Package 'codyn'

June 12, 2018

Title Community Dynamics Metrics

Version 2.0.0

Date 2018-06-11

Description Univariate and multivariate temporal and spatial diversity indices, rank abundance curves, and community stability metrics. The functions implement metrics that are either explicitly temporal and include the option to calculate them over multiple replicates, or spatial and include the option to calculate them over multiple time points. Functions fall into five categories: static diversity indices, temporal diversity indices, spatial diversity indices, rank abundance curves, and community stability metrics. The diversity indices are temporal and spatial analogs to traditional diversity indices. Specifically, the package includes functions to calculate community richness, evenness and diversity at a given point in space and time. In addition, it contains functions to calculate species turnover, mean rank shifts, and lags in community similarity between two time points.

Depends R (>= 3.2.0)

Imports assertthat, stats, permute, vegan

Suggests testthat, knitr, ggplot2, reshape2, devtools, dplyr, rmarkdown

License Apache License (== 2.0)

URL https://github.com/NCEAS/codyn/

BugReports https://github.com/NCEAS/codyn/issues

LazyData true

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Lauren Hallett [aut],
Meghan L. Avolio [aut],
Ian T. Carroll [aut],
Sydney K. Jones [aut],

A. Andrew M. MacDonald [aut],
Dan F. B. Flynn [aut],
Peter Slaughter [aut],
Julie Ripplinger [aut],
Scott L. Collins [aut],
Corinna Gries [aut],
Matthew B. Jones [aut, cre]

Maintainer Matthew B. Jones <jones@nceas.ucsb.edu>

Repository CRAN

Date/Publication 2018-06-12 05:29:23 UTC

R topics documented:

---

abundance_change *Species Abundance Changes*

Description

Calculates the abundance change for species in a replicate between two time points. Changes are on abundance values provided, if relative data is used, then changes in relative abundance will be calculated.

Usage

```
abundance_change(df, time.var, species.var, abundance.var,
  replicate.var = NULL, reference.time = NULL)
```

Arguments

`df`	A data frame containing time, species, and abundance columns and an optional column of replicates
`time.var`	The name of the time column
`species.var`	The name of the species column
`abundance.var`	The name of the abundance column
`replicate.var`	The name of the optional replicate column. If specified, replicate must be unique within the dataset and cannot be nested within treatments or blocks.

`reference.time` The name of the optional time point that all other time points should be compared to (e.g. the first year of data). If not specified, each comparison is between consecutive time points (e.g. first to second year, second to third year, etc.)

Value

The abundance_change function returns a data frame with the following fields:

- replicate.var: A column with the specified replicate.var, if it is specified.
- time.var: A column with the specified time.var and a second column, with '2' appended to the name. Time is subtracted from time2
- species.var: A column with the specified species.var.
- change: A numeric column of the change in abundance between time points. A positive value occurs when a species increases in abundance over time, and a negative value when a species decreases in abundance over time.

References

Avolio et al. Submitted to MEE

Examples

```
data(pplots)
# Without replicates
df <- subset(pplots, plot == 25)
abundance_change(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 time.var = "year")

# With replicates
df <- subset(pplots, year < 2004 & plot %in% c(6, 25, 32))
abundance_change(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 replicate.var = "plot",
                 time.var = "year")

# With reference year
df <- subset(pplots, year < 2005 & plot %in% c(6, 25, 32))
abundance_change(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 replicate.var = "plot",
                 time.var = "year",
                 reference.time = 2002)
```

abundance_difference *Abundance Differences*

Description

Calculates the abundance difference for species between two samples. Differences are on abundance values provided, if relative data is used, then differences in relative abundance will be calculated. There are three ways differences can be calculated. 1) Between treatments within a block (note: block.var and treatment.var need to be specified). 2) Between treatments, pooling all replicates into a single species pool (note: pool = TRUE, treatment.var needs to be specified, and block.var = NULL). 3) All pairwise combinations between all replicates (note: block.var = NULL, pool = FALSE and specifying treatment.var is optional. If treatment.var is specified, the treatment that each replicate belongs to will also be listed in the output).

Usage

```
abundance_difference(df, time.var = NULL, species.var, abundance.var,
  replicate.var, treatment.var = NULL, pool = FALSE, block.var = NULL,
  reference.treatment = NULL)
```

Arguments

df	A data frame containing species, abundance, replicate columns and optional time, treatment and block columns.
time.var	The name of the optional time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the replicate column. Replicate must be unique within the dataset and cannot be nested within treatments or blocks.
treatment.var	The name of the optional treatment column
pool	An argument to allow abundance values to be pooled within a treatment. The default value is "FALSE", a value of "TRUE" averages the abundances of each species within a treatment at a given time point.
block.var	The name of the optional block column
reference.treatment	The name of the optional treatment that all other treatments will be compared to (e.g. only controls will be compared to all other treatments). If not specified all pairwise treatment comparisons will be made.

Value

The abundance_difference function returns a data frame with the following attributes:

- species.var: A column that has same name and type as the species.var column.

- difference: A numeric column of the abundance differences between the two samples being compared (replicates or treatments). A numeric column of the change in abundance between consecutive timepoints. A positive value occurs when a species has greater abundance in replicate.var2 than in replicate.var and/or in treatment.var2 than in treatment.var. #'
- replicate.var: A column that has same name and type as the replicate.var column, represents the first replicate being compared. Note, a replicate column will be returned only when pool = FALSE or block.var = NULL.
- replicate.var2: A column that has the same type as the replicate.var column, and is named replicate.var with a 2 appended to it, represents the second replicate being compared. Note, a replicate.var column will be returned only when pool = FALSE and block.var = NULL.
- time.var: A column that has the same name and type as the time.var column, if time.var is specified.
- treatment.var: A column that has same name and type as the treatment.var column, represents the first treatment being compared. A treatment.var column will be returned when pool = TRUE, block.var is specified, or treatment.var is specified.
- treatment.var2: A column that has the same type as the treatment.var column, and is named treatment.var with a 2 appended to it, represents the second treatment being compared. A treatment.var column will be returned when pool = TRUE, block.var is specified, or treatment.var is specified.
- block.var: A column that has same name and type as the block.var column, if block.var is specified.

References

Avolio et al. Submitted to MEE

Examples

```
data(pplots)
# With block and no time
df <- subset(pplots, year == 2002 & block < 3)
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     treatment.var = "treatment",
                     block.var = "block",
                     replicate.var = "plot")

# With blocks and time
df <- subset(pplots, year < 2004 & block < 3)
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     treatment.var = "treatment",
                     block.var = "block",
                     replicate.var = "plot",
                     time.var = "year")

# With blocks, time and reference treatment
```

```
df <- subset(pplots, year < 2004 & block < 3)
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     treatment.var = "treatment",
                     block.var = "block",
                     replicate.var = "plot",
                     time.var = "year",
                     reference.treatment = "N1P0")

# Pooling by treatment with time
df <- subset(pplots, year < 2004)
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     treatment.var = "treatment",
                     pool = TRUE,
                     replicate.var = "plot",
                     time.var = "year")

# All pairwise replicates with treatment
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     replicate.var = "plot",
                     time.var = "year",
                     treatment.var = "treatment")

# All pairwise replicates without treatment
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
abundance_difference(df = df,
                     species.var = "species",
                     abundance.var = "relative_cover",
                     replicate.var = "plot",
                     time.var = "year")
```

`add_ranks` *Add abundance ranks*

Description

Rank species by abundance, by specified grouping. Species with zero abundance receive rank S+1, where S is the total number of species in the group.

Usage

```
add_ranks(df, abundance.var)
```

Arguments

df	A data frame containing a single record per species with its abundance
abundance.var	The name of the abundance column

Value

The add_ranks function returns a data frame with the following additional column:

- rank: A numeric column with the species rank; a rank of 1 indicates the species was most abundant in that time period. All species that are not present in that time period have the rank value S+1 where S is the number of species in the sample.

---

add_rank_abundance *Add relative abundance ranks and cumulative abundance*

Description

Rank species by abundance within the specified grouping. The rank of the lead abundant species is 1 and the most abundant species has rank 1/S, where S is the number of species in the group.

Usage

```
add_rank_abundance(df, species.var, abundance.var)
```

Arguments

df	A data frame containing a single record per species with its abundance
species.var	The name of the species column
abundance.var	The name of the abundance column

Value

The add_rank_abundance function returns a data frame with the following additional column:

- rank:
- relrank: A numeric column with the species rank divided by the maximum rank in the group; a rank of 1 indicates the species was the least abundant.
- cumabund:

check_args *Checking for errors in arguments*

Description

Check for errors in the application of arguments and input data for all `*_change` and `*_difference` functions.

Usage

```
check_args(df, time.var = NULL, species.var, abundance.var,
  replicate.var = NULL, treatment.var = NULL, pool = FALSE,
  block.var = NULL, reference.time = NULL, reference.treatment = NULL)
```

Arguments

`df`	The name of the dataframe
`time.var`	The name of the optional time column
`species.var`	The name of the species column
`abundance.var`	The name of the abundance column
`replicate.var`	The name of the replicate column
`treatment.var`	The name of the optional treatment column
`pool`	The name the optional pooling approach
`block.var`	The name of the optional block column
`reference.time`	The name of the optional reference time
`reference.treatment`	The name of the optional reference treatment

check_multispp *Utility function to stop calculations if only one species occurs in at least one replicate*

Description

Utility function to stop calculations if only one species occurs in at least one replicate

Usage

```
check_multispp(df, species.var, replicate.var)
```

Arguments

`df`	A dataframe containing time.var, species.var and abundance.var columns
`species.var`	The name of the species column from df
`replicate.var`	The name of the replicate column from df

check_names *check names of data frames*

Description

check names of data frames

Usage

```
check_names(given, data)
```

Arguments

given	Vector of variable names as supplied by user
data	Data frame containing variables

check_numeric *Utility to check for numeric abundance and time variables*

Description

Utility to check for numeric abundance and time variables

Usage

```
check_numeric(df, time.var, abundance.var)
```

Arguments

df	A dataframe containing time.var, species.var, and replicate.var columns
time.var	The name of the time column from df
abundance.var	The name of the replicate column from df

`check_single` *Utility function to ensure only a single record exists for a given species within one replicate if replicate.var given, and for one time point if time.var given.*

Description

Utility function to ensure only a single record exists for a given species within one replicate if replicate.var given, and for one time point if time.var given.

Usage

```
check_single(df, species.var, time.var = NULL, replicate.var = NULL)
```

Arguments

`df`	A dataframe containing a species.var column, and optionally a time.var and replicate.var columns
`species.var`	The name of the species column from df
`time.var`	The name of the time column from df
`replicate.var`	The name of the replicate column from df

`check_single_onerep` *Utility function to warn users that either multiple records exist within replicates, or that data may be spanning multiple replicates but no replicate.var has been specified*

Description

Utility function to warn users that either multiple records exist within replicates, or that data may be spanning multiple replicates but no replicate.var has been specified

Usage

```
check_single_onerep(df, time.var, species.var)
```

Arguments

`df`	A dataframe containing time.var, species.var and abundance.var columns
`time.var`	The name of the time column from df
`species.var`	The name of the species column from df

check_sppvar *Utility function to stop calculations if the species never change in a replicate*

Description

Utility function to stop calculations if the species never change in a replicate

Usage

```
check_sppvar(comdat)
```

Arguments

comdat	A community dataframe

codyn *Community Dynamics Metrics*

Description

Univariate and multivariate temporal and spatial diversity indices, rank abundance curves, and community stability metrics for ecologists.

Details

The functions in codyn implement metrics that are either explicitly temporal and include the option to calculate them over multiple replicates, or spatial and include the option to calculate them over multiple time points. Functions fall into five categories: static diversity indices, temporal diversity indices, spatial diversity indices, rank abundance curves, and community stability metrics. The diversity indices in codyn are temporal and spatial analogs to traditional diversity indices. Specifically, codyn includes functions to calculate community richness, evenness and diversity at a given point in space and time. In addition, codyn contains functions to calculate species turnover, mean rank shifts, and lags in community similarity between two time points. For the components of rank abundance curves, the shape of the rank abundance curve, species abundances and multivariate metrics of community composition, codyn contains functions to calculate these metrics either between time points or a single time point between two paired replicates. The community stability metrics in codyn calculate overall stability and patterns of species covariance and synchrony over time. Finally, codyn contains vignettes that describe methods and reproduce figures from published papers to help users contextualize and apply functions to their own data. Work on this package was supported by National Science Foundation grant #1262458 to the National Center for Ecological Analysis and Synthesis (NCEAS), the University of Wisconsin, and the University of New Mexico, and a SESYNC Synthesis Postdoctoral Fellowship to MLA.

Functions

- turnover: Calculates species turnover between time periods
- rank_shift: Calculates the mean relative change in species rank abundances
- rate_change: Calculates the rate change in a community over time
- rate_change_interval: Produces a data frame containing differences in species composition between samples at increasing time intervals
- community_stability: Calculates community stability over time
- variance_ratio: Computes the ratio of the variance of aggregate species abundances in a community
- synchrony: Calculates the degree synchrony in species abundances
- temporal_torus_translation: Calculates a test statistic on a null ecological community created via cyclic shifts. `confint` provides mean and confidence intervals of this null distribution
- community_structure: Calculates richness and evenness (using specified metric) for a replicate
- community_diversity: Calculates diversity (using specified metric) for a replicate
- RAC_change:Calculates changes in species richness, evenness, species' ranks, gain, and losses for a replicate over time
- abundance_change: For each species in a replicate, calculates changes in abundance over time
- curve_change: Calculates changes in the shape of the RAC curve for each replicate over time
- multivariate_change: Calculates changes in community composition and dispersion over time
- RAC_difference: Calculates differences in species richness, evenness, species' ranks, shared species between paired samples at a single point in time
- abundance_difference: Calculates differences in abundance for each species in paired samples at a single point in time
- curve_difference: Calculates differences in the shape of the RAC between paired samples at a single point in time
- multivariate_difference: Calculates differences in community composition and dispersion of all replicates between treatments at a single point in time

Author(s)

- Lauren Hallett <lauren.m.hallett@gmail.com>
- Meghan L. Avolio <meghan.avolio@jhu.edu>
- Ian T. Carroll <icarroll@sesync.org>
- Sydney K. Jones <syd@sevilleta.unm.edu>
- A. Andrew A. MacDonald <aammacdonald@gmail.com>
- Dan F. B. Flynn <flynn@fas.harvard.edu>
- Peter Slaughter <slaughter@nceas.ucsb.edu>

- Julie Ripplinger <julie.ripplinger@asu.edu>
- Scott L. Collins <scollins@sevilleta.unm.edu>
- Corinna Gries <cgries@wisc.edu>
- Matthew B. Jones <jones@nceas.ucsb.edu>

`collins08` *Konza data from Collins et al. 2008*

Description

A dataset of tallgrass prairie plant composition at one annually burned and one unburned site over time at the Konza Prairie LTER, Manhattan Kansas (Collins et al. 2008).

Usage

```
data(collins08)
```

Format

A data frame with 2058 rows and 4 variables

Details

A data frame containing a column for replicate, year, species and abundance:

- replicate: A factor column of spatial replicates with two levels ("annually burned" and "unburned")
- year: An integer column of sampling time points
- species: A factor column of species sampled
- abundance: A numeric column of abundance values

Source

Collins, Scott L., Katharine N. Suding, Elsa E. Cleland, Michael Batty, Steven C. Pennings, Katherine L. Gross, James B. Grace, Laura Gough, Joe E. Fargione, and Christopher M. Clark. (2008) "Rank clocks and plant community dynamics." Ecology 89, no. 12: 3534-41.

`community_diversity` *Community Diversity*

Description

Calculates Shannon's or Inverse Simpson's diversity of a community, but only one metric of diversity can be calculated at a time and must be specified.

Usage

```
community_diversity(df, time.var = NULL, abundance.var,
  replicate.var = NULL, metric = c("Shannon", "InverseSimpson"))
```

Arguments

`df`	A data frame containing species and abundance columns and optional columns of time and/or replicate.
`time.var`	The name of the optional time column
`abundance.var`	The name of the abundance column
`replicate.var`	The name of the optional replicate column. If specified, replicate must be unique within the dataset and cannot be nested within treatments or blocks.
`metric`	The diversity metric to return: • "Shannon": The default metric, calculates Shannon's diversity. • "InverseSimpson": Calculates inverse of Simpson's diversity.

Value

The community_diversity function returns a data frame with the following attributes:

- time.var: A column that has the same name and type as the time.var column, if time.var is specified.
- replicate.var: A column that has same name and type as the replicate.var column, if replicate.var is specified.
- Shannon: A numeric column of Shannon's diversity if metric = "Shannon"
- InverseSimpson: A numeric column of the inverse of Simpson's diversity if metric = "InverseSimpson"

References

Magurran, A.E. 2004. Measuring Biological Diversity. Blackwell Publishing, Malden MA, USA.

Examples

```
data(pplots)
#Example with both time and replicates
df <- subset(pplots, plot == 25 | plot == 6)
community_diversity(df,
                    time.var="year",
                    replicate.var = "plot",
                    abundance.var = "relative_cover") # for Shannon's diversity metric

df <- subset(pplots, plot == 25 | plot == 6)
community_diversity(df,
                    time.var="year",
                    replicate.var = "plot",
                    abundance.var = "relative_cover",
                   metric = "InverseSimpson") # for Inverse of Simpson's diversity metric

#Example with no replicates
df <- subset(pplots, plot == 25)
community_diversity(df,
                    time.var="year",
                    abundance.var = "relative_cover") # for Shannon's diversity metric

#Example with no time or replicate
df <- subset(pplots, plot == 25 & year == 2002)
community_diversity(df,
                    abundance.var = "relative_cover") # for Shannon's diversity metric
```

---

community_stability *Community Stability*

Description

Calculates the stability of the overall community over time as the temporal mean / temporal standard deviation of aggregate species abundances (Tilman 1999).

Usage

```
community_stability(df, time.var, abundance.var, replicate.var = NA)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there should be a single community represented within each time point and replicate.

Value

The community_stability function returns a numeric stability value unless a replication column is specified in the input data frame. If replication is specified, the function returns a data frame with the following attributes:

- stability: A numeric column with the stability values.
- replicate.var: A column that shares the same name and type as the replicate.var column in the input data frame.

References

Tilman, D. "The Ecological Consequences of Changes in Biodiversity: A Search for General Principles." Ecology 80, no. 5 (July 1999): 1455-74. doi:10.1890/0012-9658(1999)080[1455:TECOCI]2.0.CO;2.

Examples

```
data(knz_001d)
community_stability(knz_001d[knz_001d$subplot=="A_1",],
                    time.var = "year",
                    abundance.var = "abundance") # for one subplot
community_stability(knz_001d,
                    time.var = "year",
                    abundance.var = "abundance",
                    replicate.var = "subplot") # across all subplots
```

---

community_structure *Community Structure*

Description

Calculates species richness and evenness of a community. Evenness may be calculated as Simpson's (1/D/S), EQ, or Evar, but only one metric of evenness can be calculated at a time and must be specified.

Usage

```
community_structure(df, time.var = NULL, abundance.var,
  replicate.var = NULL, metric = c("Evar", "SimpsonEvenness", "EQ"))
```

Arguments

df	A data frame containing species and abundance columns and optional columns of time and/or replicate.
time.var	The name of the optional time column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column. If specified, replicate must be unique within the dataset and cannot be nested within treatments or blocks.
metric	The evenness metric to return: • "Evar": The default metric, calculates Evar evenness from Smith and Wilson 1996 • "SimpsonEvenness": Calculates Simpsons' evenness • "EQ": Calculates EQ evenness from Smith and Wilson 1996

Value

The community_structure function returns a data frame with the following attributes:

- time.var: A column that has the same name and type as the time.var column, if time.var is specified.
- replicate.var: A column that has same name and type as the replicate.var column, if specified.
- richness: A numeric column of species richness
- Evar: A numeric column of Evar evenness if evenness = "Evar"
- EQ: A numeric column of EQ evenness if evenness = "EQ"
- SimpsonEvenness: A numeric column of Simpsons evenness if evenness = "SimpsonEveness"

References

Smith, B. and Wilson, J. B. 1996. A consumer's guide to evenness indices. Oikos 76: 70-82.

Examples

```
data(pplots)
#Example with both time and replicates
df <- subset(pplots, plot == 25 | plot == 6)
community_structure(df,
                    time.var="year",
                    replicate.var = "plot",
                    abundance.var = "relative_cover") # for Evar evenness metric

df <- subset(pplots, plot == 25 | plot == 6)
community_structure(df,
                    time.var="year",
                    replicate.var = "plot",
                    abundance.var = "relative_cover",
                    metric = "SimpsonEvenness") # for Simpson's evenness metric

#Example with no replicates
```

```
df <- subset(pplots, plot == 25)
community_structure(df,
                    time.var="year",
                    abundance.var = "relative_cover",
                    metric = "EQ") # for EQ evenness metric

#Example with only a single time point and no replicates
df <- subset(pplots, plot == 25 & year == 2002)
community_structure(df,
                    abundance.var = "relative_cover") # for Evar evenness metric
```

`confint.cyclic_shift` *Confidence Intervals from a Cyclic Shift Permutation*

Description

Calculates confidence intervals for the S3 object produced by `cyclic_shift`

Usage

```
## S3 method for class 'cyclic_shift'
confint(object, parm = "out", level = 0.95, ...)
```

Arguments

`object`	An object of class `cyclic_shift`
`parm`	which parameter is to be given a confidence interval. At present there is only one option: the mean of the null distribution. Defaults to "out", referring to the null distribution in objects of class `cyclic_shift`.
`level`	the confidence level required.
`...`	further arguments to `quantile`

Value

A dataframe with the following columns:

- lowerCI: A numeric column with the lowest confidence interval value.
- upperCI: A numeric column with the highest confidence interval value.
- nullMean: A numeric column with the average value of the specified test statistic when calculated on a null community.

References

Hallett, Lauren M., Joanna S. Hsu, Elsa E. Cleland, Scott L. Collins, Timothy L. Dickson, Emily C. Farrer, Laureano A. Gherardi, et al. "Biotic Mechanisms of Community Stability Shift along a Precipitation Gradient." Ecology 95, no. 6 (2014): 1693-1700.

Harms, Kyle E., Richard Condit, Stephen P. Hubbell, and Robin B. Foster. "Habitat Associations of Trees and Shrubs in a 50-Ha Neotropical Forest Plot." Journal of Ecology 89, no. 6 (2001): 947-59.

Examples

```
# Calculate a covariance matrix on a null community
data(knz_001d)
a1_cyclic <- cyclic_shift(subset(knz_001d, subplot == "A_1"),
                  time.var = "year",
                  species.var = "species",
                  abundance.var = "abundance",
                  FUN = cov,
                  bootnumber = 10)

# Return CI on a1_cyclic
confint(a1_cyclic)
```

---

curve_change *Curve Change*

Description

Calculates the area difference between two rank abundance curves between two time periods. If replicate is specified, it must be measured in both time points, otherwise it will be dropped for that time period comparison.

Usage

```
curve_change(df, time.var, species.var, abundance.var, replicate.var = NULL,
  reference.time = NULL)
```

Arguments

df	A data frame containing time, species, and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column. If specified, replicate must be unique within the dataset and cannot be nested within treatments or blocks.
reference.time	The name of the optional time point that all other time points should be compared to (e.g. the first year of data). If not specified, each comparison is between consecutive time points (e.g. first to second year, second to third year, etc.)

Value

The curve_change function returns a data frame with the following attributes:

- time.var: A column with the specified time.var and a second column, with '2' appended to the name. Time is subtracted from time2.
- curve_change: A numeric column of the change in curves between time points.
- replicate.var: A column that has same name and type as the replicate.var column, if specified.

References

Avolio et al. Submitted to MEE

Examples

```
data(pplots)
# Without replicates
df <- subset(pplots, plot == 25)
curve_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           time.var = "year")

# With replicates
df <- subset(pplots, year < 2004 & plot %in% c(6, 25, 32))
curve_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           replicate.var = "plot",
           time.var = "year")

# With reference year
df <- subset(pplots, year < 2005 & plot %in% c(6, 25, 32))
curve_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           replicate.var = "plot",
           time.var = "year",
           reference.time = 2002)
```

---

curve_difference *Curve Difference*

Description

Calculates the area difference between two rank abundance curves. There are three ways differences can be calculated. 1) Between all treatments within a block (note: block.var and treatment.var need to be specified. 2) Between treatments, pooling all replicates into a single species pool (note: pool = TRUE, treatment.var needs to be specified, and block.var = NULL. 3) All pairwise combinations between all replicates (note:block.var = NULL, pool = FALSE and specifying treatment.var is optional. If treatment.var is specified, the treatment that each replicate belongs to will also be listed in the output).

Usage

```
curve_difference(df, time.var = NULL, species.var, abundance.var,
  replicate.var, treatment.var = NULL, pool = FALSE, block.var = NULL,
  reference.treatment = NULL)
```

Arguments

df	A data frame containing a species, abundance, and replicate columns and optional time, treatment, and block columns
time.var	The name of the optional time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the replicate column. Replicate must be unique within the dataset and cannot be nested within treatments or blocks.
treatment.var	The name of the optional treatment column
pool	An argument to allow values to be pooled within treatment. The default value is "FALSE", a value of "TRUE" takes the average abundance of all species within a treatment at a given time point prior to comparisons.
block.var	The name of the optional block column
reference.treatment	The name of the optional treatment that all other treatments will be compared to (e.g. only controls will be compared to all other treatments). If not specified all pairwise treatment comparisons will be made.

Value

The curve_difference function returns a data frame with the following attributes:

- curve_diff: A numeric column of the area difference in curves between the two samples being compared (replicates or treatments).
- replicate.var: A column that has same name and type as the replicate.var column, represents the first replicate being compared. Note, a replicate column will be returned only when pool is FALSE or block.var = NULL.
- replicate.var2: A column that has the same type as the replicate.var column, and is named replicate.var with a 2 appended to it, represents the second replicate being compared. Note, a replicate.var column will be returned only when pool is FALSE and block.var = NULL.
- time.var: A column that has the same name and type as the time.var column, if time.var is specified.
- treatment.var: A column that has same name and type as the treatment.var column, represents the first treatment being compared. A treatment.var column will be returned when pool is TRUE or block.var is specified, or treatment.var is specified.
- treatment.var2: A column that has the same type as the treatment.var column, and is named treatment.var with a 2 appended to it, represents the second treatment being compared. A treatment.var column will be returned when pool is TRUE or block.var is specified, or treatment.var is specified.
- block.var: A column that has same name and type as the block.var column, if block.var is specified.

References

Avolio et al. Submitted to MEE

Examples

```
data(pplots)
# With block and no time
df <- subset(pplots, year == 2002 & block < 3)
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 treatment.var = "treatment",
                 block.var = "block",
                 replicate.var = "plot")

# With blocks and time
df <- subset(pplots, year < 2004 & block < 3)
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 treatment.var = "treatment",
                 block.var = "block",
                 replicate.var = "plot",
                 time.var = "year")

# With blocks, time, and reference treatment
df <- subset(pplots, year < 2004 & block < 3)
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 treatment.var = "treatment",
                 block.var = "block",
                 replicate.var = "plot",
                 time.var = "year",
                 reference.treatment = "N1P0")

# Pooling by treatment with time
df <- subset(pplots, year < 2004)
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 treatment.var = "treatment",
                 pool = TRUE,
                 replicate.var = "plot",
                 time.var = "year")

# All pairwise replicates with treatment
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 replicate.var = "plot",
                 time.var = "year",
                 treatment.var = "treatment")

# All pairwise replicates without treatment
```

```
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
curve_difference(df = df,
                 species.var = "species",
                 abundance.var = "relative_cover",
                 replicate.var = "plot",
                 time.var = "year")
```

`curve_dissim` *Rank-Abundance Curve Dissimilarity*

Description

A function to calculate the curve difference between two communities, given the empirical relative rank by abundance curve as a `stepfun`.

Usage

```
curve_dissim(sf, sf2)
```

Arguments

`sf`	The curve for the first community.
`sf2`	The curve for the second community.

`cyclic_shift` *Cyclic Shift Permutations*

Description

Performs a user-specified function on a null ecological community created via cyclic shift permutations (Harms et al. 2001, Hallett et al. 2014). The null community is formed by randomly selected different starting years for the time series of each species. This generates a null community matrix in which species abundances vary independently but within-species autocorrelation is maintained. The user-specified function must require a species x time input.

Usage

```
cyclic_shift(df, time.var, species.var, abundance.var, replicate.var = NA,
  FUN, bootnumber)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the replicate column. Defaults to NA, indicating no replicates (i.e., data are from a single plot).
FUN	A function to calculate on the null community
bootnumber	Number of null simulations to run

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables.

Value

The cyclic_shift function returns an S3 object of class "cyclic_shift" and parameter "out". The length of the "out" parameter is the number of null iterations as specified by bootnumber. If multiple replicates are specified, null values are averaged among replicates for each iteration, but a different cyclic shift permutation is applied to each replicate within an iteration.

References

Hallett, Lauren M., Joanna S. Hsu, Elsa E. Cleland, Scott L. Collins, Timothy L. Dickson, Emily C. Farrer, Laureano A. Gherardi, et al. "Biotic Mechanisms of Community Stability Shift along a Precipitation Gradient." Ecology 95, no. 6 (2014): 1693-1700.

Harms, Kyle E., Richard Condit, Stephen P. Hubbell, and Robin B. Foster. "Habitat Associations of Trees and Shrubs in a 50-Ha Neotropical Forest Plot." Journal of Ecology 89, no. 6 (2001): 947-59.

Examples

```
# Calculate a covariance matrix on a null community
data(knz_001d)
a1_cyclic <- cyclic_shift(subset(knz_001d, subplot == "A_1"),
                  time.var = "year",
                  species.var = "species",
                  abundance.var = "abundance",
                  FUN = cov,
                  bootnumber = 10)
```

cyclic_shift_onerep *A function to calculate a non-S3 cyclic shift on one replicate*

Description

A function to calculate a non-S3 cyclic shift on one replicate

Usage

```
cyclic_shift_onerep(df, time.var, species.var, abundance.var, FUN, bootnumber)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
FUN	A function to calculate on the null community
bootnumber	The number of null model iterations returned

Value

out A vector of test statistics calculated on the null community

df_intersect *Create intersected data frames*

Description

Create intersections.

Usage

```
df_intersect(df1, df2, dataname = "species")
```

Arguments

df1	A dataframe
df2	A dataframe
dataname	The name of the column on which the two datasets will be joined and intersected

EQ — *Utility function to calculate EQ evenness from Smith and Wilson 1996*

Description

Utility function to calculate EQ evenness from Smith and Wilson 1996

Usage

```
EQ(x)
```

Arguments

x	Vector of abundance of each species If all abundances are equal it returns a 1

Evar — *Evar*

Description

A utility function to calculate Evar from Smith and Wilson 1996

Usage

```
Evar(x, S = length(x))
```

Arguments

x	the vector of abundances of each species
S	the number of species in the sample

fill_species — *Add missing abundances for species absent from a replicate, on the assumption that any species in the* species.var *column should be included for every group defined by all the remaining columns except* abundance.var.

Description

Add missing abundances for species absent from a replicate, on the assumption that any species in the species.var column should be included for every group defined by all the remaining columns except abundance.var.

Usage

```
fill_species(df, species.var, abundance.var)
```

Arguments

df	A dataframe with species, abundances, and at least one other column to group by
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df

Value

A dataframe with the same columns as df, but with NA added for species that are present in df, but not in each group.

---

fill_zeros *Add zero abundances for missing species, on the assumption that any species in the* species.var *column should be included for every group defined by all the remaining columns save* abundance.var.

Description

Add zero abundances for missing species, on the assumption that any species in the species.var column should be included for every group defined by all the remaining columns save abundance.var.

Usage

```
fill_zeros(df, species.var, abundance.var)
```

Arguments

df	A dataframe with species, abundances, and at least one other column to group by
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df

Value

A dataframe with the same columns as df, but with zeros added for species that are present in df, but not in a particular group.

knz_001d *Data from Konza Prairie, watershed 001d*

Description

Plant composition within multiple replicates at an annually burned tallgrass prairie site in the Konza Prairie LTER, Manhattan KS (Watershed 001d).

Usage

```
data(knz_001d)
```

Format

A data frame with 8768 rows and 4 variables

Details

A data frame containing a column for species, year, subplot and abundance:

- species: A factor column of species sampled
- year: An integer column of sampling time points
- subplot: A factor column of spatial replicates with 20 levels
- abundance: A numeric column of abundance values

Source

Konza Prairie LTER Dataset ID: PVC02, watershed 1D

Collins, S. L. (2000) Disturbance frequency and community stability in native tallgrass prairie. American Naturalist 155:311-325.

lagged_distances *Get lagged distances for a single replicate*

Description

Returns a data frame with two columns, interval and distance. The interval is the number of time steps between two communities, while distance is the euclidean distance of community change within one replicate lagged across intervals.

Usage

```
lagged_distances(df, time.var, species.var, abundance.var)
```

Arguments

df	data frame to compute the slope of community change for
time.var	The name of the time column from df
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df

Value

a data frame containing of time lags by species distances

---

lagged_slope *Get slope Returns the slope of community change within one replicate.*

Description

Get slope Returns the slope of community change within one replicate.

Usage

```
lagged_slope(df, time.var, species.var, abundance.var)
```

Arguments

df	data frame to compute the slope of community change for
time.var	The name of the time column from df
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df

Value

a slope of time lags by species distances

lag_i — *Get lagged values from a distance matrix Get lagged values from distance matrix at value i*

Description

Get lagged values from a distance matrix Get lagged values from distance matrix at value i

Usage

```
lag_i(i, DM, rownums, colnums)
```

Arguments

i	the index of the matrix to lag
DM	the distance matrix from which lagged values are drawn
rownums	number of rows in the distance matrix
colnums	number of columns in the distance matrix

Value

the lagged values

mean_rank_shift — *Mean Rank Shifts*

Description

A measure of the relative change in species rank abundances, which indicates shifts in relative abundances over time (Collins et al. 2008). Mean rank shifts are calculated as the average difference in species' ranks between consecutive time periods, among species that are present across the entire time series.

Usage

```
mean_rank_shift(df, time.var, species.var, abundance.var,
  replicate.var = as.character(NA))
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

Value

mean_rank_shift returns a data frame with the following columns:

- time.var_pair: A factor column that returns the two time periods being compared, separated by a dash. The name of this column is the same as the time.var column in the input dataframe followed by "_pair".
- MRS: A numeric column with the mean rank shift values.
- replicate.var: A column that has same name and type as the replicate.var column, if replication is specified.

---

multivariate_change *Using dissimilarity-based metrics to calculate changes in composition and dispersion*

Description

Calculates the changes in composition and dispersion based off a Bray-Curtis dissimilarity matrix. Composition change is the euclidean distance between the centroids of compared time periods and ranges from 0-1, where 0 are identical communities, and 1 and completely different communities. Since composition change is based on plotted distance between centroids, it is context dependent and depends on how many centroids are being plotted, thus result of composition change depends on how many time periods are being measured. Dispersion change is the difference of average dispersion of each replicate to its centroid between time periods.

Usage

```
multivariate_change(df, time.var, species.var, abundance.var, replicate.var,
  treatment.var = NULL, reference.time = NULL)
```

Arguments

df	A data frame containing time, species, abundance and replicate columns and an optional column of treatment
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the replicate column. Replicate must be unique within the dataset and cannot be nested within treatments or blocks.

`treatment.var` the name of the optional treatment column

`reference.time` The name of the optional time point that all other time points should be compared to (e.g. the first year of data). If not specified, each comparison is between consecutive time points (e.g. first to second year, second to third year, etc.)

Value

The multivariate_change function returns a data frame with the following attributes:

- time.var: A column with the specified time.var and a second column, with '2' appended to the name. Time is subtracted from time2 for dispersion change.
- composition_change: A numeric column that is the euclidean distance between the centroids of two time points.
- dispersion_change: A numeric column that is the difference in the average dispersion of the replicates around the centroid for the two time periods. A negative value indicates replicates are converging over time (there is less dispersion at time period 2 than time period 1) and a positive value indicates replicates are diverging over time (there is more dispersion at time period 2 than time period 1.
- treatment.var: A column that has same name and type as the treatment.var column, if treatment.var is specified.

References

Avolio et al. 2015; Avolio et al. Submitted MEE, Mari Anderson et al. 2006.

Examples

```
data(pplots)
# With treatment
multivariate_change(pplots,
                    time.var="year",
                    replicate.var = "plot",
                    treatment.var = "treatment",
                    species.var = "species",
                    abundance.var = "relative_cover")
# In each year there are 6 replicates and there are 4 years of data for 3
# time comparisons, thus 24 total observations in each treatment.

# With treatment and reference year
multivariate_change(pplots,
                    time.var="year",
                    replicate.var = "plot",
                    treatment.var = "treatment",
                    species.var = "species",
                    abundance.var = "relative_cover",
                    reference.time = 2002)
# In each year there are 6 replicates and there are 4 years of data for 3
# time comparisons, thus 24 total observations in each treatment.

# Without treatment
```

```
df <- subset(pplots, treatment == "N1P0")
multivariate_change(df,
                    time.var="year",
                    replicate.var = "plot",
                    species.var = "species",
                    abundance.var = "relative_cover")
# In each year there are 6 replicates and there are 4 years of data for 3
# time comparisons, thus 24 total observations.
```

`multivariate_difference` *Using dissimilarity-based metrics to calculate differences in composition and dispersion between pairs of treatments at a single time point*

Description

Calculates the difference in composition and dispersion between treatments based off a Bray-Curtis dissimilarity matrix at a single point in time. Composition difference is the euclidean distance between the centroids of different treatments. Since centroid distance is based on plotted distance between centroids, it is context dependent and depends on how many centroids are being plotted. The centroid distance between treatments depends on how many treatments are being compared. Dispersion difference is the difference of average dispersion of each replicate to its centroid between two treatments.

Usage

```
multivariate_difference(df, time.var = NULL, species.var, abundance.var,
  replicate.var, treatment.var, reference.treatment = NULL)
```

Arguments

`df`	A data frame containing an optional time column, species, abundance and replicate, and treatment columns
`time.var`	The name of the optional time column
`species.var`	The name of the species column
`abundance.var`	The name of the abundance column
`replicate.var`	The name of the replicate column. Replicate must be unique within the dataset and cannot be nested within treatments or blocks.
`treatment.var`	the name of the treatment column
`reference.treatment`	The name of the optional treatment that all other treatments will be compared to (e.g. only controls will be compared to all other treatments). If not specified all pairwise treatment comparisons will be made.

Value

The multivariate_difference function returns a data frame with the following attributes:

- treatment.var: A column that has same name and type as the treatment.var column, if treatment.var is specified.
- treatment.var2: A column that has the same type as the treatment.var column, and is named treatment.var with a 2 appended to it.
- composition_diff: A numeric column that is the euclidean distance between the centroids of two treatments at a single point in time.
- abs_dispersion_diff: A numeric column that is the absolute value of the difference in the average dispersion of the replicates around the centroid for the two treatments.
- trt_greater_disp: A column that has same type as the treatment.var column, and specifies which of the two treatments has greater dispersion.
- time.var: A characteristic column that has the same name and type as the time.var column, if specified.

References

Avolio et al. Submitted to MEE, Avolio et al. 2015, Marti Anderson et al. 2006

Examples

```
data(pplots)
# Without time
df <- subset(pplots, year == 2002)
multivariate_difference(df,
                        replicate.var = "plot",
                        treatment.var = "treatment",
                        species.var = "species",
                        abundance.var = "relative_cover")
# There are 6 replicates for each of three treatments, thus 18 total
# observations.

# Without time and with reference treatment
df <- subset(pplots, year == 2002)
multivariate_difference(df,
                        replicate.var = "plot",
                        treatment.var = "treatment",
                        species.var = "species",
                        abundance.var = "relative_cover",
                        reference.treatment = "N1P0")
# There are 6 replicates for each of three treatments, thus 18 total
# observations.

# With time
multivariate_difference(pplots,
                        time.var = "year",
                        replicate.var = "plot",
                        species.var = "species",
```

```
                        abundance.var = "relative_cover",
                        treatment.var = "treatment")
# In each year there are 6 replicates for each of three treatments, for a
# total of 18 observations.
```

---

pplots *Phosphorus plots data from Avolio et al. 2014*

Description

A dataset of tallgrass prairie plant composition in a nitrogen and phosphorus addition experiment at Konza Prairie, Manhattan Kansas (Avolio et al. 2014). This dataset is a subset of the full dataset.

Usage

```
data(pplots)
```

Format

A data frame with 1232 rows and 6 variables

Details

A data frame containing a column for replicate, year, species, abundance, block and treatment :

- plot: An integer column of spatial replicates with 18 levels (6-48)
- year: An integer column of sampling time points
- species: A factor column of species sampled
- relative_cover: A numeric column of relative cover values
- block: An integer column of dummy blocking variable, grouping treatment plots into blocks
- treatment: A factor column of nitrogen and phosphorus treatments applied to the plots

Source

Avolio, ML, Koerner, S, La Pierre, K, Wilcox, K, Wilson, GTW, Smith, MD, Collins, S. 2014. Changes in plant community composition, not diversity, to a decade of nitrogen and phosphorus additions drive changes in aboveground productivity in a tallgrass prairie. Journal of Ecology. 102: 1649-1660.

RAC_change *Rank Abundance Curve Changes*

Description

Calculates change of the five aspects of rank abundance curves (richness, evenness, rank, species gains, and species losses) for a replicate between two time points.

Usage

```
RAC_change(df, time.var, species.var, abundance.var, replicate.var = NULL,
  reference.time = NULL)
```

Arguments

df	A data frame containing time, species, and abundance columns and an optional columns of replicates.
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column. If specified, replicate must be unique within the dataset and cannot be nested within treatments or blocks.
reference.time	The name of the optional time point that all other time points should be compared to (e.g. the first year of data). If not specified, each comparison is between consecutive time points (the first and second year, second and third year, etc.)

Value

The RAC_change function returns a data frame with the following attributes:

- replicate.var: A column that has same name and type as the replicate.var column, if replicate.var is specified.
- time.var: A column with the specified time.var and a second column, with '2' appended to the name. Time is subtracted from time2.
- richness_change: A numeric column that is the change in richness between the two time periods for a replicate divided by the total number of unique species in both time periods. A positive value occurs when a there is an increase in species richness over time, and a negative value when there is a decreases in species richness over time.
- evenness_change: A numeric column that is the change in evenness(measured with Evar) between the two time periods for a replicate. A positive value occurs when evenness increases over time, and a negative value when evenness decreases in over time.
- rank_change: A numeric column that is the absolute value of the average change in species ranks between the two time periods for a replicate divided by the total number of unique species in both time periods. Species that are not present in both time periods are given the S+1 rank in the sample it is absent in, where S is the number of species in that sample.

- gains: A numeric column of the number of species that are present at time period 2 that were not present at time period 1 for a replicate divided by the total number of unique species in both time periods. This is equivalent to the turnover function with metric = "appearances".
- losses: A numeric column of the number of species that are not present at time period 2 but were present at time period 1 for a replicate divided by the total number of unique species in both time periods. This is equivalent to the turnover function with metric = "disappearance".

References

Avolio et al. submitted to MEE

Examples

```
data(pplots)
# Without replicates
df <- subset(pplots, plot == 25)
RAC_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           time.var = "year")

# With replicates
df <- subset(pplots, year < 2004 & plot %in% c(6, 25, 32))
RAC_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           replicate.var = "plot",
           time.var = "year")

# With reference year
df <- subset(pplots, year < 2005 & plot %in% c(6, 25, 32))
RAC_change(df = df,
           species.var = "species",
           abundance.var = "relative_cover",
           replicate.var = "plot",
           time.var = "year",
           reference.time = 2002)
```

RAC_difference *Rank Abundance Curve Differences*

Description

Calculates differences between two samples for four comparable aspects of rank abundance curves (richness, evenness, rank, species composition). There are three ways differences can be calculated. 1) Between treatments within a block (note: block.var and treatment.var need to be specified). 2) Between treatments, pooling all replicates into a single species pool (note: pool = TRUE, treatment.var needs to be specified, and block.var will be NULL). 3) All pairwise combinations between

all replicates (note: block.var = NULL, pool = FALSE and specifying treatment.var is optional. If treatment.var is specified, the treatment that each replicate belongs to will also be listed in the output).

Usage

```
RAC_difference(df, time.var = NULL, species.var, abundance.var, replicate.var,
  treatment.var = NULL, pool = FALSE, block.var = NULL,
  reference.treatment = NULL)
```

Arguments

df	A data frame containing a species, abundance, and replicate columns and optional time, treatment, and block columns
time.var	The name of the optional time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the replicate column. Replicate must be unique within the dataset and cannot be nested within treatments or blocks.
treatment.var	The name of the optional treatment column
pool	An argument to allow abundance values to be pooled within a treatment. The default value is "FALSE", a value of "TRUE" averages abundance of each species within a treatment at a given time point.
block.var	The name of the optional block column
reference.treatment	The name of the optional treatment that all other treatments will be compared to (e.g. only controls will be compared to all other treatments). If not specified all pairwise treatment comparisons will be made.

Value

The RAC_difference function returns a data frame with the following attributes:

- time.var: A column that has the same name and type as the time.var column, if time.var is specified.
- block.var: A column that has same name and type as the block.var column, if block.var is specified.
- replicate.var: A column that has same name and type as the replicate.var column, represents the first replicate being compared. Note, a replicate column will be returned only when pool is FALSE or block.var = NULL.
- replicate.var2: A column that has the same type as the replicate.var column, and is named replicate.var with a 2 appended to it, represents the second replicate being compared. Note, a replicate.var column will be returned only when pool is FALSE and block.var = NULL.
- treatment.var: A column that has the same name and type as the treatment.var column, represents the first treatment being compared. A treatment.var column will be returned when pool is TRUE or block.var is present, or treatment.var is specified.

- treatment.var2: A column that has the same type as the treatment.var column, and is named treatment.var with a 2 appended to it, represents the second treatment being compared. A treatment.var column will be returned when pool is TRUE or block.var is present, or treatment.var is specified.
- richness_diff: A numeric column that is the difference between the compared samples (treatments or replicates) in species richness divided by the total number of unique species in both samples. A positive value occurs when there is greater species richness in replicate.var2 than replicate.var or treatment.var2 than treatment.var.
- evenness_diff: A numeric column of the difference between the compared samples (treatments or replicates) in evenness (measured using the Evar metric). A positive value occurs when there is greater evenness in replicate.var2 than replicate.var or treatment.var2 than treatment.var.
- rank_diff: A numeric column of the absolute value of average difference between the compared samples (treatments or replicates) in species' ranks divided by the total number of unique species in both samples.Species that are not present in both samples are given the S+1 rank in the sample it is absent in, where S is the number of species in that sample.
- species_diff: A numeric column of the number of species that are different between the compared samples (treatments or replicates) divided by the total number of species in both samples. This is equivalent to the Jaccard Index.

References

Avolio et al. Submitted MEE

Examples

```
data(pplots)
# With block and no time
df <- subset(pplots, year == 2002 & block < 3)
RAC_difference(df = df,
               species.var = "species",
               abundance.var = "relative_cover",
               treatment.var = 'treatment',
               block.var = "block",
               replicate.var = "plot")

# With blocks and time
df <- subset(pplots, year < 2004 & block < 3)
RAC_difference(df = df,
               species.var = "species",
               abundance.var = "relative_cover",
               treatment.var = 'treatment',
               block.var = "block",
               replicate.var = "plot",
               time.var = "year")

# With blocks, time and reference treatment
df <- subset(pplots, year < 2004 & block < 3)
RAC_difference(df = df,
               species.var = "species",
```

```
                abundance.var = "relative_cover",
                treatment.var = 'treatment',
                block.var = "block",
                replicate.var = "plot",
                time.var = "year",
                reference.treatment = "N1P0")

# Pooling by treatment with time
df <- subset(pplots, year < 2004)
RAC_difference(df = df,
                species.var = "species",
                abundance.var = "relative_cover",
                treatment.var = 'treatment',
                pool = TRUE,
                replicate.var = "plot",
                time.var = "year")

# All pairwise replicates with treatment
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
RAC_difference(df = df,
                species.var = "species",
                abundance.var = "relative_cover",
                replicate.var = "plot",
                time.var = "year",
                treatment.var = "treatment")

# All pairwise replicates without treatment
df <- subset(pplots, year < 2004 & plot %in% c(21, 25, 32))
RAC_difference(df = df,
                species.var = "species",
                abundance.var = "relative_cover",
                replicate.var = "plot",
                time.var = "year")
```

`rank_onerep` *Function for calculating mean rank shifts*

Description

This is a function that calculates mean rank shifts

Usage

```
rank_onerep(df, time.var, species.var, abundance.var)
```

Arguments

`df`	dataframe of Community dataset. Must be in 'long' format.
`time.var`	The time variable
`species.var`	The species variable
`abundance.var`	The abundance variable

Value

a dataframe, showing years compared

`rank_shift` *Mean Rank Shifts*

Description

A measure of the relative change in species rank abundances, which indicates shifts in relative abundances over time (Collins et al. 2008). Mean rank shifts are calculated as the average difference in species' ranks between consecutive time periods, among species that are present across the entire time series.

Usage

```
rank_shift(df, time.var, species.var, abundance.var,
  replicate.var = as.character(NA))
```

Arguments

`df`	A data frame containing time, species and abundance columns and an optional column of replicates
`time.var`	The name of the time column
`species.var`	The name of the species column
`abundance.var`	The name of the abundance column
`replicate.var`	The name of the optional replicate column

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

Value

rank_shift returns a data frame with the following columns:

- time.var_pair: A factor column that returns the two time periods being compared, separated by a dash. The name of this column is the same as the time.var column in the input dataframe followed by "_pair".
- MRS: A numeric column with the mean rank shift values.
- replicate.var: A column that has same name and type as the replicate.var column, if replication is specified.

References

Collins, Scott L., Katharine N. Suding, Elsa E. Cleland, Michael Batty, Steven C. Pennings, Katherine L. Gross, James B. Grace, Laura Gough, Joe E. Fargione, and Christopher M. Clark. (2008) "Rank clocks and plant community dynamics." Ecology 89, no. 12: 3534-41.

Examples

```
# Calculate mean rank shifts within replicates
data(knz_001d)

myoutput <- rank_shift(knz_001d,
                    time.var = "year",
                    species.var = "species",
                    abundance.var = "abundance",
                    replicate.var = "subplot")

# Calculate mean rank shifts for a data frame with no replication

myoutput_singlerep <- rank_shift(subset(knz_001d, subplot=="A_1"),
                          time.var = "year",
                          species.var = "species",
                          abundance.var = "abundance")
```

---

rate_change *Rate of community change over successive time intervals*

Description

Calculates the slope of the differences in species composition within a community over increasing time intervals, which provides a measures of the rate of directional change in community composition. Differences in species composition are characterized by Euclidean distances, which are calculated on pair-wise communities across the entire time series. For example, a data set with six time intervals will have distance values for five one-year time lags (year 1 vs year 2, year 2 vs year 3 ...), four two-year time lags (year 1 vs year 3, year 2 vs year 4 ...) and so forth. These distance values are regressed against the time lag interval. The slope of the regression line is reported as an indication of the rate and direction of compositional change in the community.

Usage

```
rate_change(df, time.var, species.var, abundance.var, replicate.var = NA)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

The `rate_change` function uses linear regression to relate Euclidean distances to time lag intervals. It is recommended that fit of this relationship be verified using `rate_change_interval`, which returns the full set of community distance values and associated time lag intervals.

Value

The `rate_change` function returns a numeric rate change value unless a replication column is specified in the input data frame. If replication is specified, the function returns a data frame with the following attributes:

- rate_change: A numeric column with the synchrony values.
- replicate.var: A column that shares the same name and type as the replicate.var column in the input data frame.

References

Collins, S. L., Micheli, F. and Hartt, L. 2000. A method to determine rates and patterns of variability in ecological communities. - Oikos 91: 285-293.

Examples

```
data(knz_001d)
rate_change(knz_001d[knz_001d$subplot=="A_1",],
                       time.var = "year",
                       species.var = "species",
                       abundance.var = "abundance") # for one subplot

rate_change(knz_001d,
             time.var = "year",
             species.var = "species",
             abundance.var = "abundance",
             replicate.var = "subplot") # across all subplots
```

`rate_change_interval` *Differences in community composition over successive time lag intervals*

Description

Calculates the differences in species composition within a community over increasing time intervals. Differences in species composition are characterized by Euclidean distances, which are calculated on pair-wise communities across the entire time series. For example, a data set with 6 time intervals will have distance values for five one-year time lags (year 1 vs year 2, year 2 vs year 3 ...), 4 two-year time lags (year 1 vs year 3, year 2 vs year 4 ...) and so forth. Returns the full set of community distance values and associated time lag intervals.

Usage

```
rate_change_interval(df, time.var, species.var, abundance.var,
  replicate.var = NA)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column

Value

The rate_change_interval function returns a data frame with the following attributes:

- interval: A numeric column containing the interval length between time periods.
- distance: A numeric column containing the Euclidean distances.
- replicate.var: A column that shares the same name and type as the replicate.var column in the input data frame.

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

References

Collins, S. L., Micheli, F. and Hartt, L. 2000. A method to determine rates and patterns of variability in ecological communities. - Oikos 91: 285-293.

Examples

```
data(knz_001d)
rate_change_interval(knz_001d[knz_001d$subplot=="A_1",],
                     time.var = "year",
                     species.var = "species",
```

```
                      abundance.var = "abundance") # for one subplot


rate_change_interval(knz_001d,
                      time.var = "year",
                      species.var = "species",
                      abundance.var = "abundance",
                       replicate.var = "subplot") # across all subplots
```

S *Utility function to calculate richness*

Description

Utility function to calculate richness

Usage

```
S(x)
```

Arguments

x	Vector of abundance of each species

---

shuffle_community *A function to generate a community dataframe with a random start time for each species*

Description

A function to generate a community dataframe with a random start time for each species

Usage

```
shuffle_community(comdat)
```

Arguments

comdat	A community dataframe

Value

rand.comdat A randomized community dataframe

split_apply_combine *A Split-Apply-Combine implementation*

Description

Faster split-apply-combine for data frames, when the results of FUN are homogeneous with respect to the number, order, data type and (if applicable) levels of columns in the returned data frame.

Usage

```
split_apply_combine(df, by, FUN, ...)
```

Arguments

df	A data frame
by	The name of column(s) in the data frame that define groups to split
FUN	The function applied to each grouped data frame after splitting
...	Additional parameters to FUN

Source

https://stackoverflow.com/a/9730292/687112

stability_onerep *A function to calculate species synchrony over time within one replicate*

Description

A function to calculate species synchrony over time within one replicate

Usage

```
stability_onerep(df, x)
```

Arguments

df	A dataframe containing x column
x	The column to calculate stability on

Value

Stability of x, calculated as the mean/sd

synchrony *Species synchrony*

Description

Calculates the degree synchrony in species abundances within a community over time. Includes the option for two different synchrony metrics. The first, developed by Loreau and de Mazancourt (2008), compares the variance of the aggregated community with the variance of individual components. The second, developed by Gross et al. (2014), compares the average correlation of each individual species with the rest of the aggregated community.

Usage

```
synchrony(df, time.var, species.var, abundance.var, metric = "Loreau",
  replicate.var = NA)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
metric	The synchrony metric to return: • "Loreau": The default metric, calculates synchrony following Loreau and de Mazancourt (2008). • "Gross": Calculates synchrony following Gross et al. (2014).
replicate.var	The name of the optional replicate column

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

Value

The synchrony function returns a numeric synchrony value unless a replication column is specified in the input data frame. If replication is specified, the function returns a data frame with the following attributes:

- synchrony: A numeric column with the synchrony values.
- replicate.var: A column that shares the same name and type as the replicate.var column in the input data frame.

References

Gross, Kevin, Bradley J. Cardinale, Jeremy W. Fox, Andrew Gonzalez, Michel Loreau, H. Wayne Polley, Peter B. Reich, and Jasper van Ruijven. (2014) "Species richness and the temporal stability of biomass production: A new analysis of recent biodiversity experiments." The American Naturalist 183, no. 1: 1-12. doi:10.1086/673915.

Loreau, Michel, and Claire de Mazancourt. (2008) "Species synchrony and its drivers: Neutral and nonneutral community dynamics in fluctuating environments." The American Naturalist 172, no. 2: E48-66. doi:10.1086/589746.

Examples

```
data(knz_001d)
synchrony(knz_001d[knz_001d$subplot=="A_1",],
                    time.var = "year",
                    species.var = "species",
                    abundance.var = "abundance") # for one subplot
## Not run:
synchrony(knz_001d,
             time.var = "year",
             species.var = "species",
             abundance.var = "abundance",
             replicate.var = "subplot") # across all subplots

synchrony(knz_001d,
             time.var = "year",
             species.var = "species",
             abundance.var = "abundance",
             replicate.var = "subplot",
             metric="Gross") # With Gross et al. (2014) metric.

## End(Not run)
```

---

synch_onerep *A function to calculate species synchrony over time within one replicate*

Description

A function to calculate species synchrony over time within one replicate

Usage

```
synch_onerep(df, time.var, species.var, abundance.var, metric = "Loreau")
```

Arguments

df	A dataframe containing rep, time, species and abundance columns
time.var	The name of the time column from df
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df
metric	The synchrony metric to return. The default, "Loreau", returns synchrony as calculated by Loreau and de Mazancourt 2008. The alternative, "Gross", returns synchrony as calculated by Gross et al. 2014

Value

output The degree of species synchrony. If "Loreau", 1 is perfect synchrony and 0 is perfect asynchrony. If "Gross", 1 is perfect synchrony and -1 is perfect asynchrony.

---

transpose_community *Convert from a long form abundance dataframe to a time by species dataframe.*

Description

Convert from a long form abundance dataframe to a time by species dataframe.

Usage

```
transpose_community(df, time.var, species.var, abundance.var)
```

Arguments

df	A dataframe containing time.var, species.var and abundance.var columns
time.var	The name of the time column from df
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df

Value

A dataframe of species abundances x time

turnover *Species Turnover*

Description

Computes species turnover between time periods as the proportion of species either gained or lost relative to the total number of species observed across both time periods. Includes an option to compute turnover as just the proportion of species gained (i.e., "appearances") or lost (i.e., "disappearances").

Usage

```
turnover(df, time.var, species.var, abundance.var, replicate.var = NA,
  metric = "total")
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
replicate.var	The name of the optional replicate column
metric	The turnover metric to return: • total: The default metric, calculates summed appearances and disappearances relative to total species richness across both time periods. • appearance: Calculates the number of species that appeared in the second time period relative to total species richness across both time periods. • disappearance: Calculates the number of species that disappeared in the second time period relative to total species richness across both time periods.

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

Value

The turnover function returns a data frame with the following attributes:

- turnover: A numeric column with the turnover values. The name of this column is the same as the specified metric (default is "total").

- time.var: A column containing the second time point; the name and type of this column is the same as the time.var column in the input dataframe.
- replicate.var: A column that has same name and type as the replicate.var column, if replication is specified.

References

Cleland, Elsa E., Scott L. Collins, Timothy L. Dickson, Emily C. Farrer, Katherine L. Gross, Laureano A. Gherardi, Lauren M. Hallett, et al. (2013) "Sensitivity of grassland plant community composition to spatial vs. temporal variation in precipitation." Ecology 94, no. 8: 1687-96.

Examples

```
data(knz_001d)

# Calculate relative total turnover within replicates
total.res <- turnover(df=knz_001d,
              time.var = "year",
              species.var = "species",
              abundance.var = "abundance",
              replicate.var="subplot")

# Calculate relative species appearances within replicates
appear.res <- turnover(df=knz_001d,
              time.var = "year",
              species.var = "species",
              abundance.var = "abundance",
              replicate.var="subplot",
              metric="appearance")

# Calculate relative species disappearances within replicates
disappear.res <- turnover(df=knz_001d,
              time.var = "year",
              species.var = "species",
              abundance.var = "abundance",
              replicate.var="subplot",
              metric="disappearance")
```

---

turnover_allyears *A function to calculate species turnover between years*

Description

A function to calculate species turnover between years

Usage

```
turnover_allyears(df, time.var, species.var, abundance.var,
  metric = c("total", "disappearance", "appearance"))
```

Arguments

df	A dataframe containing time, species and abundance columns
time.var	The name of the time column from df
species.var	The name of the species column from df
abundance.var	The name of the abundance column from df
metric	The turnover metric to return; the default, total, returns summed appearances and disappearances relative to total species richness across both years • appearance: returns the number of appearances in the second year relative to total species richness across both years • disappearance: returns the number of disappearances in the second year relative to the total species richness across both years

Value

output A dataframe containing the specified turnover metric and year

---

turnover_twoyears *A function to calculate species turnover between two years*

Description

A function to calculate species turnover between two years

Usage

```
turnover_twoyears(d1, d2, species.var, metric = c("total", "disappearance",
  "appearance"))
```

Arguments

d1	A dataframe containing a species column from one year
d2	A dataframe containing a species column from the following year
species.var	The name of the species column in d1 and d2
metric	The turnover metric to return; the default, total, returns summed appearances and disappearances relative to total species richness across both years • appearance: returns the number of appearances in the second year relative to total species richness across both years • disappearance: returns the number of disappearances in the second year relative to the total species richness across both years

Value

output The specified turnover metric

variance_ratio *Variance Ratio*

Description

Computes the ratio of the variance of aggregate species abundances in a community to the sum of the variances of individual, component species. A variance ratio = 1 indicates that species do not covary, a variance ratio > 1 indicates predominately positive covariance among species and a variance ratio < 1 indicates predominately negative covariance (Schluter 1984).

Includes a cyclic shift null modeling option to test if the variance ratio significantly differs from 1. The null community is created by randomly selecting different starting points for each species' time series, which generates a community in which species abundances vary independently but within-species autocorrelation is maintained (Hallett et al. 2014). This randomization is repeated a user-specific number of times and confidence intervals are reported for the resultant null distribution of variance ratios. If the dataframe includes multiple replicates, the variance ratios for the actual and null communities are averaged within each iteration unless specified otherwise.

Usage

```
variance_ratio(df, time.var, species.var, abundance.var, bootnumber,
  replicate.var = NA, average.replicates = TRUE, level = 0.95, li, ui)
```

Arguments

df	A data frame containing time, species and abundance columns and an optional column of replicates
time.var	The name of the time column
species.var	The name of the species column
abundance.var	The name of the abundance column
bootnumber	The number of null model iterations used to calculated confidence intervals
replicate.var	The name of the (optional) replicate column
average.replicates	If true returns the variance ratio and CIs averaged
level	The confidence level for the null mean
li	(deprecated) lower confidence interval
ui	(deprecated) upper confidence interval across replicates; if false returns the variance ratio and CI for each replicate. Defaults to true.

Details

The input data frame needs to contain columns for time, species and abundance; time.var, species.var and abundance.var are used to indicate which columns contain those variables. If multiple replicates are included in the data frame, that column should be specified with replicate.var. Each replicate

should reflect a single experimental unit - there must be a single abundance value per species within each time point and replicate.

Null model confidence intervals default to the standard lowest 2.5% and upper 97.5% of the null distribution, typically these do not need to be change, but they can be user-modified to set more stringent CIs. @references Hallett, Lauren M., Joanna S. Hsu, Elsa E. Cleland, Scott L. Collins, Timothy L. Dickson, Emily C. Farrer, Laureano A. Gherardi, et al. (2014) "Biotic Mechanisms of Community Stability Shift along a Precipitation Gradient." Ecology 95, no. 6: 1693-1700. doi: 10.1890/13-0895.1

Schluter, Dolph. (1984) "A Variance Test for Detecting Species Associations, with Some Example Applications." Ecology 65, no. 3: 998-1005. doi:10.2307/1938071.

Value

The variance_ratio function returns a data frame with the following attributes:

- VR: A numeric column with the actual variance ratio value.
- lowerCI: A numeric column with the lowest confidence interval value.
- upperCI: A numeric column with the highest confidence interval value.
- nullmean: A numeric column with the average null variance ratio value.
- replicate.var: A column that has same name and type as the replicate.var column, if replication is specified.

Examples

```
data(knz_001d)

# Calculate the variance ratio and CIs averaged within replicates
# Here the null model is repeated once, for final use it is recommended to set a
# large bootnumber (eg, 10000)

res_averagedreplicates <- variance_ratio(knz_001d,
              time.var = "year",
              species.var = "species",
              abundance.var = "abundance",
              bootnumber = 1,
              replicate = "subplot")

#Calculate the variance ratio and CIs for each replicate

res_withinreplicates <- variance_ratio(knz_001d,
              time.var = "year",
              species.var = "species",
              abundance.var = "abundance",
              bootnumber = 1,
              replicate = "subplot",
              average.replicates = FALSE)
```

variance_ratio_longformdata

A function to calculate the variance ratio from a long form dataframe

Description

A function to calculate the variance ratio from a long form dataframe

Usage

```
variance_ratio_longformdata(df, time.var, species.var, abundance.var)
```

Arguments

df	A dataframe containing time.var, replicate.var, species.var and abundance.var columns
time.var	The name of the time.var column from df
species.var	The name of the species.var column from df
abundance.var	The name of the abundance.var column from df

Value

var.ratio The variance ratio of the community

variance_ratio_matrixdata

A function to calculate the variance ratio

Description

A function to calculate the variance ratio

Usage

```
variance_ratio_matrixdata(comdat)
```

Arguments

comdat	A community dataframe

Value

var.ratio The variance ratio of the community

Index