# Package 'rncl'

February 10, 2020

**Title** An Interface to the Nexus Class Library

**Version** 0.8.4

**Maintainer** Francois Michonneau <francois.michonneau@gmail.com>

**Description** An interface to the Nexus Class Library which allows parsing of NEXUS, Newick and other phylogenetic tree file formats. It provides elements of the file that can be used to build phylogenetic objects such as ape's 'phylo' or phylobase's 'phylo4(d)'. This functionality is demonstrated with 'read_newick_phylo()' and 'read_nexus_phylo()'.

**Imports** Rcpp (>= 0.11.0), progress (>= 1.1.2), stats

**Suggests** testthat, ape

**LinkingTo** Rcpp, progress

**Depends** R (>= 3.1.1)

**License** BSD_2_clause + file LICENSE

**URL** https://github.com/fmichonneau/rncl

**BugReports** https://github.com/fmichonneau/rncl/issues

**LazyData** true

**RoxygenNote** 7.0.2.9000

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** Francois Michonneau [aut, cre] (<https://orcid.org/0000-0002-9092-966X>), Ben Bolker [aut] (<https://orcid.org/0000-0002-2127-0443>), Mark Holder [aut] (<https://orcid.org/0000-0001-5575-0536>), Paul Lewis [aut] (<https://orcid.org/0000-0001-9852-8759>), Brian O'Meara [aut] (<https://orcid.org/0000-0002-0337-5997>)

**Repository** CRAN

**Date/Publication** 2020-02-10 05:10:03 UTC

# R topics documented:



---

`read_nexus_phylo` *Read phylogenetic trees from files*

---

## Description

Create phylo objects from NEXUS or Newick files

## Usage

```
read_nexus_phylo(
  file,
  simplify = FALSE,
  missing_edge_length = NA,
  show_progress = TRUE,
  ...
)

read_newick_phylo(
  file,
  simplify = FALSE,
  missing_edge_length = NA,
  show_progress = TRUE,
  ...
)

make_phylo(file, simplify = FALSE, missing_edge_length = NA, ...)
```

## Arguments

| | |
|---|---|
| `file` | Path of NEXUS or Newick file |
| `simplify` | If the file includes more than one tree, returns only the first tree; otherwise, returns a multiPhylo object |
| `missing_edge_length` | If the tree contains missing edge lengths, the value to be attributed to these edge lengths. By default, (`missing_edge_length = NA`) if at least edge length is missing, they are all removed. Otherwise, the value must be a single numeric value. In any case, a warning will be generated if the tree contains missing edge lengths. |
| `show_progress` | If `TRUE` (default)), a progress bar is displayed during the possibly time consuming step of removing the singletons from the tree. |
| `...` | additional parameters to be passed to the rncl function |

## Details

These functions read NEXUS or Newick files and return an object of class phylo/multiPhylo.

## Value

A phylo or a multiPhylo object

## Note

`make_phylo` will soon be deprecated, use `read_nexus_phylo` or `read_newick_phylo` instead.

## Author(s)

Francois Michonneau

## See Also

rncl-package

---

`rncl` *rncl: An R interface to the NEXUS Class Library*

---

## Description

rncl provides an interface to the NEXUS Class Library (NCL), a C++ library intended to parse valid NEXUS files as well as other common formats used in phylogenetic analysis. Currently, rncl focuses on parsing trees and supports both NEXUS and Newick formatted files. Because NCL is used by several phylogenetic software (e.g., MrBayes, Garli), rncl can parse files generated by these programs. However, other popular programs (including BEAST) use an extension of the NEXUS file format, and if trees can be imported, associated annotations (e.g., confidence intervals on the time since divergence) cannot.

Returns a list of the elements contained in a NEXUS file used to build phylogenetic objects in R

## Usage

```
rncl(
  file,
  file.format = c("nexus", "newick"),
  spacesAsUnderscores = TRUE,
  char.all = TRUE,
  polymorphic.convert = TRUE,
  levels.uniform = TRUE,
  show_progress = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| file | path to a NEXUS or Newick file |
| file.format | a character string indicating the type of file to be parsed. |
| spacesAsUnderscores | In the NEXUS file format white spaces are not allowed and are represented by underscores. Therefore, NCL converts underscores found in taxon labels in the NEXUS file into white spaces (e.g. species_1 will become "species 1"). If you want to preserve the underscores, set as TRUE (default). This option affects taxon labels, character labels and state labels. |
| char.all | If TRUE (default), returns all characters, even those excluded in the NEXUS file (only when NEXUS file contains DATA block). |
| polymorphic.convert | If TRUE (default), converts polymorphic characters to missing data (only when NEXUS file contains DATA block). |
| levels.uniform | If TRUE (default), uses the same levels for all characters (only when NEXUS file contains DATA block). |
| show_progress | If TRUE (default)), a progress bar is displayed during the possibly time consuming step of removing the singletons from the tree. |
| ... | additional parameters (currently not in use). |

## Details

NCL can also parse data associated with species included in NEXUS files. If you are interested in importing such data, see the phylobase package.

NEXUS is a common file format used in phylogenetics to represent phylogenetic trees, and other types of phylogenetic data. This function uses NCL (the NEXUS Class Library) to parse NEXUS, Newick or other common phylogenetic file formats, and returns the relevant elements as a list. phylo (from the ape package) or phylo4 (from the phylobase package) can be constructed from the elements contained in this list.

## Value

A list that contains the elements extracted from a NEXUS or a Newick file.

- taxaNames A vector of the taxa names listed in the TAXA block of the NEXUS file or inferred from the tree strings (if block missing or Newick file).
- treeNames A vector listing the names of the trees
- taxonLabelVector A list containing as many elements as there are trees in the file. Each element is a character vector that lists the taxon names encountered in the tree string *in the order they appear*, and therefore may not match the order they are listed in the translation table.
- parentVector A list containing as many elements as there are trees in the file. Each element is a numeric vector listing the parent node for the node given by its position in the vector. If the beginning of the vector is 5 5 6, the parent node of node 1 is 5, the parent of node 2 is 5 and the parent of node 3 is 6. The implicit root of the tree is identified with 0 (node without a parent).

- `branchLengthVector` A list containing as many elements as there are trees in the file. Each element is a numeric vector listing the edge/branch lengths for the edges in the same order as nodes are listed in the corresponding `parentVector` element. Values of -999 indicate that the value is missing for this particular edge. The implicit root as a length of 0.
- `nodeLabelsVector` A list containing as many elements as there are trees in the file. Each element is a character vector listing the node labels in the same order as the nodes are specified in the same order as nodes are listed in the corresponding `parentVector` element.
- `trees` A character vector listing the tree strings where tip labels have been replaced by their indices in the `taxaNames` vector. They do not correspond to the numbers listed in the translation table that might be associated with the tree.
- `dataTypes` A character vector indicating the type of data associated with the tree (e.g., "standard").
- `nbCharacters` A numeric vector indicating how many characters/traits are available.
- `charLabels` A character vector listing the names of the characters/traits that are available.
- `nbStates` A numeric vector listing the number of possible states for each character/trait.
- `stateLabels` A character vector listing in order, all possible states for each character/trait.
- `dataChr` A character vector with as many elements as there are characters/traits in the dataset. Each element is string that can be parsed by R to create a factor vector representing the data found in the file.
- `isRooted` A list with as many elements as there are trees in the file. Each element is a logical indicating whether the tree is rooted. NCL definition of a rooted tree differs from the one APE uses in some cases.
- `hasPolytomies` A list with as many elements as there are trees in the file. Each element is a logical indicating whether the tree contains polytomies.
- `hasSingletons` A list with as many elements as there are trees in the file. Each element is a logical indicating whether the tree contains singleton nodes, in other words nodes with a single descendant (also known as knuckles).

## Author(s)

Francois Michonneau

## References

Maddison DR, Swofford DL, Maddison WP (1997). "NEXUS: An extensible file format for systematic information". Systematic Biology 46(4) : 590-621. doi:10.1093/sysbio/46.4.590

Lewis, P. O. 2003. NCL: a C++ class library for interpreting data files in NEXUS format. Bioinformatics 19 (17) : 2330-2331.

## See Also

For examples on how to use the elements of the list returned by this function to build tree objects, inspect the source code of this package, in particular how `read_newick_phylo` and `read_nexus_phylo` work. For a more complex example that also use the data contained in NEXUS files, inspect the source code of the `readNCL` function in the phylobase package.

# Index