Package 'mums2'

June 25, 2026

Title Microbial Ecology by Tandem Mass Spectrometry

Version 0.1.1

Description Tools that researchers can use to analyze untargeted metabolomics data generated using tandem mass spectroscopy from microbial communities. The overall approach taken to analyze metabolomics data parallels that used to analyze microbial communities using 16S rRNA gene sequencing data. Thus, we have a number of methods a user is able to use to generate data. Firstly, users can import Mass Spectrometry 1(MS1) data and filter it. Users are then able to match Mass Spectrometry 2(MS2) data to the filtered (or unfiltered) MS1 data. With the matched data users are able to cluster it, annotate it, predict de novo chemical formulas and calculate alpha and beta diversity. For chemical formula predictions, this was the method used; ``Towards de novo identification of metabolites by analyzing tandem mass spectra" (Sebastian Böcker, Florian Rasche (2008) <doi:10.1093/bioinformatics/btn270>). The similarity/dissimilarity calculations we used to cluster our data together was: ``Spectral entropy outperforms MS/MS dot product similarity for small-molecule compound identification" (Li, Y., Kind, T., Folz, J. et al. (2021) <doi:10.1038/s41592-021-01331-z>) and ``Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking" (Wang, M., Carver, J., Phelan, V. et al. (2021) <doi:10.1038/nbt.3597>).

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/mums2/mums2, https://www.mums2.org/mums2/

BugReports https://github.com/mums2/mums2/issues

LinkingTo sitmo, Rcpp, RcppThread, testthat, RcppProgress

Imports clustur, Rcpp, data.table, mpactr, utils, stats, parallel, xml2, RaMS

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tidyverse, networkD3

Config/testthat/edition 3

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

NeedsCompilation yes

Author Allison Mason [aut] (ORCID: <https://orcid.org/0000-0003-1339-1592>),
Gregory Johnson [aut] (ORCID: <https://orcid.org/0009-0008-3890-0297>),
Patrick Schloss [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6935-4275>),
Anton Pervukhin [ctb, cph],
Florian Rasche [ctb, cph],
Henner Sudek [ctb, cph],
Marcel Martin [ctb, cph],
Yuanyue Li [ctb, cph]

Maintainer Patrick Schloss <pschloss@umich.edu>

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2026-06-25 05:30:02 UTC

Contents

`alpha_summary` *Alpha Diversity Summary*

Description

Alpha Diversity calculates the amount of diversity in a single sample. We can conduct this analysis using your created community object. We support the use of Shannon and Simpson diversity index.

Usage

```
alpha_summary(
  community_object,
  size,
  threshold,
  diversity_index = c("shannon", "simpson"),
  subsample = TRUE,
  number_of_threads = 1,
  iterations = 100,
  seed = 123
)
```

Arguments

`community_object`	the object created from the `create_community_object()` function.
`size`	the size you wish to rarefy your diversity matrix to.
`threshold`	the threshold you want your species to reach before it is included in the rarefaction sum.
`diversity_index`	the diversity index you wish to calculate diversity, the options are shannon, simpson, or richness. You may also compute many indexes at the same time using a vector (ie. c("shannon", "simpson")).
`subsample`	if true, we will rarefy the data before we run the diversity calculations. Default is TRUE.
`number_of_threads`	the number of threads you wish to use for this calculation.
`iterations`	the amount of times you wish to run your calculation.
`seed`	the RNG (random number generator) seed you would like to use.

Value

a data.frame object that shows the dissimilarity in samples.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)

cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data,
 cutoff = 0.3, cluster_method = "opticlust")

community_object <- create_community_matrix_object(cluster_results)

alpha_summary(community_object = community_object, size = 400,
              threshold = 100,
              diversity_index = c("shannon", "simpson", "richness"),
              subsample = TRUE, iterations = 1, number_of_threads = 1)
```

---

annotate_ms2 *Annotate LC-MS/MS features*

Description

Annotate query LC-MS/MS features in a mass_data object given a reference list.

annotate_ms2() allows for annotation of mass spectrometry features. Similarity between query and reference level 2 spectra are determined via spectral scoring methods. Currently scoring methods "gnps" and "spectral_entropy" are supported. The scoring method is specified by the score_params argument. score_params is a list of parameters for the chosen scoring method. Parameters for "gnps" and "spectral_entropy" can be created with functions modified_cosine_params() and spec_entropy_params(), respectively. If you are using an hmdb reference database, or a database that does not contain a precursormz please ensure that you expand your ppm to account for the difference.

Usage

```
annotate_ms2(mass_data, reference, scoring_params,
                    ppm, min_score,
                    chemical_min_score,
                    cluster_data = NULL, min_peaks = 0,
                    number_of_threads = 1)
```

Arguments

mass_data	The object generated from ms2_ms1_compare().
reference	Your reference database generated from the read_msp() function. We currently only support msp files.
scoring_params	Parameters for scoring method to be applied. This can be either modified_cosine_params() or spec_entropy_params().
ppm	Parts per million. MS2 scans with a difference in ppm less than or equal to this value will be scored.
min_score	Similarity score threshold to determine a match for annotation. Comparisons with scores below this value will not be reported.
chemical_min_score	data2
cluster_data	the cluster object that was generated by cluster_data(). Can also remain NULL if you wish to annotate the data without omu information.
min_peaks	the minimum number of peaks that need to be present before you compare the ms2 spectra.
number_of_threads	the number of threads you wish to use for this calculation.

Value

A data.frame with all comparisons with scores above the threshold. Information for the query scan include query_ms1_id (the variable_id for features in expression_data of the mass_data object) "query_ms2_id" (the ms2_spectrum_id in the query object), "query_mz" (the precursor mz for the scan), and "query_rt" (the retention time for the scan). query_mz and query_rt are derived from the ms2 matches data. A column ("ref_idx) is included to report the location for the matching reference molecule in "reference". Scores are reported in the "score" column. query_formula and chemical_similarity are also reported. Annotation information is returned given the information provided in the reference used as input.

a data.frame object containing annotations

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
```

```
                    format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)
 massbank <- read_msp(mums2_example("MSMS-Neg-Respect.msp"))
 annotations <- annotate_ms2(mass_data = matched_data,
   reference = massbank, scoring_params = modified_cosine_params(0.5),
   ppm = 1.6e3,
   min_score =  0.5, chemical_min_score = 0,
   number_of_threads = 2)
```

`change_rt_to_seconds_or_minute` *Change RT time to minutes or seconds*

Description

This function will change your ms1 peak table rt time to rt time in seconds or minutes. This modification happens in place (or by reference), so the `mpactr_object` will be updated.

Usage

```
change_rt_to_seconds_or_minute(mpactr_object, rt_type = "seconds")
```

Arguments

`mpactr_object`	The object created from `import_all_data()` file.
`rt_type`	how you want to convert your retention time, your options are minutes, or seconds. defaults to seconds.

Value

a modified `mpactr` object.

Examples

```
data <-
   import_all_data(peak_table =
                    mums2::mums2_example("botryllus_pt_small.csv"),
                    metadata =
                    mums2::mums2_example("boryillus_metadata.csv"),
                    format = "None")
change_rt_to_seconds_or_minute(data, "minutes")
```

cluster_data *Cluster Features*

Description

cluster_data() allows users to cluster features inside the mass data object. This is done by creating a sparse matrix using the distMs2() function and inputting that inside the clutur package. This allows us to easily cluster features that contain an ms2 spectra.

Usage

```
cluster_data(
  distance_df,
  ms2_match_data,
  cutoff = 0.3,
  cluster_method = "opticlust"
)
```

Arguments

distance_df	a distance df that was generated from the distMs2() function.
ms2_match_data	your mass data set object generated from ms2_ms1_compare().
cutoff	the cutoff value you wish to cluster to.
cluster_method	the clustering algorithm you wish to use. The options are: furthest, nearest, weighted, average, and opticlust.

Value

a shared data.frame (or a mothur_cluster object) displaying all the clustered and abundance data.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.6, precursor_thresh = 100,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)
```

```
cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data, cutoff = 0.3, cluster_method = "opticlust")
```

`combined_reference_database` *Combine Databases*

Description

Add another database to your reference database.

Usage

```
combined_reference_database(reference, other_reference)
```

Arguments

`reference`	reference database object.
`other_reference`	your other reference database object

Value

a `reference_database` that includes references from both reference databases.

Examples

```
reference <- read_msp(mums2_example("massbank_example_data.msp"))
reference2 <- read_msp(mums2_example("massbank_example_data.msp"))
combined_reference_database(reference, reference2)
```

`compute_molecular_formulas` *Compute Molecular formula Other*

Description

de novo algorithm for computing molecular formulas. Using fragmentation trees we are able to generate a resultant molecular formula. To ensure efficient we are using a greedy heurstic to generate the resultant formula. Although this may not always result in the correct prediction, it allows us to efficiently calculate a multitudeof chemical formulas.

Usage

```
compute_molecular_formulas(mass_data, parent_ppm = 3, number_of_threads = 1)
```

Arguments

mass_data	your mass_data object generated from ms2_ms1_compare()
parent_ppm	the ppm you wish to generate the candidate molecular formulas.
number_of_threads	the number of threads you wish to use for this calculation.

Value

your mass_data object with an additional character vector of all the predicted formulas.

References

Sebastian Böcker, Florian Rasche, Towards de novo identification of metabolites by analyzing tandem mass spectra, Bioinformatics, Volume 24, Issue 16, August 2008, Pages i49–i55, https://doi.org/10.1093/bioinformatics/b

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 0.1, 1)
compute_molecular_formulas(matched_data, number_of_threads = 2)
```

---

convert_to_group_averages

Convert Samples to Group Averages

Description

To account for users measuring there data in triplicates or other forms of measurement, we have implemented a function that can transform your matched data object to use group averages instead of each sample individually.

Usage

```
convert_to_group_averages(matched_data, mpactr_object)
```

Arguments

matched_data	your mass data set object generated from ms2_ms1_compare().
mpactr_object	The object created from import_all_data().

Value

a mass_data object using group averages

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

matched_data_avg <- convert_to_group_averages(matched_data,
                                              data)
```

---

create_community_matrix

create community matrix

Description

Using your community_object, we are able to convert it into a community matrix for easier usability of the object.

Usage

```
create_community_matrix(cluster_object)
```

Arguments

cluster_object	the result of the cluster_data() function. data <- import_all_data(peak_table = mums2::mums2_example("botryllus_pt_small.csv"), metadata = mums2::mums2_example("boryillus_ format = "None") matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"), data, 1, 6) dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2, score_params = modified_cosine_params(0.5), min_peaks = 0) cluster_results <- cluster_data(distance_df = dist, ms2_match_data = matched_data, cutoff = 0.3, cluster_method = "opticlust") community_matrix <- create_community_matrix_object(cluster_results)

Value

a data.frame object of your community_object.

`create_community_matrix_object` *Create Community Matrix Object.*

Description

Using the data generated from clustering or adding ms2 data to your object, we are able to create a community matrix object. The community matrix object stores the same data a community matrix but within a cpp object. We use this object to conduct analysis more efficiently.

Usage

```
create_community_matrix_object(data)

## S3 method for class 'mass_data'
create_community_matrix_object(data)

## S3 method for class 'mothur_cluster'
create_community_matrix_object(data)
```

Arguments

`data`	the result of the `cluster_data()` function, or just a mass_data object created from `ms2_ms1_compare()`.

Value

a external pointer to an Rcpp object.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)

cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data, cutoff = 0.3, cluster_method = "opticlust")
```

```
community_with_cluster <- create_community_matrix_object(cluster_results)

community_object_mass_data <- create_community_matrix_object(matched_data)
```

`dist_ms2` *Calculate pairwise distance between MS/MS features.*

Description

`dist_ms2` calculates and stores all non-zero distance values above the user defined cutoff (default = 0.3).

Usage

```
dist_ms2(
  data,
  cutoff,
  precursor_threshold,
  score_params,
  min_peaks = 6,
  number_of_threads = 1
)
```

Arguments

`data`	the object generated from `ms2_ms1_compare()`.
`cutoff`	The maximum distance value (`numeric`) to store a pairwise comparison. The default of .3 corresponds to a cosine score of .7, meaning pairs with a score of .7 or higher will be stored in the matrix.
`precursor_threshold`	Precursor mz tolerance. MS2 scans with a difference in precursor mz less than or equal to this value will be scored. Disable this by setting this value to -1 or less.
`score_params`	Parameters for scoring method to be applied. See `modified_cosine_params()` and `spec_entropy_params()` for more details.
`min_peaks`	the minimum number of peaks that need to be present before you compare the ms2 spectra.
`number_of_threads`	the number of threads you wish to use for this calculation.

Details

This function takes a `mass_data` object as input and calculates distance between ms2 peaks. Currently, MS1 features without MS2 peaks returns no distance value. Distance can be calculated with method `"gnps"` or `"spectral_entropy"`. A sparse matrix is returned.

Value

A sparse matrix of class "data.frame"

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")

matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist_gnps <- dist_ms2(data = matched_data,
 cutoff = 0.3, precursor_threshold = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)

dist_entropy <- dist_ms2(data = matched_data,
 cutoff = 0.3, precursor_threshold = 2,
 score_params = spec_entropy_params(), min_peaks = 0,
 number_of_threads = 2)
```

---

dist_shared *Distance Shared*

Description

Beta diversity is calculation that allows us to calculate the differences between two samples. We can conduct this analysis using the created community objects. The available options for beta diversity are: Bray Curtis, Jaccard Dissimilarity, Sorenson index, Hamming Distance, Morista-Horn index, and Yue & Clayton measure of dissimilarity (or thetayc).

Usage

```
dist_shared(
  community_object,
  size,
  threshold,
  diversity_index = "bray",
  subsample = TRUE,
  number_of_threads = 1,
  iterations = 100,
  seed = 123
)
```

Arguments

community_object	the object created from the create_community_object() function.
size	the size you wish to rarefy your diversity matrix to.
threshold	the threshold you want your species to reach before it is included in the rarefaction sum.
diversity_index	the diversity index you wish to calculate diversity. You can choose from: bray, jaccard, soren, hamming, morista, and thetayc.
subsample	if true, we will rarefy the data before we run the diversity calculations. Default is TRUE.
number_of_threads	the number of threads you wish to use for this calculation.
iterations	the amount of times you wish to run your calculation.
seed	the RNG (random number generator) seed you would like to use.

Value

a data.frame object that shows the dissimilarity between all samples.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 6,
 number_of_threads = 2)

cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data,
 cutoff = 0.3, cluster_method = "opticlust")

community_object <- create_community_matrix_object(cluster_results)

dist_shared(community_object = community_object,
 size = 400, threshold = 100, diversity_index = "bray",
 subsample = TRUE, number_of_threads =  1)
```

filter_cv_params *Filter Cv Parameters*

Description

Creates a list of filter cv arguments for the `filter_peak_table()` function

Usage

```
filter_cv_params(cv_threshold = NULL, copy_object = FALSE)
```

Arguments

`cv_threshold`	Coefficient of variation threshold. A lower cv_threshold will result in more stringent filtering and higher reproducibility. Recommended values between 0.2 - 0.5.
`copy_object`	A `boolean` parameter that allows users to return a copied object instead of modifying the object.

Value

a `list` object of arguments needed to call the given mpactr function when supplied to the `filter_peak_table()` wrapper function.

Examples

```
filter_cv_params(0.2)
filter_cv_params(0.2)
```

filter_group_params *Filter Group Parameters*

Description

Creates a list of filter group arguments for the `filter_peak_table()` function

Usage

```
filter_group_params(
  group_threshold = 0.01,
  group_to_remove,
  remove_ions = TRUE,
  copy_object = FALSE
)
```

Arguments

group_threshold	Relative abundance threshold at which to remove ions. Default = 0.01.
group_to_remove	Biological group name to remove ions from.
remove_ions	A boolean parameter. If TRUE failing ions will be removed from the peak table. Default = TRUE.
copy_object	A boolean parameter that allows users to return a copied object instead of modifying the object.

Value

a list object of arguments needed to call the given mpactr function when supplied to the filter_peak_table() wrapper function.

Examples

```
filter_group_params(group_to_remove = "blank")
```

---

filter_insource_ions_params

Filter Insource Ions Parameters

Description

Creates a list of filter insource ions arguments for the filter_peak_table() function

Usage

```
filter_insource_ions_params(cluster_threshold = 0.95, copy_object = FALSE)
```

Arguments

cluster_threshold	Cluster threshold for ion deconvolution. Default = 0.95.
copy_object	A boolean parameter that allows users to return a copied object instead of modifying the object.

Value

a list object of arguments needed to call the given mpactr function when supplied to the filter_peak_table() wrapper function.

Examples

```
filter_insource_ions_params()
```

filter_mispicked_ions_params
Filter Mispicked Ions Parameters

Description

Creates a list of filter mispicked ions arguments for the `filter_peak_table()` function

Usage

```
filter_mispicked_ions_params(
  ringwin = 0.5,
  isowin = 0.01,
  trwin = 0.005,
  max_iso_shift = 3,
  merge_peaks = TRUE,
  merge_method = "sum",
  copy_object = FALSE
)
```

Arguments

`ringwin`	Ringing mass window or detector saturation mass window. Default = 0.5 atomic mass units (AMU).
`isowin`	Isotopic mass window. Default = 0.01 AMU.
`trwin`	A `numeric` denoting the retention time threshold for assessing if ions should be merged. Default = 0.005.
`max_iso_shift`	A `numeric`. Default = 3.
`merge_peaks`	A `boolean` parameter to determine if peaks found to belong to the same ion should be merged in the feature table.
`merge_method`	If merge_peaks is TRUE, a method for how similar peaks should be merged. Can be one of "sum".
`copy_object`	A `boolean` parameter that allows users to return a copied object instead of modifying the object.

Value

a `list` object of arguments needed to call the given mpactr function when supplied to the `filter_peak_table()` wrapper function.

Examples

```
filter_mispicked_ions_params()
```

`filter_peak_table` *Filter Peak Table*

Description

This function is a wrapper for all of mpactr's filter functions. When called with a list of parameters that was generated from one of the following functions, it will call the subsequent filter: `filter_mispicked_ions_params()`, `filter_group_params()`, `filter_cv_params()`, and `filter_insource_ions_params()`. You can also find more information on these functions in`mpactr` documentation.

Usage

```
filter_peak_table(mpactr_object, params)

## S3 method for class 'filter_mispicked_ions'
filter_peak_table(mpactr_object, params)

## S3 method for class 'filter_group'
filter_peak_table(mpactr_object, params)

## S3 method for class 'filter_cv'
filter_peak_table(mpactr_object, params)

## S3 method for class 'filter_insource_ions'
filter_peak_table(mpactr_object, params)
```

Arguments

`mpactr_object`	the mpactr_object is an object generated from the `import_all_data()` function. This is how we begin our pipeline.
`params`	the list of arguments generated from calling one of these functions: `filter_mispicked_ions_params()`, `filter_group_params()`, `filter_cv_params()`, and `filter_insource_ions_params()`.

Value

a `mpactr` object that has been filter based on the supplied parameters.

Examples

```
data.table::setDTthreads(threads = 1)
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")
filtered_data <- data |>
```

```
filter_peak_table(filter_mispicked_ions_params())
```

`generate_a_combined_table` *Create a combined table*

Description

This function will use the generated matched data, annotations and cluster data, to create a combined dataframe of all the generated data. It has the ability to create the dataframe without annotations or clustering data. However, if annotations are supplied and a feature has more than one annotation, the data will be returned in long format.

Usage

```
generate_a_combined_table(
  matched_data,
  annotations = NULL,
  cluster_data = NULL
)
```

Arguments

`matched_data`	massdata object created from `ms2_ms1_compare()`
`annotations`	annotations table created from `annotate_ms2()`
`cluster_data`	cluster data created from `cluster_data()`

Value

a `data.frame` object.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)
```

```
cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data, cutoff = 0.3, cluster_method = "opticlust")

 massbank <- read_msp(mums2_example("massbank_example_data.msp"))
 annotations <- annotate_ms2(mass_data = matched_data,
   reference = massbank, scoring_params = modified_cosine_params(0.5),
   ppm = 1000,
   min_score =  0.1, chemical_min_score = 0, number_of_threads = 2)

generate_a_combined_table(matched_data, annotations, cluster_results)
```

`get_community_matrix` *Get Community Matrix*

Description

Returns the community `matrix` or the data that you used to create the object.

Usage

```
get_community_matrix(community_object)
```

Arguments

`community_object`
: the object created from the `create_community_object()` function.

Value

returns `matrix`, based on the community object.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

dist <- dist_ms2(data = matched_data, cutoff = 0.3, precursor_thresh = 2,
 score_params = modified_cosine_params(0.5), min_peaks = 0,
 number_of_threads = 2)
```

```
cluster_results <- cluster_data(distance_df = dist,
 ms2_match_data = matched_data, cutoff = 0.3, cluster_method = "opticlust")

community_with_cluster <- create_community_matrix_object(cluster_results)

community_object_mass_data <- create_community_matrix_object(matched_data)
```

`get_molecular_formula_preds` *Get Molecular Formula Predictions*

Description

Returns all of the molecular formula predictions.

Usage

```
get_molecular_formula_preds(mass_data)
```

Arguments

`mass_data`	The object generated from `ms2_ms1_compare()`.

Value

a `character` vector contain all of your predicted molecular formulas.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 0.1, 1)

matched_data <- compute_molecular_formulas(matched_data,
                                           number_of_threads = 2)

get_molecular_formula_preds(matched_data)
```

get_ms1_data *Get MS1 Matches*

Description

A getter that returns the generated ms2 data in your mass_data object.

Usage

```
get_ms1_data(mass_data)
```

Arguments

mass_data	The object generated from ms2_ms1_compare().

Value

a data.frame object containing ms1 data.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

get_ms1_data(matched_data)
```

get_ms2_matches *Get MS2 Matches*

Description

A getter that returns the generated ms2 data in your mass_data object.

Usage

```
get_ms2_matches(mass_data)
```

Arguments

mass_data	The object generated from ms2_ms1_compare().

Value

a data.frame object containing ms2 data.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

get_ms2_matches(matched_data)
```

---

get_ms2_peaks_data *Get Peaks Data*

Description

A getter that will return the peak data of all of the matched specturms. The peak data is the list of mz/intensities found in the ms2 file.

Usage

```
get_ms2_peaks_data(mass_data)
```

Arguments

mass_data	The object generated from ms2_ms1_compare().

Value

a list object containing peak data.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

get_ms2_peaks_data(matched_data)
```

`get_reference_data` *Get Reference Data*

Description

Will return the data inside the reference object based on the index given.

Usage

```
get_reference_data(reference, index)
```

Arguments

`reference`	reference database object.
`index`	the index of the data. The index starts at 1.

Value

returns a `list` object with all of the reference data at the specified index.

Examples

```
reference <- read_msp(mums2_example("massbank_example_data.msp"))
get_reference_data(reference, 1)
```

get_samples *Get Samples*

Description

Returns a list of your samples found in the metadata file.

Usage

```
get_samples(mass_data)
```

Arguments

mass_data	The object generated from ms2_ms1_compare().

Value

a character vector contain all of your samples.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

get_samples(matched_data)
```

import_all_data *Import all data*

Description

This function is a wrapper for the mpactr import_data function. It will import your peak table and meta data and create a mpactr_object.

Usage

```
import_all_data(peak_table, metadata, format)
```

Arguments

peak_table	The file path to your feature table file.
metadata	The file path to your metadata file or data.frame.
format	The expected exported type of your peak table, can be one of "Progenesis", "Metaboscape", "None".

Value

a mpactr object.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")
```

---

length.reference_database

Reference database length

Description

returns the length of the database

Usage

```
## S3 method for class 'reference_database'
length(x)
```

Arguments

x	reference database object.

Value

returns the length of the regerence database

Examples

```
reference <- read_msp(mums2_example("massbank_example_data.msp"))
length(reference)
```

`modified_cosine_params` *GNPS-like similarity between two MS/MS spectra*

Description

`modified_cosine_params()` generates a parameter list to perform GNPS-like cosine similarity score calculation between two MS2 spectra.

Usage

```
modified_cosine_params(frag_tolerance)
```

Arguments

`frag_tolerance`	The mz fragment tolerance threshold for aligning fragment peaks from two ms2 spectra. GNPS default = 0.5.

Details

`modified_cosine_params()` will initiate cosine scoring based on the Python code by Wang et al. (2016), which is currently used for cosine scoring in GNPS, to calculate similarity between two MS2 spectra. This scoring method will compare peaks data, apply a square root normalization to peak intensities, align peaks both with and without correction for mass shifts, and calculate similarity.

Value

A parameters list for similarity scoring method "gnps"

References

Mingxun Wang, Jeremy J. Carver, Vanessa V. Phelan, Laura M. Sanchez, Neha Garg, Yao Peng, Don Duy Nguyen et al. "Sharing and community curation of mass spectrometry data with Global Natural Products Social Molecular Networking." Nature biotechnology 34, no. 8 (2016): 828. PMID: 27504778

Examples

```
modified_cosine_params(0.5)
```

ms2_ms1_compare *Match your ms1 spectra to a ms2*

Description

We are matching your ms1 to your supplied ms2 by looking at the difference between the mz and rt.

Usage

```
ms2_ms1_compare(ms2_files, mpactr_object, mz_tolerance, rt_tolerance)
```

Arguments

ms2_files	a list of all your mgf, mzml, or mzxml files.
mpactr_object	your mpactr object created from import_all_data()
mz_tolerance	your mass-charge ratio tolerance in ppm (parts per million).
rt_tolerance	your retention time tolerance.

Value

returns a mass_data object of all of the ms2 and ms1 matches.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)
```

mums2_example *Get file paths for examples*

Description

mums2 contains a number of example files in the inst/extdata directory. This function makes them accessible in documentation that shows how file paths are used in function examples.

Usage

```
mums2_example(file = NULL)
```

Arguments

file	Name of a file. If NULL, all examples files will be listed.

Value

A file path to example data stored in the inst/extdata directory of the package.

returns a character object

Examples

```
mums2_example()

mums2_example("massbank_example_data.msp")
```

---

print.community_object *Print Community Object*

Description

S3 function for printing the community object

Usage

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

Arguments

x	the object created from the create_community_object() function.
...	other parameters that are included in the print() function.

Value

returns matrix representation of the community object and prints it to the screen.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")



matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

community_object <- create_community_matrix_object(matched_data)
print(community_object)
```

---

print.reference_database

Print reference

Description

print reference objects.

Usage

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

Arguments

x	reference database object.
...	any extra print arguments you want to include.

Value

prints customized message to the console

Examples

```
reference <- read_msp(mums2_example("massbank_example_data.msp"))
print(reference)
```

rarefy_ms *Rarefy MS1 Feature Table*

Description

rarefy_ms() performs a single subsampling of MS1 features in sample. Feature intensities are subsampled to the supplied size and accounts for intensity thresholds due to machine limits and background noise. Specifically, features whose abundance falls below the threshold after rarefying are removed. This allows for accurate representation of samples at different dilutions regardless of the desired submsampling size.

Usage

```
rarefy_ms(community_object, size, threshold, number_of_threads = 1, seed = 123)
```

Arguments

community_object	A community_object
size	The desired total sample intensity to subsample to.
threshold	The individual feature threshold. Each subsampled feature must be >= this value to be retained.
number_of_threads	the number of threads you wish to use for this calculation.
seed	the RNG (random number generator) seed you would like to use.

Value

A external_pointer that references a community matrix of rarefied feature intensities.

returns a matrix object that contains your rarefied data.

Examples

```
data <-
   import_all_data(peak_table =
                   mums2::mums2_example("botryllus_pt_small.csv"),
                   metadata =
                   mums2::mums2_example("boryillus_metadata.csv"),
                   format = "None")


matched_data <- ms2_ms1_compare(mums2_example("botryllus_v2.gnps.mgf"),
 data, 1, 6)

community_object <- create_community_matrix_object(matched_data)
rarefy_ms(community_object, 400, 10)
```

read_hmdb *Read HMDB database*

Description

This function allows you to create an hmdb database. However you are required to supply an xml hmdb file and a folder path that contains all of the ms2 spectras from the hmdb download page https://www.hmdb.ca/downloads.

Usage

```
read_hmdb(hmdb_file, ms2_folder)
```

Arguments

hmdb_file	the xml hmdb file.
ms2_folder	the folder path of your ms2 spectra files.

Value

a reference_database object.

References

Wishart DS, Guo A, Oler E, Wang F, Anjum A, Peters H, Dizon R, Sayeeda Z, Tian S, Lee BL, Berjanskii M, Mah R, Yamamoto M, Jovel J, Torres-Calzada C, Hiebert-Giesbrecht M, Lui VW, Varshavi D, Varshavi D, Allen D, Arndt D, Khetarpal N, Sivakumaran A, Harford K, Sanford S, Yee K, Cao X, Budinski Z, Liigand J, Zhang L, Zheng J, Mandal R, Karu N, Dambrova M, Schiöth HB, Greiner R, Gautam V. HMDB 5.0: the Human Metabolome Database for 2022. Nucleic Acids Res. 2022 Jan 7;50(D1):D622-D631. doi: 10.1093/nar/gkab1062. PMID: 34986597; PMCID: PMC8728138.

Examples

```
read_msp(mums2_example("massbank_example_data.msp" ))
```

read_msp *Create Reference Database*

Description

Creates a reference database by reading a download msp file. These files can be downloaded from sites like https://systemsomicslab.github.io/compms/msdial/main.html#MSP or https://mona.fiehnlab.ucdavis.edu/downloads

Usage

```
read_msp(msp_file)
```

Arguments

msp_file	the file path of your msp file

Value

a reference_database object.

Examples

```
read_msp(mums2_example("massbank_example_data.msp"))
```

---

spec_entropy_params *Entropy similarity between two MS/MS spectra*

Description

Calculate spectral entropy similarity between two MS2 spectra

Usage

```
spec_entropy_params(
  ms2_tolerance_in_da = 0.02,
  ms2_tolerance_in_ppm = -1,
  clean_spectra = TRUE,
  min_mz = 0,
  max_mz = 1000,
  noise_threshold = 0.01,
  max_peak_num = 100,
  weighted = TRUE
)
```

Arguments

ms2_tolerance_in_da	MS2 peak tolerance in Da, set to -1 to disable. Defaults to 0.02.
ms2_tolerance_in_ppm	MS2 peak tolerance in ppm, set to -1 to disable. Defaults to -1.
clean_spectra	Either TRUE or FALSE to clean the spectra prior to calculating similarity. Defaults to TRUE.
min_mz	numeric, minimum mz to keep, set to -1 to disable. Defaults to 0.
max_mz	numeric, maximum mz to keep, set to -1 to disable. Defaults to 1000.

`noise_threshold`
: Background intensity threshold, all peaks with intensity < noise_threshold * max_intensity are removed. Set to -1 to disable. Defaults to `0.01`.

`max_peak_num`
: `numeric`, maximum number of peaks to keep for score calculation. Set to -1 to disable. Defaults to `100`.

`weighted`
: `logical` whether weighted or unweighted entropy similarity will be calculated. Defaults to `TRUE`.

Details

`spec_entropy_params()` will initiate spectral entropy similarity scoring via the `msentropy` package (Li et al. 2021). For more information about parameters see there GitHub.

Value

A parameters `list` for similarity scoring method "spectral_entropy"

References

Li, Y., Kind, T., Folz, J. et al. Spectral entropy outperforms MS/MS dot product similarity for small-molecule compound identification, Nat Methods 18, 1524–1531 (2021). https://doi.org/10.1038/s41592-021-01331-z

Examples

```
spec_entropy_params()
```

Index