

# Package ‘MsBackendMgf’

September 14, 2021

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

**Version** 1.0.0

**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 (>= 4.1), Spectra (>= 1.0)

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

**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)

**RoxygenNote** 7.1.1

**Collate** 'hidden\_aliases.R' 'MsBackendMgf.R' 'functions-mgf.R'

**git\_url** <https://git.bioconductor.org/packages/MsBackendMgf>

**git\_branch** RELEASE\_3\_13

**git\_last\_commit** 291d8e9

**git\_last\_commit\_date** 2021-05-19

**Date/Publication** 2021-09-14

**Author** RforMassSpectrometry Package Maintainer [cre],  
 Laurent Gatto [aut] (<<https://orcid.org/0000-0002-1520-2268>>),  
 Johannes Rainer [aut] (<<https://orcid.org/0000-0002-6977-7147>>),  
 Sebastian Gibb [aut] (<<https://orcid.org/0000-0001-7406-4443>>)

**Maintainer**

RforMassSpectrometry Package Maintainer <maintainer@rformassspectrometry.org>

## R topics documented:

MsBackendMgf . . . . .	2
readMgf . . . . .	4

<b>Index</b>	<b>6</b>
--------------	----------

---

MsBackendMgf	<i>MS data backend for mgf files</i>
--------------	--------------------------------------

---

### 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 `MsBackendDataFrame()` backend directly and supports thus the `applyProcessing()` function to make data manipulations persistent.

New objects are created with the MsBackendMgf function. The `backendInitialize` method has to be subsequently called to initialize the object and import 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 where 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(),
  ...,
  BPPARAM = bpparam()
)

MsBackendMgf()
```

```
spectraVariableMapping(format = c("mgf"))

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

### 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.
...	Currently ignored.
BPPARAM	Parameter object defining the parallel processing setup to import data in parallel. Defaults to BPPARAM = bpparam(). See <a href="#">bpparam()</a> for more information.
format	for spectraVariableMapping: character(1) defining the format to be used. Currently only format = "mgf" is supported.
x	for export: an instance of <a href="#">Spectra()</a> class with the data that should be exported.
file	character(1) with the (full) file name to which the data should be exported.

### Value

See description above.

### Author(s)

Laurent Gatto and Johannes Rainer

### Examples

```
library(BiocParallel)
fls <- dir(system.file("extdata", package = "MsBackendMgf"),
           full.names = TRUE, pattern = "mgf$")

## Parallel processing setup: disabling parallel processing by registering
## serial processing. See ?bbparam for details and other options
register(SerialParam())

## 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()

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

```

---

readMgf

*Reading MGF files* The readMgf function imports the data from a file in MGF format reading all specified fields and returning the data as a [DataFrame\(\)](#).

---

**Description**

Reading MGF files

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

**Usage**

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

**Arguments**

<code>f</code>	character(1) with the path to an mgf file.
<code>msLevel</code>	numeric(1) with the MS level. Default is 2.
<code>mapping</code>	named character vector to rename mgf fields to spectra variables.
<code>...</code>	Additional parameters, currently ignored.

**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.

**Author(s)**

Laurent Gatto, Johannes Rainer

**Examples**

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

readMgf(f1s)
```

# Index

`applyProcessing()`, [2](#)  
`backendInitialize`, `MsBackendMgf`-method  
    (`MsBackendMgf`), [2](#)  
`bpparam()`, [3](#)  
`DataFrame()`, [4](#), [5](#)  
`export`, `MsBackendMgf`-method  
    (`MsBackendMgf`), [2](#)  
`MsBackendDataFrame()`, [2](#)  
`MsBackendMgf`, [2](#)  
`MsBackendMgf`-class (`MsBackendMgf`), [2](#)  
`readMgf`, [4](#)  
`Spectra()`, [3](#)  
`spectraVariableMapping` (`MsBackendMgf`), [2](#)