Package 'RNeXML'

May 7, 2018

Type Package

Title Semantically Rich I/O for the 'NeXML' Format

Version 2.1.1

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.

URL https://github.com/ropensci/RNeXML

BugReports https://github.com/ropensci/RNeXML/issues

License BSD_3_clause + file LICENSE

VignetteBuilder knitr

Suggests rdflib, geiger (>= 2.0), phytools (>= 0.3.93), knitr (>= 1.5), rfigshare (>= 0.3.0), knitcitations (>= 1.0.1), testthat (>= 0.10.0), phylobase (>= 0.6.8), rmarkdown (>= 0.3.3), xslt, taxize (>= 0.2.2), covr

Depends R (>= 3.0.0), ape (>= 3.1), methods (>= 3.0.0)

Imports XML (>= 3.95), plyr (>= 1.8), reshape2 (>= 1.2.2), httr (>= 0.3), uuid (>= 0.1-1), dplyr (>= 0.5.0), lazyeval (>= 0.1.0), tidyr (>= 0.3.1), stringr (>= 1.0), xml2

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' 'deprecated.R' 'get_basic_metadata.R' 'get_characters.R' 'get_level.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' 'nexml_read.R' 'nexml_validate.R' 'nexml_write.R' 'simmap.R' 'taxize_nexml.R' 'tbl_df.R'

RoxygenNote 6.0.1.9000

X-schema.org-applicationCategory Data Publication

X-schema.org-keywords metadata, nexml, phylogenetics, linked-data

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Carl Boettiger [cre, aut] (<https://orcid.org/0000-0002-1642-628X>),
Scott Chamberlain [aut] (<https://orcid.org/0000-0003-1444-9135>),
Hilmar Lapp [aut] (<https://orcid.org/0000-0001-9107-0714>),
Kseniia Shumelchyk [aut],
Rutger Vos [aut] (<https://orcid.org/0000-0001-9254-7318>)

Maintainer Carl Boettiger <cboettig@gmail.com>

Repository CRAN

Date/Publication 2018-05-07 07:55:04 UTC

R topics documented:

`add_basic_meta` *Add basic metadata*

Description

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

Usage

```
add_basic_meta(title = NULL, description = NULL,
  creator = Sys.getenv("USER"), 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, compatiable 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 acompanying 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.

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 descri 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

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

## End(Not run)
```

---

add_characters *Add character data to a nexml object*

Description

Add character data to a nexml object

Usage

```
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 ommitted will initiate a new nexml object.
append_to_existing_otus	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

```
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

```
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

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#"))
```

---

add_namespaces *add namespaces*

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

```
## 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#"))
```

`add_trees` *add_trees*

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

```
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

```
## 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

```
## 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)
```

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 heirarchy conveys important structural information, so it is not discarded by default. Occassionally 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 *Get character data.frame from nexml*

Description

Get character data.frame from nexml

Usage

```
get_characters(nex, rownames_as_col = FALSE, otu_id = FALSE,
  otus_id = FALSE)
```

Arguments

nex	a nexml object
rownames_as_col	option to return character matrix rownames (with taxon ids) as it's own column in the data.frame. Default is FALSE for compatibility with geiger and similar packages.
otu_id	logical, default FALSE. return a column with the otu id (for joining with otu metadata, etc)
otus_id	logical, default FALSE. return a column with the otus block id (for joining with otu metadata, etc)

Details

RNeXML will attempt to return the matrix using the NeXML taxon (otu) labels to name the rows and the NeXML char labels to name the traits (columns). If these are unavailable or not unique, the NeXML id values for the otus or traits will be used instead.

Value

the character matrix as a data.frame

Examples

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

# 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 it's 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` *get_citation*

Description

get_citation

Usage

```
get_citation(nexml)
```

Arguments

`nexml`	a nexml object

Value

the list of taxa

`get_flat_trees` *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 heirachical 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

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

---

get_level *get_level*

Description

get a data.frame of attribute values of a given node

Usage

```
get_level(nex, level)
```

Arguments

nex	a nexml object
level	a character vector indicating the class of node, see details

Details

level should be a character vector giving the path to the specified node group. For instance, ‘otus‘, ‘characters‘, and ‘trees‘ are top-level blocks (e.g. child nodes of the root nexml block), and can be specified directly. To get metadata for all "char" elements from all characters blocks, you must specify that ‘char‘ nodes are child nodes to ‘character‘ nodes: e.g. ‘get_level(nex, "characters/char")‘, or similarly for states: ‘get_level(nex, characters/states)‘.

The return object is a data frame whose columns are the attribute names of the elements specified. The column names match the attribute names except for "id" attribute, for which the column is renamed using the node itself. (Thus <otus id="os2"> would be rendered in a data.frame with column called "otus" instead of "id"). Additional columns are added for each parent element in the path; e.g. get_level(nex, "otus/otu") would include a column named "otus" with the id of each otus block. Even though the method always returns the data frame for all matching nodes in all blocks, these ids let you see which otu values came from which otus block. This is identical to the function call ‘get_taxa()‘. Similarly, ‘get_level(nex, "otus/otu/meta")‘ would return additional columns ’otus’ and also a column, ’otu’, with the otu parent ids of each metadata block. (This is identical to a function call to ‘get_metadata‘). This makes it easier to join data.frames as well, see examples

Value

Returns the attributes of specified class of nodes as a data.frame

---

`get_license` *get_license*

Description

get_license

Usage

```
get_license(nexml)
```

Arguments

`nexml`	a nexml object

Value

the list of taxa

`get_metadata` *get_metadata*

Description

get_metadata

Usage

```
get_metadata(nexml, level = "nexml")
```

Arguments

`nexml`	a nexml object
`level`	the name of the level of element desired, see details

Details

'level' should be either the name of a child element of a NeXML document (e.g. "otu", "characters"), or a path to the desired element, e.g. 'trees/tree' will return the metadata for all phylogenies in all trees blocks.

Value

the requested metadata as a data.frame. Additional columns indicate tha parent element of the return value.

Examples

```
## Not run:
comp_analysis <- system.file("examples", "primates.xml", package="RNeXML")
nex <- nexml_read(comp_analysis)
get_metadata(nex)
get_metadata(nex, "otus/otu")

## End(Not run)
```

`get_namespaces` *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")
nex <- nexml_read(comp_analysis)
get_namespaces(nex)
```

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 rdflib R package.

get_taxa *get_taxa*

Description

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

Usage

```
get_taxa(nexml)
```

Arguments

nexml	a nexml object

Value

the list of taxa

See Also

get_item

Examples

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

get_taxa_list *get_taxa_list*

Description

Retrieve names of all species/otus 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 recieve the list of lists format (preserving the heriarchical nature of the nexml), use get_trees_list instead.

See Also

get_trees get_flat_trees get_item

Examples

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

```
get_trees_list(nexml)
```

Arguments

`nexml`	a representation of the nexml 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 guarentees a consistent return type regardless of the number of trees present in the nexml file, and also preserves any heirarchy/grouping of trees.

See Also

`get_trees` `get_flat_trees` `get_item`

Examples

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

`meta` *Constructor function for metadata nodes*

Description

Constructor function for metadata nodes

Usage

```
meta(property = character(0), content = character(0), rel = character(0),
  href = character(0), datatype = character(0), id = character(0),
  type = character(0), children = list())
```

Arguments

`property`	specify the ontological definition together with it's 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

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

nexml_add *add elements to a new or existing nexml object*

Description

add elements to a new or existing nexml object

Usage

```
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

```
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 *publish nexml to figshare*

Description

publish nexml to figshare

Usage

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

Arguments

nexml	a nexml object (or file path to a nexml file)
file	The filename desired for the object, if nexml is not already a file. if the first argument is already a path, this value is ignored.
categories	The figshare categories, must match available set. see fs_add_categories
tags	Any keyword tags you want to add to the data.
visibility	whether the results should be published (public), or kept private, or kept as a draft for further editing before publication. (New versions can be updated, but any former versions that was once made public will always be archived and cannot be removed).
id	an existing figshare id (e.g. from fs_create), to which this file can be appended.
...	additional arguments

Value

the figshare id of the object

Examples

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

## End(Not run)
```

---

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

Details

- "tree" an ape::phylo tree, if only one tree is represented. Otherwise returns a list of lists of multiphylo trees. To consistently recieve the list of lists format (preserving the heriarchical 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 guarentees a consistent return type regardless of the number of trees present in the nexml file, and also preserves any heirarchy/grouping of trees.
- "flat_trees" a multiPhylo object (list of ape::phylo objects) Note that this method collapses any heirachical 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

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

---

nexml_publish *publish nexml files to the web and receive a DOI*

Description

publish nexml files to the web and receive a DOI

Usage

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

Arguments

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

Value

a digital object identifier to the published data

Examples

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

## End(Not run)
```

---

nexml_read *Read NeXML files into various R formats*

Description

Read NeXML files into various R formats

Usage

```
nexml_read(x, ...)

## S3 method for class 'character'
nexml_read(x, ...)

## S3 method for class 'XMLInternalDocument'
nexml_read(x, ...)

## S3 method for class 'XMLInternalNode'
nexml_read(x, ...)
```

Arguments

x	Path to the file to be read in. Or an XMLInternalDocument-class or XMLInternalNode-class
...	Further arguments passed on to xmlParse

Examples

```
# file
f <- system.file("examples", "trees.xml", package="RNeXML")
nexml_read(f)
## Not run:  # may take > 5 s
# url
url <- "https://raw.githubusercontent.com/ropensci/RNeXML/master/inst/examples/trees.xml"
nexml_read(url)
# character string of XML
str <- paste0(readLines(f), collapse = "")
nexml_read(str)
# XMLInternalDocument
library("httr")
library("XML")
x <- xmlParse(content(GET(url)))
nexml_read(x)
# XMLInternalNode
nexml_read(xmlRoot(x))

## End(Not run)
```

`nexml_to_simmap` *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

```
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 = CANONICAL_SCHEMA)
```

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("birds_orders.xml")
unlink("birds_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")

## Not run:  # takes > 5s
## 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)

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_ex` *A nexml class R object that includes simmap annotations*

Description

A nexml object with simmap stochastic character mapping annotations added to the edges, for use with the RNeXML package parsing and serializing NeXML into formats that work with the ape and phytools packages.

Usage

```
simmap_ex
```

Format

a `nexml` instance

Author(s)

Carl Boettiger

Source

Simulated tree and stochastic character mapping based on Revell 2011 (doi:10.1111/j.2041-210X.2011.00169.x)

simmap_to_nexml *simmap_to_nexml*

Description

simmap_to_nexml

Usage

```
simmap_to_nexml(phy, state_ids = NULL)
```

Arguments

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

Value

a nexml representation of the simmap

Examples

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

taxize_nexml *taxize nexml*

Description

Check taxanomic names against the specified service and add appropriate semantic metadata to the nexml OTU unit containing the corresponding identifier.

Usage

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

Arguments

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

Examples

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

## End(Not run)
```

`toPhylo` *nexml to phylo*

Description

nexml to phylo coercion

Usage

```
toPhylo(tree, otus)
```

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