Package ‘RNeXML’

February 19, 2015

Type Package
Title Implement semantically rich I/O for the NeXML format
Version 2.0.1
Description Provides access to phyloinformatic data in NeXML format. The package should add new functionality to R such as the possibility to manipulate NeXML objects in more various and refined way and compatibility with 'ape' objects. Note that for the suggested packages, Sxslt is available through Omegahat.org and rdf from carlboettiger.info using the additional repository URLs given below. Alternatively these can be installed from Github, https://github.com/cboettig/Sxslt and https://github.com/egonw/rrdf/
URL https://github.com/ropensci/RNeXML
BugReports https://github.com/ropensci/RNeXML/issues
License BSD_3_clause + file LICENSE
Additional_repositories http://www.omegahat.org/R,
http://server.carlboettiger.info:5555/R
VignetteBuilder knitr
Suggests rdf (>= 2.0.2), geiger (>= 2.0), phytools (>= 0.3.93), knitr (>= 1.5), rfigshare (>= 0.3.0), knitrca (>= 1.0.1), testthat (>= 0.8.1), phylobase (>= 0.6.8), rmarkdown (>= 0.3.3), Sxslt (>= 0.91)
Depends R (>= 3.0.0), ape (>= 3.1), methods (>= 3.0.0)
Imports XML (>= 3.95), plyr (>= 1.8), taxize (>= 0.2.2), reshape2 (>= 1.2.2), httr (>= 0.3), uuid (>= 0.1-1)
Collate 'classes.R' 'add_basic_meta.R' 'add_characters.R' 'add_meta.R' 'add_namespaces.R' 'add_trees.R' 'character_classes.R' 'concatenate_nexml.R' 'get_basic_metadata.R' 'get_characters.R' 'get_metadata.R' 'get_namespaces.R' 'get_rdf.R' 'get_taxa.R' 'get_taxa_meta.R' 'get_trees.R' 'internal_get_node_maps.R' 'internal_isEmpty.R' 'internal_name_by_id.R' 'internal_nexml_id.R' 'meta.R' 'nexmlTree.R' 'nexml_add.R' 'nexml_get.R' 'nexml_methods.R' 'nexml_publish.R'
Author  Carl Boettiger [cre, aut],
        Scott Chamberlain [aut],
        Hilmar Lapp [aut],
        Ksenia Shumelchyk [aut],
        Rutger Vos [aut]

Maintainer  Carl Boettiger <cboettig@gmail.com>

NeedsCompilation  no

Repository  CRAN

Date/Publication  2014-12-26 08:35:18

R topics documented:

'add_basic_meta' 'add_characters' 'add_meta' 'add_namespaces' 'add_trees'
'c,ListOfmeta-method' 'c,meta-method' 'c,nexml-method'
'flatten_multiphylo' 'get_characters' 'get_characters_list'
'get_citation' 'get_flat_trees' 'get_license'
'get_metadata' 'get_namespaces' 'get_rdf'
'get_taxa' 'get_taxa_list' 'get_trees' 'get_trees_list'
'meta' 'nexml_add' 'nexml_figshare' 'nexml_get'
'nexml_publish' 'nexml_read' 'nexml_to_simmap'
'nexml_validate' 'nexml_write'
'reset_id_counter' 'simmap_ex'
add_basic_meta

Description

adds Dublin Core metadata elements to (top-level) nexml

Usage

add_basic_meta(title = NULL, description = NULL, creator = NULL,
    pubdate = Sys.Date(), rights = "CC0", publisher = NULL,
    citation = NULL, nexml = new("nexml"))

Arguments

title A title for the dataset
description a description of the dataset
creator name of the data creator. Can be a string or R person object
pubdate publication date. Default is current date.
rights the intellectual property rights associated with the data. The default is Creative Commons Zero (CC0) public domain declaration, compatible with all other licenses and appropriate for deposition into the Dryad or figshare repositories. CC0 is also recommended by the Panton Principles. Alternatively, any other plain text string can be added and will be provided as the content attribute to the dc:rights property.
publisher the publisher of the dataset. Usually where a user may go to find the canonical copy of the dataset: could be a repository, journal, or academic institution.
citation a citation associated with the data. Usually an accompanying academic journal article that indicates how the data should be cited in an academic context. Multiple citations can be included here. citation can be a plain text object, but is preferably an R 'citation' or 'bibentry' object (which can include multiple citations. See examples
nexml a nexml object to which metadata should be added. A new nexml object will be created if none exists.
add_characters

Details

add_basic_meta() is just a wrapper for add_meta to make it easy to provide generic metadata without explicitly providing the namespace. For instance, add_basic_meta(title="My title", description="a description") is identical to: add_meta(list(meta("dc:title", "My title"), meta("dc:description", "a description")))

Most function arguments are mapped directly to the Dublin Core terms of the same name, with the exception of 'rights', which by default maps to the Creative Commons namespace when using CC0 license.

Value

an updated nexml object

See Also

add_trees add_characters add_meta

Examples

```r
nex <- add_basic_meta(title = "My test title",
                      description = "A description of my test",
                      creator = "Carl Boettiger <cboettig@gmail.com>",
                      publisher = "unpublished data",
                      pubdate = "2012-04-01")

## Adding citation to an R package:
nexml <- add_basic_meta(citation="ape")

## Not run:
## Use knitr CiteDOI package to add a citation by DOI:
library(knitr)
nexml <- add_basic_meta(citation="10.2307/2408428")

## End(Not run)
```

---

## add_characters

### Add character data to a nexml object

**Description**

Add character data to a nexml object

**Usage**

```r
add_characters(x, nexml = new("nexml"), append_to_existing_otus = FALSE)
```
Arguments

- **x**: character data, in which character traits labels are column names and taxon labels are row names. x can be in matrix or data.frame format.
- **nexml**: a nexml object, if appending character table to an existing nexml object. If omitted will initiate a new nexml object.
- **append_to_existing_otas**: logical. If TRUE, will add any new taxa (taxa not matching any existing otus block) to the existing (first) otus block. Otherwise (default), a new otus block is created, even though it may contain duplicate taxa to those already present. While FALSE is the safe option, TRUE may be appropriate when building nexml files from scratch with both characters and trees.

Examples

```r
library("geiger")
data(geospiza)
geiger_nex <- add_characters(geospiza$dat)
```

---

**add_meta**

*Add metadata to a nexml file*

Description

Add metadata to a nexml file

Usage

```r
add_meta(meta, nexml = new("nexml"), level = c("nexml", "otus", "trees", "characters"), namespaces = NULL, i = 1, at_id = NULL)
```

Arguments

- **meta**: a meta S4 object, e.g. ouput of the function *meta*, or a list of these meta objects
- **nexml**: (S4) object
- **level**: the level at which the metadata annotation should be added.
- **namespaces**: named character string for any additional namespaces that should be defined.
- **i**: for otus, trees, characters: if there are multiple such blocks, which one should be annotated? Default is first/only block.
- **at_id**: the id of the element to be annotated. Optional, advanced use only.

Value

the updated nexml object
add_namespaces

See Also

meta add_trees add_characters add_basic_meta

Examples

## Create a new nexml object with a single metadata element:
modified <- meta(property = "prism:modificationDate", content = "2013-10-04")
nex <- add_meta(modified) # Note: 'prism' is defined in nexml_namespaces by default.

## Write multiple metadata elements, including a new namespace:
website <- meta(href = "http://carlboettiger.info",
                rel = "foaf:homepage") # meta can be link-style metadata
nex <- add_meta(list(modified, website),
                namespaces = c(foaf = "http://xmlns.com/foaf/0.1/"))

## Append more metadata, and specify a level:
history <- meta(property = "skos:historyNote",
                content = "Mapped from the bird.orders data in the ape package using RNeXML")
nex <- add_meta(history,
                nexml = nex,
                level = "trees",
                namespaces = c(skos = "http://www.w3.org/2004/02/skos/core"))

Description

add namespaces, avoiding duplication if prefix is already defined

Usage

add_namespaces(namespaces, nexml = new("nexml"))

Arguments

namespaces a named character vector of namespaces
nexml a nexml object. will create a new one if none is given.

Value

a nexml object with updated namespaces

See Also

meta add_meta
Examples

```r
## Create a new nexml object with a single metadata element:
modified <- meta(property = "prism:modificationDate", content = "2013-10-04")
nex <- add_meta(modified)  # Note: 'prism' is defined in nexml_namespaces by default.

## Write multiple metadata elements, including a new namespace:
website <- meta(href = "http://carlboettiger.info",
                   rel = "foaf:homepage")  # meta can be link-style metadata
nex <- add_meta(list(modified, website),
                 namespaces = c(foaf = "http://xmlns.com/foaf/0.1/"))

## Append more metadata, and specify a level:
history <- meta(property = "skos:historyNote",
                 content = "Mapped from the bird.orders data in the ape package using RNeXML")
nex <- add_meta(history,
               nexml = nex,
               level = "trees",
               namespaces = c(skos = "http://www.w3.org/2004/02/skos/core#"))
```

---

Description

`add_trees`

Usage

```
add_trees(phy, nexml = new("nexml"), append_to_existing_otus = FALSE)
```

Arguments

- `phy` a phylo object, multiPhylo object, or list of mulitPhylo to be added to the nexml
- `nexml` a nexml object to which we should append this phylo. By default, a new nexml object will be created.
- `append_to_existing_otus` logical, indicating if we should make a new OTU block (default) or append to the existing one.

Value

a nexml object containing the phy in nexml format.

Examples

```
library("geiger")
data(geospiza)
geiger_nex <- add_trees(geospiza$phy)
```
c,ListOfmeta-method  
**Concatenate ListOfmeta elements into a ListOfmeta**

**Description**
Concatenate ListOfmeta elements into a ListOfmeta

**Usage**
```
# S4 method for signature 'ListOfmeta'
c(x, ..., recursive = FALSE)
```

**Arguments**
- `x, ...`: meta or ListOfmeta elements to be concatenated, e.g. see `meta`
- `recursive`: logical, if 'recursive=TRUE', the function descends through lists and combines their elements into a vector.

**Value**
a `ListOfmeta` object containing multiple meta elements.

**Examples**
```r
metalist <- c(meta(content="example", property="dc:title"),
               meta(content="Carl", property="dc:creator"))
out <- c(metalist, metalist)
out <- c(metalist, meta(content="a", property="b"))
```

---

c,meta-method  
**Concatenate meta elements into a ListOfmeta**

**Description**
Concatenate meta elements into a ListOfmeta

**Usage**
```
# S4 method for signature 'meta'
c(x, ..., recursive = FALSE)
```

**Arguments**
- `x, ...`: meta elements to be concatenated, e.g. see `meta`
- `recursive`: logical, if 'recursive=TRUE', the function descends through lists and combines their elements into a vector.
Value

a `listOfmeta` object containing multiple `meta` elements.

Examples

c(meta(content="example", property="dc:title"),
   meta(content="Carl", property="dc:creator"))

---

c.nexml-method

**Concatenate nexml files**

Description

Concatenate nexml files

Usage

```r
## S4 method for signature 'nexml'
c(x, ..., recursive = FALSE)
```

Arguments

- `x, ...`: nexml objects to be concatenated, e.g. from `write.nexml` or `read.nexml`. Must have unique ids on all elements.
- `recursive`: logical. If `recursive = TRUE`, the function recursively descends through lists (and pairlists) combining all their elements into a vector. (Not implemented).

Value

a concatenated nexml file

Examples

```r
## Not run:
f1 <- system.file("examples", "trees.xml", package="RNeXML")
f2 <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex1 <- read.nexml(f1)
nex2 <- read.nexml(f2)
nex <- c(nex1, nex2)

## End(Not run)
```
**get_characters**

Get character data.frame, accepts either nexml object, or a list of data.frames

**Usage**

get_characters(input, suffixes = FALSE, rownames_as_col = FALSE)

**Arguments**

- **input**: A nexml object (e.g., as output from `read.nexml`), or a list of data.frame's (e.g., as output from `get_characters_list`)
- **suffixes**: Add list element names as suffixes to output data.frame column names.
- **rownames_as_col**: Option to return character matrix rownames (with taxon ids) as its own column in the data.frame. Default is FALSE for compatibility with geiger and similar packages.

**flatten_multiphylo**

Flatten a multiphylo object

**Description**

Flatten a multiphylo object

**Usage**

flatten_multiphylo(object)

**Arguments**

- **object**: a list of multiphylo objects

**Details**

NeXML has the concept of multiple <trees> nodes, each with multiple child <tree> nodes. This maps naturally to a list of multiphylo objects. Sometimes this hierarchy conveys important structural information, so it is not discarded by default. Occasionally it is useful to flatten the structure though, hence this function. Note that this discards the original structure, and the nexml file must be parsed again to recover it.
get_characters_list

Examples

## Not run:
# A simple example with a discrete and a continuous trait
f <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- read.nexml(f)
get_characters(nex)

# with different row names
char_list <- get_characters_list(nex)
row.names(char_list[[1]])[1:3] <- c("taxon_18", "taxon_20", "taxon_30")
get_characters(char_list)

# A more complex example -- currently ignores sequence-type characters
f <- system.file("examples", "characters.xml", package="RNeXML")
nex <- read.nexml(f)
get_characters(nex)

## End(Not run)

get_characters_list  Extract the character matrix

Description

Extract the character matrix

Usage

get_characters_list(nexml, rownames_as_col = FALSE)

Arguments

nexml  nexml object (e.g. from read.nexml)
rownames_as_col  option to return character matrix rownames (with taxon ids) as its own column in the data.frame. Default is FALSE for compatibility with geiger and similar packages.

Value

the list of taxa

Examples

comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
get_characters_list(nex)
get_citation

Description
get_citation

Usage
get_citation(nexml)

Arguments
nexml a nexml object

Value
the list of taxa

get_flat_trees

Description
extract a single multiPhylo object containing all trees in the nexml

Usage
get_flat_trees(nexml)

Arguments
nexml a representation of the nexml object from which the data is to be retrieved

Details
Note that this method collapses any heirarchical structure that may have been present as multiple ‘trees’ nodes in the original nexml (though such a feature is rarely used). To preserve that structure, use get_trees instead.

Value
a multiPhylo object (list of ape::phylo objects). See details.

See Also
get_trees get_trees get_item
Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
get_flat_trees(nex)
```

Description

get_license

Usage

```r
get_license(nexml)
```

Arguments

- `nexml`: a `nexml` object

Value

the list of taxa

Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
get_metadata(nex)
```
get_namespaces

Description
get namespaces

Usage
get_namespaces(nexml)

Arguments
nexml a nexml object

Value
a named character vector providing the URLs defining each of the namespaces used in the nexml file. Names correspond to the prefix abbreviations of the namespaces.

Examples
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
ex <- nexml_read(comp_analysis)
get_namespaces(ex)

get_rdf

Extract rdf-xml from a NeXML file

Description
Extract rdf-xml from a NeXML file

Usage
get_rdf(file)

Arguments
file the name of a nexml file, or otherwise a nexml object.

Value
an RDF-XML object (XMLInternalDocument). This can be manipulated with tools from the XML R package, or converted into a triplestore for use with SPARQL queries from the rrdf R package.
get_taxa

Examples

## Not run:
```r
f <- system.file("examples", "meta_example.xml", package="RNeXML")
rdf <- get_rdf(f)

## Write to a file and read in with rrdf
tmp <- tempfile()
saveXML(rdf, tmp)
library(rrdf)
lib <- load.rdf(tmp)

## Perform a SPARQL query:

## End(Not run)
```

---

get_taxa  get_taxa

Description

Retrieve names of all species/otus (operational taxonomic units) included in the nexml

Usage

```r
get_taxa(nexml)
```

Arguments

- **nexml**
  
a nexml object

Value

the list of taxa

See Also

- `get_item`

Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
get_taxa(nex)
```
get_taxa_list

Description
Retrieve names of all species/otus (operational taxonomic units) included in the nexml

Usage
get_taxa_list(nexml)

Arguments
nexml a nexml object

Value
the list of taxa

See Also
get_item

get_trees
extract a phylogenetic tree from the nexml

Description
extract a phylogenetic tree from the nexml

Usage
get_trees(nexml)

Arguments
nexml a representation of the nexml object from which the data is to be retrieved

Value
an ape::phylo tree, if only one tree is represented. Otherwise returns a list of lists of multiphylo trees. To consistently receive the list of lists format (preserving the hierarchal nature of the nexml), use get_trees_list instead.

See Also
get_trees get_flat_trees get_item
get_trees_list

Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nxml_read(comp_analysis)
get_trees(nex)
```

---

get_trees_list  
*extract all phylogenetic trees in ape format*

Description

extract all phylogenetic trees in ape format

Usage

```r
get_trees_list(nexml)
```

Arguments

- `nexml`  
a representation of the nxml object from which the data is to be retrieved

Value

returns a list of lists of multiphylo trees, even if all trees are in the same 'trees' node (and hence the outer list will be of length 1) or if there is only a single tree (and hence the inner list will also be of length 1. This guarantees a consistent return type regardless of the number of trees present in the nxml file, and also preserves any heirarchy/grouping of trees.

See Also

- `get_trees`
- `get_flat_trees`
- `get_item`

Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nxml_read(comp_analysis)
get_trees_list(nex)
```
Description

Constructor function for metadata nodes

Usage

```r
text(h) -> character(h)

text(h) <- character(h)
```

Arguments

- `property`: specify the ontological definition together with its namespace, e.g. `dc:title`
- `content`: content of the metadata field
- `rel`: Ontological definition of the reference provided in `href`
- `href`: A link to some reference
- `datatype`: optional RDFa field
- `id`: optional id element (otherwise id will be automatically generated).
- `type`: optional xsi:type. If not given, will use either "LiteralMeta" or "ResourceMeta" as determined by the presence of either a property or a href value.
- `children`: Optional element containing any valid XML block (XMLInternalElementNode class, see the XML package for details).

Details

User must either provide `property+content` or `rel+href`. Mixing these will result in potential garbage. The datatype attribute will be detected automatically from the class of the content argument. Maps from R class to schema datatypes are as follows: character - xs:string, Date - xs:date, integer - xs:integer, numeric - xs:decimal, logical - xs:boolean

See Also

- `nexml_write`

Examples

```r
meta(content="example", property="dc:title")
```
**nexml_add**

*Description*

add elements to a new or existing nexml object

**Usage**

```r
nexml_add(x, nexml = new("nexml"), type = c("trees", "characters", "meta", "namespaces"), ...)
```

**Arguments**

- `x`: the object to be added
- `nexml`: an existing nexml object onto which the object should be appended
- `type`: the type of object being provided.
- `...`: additional optional arguments to the add functions

**Value**

a nexml object with the additional data

**See Also**

`add_trees`, `add_characters`, `add_meta`, `add_namespaces`

**Examples**

```r
library("geiger")
data(geospiza)
geiger_nex <- nexml_add(geospiza$phy, type="trees")
geiger_nex <- nexml_add(geospiza$dat, nexml = geiger_nex, type="characters")
```

---

**nexml_figshare**

*Description*

publish nexml to figshare

**Usage**

```r
nexml_figshare(nexml, file = "nexml.xml",
categories = "Evolutionary Biology", tags = list("phylogeny", "NeXML"),
visibility = c("public", "private", "draft"), id = NULL, ...)
```
nexml_get

Get the desired element from the nexml object

Description

Get the desired element from the nexml object

Usage

nexml_get(nexml, element = c("trees", "trees_list", "flat_trees", "metadata", "otu", "taxa", "characters", "characters_list", "namespaces"), ...)

Arguments

nexml a nexml object (from read_nexml)
element the kind of object desired, see details.
... additional arguments, if applicable to certain elements

Examples

## Not run:
data(bird.orders)
birds <- add_trees(bird.orders)
doi <- nexml_get(birds, visibility = "public", repository="figshare")

## End(Not run)
Details

- "tree" an ape::phylo tree, if only one tree is represented. Otherwise returns a list of lists of multiphylo trees. To consistently receive the list of lists format (preserving the hierarchical nature of the nexml), use trees instead.

- "trees" returns a list of lists of multiphylo trees, even if all trees are in the same ‘trees’ node (and hence the outer list will be of length 1) or if there is only a single tree (and hence the inner list will also be of length 1. This guarantees a consistent return type regardless of the number of trees present in the nexml file, and also preserves any hierarchy/grouping of trees.

- "flat_trees" a multiPhylo object (list of ape::phylo objects) Note that this method collapses any hierarchical structure that may have been present as multiple ‘trees’ nodes in the original nexml (though such a feature is rarely used). To preserve that structure, use ‘trees’ instead.

- "metadata" Get metadata from the specified level (default is top/nexml level)

- "otu" returns a named character vector containing all available metadata. names indicate property (or rel in the case of links/resourceMeta), while values indicate the content (or href for links).

- "taxa" alias for otu

For a slightly cleaner interface, each of these elements is also defined as an S4 method for a nexml object. So in place of ‘get_item(nexml, "tree")’, one could use ‘get_tree(nexml)’, and so forth for each element type.

Value

return type depends on the element requested. See details.

See Also

get_trees

Examples

```r
comp_analysis <- system.file("examples", "comp_analysis.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
nexml_get(nex, "trees")
nexml_get(nex, "characters_list")
```

Description

Publish nexml files to the web and receive a DOI

Usage

```
nexml_publish(nexml, ..., repository = "figshare")
```
nexml_read

Arguments

nexml: a nexml object (or file path)
... additional arguments, depending on repository. See examples.
repository: destination repository

Value

a digital object identifier to the published data

Examples

```r
## Not run:
data(bird.orders)
birds <- add_trees(bird.orders)
doi <- nexml_publish(birds, visibility = "public", repository="figshare")
## End(Not run)
```

Description

Read NeXML files into various R formats

Usage

`nexml_read(x, ...)`

Arguments

x: Path to the file to be read in. Or an XMLInternalDoc or XMLInternalNode
... Further arguments passed on to XML::xmlParse

Examples

```r
f <- system.file("examples", "trees.xml", package="RNeXML")
nexml_read(f)
```
### nexml_to_simmap

**Description**

nexml_to_simmap

**Usage**

nexml_to_simmap(nexml)

**Arguments**

- **nexml**: a nexml object

**Value**

a simmap object (phylo object with a $maps element for use in phytools functions).

**Examples**

```r
data(simmap_ex)
phy <- nexml_to_simmap(simmap_ex)
nex <- simmap_to_nexml(phy)
```

### nexml_validate

**validate nexml using the online validator tool**

**Description**

validate nexml using the online validator tool

**Usage**

nexml_validate(file, schema = "http://www.nexml.org/2009/nexml.xsd")

**Arguments**

- **file**: path to the nexml file to validate
- **schema**: URL of schema (for fallback method only, set by default).

**Details**

Requires an internet connection. see http://www.nexml.org/nexml/phylows/validator for more information in debugging invalid files
Value
TRUE if the file is valid, FALSE or error message otherwise

Examples
## Not run:
data(bird.orders)
birds <- nexml_write(bird.orders, "birds_orders.xml")
nexml_validate("bird_orders.xml")
unlink("bird_orders.xml") # delete file to clean up
## End(Not run)

nexml_write  Write nexml files

Description
Write nexml files

Usage
nexml_write(x = new("nexml"), file = NULL, trees = NULL,
characters = NULL, meta = NULL, ...)

Arguments
x a nexml object, or any phylogeny object (e.g. phylo, phylo4) that can be coerced into one. Can also be omitted, in which case a new nexml object will be constructed with the additional parameters specified.
file the name of the file to write out
trees phylogenetic trees to add to the nexml file (if not already given in x) see add_trees for details.
characters additional characters
meta A meta element or list of meta elements, see add_meta
...
additional arguments to add__basic_meta, such as the title. See add_basic_meta.

Value
Writes out a nexml file

See Also
add_trees add_characters add_meta nexml_read
Examples

## Write an ape tree to nexml, analgous to write.nexus:
library(ape); data(bird.orders)
write.nexml(bird.orders, file="example.xml")

## Assemble a nexml section by section and then write to file:
library(geiger)
data(geospiza)
nexml <- add_trees(geospiza$phy) # creates new nexml
nexml <- add_characters(geospiza$dat, nexml = nexml) # pass the nexml obj to append character data
nexml <- add_basic_meta(title="my title", creator = "Carl Boettiger", nexml = nexml)
nexml <- add_meta(meta("prism:modificationDate", format(Sys.Date())), nexml = nexml)

## Not run: # takes > 5s
write.nexml(nexml, file="example.xml")

## As above, but in one call (except for add_meta() call).
write.nexml(trees = geospiza$phy,
characters = geospiza$dat,
title = "My title",
creator = "Carl Boettiger",
file = "example.xml")

## Mix and match: identical to the section by section:
nexml <- add_meta(meta("prism:modificationDate", format(Sys.Date())))
write.nexml(x = nexml,
trees = geospiza$phy,
characters = geospiza$dat,
title = "My title",
creator = "Carl Boettiger",
file = "example.xml")

## End(Not run)

reset_id_counter

reset id counter

Description
reset the id counter

Usage
reset_id_counter()
**simmap_to_nexml**

**Description**

`A nexml class R object that includes simmap annotations`

**Usage**

`simmap_ex`

**Format**

a `nexml` instance

**Author(s)**

Carl Boettiger

**Source**


---

**simmap_to_nexml**

**Description**

`simmap_to_nexml`

**Usage**

`simmap_to_nexml(phy, state_ids = NULL)`

**Arguments**

- `phy` a phy object containing simmap phy$maps element, from the phytools package
- `state_ids` a named character vector giving the state names corresponding to the ids used to refer to each state in `nexml`. If null ids will be generated and states taken from the phy$states names.

**Value**

a `nexml` representation of the simmap
**taxize_nexml**

**Examples**

```r
data(simmap_ex)
phy <- nxml_to_simmap(simmap_ex)
nex <- simmap_to_nxml(phy)
```

```r
taxize_nexml(nexml, type = c("NCBI"), ...)
```

**Arguments**

- `nexml`: a nxml object
- `type`: the name of the identifier to use
- `...`: additional arguments (not implemented yet)

**Examples**

```r
## Not run:
data(bird.orders)
birds <- add_trees(bird.orders)
birds <- taxize_nexml(birds, "NCBI")
```

**tophylo**

**Description**

nexml to phylo coercion

**Usage**

`tophylo(tree, otus)`
toPhylo

Arguments

- **tree**: an nexml tree element
- **otus**: a character string of taxonomic labels, named by the otu ids. e.g. (from get_otu_maps for the otus set matching the relevant trees node.

Value

phylo object. If a "reconstructions" annotation is found on the edges, return simmap maps slot as well.
Index

add_basic_meta, 3, 6, 24
add_characters, 4, 4, 6, 19, 24
add_meta, 4, 5, 6, 19, 24
add_namespaces, 6, 19
add_trees, 4, 6, 7, 19, 24

c, ListOfmeta-method, 8
c, meta-method, 8
c, nexml-method, 9

flatten_multiphylo, 10

get_characters, 10
get_characters_list, 10, 11
get_citation, 12
get_flat_trees, 12, 16, 17
get_item, 12, 15–17
get_item (nexml_get), 20
get_license, 13
get_metadata, 13
get_namespaces, 14
get_otu (get_taxa), 15
get_otus_list (get_taxa_list), 16
get_rdf, 14
get_taxa, 15
get_taxa_list, 16
get_trees, 12, 16, 16, 17, 21
get_trees_list, 16, 17

meta, 5, 6, 8, 18

nexml_add, 19
nexml_figshare, 19
nexml_get, 20
nexml_publish, 21
nexml_read, 22, 24
nexml_to_simmap, 23
nexml_validate, 23
nexml_write, 18, 24

read.nexml, 9, 10

read.nexml (nexml_read), 22
reset_id_counter, 25

simmap_ex, 26
simmap_to_nexml, 26
taxize_nexml, 27
tophylo, 27

write.nexml, 9
write.nexml (nexml_write), 24