Package ‘opentimsr’

March 30, 2022

Encoding UTF-8

Type Package

Title An Open-Source Loader for Bruker's timsTOF Data Files

Version 1.0.13

Date 2022-03-29

Maintainer Michał Piotr Startek <michal.startek@mimuw.edu.pl>

Description A free, open-source package designed for
handling .tdf data files produced by Bruker's 'timsTOF' mass
spectrometers.
Fast, free, crossplatform, with no reading through
EULAs or messing with binary .dll files involved.

License GPL-3

URL https://github.com/michalsta/opentims

Depends R (>= 3.0.0)

Imports Rcpp (>= 0.12.0), methods, DBI, RSQLite

LazyData no

LinkingTo Rcpp

NeedsCompilation yes

SystemRequirements C++17

RoxygenNote 7.1.2

Author Michał Piotr Startek [aut, cre, cph]
(<https://orcid.org/0000-0001-5227-3447>),
Mateusz Krzysztof Łącki [aut, cph]
(<https://orcid.org/0000-0001-7415-4748>)

Repository CRAN

Date/Publication 2022-03-29 22:40:06 UTC

R topics documented:

`CloseTIMS` *Close the TIMS data handle and release all resources.*

Description

Calling this method is not mandatory - the resources will anyway be cleanly released when the garbage collector decides to destroy the OpenTIMS data handle. The only purpose of this method is to enable them to be explicitly released earlier than that.

Usage

```
CloseTIMS(opentims)
```

Arguments

`opentims`	Instance of OpenTIMS.

Value

void

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
CloseTIMS(D)

## End(Not run)
```

`download_bruker_proprietary_code`
Get Bruker's code needed for running proprietary time of flight to mass over charge and scan to drift time conversion.

Description

By using this function you aggree to terms of license precised in "https://github.com/MatteoLacki/opentims_bruker_bridge". The conversion, due to independent code-base restrictions, are possible only on Linux and Windows operating systems. Works on full open-source solution are on the way.

Usage

```
download_bruker_proprietary_code(
  target.folder,
  net_url = paste0("https://raw.githubusercontent.com/MatteoLacki/",
    "opentims_bruker_bridge/main/opentims_bruker_bridge/"),
  mode = "wb",
  ...
)
```

Arguments

`target.folder`	Folder where to store the 'dll' or 'so' file.
`net_url`	The url with location of all files.
`mode`	Which mode to use when downloading a file?
`...`	Other parameters to 'download.file'.

Value

Path to the output 'timsdata.dll' on Windows and 'libtimsdata.so' on Linux.

Examples

```
## Not run:
download_bruker_proprietary_code("your/prefered/destination/folder")

## End(Not run)
```

explore.tdf.tables *Explore the contentents of the sqlite .tdf database.*

Description

Explore the contentents of the sqlite .tdf database.

Usage

```
explore.tdf.tables(opentims, ...)
```

Arguments

`opentims`	Instance of OpenTIMS
`...`	Parameters passed to head and tail functions.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
explore.tdf.tables(D)

## End(Not run)
```

length,OpenTIMS-method *Get the overall number of peaks.*

Description

Get the overall number of peaks.

Usage

```
## S4 method for signature 'OpenTIMS'
length(x)
```

Arguments

`x`	OpenTIMS data instance.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(length(D))

## End(Not run)
```

`min_max_measurements` *Get border values for measurements.*

Description

Get the min-max values of the measured variables (except for TOFs, that would require iteration through data rather than parsing metadata).

Usage

```
min_max_measurements(opentims)
```

Arguments

`opentims`	Instance of OpenTIMS.

Value

data.frame Limits of individual extracted quantities.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
min_max_measurements(D) # this gives a small data-frame with min and max values.

## End(Not run)
```

`MS1` *Get MS1 frame numbers.*

Description

Get MS1 frame numbers.

Usage

```
MS1(opentims)
```

Arguments

`opentims`	Instance of OpenTIMS

Value

Numbers of frames corresponding to MS1, i.e. precursor ions.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(MS1(D))

## End(Not run)
```

`OpenTIMS` *Get OpenTIMS data representation.*

Description

Get OpenTIMS data representation.

Usage

```
OpenTIMS(path.d)
```

Arguments

`path.d`	Path to the TimsTOF '*.d' folder containing the data (requires the folder to contain only 'analysis.tdf' and 'analysis.tdf_bin').

Examples

```
## Not run:
D = OpenTIMS(path_to_.d_folder)
D[1] # First frame.

## End(Not run)
```

`OpenTIMS-class` *TimsTOF data accessor.*

Description

S4 class that facilitates data queries for TimsTOF data.

Slots

`path.d`	Path to raw data folder (typically *.d).
`handle`	Pointer to raw data.
`min_frame`	The index of the minimal frame.
`max_frame`	The index of the miximal frame.
`min_scan`	The minimal scan number. It is assumed to be equal to 1.
`max_scan`	The maximal scan number.
`min_intensity`	The minimal value of intensity. Set to 0, but actually 9 is more sensible.
`max_intensity`	The maximal intensity: the max over values reported in the frames.
`min_retention_time`	The lowest recorded retention time.
`max_retention_time`	The highest recorded retention time.
`min_inv_ion_mobility`	The minimal recorded inverse ion mobility.
`max_inv_ion_mobility`	The maximal recorded inverse ion mobility.
`min_mz`	The minimal recorded mass to charge ratio.
`max_mz`	The maximal recorded mass to charge ratio.
`frames`	A data.frame with information on the frames (contents of the Frames table in the sqlite db).
`all_columns`	Names of available columns.

`opentims_set_threads` *Set the number of threads to be used for data processing by OpenTIMS*

Description

A value of 0 is acceptable: it will cause OpenTIMS to use all detected cores.

Usage

```
opentims_set_threads(n)
```

Arguments

`n`	The number of worker threads to be used.

Value

void

Examples

```
## Not run:
opentims_set_threads(1)

## End(Not run)
```

`peaks_per_frame_cnts` *Get the number of peaks per frame.*

Description

Get the number of peaks per frame.

Usage

```
peaks_per_frame_cnts(opentims)
```

Arguments

`opentims`	Instance of OpenTIMS.

Value

Number of peaks in each frame.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(peaks_per_frame_cnts(D))

## End(Not run)
```

`query` *Query for raw data.*

Description

Get the raw data from Bruker's 'tdf_bin' format. Defaults to both raw data ('frame','scan','tof','intensity') and its tranformations into physical units ('mz','inv_ion_mobility','retention_time').

Usage

```
query(opentims, frames, columns = all_columns)
```

Arguments

`opentims`	Instance of OpenTIMS.
`frames`	Vector of frame numbers to extract.
`columns`	Vector of columns to extract. Defaults to all columns.

Value

data.frame with selected columns.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(query(D, c(1,20, 53)) # extract all columns
print(query(D, c(1,20, 53), columns=c('scan','intensity')) # only 'scan' and 'intensity'

## End(Not run)
```

`query_slice` *Query for raw data.*

Description

Get the raw data from Bruker's 'tdf_bin' format. Defaults to both raw data ('frame','scan','tof','intensity') and its tranformations into physical units ('mz','inv_ion_mobility','retention_time').

Usage

```
query_slice(opentims, from = NULL, to = NULL, by = 1, columns = all_columns)
```

Arguments

`opentims`	Instance of OpenTIMS.
`from`	First frame to extract.
`to`	Last frame to extract.
`by`	Every by-th frame gets extracted (starting from the first one).
`columns`	Vector of columns to extract. Defaults to all columns.

Details

We assume 'from' <= 'to'.

Value

data.frame with selected columns.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(query_slice(D, 10, 200, 4)) # extract every fourth frame between 10 and 200.
print(query_slice(D, 10, 200, 4, columns=c('scan','intensity')) # only 'scan' and 'intensity'

## End(Not run)
```

range,OpenTIMS-method *Select a range of frames to extract.*

Description

This is similar to using the from:to:by operator in Python.

Usage

```
## S4 method for signature 'OpenTIMS'
range(x, from, to, by = 1L)
```

Arguments

x	OpenTIMS data instance.
from	The first frame to extract.
to	The last+1 frame to extract. Frame with that number will not get extracted, but some below that number might.
by	Extract each by-th frame

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(head(range(D, 10,100,3))) # each third frame from 10 to 100.

## End(Not run)
```

retention_times *Get the retention time for each frame.*

Description

Get the retention time for each frame.

Usage

```
retention_times(opentims)
```

Arguments

opentims	Instance of OpenTIMS.

Value

Retention times corresponding to each frame.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(retention_times(D))

## End(Not run)
```

`rt_query` *Get the retention time for each frame.*

Description

Extract all frames corresponding to retention times inside [min_retention_time, max_retention_time] closed borders interval.

Usage

```
rt_query(
  opentims,
  min_retention_time,
  max_retention_time,
  columns = all_columns
)
```

Arguments

`opentims`	Instance of OpenTIMS.
`min_retention_time`	Lower boundry on retention time.
`max_retention_time`	Upper boundry on retention time.
`columns`	Vector of columns to extract. Defaults to all columns.

Value

data.frame with selected columns.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(rt_query(D, 10, 100)) # frames between tenth and a hundreth second of the experiment

## End(Not run)
```

setup_bruker_so — *Dynamically link Bruker's DLL to enable tof-mz and scan-inv_ion_mobility conversion.*

Description

By using this function you aggree to terms of license precised in "https://github.com/MatteoLacki/opentims_bruker_bridge". The conversion, due to independent code-base restrictions, are possible only on Linux and Windows operating systems. Works on full open-source solution are on the way.

Usage

```
setup_bruker_so(path)
```

Arguments

`path`	Path to the 'libtimsdata.so' on Linux or 'timsdata.dll' on Windows, as produced by 'download_bruker_proprietary_code'.

Examples

```
## Not run:
so_path = download_bruker_proprietary_code("your/prefered/destination/folder")
setup_bruker_so(so_path)

## End(Not run)
```

table2df — *Extract tables from sqlite database analysis.tdf.*

Description

Export a table from sqlite.

Usage

```
table2df(opentims, names)
```

Arguments

`opentims`	Instance of OpenTIMS
`names`	Names to extract from the sqlite database.

Value

A list of tables.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(head(table2df(D, "Frames"))) # Extract table "Frames".

## End(Not run)
```

`tables_names` *Extract tables from sqlite database analysis.tdf.*

Description

Extract tables from sqlite database analysis.tdf.

Usage

```
tables_names(opentims)
```

Arguments

`opentims`	Instance of OpenTIMS

Value

Names of tables.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(tables_names(D))

## End(Not run)
```

`[,OpenTIMS-method` *Get some frames of data.*

Description

Get some frames of data.

Usage

```
## S4 method for signature 'OpenTIMS'
x[i]
```

Arguments

`x`	OpenTIMS data instance.
`i`	An array of nonzero indices to extract.

Examples

```
## Not run:
D = OpenTIMS('path/to/your/folder.d')
print(head(D[10]))
print(head(D[10:100]))

## End(Not run)
```

Index