

mzR

March 24, 2012

metadata

Access the metadata from an 'mzR' object.

Description

Accessors to the analytical setup metadata of a run. `runInfo` will show a summary of the experiment as a named list, including `scanCount`, `lowMZ`, `highMZ`, `startMZ`, `endMZ`, `dStartTime` and `dEndTime`. The `instrumentInfo` method returns a named list including instrument manufacturer, model, ionisation technique, analyzer and detector. These individual pieces of information can also be directly accessed by the specific methods.

Usage

```
runInfo(object)
analyzer(object)
detector(object)
instrumentInfo(object)
ionisation(object)
manufacturer(object)
model(object)
```

Arguments

`object` An instantiated `mzR` object.

Author(s)

Steffen Neumann and Laurent Gatto

See Also

See for example [peaks](#) to access the data for the spectra in a "`mzR`" class.

Examples

```

library(msdata)
filepath <- system.file("microtofq", package = "msdata")
file <- list.files(filepath, pattern="MM14.mzML",
                  full.names=TRUE, recursive = TRUE)
mz <- openMSfile(file)
fileName(mz)
runInfo(mz)
close(mz)

```

mzR-class

Class 'mzR' and sub-classes

Description

The class `mzR` is the main class for the common mass spectrometry formats. It is a virtual class and thus not supposed to be instantiated directly.

The sub-classes implement specific APIs to access the underlying data and metadata in the files. Currently, `mzRramp` is the only available implementation. It uses the ISB 'RAMP' random access C/C++ API to access the relevant information in `mzData`, `mzXML` and `mzML` files.

Additional sub-classes using the proteowizard API and netCDF are planned.

Objects from the Class

`mzR` is a virtual class, so instances cannot be created.

Objects can be created by calls of the form `new("mzRramp", ...)`, but more often they will be created with [openMSfile](#).

Slots

fileName: Object of class `character` storing the original filename used when the instance was created.

backend: One of the implemented backends or `NULL`.

.__classVersion__: Object of class `"Versioned"`, from Biobase.

Extends

Class `"Versioned"`, directly.

Methods

Methods currently implemented for `mzR`

fileName signature(object = "mzR"): ...

Methods currently implemented for `mzRramp`

analyzer signature(object = "mzRramp"): ...

close signature(con = "mzRramp"): ...

detector signature(object = "mzRramp"): ...

```
fileName signature(object = "mzRramp"): ...
get3Dmap signature(object = "mzRramp"): ...
header signature(object = "mzRramp", scans = "missing"): ...
header signature(object = "mzRramp", scans = "numeric"): ...
header signature(object = "mzRnetCDF", scans = "missing"): ...
header signature(object = "mzRnetCDF", scans = "numeric"): ...
initializeRamp signature(object = "mzRramp"): ...
instrumentInfo signature(object = "mzRramp"): ...
ionisation signature(object = "mzRramp"): ...
isInitialized signature(object = "mzRramp"): ...
length signature(x = "mzRramp"): ...
manufacturer signature(object = "mzRramp"): ...
model signature(object = "mzRramp"): ...
peaksCount signature(object = "mzRramp", scans = "missing"): ...
peaksCount signature(object = "mzRramp", scans = "numeric"): ...
peaks signature(object = "mzRramp", scans = "missing"): ...
peaks signature(object = "mzRramp", scans = "numeric"): ...
peaks signature(object = "mzRnetCDF", scans = "missing"): ...
peaks signature(object = "mzRnetCDF", scans = "numeric"): ...
runInfo signature(object = "mzRramp"): ...
```

Author(s)

Steffen Neumann and Laurent Gatto

References

RAMP: <http://tools.proteomecenter.org/wiki/index.php?title=Software:RAMP> Proteowizard: <http://proteowizard.sourceforge.net/>

Examples

```
library(msdata)
filepath <- system.file("microtofq", package = "msdata")
file <- list.files(filepath, pattern="MM14.mzML",
                  full.names=TRUE, recursive = TRUE)
mzml <- openMSfile(file)
close(mzml)
```

mzR-package	<i>parser for mzXML, mzData and mzML files (mass spectrometry data)</i>
-------------	---

Description

The mzR package is a library purely for accessing mass spectrometry data in a wide range of formats. Several backend libraries are used, such as the ISB random access parser (RAMP) for mass spectrometry mzXML, mzData and mzML files. The package contains the original RAMP code written by the ISB, and a subset of the proteowizard library for mzML.

Details

Further information is available in the following vignette:

mzR mzR, Ramp, mzXML, mzData, mzML (source, pdf)

Author(s)

Bernd Fischer, Steffen Neumann, Laurent Gatto

Maintainers: Bernd Fischer <bernd.fischer@embl.de>, Steffen Neumann <sneumann@ipb-halle.de>, Laurent Gatto <lg390@cam.ac.uk>

References

<http://dx.doi.org/10.1074/mcp.R110.000133>

openMSfile	<i>Create and check mzR objects from netCDF, mzXML, mzData or mzML files.</i>
------------	---

Description

The openMSfile constructor will create a new format-specific mzR object, open 'filename' file and all header information is loaded as a Rcpp module and made accessible through the ramp slot of the resulting object.

Usage

```
openMSfile(filename, backend=c("Ramp", "netCDF"), verbose = FALSE)
initializeRamp(object)
isInitialized(object)
fileName(object)
```

Arguments

filename	Path name of the netCDF, mzData, mzXML or mzML file to read.
backend	A character specifying with backend API to use. Currently only 'Ramp' and 'netCDF' are available. 'pwiz' will be added in future version.
object	An instantiated mzR object.
verbose	Enable verbose output.

Author(s)

Steffen Neumann and Laurent Gatto

Examples

```
library(msdata)
filepath <- system.file("microtofq", package = "msdata")
file <- list.files(filepath, pattern="MM14.mzML",
                  full.names=TRUE, recursive = TRUE)
mz <- openMSfile(file)
fileName(mz)
isInitialized(mz)
runInfo(mz)
close(mz)
```

peaks

Access the raw data from an 'mzR' object.

Description

Access the MS raw data. The `peaks` and `peaksCount` functions return the (m/z,intensity) pairs and the number peaks in the spectrum/spectra. `peaks` returns a single matrix if `scans` is a numeric of length 1 and a list of matrices if several scans are asked for or no `scans` argument is provided (i.e all spectra in the object are returned). `peaksCount` will return a numeric of length n.

The `header` function returns a list containing `seqNum`, `acquisitionNum`, `msLevel`, `peaksCount`, `totIonCurrent`, `retentionTime`, `basePeakMZ`, `basePeakIntensity`, `collisionEnergy`, `ionisationEnergy`, `lowM`, `highMZ`, `precursorScanNum`, `precursorMZ`, `precursorCharge`, `precursorIntensity`, `mergedScan`, `mergedResultScanNum`, `mergedResultStartScanNum` and `mergedResultEndScanNum`, when available in the original file. If multiple scans are queried, a `data.frame` is returned with the scans reported along the rows.

The `get3Dmap` function performs a simple resampling between `lowMz` and `highMz` with `resMz` resolution. A matrix of dimensions `length(scans)` times `seq(lowMz, highMz, resMz)` is returned.

Usage

```
header(object, scans, ...)
```

```
peaksCount(object, scans, ...)
```

```
peaks(object, scans, ...)
```

```
get3Dmap(object, scans, lowMz, highMz, resMz, ...)
```

Arguments

<code>object</code>	An instantiated <code>mzR</code> object.
<code>scans</code>	A numeric specifying which scans to return. Optional for the <code>header</code> , <code>peaks</code> and <code>peaksCount</code> methods. If omitted, the requested data for all peaks is returned.
<code>lowMz</code> , <code>highMz</code>	Numerics defining the <code>m/z</code> range to be returned.
<code>resMz</code>	a numeric defining the <code>m/z</code> resolution.
<code>...</code>	Other arguments. Currently ignored.

Author(s)

Steffen Neumann and Laurent Gatto

See Also

[instrumentInfo](#) for metadata access and the "`mzR`" class.

Examples

```
library(msdata)
filepath <- system.file("microtofq", package = "msdata")
file <- list.files(filepath, pattern="MM14.mzML",
                  full.names=TRUE, recursive = TRUE)
mz <- openMSfile(file)
runInfo(mz)
colnames(header(mz))
close(mz)
```

Index

*Topic classes

mzR-class, 2

*Topic package, file

mzR-package, 4

analyzer (*metadata*), 1

analyzer, mzRramp-method
(*mzR-class*), 2

class:mzR (*mzR-class*), 2

class:mzRnetCDF (*mzR-class*), 2

class:mzRramp (*mzR-class*), 2

close (*mzR-class*), 2

close, mzRnetCDF-method
(*mzR-class*), 2

close, mzRramp-method (*mzR-class*),
2

detector (*metadata*), 1

detector, mzRramp-method
(*mzR-class*), 2

fileName (*openMSfile*), 4

fileName, mzR-method (*mzR-class*), 2

get3Dmap (*peaks*), 5

get3Dmap, mzRramp-method
(*mzR-class*), 2

header, 5

header (*peaks*), 5

header, mzRnetCDF, missing-method
(*mzR-class*), 2

header, mzRnetCDF, numeric-method
(*mzR-class*), 2

header, mzRramp, missing-method
(*mzR-class*), 2

header, mzRramp, numeric-method
(*mzR-class*), 2

initializeRamp (*openMSfile*), 4

initializeRamp, mzRramp-method
(*mzR-class*), 2

instrumentInfo, 6

instrumentInfo (*metadata*), 1

instrumentInfo, mzRramp-method
(*mzR-class*), 2

ionisation (*metadata*), 1

ionisation, mzRramp-method
(*mzR-class*), 2

isInitialized (*openMSfile*), 4

isInitialized, mzRnetCDF-method
(*mzR-class*), 2

isInitialized, mzRramp-method
(*mzR-class*), 2

length (*mzR-class*), 2

length, mzRnetCDF-method
(*mzR-class*), 2

length, mzRramp-method
(*mzR-class*), 2

manufacturer (*metadata*), 1

manufacturer, mzRramp-method
(*mzR-class*), 2

metadata, 1

model (*metadata*), 1

model, mzRramp-method (*mzR-class*),
2

mzR, 1, 6

mzR (*mzR-package*), 4

mzR-class, 2

mzR-package, 4

mzRnetCDF-class (*mzR-class*), 2

mzRramp-class (*mzR-class*), 2

openMSfile, 2, 4

peaks, 1, 5

peaks, mzRnetCDF, missing-method
(*mzR-class*), 2

peaks, mzRnetCDF, numeric-method
(*mzR-class*), 2

peaks, mzRramp, missing-method
(*mzR-class*), 2

peaks, mzRramp, numeric-method
(*mzR-class*), 2

peaksCount (*peaks*), 5

peaksCount, mzRramp, missing-method
(*mzR-class*), 2

peaksCount, mzRramp, numeric-method
(*mzR-class*), 2

runInfo(*metadata*), 1
runInfo, mzRramp-method
(*mzR-class*), 2

Versioned, 2