

Package ‘MsBackendMsp’

April 16, 2024

Title Mass Spectrometry Data Backend for NIST msp Files

Version 1.6.0

Description Mass spectrometry (MS) data backend supporting import and handling of MS/MS spectra from NIST MSP Format (msp) files. Import of data from files with different MSP *flavours* is supported. Objects from this package add support for MSP files to Bioconductor's Spectra package. This package is thus not supposed to be used without the Spectra package that provides a complete infrastructure for MS data handling.

Depends R (>= 4.1.0), Spectra (>= 1.5.14)

Imports BiocParallel, S4Vectors, IRanges, MsCoreUtils, methods, stats

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

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

BugReports <https://github.com/RforMassSpectrometry/MsBackendMsp/issues>

URL <https://github.com/RforMassSpectrometry/MsBackendMsp>

biocViews Infrastructure, Proteomics, MassSpectrometry, Metabolomics, DataImport

Roxygen list(markdown=TRUE)

RoxygenNote 7.2.3

Collate 'hidden_aliases.R' 'MsBackendMsp.R' 'functions-msp.R'

git_url <https://git.bioconductor.org/packages/MsBackendMsp>

git_branch RELEASE_3_18

git_last_commit 08ab217

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-04-15

Author Neumann Steffen [aut] (<<https://orcid.org/0000-0002-7899-7192>>),
 Johannes Rainer [aut, cre] (<<https://orcid.org/0000-0002-6977-7147>>),
 Michael Witting [ctb] (<<https://orcid.org/0000-0002-1462-4426>>)

Maintainer Johannes Rainer <Johannes.Rainer@eurac.edu>

R topics documented:

hidden_aliases	2
MsBackendMsp	2
readMsp	5
Index	6

hidden_aliases	<i>Internal page for hidden aliases</i>
----------------	---

Description

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

MsBackendMsp	<i>MS data backend for msp files</i>
--------------	--------------------------------------

Description

The MsBackendMsp class supports import of MS/MS spectra data from files in NIST MSP file format. MsBackendMsp extends the `MsBackendDataFrame()` backend directly and supports thus the `applyProcessing()` function to make data manipulations persistent.

New objects are created with the MsBackendMsp function. The backendInitialize method has to be subsequently called to initialize the object and import MS/MS data from (one or more) msp files.

The MsBackendMsp backend provides an export method that allows to export the data from the Spectra object (parameter x) to a file in msp format. Parameters to this function are:

- x: the Spectra object that should be exported.
- file: character(1) with the desired file name.
- mapping: named character providing the mapping between spectra variables and MSP data fields. Defaults to mapping = spectraVariableMapping(MsBackendMsp()).
- allVariables: logical(1) whether all spectra variables in x should be exported or only those defined with mapping.
- exportName: logical(1) whether a NAME field should always be exported even if not provided in x.

See the package vignette for details and examples.

The `spectraVariableMapping` function allows to provide the mapping between spectra variable names (i.e. the names that will be used for the spectra variables in the `Spectra()` object) and the data field names of the MSP file. Parameter `format` allows to select pre-defined mappings. Currently supported mapping flavors are:

- `format = "msp"`: default MSP field names. Should work with standard NIST MSP files or MSP files exported from MS-DIAL.
- `format = "mona"`: MSP file format from MoNA including LipidBlast.

Usage

```
## S4 method for signature 'MsBackendMsp'
backendInitialize(
  object,
  file,
  mapping = spectraVariableMapping(object),
  ...,
  BPPARAM = bpparam()
)

MsBackendMsp()

## S4 method for signature 'MsBackendMsp'
spectraVariableMapping(object, format = c("msp", "mona"))

## S4 method for signature 'MsBackendMsp'
export(
  object,
  x,
  file = tempfile(),
  mapping = spectraVariableMapping(MsBackendMsp()),
  allVariables = TRUE,
  exportName = TRUE,
  ...
)
```

Arguments

<code>object</code>	Instance of <code>MsBackendMsp</code> class.
<code>file</code>	character with the (full) file name(s) of the msp file(s) from which MS/MS data should be imported or exported.
<code>mapping</code>	named character vector to rename MSP fields to spectra variables (see <code>spectraVariableMapping()</code>). This allows to correctly import also custom fields or data from files with different MSP <i>flavors</i> .
<code>...</code>	Currently ignored.
<code>BPPARAM</code>	Parameter object defining the parallel processing setup to import data in parallel. Defaults to <code>BPPARAM = bpparam()</code> . See <code>bpparam()</code> for more information.

format	For spectraVariableMapping: character(1) specifying for which MSP <i>flavour</i> the mapping should be returned. Currently supported are: format = "msp" (generic MSP format, for example for MS-DIAL MSP files) and format = "mona" (MSP files in MoNA flavour).
x	For export: a <code>Spectra()</code> object that should be exported to the specified MSP file.
allVariables	logical(1) whether all spectra variables in x should be exported or only those defined with mapping.
exportName	logical(1) whether a NAME field should always be exported even if not provided in x.

Value

MsBackendMsp and backendInitialize return an instance of a MsBackendMsp class. spectraVariableMapping a named character vector with the mapping between spectra variables and MSP data fields.

Author(s)

Steffen Neumann, Michael Witting, Laurent Gatto and Johannes Rainer

Examples

```
## Import spectra from a MSP file from LipidBlast
f <- system.file("extdata", "small-export-LipidBlast.msp",
  package = "MsBackendMsp")
be <- backendInitialize(MsBackendMsp(), f)
be

be$msLevel
be$intensity
be$mz

## precursor m/z are however all missing
be$precursorMz

## Default spectra variable mapping
spectraVariableMapping(MsBackendMsp())

## In fact, to read MSP files in "LipidBlast flavour" (same as MoNA) we
## should use a different spectra variable mapping
spectraVariableMapping(MsBackendMsp(), "mona")

## Importing the data with this will correctly retrieve data
be <- backendInitialize(MsBackendMsp(), f,
  mapping = spectraVariableMapping(MsBackendMsp(), "mona"))
be$precursorMz

## Other fields are also correctly mapped, but might need to be converted
## to e.g. numeric, such as "exactmass"
be$exactmass
```

```
be$exactmass <- as.numeric(be$exactmass)

be$adduct
be$formula

## Exporting Spectra objects in MSP format.

sps <- Spectra(be)
export(MsBackendMsp(), sps, file = stdout())
```

readMsp

Reading MSP files

Description

The `readMsp` function imports the data from a file in MGF format reading all specified fields and returning the data as a `DataFrame()`.

Usage

```
readMsp(f, msLevel = 2L, mapping = spectraVariableMapping(MsBackendMsp()), ...)
```

Arguments

<code>f</code>	character(1) with the path to an MSP file.
<code>msLevel</code>	numeric(1) with the MS level. Default is 2. This value will be reported as the spectra's MS level unless the source MSP file defines the MS level.
<code>mapping</code>	named character vector to rename MSP fields to spectra variables (see <code>spectraVariableMapping()</code>). This allows to correctly import also custom fields or data from files with different MSP <i>flavors</i> .
<code>...</code>	Additional parameters, currently ignored.

Value

A `DataFrame` with each row containing the data from one spectrum in the MSP file. `m/z` and intensity values are available in columns `"mz"` and `"intensity"` in a list representation.

Author(s)

Laurent Gatto, Steffen Neumann, Johannes Rainer

Examples

```
f1s <- dir(system.file("extdata", package = "MsBackendMsp"),
  full.names = TRUE, pattern = "msp$")[1L]

readMsp(f1s)
```

Index

* **internal**

- hidden_aliases, [2](#)
- [,MsBackendDataFrame-method
(hidden_aliases), [2](#)
- applyProcessing(), [2](#)
- backendInitialize,MsBackendMsp-method
(MsBackendMsp), [2](#)
- bpparam(), [3](#)
- DataFrame(), [5](#)
- export,MsBackendMsp-method
(MsBackendMsp), [2](#)
- hidden_aliases, [2](#)
- MsBackendDataFrame(), [2](#)
- MsBackendMsp, [2](#)
- MsBackendMsp-class (MsBackendMsp), [2](#)
- readMsp, [5](#)
- Spectra(), [3](#), [4](#)
- spectraVariableMapping(), [3](#), [5](#)
- spectraVariableMapping,MsBackendMsp-method
(MsBackendMsp), [2](#)