Package 'msPurity'

October 15, 2025

Type Package

Title Automated Evaluation of Precursor Ion Purity for Mass Spectrometry Based Fragmentation in Metabolomics

Version 1.35.0

Date 2024-05-09

URL https://github.com/computational-metabolomics/msPurity/

Description msPurity R package was developed to:
1) Assess the spectral quality of fragmentation spectra by evaluating the ``precursor ion purity".
2) Process fragmentation spectra.
3) Perform spectral matching.
What is precursor ion purity? -What we call ``Precursor ion purity" is a measure of the contribution of a selected precursor peak in an isolation window used for fragmentation. The simple calculation involves dividing the intensity of the selected precursor peak by the total intensity of the isolation window. When assessing MS/MS spectra this calculation is done before and after the MS/MS scan of interest and the purity is interpolated at the recorded time of the MS/MS acquisition. Additionally, isotopic peaks can be removed, low abundance peaks are removed that are thought to have limited contribution to the resulting MS/MS spectra and the isolation efficiency of the mass spectrometer can be used to normalise the intensities used for the calculation.

Encoding UTF-8

License GPL-3 + file LICENSE

LazyData TRUE

BugReports https://github.com/computational-metabolomics/msPurity/issues/new

Depends Rcpp

Imports plyr, dplyr, dbplyr, magrittr, foreach, parallel, doSNOW, stringr, mzR, reshape2, fastcluster, ggplot2, DBI, RSQLite

Suggests MSnbase, testthat, xcms, BiocStyle, knitr, rmarkdown, msPurityData, CAMERA, RPostgres, RMySQL

VignetteBuilder knitr

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

biocViews MassSpectrometry, Metabolomics, Software

Collate 'all-generics.R' 'averaging.R' 'combineAnnotations.R'
'create-database.R' 'createDatabase.R' 'flag-filter-remove.R'
'iw-norm.R' 'matching-algs.R' 'meta_extract.R' 'msPurity.R'
'pcalc.R' 'purityA-0-class.R' 'purityA-av-spectra.R'
'purityA-constructor.R' 'purityA-create-msp.R'
'purityA-filter-frag-spectra.R' 'purityA-frag4feature.R'
'purityA-validate.R' 'purityD-class.R' 'purityD-constructor.R'
'purityD-av-spectra.R' 'purityD-dims-purity.R'
'purityD-fileList.R' 'purityD-filterp.R' 'purityD-subtract.R'
'purityD-writeOut.R' 'purityX-class.R' 'purityX-constructor.R'
'spectral-matching.R' 'spectralMatching.R' 'splinepurity.R'

git_url https://git.bioconductor.org/packages/msPurity

git_branch devel

git_last_commit d018d66

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-10-14

Author Thomas N. Lawson [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5915-7980>),
Ralf Weber [ctb],
Martin Jones [ctb],
Julien Saint-Vanne [ctb],
Andris Jankevics [ctb],
Mark Viant [ths],
Warwick Dunn [ths]

Maintainer Thomas N. Lawson <thomas.nigel.lawson@gmail.com>

Contents

`assessPuritySingle` *Assess the purity of a single LC-MS/MS or DI-MS/MS file*

Description

Given a filepath to an mzML file the precursor purity for any MS/MS scans will be outputed into a dataframe

Usage

```
assessPuritySingle(
  filepth,
  fileid = NA,
  mostIntense = FALSE,
  nearest = TRUE,
  offsets = NA,
  cores = 1,
  plotP = FALSE,
  plotdir = NULL,
  interpol = "linear",
  iwNorm = FALSE,
  iwNormFun = NULL,
  ilim = 0,
  mzRback = "pwiz",
  isotopes = TRUE,
  im = NULL,
  ppmInterp = 7
)
```

Arguments

filepth	character; mzML file path for MS/MS spectra
fileid	numeric; adds a fileid column (primarily for internal use for msPurity)
mostIntense	boolean; True if the most intense peak is used for calculation. False if the centered peak is used
nearest	boolean; True if the peak selected is as the nearest MS1 scan. If False then the preceding scan is used
offsets	vector; Overide the isolation offsets found in the mzML filee.g. c(0.5, 0.5)
cores	numeric; Number of cores to use
plotP	boolean; If TRUE a plot of the purity is to be saved
plotdir	vector; If plotP is TRUE plots will be saved to this directory
interpol	character; Type of interolation to be performed "linear", "spline" or "none"
iwNorm	boolean; If TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function
iwNormFun	function; A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric; All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
mzRback	character; Backend to use for mzR parsing
isotopes	boolean; TRUE if isotopes are to be removed
im	matrix; Isotope matrix, default removes C13 isotopes (single, double and triple bonds)
ppmInterp	numeric; Set the ppm tolerance for the precursor ion purity interpolation. i.e. the ppm tolerence between the precursor ion found in the neighbouring scans.

Value

a dataframe of the purity score of the ms/ms spectra

See Also

purityA

Examples

```
filepth <- system.file("extdata", "lcms", "mzML", "LCMSMS_1.mzML", package="msPurityData")

puritydf <- assessPuritySingle(filepth)
```

`averageAllFragSpectra,purityA-method`

Using a purityA object, average and filter MS/MS spectra for each XCMS feature within and across MS data files (ignoring intra and inter relationships)

Description

General

Average and filter fragmentation spectra for each XCMS feature within and across MS data files (ignoring intra and inter relationships).

The averaging is performed using hierarchical clustering of the m/z values of each peaks, where m/z values within a set ppm tolerance will be clustered. The clustered peaks are then averaged (or summed).

The fragmentation can be filtered on the averaged spectra (with the arguments snr, rsd, minfrac, ra)

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> **averageAllFragSpectra** -> createDatabase -> spectralMatching -> (sqlite spectral database)

Usage

```
## S4 method for signature 'purityA'
averageAllFragSpectra(
  pa,
  minfrac = 0.5,
  minnum = 1,
  ppm = 5,
  snr = 0,
  ra = 0,
  av = "median",
  sumi = TRUE,
  rmp = FALSE,
  cores = 1
)
```

Arguments

pa	object; purityA object

minfrac	numeric;minimum ratio of the peak fraction (peak count / total peaks) across all (ignoring intra and inter relationships)
minnum	numeric; minimum number of times peak is present across all fragmentation spectra (ignoring intra and inter relationships)
ppm	numeric; ppm threshold to average across all scans (ignoring intra and inter relationships)
snr	numeric; minimum signal-to-noise of the peak across all (ignoring intra and inter relationships)
ra	numeric; minimum relative abundance of the peak fraction across all (ignoring intra and inter relationships)
av	character; type of averaging to use (median or mean)
sumi	boolean; TRUE if the intensity for each peak is summed across averaged spectra
rmp	boolean; TRUE if peaks are to be removed that do not meet the threshold criteria. Otherwise they will just be flagged
cores	numeric; Number of cores for multiprocessing

Value

Returns a purityA object (pa) with the following slots now with data

- pa@av_spectra: the average spectra is recorded here stored as a list. E.g. pa@av_spectra1av_all would give the average spectra for grouped feature 1.
- pa@av_all_params: The parameters used are recorded here

Each spectra in the av_spectra list contains the following columns:

- cl: id of clustered (averaged) peak
- mz: average m/z
- i: average intensity
- snr: average signal to noise ratio
- rsd: relative standard deviation
- count: number of clustered peaks
- total: total number of potential scans to be used for averaging
- inPurity: average precursor ion purity
- ra: average relative abundance
- frac: the fraction of clustered peaks (e.g. the count/total)
- snr_pass_flag: TRUE if snr threshold criteria met
- minfrac_pass_flag: TRUE if minfrac threshold criteria
- ra_pass_flag: TRUE if ra threshold criteria met
- pass_flag: TRUE if all threshold criteria met

Examples

```
#====== XCMS ==================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ===============================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xcmsObj)
#pa <- filterFragSpectra(pa)
#pa <- averageAllFragSpectra(pa)

## Run from previously generated data
pa <- readRDS(system.file("extdata", "tests", "purityA",
                          "3_filterFragSpectra_pa.rds", package="msPurity"))
pa <- averageAllFragSpectra(pa)
```

---

averageInterFragSpectra,purityA-method

Using a purityA object, average and filter fragmentation spectra for each XCMS feature across multiple MS data files

---

Description

General

Average and filter fragmentation spectra for each XCMS feature across MS data files. This can only be run after averageIntraFragSpectra has been used.

The averaging is performed using hierarchical clustering of the m/z values of each peaks, where m/z values within a set ppm tolerance will be clustered. The clustered peaks are then averaged (or summed).

The fragmentation can be filtered on the averaged spectra (with the arguments snr, rsd, minfrac and ra)

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing

  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> averageIntraFragSpectra -> **averageInterFragSpectra** -> createDatabase -> spectralMatching -> (sqlite spectral database)

Usage

```
## S4 method for signature 'purityA'
averageInterFragSpectra(
  pa,
  minfrac = 0.5,
  minnum = 1,
  ppm = 5,
  snr = 0,
  ra = 0,
  av = "median",
  sumi = TRUE,
  rmp = FALSE,
  cores = 1
)
```

Arguments

pa	object; purityA object
minfrac	numeric; minimum ratio of the peak fraction (peak count / total peaks) across files
minnum	numeric; minimum number of times peak is present across fragmentation spectra across files
ppm	numeric; ppm threshold to average across files
snr	numeric; minimum signal-to-noise of the peak across files
ra	numeric; minimum relative abundance of the peak across files
av	character; type of averaging to use (median or mean)
sumi	boolean; TRUE if the intensity for each peak is summed across averaged spectra
rmp	boolean; TRUE if peaks are to be removed that do not meet the threshold criteria. Otherwise they will just be flagged
cores	numeric; Number of cores for multiprocessing

Value

Returns a purityA object (pa) with the following slots now with data

- pa@av_spectra: the average spectra is recorded here stored as a list. e.g. "pa@av_spectra1av_inter" would give the average spectra for grouped feature 1
- pa@av_intra_params: The parameters used are recorded here

Each spectra in the av_spectra list contains the following columns: *

- cl: id of clustered (averaged) peak

- mz: average m/z
- i: average intensity
- snr: average signal to noise ratio
- rsd: relative standard deviation
- count: number of clustered peaks
- total: total number of potential scans to be used for averaging
- inPurity: average precursor ion purity
- ra: average relative abundance
- frac: the fraction of clustered peaks (e.g. the count/total)
- snr_pass_flag: TRUE if snr threshold criteria met
- minfrac_pass_flag: TRUE if minfrac threshold criteria
- ra_pass_flag: TRUE if ra threshold criteria met
- pass_flag: TRUE if all threshold criteria met

Examples

```
#====== XCMS ===================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ================================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xcmsObj)
#pa <- averageIntraFragSpectra(pa)
#pa <- averageInterFragSpectra(pa)

## Run from previously generated data
pa <- readRDS(system.file("extdata", "tests", "purityA",
                          "4_averageIntraFragSpectra_no_filter_pa.rds",
                          package="msPurity"))
pa <- averageInterFragSpectra(pa)
```

`averageIntraFragSpectra,purityA-method`
Using a purityA object, average and filter fragmentation spectra for each XCMS feature within a MS data file

Description

General

Average and filter fragmentation spectra for each XCMS feature within a MS data file.

The averaging is performed using hierarchical clustering of the m/z values of each peaks, where m/z values within a set ppm tolerance will be clustered. The clustered peaks are then averaged (or summed).

The fragmentation can be filtered on the averaged spectra (with the arguments snr, rsd, minfrac and ra)

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.xcmsSet -> xcms.merge -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- XCMS processing (version >= 3)
  - (mzML files) -> MSnBase.readMSdata -> xcms.findChromPeaks -> xcms.groupChromPeaks-> xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> **averageIntraFragSpectra** -> averageIntraFragSpectra -> createDatabase -> spectralMatching -> (sqlite spectral database)

Usage

```
## S4 method for signature 'purityA'
averageIntraFragSpectra(
  pa,
  minfrac = 0.5,
  minnum = 1,
  ppm = 5,
  snr = 0,
  ra = 0,
  av = "median",
  sumi = TRUE,
  rmp = FALSE,
  cores = 1
)
```

Arguments

pa	object; purityA object
minfrac	numeric; minimum ratio of the peak fraction (peak count / total peaks) within each file
minnum	numeric; minimum number of times peak is present across fragmentation spectra within each file
ppm	numeric; ppm threshold to average within each file
snr	numeric; minimum signal-to-noise of the peak within each file
ra	numeric; minimum relative abundance of the peak within each file
av	character; type of averaging to use (median or mean)
sumi	boolean; TRUE if the intensity for each peak is summed across averaged spectra
rmp	boolean; TRUE if peaks are to be removed that do not meet the threshold criteria. Otherwise they will just be flagged
cores	numeric; Number of cores for multiprocessing

Value

Returns a purityA object (pa) with the following slots now with data

- pa@av_spectra: the average spectra is recorded here stored as a list. e.g. "pa@av_spectra1av_intra$1" would give the average spectra for grouped feature 1 and for file 1.
- pa@av_intra_params: The parameters used are recorded here

Each spectra in the av_spectra list contains the following columns:

- cl: id of clustered (averaged) peak
- mz: average m/z
- i: average intensity
- snr: average signal to noise ratio
- rsd: relative standard deviation
- count: number of clustered peaks
- total: total number of potential scans to be used for averaging
- inPurity: average precursor ion purity
- ra: average relative abundance
- frac: the fraction of clustered peaks (e.g. the count/total)
- snr_pass_flag: TRUE if snr threshold criteria met
- minfrac_pass_flag: TRUE if minfrac threshold criteria
- ra_pass_flag: TRUE if ra threshold criteria met
- pass_flag: TRUE if all threshold criteria met

Examples

```
#====== XCMS ================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ============================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xcmsObj)
#pa <- averageIntraFragSpectra(pa)

# Run from previously generated data (where class is 'XCMSnExp'):
pa <- readRDS(system.file("extdata", "tests", "purityA",
              "2_frag4feature_pa.rds", package="msPurity"))
pa <- averageIntraFragSpectra(pa)
```

---

averageSpectra,purityD-method

Using purityD object, calculates to average mz, intensity and signal-to-noise of multiple scans from multiple MS datafiles (mzML or .csv)

Description

Uses a purityD object with references to multiple MS files. For each file: Averages multiple scans together, see averageSpectraSingle for more information

Usage

```
## S4 method for signature 'purityD'
averageSpectra(
  Object,
  rtscn = "all",
  scanRange = NA,
  timeRange = NA,
  clustType = "hc",
  ppm = 1.5,
  snthr = 3,
  av = "median",
  missingV = "zero",
  minfrac = 0.6667,
  normTIC = FALSE,
```

```
  snMeth = "median"
)
```

Arguments

Object	object; purityD object
rtscn	character; Whether it is scans or retention time to be filtered. Use "all" if all scans to be used. ['rt', 'scns', 'all']
scanRange	vector; Scan range (if rtscn='scns') e.g. c(40, 69)
timeRange	vector; Time range (if rtscn='rt) e.g. c(10.3, 400.8) (only if using mzML file)
clustType	character; Type of clustering used either Hierarchical or just simple 1D grouping ['hc', 'simple']
ppm	numeric; The ppm error to cluster mz together
snthr	numeric; Signal to noise ratio threshold
av	character; What type of averaging to do between peaks
missingV	character; What to do with missing values (zero or ignore)
minfrac	numeric; Min fraction of scans with a grouped peak to be an accepted averaged peak
normTIC	boolean; If TRUE then RSD calculation will use the normalised intensity (intensity divided by TIC) if FALSE will use standard intensity
snMeth	character; Type of snMethod to use ['mean', 'median', 'precalc']. Precalc only applicable when using the csvFile parameter as TRUE

Value

purityD object with averaged spectra

See Also

averageSpectraSingle

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
ppDIMS <- averageSpectra(ppDIMS)
```

---

averageSpectraSingle *Calculates to average mz, intensity and signal-to-noise of multiple scans from 1 MS datafile (mzML or .csv)*

Description

Averages multiple scans of mass spectrometry data together. Each scan consisting of a minimum of intensity and mz values.

Works for either mzML or a .csv file consisting of mz, i, scanid, (optional: noise, backgroun, snr)

Signal-to-noise (SNR) can be calculated a number of ways. Default is to calculate the SN for every scan as the "Intensity of peak / the median intensity of the scan".

Alternatively if using a .csv file as input (and assigning the csvFile parameter to TRUE), a precalculated SNR can be one of the columns. The precalculated SNR can then be chosen by using the option 'precalc' for the parameter snMethod

The function will work for both LC-MS or DI-MS datasets.

Usage

```
averageSpectraSingle(
  filePth,
  rtscn = "all",
  scanRange = NA,
  timeRange = NA,
  clustType = "hc",
  ppm = 1.5,
  snthr = 3,
  cores = 1,
  av = "median",
  missingV = "ignore",
  minfrac = 0.6667,
  snMeth = "median",
  csvFile = FALSE,
  normTIC = FALSE,
  mzRback = "pwiz",
  MSFileReader = FALSE
)
```

Arguments

filePth	character; Path of the file to be processed
rtscn	character; Whether it is scans or retention time to be filtered. Use "all" if all scans to be used. ['rt', 'scns', 'all']
scanRange	vector; Scan range (if rtscn='scns') e.g. c(40, 69)
timeRange	vector; Time range (if rtscn='rt) e.g. c(10.3, 400.8) (only if using mzML file)
clustType	character; Type of clustering used either Hierarchical or just simple 1D grouping ['hc', 'simple']
ppm	numeric; The ppm error to cluster mz together
snthr	numeric; Signal to noise ratio threshold
cores	numeric; Number of cores used to perform Hierarchical clustering WARNING: memory intensive, default 2
av	character; What type of averaging to do between peaks
missingV	character; What to do with missing values (zero or ignore)

minfrac	numeric; Min fraction of scans with a grouped peak to be an accepted averaged peak
snMeth	character; Type of snMethod to use ['mean', 'median', 'precalc']. Precalc only applicable when using the csvFile parameter as TRUE
csvFile	boolean; A csv file can be used as input. Useful for thermo files where the MSFileReader API can extract peaklist. This can consist of an .csv file with the following columns c('mz', 'i', 'scanid', 'snr')
normTIC	boolean; If TRUE then RSD calculation will use the normalised intensity (intensity divided by TIC) if FALSE will use standard intensity
mzRback	character; Backend to use for mzR parsing
MSFileReader	boolean; Deprecapted. Use csvFile parameter

Value

dataframe of the median mz, intensity, signal-to-noise ratio.

Examples

```
mzmlPth <- system.file("extdata", "dims", "mzML", "B02_Daph_TEST_pos.mzML", package="msPurityData")
avP <- averageSpectraSingle(mzmlPth)
```

---

combineAnnotations *Combine Annotations*

Description

Combine the annotation results from msPurity spectral matching, MetFrag, Sirius CSI:FingerID, probmetab and any generic MS1 lookup software (e.g. results from the BEAMS software)

The annotation results are then aligned by inchikey and XCMS grouped feature.

The tool has to be run with a local compound database (available on request - contact t.n.lawson@bham.ac.uk)

Usage

```
combineAnnotations(
  sm_resultPth,
  compoundDbPth,
  metfrag_resultPth = NA,
  sirius_csi_resultPth = NA,
  probmetab_resultPth = NA,
  ms1_lookup_resultPth = NA,
  ms1_lookup_dbSource = "hmdb",
  ms1_lookup_checkAdducts = FALSE,
  ms1_lookup_keepAdducts = c("[M+H]+", "[M-H]-"),
  weights = list(sm = 0.3, metfrag = 0.2, sirius_csifingerid = 0.2, probmetab = 0,
    ms1_lookup = 0.05, biosim = 0.25),
  compoundDbType = "sqlite",
  compoundDbName = NA,
  compoundDbHost = NA,
  compoundDbPort = NA,
```

```
  compoundDbUser = NA,
  compoundDbPass = NA,
  outPth = NA,
  summaryOutput = TRUE
)
```

Arguments

sm_resultPth	character; Path to the msPurity SQLite database used for spectral matching
compoundDbPth	character; Path to local compound database with pubchem, hmdb, KEGG and metab_compound summary table (full database available on request - contact t.n.lawson@bham.ac.uk). This is only applicable if using "compoundDbType sqlite")
metfrag_resultPth	character; Path to the tsv table of metfrag results
sirius_csi_resultPth	character; Path to the tsv table of Sirius CSI:Finger ID results
probmetab_resultPth	character; Path to the tsv table of Probmetab results
ms1_lookup_resultPth	character; Path to generic tsv table of MS1 lookup results
ms1_lookup_dbSource	character; Source of the compound database used for ms1_lookup (currently only supports HMDB, KEGG or PubChem)
ms1_lookup_checkAdducts	boolean; Check if adducts match to those found in CAMERA (requires the database to have been created with CAMERA object)
ms1_lookup_keepAdducts	vecotr; Keep only adducts found from the MS1 lookup that are found in this vector
weights	list;
compoundDbType	character; Database type for compound database can be either (sqlite, postgres or mysql)
compoundDbName	character; Database name (only applicable for postgres and mysql)
compoundDbHost	character; Database host (only applicable for postgres and mysql)
compoundDbPort	character; Database port (only applicable for postgres and mysql)
compoundDbUser	character; Database user (only applicable for postgres and mysql)
compoundDbPass	character; Database pass (only applicable for postgres and mysql) - Note this is not secure!
outPth	character;
summaryOutput	boolean; If a summary dataframe is to be created

Value

purityA object with slots for fragmentation-XCMS links

Examples

```
metfrag_resultPth <- system.file("extdata", "tests", "external_annotations",
                                 "metfrag.tsv", package="msPurity")
# run the standard spectral matching workflow to get the sm_resultPth
sm_resultPth <- system.file("extdata","tests", "sm",
                        "spectralMatching_result.sqlite", package="msPurity")
compoundDbPth <- system.file("extdata", "tests", "db",
                          "metab_compound_subset.sqlite", package="msPurity")
combined <- combineAnnotations(sm_resultPth,
                            metfrag_resultPth,
                            outPth=file.path(tempdir(), 'combined.sqlite'),
                            compoundDbPth=compoundDbPth)
```

`createDatabase` *Create database*

Description

General

Create an SQLite database of an LC-MS(/MS) experiment (replaces the create_database function).

Schema details can be found here.

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> averageAllFragSpectra -> **createDatabase** -> spectralMatching -> (sqlite spectral database)

Usage

```
createDatabase(
  pa,
  xcmsObj,
  xsa = NULL,
  outDir = ".",
  grpPeaklist = NA,
  dbName = NA,
  metadata = NA,
  xset = NA
)
```

Arguments

pa	purityA object; Needs to be the same used for frag4feature function
xcmsObj	xcms object of class XCMSnExp or xcmsSet; Needs to be the same used for frag4feature function (this will be ignored when using xsa parameter)
xsa	CAMERA object (optional); if CAMERA object is used, we ignore the xset parameter input and obtain all information from the xset object nested with the CAMERA xsa object. Adduct and isotope information will be included into the database when using this parameter. The underlying xset object must be the one used for the frag4feature function
outDir	character; Out directory for the SQLite result database
grpPeaklist	dataframe (optional); Can use any peak dataframe. Still needs to be derived from the xset object though
dbName	character (optional); Name of the result database
metadata	list; A list of metadata to add to the s_peak_meta table
xset	xcms object of class XCMSnExp or xcmsSet; (Deprecated - if provided, will replace variable 'obj')

Value

path to SQLite database and database name

Examples

```
library(xcms)
library(MSnbase)
library(magrittr)
#====== XCMS ================================
## Read in MS data
msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ============================
pa  <- purityA(msmsPths)
pa <- frag4feature(pa = pa, xcmsObj = xcmsObj)
pa <- filterFragSpectra(pa, allfrag=TRUE)
pa <- averageAllFragSpectra(pa)
dbPth <- createDatabase(pa, xcmsObj, metadata=list('polarity'='positive','instrument'='Q-Exactive'))

td <- tempdir()
db_pth = createDatabase(pa = pa, xcmsObj = xcmsObj, outDir = td)
```

`createMSP,purityA-method`

Using a purityA object, create an MSP file of fragmentation spectra

Description

General

Create an MSP file for all the fragmentation spectra that has been linked to an XCMS feature via frag4feature. Can export all the associated scans individually or the averaged fragmentation spectra can be exported.

Additional metadata can be included in a dataframe (each column will be added to metadata of the MSP spectra). The dataframe must contain the column "grpid" corresponding to the XCMS grouped feature.

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> **purityA** -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> averageIntraFragSpectra -> averageIntraFragSpectra -> **createMSP** -> (MSP file)

Usage

```
## S4 method for signature 'purityA'
createMSP(
  pa,
  msp_file_pth = NULL,
  metadata = NULL,
  metadata_cols = NULL,
  xcms_groupids = NULL,
  method = "all",
  adduct_split = TRUE,
  filter = TRUE,
  msp_schema = "massbank",
  intensity_ra = "intensity_ra",
  include_adducts = ""
)
```

Arguments

`pa`	object; purityA object
`msp_file_pth`	character; Name of the output msp file, if NULL the file "frag_spectra_time stamp.msp" will be created in the current directory

metadata	data.frame; Data frame with additional coumpound infomation to include in msp output
metadata_cols	vector; Column names of meta data to incorporate into name
xcms_groupids	vector; XCMS group id's to extract ms/ms data for
method	character; "all" will export all matching ms/ms spectra to xcms features, "max" will use spectra with the highest inensity, "av_intra" will use the intra file averaged spectra (within file), "av_inter" will use the inter file (across file) averaged spectra, "av_all" will use the averaged spectra (ignoring inter and intra)
adduct_split	boolean; If either "adduct" or MS$FOCUSED_ION: PRECURSOR_TYPE column is in metadata then each adduct will have it's own MSP spectra. (Useful, if the MSP file will be used for further annotation)
filter	boolean; TRUE if filtered peaks are to be removed
msp_schema	character; Either MassBank (Europe) or MoNA style of MSP file format to be used ('massbank' or 'mona')
intensity_ra	character; Either 'intensity', 'ra' (relative abundance) or 'intensity_ra' (intensity and relative abundance) to be written to the MSP file
include_adducts	character; Additional adducts to include as a string seperated by white a space (e.g. [M+H]+ [M+Na]+)

Value

Returns a MSP file with the selected spectra and metadata

Examples

```
#====== XCMS ================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ============================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa = pa, xcmsObj = xcmsObj)
#pa <- filterFragSpectra(pa, allfrag=TRUE)
#pa <- averageAllFragSpectra(pa)
#createMSP(pa)

pa <- readRDS(system.file("extdata", "tests", "purityA",
                          "9_averageAllFragSpectra_with_filter_pa.rds",
                          package="msPurity"))
createMSP(pa)
```

create_database *Create database deprecated*

Description

Create and SQLite database of an LC-MS(/MS) experiment

msPurity::create_database is deprecated. Please use msPurity::createDatabase for future use

Usage

```
create_database(
  pa,
  xset,
  xsa = NULL,
  out_dir = ".",
  grp_peaklist = NA,
  db_name = NA
)
```

Arguments

pa	purityA object; Needs to be the same used for frag4feature function
xset	xcms object; Needs to be the same used for frag4feature function (this will be ignored when using xsa parameter)
xsa	CAMERA object [optional]; if CAMERA object is used, we ignore the xset parameter input and obtain all information from the xset object nested with the CAMERA xsa object. Adduct and isotope information will be included into the database when using this parameter. The underlying xset object must be the one used for the frag4feature function
out_dir	character; Out directory for the SQLite result database
grp_peaklist	dataframe [optional]; Can use any peak dataframe. Still needs to be derived from the xset object though
db_name	character [optional]; Name of the result database

Value

path to SQLite database and database name

Examples

```
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#            package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#xset <- xcms::xcmsSet(msmsPths)
#xset <- xcms::group(xset)

#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xset)
#pa <- averageAllFragSpectra(pa)
#db_pth <- create_database(pa, xset)
```

```
# Run from previously generated data
pa <- readRDS(system.file("extdata", "tests", "purityA",
                          "9_averageAllFragSpectra_with_filter_pa_OLD.rds",
                          package="msPurity"))
xset <- readRDS(system.file("extdata","tests", "xcms",
                            "msms_only_xset_OLD.rds", package="msPurity"))

# Need to ensure the filelists are matching
msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
                                   package="msPurityData"),
                                   full.names = TRUE, pattern = "MSMS")
pa@fileList[1] <- msmsPths[basename(msmsPths)=="LCMSMS_1.mzML"]
pa@fileList[2] <- msmsPths[basename(msmsPths)=="LCMSMS_2.mzML"]
xset@filepaths[1] <- msmsPths[basename(msmsPths)=="LCMSMS_1.mzML"]
xset@filepaths[2] <- msmsPths[basename(msmsPths)=="LCMSMS_2.mzML"]
db_pth <- create_database(pa, xset)
```

`dimsPredictPurity,purityD-method`

Using purityD object, assess anticipated purity from a DI-MS run

---

Description

Assess the precursor purity of anticpated MS/MS spectra. i.e. it 'predicts' the precursor purity of the DI-MS peaks for a future MS/MS run.

Usage

```
## S4 method for signature 'purityD'
dimsPredictPurity(
  Object,
  ppm = 1.5,
  minOffset = 0.5,
  maxOffset = 0.5,
  iwNorm = FALSE,
  iwNormFun = NULL,
  ilim = 0.05,
  sampleOnly = FALSE,
  isotopes = TRUE,
  im = NULL
)
```

Arguments

`Object`	object = purityD object
`ppm`	numeric = tolerance for target mz value in each scan
`minOffset`	numeric = isolation window minimum offset
`maxOffset`	numeric = isolation window maximum offset
`iwNorm`	boolean = if TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function

iwNormFun	function = A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric = All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
sampleOnly	boolean = if TRUE will only calculate purity for sample peaklists
isotopes	boolean = TRUE if isotopes are to be removed
im	matrix = Isotope matrix, default removes C13 isotopes (single, double and triple bonds)

Value

purityD object with predicted purity of peaks

purityD object

See Also

dimsPredictPuritySingle

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
ppDIMS <- averageSpectra(ppDIMS)
ppDIMS <- filterp(ppDIMS)
ppDIMS <- subtract(ppDIMS)
ppDIMS <- dimsPredictPurity(ppDIMS)
```

---

dimsPredictPuritySingle

Predict the precursor purity from a DI-MS dataset

Description

Given a an DI-MS dataset (either mzML or .csv file) calculate the predicted purity for a vector of mz values.

Calculated at a given offset e.g. for 0.5 +/- Da the minOffset would be 0.5 and the maxOffset of 0.5.

A ppm tolerance is used to find the target mz value in each scan.

Usage

```
dimsPredictPuritySingle(
  mztargets,
  filepth,
  minOffset = 0.5,
  maxOffset = 0.5,
  ppm = 2.5,
  mzML = TRUE,
  iwNorm = FALSE,
  iwNormFun = NULL,
```

```
  ilim = 0.05,
  mzRback = "pwiz",
  isotopes = TRUE,
  im = NULL,
  sim = FALSE
)
```

Arguments

mztargets	vector = mz targets to get predicted purity for
filepth	character = mzML file path or .csv file path
minOffset	numeric = isolation window minimum offset
maxOffset	numeric = isolation window maximum offset
ppm	numeric = tolerance for target mz value in each scan
mzML	boolean = Whether an mzML file is to be used or .csv file (TRUE == mzML)
iwNorm	boolean = if TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function
iwNormFun	function = A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric = All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
mzRback	character = backend to use for mzR parsing
isotopes	boolean = TRUE if isotopes are to be removed
im	matrix = Isotope matrix, default removes C13 isotopes (single, double and triple bonds)
sim	boolean = TRUE if file is from sim stitch experiment. Default FALSE

Value

a dataframe of the target mz values and the predicted purity score

Examples

```
mzmlPth <- system.file("extdata", "dims", "mzML", "B02_Daph_TEST_pos.mzML",
                       package="msPurityData")
predicted <- dimsPredictPuritySingle(c(173.0806, 216.1045), filepth=mzmlPth,
                             minOffset=0.5, maxOffset=0.5, ppm=5, mzML=TRUE)
```

---

filterFragSpectra,purityA-method

Filter fragmentation spectra associated with an XCMS feature

Description

General

Flag and filter features based on signal-to-noise ratio, relative abundance, intensity threshold and purity of the precursor ion.

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> **filterFragSpectra** -> averageAllFragSpectra -> createDatabase -> spectralMatching -> (sqlite spectral database)

Usage

```
## S4 method for signature 'purityA'
filterFragSpectra(
  pa,
  ilim = 0,
  plim = 0.8,
  ra = 0,
  snr = 3,
  rmp = FALSE,
  snmeth = "median",
  allfrag = FALSE
)
```

Arguments

pa	object; purityA object
ilim	numeric; min intensity of a peak
plim	numeric; min precursor ion purity of the associated precursor for fragmentation spectra scan
ra	numeric; minimum relative abundance of a peak
snr	numeric; minimum signal-to-noise of a peak within each file
rmp	boolean; TRUE if peaks are to be removed that do not meet the threshold criteria. Otherwise they will just be flagged.
snmeth	character; Method to calculate signal to noise ration (either median or mean)
allfrag	boolean; Whether to filter on all fragmentation spectra or just the fragmentation spectra grouped to XCMS feature

Value

Returns a purityA object with the pa@grped_msms spectra matrices are updated with the following columns

- snr: Signal to noise ratio (calculated at scan level)
- ra: Relative abundance (calculated at scan level)
- purity_pass_flag: Precursor ion purity flag (1 pass, 0 fail)
- intensity_pass_flag: Intensity flag (1 pass, 0 fail)
- snr_pass_flag: Signal-to-noise pass flag (1 pass, 0 fail)
- ra_pass_flag: Relative abundance pass flag (1 pass, 0 fail)
- pass_flag: Overall pass flag, all flags must pass for this to pass (1 pass, 0 fail)

Examples

```
#====== XCMS ================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ============================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xcmsObj)
#pa <- filterFragSpectra(pa)

## Run from previously generated data
pa <- readRDS(system.file("extdata", "tests", "purityA",
                          "2_frag4feature_pa.rds", package="msPurity"))
pa <- filterFragSpectra(pa)
```

---

filterp,purityD-method

Filter out peaks based on intensity and RSD criteria

Description

Uses a purityD object remove peaks from either (or both) samples and blanks that are either below an intensity threshold or greater than a Relative Standard Deviation (RSD) threshold

Usage

```
## S4 method for signature 'purityD'
filterp(Object, thr = 5000, rsd = 20, sampleOnly = TRUE)
```

Arguments

Object	object; purityD object
thr	numeric; intensity threshold
rsd	numeric; rsd threshold
sampleOnly	boolean; if only the sample (not blanks) should be filtered

Value

purityD object

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)

ppDIMS <- purityD(inDF, cores=1)
ppDIMS <- averageSpectra(ppDIMS)
ppDIMS <- filterp(ppDIMS, thr = 5000)
```

---

flag_remove *Flag and remove unwanted peaks*

Description

Filter, flag and remove unwanted peaks from xcms object (xcmsObj) of class XCMSnExp, xcmsSet or xsAnnotate. When the peaks are removed, the xcmsObj object can be regrouped (originally using xcms::group, now using xcms::groupChromPeaks). The function then checks if any blank peaks are still present and the process is repeated.

The output is a list object containing: 1) the updated xcms object, 2) the grouped peaklist and 3) the blank removed peaks

Usage

```
flag_remove(
  xcmsObj,
  pol = NA,
  rsd_i_blank = NA,
  minfrac_blank = 0.5,
  rsd_rt_blank = NA,
  ithres_blank = NA,
  s2b = 10,
  ref.class = "blank",
  egauss_thr = NA,
  rsd_i_sample = NA,
  minfrac_sample = 0.7,
```

```
  rsd_rt_sample = NA,
  ithres_sample = NA,
  minfrac_xcms = 0.7,
  mzwid = 0.017,
  bw = 5,
  out_dir = ".",
  temp_save = FALSE,
  remove_spectra_bool = TRUE,
  grp_rm_ids = NA,
  xset = NA
)
```

Arguments

xcmsObj	object; XCMSnExp, xcmsSet or xsAnnotate object
pol	str; polarity (just used for naming purpose for files being saved) [positive, negative, NA]
rsd_i_blank	numeric; RSD threshold for the blank
minfrac_blank	numeric; minimum fraction of files for features needed for the blank
rsd_rt_blank	numeric; RSD threshold for the RT of the blank
ithres_blank	numeric; Intensity threshold for the blank
s2b	numeric; fold change (sample/blank) needed for sample peak to be allowed. e.g. if s2b set to 10 and the recorded sample 'intensity' value was 100 and blank = 10. 1000/10 = 100 so sample has fold change higher than the threshold and the peak is not considered a blank
ref.class	str; A string representing the class that will be used for the blank.
egauss_thr	numeric; Threshold for filtering out non gaussian shaped peaks. Note this only works if the verbose option was set for XCMS;
rsd_i_sample	numeric; RSD threshold for the sample
minfrac_sample	numeric; minimum fraction of files for features needed for the sample
rsd_rt_sample	numeric; RSD threshold for the RT of the sample
ithres_sample	numeric; Intensity threshold for the sample
minfrac_xcms	numeric; minfrac for xcms grouping
mzwid	numeric; xcms grouping parameter (corresponds to variable 'binSize' in XCMS3)
bw	numeric; xcms grouping parameter
out_dir	str; out directory
temp_save	boolean; Assign True if files for each step saved (for testing purpsoses)
remove_spectra_bool	bool; TRUE if flagged spectra is to be removed
grp_rm_ids	vector; vector of grouped_xcms peaks to remove (corresponds to the row from xcms::group output)
xset	object, DEPRECATED; xcmsSet object

Value

list(xset, grp_peaklist, removed_peaks)

Examples

```
library(xcms)
library(MSnbase)
library(magrittr)
#read in files and data
msPths <-list.files(system.file("extdata", "lcms", "mzML", package="msPurityData"), full.names = TRUE)
ms_data = readMSData(msPths, mode = 'onDisk', msLevel. = 1)

#subset the data to focus on retention times 30-90 seconds and m/z values between 100 and 200 m/z.
rtr = c(30, 90)
mzr = c(100, 200)
ms_data = ms_data %>%  filterRt(rt = rtr) %>%  filterMz(mz = mzr)

##### perform feature detection in individual files
cwp <- CentWaveParam(snthresh = 3, noise = 100, ppm = 10, peakwidth = c(3, 30))
xcmsObj <- findChromPeaks(ms_data, param = cwp)
xcmsObj@phenoData@data$class = c('blank', 'blank', 'sample', 'sample')
xcmsObj@phenoData@varMetadata = data.frame('labelDescription' = 'sampleNames', 'class')
pdp <- PeakDensityParam(sampleGroups = xcmsObj@phenoData@data$class, minFraction = 0, bw = 5, binSize = 0.017)
xcmsObj <- groupChromPeaks(xcmsObj, param = pdp)

#### flag, filter and remove peaks, returning an updated xcmsObj (XCMSnExp or xcmsSet class), grouped_peaklist (
fr <- flag_remove(xcmsObj)

##### load from existing data
xcmsObj = readRDS(system.file("extdata", "tests", "purityA", "10_input_filterflagremove.rds", package="msPuri
```

---

frag4feature,purityA-method

Using a purityA object, link MS/MS data to XCMS features

Description

General:

Assign fragmentation spectra (MS/MS) stored within a purityA class object to grouped features within an XCMS xset object.

XCMS calculates individual chromatographic peaks for each mzML file (retrieved using xcms::chromPeaks(xcmsObj)) these are then grouped together (using xcms::groupChromPeaks). Ideally the mzML files that contain the MS/MS spectra also contain sufficient MS1 scans for XCMS to detect MS1 chromatographic features. If this is the case, to determine if a MS2 spectra is to be linked to an XCMS grouped feature, the associated acquisition time of the MS/MS event has to be within the retention time window defined for the individual peaks associated for each file. The precursor m/z value also has to be within the user ppm tolerance to XCMS feature.

See below for representation of the linking (the * —— * represent a many-to-many relationship) e.g. 1 or more MS/MS events can be linked to 1 or more individual feature and an individual XCMS feature can be linked to 1 or more grouped XCMS features

- [grouped XCMS feature - across files] * —— * [individual XCMS feature - per file] * —— * [MS/MS spectra]

Alternatively, if the "useGroup" argument is set to TRUE, the full width of the grouped peak (determined as the minimum rtmin and maximum rtmax of the all associated individual peaks) will be used. This option should be used if the mzML file with MS/MS has very limited MS1 data and so individual chromatographic peaks might not be detected with the mzML files containing the MS/MS data. However, it should be noted this may lead to potential inaccurate linking.

- [grouped XCMS peaks] * —— * [MS/MS spectra]

Example LC-MS/MS processing workflow:

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> **frag4feature** -> filterFragSpectra -> averageAllFragSpectra -> createDatabase -> spectralMatching -> (sqlite spectral database)

Additional notes:

- If using only a single file, then grouping still needs to be performed within XCMS before frag4feature can be used.
- Fragmentation spectra below a certain precursor ion purity can be be removed (see plim argument).
- A SQLite database can be created directly here but the functionality has been deprecated and the createDatabase function should now be used
- Can experience some problems when using XCMS version < 3 and obiwarp retention time correction.

Usage

```
## S4 method for signature 'purityA'
frag4feature(
  pa,
  xcmsObj,
  ppm = 5,
  plim = NA,
  intense = TRUE,
  convert2RawRT = TRUE,
  useGroup = FALSE,
  createDb = FALSE,
  outDir = ".",
  dbName = NA,
  grpPeaklist = NA,
  use_group = NA,
  out_dir = NA,
  create_db = NA,
  grp_peaklist = NA,
  db_name = NA,
  xset = NA
)
```

Arguments

pa	object; purityA object
xcmsObj	object; XCMSnExp, xcmsSet or xsAnnotate object derived from the same files as those used to create the purityA object
ppm	numeric; ppm tolerance between precursor mz and XCMS feature mz
plim	numeric; minimum purity of precursor to be included
intense	boolean; If TRUE the most intense precursor will be used. If FALSE the precursor closest to the center of the isolation window will be used
convert2RawRT	boolean; If retention time correction has been used in XCMS set this to TRUE
useGroup	boolean; Ignore individual peaks and just find matching fragmentation spectra within the (full) rtmin rtmax of each grouped feature
createDb	boolean; if yes, generate a database of MS2 spectra
outDir	string; path where (optionally generated) database file should be saved
dbName	character; name to assign to (optionally exported) database.
grpPeaklist	dataframe; Can use any peak dataframe to add to databse. Still needs to be derived from the "obj" object though
use_group	boolean; (Deprecated, to be removed - replaced with useGroup argument for style consistency)
out_dir	character; (Deprecated, to be removed - use createDatabase function) Path where database will be created
create_db	boolean; (Deprecated, to be removed - use createDatabase function) SQLite database will be created of the results
grp_peaklist	dataframe; (Deprecated, to be removed - use createDatabase function) Can use any peak dataframe to add to databse. Still needs to be derived from the xset object though
db_name	character; (Deprecated, to be removed - use createDatabase function) If create_db is TRUE, a custom database name can be used, default is a time stamp
xset	object; (Deprecated, to be removed - use xcmsObj) 'xcmsSet' object derived from the same files as those used to create the purityA object

Value

Returns a purityA object (pa) with the following slots populated:

- pa@grped_df: A dataframe of the grouped XCMS features linked to the associated fragmentation spectra precursor details is recorded here
- pa@grped_ms2: A list of fragmentation spectra associated with each grouped XCMS feature is recorded here
- pa@f4f_link_type: The linking method is recorded here (e.g. individual peaks or grouped - "useGroup=TRUE")

Examples

```
library(xcms)
library(MSnbase)
library(magrittr)
#====== XCMS ================================
## Read in MS data
```

```
msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

## Or if using the old XCMS functions
#xcmsObj <- xcms::xcmsSet(msmsPths)
#xcmsObj <- xcms::group(xcmsObj)
#xcmsObj <- xcms::retcor(xcmsObj)
#xcmsObj <- xcms::group(xcmsObj)

#====== msPurity ============================
pa <- purityA(msmsPths)
pa <- frag4feature(pa, xcmsObj)
```

`Getfiles` *Get files for DI-MS processing*

Description

Takes in a folder path and outputs the a data frame structure for purityD. Function modified from mzmatch.

Usage

```
Getfiles(
  projectFolder = NULL,
  recursive = FALSE,
  pattern = ".csv",
  check = TRUE,
  raw = FALSE,
  peakout = NA,
  cStrt = TRUE,
  mzml_out = FALSE
)
```

Arguments

`projectFolder`	character; Directory path
`recursive`	boolean; Recursively check for files
`pattern`	character; File suffix to check for

check	boolean; Check with a GUI the files
raw	(REDUNDANT)
peakout	(REDUNDANT)
cStrt	boolean; Use the first word as the class name for files
mzml_out	(REDUNDANT)

Value

dataframe of files

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
```

---

getP,purityD-method *Get peaklist for a purityD object*

Description

output peak list for a purityD object

Usage

```
## S4 method for signature 'purityD'
getP(x)
```

Arguments

x	object; purityD object

Value

peaks

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
peaks <- getP(ppDIMS)
```

get_additional_mzml_meta

Get additional mzML meta

Description

Extract the filter strings 'accession MS:1000512' from an mzML file. Called header in thermo software. Enables quick access to various information regarding each scan

Usage

```
get_additional_mzml_meta(mzml_pth)
```

Arguments

mzml_pth	character; mzML path

Value

dataframe of meta info

Examples

```
mzml_pth <- system.file("extdata", "dims", "mzML", 'B02_Daph_TEST_pos.mzML', package="msPurityData")
meta_df <- get_additional_mzml_meta(mzml_pth)
```

groupPeaks,purityD-method

Using purityD object, group multiple peaklists by similar mz values (mzML or .csv)

Description

Uses a purityD object to group all the peaklists in the 'avPeaks$processing' slot

Usage

```
## S4 method for signature 'purityD'
groupPeaks(Object, ppm = 3, sampleOnly = FALSE, clustType = "hc")
```

Arguments

Object	object = purityD object
ppm	numeric = The ppm tolerance to group peaklists
sampleOnly	= if TRUE the sample peaks will only be grouped
clustType	= if 'hc' the hierarchical clustering, if 'simple' the mz values will just be grouped using a simple 1D method

Value

data.frame of peaklists grouped together by mz

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
ppDIMS <- averageSpectra(ppDIMS)
grpedP <- groupPeaks(ppDIMS)
```

---

groupPeaksEx *Group peaklists from a list of dataframes*

Description

Group a list of dataframes by their m/z values

Usage

```
groupPeaksEx(peak_list, cores = 1, clustType = "hc", ppm = 2)
```

Arguments

peak_list	list = A list (named) of dataframes consiting of a least the following columns ['peakID', 'mz']
cores	= number of cores used for calculation
clustType	= if 'hc' the hierarchical clustering, if 'simple' the mz values will just be grouped using a simple 1D method
ppm	numeric = The ppm tolerance to group peaklists

Value

data.frame of peaklists grouped together by mz

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
ppDIMS <- averageSpectra(ppDIMS)
grpedP <- groupPeaks(ppDIMS)
```

initialize,purityD-method *Constructor for S4 class to represent a DI-MS purityD*

Description

The class used to predict purity from an DI-MS dataset.

Usage

```
## S4 method for signature 'purityD'
initialize(.Object, fileList, cores = 1, mzML = TRUE, mzRback = "pwiz")
```

Arguments

.Object	object; purityD object
fileList	data.frame; created using GetFiles, data.frame with filepaths and sample class information
cores	numeric; Number of cores used to perform Hierarchical clustering WARNING: memory intensive, default 1
mzML	boolean; TRUE if mzML to be used FALSE if .csv file to be used
mzRback	character; backend to use for mzR parsing

Value

purityD object

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
```

iwNormGauss *Gaussian normalisation for isolation window efficiency*

Description

Creates a function based on a gaussian curve shape that will normalise any intensity values within a defined isolation window.

The function that is created will output a value between 0 to 1 based on the position between the minOff and maxOff params. (The value 1.0 being equivalent to 100% efficient)

Usage

```
iwNormGauss(sdlim = 3, minOff = -0.5, maxOff = +0.5)
```

Arguments

sdlim	numerical; Standard deviation limit for gaussian curve
minOff	numerical; Offset to the 'left' for the precursor range. (Should be negative)
maxOff	character; Offset to the 'left' for the precursor range. (Should be positive)

Value

normalisation function for selected range.

Examples

```
iwNormFun <- iwNormGauss(minOff=-0.5, maxOff=0.5)
pm <- data.frame(mz=c(99.5, 99.9, 100, 100.1, 100.5),i=c(1000, 1000, 1000, 1000, 1000))
mzmax = 100.5
mzmin = 99.5
middle <- mzmax-(mzmax-mzmin)/2
adjustmz = pm$mz-middle

# normalise the intensities
pm$normi = pm$i*iwNormFun(adjustmz)
```

---

iwNormQE.5 *Q-Exactive +/- 0.5 range, normalisation for isolation window efficiency*

Description

Creates a function based on a previous experimental analysis of a Q-Exactive at +/- 0.5 isolation window efficiency. See http://pubs.acs.org/doi/abs/10.1021/acs.analchem.6b04358

The function that is created will output a value between 0 to 1 based on the position between the minOff and maxOff params

NOTE: The resulting function will work for values greater that 0.5 and less than -0.5.

This is because (on our instrument tested at least) when using a window of +/- 0.5, the isolation is NOT confined to the +/-0.5 Da window. Resulting in ions from outside the window being isolated. For this reason the function can normalise values outside of the the +/- 1 Da range. Please see above paper figure 3 for more details.

Usage

```
iwNormQE.5()
```

Value

normalisation function for +/- 0.5 range for Q-Exactive

Examples

```
iwNormFun <- iwNormQE.5()
pm <- data.frame(mz=c(99.5, 99.9, 100, 100.1, 100.5),i=c(1000, 1000, 1000, 1000, 1000))
mzmax = 100.5
mzmin = 99.5
middle <- mzmax-(mzmax-mzmin)/2
adjustmz = pm$mz-middle

# normalise the intensities
pm$normi = pm$i*iwNormFun(adjustmz)
```

`iwNormRcosine` *Raised cosine normalisation for isolation window efficiency*

Description

Creates a function based on a rasied cosine curve shape that will normalise any intensity values within a defined isolation window

The function that is created will output a value between 0 to 1 based on the position between the minOff and maxOff params

Usage

```
iwNormRcosine(minOff = -0.5, maxOff = +0.5)
```

Arguments

`minOff`	numerical; Offset to the 'left' for the precursor range. (Should be negative)
`maxOff`	character; Offset to the 'left' for the precursor range. (Should be positive)

Value

normalisation function for selected range

Examples

```
iwNormFun <- iwNormRcosine()
pm <- data.frame(mz=c(99.5, 99.9, 100, 100.1, 100.5),i=c(1000, 1000, 1000, 1000, 1000))
mzmax = 100.5
mzmin = 99.5
middle <- mzmax-(mzmax-mzmin)/2
adjustmz = pm$mz-middle

# normalise the intensities
pm$normi = pm$i*iwNormFun(adjustmz)
```

`msPurity` `msPurity` *package*

Description

msPurity Bioconductor

Author(s)

Maintainer: Thomas N. Lawson <thomas.nigel.lawson@gmail.com> (ORCID)

Other contributors:

- Ralf Weber [contributor]
- Martin Jones [contributor]
- Julien Saint-Vanne [contributor]
- Andris Jankevics [contributor]
- Mark Viant [thesis advisor]
- Warwick Dunn [thesis advisor]

See Also

Useful links:

- https://github.com/computational-metabolomics/msPurity/
- Report bugs at https://github.com/computational-metabolomics/msPurity/issues/new

`pcalc` *Perform purity calculation on a peak matrix*

Description

This is the main purity calculation that is performed in purityX, purityD and purityA.

- Takes in a matrix of peaks
- gets isolation window based on mzmin mzmax
- locates the mz target in the peak matrix
- removes isotopic peaks
- removes any peaks below limit (percentage of target peak intensity)
- normalises
- Calculates purity: Divides the target peak intensity by the total peak intensity for the isolation window

Usage

```
pcalc(
  peaks,
  mzmin,
  mzmax,
  mztarget,
  ppm = NA,
  iwNorm = FALSE,
  iwNormFun = NULL,
  ilim = 0,
  targetMinMZ = NA,
  targetMaxMZ = NA,
  isotopes = FALSE,
  im = NULL
)
```

Arguments

peaks	matrix; Matrix of peaks consisting of 2 columns: mz and i
mzmin	numeric; Isolation window (min)
mzmax	numeric; Isolation window (max)
mztarget	numeric; The mz window to target in the isolation window
ppm	numeric; PPM tolerance for the target mz value. If NA will presume targetMinMZ and targetMaxMZ will be used
iwNorm	boolean; If TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function
iwNormFun	function; A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric; All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
targetMinMZ	numeric; Range to look for the mztarget (min)
targetMaxMZ	numeric; Range to look for the mztarget (max)
isotopes	boolean; TRUE if isotopes are to be removed
im	matrix; Isotope matrix, default removes C13 isotopes (single, double and triple bonds)

Value

a vector of the purity score and the number of peaks in the window e.g c(purity, pknm)

Examples

```
pm <- rbind(c(100, 1000),c(101.003, 10))
pcalc(pm, mzmin = 98, mzmax = 102, mztarget=100, ppm=5)
pcalc(pm, mzmin = 98, mzmax = 102, mztarget=100, ppm=5, isotopes = TRUE)
```

purityA *Assess the acquired precursor ion purity of MS/MS spectra (constructor)*

Description

General:

Given a vector of LC-MS/MS or DI-MS/MS mzML file paths calculate the precursor ion purity of each MS/MS scan.

The precursor ion purity represents the measure of the contribution of a selected precursor peak in an isolation window used for fragmentation and can be used as away of assessing the spectral quality and level of "contamination" of fragmentation spectra.

The calculation involves dividing the intensity of the selected precursor peak by the total intensity of the isolation window and is performed before and after the MS/MS scan of interest and interpolated at the recorded time of the MS/MS acquisition.

Additionally, isotopic peaks are annotated and omitted from the calculation, low abundance peaks are removed that are thought to have minor contribution to the resulting MS/MS spectra and the isolation efficiency of the mass spectrometer can be used to normalise the intensities used for the calculation.

The output is a purityA S4 class object (referred to as pa for convenience throughout the manual). The object contains a slot (pa@puritydf) where the details of the purity assessments for each MS/MS scan. The purityA object can then be used for further processing including linking the fragmentation spectra to XCMS features, averaging fragmentation, database creation and spectral matching (from the created database).

Example LC-MS/MS processing workflow:

The purityA object can be used for further processing including linking the fragmentation spectra to XCMS features, averaging fragmentation, database creation and spectral matching (from the created database). See below for an example workflow:

- Purity assessments
  - (mzML files) -> **purityA** -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)
  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)
- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> averageAllFragSpectra -> createDatabase -> spectralMatching -> (sqlite spectral database)

Isolation efficiency:

When the isolation efficiency of an MS instrument is known the peak intensities within an isolation window can be normalised for the precursor purity calculation. The isolation efficiency can be estimated by measuring a single precursor across a sliding window. See figure 3 from the original msPurity paper (Lawson et al 2017). This has been experimentally measured for a Thermo Fisher Q-Exactive Mass spectrometer using 0.5 Da windows and can be set within msPurity by using msPurity::iwNormQE.5() as the input to the iwNormFunc argument.

Other options to model the isolation efficiency the gaussian isolation window msPurity::iwNormGauss(minOff=-0.5, maxOff = 0.5) or a R-Cosine window msPurity::iwNormRCosine(minOff=-0.5, maxOff=0.5). Where the minOff and maxOff can be altered depending on the isolation window size.

A user can also define their own normalisation function. The only requirement of the function is that given a value between the minOff and maxOff a normalisation value between 0-1 is returned.

Notes regarding instrument specific isolation window offsets used::

- The isolation widths offsets will be automatically determined from extracting metadata from the mzML file. However, for some vendors though this is not recorded, in these cases the offsets should be given by the user as an argument (offsets).
- In the case of Agilent only the "narrow" isolation is supported. This roughly equates to +/- 0.65 Da (depending on the instrument). If the file is detected as originating from an Agilent instrument the isolation widths will automatically be set as +/- 0.65 Da.

Usage

```
purityA(
  fileList,
  cores = 1,
  mostIntense = FALSE,
  nearest = TRUE,
  offsets = NA,
  plotP = FALSE,
  plotdir = NULL,
  interpol = "linear",
  iwNorm = FALSE,
  iwNormFun = NULL,
  ilim = 0.05,
  mzRback = "pwiz",
  isotopes = TRUE,
  im = NULL,
  ppmInterp = 7
)
```

Arguments

fileList	vector; mzML file paths
cores	numeric; Number of cores to use
mostIntense	boolean; True if the most intense peak is used for calculation. Set to FALSE if the peak closest to mz value detailed in mzML meta data.
nearest	boolean; True if the peak selected is from either the preceding scan or the nearest.
offsets	vector; Override the isolation offsets found in the mzML file e.g. c(0.5, 0.5)
plotP	boolean; If TRUE a plot of the purity is to be saved
plotdir	vector; If plotP is TRUE plots will be saved to this directory
interpol	character; type of interolation to be performed "linear" or "spline" (Spline option is only included for testing purposes, linear should be used for all standard cases, isotope removal is also not available for the spline option)
iwNorm	boolean; If TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function

iwNormFun	function; A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric; All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
mzRback	character; backend to use for mzR parsing
isotopes	boolean; TRUE if isotopes are to be removed
im	matrix; Isotope matrix, default removes C13 isotopes (single, double and triple bonds)
ppmInterp	numeric; Set the ppm tolerance for the precursor ion purity interpolation. i.e. the ppm tolerence between the precursor ion found in the neighbouring scans.

Value

Returns a purityA object (pa) with the pa@puritydf slot updated

The purity dataframe (**pa@puritydf**) consists of the following columns:

- pid: unique id for MS/MS scan
- fileid: unique id for mzML file
- seqNum: scan number
- precursorIntensity: precursor intensity value as defined in the mzML file
- precursorMZ: precursor m/z value as defined in the mzML file
- precursorRT: precursor RT value as defined in the mzML file
- precursorScanNum: precursor scan number value as defined in mzML file
- id: unique id (redundant)
- filename: mzML filename
- precursorNearest: MS1 scan nearest to the MS/MS scan
- aMz: The m/z value in the "precursorNearest" MS1 scan which most closely matches the precursorMZ value provided from the mzML file
- aPurity: The purity score for aMz
- apkNm: The number of peaks in the isolation window for aMz
- iMz: The m/z value in the precursorNearest MS1 scan that is the most intense within the isolation window.
- iPurity: The purity score for iMz
- ipkNm: The number of peaks in the isolation window for iMz
- inPurity: The interpolated purity score (the purity score is calculated at neighbouring MS1 scans and interpolated at the point of the MS/MS acquisition)
- inpkNm: The interpolated number of peaks in the isolation window

The remaining slots for purityA class include

- pa@cores: The number of CPUs to be used for any further processing with this purityA object
- pa@fileList: list of the mzML files that have been processed
- pa@mzRback: The backend library used by mzR to extract information from the mzML file (e.g. pwiz)
- pa@grped_df: If frag4feature has been performed, a dataframe of the grouped XCMS features linked to the associated fragmentation spectra precursor details is recorded here

- pa@grped_ms2: If frag4feature has been performed, a list of fragmentation spectra associated with each grouped XCMS feature is recorded here
- pa@f4f_link_type: If frag4feature has been performed, the 'linking method' is recorded here, e.g. 'group' or 'individual'. Default is 'individual', see frag4feature documentation for more details
- pa@av_spectra: if averageIntraFragSpectra, averageInterFragSpectra, or averageAllFragSpectra have been performed, the average spectra is recorded here
- pa@av_intra_params: If averageIntraFragSpectra has been performed, the parameters are recorded here
- pa@av_inter_params: if averageInterFragSpectra has been performed, the parameters are recorded here]
- pa@av_all_params: If averageAllFragSpectra has been performed, the parameters are recorded here
- pa@db_path: If create_database has been performed, the resulting path to the database is recorded here

See Also

assessPuritySingle

Examples

```
filepths <- system.file("extdata", "lcms", "mzML", "LCMSMS_1.mzML", package="msPurityData")
pa <- purityA(filepths)
```

---

purityD-class *An S4 class to represent a DI-MS purityD*

Description

The class used to assess anticipated purity from a DI-MS run

Arguments

.Object	object; purityD object
fileList	data.frame; Created using GetFiles, data.frame with filepaths and sample class information
cores	numeric; Number of cores used to perform Hierarchical clustering WARNING: memory intensive, default 1
mzML	boolean; TRUE if mzML to be used FALSE if .csv file to be used

Value

purityD object

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)
ppDIMS <- purityD(fileList=inDF, cores=1, mzML=TRUE)
```

purityX *Assessing anticipated purity of XCMS features from an LC-MS run*

Description

Constructor for the purityX class.

Given an XCMS object get the anticipated precursor purity of the grouped peaks

Usage

```
purityX(
  xset,
  purityType = "purityFWHMmedian",
  offsets = c(0.5, 0.5),
  fileignore = NULL,
  cores = 1,
  xgroups = NULL,
  iwNorm = FALSE,
  iwNormFun = NULL,
  ilim = 0.05,
  plotP = FALSE,
  mzRback = "pwiz",
  isotopes = FALSE,
  im = NULL,
  singleFile = 0,
  rtrawColumns = FALSE,
  saveEIC = FALSE,
  sqlitePth = NULL
)
```

Arguments

xset	object; xcms object
purityType	character; Area and average used for the purity predictions. Options are "purityFWHMmedian", "purityFWmedian", "purityFWHMmean", "purityFWmean"
offsets	vector; vector of the isolation window upper and lower offsets
fileignore	vector; vector of files to ignore for the prediction calculation
cores	numeric; number of cores to use
xgroups	vector; vector of xcms groups to perform prediction on
iwNorm	boolean; if TRUE then the intensity of the isolation window will be normalised based on the iwNormFun function
iwNormFun	function; A function to normalise the isolation window intensity. The default function is very generalised and just accounts for edge effects
ilim	numeric; All peaks less than this percentage of the target peak will be removed from the purity calculation, default is 5% (0.05)
plotP	boolean; TRUE if plot of the EIC of feature and associated contamination is the be save to the working directory
mzRback	character; backend to use for mzR parsing

isotopes	boolean; TRUE if isotopes are to be removed
im	matrix; Isotope matrix, default removes C13 isotopes (single, double and triple bonds)
singleFile	numeric; If just a single file for purity is to be calculated (rather than the grouped XCMS peaks). Uses the index of the files in xcmsSet object. If zero this is ignored.
rtrawColumns	boolean; TRUE if the rt_raw values are included as additional columns in the @peaks slot (only required if using the obiwarp)
saveEIC	boolean; If True extracted ion chromatograms will be saved to SQLite database
sqlitePth	character; If saveEIC True, then a path to sqlite database can be used. If NULL then a database will be created in the working directory called eics

Value

a purityX object containing a dataframe of predicted purity scores

Examples

```
msPths <- list.files(system.file("extdata", "lcms", "mzML", package="msPurityData"), full.names = TRUE, patter
xset <- readRDS(system.file("extdata", "tests", "xcms", "ms_only_xset_OLD.rds", package="msPurity"))
xset@filepaths[1] <- msPths[basename(msPths)=="LCMS_1.mzML"]
xset@filepaths[2] <- msPths[basename(msPths)=="LCMS_2.mzML"]
px <- purityX(xset, singleFile = 1)
```

---

show,purityA-method *Show method for purityA class*

Description

print statement for purityA class

Usage

```
## S4 method for signature 'purityA'
show(object)
```

Arguments

object	object; purityA object

Value

a print statement of regarding object

show,purityD-method *Show method for purityD*

Description

Show method for purityD object

Usage

```
## S4 method for signature 'purityD'
show(object)
```

Arguments

object	= purityD object

Value

a print statement of regarding object

show,purityX-method *Show method for purityX*

Description

Show method for purityX object

Usage

```
## S4 method for signature 'purityX'
show(object)
```

Arguments

object	object; purityX object

Value

a print statement of regarding object

`spectralMatching` *Spectral matching for LC-MS/MS datasets*

Description

General

Perform spectral matching to spectral libraries for an LC-MS/MS dataset.

The spectral matching is performed from a **Query** SQLite spectral-database against a **Library** SQLite spectral-database.

The SQLite schema of the spectral database can be detailed Schema details can be found here.

The query spectral-database in most cases should contain be the "unknown" spectra database generated the msPurity function createDatabase as part of a msPurity-XCMS data processing workflow.

The library spectral-database in most cases should contain the "known" spectra from either public or user generated resources. The library SQLite database by default contains data from MoNA including Massbank, HMDB, LipidBlast and GNPS. A larger database can be downloaded from here. To create a user generated library SQLite database the following tool can be used to generate a SQLite database from a collection of MSP files: msp2db. It should be noted though, that as long as the schema of the spectral-database is as described here , then any database can be used for either the library or query - even allowing for the same database to be used.

The spectral matching functionality has four main components, spectral filtering, spectral alignment, spectral matching, and summarising the results.

Spectral filtering is simply filtering both the library and query spectra to be search against (e.g. choosing the library source, instrument, retention time, precursor PPM tolerance etc).

The spectral alignment stage involves aligning the query peaks to the library peaks. The approach used is similar to modified pMatch algorithm described in Zhou et al 2015.

The spectral matching of the aligned spectra is performed against a combined intensity and m/z weighted vector - created for both the query and library spectra (wq and wl). See below:

$$w = intensity^x * mz^y$$

Where x and y represent weight factors, defaults to x=0.5 and y=2 as per MassBank. These can be adjusted by the user though.

The aligned weighted vectors are then matched using dot product cosine, reverse dot product cosine and the composite dot product. See below for dot product cosine equation.

$$dpc = wq * wl / \sqrt{\sum wq^2} * \sqrt{\sum wl^2}$$

See the vigenttes for more details regarding matching algorithms used.

Example LC-MS/MS processing workflow

- Purity assessments
  - (mzML files) -> purityA -> (pa)
- XCMS processing
  - (mzML files) -> xcms.findChromPeaks -> (optionally) xcms.adjustRtime -> xcms.groupChromPeaks -> (xcmsObj)

  - — *Older versions of XCMS* — (mzML files) -> xcms.xcmsSet -> xcms.group -> xcms.retcor -> xcms.group -> (xcmsObj)

- Fragmentation processing
  - (xcmsObj, pa) -> frag4feature -> filterFragSpectra -> averageAllFragSpectra -> createDatabase -> **spectralMatching** -> (sqlite spectral database)

Usage

```
spectralMatching(
  q_dbPth,
  l_dbPth = NA,
  q_purity = NA,
  q_ppmProd = 10,
  q_ppmPrec = 5,
  q_raThres = NA,
  q_pol = NA,
  q_instrumentTypes = NA,
  q_instruments = NA,
  q_sources = NA,
  q_spectraTypes = c("av_all", "inter"),
  q_pids = NA,
  q_rtrange = c(NA, NA),
  q_spectraFilter = TRUE,
  q_xcmsGroups = NA,
  q_accessions = NA,
  l_purity = NA,
  l_ppmProd = 10,
  l_ppmPrec = 5,
  l_raThres = NA,
  l_pol = "positive",
  l_instrumentTypes = NA,
  l_instruments = NA,
  l_sources = NA,
  l_spectraTypes = NA,
  l_pids = NA,
  l_rtrange = c(NA, NA),
  l_spectraFilter = FALSE,
  l_xcmsGroups = NA,
  l_accessions = NA,
  usePrecursors = TRUE,
  raW = 0.5,
  mzW = 2,
  rttol = NA,
  q_dbType = "sqlite",
  q_dbName = NA,
  q_dbHost = NA,
  q_dbUser = NA,
  q_dbPass = NA,
  q_dbPort = NA,
  l_dbType = "sqlite",
  l_dbName = NA,
  l_dbHost = NA,
```

```
  l_dbUser = NA,
  l_dbPass = NA,
  l_dbPort = NA,
  cores = 1,
  updateDb = FALSE,
  copyDb = FALSE,
  outPth = "sm_result.sqlite"
)
```

Arguments

q_dbPth	character; Path of the database of queries that will be searched against the library spectra. Generated from createDatabase
l_dbPth	character; path to library spectral SQLite database. Defaults to msPurityData package data.
q_purity	character; Precursor ion purity threshold for the query spectra
q_ppmProd	numeric; ppm tolerance for query product
q_ppmPrec	numeric; ppm tolerance for query precursor
q_raThres	numeric; Relative abundance threshold for query spectra
q_pol	character; Polarity of query spectra ('positive', 'negative', NA).
q_instrumentTypes	vector; Instrument types for query spectra.
q_instruments	vector; Instruments for query spectra (note that this is used in combination with q_instrumentTypes - any spectra matching either q_instrumentTypes or q_instruments will be used).
q_sources	vector; Sources of query spectra (e.g. massbank, hmdb).
q_spectraTypes	character; Spectra types of query spectra to perfrom spectral matching e.g. ('scan', 'av_all', 'intra', 'inter')
q_pids	vector; pids for query spectra (correspond to column 'pid; in s_peak_meta)
q_rtrange	vector; retention time range (in secs) of query spectra, first value mininum time and second value max e.g. c(0, 10) is between 0 and 10 seconds
q_spectraFilter	boolean; For query spectra, if prior filtering performed with msPurity, flag peaks will be removed from spectral matching
q_xcmsGroups	vector; XCMS group ids for query spectra
q_accessions	vector; accession ids to filter query spectra
l_purity	character; Precursor ion purity threshold for the library spectra (uses interpolated purity - inPurity)
l_ppmProd	numeric; ppm tolerance for library product
l_ppmPrec	numeric; ppm tolerance for library precursor
l_raThres	numeric; Relative abundance threshold for library spectra
l_pol	character; Polarity of library spectra ('positive', 'negative', NA)
l_instrumentTypes	vector; Instrument types for library spectra.
l_instruments	vector; Instruments for library spectra (note that this is used in combination with q_instrumentTypes - any spectra matching either q_instrumentTypes or q_instruments will be used).

l_sources	vector; Sources of library spectra (e.g. massbank, hmdb).
l_spectraTypes	vector; Spectra type of library spectra to perfrom spectral matching with e.g. ('scan', 'av_all', 'intra', 'inter')
l_pids	vector; pids for library spectra (correspond to column 'pid; in s_peak_meta)
l_rtrange	vector; retention time range (in secs) of library spectra, first value mininum time and second value max e.g. c(0, 10) is between 0 and 10 seconds
l_spectraFilter	boolean; For library spectra, if prior filtering performed with msPurity, flag peaks will be removed from spectral matching
l_xcmsGroups	vector; XCMS group ids for library spectra
l_accessions	vector; accession ids to filter library spectra
usePrecursors	boolean; If TRUE spectra will be filtered by similarity of precursors based on ppm range defined by l_ppmPrec and q_ppmPrec
raW	numeric; Relative abundance weight for spectra (default to 0.5 as determined by massbank for ESI data)
mzW	numeric; mz weight for spectra (default to 2 as determined by massbank for ESI data)
rttol	numeric ; Tolerance in time range between the library and query spectra retention time
q_dbType	character; Query database type for compound database can be either (sqlite, postgres or mysql)
q_dbName	character; Query database name (only applicable for postgres and mysql)
q_dbHost	character; Query database host (only applicable for postgres and mysql)
q_dbUser	character; Query database user (only applicable for postgres and mysql)
q_dbPass	character; Query database pass - Note this is not secure! use with caution (only applicable for postgres and mysql)
q_dbPort	character; Query database port (only applicable for postgres and mysql)
l_dbType	character; Library database type for compound database can be either (sqlite, postgres or mysql)
l_dbName	character; Library database name (only applicable for postgres and mysql)
l_dbHost	character; Library database host (only applicable for postgres and mysql)
l_dbUser	character; Library database user (only applicable for postgres and mysql)
l_dbPass	character; Library database pass - Note this is not secure! use with caution (only applicable for postgres and mysql)
l_dbPort	character; Library database port (only applicable for postgres and mysql)
cores	numeric; Number of cores to use
updateDb	boolean; Update the Query SQLite database with the results
copyDb	boolean; If updating the database - perform on a copy rather thatn the original query database
outPth	character; If copying the database - the path of the new database file

Value

Returns a list containing the following elements

q_dbPth

Path of the query database (this will have been updated with the annotation results if updateDb argument used)

xcmsMatchedResults

If the qeury spectra had XCMS based chromotographic peaks tables (e.g c_peak_groups, c_peaks) in the sqlite database - it will be possible to summarise the matches for each XCMS grouped feature. The dataframe contains the following columns

- lpid - id in database of library spectra
- qpid - id in database of query spectra
- dpc - dot product cosine of the match
- rdpc - reverse dot product cosine of the match
- cdpc - composite dot product cosine of the match
- mcount - number of matching peaks
- allcount - total number of peaks across both query and library spectra
- mpercent - percentage of matching peaks across both query and library spectra
- library_rt - retention time of library spectra
- query_rt - retention time of query spectra
- rtdiff - difference between library and query retention time
- library_precursor_mz - library precursor mz
- query_precursor_mz - query precursor mz
- library_precursor_ion_purity - library precursor ion purity
- query_precursor_ion_purity - query precursor ion purity
- library_accession - library accession value (unique string or number given to eith MoNA or Massbank data entires)
- library_precursor_type - library precursor type (i.e. adduct)
- library_entry_name - Name given to the library spectra
- inchikey - inchikey of the matched library spectra
- library_source_name - source of the spectra (e.g. massbank, gnps)
- library_compound_name - name of compound spectra was obtained from

matchedResults

All matched results from the query spectra to the library spectra. Contains the same columns as above but without the XCMS details. This table is useful to observe spectral matching results for all MS/MS spectra irrespective of if they are linked to XCMS MS1 features.

list of database details and dataframe summarising the results for the xcms features

Examples

```
#====== XCMS ================================
## Read in MS data
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#           package="msPurityData"), full.names = TRUE, pattern = "MSMS")
#ms_data = readMSData(msmsPths, mode = 'onDisk', msLevel. = 1)

## Find peaks in each file
#cwp <- CentWaveParam(snthresh = 5, noise = 100, ppm = 10, peakwidth = c(3, 30))
#xcmsObj  <- xcms::findChromPeaks(ms_data, param = cwp)

## Optionally adjust retention time
#xcmsObj  <- adjustRtime(xcmsObj , param = ObiwarpParam(binSize = 0.6))

## Group features across samples
#pdp <- PeakDensityParam(sampleGroups = c(1, 1), minFraction = 0, bw = 30)
#xcmsObj <- groupChromPeaks(xcmsObj , param = pdp)

#====== msPurity ============================
#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa = pa, xcmsObj = xcmsObj)
#pa <- filterFragSpectra(pa, allfrag=TRUE)
#pa <- averageAllFragSpectra(pa)
#q_dbPth <- createDatabase(pa, xcmsObj, metadata=list('polarity'='positive','instrument'='Q-Exactive'))
#sm_result <- spectralMatching(q_dbPth, cores=4, l_pol='positive')

td <- tempdir()
q_dbPth <- system.file("extdata", "tests", "db", "createDatabase_example.sqlite", package="msPurity")

rid <- paste0(paste0(sample(LETTERS, 5, TRUE), collapse=""),  paste0(sample(9999, 1, TRUE), collapse=""), ".sq
sm_out_pth <- file.path(td, rid)

result <- spectralMatching(q_dbPth, q_xcmsGroups = c(53, 89, 410), cores=1, l_accessions = c('PR100407', 'ML005
                           q_spectraTypes = 'av_all',
                           updateDb = TRUE,
                           copyDb = TRUE,
                           outPth = sm_out_pth)
```

`spectral_matching` *Spectral matching deprecated*

Description

Perform spectral matching to spectral libraries using dot product cosine on a LC-MS/MS dataset and link to XCMS features.

msPurity::spectral_matching is deprecated - please use msPurity::spectralMatching for future use

Usage

```
spectral_matching(
  query_db_pth,
```

```
  ra_thres_l = 0,
  ra_thres_q = 2,
  cores = 1,
  pol = "positive",
  ppm_tol_prod = 10,
  ppm_tol_prec = 5,
  score_thres = 0.6,
  topn = NA,
  db_name = NA,
  library_db_pth = NA,
  instrument_types = NA,
  library_sources = "massbank",
  scan_ids = NA,
  pa = NA,
  xset = NA,
  grp_peaklist = NA,
  out_dir = ".",
  ra_w = 0.5,
  mz_w = 2,
  spectra_type_q = "scans",
  ra_thres_t = NA,
  target_db_pth = NA,
  rt_range = c(NA, NA),
  rttol = NA,
  match_alg = "dpc"
)
```

Arguments

query_db_pth	character; Path of the database of targets (queries) that will be searched against the library spectra. Generated either from frag4feature or from create_database functions.
ra_thres_l	numeric; Relative abundance threshold for library spectra
ra_thres_q	numeric; Relative abundance threshold for target (query) spectra (Peaks below this RA threshold will be excluded)
cores	numeric; Number of cores to use
pol	character; Polarity ['positive' or 'negative']
ppm_tol_prod	numeric; PPM tolerance to match to product
ppm_tol_prec	numeric; PPM tolerance to match to precursor
score_thres	numeric; Dot product cosine score threshold
topn	numeric [optional]; Only use top n matches
db_name	character [optional]; Name of the result database (e.g. can use CAMERA peaklist)
library_db_pth	character [optional]; path to library spectral SQLite database. Defaults to msPurityData package data.
instrument_types	vector [optional]; Vector of instrument types, defaults to all
library_sources	vector [optional]; Vector of library sources. Default option is for massbank only but the 'lipidblast' library is also available

scan_ids	vector [optional]; Vector of unique scan ids calculated from msPurity "pid". These scans will on used for the spectral matching. All scans will be used if set to NA
pa	purityA object [optional]; If target_db_pth set to NA, a new database can be created using pa, xset and grp_peaklist
xset	xcms object [optional]; If target_db_pth set to NA, a new database can be created using pa, xset and grp_peaklist
grp_peaklist	dataframe [optional]; If target_db_pth set to NA, a new database can be created using pa, xset and grp_peaklist
out_dir	character [optional]; If target_db_pth set to NA, Out directory for the SQLite result database
ra_w	numeric; Relative abundance weight for spectra
mz_w	numeric; mz weight for spectra
spectra_type_q	character; Type of fragmentation spectra from query to match with "scans" = all individual scans, "av_intra" = averaged spectra (intra), "av_inter" = averaged spectra (inter), "av_all" = averaged all spectra ignoring inter-intra relationships
ra_thres_t	numeric [deprecated]; The relative abundance threshold for the query spectra (use ra_thres_q for future use)
target_db_pth	character [deprecated]; The query database path (use query_db_pth for future use)
rt_range	vector [optional]; Vector of rention time range to filter the library spectra (rtmin, rtmax). Default is to ignore retention time range
rttol	numeric [optional]; Tolerance in time range between the Library and Query database retention time (in seconds) NA to ignore
match_alg	character; Can either use dot product cosine (dpc) or match factor (mf) for spectral matching. Defaults to dpc

Value

list of database details and dataframe summarising the results for the xcms features

Examples

```
#msmsPths <- list.files(system.file("extdata", "lcms", "mzML",
#                        package="msPurityData"), full.names = TRUE,
#                        pattern = "MSMS")
#xset <- xcms::xcmsSet(msmsPths)
#xset <- xcms::group(xset)
#xset <- xcms::retcor(xset)
#xset <- xcms::group(xset)

#pa  <- purityA(msmsPths)
#pa <- frag4feature(pa, xset)
#pa <- averageAllFragSpectra(pa)
#db_pth <- create_database(pa, xset)
#q_dbPth <- system.file("extdata", "tests", "db",
#                       "create_database_example.sqlite", package="msPurity")
#result <- spectral_matching(q_dbPth, spectra_type_q="av_all")
```

`subtract,purityD-method`

Using Subtract MZ values based on ppm tolerance and noise ratio

Description

Uses a purityD object with references to multiple MS files. Subtract blank peaks from the sample peaks see subtractMZ for more information

Usage

```
## S4 method for signature 'purityD'
subtract(
  Object,
  byClass = TRUE,
  mapping = c("sample", "blank"),
  ppm = 5,
  s2bthres = 10
)
```

Arguments

`Object`	object; purityD object
`byClass`	boolean; subtract within each class
`mapping`	parameter not functional (TODO)
`ppm`	numeric = ppm tolerance
`s2bthres`	numeric = threshold for the samp2blank (i1/i2)

Value

purityD object with averaged spectra

See Also

`subtractMZ`

Examples

```
datapth <- system.file("extdata", "dims", "mzML", package="msPurityData")
inDF <- Getfiles(datapth, pattern=".mzML", check = FALSE, cStrt = FALSE)

ppDIMS <- purityD(inDF, cores=1)
ppDIMS <- averageSpectra(ppDIMS)
ppDIMS <- filterp(ppDIMS, thr = 5000)
ppDIMS <- subtract(ppDIMS)
```

subtractMZ *Subtract MZ values based on ppm tolerance and noise ratio*

Description

This function is intended for blank subtraction of mz values from two peaklists. It takes in 2 vectors of mz values and 2 coresponding vectors of Intensity values.

The second mz values are subtracted from the first set within an MZ tolerance.

However, if the mz match but the intensity is above a defined threshold then they are not subtracted

Usage

```
subtractMZ(mz1, mz2, i1, i2, ppm = 5, s2bthres = 10)
```

Arguments

mz1	vector = mz values to start with
mz2	vector = mz values to subtract
i1	vector = i values for mz1
i2	vector = i values for mz2
ppm	numeric = ppm tolerance
s2bthres	numeric = threshold for the samp2blank (i1/i2)

Value

a vector of the remaining mz values

Examples

```
mz1 <- c(100.001, 200.002, 300.302)
mz2 <- c(100.004, 200.003, 500.101)
i1 <- c(100, 100, 100)
i2 <- c(100, 10000, 100)

subtractMZ(mz1, mz2, i1, i2, ppm=5, s2bthres =10)
```

validate,purityA-method *Validate precursor purity predictions using LC-MS and LC-MS/MS dataset*

Description

The method is used to validate the precursor purity predictions made from an LC-MS dataset

Usage

```
## S4 method for signature 'purityA'
validate(pa, ppLCMS)
```

Arguments

pa	object; purityA object
ppLCMS	object; purityX object

Value

purityA object

`writeOut,purityD-method`

Using purityD object, save peaks as text files

Description

Uses a purityD object with references to multiple MS files. Predicts the purity of the processed sample files

Usage

```
## S4 method for signature 'purityD'
writeOut(Object, outDir, original)
```

Arguments

Object	object; purityD object
outDir	character; Directory to save text files
original	boolean; If the original (unprocessed) files are to be saved to text files

Value

purityD object

Index