" section in MALDIquant::intensityMatrix

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import the six spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)
# Transform the spectra signals according to Strejcek et al. (2018)
processed <- process_spectra(spectra_list)
# Merge the spectra to produce the feature matrix
fm <- merge_processed_spectra(list(processed))
# The feature matrix has 6 spectra as rows and
#  35 peaks as columns
dim(fm)
# Notice the difference when the interpolation is turned off
fm_no_interpolation <- merge_processed_spectra(
  list(processed),
  interpolate_missing = FALSE
)
sum(fm == 0) # 0
sum(fm_no_interpolation == 0) # 68

# Multiple runs can be aggregated using list()
# Merge the spectra to produce the feature matrix
fm_all <- merge_processed_spectra(list(processed, processed, processed))
# The feature matrix has 3×6=18 spectra as rows and
#  35 peaks as columns
dim(fm_all)
```

---

pick_spectra *Cherry-pick Bruker MALDI Biotyper spectra*

Description

Using the clusters information, and potential additional metadata as external criteria, spectra are labeled as to be picked for each cluster. Note that some spectra and therefore clusters can be explicitly removed (*masked*) from the picking decision if they have been previously picked or should be discarded, using logical columns in the metadata table. If no metadata are provided, the reference spectra of each cluster will be picked.

Usage

```
pick_spectra(
  cluster_df,
```

```
    metadata_df = NULL,
    criteria_column = NULL,
    hard_mask_column = NULL,
    soft_mask_column = NULL,
    is_descending_order = TRUE,
    is_sorted = FALSE
  )
```

Arguments

`cluster_df` A tibble with clusters information from the delineate_with_similarity or the import_spede_clusters function.

`metadata_df` Optional tibble with relevant metadata to guide the picking process (e.g., OD600).

`criteria_column` Optional character indicating the column in `metadata_df` to be used as a criteria.

`hard_mask_column` Column name in the `cluster_df` or `metadata_df` tibble indicating whether the spectra, **and the clusters to which they belong** should be discarded (`TRUE`) or not (`FALSE`) before the picking decision.

`soft_mask_column` Column name in the `cluster_df` or `metadata_df` tibble indicating whether the spectra should be discarded (`TRUE`) or not (`FALSE`) before the picking decision.

`is_descending_order` Optional logical indicating whether to sort the `criteria_column` from the highest-to-lowest value (`TRUE`) or lowest-to-highest (`FALSE`).

`is_sorted` Optional logical to indicate that the `cluster_df` is already sorted by cluster based on (usually multiple) internal criteria to pick the first of each cluster. This flag is **overridden** if a `metadata_df` is provided.

Value

A tibble with as many rows as `cluster_df` with an additional logical column named `to_pick` to indicate whether the colony associated to the spectra should be picked. If `metadata_df` is provided, then additional columns from this tibble are added to the returned tibble.

See Also

delineate_with_similarity, set_reference_spectra. For a useful utility function to soft-mask specific spectra: is_well_on_edge.

Examples

```
# 0. Load a toy example of a tibble of clusters created by
#   the `delineate_with_similarity` function.
clusters <- readRDS(
  system.file("clusters_tibble.RDS",
    package = "maldipickr"
  )
```

```
)
# 1. By default and if no other metadata are provided,
#   the function picks reference spectra for each clusters.
#
# N.B: The spectra `name` and `to_pick` columns are moved to the left
# only for clarity using the `relocate()` function.
#
pick_spectra(clusters) %>%
  dplyr::relocate(name, to_pick) # only for clarity

# 2.1 Simulate OD600 values with uniform distribution
#  for each of the colonies we measured with
#  the Bruker MALDI Biotyper
set.seed(104)
metadata <- dplyr::transmute(
  clusters,
  name = name, OD600 = runif(n = nrow(clusters))
)
metadata

# 2.2 Pick the spectra based on the highest
#   OD600 value per cluster
pick_spectra(clusters, metadata, "OD600") %>%
  dplyr::relocate(name, to_pick) # only for clarity

# 3.1 Say that the wells on the right side of the plate are
#   used for negative controls and should not be picked.
metadata <- metadata %>% dplyr::mutate(
  well = gsub(".*[A-Z]([0-9]{1,2}$)", "\\1", name) %>%
    strtoi(),
  is_edge = is_well_on_edge(
    well_number = well, plate_layout = 96, edges = "right"
  )
)

# 3.2 Pick the spectra after discarding (or soft masking)
#   the spectra indicated by the `is_edge` column.
pick_spectra(clusters, metadata, "OD600",
  soft_mask_column = "is_edge"
) %>%
  dplyr::relocate(name, to_pick) # only for clarity

# 4.1 Say that some spectra were picked before
#   (e.g., in the column F) in a previous experiment.
# We do not want to pick clusters with those spectra
#   included to limit redundancy.
metadata <- metadata %>% dplyr::mutate(
  picked_before = grepl("_F", name)
)
# 4.2 Pick the spectra from clusters without spectra
#   labeled as `picked_before` (hard masking).
pick_spectra(clusters, metadata, "OD600",
  hard_mask_column = "picked_before"
```

```
) %>%
  dplyr::relocate(name, to_pick) # only for clarity
```

`process_spectra` *Process Bruker MALDI Biotyper spectra* à la *Strejcek et al. (2018)*

Description

Process Bruker MALDI Biotyper spectra *à la* Strejcek et al. (2018)

Usage

```
process_spectra(
  spectra_list,
  spectra_names = get_spectra_names(spectra_list),
  rds_prefix = deprecated()
)
```

Arguments

`spectra_list`	A list of MALDIquant::MassSpectrum objects.
`spectra_names`	A tibble::tibble (or data.frame) of sanitized spectra names by default from get_spectra_names. If provided manually, the column `sanitized_name` will be used to name the spectra.
`rds_prefix`	**[Deprecated]** Writing to disk as RDS is no longer supported. A character indicating the prefix for the `.RDS` output files to be written in the `processed` directory. By default, no prefix are given and thus no files are written.

Details

Based on the original implementation, the function performs the following tasks:

1. Square-root transformation
2. Mass range trimming to 4-10 kDa as they were deemed most determinant by Strejcek et al. (2018)
3. Signal smoothing using the Savitzky-Golay method and a half window size of 20
4. Baseline correction with the SNIP procedure
5. Normalization by Total Ion Current
6. Peak detection using the SuperSmoother procedure and with a signal-to-noise ratio above 3
7. Peak filtering. This step has been added to discard peaks with a negative signal-to-noise ratio probably due to being on the edge of the mass range.

Value

A named list of three objects:

- `spectra`: a list the length of the spectra list of MALDIquant::MassSpectrum objects.
- `peaks`: a list the length of the spectra list of MALDIquant::MassPeaks objects.
- `metadata`: a tibble indicating the median signal-to-noise ratio (`SNR`) and peaks number for all spectra list (`peaks`), with spectra names in the `name` column.

Note

The original R code on which this function is based is accessible at: `https://github.com/strejcem/MALDIvs16S`

References

Strejcek M, Smrhova T, Junkova P & Uhlik O (2018). “Whole-Cell MALDI-TOF MS versus 16S rRNA Gene Analysis for Identification and Dereplication of Recurrent Bacterial Isolates.” *Frontiers in Microbiology* 9 `doi:10.3389/fmicb.2018.01294`.

See Also

import_biotyper_spectra and check_spectra for the inputs and merge_processed_spectra for further analysis.

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import the six spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)
# Transform the spectra signals according to Strejcek et al. (2018)
processed <- process_spectra(spectra_list)
# Overview of the list architecture that is returned
#  with the list of processed spectra, peaks identified and the
#  metadata table
str(processed, max.level = 2)
# A detailed view of the metadata with the median signal-to-noise
#  ratio (SNR) and the number of peaks
processed$metadata
```

read_biotyper_report *Importing Bruker MALDI Biotyper CSV report*

Description

The header-less table exported by the Compass software in the Bruker MALDI Biotyper device is separated by semi-colons and has empty columns which prevent an easy import in R. This function reads the report correctly as a tibble.

Usage

```
read_biotyper_report(path, best_hits = TRUE, long_format = TRUE)
```

Arguments

`path`	Path to the semi-colon separated table
`best_hits`	A logical indicating whether to return only the best hits for each target analyzed
`long_format`	A logical indicating whether the table is in the long format (many rows) or wide format (many columns) when showing all the hits. This option has no effect when `best_hits = TRUE`.

Details

The header-less table contains identification information for each target processed by the Biotyper device and once processed by the `read_biotyper_report`, the following seven columns are available in the tibble, *when using the* `best_hits = TRUE` *option*:

- `name`: a character indicating the name of the spot of the MALDI target (i.e., plate)
- `sample_name`: the character string provided during the preparation of the MALDI target (i.e., plate)
- `hit_rank`: an integer indicating the rank of the hit for the corresponding target and identification
- `bruker_quality`: a character encoding the quality of the identification with potentially multiple "+" symbol or only one "-"
- `bruker_species`: the species name associated with the MALDI spectrum analyzed.
- `bruker_taxid`: the NCBI Taxonomy Identifier of the species name in the column species
- `bruker_hash`: a hash from an undocumented checksum function probably to encode the database entry.
- `bruker_log`: the log-score of the identification.

When all hits are returned (with `best_hits = FALSE`), the default output format is the long format (`long_format = TRUE`), meaning that the previous columns remain unchanged, but all hits are now returned, thus increasing the number of rows.

When all hits are returned (with `best_hits = FALSE`) *using the wide format* `(long_format = FALSE), the two columns nan`
contain the hit rank, **creating a tibble of 52 columns**:

- bruker_01_quality
- bruker_01_species
- bruker_01_taxid
- bruker_01_hash
- bruker_01_log
- bruker_02_quality
- ...
- bruker_10_species
- bruker_10_taxid
- bruker_10_hash
- bruker_10_log

Value

A tibble of 7 columns (best_hits = TRUE) or 52 columns (best_hits = FALSE). See Details for the description of the columns.

Note

A report that contains only spectra with no peaks found will return a tibble of 0 rows and a warning message.

See Also

read_many_biotyper_reports

Examples

```
# Get a example Bruker report
biotyper <- system.file("biotyper.csv", package = "maldipickr")
# Import the report as a tibble
report_tibble <- read_biotyper_report(biotyper)
# Display the tibble
report_tibble
```

---

read_many_biotyper_reports

Importing a list of Bruker MALDI Biotyper CSV reports

Description

Importing a list of Bruker MALDI Biotyper CSV reports

Usage

```
read_many_biotyper_reports(path_to_reports, report_ids, best_hits = TRUE, ...)
```

Arguments

path_to_reports
: A vector of paths to the csv files to be imported by read_biotyper_report().

report_ids
: A vector of character names for each of the reports.

best_hits
: A logical indicating whether to return only the best hit in the read_biotyper_report() function.

...
: Name-value pairs to be passed on to dplyr::mutate()

Value

A tibble just like the one returned by the read_biotyper_report() function, except that the name of the spot of the MALDI target (i.e., plate) is registered to the original_name column (instead of the name column), and the column name consist in the provided report_ids used as a prefix of the original_name column.

Note

The report identifiers are sanitized to convert all dashes (-) as underscores (_).

See Also

read_biotyper_report

Examples

```
# List of Bruker MALDI Biotyper reports
reports_paths <- system.file(
  c("biotyper.csv", "biotyper.csv", "biotyper.csv"),
  package = "maldipickr"
)
# Read the list of reports and combine them in a single tibble
read_many_biotyper_reports(
  reports_paths,
  report_ids = c("first", "second", "third"),
  # Additional metadata below are passed to dplyr::mutate
  growth_temperature = 37.0
)
```

---

remove_spectra — *Remove (raw or processed) spectra*

Description

The remove_spectra() function is used to discard specific spectra from (1) raw spectra list by removing them, or (2) processed spectra by removing them from the spectra, peaks and metadata objects.

Usage

```
remove_spectra(spectra_list, to_remove)
```

Arguments

spectra_list	A list of MALDIquant::MassSpectrum objects OR A list of processed spectra from process_spectra
to_remove	The spectra to be removed. In the case of raw spectra: a logical vector same size of spectra_list or from check_spectra function. In the case of processed spectra: names of the spectra as formatted in get_spectra_names in the sanitized_name column.

Value

The same object as spectra_list minus the spectra in to_remove.

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
directory_biotyper_spectra <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
)
# Import only the first two spectra
spectra_list <- import_biotyper_spectra(directory_biotyper_spectra)[1:2]
# Introduce artificially an empty raw spectra
spectra_list <- c(spectra_list, MALDIquant::createMassSpectrum(0, 0))
# Empty spectra are detected by `check_spectra()`
#   and can be removed by `remove_spectra()`
spectra_list %>%
  remove_spectra(to_remove = check_spectra(.))

# Get an example processed spectra
processed_path <- system.file(
    "three_processed_spectra_with_one_peakless.RDS",
    package = "maldipickr")
processed <- readRDS(processed_path) %>% list()

# Remove a specific spectra
remove_spectra(processed, "empty_H12")
```

---

set_reference_spectra *Set a reference spectrum for each cluster*

Description

Define a high-quality spectra as a representative spectra of the cluster based on the highest median signal-to-noise ratio and the number of detected peaks

Usage

```
set_reference_spectra(cluster_df, metadata_df)
```

Arguments

`cluster_df` A tibble of n rows for each spectra produced by delineate_with_similarity function with at least the following columns:

- `name`: the rownames of the similarity matrix indicating the spectra names
- `membership`: integers stating the cluster number to which the spectra belong to. It starts from 1 to c, the total number of clusters.
- `cluster_size`: integers indicating the total number of spectra in the corresponding cluster.

`metadata_df` A tibble of n rows for each spectra produced by the process_spectra function with median signal-to-noise ratio (`SNR`), peaks number (`peaks`), and spectra names in the `name` column.

Value

A merged tibble in the same order as `cluster_df` with both the columns of `cluster_df` and `metadata_df`, as well as a logical column `is_reference` indicating if the spectrum is the reference spectra of the cluster.

See Also

delineate_with_similarity, pick_spectra

Examples

```
# Get an example directory of six Bruker MALDI Biotyper spectra
# Import the six spectra and
# Transform the spectra signals according to Strejcek et al. (2018)
processed <- system.file(
  "toy-species-spectra",
  package = "maldipickr"
) %>%
  import_biotyper_spectra() %>%
  process_spectra()

# Toy similarity matrix between the six example spectra of
#  three species. The cosine metric is used and a value of
#  zero indicates dissimilar spectra and a value of one
#  indicates identical spectra.
cosine_similarity <- matrix(
  c(
    1, 0.79, 0.77, 0.99, 0.98, 0.98,
    0.79, 1, 0.98, 0.79, 0.8, 0.8,
    0.77, 0.98, 1, 0.77, 0.77, 0.77,
    0.99, 0.79, 0.77, 1, 1, 0.99,
    0.98, 0.8, 0.77, 1, 1, 1,
    0.98, 0.8, 0.77, 0.99, 1, 1
```

```
    ),
    nrow = 6,
    dimnames = list(
      c(
        "species1_G2", "species2_E11", "species2_E12",
        "species3_F7", "species3_F8", "species3_F9"
      ),
      c(
        "species1_G2", "species2_E11", "species2_E12",
        "species3_F7", "species3_F8", "species3_F9"
      )
    )
  )
# Delineate clusters based on a 0.92 threshold applied
#  to the similarity matrix
clusters <- delineate_with_similarity(
  cosine_similarity,
  threshold = 0.92
)

# Set reference spectra with the toy example
set_reference_spectra(clusters, processed$metadata)
```

Index