Package: MsBackendMgf (via r-universe)

November 1, 2026

Title Mass Spectrometry Data Backend for Mascot Generic Format (mgf) Files

Version 1.20.1

Description Mass spectrometry (MS) data backend supporting import and export of MS/MS spectra data from Mascot Generic Format (mgf) files. Objects defined in this package are supposed to be used with the Spectra Bioconductor package. This package thus adds mgf file support to the Spectra package.

Depends R (>= 4.0), Spectra (>= 1.5.14)

Imports ProtGenerics (>= 1.35.3), BiocParallel, S4Vectors, IRanges, MsCoreUtils, methods, stats, data.table

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

License Artistic-2.0

LazyData yes

Encoding UTF-8

VignetteBuilder knitr

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

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

biocViews Infrastructure, Proteomics, MassSpectrometry, Metabolomics, DataImport

Roxygen list(markdown=TRUE)

Collate 'hidden_aliases.R' 'MsBackendMgf.R' 'functions-mgf.R'

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev

Repository Bioconductor 3.23

Date/Publication 2026-07-24 08:30:21 UTC

RemoteUrl https://github.com/bioc/MsBackendMgf

RemoteRef RELEASE_3_23

RemoteSha 56c06bd2ab4e0b1fb8d99bdc450d1dd49b74d3ab

Contents

MsBackendMgf 2
readMgf 6

Index **8**

`MsBackendMgf` *MS data backend for MGF files*

Description

The `MsBackendMgf` class supports import and export of MS/MS spectra data from/to files in Mascot Generic Format (mgf) files. After initial import, the full MS data is kept in memory. `MsBackendMgf` extends the `Spectra::MsBackendDataFrame()` backend directly and supports thus the `Spectra::applyProcessing()` function to make data manipulations persistent.

The `MsBackendAnnotatedMgf` class supports import of data from MGF files that provide, in addition to the *m/z* and intensity values, also additional annotations/metadata for each mass peak. For such MGF files it is expected that each line contains information from a single mass peak, separated by a white space (blank). The first two elements are expected to be the peak's *m/z* and intensity values, while each additional element is considered an annotation for this specific peak. See examples below for the format of a supported MGF file. The `backendInitialize()` method of `MsBackendAnnotatedMgf` does not support parameter `nlines`. Also, import of data can be considerably slower compared to the standard `MsBackendMgf` backend, because of the additionally required parsing of peak annotations. Peaks information in MGF files are not named, thus, additional peaks annotations are named using the standard naming convention for column named of data frames: the first peaks annotation is called `"V1"`, the second (if available) `"V2"` and so on.

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

The `MsBackendMgf` backend provides an `export` method that allows to export the data from the `Spectra` object (parameter `x`) to a file in mgf format. See the package vignette for details and examples.

Default mappings from fields in the MGF file to spectra variable names are provided by the `spectraVariableMapping()` function. This function returns a named character vector were names are the spectra variable names and the values the respective field names in the MGF files. This named character vector is submitted to the import and export function with parameter `mapping`. It is also possible to pass own mappings (e.g. for special MGF dialects) with the `mapping` parameter.

Usage

```
## S4 method for signature 'MsBackendMgf'
backendInitialize(
  object,
  files,
  mapping = spectraVariableMapping(object),
  nlines = -1L,
```

```
  ...,
  BPPARAM = SerialParam()
)

MsBackendMgf()

## S4 method for signature 'MsBackendMgf'
spectraVariableMapping(object, format = c("mgf"))

## S4 method for signature 'MsBackendMgf'
export(
  object,
  x,
  file = tempfile(),
  mapping = spectraVariableMapping(object),
  exportTitle = TRUE,
  ...
)

## S4 method for signature 'MsBackendAnnotatedMgf'
backendInitialize(
  object,
  files,
  mapping = spectraVariableMapping(object),
  ...,
  BPPARAM = SerialParam()
)

MsBackendAnnotatedMgf()
```

Arguments

object	Instance of MsBackendMgf class.
files	character with the (full) file name(s) of the mgf file(s) from which MS/MS data should be imported.
mapping	for backendInitialize() and export: named character vector allowing to specify how fields from the MGF file should be renamed. Names are supposed to be the spectra variable name and values of the vector the field names in the MGF file. See output of spectraVariableMapping() for the expected format and examples below or description above for details.
nlines	for backendInitialize() of MsBackendMgf: integer(1) defining the number of lines that should be imported and processed from the MGF file(s). By default (nlines = -1L) the full file is imported and processed at once. If set to a positive integer, the data is imported and processed *chunk-wise* using readMgfSplit().
...	Currently ignored.
BPPARAM	Parameter object defining the parallel processing setup. If parallel processing is enabled (with BPPARAM different than SerialParam(), the default) and length

of `files` is larger than one, import is performed in parallel on a per-file basis. If data is to be imported from a single file (i.e., length of `files` is one), parsing of the imported file is performed in parallel. See also `BiocParallel::SerialParam()` for information on available parallel processing setup options.

`format` for `spectraVariableMapping()`: `character(1)` defining the format to be used. Currently only `format = "mgf"` is supported.

`x` for `export()`: an instance of `Spectra::Spectra()` class with the data that should be exported.

`file` `character(1)` with the (full) file name to which the data should be exported.

`exportTitle` `logical(1)` whether the *TITLE* field should be included in the exported MGF file. If `TRUE` (the default) a `spectraVariable` called `"TITLE"` will be used, if no such variable is present either the `spectraNames(object)` will be used or, if they are empty, a title will be generated including the MS level, retention time and acquisition number of the spectrum.

Value

See description above.

Author(s)

Laurent Gatto, Corey Broeckling and Johannes Rainer

Examples

```
library(BiocParallel)
#' Getting the file names of all example MGF files from MsBackendMgf
fls <- dir(system.file("extdata", package = "MsBackendMgf"),
    full.names = TRUE, pattern = "^spectra(.*).mgf$")

## Create an MsBackendMgf backend and import data from test mgf files.
be <- backendInitialize(MsBackendMgf(), fls)
be

be$msLevel
be$intensity
be$mz

## The spectra variables that are available; note that not all of them
## have been imported from the MGF files.
spectraVariables(be)

## The variable "TITLE" represents the title of the spectrum defined in the
## MGF file
be$TITLE

## The default mapping of MGF fields to spectra variables is provided by
## the spectraVariableMapping function
spectraVariableMapping(MsBackendMgf())
```

```
## We can provide our own mapping e.g. to map the MGF field "TITLE" to a
## variable named "spectrumName":
map <- c(spectrumName = "TITLE", spectraVariableMapping(MsBackendMgf()))
map

## We can then pass this mapping with parameter `mapping` to the
## backendInitialize method:
be <- backendInitialize(MsBackendMgf(), fls, mapping = map)

## The title is now available as variable named spectrumName
be$spectrumName

## Next we create a Spectra object with this data
sps <- Spectra(be)

## We can use the 'MsBackendMgf' also to export spectra data in mgf format.
out_file <- tempfile()
export(sps, backend = MsBackendMgf(), file = out_file, map = map)

## The first 20 lines of the generated file:
readLines(out_file, n = 20)

## Next we add a new spectra variable to each spectrum
sps$spectrum_idx <- seq_along(sps)

## This new spectra variable will also be exported to the mgf file:
export(sps, backend = MsBackendMgf(), file = out_file, map = map)
readLines(out_file, n = 20)

####
## Annotated MGF

## An example of a supported annotated MGF file
fl <- system.file("extdata", "xfiora.mgf", package = "MsBackendMgf")

## Lines with peak data start with a numeric and information is
## separated by a whitespace. The first two elements are the peak's m/z
## and intensity while any additional information is considered as
## annotation. Information for each peak is provided in one line.
readLines(fl)

## Importing the data using an `MsBackendAnnotatedMgf`
ba <- backendInitialize(MsBackendAnnotatedMgf(), fl)
ba

## An additional peaks variable is available.
peaksVariables(ba)

ba$V1

## The length of such peaks variables is the same as the length of the
## m/z or intensity values, i.e. each peak has one value (with the value
## being `NA` if missing).
```

```
length(ba$V1[[1L]])
length(ba$mz[[1L]])

## Extracting the peaks data from a `Spectra` with a `MsBackendAnnotatedMgf`
s <- Spectra(ba)
pd <- peaksData(s, peaksVariables(ba))[[1L]]
head(pd)
class(pd)
```

---

readMgf *Reading MGF files*

Description

The readMgf() function imports the data from a file in MGF format reading all specified fields and returning the data as a S4Vectors::DataFrame().

For very large MGF files the readMgfSplit() function might be used instead. In contrast to the readMgf() functions, readMgfSplit() reads only nlines lines from an MGF file at once reducing thus the memory demand (at the cost of a lower performance, compared to readMgf()).

Usage

```
readMgf(
  f,
  msLevel = 2L,
  mapping = spectraVariableMapping(MsBackendMgf()),
  annotated = FALSE,
  ...,
  BPPARAM = SerialParam()
)

readMgfSplit(
  f,
  msLevel = 2L,
  mapping = spectraVariableMapping(MsBackendMgf()),
  nlines = 1e+05,
  BPPARAM = SerialParam(),
  ...
)
```

Arguments

f	character(1) with the path to an mgf file.
msLevel	numeric(1) with the MS level. Default is 2.
mapping	named character vector to rename mgf fields to spectra variables.

annotated	For readMgf(): logical(1) whether the MGF file contains additional peak annotations. See examples below or the documentation for MsBackendAnnotatedMgf() for information on the expected format.
...	Additional parameters, currently ignored.
BPPARAM	parallel processing setup that should be used. Only the parsing of the imported MGF file is performed in parallel.
nlines	for readMgfSplit(): integer(1) with the number of lines that should be imported and parsed in each iteration.

Value

A DataFrame with each row containing the data from one spectrum in the MGF file. m/z and intensity values are available in columns "mz" and "intensity" in a list representation. For readMgf() with annotated = TRUE also all peaks annotation columns (named "V1", etc) are provided in a list representation,

Author(s)

Laurent Gatto, Johannes Rainer, Sebastian Gibb, Corey Broeckling

Examples

```
fls <- dir(system.file("extdata", package = "MsBackendMgf"),
    full.names = TRUE, pattern = "mgf$")[1L]

readMgf(fls)

## Annotated MGF
fl <- system.file("extdata", "xfiora.mgf", package = "MsBackendMgf")
res <- readMgf(fl, annotated = TRUE)
colnames(res)
res$V1
```

Index

backendInitialize,MsBackendAnnotatedMgf-method
        *(*MsBackendMgf*)*, 2
backendInitialize,MsBackendMgf-method
        *(*MsBackendMgf*)*, 2
BiocParallel::SerialParam(), *4*

export,MsBackendMgf-method
        *(*MsBackendMgf*)*, 2

MsBackendAnnotatedMgf *(*MsBackendMgf*)*, 2
MsBackendAnnotatedMgf(), *7*
MsBackendMgf, 2
MsBackendMgf-class *(*MsBackendMgf*)*, 2

readMgf, 6
readMgfSplit *(*readMgf*)*, 6
readMgfSplit(), *3*

S4Vectors::DataFrame(), *6*
Spectra::applyProcessing(), *2*
Spectra::MsBackendDataFrame(), *2*
Spectra::Spectra(), *4*
spectraVariableMapping,MsBackendMgf-method
        *(*MsBackendMgf*)*, 2