Package: tdtr (via r-universe)

May 19, 2026

Title Read and Export Tucker-Davis Technologies Data from R

Version 0.0.5

Description Provides an R interface for reading Tucker-Davis Technologies (TDT) tank/block data through the Python 'tdt' package with reticulate, including Python-backed wrappers and explicit helpers for collecting data into ordinary R objects for downstream analysis.

License MIT + file LICENSE

URL https://github.com/nimh-dsst/tdtr, https://nimh-dsst.r-universe.dev/tdtr

BugReports https://github.com/nimh-dsst/tdtr/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports jsonlite, reticulate (>= 1.41.0), rlang, tibble

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libpng-dev python3

Repository https://nimh-dsst.r-universe.dev

Date/Publication 2026-05-19 16:15:16 UTC

RemoteUrl https://github.com/nimh-dsst/tdtr

RemoteRef HEAD

RemoteSha 6827dbcb8da4e6adff4b91025849b1cc75efc16b

Contents

`as_ranges` *Convert time windows to Python* `tdt` *ranges*

Description

Python `tdt` expects a `2 x N` numeric matrix where row 1 is start time and row 2 is stop time.

Usage

```
as_ranges(ranges)
```

Arguments

`ranges`	A numeric length-2 vector, a `2 x N` matrix, an `N x 2` matrix, or a data frame with `start` and `stop` columns.

Value

A numeric `2 x N` matrix.

`as_tdt_block` *Coerce to a materialized TDT block*

Description

Coerce to a materialized TDT block

Usage

```
as_tdt_block(x, ...)
```

Arguments

`x`	A `tdt_block`, `tdt_block_py`, or compatible object.
`...`	Passed to `collect_block()`.

Value

A materialized `tdt_block`.

`as_tibble_epocs` *Return epocs/events as a tibble*

Description

Return epocs/events as a tibble

Usage

```
as_tibble_epocs(x, store = NULL)
```

Arguments

`x`	A `tdt_block`, `tdt_block_py`, or compatible object.
`store`	Optional epoc store name.

Value

A tibble with `store`, `onset`, `offset`, and `value` columns.

as_tibble_stream *Return a bounded stream table*

Description

Return a bounded stream table

Usage

```
as_tibble_stream(x, stream, window = NULL, downsample = NULL, max_rows = NULL)
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.
stream	Stream name.
window	Optional length-2 time window in seconds.
downsample	Optional integer stride.
max_rows	Optional maximum rows to return.

Value

A tibble with time, channel, and value columns.

block_info *Return block metadata*

Description

Return block metadata

Usage

```
block_info(x)
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.

Value

A named list.

collect_block *Collect a Python-backed TDT block into R*

Description

Collect a Python-backed TDT block into R

Usage

```
collect_block(
  x,
  streams = TRUE,
  epocs = TRUE,
  snips = FALSE,
  scalars = FALSE,
  stores = NULL,
  max_bytes_warn = 500 * 1024^2,
  quiet = FALSE
)
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.
streams	If TRUE, collect stream data.
epocs	If TRUE, collect epoc/event data.
snips, scalars	Reserved for future collection of Python tdt stores.
stores	Optional stream names to collect.
max_bytes_warn	Warn before copying a Python array larger than this many bytes.
quiet	Suppress size warnings.

Value

A materialized tdt_block.

collect_epocs *Collect epocs/events into R*

Description

Collect epocs/events into R

Usage

```
collect_epocs(x, store = NULL, as = c("tibble", "list"))
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.
store	Optional epoc store name. If NULL, all epocs are collected.
as	Return shape: "tibble" or "list".

Value

A tibble or named list of tibbles.

---

collect_stream *Collect one stream into R*

Description

Collect one stream into R

Usage

```
collect_stream(
  x,
  store,
  as = c("matrix", "numeric", "list"),
  include_time = FALSE,
  max_bytes_warn = 500 * 1024^2,
  quiet = FALSE
)
```

Arguments

x	A tdt_block, tdt_block_py, tdt_sev_py, or compatible object.
store	Stream store name. The sanitized Python key or original TDT store name may be used.
as	Return shape: "matrix", "numeric", or "list".
include_time	If TRUE, return a list with data, time, and stream metadata.
max_bytes_warn	Warn before copying a Python array larger than this many bytes.
quiet	Suppress size warnings.

Value

A matrix, numeric vector, or stream metadata list.

epoc *Return one epoc/event store*

Description

Return one epoc/event store

Usage

```
epoc(x, store)
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.
store	Epoc store name.

Value

One epoc object or tibble subset.

epoc_filter_py *Filter TDT epocs through Python* tdt

Description

Filter TDT epocs through Python tdt

Usage

```
epoc_filter_py(
  data,
  epoc,
  values = NULL,
  modifiers = NULL,
  t = NULL,
  tref = FALSE,
  keepdata = TRUE,
  ...
)
```

Arguments

data	A tdt_block_py wrapper or raw Python block object.
epoc	Epoc store name.
values, modifiers, t, tref, keepdata	Arguments passed to Python tdt.epoc_filter().
...	Additional keyword arguments passed to Python tdt.epoc_filter().

Value

A tdt_block_py wrapper.

---

epoc_names *Return epoc/event store names*

Description

Return epoc/event store names

Usage

```
epoc_names(x)
```

Arguments

x	A tdt_block, tdt_block_py, epoc container, or compatible object.

Value

A character vector of epoc store names.

---

epocs *Return the epoc/event container or rows*

Description

Return the epoc/event container or rows

Usage

```
epocs(x, store = NULL)
```

Arguments

x	A tdt_block, tdt_block_py, or compatible object.
store	Optional epoc store name.

Value

For materialized blocks, a tibble. For Python-backed blocks with no store, the live Python epocs container.

is_tdt_block *Test whether an object is a materialized TDT block*

Description

Test whether an object is a materialized TDT block

Usage

```
is_tdt_block(x)
```

Arguments

x	Object to test.

Value

TRUE if x inherits from tdt_block, otherwise FALSE.

new_tdt_block *Create a materialized TDT block*

Description

Create a materialized TDT block

Usage

```
new_tdt_block(info = list(), streams = list(), epocs = NULL)
```

Arguments

info	Named list of block metadata.
streams	Named list of stream objects.
epocs	Epoc/event table. If NULL, an empty tibble is used.

Value

A tdt_block object.

`profile_tdt_memory` *Profile memory movement for a TDT read workflow*

Description

`profile_tdt_memory()` runs a small read workflow and records elapsed time, R allocations reported by `Rprofmem()`, and Python allocations reported by Python `tracemalloc` when available. Use reader arguments such as `store`, `channel`, `t1`, `t2`, `evtype`, and `ranges` to test a bounded workflow before scaling up to a larger block.

Usage

```
profile_tdt_memory(
  block_path,
  ...,
  stream = NULL,
  events = TRUE,
  summarize = TRUE,
  quiet = TRUE
)
```

Arguments

`block_path`	Path to a TDT block directory.
`...`	Arguments passed to `read_block()`.
`stream`	Optional stream store to collect after the read. Leave `NULL` to profile metadata and events without copying stream data into R.
`events`	If `TRUE`, collect epocs/events after the read.
`summarize`	If `TRUE`, call `summary()` on the Python-backed block.
`quiet`	Suppress collection size warnings.

Details

The numbers are diagnostics, not portable benchmarks. They are most useful for comparing alternative reads on the same machine and Python environment.

Value

A tibble with one row per profiled step.

ranges_from_epocs *Build time ranges from epoc/event onsets*

Description

Build time ranges from epoc/event onsets

Usage

```
ranges_from_epocs(epocs, pre, post, onset_col = "onset", drop_negative = TRUE)
```

Arguments

epocs	A data frame with an onset column.
pre	Start offset in seconds relative to onset. Use a negative value for time before onset.
post	Stop offset in seconds relative to onset.
onset_col	Name of the onset column.
drop_negative	If TRUE, drop ranges with negative start times.

Value

A numeric 2 x N matrix.

read_block *Read a TDT block through Python* tdt

Description

Read a TDT block through Python tdt

Usage

```
read_block(block_path, ..., collect = FALSE)
```

Arguments

block_path	Path to a TDT block/tank directory.
...	Extra keyword arguments passed to tdt.read_block().
collect	If FALSE, return a Python-backed tdt_block_py wrapper. If TRUE, collect into a materialized R tdt_block.

Details

`read_block()` normalizes read-time `store` filters against the block header, so the stream names returned by `stream_names()` can also be used for filtering. This matters for TDT stores whose original names are not valid Python identifiers. For example, Python `tdt` filters a store originally named `465A` with `store = "465A"`, but the returned stream is exposed under the sanitized name `_465A`. `read_block()` accepts either spelling. Use `read_block_py()` when you need Python `tdt.read_block()` store matching exactly as implemented upstream.

Value

A `tdt_block_py` wrapper or a materialized `tdt_block`.

`read_block_py` *Read a TDT block through Python* `tdt`

Description

This is the explicit Python-backed compatibility wrapper for `tdt.read_block()`. It keeps Python objects live and does not copy stream arrays into R unless a collection helper is called later.

Usage

```
read_block_py(
  block_path,
  bitwise = "",
  channel = 0,
  combine = NULL,
  headers = 0,
  nodata = FALSE,
  ranges = NULL,
  store = "",
  t1 = 0,
  t2 = 0,
  evtype = NULL,
  verbose = 0,
  sortname = "TankSort",
  export = NULL,
  scale = 1,
  dtype = NULL,
  outdir = NULL,
  prefix = NULL,
  outfile = NULL,
  dmy = FALSE,
  noepocauto = FALSE,
  ...
)
```

Arguments

block_path	Path to a TDT block/tank directory.
bitwise, channel, combine, headers, nodata, ranges, store, t1, t2, evtype, verbose, sortname, export, scale, dtype, outdir, prefix, outfile, dmy, noepocauto	Arguments passed to Python tdt.read_block().
...	Additional keyword arguments passed to Python tdt.read_block().

Details

read_block_py() passes store directly to Python tdt.read_block(). For stores whose returned names are sanitized by Python, such as _465A, pass the original TDT store ID (465A) or use read_block() for tdtr's header-based normalization.

Value

A tdt_block_py wrapper.

---

read_epocs_csv *Read an epoc/event CSV export*

Description

Read an epoc/event CSV export

Usage

```
read_epocs_csv(path, ...)
```

Arguments

path	CSV file path.
...	Passed to utils::read.csv().

Value

A tibble with store, onset, offset, and value columns.

read_sev_py *Read TDT SEV data through Python* tdt

Description

Read TDT SEV data through Python tdt

Usage

```
read_sev_py(
  sev_dir,
  channel = 0,
  event_name = "",
  t1 = 0,
  t2 = 0,
  fs = 0,
  ranges = NULL,
  verbose = 0,
  just_names = FALSE,
  export = NULL,
  scale = 1,
  dtype = NULL,
  outdir = NULL,
  prefix = NULL,
  ...
)
```

Arguments

sev_dir	Path to a SEV directory or file.
channel, event_name, t1, t2, fs, ranges, verbose, just_names, export, scale, dtype, outdir, prefix	Arguments passed to Python tdt.read_sev().
...	Additional keyword arguments passed to Python tdt.read_sev().

Value

A tdt_sev_py wrapper.

read_stream_csv *Read a stream CSV export*

Description

Read a stream CSV export

Usage

```
read_stream_csv(path, fs, name, channels = NULL, t0 = 0, ...)
```

Arguments

path	CSV file path.
fs	Sampling frequency.
name	Stream name.
channels	Optional channel labels.
t0	Stream start time in seconds.
...	Passed to utils::read.csv().

Value

A materialized stream object.

stream *Return one stream*

Description

Return one stream

Usage

```
stream(x, name)
```

Arguments

x	A tdt_block, tdt_block_py, tdt_sev_py, or compatible object.
name	Stream name. The sanitized Python key or original TDT store name may be used.

Value

A stream object. For Python-backed blocks this is a live Python stream object and does not copy stream data into R.

stream_names *Return stream names*

Description

Return stream names

Usage

```
stream_names(x)
```

Arguments

x	A tdt_block, tdt_block_py, tdt_sev_py, stream container, or compatible object.

Value

A character vector of stream names.

streams *Return the stream container*

Description

Return the stream container

Usage

```
streams(x)
```

Arguments

x	A tdt_block, tdt_block_py, tdt_sev_py, or compatible object.

Value

A stream container. For Python-backed objects this may be a live Python object and does not copy stream arrays into R.

tdt_available *Check whether Python* tdt *is available*

Description

Check whether Python tdt is available

Usage

```
tdt_available(initialize = TRUE)
```

Arguments

initialize	If TRUE, allow reticulate to initialize Python while checking for tdt.

Value

TRUE if reticulate can find Python tdt, otherwise FALSE.

tdt_config *Show Python configuration for* tdtr

Description

Show Python configuration for tdtr

Usage

```
tdt_config(initialize = FALSE)
```

Arguments

initialize	If TRUE, include reticulate::py_config() details. This may initialize Python.

Value

A list with reticulate, Python, and Python tdt availability fields.

tdtr_example_block_path *Locate the packaged TDT example block*

Description

`tdtr` includes a small raw TDT block from the official TDT example data so examples, tests, and vignettes can exercise the reticulate-backed reader. The files are stored in the same block layout used by TDT, under `inst/extdata`.

Usage

```
tdtr_example_block_path(mustWork = TRUE)
```

Arguments

`mustWork`	If `TRUE`, error when the example block is not available.

Value

A single path to the packaged `Subject1-211115-094936` block.

validate_tdt_block *Validate a materialized TDT block*

Description

Validate a materialized TDT block

Usage

```
validate_tdt_block(x)
```

Arguments

`x`	Object to validate.

Value

`x`, invisibly.

Index