Package ‘RAMClustR’

May 17, 2023

Type Package

Title Mass Spectrometry Metabolomics Feature Clustering and Interpretation

Version 1.3.0

Date 2023-05-17

Depends R (>= 3.5.0)

Imports dynamicTreeCut, fastcluster, ff, htrr, jsonlite, preprocessCore, e1071, gplots, pcaMethods, stringr, utils, webchem, ggplot2, methods

Suggests knitr, rmarkdown, xcms, testthat, patrick, MSnbase, InterpretMSSpectrum, BiocManager, xml2, stringi, readxl, curl, rentrez

License GPL (>= 2)

Description A feature clustering algorithm for non-targeted mass spectromet-ric metabolomics data. This method is compatible with gas and liquid chromatography coupled mass spectrometry, including indiscriminant tandem mass spectrometry data <DOI:10.1021/ac501530d>.

URL https://github.com/cbroeckl/RAMClustR

Encoding UTF-8

biocViews MassSpectrometry, Metabolomics

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Corey D. Broeckling [aut] (<https://orcid.org/0000-0002-6158-827X>), Fayyaz Afsar [aut], Steffen Neumann [aut], Asa Ben-Hur [aut], Jessica Prenni [aut], Helge Hecht [cre] (<https://orcid.org/0000-0001-6744-996X>), Matej Trojan [ctb], Zargham Ahmad [ctb]
Maintainer  Helge Hecht <helge.hecht@recetox.muni.cz>
Repository  CRAN
Date/Publication  2023-05-17 08:00:02 UTC

R topics documented:

adap.to.rc ........................................ 3
add_params ........................................ 5
annotate ........................................... 5
annotation.summary ................................ 7
assign.z ........................................... 8
change.annotation .................................. 9
checks .............................................. 11
check_arguments_filter.blanks ...................... 11
check_arguments_filter.cv .......................... 12
check_arguments_replace.na ......................... 12
cmpd.summary ....................................... 13
compute_do.sets .................................... 14
compute_SPECAbundAve ................................ 14
compute_wt_mean .................................... 15
create_ramclustObj ................................ 15
defineExperiment ................................... 16
define_samples ..................................... 17
do.findmain ......................................... 18
export.msfinder.formulas ............................ 20
exportDataset ....................................... 21
filter_blanks ....................................... 22
filter_good_features ................................ 23
filter_signal ........................................ 23
findfeature .......................................... 24
findmass ............................................ 25
find_good_features .................................. 26
fooddb2msfinder .................................... 26
get.taxon.cids ...................................... 27
getData ............................................ 28
getSmilesInchi ...................................... 29
get_ExpDes .......................................... 30
get_instrument_platform ............................... 31
import.adap.kdb .................................... 31
import.msfinder.formulas ............................ 32
import.msfinder.mssearch ........................... 33
import.msfinder.structures ......................... 34
import.sirius ....................................... 35
impRamSearch ........................................ 36
manual.annotation.template .......................... 37
mean_signal_intensities ................................ 38
mergeRCobjects ..................................... 38
Description

use pubchem rest and view APIs to retrieve structures, CIDs (if a name or inchikey is given), synonyms, and optionally vendor data, when available.
Usage

adap.to.rc(
    seq = "seq.csv",
    spec.abund = "signal.csv",
    msp = "spectra.msp",
    annotations = "annotations.xlsx",
    mzdec = 1,
    min.score = 700,
    manual.name = FALSE,
    qc.tag = "qc",
    blank.tag = "blank",
    factor.names = c()
)

Arguments

seq     file name/path to sequence file - expect filenames in column 1 and sample names in column 2. filenames should match those in spec.abund
spec.abund file name/path to adap-big export of signal intensities. .csv file expected
msp        file name/path to .msp file created by adap-big
annotations file name/path to annotations .xlsx file. generally 'simple_export.xlsx'
mzdec      mz decimals to report for internal storage/reporting. generally we want 0 for adap kdb
min.score   700 (out of 1000) by default
manual.name when looking up inchikey/names, should manual input be used to fill ambiguous names? generally recommend TRUE
qc.tag       a character string by which to recognize a sample as a qc sample. i.e. 'QC' or 'qc'.
blank.tag    a character string by which to recognize a sample as a blank sample. i.e. 'blank' or 'Blank'.
factor.names factor names

Details

useful for moving from chemical name to digital structure representation. greek letters are assumed to be 'UTF-8' encoded, and are converted to latin text before searching. if you are reading in your compound name list, do so with 'encoding' set to 'UTF-8'.

Value

returns a ramclustR structured object suitable for downstream processing steps.

Author(s)

Corey Broeckling
**add_params**

**Description**

add rc.feature.replace.na params in ramclustObj

**Usage**

```r
add_params(ramclustObj, params, param_name)
```

**Arguments**

- `ramclustObj`: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `params`: vector containing parameters to add
- `param_name`: name of the parameter/step

**Value**

ramclustR object with rc.feature.replace.na params added.

**annotate**

**Description**

evaluate ramSearch, MSFinder mssearch, MSFinder Structure, MSFinder Formula, and findmain output to annotate spectra of ramclustR object

**Usage**

```r
annotate(
  ramclustObj = NULL,
  standardize.names = FALSE,
  min.msms.score = 0.8,
  database.priority = NULL,
  database.priority.factor = 0.1,
  find.inchikey = TRUE,
  taxonomy.inchi = NULL,
  taxonomy.inchi.factor = 0.1,
  use.ri = TRUE,
  sri = 300,
  ri.na.factor = 0.6,
  reset = TRUE
)
```
Arguments

ramclustObj  R object - the ramclustR object which was used to write the .mat or .msp files
standardize.names  logical: if TRUE, use inchikey for standardized chemical name lookup (http://cts.fiehnlab.ucdavis.edu/)
min.msms.score  numerical: what is the minimum MSFinder similarity score acceptable. default = 6.5
database.priority  character. Formula assignment prioritization based on presence in one or more (structure) databases. Can be set to a single or multiple database names. must match database names as they are listed in MSFinder precisely. Can also be set to 'all' (note that MSFinder reports all databases matched, not just databases in MSFinder parameters). If any database is set, the best formula match to any of those databases is selected, rather than the best formula match overall. If NULL, this will be set to include all selected databases (from ramclustObj$msfinder.dbs, retrieved from search output during import.msfinder.formulas(), when available) or 'all'.
database.priority.factor  numeric, between 0 and 1. 0.1 by default. The proportion by which scores for structures not in priority database are assessed
find.inchikey  logical. default = TRUE. use chemical translation service to try to look up inchikey for chemical name.
taxonomy.inchi  vector or data frame. Only when rescore.structure = TRUE. user can supply a vector of inchikeys. If used, structures which match first block of inchikey retain full score, while all other structures are penalized.
taxonomy.inchi.factor  numeric, between 0 and 1. 0.1 by default. The proportion by which scores for structures not in taxonomy.inchi vector are assessed
use.ri  logical. default = TRUE. If retention index is available in ramclustObj (set by 'rc.calibrate.ri') and in library spectra from MSFinder, use RI similiarity to rescore.
sri  numeric. sigma value for retention index. controls decay rate of retention index curve. decay rate between 0 and 1 exported, and multiplied by spectrum score, totalscore.
ri.na.factor  numeric. between 0 and 1. 0.5 by default. how should spectrum scores be treated when no retention index is available? NA values are replaced by retention index similarities of ri.na.factor when use.ri = TRUE.
reset  logical. If TRUE, removes any previously assigned annotations.

Details

this function imports the output from the MSFinder program to annotate the ramclustR object

Value

an updated ramclustR object, with the at $msfinder.formula, $msfinder.formula.score, $ann, and $ann.conf slots updated to annotated based on output from 1. ramsearch output, 2. msfinder
mssearch, 3. msfinder predicted structure, 4. msfinder predicted formula, and 5. interpretMSSpectrum inferred molecular weight, with listed order as priority.

**Author(s)**

Corey Broeckling

**References**


---

**annotation.summary**

**annotation.summary()**

**Description**

Write a .csv file containing a summary of the annotations in the ramclustR object.

**Usage**

```r
annotation.summary(ramclustObj = NULL, outfile = NULL)
```

**Arguments**

- `ramclustObj` R object - the ramclustR object which was used to write the .mat or .msp files
- `outfile` file path/name of output csv summary file. if NULL (default) will be exported to spectra/annotationSummary.csv

**Details**

this function exports a csv file summarizing annotation evidence for each compound

**Value**

nothing
Author(s)
Corey Broeckling

References

Description
infer charge state of features in ramclustR object.

Usage
assign.z(
  ramclustObj = NULL,  
  chargestate = c(1:5),  
  mzError = 0.02,  
  nEvents = 2,  
  minPercentSignal = 10,  
  assume1 = TRUE  
)

Arguments
ramclustObj ramclustR object to annotate
chargestate integer vector. vector of integers of charge states to look for. default = c(1:5)
mzError numeric. the error allowed in charge state m/z filtering. absolute mass units
nEvents integer. the number of isotopes necessary to assign a charge state > 1. default = 2.
minPercentSignal numeric. the ratio of isotope signal (all isotopes) divided by total spectrum signal * 100 much be greater than minPercentSignal to evaluate charge state. Value should be between 0 and 100.
assume1 logical. when TRUE, m/z values for which no isotopes are found are assumed to be at z = 1.

Details
Annotation of ramclustR spectra. looks at isotope spacing for clustered features to infer charge state for each feature and a max charge state for each compound
Value
returns a ramclustR object. new slots holding:

zmax. vector with length equal to number of compounds. max charge state detected for that compound
fm. vector of inferred 'm', m/z value * z value
fz. vector of inferred 'z' values based on analysis of isotopes in spectrum.

Author(s)
Corey Broeckling

References


declare annotation
change.annate

evaluate ramSearch, MSFinder mssearch, MSFinder Structure, MSFinder Formula, and findmain output to annotate spectra of ramclustR object

Description
After running RAMSearch (msp) and MSFinder on .mat or .msp files, import the spectral search results

Usage
change.annate(
   ramclustObj = NULL,
   msfinder.dir = "C:/MSFinder/MSFINDER ver 3.22",
   standardize.names = FALSE,
   min.msms.score = 3.5,
   database.priority = "all",
   any.database.priority = TRUE,
   reset = TRUE
)
Arguments

ramclustObj  R object - the ramclustR object which was used to write the .mat or .msp files
msfinder.dir full path to MSFinder directory - used for naming refinement
standardize.names logical: if TRUE, use inchikey for standardized chemical name lookup (http://cts.fiehnlab.ucdavis.edu/)
min.msms.score numerical: what is the minimum MSFinder similarity score acceptable. default = 3.5
database.priority character. Formula assignment prioritization based on presence in one or more databases. Can be set to a single or multiple database names. must match database names as they are listed in MSFinder precisely. Can also be set to ‘all’ (note that MSFinder reports all databases matched, not just selected databases). If any database is set, the best formula match to that (those) database(s) is selected, rather than the best formula match overall.
any.database.priority logical. First priority in formula assignment is based on any of the ‘database.priority’ values. Secondary priority from all other databases (determined in original MSFinder search) if TRUE. If false, formula assignment score from MSFinder used independent of structure search results.
reset logical. If TRUE, removes any previously assigned annotations.

Details

this function imports the output from the MSFinder program to annotate the ramclustR object

Value

an updated ramclustR object, with the at $msfinder.formula, $msfinder.formula.score, $ann, and $ann.conf slots updated to annotated based on output from 1. ramsearch output, 2. msfinder mssearch, 3. msfinder predicted structure, 4. msfinder predicted formula, and 5. interpretMSSpectrum inferred molecular weight, with listed order as priority.

Author(s)

Corey Broeckling

References


---

### checks

**Description**

check if MS data contains mz and rt, and if MSMS data is present feature names and sample names are identical

**Usage**

```r
checks(
  ms1_featureDefinitions = NULL,
  ms1_featureValues = NULL,
  ms2_featureValues = NULL,
  feature_names = NULL
)
```

**Arguments**

- `ms1_featureDefinitions` dataframe with metadata with columns: mz, rt, feature names containing MS data
- `ms1_featureValues` dataframe with rownames = sample names, colnames = feature names containing MS data
- `ms2_featureValues` dataframe with rownames = sample names, colnames = feature names containing MSMS data
- `feature_names` feature names extracted from the data

---

### check_arguments_filter.blanks

**Description**

check provided arguments
Usage

check_arguments_filter.blanks(ramclustObj, sn)

Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

sn  numeric defines the ratio for 'signal'. i.e. sn = 3 indicates that signal intensity must be 3 fold higher in sample than in blanks, on average, to be retained.

check_arguments_filter.cv

check_arguments_filter.cv

Description

check provided arguments

Usage

check_arguments_filter.cv(ramclustObj, qc.tag)

Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

qc.tag  character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.

check_arguments_replace.na

check_arguments_replace.na

Description

check provided arguments

Usage

check_arguments_replace.na(ramclustObj, replace.int, replace.noise, replace.zero)
**cmpd.summary**

**Arguments**

- `ramclustObj` ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `replace.int` default = 0.1. proportion of minimum feature value to replace NA (or zero) values with
- `replace.noise` default = 0.1. proportion of `replace.int` value by which noise is added via 'jitter'
- `replace.zero` logical if TRUE, any zero values are replaced with noise as if they were NA values

**Description**

a bit of reporting for compounds, quick access summary and plot (if available)

**Usage**

```r
cmpd.summary(ramclustObj = NULL, cmpd = 1)
```

**Arguments**

- `ramclustObj` ramclustR object to annotate
- `cmpd` integer. compound number to report. i.e. 459.

**Details**

Reports name, annotation, retention time, number of features in spectrum, median and mean signal intensity, and if `interpretMSSpectrum` (do.findmain) has been run, plots an annotated MS level spectrum.

**Author(s)**

Corey Broeckling

**References**


compute_do.sets

Description
compute data frame to use in ramclustObj

Usage
compute_do.sets(ramclustObj)

Arguments
ramclustObj ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

Value
vector which is used to select data frame to use in ramclustObj

compute_SpecAbundAve

Description
further aggregate by sample names for 'SpecAbundAve' dataset

Usage
compute_SpecAbundAve(ramclustObj = NULL)

Arguments
ramclustObj ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

Value
ramclustR object with aggregate by sample names for 'SpecAbundAve' dataset
compute_wt_mean

Description
compute weighted.mean intensity of feature in ms/msms level data

Usage
compute_wt_mean(data, global.min, fmz, ensure.no.na)

Arguments
- data: feature in ms/msms level data
- global.min: minimum intensity in ms/msms level data
- fmz: feature retention time
- ensure.no.na: logical: if TRUE, any 'NA' values in msint and/or msmsint are replaced with numerical values based on 10 percent of feature min plus noise. Used to ensure that spectra are not written with NA values.

Value
weighted.mean intensity of feature in ms/msms level data

create_ramclustObj

Description
create ramclustr Object

Usage
create_ramclustObj(
    ExpDes = NULL,
    input_history = NULL,
    MSdata = NULL,
    MSMSdata = NULL,
    frt = NULL,
    fmz = NULL,
    st = NULL,
    phenoData = NULL,
    feature_names = NULL,
    sample_names = NULL,
    xcmsOrd = NULL,
    ensure.no.na = TRUE
)
**defineExperiment**

**Arguments**

- **ExpDes**: either an R object created by R ExpDes object: data used for record keeping and labelling msp spectral output.
- **input_history**: input history.
- **MSdata**: dataframe containing MS Data.
- **MSMSdata**: dataframe containing MSMS Data.
- **frt**: feature retention time, in whatever units were fed in.
- **fmz**: feature retention time.
- **st**: numeric: sigma t - time similarity decay value.
- **phenoData**: dataframe containing phenoData.
- **feature_names**: feature names extracted from the data.
- **sample_names**: sample names extracted from the data.
- **xcmsOrd**: original xcms order of features, for back-referencing when necessary.
- **ensure.no.na**: logical: if TRUE, any 'NA' values in msint and/or msmsint are replaced with numerical values based on 10 percent of feature min plus noise. Used to ensure that spectra are not written with NA values.

**Value**

an ramclustR object. This object is formatted as an hclust object with additional slots for holding feature and compound data.

**Description**

Create an Experimental Design R object for record-keeping and msp output.

**Usage**

```r
defineExperiment(csv = FALSE, force.skip = FALSE)
```

**Arguments**

- **csv**: logical or filepath. If csv = TRUE, csv template called "ExpDes.csv" will be written to your working directory. you will fill this in manually, ensuring that when you save you retain csv format. ramclustR will then read this file in and format appropriately. If csv = FALSE, a pop up window will appear (in windows, at least) asking for input. If a character string with full path (and file name) to a csv file is given, this will allow you to read in a previously edited csv file.
- **force.skip**: logical. If TRUE, ramclustR creates a pseudo-filled ExpDes object to enable testing of functionality. Not recommended for real data, as your exported spectra will be improperly labelled.
**define_samples**

**Value**

an Exp Des R object which will be used for record keeping and writing spectra data.

**Author(s)**

Corey Broeckling

**References**


---

**define_samples**

**Description**

define samples in each set

**Usage**

```r
define_samples(ramclustObj, tag)
```

**Arguments**

- **ramclustObj**: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- **tag**: character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.

**Value**

samples found using the tag
do.findmain

Description

Cluster annotation function: inference of 'M' - molecular weight of the compound giving rise to each spectrum - using the InterpretMSSpectrum::findMain function

Usage

do.findmain(
  ramclustObj = NULL,
  cmpd = NULL,
  mode = "positive",
  mzabs.error = 0.005,
  ppm.error = 10,
  ads = NULL,
  nls = NULL,
  scoring = "auto",
  plot.findmain = TRUE,
  writeMat = TRUE,
  writeMS = TRUE,
  use.z = TRUE
)

Arguments

ramclustObj ramclustR object to annotate.

cmpd integer: vector defining compound numbers to annotated. If NULL (default), all compounds

mode character: "positive" or "negative"

mzabs.error numeric: absolute mass deviation allowed, default = 0.01

ppm.error numeric: ppm mass error _added to mzabs.error, default = 10

ads character: vector of allowed adducts, i.e. c("[M+H]+"). If NULL, default positive mode values of H+, Na+, K+, and NH4+, as monomer, dimer, and trimer, are assigned. Negative mode include "[M-H]", "[M+Na-2H]", "[M+K-2H]", "[M+CH2O2-H]" as monomer, dimer, and trimer.

nls character: vector of allowed neutral losses, i.e. c("[M+H-H2O]+"). If NULL, an extensive list derived from CAMERA's will be used.

scoring character: one of 'imss', 'ramclustr', or 'auto'. default = 'auto'. see details.

plot.findmain logical: should polts be generated for evaluation? default = TRUE. PDF saved to working.directory/spectra

writeMat logical: should individual .mat files (for MSFinder) be generated in a 'mat' subdirectory in the 'spectra' folder? default = TRUE.
writeMS

logical: should individual .ms files (for Sirius) be generated in a 'ms' subdirectory in the 'spectra' folder? default = TRUE. Note that no import functions are yet written for Sirius output.

use.z

logical: if you have previously run the 'assign.z' function from ramclustR, there will be a slot reflecting the feature mass after accounting for charge (fm) - if TRUE this is used instead of feature m/z (fmz) in interpreting MS data and exporting spectra for annotation.

Details

a partially annotated ramclustR object. base structure is that of a standard R hierarchical clustering output, with additional slots described in ramclustR documentation (?ramclustR). New slots added after using the interpretMSSpectrum functionality include those described below.

Value

$M$: The inferred molecular weight of the compound giving rise to the each spectrum

$M$.ppm: The ppm error of all the MS signals annotated, high error values should be considered 'red flags'.

$M$.ann: The annotated spectrum supporting the interpretation of M

$use.findmain$: Logical vector indicating whether findmain scoring (TRUE) or ramclustR scoring (FALSE) was used to support inference of M. By default, findmain scoring is used. When ramclustR scoring differs from findmain scoring, the scoring metric which predicts higher M is selected.

$M$.ramclustr: M selected using ramclustR scoring

$M$.ppm.ramclustr: ppm error of M selected using ramclustR scoring. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

$M$.ann.ramclustr: annotated spectrum supporting M using ramclustR scoring

$M$.nann.ramclustr: number of masses annotated using ramclustR scoring. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

$M$.space.ramclustr: the 'space' of scores between the best and second best ramclustR scores. Calculated as a ratio. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

$M$.findmain: M selected using findmain scoring

$M$.ppm.findmain: ppm error of M selected using findmain scoring. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

$M$.ann.findmain: annotated spectrum supporting M using findmain scoring

$M$.nann.findmain: number of masses annotated using findmain scoring. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

$M$.space.findmain: the 'space' of scores between the best and second best findmain scores. Calculated as a ratio. Used to resolve conflicts between ramclustR and findmain M assignment when scoring = auto.

Author(s)

Corey Broeckling
References


export.msfinder.formulas

export MSFinder formula prediction results in tabular format.

Description

After running MSFinder, results have been imported to the ramclustR object. This function exports as a .csv file for ease of viewing.

Usage

export.msfinder.formulas(
  ramclustObj = NULL,
  export.all = FALSE,
  output.directory = NULL
)

Arguments

ramclustObj R object - the ramclustR object which was used to write the .mat or .msp files
export.all logical: default = FALSE. If TRUE, export all columns, if FALSE, only columns
output.directory valid path: default = NULL. If NULL, results are exported to spectra/mat directory.

Details

detail function exports a .csv file containing all returned MSFinder molecular formula hypotheses. this file is saved (by default) to the working directory spectra/mat/ directory
**Value**

an updated ramclustR object, with the RCsann and RCsann.conf slots updated to annotated based on output from 1. ramsearch output, 2. msfinder mssearch, 3. msfinder predicted structure, 4. msfinder predicted formula, and 5. interpretMSSpectrum inferred molecular weight, with listed order as priority.

**Author(s)**

Corey Broeckling

**References**


---

**Description**

export one of 'SpecAbund', 'SpecAbundAve', 'MSdata' or 'MSMSdata' from an RC object to csv

**Usage**

```r
exportDataset(
  ramclustObj = NULL,
  which.data = "SpecAbund",
  label.by = "ann",
  appendFactors = TRUE
)
```

**Arguments**

- `ramclustObj`: ramclustR object to export from
- `which.data`: name of dataset to export. SpecAbund, SpecAbundAve, MSdata, or MSMSdata
- `label.by`: either 'ann' or 'cmpd', generally. name of ramclustObj slot used as csv header for each column (compound)
- `appendFactors`: logical. If TRUE (default) the factor data frame is appended to the left side of the dataset.
filter_blanks

Details

Useful for exporting the processed signal intensity matrix to csv for analysis elsewhere.

Value

nothing is returned. file exported as csf to 'datasets/*.csv'

Author(s)

Corey Broeckling

References


filter_blanks  filter_blanks

Description

filter blanks

Usage

filter_blanks(ramclustObj, keep, d1)

Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
keep  union of which signal is at least 3x larger, output of filter_signal()
d1  MS Data

Value

ramclustObj object with feature.filter.blanks
filter_good_features

Description
filter to keep only 'good' features

Usage
filter_good_features(ramclustObj, keep)

Arguments
- ramclustObj: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- keep: features to keep. output of find_good_features().

Value
ramclustR object filtered to keep only 'good' features

filter_signal

Description
filter signal

Usage
filter_signal(ms.qc.mean, ms.blank.mean, sn)

Arguments
- ms.qc.mean: ms qc mean signal intensities
- ms.blank.mean: ms blank mean signal intensities
- sn: numeric defines the ratio for 'signal'. i.e. sn = 3 indicates that signal intensity must be 3 fold higher in sample than in blanks, on average, to be retained.

Value
union of which signal is at least 3x larger
Description

see if any features match a given mass, and whether they are plausibly M0

Usage

```r
findfeature(
    ramclustObj = NULL,
    mz = NULL,
    mztol = 0.02,
    rt = NULL,
    rttol = 2,
    iso.rttol = 2,
    zmax = 6,
    m.check = TRUE
)
```

Arguments

- `ramclustObj`: R object: the ramclustR object to explore
- `mz`: numeric: mz value to search for
- `mztol`: numeric: absolute mass tolerance around mz
- `rt`: numeric: optional rt value to search for (generally in seconds, though use whatever units your data is in)
- `rttol`: numeric: absolute retention time tolerance around rt.
- `iso.rttol`: numeric: when examining isotope patterns, feature retention time tolerance around features matching mz +/- mztol
- `zmax`: integer: maximum charge state to consider. default is 6.
- `m.check`: logical: check whether the matching masses are plausibly M0. That is, we look for ions 1 proton mass (from charge state 1:zmax) below the target m/z at the same time that have intensities consistent with target ion being a non-M0 isotope.

Details

a convenience function to perform a targeted search of all features for a mass of interest. Also performs a crude plausibility check as to whether the matched feature could be M0, based on the assumption of approximately 1 carbon per 17 m/z units and natural isotopic abundance of 1.1

Value

returns a table to the console listing masses which match, their retention time and intensity, and whether it appears to be plausible as M0
findmass

Author(s)
Corey Broeckling

Description
see if any features match a given mass, and whether they are plausibly M0

Usage
findmass(
  ramclustObj = NULL,
  mz = NULL,
  mztol = 0.02,
  rttol = 2,
  zmax = 6,
  m.check = TRUE
)

Arguments
ramclustObj R object: the ramclustR object to explore
mz numeric: mz value to search for
mztol numeric: absolute mass tolerance around mz
rttol numeric: when examining isotope patterns, feature retention time tolerance around features matching mz +- mztol
zmax integer: maximum charge state to consider. default is 6.
m.check logical: check whether the matching masses are plausibly M0. That is, we look for ions 1 proton mass (from charge state 1:zmax) below the target m/z at the same time that have intensities consistent with target ion being a non-M0 isotope.

Details
a convenience function to perform a targeted search of all features for a mass of interest. Also performs a crude plausibility check as to whether the matched feature could be M0, based on the assumption of approximately 1 carbon per 17 m/z units and natural isotopic abundance of 1.1

Value
returns a table to the console listing masses which match, their retention time and intensity, and whether it appears to be plausible as M0
find_good_features

Description

find 'good' features, acceptable CV at either MS or MSMS level results in keeping

Usage

find_good_features(ramclustObj, do.sets, max.cv, qc)

Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
do.sets  select data frame to use.
max.cv  numeric maximum allowable cv for any feature. default = 0.5
qc  QC samples found by define_samples

Value

ramclustR object
features to keep

fooddb2msfinder

Description

convenience function for converting FoodDB database export format to MSFinder custom database import format. Before running this, please have downloaded .csv files from FoodDB with the appropriate Display Field Headers (see details)

Usage

fooddb2msfinder(
  foodb.files = NULL,
  out.dir = NULL,
  out.name = "FoodDB_for_MSFinder.txt"
)
**get.taxon.cids**

**Arguments**

- `foodb.files`: default = NULL, if path is set, will read automatically. If NULL, directory selection by user.
- `out.dir`: default = NULL. Can set to existing directory with full path name. If NULL, directory selection by user.
- `out.name`: default = "FoodDB_for_MSFinder.txt".

**Details**

Input file(s) should be csv formatted, with required headers of 'Name', 'Smiles', 'Inchikey', 'Chemical formula', and 'Mono mass' - case sensitive. Output will be in tab delimited text format in directory of choice.

**Value**

Nothing is returned - output file written to directory set by 'out.dir' and name set by 'out.name'

**Author(s)**

Corey Broeckling

**References**


---

gerget.taxon.cids

**Description**

use pubchem rest to retrieve pubchem CIDS known to be found in a given species. NCBI taxid should be used as input. i.e. Homo sapiens subsp. 'Denisova' is taxid 741158

**Usage**

```r
get.taxon.cids(
  taxid = NULL,
  taxstring = NULL,
  sub taxa n = 1000,
  get.inchikey = TRUE
)
```
getData

Arguments

taxid integer NCBI taxid for the taxon to search.
taxstring taxonomy string for the taxon of interest.
sub.taxa.n integer value for the number of subtaxa to consider. Note that if the sub.taxa.n value is less than the available number of subtaxa, only the first sub.taxa.n values, as reported by rentrez, are returned. If you require specific subtaxa, you should call those taxids explicitly to ensure those results are returned.
get.inchikey logical whether to get the InChIKeys as well (default TRUE).

Details

this function enables return of a list of pubchem CIDs which can be used for prioritizing annotations. If a genus level taxid is selected, setting the sub.taxa.n option > 0 will return metabolites associated with that taxid and all (assuming n is large enough) subtaxa. i.e. setting taxid to 9605 (genus = 'Homo') will return metabolites associated with Homo sapiens, Homo heidelbergensis, Homo sapiens subsp. 'Denisova', etc.

Value

returns a vector of integer pubchem cids (and optionally inchikeys if get.inchikey was set to TRUE)

Author(s)

Corey Broeckling

Description

retrieve and parse sample names, retrieve metabolite data. returns as list of two data frames

Usage

getData(
  ramlustObj = NULL,
  which.data = "SpecAbund",
  delim = "-",
  cmpdlabel = "cmpd",
  filter = FALSE
)
getSmilesInchi

Arguments

ramclustObj ramclustR object to retrieve data from
which.data character; which dataset (SpecAbund or SpecAbundAve) to reference
delim character; "-" by default - the delimiter for parsing sample names to factors
cmpdlabel = "cmpd"; label the data with the annotation. can also be set to 'ann' for column names assigned as annotations.
filter = TRUE; logical, if TRUE, checks for $cmpd.use slot generated by rc.cmpd.cv.filter() function, and only gets acceptable compounds.

Details

convenience function for parsing sample names and returning a dataset.

Value

returns a list of length 3: $design is the experimental sample factors after parsing by the delim, $data is the dataset, $full.data is merged $des and $data data.frames.

Author(s)

Corey Broeckling

description

use PubChem API to look up full smiles and inchi notation for each inchikey

Usage

getSmilesInchi(ramclustObj = NULL, inchikey = NULL, ignore.stereo = TRUE)

Arguments

ramclustObj ramclustR object to look up smiles and inchi for each inchikey (without a smiles/inchi). Must provide one of ramclustObj or inchikey.
inchikey character vector of inchikey strings. Must provide one of ramclustObj or inchikey.
ignore.stereo logical. default = TRUE. If the Pubchem databases does not have the full inchikey string, should we search by the first (non-stereo) block of the inchikey? When true, returns the first pubchem match to the inchikey block one string. If the full inchikey is present, that is used preferentially.
Details

The $inchikey slot is used to look up parameters from pubchem. PubChem CID, a pubchem URL, smiles (canonical) and inchi are returned. If smiles and inchi slots are already present (from MSFinder, for example) pubchem smiles and inchi are used to fill in missing values only, not replace.

Value

returns a ramclustR object. New vector of $smiles and $inchi with length equal to number of compounds.

Author(s)

Corey Broeckling

References

Description

get instrument platform

Usage

get_instrument_platform(design)

Arguments

design data frame containing Experimental Design

Value

instrument platform

Description

use pubchem rest and view APIs to retrieve structures, CID\(\text{s}\) (if a name or inchikey is given), synonyms, and optionally vendor data, when available.

Usage

import.adap.kdb(
  ramclustObj = NULL,
  annotations = NULL,
  min.score = 700,
  annotate = TRUE,
  manual.name = TRUE
)

Arguments

ramclustObj ramclustR object to be annotated.
annotations file name/path to annotations .xlsx file. generally ’simple_export.xlsx’
min.score 700 (out of 1000) by default
annotate logical. TRUE by default. for now please leave default
manual.name when looking up inchikey/names, should manual input be used to fill ambiguous names? generally recommend TRUE
**Details**

useful for moving from chemical name to digital structure representation. greek letters are assumed to be 'UTF-8' encoded, and are converted to latin text before searching. if you are reading in your compound name list, do so with ’encoding’ set to ’UTF-8’.

**Value**

returns a ramclustR structured object suitable for down stream processing steps.

**Author(s)**

Corey Broeckling
import.msfinder.mssearch

import.MSFinder.mssearch

Description

After running MSFinder on .mat or .msp files, import the spectral search results

Usage

import.msfinder.mssearch(
  ramclustObj = NULL,
  mat.dir = NULL,
  msp.dir = NULL,
  dir.extension = ".mssearch"
)

Arguments

ramclustObj  R object - the ramclustR object which was used to write the .mat or .msp files
mat.dir      optional path to .mat directory
msp.dir      optional path to .msp directory
dir.extension optional directory name code specifying subset of results to use. Useful if running MSFinder from the command line for both spectral searching and interpretation.

Details

this function imports the output from the MSFinder program to annotate the ramclustR object

Value

an updated ramclustR object, with new slots at $msfinder.mssearch.details and $msfinder.mssearch.scores

References


import.msfinder.structures

write.methods

Description

write RAMClustR processing methods and citations to text file

Usage

import.msfinder.structures(ramclustObj = NULL, mat.dir = NULL, msp.dir = NULL)

Arguments

ramclustObj R object - the ramclustR object which was used to write the .mat or .msp files
mat.dir directory in which to look for mat file MSFinder output - by default the /spectra/mat in the working directory
msp.dir directory in which to look for msp file MSFinder output - by default the /spectra/msp in the working directory

Details

this function exports a file called ramclustr_methods.txt which contains the processing history, parameters used, and relevant citations.

Value

an annotated ramclustR object
nothing - new file written to working directory
Author(s)
Corey Broeckling

References

Description
After running Sirius on .ms files, import the annotation results

Usage
import.sirius(ramclustObj = NULL, ms.dir = NULL, ion.mode = NULL)

Arguments
ramclustObj R object - the ramclustR object which was used to write the .mat or .msp files
ms.dir optional path to .mat directory. default = "spectra/ms/out" subdirectory in working directory
ion.mode specify either "N" for negative ionization mode or "P" for positive ionization mode

Details
this function imports the output from the Sirius program to annotate the ramclustR object

Value
an updated ramclustR object, with new slots at $msfinder.sirius

Author(s)
Corey Broeckling
References


Description

import ramsearch output for annotating an RC object

Usage

impRamSearch(ramclustObj = NULL, ramsearchout = "spectra/results.rse")

Arguments

ramclustObj ramclustR object to annotate
ramsearchout path to .rse file to import

Details

Annotation of ramclustR exported .msp spectra is accomplished using RAMSearch. Exported ramsearch annotations (.rse) can be imported with this function

Value

returns a ramclustR object. new slots holding .rse data

Author(s)

Corey Broeckling

References


Description

export a .csv formatted template for manually editing MSFinder annotations

Usage

manual.annotation.template(
  ramclustObj = NULL,
  outfile = "manual.annotation.template.csv"
)

Arguments

ramclustObj ramclustR object to annotate
outfile output file directory and name. default = 'manual.annotation.template.csv'

Details

While unsupervised annotation is rapid and objective, subjective knowledge can be used to improve annotations. This function writes a template file containing compound name, computationally assigned inchikey, and an empty column for your manually inferred inchikey. Upon completion of manual annotation, you can reimport this file and update your ramclustR object to reflect your manual input.

Author(s)

Corey Broeckling

References


mean_signal_intensities

Description

calculate MS mean signal intensities

Usage

mean_signal_intensities(data, sample)

Arguments

data MS/MSMS data
sample sample found using the tag, output of define_samples()

Value

mean signal intensities

mergeRCobjects

Description

merge two ramclustR objects

Usage

mergeRCobjects(
  ramclustObj.1 = NULL,
  ramclustObj.2 = NULL,
  mztol = 0.02,
  rttol = 30,
  course.rt.adj = NULL,
  mzwt = 2,
  rtwt = 1,
  intwt = 3
)

Arguments

**ramclustObj.1**  
ramclustR object 1: this object will be the base for the new object. That is all the features from ramclustObj.1 will be retained.

**ramclustObj.2**  
ramclustR object 2: this object will mapped and appended to ramclustObj.1. That is only features which appear consistent with those from ramclustObj.1 will be retained.

**mztol**  
umERIC: absolute mass tolerance around mz

**rttol**  
umERIC: feature retention time tolerance. Value set by this option will be used during the initial anchor mapping phase. Two times the standard error of the loess correction will be used for the full mapping.

**course.rt.adj**  
umERIC: default = NULL. optional approximate retention time shift between ramclustObj.1 and ramclustObj.2. i.e if the retention time of ramclustObj.1 is on average 15 seconds longer than that of ramclustobj.2, enter '15'. if 1 is less than 2, enter a negative number. This is applied before mapping to enable a smaller 'rttol' value to be used.

**mzwt**  
umERIC: when mapping features, weighting value used for similarities between feature mass values (see rtwt, intwt)

**rtwt**  
umERIC: when mapping features, weighting value used for similarities between feature retention time values (see mzwt, intwt)

**intwt**  
umERIC: when mapping features, weighting value used for similarities between ranked signal intensity values (see rtwt, mzwt)

Details

Two ramclustR objects are merged with this function, mapping features between them. The first (ramclustObj.1) object use used as the template - all data in it is retained. ramclustObj.2 is mapped to ramclustObj.1 feature by feature - only mapped features are retained. A new ramlcustObj is returned, with a new SpecAbund dataset with the same column number as the ramclustObj.1$SpecAbund set.

Value

returns a ramclustR object. All values from ramclustObj.1 are retained. SpecAbund dataset from ramclustObj.1 is moved to RC$SpecAbund.1, where RC is the new ramclustObj.

Author(s)

Corey Broeckling
Description

normalize data using batch.qc

Usage

normalized_data_batch_qc(
  data = NULL,
  batch = NULL,
  order = NULL,
  qc = NULL,
  qc.inj.range = 20
)

Arguments

data feature in ms/msms level data
batch integer vector with length equal to number of injections in xset or csv file or dataframe
order integer vector with length equal to number of injections in xset or csv file or dataframe
qc logical vector with length equal to number of injections in xset or csv file or dataframe
qc.inj.range integer: how many injections around each injection are to be scanned for presence of QC samples when using batch.qc normalization? A good rule of thumb is between 1 and 3 times the typical injection span between QC injections. i.e. if you inject QC ever 7 samples, set this to between 7 and 21. smaller values provide more local precision but make normalization sensitive to individual poor outliers (though these are first removed using the boxplot function outlier detection), while wider values provide less local precision in normalization but better stability to individual peak areas.

Value

normalized data.
**normalized_data_tic**  

**Description**

normalize data using TIC

**Usage**

```r
normalized_data_tic(ramclustObj = NULL)
```

**Arguments**

- `ramclustObj`  
  `ramclustObj` containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

**Value**

`ramclustR` object with total extracted ion normalized data.

**order_datasets**

**Description**

order the datasets first by batch and run order

**Usage**

```r
order_datasets(order = NULL, batch = NULL, qc = NULL, data = NULL)
```

**Arguments**

- `order`  
  integer vector with length equal to number of injections in xset or csv file or dataframe
- `batch`  
  integer vector with length equal to number of injections in xset or csv file or dataframe
- `qc`  
  logical vector with length equal to number of injections in xset or csv file or dataframe
- `data`  
  feature in ms/msms level data

**Value**

ordered feature in ms/msms level data, order, batch, qc
Description

Main clustering function for grouping features based on their analytical behavior.

Usage

```r
ramclustR(
  xcmsObj = NULL,
  ms = NULL,
  pheno_csv = NULL,
  idmsms = NULL,
  taglocation = "filepaths",
  MStag = NULL,
  idMSMStag = NULL,
  featdelim = ".",
  timepos = 2,
  st = NULL,
  sr = NULL,
  maxt = NULL,
  deepSplit = FALSE,
  blocksize = 2000,
  mult = 5,
  hmax = NULL,
  sampNameCol = 1,
  collapse = TRUE,
  usePheno = TRUE,
  mspout = TRUE,
  ExpDes = NULL,
  normalize = "TIC",
  qc.inj.range = 20,
  order = NULL,
  batch = NULL,
  qc = NULL,
  minModuleSize = 2,
  linkage = "average",
  mzdec = 3,
  cor.method = "pearson",
  rt.only.low.n = TRUE,
  fftempdir = NULL,
  replace.zeros = TRUE
)
```

Arguments

- `xcmsObj` xcmsObject: containing grouped feature data for clustering by `ramclustR`
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ms</td>
<td>filepath: optional csv input. Features as columns, rows as samples. Column header mz_rt</td>
</tr>
<tr>
<td>pheno_csv</td>
<td>filepath: optional csv input containing phenoData</td>
</tr>
<tr>
<td>idmsms</td>
<td>filepath: optional idMSMS / MSe csv data. same dim and names as ms required</td>
</tr>
<tr>
<td>taglocation</td>
<td>character: &quot;filepaths&quot; by default, &quot;phenoData[,1]&quot; is another option. refers to xcms slot</td>
</tr>
<tr>
<td>MStag</td>
<td>character: character string in 'taglocation' to designat MS / MSe files e.g. &quot;01.cdf&quot;</td>
</tr>
<tr>
<td>idMSMStag</td>
<td>character: character string in 'taglocation' to designat idMSMS / MSe files e.g. &quot;02.cdf&quot;</td>
</tr>
<tr>
<td>featdelim</td>
<td>character: how feature mz and rt are delimited in csv import column header e.g. =&quot;_&quot;</td>
</tr>
<tr>
<td>timepos</td>
<td>integer: which position in delimited column header represents the retention time (csv only)</td>
</tr>
<tr>
<td>st</td>
<td>numeric: sigma t - time similarity decay value</td>
</tr>
<tr>
<td>sr</td>
<td>numeric: sigma r - correlational similarity decay value</td>
</tr>
<tr>
<td>maxt</td>
<td>numeric: maximum time difference to calculate retention similarity for - all values beyond this are assigned similarity of zero</td>
</tr>
<tr>
<td>deepSplit</td>
<td>logical: controls how aggressively the HCA tree is cut - see ?cutreeDynamicTree</td>
</tr>
<tr>
<td>blocksize</td>
<td>integer: number of features (scans?) processed in one block =1000,</td>
</tr>
<tr>
<td>mult</td>
<td>numeric: internal value, can be used to influence processing speed/ram usage</td>
</tr>
<tr>
<td>hmax</td>
<td>numeric: precut the tree at this height, default 0.3 - see ?cutreeDynamicTree</td>
</tr>
<tr>
<td>sampNameCol</td>
<td>integer: which column from the csv file contains sample names?</td>
</tr>
<tr>
<td>collapse</td>
<td>logical: reduce feature intensities to spectrum intensities?</td>
</tr>
<tr>
<td>usePheno</td>
<td>logical: transfer phenotype data from XCMS object to SpecAbund dataset?</td>
</tr>
<tr>
<td>mspout</td>
<td>logical: write msp formatted spectra to file?</td>
</tr>
<tr>
<td>ExpDes</td>
<td>either an R object created by R ExpDes object: data used for record keeping and labelling msp spectral output</td>
</tr>
<tr>
<td>normalize</td>
<td>character: either &quot;none&quot;, &quot;TIC&quot;, &quot;quantile&quot;, or &quot;batch.qc&quot; normalization of feature intensities. see batch.qc overview in details.</td>
</tr>
<tr>
<td>qc.inj.range</td>
<td>integer: how many injections around each injection are to be scanned for presence of QC samples when using batch.qc normalization? A good rule of thumb is between 1 and 3 times the typical injection span between QC injections. i.e. if you inject QC ever 7 samples, set this to between 7 and 21. smaller values provide more local precision but make normalization sensitive to individual poor outliers (though these are first removed using the boxplot function outlier detection), while wider values provide less local precision in normalization but better stability to individual peak areas.</td>
</tr>
<tr>
<td>order</td>
<td>integer vector with length equal to number of injections in xset or csv file</td>
</tr>
<tr>
<td>batch</td>
<td>integer vector with length equal to number of injections in xset or csv file</td>
</tr>
<tr>
<td>qc</td>
<td>logical vector with length equal to number of injections in xset or csv file.</td>
</tr>
<tr>
<td>minModuleSize</td>
<td>integer: how many features must be part of a cluster to be returned? default = 2</td>
</tr>
</tbody>
</table>
linkage character: hierarchical clustering linkage method - see ?hclust
mzdec integer: number of decimal places used in printing m/z values
cor.method character: which correlational method used to calculate 'r' - see ?cor
rt.only.low.n logical: default = TRUE At low injection numbers, correlational relationships of peak intensities may be unreliable. by default ramclustR will simply ignore the correlational r value and cluster on retention time alone. if you wish to use correlation with at n < 5, set this value to FALSE.
fftempdir valid path: if there are file size limitations on the default ff package temp directory - getOptions('fftempdir') - you can change the directory used as the ffftempdir with this option.
replace.zeros logical: TRUE by default. NA, NaN, and Inf values are replaced with zero, and zero values are sometimes returned from peak peaking. When TRUE, zero values will be replaced with a small amount of noise, with noise level set based on the detected signal intensities for that feature.

Details
Main clustering function output - see citation for algorithm description or vignette('RAMClustR') for a walk through. batch.qc. normalization requires input of three vectors (1) batch (2) order (3) qc. This is a feature centric normalization approach which adjusts signal intensities first by comparing batch median intensity of each feature (one feature at a time) QC signal intensity to full dataset median to correct for systematic batch effects and then secondly to apply a local QC median vs global median sample correction to correct for run order effects.

Value
$featclus: integer vector of cluster membership for each feature
$frt: feature retention time, in whatever units were fed in (xcms uses seconds, by default)
$fmz: feature retention time, reported in number of decimal points selected in ramclustR function
$xcmsOrd: the original XCMS (or csv) feature order for cross referencing, if need be
$clrt: cluster retention time
$clrtsd: retention time standard deviation of all the features that comprise that cluster
$nfeat: number of features in the cluster
$nsing: number of 'singletons' - that is the number of features which clustered with no other feature
$ExpDes: the experimental design object used when running ramclustR. List of two dataframes.
$cmpd: compound name. C#### are assigned in order of output by dynamicTreeCut. Compound with the most features is classified as C0001...
$ann: annotation. By default, annotation names are identical to 'cmpd' names. This slot is a placeholder for when annotations are provided
$MSdata: the MSdataset provided by either xcms or csv input
$MSMSdata: the (optional) MSe/idMSMS dataset provided be either xcms or csv input
$SpecAbund: the cluster intensities after collapsing features to clusters
$SpecAbundAve: the cluster intensities after averaging all samples with identical sample names
- 'spectra' directory is created in the working directory. In this directory a .msp is (optionally) created, which contains the spectra for all compounds in the dataset following clustering. If MSe/idMSMS data are provided, they are listed with the same compound name as the MS spectrum, with the collision energy provided in the ExpDes object provided to distinguish low from high CE spectra.

Author(s)
Corey Broeckling

References


Examples
## Choose input file with feature column names `mz_rt` (expected by default).
## Column with sample name is expected to be first (by default).
## These can be adjusted with the `featdelim` and `sampNameCol` parameters.
wd <- getwd()
filename <- system.file("extdata", "peaks.csv", package = "RAMClustR", mustWork = TRUE)
print(filename)
head(data.frame(read.csv(filename)), c(6L, 5L))

## If the file contains features from MS1, assign those to the `ms` parameter.
## If the file contains features from MS2, assign those to the `idmsms` parameter.
## If you ran `xcms` for the feature detection, assign the output to the `xcmsObj` parameter.
## In this example we use a MS1 feature table stored in a `csv` file.
setwd(tempdir())
ramclustobj <- ramclustR(ms = filename, st = 5, maxt = 1, blocksize = 1000)

## Investigate the deconvoluted features in the `spectra` folder in MSP format
## or inspect the `ramclustobj` for feature retention times, annotations etc.
print(ramclustobj$ann)
print(ramclustobj$nfeat)
print(ramclustobj$SpecAbund[, 1:6])
setwd(wd)

Description
extractor for xcms objects in preparation for clustering
Usage

rc.calibrate.ri(ramclustObj = NULL, calibrant.data = "", poly.order = 3)

Arguments

ramclustObj ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
calibrant.data character vector defining the file path/name to a csv file containing columns including 'rt', and 'ri'. Alternatively, a data.frame with those columnn names (case sensitive)
poly.order integer default = 3. polynomical order used to fit rt vs ri data, and calculate ri for all feature and metabolite rt values. poly.order should be apprciably smaller than the number of calibrant points.

Details

This function generates a new slot in the ramclustR object for retention index. Calibration is performed using a polynomial fit of order poly.order. It is the user’s responsibility to ensure that the number and span of calibrant points is sufficient to calibrate the full range of feature and compound retention times. i.e. if the last calibration point is at 1000 seconds, but the last eluting peak is at 1300 seconds, the calibration will be very poor for the late eluting compound.

Value

ramclustR object with retention index assigned for features ($fri) and compounds ($clri).

Author(s)

Corey Broeckling

References


Description

used to remove compounds which are found at similar intensity in blank samples. Only applied after clustering. see also rc.feature.filter.blanks for filtering at the feature level (only done before clustering).
Usage

rc.cmpd.filter.blanks(
  ramclustObj = NULL,
  qc.tag = "QC",
  blank.tag = "blank",
  sn = 3,
  remove.blanks = TRUE
)

Arguments

ramclustObj ramclustObj containing SpecAbund dataframe.
qc.tag character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.
blank.tag see 'qc.tag', but for blanks to use as background.
sn numeric defines the ratio for 'signal'. i.e. sn = 3 indicates that signal intensity must be 3 fold higher in sample than in blanks, on average, to be retained.
remove.blanks logical. TRUE by default. this removes any recognized blanks samples from the SpecAbund sets after they are used to filter contaminant compounds

Details

This function removes compounds which contain signal in QC samples comparable to blanks.

Value

ramclustR object with normalized data.

Author(s)

Corey Broeckling

References

Description

extractor for xcms objects in preparation for clustering

Usage

```r
crcmpd.filter.cv(ramclustObj = NULL, qc.tag = "QC", max.cv = 0.5)
```

Arguments

- `ramclustObj` : ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `qc.tag` : character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.
- `max.cv` : numeric maximum allowable cv for any feature. default = 0.3

Details

This function offers normalization by total extracted ion signal. it is recommended to first run 'rc.feature.filter.blanks' to remove non-sample derived signal.

Value

ramclustR object with total extracted ion normalized data.

Author(s)

Corey Broeckling

References

rc.cmpd.get.classyfire

classyfire

Description

use classyfire web API to look up full ClassyFire hierarchy for each inchikey

Usage

rc.cmpd.get.classyfire(
    ramclustObj = NULL,
    inchikey = NULL,
    get.all = TRUE,
    max.wait = 10,
    posts.per.minute = 5
)

Arguments

ramclustObj ramclustR object to ClassyFy. Must supply one of either ramclustObj or inchikey (see below)
inchikey vector of text inchikeys to ClassyFy. Must supply one of either ramclustObj or inchikey.
get.all logical; if TRUE, when inchikey classyfire lookup fails, submits for classificaiton. Can be slow. max.wait (below) sets max time to spend on each compound before moving on. default = FALSE.
max.wait numeric; maximum time (seconds) to wait per compound when ‘get.all’ = TRUE.
posts.per.minute integer; a limit set when ‘get.all’ is true. ClassyFire server accepts no more than 5 posts per minute when calculating new ClassyFire results. Slows down submission process to keep server from denying access.

Details

The $inchikey slot is used to look up the

Value

returns a ramclustR object. new dataframe in $classyfire slot with rows equal to number of compounds.

Author(s)

Corey Broeckling
References


Description

use pubchem rest and view APIs to retrieve structures, CIDs (if a name or inchikey is given), synonyms, and optionally vendor data, when available.

Usage

rc.cmpd.get.pubchem(ramclustObj = NULL, search.name = NULL, cmpd.names = NULL, cmpd.cid = NULL, cmpd.inchikey = NULL, cmpd.smiles = NULL, use.parent.cid = FALSE, manual.entry = FALSE, get.vendors = FALSE, priority.vendors = c("Sigma Aldrich", "Alfa Chemistry", "Acros Organics", "VWR", "Alfa Aesar", "molport", "Key Organics", "BLD Pharm"), get.properties = TRUE, all.props = FALSE, get.synonyms = TRUE, find.short.lipid.name = TRUE, find.short.synonym = TRUE, max.name.length = 30, assign.short.name = TRUE, get.bioassays = TRUE, get.pathways = TRUE, write.csv = TRUE)

Arguments

ramclustObj  RAMClust Object input. if used, ramclustObj$CID, ramclustObj$inchikey, and ramclustObj$ann are used as input, in that order, and ramclustObj is returned with $pubchem slot appended.

search.name  character. optional name to assign to pubchem search to name output .csv files.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmpd.names</td>
<td>character vector. i.e. c(&quot;caffeine&quot;, &quot;theobromine&quot;, &quot;glucose&quot;)</td>
</tr>
<tr>
<td>cmpd.cid</td>
<td>numeric integer vector. i.e. c(2519, 5429, 107526)</td>
</tr>
<tr>
<td>cmpd.inchikey</td>
<td>character vector. i.e. c(&quot;RYYVLZVUVIJVGH-UHFFFAOYSA-N&quot;, &quot;YAPQBXQYLJRXSA-UHFFFAOYSA-N&quot;, &quot;GZCGUPFRVQAUEE-SLPGGIOYSA-N&quot;)</td>
</tr>
<tr>
<td>cmpd.smiles</td>
<td>character vector. i.e. c(&quot;CN1C=NC2=C1C(=O)N(C(=O)N2C)C&quot;, &quot;CN1C=NC2=C1C(=O)NC(=O)N2C&quot;)</td>
</tr>
<tr>
<td>use.parent.cid</td>
<td>logical. If TRUE, the CID for each supplied name/inchikey is used to retrieve its parent CID (i.e. the parent of sodium palmitate is palmitic acid). The parent CID is used to retrieve all other names, properties.</td>
</tr>
<tr>
<td>manual.entry</td>
<td>logical. If TRUE, user input is enabled for compounds not matched by name. A browser window will open with the pubchem search results in your default browser.</td>
</tr>
<tr>
<td>get.vendors</td>
<td>logical. If TRUE, vendor data is returned for each compound with a matched CID. Includes vendor count and vendor product URL, if available</td>
</tr>
<tr>
<td>priority.vendors</td>
<td>character vector. i.e. c(&quot;MyFavoriteCompany&quot;, &quot;MySecondFavoriteCompany&quot;). If these vendors are found, the URL returned is from priority vendors. Priority is given by order input by user.</td>
</tr>
<tr>
<td>get.properties</td>
<td>logical. If TRUE, physicochemical property data are returned for each compound with a matched CID.</td>
</tr>
<tr>
<td>all.props</td>
<td>logical. If TRUE, all pubchem properties (<a href="https://pubchemdocs.ncbi.nlm.nih.gov/pug-rest$_Toc494865567">https://pubchemdocs.ncbi.nlm.nih.gov/pug-rest$_Toc494865567</a>) are returned. If false, only a subset (faster).</td>
</tr>
<tr>
<td>get.synonyms</td>
<td>TRUE. logical. If TRUE, retrieve pubchem synonyms. returned to $synonyms slot</td>
</tr>
<tr>
<td>find.short.lipid.name</td>
<td>TRUE. logical. If TRUE, and get.synonyms = TRUE, looks for lipid short hand names in synonyms list (i.e. PC(36:6)). returned to $short.name slot. Short names are assigned only if assign.short.names = TRUE.</td>
</tr>
<tr>
<td>find.short.synonym</td>
<td>TRUE. logical. If TRUE, and get.synonyms = TRUE, looks for lipid short synonym names, with prioritization for names with fewer numeric characters (i.e. database accession numbers or CAS numbers). returned to $short.name slot. Short names are assigned only if assign.short.names = TRUE.</td>
</tr>
<tr>
<td>max.name.length</td>
<td>20. integer. If names are longer than this value, short names will be searched for, else, retain original name.</td>
</tr>
<tr>
<td>assign.short.name</td>
<td>TRUE. If TRUE, short names from find.short.lipid.name and/or find.short.synonym = TRUE, short names are assigned the be the default annotation name ($ann slot), and original annotations are moved to $long.name slot.</td>
</tr>
<tr>
<td>get.bioassays</td>
<td>logical. If TRUE, return a table summarizing existing bioassay data for that CID.</td>
</tr>
<tr>
<td>get.pathways</td>
<td>logical. If TRUE, return a table of metabolic pathways for that CID.</td>
</tr>
<tr>
<td>write.csv</td>
<td>logical. If TRUE, write csv files of all returned pubchem data.</td>
</tr>
</tbody>
</table>
rc.cmpd.get.smiles.inchi

**Details**

useful for moving from chemical name to digital structure representation. greek letters are assumed to be 'UTF-8' encoded, and are converted to latin text before searching. if you are reading in your compound name list, do so with 'encoding' set to 'UTF-8'.

**Value**

returns a list with one or more of $pubchem (compound name and identifiers) - one row in dataframe per CID; $properties contains physicochemical properties - one row in dataframe per CID; $vendors contains the number of vendors for a given compound and selects a vendor based on 'priority.vendors' supplied, or randomly chooses a vendor with a HTML link - one row in dataframe per CID; $bioassays contains a summary of bioassay activity data from pubchem - zero to many rows in dataframe per CID

**Author(s)**

Corey Broeckling

---

**Description**

use PubChem API to look up full smiles and inchi notation for each inchikey

**Usage**

```r
rc.cmpd.get.smiles.inchi(
  ramclustObj = NULL,
  inchikey = NULL,
  ignore.stereo = TRUE
)
```

**Arguments**

- `ramclustObj` ramclustR object to look up smiles and inchi for each inchikey (without a smiles/inchi). Must provide one of ramclustObj or inchikey.
- `inchikey` character vector of inchikey strings. Must provide one of ramclustObj or inchikey.
- `ignore.stereo` logical. default = TRUE. If the Pubchem databases does not have the full inchikey string, should we search by the first (non-stereo) block of the inchikey? When true, returns the first pubchem match to the inchikey block one string. If the full inchikey is present, that is used preferentially.
Details

The `$inchikey` slot is used to look up parameters from pubchem. PubChem CID, a pubchem URL, smiles (canonical) and inchi are returned. If smiles and inchi slots are already present (from MS-Finder, for example) pubchem smiles and inchi are used to fill in missing values only, not replace.

Value

returns a ramclustR object. new vector of `$smiles` and `$inchi` with length equal to number of compounds.

Author(s)

Corey Broeckling

References


Description

replaces any NA (and optionally zero) values with small signal (20)

Usage

```r
rc.cmpd.replace.na(
  ramclustObj = NULL,
  replace.int = 0.1,
  replace.noise = 0.1,
  replace.zero = TRUE
)
```

Arguments

- `ramclustObj` ramclustObj containing SpecAbund dataset
- `replace.int` default = 0.2. proportion of minimum feature value to replace NA (or zero) values with
- `replace.noise` default = 0.2. proportion of `replace.int` value by which noise is added via 'jitter'
- `replace.zero` logical if TRUE, any zero values are replaced with noise as if they were NA values
rc.expand.sample.names

Details

noise is added by finding for each feature the minimum detected value, multiplying that value by replace.int, then adding (replace.int*replace.noise) noise. abs() is used to ensure no negative values result.

Value

ramclustR object with NA and zero values removed.

Author(s)

Corey Broeckling

References


description

rc.expand.sample.names

Usage

rc.expand.sample.names(
  ramclustObj = NULL,
  delim = "-",
  factor.names = TRUE,
  quiet = FALSE
)

Arguments

ramclustObj ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
delim what delimiter should be used to separate names into factors? '-' by default
factor.names logical or character vector. if TRUE, user will enter names one by one in console. If character vector (i.e. c("trt", "time"). names are assigned to table
quiet logical. if TRUE, user will not be prompted to enter names one by one in console.
**rc.export.msp.rc**

**Details**

This function only works on newer format ramclustObjects with a $phenoData slot. This function will split sample names by a delimiter, and enable users to name factors.

**Value**

ramclustR object with normalized data.

**Author(s)**

Corey Broeckling

**References**


---

**Description**

Cluster annotation function: inference of 'M' - molecular weight of the compound giving rise to each spectrum - using the InterpretMSSpectrum::findMain function.

**Usage**

```r
rc.export.msp.rc(ramclustObj = NULL, one.file = TRUE, mzdec = 1)
```

**Arguments**

- `ramclustObj` ramclustR object to annotate.
- `one.file` logical, should all msp spectra be written to one file? If false, each spectrum is an individual file.
- `mzdec` integer. Number of decimal points to export mass values with.

**Details**

exports files to a directory called 'spectra'. If one.file = FALSE, a new directory 'spectra/msp' is created to hold the individual msp files. if do.findman has been run, spectra are written as ms2 spectra, else as ms1.

**Value**

nothing, just exports files to the working directory
rc.feature.filter.blanks

Description
used to remove features which are found at similar intensity in blank samples

Usage
rc.feature.filter.blanks(
  ramclustObj = NULL,
  qc.tag = "QC",
  blank.tag = "blank",
  sn = 3,
  remove.blanks = TRUE
)

Arguments
ramclustObj ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
qc.tag character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.
blank.tag see 'qc.tag', but for blanks to use as background.
sn numeric defines the ratio for 'signal'. i.e. sn = 3 indicates that signal intensity must be 3 fold higher in sample than in blanks, on average, to be retained.
remove.blanks logical. TRUE by default. this removes any recognized blanks samples from the MSdata and MSMSdata sets after they are used to filter contaminant features.

Details
This function offers normalization by run order, batch number, and QC sample signal intensity. Each input vector should be the same length, and equal to the number of samples in the $MSdata set. Input vector order is assumed to be the same as the sample order in the $MSdata set.

Value
ramclustR object with normalized data.
**Author(s)**
Corey Broeckling

**References**

---

**Description**
extractor for xcms objects in preparation for clustering

**Usage**
```r
c.feature.filter.cv(ramclustObj = NULL, qc.tag = "QC", max.cv = 0.5)
```

**Arguments**
- `ramclustObj`: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `qc.tag`: character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.
- `max.cv`: numeric maximum allowable cv for any feature. default = 0.5

**Details**
This function offers normalization by total extracted ion signal. it is recommended to first run `rc.feature.filter.blanks` to remove non-sample derived signal.

**Value**
ramclustR object with total extracted ion normalized data.

**Author(s)**
Corey Broeckling

**References**
Description

normalize data using batch.qc

Usage

```
rc.feature.normalize.batch.qc(
  order = NULL,
  batch = NULL,
  qc = NULL,
  ramclustObj = NULL,
  qc.inj.range = 20
)
```

Arguments

- **order**: integer vector with length equal to number of injections in xset or csv file or dataframe
- **batch**: integer vector with length equal to number of injections in xset or csv file or dataframe
- **qc**: logical vector with length equal to number of injections in xset or csv file or dataframe
- **ramclustObj**: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- **qc.inj.range**: integer: how many injections around each injection are to be scanned for presence of QC samples when using batch.qc normalization? A good rule of thumb is between 1 and 3 times the typical injection span between QC injections. i.e. if you inject QC ever 7 samples, set this to between 7 and 21. smaller values provide more local precision but make normalization sensitive to individual poor outliers (though these are first removed using the boxplot function outlier detection), while wider values provide less local precision in normalization but better stability to individual peak areas.

Value

ramclustR object with normalized data.
**Description**

extractor for xcms objects in preparation for clustering

**Usage**

```r
rc.feature.normalize.qc(
  ramclustObj = NULL,
  order = NULL,
  batch = NULL,
  qc.tag = NULL,
  output.plot = FALSE,
  p.cut = 0.05,
  rsq.cut = 0.1,
  p.adjust = "none"
)
```

**Arguments**

- `ramclustObj`: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `order`: integer vector with length equal to number of injections in xset or csv file
- `batch`: integer vector with length equal to number of injections in xset or csv file
- `qc.tag`: character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the ‘sample.names’ slot by default.
- `output.plot`: logical: if TRUE (default), plots are output to PDF.
- `p.cut`: numeric when run order correction is applied, only features showing a run order vs signal with a linear p-value (after FDR correction) < p.cut will be adjusted. also requires r-squared < rsq.cut.
- `rsq.cut`: numeric when run order correction is applied, only features showing a run order vs signal with a linear r-squared > rsq.cut will be adjusted. also requires p values < p.cut.
- `p.adjust`: which p-value adjustment should be used? default = "none", see ?p.adjust

**Details**

This function offers normalization by run order, batch number, and QC sample signal intensity.

Each input vector should be the same length, and equal to the number of samples in the $MSdata set.

Input vector order is assumed to be the same as the sample order in the $MSdata set.
**Value**

ramclustR object with normalized data.

**Author(s)**

Corey Broeckling

**References**


---

**Description**

normalize data using quantile

**Usage**

```
rc.feature.normalize.quantile(ramclustObj = NULL)
```

**Arguments**

- `ramclustObj` ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

**Value**

ramclustR object with normalized data.

---

**Description**

extractor for xcms objects in preparation for clustering

**Usage**

```
rc.feature.normalize.tic(ramclustObj = NULL)
```
Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)

Details

This function offers normalization by total extracted ion signal. It is recommended to first run 'rc.feature.filter.blanks' to remove non-sample derived signal.

Value

ramclustR object with total extracted ion normalized data.

Author(s)

Corey Broeckling

References


Description

replaces any NA (and optionally zero) values with small signal (20)

Usage

rc.feature.replace.na(  
  ramclustObj = NULL,  
  replace.int = 0.1,  
  replace.noise = 0.1,  
  replace.zero = TRUE,  
  which.data = c("MSdata", "MSMSdata")  
)

Arguments

ramclustObj  ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
replace.int  default = 0.1. proportion of minimum feature value to replace NA (or zero) values with
replace.noise  default = 0.1. proportion of replace.int value by which noise is added via 'jitter'
replace.zero  logical if TRUE, any zero values are replaced with noise as if they were NA values
which.data  name of dataset
Details

noise is added by finding for each feature the minimum detected value, multiplying that value by replace.int, then adding (replace.int*replace.noise) noise. abs() is used to ensure no negative values result.

Value

ramclustR object with NA and zero values removed.

Author(s)

Corey Broeckling

References


description

equaler for csv objects in preparation for normalization and clustering

Usage

rc.get.csv.data(
csv = NULL,
phenoData = NULL,
idmsms = NULL,
ExpDes = NULL,
sampNameCol = 1,
st = NULL,
timepos = 2,
featdelim = "_",
ensure.no.na = TRUE
)

Arguments

csv filepath: csv input. Features as columns, rows as samples. Column header mz_rt
phenoData character: character string in ’taglocation’ to designate files as either MS / DIA(MSe, MSall, AIF, etc) e.g. ”01.mzML”
idmsms filepath: optional idMSMS / MSe csv data. same dim and names as ms required
ExpDes: either an R object created by R ExpDes object: data used for record keeping and labelling msp spectral output
sampNameCol: integer: which column from the csv file contains sample names?
st: numeric: sigma t - time similarity decay value
timepos: integer: which position in delimited column header represents the retention time
featdelim: character: how feature mz and rt are delimited in csv import column header e.g. ="-"
ensure.no.na: logical: if TRUE, any 'NA' values in msint and/or msmsint are replaced with numerical values based on 10 percent of feature min plus noise. Used to ensure that spectra are not written with NA values.

Details
This function creates a ramclustObj which will be used as input for clustering.

Value
an empty ramclustR object. this object is formatted as an hclust object with additional slots for holding feature and compound data. details on these found below.
$idrt: feature retention time, in whatever units were fed in
$idmz: feature retention time, reported in number of decimal points selected in ramclustR function
$ExpDes: the experimental design object used when running ramclustR. List of two dataframes.
$MSdata: the MSdataset provided by either xcms or csv input
$MSMSdata: the (optional) DIA(MSe, MSall, AIF etc) dataset
$xcmsOrd: original xcms order of features, for back-referencing when necessary
$msint: weighted.mean intensity of feature in ms level data
$msmsint:weighted.mean intensity of feature in msms level data

Author(s)
Corey Broeckling

References
Examples

## Choose csv input file. Features as columns, rows as samples
## Choose csv input file phenoData
filename <- system.file("extdata", "peaks.csv", package = "RAMClustR", mustWork = TRUE)
phenoData <- system.file("extdata", "phenoData.csv", package = "RAMClustR", mustWork = TRUE)

ramclustobj <- rc.get.csv.data(csv = filename, phenoData = phenoData, st = 5)

rc.get.df.data

Description

extractor for dataframe input in preparation for normalization and clustering

Usage

rc.get.df.data(
  ms1_featureDefinitions = NULL,
  ms1_featureValues = NULL,
  ms2_featureDefinitions = NULL,
  ms2_featureValues = NULL,
  phenoData = NULL,
  ExpDes = NULL,
  featureNamesColumnIndex = 1,
  st = NULL,
  ensure.no.na = TRUE
)

Arguments

ms1_featureDefinitions
dataframe with metadata with columns: mz, rt, feature names containing MS data

ms1_featureValues
dataframe with rownames = sample names, colnames = feature names containing MS data

ms2_featureDefinitions
dataframe with metadata with columns: mz, rt, feature names containing MSMS data

ms2_featureValues
dataframe with rownames = sample names, colnames = feature names containing MSMS data

phenoData
dataframe containing phenoData

ExpDes
either an R object created by R ExpDes object: data used for record keeping and labelling msp spectral output
featureNamesColumnIndex
  integer: which column in ‘ms1_featureDefinitions’ contains feature names?

st
  numeric: sigma t - time similarity decay value

ensure.no.na
  logical: if TRUE, any ‘NA’ values in msint and/or msmsint are replaced with
  numerical values based on 10 percent of feature min plus noise. Used to ensure
  that spectra are not written with NA values.

Details

This function creates a ramclustObj which will be used as input for clustering.

Value

an empty ramclustR object. this object is formatted as an hclust object with additional slots for
holding feature and compound data. details on these found below.

$frt: feature retention time, in whatever units were fed in
$fmz: feature retention time, reported in number of decimal points selected in ramclustR function
$ExpDes: the experimental design object used when running ramclustR. List of two dataframes.
$MSdata: the MSdataset provided by either xcms or csv input
$MSMSdata: the (optional) DIA(MSe, MSall, AIF etc) dataset
$xcmsOrd: original xcms order of features, for back-referencing when necessary
$msmsint: weighted.mean intensity of feature in ms level data
$msmsint: weighted.mean intensity of feature in msms level data

Author(s)

Zargham Ahmad, Helge Hecht, Corey Broeckling

References


Efficient and Confident Annotation of LC-MS Metabolomics Data through MS1 Spectrum and Time

Examples

## Choose dataframe with metadata with columns: mz, rt, feature names containing MS data
## Choose dataframe with rownames = sample names, colnames = feature names containing MS data
## Choose dataframe containing phenoData
df1 <- readRDS(system.file("extdata", "featDefinition.rds", package = "RAMClustR", mustWork = TRUE))
df2 <- readRDS(system.file("extdata", "featValues.rds", package = "RAMClustR", mustWork = TRUE))
df3 <- readRDS(system.file("extdata", "phenoData_df.rds", package = "RAMClustR", mustWork = TRUE))
ramclustr <- rc.get.df.data(ms1_featureDefinitions=df1, ms1_featureValues=df2, phenoData=df3, st=5)

rc.get.xcms.data
rc.get.xcms.data

Description
extractor for xcms objects in preparation for normalization and clustering

Usage
rc.get.xcms.data(
  xcmsObj = NULL,
  taglocation = "filepaths",
  MStag = NULL,
  MSMStag = NULL,
  ExpDes = NULL,
  mzdec = 3,
  ensure.no.na = TRUE
)

Arguments
xcmsObj xcmsObject: containing grouped feature data for clustering by ramclustR
taglocation character: "filepaths" by default, "phenoData[,1]" is another option. refers to xcms slot
MStag character: character string in 'taglocation' to designate files as either MS / DIA(MSe, MSall, AIF, etc) e.g. "01.mzML"
MSMStag character: character string in 'taglocation' to designate files as either MS / DIA(MSe, MSall, AIF, etc) e.g. "02.mzML"
ExpDes either an R object created by R ExpDes object: data used for record keeping and labelling msp spectral output
mzdec integer: number of decimal places for storing m/z values
ensure.no.na logical: if TRUE, any 'NA' values in msint and/or msmsint are replaced with numerical values based on 10 percent of feature min plus noise. Used to ensure that spectra are not written with NA values.

Details
This function creates a ramclustObj which will be used as input for clustering.
Value

an empty ramclustR object. this object is formatted as an hclust object with additional slots for holding feature and compound data. details on these found below.

$frt: feature retention time, in whatever units were fed in (xcms uses seconds, by default)
$fmz: feature retention time, reported in number of decimal points selected in ramclustR function
$ExpDes: the experimental design object used when running ramclustR. List of two dataframes.
$MSdata: the MSdataset provided by either xcms or csv input
$MSMSdata: the (optional) DIA(MSe, MSall, AIF etc) dataset provided be either xcms or csv input
$xcmsOrd: original xcms order of features, for back-referencing when necessary
$msint: weighted.mean intensity of feature in ms level data
$msmsint: weighted.mean intensity of feature in msms level data

Author(s)

Corey Broeckling

References


Description

summarize quality control for clustering and for quality control sample variation based on compound ($SpecAbund) and feature ($MSdata and $MSMSdata, if present)

Usage

rc.qc(
  ramclustObj = NULL,
  qc.tag = "QC",
  remove.qc = FALSE,
  npc = 4,
  scale = "pareto",
  outfile.basename = "ramclustQC",
  view.hist = TRUE
)

Arguments

- **ramclustObj**: ramclustR object to analyze
- **qc.tag**: qc.tag character vector of length one or two. If length is two, enter search string and factor name in $PhenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.
- **remove.qc**: logical - if TRUE (default) QC injections will be removed from the returned ramclustObj (applies to $MSdata, $MSMSdata, $SpecAbund, $PhenoData, as appropriate). If FALSE, QC samples remain.
- **npc**: number of Principle components to calculate and plot
- **scale**: "pareto" by default: PCA scaling method used
- **outfile.basename**: base name of output files. Extensions added internally. default = "ramclustQC"
- **view.hist**: logical. should histograms be plotted?

Details

plots a ramclustR summary plot. first page represents the correlation of each cluster to all other clusters, sorted by retention time. large blocks of yellow along the diagonal indicate either poor clustering or a group of coregulated metabolites with similar retention time. It is an imperfect diagnostic, particularly with lipids on reverse phase LC or sugars on HILIC LC systems. Page 2: histogram of r values from page 1 - only r values one position from the diagonal are used. Pages 3:5 - PCA results, with QC samples colored red. relative standard deviation calculated as sd(QC PC scores) / sd(all PC scores). Page 6: histogram of CV values for each compound in the dataset, QC samples only.

Value

new RC object. Saves output summary plots to pdf and .csv summary tables to new 'QC' directory. If remove.qc = TRUE, moves QC samples to new $QC slot from original position.

Author(s)

Corey Broeckling

References


Description

Main clustering function for grouping features based on their analytical behavior.

Usage

```r
rc.ramclustr(
  ramclustObj = NULL,
  st = NULL,
  sr = NULL,
  maxt = NULL,
  deepSplit = FALSE,
  blocksize = 2000,
  mult = 5,
  hmax = NULL,
  collapse = TRUE,
  minModuleSize = 2,
  linkage = "average",
  cor.method = "pearson",
  rt.only.low.n = TRUE,
  fftempdir = NULL
)
```

Arguments

- `ramclustObj` ramclustR object: containing ungrouped features. constructed by `rc.get.xcms.data`, for example
- `st` numeric: sigma t - time similarity decay value
- `sr` numeric: sigma r - correlational similarity decay value
- `maxt` numeric: maximum time difference to calculate retention similarity for - all values beyond this are assigned similarity of zero
- `deepSplit` logical: controls how aggressively the HCA tree is cut - see ?cutreeDynamicTree
- `blocksize` integer: number of features (scans?) processed in one block =1000,
- `mult` numeric: internal value, can be used to influence processing speed/ram usage
- `hmax` numeric: precut the tree at this height, default 0.3 - see ?cutreeDynamicTree
- `collapse` logical: if true (default), feature quantitative values are collapsed into spectra quantitative values.
- `minModuleSize` integer: how many features must be part of a cluster to be returned? default = 2
- `linkage` character: heirarchical clustering linkage method - see ?hclust
- `cor.method` character: which correlational method used to calculate 'r' - see ?cor
rt.only.low.n logical: default = TRUE. At low injection numbers, correlational relationships of peak intensities may be unreliable. By default, ramclustR will simply ignore the correlational r value and cluster on retention time alone. If you wish to use correlation with \( n < 5 \), set this value to FALSE.

ffttempdir valid path: if there are file size limitations on the default ff package temp directory - getOptions('ffttempdir') - you can change the directory used as the ffttempdir with this option.

Details
Main clustering function output - see citation for algorithm description or vignette('RAMClustR') for a walk through. batch.qc. normalization requires input of three vectors (1) batch (2) order (3) qc. This is a feature centric normalization approach which adjusts signal intensities first by comparing batch median intensity of each feature (one feature at a time) QC signal intensity to full dataset median to correct for systematic batch effects and then secondly to apply a local QC median vs global median sample correction to correct for run order effects.

Value
$featclus: integer vector of cluster membership for each feature
$clrt: cluster retention time
$clrtsd: retention time standard deviation of all the features that comprise that cluster
$nfeat: number of features in the cluster
$nnsing: number of 'singletons' - that is the number of features which clustered with no other feature
$cmpd: compound name. C#### are assigned in order of output by dynamicTreeCut. Compound with the most features is classified as C0001...
$sann: annotation. By default, annotation names are identical to 'cmpd' names. This slot is a placeholder for when annotations are provided
$SpecAbund: the cluster intensities after collapsing features to clusters
$SpecAbundAve: the cluster intensities after averaging all samples with identical sample names

Author(s)
Corey Broeckling

References

Description

summarize quality control for clustering and for quality control sample variation based on compound ($SpecAbund) and feature ($MSdata and $MSMSdata, if present)

Usage

rc.remove.qc(ramclustObj = NULL, qc.tag = "QC")

Arguments

ramclustObj ramclustR object to analyze

qc.tag qc.tag character vector of length one or two. If length is two, enter search string and factor name in $phenoData slot (i.e. c("QC", "sample.type"). If length one (i.e. "QC"), will search for this string in the 'sample.names' slot by default.

Details

simply moves QC samples out of the way for downstream processing. moved to a $qc slot.

Value

new RC object. moves QC samples to new $qc slot from original position.

Author(s)

Corey Broeckling

References


Description

summarize quality control for clustering and for quality control sample variation based on compound ($SpecAbund) and feature ($MSdata and $MSMSdata, if present)

Usage

rc.restore.qc.samples(ramclustObj = NULL)

Arguments

ramclustObj ramclustR object to analyze

Details

moves all of $phenoData, $MSdata, $MSMSdata, $SpecAbund back to original positions from $qc slot

Value

RC object

Author(s)

Corey Broeckling

References


**Description**

filter RC object and summarize quality control sample variation

**Usage**

```r
RCQC(
  ramclustObj = NULL,
  qctag = "QC",
  npc = 4,
  scale = "pareto",
  which.data = "SpecAbund",
  outfile = "ramclustQC.pdf"
)
```

**Arguments**

- `ramclustObj`: ramclustR object to analyze
- `qctag`: "QC" by default - rowname tag to identify QC samples
- `npc`: number of Principle components to calculate and plot
- `scale`: "pareto" by default: PCA scaling method used
- `which.data`: which dataset to use. "SpecAbund" by default
- `outfile`: name of output pdf file.

**Details**

plots a ramclustR summary plot. first page represents the correlation of each cluster to all other clusters, sorted by retention time. large blocks of yellow along the diagonalq indicate either poor clustering or a group of coregulated metabolites with similar retention time. It is an imperfect diagnostic, particularly with lipids on reverse phase LC or sugars on HILIC LC systems. Page 2: histogram of r values from page 1 - only r values one position from the diagonal are used. Pages 3:5: PCA results, with QC samples colored red. relative standard deviation calculated as sd(QC PC scores) / sd(all PC scores). Page 6: histogram of CV values for each compound int he dataset, QC samples only.

**Value**

new RC object, with QC samples moved to new slot. prints output summary plots to pdf.

**Author(s)**

Corey Broeckling
References


---

remove_blanks

**Description**

remove blanks

**Usage**

```r
remove_blanks(ramclustObj, blank)
```

**Arguments**

- `ramclustObj`: ramclustObj containing MSdata with optional MSMSdata (MSe, DIA, idMSMS)
- `blank`: blank samples found by define_samples

**Value**

ramclustObj object with blanks removed

---

replace_na

**Description**

add rc.feature.replace.na params in ramclustObj

**Usage**

```r
replace_na(data, replace.int, replace.zero, replace.noise)
```
**write.methods**

**Arguments**

- **data**
  - selected data frame to use

- **replace.int**
  - default = 0.1. proportion of minimum feature value to replace NA (or zero) values with

- **replace.zero**
  - logical if TRUE, any zero values are replaced with noise as if they were NA values

- **replace.noise**
  - default = 0.1. proportion of replace.int value by which noise is added via 'jitter'

**Value**

- selected ramclustR data frame with NA and zero values removed.
- number of features replaced

---

**Description**

write RAMClustR processing methods and citations to text file

**Usage**

```r
write.methods(ramclustObj = NULL, filename = NULL)
```

**Arguments**

- **ramclustObj**
  - R object - the ramclustR object which was used to write the .mat or .msp files

- **filename**
  - define filename/path to write, uses 'ramclustr_methods.txt' and the working directory by default.

**Details**

this function exports a file called ramclustr_methods.txt which contains the processing history, parameters used, and relevant citations.

**Value**

- an annotated ramclustR object

  - nothing - new file written to working director

**Author(s)**

Corey Broeckling
References


write.msp

Description

Cluster annotation function: inference of 'M' - molecular weight of the compound giving rise to each spectrum - using the InterpretMSSpectrum::findMain function

Usage

write.msp(ramclustObj = NULL, one.file = FALSE)

Arguments

ramclustObj ramclustR object to annotate.

one.file logical, should all msp spectra be written to one file? If false, each spectrum is an individual file.

Details

exports files to a directory called 'spectra'. If one.file = FALSE, a new directory 'spectra/msp' is created to hold the individual msp files. If do.findman has been run, spectra are written as ms2 spectra, else as ms1.

Value

nothing, just exports files to the working directory

Author(s)

Corey Broeckling
write_csv

description

write csv template called "ExpDes.csv" to your working directory. you will fill this in manually, ensuring that when you save you retain csv format. ramclustR will then read this file in and and format appropriately.

usage

write_csv(data)

arguments

data csv template to write

value

read ExpDes.csv file
Index

* MSFinder
  annotate, 5
  annotation.summary, 7
  assign.z, 8
  change.annotation, 9
  defineExperiment, 16
  export.msfinder.formulas, 20
  findfeature, 24
  findmass, 25
  fooddb2msfinder, 26
  getSmilesInchi, 29
  import.msfinder.formulas, 32
  import.msfinder.mssearch, 33
  import.msfinder.structures, 34
  manual.annotation.template, 37
  ramclustR, 42
  rc.calibrate.ri, 45
  rc.cmpd.filter.blanks, 46
  rc.cmpd.filter.cv, 48
  rc.cmpd.get.classyfire, 49
  rc.cmpd.get.smiles.inchi, 52
  rc.cmpd.replace.na, 53
  rc.expand.sample.names, 54
  rc.feature.filter.blanks, 56
  rc.feature.filter.cv, 57
  rc.feature.normalize.qc, 59
  rc.feature.normalize.tic, 60
  rc.feature.replace.na, 61
  rc.get.csv.data, 62
  rc.get.df.data, 64
  rc.get.xcms.data, 66
  rc.qc, 67
  rc.ramclustR, 69
  rc.remove.qc, 71
  rc.restore.qc.samples, 72
  RCQC, 73

* RAMClustR
  annotate, 5
  annotation.summary, 7
  assign.z, 8
  change.annotation, 9
  cmpd.summary, 13
  defineExperiment, 16
  do.findmain, 18
  export.msfinder.formulas, 20
  exportDataset, 21
  findfeature, 24
  findmass, 25
  fooddb2msfinder, 26
  getSmilesInchi, 29
  import.msfinder.formulas, 32
  import.msfinder.mssearch, 33
  import.msfinder.structures, 34
  import.sirius, 35
  impRamSearch, 36
  manual.annotation.template, 37
  mergeRCobjects, 38
  ramclustR, 42
  rc.calibrate.ri, 45
  rc.cmpd.filter.blanks, 46
  rc.cmpd.filter.cv, 48
  rc.cmpd.get.classyfire, 49
  rc.cmpd.get.smiles.inchi, 52
  rc.cmpd.replace.na, 53
  rc.expand.sample.names, 54
  rc.feature.filter.blanks, 56
  rc.feature.filter.cv, 57
  rc.feature.normalize.qc, 59
  rc.feature.normalize.tic, 60
  rc.feature.replace.na, 61
  rc.get.csv.data, 62
  rc.get.df.data, 64
  rc.get.xcms.data, 66
  rc.qc, 67
  rc.ramclustR, 69
  rc.remove.qc, 71

78
getSmilesInchi, 29
rc.cmpd.get.smiles.inchi, 52

* interpretMSSpectrum
  do.findmain, 18
  rc.export.msp.rc, 55
  write.msp, 76

* mass spectrometry
  annotate, 5
  annotation.summary, 7
  assign.z, 8
  change.annotation, 9
  cmpd.summary, 13
  defineExperiment, 16
  do.findmain, 18
  export.msfinder.formulas, 20
  findfeature, 24
  findmass, 25
  fooddb2msfinder, 26
  getSmilesInchi, 29
  import.msfinder.formulas, 32
  import.msfinder.mssearch, 33
  import.msfinder.structures, 34
  import.sirius, 35
  impRamSearch, 36
  manual.annotation.template, 37
  mergeRCobjects, 38
  ramclustR, 42
  rc.calibrate.ri, 45
  rc.cmpd.filter.blanks, 46
  rc.cmpd.filter.cv, 48
  rc.cmpd.get.classyfire, 49
  rc.cmpd.get.smiles.inchi, 52
  rc.cmpd.replace.na, 53
  rc.expand.sample.names, 54
  rc.export.msp.rc, 55
  rc.feature.filter.blanks, 56
  rc.feature.filter.cv, 57
  rc.feature.normalize.qc, 59
  rc.feature.normalize.tic, 60
  rc.feature.replace.na, 61
  rc.get.csv.data, 62
  rc.get.df.data, 64
  rc.get.xcms.data, 66
  rc.qc, 67
  rc.ramclustr, 69
  rc.remove.qc, 71
  rc.restore.qc.samples, 72
  RCQC, 73
  write.methods, 75
  write.msp, 76

* metabolomics
  annotate, 5
  annotation.summary, 7
  assign.z, 8
  change.annotation, 9
  cmpd.summary, 13
  defineExperiment, 16
  do.findmain, 18
  export.msfinder.formulas, 20
  findfeature, 24
  findmass, 25
  fooddb2msfinder, 26
  getSmilesInchi, 29
  import.msfinder.formulas, 32
  import.msfinder.mssearch, 33
  import.msfinder.structures, 34
  import.sirius, 35
  impRamSearch, 36
  manual.annotation.template, 37
  mergeRCobjects, 38
  ramclustR, 42
  rc.calibrate.ri, 45
  rc.cmpd.filter.blanks, 46
  rc.cmpd.filter.cv, 48
  rc.cmpd.get.classyfire, 49
  rc.cmpd.get.smiles.inchi, 52
  rc.cmpd.replace.na, 53
  rc.expand.sample.names, 54
  rc.export.msp.rc, 55
  rc.feature.filter.blanks, 56
  rc.feature.filter.cv, 57
  rc.feature.normalize.qc, 59
  rc.feature.normalize.tic, 60
  rc.feature.replace.na, 61
  rc.get.csv.data, 62
  rc.get.df.data, 64
  rc.get.xcms.data, 66
  rc.qc, 67
  rc.ramclustr, 69
  rc.remove.qc, 71
  rc.restore.qc.samples, 72
  RCQC, 73
  write.methods, 75
  write.msp, 76

* ramclustR
  do.findmain, 18
export.msfinder.formulas, 20
exportDataset, 21
findfeature, 24
findmass, 25
fooddb2msfinder, 26
getSmilesInchi, 29
import.msfinder.formulas, 32
import.msfinder.mssearch, 33
import.msfinder.structures, 34
import.sirius, 35
impRamSearch, 36
manual.annotation.template, 37
mergeRCoobjects, 38
ramclustR, 42
rc.calibrate.ri, 45
rc.cmpd.filter.blanks, 46
rc.cmpd.filter.cv, 48
rc.cmpd.get.classyfire, 49
rc.cmpd.get.smiles.inchi, 52
rc.cmpd.replace.na, 53
rc.expand.sample.names, 54
rc.export.msp.rc, 55
rc.feature.filter.blanks, 56
rc.feature.filter.cv, 57
rc.feature.normalize.qc, 59
rc.feature.normalize.tic, 60
rc.feature.replace.na, 61
rc.get.csv.data, 62
rc.get.df.data, 64
rc.get.xcms.data, 66
rc.qc, 67
rc.ramclustr, 69
rc.remove.qc, 71
rc.restore.qc.samples, 72
RCQC, 73
write.methods, 75
write.msp, 76

* ramclustR
  annotate, 5
  annotation.summary, 7
  assign.z, 8
  change.annotation, 9
  cmpd.summary, 13
  defineExperiment, 16

* smiles
  getSmilesInchi, 29
  rc.cmpd.get.smiles.inchi, 52

* xcms

annotate, 5
annotation.summary, 7
assign.z, 8
change.annotation, 9
defineExperiment, 16
do.findmain, 18
export.msfinder.formulas, 20
findfeature, 24
findmass, 25
fooddb2msfinder, 26
import.msfinder.formulas, 32
import.msfinder.mssearch, 33
import.msfinder.structures, 34
import.sirius, 35
impRamSearch, 36
manual.annotation.template, 37
mergeRCoobjects, 38
ramclustR, 42
rc.calibrate.ri, 45
rc.cmpd.filter.blanks, 46
rc.cmpd.filter.cv, 48
rc.cmpd.get.classyfire, 49
rc.cmpd.replace.na, 53
rc.expand.sample.names, 54
rc.export.msp.rc, 55
rc.feature.filter.blanks, 56
rc.feature.filter.cv, 57
rc.feature.normalize.qc, 59
rc.feature.normalize.tic, 60
rc.feature.replace.na, 61
rc.get.csv.data, 62
rc.get.df.data, 64
rc.get.xcms.data, 66
rc.qc, 67
rc.ramclustr, 69
rc.remove.qc, 71
rc.restore.qc.samples, 72
RCQC, 73
write.methods, 75
write.msp, 76

adap.to.rc, 3
add_params, 5
annotate, 5
annotation.summary, 7
assign.z, 8
change.annotation, 9
check_arguments_filter.blanks, 11
check_arguments_filter.cv, 12
check_arguments_replace.na, 12
checks, 11
check modulation, 13
compute do sets, 14
compute spec abundance ave, 14
compute wt mean, 15
create ramclust obj, 15
define samples, 17
define experiment, 16
do findmain, 18
export msfinder formulas, 20
export dataset, 21
filter blanks, 22
filter good features, 23
filter signal, 23
find good features, 26
find feature, 24
find mass, 25
food db2msfinder, 26
get taxon cids, 27
gt. exp des, 30
gt. instrument platform, 31
data get, 28
get smiles inchi, 29
import adap kdb, 31
import msfinder formulas, 32
import msfinder ms search, 33
import msfinder structures, 34
import siri us, 35
imp ram search, 36
manual annotation template, 37
mean signal intensities, 38
merge R objects, 38
normalized data batch qc, 40
normalized data tic, 41
order datasets, 41
ramclust R, 42
rc calibrate ri, 45
rc cmpd filter blanks, 46
rc cmpd filter cv, 48
rc cmpd get classy fire, 49
rc cmpd get pubchem, 50
rc cmpd get smiles inchi, 52
rc cmpd replace na, 53
rc expand sample names, 54
rc export msp rc, 55
rc feature filter blanks, 56
rc feature filter cv, 57
rc feature normalize batch qc, 58
rc feature normalize q c, 59
rc feature normalize quantile, 60
rc feature normalize tic, 60
rc feature replace na, 61
rc get csv data, 62
rc get df data, 64
rc get xcms data, 66
rc qc, 67
rc ramclustr, 69
rc remove qc, 71
rc restore qc samples, 72
RCQC, 73
remove blanks, 74
replace na, 74
write methods, 75
write msp, 76
write csv, 77