Package 'multirich'

May 15, 2021

Type Package

Title Calculate Multivariate Richness via UTC and sUTC

Version 2.1.3

Date 2021-05-13

Author Alexander Keyel

Maintainer Alexander Keyel <skeyel@gmail.com>

Description Functions to calculate Unique Trait Combinations (UTC) and scaled Unique Trait Combinations (sUTC) as measures of multivariate richness. The package can also calculate beta-diversity for trait richness and can partition this into nestedness-related and turnover components. The code will also calculate several measures of overlap. See Keyel and Wiegand (2016) <doi:10.1111/2041-210X.12558> for more details.

License GPL (>= 2)

Suggests knitr, rmarkdown, qpdf

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-14 23:30:11 UTC

R topics documented:

`multirich-package` *multirich (package)*

Description

Package to calculate the number of unique trait combinations and scale the number of unique trait combinations by the total number of trait combinations possible as measures of multivariate richness.

Details

Package:	multirich
Type:	Package
Version:	2.1.1
Date:	2015-09-14
License:	GPL-2 (or later)

Author(s)

Alexander "Sasha" Keyel
Maintainer: Sasha Keyel <skeyel@gmail.com>

References

Keyel, A.C. and K. Wiegand. (in review) Validating the use of Unique Trait Combinations for measuring multivariate functional richness.

calc.mvd *Calculate multivariate richness and functional overlap*

Description

Function to reduce to functional units, and calculate multivariate richness and functional overlap

Usage

```
calc.mvd(in.mat, in.com, traitspace, calc.ovr)
```

Arguments

in.mat	A record by trait matrix
in.com	A record by community matrix
traitspace	The total trait space for scaling purposes
calc.ovr	An indicator for whether overlap metrics should be calculated

calc.mvo *Calculate functional overlap*

Description

Calculate functional overlap

Usage

```
calc.mvo(in.mat, dups)
```

Arguments

in.mat	a record x trait matrix
dups	dups indicate which records in the record x trait matrix are duplicates

check.breaks *Check breaks*

Description

Check that break point lists are all the same length

Usage

```
check.breaks(breaks, in.mat)
```

Arguments

breaks	A list containing break points to use for each continuous trait. Categorical traits are unmodified, and should be a list containing the text "cat"
in.mat	A record x trait matrix

check.traits *Check trait*

Description

Check trait for compatibility with assembly procedure & adjust as necessary

Usage

```
check.traits(in.mat, tr1.lim, tr2.lim, sim.type, filter.vals = "none")
```

Arguments

in.mat	A record x trait matrix
tr1.lim	The minimum and maximum allowable values for trait 1
tr2.lim	The minimum and maximum allowable values for trait 2
sim.type	"random" runs random assembly (or if any sim.type other than limiting or filter is specified), "limiting" runs limiting similarity, and "filter" applies an environmental filter.
filter.vals	Vector of filter values in text form with a semi-colon delimiter (sorry!) e.g., c("1,1", "1;2") would only allow values of 1,1 and 2,2 in the data.
tr1	Trait values for trait 1
tr2	Trait values for trait 2

Note

This function is very inefficient & could be optimized!

data.preprocess *Preprocess data*

Description

Preprocess data

Usage

```
data.preprocess(in.mat, log.trans = 0, st.range = 0, col.mins, col.maxs)
```

Arguments

in.mat	A record x trait matrix
log.trans	Whether or not to do a log transformation. It is recommended to do all log-transformations outside of the multirich package.
st.range	Must equal zero. This option may be added in future versions of the package
col.mins	Minimum column values to use. "use data" will extract these from the dataset
col.maxs	Maximum column values to use. "use data" will extract these from the dataset

df.categorize *Categorize data*

Description

Currently by rounding, which is a sub-optimal approach for many questions

Usage

```
df.categorize(in.mat, cell.res)
```

Arguments

in.mat	A record x trait matrix
cell.res	The number of decimal places to round the data to.

expand.breaks *Expand trait breaks*

Description

Function to match longer lists of trait breaks. This may be a sub-optimal solution

Usage

```
expand.breaks(in.lst, target.reps)
```

Arguments

in.lst	the list of trait breakpoints to be expanded
target.reps	the target length of the trait breakpoint list.

Value

an expanded list of trait breakpoints

get.breaks *Function to calculate break points for a trait*

Description

takes range of values, and then creates break points for all possible sub-categorizations of those values WARNING: Currently only gives breakpoints for integer values!

Usage

```
get.breaks(min.val, max.val)
```

Arguments

min.val	The minimum value to be used. Can be taken from the data, or can be based on a priori knowledge
max.val	The maximum value to be used in generating break points. Can be taken from the data or can be based on a priori knowledge.

get.traitspace *Calculate trait space (simple)*

Description

Function to calculate the traitspace using minimum and maximum column values. Creates a hypercubic traitspace.

Usage

```
get.traitspace(in.mat, col.res, col.mins, col.maxs)
```

Arguments

in.mat	A record x trait matrix
col.res	The number of decimal places to use for rounding (resolution) purposes
col.mins	The minimum values to use for the trait space. "use data" extracts minimums from in.mat
col.maxs	The maximum values to use for the trait space. "use data" extracts maximums from in.mat

listtotext *List to text*

Description

Take a list (or vector) and convert it to text separated by separator. Does not work for nested lists ORIGINALLY FROM MYR.R script

Usage

```
listtotext(inlist, sep)
```

Arguments

inlist	An unnested list
sep	A separator to separate elements of the list.

make.trait *Make trait*

Description

Create integer trait values from a uniform random distribution

Usage

```
make.trait(trlim, n.recs)
```

Arguments

trlim	A vector containing the minimum and maximum values for the trait
n.recs	The number of trait values to create for trait

matrix.check *Check that input to mvfd is in matrix format*

Description

Check that input to mvfd is in matrix format

Usage

```
matrix.check(in.mat)
```

Arguments

in.mat	An input to be tested for matrix formatting

mvfd *mvfd Main function to calculate functional richness*

Description

Main function to calculate multivariate richness. Goal is to mirror format of dbfd from Laliberte & Shipley 2011.

Usage

```
mvfd(
  in.mat,
  in.com = "none",
  unequal.abund = F,
  resolution = 0,
  st.range = 0,
  log.trans = 0,
  col.mins = "use data",
  col.maxs = "use data",
  traitspace = "use data",
  calc.ovr = 1,
  force.matrix = TRUE
)
```

Arguments

in.mat	A record x trait matrix. Needs to be in matrix format, not as a dataframe
in.com	A community x species matrix. If only one community with all the species is considered, you can enter "none", and the code will auto-create the community needed for the script.
unequal.abund	A feature not currently in script, intended as an option to indicate whether abundance data should be incorporated.
resolution	This input controls rounding of the data (categorization was acheived by rounding for simplicity). 0 indicates integers, 1 = 1 decimal place, 2 = 2 decimal places, -1 = 10's place., etc.)
st.range	Option to control range standardization. This was never properly scripted and should remain 0. Any desired range transformations should be done prior to this script.
log.trans	Option to control whether or not data are log transformed. This has not been properly tested as I found it easier to log-transform the data manually in Excel.
col.mins	If option is "use data" the function will get the minimum from the data. Otherwise a vector of minimum values to use can be specified
col.maxs	If option is "use data" the function will get the maximum values from the data. Otherwise a vector of maximum values can be specified.
traitspace	If set to "use data", the function will estimate the traitspace as the product of trait ranges. Otherwise, the specified traitspace will be used for scaling (e.g., if you want to input a traitspace based on a convex hull)
calc.ovr	An option to determine whether overlap is calculated. This may be slow or buggy, so in some cases it may be easier to turn it off.
force.matrix	an option to determine whether to try to force an input into matrix format

Details

Currently takes a record x trait matrix and an optional community matrix.

Author(s)

A.C. Keyel

Examples

```
# Compare functional diveristy between species using the Iris dataset
ind.mat = iris
ind.mat$Species = NULL
ind.lbl = sprintf("Ind_%s",seq(1,nrow(iris)))
ind.mat = as.matrix(ind.mat) #Needs to be in matrix format
rownames(ind.mat) = ind.lbl
com.base = iris$Species
pool = rep(1,nrow(iris))
com1 = sapply(com.base, function(x){ifelse(x == "setosa",1,0)})
com2 = sapply(com.base, function(x){ifelse(x == "versicolor",1,0)})
com3 = sapply(com.base, function(x){ifelse(x == "virginica",1,0)})
com.vec = c(pool,com1,com2,com3)
com.lbl = c("pool","com1","com2","com3")
com.mat = matrix(com.vec,nrow = 4,byrow = TRUE,dimnames = list(com.lbl,ind.lbl))

mvr.out = mvfd(ind.mat,com.mat)
```

---

part.mvr.beta *Partition multivariate richness Beta Component*

Description

Function to calculate dissimilarity between a pair of communities, and partition it into nestedness-related & turnover components Either the Sorensen index or the Jaccard index can be used to calculate dissimilarity

Usage

```
part.mvr.beta(in.mat, in.com, index.rows, index.type = "Sorensen")
```

Arguments

in.mat	A record x trait matrix. Needs to be in matrix format, not as a dataframe
in.com	A community x record matrix
index.rows	A vector with 2 elements, that gives the row number for the pair of communities of interest.
index.type	specifies which index to use. Options are Sorensen (default) & Jaccard

Value

- aa Overlap between the two communities
- dissimilarity Dissimilarity between the two communities
- turnover This gives the turnover between the two communities. To get the percent of dissimilarity due to turnover, divide by total dissimilarity.
- Bnes Nestedness-related dissimilarity between the two communities. To get the percent of dissimilarity due to this, divide by total dissimilarity.

Author(s)

A.C. Keyel

References

Baselga, A. 2010. Partitioning the turnover and nestedness components of beta diversity. Global Ecology and Biogeography 19: 134-143
Baselga, A. 2012. The relationship between species replacement, dissimilarity derived from nestedness, and nestedness. Global Ecology and Biogeography 21: 1223-1232
Villeger, S. Grenouillet, G., and Brosse, S. 2013. Decomposing functional Beta-diversity reveals that low functional Beta-diversity is driven by low functional turnover in European fish assemblages. Global Ecology and Biogeography 22: 671-681.

Examples

```
## Partition beta diversity for two species in the iris dataset

# Set up record x trait matrix
ind.mat = iris
ind.mat$Species = NULL
ind.lbl = sprintf("Ind_%s",seq(1,nrow(iris)))
ind.mat = as.matrix(ind.mat) #Needs to be in matrix format
rownames(ind.mat) = ind.lbl

# Set up community matrix
com.base = iris$Species
pool = rep(1,nrow(iris))
com1 = sapply(com.base, function(x){ifelse(x == "setosa",1,0)})
com2 = sapply(com.base, function(x){ifelse(x == "versicolor",1,0)})
com3 = sapply(com.base, function(x){ifelse(x == "virginica",1,0)})
com.vec = c(pool,com1,com2,com3)
com.lbl = c("pool","com1","com2","com3")
com.mat = matrix(com.vec, nrow = 4,byrow = TRUE,dimnames = list(com.lbl,ind.lbl))

# Specify the communities to compare
index.rows = c(2,4) #compare species 1 & 3 (+1 due to the pool being the first community)

# Do the diversity partitioning
part.out = part.mvr.beta(ind.mat,com.mat,index.rows,index.type = "Sorensen")
  com.overlap = part.out[[1]]
  #0: no overlap
```

```
com.dis = part.out[[2]]
#1: complete dissimilarity
com.turn = part.out[[3]]
#1: This gives the absolute amount of dissimilarity due to turnover.
# For percent dissimilarity due to turnover, you need to divide by overall dissimilarity
com.nest = part.out[[4]]
#0: This gives the absolute amount of dissimilarity due to nestedness.
# For percent, divide by total dissimilarity
```

`plot.sensitivity` *Plot sensitivity analysis results*

Description

Goal is to create a plot similar to a ROC plot, where at one side, everything is in a single category, and on the other end, everything is in its own category.

Usage

```
## S3 method for class 'sensitivity'
plot(sutc.vals, traitspaces, n, com.names = "none", in.pdf = "none")
```

Arguments

`sutc.vals`	A list containing scaled unique trait combinations (see above)
`traitspaces`	A vector containing the traitspace for each combination value
`n`	The sample size of the trait x record matrix
`com.names`	The names of the communities being evaluated (for labeling purposes)
`in.pdf`	A pdf file to be generated, containing the two plots. "none" skips generating a pdf, and "no.plot" disables the function

---

`sensitivity.analysis` *Sensitivity Analysis*

Description

Run a sensitivity analysis on the data to see to what extent richness is sensitive to choice of break points.

Usage

```
sensitivity.analysis(
  in.mat,
  in.com,
  breaks,
  out.pdf,
  in.traitspaces = "use data"
)
```

Arguments

in.mat	A record x trait matrix
in.com	A community x record matrix
breaks	A list containing break points to use for each trait. Categorical traits are unmodified, and should be a list containing the text "cat" and the number of categories. e.g., list("cat",2). For non-categorical traits, this list needs to contain the same number of elements for each trait. See example below for format.
out.pdf	A pdf to be created containing the results of the sensitivity analysis. "none" plots to the active R window. "no.plot" will disable the plot entirely.
in.traitspaces	a vector of trait spaces, if pre-existing trait spaces are desired. Otherwise, the default of 'use data' will calculate traitspaces based on the range of values present in the data.

Examples

```
# Example of a sensitivity analysis using simulated traits

# Set up example
#Adding 0.5 is to give even probabilities when rounding
n.recs = 10
tr1 = round(runif(n.recs,1 - 0.5, 10 +0.5),0)
tr2 = round(runif(n.recs,1 - 0.5, 4 + 0.5),0)

# Set up row & col names
row.nams = sprintf("Record_%s", seq(1,n.recs))
col.nams = c("tr1","tr2")

#Create matrix
in.mat = matrix(c(tr1,tr2), ncol = 2, dimnames = list(row.nams, col.nams))

# Get break points
tr1.breaks = get.breaks(1,10)
tr2.breaks = get.breaks(1,4)
tr2.breaks = expand.breaks(tr2.breaks, 9)

breaks = list(tr1.breaks, tr2.breaks)

# Actually run sensitivity analysis
# Note that the plot & results will vary as this depends on random numbers
results = sensitivity.analysis(in.mat,"none", breaks, "none")
```

---

sensitivity.boxplots *Make boxplots based on multiple simulations*

Description

Takes the output results from multiple simulation iterations and plots boxplots displaying the results

Usage

```
sensitivity.boxplots(results.mat, traitspaces, n.recs)
```

Arguments

results.mat	An input matrix with ncol equal to the number of elements in traitspaces, and a given row containing the values for each trait space for that simulation run.
traitspaces	A vector identifying the traitspaces at which the simulation analysis was carried out
n.recs	The number of records in the original matrix. This is the point where if every record were unique, the entire traitspace would be full.

---

utc.sim *Sensitivity Simulator*

Description

Function to help run simulations for sensitivity analysis. Note that for simplicity, the number of traits is restricted to two, each with the same range of values, as can be done for real traits by standardizing by range.

Usage

```
utc.sim(
  n.reps,
  n.recs,
  tr1.lim,
  tr2.lim,
  sim.type = "random",
  filter.vals = "none"
)
```

Arguments

n.reps	The number of replicates for the simulation
n.recs	The number of records in the record x trait matrix
tr1.lim	The minimum and maximum allowable values for trait 1
tr2.lim	The minimum and maximum allowable values for trait 2
sim.type	"random" runs random assembly (or if any sim.type other than limiting or filter is specified), "limiting" runs limiting similarity, and "filter" applies an environmental filter.
filter.vals	Vector of filter values in text form with a semi-colon delimiter (sorry!) e.g., c("1,1", "1;2") would only allow values of 1,1 and 2,2 in the data.

Index