

# Package ‘mzR’

April 15, 2017

**Type** Package

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

**Version** 2.8.1

**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, XML

**VignetteBuilder** knitr

**LinkingTo** Rcpp, zlibbioc

**RcppModules** Ramp, Pwiz, Ident

**SystemRequirements** C++11, GNU make, NetCDF

**URL** <https://github.com/sneumann/mzR/>

**BugReports** <https://github.com/sneumann/mzR/issues/>

**biocViews** Infrastructure, DataImport, Proteomics, Metabolomics,  
MassSpectrometry

**NeedsCompilation** yes

## R topics documented:

mzR-package	2
isolationWindow-methods	3
metadata	3
mzR-class	5
openMSfile	7
peaks	8
pwiz.version	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

mzR-package	<i>parser for mzXML, mzData, mzML and mzid 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) 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 <l390@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>

---

 isolationWindow-methods

*Returns the ion selection isolation window*


---

### Description

The methods return matrices of lower (column low) and upper (column high) isolation window offsets. Matrices are returned as a list of length equal to the number of input files (provided as file names of raw mass spectrometry data objects, see below). By default (i.e when `unique. = TRUE`), only unique offsets are returned, as they are expected to be identical for all spectra per acquisition. If this is not the case, a message is displayed.

### Methods

`signature(object = "character", unique. = "logical", simplify = "logical")` Returns the isolation window for the file object. By default, only unique isolation windows are returned per file (`unique = TRUE`); if set to `FALSE`, a matrix with as many rows as there are MS2 spectra. If only one file passed as an input and `simplify` is set to `TRUE` (default), the resulting list of length 1 is simplified to a matrix.

`signature(object = "mzRpwiz", unique. = "logical", simplify = "logical")` As above for `mzRpwiz` objects.

`signature(object = "mzRRamp", unique. = "logical", simplify = "logical")` As above for `mzRRamp` object.

### Author(s)

Laurent Gatto <lg390@cam.ac.uk> based on the functionality from the `msPurity::get_isolation_offsets` function.

### Examples

```
library("msdata")
f <- msdata::proteomics(full.names = TRUE, pattern = "TMT_")
isolationWindow(f)

rw <- openMSfile(f)
isolationWindow(rw)
str(isolationWindow(rw, unique = FALSE))
```

---

 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 used 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

<code>object</code>	An instantiated <code>mzR</code> object.
<code>x</code>	An instantiated <code>mzR</code> object.
<code>...</code>	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 instantiated directly.

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

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 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"): ...

Methods currently implemented for mzRp wiz

**analyzer** signature(object = "mzRp wiz"): ...  
**detector** signature(object = "mzRp wiz"): ...  
**instrumentInfo** signature(object = "mzRp wiz"): ...  
**ionisation** signature(object = "mzRp wiz"): ...  
**length** signature(x = "mzRp wiz"): ...  
**manufacturer** signature(object = "mzRp wiz"): ...  
**model** signature(object = "mzRp wiz"): ...  
**runInfo** signature(object = "mzRp wiz"): ...  
**chromatogramsInfo** signature(object = "mzRp wiz"): ...

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.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)

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

---

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 or pwiz slot of the resulting object.

The openIDfile constructor will create a new format-specific mzR object, open 'filename' file and all information is loaded as a Rcpp module. The mzid format is supported through 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 specifying 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 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 `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 omitted, 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)
```

---

pwiz.version

*Get the version number of pwiz backend.*

---

**Description**

Get the version number of pwiz backend.

**Usage**

```
pwiz.version()
```

# Index

## \*Topic **classes**

mzR-class, 5

## \*Topic **methods**

isolationWindow-methods, 3

## \*Topic **package, file**

mzR-package, 2

analyzer (metadata), 3

analyzer, mzRnetCDF-method (mzR-class), 5

analyzer, mzRpwiz-method (mzR-class), 5

analyzer, mzRramp-method (mzR-class), 5

chromatogramsInfo (metadata), 3

chromatogramsInfo, mzRpwiz-method  
(mzR-class), 5

class:mzR (mzR-class), 5

class:mzRident (mzR-class), 5

class:mzRnetCDF (mzR-class), 5

class:mzRpwiz (mzR-class), 5

class:mzRramp (mzR-class), 5

close (mzR-class), 5

close, mzRnetCDF-method (mzR-class), 5

close, mzRpwiz-method (mzR-class), 5

close, mzRramp-method (mzR-class), 5

database (metadata), 3

database, mzRident-method (mzR-class), 5

detector (metadata), 3

detector, mzRnetCDF-method (mzR-class), 5

detector, mzRpwiz-method (mzR-class), 5

detector, mzRramp-method (mzR-class), 5

enzymes (metadata), 3

enzymes, mzRident-method (mzR-class), 5

fileName (openMSfile), 7

fileName, mzR-method (mzR-class), 5

get3Dmap (peaks), 8

get3Dmap, mzRpwiz-method (mzR-class), 5

get3Dmap, mzRramp-method (mzR-class), 5

header, 8

header (peaks), 8

header, mzRnetCDF, missing-method  
(mzR-class), 5

header, mzRnetCDF, numeric-method  
(mzR-class), 5

header, mzRpwiz, missing-method  
(mzR-class), 5

header, mzRpwiz, numeric-method  
(mzR-class), 5

header, mzRramp, missing-method  
(mzR-class), 5

header, mzRramp, numeric-method  
(mzR-class), 5

initializeRamp (openMSfile), 7

initializeRamp, mzRramp-method  
(mzR-class), 5

instrumentInfo, 9

instrumentInfo (metadata), 3

instrumentInfo, mzRnetCDF-method  
(mzR-class), 5

instrumentInfo, mzRpwiz-method  
(mzR-class), 5

instrumentInfo, mzRramp-method  
(mzR-class), 5

ionisation (metadata), 3

ionisation, mzRnetCDF-method  
(mzR-class), 5

ionisation, mzRpwiz-method (mzR-class), 5

ionisation, mzRramp-method (mzR-class), 5

isInitialized (openMSfile), 7

isInitialized, mzRnetCDF-method  
(mzR-class), 5

isInitialized, mzRramp-method  
(mzR-class), 5

isolationWindow

(isolationWindow-methods), 3

isolationWindow, character-method  
(isolationWindow-methods), 3

isolationWindow, mzRpwiz-method  
(isolationWindow-methods), 3

isolationWindow, mzRramp-method  
(isolationWindow-methods), 3

isolationWindow-methods, 3

- length (mzR-class), 5
- length, mzRident-method (mzR-class), 5
- length, mzRnetCDF-method (mzR-class), 5
- length, mzRp wiz-method (mzR-class), 5
- length, mzRramp-method (mzR-class), 5
  
- manufacturer (metadata), 3
- manufacturer, mzRnetCDF-method (mzR-class), 5
- manufacturer, mzRp wiz-method (mzR-class), 5
- manufacturer, mzRramp-method (mzR-class), 5
- metadata, 3
- model (metadata), 3
- model, mzRnetCDF-method (mzR-class), 5
- model, mzRp wiz-method (mzR-class), 5
- model, mzRramp-method (mzR-class), 5
- modifications (metadata), 3
- modifications, mzRident-method (mzR-class), 5
- mzidInfo (metadata), 3
- mzidInfo, mzRident-method (mzR-class), 5
- mzR, 4, 9
- mzR (mzR-package), 2
- mzR-class, 5
- mzR-package, 2
- mzRident-class (mzR-class), 5
- mzRnetCDF-class (mzR-class), 5
- mzRp wiz-class (mzR-class), 5
- mzRramp-class (mzR-class), 5
  
- openIDfile (openMSfile), 7
- openMSfile, 5, 7
  
- para (metadata), 3
- para, mzRident-method (mzR-class), 5
- peaks, 4, 8
- peaks, mzRnetCDF-method (mzR-class), 5
- peaks, mzRp wiz-method (mzR-class), 5
- peaks, mzRramp-method (mzR-class), 5
- peaksCount (peaks), 8
- peaksCount, mzRp wiz, missing-method (mzR-class), 5
- peaksCount, mzRp wiz, numeric-method (mzR-class), 5
- peaksCount, mzRramp, missing-method (mzR-class), 5
- peaksCount, mzRramp, numeric-method (mzR-class), 5
- psms (metadata), 3
- psms, mzRident-method (mzR-class), 5
- pwiz.version, 9
  
- runInfo (metadata), 3
- runInfo, mzRnetCDF-method (mzR-class), 5
- runInfo, mzRp wiz-method (mzR-class), 5
- runInfo, mzRramp-method (mzR-class), 5
  
- sampleInfo (metadata), 3
- sampleInfo, mzRp wiz-method (mzR-class), 5
- score (metadata), 3
- score, mzRident-method (mzR-class), 5
- softwareInfo (metadata), 3
- softwareInfo, mzRident-method (mzR-class), 5
- softwareInfo, mzRp wiz-method (mzR-class), 5
- sourceInfo (metadata), 3
- sourceInfo, mzRident-method (mzR-class), 5
- sourceInfo, mzRp wiz-method (mzR-class), 5
- substitutions (metadata), 3
- substitutions, mzRident-method (mzR-class), 5
  
- tolerance (metadata), 3
- tolerance, mzRident-method (mzR-class), 5
  
- Versioned, 5