Package 'mzR'

April 23, 2016

Type Package

Title parser for netCDF, mzXML, mzData and mzML and mzIdentML files (mass spectrometry data)

Version 2.4.1

Author Bernd Fischer, Steffen Neumann, Laurent Gatto, Qiang Kou

Maintainer Bernd Fischer <b.fischer@dkfz.de>,
Steffen Neumann <sneumann@ipb-halle.de>,
Laurent Gatto <lg390@cam.ac.uk>,
Qiang Kou <qkou@umail.iu.edu>

Description mzR provides a unified API to the common file formats and parsers available for mass spectrometry data. It comes with a wrapper for the ISB random access parser for mass spectrometry mzXML, mzData and mzML files. The package contains the original code written by the ISB, and a subset of the proteowizard library for mzML and mzIdentML. The netCDF reading code has previously been used in XCMS.

License Artistic-2.0

LazyLoad yes

Depends Rcpp (>= 0.10.1), methods, utils

Imports Biobase, BiocGenerics (>= 0.13.6), ProtGenerics

Suggests msdata (>= 0.3.5), RUnit, mzID, BiocStyle, knitr

VignetteBuilder knitr

LinkingTo Rcpp, zlibbioc

RcppModules Ramp, Pwiz, Ident

SystemRequirements GNU make, NetCDF

URL https://github.com/sneumann/mzR/

BugReports https://github.com/sneumann/mzR/issues/new

biocViews Infrastructure, DataImport, Proteomics, Metabolomics, MassSpectrometry

NeedsCompilation yes

R topics documented:

`mzR-package` *parser for mzXML, mzData, mzML and mzid files (mass spectrometry data)*

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 acces parser (RAMP) and ProteoWizard (pwiz) for mass spectrometry mzXML, mzData, mzML and mzid files. The package contains the original RAMP code written by the ISB, and a subset of the proteowizard library for mzML and mzid.

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, Qiang Kou

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

References

Nat Biotechnol. 2012 Oct 10;30(10):918-20. doi: 10.1038/nbt.2377. A cross-platform toolkit for mass spectrometry and proteomics. Chambers MC, Maclean B, Burke R, Amodei D, Ruderman DL, Neumann S, Gatto L, Fischer B, Pratt B, Egertson J, Hoff K, Kessner D, Tasman N, Shulman N, Frewen B, Baker TA, Brusniak MY, Paulse C, Creasy D, Flashner L, Kani K, Moulding C, Seymour SL, Nuwaysir LM, Lefebvre B, Kuhlmann F, Roark J, Rainer P, Detlev S, Hemenway T, Huhmer A, Langridge J, Connolly B, Chadick T, Holly K, Eckels J, Deutsch EW, Moritz RL, Katz JE, Agus DB, Maccoss M, Tabb DL, Mallick P. http://www.ncbi.nlm.nih.gov/pubmed/23051804

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, dStartTime and dEndTime. The instrumentInfo method returns a named list including instrument manufacturer, model, ionisation technique, analyzer and detector. mzRpwiz will give more additional information including information on sample, software using and original source file. These individual pieces of information can also be directly accessed by the specific methods. mzidInfo is used for the mzR object created from a mzid file. It returns basic information on this mzid file including file provider, creation date, software, database, enzymes and spectra data format. The mzidInfo will return the scoring results in identification. It will return different results for different searching software used.

Usage

```
runInfo(object)
chromatogramsInfo(object)
analyzer(object)
detector(object)
instrumentInfo(object)
ionisation(object)
softwareInfo(object)
sampleInfo(object)
sourceInfo(object)
model(object)
mzidInfo(object)
modifications(object, ...)
psms(object, ...)
substitutions(object)
database(object, ...)
enzymes(object)
tolerance(object)
score(x, ...)
para(object)
```

Arguments

object	An instantiated mzR object.
x	An instantiated mzR object.
...	Additional arguments, currently ignored.

Author(s)

Steffen Neumann, Laurent Gatto and Qiang Kou

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)
instrumentInfo(mz)
close(mz)

file <- system.file("mzid", "Tandem.mzid.gz", package="msdata")
mzid <- openIDfile(file)
softwareInfo(mzid)
enzymes(mzid)
```

---

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 instanciated directly.

The sub-classes implement specific APIs to access the underlying data and metadata in the files. Currently, mzRramp and mzRpwiz are available implementations. mzRramp uses the ISB 'RAMP' random access C/C++ API, and mzRpwiz uses Proteowizard to access the relevant information in mzData, mzXML and mzML files. You can also open mz5 file by using mzRpwiz.

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.

After creating a mzR, you can write it into a file. mzXML, mzML, mgf formats are supported.

Slots

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

backend: One of the implemented backens 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"): ...

Methods currently implemented for mzRpwiz

analyzer signature(object = "mzRpwiz"): ...
detector signature(object = "mzRpwiz"): ...
instrumentInfo signature(object = "mzRpwiz"): ...
ionisation signature(object = "mzRpwiz"): ...
length signature(x = "mzRpwiz"): ...

manufacturer `signature(object = "mzRpwiz")`: ...

model `signature(object = "mzRpwiz")`: ...

runInfo `signature(object = "mzRpwiz")`: ...

chromatogramsInfo `signature(object = "mzRpwiz")`: ...

Methods currently implemented for `mzRident`

mzidInfo `signature(object = "mzRident")`: ...

psms `signature(object = "mzRident")`: ...

modifications `signature(object = "mzRident")`: ...

substitutions `signature(object = "mzRident")`: ...

database `signature(x = "mzRident")`: ...

enzymes `signature(object = "mzRident")`: ...

sourceInfo `signature(object = "mzRident")`: ...

tolerance `signature(object = "mzRident")`: ...

score `signature(object = "mzRident")`: ...

para `signature(object = "mzRident")`: ...

Author(s)

Steffen Neumann, Laurent Gatto, Qiang Kou

References

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

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)

## using the pwiz backend
mzml <- openMSfile(file, backend = "pwiz")
```

`openMSfile` *Create and check mzR objects from netCDF, mzXML, mzData or mzML files.*

Description

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

The `openIDfile` constructor will create a new format-specifc `mzR` object, open 'filename' file and all information is loaded as a Rcpp module. The mzid format is supported throught `pwiz` backend. Only mzIdentML 1.1 is supported.

Usage

```
openMSfile(filename, backend=c("Ramp", "pwiz", "netCDF"), verbose = FALSE)

initializeRamp(object)

isInitialized(object)

fileName(object, ...)

openIDfile(filename, verbose = FALSE)
```

Arguments

`filename`	Path name of the netCDF, mzData, mzXML or mzML file to read/write.
`backend`	A `character` specifiying with backend API to use. Currently 'Ramp', 'netCDF' and 'pwiz' are available.
`object`	An instantiated mzR object.
`verbose`	Enable verbose output.
`...`	Additional arguments, currently ignored.

Author(s)

Steffen Neumann, Laurent Gatto, Qiang Kou

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)

## Not run:
   ## to use another backend
   mz <- openMSfile(file, backend = "pwiz")
   mz

## End(Not run)

 file <- system.file("mzid", "Tandem.mzid.gz", package="msdata")
 mzid <- openIDfile(file)
 softwareInfo(mzid)
 enzymes(mzid)
```

---

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 oject are retured). 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 reMz resolution. A matrix of dimensions length(scans) times seq(lowMz,highMz,resMz) is returned.

Usage

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

peaksCount(object, scans, ...)

peaks(object, ...)

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

Arguments

object	An instantiated mzR object.

scans	A numeric specifying which scans to return. Optional for the header, peaks and peaksCount methods. If ommited, the requested data for all peaks is returned.
lowMz, highMz	Numerics defining the m/z range to be returned.
resMz	a numeric defining the m/z resolution.
...	Other arguments. A scan parameter can be passed to peaks.

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