

# Package ‘taxlist’

October 14, 2022

**Version** 0.2.3

**Encoding** UTF-8

**Title** Handling Taxonomic Lists

**Depends** R(>= 3.5.0)

**Imports** biblio, foreign, methods, stats, taxize, stringdist, stringr,  
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)

**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' 'deprecated-functions.R'  
'replace\_x.R' 'dissect\_name.R' 'clean\_strings.R'  
'taxlist-class.R' 'clean.R' 'as.R' 'taxon\_views.R'  
'count\_taxa.R' 'taxon\_names.R' 'taxon\_relations.R'  
'taxon\_traits.R' 'levels.R' 'accepted\_name.R' 'get\_children.R'  
'merge\_taxa.R' 'Extract.R' 'subset.R' 'backup\_object.R'  
'summary.R' 'df2taxlist.R' 'tv2taxlist.R' 'tnrs.R'  
'tax2traits.R' 'match\_names.R' 'print\_name.R' 'indented\_list.R'  
'Easplist-data.R' 'taxlist-package.R'

**VignetteBuilder** knitr

**RoxygenNote** 7.2.1

**NeedsCompilation** no

**Author** Miguel Alvarez [aut, cre] (<<https://orcid.org/0000-0003-1500-1834>>),  
 Zachary Foster [ctb] (<<https://orcid.org/0000-0002-5075-0948>>),  
 Sam Levin [rev],  
 Margaret Siple [rev]

**Maintainer** Miguel Alvarez <kamapu78@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-09-12 06:30:02 UTC

## R topics documented:

taxlist-package . . . . .	3
accepted_name . . . . .	3
as.list . . . . .	5
backup_object . . . . .	6
clean . . . . .	8
clean_strings . . . . .	9
count_taxa . . . . .	10
Deprecated-functions . . . . .	11
df2taxlist . . . . .	12
dissect_name . . . . .	13
Easlist-data . . . . .	14
Extract . . . . .	15
get_children . . . . .	16
indented_list . . . . .	17
levels . . . . .	19
match_names . . . . .	20
merge_taxa . . . . .	22
print_name . . . . .	23
replace_x . . . . .	25
subset . . . . .	27
summary . . . . .	28
tax2traits . . . . .	30
taxlist-class . . . . .	31
taxon_names . . . . .	32
taxon_relations . . . . .	34
taxon_traits . . . . .	36
taxon_views . . . . .	38
tnrs . . . . .	39
tv2taxlist . . . . .	40
<b>Index</b>	<b>42</b>

---

taxlist-package	<i>taxlist: Handling taxonomic lists.</i>
-----------------	---

---

## Description

The class `taxlist` is defined in this package using the S4 language. The main task of `taxlist` objects is to keep the taxonomic coherence of information included in taxonomic lists and to implement functions (methods) for a proper data handling. Objects of class `taxlist` can be included in further objects, for instance in biodiversity records as done in the package `vegetable`.

## Details

The class `taxlist` is defined in this package using the S4 language. The main task of `taxlist` objects is to keep the taxonomic coherence of information included in taxonomic lists and to implement functions (methods) for a proper data handling. Objects of class `taxlist` can be included in further objects, for instance in biodiversity records as done in the package `vegetable`.

For a more detailed description of this package, see Alvarez & Luebert (2018).

## Author(s)

Miguel Alvarez <kamapu78@gmail.com>

## 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

---

accepted_name	<i>Manage accepted names, synonyms and basionyms</i>
---------------	--

---

## 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 functions `accepted_name`, `basionym` and `synonyms` can be used either to display the respective usage names or to set usage names in one of those categories.

## Usage

```
accepted_name(taxlist, ConceptID, ...)

## S4 method for signature 'taxlist,numeric'
accepted_name(taxlist, ConceptID, show_traits = FALSE, ...)

## S4 method for signature 'taxlist,missing'
```

```

accepted_name(taxlist, ConceptID, ...)

accepted_name(taxlist, ConceptID) <- value

## S4 replacement method for signature 'taxlist,numeric,numeric'
accepted_name(taxlist, ConceptID) <- value

synonyms(taxlist, ConceptID, ...)

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

## S4 method for signature 'taxlist,missing'
synonyms(taxlist, ConceptID, ...)

basionym(taxlist, ConceptID, ...)

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

## S4 method for signature 'taxlist,missing'
basionym(taxlist, ConceptID, ...)

basionym(taxlist, ConceptID) <- value

## S4 replacement method for signature 'taxlist,numeric,numeric'
basionym(taxlist, ConceptID) <- value

```

### Arguments

taxlist	An object of class <code>taxlist</code> .
ConceptID	Integer containing concept IDs where to request or set names for one category.
...	Further arguments passed among methods.
show_traits	Logical value, whether traits should be included in the output of <code>accepted_name</code> or not.
value	Integer containing usage IDs to be set to the respective category in the respective taxon concept.

### Details

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 and there is no replacing method for synonyms. 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.

## 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)
```

---

as.list

*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)

## S4 method for signature 'taxlist'
as.list(x, ...)

as(from) <- value

## S4 replacement method for signature 'taxlist'
as(from) <- value
```

**Arguments**

x	An object of class <code>taxlist</code> or any S4 class.
...	further arguments passed to or from other methods.
from	An object of class <code>taxlist</code> , which will be coerced by the replacement method.
value	A character value indicating the class to be coerced to. Here it is always "list".

**Details**

Coerce `taxlist` objects to lists.

**Value**

An object of class `list`.

**Author(s)**

Miguel Alvarez <kamapu78@gmail.com>

**Examples**

```
Easplist <- as.list(Easplist)
class(Easplist)
```

---

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,  
  overwrite = FALSE  
)  
  
load_last(file, fext = ".rda")
```

**Arguments**

...	Names of the objects to be saved (either symbols or character strings).
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.
overwrite	A logical value indicating whether existing files must be overwritten or not.
fext	A character value indicating the file extension (including the dot symbol).

**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**

An R image with extension **\*.rda**.

**Author(s)**

Miguel Alvarez <kamapu78@gmail.com>

**See Also**

[save load](#).

## Examples

```
## Not run:
## A subset with Pseudognaphalium and relatives
Pseudognaphalium <- subset(x = Easplist, subset = grepl(
  "Pseudognaphalium",
  TaxonName
), slot = "names")
Pseudognaphalium <- get_parents(Easplist, Pseudognaphalium)

## Create a backup with date stamp
backup_object(Pseudognaphalium, file = "Pseudonaphalium")

## The same
backup_object(objects = "Pseudognaphalium", file = "Pseudonaphalium")

## To load the last backup into a session
load_last("Pseudognaphalium")

## End(Not run)

## 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

<code>object</code>	A <a href="#">taxlist</a> object.
<code>...</code>	Further arguments passed from or to other methods.
<code>times</code>	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, ]
## Not run:
summary(Easplist)

## End(Not run)

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

---

clean_strings	<i>Cleaning character strings.</i>
---------------	------------------------------------

---

**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 <a href="#">iconv()</a> .

**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**

```
library(taxlist)
clean_strings(" Cyperus    papyrus L.    ")
```

---

count_taxa	<i>Count taxa within a taxlist object.</i>
------------	--

---

**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 <a href="#">taxlist</a> 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_taxa(Easplist, level = "species")

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

---

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()
```

---

```
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 <b>TaxonUsageID</b> (integer with IDs for each name), <b>TaxonConceptID</b> (integer with IDs for the respective taxon concepts), and <b>TaxonName</b> (character) are mandatory. Other optional columns are <b>AuthorName</b> (character with names' authorities), <b>AcceptedName</b> (logical indicating whether the name is an accepted name or a synonym and will be set as TRUE by default), <b>Level</b> (factor sorting taxonomic ranks in the bottom-up direction), <b>Parent</b> (integer, the taxon concept ID of the parental taxon), and <b>ViewID</b> (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 <b>taxonNames</b> .
...	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 <b>TaxonConceptID</b> is mandatory.
taxonViews	A data frame or <a href="#">lib_df</a> with references of taxonomic views (optional). If provided, the column <b>ViewID</b> is mandatory 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 <b>Level</b> will be preserved (if factor) or coerced to factor, except in the case that no column <b>Level</b> is provided.
clean_strings	Logical value, whether function <a href="#">clean_strings()</a> 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
Cyperus <- df2taxlist(Cyperus)
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**

```
Easplist <- subset(x = Easplist, subset = Level == "variety", slot = "relations")
Easplist <- accepted_name(Easplist)[c(1:10), "TaxonName"]

# split name
dissect_name(Easplist)

# re-paste the two first words
dissect_name(Easplist, 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

<code>x</code>	Object of class <code>taxlist</code> .
<code>i</code>	Integer or logical vector used as index for access to taxon concepts, referring to the rows in slot <code>'taxonRelations'</code> . 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.
<code>j</code>	Integer, logical or character vector used as index for access to variables in slot <code>'taxonTraits'</code> . These indices can be used to reduce the number of variables in the mentioned slot.
<code>drop</code>	A logical value passed to <code>Extract</code> .
<code>name</code>	A symbol or character value for the method <code>\$</code> , corresponding to a variable either at slot <code>'taxonTraits'</code> or slot <code>'taxonRelations'</code> .

### 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 ten concepts in this list
summary(Easplist[1:10, ], "all")
```

---

get_children	<i>Retrieve children or parents of taxon concepts</i>
--------------	---

---

### Description

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

### Usage

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

### Arguments

taxlist	A <a href="#">taxlist</a> object.
ConceptID	Concept IDs for selecting parents or children or a subset of taxlist.
...	Further arguments passed among methods.

### 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(Easplist, charmatch("Ebenaceae", TaxonName))
Ebenaceae <- get_children(Easplist, Ebenaceae)

summary(Ebenaceae)
summary(object = Ebenaceae, ConceptID = "all", maxsum = 100)

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

summary(Diostri)
summary(Diostri, "all")
```

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 <a href="#">taxlist</a> 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"
)
```

---

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

```
## S4 method for signature 'taxlist'
levels(x)

## S4 replacement method for signature 'taxlist'
levels(x) <- value
```

### Arguments

x                    A [taxlist](#) object.  
value                A character vector with replacement values for levels of 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` is also implemented.

### Value

A character vector or a [taxlist](#) object with added or modified taxonomic levels.

### Author(s)

Miguel Alvarez <kamapu78@gmail.com>

**Examples**

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

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

---

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 `stringsim()`.

**Usage**

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

## S4 method for signature 'character,character'
match_names(x, object, best = 5, clean = TRUE, decreasing = TRUE, ...)

## S4 method for signature 'character,taxlist'
match_names(
  x,
  object,
  clean = TRUE,
  output = "data.frame",
  best = 5,
  show_concepts = FALSE,
  accepted_only = FALSE,
  method = "lcs",
  sort_by,
  order_args = list(),
  ...
)
```

**Arguments**

<code>x</code>	A character vector with names to be compared.
<code>object</code>	An object of class <code>taxlist</code> to be compared with.
<code>best</code>	Integer value indicating how many from the best matches have to be displayed (only working for <code>output="list"</code> ).

clean	Logical value, whether leading, tailing and double blanks should be deleted from x.
decreasing	Logical value indicating whether retrieved names should be sorted by decreasing or increasing similarity value. In the character method, the sorting corresponds to similarities between the queried value and the reference vector (argument object). In the taxlist method using 'output = "data.frame"', the order corresponds to the similarity of the best match (by default, no sorting is done). This argument is passed to <code>order()</code> .
output	Character value indicating the type of output. Alternative values are "list" (taxon concepts ID's sorted by similarity for each queried name) or "data.frame" (a table including the best match for every queried name).
show_concepts	Logical value, whether respective concepts should be displayed in output or not.
accepted_only	Logical value, whether only accepted names should be matched or all usage names (including synonyms).
method, ...	Further arguments passed to <code>stringsim()</code> .
sort_by	A character vector including the output columns used for sorting the output table. Used only in 'character,taxlist-method'. The function checks the presence of these values as columns in the output data frame.
order_args	A named list including arguments passed to <code>order()</code> in the 'character,taxlist-method'.

**Author(s)**

Miguel Alvarez <kamapu78@gmail.com>

**See Also**

`stringsim()`

**Examples**

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

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

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

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

---

merge_taxa	<i>Merge concepts or move names</i>
------------	-------------------------------------

---

### Description

Merge taxon concepts from a [taxlist](#) object into single ones.

### Usage

```
merge_taxa(object, concepts, level, ...)

## S4 method for signature 'taxlist,numeric,missing'
merge_taxa(object, concepts, print_output = FALSE, ...)

## S4 method for signature 'taxlist,missing,character'
merge_taxa(object, concepts, level, ...)

change_concept(taxlist, UsageID) <- value

## S4 replacement method for signature 'taxlist'
change_concept(taxlist, UsageID) <- value
```

### Arguments

object, taxlist	Object of class <a href="#">taxlist</a> .
concepts	Numeric (integer) vector including taxon concepts to be merged.
level	Character vector indicating the lowest level for merging.
...	Further arguments to be passed to or from other methods.
print_output	Logical value indicating whether the merged concept should be displayed in the console.
UsageID	Numeric vector with taxon usage IDs to be changed from concept.
value	Numeric vector with taxon concept IDs to be assigned to the names.

### 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))

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

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

---

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"),
  ...
)

## S3 method for class 'taxlist'
print_name(
```

```

    object,
    id,
    concept = TRUE,
    include_author = TRUE,
    secundum,
    style = "markdown",
    ...
)

```

### Arguments

object	An object of class <code>taxlist</code> .
...	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.
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 <code>taxonViews</code> that will be mentioned as <i>secundum</i> (according to).

### Value

A character value including format to italic font.

### Author(s)

Miguel Alvarez <kamapu78@gmail.com>

### See Also

`ape::mixedFontLabel()`.

### Examples

```

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)

## 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)

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

```

---

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 and data frames. 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).

The function `insert_rows()` will add rows and columns at the same time. This function will be used when a new table is appended to another but sharing only part of the columns.

**Usage**

```
replace_x(x, old, new)

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

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

insert_rows(x, y)
```

**Arguments**

x	A vector to be modified. In the case of <code>insert_rows()</code> , x is a data frame.
old	A vector with values to be replaced by <code>replace_x()</code> 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.
y	A data frame including rows (and columns) to be inserted in x.

**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))

## 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)

```

---

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

```
Easplist <- subset(
  x = Easplist, subset = life_form == "reed_plant",
  slot = "traits"
)
summary(Easplist)

summary(as.factor(Easplist$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 <a href="#">taxlist</a> 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 <code>slottaxonViews</code> 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 `grep1()`, 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 for 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, ...)

## S4 method for signature 'taxlist'
tax2traits(object, get_names = FALSE, ...)
```

**Arguments**

object	An object of class <code>taxlist</code> .
...	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**

```
library(taxlist)

showClass("taxlist")

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

---

taxon_names	<i>Handle information on taxon usage names.</i>
-------------	---

---

**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, ...)

## S4 method for signature 'taxlist'
taxon_names(taxlist, ...)

taxon_names(taxlist) <- value

## S4 replacement method for signature 'taxlist,data.frame'
taxon_names(taxlist) <- value

add_synonym(taxlist, ConceptID, ...)
```



```
## S4 method for signature 'taxlist'
add_synonym(taxlist, ConceptID, TaxonName, AuthorName, ...)

update_name(taxlist, UsageID, ...)

## S4 method for signature 'taxlist,numeric'
update_name(taxlist, UsageID, ...)

delete_name(taxlist, UsageID, ...)

## S4 method for signature 'taxlist,numeric'
delete_name(taxlist, UsageID, ...)
```

### Arguments

taxlist	A <a href="#">taxlist</a> object to be modified.
...	Further arguments passed among methods. In <code>update_name</code> are vectors including the variables to be updated for the respective taxon usage ID.
value	A data frame used as new slot <code>taxonNames</code> in <code>taxlist</code> .
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")
Easplist <- add_synonym(
  taxlist = Easplist, ConceptID = 51793,
  TaxonName = "Maba scabra", AuthorName = "Chiov."
)
summary(Easplist, "Diospyros scabra")

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

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

```

---

taxon_relations	<i>Retrieve or replace slot taxonRelations in taxlist objects</i>
-----------------	---

---

## Description

Retrieve the content of slot taxonRelations from a [taxlist](#) object or replace it by a new data frame.

## Usage

```

taxon_relations(taxlist, ...)

## S4 method for signature 'taxlist'
taxon_relations(taxlist, ...)

taxon_relations(taxlist) <- value

## S4 replacement method for signature 'taxlist,data.frame'
taxon_relations(taxlist) <- value

add_concept(taxlist, TaxonName, ...)

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

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

```

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

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

### Arguments

taxlist	A <a href="#">taxlist</a> object.
...	Further arguments passed among methods.
value	A data.frame object to be set as slot taxonRelations.
TaxonName	Character vector with the accepted name for the new taxon concepts.
Level	Character vector indicating the level of the concept in the list.
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.

### 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"
)
Euclea <- get_children(Easplist, Euclea)
```

```

Euclea
taxon_relations(Euclea)

## Subset with family Ebenaceae and children
Ebenaceae <- subset(Easplist, charmatch("Ebenaceae", TaxonName))
Ebenaceae <- get_children(Easplist, Ebenaceae)

Ebenaceae
summary(object = Ebenaceae, ConceptID = "all", maxsum = 100)

## Adding a new concept
Ebenaceae <- add_concept(
  taxlist = Ebenaceae, TaxonName = "Euclea acutifolia",
  AuthorName = "E. Mey. ex A. DC.", Level = "species", Parent = 55707, ViewID = 1
)

## A summary again
Ebenaceae
summary(Ebenaceae, "all", maxsum = 100)

## Display two Typha species
summary(Easplist, c("Typha domingensis", "Typha latifolia"))

## Update a concept
summary(Easplist, "Corchorus olitorius")
Easplist <- update_concept(
  taxlist = Easplist, ConceptID = 155,
  Level = "subspecies"
)
summary(Easplist, "Corchorus olitorius")

```

---

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, ...)

## S4 method for signature 'taxlist'
taxon_traits(taxlist, ...)

taxon_traits(taxlist) <- value

```

```
## S4 replacement method for signature 'taxlist,data.frame'
taxon_traits(taxlist) <- value

update_trait(taxlist, ConceptID, ...)

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

### Arguments

taxlist	A <a href="#">taxlist</a> object.
...	Further arguments to be passed among methods.
value	Data frame to be set as slot taxonTraits.
ConceptID	A numeric vector with the respective taxon concept IDs.

### 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

```
head(taxon_traits(Easplist))

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

# Update
Easplist <- update_trait(
  taxlist = Easplist, ConceptID = 355,
  life_form = "annual"
)
accepted_name(taxlist = Easplist, 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, ...)

## S4 method for signature 'taxlist'
taxon_views(taxlist, ...)

taxon_views(taxlist) <- value

## S4 replacement method for signature 'taxlist,data.frame'
taxon_views(taxlist) <- value

add_view(taxlist, ...)

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

## Arguments

<code>taxlist</code>	A <code>taxlist</code> object.
<code>...</code>	Further arguments to be passed among methods.
<code>value</code>	An object of class <code>data.frame</code> containing the references used to define the circumscription of taxon concepts included in <code>taxlist</code> .

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

taxon_views(Easplist)
```

---

tnrs

*Taxonomic Name Resolution Service*

---

**Description**

Methods of `taxize::tnrs()` for `taxlist` objects.

**Usage**

```
tnrs(query, ...)
```

## S4 method for signature 'character'

```
tnrs(query, ...)
```

## S4 method for signature 'taxlist'

```
tnrs(query, min_score = 0.8, source = "iPlant_TNRS", ...)
```

**Arguments**

query	Either a character vector or a taxlist object with names to search.
...	Further arguments passed to <code>taxize::tnrs()</code> .
min_score	Minimum value of score for considering accepted names as suggested by the output.
source	Source database.

**Details**

This function checks for matching of taxon names in `taxlist` objects with the Taxonomic Name Resolution Service (TNRS). Misspelled names as well as author names will be replaced in the the new object and new accepted names will be inserted.

A method for character vectors is defined for the original function.

**Value**

A data frame or an object of class `taxlist`.

**Author(s)**

Miguel Alvarez <kamapu78@gmail.com>

**See Also**

`taxize::tnrs()`

---

tv2taxlist

*Import species lists from Turboveg databases*

---

**Description**

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

Internally the functions `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 <code>tv.home()</code> from <code>vegdata-package</code> is called.
...	Further arguments passed to <code>df2taxlist()</code> .



**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, 14
  - [ (Extract), 15
  - [, taxlist-method (Extract), 15
  - \$ (Extract), 15
  - \$, taxlist-method (Extract), 15
- accepted\_name, 3
- accepted\_name, taxlist, missing-method (accepted\_name), 3
- accepted\_name, taxlist, numeric-method (accepted\_name), 3
- accepted\_name<- (accepted\_name), 3
- accepted\_name<-, taxlist, numeric, numeric-method (accepted\_name), 3
- add\_concept (taxon\_relations), 34
- add\_concept(), 33, 38
- add\_concept, taxlist, character-method (taxon\_relations), 34
- add\_concept, taxlist, taxlist-method (taxon\_relations), 34
- add\_level (Deprecated-functions), 11
- add\_parent (Deprecated-functions), 11
- add\_synonym (taxon\_names), 32
- add\_synonym(), 5, 35
- add\_synonym, taxlist-method (taxon\_names), 32
- add\_trait (Deprecated-functions), 11
- add\_view (taxon\_views), 38
- add\_view(), 35
- add\_view, taxlist-method (taxon\_views), 38
- ape::mixedFontLabel(), 24
- as.list, 5
- as.list, taxlist-method (as.list), 5
- as<- (as.list), 5
- as<- , taxlist-method (as.list), 5
- backup\_object, 6
- basionym (accepted\_name), 3
- basionym, taxlist, missing-method (accepted\_name), 3
- basionym, taxlist, numeric-method (accepted\_name), 3
- basionym<- (accepted\_name), 3
- basionym<- , taxlist, numeric, numeric-method (accepted\_name), 3
- change\_concept<- , 5
- change\_concept<- (merge\_taxa), 22
- change\_concept<- , taxlist-method (merge\_taxa), 22
- clean, 8
- clean, taxlist-method (clean), 8
- clean\_strings, 9
- clean\_strings(), 12
- clean\_strings, character-method (clean\_strings), 9
- clean\_strings, data.frame-method (clean\_strings), 9
- clean\_strings, factor-method (clean\_strings), 9
- count\_taxa, 10
- count\_taxa, character, missing-method (count\_taxa), 10
- count\_taxa, factor, missing-method (count\_taxa), 10
- count\_taxa, formula, taxlist-method (count\_taxa), 10
- count\_taxa, taxlist, missing-method (count\_taxa), 10
- data.frame, 38
- delete\_name (taxon\_names), 32
- delete\_name, taxlist, numeric-method (taxon\_names), 32
- Deprecated-functions, 11
- df2taxlist, 12
- df2taxlist(), 40, 41

- df2taxlist, character-method  
(df2taxlist), 12
- df2taxlist, data.frame-method  
(df2taxlist), 12
- df2taxlist.character(df2taxlist), 12
- df2taxlist.data.frame(df2taxlist), 12
- dissect\_name, 13
- Easplist (Easplist-data), 14
- Easplist-data, 14
- Extract, 15, 15
- get\_children, 16
- get\_children(), 19
- get\_children, taxlist, numeric-method  
(get\_children), 16
- get\_children, taxlist, taxlist-method  
(get\_children), 16
- get\_parents (get\_children), 16
- get\_parents(), 19
- get\_parents, taxlist, numeric-method  
(get\_children), 16
- get\_parents, taxlist, taxlist-method  
(get\_children), 16
- grepl(), 29
- iconv(), 9
- indented\_list, 17
- indented\_list, taxlist-method  
(indented\_list), 17
- insert\_rows (replace\_x), 25
- length(), 11
- levels, 19
- levels, taxlist-method (levels), 19
- levels<- (levels), 19
- levels<-, taxlist-method (levels), 19
- lib\_df, 12
- list, 6
- load, 7
- load\_last (backup\_object), 6
- match\_names, 20
- match\_names, character, character-method  
(match\_names), 20
- match\_names, character, taxlist-method  
(match\_names), 20
- merge\_taxa, 22
- merge\_taxa, taxlist, missing, character-method  
(merge\_taxa), 22
- merge\_taxa, taxlist, numeric, missing-method  
(merge\_taxa), 22
- object.size(), 30
- order(), 21
- print (summary), 28
- print, taxlist-method (summary), 28
- print\_name, 23
- print\_name, character-method  
(print\_name), 23
- print\_name, taxlist-method (print\_name),  
23
- print\_name.character (print\_name), 23
- print\_name.taxlist (print\_name), 23
- read.dbf(), 40
- replace\_idx (replace\_x), 25
- replace\_na (replace\_x), 25
- replace\_view (Deprecated-functions), 11
- replace\_x, 25
- S4\_to\_list (as.list), 5
- save, 7
- save(), 6
- show, taxlist-method (summary), 28
- stringsim(), 20, 21
- strsplit(), 13, 14
- subset, 15, 27
- subset(), 28
- subset, taxlist-method (subset), 27
- summary, 28
- summary, taxlist-method (summary), 28
- synonyms (accepted\_name), 3
- synonyms, taxlist, missing-method  
(accepted\_name), 3
- synonyms, taxlist, numeric-method  
(accepted\_name), 3
- tax2traits, 30
- tax2traits, taxlist-method (tax2traits),  
30
- taxize::tnrs(), 39, 40
- taxlist, 3–6, 8–20, 22–25, 27–41
- taxlist-class, 31
- taxlist-package, 3
- taxlist2taxmap (Deprecated-functions),  
11
- taxmap2taxlist (Deprecated-functions),  
11

taxon\_names, [32](#)  
taxon\_names, taxlist-method  
    (taxon\_names), [32](#)  
taxon\_names<- (taxon\_names), [32](#)  
taxon\_names<-, taxlist, data.frame-method  
    (taxon\_names), [32](#)  
taxon\_relations, [34](#)  
taxon\_relations, taxlist-method  
    (taxon\_relations), [34](#)  
taxon\_relations<- (taxon\_relations), [34](#)  
taxon\_relations<-, taxlist, data.frame-method  
    (taxon\_relations), [34](#)  
taxon\_traits, [36](#)  
taxon\_traits, taxlist-method  
    (taxon\_traits), [36](#)  
taxon\_traits<- (taxon\_traits), [36](#)  
taxon\_traits<-, taxlist, data.frame-method  
    (taxon\_traits), [36](#)  
taxon\_views, [38](#)  
taxon\_views(), [35](#)  
taxon\_views, taxlist-method  
    (taxon\_views), [38](#)  
taxon\_views<- (taxon\_views), [38](#)  
taxon\_views<-, taxlist, data.frame-method  
    (taxon\_views), [38](#)  
tnrs, [39](#)  
tnrs, character-method (tnrs), [39](#)  
tnrs, taxlist-method (tnrs), [39](#)  
tv.home(), [40](#)  
tv2taxlist, [40](#)  
  
update\_concept (taxon\_relations), [34](#)  
update\_concept, taxlist, numeric-method  
    (taxon\_relations), [34](#)  
update\_name (taxon\_names), [32](#)  
update\_name, taxlist, numeric-method  
    (taxon\_names), [32](#)  
update\_trait (taxon\_traits), [36](#)  
update\_trait, taxlist, numeric-method  
    (taxon\_traits), [36](#)  
  
vegdata-package, [40](#)