

Package ‘ecan’

October 5, 2026

Title Ecological Analysis and Visualization

Version 0.3.0

Description Support ecological analyses such as ordination and clustering.

Contains consistent and easy wrapper functions of 'stat', 'vegan', and 'labdsv' packages, and visualisation functions of ordination and clustering.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5.0)

URL <https://github.com/matutosi/ecan>,
<https://github.com/matutosi/ecan/tree/develop>,
<https://matutosi.github.io/ecan/>

BugReports <https://github.com/matutosi/ecan/issues>

Imports MASS, cluster, dendextend, dplyr, ggplot2, jsonlite, labdsv,
magrittr, purrr, rlang, stringr, tibble, tidyr, vegan

Suggests ggdendro, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Toshikazu Matsumura [aut, cre]

Maintainer Toshikazu Matsumura <matutosi@gmail.com>

Repository CRAN

Date/Publication 2026-10-05 02:30:02 UTC

Contents

cluster	2
d	4
df2table	4

draw_layer_construction	5
gen_example	6
ind_val	8
is_one2multi	9
ordination	10
pad2longest	11
pseudospecies	12
read_biss	12
shdi	13
twinspan	14
tw_downweight	16
tw_hill_const	17
tw_inertia	18
tw_preference	18
tw_ra	19
tw_species_data	20
tw_two_way	21
Index	22

cluster	<i>Helper function for clustering methods</i>
---------	---

Description

- Helper function for clustering methods
- Helper function for calculate distance
- Add group names to hclust labels.
- Add colors to dendrogram

Usage

```
cluster(x, c_method, d_method)

distance(x, d_method)

cls_add_group(cls, df, indiv, group, pad = TRUE)

cls_color(cls, df, indiv, group)
```

Arguments

- x A community data matrix. rownames: stands. colnames: species.
- c_method A string of clustering method. "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC), "centroid" (= UPGMC), or "diana".

d_method	A string of distance method. "correlation", "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao", "mahalanobis", "chisq", "chord", "aitchison", or "robust.aitchison".
cls	A result of cluster or dendrogram.
df	A data.frame to be added into ord scores
indiv, group	A string to specify individual and group name of column in df.
pad	A logical to specify padding strings.

Value

cluster() returns result of clustering. \$clustering_method: c_method \$distance_method: d_method

distance() returns distance matrix.

Inputing cls return a color vector,
inputing dend return a dend with color.

Examples

```
library(dplyr)
df <-
  tibble::tibble(
    stand = paste0("ST_", c("A", "A", "A", "B", "B", "C", "C", "C", "C")),
    species = paste0("sp_", c("a", "e", "d", "e", "b", "e", "d", "b", "a")),
    abundance = c(3, 3, 1, 9, 5, 4, 3, 3, 1))
cls <-
  df2table(df) %>%
  cluster(c_method = "average", d_method = "bray")
library(ggdendro)
# show standard cluster
ggdendro::ggdendrogram(cls)

# show cluster with group
data(dune, package = "vegan")
data(dune.env, package = "vegan")
cls <-
  cluster(dune, c_method = "average", d_method = "bray")
df <- tibble::rownames_to_column(dune.env, "stand")
cls <- cls_add_group(cls, df, indiv = "stand", group = "Use")
ggdendro::ggdendrogram(cls)
```

d	<i>Calculating diversity</i>
---	------------------------------

Description

Calculating diversity

Usage

d(x)

h(x, base = exp(1))

Arguments

x, base A numeric vector.

Value

A numeric vector.

df2table	<i>Convert data.frame and table to each other.</i>
----------	--

Description

Convert data.frame and table to each other.

Usage

df2table(df, st = "stand", sp = "species", ab = "abundance")

table2df(tbl, st = "stand", sp = "species", ab = "abundance")

dist2df(dist)

Arguments

df A data.frame.

st, sp, ab A string.

tbl A table. community matrix. rownames: stands. colnames: species.

dist A distance table.

Value

df2table() return table,
table2df() return data.frame,
dist2df() return data.frame.

Examples

```
tibble::tibble(  
  st = paste0("st_", rep(1:2, times = 2)),  
  sp = paste0("sp_", rep(1:2, each = 2)),  
  ab = runif(4)) %>%  
dplyr::bind_rows(., .) %>%  
print() %>%  
df2table("st", "sp", "ab")
```

draw_layer_construction

Draw layer construction plot

Description

Draw layer construction plot

Add mid point and bin width of layer heights.

Compute mid point of layer heights.

Compute bin width of layer heights.

Usage

```
draw_layer_construction(  
  df,  
  stand = "stand",  
  height = "height",  
  cover = "cover",  
  group = "",  
  ...  
)  
  
add_mid_p_bin_w(df, height = "height")  
  
mid_point(x)  
  
bin_width(x)
```

Arguments

df	A dataframe including columns: stand, layer height and cover. Optional column: stand group.
stand, height, cover, group	A string to specify stand, height, cover, group column.
...	Extra arguments for geom_bar().
x	A numeric vector.

Value

draw_layer_construction() returns gg object, add_mid_p_bin_w() returns dataframe including mid_point and bin_width columns. mid_point() and bin_width() return a numeric vector.

Examples

```
library(dplyr)
n <- 10
height_max <- 20
ly_list <- c("B", "S", "K")
st_list <- LETTERS[1]
sp_list <- letters[1:9]
st_group <- NULL
sp_group <- rep(letters[24:26], 3)
cover_list <- 2^(0:4)
df <- gen_example(n = n, use_layer = TRUE,
                  height_max = height_max, ly_list = ly_list,
                  st_list = st_list, sp_list = sp_list,
                  st_group = st_group, sp_group = sp_group,
                  cover_list = cover_list)

# select stand and summarise by sp_group
df %>%
  dplyr::group_by(height, sp_group) %>%
  dplyr::summarise(cover = sum(cover), .groups = "drop") %>%
  draw_layer_construction(group = "sp_group", colour = "white")
```

gen_example

Generate vegetation example

Description

Stand, species, and cover are basic. Layer, height, st_group, are sp_group optional.

Usage

```
gen_example(
  n = 300,
  use_layer = TRUE,
  height_max = 20,
  ly_list = "",
  st_list = LETTERS[1:9],
  sp_list = letters[1:9],
  st_group = NULL,
  sp_group = NULL,
  cover_list = 2^(0:6)
)
```

Arguments

<code>n</code>	A numeric to generate no of occurrences.
<code>use_layer</code>	A logical. If FALSE, <code>height_max</code> and <code>ly_list</code> will be omitted.
<code>height_max</code>	A numeric. The highest layer of samples.
<code>ly_list, st_list, sp_list, st_group, sp_group</code>	A string vector. <code>st_group</code> and <code>sp_group</code> are optional (default is NULL). Length of <code>st_list</code> and <code>sp_list</code> should be the same as <code>st_group</code> and <code>sp_group</code> , respectively.
<code>cover_list</code>	A numeric vector.

Value

A dataframe with columns: stand, layer, species, cover, `st_group` and `sp_group`.

Examples

```
n <- 300
height_max <- 20
ly_list <- c("B1", "B2", "S1", "S2", "K")
st_list <- LETTERS[1:9]
sp_list <- letters[1:9]
st_group <- rep(LETTERS[24:26], 3)
sp_group <- rep(letters[24:26], 3)
cover_list <- 2^(0:6)
gen_example(n = n, use_layer = TRUE,
            height_max = height_max, ly_list = ly_list,
            st_list = st_list, sp_list = sp_list,
            st_group = st_group, sp_group = sp_group,
            cover_list = cover_list)
```

ind_val

*Helper function for Indicator Species Analysis***Description**

Calculating diversity indices such as species richness (s), Shannon's H' (h), Simpson's D (d), Simpson's inverse D (i).

Usage

```
ind_val(
  df,
  stand = NULL,
  species = NULL,
  abundance = NULL,
  group = NULL,
  row_data = FALSE
)
```

Arguments

df	A data.frame, which has three cols: stand, species, abundance. Community matrix should be converted using table2df().
stand, species, abundance	A text to specify each column. If NULL, 1st, 2nd, 3rd column will be used.
group	A text to specify group column. The groups are given in the order of the levels when it is a factor, otherwise in the order of sort().
row_data	A logical. TRUE: return row result data of labdsv::indval().

Value

A data.frame.

Examples

```
library(dplyr)
library(tibble)
data(dune, package = "vegan")
data(dune.env, package = "vegan")
df <-
  dune %>%
  table2df(st = "stand", sp = "species", ab = "cover") %>%
  dplyr::left_join(tibble::rownames_to_column(dune.env, "stand"))
ind_val(df, abundance = "cover", group = "Moisture")
```

is_one2multi	<i>Check cols one-to-one, or one-to-multi in data.frame</i>
--------------	---

Description

Check cols one-to-one, or one-to-multi in data.frame

Usage

```
is_one2multi(df, col_1, col_2)

is_one2one(df, col_1, col_2)

is_multi2multi(df, col_1, col_2)

cols_one2multi(df, col, include_self = TRUE, inculde_self)

select_one2multi(df, col, include_self = TRUE, inculde_self)

unique_length(df, col_1, col_2)
```

Arguments

df	A data.frame
col, col_1, col_2	A string to specify a colname.
include_self	A logical. If TRUE, return value including input col.
inculde_self	Deprecated: the misspelt name of include_self, kept so that older code keeps working.

Value

is_one2multi(), is_one2one(), is_multi2multi() return a logical. cols_one2multi() returns strings of colnames that has one2multi relation to input col. unique_length() returns a list.

Examples

```
df <- tibble::tibble(
  x      = rep(letters[1:6], each = 1),
  x_grp  = rep(letters[1:3], each = 2),
  y      = rep(LETTERS[1:3], each = 2),
  y_grp  = rep(LETTERS[1:3], each = 2),
  z      = rep(LETTERS[1:3], each = 2),
  z_grp  = rep(LETTERS[1:3], times = 2))

unique_length(df, "x", "x_grp")

is_one2one(df, "x", "x_grp")
```

```
is_one2one(df, "y", "y_grp")
is_one2one(df, "z", "z_grp")
```

ordination	<i>Helper function for ordination methods</i>
------------	---

Description

Helper function for ordination methods

Usage

```
ordination(tbl, o_method, d_method = NULL, ...)

ord_plot(ord, score = "st_scores", x = 1, y = 2)

ord_add_group(ord, score = "st_scores", df, indiv, group)

ord_extract_score(ord, score = "st_scores", row_name = NULL)
```

Arguments

<code>tbl</code>	A community data matrix. rownames: stands. colnames: species.
<code>o_method</code>	A string of ordination method. "pca", "ca", "dca", "pcoa", or "nmds". "fspa" is removed, because package dave was archived.
<code>d_method</code>	A string of distance method. "correlation", "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao", "mahalanobis", "chisq", "chord", "aitchison", or "robust.aitchison".
<code>...</code>	Other parameters for PCA.
<code>ord</code>	A result of ordination().
<code>score</code>	A string to specify score for plot. "st_scores" means stands and "sp_scores" species.
<code>x, y</code>	A column number for x and y axis.
<code>df</code>	A data.frame to be added into ord scores
<code>indiv, group, row_name</code>	A string to specify indiv, group, row_name column in df.

Value

`ordination()` returns result of ordination. `$st_scores`: scores for stand. `$sp_scores`: scores for species. `$eig_val`: eigen value for stand. `$results_raw`: results of original ordination function. `$ordination_method`: `o_method`. `$distance_method`: `d_method`. `ord_plot()` returns ggplot2 object. `ord_extract_score()` extracts stand or species scores from ordination result. `ord_add_group()` adds group data.frame into ordination scores. The columns that are one-to-multi to indiv are added, and the column named by group is always among them.

Examples

```

library(ggplot2)
library(vegan)
data(dune)
data(dune.env)

df <-
  table2df(dune) %>%
  dplyr::left_join(tibble::rownames_to_column(dune.env, "stand"))
sp_dammy <-
  tibble::tibble("species" = colnames(dune),
                 "dammy_1" = stringr::str_sub(colnames(dune), 1, 1),
                 "dammy_6" = stringr::str_sub(colnames(dune), 6, 6))
df <- dplyr::left_join(df, sp_dammy)

ord_dca <- ordination(dune, o_method = "dca")
ord_pca <-
  df %>%
  df2table() %>%
  ordination(o_method = "pca")

ord_dca_st <-
  ord_extract_score(ord_dca, score = "st_scores")
ord_pca_sp <-
  ord_add_group(ord_pca,
                score = "sp_scores", df, indiv = "species", group = "dammy_1")

```

pad2longest

Pad a string to the longest width of the strings.

Description

Pad a string to the longest width of the strings.

Usage

```
pad2longest(string, side = "right", pad = " ")
```

Arguments

string	Strings.
side	Side on which padding character is added (left, right or both).
pad	Single padding character (default is spaces).

Value

Strings.

Examples

```
x <- c("a", "ab", "abc")
pad2longest(x, side = "right", pad = " ")
```

pseudospecies	<i>Pseudospecies transformation</i>
---------------	-------------------------------------

Description

Expands each species into binary pseudospecies by cut levels. A pseudospecies of a cut level is present when the abundance is larger than zero and not less than the cut level. Pseudospecies that occur in no stand are dropped.

Usage

```
pseudospecies(x, cut_levels = c(0, 2, 5, 10, 20))
```

Arguments

x	A community data matrix or data.frame. rownames: stands, colnames: species.
cut_levels	A numeric vector of pseudospecies cut levels.

Value

A binary matrix of stands by pseudospecies with "species", "level" and "cut_levels" attributes.

Examples

```
data(dune, package = "vegan")
psp <- pseudospecies(dune)
dim(psp)
head(colnames(psp))
```

read_biss	<i>Read data from BiSS (Biodiversity Investigation Support System) to data frame.</i>
-----------	---

Description

BiSS data is formatted as JSON.

Usage

```
read_biss(txt, join = TRUE)
```

Arguments

txt A JSON string, URL or file.

join A logical. TRUE: join plot and occurrence, FALSE: do not join.

Value

A data frame.

Examples

```
path <- system.file("extdata", "biss_example.json", package = "ecan")
read_biss(path)
```

shdi	<i>Helper function for calculating diversity</i>
------	--

Description

Calculating diversity indices such as species richness (s), Shannon's H' (h), Simpson's D (d), Simpson's inverse D (i).

Usage

```
shdi(df, stand = NULL, species = NULL, abundance = NULL)
```

Arguments

df A data.frame, which has three cols: stand, species, abundance. Community matrix should be converted using table2df().

stand, species, abundance A text to specify each column. If NULL, 1st, 2nd, 3rd column will be used.

Value

A data.frame. Including species richness (s), Shannon's H' (h), Simpson's D (d), Simpson's inverse D (i).

Examples

```
data(dune, package = "vegan")
df <- table2df(dune)
shdi(df)
```

twinspan

*Two-way indicator species analysis (TWINSpan)***Description**

A native R implementation of TWINSpan (Hill 1979) and modified TWINSpan (Roleček et al. 2009). The algorithm divides stands (samples) hierarchically by the first axis of a correspondence analysis (reciprocal averaging) of pseudospecies, refines the division with differential species, and summarises it with a small set of indicator pseudospecies.

Usage

```
twinspan(
  x,
  cut_levels = c(0, 2, 5, 10, 20),
  min_size = 5,
  max_depth = 6,
  max_indicators = 7,
  diff_threshold = 1/3,
  refine_iter = 5,
  modified = FALSE,
  n_clusters = NULL,
  use_indicator = FALSE,
  downweight = TRUE,
  polish = c("hill", "ecan"),
  species = TRUE
)
```

```
## S3 method for class 'twinspan'
as.hclust(x, ...)
```

```
## S3 method for class 'twinspan'
print(x, ...)
```

Arguments

<code>x</code>	A community data matrix or data.frame. rownames: stands, colnames: species.
<code>cut_levels</code>	A numeric vector of pseudospecies cut levels.
<code>min_size</code>	An integer. Groups smaller than this are not divided.
<code>max_depth</code>	An integer of the maximum number of division levels. The default (6) is the same as the original TWINSpan (its levmax).
<code>max_indicators</code>	An integer of the maximum number of indicator pseudospecies used to summarise a division.
<code>diff_threshold</code>	A numeric in (0, 1]. A pseudospecies is a differential species when the absolute value of its preference is not less than this. The default (1/3) corresponds to a 2:1 frequency ratio.

<code>refine_iter</code>	An integer of the maximum number of refinement steps.
<code>modified</code>	A logical. TRUE: modified TWINSpan. The most heterogeneous group is divided first.
<code>n_clusters</code>	An integer of the number of groups to stop at, or NULL for no limit.
<code>use_indicator</code>	A logical. TRUE: use the indicator ordination for the final division (as in the original TWINSpan). FALSE: use the refined ordination.
<code>downweight</code>	A logical. TRUE (the default, as in the original TWINSpan): downweight the rare pseudospecies in the ordination, in the way of <code>decorana()</code> . See <code>tw_downweight()</code> .
<code>polish</code>	A string. "hill" (the default): divide in the way of the original TWINSpan. "ecan": the earlier way of this package. <code>diff_threshold</code> , <code>refine_iter</code> and <code>use_indicator</code> are used only by "ecan".
<code>species</code>	A logical. TRUE: classify pseudospecies as well as stands, which is needed for <code>tw_two_way()</code> .
<code>...</code>	Ignored.

Details

The package is written in plain R and needs no compiler. It is not a port of Hill's FORTRAN program, but `polish = "hill"` (the default) follows the steps of that program: the rare pseudospecies are downweighted as `WEIGHT` does, the axis is polished twice as `POLISH` does, the stands are divided at the middle of the range of the polished axis, and the stands of the critical zone around that point are placed by the indicator pseudospecies, whose number and threshold are the ones that misclassify fewest stands. The constants of the original are in `tw_hill_const()`.

The two halves of a division are put in the order of the original as well: the half that resembles the group next to the one being divided comes first, so that neighbouring groups stay together.

On the dune, sipoo, varespec, mite, BCI and pyrifos data of `vegan` this reproduces the original program exactly: the same groups with the same numbers, the same divisions and the same eigenvalues. It does so on twenty randomly generated data sets as well. The species are classified in the way of the original as well, that is on how faithful each of them is to the groups of stands rather than on the pseudospecies table itself, and without indicators: see `tw_species_data()`. The species groups are those of the original too.

`polish = "ecan"` keeps the earlier way of this package, which was written from the published description alone: the division is refined with the pseudospecies whose preference reaches `diff_threshold`, and the stands are divided at the centroid of the axis. It is kept because it needs no zone or indicator to place a stand, but it does not follow the original as closely.

If the results of the original program are needed, the `twinspan` package of Oksanen (<https://github.com/jarioksa/twinspan>, MIT licensed) calls Hill's FORTRAN code itself.

Value

`twinspan()` returns a list with class "twinspan". `$classification`: a tibble of stand, group, path and depth. `$species_classification`: a tibble of species, group, path and depth (of pseudospecies when `polish = "ecan"`). `$nodes`: a list of the nodes of the division tree. `$pseudospecies`: the pseudospecies matrix. `$call`, and the parameters above.

`as.hclust()` returns an "hclust" object so that `cls_color()`, `cls_add_group()` and `ggdendro::ggdendrogram()` can be used.

print() returns the object invisibly.

References

Hill, M.O. (1979) TWINSpan: a FORTRAN program for arranging multivariate data in an ordered two-way table by classification of the individuals and attributes. Cornell University, Ithaca.

Roleček, J., Tichý, L., Zelený, D. and Chytrý, M. (2009) Modified TWINSpan classification in which the hierarchy respects cluster heterogeneity. *Journal of Vegetation Science* 20: 596-602.

Examples

```
data(dune, package = "vegan")
tw <- twinspan(dune)
tw
tw$classification

# modified TWINSpan with a fixed number of groups
tw_mod <- twinspan(dune, modified = TRUE, n_clusters = 4)
table(tw_mod$classification$group)

# use with the clustering helpers of ecan
library(ggdendro)
ggdendro::ggdendrogram(stats::as.hclust(tw))
```

tw_downweight

Downweighting of rare pseudospecies

Description

Gives a weight to each pseudospecies, so that the rare ones weigh less in the ordination. The original TWINSpan downweights them before the correspondence analysis, and twinspan() does the same by default. The weights are used only in the ordination: the preference of the pseudospecies is counted on the raw occurrences.

Usage

```
tw_downweight(
  y,
  method = c("hill", "decorana"),
  fraction = 5,
  frq_lim = 0.2,
  w_min = 0.01,
  rw = NULL
)
```

Arguments

y	A binary matrix of stands by pseudospecies.
method	A string, "hill" or "decorana".
fraction	A numeric of the downweighting fraction of "decorana".
frq_lim	A numeric of the frequency above which "hill" does not downweight.
w_min	A numeric of the smallest weight of "hill".
rw	A numeric vector of stand weights, or NULL.

Details

Two ways are available. "hill" is the WEIGHT subroutine of the original TWINSpan: a pseudospecies occurring in a smaller proportion of the stands than frq_lim is weighted in proportion to that shortfall, and no weight falls below w_min. "decorana" is the downweighting of decorana() and of vegan::downweight(), where the frequencies are compared with the most frequent pseudospecies instead of a fixed proportion.

Value

A numeric vector of the weight of each pseudospecies.

Examples

```
data(dune, package = "vegan")
psp <- pseudospecies(dune)
summary(tw_downweight(psp))
summary(tw_downweight(psp, method = "decorana"))
```

tw_hill_const

Constants of the original TWINSpan

Description

The values that Hill's FORTRAN program sets for one division. They are used when twinspan(polish = "hill"), and are collected here so that the correspondence with the original is easy to check.

Usage

```
tw_hill_const()
```

Value

A named list of the constants. rat_lim, frq_lim, feeble, icw_exp, ipr_exp, cwt_min, cr_long, cr_cut, polish_iter, mz_crit, mz_out and mz_ind.

Examples

```
unlist(tw_hill_const())
```

tw_inertia	<i>Total inertia of a pseudospecies matrix</i>
------------	--

Description

Used as the heterogeneity of a group in modified TWINSpan.

Usage

```
tw_inertia(y, w = NULL)
```

Arguments

y	A binary matrix of stands by pseudospecies.
w	A numeric vector of pseudospecies weights, or NULL.

Value

A numeric of the total inertia (0 when it cannot be computed).

Examples

```
data(dune, package = "vegan")
tw_inertia(pseudospecies(dune))
```

tw_preference	<i>Preference of pseudospecies for one side of a division</i>
---------------	---

Description

The preference is $(f2 - f1) / (f2 + f1)$, where $f1$ and $f2$ are the relative frequencies of the pseudospecies in the negative and the positive group. It ranges from -1 (only in the negative group) to 1.

Usage

```
tw_preference(y, positive)
```

Arguments

y	A binary matrix of stands by pseudospecies.
positive	A logical vector. TRUE: the stand is in the positive group.

Value

A numeric vector of the preference of each pseudospecies.

Examples

```
data(dune, package = "vegan")
psp <- pseudospecies(dune)
pos <- tw_ra(psp)$sample > 0
summary(tw_preference(psp, pos))
```

tw_ra	<i>Reciprocal averaging (first correspondence analysis axis)</i>
-------	--

Description

Reciprocal averaging (first correspondence analysis axis)

Usage

```
tw_ra(y, w = NULL, rw = NULL, max_iter = 999, tol = 1e-10)
```

Arguments

y	A binary matrix of stands by pseudospecies.
w	A numeric vector of pseudospecies weights, or NULL.
rw	A numeric vector of stand weights, or NULL.
max_iter	An integer of the maximum number of iterations.
tol	A numeric of the convergence tolerance.

Value

A list of stand scores (\$sample), pseudospecies scores (\$species), the eigenvalue (\$eig) and \$converged.

Examples

```
data(dune, package = "vegan")
ra <- tw_ra(pseudospecies(dune))
ra$eig
```

tw_species_data	<i>Data on which the species of TWINSpan are classified</i>
-----------------	---

Description

The original TWINSpan does not classify the species on the pseudospecies table itself, but on how faithful each species is to the groups of stands. Every group of the hierarchy, the terminal ones and the ones that were divided further, becomes three pseudo-quadrats, at the cut levels 0.8, 2 and 6 of the ratio between the frequency of the species inside the group and its frequency outside. A species weighs as much as it occurs, and a group weighs as much as it holds, multiplied by $\sqrt{2}$ for every level it stands above the deepest one, and doubled for the two upper cut levels.

Usage

```
tw_species_data(
  object,
  psp = object$pseudospecies,
  sp_map = attr(bsp, "species"),
  levmax = object$max_depth
)
```

Arguments

object	A "twinspan" object, or the result of <code>tw_tree()</code> .
psp	The pseudospecies matrix of that object.
sp_map	The species of each pseudospecies.
levmax	The deepest level of division.

Value

A list of the binary matrix (`$y`) of species by pseudo-quadrats, the weights of the species (`$rw`) and of the pseudo-quadrats (`$cw`), and the ratios (`$ratio`).

Examples

```
data(dune, package = "vegan")
tw <- twinspan(dune)
str(tw_species_data(tw))
```

tw_two_way

Ordered two-way table of a TWINSpan result

Description

Arranges the community data with the stands and the species in the order of their division paths, as in the printed output of TWINSpan. The dichotomy of each stand is shown by the digits below the table.

Usage

```
tw_two_way(object, cells = c("level", "abundance"))
```

```
## S3 method for class 'tw_two_way'  
print(x, ...)
```

Arguments

object	A "twinspan" object made with species = TRUE.
cells	A string. "level": the pseudospecies cut level of each cell. "abundance": the original values.
x	A "tw_two_way" object.
...	Ignored.

Value

tw_two_way() returns a character matrix with class "tw_two_way", "stand_path" and "species_path" attributes.

print() returns the object invisibly.

Examples

```
data(dune, package = "vegan")  
tw_two_way(twinspan(dune))
```

Index

`add_mid_p_bin_w`
 (`draw_layer_construction`), 5
`as.hclust.twinspan` (`twinspan`), 14

`bin_width` (`draw_layer_construction`), 5

`cls_add_group` (`cluster`), 2
`cls_color` (`cluster`), 2
`cluster`, 2
`cols_one2multi` (`is_one2multi`), 9

`d`, 4
`df2table`, 4
`dist2df` (`df2table`), 4
`distance` (`cluster`), 2
`draw_layer_construction`, 5

`gen_example`, 6

`h` (`d`), 4

`ind_val`, 8
`is_multi2multi` (`is_one2multi`), 9
`is_one2multi`, 9
`is_one2one` (`is_one2multi`), 9

`mid_point` (`draw_layer_construction`), 5

`ord_add_group` (`ordination`), 10
`ord_extract_score` (`ordination`), 10
`ord_plot` (`ordination`), 10
`ordination`, 10

`pad2longest`, 11
`print.tw_two_way` (`tw_two_way`), 21
`print.twinspan` (`twinspan`), 14
`pseudospecies`, 12

`read_biss`, 12

`select_one2multi` (`is_one2multi`), 9
`shdi`, 13

`table2df` (`df2table`), 4
`tw_downweight`, 16
`tw_hill_const`, 17
`tw_inertia`, 18
`tw_preference`, 18
`tw_ra`, 19
`tw_species_data`, 20
`tw_two_way`, 21
`twinspan`, 14

`unique_length` (`is_one2multi`), 9