Package 'opentimsr'

November 5, 2026

Encoding UTF-8

Type Package

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

Version 1.2.3

Date 2026-11-03

Maintainer Mateusz Krzysztof Łącki <matteo.lacki@gmail.com>

Description Reads raw data from Bruker's timsTOF mass spectrometers, stored in analysis.tdf and analysis.tdf_bin files inside .d folders. Frames can be selected by number or by retention time, and each peak is reported with its scan, time-of-flight index, intensity, mass-to-charge ratio and inverse ion mobility. Mass-to-charge ratios and ion mobilities are computed with built-in open-source calibration, or optionally with Bruker's proprietary library.
See Łącki et al. (2021) <doi:10.1021/acs.jproteome.0c00962>.

License MIT + file LICENSE

Copyright See file inst/COPYRIGHTS

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

BugReports https://github.com/michalsta/opentims/issues

Depends R (>= 4.3.0)

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

LinkingTo Rcpp

NeedsCompilation yes

SystemRequirements C++20, libzstd

Suggests testthat (>= 3.0.0)

Config/roxygen2/version 8.1.1

Author Michał Piotr Startek [aut, cph] (ORCID:
<https://orcid.org/0000-0001-5227-3447>),
Mateusz Krzysztof Łącki [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7415-4748>),
mandreyel [ctb, cph] (Author and copyright holder of the bundled 'mio' library (https://github.com/mandreyel/mio))

Repository CRAN

Date/Publication 2026-11-04 23:20:02 UTC

Contents

CloseTIMS . . . 2
download_bruker_proprietary_code . . . 3
explore.tdf.tables . . . 4
get_separate_frames . . . 4
length,OpenTIMS-method . . . 5
min_max_measurements . . . 6
MS1 . . . 6
OpenTIMS . . . 7
OpenTIMS-class . . . 7
opentims_set_threads . . . 8
peaks_per_frame_cnts . . . 9
query . . . 9
query_slice . . . 10
range,OpenTIMS-method . . . 11
retention_times . . . 11
rt_query . . . 12
setup_bruker_so . . . 13
setup_opensource . . . 14
table2df . . . 14
tables_names . . . 15
use_inv_ion_mobility_lookup . . . 15
use_mz_lookup . . . 16
[,OpenTIMS-method . . . 17

Index **18**

`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

No return value, called for its side effect of closing the data handle.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
CloseTIMS(D)
```

`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

Bruker's 'timsdata' library is proprietary software owned by Bruker. It is not part of opentimsr and is not distributed with it: this function downloads it only when called, from "https://github.com/MatteoLacki/opentims_bruker_b and by using it you agree to the license terms stated there. The library is available only for 64-bit x86 Linux and Windows; elsewhere this function stops with an error. opentimsr works without it, using its built-in open-source converters (see `setup_opensource`).

Usage

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

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?
`overwrite`	Download again even if the file is already in `target.folder`? By default an existing file is reused. A library already loaded with `setup_bruker_so` is never modified in place; on Windows it cannot be replaced at all until R is restarted.
`...`	Other parameters to 'download.file'.

Value

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

Examples

```
if (interactive()) {
  download_bruker_proprietary_code(tempdir())
}
```

`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.

Value

No return value, called for its side effect of printing the tables.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
if (interactive()) explore.tdf.tables(D)
CloseTIMS(D)
```

`get_separate_frames` *Get frames as separate data frames.*

Description

Like `query`, but returns one data frame per frame instead of one table with the peaks of all frames. Frames are decoded in parallel (see `opentims_set_threads`). A frame requested more than once is returned once.

Usage

```
get_separate_frames(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

A named list of data.frames, one per distinct frame, in order of first request and named by frame number. Each data.frame has the requested columns, with one row per peak, as returned by query.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
frames = get_separate_frames(D, c(1, 2))
names(frames)
head(frames[["2"]])
get_separate_frames(D, 1, columns=c('scan','intensity')) # only 'scan' and 'intensity'
CloseTIMS(D)
```

---

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.

Value

The total number of peaks in the dataset.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
length(D) # number of peaks in the dataset
CloseTIMS(D)
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
min_max_measurements(D) # small data frame with min and max values
CloseTIMS(D)
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
MS1(D)
CloseTIMS(D)
```

`OpenTIMS` *Get OpenTIMS data representation.*

Description

Select the converters for 'mz' and 'inv_ion_mobility' first, with `setup_opensource` or `setup_bruker_so`; without them, only the raw columns can be queried from the returned object.

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').

Value

An object of class `OpenTIMS`, used as the data handle by the other functions.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
D[1] # first frame
CloseTIMS(D)
```

`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: 0, as scans are numbered from 0, like in Bruker's own tables and libraries.

`max_scan` The maximal scan number: the largest NumScans minus 1.

`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

Frames are decoded in parallel by `query`, `query_slice`, `rt_query`, `[` and `range`. By default all detected cores are used; a value of 0 restores that default. When the environment variable `_R_CHECK_LIMIT_CORES_` is set (as during `R CMD check --as-cran`), the default is two threads.

Usage

```
opentims_set_threads(n)
```

Arguments

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

Value

No return value, called for its side effect of setting the number of threads.

Examples

```
opentims_set_threads(1)
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
peaks_per_frame_cnts(D)
CloseTIMS(D)
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
query(D, c(1, 2)) # extract all columns
query(D, c(1, 2), columns=c('scan','intensity')) # only 'scan' and 'intensity'
CloseTIMS(D)
```

---

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
query_slice(D, 1, 2)    # frames 1 to 2
query_slice(D, 1, 2, 2) # every second frame between 1 and 2
query_slice(D, 1, 2, columns=c('scan','intensity')) # only 'scan' and 'intensity'
CloseTIMS(D)
```

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, na.rm = FALSE)
```

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.
na.rm	Needed to comply with the generic. There should not be any NA frames, since the from and to are checked against the limits of the data.

Value

For the selected frames, a data.frame with columns 'frame', 'scan', 'tof' and 'intensity', one row per peak.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
head(range(D, 1, 3))    # frames 1 and 2
head(range(D, 1, 3, 2)) # every second frame, i.e. only frame 1
CloseTIMS(D)
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
retention_times(D)
CloseTIMS(D)
```

---

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
rt_query(D, 0.6, 0.7) # frames recorded between 0.6 and 0.7 seconds
CloseTIMS(D)
```

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

Description

Bruker's 'timsdata' library is proprietary software owned by Bruker. It is not part of opentimsr and is not distributed with it; obtain it with `download_bruker_proprietary_code`, whose license terms apply. The library is available only for 64-bit x86 Linux and Windows. opentimsr works without it, using its built-in open-source converters (see `setup_opensource`).

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'.

Value

No return value, called for its side effect of switching the conversion backend.

Examples

```
if (interactive()) {
  so_path = download_bruker_proprietary_code(tempdir())
  setup_bruker_so(so_path)
}
```

setup_opensource *Use the built-in open-source tof-to-mz and scan-to-inv_ion_mobility converters.*

Description

Activates the built-in open-source converters for tof-to-mz and scan-to-inv_ion_mobility transformations. No proprietary Bruker libraries are required.

Usage

```
setup_opensource()
```

Details

No converters are selected when the package is loaded: call either this function or `setup_bruker_so` before `OpenTIMS`. The choice is fixed when a dataset is opened. A dataset opened before either is called still returns the raw columns ('frame', 'scan', 'tof', 'intensity'), but requesting 'mz' or 'inv_ion_mobility' raises an error.

Value

No return value, called for its side effect of switching the conversion backend.

Examples

```
setup_opensource()
```

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
head(table2df(D, "Frames")$Frames) # extract table "Frames"
CloseTIMS(D)
```

---

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

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
tables_names(D)
CloseTIMS(D)
```

---

use_inv_ion_mobility_lookup *Use a lookup table for inverse ion mobility conversion with Bruker's library.*

Description

Like use_mz_lookup, for inverse ion mobility. opentimsr uses no pressure compensation, so this conversion is the same in every frame and the table is exact for all frames.

Usage

```
use_inv_ion_mobility_lookup(opentims, frame = 1L)
```

Arguments

opentims	Instance of OpenTIMS.
frame	Frame whose calibration to use, or NULL for exact conversion.

Value

No return value, called for its side effect of changing the conversion.

Examples

```
if (interactive()) {
  setup_bruker_so(download_bruker_proprietary_code(tempdir()))
  D = OpenTIMS(system.file("extdata", "test.d", package = "opentimsr"))
  use_inv_ion_mobility_lookup(D)
  CloseTIMS(D)
}
```

---

use_mz_lookup *Use a lookup table for m/z conversion with Bruker's library.*

Description

Bruker's library converts one frame at a time and can only be called by one thread at a time, which limits parallel extraction. After this call, m/z values come from a table holding the m/z of every tof index for frame, computed once by Bruker's library and then used for all frames.

Usage

```
use_mz_lookup(opentims, frame = 1L)
```

Arguments

opentims	Instance of OpenTIMS.
frame	Frame whose calibration to use, or NULL for exact conversion.

Details

Values are exact for frame and approximate for other frames: Bruker's m/z calibration changes slightly from frame to frame (by at most 0.16 ppm over four test datasets). The function can be called again with another frame, which refills the same table, or with NULL to return to exact conversion. Only available with Bruker's conversion (see setup_bruker_so).

Value

No return value, called for its side effect of changing the conversion.

Examples

```
if (interactive()) {
  setup_bruker_so(download_bruker_proprietary_code(tempdir()))
  D = OpenTIMS(system.file("extdata", "test.d", package = "opentimsr"))
  use_mz_lookup(D)        # calibration of frame 1 for all frames
  use_mz_lookup(D, 2)     # calibration of frame 2 instead
  use_mz_lookup(D, NULL)  # back to exact conversion
  CloseTIMS(D)
}
```

`[,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`	Frame numbers to extract.

Value

For the requested frames, a data.frame with columns 'frame', 'scan', 'tof' and 'intensity', one row per peak.

Examples

```
path.d = system.file("extdata", "test.d", package = "opentimsr")
setup_opensource()
D = OpenTIMS(path.d)
head(D[1])   # first frame
head(D[1:2]) # frames 1 and 2
CloseTIMS(D)
```

Index

[,OpenTIMS-method, 17

CloseTIMS, 2

download_bruker_proprietary_code, 3, *13*

explore.tdf.tables, 4

get_separate_frames, 4

length,OpenTIMS-method, 5

min_max_measurements, 6
MS1, 6

OpenTIMS, 7, *14*
OpenTIMS-class, 7
opentims_set_threads, *4*, 8

peaks_per_frame_cnts, 9

query, *4, 5*, *8*, 9
query_slice, *8*, 10

range,OpenTIMS-method, 11
retention_times, 11
rt_query, *8*, 12

setup_bruker_so, *3*, *7*, 13, *14*, *16*
setup_opensource, *3*, *7*, *13*, 14

table2df, 14
tables_names, 15

use_inv_ion_mobility_lookup, 15
use_mz_lookup, *15*, 16