Package ‘ptairMS’

March 7, 2024

Title Pre-processing PTR-TOF-MS Data
Version 1.10.0
Date 2020-03-18
Description This package implements a suite of methods to preprocess data from PTR-TOF-MS instruments (HDF5 format) and generates the 'sample by features' table of peak intensities in addition to the sample and feature metadata (as a single ExpressionSet object for subsequent statistical analysis). This package also permit usefull tools for cohorts management as analyzing data progressively, visualization tools and quality control. The steps include calibration, expiration detection, peak detection and quantification, feature alignment, missing value imputation and feature annotation. Applications to exhaled air and cell culture in headspace are described in the vignettes and examples. This package was used for data analysis of Gassin Delyle study on adults undergoing invasive mechanical ventilation in the intensive care unit due to severe COVID-19 or non-COVID-19 acute respiratory distress syndrome (ARDS), and permit to identify four potential biomarkers of the infection.

License GPL-3
Encoding UTF-8
LazyData false

BugReports https://github.com/camilleroquencourt/ptairMS/issues
RoxygenNote 7.2.3
Imports Biobase, bit64, chron, data.table, doParallel, DT, enviPat, foreach, ggplot2, graphics, grDevices, ggrepur, gridExtra, Hmisc, methods, minpack.lm, MSnbase, parallel, plotly, rhdf5, rlang, Rcpp, shiny, shinyscreenshot, signal, scales, stats, utils

Suggests knitr, rmarkdown, BiocStyle, testthat (>= 2.1.0), ptairData, ropls

VignetteBuilder knitr
LinkingTo Rcpp
NeedsCompilation: yes

biocViews: Software, MassSpectrometry, Preprocessing, Metabolomics,
          PeakDetection, Alignment

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

git_branch: RELEASE_3_18

git_last_commit: 4b718ce

git_last_commit_date: 2023-10-24

Repository: Bioconductor 3.18

Date/Publication: 2024-03-06

Author: camille Roquencourt [aut, cre]

Maintainer: camille Roquencourt <camille.roquencourt@hotmail.fr>

R topics documented:

- ptairMS-package                      .................................................. 3
- aggregate                            ...................................................... 4
- align                                .......................................................... 4
- alignSamples                        ...................................................... 5
- annotateVOC                         ...................................................... 7
- calibration                         ...................................................... 9
- calibrationFun                      ...................................................... 10
- changeTimeLimits                    ................................................... 11
- convert_to_mzML                     ................................................... 11
- createPtrSet                        ..................................................... 12
- cumulative_fit_function             ................................................ 14
- deadTimeCorr                        ..................................................... 15
- defineKnots                          ..................................................... 15
- detectPeak                          ...................................................... 17
- determinePeakShape                  ................................................... 20
- exportSampleMetadata                .................................................. 20
- extractEIC                          ..................................................... 21
- fit_averagePeak                     ..................................................... 21
- fit_averagePeak_function            ................................................ 22
- formula2mass                        ..................................................... 23
- getDirectory                        ..................................................... 23
- getFileName                         ..................................................... 24
- getPeakList                         ..................................................... 25
- getSampleMetadata                   .................................................. 26
- importSampleMetadata                .................................................. 26
- impute                              ......................................................... 27
- imputeMat                           ......................................................... 28
- initializeFit                       ....................................................... 28
- LocalMaximaSG                       ..................................................... 30
- makeSubGroup                        ..................................................... 30
- OptimalWindowsSG                    ..................................................... 31
Description

This package implements a suite of methods to preprocess data from PTR-TOF-MS instruments (HDF5 format) and generates the ’sample by features’ table of peak intensities in addition to the sample and feature metadata (as a single ExpressionSet object for subsequent statistical analysis). This package also permit useful tools for cohorts management as analyzing data progressively, visualization tools and quality control. The steps include calibration, expiration detection, peak detection and quantification, feature alignment, missing value imputation and feature annotation. Applications to exhaled air and cell culture in headspace are described in the vignettes and examples. This package was used for data analysis of Gassin Delyle study on adults undergoing invasive mechanical ventilation in the intensive care unit due to severe COVID-19 or non-COVID-19 acute respiratory distress syndrome (ARDS), and permit to identify four potential biomarkers of the infection.

Author(s)

Maintainer: camille Roquencourt <camille.roquencourt@hotmail.fr>
aggregate

aggregate peakgroup for align function

Description

aggregate peakgroup for align function

Usage

aggregate(subGroupPeak, n.exp)

Arguments

subGroupPeak  teh group tp aggregate
n.exp        number of expiration done in the file

Value

a matrix with the median of mz, mean of ppb, ppb in background, and percentage of expiration where the peak is detected @keywords internal

align

Alignment with kernel gaussian density

Description

Alignment with kernel gaussian density

Usage

align(peakTab, ppmGroup = 70, dmzGroup = 0.001)

Arguments

peakTab  table with comlumn : mass, quantification, and groups number to aligned
ppmGroup width of sub group created beafore density estimation in ppm
dmzGroup width of sub group created beafore density estimation in Da

Value

A list containing groups formed by alignment.
**alignSamples**  
Alignment between samples

**Description**

AlignSamples performs alignment between samples (i.e. the matching of variables between the peak lists within the ptrSet object) by using a kernel gaussian density (Delabriere et al, 2017). This function returns an ExpressionSet, which contains the matrix of peak intensities, the sample metadata (borrowed from the input ptrSet) and the variable metadata which contains the peak intensities in the background. Two filters may be applied to:

- keep only variables with a significant higher intensity in the expirations compared to the background (i.e., a p-value less than pValGreaterThres) for at least fracExp
- keep only variables which are detected in more than fracGroup percent of the samples (or group)

If you do not want to apply those filters, set fracGroup to 0 and pValGreaterThres to 1.

**Usage**

```r
alignSamples(
  X,
  ppmGroup = 70,
  fracGroup = 0.8,
  group = NULL,
  fracExp = 0.3,
  pValGreaterThres = 0.001,
  pValLessThres = 0,
  quantiUnit = c("ppb", "ncps", "cps")[1],
  bgCorrected = TRUE,
  dmzGroup = 0.001
)
```

```r
## S4 method for signature 'ptrSet'
alignSamples(
  X,
  ppmGroup = 70,
  fracGroup = 0.8,
  group = NULL,
  fracExp = 0.3,
  pValGreaterThres = 0.001,
  pValLessThres = 0,
  quantiUnit = c("ppb", "ncps", "cps")[1],
  bgCorrected = TRUE,
  dmzGroup = 0.001
)
```
alignSamples

Arguments

X         ptrSet already processed by the detectPeak function
ppmGroup  ppm maximal width for an mz group
fracGroup only variables present in fracGroup percent of at least one group will be kept (if 0 the filter is not applied)
group     character: sampleMetadata data column name. If NULL, variables not present in fracGroup percent of samples will be deleted. Else, variables not present in fracGroup percent in in at least one group group will be removed.
fracExp    fraction of samples which must have their p-value less than pValGreaterThres and pValLessThres
pValGreaterThres threshold of the p-value for the unilateral testing that quantification (in cps) of expiration points are greater than the intensities in the background.
pValLessThres threshold of the p-value for the unilateral testing that quantification (in cps) of expiration points are less than the intensities of the background.
quantiUnit ppb, ncp or cps
bgCorrected logical: should the peak table contain the background corrected values?
dmzGroup minimum mz width to be used for grouping the features (required for low masses)

Value

an ExpressionSet (Biobase object)

References

Delabriere et al., 2017

Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw,
setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525),
fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mzNominal=c(21,60,79))
eset <- alignSamples(exhaledPtrset,pValGreaterThres=0.05)
Biobase::exprs(eset)
Biobase::fData(eset)
Biobase::pData(eset)
Putative annotation of VOC mz by using the reference compilation from the literature

Description

Putatively annotate VOC mz by using the reference compilation from the literature, and propose an isotope detection.

Usage

annotateVOC(
  x,
  ionMassColname = "ion_mass",
  ppm = 20,
  prefix = "vocDB_",
  fields = c("ion_mass", "ion_formula", "formula", "mass_monoiso", "name_iupac",
            "pubchem_cid", "inchi", "inchikey", "ref_year", "ref_pmid", "disease_name",
            "disease_meshid")[c(1, 2, 5)]
)

## S4 method for signature 'ExpressionSet'
annotateVOC(
  x,
  ionMassColname = "ion_mass",
  ppm = 50,
  prefix = "vocDB_",
  fields = c("ion_mass", "ion_formula", "formula", "mass_monoiso", "name_iupac",
            "pubchem_cid", "inchi", "inchikey", "ref_year", "ref_pmid", "disease_name",
            "disease_meshid")[c(1, 2, 5)]
)

## S4 method for signature 'data.frame'
annotateVOC(
  x,
  ionMassColname = "ion_mass",
  ppm = 50,
  prefix = "vocDB_",
  fields = c("ion_mass", "ion_formula", "formula", "mass_monoiso", "name_iupac",
            "pubchem_cid", "inchi", "inchikey", "ref_year", "ref_pmid", "disease_name",
            "disease_meshid")[c(1, 2, 5)]
)

## S4 method for signature 'numeric'
annotateVOC(
  x,
  ionMassColname = "",
  ppm = 50,
  prefix = "vocDB_",
  fields = c("ion_mass", "ion_formula", "formula", "mass_monoiso", "name_iupac",
            "pubchem_cid", "inchi", "inchikey", "ref_year", "ref_pmid", "disease_name",
            "disease_meshid")[c(1, 2, 5)]
)
annotateVOC

ppm = 50,
prefix = "vocDB_",
fields = c("ion_mass", "ion_formula", "formula", "mass_monoiso", "name_iupac",
        "pubchem_cid", "inchi", "inchikey", "ref_year", "ref_pmid", "disease_name",
        "disease_meshid")[c(1, 2, 5)]
)

Arguments

x Expression set object (resp. data.frame) (resp. numeric vector) containing the
    PTR-MS processed data (resp. containing a column with the ion mass values)
    (resp. containing the ion mass values)
ionMassColname Character: column name from the fData (resp. from the data.frame) containing
    the ion mass values; [default: 'ion_mass']; this argument is not used when x is
    a numeric vector
ppm Numeric: tolerance
prefix Character: prefix for the new 'annotation' columns [default: 'vocDB_']
fields Character vector: fields of the 'vocDB' database to be queried among: 'ion_mass'
    [default], 'ion_formula' [default], 'formula', 'mass_monoiso', 'name_iupac'
    [default], 'pubchem_cid', 'inchi', 'inchikey', 'ref_year', 'ref_pmid', 'disease_name',
    'disease_meshid'

Value

Returns the data.frame with additional columns containing the vocDB informations for the matched
ion_mass values as well as the detected isotopes

Examples

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw,
    setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525),
    fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mz=c(21,59,77))
exhaled.eset <-alignSamples(exhaledPtrset ,pValGreaterThres=0.05)
# Expression Set
exhaled.eset <- annotateVOC(exhaled.eset)
head(Biobase::fData(exhaled.eset))
# Data frame
exhaled_fdata.df <- Biobase::fData(exhaled.eset)
exhaled_fdata.df <- annotateVOC(exhaled_fdata.df)
head(exhaled_fdata.df)
# Numeric
ionMass.vn <- as.numeric(Biobase::featureNames(exhaled.eset))
annotated_ions.df <- annotateVOC(ionMass.vn)
head(annotated_ions.df)
**calibration**

*Calibrates the mass axis with references masses*

**Description**

To convert Time Of Flight (TOF) axis to mass axis, we use the formula: \( m_z = ((t_{of}\text{-}b)/a)^2 \) (Muller et al. 2013). To estimate those parameters, references peaks with accurate known masses and without overlapping peak are needed. The best is that the references masses covers a maximum of the mass range.

**Usage**

```r
calibration(
  x,
  mzCalibRef = c(21.022, 29.013424, 41.03858, 60.0525, 203.943, 330.8495),
  calibrationPeriod = 60,
  tol = 70,
  ...
)
```

### S4 method for signature 'ptrRaw'

```r
calibration(
  x,
  mzCalibRef = c(21.022, 29.013424, 41.03858, 59.049141, 75.04406, 203.943, 330.8495),
  calibrationPeriod = 60,
  tol = 70,
  ...
)
```

### S4 method for signature 'ptrSet'

```r
calibration(
  x,
  mzCalibRef = c(21.022, 29.013424, 41.03858, 75.04406, 203.943, 330.8495),
  calibrationPeriod = 60,
  tol = 70,
  fileNames = getParameters(x)$listFile
)
```

**Arguments**

- `x` a **ptrRaw-class** or **ptrSet-class** object
- `mzCalibRef` Vector of accurate mass values of intensive peaks and 'unique' in a nominal mass interval (without overlapping)
- `calibrationPeriod` in second, coefficient calibration are estimated for each sum spectrum of `calibrationPeriod` seconds
tol the maximum error tolerated in ppm. If more than tol warnings.

""

fileNames file to recalibrate

Value

the same ptrRaw or ptrSet as in input, with the following modified element:

• mz: the new mz axis calibrated
• rawM: same raw matrix with the new mz axis in rownames
• calibMassRef: reference masses used for the calibration
• calibMzToTof and calibTofToMz: function to convert TOF to mz
• calibError: the calibration error to the reference masses in ppm
• calibrationIndex: index time of each calibration period

Examples

### ptrRaw object

library(ptairData)
filePath <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5', package = 'ptairData')
raw <- readRaw(filePath, calib = FALSE)
rawCalibrated <- calibration(raw)

calibrationFun calibration function

description

Performs calibration on sp with mzCalibRef reference masses and mzToTofFunc as previous calibration function

Usage

calibrationFun(sp, mz, mzCalibRef, calibCoef, peakShape, tol)

Arguments

sp spectrum
mz mass axis
mzCalibRef masses of know reference peaks
calibCoef coefficient of the previous calibration
peakShape a list with reference axis and a reference peak shape centered in zero
tol maximum error tolarated in ppm
changeTimeLimits

Value
list

Description
Shiny application to modify and view expiration limits
This function runs a shiny app, where you can check the automatic expiration detection, knots location, and modify it.

Usage
changeTimeLimits(ptrSet)

Arguments
ptrSet a ptrSet object

Value
the ptrSet object modified

Examples
library(ptairData)
directory <- system.file("extdata/exhaledAir", package = "ptairData")
ptrSet <- createPtrSet(directory, setName="ptrSet", mzCalibRef=c(21.022,59.049), fracMaxTIC=0.8)
## Not run: ptrSet <- changeTimeLimits(ptrSet)

convert_to_mzML

Convert a h5 file to mzML

Description
convert_to_mzML create a mzML file from a h5 file in the same directory with the writeMLData of the MSnbase package

Usage
convert_to_mzML(file)
createPtrSet

Arguments

dir       A .h5 file path

Value

create a mzML file in the same directory of the h5 input file

Examples

library(ptairData)
filePathRaw <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5',
                          package = 'ptairData')
# write a mzml file in the same directory
convert_to_mzML(filePathRaw)
file_mzML <- gsub(".h5", ".mzML", filePathRaw)
file.remove(file_mzML)

createPtrSet

Description

This function creates a ptrSet-class S4 object. It opens each file and:

• performs an external calibration by using the mzCalibRef reference masses on the sum spectra
  every calibrationPeriod second
• quantifies the primary ion (H3O+ isotope by default) on the average total ion spectrum.
• calculates expiration on the mzBreathTracer trace. The part of the trace where the intensity is
  higher than fracMaxTIC * max(trace) is considered as expiration. If fracMaxTIC is different
to zero, this step is skipped
• defines the set of knots for the peak analysis (see detectPeak)
• provides a default sampleMetadata based on the tree structure of the directory and the acqui-
sition date (a data.frame with file names as row names)
• If saveDir is not NULL, the returned object will be saved as a .Rdata in saveDir with the
setName as name

Usage

createPtrSet(
  dir,
  setName,
  mzCalibRef = c(21.022, 29.013424, 41.03858, 60.0525, 203.943, 330.8495),
  calibrationPeriod = 60,
  fracMaxTIC = 0.8,
  mzBreathTracer = NULL,
  knotsPeriod = 3,
)
createPtrSet

mzPrimaryIon = 21.022,
  saveDir = NULL

Arguments

dir character. a directory path which contains several h5 files, possibly organized in
  subfolder

setName character. name of the ptrSet object. If 'saveDir' is not null, the object will be
  saved with this name.

mzCalibRef vector of the reference mass values; those masses should be accurate, and the
  corresponding peaks should be of high intensity and 'unique' in a nominal mass
  interval (without overlapping peaks) to performs calibration. See ?calibration.

calibrationPeriod in second, coefficient calibration are estimated for each sum spectrum of calibrationPeriod
  seconds

fracMaxTIC Fraction (between 0 and 1) of the maximum of the Total Ion Current (TIC)
  amplitude after baseline removal. Only the part of the spectrum where the TIC
  intensity is higher than 'fracMaxTIC * max(TIC)' will be analyzed. If you want
  to analyze the entire spectrum, set this parameter to 0.

mzBreathTracer integer: nominal mass of the Extracted Ion Current (EIC) used to compute the
  expiration time limits. If NULL, the limits will be computed on the Total Ion
  Current (TIC).

knotsPeriod period in second (time scale) between two knots for the two dimensional mod-
  eling

mzPrimaryIon Exact mass of the primary ion isotope

saveDir Directory where the ptrSet object will be saved as .RData. If NULL, nothing will
  be saved.

Value

a ptrSet object with slots:

• Parameter: list containing the parameters used for createPrtSet, detectPeak and alignTimePeriods
  functions.

• sampleMetadata: data frame containing information about the data, with file names in row
  names

• mzCalibRef: list containing for each file the masses used for the calibration (see ?ptairMS::calibration
  for more details)

• signalCalibRef: mz and intensity +/- 0.4Da around the calibration masses

• errorCalibPpm: list containing for each file the accuracy error in ppm at each calibration
  masses

• coefCalib: list containing the calibration coefficients 'a' and 'b' which enable to convert tof
  to mz for each file (see calibration function for more details.

• resolution: estimated resolution \( \frac{m}{\Delta m} \) for each calibration masses within each file
• TIC: The Total Ion Current for each file
• timeLimits: list containing, for each file, a list of two elements: the matrix of time limit for each file (if fracMaxTIC is different to zero), and the background index. See timeLimits for more details
• peakList: list containing for each file an expression set eSet, with m/z peak center, quantification for background and exhaled air in cps, ppb and ncps, and quantity for each time points. See getPeakList for more details.

Examples

library(ptairData)
directory <- system.file('extdata/mycobacteria', package = 'ptairData')
ptrSet<-createPtrSet(dir=directory,setName='ptrSet',
mzCalibRef=c(21.022,59.049),
fracMaxTIC=0.9,saveDir= NULL)

---

cumulative_fit_function

Create cumulative function fit

Description

Create cumulative function fit

Usage

cumulative_fit_function(fit_function_str, par_var_str, par_fix_str, n.peak)

Arguments

fit_function_str
  fit function who will be use in character
par_var_str
  parameters of fit function who change with the peak in a vector of character
par_fix_str
  parameters of fit function independent of the peak in a vector of character
n.peak
  number of peak

Value

a list:
  init.names: names of parameters for the initialization
  func.eval: function who will be fitted
**deadTimeCorr**

*Dead time correction on raw data*

**Description**

Dead time correction on raw data

**Usage**

```
deadTimeCorr(raw, ve, vne, r, threshold = 0.1)
```

**Arguments**

- `raw`: ptrRaw object
- `ve`: extending dead time
- `vne`: non extending dead time
- `r`: number of extraction
- `threshold`: only bin of intensity more than threshold*r which be corrected

**Value**

a ptrRaw object with the raw matrix corrected

**defineKnots**

*Define the knots location*

**Description**

defineKnots function determine the knots location for a ptrSet or ptrRaw object. There is three possibilities:

- method = expiration in the expiration periods, a knots is placed every knotsPeriod seconds, and 1 knots in the middle of two expiration, one at begin and at the end
- method= uniforme, the knots are placed uniformly every knotsPeriod time points
- give in knotsList a list of knot, with all base name file in name of the list element. All file must be informed. The knots location must be contained in the time axis
Usage

```r
defineKnots(
    object,
    knotsPeriod = 3,
    method = c("expiration", "uniform", "manual")[1],
    knotsList = NULL,
    ...
)
```

## S4 method for signature 'ptrRaw'
```r
defineKnots(
    object,
    knotsPeriod = 3,
    method = c("expiration", "uniform")[1],
    knotsList = NULL,
    timeLimit = list(NULL)
)
```

## S4 method for signature 'ptrSet'
```r
defineKnots(
    object,
    knotsPeriod = 3,
    method = c("expiration", "uniform")[1],
    knotsList = NULL
)
```

Arguments

- **object**: ptrSet object
- **knotsPeriod**: the period in second (times scale) between two knots for the two dimensional modelization
- **method**: expiration or uniform
- **knotsList**: a list of knot location for each files, with all base name file in name of the list element
- **timeLimit**: index time of the expiration limits and background

Value

- numeric vector of knots
- a list with numeric vector of knots for each file

Examples

```r
library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
```
detectPeak

mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL)

### placed knots every 2 times points
exhaledPtrset <- defineKnots(exhaledPtrset, knotsPeriod = 2, method = 'uniform')

### placed knots every 3 times points in the expiration (default)
exhaledPtrset <- defineKnots(exhaledPtrset, knotsPeriod = 3, method = 'expiration')

detectPeak  

Detection and quantification of peaks for a ptrSet object.

Description

The detectPeak function detects peaks on the average total spectrum around nominal masses, for all files present in ptrSet which have not already been processed. The temporal evolution of each peak is then evaluated by using a two-dimensional penalized spline regression. Finally, the expiration points (if defined in the ptrSet) are averaged, and a t-test is performed between expiration and ambient air. The peakList can be accessed with the getPeakList function, which returns the information about the detected peaks in each file as a list of ExpressionSet objects. The peak detection steps within each file are as follows:

for each nominal mass:

- correction of the calibration drift
- peak detection on the average spectrum
- estimation of temporal evolution
- t-test between expiration and ambient air

Usage

detectPeak(
  x, 
  ppm = 130, 
  minIntensity = 10, 
  minIntensityRate = 0.01, 
  mzNominal = NULL, 
  resolutionRange = NULL, 
  fctFit = NULL, 
  smoothPenalty = NULL, 
  parallelize = FALSE, 
  nbCores = 2, 
  saving = TRUE, 
  saveDir = getParameters(x)$saveDir, 
  ...
)

## S4 method for signature 'ptrRaw'
detectPeak(}
detectPeak(
  x,
  ppm = 130,
  minIntensity = 10,
  minIntensityRate = 0.01,
  mzNominal = NULL,
  resolutionRange = NULL,
  fctFit = NULL,
  smoothPenalty = NULL,
  timeLimit,
  knots = NULL,
  mzPrimaryIon = 21.022,
  ...
)

## S4 method for signature 'ptrSet'
detectPeak(
  x,
  ppm = 130,
  minIntensity = 10,
  minIntensityRate = 0.01,
  mzNominal = NULL,
  resolutionRange = NULL,
  fctFit = NULL,
  smoothPenalty = 0,
  parallelize = FALSE,
  nbCores = 2,
  saving = TRUE,
  saveDir = getParameters(x)$saveDir,
  ...
)

Arguments

x a `ptrSet` object

ppm minimum distance in ppm between two peaks

minIntensity minimum intensity for peak detection. The final threshold for peak detection will be: max(minIntensity, threshold noise). The threshold noise corresponds to max(max(noise around the nominal mass), minIntensityRate * max(intensity within the nominal mass))

minIntensityRate Fraction of the maximum intensity to be used for noise thresholding

mzNominal nominal masses at which peaks will be detected; if NULL, all nominal masses of the mass axis

resolutionRange vector with the minimum, average, and maximum resolution of the PTR instrument. If NULL, the values are estimated by using the calibration peaks.

fctFit function for the peak quantification: should be sech2 or averagePeak. If NULL, the best function is selected by using the calibration peaks.
detectPeak

smoothPenalty    second order penalty coefficient of the p-spline used for two-dimensional regression. If NULL, the coefficient is estimated by generalized cross validation (GCV) criteria
parallelize    Boolean. If TRUE, loops over files are parallelized
nbCores    number of cluster to use for parallel computation.
saving    boolean. If TRUE, the object will be saved in saveDir with the setName parameter of the createPtrSet function
saveDir    The directory where the ptrSet object will be saved as .RData. If NULL, nothing will be saved
...    may be used to pass parameters to the processFileTemporal function
timeLimit    index time of the expiration limits and background. Should be provided by timeLimits function
knots    numeric vector corresponding to the knot values, which used for the two dimensional regression for each file. Should be provided by defineKnots function
mzPrimaryIon    the exact mass of the primary ion isotope

Value

an S4 ptrSet object, that contains the input ptrSet completed with the peakLists.

References

Muller et al 2014, Holzinger et al 2015, Marx and Eilers 1992

Examples

## For ptrRaw object
library(ptairData)
filePath <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5', package = 'ptairData')
raw <- readRaw(filePath,mzCalibRef=c(21.022,59.049),calib=TRUE)
timeLimit<-timeLimits(raw,fracMaxTIC=0.7)
knots<-defineKnots(object = raw,timeLimit=timeLimit)
raw <- detectPeak(raw, timeLimit=timeLimit, mzNominal = c(21,59),
smoothPenalty=0,knots=knots,resolutionRange=c(2000,5000,8000))

## For a ptrSet object
library(ptairData)
directory <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset<createPtrSet(dir=directory,setName="exhaledPtrset",
mzCalibRef=c(21.022,59.049),
fracMaxTIC=0.9,saveDir= NULL)
exhaledPtrset <- detectPeak(exhaledPtrset)
peakListEset<-getPeakList(exhaledPtrset)
Biobase::fData(peakListEset[[1]])
Biobase::exprs(peakListEset[[1]])
determinePeakShape  \hspace{1cm} Determine peak shape from raw data in tof

**Description**

This function uses the method described by average and al 2013, for determining a peak shape from the raw data:

$$peak_i(Delta_i, A_i, t_i) = \text{interpolation}(x = \text{tof.ref} \times \Delta_i + t_i, y = A_i \times \text{peak.ref}, xout= \text{TOF}_i)$$

where peak.ref and tof.ref are peaks reference used for mass calibration.

**Usage**

```r
determinePeakShape(raw, plotShape = FALSE)
```

**Arguments**

- `raw` a `ptrRaw-class` object
- `plotShape` if true plot each reference peak and the average peak (the peak shape)

**Value**

A list of two vectors which are the reference peak normalized tof and intensity.

---

exportSampleMetada  \hspace{1cm} export sampleMetadata

**Description**

export sampleMetadata

**Usage**

```r
exportSampleMetada(set, saveFile)
```

**Arguments**

- `set` a `ptrSet` object
- `saveFile` a file path in tsv extension where the data.frame will be exported

**Value**

nothing
extractEIC

**Examples**

```r
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
saveFile<-file.path(getwd(),'sampleMetadata.tsv')
#exportSampleMetadata(exhaledPtrset ,saveFile)
```

**extractEIC**

*extract all raw EIC from a pre-defined peak List*

**Description**

extract all raw EIC from a pre-defined peak List

**Usage**

```r
extractEIC(raw, peak, peakQuantil = 0.01, fctFit = "sech2")
```

**Arguments**

- `raw` ptrRaw object
- `peak` a data.frame with a column named 'Mz'. The Mz of the VOC detected
- `peakQuantil` the quantile of the peak shape to determine the borne of the EIC
- `fctFit` function used to fit peak

**Value**

list containing all EIC and the Mz borne for all peak

**fit_averagePeak**

*Fit peak with average function*

**Description**

Fit peak with average function

**Usage**

```r
fit_averagePeak(initTof, l.shape, sp, bin, lower.cons, upper.cons)
```
Arguments

- `initTof`: list of initialisation in tof
- `l.shape`: peak shape average
- `sp`: spectrum
- `bin`: tof axis
- `lower.cons`: lower constrain for fit
- `upper.cons`: upper constrain for fit

Value

- list with fit information

---

**Description**

fit function average

**Usage**

```r
fit_averagePeak_function(t, delta, h, intervRef, peakShape, bin)
```

Arguments

- `t`: tof center of peak
- `delta`: FWHM of peak
- `h`: peak height
- `intervRef`: reference interval for peak shape
- `peakShape`: peak shape estimated on intervalRef
- `bin`: bin interval of peak will be fitted

Value

- peak function made on an average of reference peaks normalized
**formula2mass**

*Compute exact mass.*

---

**Description**

Compute exact mass from an elemental formula

**Usage**

```r
formula2mass(formula.vc, protonate.l = TRUE)
```

**Arguments**

- `formula.vc`: Vector of molecular formulas.
- `protonate.l`: Should a proton be added to the formula?

**Value**

Vector of the corresponding (protonated) masses.

**Examples**

```r
formula2mass("CO2")
```

---

**getDirectory**

*get the files directory of a ptrSet*

---

**Description**

get the files directory of a ptrSet

**Usage**

```r
getDirectory(ptrSet)
```

**Arguments**

- `ptrSet`: ptrSet object

**Value**

the directory in absolute path as character
Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
getDirectory(exhaledPtrset )

getFileNames

get the file names containing in the directory of a ptrSet or ptrRaw

Description

get the file names containing in the directory of a ptrSet or ptrRaw
get the file names containing in the directory of a ptrSet

Usage

getFileNames(object, fullNames = FALSE)

## S4 method for signature 'ptrSet'
getFileNames(object, fullNames = FALSE)

## S4 method for signature 'ptrRaw'
getFileNames(object, fullNames = FALSE)

Arguments

object ptrSet object
fullNames logical: if TRUE, it return the the directory path is prepended to the file names.

Value

a vector of character that contains all file names

Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
getFileNames(exhaledPtrset )
getPeakList

get the peak list of a ptrSet object

Description

get the peak list of a ptrSet object

Usage

getPeakList(set)

Arguments

set ptrSet object

Value

a list of expressionSet, where:

- assay Data contains the matrix with m/z peak center in row names, and the quantification in cps at each time point
- feature Data the matrix with m/z peak center in row names, and the following columns:
  - quanti_unit: the mean of the quantification over the expiration/headspace time limits defined
  - background_unit: the mean of the quantification over the background time limits defined
  - diffAbs_unit: the mean of the quantification over the expiration/headspace time limits defined after subtracting the baseline estimated from the background points defined
  - pValLess/ pValGreater: the p-value of the unilateral t-test, who test that quantification (in cps) of expiration points are less/greater than the intensity of the background.
  - lower/upper: integration boundaries
  - parameter peak: estimated peak parameter

Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mz=c(21,59))
peakList<- getPeakList(exhaledPtrset)
X<-Biobase::exprs(peakList[[1]])
Y<- Biobase::fData(peakList[[1]])
head(Y)
**getSampleMetadata**  
*get sampleMetadata of a ptrSet*

**Description**  
get sampleMetadata of a ptrSet

**Usage**  
getSampleMetadata(set)

**Arguments**  
set 
a ptrSet object

**Value**  
a data.frame

**Examples**  

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")  
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",  
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL)  
SMD<-getSampleMetadata(exhaledPtrset)

---

**importSampleMetadata**  
*import a sampleMetadata from a tsv file to a ptrSet object*

**Description**  
import a sampleMetadata from a tsv file to a ptrSet object

**Usage**  
importSampleMetadata(set, file)

**Arguments**  
set 
a ptrSet object

file 
a tsv file contains the sample metadata to import, with all file names in row name (the first column on the excel).

**Value**  
a ptrSet with the new sample Metadata
Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
saveFile<-file.path(getwd(),'sampleMetadata.tsv')
#exportSampleMetadata(exhaledPtrset,saveFile)
#exhaledPtrset<-importSampleMetadata(exhaledPtrset,saveFile)

impute Imputes the missing values

Description

Imputes missing values by returning back to the raw data and fitting the peak shape function on the noise (or on the residuals signals if peaks have already been detected).

Usage

impute(eSet, ptrSet, parallelize = FALSE, nbCores = 2)

Arguments

eSet ExpressionSet returned by the alignSamples function
ptrSet ptrSet-class object processed by the detectPeak function
parallelize boolean. If TRUE, the loop over all files will be parallelized
nbCores number of clusters to use for parallel computation.

Value

the same ExpressionSet as in input, with the imputed missing values in the assayData slot

Examples

library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw,
setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525),
fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mz=c(21,52))
eSet <- alignSamples(exhaledPtrset,mz=c(21,52))
Biobase::exprs(eSet)
eSet <- impute(eSet,exhaledPtrset)
Biobase::exprs(eSet)
imputeMat

**Impute missing values on an matrix set from an ptrSet**

**Description**

Imputing missing values by returning back to the raw data and fitting the peak shape function on the noise / residuals

**Usage**

```r
imputeMat(X, ptrSet, quantiUnit)
```

**Arguments**

- **X**
  - the peak table matrix with missing values
- **ptrSet**
  - processed by detectPeak function
- **quantiUnit**
  - the unit of the quantities in the matrix X (ppb, cps or ncps)

**Value**

the same matrix as in input, with missing values imputing

**Examples**

```r
library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw,
set_Name="exhaledPtrset", mz_Calib_Ref = c(21.022, 60.0525),
frac_Max_TIC = 0.7, save_Dir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mz=c(21,52))
eSet <- alignSamples(exhaledPtrset,pValGreaterThres=0.05,fracGroup=0)
X <-Biobase::exprs(eSet)
X <- imputeMat(X,exhaledPtrset,quantiUnit='ppb')
plotFeatures(exhaledPtrset,mz = 52.047,typePlot = "ggplot",colorBy = "subfolder")
```

---

**initializeFit**

**initialization for apply fit function in the spectrum**

**Description**

initialization for apply fit function in the spectrum
**initializeFit**

**Usage**

```
initializeFit(
  i,
  sp.i.fit,
  sp.i,
  mz.i,
  calibCoef,
  resmean,
  minpeakheight,
  noiseacf,
  ppmPeakMinSep,
  daSeparation,
  d,
  plotAll,
  c
)
```

**Arguments**

- **i** the nominal mass
- **sp.i.fit** the vector who will be fetted (spectrum pf residual)
- **sp.i** the spectrum around a nominal mass
- **mz.i** the mass vector around a nominal mass
- **calibCoef** calibration coefficient
- **resmean** resolution m/delta(m) mean
- **minpeakheight** the minimum peak intensity
- **noiseacf** aytocorelation of the noise
- **ppmPeakMinSep** the minimum distance between two peeks in ppm
- **daSeparation** the minimum distance between two peeks in da
- **d** the degree of savitzky golay filter
- **plotAll** bollean if TRUE, it plot all the initialiation step
- **c** the number of current itteration

**Value**

a list with fit input
LocalMaximaSG  
*Find local maxima with Savitzky Golay filter*

**Description**
Apply Savitzky Golay filter to the spectrum and find local maxima such that: second derivate Savitzky Golay filter < 0 and first derivate = 0 and intensity > minPeakHeight

**Usage**
LocalMaximaSG(sp, minPeakHeight = -Inf, noiseacf = 0.1, d = 3)

**Arguments**
- **sp** the array of spectrum values
- **minPeakHeight** minimum intensity of a peak
- **noiseacf** autocorrelation of the noise
- **d** the degree of Savitzky Golay filter, default 3

**Value**
array with peak's index in the spectrum

**Examples**
spectrum<-dnorm(x=seq(-5,5,length.out = 100))
index.max<-LocalMaximaSG(spectrum)

makeSubGroup  
*Use in align function. return a peak group thanks to kernel gaussian density in a peak matrix.*

**Description**
Use in align function. return a peak group thanks to kernel gaussian density in a peak matrix.

**Usage**
makeSubGroup(subpeakl, den, plim)

**Arguments**
- **subpeakl** a matrix with mz, ppb, background and group column.
- **den** the kernel gaussian density estimated on subpeakl
- **plim** the limit of a peak in the density of the group who will be fromed
**OptimalWindowsSG**

*Find optimal window’s size for Savitzky Golay filter*

**Value**

the sub peakgroup

**Description**

Find optimal window’s size for Savitzky Golay filter

**Usage**

OptimalWindowsSG(sp, noiseacf, d = 3)

**Arguments**

- `sp`: the array of spectrum values
- `noiseacf`: autocorrelation of the noise
- `d`: the degree of Savitzky Golay filter

**Value**

the optimal size of Savitzky Golay filter’s windows

**PeakList**

*Detection and quantification of peaks on a sum spectrum.*

**Description**

Detection and quantification of peaks on a sum spectrum.

**Usage**

PeakList(
  raw,
  mzNominal = unique(round(getRawInfo(raw)$mz)),
  ppm = 130,
  resolutionRange = c(3000, 5000, 8000),
  minIntensity = 5,
  fctFit = c("sech2", "averagePeak")[1],
  peakShape = NULL,
  maxIter = 1,
  R2min = 0.995,
  autocorNoiseMax = 0.3,
  plotFinal = FALSE,


```r
plotAll = FALSE,
thNoiseRate = 1.1,
minIntensityRate = 0.01,
countFacFWHM = 10,
daSeparation = 0.005,
d = 3,
windowSize = 0.4
)
```

## S4 method for signature 'ptrRaw'
PeakList(
  raw,
  mzNominal = unique(round(getRawInfo(raw)$mz)),
  ppm = 130,
  resolutionRange = c(300, 5000, 8000),
  minIntensity = 5,
  fctFit = c("sech2", "averagePeak")[1],
  peakShape = NULL,
  maxIter = 3,
  R2min = 0.995,
  autocorNoiseMax = 0.3,
  plotFinal = FALSE,
  plotAll = FALSE,
  thNoiseRate = 1.1,
  minIntensityRate = 0.01,
  countFacFWHM = 10,
  daSeparation = 0.005,
  d = 3,
  windowSize = 0.4
)

### Arguments

- **raw** `ptrRaw-class` object
- **mzNominal** the vector of nominal mass where peaks will be detected
- **ppm** the minimum distance between two peaks in ppm
- **resolutionRange** vector with resolution min, resolution Mean, and resolution max of the PTR
- **minIntensity** the minimum intenisty for peaks detection. The final threshold for peak detection will be: \( \text{max (minPeakDetect, thresholdNoise)} \). The threshold-Noise correspond to \( \text{max(thNoiseRate * max(noise around the nominal mass), minIntensityRate * max(intensity in the nominal mass))} \). The noise around the nominal mass correspond: \([m-windowSize-0.2,m-windowSize]U[m+windowSize,m+WindowSize+0.2]\)
- **fctFit** the function for the quantification of Peak, should be average or Sech2
- **peakShape** a list with reference axis and a reference peak shape centered in zero
- **maxIter** maximum iteration of residual analysis
- **R2min** R2 minimum to stop the iterative residual analysis
the autocorelation threshold for Optimal windows Savitzky Golay filter in `OptimalWindowSG` ptairMS function. See `?OptimalWindowSG`.

- `plotFinal` boolean. If TRUE, plot the spectrum for all nominal masses, with the final fitted peaks.
- `plotAll` boolean. If TRUE, plot all steps to get the final fitted peaks.
- `thNoiseRate` The rate which multiplies the max noise intensity.
- `minIntensityRate` The rate which multiplies the max signal intensity.
- `countFacFWHM` integer. We will sum the fitted peaks on a window’s size of `countFacFWHM * FWHM`, centered in the mass peak center.
- `daSeparation` the minimum distance between two peaks in Da for nominal mass < 17.
- `d` the degree for the Savitzky Golay filter.
- `windowSize` peaks will be detected only around m - `windowSize`; m + `windowSize`, for all m in `mzNominal`.

### Value

A list containing:

- `peak`: a data.frame, with for all peak detected: the mass center, the intensity count in cps, the peak width (delta_mz), correspond to the Full Width Half Maximum (FWHM), the resolution m/delta_m, the other parameters values estimated of `fitFunc`.
- `warnings`: warnings generated by the peak detection algorithm per nominal masses.
- `infoPlot`: elements needed to plot the fitted peak per nominal masses.

### Examples

```r
library(ptairData)

# reading sample data, saving the file path
filePath <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5', package = 'ptairData')
file <- readRaw(filePath)

# creating peak list
peakList <- PeakList(file, mzNominal = c(21,63))
peakList$peak

# plotting the peak list
plot(peakList)
```

---

### Description

`plot` a `ptrSet` object

---

### plot,ptrSet,ANY-method

Plot a `ptrSet` object.
Usage

## S4 method for signature 'ptrSet,ANY'
plot(x, y, typePlot = "")

Arguments

x a ptrSet object
y not use
typePlot could be: calibError, resolution, peakShape, or a empty character if you want all.

Value

plot

Examples

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
plot(exhaledPtrset )
plot(exhaledPtrset , typePlot = 'calibError')
plot(exhaledPtrset , typePlot = 'resolution')
plot(exhaledPtrset , typePlot = 'peakShape')

plotCalib

Plot the calibration peaks after calibration

Description

Plot the calibration peaks after calibration

Usage

plotCalib(object, ppm = 2000, ...)

## S4 method for signature 'ptrRaw'
plotCalib(object, ppm = 2000, ...)

## S4 method for signature 'ptrSet'
plotCalib(object, ppm = 2000, pdfFile = NULL, fileNames = NULL, ...)
**plotFeatures**

**Arguments**

- **object**: a ptrSet or ptrRaw object
- **ppm**: the width of plot windows
- **...**: not used
- **pdfFile**: is different of NULL, the file path to save the plots in pdf
- **fileNames**: the name of the files in the ptrSet object to plot. If NULL, all files will be plotted

**Value**

plot

**Examples**

```r
## ptrSet
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
plotCalib(exhaledPtrset ,fileNames=getFileNames(exhaledPtrset )[1])

## ptrRaw
filePath<-system.file(  
'extdata/exhaledAir/ind1/ind1-1.h5',  
package = 'ptairData')
raw <- readRaw(filePath,mzCalibRef=c(21.022,59.049))
plotCalib(raw)
```

---

**Plot raw average spectrum around a mzRange**

**Description**

Plot the raw data spectrum for several files in a ptrSet object around the mz masses. The expiration average spectrum is in full lines, and background in dashed lines.

**Usage**

```r
plotFeatures(  
  set,  
  mz,  
  typePlot = "plotly",  
  addFeatureLine = TRUE,  
  ppm = 2000,  
  pdfFile = NULL,  
  fileNames = NULL,  
  colorBy = "rownames"
)
```
## S4 method for signature 'ptrSet'
plotFeatures(
  set,
  mz,
  typePlot = "plotly",
  addFeatureLine = TRUE,
  ppm = 2000,
  pdfFile = NULL,
  fileNames = NULL,
  colorBy = "rownames"
)

### Arguments

- **set**: a 
  ptrSet-class object
- **mz**: the mz values to plot
- **typePlot**: set "plotly" to get an interactive plot, or "ggplot"
- **addFeatureLine**: boolean. If TRUE a vertical line at the mz masses is plotted
- **ppm**: windows size of the plot round mz in ppm
- **pdfFile**: a file path to save a pdf with a individual plot per file
- **fileNames**: vector of character. The file names you want to plot. If NULL, it plot all files
- **colorBy**: character. A column name of sample metadata by which the line are colored.

### Value

a plotly or ggplot2 object

### Examples

```r
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw,
  setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525),
  fracMaxTIC = 0.7, saveDir = NULL )
plotF<-plotFeatures(exhaledPtrset,mz=59.049,type="ggplot",colorBy="subfolder")
print(plotF)
```

---

## plotPeakShape

plot the average peak shape of reference calibration masses for a ptrSet

### Description

plot the average peak shape of reference calibration masses for a ptrSet

### Usage

```r
plotPeakShape(set, showAverage = FALSE)
```
Arguments

- set: ptrSet object
- showAverage: boolean

Value

- ggplot object

Description

Displays the image of the matrix of intensities, the TIC and the TIS, for the selected m/z and time ranges

Usage

```r
plotRaw(
  object,
  mzRange,
  timeRange = c(NA, NA),
  type = c("classical", "plotly")[1],
  ppm = 2000,
  palette = c("heat", "revHeat", "grey", "revGrey", "ramp")[1],
  showVocDB = TRUE,
  figure.pdf = "interactive",
  ...
)
```

## S4 method for signature 'ptrRaw'
```r
plotRaw(
  object,
  mzRange,
  timeRange = c(NA, NA),
  type = c("classical", "plotly")[1],
  ppm = 2000,
  palette = c("heat", "revHeat", "grey", "revGrey", "ramp")[1],
  showVocDB = TRUE,
  figure.pdf = "interactive",
  ...
)
```

## S4 method for signature 'ptrSet'
```r
plotRaw(
  object,
  ...
)
```
plotRaw

mzRange, timeRange = c(NA, NA),
type = c("classical", "plotly")[1],
ppm = 2000,
palette = c("heat", "revHeat", "grey", "revGrey", "ramp")[1],
showVocDB = TRUE,
figure.pdf = "interactive",
fileNames = NULL,
...
)

Arguments

object An S4 object of class ptrRaw-class or ptrSet
mzRange Either a vector of 2 numerics indicating the m/z limits or an integer giving a
nominal m/z
timeRange Vector of 2 numerics giving the time limits
type Character: plot type; either 'classical' [default] or 'plotly'
ppm Integer: Half size of the m/z window when mzRange is set to a nominal mass
palette Character: Color palette for the 'classical' plot; either 'heat' [default], 'revHeat',
'grey', 'revGrey' or 'ramp'
showVocDB Logical: Should putative m/z annotations from the internal package database be
displayed (default is TRUE)
figure.pdf Character: Either 'interactive' [default], or the filename of the figure to be saved
(with the 'pdf' extension); only available for the 'classical' display
... not used
fileNames vector of character. The file names of the ptrSer that you want to plot. Could be
in basename or fullname.

Value

Invisibly returns a list of the raw (sub)matrix 'rawsubM' and the voc (sub)database 'vocsubDB'

Examples

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
ptairMS::plotRaw(exhaledPtrset ,mzRange=59,fileNames=’ind1-1.h5’)

patientRaw <- ptairMS::readRaw(system.file("extdata/exhaledAir/ind1/ind1-1.h5", package = "ptairData"),
mzCalibRef=c(21.022,59.049,75.05))
ptairMS::plotRaw(patientRaw, mzRange = 59)
ptairMS::plotRaw(patientRaw, mzRange = 59, type = 'plotly')
plotTIC

plot the Total Ion spectrum (TIC) for one or several files.

Description

plot the Total Ion spectrum (TIC) for one or several files.

Usage

plotTIC(
  object,
  type = c("plotly", "ggplot")[1],
  baselineRm = FALSE,
  showLimits = FALSE,
  ...
)

## S4 method for signature 'ptrRaw'
plotTIC(object, type, baselineRm, showLimits, fracMaxTIC = 0.8, ...)

## S4 method for signature 'ptrSet'
plotTIC(
  object,
  type,
  baselineRm,
  showLimits,
  pdfFile = NULL,
  fileNames = NULL,
  colorBy = "rownames",
  normalizePrimariIon = FALSE,
  ...
)

Arguments

object ptrSet or ptrRaw S4 object
type set 'plotly' to get an interactive plot, and 'ggplot' for classical plot.
baselineRm logical. If TRUE, remove the baseline of the TIC
showLimits logical. If TRUE, add the time limits to the plot (obtain with the 'fracMaxTIC' argument or 'createPtrSet' function)
... not used
fracMaxTIC Percentage (between 0 and 1) of the maximum of the Total Ion Current (TIC) amplitude with baseline removal. We will analyze only the part of the spectrum where the TIC intensity is higher than 'fracMaxTIC * max(TIC)'. If you want to analyze the entire spectrum, set this parameter to 0.
ptrRaw-class

PTR-TOF-MS raw data from a rhdf5 file

Description

A ptrRaw object contains PTR-TOF-MS raw data from one rhdf5 file. It is created with the readRaw function.

Slots

- **name**: the file name
- **rawM**: the raw intensities matrix
- **mz**: array of the m/z axis
- **time**: numeric vector of acquisition time (in seconds)
- **calibMzToTof**: function to convert m/z to ToF
- **calibTofToMz**: function to convert tof to m/z
- **calibCoef**: calibration coefficients (a,b) such that: \( m/z = ((tof-b)/a)^2 \) for each calibration period
- **indexTimeCalib**: index time of each calibration period

Value

a plotly or ggplot2 object.

Examples

```r
### ptrRaw object

library(ptairData)
filePath <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5', package = 'ptairData')
raw <- readRaw(filePath)
p <- plotTIC(raw)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
plotTIC(exhaledPtrset, type="ggplot")
```
ptrSet-class

A set of PTR-TOF-MS raw data informations

Description

A ptrSet object is related to a directory that contains several PTR-TOF-MS raw data in rhdf5 format. It is created with the createPtrSet function. This object could be updated when new files are added with the updatePtrSet function.

Slots

- parameter the input parameters value of the function createPtrSet and detectPeak
- sampleMetadata dataframe of sample metadata, with file names in row names, suborders names and acquisition date in columns
- date acquisition date for each file
- mzCalibRef the masses used for calibration for each file
- signalCalibRef the spectrum of mass calibration for each file
- errorCalibPpm the calibration error for each file
- coefCalib the coefficients of mass axis calibration of each calibration periods for each file
- indexTimeCalib index time of each calibration period for each file
- primaryIon the quantity in count per acquisition time of the isotope of primary ion for each file
- resolution estimation of the resolution for each file based on the calibration reference masses
- prinRecAct drift information (temperature, pressure and voltage)
- ptrTransmission transmission curve for each file

References

https://www.hdfgroup.org
TIC  the total ion current (TIC) for each file
breathTracer  the tracer for expiration/head spaces detection
timeLimit  the index of time limit for each file
knots  numeric vector correspond to the knot that will be used for the two dimensional regression for each file
fctFit  the peak function used for peak deconvolution for each file
peakShape  average normalized peak shape of the calibration peak for each file
peakList  individual peak list in eSet

---

**Description**

`readRaw` reads a h5 file with rhdf5 library, and calibrates the mass axis with `mzCalibRef` masses each `calibrationPeriod` seconds. It returns a `ptrRaw-class` S4 object, that contains raw data.

**Usage**

```r
readRaw(
  filePath,
  calib = TRUE,
  mzCalibRef = c(21.022, 29.013424, 41.03858, 60.0525, 203.943, 330.8495),
  calibrationPeriod = 60,
  tolCalibPpm = 70,
  maxTimePoint = 900
)
```

**Arguments**

- **filePath**: h5 absolute file path full name.
- **calib**: boolean. If true, an external calibration is performed on the `calibrationPeriod` sum spectrum with `mzCalibRef` reference masses.
- **mzCalibRef**: calibration parameter. Vector of exact theoretical masses values of an intensive peak without overlapping.
- **calibrationPeriod**: in second, coefficient calibration are estimated for each sum spectrum of `calibrationPeriod` seconds.
- **tolCalibPpm**: calibration parameter. The maximum error tolerated in ppm. A warning appears for error greater than `tolCalibPpm`.
- **maxTimePoint**: number maximal of time point to read.
Value

a ptrRaw object, including slot

- rawM the data raw matrix, in count of ions
- mz the mz axis
- time time acquisition in second

Examples

```r
library(ptairData)
filePathRaw <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5', package = 'ptairData')
raw <- readRaw(filePath=filePathRaw, mzCalibRef=c(21.022, 60.0525), calib=FALSE)
```

**resetSampleMetadata**

reset the default sampleMetadata, containing the suborders names and the acquisition dates as columns.

Description

reset the default sampleMetadata, containing the suborders names and the acquisition dates as columns.

Usage

```r
resetSampleMetadata(ptrset)
```

Arguments

- `ptrset` a ptrser object

Value

a data.frame

Examples

```r
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset", 
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
SMD<- resetSampleMetadata(exhaledPtrset)
```
**Description**

This function is useful when you want to change the parameters of the detect peak function. First delete the peakList with `rmPeakList`, and apply `detectPeak` with the new parameters.

**Usage**

```r
rmPeakList(object)
```

**Arguments**

- `object`: ptrSet object

**Value**

a ptrSet

**Examples**

```r
library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset <- rmPeakList(exhaledPtrset )
```

---

**RunShinnyApp**

*Graphical interface of ptairMS workflow*

**Description**

The whole workflow of ptairMS can be run interactively through a graphical user interface, which provides visualizations (expiration phases, peaks in the raw data, peak table, individual VOCs), quality controls (calibration, resolution, peak shape and evolution of reagent ions depending on time), and exploratory data analysis.

**Usage**

```r
RunShinnyApp()
```

**Value**

Shinny app
setSampleMetadata

Examples

## Not run: RunShinyApp()

--

setSampleMetadata set sampleMetadata in a ptrSet

Description

Insert a sample metadata data.frame in a ptrSet object. The dataframe must have all file names in rownames.

Usage

setSampleMetadata(set, sampleMetadata)

Arguments

set a ptrSet object

sampleMetadata a data.frame with all file names of the ptrSet in row names

Value

the ptrSet object in argument with the sampleMetadata modified

Examples

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset", mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
SMD<-getSampleMetadata(exhaledPtrset )
colnames(SMD)[1]<-‘individual’
exhaledPtrset<-setSampleMetadata(exhaledPtrset ,SMD)

--

show,ptrRaw-method show a ptrRaw object

Description

It indicates the files, the mz range, time acquisition range, and calibration error.

Usage

## S4 method for signature 'ptrRaw'
show(object)
Arguments
    object       a ptrRaw object

Value
    nothing

Description
    It indicates the directory, the number of files that contain the directory at the moment, and the
    number of processed files. The two numbers are different, use updatePtrSet function.

Usage
    ## S4 method for signature 'ptrSet'
    show(object)

Arguments
    object       a ptrSet object

Value
    nothing

Description
    Baseline estimation

Usage
    snipBase(sp, widthI = 11, iteI = 5)

Arguments
    sp           an array with spectrum values
    widthI       width of interval
    iteI         number of iteration

Value
    baseline estimation of the spectrum
timeLimits

Calculates time limits on the breath tracer

Description
This function derives limits on the breath tracer indicated, where the intensity is greater than fracMaxTIC*max(tracer). By setting fracMaxTIC close to 1, the size of the limits will be restricted. This function also determines the index corresponding to the background, where variation between two successive points can be controlled with derivThreshold parameter.

Usage

timeLimits(
  object,
  fracMaxTIC = 0.5,
  fracMaxTICBg = 0.2,
  derivThresholdExp = 1,
  derivThresholdBg = 0.05,
  mzBreathTracer = NULL,
  minPoints = 2,
  degreeBaseline = 1,
  baseline = TRUE,
  redefineKnots = TRUE,
  plotDel = FALSE
)

## S4 method for signature 'ptrRaw'
timeLimits(
  object,
  fracMaxTIC = 0.6,
  fracMaxTICBg = 0.2,
  derivThresholdExp = 0.5,
  derivThresholdBg = 0.05,
  mzBreathTracer = NULL,
  minPoints = 2,
  degreeBaseline = 1,
  baseline = TRUE,
  redefineKnots = TRUE,
  plotDel = FALSE
)

## S4 method for signature 'ptrSet'
timeLimits(
  object,
  fracMaxTIC = 0.5,
  fracMaxTICBg = 0.2,
  derivThresholdExp = 1,
```r
derivThresholdBg = 0.05,
minPoints = 2,
degreeBaseline = 1,
baseline = TRUE,
redefineKnots = TRUE,
plotDel = FALSE
)
```

**Arguments**

- `object`: a ptrRaw or ptrSet object
- `fracMaxTIC`: between 0 and 1. Percentage of the maximum of the tracer amplitude with baseline removal. If you want a finer limitation, increase `fracMaxTIC`, indeed decrease `fracMaxTICBg` same as `fracMaxTIC` but for background detection (lower than `fracMaxTIC`*max(TIC))
- `derivThresholdExp`: the threshold of the difference between two successive points of the expiration
- `derivThresholdBg`: the threshold of the difference between two successive points of the background
- `mzBreathTracer`: NULL or a integer. Correspond to a nominal masses of Extract Ion Current (EIC) whose limits you want to compute. If NULL, the limits are calculated on the Total Ion Current (TIC).
- `minPoints`: minimum duration of an expiration (in index).
- `degreeBaseline`: the degree of polynomial baseline function
- `baseline`: logical, should the trace be baseline corrected?
- `redefineKnots`: logical, should the knot location must be redefined with the new times limits?
- `plotDel`: boolean. If TRUE, the trace is plotted with limits and threshold.

**Value**

a list with expiration limits (a matrix of index, where each column correspond to one expiration, the first row it is the beginning and the second the end, or NA if no limits are detected) and index of the background.

**Examples**

```r
## ptrRaw object
library(ptairData)
filePath <- system.file('extdata/exhaledAir/ind1', 'ind1-1.h5',
package = 'ptairData')
raw <- readRaw(filePath)
timLim <- timeLimits(raw, fracMaxTIC=0.9, plotDel=TRUE)
timLim_acetone <- timeLimits(raw, fracMaxTIC=0.5, mzBreathTracer = 59, plotDel=TRUE)
```
TransmissionCurve

Estimation of the transmission curve

Description

Estimation of the transmission curve

Usage

TransmissionCurve(x, y)

Arguments

x: masses
y: transmission data

Value

a numeric vector

updatePtrSet

update a ptrSet object

Description

When new files are added to a directory which has already a ptrSet object associated, run updatePtrSet to add the new files in the object. The information on the new files are added to object with the same parameter used for the function createPtrSet who has created the object. updatePtrSet also delete from the ptrSet deleted files in the directory.

Usage

updatePtrSet(ptrset)

Arguments

ptrset: a ptrset object

Value

teh same ptrset object than ininput, but completed with new files and without deleted files in the directory
Examples

dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
##add or delete files in the directory
# exhaledPtrset<- updatePtrSet(exhaledPtrset)

width

Calculate the FWHM (Full Width at Half Maximum) in raw data

Description

Calculate the FWHM (Full Width at Half Maximum) in raw data

Usage

width( tof, peak, fracMaxTIC = 0.5)

Arguments

tof A vector of tof interval
peak A vector of peak Intensity
fracMaxTIC the fraction of the maximum intensity to compute the width

Value

the delta FWHM in tof

writeEset

Exporting an ExpressionSet instance into 3 tabulated files 'dataMatrix.tsv', sampleMetadata.tsv', and 'variableMetadata.tsv'

Description

Note that the dataMatrix is transposed before export (e.g., the samples are written column wise in the 'dataMatrix.tsv' exported file).

Usage

writeEset(x, dirName, overwrite = FALSE, verbose = TRUE)

## S4 method for signature 'ExpressionSet'
writeEset(x, dirName, overwrite = FALSE, verbose = TRUE)
writeEset

Arguments

- **x**: An S4 object of class ExpressionSet
- **dirName**: Character: directory where the tables should be written
- **overwrite**: Logical: should existing files be overwritten?
- **verbose**: Logical: should messages be printed?

Value

No object returned.

Examples

```r
library(ptairData)
dirRaw <- system.file("extdata/exhaledAir", package = "ptairData")
exhaledPtrset <- createPtrSet(dir=dirRaw, setName="exhaledPtrset",
mzCalibRef = c(21.022, 60.0525), fracMaxTIC = 0.7, saveDir = NULL )
exhaledPtrset<-detectPeak(exhaledPtrset,mz=c(21,59,77))
eset <- ptairMS::alignSamples(exhaledPtrset )
writeEset(eset, dirName = file.path(getwd(), "processed_dataset"))
unlink(file.path(getwd(), "processed_dataset"),recursive = TRUE)
```
Index

* internal
  align, 4
  calibrationFun, 10
cumulative_fit_function, 14
deadTimeCorr, 15
determinePeakShape, 20
extractEIC, 21
fit_averagePeak, 21
fit_averagePeak_function, 22
initializeFit, 28
makeSubGroup, 30
OptimalWindowsSG, 31
plotPeakShape, 36
ptairMS-package, 3
snipBase, 46
TransmissionCurve, 49
width, 50
aggregate, 4
align, 4
alignSamples, 5, 27
alignSamples, ptrSet-method (alignSamples), 5
annotateVOC, 7
annotateVOC, data.frame-method (annotateVOC), 7
annotateVOC, ExpressionSet-method (annotateVOC), 7
annotateVOC, numeric-method (annotateVOC), 7
calibration, 9, 13
calibration, ptrRaw-method (calibration), 9
calibration, ptrSet-method (calibration), 9
calibrationFun, 10
defaultTimeLimits, 11
defaultTimeLimits, ptrSet-method (defaultTimeLimits), 11
cumulative_fit_function, 14
deadTimeCorr, 15
defineKnots, 15, 19
defineKnots, ptrRaw-method (defineKnots), 15
defineKnots, ptrSet-method (defineKnots), 15
detectPeak, 6, 12, 17, 27, 41
detectPeak, ptrRaw-method (detectPeak), 17
detectPeak, ptrSet-method (detectPeak), 17
determinePeakShape, 20
eSet, 14, 41, 42
exportSampleMetadata, 20
ExpressionSet, 5, 6
extractEIC, 21
fit_averagePeak, 21
fit_averagePeak_function, 22
formula2mass, 23
gDirectory, 23
gFileNames, 24
getFileNames, ptrRaw-method (getFileNames), 24
getFileNames, ptrSet-method (getFileNames), 24
gPeakList, 14, 17, 25
gSampleMetadata, 26
importSampleMetadata, 26
impute, 27
imputeMat, 28
initializeFit, 28
LocalMaximaSG, 30
makeSubGroup, 30
OptimalWindowsSG, 31
PeakList, 31
PeakList, ptrRaw-method (PeakList), 31
plot, ptrSet, ANY-method, 33
plot, ptrSet-method
  (plot, ptrSet, ANY-method), 33
plot ptrSet (plot, ptrSet, ANY-method), 33
plotCalib, 34
plotCalib, ptrRaw-method (plotCalib), 34
plotCalib, ptrSet-method (plotCalib), 34
plotFeatures, 35
plotFeatures, ptrSet-method
  (plotFeatures), 35
plotPeakShape, 36
plotRaw, 37
plotRaw, ptrRaw-method (plotRaw), 37
plotRaw, ptrSet-method (plotRaw), 37
plotTIC, 39
plotTIC, ptrRaw-method (plotTIC), 39
plotTIC, ptrSet-method (plotTIC), 39
ptairMS (ptairMS-package), 3
ptairMS-package, 3
ptrRaw-class, 40
ptrSet, 18
ptrSet-class, 41
readRaw, 40, 42
resetSampleMetadata, 43
rmPeakList, 44
RunShinyApp, 44
setSampleMetadata, 45
show, ptrRaw-method, 45
show, ptrSet-method, 46
snipBase, 46
timeLimits, 14, 19, 47
timeLimits, ptrRaw-method (timeLimits), 47
timeLimits, ptrSet-method (timeLimits), 47
TransmissionCurve, 49
updatePtrSet, 41, 49
width, 50
writeEset, 50
writeEset, ExpressionSet-method
  (writeEset), 50