Package 'DeadCanMove'

July 20, 2015

Type Package

Title Assess how spatial roadkill patterns change with temporal sampling scheme

Version 0.2

Date 2015-05-8

Author Barbosa A.M., Marques J.T., Santos S.M., Lourenco A., Medinas D., Beja P., Mira A.

Maintainer A. Marcia Barbosa <barbosa@uevora.pt>

Description

From a baseline data frame of dead individuals recorded daily at different road stretches, simulate varying sub-sampling schemes, calculate roadkill hotspots based on each scheme, and compare hotspot patterns between sampling schemes and the baseline data.

License GPL-3

R topics documented:

`DeadCanMove-package` *Assess how spatial roadkill patterns change with temporal sampling scheme*

Description

From a baseline data frame of dead individuals recorded daily at different road stretches, simulate varying sub-sampling schemes, calculate roadkill hotspots based on each scheme, and compare hotspot patterns between sampling schemes and the baseline data.

Details

Package:	DeadCanMove
Type:	Package
Version:	0.2
Date:	2015-05-8
License:	GPL-3

Author(s)

Barbosa A.M., Marques J.T., Santos S.M., Lourenco A., Medinas D., Beja P., Mira A.

Maintainer: A. Marcia Barbosa <barbosa@uevora.pt>

References

Santos S.M., Marques J.T., Lourenco A., Medinas D., Barbosa A.M., Beja P., Mira A. (in review) Sampling effects on the identification of roadkill hotspots and implications for survey design.

Examples

```
data(roadkills)

hc <- hotspots.comparison(dataset = roadkills,
sampl.columns = 4:ncol(roadkills), sampl.intervals = 1:5,
region.column = "segment", group.column = "taxon", include.all.together = TRUE,
confidence = 0.95, min.total.events = 80, min.hotspot.threshold = 2,
comp.method = "Phi", plot = TRUE, sep.plots = FALSE,
omit.baseline.interval = TRUE, ylim = c(0, 1))

hc
```

```
## Not run:
# calculate hotspots for the seqsubmats:

seqsubmats.hs <- vector("list", length(seqsubmats))
for (g in 1:length(seqsubmats)) {
  seqsubmats.hs[[g]] <- sequential.hotspots(dataset = roadkills, submats = seqsubmats[[g]], region.column = "segm
}; rm(g)
names(seqsubmats.hs) <- paste0(names(seqsubmats), ".hs")


# calculate hotspot correlation, loss, gain or balance for different replicates
# per sampling scheme and animal group:

replicate.corrs <- repl.hs.comp(seqsubmats.hs = seqsubmats.hs,
hs.baseline = hs.baseline, method = "Phi")

replicate.gains <- repl.hs.comp(seqsubmats.hs = seqsubmats.hs,
hs.baseline = hs.baseline, method = "gain")

## End(Not run)
```

`binary.comp.methods` *Binary comparison methods*

Description

This function provides the methods implemented for calculating binary similarity between hotspots obtained from different sampling schemes and those obtained from the baseline (reference) sampling scheme. It is used by functions `binary.comparison`, `sequential.corr` and `hotspots.comparison`.

Usage

```
binary.comp.methods()
```

Value

A character vector naming the methods implemented.

Author(s)

A. Marcia Barbosa

See Also

`binary.comparison`, `sequential.corr`

Examples

```
binary.comp.methods()
```

`binary.comparison` *Binary comparison*

Description

Compares two binary vectors using the coefficient specified in `method`.

Usage

```
binary.comparison(x, y, method)
```

Arguments

`x`	a binary (0-1) vector
`y`	a binary (0-1) vector to compare with `x`
`method`	the comparison measure to use. Current options are "Phi", "Mathews", "Yule", "Jaccard", "Baroni", "kappa", "CCR", "TSS", "gain", "loss", and "balance".

Value

A numeric value indicating the cofficient of association specified in `method`.

Author(s)

A. Marcia Barbosa

See Also

`binary.comp.methods`

Examples

```
bin1 <- sample(c(0, 1), 100, replace = TRUE)
bin2 <- sample(c(0, 1), 100, replace = TRUE)

binary.comparison(bin1, bin2, method = "Phi")
binary.comparison(bin1, bin2, method = "kappa")
```

getBoxplots *Get boxplots*

Description

Get boxplots

Usage

```
getBoxplots(corrs.list, ...)
```

Arguments

corrs.list	a list of corrs.tables given by function schemeCorrs
...	additional arguments for the boxplot function (e.g. ylim = c(0, 1), las = 2, main = as.character(bquote(corrs.list)))

Value

Box plots

Author(s)

A. Marcia Barbosa

See Also

schemeCorrs

hotspot.numbers *Hotspot numbers*

Description

This function calculates the total numbers of events and the hotspot thresholds for each group and sampling interval.

Usage

```
hotspot.numbers(hotspots.list, sampl.intervals, groups, include.all.together =
TRUE,
min.total.events = 0, min.hotspot.threshold = 2)
```

Arguments

`hotspots.list`	results of the sequential.hotspots function
`sampl.intervals`	integer vector of the sampling intervals to analyse (at the moment, these intervals must be consecutive and start with one)
`groups`	taxa or groups to analyse separately (e.g. as.character(unique(dataset$group)))
`include.all.together`	logical, whether to run the analysis also for all groups combined
`min.total.events`	minimum total number of events to calculate hotspots for a group
`min.hotspot.threshold`	minimum number of events for a region to be considered a hotspot

Value

A list of the following matrices:

`N.events`

`HS.threshold`

`N.hotspots`

`events.in.HS`

Author(s)

A. Marcia Barbosa, J. Tiago Marques, Sara M. Santos

See Also

`hotspots`, `sequential.hotspots`

Examples

```
data(roadkills)

submats <- sequential.submatrix(dataset = roadkills,
sampl.columns = 4:ncol(roadkills), window.size = 1, gap.size = 0:2,
group.column = "taxon", include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = 120)

hsl <- sequential.hotspots(dataset = roadkills, submats = submats,
region.column = "segment", first.subsampl.col = 4, confidence = 0.95)

hsn <- hotspot.numbers(hotspots.list = hsl, sampl.intervals = 1:3,
groups = as.character(unique(roadkills$taxon)), include.all.together = TRUE,
min.hotspot.threshold = 2)

hsn
```

`hotspots` *Calculate roadkill hotspots*

Description

This function identifies the hotspot regions in a dataset, or in a submatrix compared to the total dataset, using an adaptation of the method of Malo et al. (2004).

Usage

```
hotspots(dataset, submat = NULL, region.column, subsampl.columns,
hotspots = TRUE, confidence = 0.95, min.total.events = 0, min.hotspot.threshold = 2)
```

Arguments

`dataset`	name of the matrix or dataframe containing the complete data
`submat`	name of the matrix or dataframe containing the data of the group and sampling window/gap for which to calculate hotspots
`region.column`	name or index number of the column containing the regions (road sectors, sites) to classify as hotspots or non-hotspots
`subsampl.columns`	index numbers of the consecutive columns of submat (or, if there is no submat, of the dataset) containing the (daily) sampling data, e.g. 4:180
`hotspots`	logical, whether to calculate the hotspots
`confidence`	confidence threshold to consider hotspots
`min.total.events`	minimum total number of events to calculate hotspots. Not totally implemented yet!
`min.hotspot.threshold`	minimum number of events for a region to be considered a hotspot. If the Malo method says that regions with less than this value are hotspots, the value returned is NA. The default threshold is 2.

Value

A list with elements `threshold` (an integer value indicating the number of deaths obtained as a threshold for considering a site a roadkill hotspot) and `hotspots` (a data frame showing the total number of deaths per region and whether or not it was considered a hospot.)

Author(s)

A. Marcia Barbosa, J. Tiago Marques, Sara M. Santos

References

Malo, J.E., Suarez, F., Diez, A. (2004) Can we mitigate animal-vehicle accidents using predictive models? J. Appl. Ecol. 41, 701-710 (doi: 10.1111/j.0021-8901.2004.00929.x)

See Also

sequential.hotspots

Examples

```
data(roadkills)

hs <- hotspots(dataset = roadkills, submat = NULL, region.column = "segment",
subsampl.columns = 4:ncol(roadkills), confidence = 0.95)

hs
```

---

hotspots.comparison *Hotspots comparison*

Description

This is a wrapper for most of the functions in this package (one function to rule them all). You'll probably only need to use this one, which in turn calls each of the other functions and does all the calculations in one step.

Usage

```
hotspots.comparison(dataset, sampl.columns, sampl.intervals, region.column,
group.column, include.all.together = TRUE, confidence = 0.95,
min.total.events = 80, min.hotspot.threshold = 2, comp.method = "Phi",
plot = TRUE, sep.plots = FALSE, omit.baseline.interval = TRUE, ...)
```

Arguments

dataset	name of the matrix or dataframe to analyze
sampl.columns	index numbers of the columns containing the (daily) sampling data, e.g. 4:180
sampl.intervals	intervals at which to extract sampling data, e.g. 1:30; currently must be consecutive and start with 1
region.column	name or index number of the column containing the regions (road segments, sites) to classify as hotspots or non-hotspots
group.column	name or index number of the column containing the taxa or groups to analyse separately, e.g. 3 or "Family"; if NULL, all records will be used together
include.all.together	logical, whether to get subsampling matrices also for the complete data (including all groups combined)
confidence	confidence threshold to consider hotspots (see Malo et al. 2004); defaults to 0.95
min.total.events	minimum total number of events (e.g. deaths) to calculate hotspots for a group

`min.hotspot.threshold`
: minimum number of events for a region to be considered a hotspot

`comp.method`
: the method with which to compare the hotspots obtained with increasing sampl.intervals with those of the baseline scenario; type `binary.comp.methods()` for available options

`plot`
: logical, whether to plot the correlations between subsamples and baseline for each group (may cause function to fail if `sep.plots = FALSE` and figure margins are too large for the number of resulting plots)

`sep.plots`
: logical, whether to present the plots in separate windows rather than all in the same window

`omit.baseline.interval`
: logical, whether to omit the first column (correlation of baseline hotspots with themselves) from calculations and results

`...`
: additional arguments to pass to the `plot` function

Value

A list with 9 elements:

`hotspots.list`

`N.events`

`HS.threshold`

`N.hotspots`

`events.in.HS`

`event.corrs`

`event.loss`

`event.gain`

`event.balance`

Author(s)

A. Marcia Barbosa, J. Tiago Marques, Sara M. Santos

References

Malo, J.E., Suarez, F., Diez, A. (2004) Can we mitigate animal-vehicle accidents using predictive models? J. Appl. Ecol. 41, 701-710 (doi: 10.1111/j.0021-8901.2004.00929.x)

See Also

`hotspots`

Examples

```
data(roadkills)

hc <- hotspots.comparison(dataset = roadkills, sampl.columns = 4:ncol(roadkills),
sampl.intervals = 1:5, region.column = "segment", group.column = "taxon",
include.all.together = TRUE, confidence = 0.95, min.total.events = 80,
min.hotspot.threshold = 2, comp.method = "Phi", plot = TRUE, sep.plots = FALSE,
omit.baseline.interval = TRUE, ylim = c(0, 1))
```

---

jumping.window *Jumping window*

Description

This function extracts a moving (a.k.a. running, rolling, sliding) window but with no overlap between windows and with the option for gaps between windows.

Usage

```
jumping.window(sampl.columns, window.size, gap.size, start.column = 1, J = FALSE)
```

Arguments

sampl.columns	index numbers of the consecutive columns with the sampling data (e.g. 3:180) from which to extract the jumping windows. Can also be any vector from which to extract a jumpting window.
window.size	size of each sampling window/season (consecutive sampling days each time)
gap.size	size of the gap between sampling windows. Can be zero or a positive integer.
start.column	column of sampl.columns where to actually start the sampling windows. The default is 1, but e.g. with a gap size of 1 between windows, the start column can be either 1 or 2.
J	logical, whether to provide the results in the form of J for function carcass::etterson. Defaults to FALSE.

Details

This function is now integrated within submatrix to provide for additional sampling scheme options.

Value

This function returns a vector containing the elements of sampl.columns that are included in the extracted windows; elements falling within the gaps are left out.

Author(s)

A. Marcia Barbosa, J. Tiago Marques

Examples

```
data(roadkills)

names(roadkills)

jumping.window(4:ncol(roadkills), window.size = 1, gap.size = 0)

jumping.window(4:ncol(roadkills), window.size = 1, gap.size = 1)

w3g5 <- jumping.window(4:ncol(roadkills), window.size = 3, gap.size = 5)

w3g5

sm <- submatrix(dataset = roadkills, sampl.columns = w3g5, sampl.interval = 1,
group.column = "taxon", group.names = NULL, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE)
```

`plotEventCorrs` *Plot correlations between events in each subsampling dataset and the baseline dataset*

Description

This function plots the correlation with baseline against sampling interval for each group

Usage

```
plotEventCorrs(event.corrs, sep.plots = FALSE, ...)
```

Arguments

`event.corrs`	a matrix of correlations resulting from the `sequential.corr` function
`sep.plots`	logical, whether to place each plot in a separate window
`...`	additional arguments to pass to the `plot` function

Author(s)

A. Marcia Barbosa

See Also

`plot`, `binary.comparison`, `sequential.corr`

Examples

```
data(roadkills)

submats <- sequential.submatrix(dataset = roadkills,
sampl.columns = 4:ncol(roadkills), window.sizes = 1, gap.sizes = 0:2,
group.column = "taxon", include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = 80)

hsl <- sequential.hotspots(dataset = roadkills, submats = submats,
region.column = "segment", first.subsampl.col = 4, confidence = 0.95)

hsn <- hotspot.numbers(hotspots.list = hsl, sampl.intervals = 1:3,
groups = as.character(unique(roadkills$taxon)), include.all.together = TRUE,
min.total.events = 0, min.hotspot.threshold = 2)

seqcorr <- sequential.corr(hotspots.list = hsl,
hotspots.thresholds = hsn$HS.threshold, comp.method = "Phi",
baseline.gap = 0, messages = "TRUE")

plotEventCorrs(event.corrs = seqcorr, sep.plots = FALSE, ylim = c(0, 1),
pch = 20)
```

---

repl.hs.comp *Replicate hotspot comparison*

Description

This function calculates hotspot correlation, loss, gain or balance for the different replicates per sampling scheme and taxonomic group.

Usage

```
repl.hs.comp(seqsubmats.hs, hs.baseline, method = "Phi", stats = TRUE, plot = TRUE, plot.mean = TRUE,
```

Arguments

seqsubmats.hs	hotspots for the seqsubmats
hs.baseline	hotspots for the baseline
method	binary comparison method to use. See binary.comparison for available options.
stats	logical, whether to calculate also the stats (mean, min, max, sd) of the replicate comparison for each group.
plot	logical, whether to plot the hotspot comparison values per replicate per group.
plot.mean	logical, whether to plot (with a white circle) the mean value of the replicates per group.

ylim	limits for the y axis. The default is NULL for automatic limits, but you may want to use ylim = c(0,1) for e.g. phi correlations to be directly comparable among plots.
horiz.line	optionally, a numeric value indicating the y axis value for a horizontal threshold line to be drawn.

Value

A list.

Author(s)

A. Marcia Barbosa

See Also

binary.comparison

Examples

```
## Not run:
replicate.corrs <- repl.hs.comp(seqsubmats.hs = seqsubmats.hs,
hs.baseline = hs.baseline, method = "Phi")

replicate.gains <- repl.hs.comp(seqsubmats.hs = seqsubmats.hs,
hs.baseline = hs.baseline, method = "gain")


## End(Not run)
```

---

roadkills *Imaginary roadkill data*

Description

An imaginary dataset of roadkill data for 5 "taxonomic" groups.

Usage

```
data(roadkills)
```

Format

A data frame with 900 observations on the following variables:

`individ` an integer vector attributing an identifier to each recorded individual

`segment` a numeric vector identifying the road segment at which each individual was recorded

`group` a character vector indicating the "taxonomic" group to which each individual belongs

`day1` a numeric vector indicating whether the individual was found (1) or not (0) on that sampling day (the same for all remaining days in the data frame)

Details

Each row corresponds to an individual recorded at a particular road stretch (`segment`), with a 1 if it was present and a 0 if it was not present at that segment on each of the sampling days. Individuals were not manually removed from the road, so each individual has value 1 in all days in which its body was detected on the road.

Source

Freely modified from data collected by: Santos S.M., Carvalho F., Mira A. (2011) How long do the dead survive on the road? Carcass persistence probability and implications for road-kill monitoring surveys. PLoS ONE 6(9), e25383 (doi:10.1371/journal.pone.0025383)

Examples

```
data(roadkills)

roadkills[1:20, 1:10]
```

---

schemeCorrs *Scheme correlations*

Description

Get correlation between each sampling scheme and the corresponding baseline

Usage

```
schemeCorrs(dataset, submats, submats.baseline, region.column, group.column, first.subsampl.col)
```

Arguments

`dataset`	name of the matrix or dataframe containing the complete data
`submats`	a list of the submatrices for which to calculate the correlation (result of the `sequential.submatrix` function)
`submats.baseline`	a list of the submatrices corresponding to the baseline sampling scheme for each group
`region.column`	name or index number of the column containing the regions (road segments, sites) to classify as hotspots or non-hotspots
`group.column`	name or index number of the column containing the taxonomic groups
`first.subsampl.col`	index number of the first column containing subsampling data

Value

This function returns a list of corrs.tables.

Author(s)

A. Marcia Barbosa

See Also

getBoxplots

`sequential.corr` *Sequential correlation*

Description

This function calculates the correlation between the hotspots obtained from each submatrix and those of the baseline (sub)matrix of the corresponding group.

Usage

```
sequential.corr(hotspots.list, hotspots.thresholds, comp.method = "Phi",
baseline.interval = 1, baseline.gap = 0, messages = "TRUE")
```

Arguments

`hotspots.list`	a list of hotspot tables resulting from the `sequential.hotspots` function
`hotspots.thresholds`	a matrix of hotspots thresholds (element 2 of the results of the `hotspot.numbers` function)
`comp.method`	characer value indicating the correlation coefficient to use; type `binary.comp.methods()` for available options
`baseline.interval`	the sampling interval with which to correlate all the other sampling intervals for each group; defaults to 1 (take every sample)
`baseline.gap`	the sampling gap with which to correlate all other sampling schemes for each group; defaults to 0 (no gap between samples)
`messages`	logical, whether to display messages

Value

A matrix of correlations (or whatever index was defined in `method`) between the hotspots obtained for each group and sampling scheme, and the hotspots obtained from the baseline data for the group under analysis.

Author(s)

A. Marcia Barbosa

See Also

binary.comparison

Examples

```
data(roadkills)

submats <- sequential.submatrix(dataset = roadkills,
sampl.columns = 4:ncol(roadkills), window.size = 1, gap.size = c(0, 1, 2),
group.column = "taxon", include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = 120)

hsl <- sequential.hotspots(dataset = roadkills, submats = submats,
region.column = "segment", first.subsampl.col = 4, confidence = 0.95)

hsn <- hotspot.numbers(hotspots.list = hsl, sampl.intervals = 1:3,
groups = as.character(unique(roadkills$taxon)), include.all.together = TRUE,
min.total.events = 0, min.hotspot.threshold = 2)

seqcorr <- sequential.corr(hotspots.list = hsl,
hotspots.thresholds = hsn$HS.threshold, comp.method = "Phi",
baseline.gap = 0, messages = "TRUE")

seqcorr
```

---

sequential.estimateN *Sequential estimate N*

Description

This function estimates the actual numbers of animal casualties given the observed numbers and a set of estimators, sequentially for all given submats. Requires package **carcass**.

Usage

```
sequential.estimateN(submats, submats.N, first.subsampl.col, region.column,
persist, effic, estimators = c("korner", "huso", "erickson", "etterson"),
margin = 0.01, ...)
```

Arguments

submats	result of the sequential.submatrix function.
submats.N	result of the sequential.Nevents function.
first.subsampl.col	index number of the first column containing the (sub)sampling data in submats
region.column	name or index number of the column containing the regions (road segments, sites) to classify as hotspots or non-hotspots

persist	named vector of persistence per group; group names must match those in the data
effic	named vector of detection efficiency per group; group names must match those in the data
estimators	character vector of the estimator(s) to use. The default is all estimators available.
margin	numeric value to subtract from and add to the estimator(s) to get p.lower and p.upper, respectively, when using function estimateN in package **carcass**
...	currently not in use

Value

This function returns a list.

Author(s)

A. Marcia Barbosa, J. Tiago Marques, Sara Santos

See Also

function estimateN in package **carcass**

---

sequential.hotspots *Calculate roadkill hotspots for a series of (sub)sampling datasets*

Description

This function applies hotspots sequencially to a given set of submatrices to identify the hotspot regions in each dataset, using an adaptation of the method of Malo et al. (2004).

Usage

```
sequential.hotspots(dataset, submats, region.column, first.subsampl.col, confidence = 0.95)
```

Arguments

dataset	name of the matrix or dataframe containing the complete data
submats	a list of the submatrices for which to calculate the hotspots (result of the sequential.submatrix function)
region.column	name or index number of the column containing the regions (road segments, sites) to classify as hotspots or non-hotspots
first.subsampl.col	index number of the first column containing subsampling data
confidence	confidence threshold to consider hotspots. The default is 0.95

Value

A list of 2 elements:

`hotspots.thresholds`
: A named integer vector

`hotspots.maps`
: A list of data frames, each showing the total number of events (deaths) per region and whether or not it was considered a hospot.

Author(s)

A. Marcia Barbosa

References

Malo, J.E., Suarez, F., Diez, A. (2004) Can we mitigate animal-vehicle accidents using predictive models? J. Appl. Ecol. 41, 701-710 (doi: 10.1111/j.0021-8901.2004.00929.x)

See Also

`hotspots`

Examples

```
data(roadkills)

submats <- sequential.submatrix(dataset = roadkills,
sampl.columns = 4:ncol(roadkills), window.sizes = 1, gap.sizes = 1:3,
group.column = "taxon", include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = 85)

shs <- sequential.hotspots(dataset = roadkills, submats = submats,
region.column = "segment", first.subsampl.col = 4, confidence = 0.95)

shs
str(shs)
```

---

sequential.Nevents *Sequential numbers of events*

Description

Applies function link{hotspots} (with hotspots=FALSE) sequentially to a given set of submatrices

Usage

```
sequential.Nevents(dataset, submats, region.column, first.subsampl.col, estimate = FALSE)
```

Arguments

dataset	name of the matrix or dataframe containing the complete data
submats	a list of the submatrices for which to calculate the hotspots (result of the sequential.submatrix function)
region.column	name or index number of the column containing the regions (road sectors, sites) to classify as hotspots or non-hotspots
first.subsampl.col	index number of the first column containing subsampling data
estimate	logical, whether to add estimates from package **carcass**

Author(s)

A. Marcia Barbosa

See Also

link{hotspots}, link{sequential.hotspots}

---

sequential.seqsubmat *Sequential sequential.submatrix*

Description

Applies sequential.submatrix sequentially to a set of gap sizes, with window.size =1

Usage

```
sequential.seqsubmat(dataset, sampl.columns, group.column, gap.sizes, n.replicates.limit)
```

Arguments

dataset	name of the matrix or dataframe to analyze
sampl.columns	numbers of the consecutive columns with the (daily) sampling data, e.g. 4:180
group.column	name or index number of the column containing the taxa or groups to analyse separately, e.g. 3 or "Family"; if NULL, all records will be used together
gap.sizes	integer vector of the size(s) of the gaps between sampling days. Must be either 0 (for no gap) or a vector of positive integers.
n.replicates.limit	Maximum number of replicates allowed

Value

This function returns a list of submatrices

Author(s)

A. Marcia Barbosa

See Also

sequential.submatrix

Examples

```
data(roadkills)

seqsubmats <- sequential.seqsubmat(dataset = roadkills, sampl.columns = 4:ncol(roadkills), group.column = "taxon

# seqsubmats for gap 0 (baseline scenario) must be obtained separately:
seqsubmats.baseline <- sequential.seqsubmat(dataset = roadkills, sampl.columns = 4:ncol(roadkills), group.column
```

---

sequential.submatrix *Extract subsampling matrices for a series of subsampling schemes*

Description

This function applies submatrix sequencially to all specified sampling schemes and taxonomic groups.

Usage

```
sequential.submatrix(dataset, sampl.columns, sampl.intervals = NULL,
window.sizes = NULL, gap.sizes = NULL, start.columns = 1, all.combinations = TRUE,
group.column = NULL, include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = NULL)
```

Arguments

dataset	name of the matrix or dataframe to analyze
sampl.columns	numbers of the consecutive columns with the (daily) sampling data, e.g. 4:180
sampl.intervals	a vector of the intervals at which to extract sampling data, e.g. 5 to take one every five samples
window.sizes	the size (in sampling time units, e.g. days) of the sampling periods
gap.sizes	the size of the gaps between sampling periods
start.columns	vector of start columns, see submatrix
all.combinations	logical, whether to use all window x gap size combinations (the default, TRUE) or just the number corresponding to the length of window.sizes and gap sizes (in which case window.sizes and gap.sizes must have the same length)

`group.column` name or index number of the column containing the taxa or groups to analyse separately, e.g. 3 or "Family"; if NULL, all records will be used together

`include.all.together` logical, whether to get subsampling matrices also for the complete data (including all groups together)

`remove.zeros` logical, whether to remove rows where all extracted samples have zero observations

`keep.nonsampl.columns` logical, whether to include also the non-sampling columns of `dataset` in the resulting submatrices)

`n.subsampl.columns` number of subsampl.columns

Value

A list of submatrices

Author(s)

A. Marcia Barbosa

See Also

`submatrix`, `subset`

Examples

```
data(roadkills)

submats1 <- sequential.submatrix(dataset = roadkills, sampl.columns = 4:ncol(roadkills),
sampl.intervals = c(1, 3), group.column = "taxon", include.all.together = TRUE,
remove.zeros = TRUE, keep.nonsampl.columns = TRUE, n.subsampl.columns = 85)

names(submats1)
head(submats1[[1]])

submats2 <- sequential.submatrix(dataset = roadkills, sampl.columns = 4:ncol(roadkills),
window.sizes = c(1,3,5), gap.sizes = 1:3, start.columns = 1, all.combinations = TRUE,
group.column = "taxon", include.all.together = TRUE, remove.zeros = TRUE,
keep.nonsampl.columns = TRUE, n.subsampl.columns = 85)

names(submats2)
head(submats2[[1]])
```

submatrix *Extract a submatrix for a given taxomic group and/or sampling scheme*

Description

Given a baseline dataset, this function extracts a sub-dataset for a given taxomic group and/or sampling scheme, defined either by a sampling interval (periodicity) or by a window size (consecutive sampling days each time) and a gap size (gaps between sampling windows).

Usage

```
submatrix(dataset, sampl.columns, sampl.interval = NULL, window.size = NULL,
gap.size = NULL, start.column = 1, group.column = NULL, group.names = NULL,
remove.zeros = TRUE, keep.nonsampl.columns = TRUE)
```

Arguments

dataset	name of the matrix or dataframe to analyze
sampl.columns	index numbers of the (consecutive) columns containing the baseline (daily) sampling data, e.g. 3:180
sampl.interval	interval at which to extract sampling data, e.g. 5 (to take one every five samples)
window.size	instead of sampl.interval (for one sampling every so many time units), size (in sampling time units, e.g. days) of each sampling window (e.g. 3 for 3 consecutive days sampling each time); must be complemented with gap.size, for the gap between sampling windows.
gap.size	size (in sampling time units, e.g. days) of the gaps between sampling windows – e.g. 1 for 1 gap (non-sampled) day between sampling windows.
start.column	column of sampl.columns where to actually start the sampling (e.g. with a gap size of 1 between sampling windows, the start column can be either 1 or 2)
group.column	name or index number of the column containing the taxa or groups to analyse, e.g. 3 or "Family"
group.names	name(s) of the group(a) to extract, e.g. c("Mustelidae", "Procyonidae"); if NULL (the default), all groups in group.names are extracted
remove.zeros	logical indicating whether to remove rows where all extracted days have zero observations
keep.nonsampl.columns	logical indicating whether to keep the non-sampling columns in the extracted result

Value

This function returns a subset of dataset containing the taxonomic groups and sampling columns resulting from the given sampling scheme.

Author(s)

A. Marcia Barbosa, J. Tiago Marques

See Also

sequential.submatrix, subset

Examples

```
data(roadkills)

submat1 <- submatrix(dataset = roadkills, sampl.columns = 4:ncol(roadkills),
sampl.interval = 3, start.column = 1, group.column = "taxon", group.names = NULL)

head(submat1)

submat2 <- submatrix(dataset = roadkills, sampl.columns = 4:ncol(roadkills),
window.size = 5, gap.size = 2, start.column = 1, group.column = "taxon",
group.names = NULL)

head(submat2)
```

Index