

Package: MsBackendTimsTof (via r-universe)

October 26, 2026

Title Mass Spectrometry Data Backend for Bruker TimsTOF Files

Version 0.99.0

Description Mass spectrometry (MS) data backend supporting import and export of (ion mobility) MS data from Bruker TimsTOF files. The backend uses the opentimsr R package for raw data access, using either the proprietary C++ library from Bruker or an open reimplementation. The backend thus supports import of MS data from the original raw data files without needing to convert them to an open format.

Depends R (>= 4.5), Spectra (>= 1.5.17)

Imports IRanges, BiocParallel, methods, opentimsr (>= 1.0.11), MsCoreUtils, S4Vectors, ProtGenerics, RSQLite, DBI, data.table, dplyr

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

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

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

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

biocViews Infrastructure, MassSpectrometry, Metabolomics, DataImport

Roxygen list(markdown=TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs cmake make libuv1-dev libzstd-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-10-26 14:21:47 UTC

RemoteUrl <https://github.com/BiocStaging/MsBackendTimsTof>

RemoteRef HEAD

RemoteSha a3df811f5f9852eaab6f2f765f3ce7cd05446bd7

Contents

MsBackendTimsTof	2
setup_converter_library	6
Index	8

MsBackendTimsTof	<i>TimsTOF MS data backend</i>
------------------	--------------------------------

Description

The MsBackendTimsTof class adds support for Bruker TimsTOF data files (.d files) to [Spectra\(\)](#) objects. New objects are created with the MsBackendTimsTof() function and initialized using the backendInitialize() function. To ensure a small memory footprint, only general information is kept in memory (such as number of frames and scans) and all data (specifically the peaks data) are retrieved from the original data files on-the-fly. MsBackendTimsTof directly extends the [MsBackendCached\(\)](#) backend from the *Spectra* package and supports therefore adding or locally caching spectra variables.

Usage

```
MsBackendTimsTof()

## S4 method for signature 'MsBackendTimsTof'
backendInitialize(object, files, ..., BPPARAM = bpparam())

## S4 method for signature 'MsBackendTimsTof'
length(x)

## S4 method for signature 'MsBackendTimsTof'
peaksData(object, columns = c("mz", "intensity"))

## S4 method for signature 'MsBackendTimsTof'
peaksVariables(object)

## S4 method for signature 'MsBackendTimsTof'
mz(object)

## S4 method for signature 'MsBackendTimsTof'
intensity(object)

## S4 method for signature 'MsBackendTimsTof'
runtime(object)

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

```
## S4 method for signature 'MsBackendTimsTof,ANY'
extractByIndex(object, i)

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

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

## S4 method for signature 'MsBackendTimsTof'
show(object)

## S4 method for signature 'MsBackendTimsTof'
x$name

## S4 method for signature 'MsBackendTimsTof'
selectSpectraVariables(object, spectraVariables = spectraVariables(object))

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

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

## S4 method for signature 'MsBackendTimsTof'
spectraNames(object)

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

Arguments

object	MsBackendTimsTof object.
files	character specifying TimsTOF '*.d' folders names.
...	Additional arguments.
BPPARAM	Parameter object defining the parallel processing setup to import data in parallel. Defaults to BPPARAM = <code>bpparam()</code> . See bpparam() for more information.
x	MsBackendTimsTof object.
columns	For <code>spectraData()</code> : names of the spectra variables to extract from object. For <code>peaksData()</code> : names of the peak variables to extract. Defaults to <code>columns = c("mz", "intensity")</code> .
i	For <code>[]</code> : integer, logical to subset the object.
j	For <code>[]</code> : not supported.
drop	For <code>[]</code> : not considered.
name	For <code>\$</code> : the name of the variable to access.

spectraVariables	character with the names of the spectra variables that should be retained in the returned object.
value	For <code>\$<-</code> : the value to assign to the spectra variable.
initial	For <code>tic()</code> : <code>logical(1)</code> whether the original total ion count should be returned (<code>initial = TRUE</code> , the default) or whether it should be calculated on the spectra's intensities (<code>initial = FALSE</code>).

Details

The (DDA) MS2 data for MS2 spectra is retrieved from the *analytis.tdf* SQLite databases within the *.d* folder. Per default, the *m/z* of the largest peak in the isolation window is reported as *precursorMz* (SQLite database column "LargestPeakMz"). Alternatively, the content of database columns "AverageMz" or "IsolationMz" could be reported instead. To use e.g. the "AverageMz" as *precursorMz* set options(`TIMSTOF_PRECURSOR_MZ = "AverageMz"`).

The reported *isolationWindowLowerMz* and *isolationWindowUpperMz* are calculated based on the "IsolationMz" and "IsolationWidth" database columns, i.e., `isolationWindowLowerMz <- IsolationMz - IsolationWidth / 2` and `isolationWindowUpperMz <- IsolationMz + IsolationWidth / 2`.

Value

See information for the respective function.

Available methods

The methods listed below are specifically implemented for *MsBackendTimsTof*. The backend inherits and thus supports all other methods from the [Spectra::MsBackend](#) class.

- `$`: access any of the *spectraVariables* of the backend.
- `$<-`: add a new spectra variable or change values for an existing spectra variables. Values can be changed for any spectra variable except *peaks variables* ([Spectra::peaksVariables\(\)](#)) or special internal variables "file" and "frameId". Note that changes to spectra variables are only cached within the object but not propagated to the original data files.
- `[`: subset the backend. Only subsetting by element (*row/i*) is allowed. First the `@indices` slot of object is subsetted and then the `frames` and `fileNames` slots are subsetted accordingly. Note that `[` does not update the values of frames variables (such as "MaxIntensity", "SummedIntensities", "NumScans" and "NumPeaks").
- `backendInitialize()`: initializes object (the *MsBackendTimsTof* object) using TimsTOF data files whose path is specified by `files`. This method is supposed to be called right after creating a *MsBackendTimsTof* object with `MsBackendTimsTof()` function.
- `dataStorage()`: gets a character of length equal to the number of spectra in object with the names of the `'*.d'` folders where each spectrum is stored.
- `intensity()`: gets the intensity values from the spectra in the backend. 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.

- `mz()`: gets the mass-to-charge ratios (m/z) from the spectra in the backend. Returns a `NumericList()` of numeric vectors (m/z values for each spectrum). The length of the list is equal to the number of spectra in object.
- `peaksData()`: gets the peak matrices of the spectra in the backend. Returns a list of matrix with columns defined by parameter `columns` (which defaults to `columns = c("mz", "intensity")`). Use `peaksVariables()` to list all supported and available columns for a backend. The length of the list is equal to the number of spectra in object.
- `peaksVariables()`: gets the supported peak variables (columns) for the backend.
- `rtime()`: gets the retention times for each spectrum. Returns a numeric vector (length equal to the number of spectra) with the retention time for each spectrum.
- `selectSpectraVariables()`: reduces the available spectra variables to the ones specified with parameter `spectraVariables`. For *core spectra variables* (`coreSpectraVariables()`) only their values will be removed, but not the variable itself.
- `spectraData()`: gets spectra variables (specified by `columns`) from object.
- `spectraNames()`: returns an *ID*/name for each spectrum. As IDs the index of the spectrum within the object after the initialization is used. This index/spectra name is unique and stable for each spectrum within the same object.
- `spectraVariables()`: returns a character vector with the spectra variables names of core spectra variables defined in the Spectra package and other additional variables contained in object. Note that also "mz" and "intensity" (which are by default not returned by the `spectraVariables`, `Spectra` method) are returned.
- `tic()`: calculates the total ion count from the intensities of each spectrum (for `initial = FALSE`). For `initial = TRUE` NA is returned for all spectra.

Note

For some of the information a dedicated *converter library* is required. By default an open source library from the *opentimsr* package is used, but alternatively also the shared C++ library from Bruker can be selected. See [setup_converter_library\(\)](#) for more information.

Author(s)

Andrea Vicini, Johannes Rainer

Examples

```
## Define the test file
f1 <- system.file("ddaPASEF.d", package = "MsBackendTimsTof")

## Create a MsBackend instance for that file
be <- backendInitialize(MsBackendTimsTof(), f1)
be

## Available spectra variables
spectraVariables(be)

## Available peaks variables
peaksVariables(be)
```

```
## Subset to 10 randomly selected spectra.
be_sub <- be[sort(sample(seq_along(be), 10))]]
rtime(be_sub)

## Get the peaks data, i.e. m/z, intensity as well as tof and ion mobility
pd <- peaksData(be_sub,
  columns = c("mz", "intensity", "tof", "inv_ion_mobility"))
pd

## Filter to data set to MS2 spectra only
be_ms2 <- filterMsLevel(be, 2L)
be_ms2

## Get the full spectra data (including peaks variables)
spectraData(be_ms2)

## Add a new spectra variable
be$new_var <- seq_along(be)

head(be$new_var)

## Changing values of a spectra variable. Note that these are only changed
## *locally* within the object, but not in the original data files.
head(rtime(be))
be$rtime <- rtime(be) + 10
head(rtime(be))
```

setup_converter_library

Setup converter library for Bruker files

Description

Function to setup the required built-in open-source converters or the Bruker library to convert tof-to-mz and scan-to-inv_ion_mobility. When the Bruker option is selected the library is downloaded automatically using `opentimsr::download_bruker_proprietary_code()`.

Usage

```
setup_converter_library(opensource = TRUE, path = NULL, force = FALSE)
```

Arguments

opensource	logical(1) determine if use the built-in open-source converters of <i>opentimsr</i> (defaults) or the proprietary library from Bruker. Defaults to opensource = TRUE.
path	character(1) where save the Bruker library. If NULL the library is cached using <i>BiocFileCache</i> . Used only with the Bruker library. Defaults to path = NULL.

force logical(1) force to redownload the Bruker library (default: FALSE). See the notes section for the use of force = TRUE on Windows.

Value

The function is called to register and load a converter library and does not return a value.

Note

This function is called during package startup, thus in most cases it is not required to be used. This will use the `opentimsr::setup_opensource()` function. To use Bruker's proprietary library, call `setup_converter_library()` with parameter `opensource = FALSE`. This will download the library with `opentimsr::download_bruker_proprietary_code()` and cache the file in the local *BiocFileCache*.

On Windows, the Bruker DLL is bound to the `opentimsr` package. To allow the Bruker library to be re-downloaded, the package must be temporarily detached so the old file can be deleted, then reattached afterwards. This is done automatically.

Author(s)

Gabriele Tomè

Examples

```
## To setup the open-source built-in library
setup_converter_library()

## Use `setup_converter_library(opensource = FALSE)` to download and use the
## proprietary library from Bruker. The library file is downloaded and
## cached with *BiocFileCache* package. Alternatively, to download the
## library to a specific local path use the `path` parameter.
```

Index

[,MsBackendTimsTof,ANY,ANY-method
(MsBackendTimsTof), [2](#)

[,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

\$,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

\$<-,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

backendInitialize,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

bpparam(), [3](#)

coreSpectraVariables(), [5](#)

dataStorage,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

extractByIndex,MsBackendTimsTof,ANY-method
(MsBackendTimsTof), [2](#)

intensity,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

length,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

MsBackendCached(), [2](#)

MsBackendTimsTof, [2](#)

MsBackendTimsTof-class
(MsBackendTimsTof), [2](#)

mz,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

NumericList(), [4](#), [5](#)

opentimsr::download_bruker_proprietary_code(),
[7](#)

opentimsr::setup_opensource(), [7](#)

peaksData,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

peaksVariables,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

precScanNum,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

runtime,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

selectSpectraVariables,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

setup_converter_library, [6](#)

setup_converter_library(), [5](#)

show,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

Spectra(), [2](#)

Spectra::MsBackend, [4](#)

Spectra::peaksVariables(), [4](#)

spectraData,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

spectraNames,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)

tic,MsBackendTimsTof-method
(MsBackendTimsTof), [2](#)