Package ‘MsQuality’

October 15, 2023

Type Package

Title MsQuality - Quality metric calculation from Spectra and MsExperiment objects

Version 1.0.0

Date 2023-04-18

VignetteBuilder knitr

Description The MsQuality provides functionality to calculate quality metrics for mass spectrometry-derived, spectral data at the per-sample level. MsQuality relies on the mzQC framework of quality metrics defined by the Human Proteom Organization-Proteomics Standards Initiative (HUPO-PSI). These metrics quantify the quality of spectral raw files using a controlled vocabulary. The package is especially addressed towards users that acquire mass spectrometry data on a large scale (e.g. data sets from clinical settings consisting of several thousands of samples). The MsQuality package allows to calculate low-level quality metrics that require minimum information on mass spectrometry data: retention time, m/z values, and associated intensities. MsQuality relies on the Spectra package, or alternatively the MsExperiment package, and its infrastructure to store spectral data.

Depends R (>= 4.2.0)

Imports ggplot2 (>= 3.3.5), htmlwidgets (>= 1.5.3), methods (>= 4.2.0), MsExperiment (>= 0.99.0), plotly (>= 4.9.4.1), ProtGenerics (>= 1.24.0), shiny (>= 1.6.0), shinydashboard (>= 0.7.1), Spectra (>= 1.2.0), stats (>= 4.2.0), stringr (>= 1.4.0), tibble (>= 3.1.4), tidyr (>= 1.1.3), BiocParallel

Suggests BiocGenerics (>= 0.24.0), BiocStyle (>= 2.6.1), dplyr (>= 1.0.5), knitr (>= 1.11), msdata (>= 0.32.0), mzR (>= 2.32.0), rmarkdown (>= 2.7), S4Vectors (>= 0.29.17), testthat (>= 2.2.1)

biocViews Metabolomics, Proteomics, MassSpectrometry, QualityControl

URL `https://www.github.com/tnaake/MsQuality/`

BugReport https://www.github.com/tnaake/MsQuality/issues/

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

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

git_branch RELEASE_3_17

git_last_commit e4e396a

git_last_commit_date 2023-04-25

Date/Publication 2023-10-15

Author Thomas Naake [aut, cre] (<https://orcid.org/0000-0001-7917-5580>), Johannes Rainer [aut] (<https://orcid.org/0000-0002-6977-7147>)

Maintainer Thomas Naake <thomasnaake@googlemail.com>

R topics documented:

`MsQuality-package` *MsQuality - Quality metric calculation from Spectra and MsExperiment objects*

Description

MsQuality enables to calculate quality metrics of mass spectrometry data. It is build upon Spectra and MsExperiment objects.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

Author(s)

NA Maintainer: NA

Examples

```
## Not run: calculateMetrics(object = spectra)
## Not run: calculateMetrics(object = mse)
```

`.rtOrderSpectra` *Order Spectra according to increasing retention time*

Description

The function ‘.rtOrderSpectra‘ orders the features in a ‘Spectra‘ object according to the (increasing) retention time values.

Usage

```
.rtOrderSpectra(spectra)
```

Arguments

`spectra`	‘Spectra‘ object

Details

Internal function in quality metric functions.

Value

‘Spectra‘ object with the features ordered according to the (increasing) retention time

Author(s)

Johannes Rainer

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0001847", "HMDB0000001", "HMDB0000001"),
    name = c("Caffeine", "1-Methylhistidine", "1-Methylhistidine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876),
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16))
spd$intensity <- list(
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994),
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643))
spd$rtime <- c(15.84, 9.44, 9.44)
sps <- Spectra(spd)
MsQuality:::.rtOrderSpectra(sps)
```

`areaUnderTic` *Area under TIC (QC:4000077)*

Description

"The area under the total ion chromatogram." [PSI:QC] id: QC:4000077

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level, (2) the sum of the ion counts are obtained and returned.

Usage

```
areaUnderTic(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000022 ! chromatogram metric

The sum of the TIC is returned as an equivalent to the area.

Value

‘numeric(1)‘

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
sps <- Spectra(spd)
areaUnderTic(spectra = sps, msLevel = 2L)
```

areaUnderTicRtQuantiles

Area under TIC RT quantiles (QC:4000078)

Description

"The area under the total ion chromatogram of the retention time quantiles. Number of quantiles are given by the n-tuple." [PSI:QC] id: QC:4000078

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the ‘Spectra‘ object is ordered according to the retention time,

(3) the 0%, 25%, 50%, 75%, and 100% quantiles of the retention time values are obtained,

(4) the ion count of the intervals between the 0%/25%, 25%/50%, 50%/75%, and 75%/100% are obtained

(5) the ion counts of the intervals are summed (TIC) and the values returned

Usage

```
areaUnderTicRtQuantiles(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000004 ! n-tuple is_a: QC:4000010 ! ID free is_a: QC:4000022 ! chromatogram metric

The sum of the TIC is returned as an equivalent to the area.

Value

‘numeric(4)‘

Note

This function interprets the *quantiles* from the [PSI:QC] definition as *quartiles*, i.e. the 0, 25, 50, 75 and 100% quantiles are used.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
areaUnderTicRtQuantiles(spectra = sps, msLevel = 2L)
```

`calculateMetrics` *Calculate QC metrics from a Spectra or MsExperiment object*

Description

Calculate QC metrics from a `Spectra` or `MsExperiment` object. `calculateMetrics` is a wrapper for the functions `calculateMetricsFromSpectra` and `calculateMetricsFromMsExperiment`.

Usage

```
calculateMetrics(object, metrics = qualityMetrics(object), ...)
```

Arguments

`object`	`Spectra` or `MsExperiment` object
`metrics`	`character` specifying the quality metrics to be calculated on `object`
`...`	arguments passed to the quality metrics functions defined in `metrics`

Details

The metrics are defined by the argument `metrics`. Further arguments passed to the quality metric functions can be specified by the `params` argument. `params` can contain named entries which are matched against the formal arguments of the quality metric functions.

Value

`data.frame` containing in the columns the metrics for the different spectra and in rows the samples

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(msdata)
library(Spectra)
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)
spectra <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

#' ## calculate the metrics
## additional parameters passed to the quality metrics functions
## (MsLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange) passed to ...
calculateMetrics(object = spectra, metrics = metrics,
    msLevel = 1, change = "jump", relativeTo = "Q1")
calculateMetrics(object = spectra, metrics = metrics,
    msLevel = 1, change = "fall", relativeTo = "previous")
```

`calculateMetricsFromMsExperiment` *Calculate QC metrics from a MsExperiment object*

Description

The function `calculateMetricsFromMsExperiment` calculates quality metrics from a `MsExperiment` object. Each spectra in the `msexp` object should refer to one mzML file/to one sample.

Usage

```
calculateMetricsFromMsExperiment(
  msexp,
  metrics = qualityMetrics(msexp),
  ...,
  BPPARAM = bpparam()
)
```

Arguments

msexp	MsExperiment object
metrics	character specifying the quality metrics to be calculated on msexp
...	arguments passed to the quality metrics functions defined in metrics
BPPARAM	Parallel processing setup. Defaults to ‘BPPARAM = bpparam()‘. See [bpparam()] for details on parallel processing with ‘BiocParallel‘.

Details

The metrics are defined by the argument metrics. Further arguments passed to the quality metric functions can be specified by the params argument. params can contain named entries which are matched against the formal arguments of the quality metric functions.

Value

data.frame containing in the columns the metrics for the different spectra (in rows)

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(msdata)
library(MsExperiment)
library(S4Vectors)

msexp <- MsExperiment()
sd <- DataFrame(sample_id = c("QC1", "QC2"),
    sample_name = c("QC Pool", "QC Pool"), injection_idx = c(1, 3))
sampleData(msexp) <- sd

## define file names containing spectra data for the samples and
## add them, along with other arbitrary files to the experiment
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)
experimentFiles(msexp) <- MsExperimentFiles(
    mzML_files = fls,
    annotations = "internal_standards.txt")
## link samples to data files: first sample to first file in "mzML_files",
## second sample to second file in "mzML_files"
msexp <- linkSampleData(msexp, with = "experimentFiles.mzML_files",
    sampleIndex = c(1, 2), withIndex = c(1, 2))
msexp <- linkSampleData(msexp, with = "experimentFiles.annotations",
     sampleIndex = c(1, 2), withIndex = c(1, 1))

library(Spectra)
## import the data and add it to the mse object
spectra(msexp) <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
```

```
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## additional parameters passed to the quality metrics functions
## (msLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange) passed to ...
calculateMetricsFromMsExperiment(msexp = msexp, metrics = metrics,
    msLevel = 1, change = "jump", relativeTo = "Q1")

calculateMetricsFromMsExperiment(msexp = msexp, metrics = metrics,
    msLevel = 1, change = "fall", relativeTo = "previous")
```

---

calculateMetricsFromOneSampleSpectra

Calculate QC metrics from a Spectra object containing only spectral data from one sample

---

Description

The function `calculateMetricsFromOneSampleSpectra` calculates quality metrics from a `Spectra` containing spectral data from one sample.

Usage

```
calculateMetricsFromOneSampleSpectra(
  spectra,
  metrics = qualityMetrics(spectra),
  f = spectra$dataOrigin,
  ...
)
```

Arguments

`spectra`	`Spectra` object
`metrics`	`character` specifying the quality metrics to be calculated on `spectra`
`f`	`character`, grouping parameter for `spectra`
`...`	arguments passed to the quality metrics functions defined in `metrics`

Details

The metrics are defined by the argument `metrics`. Further arguments passed to the quality metric functions can be specified by the `params` argument. `params` can contain named entries which are matched against the formal arguments of the quality metric functions.

The `Spectra` object will only contain spectral data from one data origin (e.g. `spectra$dataOrigin` is of length 1). The grouping is specified by the argument `f`.

Value

named `numeric` vector

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(msdata)
library(Spectra)
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)[1]
spectra <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## calculate the metrics
## additional parameters passed to the quality metrics functions
## (MsLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange) passed to ...
MsQuality:::calculateMetricsFromOneSampleSpectra(spectra = spectra,
    metrics = metrics, msLevel = 1, change = "jump", relativeTo = "Q1")
MsQuality:::calculateMetricsFromOneSampleSpectra(spectra = spectra,
    metrics = metrics, msLevel = 1, change = "fall", relativeTo = "previous")
```

---

calculateMetricsFromSpectra

Calculate QC metrics from a Spectra object

Description

The function calculateMetricsFromSpectra calculates quality metrics from a Spectra object. The function will calculate the metrics per sample according to the grouping parameter f, e.g. dataOrigin information. Samples will be processed in parallel using the default parallel processing setup ([bpparam()]) or with the parallel processing setup defined with parameter ‘BPPARAM‘.

Usage

```
calculateMetricsFromSpectra(
  spectra,
  metrics = qualityMetrics(spectra),
  f = dataOrigin(spectra),
  ...,
  BPPARAM = bpparam()
)
```

Arguments

spectra	Spectra object
metrics	character specifying the quality metrics to be calculated on spectra

f	`character` defining which spectra in ‘spectra‘ belong to one sample. Defaults to ‘f = dataOrigin(spectra)‘. Spectra from the same original data file are processed together (and in parallel for different files).
...	arguments passed to the quality metrics functions defined in `metrics`
BPPARAM	Parallel processing setup. Defaults to ‘BPPARAM = bpparam()‘. See [bpparam()] for details on parallel processing with ‘BiocParallel‘.

Details

The metrics are defined by the argument `metrics`. Further arguments passed to the quality metric functions can be specified by the `params` argument. `params` can contain named entries which are matched against the formal arguments of the quality metric functions.

Value

`data.frame` containing in the columns the metrics for the different spectra (in rows)

Author(s)

Thomas Naake, <`thomasnaake@googlemail.com`>, Johannes Rainer

Examples

```
library(msdata)
library(Spectra)

## define file names containing spectra data for the samples
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)

## import the data and add it to the spectra object
spectra <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## calculate the metrics
## additional parameters passed to the quality metrics functions
## (msLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange) passed to ...
calculateMetricsFromSpectra(spectra = spectra, metrics = metrics,
    msLevel = 1, change = "jump", relativeTo = "Q1")
calculateMetricsFromSpectra(spectra = spectra, metrics = metrics,
    msLevel = 1, change = "fall", relativeTo = "previous")
```

`extentIdentifiedPrecursorIntensity`
Extent of identified precursor intensity (QC:4000125)

Description

"Ratio of 95th over 5th percentile of precursor intensity for identified peptides" [PSI:QC] id: QC:4000125

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the intensities of the precursor ions are obtained,

(3) the 5% and 95% quantile of these intensities are obtained (‘NA‘ values are removed),

(4) the ratio between the 95% and the 5% intensity quantile is calculated and returned.

Usage

```
extentIdentifiedPrecursorIntensity(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

Can be used to approximate the dynamic range of signal is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

Precursor intensity values that are ‘NA‘ are removed.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000125*, the ‘Spectra‘ object should be prepared accordingly, i.e. being subsetted to spectra with identification data.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorIntensity <- c(100, 100, 100)
sps <- Spectra(spd)
extentIdentifiedPrecursorIntensity(spectra = sps, msLevel = 2L)
```

`Lee_2019` *Example data for* `MsQuality`*: data set of Lee et al. (2019)*

Description

The data set of Lee et al. (2019) contains metabolite information measured by reverse phase liquid chromatography (RPLC) coupled to mass spectrometry and hydrophilic interaction liquid chromatography (HILIC) coupled to mass spectrometry (file ‘STables - rev1.xlsx‘ in the Supplementary Information).

It will be used as an example data set in the vignette to show the functionality of the packages. The file contains `Spectra` and `MsExperiment` objects that store the mass spectrometry data.

Format

`Spectra` and `MsExperiment`

Value

`Spectra` and `MsExperiment` objects

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Source

See the file `Lee2019-data-source.R` in `scripts` for the source code how `sps_hilic` and `sps_rplc` were created.

References

Lee et al. (2019). A large-scale analysis of targeted metabolomics data from heterogeneous biological samples provides insights into metabolite dynamics. Metabolomics, 103, doi: 10.1007/s11306-019-1564-8.

`Lee_2019_meta_vals` *Example data for* `MsQuality`*: data set of Lee et al. (2019)*

Description

The data set of Lee et al. (2019) contains metabolite information measured by reverse phase liquid chromatography (RPLC) coupled to mass spectrometry and hydrophilic interaction liquid chromatography (HILIC) coupled to mass spectrometry (file ‘STables - rev1.xlsx‘ in the Supplementary Information). The xlsx sheets ‘Methods‘ and ‘Raw data‘ were stored as txt files.

`Lee_2019_meta_vals` contains two data frame objects: one containing information on metabolite meta-data and one containing intensity values on metabolites. The object will be used as an example data set in the vignette to show the functionality of the packages.

Format

`data.frame`

Value

`data.frame`

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Source

path_to_meta <- "Lee_et_al_2019_Stables_rev1_Methods.txt" meta <- read.delim(path_to_meta, dec = ".", header = TRUE)

## print number of metabolites per measurement (meta data) table(meta$Method)

path_to_vals <- "Lee_et_al_2019_Stables_rev1_Raw_data.txt" vals <- read.delim(path_to_vals, dec = ".", header = TRUE)

## print number of metabolites per measurement (intensity data) table(grepl(vals$Metabolite, pattern = "_rp$")) table(grepl(vals$Metabolite, pattern = "_hn$"))

## save the two objects as an RData object save(meta, vals, file = "Lee_2019_meta_vals.RData", compress = "xz")

References

Lee et al. (2019). A large-scale analysis of targeted metabolomics data from heterogeneous biological samples provides insights into metabolite dynamics. Metabolomics, 103, doi: 10.1007/s11306-019-1564-8.

`meanCharge` *Mean charge in identified spectra (QC:4000177) or mean precursor charge in all MS2 (QC:4000182)*

Description

"Mean charge in identified spectra" [PSI:QC] id: QC:4000177

"Mean precursor charge in all MS2" [PSI:QC] id: QC:4000182

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor charge is obtained,

(3) the mean of the precursor charge values is calculated and returned.

Usage

```
meanCharge(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000177*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorCharge <- c(1L, 1L, 1L)
sps <- Spectra(spd)
meanCharge(spectra = sps, msLevel = 2L)
```

---

medianCharge — *Median charge in identified spectra (QC:4000178) or median precursor charge in all MS2 (QC:4000183)*

Description

"Median charge in identified spectra" [PSI:QC] id: QC:4000178

"Median precursor charge in all MS2" [PSI:QC] id: QC:4000183

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor charge is obtained,

(3) the median of the precursor charge values is calculated and returned.

Usage

```
medianCharge(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000178*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
sps <- Spectra(spd)
spd$precursorCharge <- c(1L, 1L, 1L)
medianCharge(spectra = sps, msLevel = 2L)
```

`medianPrecursorMz` *Precursor median m/z for IDs (QC:4000065)*

Description

"Median m/z value for all identified peptides (unique ions) after FDR." [PSI:QC] id: QC:4000065

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor m/z values are obtained,

(3) the median value is returned (‘NAs‘ are removed).

Usage

```
medianPrecursorMz(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000023 ! MS1 metric is_a: QC:4000025 ! ion source metric

Value

‘numeric(1)‘

Note

‘medianPrecursorMz‘ will calculate the *precursor* median m/z of all Spectra within ‘spectra‘. If the calculation needs be done according to *QC:4000065*, the ‘Spectra‘ object should be prepared accordingly, i.e. filtered with e.g. [filterPrecursorMz()] or subsetted to spectra with identification data.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorMz <- c(170.16, 170.16, 195.0876)
sps <- Spectra(spd)
medianPrecursorMz(spectra = sps, msLevel = 2L)
```

`medianTicOfRtRange` *Median of TIC values in the shortest RT range in which half of the peptides are identified (QC:4000132)*

Description

"Median of TIC values in the shortest RT range in which half of the peptides are identified" [PSI:QC] id: QC:4000132

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the ‘Spectra‘ object is ordered according to the retention time,

(3) the number of features in ‘spectra‘ is obtained and the number for half of the features is calculated,

(4) iterate through the features (always by taking the neighbouring half of features) and calculate the retention time range of the set of features,

(5) retrieve the set of features with the minimum retention time range,

(6) calculate from the set of (5) the median TIC (‘NA‘ values are removed) and return it.

Usage

```
medianTicOfRtRange(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

The function ‘medianTicOfRtRange‘ uses the function ‘ionCount‘ as an equivalent to the TIC.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000132*, the ‘Spectra‘ object should be prepared accordingly, i.e. being subsetted to spectra with identification data.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
medianTicOfRtRange(spectra = sps, msLevel = 2L)
```

`medianTicRtIqr` *Median of TIC values in the RT range in which the middle half of peptides are identified (QC:4000130)*

Description

"Median of TIC values in the RT range in which half of peptides are identified (RT values of Q1 to Q3 of identifications)" [PSI:QC] id: QC:4000130

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the ‘Spectra‘ object is ordered according to the retention time,

(3) the features between the 1st and 3rd quartile are obtained (half of the features that are present in ‘spectra‘),

(4) the ion count of the features within the 1st and 3rd quartile is obtained,

(5) the median value of the ion count is calculated (‘NA‘ values are removed) and the median value is returned.

Usage

```
medianTicRtIqr(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

The function ‘medianTicRtIqr‘ uses the function [ionCount()] as an equivalent to the TIC.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000130*, the ‘Spectra‘ object should be prepared accordingly, i.e. being subsetted to spectra with identification data.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
medianTicRtIqr(spectra = sps, msLevel = 2L)
```

`msSignal10xChange` *MS1 signal jump/fall (10x) count (QC:4000172/QC:4000173)*

Description

"The count of MS1 signal jump (spectra sum) by a factor of ten or more (10x) between two subsequent scans" [PSI:QC] id: QC:4000172

"The count of MS1 signal decline (spectra sum) by a factor of ten or more (10x) between two subsequent scans" [PSI:QC] id: QC:4000173

#' The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the intensity of the precursor ions within ‘spectra‘ are obtained,

(3) the intensity values of the features are obtained via the ion count,

(4) the signal jumps/declines of the intensity values with the two subsequent intensity values is calculated,

(5) in the case of *QC:4000172*, the signal jumps by a factor of ten or more are counted and returned; in the case of *QC:4000173*, the signal declines by a factor of ten or more are counted and returned.

Usage

```
msSignal10xChange(spectra, change = "jump", msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`change`	‘character(1)‘, one of ‘"jump"‘ or ‘"fall"‘
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric

An unusual high count of signal jumps or falls can indicate ESI stability issues.

The function ‘msSignal10xChange‘ uses the function ‘ionCount‘ as an equivalent to the TIC.

Value

‘numeric(1)‘

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
msSignal10xChange(spectra = sps, change = "jump", msLevel = 2L)
msSignal10xChange(spectra = sps, change = "fall", msLevel = 2L)
```

`mzAcquisitionRange` *m/z acquisition range (QC:4000138)*

Description

"Upper and lower limit of m/z values at which spectra are recorded." [PSI:QC] id: QC:4000138

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the m/z values of the peaks within ‘spectra‘ are obtained,

(3) the minimum and maximum m/z values are obtained and returned.

Usage

```
mzAcquisitionRange(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric is_a: QC:4000004 ! n-tuple

Value

‘numeric(2)‘

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
sps <- Spectra(spd)
mzAcquisitionRange(spectra = sps, msLevel = 2L)
```

---

numberSpectra *Number of MS1 or MS2 spectra (QC:4000059/QC:4000060)*

Description

"The number of MS1 events in the run." [PSI:QC] id: QC:4000059 "The number of MS2 events in the run." [PSI:QC] id: QC:4000060

For *QC:4000059*, ‘msLevel‘ is set to 1. For *QC:4000060*, ‘msLevel‘ is set to 2.

Usage

```
numberSpectra(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000023 ! MS1 metric

Value

'numeric(1)'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
sps <- Spectra(spd)
numberSpectra(spectra = sps, msLevel = 1L)
numberSpectra(spectra = sps, msLevel = 2L)
```

---

plotMetric — *Visualize a quality metric*

Description

The function `plotMetric` visualizes the metric values per sample. The function accepts the output of `calculateMetrics` or, `calculateMetricsFromSpectra`, or `calculateMetricsFromMsExperiment` and a vector specifying the metric to display.

Usage

```
plotMetric(qc, metric = "areaUnderTic", plotly = TRUE)
```

Arguments

qc	matrix/data.frame
metric	character
plotly	logical(1)

Details

`plotMetric` will select all columns that start with `metric`. The different levels in the `name` column in the returned tibble correspond to the columns that were selected and do not contain the `metric` prefix. In case there is no additional specification (e.g. for the metric `rtDuration` only the column `rtDuration` will be selected), the `name` column will include the `metric` (`rtDuration`).

Value

gg `plotly`

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(msdata)
library(MsExperiment)
library(S4Vectors)
msexp <- MsExperiment()
sd <- DataFrame(sample_id = c("QC1", "QC2"),
    sample_name = c("QC Pool", "QC Pool"), injection_idx = c(1, 3))
sampleData(msexp) <- sd

## define file names containing spectra data for the samples and
## add them, along with other arbitrary files to the experiment
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)

library(Spectra)
## import the data and add it to the msexp object
spectra(msexp) <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## calculate the metrics
## additional parameters passed to the quality metrics functions
## (msLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange)
qc <- calculateMetricsFromMsExperiment(msexp = msexp, metrics = metrics,
    msLevel = 1, relativeTo = "Q1", change = "jump")
rownames(qc) <- c("Sample 1", "Sample 2")

## do the actual plotting
plotMetric(qc, metric = "areaUnderTic", plotly = TRUE)
```

plotMetricTibble *Helper function for plotMetric*

Description

The function `plotMetricTibble` is a helper function for the function `plotMetric`. It returns a tibble in long format that is interpretable by `ggplot2`.

Usage

```
plotMetricTibble(qc, metric)
```

Arguments

`qc`	`data.frame`
`metric`	`character`

Details

`plotMetricRibble` will select all columns that start with `metric`. The different levels in the `name` column in the returned tibble correspond to the columns that were selected and do not contain the `metric` prefix. In case there is no additional specification (e.g. for the metric `rtDuration` only the column `rtDuration` will be selected), the `name` column will include the `metric` (`rtDuration`).

Value

`tibble`

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(msdata)
library(MsExperiment)
library(S4Vectors)
msexp <- MsExperiment()
sd <- DataFrame(sample_id = c("QC1", "QC2"),
    sample_name = c("QC Pool", "QC Pool"), injection_idx = c(1, 3))
sampleData(msexp) <- sd

## define file names containing spectra data for the samples and
## add them, along with other arbitrary files to the experiment
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)
experimentFiles(msexp) <- MsExperimentFiles(
    mzML_files = fls,
    annotations = "internal_standards.txt")
## link samples to data files: first sample to first file in "mzML_files",
## second sample to second file in "mzML_files"
msexp <- linkSampleData(msexp, with = "experimentFiles.mzML_files",
    sampleIndex = c(1, 2), withIndex = c(1, 2))
msexp <- linkSampleData(msexp, with = "experimentFiles.annotations",
    sampleIndex = c(1, 2), withIndex = c(1, 1))

library(Spectra)
## import the data and add it to the mse object
spectra(msexp) <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## calculate the metrics
## additional parameters passed to the quality metrics functions
## (msLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange)
qc <- calculateMetricsFromMsExperiment(msexp = msexp, metrics = metrics,
    msLevel = 1, relativeTo = "Q1", change = "jump")
rownames(qc) <- c("Sample 1", "Sample 2")
plotMetricTibble(qc, metric = "areaUnderTic")
```

precursorIntensityMean

Precursor intensity distribution mean (QC:4000168), Identified precursor intensity distribution mean (QC:4000229), or Unidentified precursor intensity distribution mean (QC:4000234)

Description

"From the distribution of precursor intensities, the mean." [PSI:QC] id: QC:4000168

"From the distribution of identified precursor intensities, the mean" [PSI:QC] id: QC:4000229

"From the distribution of unidentified precursor intensities, the mean" [PSI:QC] id: QC:4000234

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the intensity of the precursor ions within ‘spectra‘ are obtained,

(3) the mean of the precursor intensity values is obtained (‘NA‘ values are removed) and returned.

Usage

```
precursorIntensityMean(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric

The intensity distribution of the precursors informs about the dynamic range of the acquisition.

The intensity distribution of the identified precursors informs about the dynamic range of the acquisition in relation to identifiability.

The intensity distribution of the unidentified precursors informs about the dynamic range of the acquisition in relation to identifiability.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were (not) identified. If the calculation needs to be done according to *QC:4000229*/*QC:4000234*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorIntensity <- c(100.0, 100.0, 100.0)
sps <- Spectra(spd)
precursorIntensityMean(spectra = sps, msLevel = 2L)
```

`precursorIntensityQuartiles` *Precursor intensity distribution Q1, Q2, Q3 (QC:4000167), Identified precursor intensity distribution Q1, Q2, Q3 (QC:4000228), or Unidentified precursor intensity distribution Q1, Q2, Q3 (QC:4000233)*

Description

"From the distribution of precursor intensities, the quartiles Q1, Q2, Q3" [PSI:QC] id: QC:4000167

"From the distribution of identified precursor intensities, the quartiles Q1, Q2, Q3" [PSI:QC] id: QC:40000228

"From the distribution of unidentified precursor intensities, the quartiles Q1, Q2, Q3" [PSI:QC] id: QC:4000233

The metric is calculated as follows: (1) the 'Spectra' object is filtered according to the MS level,

(2) the intensity of the precursor ions within 'spectra' are obtained,

(3) the 25%, 50%, and 75% quantile of the precursor intensity values are obtained ('NA' values are removed) and returned.

Usage

```
precursorIntensityQuartiles(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	'Spectra' object
`msLevel`	'integer'
`...`	not used here

Details

is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric is_a: QC:4000004 ! n-tuple

The intensity distribution of the precursors informs about the dynamic range of the acquisition.

The intensity distribution of the identified precursors informs about the dynamic range of the acquisition in relation to identifiability.

The intensity distribution of the unidentified precursors informs about the dynamic range of the acquisition in relation to identifiability.

Value

'numeric(3)'

Note

The ‘Spectra‘ object might contain features that were (not) identified. If the calculation needs to be done according to *QC:4000228*/*QC:4000233*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorIntensity <- c(100.0, 100.0, 100.0)
sps <- Spectra(spd)

precursorIntensityQuartiles(spectra = sps, msLevel = 2L)
```

---

precursorIntensityRange

Precursor intensity range (QC:4000144)

Description

"Minimum and maximum precursor intensity recorded." [PSI:QC] id: QC:4000144

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the intensity of the precursor ions within ‘spectra‘ are obtained,

(3) the minimum and maximum precursor intensity values are obtained and returned.

Usage

```
precursorIntensityRange(spectra, msLevel = 1, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric is_a: QC:4000004 ! n-tuple

The intensity range of the precursors informs about the dynamic range of the acquisition.

Value

‘numeric(2)‘

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorIntensity <- c(100.0, 100.0, 100.0)
sps <- Spectra(spd)
precursorIntensityRange(spectra = sps, msLevel = 2L)
```

`precursorIntensitySd` *Precursor intensity distribution sigma (QC:4000169), Identified precursor intensity distribution sigma (QC:4000230), or Unidentified precursor intensity distribution sigma (QC:4000235)*

Description

"From the distribution of precursor intensities, the sigma value." [PSI:QC] id: QC:4000169

"From the distribution of identified precursor intensities, the sigma value" [PSI:QC] id: QC:4000230

"From the distribution of unidentified precursor intensities, the sigma value" [PSI:QC] id: QC:4000235

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the intensity of the precursor ions within ‘spectra‘ are obtained,

(3) the standard deviation of precursor intensity values is obtained (‘NA‘ values are removed) and returned.

Usage

```
precursorIntensitySd(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000001 ! QC metric

The intensity distribution of the precursors informs about the dynamic range of the acquisition.

The intensity distribution of the identified precursors informs about the dynamic range of the acquisition in relation to identifiability.

The intensity distribution of the unidentified precursors informs about the dynamic range of the acquisition in relation to identifiability.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were (not) identified. If the calculation needs to be done according to *QC:4000230*/*QC:4000235*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorIntensity <- c(100.0, 100.0, 100.0)
sps <- Spectra(spd)
precursorIntensitySd(spectra = sps, msLevel = 2L)
```

`qualityMetrics` *Get a vector of quality metrics than can be applied to* `object`

Description

The function `qualityMetrics` returns a character vector with available quality metrics depending on `object`.

Usage

```
qualityMetrics(object)
```

Arguments

`object`	object of type `Spectra` or `MsExperiment`

Details

`object` is a `Spectra` or `MsExperiment`.

Value

`character`

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(Spectra)
spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$dataOrigin <- rep("sample_1", 3)
sps <- Spectra(spd)

qualityMetrics(object = sps)
```

---

ratioCharge1over2 — *Charged peptides ratio 1+ over 2+ (QC:4000174) or Charged spectra ratio 1+ over 2+ (QC:4000179)*

Description

"Ratio of 1+ peptide count over 2+ peptide count in identified spectra" [PSI:QC] id: QC:4000174

"Ratio of 1+ spectra count over 2+ spectra count in all MS2" [PSI:QC] id: QC:4000179

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor charge is obtained,

(3) the number of precursors with charge 1+ is divided by the number of precursors with charge 2+ and the ratio is returned.

Usage

```
ratioCharge1over2(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

‘NA‘ is returned if there are no features with precursor charge of 1+ or 2+.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000174*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorCharge <- c(1L, 1L, 1L)
sps <- Spectra(spd)
ratioCharge1over2(spectra = sps, msLevel = 2L)
```

`ratioCharge3over2` *Charged peptides ratio 3+ over 2+ (QC:4000175) or charged spectra ratio 3+ over 2+ (QC:4000180)*

Description

"Ratio of 3+ peptide count over 2+ peptide count in identified spectra" [PSI:QC] id: QC:4000175

"Ratio of 3+ peptide count over 2+ peptide count in all MS2" [PSI:QC] id: QC:4000180

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor charge is obtained,

(3) the number of precursors with charge 3+ is divided by the number of precursors with charge 2+ and the ratio is returned.

Usage

```
ratioCharge3over2(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

‘NA‘ is returned if there are no features with precursor charge of 2+ or 3+.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000175*, the ‘Spectra‘ object should be prepared accordingly.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorCharge <- c(1L, 1L, 1L)
sps <- Spectra(spd)
ratioCharge3over2(spectra = sps, msLevel = 2L)
```

`ratioCharge4over2` *Charged peptides ratio 4+ over 2+ (QC:4000176) or charged spectra ratio 4+ over 2+ (QC:4000181)*

Description

"Ratio of 4+ peptide count over 2+ peptide count in identified spectra" [PSI:QC] id: QC:4000176

"Ratio of 4+ peptide count over 2+ peptide count in all MS2" [PSI:QC] id: QC:4000181

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the precursor charge is obtained,

(3) the number of precursors with charge 4+ is divided by the number of precursors with charge 2+ and the ratio is returned.

Usage

```
ratioCharge4over2(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000001 ! QC metric

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000176*, the ‘Spectra‘ object should be prepared accordingly.

‘NA‘ is returned if there are no features with precursor charge of 2+ or 3+.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$precursorCharge <- c(1L, 1L, 1L)
sps <- Spectra(spd)
ratioCharge4over2(spectra = sps, msLevel = 2L)
```

`rtAcquisitionRange` *Retention time acquisition range (QC:4000139)*

Description

"Upper and lower limit of time at which spectra are recorded." [PSI:QC] id: QC:4000139

#' The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the retention time values of the features within ‘spectra‘ are obtained,

(3) the minimum and maximum retention time values are obtained and returned.

Usage

```
rtAcquisitionRange(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000004 ! n-tuple

Value

'numeric(2)'

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
rtAcquisitionRange(spectra = sps, msLevel = 2L)
```

`rtDuration` *RT duration (QC:4000053)*

Description

"The retention time duration of the MS run in seconds, similar to the highest scan time minus the lowest scan time." [PSI:QC] id: QC:4000053

The metric is calculated as follows: (1) the retention time associated to the individual Spectra is obtained,

(2) the maximum and the minimum of the retention time is obtained,

(3) the difference between the maximum and the minimum is calculated and returned.

Usage

```
rtDuration(spectra, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`...`	not used here

Details

is_a: QC:4000003 ! single value is_a: QC:4000010 ! ID free is_a: QC:4000021 ! retention time metric

Retention time values that are ‘NA‘ are removed.

Value

‘numeric(1)‘

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
rtDuration(spectra = sps)
```

`rtIqr` *Interquartile RT period for peptide identifications (QC:4000072)*

Description

"The interquartile retention time period, in seconds, for all peptide identifications over the complete run." [PSI:QC] id: QC:4000072

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the retention time values are obtained,

(3) the interquartile range is obtained from the values and returned (‘NA‘ values are removed).

Usage

```
rtIqr(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

Longer times indicate better chromatographic separation. is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000022 ! chromatogram metric

Retention time values that are ‘NA‘ are removed.

Value

‘numeric(1)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000072*, the ‘Spectra‘ object should be prepared accordingly, i.e. subsetted to spectra with identification data.

The stored retention time information in ‘spectra‘ might have a different unit than seconds. ‘rtIqr‘ will return the IQR based on the values stored in ‘spectra‘ and will not convert these values to seconds.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
rtIqr(spectra = sps, msLevel = 2L)
```

`rtIqrRate` *Peptide identification rate of the interquartile RT period (QC:4000073)*

Description

"The identification rate of peptides for the interquartile retention time period, in peptides per second." [PSI:QC] id: QC:4000073

The metric is calculated as follows: (1) the ‘Spectra‘ object is filtered according to the MS level,

(2) the retention time values are obtained,

(3) the 25% and 75% quantiles are obtained from the retention time values (‘NA‘ values are removed),

(4) the number of eluted features between this 25% and 75% quantile is calculated,

(5) the number of features is divided by the interquartile range of the retention time and returned.

Usage

```
rtIqrRate(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

Higher rates indicate efficient sampling and identification. is_a: QC:4000003 ! single value is_a: QC:4000009 ! ID based is_a: QC:4000022 ! chromatogram metric

Value

‘numeric(2)‘

Note

The ‘Spectra‘ object might contain features that were not identified. If the calculation needs to be done according to *QC:4000073*, the ‘Spectra‘ object should be prepared accordingly, i.e. being subsetted to ‘spectra‘ with identification data.

The stored retention time information in ‘spectra‘ might have a different unit than seconds. ‘rtIqr‘ will return the IQR based on the values stored in ‘spectra‘ and will not convert these values to seconds.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
rtIqrRate(spectra = sps, msLevel = 2L)
```

`rtOverMsQuarters` *MS1/MS2 quantiles RT fraction (QC:4000055/QC:4000056)*

Description

"The interval used for acquisition of the first, second, third, and fourth quarter of all MS1 events divided by RT-Duration." [PSI:QC] id: QC:4000055

"The interval used for acquisition of the first, second, third, and fourth quarter of all MS2 events divided by RT-Duration." [PSI:QC] id: QC:4000056

The metric is calculated as follows: (1) the retention time duration of the whole ‘Spectra‘ object is determined (taking into account all the MS levels),

(2) the ‘Spectra‘ object is filtered according to the MS level and subsequently ordered according to the retention time

(3) the MS events are split into four (approximately) equal parts,

(4) the relative retention time is calculated (using the retention time duration from (1) and taking into account the minimum retention time),

(5) the relative retention time values associated to the MS event parts are returned.

Usage

```
rtOverMsQuarters(spectra, msLevel = 1L, ...)
```

Arguments

`spectra`	‘Spectra‘ object
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: QC:4000004 ! n-tuple is_a: QC:4000010 ! ID free is_a: QC:4000021 ! retention time metric is_a: QC:4000023 ! MS1 metric

The function returns ‘c(NaN, NaN, NaN, NaN)‘ if the filtered ‘spectra‘ object has less than 4 scan events.

Value

‘numeric(4)‘

Note

‘rtDuration‘ considers the total runtime (including MS1 and MS2 scans).

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`, Johannes Rainer

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L, 2L),
    polarity = c(1L, 1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847", "unknown"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine", "unknown"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876),
    c(83.0603, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994),
    c(3.146, 61.611))
spd$rtime <- c(9.44, 9.44, 15.84, 15.81)
sps <- Spectra(spd)
rtOverMsQuarters(spectra = sps, msLevel = 2L)
```

`rtOverTicQuantiles` *RT over TIC quantile (QC:4000054)*

Description

"The interval when the respective quantile of the TIC accumulates divided by retention time duration. The number of quantiles observed is given by the size of the tuple." [PSI:QC] id: QC:4000054

The metric is calculated as follows: (1) the ‘Spectra‘ object is ordered according to the retention time,

(2) the cumulative sum of the ion count is calculated (TIC),

(3) the quantiles are calculated according to the ‘probs‘ argument, e.g. when ‘probs‘ is set to ‘c(0, 0.25, 0.5, 0.75, 1)‘ the 0%, 25%, 50%, 75% and 100% quantile is calculated,

(4) the retention time/relative retention time (retention time divided by the total run time taking into account the minimum retention time) is calculated,

(5) the (relative) duration of the LC run after which the cumulative

Usage

```
rtOverTicQuantiles(
  spectra,
  probs = seq(0, 1, 0.25),
  msLevel = 1L,
  relative = TRUE,
  ...
)
```

Arguments

spectra	‘Spectra‘ object
probs	‘numeric‘ defining the quantiles. See ‘probs = seq(0, 1, 0.25)‘.
msLevel	‘integer‘
relative	‘logical‘, if set to ‘TRUE‘ the relative retention time will be returned instead of the abolute retention time
...	not used here

Details

is_a: QC:4000004 ! n-tuple is_a: QC:4000010 ! ID free is_a: QC:4000021 ! retention time metric is_a: QC:4000022 ! chromatogram metric

Value

‘numeric‘ of length equal to length ‘probs‘ with the relative duration (duration divided by the total run time) after which the TIC exceeds the respective quantile of the TIC.

Author(s)

Thomas Naake, <thomasnaake@googlemail.com>, Johannes Rainer

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
spd$rtime <- c(9.44, 9.44, 15.84)
sps <- Spectra(spd)
rtOverTicQuantiles(spectra = sps, msLevel = 2L)
```

shinyMsQuality *Shiny application to visualize quality metrics*

Description

The function `shinyMsQuality` function starts a shiny application to visualize the quality metrics interactively. It allows to display all metrics contained in qc.

The function accepts the output of `calculateMetrics`, `calculateMetricsFromSpectra`, or `calculateMetricsFromMsExp`

Usage

```
shinyMsQuality(qc)
```

Arguments

`qc`	`matrix`, contains the calculated quality metrics, the columns contain the metrics and the rows the samples

Details

The plots within the shiny application can be saved by clicking on the download button.

Value

`shiny`

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(msdata)
library(MsExperiment)
library(S4Vectors)
msexp <- MsExperiment()
sd <- DataFrame(sample_id = c("QC1", "QC2"),
    sample_name = c("QC Pool", "QC Pool"), injection_idx = c(1, 3))
sampleData(msexp) <- sd

## define file names containing spectra data for the samples and
## add them, along with other arbitrary files to the experiment
fls <- dir(system.file("sciex", package = "msdata"), full.names = TRUE)
experimentFiles(msexp) <- MsExperimentFiles(
    mzML_files = fls,
    annotations = "internal_standards.txt")
## link samples to data files: first sample to first file in "mzML_files",
## second sample to second file in "mzML_files"
msexp <- linkSampleData(msexp, with = "experimentFiles.mzML_files",
    sampleIndex = c(1, 2), withIndex = c(1, 2))
msexp <- linkSampleData(msexp, with = "experimentFiles.annotations",
    sampleIndex = c(1, 2), withIndex = c(1, 1))

library(Spectra)
## import the data and add it to the mse object
spectra(msexp) <- Spectra(fls, backend = MsBackendMzR())

## define the quality metrics to be calculated
metrics <- c("areaUnderTic", "rtDuration", "msSignal10xChange")

## calculate the metrics
## additional parameters passed to the quality metrics functions
## (msLevel is an argument of areaUnderTic and msSignal10xChange,
## relativeTo is an argument of msSignal10xChange)
qc <- calculateMetricsFromMsExperiment(msexp = msexp, metrics = metrics,
    msLevel = 1, relativeTo = "Q1", change = "jump")
rownames(qc) <- c("Sample 1", "Sample 2")

if (interactive())
    shinyMsQuality(qc = qc)
```

ticQuartileToQuartileLogRatio

MS1 TIC-change quartile ratios (MS:4000057) or MS1 TIC quartile ratios (MS:4000058)

Description

For calculation of MS:400057 set `mode = "TIC_change"`.

"The log ratios of successive TIC-change quartiles. The TIC changes are the list of MS1 total ion current (TIC) value changes from one to the next scan, produced when each MS1 TIC is subtracted from the preceding MS1 TIC. The metric's value triplet represents the log ratio of the TIC-change Q2 to Q1, Q3 to Q2, TIC-change-max to Q3" [PSI:MS] id: MS:4000057

For calculation of MS:400058 set `mode = "TIC"`.

The log ratios of successive TIC quartiles. The metric's value triplet represents the log ratios of TIC-Q2 to TIC-Q1, TIC-Q3 to TIC-Q2, TIC-max to TIC-Q3." [PSI:MS] id: MS:4000058

Usage

```
ticQuartileToQuartileLogRatio(
  spectra,
  relativeTo = "previous",
  mode = "TIC_change",
  msLevel = 1L,
  ...
)
```

Arguments

`spectra`	‘Spectra‘ object
`relativeTo`	‘character(1)‘, one of ‘"Q1"‘ or ‘"previous"‘
`mode`	‘character(1)‘, one of ‘"TIC_change"‘ or ‘"TIC"‘
`msLevel`	‘integer‘
`...`	not used here

Details

is_a: MS:4000004 ! n-tuple relationship: has_metric_category MS:4000009 ! ID free metric relationship: has_metric_category MS:4000012 ! single run based metric relationship: has_metric_category MS:4000017 ! chromatogram metric relationship: has_metric_category MS:4000021 ! MS1 metric relationship: has_value_type xsd:float ! The allowed value-type for this CV term relationship: has_value_concept STATO:0000105 ! log signal intensity ratio

The metric is calculated as follows: (1) the TIC (‘ionCount‘) of the ‘spectra‘ is calculated per scan event (with spectra ordered by retention time),

(2) for *MS:4000057*, the differences between TIC values are calculated between subsequent scan events, for *MS:4000058*, the TIC values between subsequent scan events are taken as they are,

(3) for *MS:4000057* and *MS:4000058* the ratios between the 25%, 50%, 75%, and 100% quantile to the 25 calculated. Alternatively, if ‘relativeTo = "Q1"‘, the ratios are calculated between the 50%/25%, 75%/25%, and 100%/25% quantiles.

The ‘log‘ values of the ratios are returned.

Value

‘numeric(1)‘

Note

This function interprets the *quantiles* from the [PSI:QC] definition as *quartiles*, i.e. the 0, 25, 50, 75 and 100% quantiles are used.

Author(s)

Thomas Naake, `<thomasnaake@googlemail.com>`

Examples

```
library(S4Vectors)
library(Spectra)

spd <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    polarity = c(1L, 1L, 1L),
    id = c("HMDB0000001", "HMDB0000001", "HMDB0001847"),
    name = c("1-Methylhistidine", "1-Methylhistidine", "Caffeine"))
## Assign m/z and intensity values
```

```
spd$mz <- list(
    c(109.2, 124.2, 124.5, 170.16, 170.52),
    c(83.1, 96.12, 97.14, 109.14, 124.08, 125.1, 170.16),
    c(56.0494, 69.0447, 83.0603, 109.0395, 110.0712,
        111.0551, 123.0429, 138.0662, 195.0876))
spd$intensity <- list(
    c(3.407, 47.494, 3.094, 100.0, 13.240),
    c(6.685, 4.381, 3.022, 16.708, 100.0, 4.565, 40.643),
    c(0.459, 2.585, 2.446, 0.508, 8.968, 0.524, 0.974, 100.0, 40.994))
sps <- Spectra(spd)

## MS:4000057
ticQuartileToQuartileLogRatio(spectra = sps, relativeTo = "previous",
    msLevel = 2L, mode = "TIC_change")
ticQuartileToQuartileLogRatio(spectra = sps, relativeTo = "Q1",
    msLevel = 2L, mode = "TIC_change")

## MS:4000058
ticQuartileToQuartileLogRatio(spectra = sps, relativeTo = "previous",
    msLevel = 2L, mode = "TIC")
ticQuartileToQuartileLogRatio(spectra = sps, relativeTo = "Q1",
    msLevel = 2L, mode = "TIC")
```

Index