Package: MsBackendTimsTof (via r-universe)

November 3, 2024

Title Mass spectrometry Data Backend for Bruker TimsTOF Files

Version 0.1.5

Description Mass spectrometry (MS) data backend supporting import and export of (ion mobility) MS data from Bruker TimsTOF files. The backend uses the opentimsr package which relies on the proprietory vendor C++ library for raw data access. The backend thus supports import of MS data from the original raw data files.

Depends R (>= 4.1), Spectra (>= 1.5.17)

Imports IRanges, BiocParallel, methods, opentimsr (>= 1.0.11), MsCoreUtils, S4Vectors, ProtGenerics

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

Remotes michalsta/opentims/opentimsr

License Artistic-2.0

LazyData yes

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)

RoxygenNote 7.2.3

Repository https://rformassspectrometry.r-universe.dev

RemoteUrl https://github.com/rformassspectrometry/MsBackendTimsTof

RemoteRef HEAD

RemoteSha 2e328e104424d2822b1391628a76a524369e5c94

Contents

`MsBackendTimsTof` *TimsTOF data backend*

Description

The `MsBackendTimsTof` class supports Bruker TimsTOF data files. New objects are created with the `MsBackendTimsTof` 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) is retrieved from the original file on-the-fly. By extending the `MsBackendCached()` backend from the `Spectra` package, adding or (locally) changing spectra values is also supported.

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'
rtime(object)

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

## 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'
msLevel(object, ...)

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

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

## 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 = bpparam(). See bpparam() for more information.
x	MsBackendTimsTof object.
columns	For spectraData: names of the spectra variables to extract from object. For peaksData: names of the peak variables to extract. Defaults to columns = c("mz", "intensity").
i	For [: integer, logical to subset the object.
j	For [: not supported.
drop	For [: not considered.
name	For $: 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 $<-: the value to assign to the spectra variable.

initial
: For tic: logical(1) whether the original total ion count should be returned (initial = TRUE, the default) or whether it should be calculated on the spectras' intensities (initial = FALSE).

Available methods

The following methods are implemented:

- $: 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* (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.
- msLevel: gets the spectra MS level. Returns an integer vector (of length equal to the number of spectra) with the MS level for each spectrum.
- 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.

Author(s)

Andrea Vicini, Johannes Rainer

Examples

```
## Load the opentimsr package to retrieve the required shared library
## from Bruker.
so_folder <- tempdir()
library(opentimsr)
so_file <- download_bruker_proprietary_code(so_folder, method = "wget")
setup_bruker_so(so_file)
path_d_folder <- system.file("ddaPASEF.d",
                             package = "MsBackendTimsTof")

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

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

## Available spectra variables
spectraVariables(be)

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

pd <- peaksData(be_sub, columns = c("mz", "intensity", "tof", "inv_ion_mobility"))

## 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))
```

Index