Package: RcppTskit (via r-universe)

May 10, 2026

Type Package

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

Version 0.3.0

Date 2026-03-01

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 functions because the 'Python' API and 'reticulate' already cover most needs. The provided 'RcppTskit R' API mirrors the 'tskit Python' API, while the 'RcppTskit C++' API mirrors the 'tskit C' API. Users should explore the 'RcppTskit' help pages of 'R' functions, while developers should explore the 'RcppTskit:::rtsk_*' low-level 'R' and 'C++" functions.

License MIT + file LICENSE

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

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

Depends R (>= 4.0.0)

Imports bit64, 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

Config/pak/sysreqs libpng-dev python3

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

Date/Publication 2026-04-10 06:03:43 UTC

RemoteUrl https://github.com/highlanderlab/RcppTskit

RemoteRef HEAD

RemoteSha 45a46563597fa1b39015319b367c2d7ede0435ef

RemoteSubdir RcppTskit

Contents

get_tskit_py 2
kastore_version 3
TableCollection 4
tc_load 19
tc_py_to_r 20
TreeSequence 21
ts_load 34
ts_py_to_r 36
tskit_version 37

Index **38**

`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, method-calling looks Pythonic and hence resembles 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 table collection object.

Public fields

`xptr` external pointer to the table collection

Methods

Public methods:

- `TableCollection$new()`
- `TableCollection$dump()`
- `TableCollection$write()`
- `TableCollection$tree_sequence()`
- `TableCollection$num_provenances()`
- `TableCollection$num_populations()`
- `TableCollection$num_migrations()`
- `TableCollection$num_individuals()`
- `TableCollection$individual_table_add_row()`
- `TableCollection$num_nodes()`
- `TableCollection$node_table_add_row()`
- `TableCollection$num_edges()`
- `TableCollection$edge_table_add_row()`

- TableCollection$num_sites()
- TableCollection$site_table_add_row()
- TableCollection$num_mutations()
- TableCollection$mutation_table_add_row()
- TableCollection$sequence_length()
- TableCollection$time_units()
- TableCollection$has_index()
- TableCollection$build_index()
- TableCollection$drop_index()
- TableCollection$has_reference_sequence()
- TableCollection$file_uuid()
- TableCollection$r_to_py()
- TableCollection$print()
- TableCollection$clone()

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

Usage:

```
TableCollection$new(
  file,
  skip_tables = FALSE,
  skip_reference_sequence = FALSE,
  xptr = 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 genome sequence information.

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

Details: See the `tskit` Python equivalent at https://github.com/tskit-dev/tskit/blob/dc394d72d121c99c6dcad88f7a4873880924dd72/python/tskit/tables.py#L3463. TODO: Update URL to TableCollection.load() method #104 https://github.com/HighlanderLab/RcppTskit/issues/104

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 `tskit` Python equivalent 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 `tskit` Python equivalent 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 `num_provenances()`**:** Get the number of provenances in a table collection.

Usage:

```
TableCollection$num_provenances()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `num_populations()`**:** Get the number of populations in a table collection.

Usage:

```
TableCollection$num_populations()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `num_migrations()`**:** Get the number of migrations in a table collection.

Usage:

```
TableCollection$num_migrations()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `num_individuals()`**:** Get the number of individuals in a table collection.

Usage:

```
TableCollection$num_individuals()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `individual_table_add_row()`**:** Add a row to the individuals table.

Usage:

```
TableCollection$individual_table_add_row(
  flags = 0L,
  location = NULL,
  parents = NULL,
  metadata = NULL
)
```

Arguments:

`flags` integer scalar flags for the new individual.

`location` numeric vector with the location of the new individual; can be `NULL` if unknown.

`parents` integer vector with parent individual IDs (0-based); can be `NULL` if unknown

`metadata` for the new individual; accepts `NULL`, a raw vector, or a character of length 1.

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/stable/python-api.html#tskit.IndividualTable.add_row.

Returns: Integer row ID (0-based) of the newly added individual.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_individuals())
new_id <- tc$individual_table_add_row()
new_id <- tc$individual_table_add_row(location = c(5, 8))
new_id <- tc$individual_table_add_row(flags = 0L)
new_id <- tc$individual_table_add_row(parents = c(0L, 2L))
new_id <- tc$individual_table_add_row(metadata = "abc")
new_id <- tc$individual_table_add_row(metadata = charToRaw("cba"))
(n_after <- tc$num_individuals())
```

Method `num_nodes()`**:** Get the number of nodes in a table collection.

Usage:

```
TableCollection$num_nodes()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `node_table_add_row()`**:** Add a row to the nodes table.

Usage:

```
TableCollection$node_table_add_row(
  flags = 0L,
  time = 0,
  population = -1L,
  individual = -1L,
  metadata = NULL
)
```

Arguments:

`flags` integer scalar flags for the new node.

`time` numeric scalar time value for the new node.

`population` integer scalar population row ID (0-based); use `-1` if not known - `NULL` maps to `-1` (`TSK_NULL`).

`individual` integer scalar individual row ID (0-based); use `-1` if not known - `NULL` maps to `-1` (`TSK_NULL`).

`metadata` for the new node; accepts `NULL`, a raw vector, or a character of length 1.

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/stable/python-api.html#tskit.NodeTable.add_row.

Returns: Integer row ID (0-based) of the newly added node.

Examples:

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

```
new_id <- tc$node_table_add_row()
new_id <- tc$node_table_add_row(time = 2.5)
new_id <- tc$node_table_add_row(flags = 1L, time = 3.5, population = 0L)
new_id <- tc$node_table_add_row(flags = 1L, time = 4.5, individual = 0L)
new_id <- tc$node_table_add_row(metadata = "abc")
new_id <- tc$node_table_add_row(metadata = charToRaw("cba"))
(n_after <- tc$num_nodes())
```

Method `num_edges()`**:** Get the number of edges in a table collection.

Usage:

```
TableCollection$num_edges()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `edge_table_add_row()`**:** Add a row to the edges table.

Usage:

```
TableCollection$edge_table_add_row(left, right, parent, child, metadata = NULL)
```

Arguments:

`left` numeric scalar left coordinate for the new edge.

`right` numeric scalar right coordinate for the new edge.

`parent` integer scalar parent node row ID (0-based).

`child` integer scalar child node row ID (0-based).

`metadata` for the new edge; accepts `NULL`, a raw vector, or a character of length 1.

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/stable/python-api.html#tskit.EdgeTable.add_row.

Returns: Integer row ID (0-based) of the newly added edge.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
child <- tc$node_table_add_row(time = 0.0)
(n_before <- tc$num_edges())
new_id <- tc$edge_table_add_row(
  left = 0, right = 50, parent = 16L, child = child
)
new_id <- tc$edge_table_add_row(
  left = 50, right = 75, parent = 17L, child = child, metadata = "abc"
)
new_id <- tc$edge_table_add_row(
 left = 75, right = 100, parent = 18L, child = child, metadata = charToRaw("cba")
)
(n_after <- tc$num_edges())
```

Method `num_sites()`**:** Get the number of sites in a table collection.

Usage:

```
TableCollection$num_sites()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `site_table_add_row()`**:** Add a row to the sites table.

Usage:

```
TableCollection$site_table_add_row(position, ancestral_state, metadata = NULL)
```

Arguments:

`position` numeric scalar site position.

`ancestral_state` character string for the new site.

`metadata` for the new site; accepts `NULL`, a raw vector, or a character of length 1.

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/stable/python-api.html#tskit.SiteTable.add_row.

Returns: Integer row ID (0-based) of the newly added site.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_sites())
new_id <- tc$site_table_add_row(position = 0.5, ancestral_state = "A")
new_id <- tc$site_table_add_row(position = 2.5, ancestral_state = "T", metadata = "abc")
(n_after <- tc$num_sites())
```

Method `num_mutations()`**:** Get the number of mutations in a table collection.

Usage:

```
TableCollection$num_mutations()
```

Returns: A signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `mutation_table_add_row()`**:** Add a row to the mutations table.

Usage:

```
TableCollection$mutation_table_add_row(
  site,
  node,
  derived_state,
  parent = -1L,
  metadata = NULL,
  time = NaN
)
```

Arguments:

`site` integer scalar site row ID (0-based).

`node` integer scalar node row ID (0-based).

`derived_state` character string for the new mutation.

`parent` integer scalar parent mutation row ID (0-based); use `-1` if not known - `NULL` maps to `-1` (`TSK_NULL`).

`metadata` for the new mutation; accepts `NULL`, a raw vector, or a character of length 1.

`time` numeric scalar mutation time; use `NaN` if not known - `NULL` maps to `NaN` (`TSK_UNKNOWN_TIME`).

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/stable/python-api.html#tskit.MutationTable.add_row.

Returns: Integer row ID (0-based) of the newly added mutation.

Examples:

```
ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_mutations())
# From inspection of tc we have:
# node13(time=0) <- node16(time=0.02...) <- node20(time=0.08...)
# Add mutation above 16L
m0 <- tc$mutation_table_add_row(site = 0L, node = 16L, derived_state = "T", time = 0.03)
# Add mutation above 13L
m1 <- tc$mutation_table_add_row(
  site = 0L,
  node = 13L,
  parent = m0,
  time = 0.01,
  derived_state = "C",
  metadata = "abc"
)
(n_after <- tc$num_mutations())
```

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

Usage:

```
TableCollection$sequence_length()
```

Returns: A numeric.

Examples:

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

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

Usage:

```
TableCollection$time_units()
```

Returns: A character.

Examples:

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

Method `has_index()`**:** Get whether the table collection has edge indexes.

Usage:

```
TableCollection$has_index()
```

Returns: A logical.

Examples:

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

Method `build_index()`**:** Build edge indexes for this table collection.

Usage:

```
TableCollection$build_index()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TableCollection.build_index.

Returns: No return value; called for side effects.

Examples:

```
tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$has_index()
tc$drop_index()
tc$has_index()
tc$build_index()
tc$has_index()
```

Method `drop_index()`**:** Drop edge indexes for this table collection.

Usage:

```
TableCollection$drop_index()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TableCollection.drop_index.

Returns: No return value; called for side effects.

Examples:

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

Method `has_reference_sequence()`**:** Get whether the table collection has a reference genome sequence.

Usage:

```
TableCollection$has_reference_sequence()
```

Returns: A logical.

Examples:

```
tc_file1 <- system.file("examples/test.trees", package = "RcppTskit")
tc_file2 <- system.file("examples/test_with_ref_seq.trees", package = "RcppTskit")
tc1 <- tc_load(tc_file1)
tc1$has_reference_sequence()
tc2 <- tc_load(tc_file2)
tc2$has_reference_sequence()
```

Method `file_uuid()`**:** Get the UUID string of the file the table collection was loaded from.

Usage:

```
TableCollection$file_uuid()
```

Returns: A character; `NA_character_` when file is information is unavailable.

Examples:

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

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: `TableCollection` object 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. All columns are characters since output types differ across the entries. Use individual getters to obtain raw values before they are converted to character.

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$num_provenances`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_provenances()

## ------------------------------------------------
## Method `TableCollection$num_populations`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_populations()

## ------------------------------------------------
## Method `TableCollection$num_migrations`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_migrations()

## ------------------------------------------------
## Method `TableCollection$num_individuals`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_individuals()

## ------------------------------------------------
## Method `TableCollection$individual_table_add_row`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_individuals())
new_id <- tc$individual_table_add_row()
new_id <- tc$individual_table_add_row(location = c(5, 8))
```

```
new_id <- tc$individual_table_add_row(flags = 0L)
new_id <- tc$individual_table_add_row(parents = c(0L, 2L))
new_id <- tc$individual_table_add_row(metadata = "abc")
new_id <- tc$individual_table_add_row(metadata = charToRaw("cba"))
(n_after <- tc$num_individuals())

## ------------------------------------------------
## Method `TableCollection$num_nodes`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_nodes()

## ------------------------------------------------
## Method `TableCollection$node_table_add_row`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_nodes())
new_id <- tc$node_table_add_row()
new_id <- tc$node_table_add_row(time = 2.5)
new_id <- tc$node_table_add_row(flags = 1L, time = 3.5, population = 0L)
new_id <- tc$node_table_add_row(flags = 1L, time = 4.5, individual = 0L)
new_id <- tc$node_table_add_row(metadata = "abc")
new_id <- tc$node_table_add_row(metadata = charToRaw("cba"))
(n_after <- tc$num_nodes())

## ------------------------------------------------
## Method `TableCollection$num_edges`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_edges()

## ------------------------------------------------
## Method `TableCollection$edge_table_add_row`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
child <- tc$node_table_add_row(time = 0.0)
(n_before <- tc$num_edges())
new_id <- tc$edge_table_add_row(
  left = 0, right = 50, parent = 16L, child = child
)
new_id <- tc$edge_table_add_row(
  left = 50, right = 75, parent = 17L, child = child, metadata = "abc"
)
new_id <- tc$edge_table_add_row(
  left = 75, right = 100, parent = 18L, child = child, metadata = charToRaw("cba")
```

```
)
(n_after <- tc$num_edges())

## ------------------------------------------------
## Method `TableCollection$num_sites`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_sites()

## ------------------------------------------------
## Method `TableCollection$site_table_add_row`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_sites())
new_id <- tc$site_table_add_row(position = 0.5, ancestral_state = "A")
new_id <- tc$site_table_add_row(position = 2.5, ancestral_state = "T", metadata = "abc")
(n_after <- tc$num_sites())

## ------------------------------------------------
## Method `TableCollection$num_mutations`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$num_mutations()

## ------------------------------------------------
## Method `TableCollection$mutation_table_add_row`
## ------------------------------------------------

ts_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(ts_file)
(n_before <- tc$num_mutations())
# From inspection of tc we have:
# node13(time=0) <- node16(time=0.02...) <- node20(time=0.08...)
# Add mutation above 16L
m0 <- tc$mutation_table_add_row(site = 0L, node = 16L, derived_state = "T", time = 0.03)
# Add mutation above 13L
m1 <- tc$mutation_table_add_row(
  site = 0L,
  node = 13L,
  parent = m0,
  time = 0.01,
  derived_state = "C",
  metadata = "abc"
)
(n_after <- tc$num_mutations())

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

```
## Method `TableCollection$sequence_length`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$sequence_length()

## ------------------------------------------------
## Method `TableCollection$time_units`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$time_units()

## ------------------------------------------------
## Method `TableCollection$has_index`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$has_index()

## ------------------------------------------------
## Method `TableCollection$build_index`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$has_index()
tc$drop_index()
tc$has_index()
tc$build_index()
tc$has_index()

## ------------------------------------------------
## Method `TableCollection$drop_index`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$has_index()
tc$drop_index()
tc$has_index()

## ------------------------------------------------
## Method `TableCollection$has_reference_sequence`
## ------------------------------------------------

tc_file1 <- system.file("examples/test.trees", package = "RcppTskit")
tc_file2 <- system.file("examples/test_with_ref_seq.trees", package = "RcppTskit")
tc1 <- tc_load(tc_file1)
tc1$has_reference_sequence()
```

```
tc2 <- tc_load(tc_file2)
tc2$has_reference_sequence()

## ------------------------------------------------
## Method `TableCollection$file_uuid`
## ------------------------------------------------

tc_file <- system.file("examples/test.trees", package = "RcppTskit")
tc <- tc_load(tc_file)
tc$file_uuid()

## ------------------------------------------------
## 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 genome sequence information.

Details

See the tskit Python equivalent at https://github.com/tskit-dev/tskit/blob/dc394d72d121c99c6dcad88f7a4873
python/tskit/tables.py#L3463. TODO: Update URL to TableCollection.load() method #104
https://github.com/HighlanderLab/RcppTskit/issues/104

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 temporary file on 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?

Details

Because this transfer is via a temporary file, the file UUID property changes.

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, method-calling looks Pythonic and hence resembles 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

`xptr` 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$sequence_length()`
- `TreeSequence$discrete_genome()`
- `TreeSequence$has_reference_sequence()`
- `TreeSequence$time_units()`
- `TreeSequence$discrete_time()`
- `TreeSequence$min_time()`
- `TreeSequence$max_time()`
- `TreeSequence$metadata_length()`
- `TreeSequence$file_uuid()`
- `TreeSequence$clone()`

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

Usage:

```
TreeSequence$new(
  file,
  skip_tables = FALSE,
  skip_reference_sequence = FALSE,
  xptr = 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 genome sequence information.

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

Details: See the `tskit` Python equivalent 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 `tskit` Python equivalent 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 tskit Python equivalent 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. All columns are characters since output types differ across the entries. Use individual getters to obtain raw values before they are converted to character.

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: `TreeSequence` object 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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_provenances.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_populations.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_migrations.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_individuals.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_samples.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_nodes.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_nodes.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_trees.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_sites.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.num_mutations.

Returns: A signed 64 bit integer `bit64::integer64`.

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()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.sequence_length.

Returns: A numeric.

Examples:

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

Method `discrete_genome()`**:** Get the discrete genome status.

Usage:

```
TreeSequence$discrete_genome()
```

Details: Returns `TRUE` if all genomic coordinates in the tree sequence are discrete integer values.
See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.discrete_genome.

Returns: A logical.

Examples:

```
ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_non_discrete_genome.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$discrete_genome()
ts2 <- ts_load(ts_file2)
ts2$discrete_genome()
```

Method `has_reference_sequence()`**:** Get whether the tree sequence has a reference genome sequence.

Usage:

```
TreeSequence$has_reference_sequence()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.has_reference_sequence.

Returns: A logical.

Examples:

```
ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_with_ref_seq.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$has_reference_sequence()
ts2 <- ts_load(ts_file2)
ts2$has_reference_sequence()
```

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

Usage:

```
TreeSequence$time_units()
```

Details: See the `tskit` Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.time_units.

Returns: A character.

Examples:

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

Method `discrete_time()`**:** Get the discrete time status.

Usage:

```
TreeSequence$discrete_time()
```

Details: Returns TRUE if all time values in the tree sequence are discrete integer values.
See the tskit Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.discrete_time.

Returns: A logical.

Examples:

```
ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_discrete_time.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$discrete_time()
ts2 <- ts_load(ts_file2)
ts2$discrete_time()
```

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

Usage:

```
TreeSequence$min_time()
```

Details: See the tskit Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.min_time.

Returns: A numeric.

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()
```

Details: See the tskit Python equivalent at https://tskit.dev/tskit/docs/latest/python-api.html#tskit.TreeSequence.max_time.

Returns: A numeric.

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, each as a signed 64 bit integer `bit64::integer64`.

Examples:

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

Method `file_uuid()`**:** Get the UUID string of the file the tree sequence was loaded from.

Usage:

```
TreeSequence$file_uuid()
```

Returns: A character; `NA_character_` when file is information is unavailable.

Examples:

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

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$discrete_genome`
## ------------------------------------------------

ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_non_discrete_genome.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$discrete_genome()
ts2 <- ts_load(ts_file2)
ts2$discrete_genome()

## ------------------------------------------------
## Method `TreeSequence$has_reference_sequence`
## ------------------------------------------------

ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_with_ref_seq.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$has_reference_sequence()
ts2 <- ts_load(ts_file2)
ts2$has_reference_sequence()

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

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

## ------------------------------------------------
## Method `TreeSequence$discrete_time`
## ------------------------------------------------

ts_file1 <- system.file("examples/test.trees", package = "RcppTskit")
ts_file2 <- system.file("examples/test_discrete_time.trees", package = "RcppTskit")
ts1 <- ts_load(ts_file1)
ts1$discrete_time()
ts2 <- ts_load(ts_file2)
ts2$discrete_time()

## ------------------------------------------------
## 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()

## ------------------------------------------------
## Method `TreeSequence$file_uuid`
## ------------------------------------------------

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

`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 genome sequence information.

Details

See the `tskit` Python equivalent 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 temporary file on 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?

Details

Because this transfer is via a temporary file, the file UUID property changes.

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)
```

`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()
```

Index

check_tskit_py *(*get_tskit_py*)*, 2

get_tskit_py, 2, *13*, *24*

kastore_version, 3

TableCollection, 4, *5*, *20, 21*, *23, 24*
TableCollection$dump, *6*, *14*, *21*
TableCollection$new, *20*
TableCollection$r_to_py, *21*
tc_load, *14*, 19, *21*
tc_py_to_r, *14*, 20
tc_read *(*tc_load*)*, 19
TreeSequence, *6*, 21, *22, 23*, *35, 36*
TreeSequence$dump, *23*, *30*, *36*
TreeSequence$new, *35*
TreeSequence$r_to_py, *36*
ts_load, *22*, *30*, 34, *36*
ts_py_to_r, *30*, 36
ts_read *(*ts_load*)*, 34
tskit_version, 37