Package: taxlist (via r-universe)

October 16, 2024

Version 0.3.1

Encoding UTF-8

Title Handling Taxonomic Lists

Depends R(>= 4.3)

Imports biblio (>= 0.0.8), foreign, methods, stats, stringdist, stringi, stringr, tools, utils, vegdata

Suggests ape, knitr, rmarkdown, taxa, testthat

LazyData true

Description Handling taxonomic lists through objects of class 'taxlist'. This package provides functions to import species lists from 'Turboveg' (<https://www.synbiosys.alterra.nl/turboveg/>) and the possibility to create backups from resulting R-objects. Also quick displays are implemented as summary-methods.

License GPL (>= 2)

Roxygen list(markdown = TRUE)

URL https://cran.r-project.org/package=taxlist, https://github.com/ropensci/taxlist, https://docs.ropensci.org/taxlist/

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

Collate 'imports.R' 'internal.R' 'levels.R' 'prune_levels.R' 'deprecated-functions.R' 'id_generator.R' 'replace_x.R' 'reindex.R' 'insert_rows.R' 'dissect_name.R' 'clean_strings.R' 'taxlist-class.R' 'matched_names-class.R' 'clean.R' 'coerce-methods.R' 'taxon_views.R' 'count_taxa.R' 'taxon_names.R' 'taxon_relations.R' 'add_concept.R' 'taxon_traits.R' 'accepted_name.R' 'get_children.R' 'merge_to_parent.R' 'merge_taxa.R' 'Extract.R' 'subset.R' 'backup_object.R' 'summary.R' 'df2taxlist.R' 'tv2taxlist.R' 'tax2traits.R' 'match_names.R' 'print_name.R' 'indented_list.R' 'Easplist-data.R' 'taxlist-package.R' 'taxlist2df.R' 'parents.R'

VignetteBuilder knitr

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/ropensci/taxlist

RemoteRef main

RemoteSha bc3085cfa584bb59e4c7dd1dec1aadc356fdc1a0

Contents

accepted_name . . . 3
add_concept . . . 5
as . . . 6
backup_object . . . 7
clean . . . 9
clean_strings . . . 10
count_taxa . . . 11
Deprecated-functions . . . 12
df2taxlist . . . 13
dissect_name . . . 14
Easplist-data . . . 15
Extract . . . 16
get_children . . . 17
id_generator . . . 18
indented_list . . . 19
insert_rows . . . 21
levels . . . 22
matched_names-class . . . 23
match_names . . . 23
merge_taxa . . . 25
merge_to_parent . . . 27
parents . . . 28
print_name . . . 28
prune_levels . . . 31
reindex . . . 31
replace_x . . . 33
subset . . . 34
summary . . . 35
tax2traits . . . 37
taxlist-class . . . 38
taxlist2df . . . 39
taxon_names . . . 40
taxon_relations . . . 41
taxon_traits . . . 43
taxon_views . . . 44
tv2taxlist . . . 46

Index **47**

accepted_name *Manage accepted names, synonyms and basionyms*

Description

Taxon usage names for a taxon concept can be divided into three categories: accepted names, basionyms and synonyms. Each single taxon concept may at least have an accepted name, while basionym and synonyms are optional.

The function `accepted_name()` retrieves the accepted names for the indicated taxon concepts or for the whole taxlist object. By using `show_traits=TRUE`, the respective taxon traits will be displayed as well, providing an overview of taxa included in the object. The replacement method for this function will set the respective usage name IDs as accepted names for the respective taxon concept, provided that these names are already set as synonyms in the respective concepts.

The function `synonyms()` is working in a similar way as `accepted_name()`, but this function does not include taxon traits in the output. Alternatives for inserting new synonyms into a taxon concept are either moving synonyms from other taxa by using change_concept<- or inserting new names in the object by using `add_synonym()`.

The function `basionym()` is retrieving and setting basionyms in the respective taxon concepts similarly to `accepted_name`, but this function does not retrieve any information on taxon traits, either.

The fucntion `change_concept<-` replace a taxon usage name (argument `'UsageID'`) to a different taxonomic concept (argument `'value'`).

Usage

```
accepted_name(taxlist, ...)

## S3 method for class 'taxlist'
accepted_name(taxlist, ConceptID, show_traits = FALSE, ...)

accepted_name(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
accepted_name(taxlist, ConceptID, ...) <- value

synonyms(taxlist, ...)

## S3 method for class 'taxlist'
synonyms(taxlist, ConceptID, ...)

basionym(taxlist, ...)

## S3 method for class 'taxlist'
basionym(taxlist, ConceptID, ...)

basionym(taxlist, ...) <- value
```

```
## S3 replacement method for class 'taxlist'
basionym(taxlist, ConceptID, ...) <- value

change_concept(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
change_concept(taxlist, UsageID, ...) <- value
```

Arguments

taxlist	An object of class taxlist.
...	Further arguments passed among methods.
ConceptID	Integer containing concept IDs where to request or set names for one category.
show_traits	Logical value, whether traits should be included in the output of accepted_name or not.
value	Integer containing usage IDs to be set to the respective category in the respective taxon concept.
UsageID	Numeric vector with taxon usage IDs that will be changed to a different taxonomic concept.

Value

Most of the methods return information in data frames, while replacement methods do it as taxlist objects.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

add_synonym() change_concept<-

Examples

```
## Set a different accepted name for Cyclosorus interruptus
summary(Easplist, "Cyclosorus interruptus")
accepted_name(Easplist, 50074) <- 53097
summary(Easplist, 50074)

## Inserting a new name first
summary(Easplist, "Basella alba")
Easplist <- add_synonym(taxlist = Easplist, ConceptID = 68,
    TaxonName = "Basella cordifolia", AuthorName = "Lam.")
summary(Easplist, 68)
accepted_name(Easplist, 68) <- 56139
summary(Easplist, 68)
```

```
## Display synonyms
head(synonyms(taxlist = Easplist))

## Synonyms for an specific concept
synonyms(taxlist = Easplist, ConceptID = 20)

## Basionym for Cyclosrus interruptus
summary(Easplist, 50074)
basionym(Easplist, 50074) <- 53097

summary(Easplist, 50074)
basionym(Easplist, 50074)

## Move the name Typha aethiopica to concept 573 (T. latifolia)
summary(Easplist, c(50105, 573))
change_concept(Easplist, 53130) <- 573
summary(Easplist, c(50105, 573))
```

---

add_concept *Add new taxonomic concepts into taxlist objects*

Description

Alternative methods to add new concepts into existing `taxlist` objects.

Usage

```
add_concept(taxlist, TaxonName, ...)

## S4 method for signature 'taxlist,data.frame'
add_concept(taxlist, TaxonName, ...)

## S4 method for signature 'taxlist,character'
add_concept(taxlist, TaxonName, ...)

## S4 method for signature 'taxlist,taxlist'
add_concept(taxlist, TaxonName, insert_view = FALSE, ...)

update_concept(taxlist, ConceptID, ...)
```

Arguments

`taxlist`	A taxlist object.
`TaxonName`	Character vector with the accepted name for the new taxon concepts.
`...`	Further arguments passed among methods.
`insert_view`	A numeric (integer) vector, indicating the views to be inserted in `taxlist` or the value `TRUE` (see details).
`ConceptID`	Concept IDs to be updated.

as *Coerce taxlist objects to lists.*

Description

Coercion of S4 objects to lists can be applied to explore their content, avoiding errors caused by their validation.

Usage

```
S4_to_list(x)
```

Arguments

x	An object of class taxlist or any S4 class.

Details

Coerce taxlist objects to lists.

Value

An object of class list.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Coerce taxlist to list
tax_list <- as(Easplist, "list")

## Coerce data frame to taxlist
Cyperus <- read.csv(file = file.path(path.package("taxlist"), "cyperus",
        "names.csv"))
Cyperus$AcceptedName <- !Cyperus$SYNONYM
head(Cyperus)

as(Cyperus, "taxlist")
```

backup_object *Make and load backups of R objects*

Description

When work with data becomes risky, the best practice is to produce backup files. The function of `backup_object` is a wrapper of `save()`, adding a time stamp and a suffix to the name of the resulting file (an R image file with extension ***.rda**). The function `load_last` is adapted to this style, loading the newest version to the session.

Usage

```
backup_object(
  ...,
  objects = character(),
  file,
  stamp = TRUE,
  sep = "_",
  date_format = "%Y-%m-%d",
  time_format = "%H:%M:%S",
  overwrite = FALSE
)

sort_backups(
  name,
  path = ".",
  date_format = "%Y-%m-%d",
  fext = ".rda",
  sep = "_"
)

load_last(file, path, ..., choice)
```

Arguments

`...`	Names of the objects to be saved (either symbols or character strings) in `backup_object()`. In `load_last()`, arguments passed to `sort_backups()`.
`objects`	A character vector indicating the names of objects to be included in the backup file.
`file`	A character value indicating the name of the backup file, without the extension.
`stamp`	A logical value indicating whether time should be stamped in the backup name or not.
`sep`	A character value used to separate backup's name from stamp and from the suffix.
`date_format`	A character value indicating the format used for the file stamp. See `strptime()`.

time_format	A character value indicating the format used for the the time (not including the date), which will be used for the invisible report in backup_object(). See strptime().
overwrite	A logical value indicating whether existing files must be overwritten or not.
name	A character value indicating the root of the backup's name.
path	A character value indicating the path to the folder containing the backup files.
fext	A character value indicating the file extension (including the dot symbol).
choice	An integer value indicating the backup file to be used for recovery. This value refers to the row in the output of sort_backups(). If not provided, load_last() will select the newest backup.

Details

In both functions the argument file may include either the path relative to the working directory or the absolute path to the file, excluding stamps and extension. For overwrite=FALSE (the default), a numeric suffix will be added to the backup's name, if another backup was produced at the same day. For overwrite=TRUE no suffix will be included in the file and existing files will be overwritten.

The function load_last() will load the newest version among backups stored in the same folder, provided that the backup name includes a time stamp.

Value

The function backup_object() writes an R-image with extension ***.rda** and an invisible vector with the name of the backup, its absolute path and a time stamp.

The function sort_backups() returns a data frame including the sorted names of backup files from the oldest to the newest.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

save(), load().

Examples

```
## A subset with Pseudognaphalium and relatives
Pseudognaphalium <- subset(x = Easplist, subset = grepl("Pseudognaphalium",
        TaxonName), slot = "names", keep_parents = TRUE)

## Create a backup with date stamp in tempdir
backup_object(Pseudognaphalium, file = file.path(tempdir(), "Pseudognaphalium"))

## Retrieve paths of backup
info_back <- backup_object(Pseudognaphalium, file = file.path(tempdir(),
        "Pseudognaphalium"))
info_back
```

```
## Display all backups
sort_backups("Pseudognaphalium", tempdir())

## Delete object
rm(list = "Pseudognaphalium")

## To load the last backup into a session
load_last("Pseudognaphalium", path = tempdir())

## Load pre-installed backup
load_last(file.path(path.package("taxlist"), "extdata", "Podocarpus"))
```

`clean` *Delete orphaned records*

Description

Manipulation of slots may generate orphaned entries in taxlist objects. The function `clean` deletes such entries and restores the consistency of the objects.

Usage

```
clean(object, ...)

## S4 method for signature 'taxlist'
clean(object, times = 2, ...)
```

Arguments

`object`	A taxlist object.
`...`	Further arguments passed from or to other methods.
`times`	An integer indicating how many times the cleaning should be repeated.

Details

Cleaning of objects will follow the deletion of orphaned names, orphaned taxon trait entries, and orphaned parent entries.

Value

A clean taxlist object.

Author(s)

Miguel Alvarez.

Examples

```
## Direct manipulation of slot taxonRelations generates an invalid object
Easplist@taxonRelations <- Easplist@taxonRelations[1:5, ]

## Now apply cleaning
Easplist <- clean(Easplist)
summary(Easplist)
```

---

clean_strings *Cleaning character strings.*

Description

Multiple, leading and trailing white spaces as well as wrong encodings may cause serious problems in information dealing with taxonomic names. The function `clean_strings` get rid of them.

Usage

```
clean_strings(x, ...)

## S4 method for signature 'character'
clean_strings(x, from = "utf8", to = "utf8", ...)

## S4 method for signature 'factor'
clean_strings(x, from = "utf8", to = "utf8", ...)

## S4 method for signature 'data.frame'
clean_strings(x, from = "utf8", to = "utf8", ...)
```

Arguments

`x`	Object to be cleaned.
`...`	Further arguments passed among methods (not yet in use).
`from, to`	Arguments passed to `iconv()`.

Details

This function automatically deletes leading, trailing and multiple white spaces, either in strings (method `character`), levels (method `factor` or in single columns (method `data.frame`).

Value

The same as input `x`.

Author(s)

Miguel Alvarez.

Examples

```
## Leading, trailing and multiple spaces
clean_strings(" Cyperus    papyrus L.     ")
```

`count_taxa` *Count taxa within a taxlist object.*

Description

Counting number of taxa within taxlist objects or character vectors containing taxon names.

Usage

```
count_taxa(object, data, ...)

## S4 method for signature 'character,missing'
count_taxa(object, na.rm = TRUE, ...)

## S4 method for signature 'factor,missing'
count_taxa(object, na.rm = TRUE, ...)

## S4 method for signature 'taxlist,missing'
count_taxa(object, level, ...)

## S4 method for signature 'formula,taxlist'
count_taxa(object, data, include_na = FALSE, suffix = "_count", ...)
```

Arguments

`object`	An object containing a taxonomic list or a formula.
`data`	An object of class taxlist in the `formula` method.
`...`	further arguments passed among methods.
`na.rm`	Logical value, whether NAs have to be removed from the input vector or not.
`level`	Character value indicating the taxonomic rank of counted taxa.
`include_na`	Logical value indicating whether `NA` values in a taxon trait should be considered for counting taxa or just ignored (only used in `formula` method).
`suffix`	Character value used as suffix for the counted rank in the output data frame (only used in `formula` method).

Details

This function is written by convenience in order to reduce code for counting taxa within taxlist objects and it is just a wrapper of `length()`.

Value

An integer with the number of taxa.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## factor method
count_taxa(iris$Species)

## taxlist method
count_taxa(Easplist)

## count only species
count_taxa(Easplist, level = "species")

## using a formula
count_taxa(~life_form, Easplist, include_na = TRUE)
```

Deprecated-functions *Deprecated functions*

Description

Most of those functions have been replaced by alternative 'update' ones.

Usage

```
add_parent()

add_trait()

add_level()

replace_view()

taxlist2taxmap()

taxmap2taxlist()

taxmap2taxlist()

tnrs()
```

`df2taxlist` *Convert data frames and strings into taxlist objects*

Description

Function converting template data frame into taxlist object. Also character vectors including taxonomic names will be converted but without any information on taxonomic ranks and parental taxa.

Usage

```
df2taxlist(x, ...)

## S3 method for class 'data.frame'
df2taxlist(x, taxonTraits, taxonViews, levels, clean_strings = TRUE, ...)

## S3 method for class 'character'
df2taxlist(x, ...)
```

Arguments

`x`	A data frame or a character vector with taxonomic names. If x is a data frame, the columns **TaxonUsageID** (integer with IDs for each name), **TaxonConceptID** (integer with IDs for the respective taxon concepts), and **TaxonName** (character) are mandatory. Other optional columns are **AuthorName** (character with names' authorities), **AcceptedName** (logical indicating whether the name is an accepted name or a synonym and will be set as TRUE by default), **Level** (factor sorting taxonomic ranks in the bottom-up direction), **Parent** (integer, the taxon concept ID of the parental taxon), and **ViewID** (integer pointing to the ID of taxonomic view, usually a bibliographic reference, and will be used only if 'taxonViews' is provided. Any further column not included in the prototype of taxlist will be considered as names' attributes and inserted in slot **taxonNames**.
`...`	Further arguments passed among methods. For the `'character-method'`, arguments will be passed to the `'data.frame-method'`.
`taxonTraits`	A data frame with attributes of taxonomic concepts (optional). If provided, the column **TaxonConceptID** is mandatorial.
`taxonViews`	A data frame or biblio::lib_df with references of taxonomic views (optional). If provided, the column **ViewID** is mandatorial and have to match the homonymous column at 'x'.
`levels`	A character vector setting the levels or taxonomic ranks from the bottom to the top. This argument is optional and if missing, the column **Level** will be preserved (if factor) or coerced to factor, except in the case that no column **Level** is provided.
`clean_strings`	Logical value, whether function `clean_strings()` should be applied to 'x' or not.

Value

A taxlist object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>.

Examples

```
Cyperus <- read.csv(file = file.path(path.package("taxlist"), "cyperus",
  "names.csv"))
head(Cyperus)

## Convert to 'taxlist' object
Cyperus$AcceptedName <- !Cyperus$SYNONYM
df2taxlist(Cyperus)

## Create a 'taxlist' object from character vectors
Plants <- df2taxlist(c("Triticum aestivum", "Zea mays"), AuthorName = "L.")
summary(Plants, "all")
```

---

dissect_name *Dissect Scientific Names into their Elements*

Description

Depending the degree of resolution and specific roles of nomenclature, strings containing taxon usage names (scientific names) are constructed with different parts. A string with names can be consequently split into those elements, meanwhile the number of elements may suggest the taxonomic ranks.

This function is a wrapper of `strsplit()`, while name element can be re-pasted if indicated in argument `repaste`.

Usage

```
dissect_name(x, split = " ", fixed = TRUE, repaste, ...)
```

Arguments

`x`	A character vector containing taxon names.
`split, fixed, ...`	Arguments passed to `strsplit()`.
`repaste`	An integer vector indicating the elements of the name selected for the output.

Value

A character matrix with as many rows as names in the input vector. If `repaste` is indicated, then the output will be a character vector.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

strsplit()

Examples

```
# A list of variety names
sp_list <- subset(x = Easplist, subset = Level == "variety", slot = "relations")
sp_list <- accepted_name(sp_list)[c(1:10), "TaxonName"]

# split name
dissect_name(sp_list)

# re-paste the two first words (species name)
dissect_name(sp_list, repaste = c(1:2))
```

Easplist-data *List of vascular plants from East Africa*

Description

Example of an incomplete taxonomic list including taxa recorded in East Africa.

Usage

```
Easplist
```

Format

An object of class taxlist.

Details

This list is a subset of the taxonomic list implemented in the database SWEA-Dataveg. Since this list is being complemented regarding stored vegetation plots, it is an incomplete list.

Source

African Plant Database, SWEA-Dataveg.

Examples

```
summary(Easplist)
```

Extract *Extract or Replace Parts of taxlist Objects*

Description

Quick access to slots `taxonTraits` and `taxonRelations` within taxlist objects.

Usage

```
## S4 method for signature 'taxlist'
x[i, j, drop = FALSE]

## S4 method for signature 'taxlist'
x$name
```

Arguments

`x`	Object of class taxlist.
`i`	Integer or logical vector used as index for access to taxon concepts, referring to the rows in slot 'taxonRelations'. These indices can be used to produce a object with a subset of taxon concepts. It is not recommended to use character values for this index.
`j`	Integer, logical or character vector used as index for access to variables in slot 'taxonTraits'. These indices can be used to reduce the number of variables in the mentioned slot.
`drop`	A logical value passed to `Extract`.
`name`	A symbol or character value for the method `$`, corresponding to a variable either at slot 'taxonTraits' or slot 'taxonRelations'.

Value

The method `$` retrieves a vector, while `[` retrieves a subset of the input taxlist object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>.

See Also

taxlist `subset`

Examples

```
## Statistics on life forms
summary(as.factor(Easplist$life_form))

## First concepts in this list
summary(Easplist[1:5, ], "all")
```

`get_children` *Retrieve children or parents of taxon concepts*

Description

Retrieve all children or all parents of a queried taxon concept.

Usage

```
get_children(taxlist, ...)

## S3 method for class 'taxlist'
get_children(taxlist, ConceptID, ...)

get_parents(taxlist, ...)

## S3 method for class 'taxlist'
get_parents(taxlist, ConceptID, ...)
```

Arguments

`taxlist`	A taxlist object.
`...`	Further arguments passed among methods.
`ConceptID`	Concept IDs for selecting parents or children or a subset of `taxlist`.

Details

This function produces subsets of taxlist objects including all children or parents of queried taxon concepts. Multiple concepts can be queried in these function. The argument `ConceptID` can be a vector of concept IDs or a subset of the input `taxlist` object.

Value

A taxlist object with a subset including requested concepts with children or parents.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Subset with family Ebenaceae and children
Ebenaceae <- subset(x = Easplist, subset = TaxonName == "Ebenaceae")
Ebenaceae

Ebenaceae <- get_children(Easplist, Ebenaceae)
Ebenaceae
```

```
## Get parents of Diospyros tricolor
Diostri <- subset(x = Easplist, subset = TaxonConceptID == 52403,
    slot = "relations")
Diostri

Diostri <- get_parents(Easplist, Diostri)
Diostri
```

`id_generator` *Generate Identifiers*

Description

Creating identifiers for new elements in a database.

The function `id_solver()` wil compare to set of identifiers and modify the second to avoid duplicated IDs.

Usage

```
id_generator(
  len,
  minvalue = 1,
  nchar = 10,
  mode = c("numeric", "character"),
  ...
)

id_solver(insert, to, suffix = c("numeric", "character"), sep = "")
```

Arguments

`len`	Numeric value indicating the length of the retrieved vector with identifiers.
`minvalue`	Numeric value indicating the minimum value in the vector of identifiers. Used only for `'mode = "numeric"'`.
`nchar`	Numeric value indicating the number of characters included in the retrieved identifiers. Used only for `'mode = "character"'`.
`mode`	Character value indicating the type of identifier created, which is either numeric (the default) or charcter.
`...`	Further parameters passed to `stringi::stri_rand_strings()`, actually to the argument `'pattern'`.
`insert`	A vector (either numeric or character) containing IDs of elements that will be inserted in a database.
`to`	A vector (either numeric or character) containing IDs of elements thar already exist in target database.

suffix	A character vector indicating the mode used for the suffix. Only 'numeric' or 'character' and partial matchings are accepted here. This argument is only used for character IDs. If 'suffix = "character"', a letter of the alphabet (vector 'letters') will be appended to duplicated IDs.
sep	A character value used as separator between original character ID and the appended suffix.

Value

A vector with IDs created by id_generator(), either as numeric or character. In the case of id_solver(), a vector, which is either identical to 'insert' (if no conflicts) or a vector witht he same properties but with resolved IDs.

Examples

```
## Creating numeric IDs
id_generator(len = 10, minvalue = 5)

## Creating character IDs
id_generator(len = 10, mode = "character")

## Solving duplicates in numeric identifiers
id_solver(insert = c(3, 7, 5, 10), to = c(1:5))

## Solving duplicates in bibtexkeys
db_refs <- c("Alvarez2003", "Schmitz1988", "Li2023")
new_refs <- c("Alvarez2003", "Li2023", "Mueller1953", "Alvarez2003a")
any(duplicated(c(db_refs, new_refs)))

solved_refs <- id_solver(insert = new_refs, to = db_refs, suffix = "character")
solved_refs
any(duplicated(c(db_refs, solved_refs)))
```

---

indented_list *Print hierarchical structure in indented lists*

Description

Print taxonomic hierarchies (ranks and parent-child relationships) from taxlist objects in an indented list.

Usage

```
indented_list(object, ...)

## S4 method for signature 'taxlist'
indented_list(
  object,
  filter,
```

```
  keep_children = TRUE,
  keep_parents = TRUE,
  rankless_as,
  indent = " ",
  lead_br = "",
  print = TRUE,
  author = TRUE,
  level = FALSE,
  synonyms = FALSE,
  syn_encl = c("= ", ""),
  secundum,
  alphabetical = FALSE,
  ...
)
```

Arguments

object	A taxlist object containing taxonomic concepts.
...	Further arguments (not used yet).
filter	A character value (optional) that will be matched with the taxon usage names to produce a subset of 'object'. Note that this filter will be also applied to synonyms, independent of the argument applied in parameter 'synonyms'.
keep_children	A logical value indicating whether children of matched concept should be included in the result.
keep_parents	A logical value indicating whether parents of matched concept should be included in the result.
rankless_as	A character vector indicating a level (taxonomic rank) to which rankless taxa may be set before doing the list.
indent	Symbol used for indentation. This symbol will be multiplied by the depth of the taxonomic rank. The default is a blank space. This can be also provided as a named vector, with a different indentation symbol for the respective taxonomic ranks.
lead_br	Optional line break symbol leading before the indentation. It may be required for r-markdown documents.
print	A logical value indicating whether the indented list should be printed in the console or not (default = TRUE).
author	A logical value indicating whether the author should be printed with the name (default = TRUE).
level	A logical value indicating whether the name of the level (taxonomic rank) should be included before the name or not (default = FALSE).
synonyms	A logical value indicating whether the synonyms should be included after accepted names or not (default = FALSE).
syn_encl	A character vector of length 2 including the symbols used to enclose synonyms. First value will be set before the synonyms and second value, after the synonyms.

secundum	A character value matching a name in slot 'taxonViews', which will be printed as secundum (taxon view). It is not printed by default.
alphabetical	A logical value indicating whether taxa may be sorted by names or by IDs. The default is FALSE, thus taxa are sorted by IDs. Note that argument TRUE may not work properly if the object contains homonymous taxa.

Value

If 'print = TRUE', the indented list is printed in the console. The result, which is a data frame with the elements used to format the names, can be also assigned to an object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Show taxonomy of papyrus
indented_list(Easplist, "papyrus")

## Include synonyms and taxon views
indented_list(Easplist, "papyrus", level = TRUE, synonyms = TRUE,
    secundum = "secundum")
```

---

insert_rows *Insert additional rows to a data frame.*

Description

Adding new rows from data frame sharing some columns. Data contained in `y` is assumed to be additional data and will be appended.

Columns occurring in only one of the data frames will be added to the output.

Usage

```
insert_rows(x, y, ...)

## S4 method for signature 'data.frame,data.frame'
insert_rows(x, y, ...)
```

Arguments

x	A data frame.
y	A data frame including rows (and columns) to be inserted in `x`.
...	Addicional arguments passed among methods.

Value

A data frame.

Examples

```
## Merge data frames including new columns
data(iris)
iris$Species <- paste(iris$Species)
new_iris <- data.frame(Species = rep("humilis", 2), Height = c(15, 20),
    stringsAsFactors = FALSE)
insert_rows(iris, new_iris)
```

---

levels *Set and retrieves hierarchical levels*

Description

Taxonomic hierarchies can be set as levels in taxlist objects, ordered from lower to higher levels.

Add taxonomic levels for specific taxon concepts in a taxlist object. Also changes in concept circumscription may implicate changes in its taxonomic hierarchy.

Usage

```
levels(x)

## S3 method for class 'taxlist'
levels(x)

levels(x) <- value

## S3 replacement method for class 'taxlist'
levels(x) <- value
```

Arguments

x	A taxlist object.
value	A character vector with replacement values for levels o x.

Details

Taxonomic levels will be handled as factors in the taxlist objects. Those levels are useful for creating subsets of related groups (e.g. by functions `get_children()` or `get_parents()`).

Levels in combination to parent-child relationships will be further used for checking consistency of taxonomic lists.

A replacement method of the form `levels(x) <- value` it is also implemented.

Value

A `character` vector or a taxlist object with added or modified taxonomic levels.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

`prune_levels()`

Examples

```
## Get levels of species list
levels(Easplist)

## Add aggregate as new taxonomic level
levels(Easplist) <- c("form", "variety", "subspecies", "species", "complex",
    "aggregate", "genus", "family")
summary(Easplist)
```

---

matched_names-class *Names matched with a reference taxonomic list*

Description

An S3 class containing results of names compared with a reference list. This class enables further methods applied to these outputs, for instance an interactive selection of multiple choices.

---

match_names *Search matchings between character and taxlist objects*

Description

Names provided in a character vector will be compared with names stored in slot `taxonNames` within an object of class taxlist by using the function `stringdist::stringsim()`.

Usage

```
match_names(x, object, ...)

## S4 method for signature 'character,character'
match_names(
  x,
  object,
  UsageID,
```

```
  best = 1,
  nomatch = TRUE,
  method = "lcs",
  cutlevel = NULL,
  ...
)

## S4 method for signature 'character,missing'
match_names(x, best, cutlevel, nomatch = TRUE, ...)

## S4 method for signature 'character,taxlist'
match_names(
  x,
  object,
  show_concepts = FALSE,
  accepted_only = FALSE,
  include_author = FALSE,
  ...
)
```

Arguments

x	A character vector with names to be compared.
object	Either a character vector or a taxlist object containing the taxonomic list for comparison. If missing, the similarity of each name in 'x' will be compared with the rest of the names in the same vector.
...	Further arguments passed among methods.
UsageID	A vector with IDs for single usage names in the compared list. If the IDs are duplicated or not as much as names in 'object', the function retrieves an error message. If missing, this function will number every name anew (see column 'TaxonUsageID' in the output object).
best	Integer value indicating how many matches should be displayed in the output. Matches with the same value of similarity will be considered as one. Note that this argument will be overrode by 'cutlevel'.
nomatch	A logical value indicating wheter names without matches should be included in the output (`'nomatch = TRUE'`) or not (`'nomatch = FALSE'`).
method	Further arguments passed to `stringdist::stringsim()`.
cutlevel	A numeric value indicating a cut level of similarity, considering as match names with similarities equal or bigger than the cut value. This argument overrides 'best'.
show_concepts	Logical value indicating whether the respective taxon concepts should be displayed in output or not.
accepted_only	Logical value indicating whether only accepted names should be matched or all usage names (including synonyms).
include_author	A logical value indicating whether the author name in object (method for taxlist) should be included in the matching list or not.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

stringdist::stringsim()

Examples

```
## Names to be compared
species <- c("Cyperus papyrus", "Typha australis", "Luke Skywalker")

## Comparing character vectors
match_names(c("Cyperus paper", "TIE fighter"), species)

## Retrieve taxon usage names
match_names(species, Easplist)

## Display accepted names in output
match_names(x = species, object = Easplist, show_concepts = TRUE)

# Using cut value for similarity
match_names(x = species, object = Easplist, cutlevel = 0.8)
```

---

merge_taxa *Merge concepts or move names*

Description

Merge taxon concepts form a taxlist object into single ones.

Usage

```
merge_taxa(object, ...)

## S3 method for class 'taxlist'
merge_taxa(
  object,
  concepts,
  level = NULL,
  delete_nomatch = FALSE,
  print_output = FALSE,
  ...
)
```

Arguments

object, taxlist	Object of class taxlist.
...	Further arguments to be passed to or from other methods.
concepts	Numeric (integer) vector including taxon concepts to be merged.
level	Character vector with queried taxonomic ranks. This setting works only if concepts are missing. ranks in between will be merged to their respective parents by merge_to_parent(). Non queried ranks as well as rankless concepts will be deleted from the output.
delete_nomatch	A logical value indicating whether no matched ranks (i.e. top rank and rankless concepts) should be deleted from the output or not.
print_output	Logical value indicating whether the merged concept should be displayed in the console. Thi works only if a vector is provided at concepts.

Details

Taxon concepts indicated in argument concepts will be merged into a single concept. The new concept inherits the ID and respective attributes from slots taxonRelations and taxonTraits from the first taxon concept indicated in argument concepts.

For convenience the resulting concept can be displayed by setting print_output=TRUE but only when using argument concepts.

An alternative application of this function is implemented through the argument level, where all lower rank taxa will be merged to the indicated level or higher (if parent of merged taxa are at a higher rank).

Value

An object of class taxlist.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Merge Cyperus papyrus and Cyperus dives
summary(Easplist, c(206, 197))

merged_cyperus <- merge_taxa(object = Easplist, concepts = c(206, 197),
    print_output = TRUE)

## Subset with Kyllinga species
ky <- subset(Easplist, TaxonName == "Kyllinga", keep_children = TRUE,
    keep_parents = TRUE)
ky
indented_list(ky)

## Merge to species and family
merge_taxa(ky, level = c("species", "family"))
```

```
## Merge to variety and genus
merge_taxa(ky, level = c("variety", "genus"))
```

`merge_to_parent` *Merge taxa to their respective parents*

Description

Aggregation of taxon concepts to their respective parents. All names of aggregated concepts will become synonyms in the target parent.

Usage

```
merge_to_parent(object, ...)

## S3 method for class 'taxlist'
merge_to_parent(object, concept_id, ...)
```

Arguments

`object`	An object of class taxlist.
`...`	Further arguments passed among methods.
`concept_id`	A vector of IDs (TaxonConceptID) of taxa that will be aggregated into their respective parents. Note that if one of the IDs is simultaneously the parent of another ID in the vector, this function will retrieve an error message.

Value

An object of class taxlist with merged taxa.

See Also

`merge_taxa()`

Examples

```
## Subset with Kyllinga species
ky <- subset(Easplist, TaxonName == "Kyllinga", keep_children = TRUE,
    keep_parents = TRUE)
ky
indented_list(ky)

## Merge two species with the genus
summary(ky, c(346, 50400))
summary(ky, "Kyllinga", exact = TRUE)
ky <- merge_to_parent(ky, c(346, 50400))

summary(ky, "Kyllinga", exact = TRUE)
```

parents *Retrieve parents for specific concepts*

Description

Retrieve IDs of parents for selected taxa in a taxonomic list.

Usage

```
parents(taxlist, level, ...)

## S4 method for signature 'taxlist,character'
parents(taxlist, level, concept, ...)
```

Arguments

taxlist	An object of class taxlist containing a taxonomic list.
level	A character value indicating the level at which the parents will be extracted (upwards in the taxonomic ranks).
...	Further arguments passed among methods.
concept	A vector containing concept IDs. The taxa for which the parents will be retrieved. If not provided, parents for every single taxon concept in 'taxlist' will be retrieved.

Examples

```
# Random selection of 5 taxa
IDs <- sample(Easplist@taxonRelations$TaxonConceptID, 5)

# Print names and names of parents
print_name(Easplist, IDs)
print_name(Easplist, parents(Easplist, "genus", IDs))
print_name(Easplist, parents(Easplist, "family", IDs))
```

print_name *Format usage names for publications*

Description

When writing on bio-diversity, usage names could be automatically inserted in documents including the typical italic format for different elements of a scientific name. The function `print_name` can be applied either in markdown documents or for graphics.

In **Rmarkdown** documents use `*Cyperus papyrus* L.` for inserting a formatted a species name.

Usage

```
print_name(object, ...)

## S3 method for class 'character'
print_name(
  object,
  second_mention = FALSE,
  style = "markdown",
  isolate = c("var.", "ssp.", "subsp.", "f.", "fma."),
  trim = c("spp.", "sp.", "species"),
  italics = TRUE,
  collapse,
  ...
)

## S3 method for class 'taxlist'
print_name(
  object,
  id,
  concept = TRUE,
  include_author = TRUE,
  secundum,
  style = "markdown",
  italics = TRUE,
  collapse,
  ...
)
```

Arguments

`object`	An object of class taxlist.
`...`	Further arguments passed among methods.
`second_mention`	Logical value, whether the genus name should be abbreviated or not.
`style`	Character value indicating the alternative format for italics. The available options are `"markdown"` (called within Rmarkdown documents), `"html"` (for documents rendered into html files), `"expression"` (used for labels in graphics), and `"knitr"` (format in LaTeX code).
`isolate`	A character vector with words (usually abbreviations) appearing in the middle of scientific names, which are not formatted in italics.
`trim`	A character vectors with words appearing at the end of scientific names that are not formatted in italics, either.
`italics`	A logical value indicating whether the names should be italized or not.
`collapse`	A character value or vector used to collapse the names and passed to `paste0()`. If its lenght is 2, the second value will be used to connect the two last names. Note that collapse is not yet implemented for `style = "expression"`.
`id`	Integer containing either a concept or a name ID.

concept	Logical value, whether id corresponds to a concept ID or a taxon usage name ID.
include_author	Logical value, whether authors of the name should be mentioned or not.
secundum	Character value indicating the column in slot taxonViews that will be mentioned as *secundum* (according to).

Value

A character value including format to italic font.

See Also

ape::mixedFontLabel().

Examples

```
## Example subspecies
summary(Easplist, 363, secundum = "secundum")

## Empty plot
plot(x = NA, xlim = c(0, 5), ylim = c(7, 1), bty = "n", xaxt = "n", xlab = "",
  ylab = "options")

## Accepted name with author
text(x = 0, y = 1, labels = print_name(Easplist, 363, style = "expression"),
  pos = 4)

## Including taxon view
text(x = 0, y = 2, labels = print_name(Easplist, 363, style = "expression",
  secundum = "secundum"), pos = 4, cex = 0.7)

## Second mention in text
text(x = 0, y = 3, labels = print_name(Easplist, 363, style = "expression",
  second_mention = TRUE), pos = 4)

## Using synonym
text(x = 0, y = 4, labels = print_name(Easplist, 50037, style = "expression",
  concept = FALSE), pos = 4)

## Markdown style
text(0, 5, labels = print_name(Easplist, 363, style = "markdown"), pos = 4)

## HTML style
text(0, 6, labels = print_name(Easplist, 363, style = "html"), pos = 4,
    cex = 0.7)

## LaTeX style for knitr
text(x = 0, y = 7, labels = print_name(Easplist, 363, style = "knitr"), pos = 4,
    cex = 0.7)
```

prune_levels *Prune not used taxonomic ranks*

Description

Taxonomic ranks without taxon concepts will be pruned in taxlist objects.

Usage

```
prune_levels(object, ...)

## S3 method for class 'taxlist'
prune_levels(object, ...)
```

Arguments

object	An object of class taxlist.
...	Further arguments passed among methods (not yet in use).

Value

An object of class taxlist with pruned taxonomic ranks.

See Also

levels()

Examples

```
## Subset species belonging to Cyperus
Cyperus <- subset(Easplist, TaxonName == "Cyperus", slot = "taxonNames",
    keep_children = TRUE, keep_parents = TRUE)
Cyperus

## Prune not used ranks
prune_levels(Cyperus)
```

reindex *Re-index elements of taxlist objects*

Description

The assignment of new identifiers must take into account all possible occurrences of such indices in taxlist objects in order to maintain their validity.

Usage

```
reindex(object, ...)

## S3 method for class 'taxlist'
reindex(object, old, new, idx = "TaxonConceptID", ...)

reindex(object, ...) <- value

## S3 replacement method for class 'taxlist'
reindex(object, ...) <- value
```

Arguments

object	A taxlist object.
...	Further arguments to be passed among methods.
old	A vector with old identifiers to be re-indized. This may contain all identifiers or only a part of them. If only a part, the rest of indices will be preserved. If the changes insert duplicated identifiers, an error message will be retrieved. If missing, all identifiers in 'object' will be considered.
new, value	A vector with the new identifiers. It has to be of the same length as 'old'.
idx	Name of the index to be changed, which means "TaxonConceptID", "TaxonUsageID", or "ViewID" for taxon concepts, taxon usage names, or taxon views, respectively. You can also use the aliases "concept", "usage", and "view".

Value

An object of class taxlist with modified identifiers.

Examples

```
## Copy taxonomic list
sp_list <- Easplist
summary(sp_list, "papyrus")

## Re-index taxon concepts
reindex(sp_list) <- id_generator(nrow(sp_list@taxonRelations),
    mode = "character")

## Re-index taxon usage names
reindex(sp_list, idx = "TaxonUsageID") <- id_generator(nrow(sp_list@taxonNames),
    mode = "character")

## Re-index taxon views
reindex(sp_list, idx = "ViewID") <- id_generator(nrow(sp_list@taxonViews),
    mode = "character")

## Check result
validObject(sp_list)
summary(sp_list, "papyrus")
```

replace_x *Data manipulation.*

Description

This is a series of functions designed for a fast coding of replacements both, as internal functions and in workflows dealing with information stored in vectors. Such functions are especially useful when handling with functional traits stored in taxlist objects.

`replace_x()` is used to exchange values in vectors. `replace_idx()` changes values in vectors by matching indices or conditions. The function `replace_na()` works in the same way as `replace_idx()` but will only insert values in empty elements (NAs).

Usage

```
replace_x(x, old, new)

replace_idx(x, idx1 = x, idx2 = idx1, new)

replace_na(x, idx1, idx2 = idx1, new)
```

Arguments

`x`	A vector to be modified. In the case of `insert_rows()`, `x` is a data frame.
`old`	A vector with values to be replaced by `replace_x()` in a vector.
`new`	A vector containing values to be inserted, either comparing values or using indices.
`idx1`, `idx2`	Indices applied for value replacements to match `x` with `new`, respectively. If `idx2` is not provided, it will be assumed as equivalent to `idx1`.

Value

A vector or data frame with the modified values.

Author(s)

Miguel Alvarez.

Examples

```
## Replace values in vector
replace_x(x = letters, old = c("b", "p", "f"), new = c("bee", "pork", "fungus"))

## Replace values using indices
replace_idx(x = letters, idx1 = 1:length(letters), idx2 = c(2, 7, 17),
  new = c("second", "seventh", "seventeenth"))

## Replace values if they are NAs
```

```
letters[2] <- NA
replace_na(x = letters, idx1 = 1:length(letters), idx2 = c(1:3),
  new = c("alpha", "beta", "zeta"))

## The same applications but this time for functional traits
summary(as.factor(Easplist$life_form))

# Merge annuals
Easplist@taxonTraits$lifeform <- replace_x(x = Easplist@taxonTraits$life_form,
  old = c("obligate_annual", "facultative_annual"), new = c("annual", "annual"))
summary(as.factor(Easplist$lifeform))

# The same effect
Easplist@taxonTraits$lifeform <- replace_idx(x = Easplist@taxonTraits$life_form,
  idx1 = grepl("annual", Easplist@taxonTraits$life_form), idx2 = TRUE,
  new = "annual")
summary(as.factor(Easplist$lifeform))
```

`subset` *Subset method for taxlist objects*

Description

Subset of taxlist objects will be done applying either logical operations or pattern matchings. Subsets can be referred to information contained either in the slot `taxonNames`, `taxonRelations` or `taxonTraits`.

Usage

```
## S4 method for signature 'taxlist'
subset(
  x,
  subset,
  slot = "names",
  keep_children = FALSE,
  keep_parents = FALSE,
  ...
)
```

Arguments

`x`	Object of class taxlist.
`subset`	Logical vector or logical operation to apply as subset.
`slot`	Character value indicating the slot to be used for the subset.
`keep_children`	Logical value applied to hierarchical structures.
`keep_parents`	Logical value applied to hierarchical structures.
`...`	Further arguments to be passed to or from other methods.

Details

The argument `subset` will be applied to the slot specified in argument `slot`. This argument also allows partial matchings.

Arguments `keep_children` and `keep_parents` are applied to objects including parent-child relationships. When those arguments are set as `FALSE` (the default), children or parents of selected taxon concepts will not be included in the subset.

Be aware that `subset()` won't work properly inside of function definitions.

Value

An object of class taxlist.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Produce a data set with only reed plants
sp_list <- subset(x = Easplist, subset = life_form == "reed_plant",
    slot = "taxonTraits", keep_parents = TRUE)
sp_list

summary(as.factor(sp_list$life_form))
```

---

summary *Print overviews for taxlist Objects and their content*

Description

A method to display either an overview of the content of taxlist objects or an overview of selected taxa.

Usage

```
## S4 method for signature 'taxlist'
summary(
  object,
  ConceptID,
  units = "Kb",
  check_validity = TRUE,
  display = "both",
  maxsum = 5,
  secundum = NULL,
  exact = FALSE,
  ...
)
```

```
## S4 method for signature 'taxlist'
show(object)

## S4 method for signature 'taxlist'
print(x, ...)
```

Arguments

object, x	A taxlist object.
ConceptID	IDs of concepts to be displayed in the summary.
units	Character value indicating the units shown in the object's allocated space.
check_validity	Logical value indicating whether the validity of object should be checked or not.
display	Character value indicating the field to be displayed (see details).
maxsum	Integer indicating the maximum number of displayed taxa.
secundum	A character value indicating the column from slottaxonViews to be displayed in the summary.
exact	A logical value indicating whether taxon names should match the exact argument in parameter 'ConceptID'. It works only if 'ConceptID' is provided as character value and is not the keyword 'all'.
...	Further arguments passed to or from another methods.

Details

A general overview indicating number of names, concepts and taxon views included in taxlist objects. If argument ConceptID is a vector with concept IDs or names to be matched by grepl(), then a display of all names included in each concept will be produced. Alternative you can use taxon="all" in order to get the listing of names for all concepts included in the object (truncated to the input number of maxsum).

For summaries applied to concepts, there are three alternative displays of names using the argument display. Use display="name" to show the value TaxonName, display="author" to show the value AuthorName or display="both" to show both values. Such values are taken from slot taxonNames.

For big objects it will be recommended to set units="Mb" (see also object.size() for further alternatives).

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

taxlist

Examples

```
## summary of the object
summary(Easplist, units = "Mb")

## the same output
summary(Easplist)
show(Easplist)
print(Easplist)
Easplist

## summary for two taxa
summary(Easplist, c(51128, 51140))

## summary by matching a name
summary(Easplist, "Acmella")

## summary for the first 10 taxa
summary(object = Easplist, ConceptID = "all", maxsum = 10)
```

`tax2traits` *Set taxonomic information as taxon traits*

Description

Taxonomic classification can be included in taxlist objects within the information provided at slot `taxonRelations`. Nevertheless, for statistical analyses it may be more convenient to insert such information in the slot `taxonTraits`.

Usage

```
tax2traits(object, ...)

## S3 method for class 'taxlist'
tax2traits(object, get_names = FALSE, ...)
```

Arguments

`object`	An object of class taxlist.
`...`	Further arguments to be passed among methods.
`get_names`	Logical value indicating whether taxon names should be retrieved instead of taxon IDs.

Details

This function can only be applied to objects containing parent-child relationships and information on taxonomic levels.

Value

An object of class taxlist with taxonomy added as traits.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>.

Examples

```
## Family Acanthaceae with children
Acanthaceae <- subset(x = Easplist, subset = TaxonName == "Acanthaceae",
  slot = "names", keep_children = TRUE)
summary(Acanthaceae)

## Insert taxonomy to taxon traits
Acanthaceae <- tax2traits(Acanthaceae, get_names = TRUE)
head(taxon_traits(Acanthaceae))
```

---

taxlist-class *An S4 class to represent taxonomic lists.*

Description

Class for taxonomic lists including synonyms, hierarchical ranks, parent-child relationships, taxon views and taxon traits.

Note that each taxon becomes an identifier, represented by the column **TaxonConceptID** in the slot **taxonRelations**, analogous to a primary key in a relational database. This identifier is restricted to an integer in `taxlist` and is specific for the object.

In the same way, each taxon usage name has an identifier in the column **TaxonUsageID**, slot **taxonNames**. The column **ViewID** in slot **taxonViews** is the identifier of the taxon view.

Slots

`taxonNames` (`data.frame`) Table of taxon usage names (accepted names and synonyms).

`taxonRelations` (`data.frame`) Relations between concepts, accepted names, basionyms, parents and hierarchical level.

`taxonTraits` Table of taxon traits.

`taxonViews` References used to determine the respective concept circumscription.

Author(s)

Miguel Alvarez

References

Alvarez M, Luebert F (2018). The taxlist package: managing plant taxonomic lists in R. *Biodiversity Data Journal* 6: e23635. doi:10.3897/bdj.6.e23635

Examples

```
## Class 'taxlist'
showClass("taxlist")

## Create an empty object
sp_list <- new("taxlist")
sp_list
```

`taxlist2df` *Coerce taxlist objects to data frames*

Description

Transform taxlist objects into data frames.

Usage

```
taxlist2df(x, ...)

## S3 method for class 'taxlist'
taxlist2df(
  x,
  include_traits = FALSE,
  include_views = FALSE,
  standard = c("taxlist", "dwc"),
  ...
)
```

Arguments

`x`	A taxlist object to be coerced.
`...`	Further arguments passed among methods.
`include_traits`	A logical value indicating whether taxon concept attributes have to be included in the output or not.
`include_views`	A logical value indicating whether taxon views have to be included in the output or not.
`standard`	A character value indicating the standard used to name columns in the output data frame. Per default `taxlist` names are used but it can be set to `dwc` for renaming some columns according to Darwin Core.

taxon_names *Handle information on taxon usage names.*

Description

The slot taxonNames in taxlist objects contains taxon usage names for the respective taxon. These functions assist on the access and modification of entries for names.

Usage

```
taxon_names(taxlist, ...)

## S3 method for class 'taxlist'
taxon_names(taxlist, ...)

taxon_names(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
taxon_names(taxlist, ...) <- value

add_synonym(taxlist, ...)

## S3 method for class 'taxlist'
add_synonym(taxlist, ConceptID, TaxonName, AuthorName, ...)

update_name(taxlist, ...)

## S3 method for class 'taxlist'
update_name(taxlist, UsageID, ...)

delete_name(taxlist, ...)

## S3 method for class 'taxlist'
delete_name(taxlist, UsageID, ...)
```

Arguments

taxlist	A taxlist object to be modified.
...	Further arguments passed among methods. In update_name are vectors including the variables to be updated for the respective taxon usage ID.
value	A data frame used as new slot taxonNames in taxlist.
ConceptID	Numeric vector indicating the concept ID to which the synonyms will be added.
TaxonName, AuthorName	Character values used for the new names (synonyms).
UsageID	Numeric vector indicating the taxon usage IDs to be updated.

Details

The replacement method `taxon_names<-` is a quick alternative to include names in empty taxlist objects.

The function `add_synonym()` works only for adding names to existing taxon concepts. For adding new taxon concepts as well you should use `add_concept()`.

Value

A data frame or, in the case of the replacement method, a taxlist object with modified slot `taxonNames`.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

taxlist

Examples

```
## Display of slot 'taxonNames'
Euclea <- subset(x = Easplist, subset = charmatch("Euclea", TaxonName),
  slot = "names", keep_children = TRUE)
Euclea
taxon_names(Euclea)

## Insert a synonym to Diospyros scabra
summary(Easplist, "Diospyros scabra")
sp_list <- add_synonym(taxlist = Easplist, ConceptID = 51793,
  TaxonName = "Maba scabra", AuthorName = "Chiov.")
summary(sp_list, "Diospyros scabra")

## Delete a synonym of Launaea cornuta
summary(sp_list, "Launaea cornuta")
sp_list <- delete_name(sp_list, 53821)
summary(sp_list, "Launaea cornuta")

## Hypothetical correction in author name in Launaea cornuta
sp_list <- update_name(taxlist = sp_list, UsageID = 355, AuthorName = "L.")
summary(sp_list, "Launaea cornuta")
```

`taxon_relations` *Retrieve or replace slot taxonRelations in taxlist objects*

Description

Retrieve the content of slot `taxonRelations` from a taxlist object or replace it by a new data frame.

Usage

```
taxon_relations(taxlist, ...)

## S3 method for class 'taxlist'
taxon_relations(taxlist, ...)

taxon_relations(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
taxon_relations(taxlist, ...) <- value

## S4 method for signature 'taxlist,numeric'
update_concept(taxlist, ConceptID, ...)
```

Arguments

`taxlist`	A taxlist object.
`...`	Further arguments passed among methods.
`value`	A `data.frame` object to be set as slot `taxonRelations`.
`ConceptID`	Concept IDs to be updated.

Details

The replacement method `taxon_relations<-` should be only used when constructing taxlist objects from an empty one (prototype).

New concepts should be first added to a taxlist object using their respective accepted names. Synonyms can be further provided using the function `add_synonym()`.

Additional named vectors can be provided to be included in slot `taxonNames`, in the cases where those variables already exist, otherwise they will be ignored.

It is recommended also to provide a concept view as `ViewID` (see `taxon_views()`). For adding a new view, use `add_view()`.

Value

An object of class taxlist with added names and concepts.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

taxlist

Examples

```
## Subset for the genus Euclea and display of slot 'taxonNames'
Euclea <- subset(x = Easplist, subset = charmatch("Euclea", TaxonName),
  slot = "names", keep_children = TRUE)
Euclea
taxon_relations(Euclea)
```

`taxon_traits` *Manipulation of taxon traits in taxlist objects.*

Description

The slot `taxonTraits` in taxlist objects contains attributes of taxon concepts (e.g. functional traits). These functions are suitable for replacing, retrieving and appending trait information in taxonomic lists.

Usage

```
taxon_traits(taxlist, ...)

## S3 method for class 'taxlist'
taxon_traits(taxlist, ...)

taxon_traits(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
taxon_traits(taxlist, ...) <- value

update_trait(taxlist, ...)

## S3 method for class 'taxlist'
update_trait(taxlist, taxonTraits, ...)
```

Arguments

`taxlist`	A taxlist object.
`...`	Further arguments to be passed among methods.
`value`	Data frame to be set as slot `taxonTraits`.
`taxonTraits`	a data frame with taxon traits to be inserted in `'taxlist'`. A column `'TaxonConceptID'` is mandatory in this table. If some taxon concept IDs are not occurring in `'taxlist'`, an error message is retrieved by `update_trait()`.

Details

Taxon traits are contained in a data frame at the slot `taxonTraits` in taxlist objects. To optimise space, this data frame contain only entries for those concepts with information, while taxa with no information are skipped from this table. Thus appending new variables may also have to include new rows in this slot, which is automatically carried out by this function.

The replacement method `taxon_traits<-` should be only used when constructing taxlist objects from an empty one.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

taxlist.

Examples

```
## Displaying taxon traits
head(taxon_traits(Easplist))

## Updating traits for Launaea cornuta
summary(Easplist, "Launaea cornuta")
accepted_name(taxlist = Easplist, ConceptID = 355, show_traits = TRUE)

sp_list <- update_trait(taxlist = Easplist, taxonTraits = data.frame(
        TaxonConceptID = 355,
        life_form = "annual"))
accepted_name(taxlist = sp_list, ConceptID = 355, show_traits = TRUE)
```

---

taxon_views *Management of concept views in taxonomic lists.*

Description

Retrieve or replace slot `taxonViews` in an object of class taxlist

Usage

```
taxon_views(taxlist, ...)

## S3 method for class 'taxlist'
taxon_views(taxlist, ...)

taxon_views(taxlist, ...) <- value

## S3 replacement method for class 'taxlist'
taxon_views(taxlist, ...) <- value
```

```
add_view(taxlist, taxonViews, ...)

## S4 method for signature 'taxlist,data.frame'
add_view(taxlist, taxonViews, ...)
```

Arguments

`taxlist`	A taxlist object.
`...`	Further arguments to be passed among methods.
`value`	An object of class data.frame containing the references used to define the circumscription of taxon concepts included in `taxlist`.
`taxonViews`	A data frame with taxon views to be inserted in `'taxlist'`.

Details

Taxon views indicate in taxlist objects the references determining the circumscription of the respective taxon concepts. When adding a new concept (see `add_concept()`), the respective reference may not yet occur in the input taxlist object.

The term taxon view was introduced by **Zhong et al. (1996)** and corresponds to the reference used for the definition of a concept.

This function retrieves the slot `taxonViews` from objects of the class taxlist.

The replacement method `taxon_views<-` replaces the whole content of slot `taxonViews` and it is only recommended to use when constructing a new taxlist object from an empty prototype.

Value

An object of class taxlist with added views.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

References

Zhong Y, Jung S, Pramanik S, Beaman JH (1996). Data model and comparison and query methods for interacting classifications in a taxonomic database. *Taxon* 45: 223–241. doi:10.1093/bioinformatics/15.2.149

See Also

taxlist

Examples

```
## See existing views
taxon_views(Easplist)

## Add a new view
sp_list <- add_view(taxlist = Easplist, taxonViews = data.frame(
        secundum = "Beentje et al. (1952)",
        Title = "Flora of Tropical East Africa",
        URL = "http://www.kew.org/science/directory/projects/FloraTropEAfrica.html"))

taxon_views(sp_list)
```

`tv2taxlist` *Import species lists from Turboveg databases*

Description

Importing species lists from Turboveg 2 databases into a taxlist object.

Internally the funcions `foreign::read.dbf()` and `df2taxlist()` are called.

Usage

```
tv2taxlist(taxlist, tv_home = tv.home(), ...)
```

Arguments

`taxlist`	Character value indicating the name of a species list in Turboveg.
`tv_home`	Character value indicating the path to the main Turboveg folder. By default the function `vegdata::tv.home()` is called.
`...`	Further arguments passed to `df2taxlist()`.

Value

A taxlist object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

See Also

`df2taxlist()`

Examples

```
## Cyperus data set installed as Turboveg species list
Cyperus <- tv2taxlist(taxlist = "cyperus",
  tv_home = file.path(path.package("taxlist"), "tv_data"))
Cyperus
```

Index

∗ **datasets**
    Easplist-data, 15
[*(*Extract*)*, 16
[,taxlist-method *(*Extract*)*, 16
$ *(*Extract*)*, 16
$,taxlist-method *(*Extract*)*, 16

accepted_name, 3
accepted_name,taxlist-method
        *(*accepted_name*)*, 3
accepted_name.taxlist *(*accepted_name*)*, 3
accepted_name<- *(*accepted_name*)*, 3
accepted_name<-,taxlist-method
        *(*accepted_name*)*, 3
accepted_name<-.taxlist
        *(*accepted_name*)*, 3
add_concept, 5
add_concept(), *41*, *45*
add_concept,taxlist,character-method
        *(*add_concept*)*, 5
add_concept,taxlist,data.frame-method
        *(*add_concept*)*, 5
add_concept,taxlist,taxlist-method
        *(*add_concept*)*, 5
add_level *(*Deprecated-functions*)*, 12
add_parent *(*Deprecated-functions*)*, 12
add_synonym *(*taxon_names*)*, 40
add_synonym(), *3*, *4*, *42*
add_synonym,taxlist-method
        *(*taxon_names*)*, 40
add_synonym.taxlist *(*taxon_names*)*, 40
add_trait *(*Deprecated-functions*)*, 12
add_view *(*taxon_views*)*, 44
add_view(), *42*
add_view,taxlist,data.frame-method
        *(*taxon_views*)*, 44
ape::mixedFontLabel(), *30*
as, 6

backup_object, 7
basionym *(*accepted_name*)*, 3
basionym,taxlist-method
        *(*accepted_name*)*, 3
basionym.taxlist *(*accepted_name*)*, 3
basionym<- *(*accepted_name*)*, 3
basionym<-,taxlist-method
        *(*accepted_name*)*, 3
basionym<-.taxlist *(*accepted_name*)*, 3
biblio::lib_df, *13*

change_concept<-, *3*, *4*
change_concept<- *(*accepted_name*)*, 3
change_concept<-,taxlist-method
        *(*accepted_name*)*, 3
change_concept<-.taxlist
        *(*accepted_name*)*, 3
clean, 9
clean,taxlist-method *(*clean*)*, 9
clean_strings, 10
clean_strings(), *13*
clean_strings,character-method
        *(*clean_strings*)*, 10
clean_strings,data.frame-method
        *(*clean_strings*)*, 10
clean_strings,factor-method
        *(*clean_strings*)*, 10
coerce,character,taxlist-method *(*as*)*, 6
coerce,data.frame,taxlist-method *(*as*)*, 6
coerce,taxlist,data.frame-method *(*as*)*, 6
coerce,taxlist,list-method *(*as*)*, 6
count_taxa, 11
count_taxa,character,missing-method
        *(*count_taxa*)*, 11
count_taxa,factor,missing-method
        *(*count_taxa*)*, 11
count_taxa,formula,taxlist-method
        *(*count_taxa*)*, 11
count_taxa,taxlist,missing-method
        *(*count_taxa*)*, 11

data.frame, *45*
delete_name *(*taxon_names*)*, 40
delete_name,taxlist-method *(*taxon_names*)*, 40
delete_name.taxlist *(*taxon_names*)*, 40
Deprecated-functions, 12
df2taxlist, 13
df2taxlist(), *46*
df2taxlist,character-method *(*df2taxlist*)*, 13
df2taxlist,data.frame-method *(*df2taxlist*)*, 13
df2taxlist.character *(*df2taxlist*)*, 13
df2taxlist.data.frame *(*df2taxlist*)*, 13
dissect_name, 14

Easplist *(*Easplist-data*)*, 15
Easplist-data, 15
Extract, *16*, 16

foreign::read.dbf(), *46*

get_children, 17
get_children(), *22*
get_children,taxlist-method *(*get_children*)*, 17
get_children.taxlist *(*get_children*)*, 17
get_parents *(*get_children*)*, 17
get_parents(), *22*
get_parents,taxlist-method *(*get_children*)*, 17
get_parents.taxlist *(*get_children*)*, 17
grepl(), *36*

iconv(), *10*
id_generator, 18
id_solver *(*id_generator*)*, 18
indented_list, 19
indented_list,taxlist-method *(*indented_list*)*, 19
insert_rows, 21
insert_rows,data.frame,data.frame-method *(*insert_rows*)*, 21

length(), *11*
levels, 22
levels(), *31*
levels,taxlist-method *(*levels*)*, 22
levels.taxlist *(*levels*)*, 22
levels<- *(*levels*)*, 22
levels<-,taxlist-method *(*levels*)*, 22
levels<-.taxlist *(*levels*)*, 22
list, *6*
load(), *8*
load_last *(*backup_object*)*, 7

match_names, 23
match_names,character,character-method *(*match_names*)*, 23
match_names,character,missing-method *(*match_names*)*, 23
match_names,character,taxlist-method *(*match_names*)*, 23
matched_names-class, 23
merge_taxa, 25
merge_taxa(), *27*
merge_taxa,taxlist-method *(*merge_taxa*)*, 25
merge_taxa.taxlist *(*merge_taxa*)*, 25
merge_to_parent, 27
merge_to_parent(), *26*
merge_to_parent,taxlist-method *(*merge_to_parent*)*, 27
merge_to_parent.taxlist *(*merge_to_parent*)*, 27

object.size(), *36*

parents, 28
parents,taxlist,character-method *(*parents*)*, 28
paste0(), *29*
print *(*summary*)*, 35
print,taxlist-method *(*summary*)*, 35
print_name, 28
print_name,character-method *(*print_name*)*, 28
print_name,taxlist-method *(*print_name*)*, 28
print_name.character *(*print_name*)*, 28
print_name.taxlist *(*print_name*)*, 28
prune_levels, 31
prune_levels(), *23*
prune_levels,taxlist-method *(*prune_levels*)*, 31
prune_levels.taxlist *(*prune_levels*)*, 31

reindex, 31

reindex,taxlist-method (reindex), 31
reindex.taxlist (reindex), 31
reindex<- (reindex), 31
reindex<-,taxlist-method (reindex), 31
reindex<-.taxlist (reindex), 31
replace_idx (replace_x), 33
replace_na (replace_x), 33
replace_view (Deprecated-functions), 12
replace_x, 33

S4_to_list (as), 6
save(), *7, 8*
show,taxlist-method (summary), 35
sort_backups (backup_object), 7
stringdist::stringsim(), *23–25*
stringi::stri_rand_strings(), *18*
strptime(), *7, 8*
strsplit(), *14, 15*
subset, *16*, 34
subset(), *35*
subset,taxlist-method (subset), 34
summary, 35
summary,taxlist-method (summary), 35
synonyms (accepted_name), 3
synonyms,taxlist-method (accepted_name), 3
synonyms.taxlist (accepted_name), 3

tax2traits, 37
tax2traits,taxlist-method (tax2traits), 37
tax2traits.taxlist (tax2traits), 37
taxlist, *3–6*, *9*, *11*, *13–17*, *19, 20*, *22–29*, *31–46*
taxlist-class, 38
taxlist2df, 39
taxlist2df,taxlist-method (taxlist2df), 39
taxlist2df.taxlist (taxlist2df), 39
taxlist2taxmap (Deprecated-functions), 12
taxmap2taxlist (Deprecated-functions), 12
taxon_names, 40
taxon_names,taxlist-method (taxon_names), 40
taxon_names.taxlist (taxon_names), 40
taxon_names<- (taxon_names), 40
taxon_names<-,taxlist-method (taxon_names), 40
taxon_names<-.taxlist (taxon_names), 40
taxon_relations, 41
taxon_relations,taxlist-method (taxon_relations), 41
taxon_relations.taxlist (taxon_relations), 41
taxon_relations<- (taxon_relations), 41
taxon_relations<-,taxlist-method (taxon_relations), 41
taxon_relations<-.taxlist (taxon_relations), 41
taxon_traits, 43
taxon_traits,taxlist-method (taxon_traits), 43
taxon_traits.taxlist (taxon_traits), 43
taxon_traits<- (taxon_traits), 43
taxon_traits<-,taxlist-method (taxon_traits), 43
taxon_traits<-.taxlist (taxon_traits), 43
taxon_views, 44
taxon_views(), *42*
taxon_views,taxlist-method (taxon_views), 44
taxon_views.taxlist (taxon_views), 44
taxon_views<- (taxon_views), 44
taxon_views<-,taxlist-method (taxon_views), 44
taxon_views<-.taxlist (taxon_views), 44
tnrs (Deprecated-functions), 12
tv2taxlist, 46

update_concept (add_concept), 5
update_concept,taxlist,numeric-method (taxon_relations), 41
update_name (taxon_names), 40
update_name,taxlist-method (taxon_names), 40
update_name.taxlist (taxon_names), 40
update_trait (taxon_traits), 43
update_trait,taxlist-method (taxon_traits), 43
update_trait.taxlist (taxon_traits), 43

vegdata::tv.home(), *46*