Package 'MsBackendMassbank'

May 18, 2024

Title Mass Spectrometry Data Backend for MassBank record Files

Version 1.12.0

Description Mass spectrometry (MS) data backend supporting import and export of MS/MS library spectra from MassBank record files. Different backends are available that allow handling of data in plain MassBank text file format or allow also to interact directly with MassBank SQL databases. Objects from this package are supposed to be used with the Spectra Bioconductor package. This package thus adds MassBank support to the Spectra package.

Depends R (>= 4.0), Spectra (>= 1.9.12)

Imports BiocParallel, S4Vectors, IRanges, methods, ProtGenerics (>= 1.35.3), MsCoreUtils, DBI, utils

Suggests testthat, knitr (>= 1.1.0), roxygen2, BiocStyle (>= 2.5.19), RSQLite, rmarkdown

License Artistic-2.0

LazyData yes

Encoding UTF-8

VignetteBuilder knitr

BugReports https://github.com/RforMassSpectrometry/MsBackendMassbank/issues

URL https://github.com/RforMassSpectrometry/MsBackendMassbank

biocViews Infrastructure, MassSpectrometry, Metabolomics, DataImport

Roxygen list(markdown=TRUE)

RoxygenNote 7.3.1

Collate 'hidden_aliases.R' 'MsBackendMassbank.R' 'MsBackendMassbankSql-functions.R' 'MsBackendMassbankSql.R' 'functions-massbank.R'

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

git_branch RELEASE_3_19

git_last_commit 78e5e72

git_last_commit_date 2024-04-30

Contents

---

hidden_aliases   *Internal page for hidden aliases*

Description

For S4 methods that require a documentation entry but only clutter the index.

---

metaDataBlocks   *Metadata blocks to be read*

Description

metaDataBlocks returns a data.frame with the MassBank metadata blocks and whether they should be imported by default from the MassBank text files.

Usage

```
metaDataBlocks()
```

Value

A data.frame with metadata blocks.

Author(s)

Michael Witting

Examples

```
metaDataBlocks()
```

MsBackendMassbank   *MS data backend for mgf files*

Description

The MsBackendMassbank class supports import of MS/MS spectra data from MS/MS spectrum data from Massbank files. After initial import, the full MS data is kept in memory. MsBackendMassbank extends the `MsBackendDataFrame()` backend directly and supports thus the `applyProcessing()` function to make data manipulations persistent.

New objects are created with the MsBackendMassbank function. The backendInitialize method has to be subsequently called to initialize the object and import MS/MS data from (one or more) MassBank files. Optional parameter `nonStop` allows to specify whether the import returns with an error if one of the text files lacks required data, such as `mz` and `intensity` values (default `nonStop = FALSE`), or whether only affected file(s) is(are) skipped and a warning is shown (`nonStop = TRUE`). Note that any other error will abort import regardless of parameter `nonStop`.

Usage

```
## S4 method for signature 'MsBackendMassbank'
backendInitialize(
  object,
  files,
  metaBlocks = metaDataBlocks(),
  nonStop = FALSE,
  ...,
  BPPARAM = bpparam()
)

MsBackendMassbank()

## S4 method for signature 'MsBackendMassbank'
spectraVariableMapping(object, format = c("Massbank"))

## S4 method for signature 'MsBackendMassbank'
export(
  object,
  x,
  file = tempfile(),
  mapping = spectraVariableMapping(MsBackendMassbank()),
  ...
)
```

Arguments

object	Instance of `MsBackendMassbank` class.
files	character with the (full) file name(s) of the MassBank file(s) from which MS/MS data should be imported.

metaBlocks	data.frame indicating which metadata shall be imported. Default is [metaDataBlocks()]().
nonStop	logical(1) whether import should be stopped if an xml file does not contain all required fields. Defaults to nonStop = FALSE.
...	Currently ignored.
BPPARAM	Parameter object defining the parallel processing setup to import data in parallel. Defaults to BPPARAM = bpparam(). See [bpparam()]() for more information.
format	for spectraVariableMapping: character(1) defining the format to be used. Currently only format = "Massbank" is supported.
x	[Spectra()]() object that should be exported.
file	for export: character(1) defining the output file.
mapping	for export: named character vector allowing to specify how fields from the Massbank file should be renamed. Names are supposed to be the spectra variable name and values of the vector the field names in the Massbank file. See output of spectraVariableMapping(MsBackendMassbank()) for the expected format.

Value

backendInitialize and MsBackendMassbank return an instance of MsBackendMassbank-class.

Author(s)

Michael Witting

Examples

```
## Create an MsBackendMassbank backend and import data from a test file.
fls <- dir(system.file("extdata", package = "MsBackendMassbank"),
    full.names = TRUE, pattern = "txt$")
be <- backendInitialize(MsBackendMassbank(), fls)
be

be$msLevel
be$intensity
be$mz

## Initializing a backend reading additional metadata columns/information
mb <- metaDataBlocks()
mb
mb[1, 2] <- TRUE

be <- backendInitialize(MsBackendMassbank(), fls, metaBlocks = mb)
spectraVariables(be)
be$instrument
```

MsBackendMassbankSql   *MS backend accessing the MassBank MySQL database*

Description

The MsBackendMassbankSql provides access to mass spectrometry data from MassBank by directly accessing its MySQL/MariaDb database. In addition it supports adding new spectra variables or *locally* changing spectra variables provided by MassBank (without changing the original values in the database).

Note that MsBackendMassbankSql requires a local installation of the MassBank database since direct database access is not supported for the *main* MassBank instance.

Also, some of the fields in the MassBank database are not directly compatible with Spectra, such as the *collision energy* which is not available as a numeric value. The collision energy as available in MassBank is reported as spectra variable "collision_energy_text". Also, precursor m/z values reported for some spectra can not be converted to a numeric and hence NA is reported with the spectra variable precursorMz for these spectra. The variable "precursor_mz_text" can be used to get the *original* precursor m/z reported in MassBank.

Finally, MsBackendMassbankSql does **not support** parallel processing because the database connection stored within the object can not be shared acrcoss parallel processes. All functions on Spectra objects with a MsBackendMassbankSql will (silently) disable parallel processing even if the user provides a dedicated parallel processing setup with the BPPARAM parameter.

Usage

```
MsBackendMassbankSql()

## S4 method for signature 'MsBackendMassbankSql'
backendInitialize(object, dbcon, ...)

## S4 method for signature 'MsBackendMassbankSql'
peaksData(object, columns = peaksVariables(object))

## S4 method for signature 'MsBackendMassbankSql'
dataStorage(object)

## S4 replacement method for signature 'MsBackendMassbankSql'
intensity(object) <- value

## S4 replacement method for signature 'MsBackendMassbankSql'
mz(object) <- value

## S4 method for signature 'MsBackendMassbankSql'
reset(object)

## S4 method for signature 'MsBackendMassbankSql'
spectraData(object, columns = spectraVariables(object))
```

```
## S4 method for signature 'MsBackendMassbankSql'
spectraNames(object)

## S4 replacement method for signature 'MsBackendMassbankSql'
spectraNames(object) <- value

## S4 method for signature 'MsBackendMassbankSql'
tic(object, initial = TRUE)

## S4 method for signature 'MsBackendMassbankSql'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'Spectra'
compounds(object, ...)

## S4 method for signature 'MsBackendMassbankSql'
compounds(object, ...)

## S4 replacement method for signature 'MsBackendMassbankSql'
x$name <- value

## S4 method for signature 'MsBackendMassbankSql'
precScanNum(object)

## S4 method for signature 'MsBackendMassbankSql'
backendBpparam(object, BPPARAM = bpparam())
```

Arguments

object	Object extending MsBackendMassbankSql.
dbcon	For backendInitialize,MsBackendMassbankSql: SQL database connection to the MassBank (MariaDb) database.
...	Additional arguments.
columns	For spectraData accessor: optional character with column names (spectra variables) that should be included in the returned DataFrame. By default, all columns are returned. For peaksData accessor: optional character with requested columns in the individual matrix of the returned list. Use peaksVariables(object) for supported columns.
value	replacement value for <- methods. See individual method description or expected data type.
initial	For tic: logical(1) whether the initially reported total ion current should be reported, or whether the total ion current should be (re)calculated on the actual data (initial = FALSE).
x	Object extending MsBackendMassbankSql.
i	For [: integer, logical or character to subset the object.

j	For [: not supported.
drop	For [: not considered.
name	name of the variable to replace for <- methods. See individual method description or expected data type.
BPPARAM	for backendBpparam: BiocParallel parallel processing setup. See bpparam() for more information.
spectraVariables	
	For selectSpectraVariables: character with the names of the spectra variables to which the backend should be subsetted.

Value

See documentation of respective function.

Supported Backend functions

The following functions are supported by the MsBackendMassbankSql.

- [: subset the backend. Only subsetting by element (*row*/i) is allowed

- $, $<-: access or set/add a single spectrum variable (column) in the backend.

- acquisitionNum: returns the acquisition number of each spectrum. Returns an integer of length equal to the number of spectra (with NA_integer_ if not available).

- peaksData returns a list with the spectras' peak data. The length of the list is equal to the number of spectra in object. Each element of the list is a matrix with columns "mz" and "intensity". For an empty spectrum, a matrix with 0 rows and two columns (named mz and intensity) is returned. Parameter columns allows to select which peaks variables to return, but supports currently only "mz" and "intensity".

- backendBpparam: whether the backend supports parallel processing. Takes a MsBackendMassbankSql and a parallel processing setup (see bpparam() for details) as input and **always** returns a SerialParam(). This function can be used to test whether a provided parallel processing setup is supported by the backend and returns the supported setup.

- backendInitialize: initialises the backend by retrieving the IDs of all spectra in the database. Parameter dbcon with the connection to the MassBank MySQL database is required.

- dataOrigin: gets a character of length equal to the number of spectra in object with the *data origin* of each spectrum. This could e.g. be the mzML file from which the data was read.

- dataStorage: returns "<MassBank>" for all spectra.

- centroided, centroided<-: gets or sets the centroiding information of the spectra. centroided returns a logical vector of length equal to the number of spectra with TRUE if a spectrum is centroided, FALSE if it is in profile mode and NA if it is undefined. See also isCentroided for estimating from the spectrum data whether the spectrum is centroided. value for centroided<- is either a single logical or a logical of length equal to the number of spectra in object.

- collisionEnergy, collisionEnergy<-: gets or sets the collision energy for all spectra in object. collisionEnergy returns a numeric with length equal to the number of spectra (NA_real_ if not present/defined), collisionEnergy<- takes a numeric of length equal to the number of spectra in object. Note that the collision energy description from MassBank are provided as spectra variable "collisionEnergyText".

- intensity: gets the intensity values from the spectra. Returns a [NumericList()](#) of numeric vectors (intensity values for each spectrum). The length of the list is equal to the number of spectra in object.

- ionCount: returns a numeric with the sum of intensities for each spectrum. If the spectrum is empty (see isEmpty), NA_real_ is returned.

- isCentroided: a heuristic approach assessing if the spectra in object are in profile or centroided mode. The function takes the qtl th quantile top peaks, then calculates the difference between adjacent m/z value and returns TRUE if the first quartile is greater than k. (See Spectra:::.isCentroided for the code.)

- isEmpty: checks whether a spectrum in object is empty (i.e. does not contain any peaks). Returns a logical vector of length equal number of spectra.

- isolationWindowLowerMz, isolationWindowLowerMz<-: gets or sets the lower m/z boundary of the isolation window.

- isolationWindowTargetMz, isolationWindowTargetMz<-: gets or sets the target m/z of the isolation window.

- isolationWindowUpperMz, isolationWindowUpperMz<-: gets or sets the upper m/z boundary of the isolation window.

- isReadOnly: returns a logical(1) whether the backend is *read only* or does allow also to write/update data.

- length: returns the number of spectra in the object.

- lengths: gets the number of peaks (m/z-intensity values) per spectrum. Returns an integer vector (length equal to the number of spectra). For empty spectra, 0 is returned.

- msLevel: gets the spectra's MS level. Returns an integer vector (of length equal to the number of spectra) with the MS level for each spectrum (or NA_integer_ if not available).

- mz: gets the mass-to-charge ratios (m/z) from the spectra. Returns a [NumericList()](#) or length equal to the number of spectra, each element a numeric vector with the m/z values of one spectrum.

- polarity, polarity<-: gets or sets the polarity for each spectrum. polarity returns an integer vector (length equal to the number of spectra), with 0 and 1 representing negative and positive polarities, respectively. polarity<- expects an integer vector of length 1 or equal to the number of spectra.

- precursorCharge, precursorIntensity, precursorMz, precScanNum, precAcquisitionNum: get the charge (integer), intensity (numeric), m/z (numeric), scan index (integer) and acquisition number (interger) of the precursor for MS level 2 and above spectra from the object. Returns a vector of length equal to the number of spectra in object. NA are reported for MS1 spectra of if no precursor information is available.

- reset: restores the backend to its original state, i.e. deletes all locally modified data and reinitializes the backend to the full data available in the database.

- rtime, rtime<-: gets or sets the retention times for each spectrum (in seconds). rtime returns a numeric vector (length equal to the number of spectra) with the retention time for each spectrum. rtime<- expects a numeric vector with length equal to the number of spectra.

- scanIndex: returns an integer vector with the *scan index* for each spectrum. This represents the relative index of the spectrum within each file. Note that this can be different to the acquisitionNum of the spectrum which is the index of the spectrum as reported in the mzML file.

- `selectSpectraVariables`: reduces the information within the backend to the selected spectra variables.

- `smoothed,smoothed<-`: gets or sets whether a spectrum is *smoothed*. smoothed returns a `logical` vector of length equal to the number of spectra. smoothed<- takes a `logical` vector of length 1 or equal to the number of spectra in `object`.

- `spectraData`: gets general spectrum metadata (annotation, also called header). `spectraData` returns a `DataFrame`. Note that replacing the spectra data with `spectraData<-` is not supported.

- `spectraNames`: returns a `character` vector with the names of the spectra in `object`.

- `spectraVariables`: returns a `character` vector with the available spectra variables (columns, fields or attributes) available in `object`. This should return **all** spectra variables which are present in `object`, also `"mz"` and `"intensity"` (which are by default not returned by the `spectraVariables,Spectra` method).

- `tic`: gets the total ion current/count (sum of signal of a spectrum) for all spectra in `object`. By default, the value reported in the original raw data file is returned. For an empty spectrum, `NA_real_` is returned.

Not supported Backend functions

The following functions are not supported by the `MsBackendMassbankSql` since the original data can not be changed.

`backendMerge`, `export`, `filterDataStorage`, `filterPrecursorScan`, `peaksData<-`, `filterAcquisitionNum`, `intensity<-`, `mz<-`, `precScanNum`, `spectraData<-`, `spectraNames<-`.

Retrieving compound annotations for spectra

While compound annotations are also provided *via* the `spectraVariables` of the backend, it would also be possible to use the compounds function on a `Spectra` object (that uses a `MsBackendMassbankSql` backend) to retrieve compound annotations for the specific spectra.

Author(s)

Johannes Rainer

Examples

```
## Create a connection to a database with MassBank data - in the present
## example we connect to a tiny SQLite database bundled in this package
## as public access to the MassBank MySQL is not (yet) supported. See the
## vignette for more information on how to install MassBank locally and
## enable MySQL database connections
library(RSQLite)
con <- dbConnect(SQLite(), system.file("sql", "minimassbank.sqlite",
    package = "MsBackendMassbank"))

## Given that we have the connection to a MassBank databas we can
## initialize the backend:
be <- backendInitialize(MsBackendMassbankSql(), dbcon = con)
be
```

```
## Access MS level
msLevel(be)
be$msLevel

## Access m/z values
be$mz

## Access the full spectra data (including m/z and intensity values)
spectraData(be)

## Add a new spectra variable
be$new_variable <- "b"
be$new_variable

## Subset the backend
be_sub <- be[c(3, 1)]

spectraNames(be)
spectraNames(be_sub)
```

Index