Package 'seasonality'

October 21, 2024

Title Define biological seasons

Version 1.0.1

Date 2024-10-21

Depends cluster,
R (>= 2.10)

Description This package provides a set of functions to split year-round space-use measurements into biological seasons, completed with additional functions to explore and simplify these seasons. Reference: Basille M., Fortin D., Dussault C., Ouellet J.-P., Courtois R. (2013) Ecologically based definition of seasons clarifies predator-prey interactions. Ecography, 36:220–229. DOI: 10.1111/j.1600-0587.2011.07367.x

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/basille/seasonality

BugReports https://github.com/basille/seasonality/issues

RoxygenNote 7.3.2

Contents

bsSeasons *Season bootstrap*

Description

Bootstrap procedures to remove the less important seasons.

Usage

```
bsSeasons(data, ind, nclust, iter = 100, simplify = FALSE, win = 3, tol = 1)

bsWeights(bsSeasons)

bsCriterion(seasons, bsWeights, threshold = 0.75)

bsPlot(bsSeasons, seasons = NULL, bsWeights, title)
```

Arguments

data	A data frame indicating the initial data on which to run the bootstrap.
ind	A individual-year table indicating the name of the individual (column id), repeated as many times as monitoring periods.
nclust	The number of clusters to apply to the k-means.
iter	The number of iterations of the bootstrap.
simplify	Logical. Whether to simplify the resulting seasons.
win	If simplify, the length of the moving window.
tol	If simplify, the tolerance to be used.
bsSeasons	Bootstrap seasons.
seasons	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.
bsWeights	The bootsrap weights, as given by bsWeights.
threshold	The weight threshold.
title	A title for the strip.

Details

The function bsSeasons samples individual animal-years with replacement, and run the K-means clustering with the estimated number of clusters for the whole data set.

The weight is then given by the function bsWeights, which gives, for each day of the year (from 1:365) the number of changes in the last and next two days. This weight is then used by the function bsCriterion to retain only seasons which are within a given threshold of weight (based on the bootstrap data set).

bsPlot plots the result of the bootstrap procedure.

Value

A list of length `iter`, each element of which giving the clustering of one bootstrap iteration.

`bsSeasons` returns a list of vectors of the same length as `seasons`, giving the seasons for each bootstrap loop.

`bsWeights` returns a vector of the same length as `seasons`, with the weight of each day.

`bsCriterion` returns a vector of the same length as `seasons`, with the index of the clusters kept.

Examples

```
### Compute the bootstrap seasons:
caribou$bs <- bsSeasons(data = caribou$move, ind = caribou$ind,
    nclust = 8)

### Compute the weights, and identify the final seasons:
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
weights <- bsWeights(caribou$bs)
seasonsbs <- bsCriterion(seasons, weights, threshold = .8)

### Visualize the final seasons:
bsPlot(seasonsbs, seasons, weights, title = "Caribou")
```

---

caribou *Caribou data set used in Basille et al. (2012).*

Description

Caribou data set used in Basille et al. (2012).

Usage

```
caribou
```

Format

'caribou' A list with 4 elements:

ind A 91×2 data frame, providing the code of the individual ('ind') and the year ('year').

move A list with 9 data frames of environmental variables, each with 365 rows (julian day) and 91 columns (individual-year), range-standardised by individual-year.

window A 365×9 data frame, providing the value of environmental variables over a 15-day moving window for all caribou combined.

bs A list of 100 repetitions, providing season numbers as integers from 0 to i for each day of the year.

Source

Basille M., Fortin D., Dussault C., Ouellet J.-P., Courtois R. (2013) Ecologically based definition of seasons clarifies predator-prey interactions. Ecography, 36:220–229. DOI: 10.1111/j.1600-0587.2011.07367.x

Examples

```
## We work with the caribou dataset.
## `caribou$window` can be retrieved from `caribou$move`:
##
## We first compute the weights for each individual-year:
caribou$ind$weights <- rep(1/table(caribou$ind$id),
    times = table(caribou$ind$id))
## We then compute the weighted mean for each environmental variable:
cari_win <- data.frame(lapply(caribou$move, apply, 1, weighted.mean,
    w = caribou$ind$weights, na.rm = TRUE))
## And finally range-standardise the resulting data:
cari_win <- data.frame(scale(cari_win,
    center = apply(cari_win, 2, min, na.rm = TRUE),
    scale = apply(cari_win, 2, \(x) diff(range(x, na.rm = TRUE)))),
    row.names = as.character(1:365))
head(cari_win)
summary(cari_win)
all.equal(cari_win, caribou$window)

## We then compute the gap statistic for 1-10 clusters:
(caribou$gap <- gap(caribou$window)) # Different results from Basille et al.
plot(caribou$gap)
## And compute seasons with the K-means for 8 clusters:
set.seed(1)
caribou$seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sPrint(caribou$seasons)

## Bootstrap approach:
caribou$bs <- bsSeasons(data = caribou$move, ind = caribou$ind,
    nclust = 8)
## Compute the bootstrap weights, and identify 'true' seasons:
caribou$bsweights <- bsWeights(caribou$bs)
caribou$seasonsbs <- bsCriterion(caribou$seasons, caribou$bsweights,
    threshold = .9)
sPrint(caribou$seasonsbs)
## Check visually:
bsPlot(caribou$seasonsbs, caribou$seasons, caribou$bsweights,
    title = "Bootstrap on Caribou seasons")

## Simplify to get the final seasons:
sPrint(sFormat(sSimple(caribou$seasonsbs)))
## We end up with the following seasons:
## - December 28–January 2 (winter 1)
## - January 3–April 15 (winter 2)
## - April 16–May 21 (spring dispersal)
## - May 22–May 27 (pre-calving)
```

```
## - May 28-September 17 (calving and summer)
## - September 18-December 27 (autumn)
##
## We can visualize the characteristics of the final seasons:
sBoxplot(caribou$window, sFormat(sSimple(caribou$seasonsbs)))
```

---

gap *Gap statistic*

Description

Compute the gap statistic (weighted by default).

Usage

```
gap(
  data,
  from = 1,
  to = 10,
  nsim = 50,
  ref.dist = c("pc", "unif"),
  clust.method = "k-means",
  dist.method = "euclidean",
  weighted = TRUE,
  tol = 1,
  seed = 1
)

## S3 method for class 'gap'
plot(x, ...)
```

Arguments

data	A matrix, or a data frame coercible to a matrix. Input data should be of the form obs $\times$ var.
from	The minimal number of clusters for which the gap statistic is computed.
to	The maximal number of clusters for which the gap statistic is computed.
nsim	The number of simulations used to compute the gap statistic.
ref.dist	A character string specifying the reference distribution: unif Generates each reference variable uniformly over the range of the observed values for that variable; pc Generates the reference variables from a uniform distribution over a box aligned with the principal components of the data.
clust.method	A character string specifying the cluster analysis method to be used. This should be one of: "ward", "single", "complete", "average", "mcquitty", "median", "centroid", "pam", "k-means", "diana". Only tested for "k-means", which is the default.

dist.method	The distance measure to be used. Only tested for "euclidean". See dist for other metrics.
weighted	Logical. Whether the gap statistic should be weighted or not (default is TRUE).
tol	An number specifying the multiplier to reject the null model. The tolerance is analogous to setting the alpha level in the standard hypothesis testing framework, where increased tolerance is similar to selecting a smaller alpha rejection region. Tibshirani et al. (2001) used a tolerance of 1 (default behaviour), but larger values of tolerance increase the strength of evidence required to include additional clusters;
seed	A single value, interpreted as an integer, used a seed in the clustering method.
x	An object of class gap.
...	Further arguments passed to or from other methods.

Details

The package clusterSim proposes a index.Gap function to compute the gap statistic. It can be used with many different clustering methods ("ward", "single", "complete", "average", "mcquitty", "median", "centroid", "pam", "k-means", "diana"), and with uniform or pc-based reference distributions.

Bram Van Moorter modified it into index.gap.modif (http://ase-research.org/moorter/p7_gap.statistic.r), which uses k-means as a default, returns values when only one large cluster is made, and instead of calculating gap-differences, it now returns the original gap-value.

It seems however that the algorithm to compute W_k in index.Gap is not correct; in addition the index.Gap function is quite poorly written and thus difficult to understand; last but not least, it does not allow to compute the weighted gap statistic. The weighted gap statistic have been shown to provide more robust and consistent results, and allows in a multi-layer approach to derive nested clusters.

Value

gap returns a k x p data frame of class gap with the following variables:

nCluster The number of clusters k;

logWk0 $\log(W_k)$ (from the data) where $W_k = \sum_{m=1}^{k} \frac{1}{2n_m} D_m$, or $W_k = \sum_{m=1}^{k} \frac{1}{2n_m(n_m-1)} D_m$ if weighted, D_m being the (complete) sum of pairwise distances;

logWk $E_n^* \log(W_k)$ (from the simulated data sets);

Gap The gap statistic as $\mathrm{Gap}_n(k) = E_n^*\{\log(W_k)\} - \log(W_k) = (1/B)\sum_b^B = 1 \log(W_{kb}^*) - \log(W_k)$, B being the number of simulated data sets;

sdGap The standard deviation of the gap statistic, as $[s_k = (1/B)\{\sum_b^B = 1 \log(W_{kb}^*) - (1/B)\sum_b^B = 1 \log(W_{kb}^*)\}^2]^{1/2}\sqrt{(1+1/B)}$;

k The estimated number of clusters with the classical approach, indicated by an asterisk, with a tolerance T, such as $\mathrm{Gap}(k) \geq \mathrm{Gap}(k+1) - T * s_{k+1}$;

D Differences of gap, as $D\mathrm{Gap}_n(k) = \mathrm{Gap}_n(k) - \mathrm{Gap}_n(k-1)$;

DD differences of Dgap, as $DD\mathrm{Gap}_n(k) = D\mathrm{Gap}_n(k) - D\mathrm{Gap}_n(k+1)$;

DDk The estimated number of clusters with the DD-weighted approach, indicated by an asterisk; the number of clusters k is given when DDGap is maximum.

References

Tibshirani, R.; Walther, G. & Hastie, T. (2001) Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), Blackwell Publishers Ltd., 63: 411-423, DOI: 10.1111/1467-9868.00293

Yan, M. & Ye, K. (2007) Determining the number of clusters using the weighted gap statistic. Biometrics, 63: 1031-1037, DOI: 10.1111/j.1541-0420.2007.00784.x

Basille, M.; Fortin, D.; Dussault, C.; Ouellet, J.-P. & Courtois, R. (2013) Ecologically based definition of seasons clarifies predator-prey interactions. Ecography, 36:220–229, DOI: 10.1111/j.1600-0587.2011.07367.x

Examples

```
### Simple simulation
set.seed(1)
X <- matrix(rnorm(30, mean = 5), ncol = 3)
set.seed(1)
Y <- rbind(matrix(rnorm(300, mean = 5), ncol = 3),
    matrix(rnorm(300, mean = 10), ncol = 3))

### K-means tests
## Beware of the case of only 1 group:
(GG1 <- gap(X, to = 9, ref.dist = "unif"))
plot(GG1)
## Two groups:
(GG2 <- gap(Y))
plot(GG2)

### Caribou data
carigap <- gap(caribou$window)
plot(carigap)
```

---

sBoxplot *Season boxplots*

Description

Season boxplots.

Usage

```
sBoxplot(
  data,
  seasons,
  temporal = TRUE,
  months = c("rectangles", "lines"),
  cluster = TRUE,
  multi = FALSE,
  samescale = TRUE
)
```

Arguments

data	The original data on which the clustering was made (see gap).
seasons	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.
temporal	Logical. If TRUE, produces boxplots along the year (X-axis); if FALSE, produces boxplots for each cluster using their index.
months	Draws the months with background rectangles (rectangle) or dotted lines (lines).
cluster	Logical. Indicates the cluster index above the graph.
multi	Logical. Allows for comparison between several clusterings, by displaying them side by side. If yes, requires a list of data and seasons, corresponding to each clustering.
samescale	Logical. In case of comparison, use the same scale for common variables.

Examples

```
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sBoxplot(caribou$window, sSimple(seasons))
```

---

sFormat *Reorder the seasons.*

Description

Reorder the seasons as a succession of unique numbers, from 1 to the last season (useful in case of duplicated clusters, as duplicates get a new index).

Usage

```
sFormat(seasons)
```

Arguments

seasons	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.

Value

A vector of the same length as seasons, with the index of the clusters reordered.

Examples

```
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sPrint(seasons)
sPrint(sFormat(seasons))
```

sPlot *Plot the seasons*

Description

Plot the seasons.

Usage

```
sPlot(
  seasons,
  add.lines = FALSE,
  months = FALSE,
  main = "Seasons",
  ylab = substitute(seasons),
  ...
)
```

Arguments

seasons	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.
add.lines	Logical. Adds dotted lines delineating the seasons.
months	Logical. Draws monthtly delineations.
main	An overall title for the plot.
ylab	A title for the y axis.
...	Further arguments passed to the lines call.

Examples

```
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sPlot(sSimple(seasons))
```

sPrint *Print seasons*

Description

Print a sequence of seasons in a friendly way.

Usage

```
sPrint(seasons, ndays = FALSE)
```

Arguments

seasons	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.
ndays	Logical. Returns the rank of the days at which a new season starts.

Value

A vector indicating the dates at which a new season starts.

Examples

```
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sPrint(seasons)
```

---

sSimple *Simplify the seasons.*

Description

Simplify the seasons after the initial clustering, by removing the smallest seasons.

Usage

```
sSimple(clust, win = 3, tol = 1)
```

Arguments

clust	The result of a season clustering: A vector of integers (from 1:k) indicating the cluster to which each day is allocated.
win	The length of the moving window.
tol	The tolerance to be used.

Details

The function works on a moving window of length (current day + win) days. For a given day, if all other days (with a tolerance of tol days) have the same value as the focus day, the day is kept as is; otherwise, the day takes the value of the day before.

Value

A vector of the same length as seasons.

Examples

```
set.seed(1)
seasons <- kmeans(caribou$window, 8, iter.max = 100)$cluster
sPrint(seasons)
sPrint(sSimple(seasons))
```

Index