Package: vegtable (via r-universe)

September 8, 2024

Version 0.1.9

Encoding UTF-8

Title Handling Vegetation Data Sets

Depends R(>= 3.5.0), taxlist (>= 0.2.4)

Imports foreign, methods, qdapRegex, sp, stats, stringi, utils, vegdata

Suggests biblio, knitr, rmarkdown, sf, testthat, vegan

Description Import and handling data from vegetation-plot databases, especially data stored in 'Turboveg 2' (<https://www.synbiosys.alterra.nl/turboveg/>). Also import/export routines for exchange of data with 'Juice' (<https://www.sci.muni.cz/botany/juice/>) are implemented.

LazyData true

Roxygen list(markdown = TRUE)

License GPL (>= 2)

URL https://github.com/kamapu/vegtable, http://kamapu.github.io/vegtable/

BugReports https://github.com/kamapu/vegtable/issues

Collate 'imports.R' 'NULLing.R' 'coverconvert-class.R' 'vegtable-class.R' 'shaker-class.R' 'df2coverconvert.R' 'new_relation.R' 'relation2header.R' 'cover_trans.R' 'clean.R' 'coerce-methods.R' 'add_releves.R' 'header.R' 'Extract.R' 'veg_relation.R' 'vegtable_stat.R' 'df2vegtable.R' 'used_synonyms.R' 'taxa2samples.R' 'subset.R' 'names.R' 'tv2vegtable.R' 'crosstable.R' 'veg_aggregate.R' 'write_juice.R' 'layers2samples.R' 'make_cocktail.R' 'summary.R' 'count_taxa.R' 'trait_stats.R' 'update_det.R' 'veg_diversity.R' 'new_layer.R' 'aspect_conv-data.R' 'braun_blanquet-data.R' 'dune_veg-data.R' 'Kenya_veg-data.R' 'Wetlands-data.R' 'veg_layers-data.R'

RoxygenNote 7.3.1

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

RemoteUrl https://github.com/kamapu/vegtable

RemoteRef HEAD

RemoteSha 9a97a87940130cb38f0fea22399d5c80504c05ce

Contents

`add_releves` *Merge relevés from data frames into vegtable objects*

Description

Addition of plot observations into existing data sets may implicate merging data frames with vegtable objects.

Since this function will only update slots **samples** and **header**, consistency with slots **layers**, **relations** and **species** have to be checked and accordingly updated in advance.

Usage

```
add_releves(vegtable, releves, ...)

## S4 method for signature 'vegtable,data.frame'
add_releves(
  vegtable,
  releves,
  header,
  abundance,
  split_string,
  usage_ids = FALSE,
  layers = FALSE,
  layers_var,
  format = "crosstable",
  preserve_ids = FALSE,
  ...
)

add_releves(vegtable, ...) <- value

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

Arguments

`vegtable`	An object of class vegtable.
`releves`	A data frame including plot observations to be added into `vegtable`.
`...`	Further arguments passed to function `cross2db()` (i.e. `na_strings`).
`header`	A data frame (optional) including header information for plots.
`abundance`	A character value (or vector of length 2) indicating the names of abundance variable in `vegtable`.
`split_string`	Character value used to split mixed abundance codes.
`usage_ids`	Logical value indicating whether species are as taxon usage ids (integers) or names in `releves`.

`layers`	Logical value indicating whether layers are included in `releves` or not.
`layers_var`	Name of the layer variable in `vegtable`.
`format`	Character value indicating input format of `releves` (either `"crosstable"` or `"databaselist"`).
`preserve_ids`	A logical value, whether IDs in input data set should used as `ReleveID` or not. Those IDs have to be integers and if one of those already exists in `vegtable`, an error will be retrieved.
`value`	A data frame containing new plot observations. I is passed to parameter 'releves' by the replace method.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

See Also

`cross2db()`

`as` *Coerce objects to lists*

Description

Coerce vegtable objects to a list with every slot as a component of the list. This way allows to explore content and solve problems when validity checks fail.

Coercion is applied for different classes by vegtable.

Usage

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

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

Arguments

`x`	An object to be coerced.
`...`	further arguments passed from or to other methods.

Value

An object of class `list`.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## vegtable as list
veg <- as(Kenya_veg, "list")
names(veg)

## coverconvert as list
as(Kenya_veg@coverconvert, "list")
```

`aspect_conv-data` *Conversion of aspect classes to azimuth*

Description

Conversion table required to transform values of aspect to azimuth in degrees.

Usage

```
aspect_conv
```

Format

A numeric vector of values in degrees for the symbols used as names.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
aspect_conv[c("N", "S", "ENE", "SSW")]
```

`braun_blanquet-data` *Conversion of Braun-Blanquet codes to cover percentage*

Description

Cover values conversion as coverconvert object.

Object of class coverconvert contains conversion tables usually from a categorical variable (a cover scale) to a numerical one (equivalent percentage cover value). Cover values are stored as range for each level in the scale (minimum and maximum cover value).

Usage

```
braun_blanquet
```

Format

An object of class `coverconvert`.

See Also

coverconvert `cover_trans()`

Examples

```
names(braun_blanquet)
summary(braun_blanquet)
summary(braun_blanquet$b_bbds)
```

`clean` *Clean orphaned records in vegtable object*

Description

Delete entries in slots `header` and `species` orphaned by manipulation of slots.

Orphaned records generated by modifications in some slots may cause a loss on the validity of vegtable objects. This function should be applied to optimise the allocated size of a vegtable object, as well. Since running cleaning only once does not assure the deletion of all orphaned entries, it is recommended to run it at least twice. This repetition of cleaning is controlled by the argument `times`.

Usage

```
clean_once(object)

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

Arguments

`object`	A vegtable object.
`times`	Numeric value indicating how many times should be the cleaning be repeated.
`...`	Further arguments passed from or to other methods.

Value

A clean vegtable object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Create an invalid object
veg <- Kenya_veg
veg@header <- veg@header[1:10, ]

## Resolve invalidity
veg <- clean(veg)
```

`count_taxa` *Count taxa included in vegtable objects*

Description

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

This function provides a quick calculation of taxa in vegtable objects, considering only records in slot samples. Such records can be also merged from lower ranks.

For the formula method, units without any requested taxa will not appear in the output data frame. If no taxa at all is occurring at the requested level in any unit, an error message will be retrieved.

Usage

```
## S4 method for signature 'vegtable,missing'
count_taxa(object, level, include_lower = FALSE, ...)

## S4 method for signature 'formula,vegtable'
count_taxa(
  object,
  data,
  include_lower = FALSE,
  suffix = "_count",
  in_header = TRUE,
  ...
)

count_taxa(data, ...) <- value

## S4 replacement method for signature 'vegtable,formula'
count_taxa(data, ...) <- value
```

Arguments

`object`	An object of class vegtable or a formula.
`level`	Character value indicating the taxonomic rank of counted taxa.
`include_lower`	Logical value, whether lower taxonomic ranks should be included at the requested level.

`...`	further arguments passed among methods.
`data`	An object of class vegtable.
`suffix`	Character value used as suffix on the calculated variable.
`in_header`	Logical value, whether the result should be included in the slot header of the input vegtable object or not. If the formula term is related to a categorical variable at header, the result will be inserted in the respective table at slot **relations**.
`value`	A formula passed to parameter 'object' by the replace method.

Value

An data frame with the number of taxa from requested level at requested units for the formula method, or just an integer value.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Different alternatives
count_taxa(Kenya_veg)
head(count_taxa(~ReleveID, Kenya_veg, in_header = FALSE))
head(count_taxa(species ~ ReleveID, Kenya_veg, in_header = FALSE))
head(count_taxa(species ~ ReleveID, Kenya_veg, TRUE, in_header = FALSE))
head(count_taxa(family ~ ReleveID, Kenya_veg, TRUE))
```

`coverconvert` *Cover conversion tables*

Description

Cover conversion tables for vegtable objects.

This class implements conversions from different cover scales in percentage cover. For transformations to percentage cover, the function `cover_trans()` should be than used.

Slots

`value` List containing the levels of each scale.

`conversion` List with the respective start and end cut levels for the scale levels.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

See Also

`tv2coverconvert()` braun_blanquet.

`cover_trans` *Convert cover scales to percent cover*

Description

Convert values of a categorical cover scale to percentage values.

This function requires as input a coverconvert object which contains the conversion tables.

Usage

```
## S4 method for signature 'character,coverconvert'
cover_trans(x, conversion, from = NULL, rule = "top", zeroto = 0.1, ...)

## S4 method for signature 'factor,coverconvert'
cover_trans(x, conversion, ...)

## S4 method for signature 'numeric,coverconvert'
cover_trans(x, conversion, ...)

## S4 method for signature 'vegtable,missing'
cover_trans(x, to, replace = FALSE, rule = "top", zeroto = 0.1, ...)
```

Arguments

`x`	Either a factor or character vector, or a vegtable object.
`conversion`	An object of class vegtable.
`from`	Scale name of values in `x` as character value.
`rule`	A character value indicating the rule applied for cover transformation. Three rules are implemented for transformation, either `top` (values transformed to the top of the range), `middle` (transformation at the midpoint), and `bottom` (conversion at the lowest value of the range). In the later case, if the bottom is zero cover, a fictive bottom value can be set by `'zeroto'`
`zeroto`	Value set for transformation of classes with bottom at 0% cover.
`...`	Further arguments passed from or to other methods.
`to`	Name of the column in slot `samples` for writing converted values.
`replace`	Logical value indicating whether existing cover values should be replaced by the new computed values or not.

Value

Either a vector or a vegtable object.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Check the available scales
summary(Kenya_veg@coverconvert)

## Conversion by default 'top' rule
Kenya_veg <- cover_trans(Kenya_veg, to = "percent")
summary(as.factor(Kenya_veg@samples$percent))

## Conversion by 'middle' rule
Kenya_veg <- cover_trans(Kenya_veg, to = "percent", rule = "middle", replace = TRUE)
summary(as.factor(Kenya_veg@samples$percent))

## Conversion by 'bottom' rule
Kenya_veg <- cover_trans(Kenya_veg, to = "percent", rule = "bottom", replace = TRUE)
summary(as.factor(Kenya_veg@samples$percent))
```

`crosstable` *Generating cross tables from database lists*

Description

cross table is the most common format required by statistical packages used to analyse vegetation data (e.g. vegan).

You may use for convenience a formula as `'abundance ~ plot + species + ...'`. Additional variables used for rows (`...`) can be for instance the layers. For objects of class vegtable, the formula can also include variables from the species list (for example `AcceptedName`, `AuthorName`) or even taxon traits.

If required, tables already formatted as cross tables can be converted into column-oriented tables by using the function `cross2db()`.

Usage

```
crosstable(formula, data, ...)

## S4 method for signature 'formula,data.frame'
crosstable(
  formula,
  data,
  FUN,
  na_to_zero = FALSE,
  use_nas = TRUE,
  as_matrix = FALSE,
  ...
)

## S4 method for signature 'formula,vegtable'
crosstable(
```

```
  formula,
  data,
  FUN,
  level,
  include_lower = FALSE,
  na_to_zero = FALSE,
  use_nas = TRUE,
  ...
)

cross2db(object, ...)

## S3 method for class 'data.frame'
cross2db(object, layers = FALSE, na_strings, ...)

## S3 method for class 'matrix'
cross2db(object, ...)
```

Arguments

`formula`	A formula indicating the variables used in the cross table. This formula can be represented as `'abundance ~ cols + rows'`, where `'abundance'` is the numeric variable quantified for a row in a column, for instance the abundance of a species in a plot. Further variables can be set as additional rows indices in a cross table.
`data`	Either a data frame or an object of class vegtable.
`...`	Further arguments passed to the function `stats::aggregate()`.
`FUN`	Function used to aggregate values in the case of a multiple occurrence of a species in a plot, for instance.
`na_to_zero`	A logical value indicating whether zeros should be inserted into empty cells or not.
`use_nas`	Logical value indicating whether NAs should be considered as levels for categorical variables or not.
`as_matrix`	A logical value, whether output should be done as matrix or data frame.
`level`	A character vector with taxonomic ranks (levels) requested in the cross table.
`include_lower`	A logical value indicating wether lower value to the requested levels should be merged or not. It works only if `'level'` is not missing. Note that if you like to include higher ranks or rankless taxa in the cross table, you will rahter need to run `merge_taxa()` on slot **species**.
`object`	A data frame or a matrix including a cross table. Note that `cross2db()` assumes observations as columns and species (and layers) as rows in the `data.frame-method` but species as columns and observations as rows in the `matrix-method`.
`layers`	Logical value, whether the cross table includes a layer column or not.
`na_strings`	Character vector indicating no records in the cross table.

Value

An object of class data.frame.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
# Produce a subset
veg <- subset(Kenya_veg, REFERENCE == 2331, slot = "header")

## transform cover to percentage
veg <- cover_trans(veg, to = "cover_perc", rule = "middle")

## cross table of the first 5 plots
Cross <- crosstable(cover_perc ~ ReleveID + AcceptedName + AuthorName,
  data = veg[1:5, ], FUN = mean, na_to_zero = TRUE)
head(Cross)

## cross table of recorded genera
Cross <- crosstable(cover_perc ~ ReleveID + AcceptedName + AuthorName,
    data = veg, FUN = mean, level = "genus")
head(Cross[ , 1:7])

## cross table of data merged to genus
Cross <- crosstable(cover_perc ~ ReleveID + AcceptedName + AuthorName,
    data = veg, FUN = sum, level = "genus", include_lower = TRUE)
head(Cross[ , 1:7])

## the same for families
Cross <- crosstable(cover_perc ~ ReleveID + AcceptedName + AuthorName,
    data = veg, FUN = sum, level = "family", include_lower = TRUE)
head(Cross[ , 1:7])
```

`df2coverconvert` *Create coverconvert objects*

Description

The class coverconvert contains tables for transforming cover values to percentage using the function `cover_trans()`. These objects can be created from conversion tables imported as data frames.

Usage

```
df2coverconvert(x, ...)

## S3 method for class 'list'
df2coverconvert(x, ...)

## S3 method for class 'data.frame'
df2coverconvert(x, name, ...)
```

Arguments

`x`	Either a data frame or a list of data frames containing the conversion table. Three columns are mandatory in such data frames, namely **value** (factor with the symbols for each class in the cover scale, sorted from the lowest to the highest value), **bottom** (numeric value with the bottom values of each class), and **top** (numeric value with the top values of each class). The values **bottom** and **top** are usually as cover percentage but they may refer to any other numeric abundance.
`...`	Further arguments passed among methods.
`name`	A character value used as name of the cover scale in the data frame method. In the list method, this name will be extracted from the names of the elements in the list.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Convert object into list
cov <- as(Kenya_veg@coverconvert, "list")

## Convert back to coverconvert
cov <- df2coverconvert(cov)
```

`df2vegtable` *Convert a data frame into a vegtable object.*

Description

Conversion of a data frame containing a cross table of abundance or cover of species in single plots.

This function coerces a data frame containing a vegetation cross table into a vegtable object. The input data frame `x` may include information on the layers or not.

Usage

```
df2vegtable(x, species, layer, ...)

## S4 method for signature 'data.frame,numeric,numeric'
df2vegtable(x, species, layer, ...)

## S4 method for signature 'data.frame,numeric,missing'
df2vegtable(x, species, layer, ...)
```

Arguments

`x`	A data frame formatted for a taxlist object.
`species`	Numeric or integer indicating the position of the column with species names.
`layer`	Numeric or integer indicating the position of the column with layers.
`...`	Further arguments passed from or to other methods.

Value

A vegtable object.

Author(s)

Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Creating data set 'dune_veg'
library(vegan)

## Load data from vegan
data(dune)
data(dune.env)

## Conversion to vegtable
dune_veg <- data.frame(
  species = colnames(dune), t(dune),
  stringsAsFactors = FALSE, check.names = FALSE
)
dune_veg <- df2vegtable(dune_veg, species = 1)

summary(dune_veg)

## Adding environmental variables
dune.env$ReleveID <- as.integer(rownames(dune.env))
header(dune_veg) <- dune.env

summary(dune_veg)
```

`dune_veg-data` *Dutch dune meadows as vegtable*

Description

Data set from the package vegan::vegan, converted to a vegtable object.

Usage

```
dune_veg
```

Format

An object of class `vegtable`.

Source

Original data were imported from vegan::dune.

References

Jongman RHG, ter Braak CJF, van Tongeren OFR (1987). *Data analysis in community and landscape ecology.* Pudoc, Wageningen, NL.

Examples

```
summary(dune_veg)
```

`Extract` *Select or replace elements in objects*

Description

Methods for quick access to slot `header` of vegtable objects or for access to single cover scales in coverconvert objects. Also replacement methods are implemented.

Usage

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

## S4 replacement method for signature 'vegtable,ANY'
x$name <- value

## S4 method for signature 'coverconvert'
x$name

## S4 method for signature 'coverconvert'
x[i]

## S4 replacement method for signature 'coverconvert,coverconvert'
x$name <- value

## S4 method for signature 'vegtable'
x[i, j, ..., drop = FALSE]

## S4 replacement method for signature 'vegtable'
x[i, j] <- value
```

Arguments

`x`	Object of class vegtable.
`name`	A name to access.
`value`	Either a vectors or a list, used as replacement.
`i, j`	Indices for access.
`...`	Further arguments passed to or from other methods.
`drop`	A logical value passed to Extract.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Range of latitude values in database
range(Kenya_veg$LATITUDE)

## Summary of countries
summary(Kenya_veg$COUNTRY)
summary(droplevels(Kenya_veg$COUNTRY))

## First 5 samples
summary(Kenya_veg[1:5, ])
```

`header` *Retrieve or replace slot header in vegtable objects*

Description

Retrieve or replace the content of slot `header` in vegtable objects.

Usage

```
header(x, ...)

## S4 method for signature 'vegtable'
header(x, ...)

header(x) <- value

## S4 replacement method for signature 'vegtable,data.frame'
header(x) <- value
```

Arguments

`x`	Object of class vegtable.
`...`	Further arguments passed to or from other methods.
`value`	Data frame to be set as slot `header`.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
head(header(Kenya_veg))
```

`Kenya_veg-data` *Vegetation-plots from Kenya*

Description

A subset of `http://www.givd.info/ID/AF-00-006SWEA-Dataveg` including five references providing plots collected in Kenya.

Usage

```
Kenya_veg
```

Format

An object of class vegtable.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`> and Michael Curran <`currmi01@gmail.com`>

Source

`http://www.givd.info/ID/AF-00-006`

References

Bronner G (1990). *Vegetation and land use in the Mathews Range area, Samburu-District, Kenya*. J. Cramer, Berlin.

Bussmann RW (1994). *The forests of Mount Kenya – vegetation, ecology, destruction and management of a tropical mountain forest ecosystem*. Universität Bayreuth.

Bussmann RW (2002). Islands in the desert – forest vegetation of Kenya’s smaller mountains and highland areas. *Journal of East African Natural History* 91: 27–79.

Fujiwara K, Furukawa T, Kiboi SK, Mathenge S, Mutiso P, Hayashi H, Meguro S (2014). Forest types and biodiversity around the Great Rift Valley in Kenya. *Contributii Botanice* 49: 143–178.

Schmitt K (1991). *The vegetation of the Aberdare National Park Kenya.* Universitätsverlag Wagner, Innsbruck.

Examples

```
summary(Kenya_veg)
```

`layers2samples` *Add information from slot 'layers' into slot 'samples'*

Description

Slot layers may include additional information that should be moved to samples in order to use it by `subset()`, `aggregate()` or `crosstable()` methods.

If names of variables are not provided, all variables from the respective layer table will be inserted in slot `samples`.

Usage

```
layers2samples(object, layer, variable, ...)

## S4 method for signature 'vegtable,character,character'
layers2samples(object, layer, variable, ...)

## S4 method for signature 'vegtable,character,missing'
layers2samples(object, layer, variable, ...)
```

Arguments

`object`	An object of class vegtable.
`layer`	Character value indicating a target layer.
`variable`	Character vector with the names of variables to be inserted in slot `samples`.
`...`	Further arguments to be passed among methods.

Value

An object of class vegtable with variables added to samples.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>.

`make_cocktail` *Produce a Cocktail classification*

Description

Classification of vegtable objects according to **Cocktail** algorithms.

Cocktail algorithms are logical functions selecting plots according to either occurrence of species groups and cover values of single species. A group will be declared as occurring in a plot when at least a half of its members is present in the plot.

This function inserts single columns with logical values indicating whether a plot is classified in the vegetation unit or not. An additional column (name provided in argument `syntax`) compile all vegetation units, indicating with a `+` symbol those plots classified in more than one vegetation unit. When only a part of the formulas will be used, it should be specified by the argument `which`.

These functions are implemented for constructing or complementing shaker objects. Note that construction of those objects will always require a `companion` object, which is either an object of class taxlist or vegtable.

Usage

```
set_group(shaker, companion, group, ...)

## S4 method for signature 'shaker,taxlist,character'
set_group(
  shaker,
  companion,
  group,
  group_id,
  authority = FALSE,
  enc_cont = "latin1",
  enc_gr = "utf8",
  ...
)

## S4 method for signature 'shaker,vegtable,character'
set_group(shaker, companion, group, ...)

set_pseudo(shaker, companion, pseudo, ...)

## S4 method for signature 'shaker,taxlist,character'
set_pseudo(
  shaker,
  companion,
  pseudo,
  pseudo_id,
  authority = FALSE,
  enc_cont = "latin1",
```

```
  enc_gr = "utf8",
  ...
)

## S4 method for signature 'shaker,vegtable,character'
set_pseudo(shaker, companion, pseudo, ...)

set_formula(shaker, companion, formula, ...)

## S4 method for signature 'shaker,taxlist,character'
set_formula(
  shaker,
  companion,
  formula,
  formula_id,
  authority = FALSE,
  enc_cont = "latin1",
  enc_gr = "utf8",
  ...
)

## S4 method for signature 'shaker,vegtable,character'
set_formula(shaker, companion, formula, ...)

make_cocktail(shaker, vegtable, ...)

## S4 method for signature 'shaker,vegtable'
make_cocktail(
  shaker,
  vegtable,
  which,
  cover,
  syntax = "Syntax",
  FUN = sum,
  in_header = TRUE,
  ...
)
```

Arguments

`shaker`	An object of class shaker containing the respective cocktail definitions.
`companion`	Either a taxlist or a vegtable object.
`...`	Further arguments passes from or to other methods.
`authority`	Logical value indicating whether author names should be included in the taxon name or not.
`enc_cont`, `enc_gr`	Encodings used for special characters.

`pseudo`, `group`	Character vector with names of taxa included in a pseudo-species or a species group.
`pseudo_id`, `group_id`, `formula_id`	Character value as name of the pseudo-species, species group or defined vegetation unit.
`formula`	Character vector including a formula as definition of a vegetation unit.
`vegtable`	An object of class vegtable containing the vegetation observations to be classified.
`which`	Integer or character indicating the definition to be applied for classification.
`cover`	Name of the cover variable in `vegtable`.
`syntax`	Character value indicating the name of the retrieved variable including the final classification of plots.
`FUN`	Function used for merging multiple occurrence of species in a single plot.
`in_header`	Logical value indicating whether results of Cocktail classification should be inserted to the header of the input vegtable or not. In the second case, a data frame is provided as output.

Value

A data frame corresponding to the slot `header` of input object `vegtable`, including the results of Cocktail classification for the respective plots.

A shaker object.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

References

Alvarez M (2017). Classification of aquatic and semi-aquatic vegetation in two East African sites: Cocktail definitions and syntaxonomy. *Phytocoenologia*.

Bruelheide H (2000). A new measure of fidelity and its application to defining species groups. *Journal of Vegetation Science* 11: 167–178.

Kočí M, Chytrý M, Tichý L (2003). Formalized reproduction of an expert-based phytosociological classification: a case study of subalpine tall-forb vegetation. *Journal of Vegetation Science* 14: 601–610.

See Also

shaker vegtable Wetlands

Examples

```
## Example from Alvarez (2017)
Wetlands_veg <- make_cocktail(Wetlands, Wetlands_veg, cover = "percen")
summary(as.factor(Wetlands_veg@header$Syntax))

## Same but only for two vegetation units
Wetlands_veg <- make_cocktail(Wetlands, Wetlands_veg,
  which = c("HY1", "HY2"), cover = "percen"
)
summary(as.factor(Wetlands_veg$Syntax))

## Construct the 'shaker' object anew
Wetlands <- new("shaker")

## Set a pseudo-species
Wetlands <- set_pseudo(Wetlands, Wetlands_veg, c(
  "Cyperus latifolius",
  "Cyperus exaltatus"
))

## Set a species group
Wetlands <- set_group(Wetlands, Wetlands_veg,
  group_id = "Cyperus papyrus",
  group = c(
    "Cyperus papyrus",
    "Cyclosorus interruptus",
    "Lepistemon owariense"
  )
)

## Set a fromula
Wetlands <- set_formula(Wetlands, Wetlands_veg,
  formula_id = "HE1",
  formula = "groups:'Cyperus papyrus' | species:'Cyperus papyrus > 50'"
)

## Summaries
summary(Wetlands)
summary(Wetlands, Wetlands_veg)
```

`names` *Retrieve names of vegtable and coverconvert objects*

Description

Quick access to column names in slot header and names of conversion codes.

These methods provide a quick display of the contents in coverconvert and vegtable objects.

Usage

```
## S4 method for signature 'vegtable'
names(x)

## S4 method for signature 'vegtable'
dimnames(x)

## S4 method for signature 'coverconvert'
names(x)

## S4 replacement method for signature 'coverconvert'
names(x) <- value
```

Arguments

`x`	An object of class coverconvert or vegtable.
`value`	A character vector used for replacement methods.

Value

A list containing the names from each slot.

Either a vector or a list (in the case of `dimnames()`) with the names of variables.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>.

Examples

```
names(Kenya_veg@coverconvert)
names(Kenya_veg)
dimnames(Kenya_veg)
```

`new_layer` *Insert new classification of vegetation layers*

Description

A new information table for vegetation layers.

Usage

```
new_layer(object, layer, ...)

## S4 method for signature 'vegtable,data.frame'
new_layer(object, layer, ...)
```

```
new_layer(object, ...) <- value

## S4 replacement method for signature 'vegtable,data.frame'
new_layer(object, ...) <- value

## S4 replacement method for signature 'vegtable,character'
new_layer(object, levels, ...) <- value
```

Arguments

`object`	A vegtable object.
`layer`	A data frame including information on vegetation layers.
`...`	Further arguments passed among methods.
`value`	Either a data frame or a character value. In the second case, this value indicates the name of the variable at slot samples that will be set as layer information.
`levels`	A character vector used to set the levels of the new layer. This is only used in the replacement method using a character value. This input is mandatory when the new layer does not exist in slot samples, otherwise an error message will be retrieved.

Value

A vegtable object with the inserted new relation.

See Also

`layers2samples()`

Examples

```
## Modify name in samples
names(Kenya_veg@samples) <- replace_x(names(Kenya_veg@samples),
    old = "LAYER", new = "layer")

## Add installed data frame to layers
new_layer(Kenya_veg) <- veg_layers

## Take a look in the result
summary(Kenya_veg@samples$layer)

## Do it with existing values
data(Kenya_veg)
new_layer(Kenya_veg) <- "LAYER"
summary(Kenya_veg@samples$LAYER)
```

`new_relation` *Insert a new variable as relation in vegtable object*

Description

Insert a new variable in slot **header** with a respective table at slot **relations**. The respective variable in header will be set as factor.

Existing categorical variables can also be set as relations. If such variables are factors, its levels can be preserved (missing argument in `'levels'`) or reset.

Usage

```
new_relation(object, ...)

## S3 method for class 'vegtable'
new_relation(object, relation, levels, ...)

new_relation(object, levels) <- value

## S4 replacement method for signature 'vegtable,character,character'
new_relation(object, levels) <- value

## S4 replacement method for signature 'vegtable,missing,character'
new_relation(object) <- value
```

Arguments

`object`	A vegtable object.
`...`	Further arguments passed among methods.
`relation, value`	A character value indicating the name of the new relation. The parameter 'value' is used for the replacement method
`levels`	A character vector with the levels for the inserted factor. This may be missing for variables that already exist in slot **header**.

Value

A vegtable object with the inserted new relation.

Examples

```
## A brand new variable
new_relation(Kenya_veg, levels = c("forest", "grassland", "cropland")) <- "land_use"

## Set an existing variable as relation
new_relation(Kenya_veg) <- "REMARKS"
```

`relation2header` *Insert variables from relations into header*

Description

Information associated to categories listed in slot **relations** can be inserted to slot **header** for further statistical comparisons.

Usage

```
relation2header(vegtable, ...)

## S3 method for class 'vegtable'
relation2header(vegtable, relation, vars, ...)
```

Arguments

`vegtable`	An vegtable object.
`...`	Further arguments passed among methods
`relation`	A character value indicating the relation to be used for inserting new variables in slot header.
`vars`	A selection of variables from the relation to be inserted in header. This function will check the existence of the variables in the respective relation and retrieve an error if none is matching the names. If missing in the arguments, all variables of the respective relation will be inserted.

Value

A vegtable object.

Author(s)

Miguel Alvarez, kamapu78@gmail.com

Examples

```
## Insert publication year of the source into header
veg <- relation2header(Kenya_veg, "REFERENCE", "YEAR")

## Show the frequency of plots per publication year
summary(as.factor(veg$YEAR))
```

`shaker-class` *Class containing Cocktail algorithms.*

Description

Objects used for collecting Cocktail definitions.

These objects work as **expert systems** for recognition of defined vegetation units among plots of a vegtable object. A `shaker` object will be always dependent on a vegtable object, which is called `companion`. Since modifications in the `companion` may affect the functionality of the `shaker` object, it will be recommended to create the last during a session by a source script instead of recycling them from old R images.

Slots

- `pseudos` List containing IDs of taxa that will be merged into pseudo-species.
- `groups` List containing IDs of taxa belonging to the same Cocktail group.
- `dominants` A data frame including lists of species used as dominant species in Cocktail algorithms, as well as operators and cover values used in the formulas.
- `formulas` List with formulas that will be used as definitions for vegetation units.

Author(s)

Miguel Alvarez `<kamapu78@gmail.com>`

See Also

`make_cocktail()` `set_pseudo()` `set_group()` `set_formula()`

Examples

```
showClass("shaker")
```

`subset` *Subset functions for vegtable objects*

Description

Produce subsets of vegtable objects.

Logical operations can be applied either to the plots, or the relations, which are the main slots in that class.

This method can be referred to the slot `species` the same way as `taxlist::subset()`, then the rest of the data will include only references to the subset of species list.

Usage

```
## S4 method for signature 'vegtable'
subset(
  x,
  subset,
  slot = "header",
  keep_children = FALSE,
  keep_parents = FALSE,
  relation,
  ...
)
```

Arguments

`x`	A vegtable object for subset.
`subset`	Logical expression for the subset.
`slot`	Character value indicating the slot used as reference for subset. At the moment only the values "taxonNames", "taxonRelations", "taxonTraits", "header", "samples", and "relations" are accepted. The three first values will be applied to the respective slots in the contained taxlist object (slot **species**).
`keep_children`	Argument passed to `taxlist::get_children()`.
`keep_parents`	Argument passed to `taxlist::get_parents()`.
`relation`	Character value indicating the relation (slot **relations**) to be used as reference for subset.
`...`	Further arguments passed from or to other methods.

Value

A S4 object of class vegtable.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Subset by taxon name
Kenya_sub <- subset(
  x = Kenya_veg, subset = TaxonName == "Tagetes",
  slot = "taxonNames", keep_children = TRUE, keep_parents = TRUE
)
summary(Kenya_sub)
summary(Kenya_sub@species)

## Subset by taxon relations
Kenya_sub <- subset(
  x = Kenya_veg, subset = Level == "species",
  slot = "taxonRelations"
```

```
)
summary(Kenya_sub)
summary(Kenya_sub@species)

## Subset by taxon traits
Kenya_sub <- subset(
  x = Kenya_veg, subset = lf_behn_2018 == "obligate_annual",
  slot = "taxonTraits"
)
summary(Kenya_sub)
summary(Kenya_sub@species)

## Subset by header
Kenya_sub <- subset(x = Kenya_veg, subset = ALTITUDE <= 1000, slot = "header")
summary(Kenya_sub)

## Subset by samples (after converting coverage)
Kenya_veg <- cover_trans(x = Kenya_veg, to = "cover_percentage", rule = "middle")
Kenya_sub <- subset(x = Kenya_veg, subset = cover_percentage >= 50, slot = "samples")
summary(Kenya_sub)

## Subset by relations
Kenya_sub <- subset(
  x = Kenya_veg, subset = as.integer(YEAR) >= 2000,
  slot = "relations", relation = "REFERENCE"
)
summary(Kenya_sub)
```

`summary` *Summary method for vegtable objects*

Description

Display summaries for vegtable objects.

Those methods are implemented for objects of the classes vegtable, coverconvert and shaker.

The method for class `vegtable` retrieves the metadata, the size of the object, its validity and additional statistics on the content of input object.

For objects of class shaker, the function `summary()` will either retrieve general statistics when `companion` is missing, or a more detailed display when accompanied by a taxlist or vegtable object.

Usage

```
## S4 method for signature 'vegtable'
summary(object, units = "Kb", ...)

## S4 method for signature 'coverconvert'
summary(object, ...)
```

```
## S4 method for signature 'shaker'
summary(object, companion, authority = FALSE, ...)

## S4 method for signature 'vegtable'
show(object)

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

## S4 method for signature 'coverconvert'
show(object)

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

## S4 method for signature 'shaker'
show(object)

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

Arguments

`object, x`	Object to be summarized.
`units`	Units used for object size (passed to `format()`).
`...`	further arguments to be passed to or from other methods.
`companion`	Companion object (either a taxlist or a vegtable object.
`authority`	Logical value indicating whether authors should be displayed or not.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Summary for 'vegtable' objects
summary(Wetlands_veg)
## Summary for 'coverconvert' objects
summary(braun_blanquet)
## Summary for 'shaker' objects (alone and with companion)
summary(Wetlands, Wetlands_veg)
```

`taxa2samples` *Insert taxon information into samples*

Description

For statistical purposes it may be necessary to insert information on recorded taxa into the slot samples, which contain the records of taxa in sampling plots. This can be also done selectivelly for specific taxonomic ranks and lower ranks can be aggregated to their parental ones.

If column **TaxonConceptID** is already existing in `'objec@samples'`, this column will get overwritten, retrieving a warning message.

Usage

```
taxa2samples(object, ...)

## S3 method for class 'vegtable'
taxa2samples(
  object,
  merge_to,
  include_levels,
  add_relations = FALSE,
  add_traits = FALSE,
  ...
)
```

Arguments

`object`	A vegtable object.
`...`	Further arguments passed among methods.
`merge_to`	Character value indicating the level (taxonomic rank) to which taxa of lower rank have to be merged.
`include_levels`	Character vector indicating the levels to be considered in the output object. This will set the values of **TaxonConceptID** and any respective values inserted from slots **taxonRelations** and **taxonTraits** as NA.
`add_relations`	A logical value indicating whether the content of slot **taxonRelations** have to be inserted in slot **samples** or not.
`add_traits`	A logical value indicating whether the content of slot **taxonTraits** have to be inserted in slot **samples** or not.

Value

An object of class vegtable.

Author(s)

Miguel Alvarez `<kamapu78@gmail.com>`

Examples

```
## Add only variable TaxonConceptID
veg <- taxa2samples(Kenya_veg)
head(veg@samples)

## Add also information from slots taxonRelations and taxonTraits
veg <- taxa2samples(Kenya_veg, add_relations = TRUE, add_traits = TRUE)
head(veg@samples)

## Different ranks recorded at samples
veg <- taxa2samples(Kenya_veg, add_relations = TRUE)
summary(veg@samples$Level)

## Aggregate taxa to family level
veg <- taxa2samples(Kenya_veg, add_relations = TRUE, merge_to = "family")
summary(veg@samples$Level)
```

`trait_stats` *Statistics and proportion for taxon traits*

Description

Calculation of statistics and proportions of taxon traits for plot observations or groups of observations, considering data relationships, taxonomic ranks and the handling of not available values.

In `trait_stats()` you can use customized functions, which have to be defined as `foo(x, w, ...)`, where `'x'` is the (numeric) taxon trait and `'w'` is the weight (e.g. the abundance).

With the arguments `taxon_levels` and `merge_to` the used taxonomic ranks can be defined, where the first one indicates which ranks have to be considered in the calculations and the second one determine the aggregation of taxa from a lower level to a parental one.

Usage

```
trait_stats(trait, object, ...)

## S4 method for signature 'character,vegtable'
trait_stats(
  trait,
  object,
  FUN,
  head_var = "ReleveID",
  taxon_levels,
  merge_to,
  weight,
  suffix = "_stats",
  in_header = TRUE,
  ...
```

```
)

## S4 method for signature 'formula,vegtable'
trait_stats(trait, object, ...)

trait_proportion(trait, object, ...)

## S4 method for signature 'character,vegtable'
trait_proportion(
  trait,
  object,
  head_var = "ReleveID",
  trait_levels,
  taxon_levels,
  merge_to,
  include_nas = TRUE,
  weight,
  suffix = "_prop",
  in_header = TRUE,
  ...
)

## S4 method for signature 'formula,vegtable'
trait_proportion(trait, object, ...)
```

Arguments

`trait`	Either a character value indicating the name of trait variable or a formula as `'trait ~ head_var'`. Note that you can add multiple variables in the form `trait_1 + ... + trait_n ~ head_var`.
`object`	A vegtable object.
`...`	Further arguments passed among methods. In the case of the character method, they are passed to 'FUN'.
`FUN`	A function usually defined as `foo(x)` or as `foo(x, w)` for weighted statistics.
`head_var`	Character value, the name of the variable at slot header to be used as aggregation level for the calculation of statistics or proportions. If not provided, the function will use **ReleveID** by default.
`taxon_levels`	Character vector indicating the selected taxonomic ranks to be considered in the output.
`merge_to`	Character value indicating the taxonomic rank for aggregation of taxa. All ranks lower than the one indicated here will be assigned to the respective parents at the required taxonomic rank.
`weight`	Character value indicating the name of the variable at slot **samples** used as weight for the proportions. Usually the numeric abundance.
`suffix`	A suffix added to the name of the trait variable or to the levels of categorical trait variables. I is meant to avoid homonymous variables within the same object.

`in_header`	Logical value indicating whether the output should be inserted in the slot **header** or provided as data frame. In the case that `'head_var'` (or the right term in the formula method) is different from **ReleveID**, the statistics and proportions will be inserted in the respective data frame at slot **relations**.
`trait_levels`	Character vector indicating a selection of levels from a trait, in the case that some levels should be ignored in the output. Trait levels that are skipped at output will be still used for the calculation of proportions. This argument gets only applied for the character method.
`include_nas`	Logical value indicating whether NAs should be considered for the calculation of proportions or not.

Value

A data frame with the proportions of traits levels or statistics for the trait variable, or an object of class vegtable including those results at the slot `header`.

Author(s)

Miguel Alvarez `<kamapu78@gmail.com>`

Examples

```
veg <- cover_trans(Kenya_veg, to = "cover")
veg <- trait_proportion("lf_behn_2018", veg,
  trait_levels = "obligate_annual", weight = "cover", include_nas = FALSE
)
summary(veg$obligate_annual_prop)
```

`tv2vegtable` *Import of vegetation data from Turboveg databases*

Description

Import function for **Turboveg** databases into an object of class vegtable. Most of the contents of **Turboveg** databases are included in DBF files and therefore imported by the function `foreign::read.dbf()`. The automatic setting of database path will be done by the function `vegdata::tv.home()` but it can be customised by the argument `tv_home`.

The species list will be imported by using the function `taxlist::tv2taxlist()` and therefore formatted as a taxlist object. Similarly, conversion tables will be handled as coverconvert objects.

Empty columns in the header will be deleted in the imported object.

The function `tv2coverconvert()` reads the content of cover conversion tables stored in **Turboveg** and attempts to reformat them in a more comprehensive structure.

This function is used by `tv2vegtable()` to import the respective conversion table from **Turboveg** databases. Note that conversion tables in **Turboveg** have only stored the middle point for each cover class in a scale, thus it will be recommended to rebuild the `coverconvert` slot or use braun_blanquet.

Usage

```
tv2vegtable(
  db,
  tv_home = tv.home(),
  skip_empty_relations = TRUE,
  skip_scale,
  clean = TRUE
)

tv2coverconvert(file, as.is = TRUE)
```

Arguments

`db`	Name of **Turboveg** data base as character value.
`tv_home`	**Turboveg** installation path as character value.
`skip_empty_relations`	Logical value indicating whether empty relations may be excluded from imported database or not.
`skip_scale`	Character value indicating scales to be excluded in slot `coverconvert`.
`clean`	Logical value indicating whether output object should be cleaned or not.
`file`	A connection to a DBF file containing conversion table in **Turboveg**.
`as.is`	A logical value passed to `read.dbf()`.

Value

A vegtable object in the case of `tv2vegtable()`. A coverconvert object in the case of `tv2coverconvert()`.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

See Also

`taxlist::tv2taxlist()` `foreign::read.dbf()` `vegdata::tv.home()`

Examples

```
## Installed 'Turboveg' version of 'Fujiwara et al. (2014)'
# TV_Home <- file.path(path.package("vegtable"), "tv_data")
# Veg <- tv2vegtable("Fujiwara_2014", TV_Home)
# summary(Veg)
## Installed 'Turboveg' version of "Fujiwara et al. (2014)"
TV_Home <- file.path(path.package("vegtable"), "tv_data", "popup", "Swea")
Table <- tv2coverconvert(file.path(TV_Home, "tvscale.dbf"))

## First scale have to be deleted from conversion table
Table@value <- Table@value[-1]
Table@conversion <- Table@conversion[-1]
```

```
summary(Table)

## Compare the 'Turboveg' version with a vegtable version
data(braun_blanquet)
summary(Table$br_bl)
summary(braun_blanquet$br_bl)
```

`update_det` *Update by determined specimens*

Description

Reference specimens can be integrated in slot **layers** within a vegtable object. Updated entries in the specimens can be updated in slot **samples** by using this function. Alternatively expert opinions can be inserted and applied in case of disagreement with the original records.

Usage

```
update_det(x, specimens, ...)

## S4 method for signature 'vegtable,character'
update_det(x, specimens, ...)
```

Arguments

`x`	A vegtable object to be updated.
`specimens`	A character vector indicating the names of tables included in slot **layers** with updates to be applied. Note that they will be applied in the same order of the vector in the case of multiple updates.
`...`	Further arguments (not yet in use).

`used_synonyms` *Retrieve synonyms or taxon concepts used in a data set*

Description

Plots records are rather linked to plant names than plant taxon concepts and `used_synonyms()` lists all synonyms linked to records in a vegtable object, including their respective accepted names.

On the other side, the function `used_concepts()` produces a subset of the taxonomic list embeded in the slot **species** including only taxonomic concepts linked to records in the slot **samples**.

Usage

```
used_synonyms(x, ...)

## S3 method for class 'vegtable'
used_synonyms(x, ...)

used_concepts(x, ...)

## S3 method for class 'vegtable'
used_concepts(
  x,
  keep_children = FALSE,
  keep_parents = FALSE,
  keep_synonyms = TRUE,
  ...
)
```

Arguments

`x`	A vegtable object.
`...`	Further arguments to be passed from or to another methods.
`keep_children`	A logical argument indicating whether children of selected taxa should be included in the output or not. This argument passed to `get_children()`.
`keep_parents`	A logical value indicating whether parents of selected taxa should be included in the output or not. This argument passed to `get_parents()`.
`keep_synonyms`	A logical value indicating whether synonyms should be included or not.

Value

The function `used_synonyms()` returns a data frame including following variables:

SynonymID ID of the taxon usage name applied as synonym.

Synonym The synonym itself.

SynonymAuthor Author of synonym.

TaxonConceptID ID of the respective taxon concept.

AcceptedNameID ID of the taxon usage name set as accepted name of the taxon concept.

AcceptedName The respective accepted name.

AcceptedNameAuthor The author of the accepted name.

The function `used_concepts()` returns a taxlist object including only taxa occurring in the plot observations of the input vegtable object.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

See Also

`accepted_name()`

Examples

```
## Synonyms used in the Kenya_veg
Synonyms <- used_synonyms(Kenya_veg)
head(Synonyms)

## Subset species list to used concepts
species <- used_concepts(Kenya_veg)
Kenya_veg@species
species
```

`vegtable-class` *Class vegtable.*

Description

Class holding vegetation-plot data sets. Designed to content all information stored in **Turboveg** databases in just one object.

This class was designed to include information of relevés, header data and species in just one object. Objects can be created by calls of the form `new("vegtable", ...)`.

Slots

`description`	A named character vector containing metadata.
`samples`	A data frame with samples list.
`header`	A data frame with plots data.
`species`	Species list as a taxlist object.
`layers`	A list including strata within samples as data frames.
`relations`	A list including popup lists as data frames.
`coverconvert`	A scale conversion object of class coverconvert.
`syntax`	A list including syntaxonomic lists either as data frames or as taxlist objects.

See Also

`tv2vegtable()`

Examples

```
showClass("vegtable")
```

`vegtable_stat` *General statistics from vegtable objects*

Description

This function calculates general statistics of local **Turboveg** databases as required by GIVD (Global Index of Vegetation-Plot Databases, `https://www.givd.info`).

This function is based on a script delivered by GIVD for summarising statistics required in the descriptions of databases (see meta data in the page of the Global Index for Vegetation-Plot Databases).

Usage

```
vegtable_stat(vegtable, ...)

## S3 method for class 'vegtable'
vegtable_stat(vegtable, ...)
```

Arguments

`vegtable`	An object of class vegtable.
`...`	Further arguments passed among methods.

Author(s)

GIVD. Adapted by Miguel Alvarez <kamapu78@gmail.com>

Examples

```
## Statistics for GIVD
vegtable_stat(Kenya_veg)
```

`veg_aggregate` *Aggregating information into a data frame*

Description

Compute summarizing tables from vegtable objects. This function works in a similar way as `crosstable()`.

Usage

```
veg_aggregate(object, data, FUN, ...)

## S4 method for signature 'formula,vegtable,function'
veg_aggregate(object, data, FUN, use_nas = TRUE, ...)
```

Arguments

`object`	A formula indicating the variables used for the summary. As in `crosstable()`, the keywords `"TaxonName"` and `"AcceptedName"` can be used to retrieve taxonomic names, where the second will set the accepted name for names considered as synonyms.
`data`	Either a data frame or an object of class vegtable.
`FUN`	Function used to aggregate values.
`...`	Further arguments passed to the function `stats::aggregate()`.
`use_nas`	Logical value indicating whether NA's should be included in categorical variables or not.

Value

An object of class data.frame.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

See Also

`aggregate()`

Examples

```
## Transform cover to percentage cover
veg <- cover_trans(x = Kenya_veg, to = "cover")

## Frequency of taxa per publication
atab <- veg_aggregate(object = cover ~ AcceptedName + REFERENCE, data = veg, FUN = length)
head(atab)

## Life form proportions per plot
atab <- veg_aggregate(object = cover ~ lf_behn_2018 + ReleveID, data = veg, FUN = sum)
head(atab)
```

`veg_diverstiy` *Calculation of statistics at plot level*

Description

Calculation of diversity statistics at the plot level allowing for customized functions defined as `foo(x, ...)`, where `x` is the vector of abundance values.

This function calls `taxa2samples()` to derive taxa from taxon usage names in slot **samples** and multiple records of species in a single plot will be merged by stats::aggregate.

The functions `shannon()`, `evenness()`, and `dominance()` calculate the diversity index of Shannon, the evenness, and the dominance (*1 - evenness*), respectively. Dominance is the complementary value to evenness (i.e. `1 - evenness`).

The function `simpson()` calculates the Simpson's index using the alternative for vegetation plot observations.

The function `richness()` counts the number of taxa per plot and can be used as alternative to vegtable::count_taxa.

Usage

```
shannon(x, na.rm = TRUE, ...)

evenness(x, ...)

dominance(x, ...)

simpson(x, na.rm = TRUE, ...)

richness(x, na.rm = TRUE, ...)

veg_diversity(object, ...)

## S3 method for class 'vegtable'
veg_diversity(
  object,
  weight,
  FUN = shannon,
  aggr_fun = mean,
  arg_fun = list(),
  var_name,
  in_header = TRUE,
  ...
)
```

Arguments

`x`	A numeric vector containing the abundance of single species.
`na.rm`	A logical value indicating whether NA values should be removed from the abundance vector or not.
`...`	Further arguments passed among methods. In `'evenness()'` and `'dominance()'`, these arguments are passed to `'shannon()'`. In `'veg_diversity()'`, these arguments are passed to `aggregate()` (actually to `'FUN'`).

`object`	A vegtable object.
`weight`	A character value indicating the name of the column at slot **samples** which will be used as species abundance.
`FUN`	A function used to calculate the diversity index.
`aggr_fun`	A function used to aggregate abundance values for multiple records of a taxon in a plot observation. Average value is used by default.
`arg_fun`	A named list with parameters and arguments passed to `taxa2samples()`, which will retrieve the respective taxon concept for each taxon usage name and can be used to merge taxa at a determined taxonomic rank, for instance to merge all sub-specific taxa into their respective species (i.e. `'merge_to = "species"'`).
`var_name`	A character value used as name for the calculated index. If missing, the name of the function will be used.
`in_header`	A logical value indicating whether the results should be included as variables in the slot **header** of the input object. If `'in_header = TRUE'`, you may assign the result of the function to the input object.

Value

Functions `shannon()`, `evenness()`, `dominance()`, `simpson()`, and `richness()` return a numeric value (the calculated index).

Funtion `veg_diversity()` produce either a data frame with calculated values per plot observation (option `'in_header = FALSE'`) or a vegtable object with the calculated values inserted in the slot **header** (option `'in_header = TRUE'`).

Examples

```
## Compare Evenness with Shannon index
Kenya_veg <- cover_trans(x = Kenya_veg, to = "cover")
Kenya_veg <- veg_diversity(object = Kenya_veg, weight = "cover")
Kenya_veg <- veg_diversity(object = Kenya_veg, weight = "cover", FUN = evenness)

with(Kenya_veg@header, plot(shannon, evenness))
```

`veg_layers-data` *Information on vegetation layers*

Description

A data frame with a standard classification on vegetation layers.

Usage

```
veg_layers
```

Format

A data frame.

Examples

```
summary(veg_layers)
```

`veg_relation` *Retrieve or replace relations in vegtable objects*

Description

Tables providing information about levels of categorical variables in the header are called popups in **Turboveg** databases but `relations` in vegtable objects. Such variables will be converted into factors in the slot `header` according to the levels and their sorting in the respective relation.

Usage

```
veg_relation(vegtable, relation, ...)

## S4 method for signature 'vegtable,character'
veg_relation(vegtable, relation, match_header = FALSE, ...)

veg_relation(vegtable) <- value

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

Arguments

`vegtable`	An object of class vegtable.
`relation`	A character value indicating the relation table to be retrieved or replaced.
`...`	Further arguments to be passed among methods.
`match_header`	A logical vector, whether only levels occurring in slot `header` should be considered or all.
`value`	A data frame containing the new veg_relation.

Value

This function retrieves and object of class `data.frame`. In the replacement method, an object of class vegtable, including `value` in the slot `relations`.

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## overview of references
veg_relation(Kenya_veg, "REFERENCE")
```

`Wetlands-data` *Vegetation-plots from Tanzania*

Description

A subset of `http://www.givd.info/ID/AF-00-006SWEA-Dataveg` with plots sampled in Tanzania.

Usage

```
Wetlands
```

Format

An object of class shaker (`Wetlands`) and the respective companion as vegtable object (`Wetlands_veg`).

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Source

`http://www.givd.info/ID/AF-00-006`.

References

Alvarez M (2017). Classification of aquatic and semi-aquatic vegetation in two East African sites: Cocktail definitions and syntaxonomy. *Phytocoenologia*.

Examples

```
summary(Wetlands)
summary(Wetlands_veg)
```

`write_juice` *Exporting tables for Juice*

Description

This function produce txt files as inport formats for **Juice** (https://www.sci.muni.cz/botany/juice/).

This function produces two output files to be imported into a **Juice** file: A vegetation table produced by `crosstable()` and a header table. Both tables share the file name plus a suffix (`table` for the vegetation table and `header` for the header).

For the import in **Juice**, you go to the menu `File -> Import -> Table -> from Spreadsheet File (e.g. EXCEL Table)` and then follow the wizard. Do not forget to select the proper settings in the wizard: 1) 'Character delimiting columns: Comma' (for default argument values). 2) 'Use the second column as layer information: Unchecked'. 3) 'Cover values: Percentage Values'.

To further import the header table you need to go to the menu `File -> Import -> Header Data -> From Comma Delimited`

In the header (see **Value**), the first column (`Table number`) corresponds to the plot number assigned by **Juice** at import, while the column (`Releve number`) is the number originally assigned to the plot (e.g. **Turboveg** ID).

Usage

```
write_juice(data, file, formula, ...)

## S4 method for signature 'vegtable,character,formula'
write_juice(
  data,
  file,
  formula,
  FUN,
  db_name = "Plot Observations",
  header,
  coords,
  sep = ",",
  ...
)

read_juice(file, encoding = "LATIN-1", sep = ";", na = "", ...)
```

Arguments

`data`	An object of class vegtable.
`file`	Character value indicating the name of output files (without file extension).
`formula`	A formula passed to `crosstable()`.
`...`	Further arguments. While `write_juice()` passes them to the function `crosstable()`, `read_juice()` passes those arguments to `readLines()`.

`FUN`	Funtion passed to `crosstable()`.
`db_name`	Name for data set displayed in inport wizard.
`header`	Variables of header to be exported.
`coords`	Names of coordinate variables in header of `data`.
`sep`	Separator used to split rows into columns.
`encoding`	Argument passed to `readLines`.
`na`	Character used as not available values.

Value

For `read_juice()`, a list with two elements: A data frame of species by plot (`cross_table`), and a data frame with header data (`header`).

Author(s)

Miguel Alvarez <`kamapu78@gmail.com`>

Examples

```
## Subset and transform cover values to percentage
vegetation <- Kenya_veg[1:20, ]
vegetation <- cover_trans(x = vegetation, to = "cover_percent", rule = "middle")

## Write in tempdir
write_juice(data = vegetation, file = file.path(tempdir(), "SWEA"),
    formula = cover_percent ~ ReleveID + AcceptedName, FUN = mean,
    header = c("ReleveID", "COMM_TYPE"))
## Installed 'Juice' version of 'Wetlands_veg'
Veg <- file.path(path.package("vegtable"), "juice", "Wetlands_juice.txt")
Veg <- read_juice(Veg)

summary(Veg)
```

Index