Package 'RcppTskit'

May 7, 2026

Type Package

Title 'R' Access to the 'tskit C' API

Version 0.2.0

Date 2026-01-27

Description 'Tskit' enables efficient storage, manipulation, and analysis of ancestral recombination graphs (ARGs) using succinct tree sequence encoding. The tree sequence encoding of an ARG is described in Wong et al. (2024) <doi:10.1093/genetics/iyae100>, while `tskit` project is described in Jeffrey et al. (2026) <doi:10.48550/arXiv.2602.09649>. See also <https://tskit.dev> for project news, documentation, and tutorials. 'Tskit' provides 'Python', 'C', and 'Rust' application programming interfaces (APIs). The 'Python' API can be called from 'R' via the 'reticulate' package to load and analyse tree sequences as described at <https://tskit.dev/tutorials/tskitr.html>. 'RcppTskit' provides 'R' access to the 'tskit C' API for cases where the 'reticulate' option is not optimal; for example, high-performance or low-level work with tree sequences. Currently, 'RcppTskit' provides a limited set of 'R' functions because the 'Python' API and 'reticulate' already covers most needs.

License MIT + file LICENSE

URL https://github.com/HighlanderLab/RcppTskit

BugReports https://github.com/HighlanderLab/RcppTskit/issues

Depends R (>= 4.0.0)

Imports methods, R6, Rcpp (>= 1.0.8), reticulate (>= 1.43.0)

Suggests covr, knitr, quarto, spelling, testthat (>= 3.0.0)

LinkingTo Rcpp (>= 0.12.10)

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

NeedsCompilation yes

Author Gregor Gorjanc [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-8008-2787>),
Tskit Developers [cph] (Authors of included tskit C library)

Maintainer Gregor Gorjanc <gregor.gorjanc@gmail.com>

Repository CRAN

Date/Publication 2026-02-27 16:02:24 UTC

Contents

`get_tskit_py` *Get the reticulate* `tskit` *Python module*

Description

This function imports the reticulate Python `tskit` module. If it is not yet installed, it attempts to install it first.

Usage

```
get_tskit_py(object_name = "tskit", force = FALSE)

check_tskit_py(object, stop = FALSE)
```

Arguments

`object_name`	character name of the object holding the reticulate `tskit` module. If this object exists in the global R environment and is a reticulate Python module, it is returned. Otherwise, the function attempts to install and import `tskit` before returning it.
`force`	logical; force installation and/or import before returning the reticulate Python module.
`object`	reticulate Python module.
`stop`	logical; whether to throw an error in `check_tskit_py`.

Details

This function is meant for users running `tskit <- get_tskit_py()` or similar code, and for other functions in this package that need the `tskit` reticulate Python module. The point of `get_tskit_py` is to avoid importing the module repeatedly; if it has been imported already, we reuse that instance. This process can be sensitive to the reticulate Python setup, module availability, and internet access.

Value

`get_tskit_py` returns the reticulate Python module `tskit` if successful. Otherwise it throws an error (when `object_name` exists but is not a reticulate Python module) or returns `simpleError` (when installation or import failed). `check_tskit_py` returns `TRUE` if `object` is a reticulate Python module or `FALSE` otherwise.

Functions

- `check_tskit_py()`: Test whether `get_tskit_py` returned a reticulate Python module object

Examples

```
## Not run:
  tskit <- get_tskit_py()
  is(tskit)
  if (check_tskit_py(tskit)) {
    tskit$ALLELES_01
  }

## End(Not run)
```

---

kastore_version *Report the version of installed kastore C API*

Description

Report the version of installed kastore C API

Usage

```
kastore_version()
```

Details

The version is stored in the installed header `kastore.h`.

Value

A named vector with three elements `major`, `minor`, and `patch`.

Examples

```
kastore_version()
```

TableCollection *Table collection R6 class (TableCollection)*

Description

An R6 class holding an external pointer to a table collection object. As an R6 class, its methods look Pythonic and therefore resemble the tskit Python API. Since the class only holds the pointer, it is lightweight. Currently there is a limited set of R methods for working the tree sequence.

Public fields

`pointer` external pointer to the table collection

Methods

Public methods:

- `TableCollection$new()`
- `TableCollection$dump()`
- `TableCollection$write()`
- `TableCollection$tree_sequence()`
- `TableCollection$r_to_py()`
- `TableCollection$print()`
- `TableCollection$clone()`

Method `new()`**:** Create a `TableCollection` from a file or a pointer.

Usage:

```
TableCollection$new(
  file,
  skip_tables = FALSE,
  skip_reference_sequence = FALSE,
  pointer = NULL
)
```

Arguments:

`file` a string specifying the full path of the tree sequence file.

`skip_tables` logical; if `TRUE`, load only non-table information.

`skip_reference_sequence` logical; if `TRUE`, skip loading reference sequence information.

`pointer` an external pointer (`externalptr`) to a table collection.

Details: See the corresponding Python function at https://github.com/tskit-dev/tskit/blob/dc394d72d121c99c6dcad88f7a4873880924dd72/python/tskit/tables.py#L3463.

Returns: A `TableCollection` object.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
is(tc)
tc
```

Method `dump()`**:** Write a table collection to a file.

Usage:

```
TableCollection$dump(file)
```

Arguments:

`file` a string specifying the full path of the tree sequence file.

Details: See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TableCollection.dump.

Returns: No return value; called for side effects.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
dump_file <- tempfile()
tc$dump(dump_file)
tc$write(dump_file) # alias
\dontshow{file.remove(dump_file)}
```

Method `write()`**:** Alias for `TableCollection$dump`.

Usage:

```
TableCollection$write(file)
```

Arguments:

`file` see `TableCollection$dump`.

Method `tree_sequence()`**:** Create a `TreeSequence` from this table collection.

Usage:

```
TableCollection$tree_sequence()
```

Details: See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TableCollection.tree_sequence.

Returns: A `TreeSequence` object.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
ts <- tc$tree_sequence()
is(ts)
```

Method `r_to_py()`**:** This function saves a table collection from R to disk and loads it into reticulate Python for use with the `tskit` Python API.

Usage:

```
TableCollection$r_to_py(tskit_module = get_tskit_py(), cleanup = TRUE)
```

Arguments:

`tskit_module` reticulate Python module of `tskit`. By default, it calls `get_tskit_py` to obtain the module.

`cleanup` logical; delete the temporary file at the end of the function?

Details: See https://tskit.dev/tutorials/tables_and_editing.html#tables-and-editing on what you can do with the tables.

Returns: Table collection in reticulate Python.

Examples:

```
\dontrun{
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")
  tc_r <- tc_load(ts_file)
  is(tc_r)
  tc_r$print()

 # Transfer the table collection to reticulate Python and use tskit Python API
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    tc_py <- tc_r$r_to_py()
    is(tc_py)
    tmp <- tc_py$simplify(samples = c(0L, 1L, 2L, 3L))
    tmp
    tc_py$individuals$num_rows # 2
    tc_py$nodes$num_rows # 8
    tc_py$nodes$time # 0.0 ... 5.0093910
  }
}
```

Method `print()`**:** Print a summary of a table collection and its contents.

Usage:

```
TableCollection$print()
```

Returns: A list with two data.frames; the first contains table collection properties and their values; the second contains the number of rows in each table and the length of their metadata.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(file = ts_file)
tc$print()
tc
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TableCollection$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

tc_py_to_r, tc_load, and TableCollection$dump.

Examples

```
## ------------------------------------------------
## Method `TableCollection$new`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
is(tc)
tc

## ------------------------------------------------
## Method `TableCollection$dump`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
dump_file <- tempfile()
tc$dump(dump_file)
tc$write(dump_file) # alias


## ------------------------------------------------
## Method `TableCollection$tree_sequence`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- TableCollection$new(file = ts_file)
ts <- tc$tree_sequence()
is(ts)

## ------------------------------------------------
## Method `TableCollection$r_to_py`
## ------------------------------------------------

## Not run:
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")
  tc_r <- tc_load(ts_file)
  is(tc_r)
  tc_r$print()

  # Transfer the table collection to reticulate Python and use tskit Python API
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    tc_py <- tc_r$r_to_py()
    is(tc_py)
    tmp <- tc_py$simplify(samples = c(0L, 1L, 2L, 3L))
    tmp
    tc_py$individuals$num_rows # 2
```

```
    tc_py$nodes$num_rows # 8
    tc_py$nodes$time # 0.0 ... 5.0093910
  }

## End(Not run)

## ------------------------------------------------
## Method `TableCollection$print`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(file = ts_file)
tc$print()
tc
```

`tc_load` *Load a table collection from a file*

Description

Load a table collection from a file

Usage

```
tc_load(file, skip_tables = FALSE, skip_reference_sequence = FALSE)

tc_read(file, skip_tables = FALSE, skip_reference_sequence = FALSE)
```

Arguments

`file`	a string specifying the full path to a tree sequence file.
`skip_tables`	logical; if `TRUE`, load only non-table information.
`skip_reference_sequence`	logical; if `TRUE`, skip loading reference sequence information.

Details

See the corresponding Python function at https://github.com/tskit-dev/tskit/blob/dc394d72d121c99c6dcad88f7a
python/tskit/tables.py#L3463.

Value

A `TableCollection` object.

Functions

- `tc_read()`: Alias for `tc_load()`

See Also

TableCollection$new

Examples

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
is(tc)
tc
```

---

tc_py_to_r *Transfer a table collection from reticulate Python to R*

Description

This function saves a table collection from reticulate Python to disk and reads it into R for use with RcppTskit.

Usage

```
tc_py_to_r(tc, cleanup = TRUE)
```

Arguments

tc	table collection in reticulate Python.
cleanup	logical; delete the temporary file at the end of the function?

Value

A TableCollection object.

See Also

TableCollection$r_to_py tc_load, and TableCollection$dump.

Examples

```
## Not run:
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")

  # Use the tskit Python API to work with a table collection (via reticulate)
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    tc_py <- tskit$TableCollection$load(ts_file)
    is(tc_py)
    tc_py$individuals$num_rows # 8
    tmp <- tc_py$simplify(samples = c(0L, 1L, 2L, 3L))
    tmp
```

```
  tc_py$individuals$num_rows # 2
  tc_py$nodes$num_rows # 8
  tc_py$nodes$time # 0.0 ... 5.0093910

  # Transfer the table collection to R and use RcppTskit
  tc_r <- tc_py_to_r(tc_py)
  is(tc_r)
 tc_r$print()
}

## End(Not run)
```

TreeSequence *Succinct tree sequence R6 class (TreeSequence)*

Description

An R6 class holding an external pointer to a tree sequence object. As an R6 class, its methods look Pythonic and therefore resemble the tskit Python API. Since the class only holds the pointer, it is lightweight. Currently there is a limited set of R methods for working with the tree sequence.

Public fields

`pointer` external pointer to the tree sequence

Methods

Public methods:

- `TreeSequence$new()`
- `TreeSequence$dump()`
- `TreeSequence$write()`
- `TreeSequence$dump_tables()`
- `TreeSequence$print()`
- `TreeSequence$r_to_py()`
- `TreeSequence$num_provenances()`
- `TreeSequence$num_populations()`
- `TreeSequence$num_migrations()`
- `TreeSequence$num_individuals()`
- `TreeSequence$num_samples()`
- `TreeSequence$num_nodes()`
- `TreeSequence$num_edges()`
- `TreeSequence$num_trees()`
- `TreeSequence$num_sites()`
- `TreeSequence$num_mutations()`
- `TreeSequence$sequence_length()`

- TreeSequence$time_units()
- TreeSequence$min_time()
- TreeSequence$max_time()
- TreeSequence$metadata_length()
- TreeSequence$clone()

Method `new()`**:** Create a TreeSequence from a file or a pointer. See ts_load for details and examples.

Usage:

```
TreeSequence$new(
  file,
  skip_tables = FALSE,
  skip_reference_sequence = FALSE,
  pointer = NULL
)
```

Arguments:

`file` a string specifying the full path of the tree sequence file.

`skip_tables` logical; if `TRUE`, load only non-table information.

`skip_reference_sequence` logical; if `TRUE`, skip loading reference sequence information.

`pointer` an external pointer (`externalptr`) to a tree sequence.

Details: See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.load.

Returns: A TreeSequence object.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- TreeSequence$new(file = ts_file)
is(ts)
ts
ts$num_nodes()
# Also
ts <- ts_load(ts_file)
is(ts)
```

Method `dump()`**:** Write a tree sequence to a file.

Usage:

```
TreeSequence$dump(file)
```

Arguments:

`file` a string specifying the full path of the tree sequence file.

Details: See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.dump.

Returns: No return value; called for side effects.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
dump_file <- tempfile()
ts$dump(dump_file)
ts$write(dump_file) # alias
\dontshow{file.remove(dump_file)}
```

Method `write()`**:** Alias for `TreeSequence$dump`.

Usage:

```
TreeSequence$write(file)
```

Arguments:

`file` see `TreeSequence$dump`.

Method `dump_tables()`**:** Copy the tables into a `TableCollection`.

Usage:

```
TreeSequence$dump_tables()
```

Details: See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.dump_tables.

Returns: A `TableCollection` object.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
tc <- ts$dump_tables()
is(tc)
```

Method `print()`**:** Print a summary of a tree sequence and its contents.

Usage:

```
TreeSequence$print()
```

Returns: A list with two data.frames; the first contains tree sequence properties and their values; the second contains the number of rows in each table and the length of their metadata.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$print()
ts
```

Method `r_to_py()`**:** This function saves a tree sequence from R to disk and loads it into reticulate Python for use with the `tskit` Python API.

Usage:

```
TreeSequence$r_to_py(tskit_module = get_tskit_py(), cleanup = TRUE)
```

Arguments:

`tskit_module` reticulate Python module of `tskit`. By default, it calls `get_tskit_py` to obtain the module.

`cleanup` logical; delete the temporary file at the end of the function?

Returns: Tree sequence in reticulate Python.

Examples:

```
\dontrun{
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")
  ts_r <- ts_load(ts_file)
  is(ts_r)
  ts_r$num_individuals() # 8

  # Transfer the tree sequence to reticulate Python and use tskit Python API
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    ts_py <- ts_r$r_to_py()
    is(ts_py)
    ts_py$num_individuals # 8
    ts2_py <- ts_py$simplify(samples = c(0L, 1L, 2L, 3L))
    ts_py$num_individuals # 8
    ts2_py$num_individuals # 2
    ts2_py$num_nodes # 8
    ts2_py$tables$nodes$time # 0.0 ... 5.0093910
  }
}
```

Method `num_provenances()`**:** Get the number of provenances in a tree sequence.

Usage:

```
TreeSequence$num_provenances()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_provenances()
```

Method `num_populations()`**:** Get the number of populations in a tree sequence.

Usage:

```
TreeSequence$num_populations()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_populations()
```

Method `num_migrations()`**:** Get the number of migrations in a tree sequence.

Usage:

```
TreeSequence$num_migrations()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_migrations()
```

Method `num_individuals()`**:** Get the number of individuals in a tree sequence.

Usage:

```
TreeSequence$num_individuals()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_individuals()
```

Method `num_samples()`**:** Get the number of samples (of nodes) in a tree sequence.

Usage:

```
TreeSequence$num_samples()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_samples()
```

Method `num_nodes()`**:** Get the number of nodes in a tree sequence.

Usage:

```
TreeSequence$num_nodes()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_nodes()
```

Method `num_edges()`**:** Get the number of edges in a tree sequence.

Usage:

```
TreeSequence$num_edges()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_edges()
```

Method `num_trees()`**:** Get the number of trees in a tree sequence.

Usage:

```
TreeSequence$num_trees()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_trees()
```

Method `num_sites()`**:** Get the number of sites in a tree sequence.

Usage:

```
TreeSequence$num_sites()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_sites()
```

Method `num_mutations()`**:** Get the number of mutations in a tree sequence.

Usage:

```
TreeSequence$num_mutations()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_mutations()
```

Method `sequence_length()`**:** Get the sequence length.

Usage:

```
TreeSequence$sequence_length()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$sequence_length()
```

Method `time_units()`**:** Get the time units string.

Usage:

```
TreeSequence$time_units()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$time_units()
```

Method `min_time()`**:** Get the min time in node table and mutation table.

Usage:

```
TreeSequence$min_time()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$min_time()
```

Method `max_time()`**:** Get the max time in node table and mutation table.

Usage:

```
TreeSequence$max_time()
```

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$max_time()
```

Method `metadata_length()`**:** Get the length of metadata in a tree sequence and its tables.

Usage:

```
TreeSequence$metadata_length()
```

Returns: A named list with the length of metadata.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$metadata_length()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TreeSequence$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

`ts_load`

`ts_py_to_r`, `ts_load`, and `TreeSequence$dump`.

Examples

```
## ------------------------------------------------
## Method `TreeSequence$new`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- TreeSequence$new(file = ts_file)
is(ts)
ts
ts$num_nodes()
# Also
ts <- ts_load(ts_file)
is(ts)

## ------------------------------------------------
## Method `TreeSequence$dump`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
dump_file <- tempfile()
ts$dump(dump_file)
ts$write(dump_file) # alias


## ------------------------------------------------
## Method `TreeSequence$dump_tables`
## ------------------------------------------------
```

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
tc <- ts$dump_tables()
is(tc)

## ------------------------------------------------
## Method `TreeSequence$print`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$print()
ts

## ------------------------------------------------
## Method `TreeSequence$r_to_py`
## ------------------------------------------------

## Not run:
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")
  ts_r <- ts_load(ts_file)
  is(ts_r)
  ts_r$num_individuals() # 8

  # Transfer the tree sequence to reticulate Python and use tskit Python API
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    ts_py <- ts_r$r_to_py()
    is(ts_py)
    ts_py$num_individuals # 8
    ts2_py <- ts_py$simplify(samples = c(0L, 1L, 2L, 3L))
    ts_py$num_individuals # 8
    ts2_py$num_individuals # 2
    ts2_py$num_nodes # 8
    ts2_py$tables$nodes$time # 0.0 ... 5.0093910
  }

## End(Not run)

## ------------------------------------------------
## Method `TreeSequence$num_provenances`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_provenances()

## ------------------------------------------------
## Method `TreeSequence$num_populations`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
```

```
ts <- ts_load(ts_file)
ts$num_populations()

## ------------------------------------------------
## Method `TreeSequence$num_migrations`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_migrations()

## ------------------------------------------------
## Method `TreeSequence$num_individuals`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_individuals()

## ------------------------------------------------
## Method `TreeSequence$num_samples`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_samples()

## ------------------------------------------------
## Method `TreeSequence$num_nodes`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_nodes()

## ------------------------------------------------
## Method `TreeSequence$num_edges`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_edges()

## ------------------------------------------------
## Method `TreeSequence$num_trees`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_trees()

## ------------------------------------------------
## Method `TreeSequence$num_sites`
```

```
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_sites()

## ------------------------------------------------
## Method `TreeSequence$num_mutations`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$num_mutations()

## ------------------------------------------------
## Method `TreeSequence$sequence_length`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$sequence_length()

## ------------------------------------------------
## Method `TreeSequence$time_units`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$time_units()

## ------------------------------------------------
## Method `TreeSequence$min_time`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$min_time()

## ------------------------------------------------
## Method `TreeSequence$max_time`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$max_time()

## ------------------------------------------------
## Method `TreeSequence$metadata_length`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
ts$metadata_length()
```

tskit_version *Report the version of installed* tskit *C API*

Description

Report the version of installed tskit C API

Usage

```
tskit_version()
```

Details

The version is defined in the installed header tskit/core.h.

Value

A named vector with three elements major, minor, and patch.

Examples

```
tskit_version()
```

ts_load *Load a tree sequence from a file*

Description

Load a tree sequence from a file

Usage

```
ts_load(file, skip_tables = FALSE, skip_reference_sequence = FALSE)

ts_read(file, skip_tables = FALSE, skip_reference_sequence = FALSE)
```

Arguments

file	a string specifying the full path to a tree sequence file.
skip_tables	logical; if TRUE, load only non-table information.
skip_reference_sequence	logical; if TRUE, skip loading reference sequence information.

Details

See the corresponding Python function at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.load.

Value

A TreeSequence object.

Functions

- ts_read(): Alias for ts_load()

See Also

TreeSequence$new

Examples

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
ts <- ts_load(ts_file)
is(ts)
ts
ts$num_nodes()
# Also
ts <- TreeSequence$new(file = ts_file)
is(ts)
```

---

ts_py_to_r *Transfer a tree sequence from reticulate Python to R*

Description

This function saves a tree sequence from reticulate Python to disk and reads it into R for use with RcppTskit.

Usage

```
ts_py_to_r(ts, cleanup = TRUE)
```

Arguments

ts	tree sequence in reticulate Python.
cleanup	logical; delete the temporary file at the end of the function?

Value

A TreeSequence object.

See Also

TreeSequence$r_to_py ts_load, and TreeSequence$dump.

Examples

```
## Not run:
  ts_file <- system.file("examples/test.trees", package = "RcppTskit")

  # Use the tskit Python API to work with a tree sequence (via reticulate)
  tskit <- get_tskit_py()
  if (check_tskit_py(tskit)) {
    ts_py <- tskit$load(ts_file)
    is(ts_py)
    ts_py$num_individuals # 8
    ts2_py <- ts_py$simplify(samples = c(0L, 1L, 2L, 3L))
    ts_py$num_individuals # 8
    ts2_py$num_individuals # 2
    ts2_py$num_nodes # 8
    ts2_py$tables$nodes$time # 0.0 ... 5.0093910

    # Transfer the tree sequence to R and use RcppTskit
    ts2_r <- ts_py_to_r(ts2_py)
    is(ts2_r)
    ts2_r$num_individuals() # 2
  }

## End(Not run)
```

Index