Package 'adehabitat'

July 22, 2015

Version 1.8.18

Date 2015-07-22

Title Analysis of Habitat Selection by Animals

Author Clement Calenge, contributions from Mathieu Basille, Stephane Dray and Scott Fortmann-Roe

Maintainer Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

Depends R (>= 2.10.0), ade4, tkrplot, shapefiles, sp, graphics

Suggests spatstat, MASS, CircStats

Imports grDevices, methods, stats, utils

Description A collection of tools for the analysis of habitat selection by animals.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-07-22 16:15:32

R topics documented:

`adehabitat-package` *adehabitat: a Package for the Analysis of the Space Use by Animals*

Description

The package adehabitat has been designed to allow the analysis of the space use by animals. This package is organised in four main parts: (i) management of raster maps, (ii) habitat selection/ecological niche analysis, (iii) home range estimation and (iv) analysis of animals trajectories. These four parts interact with each other to facilitate the analysis. Note that adehabitat strongly relies on the package ade4, which provides numerous functions for the analysis of multivariate data.

Details

Package:	adehabitat
Type:	Package
Version:	1.8.3
Date:	2009-05-04
License:	GPL version 2 or newer

The four parts of the packages are described more in detail below:

- Management of raster mapsHowever, it provides numerous functions allowing to import and export raster maps from/to Geographic information system, to compute buffers around points or lines, to identify the value of environmental variables at given spatial locations, to count the number of points of a pattern in each pixel of a map, etc. Two basic object classes allow to manage ratser maps with adehabitat: the class `asc` is intended to store basic raster map (see `help(import.asc)`), whereas the class `kasc` is intended to store multi-layer maps (all covering the same area with the same resolution, see `help(as.kasc)`). For additional information this part of the package, see the tutorial available in the package. Type `demo(rastermaps)` for a demonstration of the package capabilities. Note that the package sp also provides many interesting functions to manage raster maps, and adehabitat provide functions of conversion to the classes of the package (see `help(kasc2spixdf)`).

- Habitat selection/ecological niche analysismany methods have been included in the package to render statistical methods allowing the analysis of habitat selection available to wildlife ecologists. Many of them are factor analyses of the niche or of the habitat selection, but other methods are also available. They include the selection ratios (see `help(wi)`), the Ecological niche factor analysis (see `help(enfa)`), the Mahalanobis distances (see `help(mahasuhab)`) and their factorial decomposition (the MADIFA, see `help(madifa)`) or the algorithm DOMAIN (see `help(domain)`). Other common methods, such as the resource selection functions can also be used with the rest of the R environment. Note that the package also include functions allowing the analysis of habitat selection using radio-tracking data, such as the compositional analysis (see `help(compana)`), the eigenanalysis of selection ratios (see `help(eisera)`) or the K-select analysis (see `help(kselect)`). An overview of these methods is available by typing `demo(nichehs)`.

- home range estimationmany methods are available to estimate the home range of animals relocated using radio-tracking data. Common methods of estimation are available, such as the Minimum convex polygon (see `help(mcp)`), the kernel estimation of the utilization distribution (see `help(kernelUD)`), the cluster home range (see `help(clusthr)`) or the nearest neighbour convex hull (see `help(NNCH)`). Note that Paolo Cavallini has designed a website dedicated to the analysis of space use by animals, which contain a wiki page, a tutorial for the home range estimation using R and adehabitat and a forum (URL: http://www.faunalia.it/animov/index.php). Several methods of these part of the package have been included following discussions that arose on this forum (especially, the nearest neighbour convex hull and the brownian bridge kernel). Type `demo(homerange)` for examples of use of these functions.

- The analysis of animals' trajectoriesThis part is the most recent one in the package. A new class designed to store animals' trajectories has been included in the package, the class `ltraj` (see `help(as.ltraj)`). Two types of trajectories can be handled with adehabitat: for trajectories of type I, the time is not recorded for the relocations (e.g. the sampling of the tracks of an animal in the snow). For trajectories of type II, the time has been recorded during sampling (e.g. radio-tracking, GPS, Argos monitoring). Many descriptive parameters are automatically computed (relative or turning angles, distance between successive relocations, mean squared displacement). Many functions allow the management and analysis of trajectories, through the analysis of these parameters (e.g. tests of independence, see `help(wawotest,indmove)`, first passage time, see `fpt`). The rediscretisation of trajectories of type I is also possible (`help(redisltraj)`). Many graphical functions are available for the exploration of the trajectory properties (`plot,plotltr,sliwinltr`), etc. A new partitioning algorithm has been added (but it is still under research) to partition animals trajectories into segment with homogeneous properties (see `modpartltraj`). Further details can be found on the help page of the function `as.ltraj`. For a demonstration, type `demo(managltraj)` or `demo(analysisltraj)`.

Author(s)

Clement Calenge, with contributions from Mathieu Basille, Stephane Dray, Manuela Royer and Scott Fortmann-Roe

Maintainer: Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Calenge, C. (2006) The package adehabitat for the R software: a tool for the analysis of space and habitat use by animals. Ecological Modelling, 197, 516-519

See Also

`ade4`

Examples

```
## Not run:

## For examples of use of mapping capabilities
demo(rastermaps)

## For examples of use of functions for
## habitat selection and niche analysis
demo(nichehs)

## For example of home range estimation
demo(homerange)

## For example of trajectory management and analysis
demo(managltraj)
demo(analysisltraj)



## End(Not run)
```

`albatross` *Argos Monitoring of Adult Albatross Movement*

Description

This data set contains the relocations of 6 adult albatross monitored in the Crozets Islands by the team of H. Weimerskirch from the CEBC-CNRS (Centre d'Etudes Biologiques de Chize, France).

Usage

```
data(albatross)
```

Format

This data set is an object of class `ltraj`.

Details

The coordinates are given in meters (UTM - zone 42).

Source

http://suivi-animal.u-strasbg.fr/index.htm

Examples

```
data(albatross)

plot(albatross)
```

`angles` *Compute Turning Angles - Deprecated*

Description

`angles` computes the turning angles (in radians) between consecutive moves from an object of class `traj`. See examples for a clearer definition.

Usage

```
angles(x, id = levels(x$id), burst = levels(x$burst),
       date = NULL, slsp =  c("remove", "missing"))
```

Arguments

`x`	an object of class `traj`
`id`	a character vector giving the identity of the animals for which the angles are to be computed
`burst`	a character vector giving the identity of the circuits for which the angles are to be computed (see `traj`)
`date`	a vector of class `POSIXct` of length 2 (beginning, end) delimiting the period of interest
`slsp`	a character string. If `"remove"`, successive relocations located at the same place are replaced by a single relocation, allowing the computation of the angles. If `"missing"`, a missing value is returned for the angles when successive relocations located at the same place.

Value

Returns a data frame with the following components:

`id`	the identity of the animal
`x`	the x coordinate of the relocation at which the angle is computed
`y`	the y coordinate of the relocation at which the angle is computed
`date`	a vector of class `POSIXct`, giving the date at which the relocation has been taken
`burst`	the id of the circuit (see `help(traj)`)
`angles`	the turning angles between the successive moves.

Note

The function `angles` is deprecated. The class `ltraj` computes the turning angles automatically (see `ltraj`).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Turchin, P. (1998) *Quantitative analysis of movement. Measuring and modeling population redistribution in animals and plants.* Sunderland, Massachusetts: Sinauer Associates.

See Also

`speed` for computation of movement speeds, `traj` for additional information about objects of class `traj`

Examples

```
## Not run:
## loads an object of class "traj"
data(puechcirc)
puechcir <- ltraj2traj(puechcirc)
puechcir

## Gets a part of the trajectory of the wild boar named
## CH93 and draws it
## Also displays the turning angles
toto <- puechcir[2:5,]
plot(toto$x, toto$y, asp = 1, ylim = c(3158300, 3158550),
     pch = 16,
     main = "Turning angles between\nthree consecutive moves",
     xlab="X", ylab="Y")
lines(toto$x, toto$y)
lines(c(toto$x[2], 700217.6),
      c(toto$y[2], 3158310), lty=2)
```

```
lines(c(toto$x[3],700289),
      c(toto$y[3],3158546), lty=2)
ang1x <- c(700234.8, 700231.9, 700231, 700233.7, 700238.8, 700243.2)
ang1y <- c(3158332, 3158336, 3158341, 3158347, 3158350, 3158350)
ang2x <- c(700283.3, 700278.8, 700275.4, 700272.4, 700271.2, 700271.6,
           700274.7)
ang2y <- c(3158522, 3158522, 3158520, 3158517, 3158514, 3158508, 3158504)
lines(ang1x, ang1y)
lines(ang2x, ang2y)
text(700216.1, 3158349, expression(theta[1]), cex=2)
text(700247.7, 3158531, expression(theta[2]), cex=2)
text(c(700301, 700231), c(3158399, 3158487),
     c("Beginning", "End"), pos=4)



## Computation of the turning angles with real data
## on wild boars
plot(puechcir)
ang <- angles(puechcir)

## The angles are in the column angles:
ang[1:4,]

## End(Not run)
```

area2asc *Converts a Polygon to Raster*

Description

`area2asc` converts a polygon map to a raster map of class `asc`. It is an alias for `mcp.rast`.

Usage

```
area2asc(poly, w, border=c("include", "exclude"))
```

Arguments

`poly`	a data frame with 2 columns giving the coordinates of a polygon object
`w`	an object of class `kasc`, or of class `asc`
`border`	a character string indicating what happens when the center of the pixel is located exactly on the limit of the polygon (`"include"` indicates that the pixel is considered to be inside the polygon).

Details

The raster map is needed to pass the format for the output raster object to the function.

Value

Returns an object of class asc.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

hr.rast

Examples

```
data(puechabon)
toto <- puechabon$kasc
loc <- puechabon$locs

## gets the coordinates of the relocations for the wild boar #1
wb1 <- loc[loc$Name == "Chou",]
wb1 <- cbind(wb1$X, wb1$Y)
nbpol <- chull(wb1)
xycoord <- wb1[nbpol,]

## rasterization of wb1
tutu <- area2asc(xycoord, toto)
image(tutu)

polygon(xycoord, lwd = 2)
```

---

area2dxf *Exportation of Areas*

Description

area2dxf exports a data frame of class area in a DXF file. Such files can be read by virtually all Geographic Information Systems.

Usage

```
area2dxf(x, file, lay = 1:nlevels(factor(x[, 1])))
```

Arguments

x	an object of class area
file	a character string giving the name of the DXF file to be created
lay	an optional vector with a length equal to the number of polygons in x. This vector is then stored in the field "Layer" of the DXF file (see examples), for GIS mapping

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

area, area.plot

Examples

```
## Not run:
  ## Loads the dataset elec88 from the package ade4
  data(elec88)
  ar <- as.area(elec88$area)
  area.plot(ar, values=elec88$tab[,1])

  ## exportation of the election results toward a dxf file
  area2dxf(ar, file = "Dept", lay = elec88$tab[,1])

  ## Removes the file
  file.remove("Dept.dxf")

## End(Not run)
```

as.area   *Objects of Class "area"*

Description

Objects of class area are used to store the information on the shape of vectorised objects.
An object of class area is a data frame with three variables. The first variable is a factor defining the polygons.
The second and third variables are the xy coordinates of the polygon vertices in the order where they are found. This kind of objects are current in the package ade4, though this package does not define area as a special class.

Usage

```
as.area(x)
```

Arguments

x	a data frame with three columns

Value

Returns an object of class `area`

See Also

`area.plot` (package ade4) for other operations on objects of class `area`, `plot.area` to display objects of this class, and `area2dxf` for exportation of the objects `area` toward a GIS.

Examples

```
data(elec88)
ar <- as.area(elec88$area)
plot(ar)
```

`as.kasc` *Working with Several Raster Maps*

Description

`as.kasc` converts a list of matrices of class `asc` into a data frame of class `kasc`.
`getkasc` converts one of the components of a data frame of class `kasc` into a matrix of class `asc`.
`image.kasc` displays a image of maps of class `kasc`.

Usage

```
as.kasc(l)
getkasc(x, var)
## S3 method for class 'kasc'
image(x,  var = names(x),
           mar = if (length(var) > 1) c(0,0,2,0) else c(5.1,4.1,4.1,2.1),
           axes = (length(var) == 1),
           clfac = NULL, col = gray((240:1)/256), mfrow = NULL, ...)
## S3 method for class 'kasc'
print(x, ...)
```

Arguments

`l`	a list of objects of class `asc`
`x`	an object of class `kasc`
`var`	for `getkasc`, a character string or a column number. For `image.kasc`, either a vector of character strings, giving the name of one (or several) variable(s) in x, or a vector of integers, giving the number of the columns to be displayed.
`mar`	this parameter is passed as the parameter `mar` of the function `par` (see `help(par)`). By default, this parameter depends of the number of variables in `x`
`axes`	logical. If `TRUE`, axes are drawn on the maps.

`clfac`	an optional list of vectors of character strings. Each element of the list is a vector corresponding to one factor in `x`. Each element of the vector gives the color name associated with each level of the corresponding factor (see Examples).
`col`	a character vector. For numeric maps, the colors to be used
`mfrow`	A vector of the form `c(nr, nc)`. Subsequent figures will be drawn in an `nr`-by-`nc` array on the device by rows
`...`	additional parameters to be passed to the generic functions `image` and `print`

Value

`as.kasc` returns an object of class `kasc`. The matrices of class `asc` are converted into vectors and coerced into a data frame of class `kasc`. This data frame has the following attributes:

`xll`	the x coordinate of the center of the lower left pixel of the map
`yll`	the y coordinate of the center of the lower left pixel of the map
`cellsize`	the size of a pixel on the studied map
`nrow`	the number of rows of the map. Caution ! the number of rows of the raster map is the number of columns of the matrix of class `asc`
`ncol`	the number of columns of the map. Caution ! the number of columns of the raster map is the number of rows of the matrix of class `asc`

The different maps in the input list of objects of class `asc` should describe the same area (same attributes: same `cellsize`, same `xll`, same `yll`, and same dimensions).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc` for additionnal information on objects of class `asc`. `image.kasc` and `explore.kasc` for graphical displays of objects of class `kasc`.

Examples

```
data(puechabon)

## kasc is a list of class kasc
(kasc <- puechabon$kasc)


## Stores two elements of the kasc in objects
## of class asc
(asp <- getkasc(kasc, "Aspect"))
(sl <- getkasc(kasc, "Slope"))


## Displays the slopes on the area (numeric)
```

```
image(sl, main = "Slope", xlab = "Lambert X", ylab = "Lambert Y")


## Aspect is a factor:
## cl is the list of color associated with the levels
## of asp
cl <- colasc(asp, NorthEast = "blue", SouthEast = "red",
             SouthWest = "orange", NorthWest = "green")
image(asp, clfac = cl, main = "Aspect", xlab = "Lambert X",
      ylab = "Lambert Y")
legend(706500, 3162000, legend=levels(asp), fill=cl, cex=0.7)



## Creation of a new kasc with elevation
## and slopes
cuicui <- as.kasc(list(Slope = sl, Aspect = asp))

## Displays the kasc object
## with random colors for aspect and grey levels for slopes
image(cuicui)

## with cm.colors for slopes (numeric)
## and cl for aspect (factor)
image(cuicui, col = cm.colors(256), clfac = list(Aspect = cl))


## plots only slope
image(cuicui, var = "Slope", main="Slope")
## similar to
image(cuicui, var = 1, main = "Slope")
```

as.ltraj *Working with Trajectories in 2D Space: the Class ltraj*

Description

The class `ltraj` is intended to store trajectories of animals. Trajectories of type II correspond to trajectories for which the time is available for each relocation (mainly GPS and radio-tracking). Trajectories of type I correspond to trajectories for which the time has not been recorded (e.g. sampling of tracks in the snow).
`as.ltraj` creates an object of this class.
`summary.ltraj` returns the number of relocations (and missing values) for each "burst" of relocations and each animal.
`traj2ltraj`, and the reciprocal function `ltraj2traj` respectively converts an object of class `ltraj` to an object of class `traj`, and conversely.
`rec` recalculates the descriptive parameters of an object of class ltraj (e.g. after a modification of the contents of this object, see examples)

Usage

```
as.ltraj(xy, date, id, burst = id, typeII = TRUE,
         slsp = c("remove", "missing"))
## S3 method for class 'ltraj'
print(x, ...)
## S3 method for class 'ltraj'
summary(object, ...)
traj2ltraj(traj, slsp =  c("remove", "missing"))
ltraj2traj(x)
rec(x, slsp = c("remove", "missing"))
```

Arguments

`x, object`	an object of class `ltraj`
`xy`	a data.frame containing the x and y coordinates of the relocations
`date`	for trajectories of type II, a vector of class `POSIXct` giving the date for each relocation. For trajectories of type I, this argument is not taken into account.
`id`	either a character string indicating the identity of the animal or a factor with length equal to `nrow(xy)`
`burst`	either a character string indicating the identity of the burst of relocations or a factor with length equal to `nrow(xy)`
`typeII`	logical. `TRUE` indicates a trajectory of type II (time recorded, e.g. radio-tracking), whereas `FALSE` indicates a trajectory of type I (time not recorded, e.g. sampling of tracks in the snow)
`slsp`	a character string used for the computation of the turning angles (see details)
`traj`	an object of class `traj`
`...`	For other functions, arguments to be passed to the generic functions `summary` and `print`

Details

Objects of class `ltraj` allow the analysis of animal movements. They contain the descriptive parameters of the moves generally used in such studies (coordinates of the relocations, date, time lag, relative and absolute angles, length of moves, increases in the x and y direction, and dispersion R2n, see below).

The computation of turning angles may be problematic when successive relocations are located at the same place. In such cases, at least one missing value is returned. For example, let r1, r2, r3 and r4 be 4 successive relocations of a given animal (with coordinates (x1,y1), (x2,y2), etc.). The turning angle in r2 is computed between the moves r1-r2 and r2-r3. If r2 = r3, then a missing value is returned for the turning angle at relocation r2. The argument `slsp` controls the value returned for relocation r3 in such cases. If `slsp == "missing"`, a missing value is returned also for the relocation r3. If `slsp == "remove"`, the turning angle computed in r3 is the angle between the moves r1-r2 and r3-r4.

For a given individual, trajectories are often sampled as "bursts" of relocations. For example, when an animal is monitored using radio-tracking, the data may consist of several circuits of activity (two successive relocations on one circuit are often highly autocorrelated, but the data from two circuits may be sampled at long intervals in time). These bursts are indicated by the attribute `burst`. Note that the bursts should be unique: do not use the same burst id for bursts collected on different animals.

Two types of trajectories can be stored in objects of class `ltraj`: trajectories of type I correspond to trajectories where the time of relocations is not recorded. It may be because it could not be noted at the time of sampling (e.g. sampling of animals' tracks in the snow) or because the analyst decided that he did not want to take it into account, i.e. to study only its geometrical properties. In this case, the variable `date` in each burst of the object contains a vector of integer giving the order of the relocations in the trajectory (i.e. 1, 2, 3, ...). Trajectories of type II correspond to trajectories for which the time is available for each relocation. It is stored as a vector of class `POSIXct` in the column `date` of each burst of relocations. The type of trajectory should be defined when the object of class `ltraj` is defined, with the argument `typeII`.

Concerning trajectories of type II, in theory, it is expected that the time lag between two relocations is constant in all the bursts and all the ids of one object of class `ltraj` (don't mix animals located every 10 minutes and animals located every day in the same object). Indeed, some of the descriptive parameters of the trajectory do not have any sense when the time lag varies. For example, the distribution of relative angles (angles between successive moves) depends on a given time scale; the angle between two during 10-min moves of a whitestork does not have the same biological meaning as the angle between two 1-day move. If the time lag varies, the underlying process varies too. For this reason, most functions of adehabitat have been developed for "regular" trajectories, i.e. trajectories with a constant time lag (see `help(sett0)`). Furthermore, several functions are intended to help the user to transform an object of class `ltraj` into a regular object (see for example `help(sett0)`, and particularly the examples to see how regular trajectories can be obtained from GPS data).

Nevertheless, the class `ltraj` allows for variable time lag, which often occur with some modes of data collection (e.g. with Argos collars). But *we stress that their analysis is still an open question!!*

Finally, the class `ltraj` deals with missing values in the trajectories. Missing values are frequent in the trajectories of animals collected using telemetry: for example, GPS collar may not receive the signal of the satellite at the time of relocation. Most functions dealing with the class `ltraj` have a specified behavior in case of missing values.

It is recommended to store the missing values in the data *before* the creation of the object of class `ltraj`. For example, the GPS data imported within R contain missing values. It is recommended to *not remove* these missing values before the creation of the object!!! These missing values may present patterns (e.g. failure to locate the animal at certain time of the day or in certain habitat types), and *the analysis of these missing values should be part of the analysis of the trajectory* (e.g. see `help(runsNAltraj)` and `help(plotNAltraj)`.

However, sometimes, the data come without any information concerning the location of these missing values. If the trajectory is approximately regular (i.e. approximately constant time lag), it is possible to determine where these missing values should occur in the object of class `ltraj`. This is the role of the function `setNA`. For example of use of this class, type `demo(ltraj)`.

Value

`summary.ltraj` returns a data frame.
`ltraj2traj` returns an object of class `traj`.
All other functions return objects of class `ltraj`. An object of class `ltraj` is a list with one component per burst of relocations. Each component is a data frame with two attributes: the attribute `"id"` indicates the identity of the animal, and the attribute `"burst"` indicates the identity of the burst. Each data frame stores the following columns:

`x`	the x coordinate for each relocation
`y`	the y coordinate for each relocation
`date`	the date for each relocation (type II) or a vector of integer giving the order of the relocations in the trajectory.
`dx`	the increase of the move in the x direction. At least two successive relocations are needed to compute `dx`. Missing values are returned otherwise.
`dy`	the increase of the move in the y direction. At least two successive relocations are needed to compute `dy`. Missing values are returned otherwise.
`dist`	the length of each move. At least two successive relocations are needed to compute `dist`. Missing values are returned otherwise.
`dt`	the time interval between successive relocations
`R2n`	the squared net displacement between the current relocation and the first relocation of the trajectory
`abs.angle`	the angle between each move and the x axis. At least two successive relocations are needed to compute `abs.angle`. Missing values are returned otherwise.
`rel.angle`	the turning angles between successive moves. At least three successive relocations are needed to compute `rel.angle`. Missing values are returned otherwise.

Note

The class `ltraj` is a better alternative to the class `traj`. Indeed, the class `ltraj` already contains the basic information needed for the modelling of movement processes. Type `demo(managltraj)` for example of management of such objects, and `demo(analysisltraj)` for example of analysis.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>
Stephane Dray <`dray@biomserv.univ-lyon1.fr`>

References

Calenge, C., Dray, S. and Royer, M. (in prep.) Studying Animals movements with the R software: what is a trajectory?

See Also

`is.regular` and `sett0` for additional information on "regular" trajectories. `setNA` and `runsNAltraj` for additional information on missing values in trajectories. `c.ltraj` to combine several objects of class `ltraj`, `Extract.ltraj` to extract or replace bursts of relocations, `plot.ltraj` and `trajdyn` for graphical displays, `gdltraj` to specify a time period. For further information on the class `traj`, see `traj`.

Examples

```
data(puechabon)
locs <- puechabon$locs
locs[1:4,]
xy <- locs[,c("X","Y")]

######################################################
##
## Example of a trajectory of type I (time not recorded)

(litrI <- as.ltraj(xy, id = locs$Name, typeII=FALSE))
plot(litrI)

## The components of the object of class "ltraj"
head(litrI[[1]])


######################################################
##
## Example of a trajectory of type II (time recorded)


### Conversion of the date to the format POSIX
da <- as.character(locs$Date)
da <- as.POSIXct(strptime(as.character(locs$Date),"%y%m%d"))


### Creation of an object of class "ltraj", with for
### example the first animal
(tr1 <- as.ltraj(xy[locs$Name=="Brock",],
                 date = da[locs$Name=="Brock"],
                 id="Brock"))

## The components of the object of class "ltraj"
head(tr1[[1]])

## With all animals
(litr <- as.ltraj(xy, da, id = locs$Name))

## Change something manually in the first burst:
head(litr[[1]])
litr[[1]][3,"x"] <- 700000
```

```
## Recompute the trajectory
litr <- rec(litr)
## Note that descriptive statistics have changed (e.g. dx)
head(litr[[1]])
```

as.sahrlocs *Exploratory Analysis of Habitat Selection*

Description

as.sahrlocs creates objects of class sahrlocs. This class has a central place in habitat selection studies relying on radio-tracking data. Niche analysis and K-select analysis can be performed using this class of objects. This class may also be used for exploratory purposes. This class of object has three main components: an object of class kasc describing the study area, an object of class kasc describing what is available to the animals, and an object of class kasc describing the relocations of the animals (sahrlocs = Study Area - Home Range - reLOCationS). getsahrlocs converts one of the components of an object of class sahrlocs into an object of class kasc.

Usage

```
as.sahrlocs(mlocs, mhr, msa, descan = NULL)
getsahrlocs(x, what = c("sa", "hr", "locs"))
## S3 method for class 'sahrlocs'
print(x, ...)
```

Arguments

mlocs	an object of class kasc returned by the function count.points.id
mhr	an object of class kasc returned by the function hr.rast, by the function buffer.ani, or by any other user-defined function
msa	an object of class kasc describing the study area
descan	an optionnal data frame with the number of rows equal to the number of monitored animals. Each column of this data frame gives any type of information on the monitored animals (e.g. sex, age, and so on)
what	a character string giving the component of the sahrlocs object to be converted. Should be either "sa" (study area), "hr" (home ranges) or "locs" (relocations)
x	an object of class sahrlocs
...	additionnal parameters to be passed to the generic function print

Details

The different maps in the input list of objects kasc should describe the same area (same attributes: same cellsize, same xll, same yll and same dimensions).

Value

Returns one object of class `sahrlocs`, which is a list containing the input arguments for the function `as.sahrlocs`. Objects of class `sahrlocs` have the same attributes as objects of class `kasc` (`xll`, `yll`, `cellsize`, `nrow`, and `ncol`).

`getsahrlocs` returns an object of class `kasc` (see `kasc`).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for additionnal information on objects of class `kasc`, `hr.rast` and `buffer.ani` for the creation of the `"hr"` component of this object, `count.points.id` for the creation of the `"locs"` component of this object, `plot.sahrlocs` and `image.sahrlocs` for a graphical display of such objects.

Examples

```
data(puechabon)
kasc <- puechabon$kasc
locs <-  puechabon$locs

## Computes the home ranges of the animals...
cp <- mcp(locs[,4:5], locs[,1])

## ... and converts it to raster
cprast <- hr.rast(cp, kasc)
locrast <- count.points.id(locs[,4:5], locs[,1], kasc)

## Creation of the sahrlocs object
(sahr <- as.sahrlocs(locrast, cprast, kasc))

## adds information on the monitored animals
age <- factor(tapply(locs[,2], locs[,1], mean))
sex <- factor(tapply(locs[,3], locs[,1], mean))
info <- as.data.frame(cbind(sex, age))
(sahr <- as.sahrlocs(locrast, cprast, kasc, info))


## Gets the "study area" component of the object
toto <- getsahrlocs(sahr)
image(toto)
```

as.traj *Working with Trajects in 2D Space*

Description

The class `traj` is intended to explore trajects of animals monitored using radio-tracking.
`as.traj` creates an object of this class.
`summary.traj` returns the number of relocations for each "burst" of relocations and each animal.
`plot.traj` allows various graphical displays of the trajects.
`getburst` returns an object of class `traj` satisfying the specified criteria (selection of one focus animal, of a period of interest, of special "bursts" (see details)).
`traj2df`, and the reciprocal function `df2traj` respectively converts an object of class `traj` to an object of class `data.frame`, and conversely.

Usage

```
as.traj(id, xy, date, burst = id, ...)
## S3 method for class 'traj'
print(x, ...)
## S3 method for class 'traj'
summary(object, id = levels(object$id), date = NULL, ...)
## S3 method for class 'traj'
plot(x, id = levels(x$id), burst = levels(x$burst), date = NULL,
          asc = NULL, area = NULL,
          xlim = range(x$x), ylim = range(x$y),
          colasc = gray((256:1)/256), colpol = "green",
          addpoints = TRUE, addlines = TRUE,
          perani = TRUE, final = TRUE, ...)
getburst(x, burst = levels(x$burst),
         id = levels(x$id), date = NULL)
traj2df(x)
df2traj(df)
```

Arguments

`id`	a factor or a character vector giving for each relocation the identity of the individual monitored in `as.traj`. a character vector containing the identity of the individuals of interest in other functions
`xy`	a data frame with 2 columns containing the x and y coordinates of the relocations
`date`	a vector of class `POSIXct` giving the date for each relocation in `as.traj`. a vector of class `POSIXct` with length 2, indicating the beginning and the end of the period of interest in other functions
`burst`	a factor or a character vector giving the identity of each "burst" of relocations in `as.traj` (e.g. the circuit id, see details). The burst level needs to be unique (two animals cannot have the same burst levels). a character vector containing the burst levels of interest in `plot.traj` and `getburst`

`x`	an object of class `traj`
`object`	an object of class `traj`
`asc`	an object of class `asc`
`area`	an object of class `area` (see `help(area)`)
`xlim`	the ranges to be encompassed by the x axis
`ylim`	the ranges to be encompassed by the y axis
`colasc`	a character vector giving the colors of the map of class `asc`
`colpol`	a character vector giving the colors of the polygon contour map, when `area` is not `NULL`
`addlines`	logical. If `TRUE`, lines joining consecutive relocations are drawn
`addpoints`	logical. If `TRUE`, points corresponding to each relocation are drawn
`perani`	logical. If `TRUE`, one plot is drawn for each level of the factor `id`, and for a given animal, the several bursts are superposed on the same plot. If `FALSE`, one plot is drawn for each level of the factor `burst`
`final`	logical. If `TRUE`, the initial and final relocations of each burst are indicated in blue and red, respectively
`df`	a data frame to be converted to the class `traj`
`...`	other optional vectors containing some variables measured at each relocation (e.g. temperature, wind, elevation, etc.) in `as.traj`. For other functions, arguments to be passed to the generic functions `plot`, `summary` and `print`

Details

For a given individual, trajects are often sampled as "bursts" of relocations (Dunn and Gipson, 1977). For example, when an animal is monitored using radio-tracking, the data may consist of several circuits of activity (two successive relocations on one circuit are often highly autocorrelated, but the data from two circuits may be sampled at long intervals in time). These bursts are indicated by the factor `burst`.

Value

An object of class `traj` is a data frame with one column named `id`, one column named `x`, one column named `y`, one column named `date` and one column named `burst`. This class therefore inherits from the class `data.frame`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Dunn, J.E. and Gipson, P.S. (1977) Analysis of radio telemetry data in studies of home range. *Biometrics*. **59**, 794–800.

See Also

as.POSIXct and strptime for additional information of the class POSIX.

Examples

```
data(puechabon)
locs <- puechabon$locs
locs[1:4,]

### Conversion of the date to the format POSIX
da <- as.character(locs$Date)
da <- as.POSIXct(strptime(as.character(locs$Date),
               "%y%m%d"))


### Creation of the object of class "traj"
(tr <- as.traj(id = locs$Name, xy = locs[,c("X", "Y")],
               date = da))
summary(tr)
plot(tr)

### Displays on maps of the study area
k <- puechabon$kasc
ele <- getkasc(k, "Elevation")
plot(tr, asc = ele)
```

---

asc2im *Conversion of Maps of Class 'asc' and 'im' (Package spatstat)*

Description

These functions convert maps of class asc to objects of class im (package spatstats) and conversely.

Usage

```
asc2im(x)
im2asc(x)
```

Arguments

x	an object of class asc or im

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

asc for additionnal information on objects of class asc, and im for additionnal information on objects of class im

Examples

```
if (require(spatstat)) {

#######################
### Conversion asc -> im

  data(puechabon)
  el <- getkasc(puechabon$kasc, "Elevation")
  image(el, main = "An object of class \"asc\"")
  elim <- asc2im(el)
  image(elim, main = "An object of class \"im\"")

#######################
### Conversion im -> asc
  u <- matrix(rnorm(10000), 100, 100)
  haha <- im(u)
  image(haha, main = "class im")
  hihi <- im2asc(haha)
  image(hihi, main = "class asc")

}
```

---

ascgen *Creation of Raster Maps*

Description

ascgen creates an object of class asc using a set of points.

Usage

```
ascgen(xy = NULL, cellsize = NULL, nrcol = 10, count = TRUE)
```

Arguments

xy	a data frame with two columns containing the x and y coordinates of the points
cellsize	the cellsize attribute of the object of class asc to be created
nrcol	the size of the square raster map to be created (number of rows and columns)
count	logical. If TRUE, the object of class asc contains the number of points in each cell. If FALSE, all the cells are set to zero

Value

Returns an object of class `asc`.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

See Also

`asc` for additional information on objects of class `asc`.

Examples

```
data(puechabon)
lo <- puechabon$locs[,c("X","Y")]
plot(lo, asp = 1, pch = 16)

## lo contains the relocations of wild boars
rast <- ascgen(lo, cellsize = 100)
image(rast)

## Alternatively, one can specify the size of the square raster map
rast <- ascgen(lo, nrcol = 10)
rast
image(rast)

## can be used for further analyses
## (e.g. correspondence analyses)
locs <- puechabon$locs[, c("Name", "X", "Y")]
o <- count.points.id(locs[,2:3], locs[,1], rast)
image(o)
```

`bauges` *Census of chamois (Rupicapra rupicapra) in the Bauges mountains*

Description

This data set contains the relocations of 198 chamois groups in the Bauges mountains, as well as maps of 7 environmental variables on the study area.

Usage

```
data(bauges)
```

Details

This dataset contains a subsample of the data collected by volunteers and professionals working in various French wildlife and forest management, from 1994 to 2004 in the wildlife reserve of "les Bauges" (French Alps). Note that both the maps and the relocations have been slightly destroyed to preserve copyright on the data.

Source

Daniel Maillard, Office National de la chasse et de la faune sauvage, 95 rue Pierre Flourens, 34000 Montpellier, France

Examples

```
data(bauges)

image(bauges$kasc)
image(getkasc(bauges$kasc, "Elevation"))
points(bauges$locs, pch=16)
```

`bear` *GPS monitoring of one brown bear*

Description

These data contain the relocations of one female brown bear monitored using GPS collars during May 2004 in Sweden.

Usage

```
data(bear)
```

Source

Scandinavian Bear Research Project. Skandinaviska Bjornprojektet. Tackasen - Kareliusvag 2. 79498 Orsa, Sweden. email: info@bearproject.info

Examples

```
data(bear)
plot(bear)
```

`bighorn` *Radio-Tracking of Bighorn Sheeps*

Description

This data set describes the habitat use and availability for 6 bighorn sheeps monitored by radio-tracking (Arnett et al. 1989, in Manly et al., 2003, p. 67-74). 10 habitat types are considered.

Usage

```
data(bighorn)
```

Details

The object `bighorn` is a list, with the following components:

`used`	the number of resource units used by each animal (in rows) in each habitat category (in columns).
`availTrue`	the availability of each habitat category.
`availEstimated`	a sample of available resource units in each habitat category.

References

Manly, B.F.J., McDonald, L.L., Thomas, D.L., McDonald, T.L. & Erickson, W.P. (2003) *Resource selection by animals - Statistical design and Analysis for field studies. Second edition.* London: Kluwer academic publishers.

`biv.test` *Bivariate Test*

Description

`biv.plot` displays a bivariate plot. `biv.test` displays the results of a bivariate randomisation test.

Usage

```
biv.plot(dfxy, br = 10, points = TRUE, density = TRUE,
         kernel = TRUE, o.include = FALSE, pch, cex, col, h, sub,
         side = c("top", "bottom", "none"), ...)
biv.test(dfxy, point, br = 10, points = TRUE, density = TRUE,
         kernel = TRUE, o.include = FALSE, pch, cex, col, Pcol, h, sub,
         side = c("top", "bottom", "none"), ...)
```

Arguments

`dfxy`	a data frame with N lines (couples of values) and two columns
`br`	a parameter used to define the numbers of breaks of the histograms. A larger value leads to a larger number of breaks
`points`	logical. Whether the points should be displayed
`density`	logical. Whether the kernel density estimation should be displayed for the marginal histograms
`kernel`	logical. Whether the kernel density estimation should be displayed for the bivariate plot
`o.include`	logical. If `TRUE`, the origin is included in the plot
`pch`	plotting "character", i.e., symbol to use for the points. (see `?points`)
`cex`	character expansion for the points
`col`	color code or name for the points, see `?par`

`h`	vector of bandwidths for x and y directions, used in the function `kde2d` of the package `MASS`. Defaults to normal reference bandwidth (see `?kde2d`)
`sub`	a character string to be inserted in the plot as a title
`side`	if `"top"`, the x and y scales of the grid are upside, if `"bottom"` they are downside, if `"none"` no legend
`point`	a vector of length 2, representing the observation to be compared with the simulated values of the randomisation test
`Pcol`	color code or name for the observed point
`...`	further arguments passed to or from other methods

Details

`biv.test` is used to display the results of a bivariate randomisation test. An example of use of the function is provided in the function `niche.test`.

The x-axis of the main window corresponds to the first column of `dfxy`; the y-axis corresponds to the second column. Kernel density is estimated to indicate the contours of the distribution of randomised values. The two marginal histograms correspond to the univariate tests on each axis, for which the p-values are computed with `as.randtest` (package ade4, one-sided tests).

Warning

`biv.plot` and `biv.test` uses the function `kde2d` of the package `MASS`.

Author(s)

Mathieu Basille <`basille@ase-research.org`>

See Also

`as.randtest` (package ade4), `niche.test`

Examples

```
x = rnorm(1000,2)
y = 2*x+rnorm(1000,2)
dfxy = data.frame(x, y)

biv.plot(dfxy)
biv.plot(dfxy, points=FALSE, col="lightblue", br=20)

p = c(3, 4)
biv.test(dfxy, p)
biv.test(dfxy, p, points=FALSE, Pcol="darkred", col="lightblue", br=20)
```

`buffer` *Compute Buffers*

Description

`buffer` compute buffers around a set of locations.
`buffer.ani` is to be used when the points can be grouped into several categories (e.g. the relocations of several animals monitored using radio-tracking; the function `buffer` is then applied to each animal).
`buffer.line` compute buffers around a line. `buffer.ltraj` compute buffers around one or several animals trajectories.

Usage

```
buffer(pts, x, dist)
buffer.ani(pts, fac, x, dist)
buffer.line(xy, x, dist)
buffer.ltraj(ltraj, x, dist, perani=FALSE)
```

Arguments

`pts`	a data frame with two columns (x and y coordinates of the points)
`x`	either an object of class `asc` or `kasc` with the same attributes as those desired for the output, or an object of class `mapattr` (see `storemapattr`)
`dist`	a value of distance
`fac`	a factor defining the categories of the points
`xy`	a data frame containing the coordinates of the vertices of the lines
`ltraj`	an object of class `ltraj`
`perani`	logical. If `FALSE`, one buffer is computed for each burst of relocation. If `TRUE`, the buffers are computed per levels of the attribute `id`.

Value

`buffer` and `buffer.line` return an object of class `asc`, with 1 for pixels located within a specified distance of given points, and `NA` otherwise.
`buffer.ani` returns a data frame of class `kasc`, with each column corresponding to one level of the factor `fac`.
`buffer.ltraj` returns a data frame of class `kasc`, with each column corresponding to one level of the attribute `id` of the object of class `ltraj` passed as argument.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

kasc for additionnal information on objects of class kasc, asc for further information on objects of class asc, storemapattr for further information on objects of class mapattr.

Examples

```
data(puechabon)

# locs is the data frame containing the
# relocations of wild boars monitored by radio-tracking
locs <- puechabon$locs

# sa is the "kasc" object of maps of the study area
sa <- puechabon$kasc

# Buffer of 500 m around all relocations
bu <- buffer(locs[,4:5], sa, 500)
image(bu)

# displays all the pixels of the study area within 500 m
# of a relocation of each monitored wild boar
buani <- buffer.ani(locs[,4:5], locs[,1], sa, 500)
image(buani)


## buffer around a trajectory
data(puechcirc)
image(buffer.ltraj(puechcirc,sa, 100))

## buffer around a line
gc <- getcontour(getkasc(sa,1))
out <- buffer.line(gc[,2:3], sa, 300)
image(out)
lines(gc[,2:3], lwd=2)
```

---

burst *ID and Bursts of an Object of Class ltraj*

Description

Functions to get or set the attribute "id" or "burst" of the components of an object of class ltraj.

Usage

```
burst(ltraj)
burst(ltraj) <- value
```

```
id(ltraj)
id(ltraj) <- value
```

Arguments

`ltraj`	an object of class `ltraj`
`value`	a character vector of up to the same length as `ltraj`

Details

The functions `id` and `burst` are accessor functions, and `id<-` and `burst<-` are replacement function.

Value

For `id` and `burst`, a character vector of the same length as `ltraj`.

For `id<-` and `burst<-`, the updated object. (Note that the value of `burst(x) <- value` is that of the assignment, `value`, not the return value from the left-hand side.)

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj`, `names`

Examples

```
data(puechcirc)
puechcirc

## To see the ID and the burst
id(puechcirc)
burst(puechcirc)

## Change the burst
burst(puechcirc) <- c("glou", "toto", "titi")
puechcirc

burst(puechcirc)[2] <- "new name"
puechcirc

## Change the ID
id(puechcirc)[id(puechcirc)=="CH93"] <- "WILD BOAR"
puechcirc
```

c.ltraj *Combine Bursts of Relocations in Objects of Class "ltraj"*

Description

This function combines several objects of class `ltraj`.

Usage

```
## S3 method for class 'ltraj'
c(...)
```

Arguments

`...`	objects of class `ltraj` to be combined

Value

An object of class `ltraj`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for further information on the class `ltraj`, `Extract.ltraj` to extract or replace bursts of relocations, `plot.ltraj` and `trajdyn` for graphical displays, `gdltraj` to specify a time period

Examples

```

data(puechcirc)

(i <- puechcirc[1])
(j <- puechcirc[3])

(toto <- c(i,j))
```

`capreochiz` *GPS Monitoring of one Roe Deer in Chize (France)*

Description

This dataset contains the relocations of a roe deer collected using GPS collars in the Chize reserve (Deux-Sevres, France) by the ONCFS (Office national de la chasse et de la faune sauvage).

Usage

```
data(capreochiz)
```

Format

This dataset is a list containing the relocations in an object of class `ltraj` names `locs` and information on these relocations in an object named `"info"` (DOP, status, etc.).

Source

Sonia Said, Office national de la chasse et de la faune sauvage, CNERA-CS, 1 place Exelmans, 55000 Bar-le-Duc (France).

Examples

```
data(capreochiz)

plot(capreochiz$locs)

head(capreochiz$info)
```

`capreotf` *GPS Monitoring of one Roe Deer in Trois-Fontaines (France)*

Description

This dataset contains the relocations of a roe deer collected from May 1st to May 4th 2004 (every 5 minutes) using GPS collars in the wildlife reserve of Trois-Fontaines (Haute Marne, France) by the ONCFS (Office national de la chasse et de la faune sauvage).

Usage

```
data(capreotf)
```

Format

This dataset is a regular object of class `ltraj` (i.e. constant time lag).

Source

Sonia Said, Office national de la chasse et de la faune sauvage, CNERA-CS, 1 place Exelmans, 55000 Bar-le-Duc (France).

Examples

```
data(capreotf)

plot(capreotf)
```

---

chamois *Location of Chamois Groups in the Chartreuse Mountains*

Description

This data set describes the habitat use and availability by the chamois of the Chartreuse mountains (Isere, France), in 1992 and 1997. These data have been gathered during the hunting season (Fall).

Usage

```
data(chamois)
```

Details

The object `chamois` is a list containing the following components:
`locs` is a data frame containing the `x` and `y` coordinates of 198 chamois groups.
`map` is a map of class `kasc` describing the vegetation (Forest or Open areas), the distance from the ecotone Open/Forest and the slopes on the area.

References

Federation Departementale des Chasseurs de l'Isere, 65 av Jean Jaures, 38320 Eybens. France.

Chi *The Chi Distribution*

Description

Density, distribution function, quantile function and random generation for the chi distribution with `df` degrees of freedom.

Usage

```
dchi(x, df = 2)
pchi(q, df = 2, lower.tail = TRUE, ...)
qchi(p, df = 2, lower.tail = TRUE)
rchi(n, df = 2)
```

Arguments

`x,q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`df`	degrees of freedom (non-negative, but can be non-integer).
`lower.tail`	logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].
`...`	additional arguments to be passed to the function `integrate`.

Details

The chi distribution with `df` = n > 0 degrees of freedom has density

$$f_n(x) = 2^{1-n/2} x^{n-1} e^{\frac{-(x^2)}{2}} / \Gamma(n/2)$$

for x > 0. This distribution is used to describe the square root of a variable distributed according to a chi-square distribution.

Value

`dchi` gives the density, `pchisq` gives the distribution function, `qchisq` gives the quantile function, and `rchisq` generates random deviates.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Evans, M., Hastings, N. and Peacock, B. (2000) Statistical Distributions, 3rd ed. Wiley, New York.

See Also

`Chisquare`

Examples

```
opar <- par(mfrow = c(2,2))

hist(rchi(100), ncla = 20, main="The Chi distribution")

plot(tutu <- seq(0, 5, length=20), dchi(tutu, df = 2), xlab = "x",
     ylab = "probability density", type = "l")

plot(tutu, pchi(tutu), xlab = "x", ylab = "Repartition function",
     type = "l")

par(opar)
```

---

clusthr *Estimation of Home Range by Clustering*

Description

`clusthr` allows the estimation of the home range by clustering (see details).
`plot.clusthr` plots the results.
`clusthr.area` computes the home-range size for given percents of relocations included in the home range, and plots the results.
`getverticesclusthr` computes the coordinates of the vertices of the home ranges (`kver.rast` and `kver2shapefile` can be used to export the results toward GIS, see the help page of these functions).

Usage

```
clusthr(xy, id = NULL)
## S3 method for class 'clusthr'
print(x, ...)
## S3 method for class 'clusthr'
plot(x, whi = names(x), pch = 21,
             bgpts = "white", colpts = "black", cex = 0.7,
             plotit = TRUE, colpol = "grey", ...)
clusthr.area(x, percent = seq(20, 100, by = 5),
             unin = c("m", "km"), unout = c("ha", "km2", "m2"),
             plotit = TRUE)
getverticesclusthr(x, whi = names(x), lev=95)
```

Arguments

`xy`	a data frame with two columns containing the coordinates of the relocation of the monitored animals
`id`	a factor giving the identity of the animal for each relocation

`x`	an object of class `clusthr` returned by the function `clusthr`
`whi`	a vector of character indicating the animals to be plotted
`pch`	either an integer specifying a symbol or a single character to be used as the default in plotting points. See `points` for possible values and their interpretation.
`bgpts`	background ("fill") color for the open plot symbols given by `pch=21:25`
`colpts`	character. The color of the points
`cex`	The size of the points (see `help(par)`)
`plotit`	logical. Whether the plot should be drawn.
`colpol`	a character string indicating the colors to be used. Can be set to `"grey"`, `"terrain.colors"`, `"heat.colors"`, `"cm.colors"`, or `NA`
`percent,lev`	100 minus the proportion of outliers to be excluded from the home range
`unin`	the units of the relocations coordinates. Either `"m"` (default) for meters or `"km"` for kilometers
`unout`	the units of the output areas. Either `"m2"` for square meters, `"km2"` for square kilometers or `"ha"` for hectares (default)
`...`	additional arguments to be passed to the functions `plot` and `print`.

Details

This method estimates home range using a modification of single-linkage cluster analysis developped by Kenward et al. (2001). The clustering process is described hereafter: the three locations with the minimum mean of nearest-neighbour joining distances (NNJD) form the first cluster. At each step, two distances are computed: (i) the minimum mean NNJD between three locations (which corresponds to the next potential cluster) and (ii) the minimum of the NNJD between a cluster "c" and the closest location. If (i) is smaller that (ii), another cluster is defined with these three locations. If (ii) is smaller than (i), the cluster "c" gains a new location. If this new location belong to another cluster, the two cluster fuses. The process stop when all relocations are assigned to the same cluster.

At each step of the clustering process, the proportion of all relocations which are assigned to a cluster is computed (so that the home range can be defined to enclose a given proportion of the relocations at hand, i.e. to an uncomplete process). At a given step, the home range is defined as the set of minimum convex polygon enclosing the relocations in the clusters.

Value

`clusthr` returns a list of class `clusthr`. This list has one component per animal (named as the levels of argument `id`). Each component is itself a list, with the following sub-components:

`xy`	the animals' relocations
`results`	a data.frame with three columns: `step` indicates the step number of the algorithm, `clust` corresponds to the cluster assigned to some relocations, and `reloc` indicates the relocation(s) which is (are) assigned to the cluster "clust" at step "step"

`plot.clusthr` returns an invisible list (see `invisible`), with one component per animal. Each component is itself a list with one component per step of the clustering process. At each step, an object of class `area` describes the home range of the animal

`clusthr.area` returns a data.frame of class `hrsize`, which can be plotted using the generic function `plot`.

`getverticesclusthr` returns a list of class `kver`, where each element is an object of class `area`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Kenwward R.E., Clarke R.T., Hodder K.H. and Walls S.S. (2001) Density and linkage estimators of homre range: nearest neighbor clustering defines multinuclear cores. *Ecology*, **82**, 1905-1920.

See Also

`kver` for further information on objects of class `kver`.

Examples

```
data(puechabon)
lo<-puechabon$locs[,c("X","Y")]

## Home Range Estimation
res <- clusthr(lo, puechabon$locs$Name)

## Displays the home range
plot(res)

## Computes the home range size
clusthr.area(res)
```

`colasc` *Creates a Vector of Colors for a Raster Map of Type 'factor'*

Description

`colasc` creates a vector of colors for a raster map of class `asc` and of type `"factor"`.

Usage

```
colasc(x, ...)
```

Arguments

`x`	an object of class `asc`.
`...`	arguments named as the levels of the factor, with character values equal to the colors for these levels (see examples)

Value

Returns a character vector.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc`

Examples

```
data(puechabon)

## gets the aspect in an asc object
asp <- getkasc(puechabon$kasc, "Aspect")

## creates the vector of colors
cl <- colasc(asp, NorthEast = "blue", SouthEast = "red",
             SouthWest = "orange", NorthWest = "green")

## graphical display
image(asp, clfac = cl, main = "Aspect", xlab = "Lambert X",
      ylab = "Lambert Y")
legend(706500, 3162000, legend=levels(asp), fill = cl, cex = 0.7)
```

---

compana *Compositional Analysis of Habitat Use*

Description

compana performs a classical compositional analysis of habitat use (Aebischer et al., 1993).

Usage

```
compana(used, avail, test = c("randomisation", "parametric"),
        rnv = 0.01, nrep = 500, alpha = 0.1)
```

Arguments

`used`	a matrix or a data frame describing the percentage of use of habitats (in columns) by animals (in rows).
`avail`	a matrix or a data frame describing the percentage of availability of habitats (in columns) by animals (in rows).
`test`	a character string. If `"randomisation"`, randomisation tests are performed for both the habitat ranking and the test of habitat selection. If `"parametric"`, usual parametric tests are performed (chi-square for the test of habitat selection and t-tests for habitat ranking).
`rnv`	the number replacing the 0 values occurring in the matrix `used`.
`nrep`	the number of repetitions in the randomisation tests.
`alpha`	the alpha level for the tests.

Details

The compositional analysis of habitat use has been recommended by Aebischer et al. (1993) for the analysis of habitat selection by several animals, when the resources are defined by several categories (e.g. vegetation types).

This analysis is carried out in two steps: first the significance of habitat selection is tested (using a Wilks lambda). Then, a ranking matrix is built, indicating whether the habitat type in row is significantly used more or less than the habitat type in column. When this analysis is performed on radio-tracking data, Aebischer et al. recommend to study habitat selection at two levels: (i) selection of the home range within the study area, and (ii) selection of the relocations within the home range. The first level is termed second-order habitat selection on Johnson's scale (1980), and the second one, third-order habitat selection.

When zero values are found in the matrix of used habitats, they are replaced by a small value (by default, 0.01), according to the recommendations of Aebischer et al. (1993).

When zero values are found in the matrix of available habitats, the function `compana` uses the procedure termed "weighted mean lambda" described in Aebischer et al. (1993: Appendix 2), instead of the usual lambda (see examples). Zero values can be found in the matrix of available habitats when the third-order habitat selection is under focus. In this case, it may occur that some habitat types are available to some animals and not to the others.

Note that this method rely on the following hypotheses: (i) independence between animals, and (ii) all animals are selecting habitat in the same way (in addition to "traditional" hypotheses in these kinds of studies: no territoriality, all animals having equal access to all available resource units, etc.). The function `eisera` can be used as a preliminary to identify whether this is indeed the case (see examples).

Value

Returns a list of the class `compana`:

`used`	the matrix of used habitats
`avail`	the matrix of available habitats
`type.test`	a character string. Either `"randomisation"` or `"parametric"`
`test`	the results of the test of habitat selection
`rm`	the ranking matrix: a square matrix with `nh` rows and `nh` columns, where `nh` is the number of habitat types under study. At the intersection of the row `i` and of the column `j`, there is a `"+"` when the habitat `i` is more used than the habitat in column, and `"-"` otherwise. When the difference is significant, the sign is tripled.
`rmnb`	the matrix containing the number of animals used to perform the tests in `rm`.
`rank`	the rank of the habitat types. It is equal to the number of `"+"` for each habitat type in row of `rm`.
`rmv`	the matrix of statistics used to build `rm`. If `(test = "parametric")`, the matrix contains the values of `t`, in the t-test comparing the row and the column habitat. If `(test = "randomisation")`, the matrix contains the mean difference between the used and available log-ratios (see Aebischer et al., 1993).
`profile`	the profile of preferences: resource types are sorted so that the left type is the most preferred and the right type is the most avoided. Habitats for which the intensity of habitat selection is similar (no significant difference) are connected by a line.

Note

In the examples below, the results differ from those published in Aebischer et al. (squirrel example, selection of the relocations within the home range). In fact, there has been a confusion in the column names in the paper. Actually, Aebischer (pers. com.) indicated that the ranking matrix given in this example is correct.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

References

Aebischer, N. J. and Robertson, P. A. (1992) Practical aspects of compositional analysis as applied to pheasant habitat utilisation. pp. 285–293 In: Priede, G. and Swift, S. M. *Wildlife telemetry, remote monitoring and tracking of animals*.

Aebischer, N. J., Robertson, P. A. and Kenward, R. E. (1993) Compositional analysis of habitat use from animal radiotracking data. *Ecology*, **74**, 1313–1325.

Johnson, D. H. (1980) The comparison of usage and availability measurements for evaluating resource preference. *Ecology*, **61**, 65–71.

See Also

`eisera` to perform an eigenanalysis of selection ratios, preliminary to the use of compositional analysis.

Examples

```
## The examples presented here
## are the same as those presented in
## the paper of Aebischer et al. (1993)


#############################
## Pheasant dataset: first
## example in Aebischer et al.

data(pheasant)

## Second order habitat selection
## Selection of home range within the
## study area (example of parametric test)
pheana2 <- compana(pheasant$mcp, pheasant$studyarea,
                   test = "parametric")
pheana2

## The ranking matrix:
print(pheana2$rm, quote = FALSE)

## Third order habitat selection
## (relocation within home range)
## We remove the first pheasant of the analysis
## (as in the paper of Aebischer et al.)
## before the analysis
pheana3 <- compana(pheasant$locs[-1,], pheasant$mcp[-1,c(1,2,4)])
pheana3

## The ranking matrix:
print(pheana3$rm, quote = FALSE)



#############################
## Squirrel data set: second
## example in Aebischer et al.

data(squirrel)

## Second order habitat selection
## Selection of home range within the
## study area
squiana2 <- compana(squirrel$mcp, squirrel$studyarea)
squiana2

## The ranking matrix:
print(squiana2$rm, quote = FALSE)
```

```
## However, note that here, the hypothesis of identical use
## on which this analysis relies is likely to be false.
## Indeed, an eisera indicates:

us <- round(30 * squirrel$locs / 100)
av <- squirrel$studyarea
ii <- eisera(us, av, scannf = FALSE)
scatter(ii, grid = FALSE, clab = 0.7)

## There are clearly two groups of animals.  In such cases,
## compositional analysis is to be avoided in this case.




## Third order habitat selection
## (relocation within home range)
## We remove the second column
## (as in the paper of Aebischer et al.)
squiana3 <- compana(squirrel$locs[,-2], squirrel$mcp[,-2])
squiana3

## The ranking matrix:
print(squiana3$rm, quote = FALSE)
```

convnum   *Conversion from Factor to Numeric for Raster Map*

Description

The objects of class kasc may contain maps of type "numeric" (e.g. the elevation) or of type "factor" (e.g. the type of vegetation). With convnum, factor maps are transformed into a collection of k maps of type "numeric" (where k is the number of levels of the factor), with zero means and unit variance (see dudi.mix in the ade4 package for further details). Additionaly, maps of type "numeric" are also centered and scaled (see dudi.mix in the ade4 package for further details).

Usage

```
convnum(kasc)
```

Arguments

kasc	an object of class kasc

Value

Returns a list with the following components:

`kasc`	an object of class `kasc`
`weight`	the weights associated with each map of the object kasc, so that the collection of maps defining a categorical variable has the same weight as a numeric variable (see examples).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`dudi.mix` (package ade4), `kasc`

Examples

```
## loads the data
data(puechabon)
kasc <- puechabon$kasc
image(kasc)

## scales all the variables
toto <- convnum(kasc)
image(toto$kasc)

## the Aspect has four levels:
## four variables have been defined.
toto$weight

## The sum of the weights given to each
## level of aspect is equal to:
toto$weight[2:5]
sum(toto$weight[2:5])

## The same weight is therefore given to the factor variable Aspect and
## to any continuous variable, e.g. the elevation.
```

`count.points.id` *Number of Points in Each Pixel of a Raster Map*

Description

`count.points` counts the number of points in each pixel of a raster map of class `kasc` or `asc`.
`count.points.id` counts the number of points in each pixel of a raster map of class `kasc` or `asc`, for different sets of points (e.g. the relocations of several animals monitored using radio-tracking)

Usage

```
count.points(xy, w)
count.points.id(xy, id, w)
```

Arguments

`xy`	a data frame with 2 columns containing the x and y coordinates of the points.
`id`	a factor giving, for each point, the membership of a point to a set.
`w`	an object of class `asc`, `kasc` or `mapattr`.

Value

`count.points` returns an object of class `asc` containing the number of points in each cell of the raster map.
`count.points.id` returns an object of class `kasc`, with one column per level of the factor `id`, containing the number of points numbered in each cell of the raster map.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for additionnal information on objects of class `kasc`, and `storemapattr` for further information on objects of class `mapattr`.

Examples

```
data(puechabon)
kasc <- puechabon$kasc
locs <-  puechabon$locs

## Counts the number of relocations of each wild boar
## per pixel of the raster map
(nlocrast <- count.points.id(locs[,4:5], locs[,1], kasc))
image(nlocrast)

## Counts the number of all relocations
## per pixel of the raster map
(nlocrast <- count.points(locs[,c("X","Y")], kasc))
image(nlocrast)
```

`cutltraj` *Split Trajectories into Several Bursts*

Description

The function `cutltraj` split the bursts in an object of class `ltraj` into several "sub-bursts", according to some specified criterion.
The function `bindltraj` binds the bursts an object of class `ltraj` with the same attributes `"id"` into one unique burst.

Usage

```
cutltraj(ltraj, criterion, value.NA = FALSE, nextr = TRUE, ...)
bindltraj(ltraj, ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`criterion`	a character string giving any syntactically correct R logical expression implying the descriptive parameters in `x`
`value.NA`	logical. The value that should be used to replace the missing values.
`nextr`	logical. Whether the current "sub-burst" should stop after (`nextr = TRUE`) or before (`nextr = FALSE`) the first relocation matching `criterion`
`...`	additional arguments to be passed to other functions

Details

Splitting a trajectory may be of interest in many situations. For example, if it is known that two kinds of activities of the monitored animals correspond to different properties of, say, the distance between successive relocations, it may be of interest to split the trajectory according to the values of these distances.

Two options are available in `cutltraj`, depending on the value of `nextr`. If `nextr = FALSE`, any sequence of successive relocations that *do not* match the criterion is considered as a new burst. For example, if for a given burst, the criterion returns the vector `(FALSE, FALSE, FALSE, TRUE, TRUE, TRUE,    FALSE, FALS`
then the function `cutltraj` creates two new bursts of relocations, the first one containing the first 3 relocations and the second one the last 3 relocations.

If `nextr = TRUE`, any sequence of successive relocations that *do not* match the criterion, *as well as the first relocation that does match it after this sequence* is considered as a new burst. This option is available because many of the descriptive parameters associated to a given relocation in an object of class `ltraj` measure some specific feature concerning the position of the next relocation. For example, one may want to consider as a burst any sequence of relocations for which the time lag is below one hour (the criterion is `"dt > 3600"`. The first relocation for which this criterion is TRUE belong to the burst, and it is the next one which is excluded from the burst. For example, if for a given burst, the criterion returns the vector (`FALSE, FALSE, FALSE, TRUE, TRUE,  TRUE, FALSE, FALSE, FALSE`), then the function `cutltraj` creates two new bursts of relocations, the first one containing the first 4 relocations and the second one the last 3 relocations.

Value

An object of class `ltraj`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information about objects of class `ltraj` (and especially concerning the names of the descriptive parameters that can be used in `cutltraj`). `is.sd` (especially the examples of this help page) for other examples of use of this function

Examples

```
#######################################################
##
## GPS monitoring of one bear

data(bear)

## We want to study the trajectory of the day at the scale
## of the day. We define one trajectory per day. The trajectory should begin
## at 22H00
## The following function returns TRUE if the date is comprised between
## 21H00 and 22H00 (i.e. correspond to the relocation taken at 21H30)

foo <- function(date) {
     da <- as.POSIXlt(date, "GMT")
     ho <- da$hour + da$min/60
     return(ho>21&ho<22)
}

## We cut the trajectory into bursts after the relocation taken at 21H30:

bea1 <- cutltraj(bear, "foo(date)", nextr = TRUE)
bea1

## Remove the first and last burst:
bea2 <- bea1[-c(1,length(bea1))]


#######################################################
##
## Bind the trajectories
```

```
bea3 <- bindltraj(bea2)
bea3
```

---

distfacmap *Compute distances to the different levels of a factor map*

Description

This function computes map of distances to patches belonging to the different levels of a map of class `asc` and of type `factor`.

Usage

```
distfacmap(x)
```

Arguments

`x`	an object of class `asc` and of type `factor`

Value

An object of class `kasc`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc` for further info on objects of class `asc`. To compute distance maps from points, lines or polygons, see `distmap` in the package `spatstat`.

Examples

```
## Not run:
data(puechabon)
asp <- getkasc(puechabon$kasc, "Aspect")
image(asp)
sor <- distfacmap(asp)
image(sor)

## End(Not run)
```

domain — *Estimation of the Potential Distribution of a Species*

Description

`domain` uses the DOMAIN algorithm to estimate the potential distribution of a species based on a list of species occurrences and on maps of the area.

Usage

```
domain(kasc, pts, type = c("value", "potential"), thresh = 0.95)
```

Arguments

`kasc`	an object of class `kasc`
`pts`	a data frame giving the x and y coordinates of the species occurrences.
`type`	a character string. The `"value"` of the suitability may be returned or the `"potential"` area of distribution
`thresh`	if `value = "potential"`, a threshold value should be supplied for the suitability (by default 0.95)

Details

This function implements the DOMAIN algorithm described in Carpenter et al. (1993).

Value

Returns a matrix of class `asc`.

Warning

`domain` is restricted to maps containing only numerical variables (i.e. no factors).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Carpenter, G., Gillison, A.N. and Winter, J. (1993) DOMAIN: a flexible modelling procedure for mapping potential distributions of plants and animals. *Biodiversity and conservation*, **2**, 667–680.

See Also

`kasc` for additionnal information on objects of class `kasc`, `asc` for additionnal information on matrices of class `asc`.

Examples

```
## Preparation of the data
data(puechabon)
kasc <- puechabon$kasc
kasc$Aspect <- NULL
pts <- puechabon$locs[puechabon$locs$Name == "Brock", 4:5]

## View of the data
elevation <- getkasc(kasc, "Elevation")
image(elevation)
points(pts, col = "red", pch = 16)

## Estimation of habitat suitability map
hsm <- domain(kasc, pts)
image(hsm, col = grey((1:256)/256))
contour(hsm, add = TRUE)
## Lighter areas are the most preferred areas

## Potential distribution
hsm <- domain(kasc, pts, type = "potential")
image(elevation, main = "Habitat suitability map")
image(hsm, add = TRUE, col = "orange")
points(pts, col = "red", pch = 16)
```

---

dunnfa *Factorial Analysis of the Specialization in Habitat Selection Studies. Unpublished Work of James Dunn (University of Arkansas)*

Description

dunnfa performs a factorial decomposition of the Mahalanobis distances in habitat selection studies (see details).

Usage

```
dunnfa(dudi, pr, scannf = TRUE, nf = 2)
## S3 method for class 'dunnfa'
print(x, ...)
```

Arguments

dudi	an object of class pca
pr	a vector giving the utilization weights associated to each unit
scannf	logical. Whether the eigenvalues barplot should be displayed
nf	an integer indicating the number of kept axes
x	an object of class dunnfa
...	additional arguments to be passed to the function print

Details

This analysis is in essence very similar to the MADIFA (see `?madifa`). The Mahalanobis distances are often used in the context of niche-environment studies (Clark et al. 1993, see the function `mahasuhab`). Each resource unit takes a value on a set of environmental variables. Each environmental variable defines a dimension in a multidimensionnal space, namely the ecological space. A set of points (resource units) describes what is available to the species. For each point, a "utilization weight" measures the intensity of use of the point by the species. The set of points for which the utilization weight is greater than zero defines the "niche". The Mahalanobis distance between any resource unit in this space (e.g. the point defined by the values of environmental variables in a pixel of a raster map) and the centroid of the niche (the distribution of used resource units) can be used to give a value of eccentricity to this point.

For a given distribution of available resource units, for which a measure of Mahalanobis distances is desired, the MADIFA (MAhalanobis DIstances Factor Analysis) partitions the ecological space into a set of axes, so that the first axes maximises the average proportion of their squared Mahalanobis distances. James Dunn (formerly University of Arkansas) proposed the analysis programmed in the function `dunnfa`, as an alternative to the MADIFA (unpublished results). This analysis is closely related to both the ENFA (Ecological niche factor analysis, Hirzel et al. 2002) and the MADIFA.

The analysis proposed by James Dunn searches, in the multidimensional space defined by environmental variables, synthesis variables which maximise the ratio (variance of the scores of available resource units) / (variance of the scores of used resource units). This ratio is sometimes called "specialization" in the ecological literature (Hirzel et al. 2002). It is therefore very similar to the ENFA (which also maximises the specialization), except that the factorial axes returned by this analysis are not required to be *orthogonal to the marginality axis*.

James Dunn demonstrated that this analysis also partitions the Mahalanobis distances into uncorrelated axes, which makes it similar to the MADIFA (the difference is that the MADIFA maximises the mean squared Mahalanobis distances on the first axes, whereas the DUNNFA maximises the specialization on the first axes). Therefore, as for the MADIFA, the DUNNFA can be used to build reduced rank habitat suitability map.

Note that although this analysis could theoretically be used with all kinds of variables, it it currently implemented only for numeric variables.

Value

`dunnfa` returns a list of class `dunnfa` containing the following components:

`call`	original call.
`tab`	a data frame with n rows and p columns (original data frame centered by column for the uniform weighting).
`pr`	a vector of length n containing the number of points in each pixel of the map.
`nf`	the number of kept axes.
`eig`	a vector with all the eigenvalues of the analysis.

liA	row coordinates (centering on the centroid of the cloud of available points), data frame with n rows and nf columns.
liU	row coordinates (centering on the centroid of the cloud of available points), data frame with p rows and nf columns.
mahasu	a vector of length n containing the reduced-rank squared Mahalanobis distances for the n units.
co	column (environmental variables) coordinates, a data frame with p rows and nf columns
cor	the correlation between the DUNNFA axes and the original variable

Note

This analysis was developed by James Dunn during an e-mail discussion on the MADIFA, and is still unpublished work. Implemented in adehabitat with his autorization.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Clark, J.D., Dunn, J.E. and Smith, K.G. (1993) A multivariate model of female black bear habitat use for a geographic information system. *Journal of Wildlife Management*, **57**, 519–526.

Hirzel, A.H., Hausser, J., Chessel, D. & Perrin, N. (2002) Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? *Ecology*, **83**, 2027–2036.

Calenge, C., Darmon, G., Basille, M., Loison, A. and Jullien J.M. (2008) The factorial decomposition of the Mahalanobis distances in habitat selection studies. *Ecology*, **89**, 555–566.

See Also

madifa, enfa and gnesfa for related methods. mahasuhab for details about the Mahalanobis distances.

Examples

```
## Not run:
data(bauges)

map <- bauges$kasc
locs <- bauges$loc

## We prepare the data for the analysis
(datadun1 <- data2enfa(map, locs))

## We then perform the PCA before the analysis
pc <- dudi.pca(datadun1$tab, scannf = FALSE)
```

```
(dun <- dunnfa(pc, datadun1$pr, nf=2,
               scannf = FALSE))

## We should keep only one axis:
barplot(dun$eig)


## The correlation of the variables with the first two axes:
s.arrow(dun$cor)

## A factorial map of the niche (centering on the available points)
scatterniche(dun$liA, dun$pr, pts=TRUE)

## a map of the reduced rank Maalanobis distances
## (here, with one axis)
dun2 <- dunnfa(pc, datadun1$pr, nf=1,
              scannf = FALSE)
kas <- getkasc(df2kasc(data.frame(dun2$mahasu,dun2$mahasu),
               datadun1$index, map), 1)
image(kas)


## Compute the specialization on the row scores of
## the analysis:
apply(dun$liA, 2, function(x) {
   varav <- sum((x - mean(x))^2) / length(x)
   meanus <- sum(dun$pr*x)/sum(dun$pr)
   varus <- sum(dun$pr * (x - meanus )^2)/sum(dun$pr)
   return(varav/varus)
})
## The eigenvalues:
dun$eig


## End(Not run)
```

eisera *Eigenanalysis of Selection Ratios*

Description

Performs an eigenanalysis of selection ratios.

Usage

```
eisera(used, available, scannf = TRUE, nf = 2)
## S3 method for class 'esr'
print(x, ...)
```

```
## S3 method for class 'esr'
scatter(x, xax = 1, yax = 2,
            csub = 1, possub = "bottomleft", ...)
```

Arguments

`used`	a data frame containing the *number* of relocations of each animal (rows) in each habitat type (columns)
`available`	a data frame containing the *proportion* of availability of each habitat type (columns) to each animal (rows)
`scannf`	logical. Whether the eigenvalues bar plot should be displayed
`nf`	if `scannf = FALSE`, an integer indicating the number of kept axes
`x`	an object of class `esr`
`xax`	the column number for the x-axis
`yax`	the column number for the y-axis
`csub`	a character size for the legend, used with `par("cex")*csub`
`possub`	a string of characters indicating the sub-title position ("topleft", "topright", "bottomleft", "bottomright")
`...`	further arguments passed to or from other methods

Details

The eigenanalysis of selection ratios has been developped to explore habitat selection by animals monitored using radio-tracking, when habitat is defined by several categories (e.g. several vegetation types, see Calenge and Dufour 2006).

This analysis can be used for both designs II (same availability for all animals, e.g. selection of the home range within the study area) and designs III (different availability, e.g. selection of the sites within the home range). In the latter case, when some available proportions are equal to zero, the selection ratios are replaced by their expectation under random habitat use, following the recommendations of Calenge and Dufour (2006).

Value

A list of class `esr` and `dudi` containing also:

`available`	available proportions
`used`	number of relocations
`wij`	selection ratios

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Calenge, C. and Dufour, A.B. (2006) Eigenanalysis of selection ratios from animal radio-tracking data. *Ecology*. **87**, 2349–2355.

See Also

wi for further information about the selection ratios, compana for compositional analysis.

Examples

```
############################################################
############################################################
###
###  Example given in Calenge and Dufour 2006 (design II)


data(squirrel)

## computation of the number of relocations in each habitat type
## from the data given by Aebischer et al. (1993).
## squirrel$locs give the percentage of relocations in each habitat
## type, and Aebischer et al. (1993) indicate that there are 30
## relocations per animal.
## We therefore compute the number of relocations in each habitat type
## using:
us <- round(30 * squirrel$locs / 100)

## Habitat availability
av <- squirrel$studyarea

## Eigenanalysis of selection ratios
ii <- eisera(us, av, scannf = FALSE)

scatter(ii, grid = FALSE, clab = 0.7)

## The following graph may help the interpretation
## (see Calenge and Dufour 2006)
data(squirreloc)
locs <- squirreloc$locs
are <- squirreloc$map
co <- attr(are, "info")

li <- split(locs[,2:3], locs[,1])
opar <- par(mfrow=n2mfrow(length(li)), mar=c(0,0,2,0))
lapply(1:length(li), function(i) {
plot(are, colp = co[,2], main=names(li)[i], axes=FALSE)
points(li[[i]], pch=16, cex=1.5)
box()
})
plot(0,0, axes=FALSE, ty="n", xlim=c(-1,1), asp=1)
legend(-0.8,0.8, unique(co[,1]), fill=unique(co[,2]))
```

```
par(opar)


##########################################################
##########################################################
###
###  Example of design III

iii <- eisera(us, squirrel$mcp, scannf = FALSE)
scatter(iii, grid = FALSE, clab = 0.7)
```

---

enfa *Ecological Niche Factor Analysis*

Description

`enfa` performs an Ecological Niche Factor Analysis. `hist.enfa` draws histograms of the row scores or of the initial variables of the ENFA.
`data2enfa` prepares data (kasc and localizations) to be analyzed by the ENFA.

Usage

```
enfa(dudi, pr, scannf = TRUE, nf = 1)
## S3 method for class 'enfa'
hist(x, scores = TRUE, type = c("h", "l"), adjust = 1, Acol, Ucol,
          Aborder, Uborder, Alwd = 1, Ulwd = 1, ...)
data2enfa(kasc, pts)
```

Arguments

`dudi`	a duality diagram, object of class `dudi` (see Details)
`pr`	a vector giving the utilization weights associated to each unit
`scannf`	logical. Whether the eigenvalues barplot should be displayed
`nf`	an integer indicating the number of kept specialization axes
`x`	an object of class `enfa`
`scores`	logical. If `TRUE`, the histograms display the row scores of the ENFA. If `FALSE`, they display the niche on the environmental variables (in this case, this is equivalent to `histniche`)
`type`	what type of plot should be drawn. Possible types are: `"h"` for histograms, `"l"` for kernel density estimates (see `?density`). By default, `type = "h"` is used. If `type = "l"` is used, the position of the mean of each distribution is indicated by dotted lines
`adjust`	if `type = "l"`, a parameter used to control the bandwidth of the density estimates (see `?density`)

`Acol`	if `type = "h"`, a color to be used to fill the histogram of the available pixels. if `type = "l"`, a color to be used for the kernel density estimates of the available pixels
`Ucol`	if `type = "h"`, a color to be used to fill the histogram of the used pixels. if `type = "l"`, a color to be used for the kernel density estimates of the used pixels
`Aborder`	color for the border of the histograms of the available pixels
`Uborder`	color for the border of the histograms of the used pixels
`Alwd`	if `type = "l"`, the line width of the kernel density estimates of the available pixels
`Ulwd`	if `type = "l"`, the line width of the kernel density estimates of the used pixels
`kasc`	a raster map of class `kasc`
`pts`	a data frame with two columns, giving the coordinates of the species locations
`...`	further arguments passed to or from other methods

Details

The niche concept, as defined by Hutchinson (1957), considers the ecological niche of a species as an hypervolume in the multidimensional space defined by environmental variables, within which the populations of a species can persist. The Ecological Niche Factor Analysis (ENFA) has been developped by Hirzel et al. (2002) to analyse the position of the niche in the ecological space. Nicolas Perrin (1984) described the position of the niche in the n-dimensional space using two measures: the M-specialization (hereafter termed marginality) and the S-specialization (hereafter termed specialization). The marginality represents the squared distance of the niche barycentre from the mean available habitat. A large specialization corresponds to a narrow niche relative to the habitat conditions available to the species.

The ENFA first extracts an axis of marginality (vector from the average of available habitat conditions to the average used habitat conditions). Then the analysis extracts successives orthogonal axes (i.e. uncorrelated), which maximises the specialization of the species. The calculations used in the function are described in Hirzel et al. (2002).

The function `enfa` can be used on both quantitative variables and qualitative variables (though the interpretation of the results of the ENFA for qualitative variables is still under research), provided that the table containing the values of habitat variables (columns) for each resource unit (rows) is correctly transformed (e.g. column-centered and standardised for tables containing only quantitative variables), and that appropriate column weights are given (e.g. the sum of the weights for the levels of a factor should be the same as the weight of one quantitative variable). Therefore, the function `enfa` requires that a preliminary multivariate analysis is performed on the table (using analysis of the family of duality diagram, e.g. principal component analysis or Hill and Smith analysis). The object returned by this preliminary analysis contains the appropriate weights and transformation of the original data frame. For example, the function `dudi.mix` can be used first on the data.frame containing the value of both quantitative (e.g. slope, elevation) and qualitative habitat variables (e.g. vegetation) for each pixel of a raster map. The result of this analysis can then be passed as argument to the function `enfa` (see examples below).

Value

`enfa` returns a list of class `enfa` containing the following components:

`call`	original call.
`tab`	a data frame with n rows and p columns.
`pr`	a vector of length n containing the number of points in each pixel of the map.
`nf`	the number of kept specialization axes.
`m`	the marginality (squared length of the marginality vector).
`s`	a vector with all the eigenvalues of the analysis.
`lw`	row weights, a vector with n components.
`li`	row coordinates, data frame with n rows and nf columns.
`cw`	column weights, a vector with p components.
`co`	column coordinates, data frame with p rows and nf columns.
`mar`	coordinates of the marginality vector.

`data2enfa` returns a list of class `dataenfa` containing the following components:

`tab`	a data frame with n rows and p columns.
`pr`	a vector of length n containing the number of points in each pixel of the map.
`index`	an integer vector giving the position of the rows of `tab` in the initial object of class `kasc`.
`attr`	an object of class `mapattr` with the attributes of the initial `kasc`.

Author(s)

Mathieu Basille <`basille@ase-research.org`>

References

Hutchinson, G.E. (1957) Concluding Remarks. *Cold Spring Harbor Symposium on Quantitative Biology*, **22**: 415–427.

Perrin, N. (1984) Contribution a l'ecologie du genre Cepaea (Gastropoda) : Approche descriptive et experimentale de l'habitat et de la niche ecologique. These de Doctorat. Universite de Lausanne, Lausanne.

Hirzel, A.H., Hausser, J., Chessel, D. and Perrin, N. (2002) Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? *Ecology*, **83**, 2027–2036.

Basille, M., Calenge, C., Marboutin, E., Andersen, R. and Gaillard, J.M. (2008) Assessing habitat selection using multivariate statistics: Some refinements of the ecological-niche factor analysis. *Ecological Modelling*, **211**, 233–240.

See Also

`niche`, `kselect` for other types of analysis of the niche, when several species are under studies, `niche.test` to perform a test of the marginality and the tolerance of the niche, and `scatter.enfa` to have a graphical display of objects of class `enfa`. See `madifa` for another factorial analysis of the ecological niche.

Examples

```
data(lynxjura)

map <- lynxjura$map

## We keep only "wild" indices.
tmp <- lynxjura$locs[,4]!="D"
locs <- lynxjura$locs[tmp, c("X","Y")]
hist(map, type = "l")
## The variable artif is far from symetric

## We perform a square root transformation
## of this variable
## We therefore normalize the variable 'artif'
map[,4] <- sqrt(map[,4])
hist(map, type = "l")

## We prepare the data for the ENFA
(dataenfa1 <- data2enfa(map, locs))

## We then perform the PCA before the ENFA
pc <- dudi.pca(dataenfa1$tab, scannf = FALSE)

## The object 'pc' contains the transformed table (i.e.
## centered so that all columns have a mean of 0
## and scaled so that all columns have a variance of 1
## 'pc' also contains the weights of the habitat variables,
## and the weights of the pixels in the analysis

(enfa1 <- enfa(pc, dataenfa1$pr,
               scannf = FALSE))
hist(enfa1)
hist(enfa1, scores = FALSE, type = "l")

## randomization test and scatterplot
## Not run:
(renfa <- randtest(enfa1))
plot(renfa)
scatter(enfa1)

## End(Not run)
```

`engen2008II` *Measuring Habitat Selection Using the Method of Engen et al. (2008)*

Description

These functions implements the method described by Engen et al. to measure the preference of animals for habitat variables in habitat selection studies.

Usage

```
engen2008II(us, av, id, nsim = 500, nsimra = 500)

engen2008I(us, av, nsimra=500)

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

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

Arguments

`us`	a data frame containing the value of numeric habitat variables (columns) in each site (rows) used by the animals.
`av`	a data frame containing the value of numeric habitat variables (columns) in each site (rows) available to the animals.
`id`	a factor with as many elements as there are rows in `us`, indicating the ID of the animal that used the corresponding rows in `us`.
`nsim`	the number of randomizations used in the calculation of the total variance.
`nsimra`	the number of random allocation of ranks used in the calculation of the normal score (see details).
`x`	an object of class `engenetalI` or `engenetalII`
`...`	additional arguments to be passed to other functions (currently unused)

Details

Engen et al. (2008) proposed an original approach to measure the preference of animals for values of each particular variable of a multivariate set of environmental variables. Their approach was originally developed for the case where there is a sample of used site is for each animal in a sample of identified animals (e.g. using radiotelemetry or GPS), with several sites per animal (i.e., design II according to the classification of Thomas and Taylor, 1990). However, we extended this approach to also include the case where habitat use is described by a sample of used site, with one site per unidentified animal (i.e., design I).

The original approach is the following: first, a normal score transformation of each habitat variable is performed: for each variable, the empirical cumulative distribution is computed, by dividing the rank of the value of each available site by the number of observations. Note that the ties are ranked randomly. Then, the inverse of the standard normal integral (see `?qnorm`) of the cumulative distribution function is computed for all available sites: this results into a perfectly normal distribution of the habitat variables for the available sites. Then, the value of the cumulative distribution – estimated from the available sites – is computed for each used site. Then, the inverse of the standard normal integral is computed for each one.

Engen et al. (2008) suppose the following model describing how habitat use results from habitat availability. Let

$$Z_{ij}$$

be the value of a given habitat variable (transformed according to the normal score) for the j-th site used by the i-th animal. Then this value can be described by the model:

$$Z_{ij} = \mu + U_i + V_{ij}$$

where

$$\mu$$

is the preference for the habitat variable (0 indicates a non-preference),

$$U_i$$

and

$$V_{ij}$$

are normal distributions with means equal to zero and variances equal to

$$\sigma^2$$

and

$$\tau^2$$

respectively. Engen et al. give fomula for the estimation of these parameters. Their estimation is done by first estimating the total variance

$$\sigma^2 + \tau^2$$

(this variance is estimated by sampling randomly one observation per animal – the parameter `nsim` controls the number of samples used in this computation; see Engen et al. 2008). Note that the correlation between the value observed for two used units sampled from all the units used by a given animal is

$$\rho = \sigma^2/(\sigma^2 + \tau^2)$$

. A large value of rho indicates a large variation in the habitat used between animals (or a small within-animal variation). The main parameter of concern is here the preference. The function `engen2008II` allows to estimate these parameters.

The function `engen2008I` extends this model for design I studies (a sample of used sites and a sample of available sites, animals not identified), by considering the following model for these studies:

$$Z_{ij} = \mu + V_{ij}$$

where

$$\mu$$

is the preference for the habitat variable (0 indicates a non-preference), and

$$V_{ij}$$

are normal distributions with means equal to zero and variances equal to

$$\sigma^2$$

.

Note that the habitat variables may be correlated on the study area. In this case, observed preference for a given variable may be an artefact of other variables prefered by animals. Supposing that the data.frame containing the

$$Z_{ij}$$

's is a realization of a multivariate normal distribution, we can compute, for each habitat variable and each used site, the *conditional* mean

$$m_{ij}$$

and *conditional* standard deviation

$$s_{ij}$$

of this variable at this site, *given* the values of the other habitat variables at this site (this is done using the algorithm described by Ripley, 1987, p.98). We then compute the standardized values

$$P_{ij} = (Z_{ij} - m_{ij})/s_{ij}$$

. The preference is then computed using these standardized values.

Because there may be ties in the distribution of values of habitat variables, the results may vary depending on the random order chosen for ties when computing the normal scores. Engen et al. recommended to repeat the function a large number of times, and to use the mean values as estimates of the parameters. This is what the function does, and the number of randomization is controlled by the parameter `nsimra`.

Note that all these methods rely on the following hypotheses: (i) independence between animals, (ii) independence between sites, and (iii) all animals are selecting habitat in the same way (in addition to "traditional" hypotheses in these kinds of studies: no territoriality, all animals having equal access to all available resource units, etc., see Manly et al. 2002 for further details).

That the examples below provide an illustration and discussion of interesting and (at first sight) surprising properties of this method.

Value

`engen2008I` returns a list of class `engenetalI`, and `engen2008II` returns a list of class `engenetalII`. Both types of list contain two elements:

`raw`	this is a list containing one data frame per habitat variable, containing the value of the correlation rho (for `engenetalII` objects), mean preferences and standard error of these preferences (columns) for each randomization performed (rows);
`results`	a data frame containing the mean values over all the randomizations, of these parameters (columns) for each habitat variable (rows).

Note

Be patient! these functions can be very long (depending on the number of sites and on the value of `simra`)

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Engen, S., Grotan, V., Halley, D. and Nygard, T. (2008) An efficient multivariate approach for estimating preference when individual observations are dependent. *Journal of Animal Ecology*, **77**: 958–965.

Thomas, D. and Taylor, E. (1990) Study designs and tests for comparing resource use and availability. *Journal of Wildlife Management*, **54**, 322–330.

Ripley, B. (1987) Stochastic Simulation. John Wiley and Sons.

See Also

`niche`, `madifa`, `gnesfa` for another approach to tackle the study of habitat selection. For categorical variables, see `kselect`

Examples

```
## Not run:

#################################
#################################
#################################


## Practical use of engen2008II

data(puechabon)
kasc <- puechabon$kasc

## Removes the aspect (no factor allowed in the function)
kasc$Aspect <- NULL

## engen2008II:
avail <- kasc2df(kasc)$tab
use <- join.kasc(puechabon$locs[,c("X","Y")], kasc)
id <- puechabon$locs$Name

## This function can be very long:
engen2008II(use, avail, id, nsimra=10)
```

```
## Practical use of engen2008I

data(lynxjura)
ka <- lynxjura$map
lo <- lynxjura$locs[,1:2]
av <- kasc2df(ka)$tab
us <- join.kasc(lo, ka)

## Idem, be patient here:
engen2008I(us, av, nsimra=10)




#################################
#################################
#################################
##
##  For a deeper discussion on
##  this method... a simulation:



#################################
##
## First, simulation of a dataset
## copy and paste this part into R,
## but skip the reading of the
## comments if you are not interested
## into this simulation

## simulate the available points

set.seed(235)
av <- cbind(rnorm(1000, mean=0, sd=3), rnorm(1000, mean=0, sd=0.5))
tt <- cbind(c(cos(-pi/4),sin(-pi/4)),c(cos(pi/4), sin(pi/4)))
av <- as.data.frame(as.matrix(av)%*%tt)

## simulate the used points: we simulate a selection on the first
## principal component of the PCA of the data.frame describing the
## availability. In other words, we simulate the case where the
## habitat selection occurs on the "common part" of the two habitat
## variable (no preference for one particular variable).

us <- do.call("rbind", lapply(1:5, function(i) {
    us1 <- cbind(rnorm(30, mean=rnorm(1, -4, 1), sd=0.5),
                 rnorm(30, mean=rnorm(1, 0, 0.5), sd=0.2))

    return(us1%*%tt)
}))
colnames(us) <- colnames(av) <- c("var1", "var2")
id <- gl(5,30)
```

```
#################################
##
## Study of the habitat selection on these data
## The data are:
## - us: a matrix containing the used sites for two
##       habitat variables
## - av: a matrix containing the available sites for two
##       habitat variables
## - id: a vector containing the id of 5 animals

## First illustrate the use and availability of the two variables:

plot(av, xlab="Habitat variable 1", ylab="Habitat variable 2",
     col="grey", pch=16)
lius <- split(as.data.frame(us), id)
junk <- lapply(1:5, function(i) points(lius[[i]], pch=16, col=i))

## -----> ***It is very clear that there is a selection***:
## the animals select the low values of both habitat variables.
## (this is what we actually simulated


## Now perform the method of Engen et al. (2008):
engen2008II(us, av, id)


## Surprisingly, the method seems to fail to identify the clear
## habitat selection identified graphically...
##
## In fact, it does not fail:
## this method identifies the part of habitat selection that is clearly
## attributable to a given variable.  Here the animals select the
## the common factor expressed in the two variables, and it is impossible
## to identify whether the selection is due only to the variable 1 or to
## the variable 2: it is caused by both variable simultaneously.
## Once the selection on the variable 2 (including the common part)
## has been removed, there is no longer appearant selection on
## variable 1.  Once the selection caused by the variable 1
## (including the common part) has been removed, there is no
## longer selection on variable 2...
##
## For this reason, Engen et al. recommended to use this method
## concurrently with other factor analyses of the habitat selection
## such as madifa, kselect, niche (in ade4 package), etc.
##
## Note also the strong correlation between the value of two random
## points used by a given animal. This indicates a strong variability
## among animals...
## End(Not run)
```

`explore.kasc` *Interactive Exploration of Maps of Class kasc (requires the package tkrplot)*

Description

This interface allows to explore distances, values, etc. on a map of class `kasc`.

Usage

```
explore.kasc(ka, coltxt = "blue", hscale = 1, vscale = 1, ...)
```

Arguments

`ka`	An object of class `kasc`
`coltxt`	character. the color of the text to be printed
`hscale`	passed to tkrplot
`vscale`	passed to tkrplot
`...`	not yet implemented

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for further information about objects of class `kasc` and `image.kasc` for graphical display

Examples

```
## Not run:
if (require(tkrplot)) {
 data(puechabon)
 explore.kasc(puechabon$kasc)
}

## End(Not run)
```

`Extract.ltraj`	*Extract or Replace Parts of an Object of Class ltraj*

Description

Extract or replace subsets of objects of class `ltraj`.

Usage

```
  ## S3 method for class 'ltraj'
x[i, id, burst]
  ## S3 replacement method for class 'ltraj'
x[i, id, burst] <- value
```

Arguments

`x`	an object of class `ltraj`
`i`	numeric. The elements to extract or replace
`id`	a character vector indicating the identity of the animals to extract or replace
`burst`	a character vector indicating the identity of the bursts of relocations to extract or replace
`value`	an object of class `ltraj`

Details

Objects of class `ltraj` contain several bursts of relocations. This function subsets or replaces these bursts, based on their indices or on the attributes `id` *or* `burst`.

Value

An object of class `ltraj`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj`, `gdltraj`

Examples

```
data(puechcirc)
puechcirc

## Extract the second and third bursts
(toto <- puechcirc[2:3])

## Extracts all bursts collected on the animal JE
puechcirc[id = "JE93"]


## Replace one burst
toto[2] <- puechcirc[1]
toto
```

`findmaxasc` *Find Local Maxima on a Map of Class 'asc'*

Description

`findmaxasc` finds the local maxima on a map of class `asc`.

Usage

```
findmaxasc(asc)
```

Arguments

`asc`	a map of class `asc`

Details

This function may be useful, among other things, to identify the local modes of the utilization distribution of an animal (e.g. estimated using `kernelUD`).

Value

a data frame with two columns containing the x and y coordinates of the local maxima.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`as.asc`, `kernelUD`

Examples

```
## example, to find the mode of the utilization distribution (UD) of an
## animal:

## load the data
data(puechabon)
loc <- puechabon$locs[, c("X", "Y")]
id <- puechabon$locs[, "Name"]

## Estimation of UD for the four animals
(ud <- kernelUD(loc, id))
image(ud)


## Now consider the UD of Chou
map <- ud$Chou$UD
image(map)

## Find the local maxima:
maxim <- findmaxasc(map)
points(maxim, col="red", pch=16, cex=1.5)
```

`fpt` *Computation of the First Passage Time From Trajectories*

Description

These functions compute the first passage time using trajectories of class `"ltraj"` of type II (time recorded).

Usage

```
fpt(lt, radii, units = c("seconds", "hours", "days"))
varlogfpt(f, graph = TRUE)
meanfpt(f, graph = TRUE)
## S3 method for class 'fipati'
plot(x, scale, warn = TRUE, ...)
```

Arguments

`lt`	an object of class `"ltraj"` of type II (time recorded)
`radii`	a numeric vector giving the radii of the circles
`units`	The time units of the results
`f,x`	an object of class `fipati` returned by the function `fpt`

`graph`	logical. Whether the results should be plotted
`scale`	the value of the radius to be plotted
`warn`	logical. Whether the function should warn the user when the given scale does not correspond to possible radii available in the object of class `fipati`
`...`	additional arguments to be passed to the generic function `plot`

Details

The first passage time (FPT) is a parameter often used to describe the scale at which patterns occur in a trajectory. For a given scale r, it is defined as the time required by the animals to pass through a circle of radius r. Johnson et al. (1992) indicated that the mean first passage time scales proportionately to the square of the radius of the circle for an uncorrelated random walk. They used this property to differenciate facilitated diffusion and impeded diffusion, according to the value of the coefficient of the linear regression `log(FPT) = a * log(radius) + b`. Under the hypothesis of a random walk, `a` should be equal to 2 (higher for impeded diffusion, and lower for facilitated diffusion). Note however, that the value of a converges to 2 only for large values of radius.

Fauchald & Tveraa (2003) proposed another use of the FPT. Instead of computing the mean of FPT, they propose the use of the variance of the log(FPT). This variance should be high for scales at which patterns occur in the trajectory (e.g. area restricted search). This method is often used to determine the scale at which an animal seaches for food.

Value

`fpt` computes the FPT for each relocation and each radius, and for each animals. This function returns an object of class `"fipati"`, i.e. a list with one component per animal. Each component is a data frame with each column corresponding to a value of `radii` and each row corresponding to a relocation. An object of class `fipati` has an attribute named `"radii"` corresponding to the argument `radii` of the function `fpt`.

`meanfpt` and `varlogfpt` return a data frame giving respectively the mean FPT and the variance of the log(FPT) for each animal (rows) and rach radius (column). These objects also have an attribute `"radii"`.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

References

Johnson, A. R., Milne, B.T., & Wiens, J.A. (1992) Diffusion in fractal landscapes: simulations and experimental studies of tenebrionid beetle movements. *Ecology* **73**: 1968–1983.

Fauchald, P. & Tveraa, T. (2003) Using first passage time in the analysis of area restricted search and habitat selection. *Ecology* **84**: 282–288.

See Also

`ltraj` for additional information on objects of class `ltraj`

Examples

```
data(puechcirc)
i <- fpt(puechcirc, seq(300,1000, length=30))
plot(i, scale = 500, warn = FALSE)

toto <- meanfpt(i)
toto
attr(toto, "radii")


toto <- varlogfpt(i)
toto
attr(toto, "radii")
```

`gdltraj` *Working with Trajectories: Specify a Time Period*

Description

Gets the parts of the trajectories stored in an object of class `ltraj` of type II (time recorded), corresponding to a specified time period.

Usage

```
gdltraj(x, min, max, type = c("POSIXct", "sec", "min", "hour", "mday",
        "mon", "year", "wday", "yday"))
```

Arguments

`x`	an object of class `ltraj` of type II (time recorded)
`min`	numeric. The beginning of the period to consider
`max`	numeric. The end of the period to consider
`type`	character. The time units of `min` and `max`

Details

The limits of the period to consider may correspond to any of the components of the list of class `POSIXlt` (hour, day, month, etc.; see `help(POSIXlt)`), or to dates stored in objects of class `POSIXct` (see examples).

Value

an object of class `ltraj`.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

ltraj for further information about objects of class ltraj, POSIXlt for further information about objects of class POSIXlt

Examples

```
data(puechcirc)
plot(puechcirc, perani = FALSE)

## Gets all the relocations collected
## between midnight and 3H AM
toto <- gdltraj(puechcirc, min = 0, max = 3, type="hour")
plot(toto, perani = FALSE)

## Gets all relocations collected between the 15th
## and the 25th august 1993
lim <- as.POSIXct(strptime(c("15/08/1993", "25/08/1993"),
                  "%d/%m/%Y"))
tutu <- gdltraj(puechcirc, min = lim[1],
                max = lim[2], type="POSIXct")
plot(tutu, perani = FALSE)
```

---

getascattr — *Copy the Attributes of an Object of Class 'asc' or 'kasc' to another Object*

Description

getascattr copies the attributes of an object of class asc to another matrix of the same size. getkascattr performs the same operation for objects of class kasc.

Usage

```
getascattr(xfrom, xto, type = c("numeric", "factor"), lev = NULL)
getkascattr(xkfrom, xkto)
```

Arguments

xfrom	an object of class asc
xto	a matrix with the same number of rows and columns as xfrom
type	a character string giving the type of the map ("factor" for maps of categorical variables, and "numeric" otherwise)

`lev`	if `type = "factor"`, a character vector giving the levels of the mapped variable (see `help(asc)`)
`xkfrom`	an object of class `kasc`
`xkto`	a data frame with the same number of rows and columns as `xkfrom`

Value

`getascattr` returns a raster matrix of class `asc`,
`getkascattr` returns a data frame of class `kasc`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for additionnal information on objects of class `kasc`, `asc` for additionnal information on objects of class `asc`

Examples

```
data(puechabon)

## my.map is a map of elevation
my.map <- getkasc(puechabon$kasc, "Elevation")
sl <- getkasc(puechabon$kasc, "Slope")
attributes(sl) <- NULL
sl <- matrix(sl, ncol = ncol(my.map))

## sl is a matrix with the same size as my.map
toto <- getascattr(my.map, sl)
image(toto)

## Same rationale with aspect
asp <- getkasc(puechabon$kasc, "Aspect")
le <- levels(asp)
attributes(asp) <- NULL
asp <- matrix(asp, ncol = ncol(my.map))

## asp is now a matrix with the same size as my.map
tutu <- getascattr(my.map, asp, typ = "factor", lev = le)
cl <- colasc(tutu, NorthEast = "blue", SouthEast = "red",
           SouthWest = "orange", NorthWest = "green")
image(tutu, clfac = cl)
```

getcontour *Computes the Contour Polygon of a Raster Object*

Description

getcontour computes the contour polygon of a raster object of class asc. When the object is made of several parts, the function returns one polygon per part.

Usage

```
getcontour(x)
```

Arguments

x	an object of class asc

Value

Returns an object of class area.

Warning

Holes in the polygons are not taken into account by the function.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

asc for additionnal information on objects of class asc, area for additional information on objects of class area, and area2dxf to export the results toward a GIS.

Examples

```
data(puechabon)
kasc <- puechabon$kasc

##########################################
## Example with one object:
## Gets the first map of the "kasc" object
## Map of the elevation
elev <- getkasc(kasc, "Elevation")
image(elev)

## Get the contour polygon
cpol <- getcontour(elev)
```

```
## Draw the polygon
i <- cpol[,2:3]
polygon(i, col = "green", lwd = 2)

########################################
## Example with two objects:
## home ranges of wild boar

hr <- getsahrlocs(puechabon$sahr, "hr")
u <- getkasc(hr, "Jean")
image(u)

## Get the contour polygons
p <- getcontour(u)
plot(p, lwd = 2)
```

`getXYcoords` *Computes the X and Y Coordinates of the Pixels of a Raster Map*

Description

`getXYcoords` computes the geographical coordinates of the rows and columns of pixels of a raster map.

Usage

```
getXYcoords(w)
```

Arguments

`w`	an object of class `asc`, `kasc`, `sahrlocs`, or `mapattr`.

Value

Returns a list with two components:

`x`	the x coordinates of the columns of pixels of the map
`y`	the y coordinates of the rows of pixels of the map

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc`, `kasc`, `as.sahrlocs`, `storemapattr` for additionnal informations on objects of class `area`, `kasc`, `sahrlocs`, `mapattr` respectively

Examples

```
data(puechabon)
(elev <- getkasc(puechabon$kasc, "Elevation"))
(coords <- getXYcoords(elev))
nrow(elev) == length(coords$x)
ncol(elev) == length(coords$y)
```

---

gnesfa *General Niche-Environment System Factor Analysis*

Description

The function gnesfa allows to perform a general niche-environment system factor analysis.

Usage

```
gnesfa(dudi, Focus, Reference,
       centering = c("single", "twice"),
       scannf = TRUE, nfFirst = 2, nfLast = 0)
## S3 method for class 'gnesfa'
print(x, ...)
```

Arguments

dudi	an object of class dudi
Focus	a vector containing the focus weights
Reference	a vector containing the reference weights
centering	a character string indicating the type of centering (see details)
scannf	a logical value indicating whether the eigenvalues bar plot should be displayed
nfFirst	the number of first axes to be kept
nfLast	the number of last axes to be kept
x	an object of class GNESFA
...	further arguments to be passed to other functions

Details

The GNESFA is an algorithm which generalises several factor analyses of the ecological niche. A table X gives the values of P environmental variables in N resource units (e.g. the pixels of a raster map). A distribution of weights D describes the availability of the resource units to the species (if not specified, these weights are considered to be uniform). Another distribution of weights Dp describes the use of the resource units by the species (for example the proportion of relocations in each pixel of a raster map).

Each environmental variable defines a dimension in a multidimensional space, the ecological space. The N resource units define a cloud of points in this space. Each point is associated to two weights.

The GNESFA finds, in the ecological space, the directions on which these two distributions of weights are the most different.

The GNESFA relies on a choice of the analyst, followed by three steps. Before all, the analyst has to choose one distribution of weights as the Reference distribution, and the other one as the Focus distribution; (i) The first table X is centred on the centroid of the Reference distribution; (ii) a principal component analysis of this Reference distribution is performed; (iii) the cloud of points is distorted, so that the Reference distribution takes a standard spherical shape; (iv) a non centred principal component analysis of the Focus distribution allows to identify the directions of the ecological space where the two distributions are the most different.

Depending on the distribution chosen as Reference, this algorithm returns results with different meanings (see examples). This algorithm is closely related to several common analyses of habitat selection/niche (ENFA, MADIFA, Mahalanobis distances, selection ratios, etc.). The examples below give some examples of the mathematical properties of this algorithm.

Note that the function takes a parameter named `centering`. Indeed, two types of centering can be performed prior to the GNESFA. The choice `"single"` consists in the centering of the cloud of point in the ecological space on the centroid of the Reference distribution. The choice `"twice"` consist to center the cloud of points on both the centroid of the Reference distribution and the centroid of the Focus distribution. This is done by projecting the cloud of points on the hyperplane orthogonal to the marginality vector (the vector connecting the two centroids. If this choice is done, the GNESFA is identical to the commonly used Ecological Niche Factor Analysis (see examples).

Value

`gnesfa` returns a list of class `gnesfa` containing the following components:

`call`	original call.
`centering`	The type of centering required.
`tab`	a data frame with n rows and p columns.
`Reference`	a vector of length n containing the Reference weights.
`Focus`	a vector of length n containing the Focus weights.
`nfFirst`	the number of kept first axes.
`nfLast`	the number of kept last axes.
`eig`	a vector with all the eigenvalues of the analysis.
`li`	row coordinates, data frame with n rows and nf columns.
`l1`	row normed coordinates, data frame with n rows and nf columns.
`co`	column scores, data frame with p rows and nf columns.
`cor`	the correlation between the GNESFA axes and the original variables

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Calenge, C. and Basille, M. (in prep.) A General Framework for the Exploration of the Ecological Niche.

See Also

`madifa`, `mahasuhab`, `enfa`, `wi` for closely related methods (see Examples)

Examples

```
## Not run:

#################################################################
##
## Study of the habitat selection by the chamois in the French
## mountains of Les Bauges


## Loads the data
data(bauges)
names(bauges)
kasc <- bauges$kasc
locs <- bauges$locs


## displays the data
image(kasc)
image(getkasc(kasc,1))
points(locs, col = "red", pch = 16)


## Prepares the data for the GNESFA:
litab <- kasc2df(kasc)
pc <- dudi.pca(litab$tab, scannf = FALSE)
Dp <- count.points(locs, kasc)[litab$index]

## Example of use with Dp = Reference
gn <- gnesfa(pc, Reference = Dp, scannf=FALSE)

## One main axis:
barplot(gn$eig)

## The correlation with variables indicate that
## the elevation, the proximity to grass and to
## deciduous forests:
s.arrow(gn$cor)

## The factorial map of the niche...
scatterniche(gn$li, Dp, pts = TRUE)

## The chamois is rather located at high elevation,
## in the grass, far from deciduous forests




##########################################################
##########################################################
##
##
##       Some interesting properties of the GNESFA
##
##
##########################################################
##########################################################




################################
################################
##
## Interesting properties of the
## choice: Dp as Reference
## identical to the MADIFA
## (Calenge et al. in revision,
## See the help page of the function madifa
## for other properties)

gn <- gnesfa(pc, Reference = Dp, scannf=FALSE,
             nfFirst = 7)
gn

## This is the same as the MADIFA:
mad <- madifa(pc, Dp, scannf=FALSE)

## Indeed:
plot(gn$li[,1], mad$li[,1])
cor(gn$li[,1], mad$li[,1])


## And consequently the sum of the squared scores,
## On the axes of the GNESFA...
su <- apply(gn$l1,1,function(x) sum(x^2))


## ... is equal to the Mahalanobis distances between
## the points and the centroid of the niche
## (Clark et al. 1993, see the help page of mahasuhab)

su2 <- mahasuhab(kasc, locs)[litab$index]


## Indeed:
all(su - su2 < 1e-7)
```

```
plot(su, su2)




################################
################################
##
## Centering twice is identical to
## the ENFA (Hirzel et al. 2002, see the help
## page of the function enfa)...


#######
##
## ... If Dp is the Reference:

gn <- gnesfa(pc, Reference = Dp, center = "twice", scannf = FALSE)
gn

enf <- enfa(pc, Dp, scannf = FALSE)
plot(enf$li[,2], gn$li[,1])
cor(enf$li[,2], gn$li[,1])

## The first specialization axis of the ENFA
## is the first axis of the GNESFA!


#######
##
## ... If Dp is the Focus:

gn <- gnesfa(pc, Focus = Dp, center = "twice",
             scannf = FALSE, nfFirst = 6)
plot(enf$li[,2], gn$li[,6])
cor(enf$li[,2], gn$li[,6])

## The first specialization axis of the ENFA
## is the last axis of the GNESFA!


#######
##
## Whatever the distribution chosen as Reference,
## projecting the cloud of points on the hyperplane
## orthogonal to the marginality axis, and performing
## a GNESFA in this subspace is identical to an ENFA!


## The marginality axis of the ENFA is identical
## to the component "projmar" of the GNESFA

plot(enf$li[,1],gn$projmar)
```

```
cor(enf$li[,1],gn$projmar)




################################
################################
##
## Interesting properties of the
## case: Dp as Focus, one categorical
## variable. Relationships with the selection
## ratios of Manly et al. (1972, see the
## help page of wi)


## For example, take the Elevation, and
## define a factor with 4 levels
elev <- data.frame(el = cut(litab$tab$Elevation, 4))

## Now, compute the complete disjonctive table
dis <- acm.disjonctif(elev)
head(dis)

## Now perform the GNESFA with Dp as Focus:
pc <- dudi.pca(dis, scannf = FALSE)
gn <- gnesfa(pc, Dp, scannf = FALSE, nfFirst = 3)


#######
##
## This analysis is closely related to the concept of
## selection ratios

## Compute the percentage of use of each level:
us <- apply(dis, 2, function(x) sum(x*Dp)/sum(Dp))
av <- apply(dis, 2, function(x) sum(x)/length(x))

## The selection ratios
wi <- widesI(us, av)$wi

## Compute the sum of the eigenvalue
sum(gn$eig)

## Compute the sum of the selection ratios - 1
sum(wi) - 1

## In other words, when the GNESFA (Dp as Focus) is
## applied on only one categorical variable, this
## analysis finds a set of axes which partition the
## sum of the selection ratios so that it is maximum
## on the first axes!!
## End(Not run)
```

hbrown — *Estimates the value of h for a Brownian motion*

Description

hbrown estimates the scaling factor h (used in the Brownian motion, see help(simm.brown)) from a trajectory.

Usage

```
hbrown(x)
```

Arguments

x	an object of class ltraj

Value

a vector with one estimate per burst of the object of class ltraj.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

simm.brown

Examples

```
toto <- simm.brown(1:200, h=4)
hbrown(toto)

toto <- simm.brown(1:200, h=20)
hbrown(toto)
```

hist.kasc *Histograms of Mapped Variables*

Description

`hist.kasc` performs histograms of the variables mapped in objects of class `kasc`.

Usage

```
## S3 method for class 'kasc'
hist(x, type = c("h", "l", "b"), adjust = 1, col,
          border, lwd = 1, ...)
```

Arguments

`x`	a raster map of class `kasc`
`type`	what type of plot should be drawn. Possible types are: * `"h"` for histograms, * `"l"` for kernel density estimates (see `?density`). * `"b"` for both histograms and kernel density estimates (see `?density`). By default, `type = "h"` is used. If `type = "l"` is used, the position of the mean of each distribution is indicated by dotted lines
`adjust`	if `type = "l"`, a parameter used to control the bandwidth of the density estimate (see `?density`)
`col`	color for the histogram
`border`	color for the border of the histogram
`lwd`	if `type = "l"`, line width for the density estimate
`...`	further arguments passed to or from other methods

Author(s)

Mathieu Basille <`basille@ase-research.org`>

See Also

`kasc`

Examples

```
## Example with factors and numeric variables
data(puechabon)
hist(puechabon$kasc, type = "h")

## Aspect is a factor, then it's not possible to use
## kernel density estimates for it :
```

```
hist(puechabon$kasc)

## Removing the factor Aspect, and smoothing gives :
hist(puechabon$kasc[-2], type = "l")
```

---

hist.ltraj *Histogram of the Descriptive Parameters of a Trajectory*

Description

This function draws an histogram of any tranformation of the descriptive parameters of a trajectory in objects of class `ltraj`.

Usage

```
## S3 method for class 'ltraj'
hist(x, which = "dx/sqrt(dt)", ...)
```

Arguments

`x`	an object of class `ltraj`
`which`	a character string giving any syntactically correct R expression implying the descriptive elements in `x`.
`...`	parameters to be passed to the generic function `hist`.

Value

a list of objects of class `"histogram"`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`hist`, `ltraj` for additional information on the descriptive parameters of the trajectory, `qqnorm.ltraj` for examination of distribution.

Examples

```
## Simulation of a Brownian Motion
a <- simm.brown(c(1:300, seq(301,6000,by=20)))
plot(a, addpoints = FALSE)


## dx/sqrt(dt) and dy/sqrt(dt) are normally distributed (see
## ?qqchi)
hist(a, "dx/sqrt(dt)", freq = FALSE)
lines(tutu <- seq(-5,5, length=50), dnorm(tutu), col="red")

hist(a, "dy/sqrt(dt)", freq = FALSE)
lines(tutu, dnorm(tutu), col="red")


## Look at the distribution of distances between
## successive relocations
hist(a, "dist/sqrt(dt)", freq = FALSE)
lines(tutu <- seq(0,5, length=50), dchi(tutu), col="red")
```

---

histniche *Histograms of the Ecological Niche*

Description

histniche draws histograms of the niche-environment system: an histogram of the available resource units (environment) is drawn on the same graph as an histogram of the used resource units (i.e. the niche), for comparison.

Usage

```
histniche(x, pr, type = c("h", "l"), adjust = 1,
          Acol, Ucol, Aborder, Uborder, Alwd = 1,
          Ulwd = 1, ylim, ncla = 15, ...)
```

Arguments

x	a data frame giving the value of environmental variables (columns) in resource units (rows, e.g. the pixels of a raster map)
pr	a vector of integers with the same length as nrow(x) (giving for example the number of detections in the pixels)
type	what type of plot should be drawn. Possible types are: * "h" for histograms, * "l" for kernel density estimates (see ?density). By default, type = "h" is used. If type = "l" is used, the position of the mean of each distribution is indicated by dotted lines
adjust	if type = "l", a parameter used to control the bandwidth of the density estimate (see ?density)
Acol	color for the histograms of the available pixels
Ucol	color for the histograms of the used pixels
Aborder	if type = "h", color for the border of the histograms of the available pixels (see help(hist.default))

`Uborder`	if `type = "h"`, color for the border of the histograms of the used pixels (see `help(hist.default)`)
`Alwd`	if `type = "l"`, line width for the density estimate of the available pixels
`Ulwd`	if `type = "l"`, line width for the density estimate of the used pixels
`ylim`	the limits for the y axis
`ncla`	The number of classes of the histogram
`...`	further arguments passed to or from other methods

Author(s)

Mathieu Basille `<basille@ase-research.org>`

Examples

```
## Not run:

data(puechabon)
cp <- count.points(puechabon$locs[,c("X","Y")], puechabon$kasc)
puechabon$kasc
li <- kasc2df(puechabon$kasc)
cpi <- c(cp)[li$index]

histniche(li$tab, cpi)
histniche(li$tab, cpi, ty="l")


## End(Not run)
```

`hr.rast` *Rasterisation of Objects of Class 'area'*

Description

Converts an object of class `area` (used in many functions of the package `ade4`) to an object of class `kasc` (rasterisation).

Usage

```
hr.rast(mcp, w, border=c("include", "exclude"))
```

Arguments

`mcp`	an object of class `area`
`w`	a raster map of class `kasc` or of class `asc`
`border`	a character string indicating what happens when the center of the pixel is located exactly on the limit of the polygon (`"include"` indicates that the pixel is considered to be inside the polygon).

Value

Returns an object of class kasc.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

kasc for additional information on objects of class kasc, area for further information on the class area

Examples

```
data(puechabon)
kasc <- puechabon$kasc
locs <-  puechabon$locs

## Computes the home range of the animals
cp <- mcp(locs[,4:5], locs[,1])
area.plot(cp)

## Converts the home range to raster
cprast <- hr.rast(cp, kasc)
image(cprast)
```

---

hseal *Argos Monitoring of Hooded Seal*

Description

This data set contains the trajectory of one hooded seal.

Usage

```
data(hseal)
```

Format

The dataset hseal is an object of class ltraj. The coordinates are stored in meters (UTM - zone 21).

Source

Jonsen, I.D., Flemming, J.M. and Myers, R.A. (2005). Robust state-space modeling of animal movement data. *Ecology*, **86**, 2874–2880.

Examples

```
data(hseal)

plot(hseal)
```

---

ibex *GPS Monitoring of Four Ibex in the Belledonne Mountain*

Description

This dataset is an object of class "ltraj" (regular trajectory, relocations every 4 hours) containing the GPS relocations of four ibex during 15 days in the Belledonne mountain (French Alps).

Usage

```
data(ibex)
```

Source

Office national de la chasse et de la faune sauvage, CNERA Faune de Montagne, 95 rue Pierre Flourens, 34000 Montpellier, France.

Examples

```
data(ibex)
plot(ibex)
```

---

ibexraw *GPS Monitoring of Four Ibex in the Belledonne Mountain (irregular data)*

Description

This dataset is an object of class "ltraj" (irregular trajectory, relocations roughly every 4 hours) containing the raw GPS relocations of four ibex during 15 days in the Belledonne mountain (French Alps).

Usage

```
data(ibexraw)
```

Details

This dataset is nearly the same as the dataset `ibex`, except that the timing of relocations has not been rounded and the missing values have not been placed in the trajectory.

Source

Office national de la chasse et de la faune sauvage, CNERA Faune de Montagne, 95 rue Pierre Flourens, 34000 Montpellier, France.

Examples

```
data(ibexraw)
plot(ibexraw)
```

`image.asc` *Displays a Color Image of a Matrix of Class 'asc'*

Description

These functions display a raster matrix of class `asc`.

Usage

```
## S3 method for class 'asc'
image(x, col = gray((240:1)/256), clfac = NULL, ...)
## S3 method for class 'asc'
contour(x, ...)
## S3 method for class 'asc'
persp(x, ...)
## S3 method for class 'asc'
plot(x, ...)
```

Arguments

`x`	a matrix of class `asc`
`col`	for maps of type `"numeric"`, the colors to be used (see `help(par)`)
`clfac`	for maps of type `"factor"`, a character vector giving the names of colors for each level of the factor (see `help(colasc)`)
`...`	additional arguments to be passed to the generic function `image`, `persp`, `contour`, and `filled.contour` (but see below)

Note

The function `plot.asc` uses the function `filled.contour`. The output produced by `filled.contour` is actually a combination of two plots; one is the filled contour and one is the legend. Two separate coordinate systems are set up for these two plots, but they are only used internally - once the function has returned these coordinate systems are lost. If you want to annotate the main contour plot, for example to add points, you can specify graphics commands in the `plot.axes` argument of the function `filled.contour` (this argument is to be passed to the function `plot.asc`). An example is given below.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

image, contour, persp, filled.contour, asc

Examples

```
data(puechabon)

# Case of a continuous variable: the elevation
my.map <- getkasc(puechabon$kasc, "Elevation")
image(my.map, main = "Elevation in Puechabon")
contour(my.map, add = TRUE)

# use of the function plot.asc: adding points on the map
plot(my.map, plot.axes = {points(puechabon$locs[,c("X","Y")])},
     main = "Elevation")

# Case of a factor: the aspect
asp <- getkasc(puechabon$kasc, "Aspect")
cl <- colasc(asp, NorthEast = "blue", SouthEast = "red",
             SouthWest = "orange", NorthWest = "green")

## graphical display
image(asp, clfac = cl, main = "Aspect", xlab = "Lambert X",
      ylab = "Lambert Y")
legend(706500, 3162000, legend = levels(asp), fill = cl,
       cex = 0.7)


opar<-par(mar = c(0,0,3,0), bg = "slategray")
persp(my.map, scale = FALSE, box = FALSE, border = NA, shade = 0.75,
      col = "darkolivegreen3", expand = 2, theta = -60, phi = 30,
      main = "The topography of Puechabon")
par(opar)
```

---

image.sahrlocs *Graphical Display of the Habitat Composition of the Home Ranges of Animals Monitored Using Radio-Tracking*

Description

image.sahrlocs allows a gray-level display of the composition of home ranges (different colors are used for factors). For a given variable, the minimum gray level (default is "white") and the maximum gray level (default is "black") represents respectively the minimum and the maximum of the variable ****on the study area****.

Usage

```
## S3 method for class 'sahrlocs'
image(x, ani = names(x$hr), var = names(x$sa),
               mar = c(0, 0, 0, 0), axes = FALSE, dfidxy = NULL, colpts =
               "black", pch = 21, bg = "white", inv = FALSE, cexpts = 0.6,
               csub = 2, possub = c("bottomleft", "bottomright", "topleft",
               "topright"), ...)
```

Arguments

`x`	an object of class `sahrlocs`
`ani`	a character vector giving the names of the variables of the `"hr"` component (the animals) for which a display is wanted
`var`	a character vector giving the names of the variables of the `"sa"` component (the habitat variables) for which a display is wanted
`mar`	the graphical parameter `mar` (see `par`)
`axes`	logical. Whether the axes should be plotted
`csub`	the character size for the legend, used with `par("cex")*csub`
`possub`	a character string indicating the sub-title position (`"topleft"`, `"topright"`, `"bottomleft"`, `"bottomright"`)
`dfidxy`	an optional data frame with three columns giving the identity and the coordinates of the relocations of each animal. (if not `NULL`, the relocations of each animal are plotted in its home range, see Examples)
`colpts`	if `dfidxy` is not `NULL`, the color of the points to be used for the plot of the relocations
`pch`	if `dfidxy` is not `NULL`, the size of the points to be used for the plot of the relocations (see `par`)
`bg`	if `dfidxy` is not `NULL`, the background color to be used for the plot of the relocations (see `par`)
`inv`	by default, lower values of the mapped variables are brighter. If `FALSE`, the lower values are darker
`cexpts`	if `dfidxy` is not `NULL`, the size of the points
`...`	additionnal parameters to be passed to the generic function `image`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`as.sahrlocs` for additionnal information on objects of class `sahrlocs`

Examples

```
data(puechabon)
(sahr <- puechabon$sahr)

## Displays all the variables for a given animal
image(sahr, ani = "Chou")

## Displays all the animals for a given variable
image(sahr, var = "Elevation")

## Load and displays the relocations of the animals
locs <-  puechabon$locs[,c(1,4:5)]
image(sahr, var = "Elevation", dfidxy = locs, pch = 21)
```

---

import.asc *Arcview ASCII Raster File Importation And Exportation*

Description

`import.asc` imports ESRI ArcInfo ASCII raster file ; conversely, `export.asc` allows to export an asc object to a ESRI ArcInfo ASCII raster file.

Usage

```
import.asc(file, type = c("numeric", "factor"), lev = NULL,
           levnb = 1, labnb = 3)
export.asc(x, file)
as.asc(x, xll = 1, yll = 1, cellsize = 1, type = c("numeric", "factor"),
       lev = levels(factor(x)))
## S3 method for class 'asc'
print(x, ...)
```

Arguments

`file`	a character string. The name of an Arcview ASCII Raster file
`type`	a character string. Either `"numeric"` or `"factor"`
`lev`	if `type = "factor"`, either a vector giving the labels of the factor levels, or the name of a file giving the correspondence table of the map (see details)
`levnb`	if `lev` is the name of a file containing a correspondence table exported from Arcview, the column number in this table where the factor levels are stored (i.e. the numbers indicating the levels of the factor)
`labnb`	if `lev` is the name of a file containing a correspondence table exported from Arcview, the column number in this table where the factor labels are stored (i.e. the character strings indicating the labels associated with each level of the factor)
`x`	a matrix of class `asc`. For the function `as.asc`, a matrix
`xll`	the x coordinate of the center of the lower left pixel of the map

`yll`	the y coordinate of the center of the lower left pixel of the map
`cellsize`	the size of a pixel on the studied map
`...`	additionnal arguments to be passed to the function `print`

Details

With Arcview 3.x: ASCII raster files are created using the command `"File -> Export data source"`
With Arcview 8/9: ASCII raster files are created by the command `"Arc Toolbox -> Conversion tools -> From raster -`
`ASCII"`.
With GRASS, the best way to import a raster map within R is to use the package `spgrass6`, and then to use the function `spixdf2kasc` to convert the files to format that can be managed by adehabitat.
ASCII raster files may also be created using other free programs, such as landserf (`http://www.soi.city.ac.uk/~jwo/landserf/`).

Raster maps are stored as matrices of class `asc` with adehabitat. They may be of type `"numeric"` (e.g. elevation on an area) or `"factor"` (e.g. the type of vegetation on an area). If the map is of type `factor`, the levels should be indicated. The `".asc"` files store the values of the mapped variable with numeric values. Each level of the factor is coded on the map by a number. The argument `lev` of `import.asc` or `as.asc` gives the labels corresponding to each number. Alternatively, these levels may be specified using a correspondence table exported from Arcview (with this software, command `"Theme -> table"`, then `"File -> Export"`, and finally export in delimited text format). An example of such file is provided in the directory "ascfiles" of the package, see the examples below. `export.asc` allows only exportation of numeric maps.

Value

Returns a raster matrix of the class `asc`, with the following attributes :

`xll`	the x coordinate of the center of the lower left pixel of the map
`yll`	the y coordinate of the center of the lower left pixel of the map
`cellsize`	the size of a pixel on the studied map
`type`	either `"numeric"` or `"factor"`.
`levels`	if `type = "factor"`, the levels of the factor.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Arcview: `http://www.esri.com`
Landserf: `http://www.soi.city.ac.uk/~jwo/landserf/`

See Also

`image.asc`

Examples

```
## Not run:
## Importation of asc files: numeric
## Path of the file to be imported
(file1 <-  paste(system.file(package = "adehabitat"),
               "ascfiles/elevation.asc", sep = "/"))
el <- import.asc(file1)
image(el)
el


## Importation of asc files: factor
(file2 <- paste(system.file(package = "adehabitat"),
              "ascfiles/aspect.asc", sep = "/"))
(levelfile <- paste(system.file(package = "adehabitat"),
              "ascfiles/aspect.txt", sep = "/"))
asp <- import.asc(file2, lev = levelfile, type = "factor")
image(asp)
asp


## map of white noise
wafwaf <- matrix(rnorm(10000), 100, 100)
wafwaf <- as.asc(wafwaf)
image(wafwaf)

## exportation of a map
export.asc(wafwaf, "foo.asc")

## remove the created file:
file.remove("foo.asc")

## End(Not run)
```

---

indmove *Testing Independence in Regular Trajectory Parameters*

Description

The function `indmove` tests for the independence between successive components `c(dx, dy)` for each burst in a regular object of class `ltraj`.

The function `indmove.detail` tests for the independence between successive `dx` or `dy` for each burst in a regular object of class `ltraj`.

The function `testang.ltraj` tests for the independence between successive angles (relative or absolute) for each burst in a regular object of class `ltraj`.

The function `testdist.ltraj` tests for the independence between successive distances between successive relocations for each burst in a regular object of class `ltraj`.

Usage

```
indmove(ltr, nrep = 200, conflim = seq(0.95, 0.5, length=5),
        sep = ltr[[1]]$dt[1], units = c("seconds", "minutes",
                                        "hours", "days"),
        plotit = TRUE)


testang.ltraj(x, which = c("absolute", "relative"),
              nrep = 999, alter = c("two-sided","less","greater"))


testdist.ltraj(x, nrep = 999, alter = c("two-sided","less","greater"))


indmove.detail(x, detail=c("dx","dy"), nrep=999,
               alter = c("two-sided","less","greater"))
```

Arguments

`ltr,x`	an object of class `ltraj`
`conflim`	a vector giving the limits of the confidence intervals to be plotted
`nrep`	number of simulations
`units`	a character string indicating the time units for the result
`alter`	a character string specifying the alternative hypothesis, must be one of "greater", "less" or "two-sided" (default)
`which`	a character string indicating whether the absolute or relative angles are under focus
`detail`	a character string indicating whether `"dx"` or `"dy"` should be tested for independence
`plotit`	logical. Whether the results should be plotted on a graph
`sep`	used in the case of variable time lag between relocations. Indicates the theoretical time lag between two relocations

Details

The function `indmove` randomises the order of the increments `c(dx, dy)` in a trajectory. The criteria of the test is the Mean Squared Displacement (`R^2_n`) (Root & Kareiva 1984).

The function `testang.ltraj` randomises the order of the angles in a trajectory. The criteria of the test is `f^2 = sum_(i=1)^(n-1) 2*(1 -     cos(angle[i+1] - angle[i]))`. This measure corresponds to the mean squared length of the segment joining two successive angles on the trigonometric circle (see examples for an illustration)

The function `testdist.ltraj` randomises the order of the distances between successive relocations in a trajectory. The criteria of the test is `sum_(i=1)^(n-1) (dist[i+1] -    dist[i])^2` (Neuman 1941, Neuman et al. 1941). The same criteria is used in `indmove.detail()`.

.

Note that these functions require "regular" trajectories, i.e. trajectories for which the relocations are separated by a constant time lag.

Finally, note that the functions `testang.ltraj` and `testdist.ltraj` are not affected by the presence of missing values in the bursts of relocations. The function `indmove` may be greatly affected by these missing values (they are removed prior to the test).

Value

`indmove()` returns a list with one component per burst. Each component is a list of two data frames. The data frame `Time` contains the time points at which R2n is computed for the observation (first column) and the simulations (other ones). The data frame `R2n` contains the values for the R2n (same dimensions).

`testang.ltraj()`, `testdist.ltraj` and `indmove.detail` return lists of objects of class `randtest`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>
Stephane Dray <`dray@biomserv.univ-lyon1.fr`>

References

Root, R.B. & Kareiva, P.M. (1984) The search for resources by cabbage butterflies (Pieris Rapae): Ecological consequences and adaptive significance of markovian movements in a patchy environment. *Ecology*, **65**: 147–165.

Neumann, J.V., Kent, R.H., Bellinson, H.R. & Hart, B.I. (1941) The mean square successive difference. *Annals of Mathematical Statistics*, **12**: 153–162

Neumann, J.V. (1941) Distribution of the ration of the mean square successive difference to the variance. *The Annals of Mathematical Statistics*, **12**: 367–395

See Also

`ltraj`

Examples

```
## theoretical independence between
br <- simm.brown(1:1000)
```

```
testang.ltraj(br)
testdist.ltraj(br)

## Not run:
indmove(br)

## End(Not run)

## Illustration of the statistic used for the test of the independence
## of the angles
opar <- par(mar = c(0,0,4,0))
plot(0,0, asp=1, xlim=c(-1, 1), ylim=c(-1, 1), ty="n", axes=FALSE,
main="Criteria f for the measure of independence between successive
angles at time i-1 and i")
box()
symbols(0,0,circle=1, inches=FALSE, lwd=2, add=TRUE)
abline(h=0, v=0)
x <- c( cos(pi/3), cos(pi/2 + pi/4))
y <- c( sin(pi/3), sin(pi/2 + pi/4))
arrows(c(0,0), c(0,0), x, y)
lines(x,y, lwd=2, col="red")
text(0, 0.9, expression(f^2 == 2*sum((1 - cos(alpha[i]-alpha[i-1])),
i==1, n-1)), col="red")
foo <- function(t, alpha)
{
  xa <- sapply(seq(0, alpha, length=20), function(x) t*cos(x))
  ya <- sapply(seq(0, alpha, length=20), function(x) t*sin(x))
  lines(xa, ya)
}
foo(0.3, pi/3)
foo(0.1, pi/2 + pi/4)
foo(0.11, pi/2 + pi/4)
text(0.34,0.18,expression(alpha[i]), cex=1.5)
text(0.15,0.11,expression(alpha[i-1]), cex=1.5)
par(opar)
```

`is.regular` *Regular Trajectories*

Description

`is.regular` tests whether a trajectory is regular (i.e. constant time lag between successive relocations).

Usage

```
is.regular(ltraj)
```

Arguments

`ltraj`	an object of class `ltraj`

Value

`is.regular` returns a logical value

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

See Also

`ltraj`

Examples

```
data(capreotf)
is.regular(capreotf)
plotltr(capreotf, "dt")

data(albatross)
is.regular(albatross)
plotltr(albatross, "dt")
```

`is.sd` *Handling of Trajectories of the Same Duration*

Description

`is.sd` tests whether the bursts of relocations in an object of class `ltraj` contain the same number of relocations, and cover the same duration ("sd" = "same duration").
`sd2df` gets one of the descriptive parameters of a regular "sd" trajectory (e.g. `"dt"`, `"dist"`, etc.) and returns a data frame with one relocation per row, and one burst per column.

Usage

```
is.sd(ltraj)
sd2df(ltraj, what)
```

Arguments

`ltraj`	an object of class `ltraj`
`what`	a character string indicating the descriptive parameter of the trajectory to be exported

Value

`is.sd` returns a logical value.
`sd2df` returns a data frame with one column per burst of relocations, and one row per relocation.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

set.limits for additional information about "sd" regular trajectories

Examples

```
## Takes the example from the help page of cutltraj (bear):
data(bear)

## We want to study the trajectory of the animal at the scale
## of the day. We define one trajectory per day. The trajectory should begin
## at 22H00.
## The following function returns TRUE if the date is comprised between
## 21H00 and 22H00 and FALSE otherwise (i.e. correspond to the
## relocation taken at 21H30)

foo <- function(date) {
     da <- as.POSIXlt(date, "GMT")
     ho <- da$hour + da$min/60
     return(ho>21.1&ho<21.9)
}

## We cut the trajectory into bursts after the relocation taken at 21H30:

bea1 <- cutltraj(bear, "foo(date)", nextr = TRUE)
bea1

## Remove the first and last burst:
bea2 <- bea1[-c(1,length(bea1))]

## Is the resulting object "sd" ?
is.sd(bea2)

## Converts to data frame:
df <- sd2df(bea2, "dist")

## Plots the average distance per hour
meandi <- apply(df[-nrow(df),], 1, mean, na.rm = TRUE)
sedi <- apply(df[-nrow(df),], 1, sd, na.rm = TRUE) / sqrt(ncol(df))
plot(seq(0, 23.5, length = 47),
     meandi,
     ty = "b", pch = 16, xlab = "Hours (time 0 = 22H00)",
     ylab="Average distance covered by the bear in 30 mins",
     ylim=c(0, 500))
lines(seq(0, 23.5, length = 47),
      meandi+sedi, col="grey")
lines(seq(0, 23.5, length = 47),
      meandi-sedi, col="grey")
```

join.asc — *Finds the Value of Mapped Variables at some Specified Locations (Spatial Join)*

Description

`join.asc` finds the value of a mapped variable at some specified locations.
`join.kasc` is the same function as `join.asc`, with several maps.

Usage

```
join.asc(pts, x)
join.kasc(pts, w)
```

Arguments

`x`	an object of class `asc`
`w`	an object of class `kasc`
`pts`	a data frame with two columns containing the x and y coordinates of the points to be placed on the map

Value

`join.asc` returns a vector with length equals to the number of points in `pts`.
`join.kasc` returns a data frame with a number of columns equals to the number of variables in the object of class `kasc`, and with each row corresponding to the rows of `pts`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` and `asc`

Examples

```
data(puechabon)
x <- puechabon$kasc

## for each relocation, finds the values of the variables in x
toto <- join.kasc(pts = cbind(puechabon$locs$X, puechabon$locs$Y), x)
toto[1:4,]
```

`kasc2df` *Conversion of Objects of Class kasc*

Description

An object of class `kasc` stores several maps in a data frame (one column per variable, and one row per pixel of the raster map). However, the features mapped are rarely rectangle-shaped, whereas the map are inevitably rectangles. Therefore, a lot of pixels of the maps do not contain data. The pixels of the map that do not contain data are `NA` in this data frame. `kasc2df` will "clean" the object of class `kasc` from these NAs, and will return a data frame containing only mapped values that can be used in subsequent analysis.
After these analyses, `df2kasc` may be used to convert the modified data frame to an object of class `kasc` for mapping (e.g. for maps of factorial axes, using `dudi` analyses, see `help(dudi.pca)`).

Usage

```
kasc2df(x, var = names(x))
df2kasc(df, index, x)
```

Arguments

`x`	an object of class `kasc` in `kasc2df` an object of class `kasc` or `mapattr` in `df2kasc`
`var`	a character vector. The names of the variables in the `kasc` that are to be kept in the output
`df`	a data frame resulting from a computation of the component `tab` of the list previously returned by the function `kasc2df` (see section `Value` below). This computation may be any form of analysis (Principal component analysis, modelling techniques, etc.)
`index`	an integer vector giving the position of the rows of `df` in the returned `kasc` (such an index can be computed using `kasc2df`)

Value

`kasc2df` returns a list with the following components:

`tab`	a data frame without NAs, with a number of variables equals to `length(var)`.
`index`	a vector of indices of the rows of the `kasc` kept for the analyses (that is, not `NA`).

`df2kasc` returns an object of class `kasc`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for additional information on objects of class `kasc`.

Examples

```
data(puechabon)
kasc <- puechabon$kasc

# Display the kasc object
image(kasc)

# Preparation for Principal component analysis
x <- kasc2df(kasc)
x$tab <- x$tab[, (names(x$tab) != "Aspect")]

# Principal component analysis
ana <- dudi.pca(x$tab, scannf = FALSE)

s.corcircle(ana$co)
s.label(ana$li, clab = 0)


## Map of the scores of the rows
scores <- df2kasc(ana$li, x$index, kasc)
image(scores)
```

`kasc2spixdf` *Conversion of maps from/to the package "sp"*

Description

These functions convert maps of classes available in adehabitat toward classes available in the package `sp` and conversely.

`kasc2spixdf` converts an object of class `kasc` into an object of class `SpatialPixelsDataFrame`.

`asc2spixdf` converts an object of class `asc` into an object of class `SpatialGridDataFrame`.

`spixdf2kasc` converts an object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame` into an object of class `asc` or `kasc`.

`area2spol` converts an object of class `area` into an object of class `SpatialPolygons`.

`spol2area` converts an object of class `SpatialPolygons` or `SpatialPolygonsDataFrame` into an object of class `area`.

`attpol2area` gets the data attribute of an object of class `SpatialPolygonsDataFrame` and stores it into a data frame.

`traj2spdf` converts an object of class `traj` into an object of class `SpatialPointsDataFrame`.

`traj2sldf` converts an object of class `traj` into an object of class `SpatialLinesDataFrame`.

`ltraj2spdf` converts an object of class `ltraj` into an object of class `SpatialPointsDataFrame`.

`ltraj2sldf` converts an object of class `ltraj` into an object of class `SpatialLinesDataFrame`.

`kver2spol` converts an object of class `kver` into an object of class `SpatialPolygons`.

Usage

```
kasc2spixdf(ka)
asc2spixdf(a)
spixdf2kasc(sg)
area2spol(ar)
spol2area(sr)
attpol2area(srdf)
traj2spdf(tr)
traj2sldf(tr, byid = FALSE)
ltraj2spdf(ltr)
ltraj2sldf(ltr, byid = FALSE)
kver2spol(kv)
```

Arguments

`ka`	an object of class `kasc`.
`a`	an object of class `asc`.
`sg`	an object of class `SpatialPixelsDataFrame` or `SpatialGridDataFrame`.
`ar`	an object of class `area`.
`sr`	an object of class `SpatialPolygons` or `SpatialPolygonsDataFrame`.
`srdf`	an object of class `SpatialPolygonsDataFrame`.
`tr`	an object of class `traj`.
`ltr`	an object of class `ltraj`.
`kv`	an object of class `kver`.
`byid`	logical. If `TRUE`, one objects of class `Lines` correspond to one animal. if `FALSE`, one object of class `Lines` correspond to one burst.

Details

We describe here more in detail the functions `spol2area` and `attpol2area`. Objects of class `area` do not deal with holes in the polygons, whereas the objects of class `SpatialPolygons` do. Therefore, when holes are present in the `SpatialPolygons` object passed as argument, the function `spol2area` ignore them and returns only the external contour of the polygon (though a warning is returned).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>, code of `spixdf2kasc` kindly provided by Roger Bivand <`Roger.Bivand@nhh.no`>

See Also

`asc` for information on objects of class `asc`, `kasc` for info on objects of class `kasc`, `area` for info on objects of class `area`, `traj` for objects of class `traj`, `ltraj` for objects of class `ltraj`.

Examples

```
## Not run:
if (require(sp)) {

#########################################
##
## Conversion kasc -> SpatialPixelsDataFrame
##

data(puechabon)
toto <- kasc2spixdf(puechabon$kasc)
image(toto)
summary(toto)

#### and conversely
toto <- spixdf2kasc(toto)
image(toto)
hist(toto)

data(meuse.grid)
m <- SpatialPixelsDataFrame(points = meuse.grid[c("x", "y")],
                           data = meuse.grid)
i <- spixdf2kasc(m)
image(i)


### conversion asc -> SpatialPixelsDataFrame
cuicui <- asc2spixdf(getkasc(toto,1))
image(cuicui)


#########################################
##
## Conversion kver -> SpatialPolygons
##

hr <- getverticeshr(kernelUD(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
                    grid=100))

plot(hr)
class(hr)
spo <- kver2spol(hr)
plot(spo)
class(spo)

#########################################
##
## Conversion area -> SpatialPolygons
##

data(elec88)
ar <- as.area(elec88$area)
plot(ar)
toto <- area2spol(ar)
plot(toto)


#########################################
##
## Conversion SpatialPolygons -> area
##

## First create an object of class "SpatialRingsDataFrame"
tutu <- SpatialPolygonsDataFrame(toto, elec88$tab)

## and then conversion:
coincoin <- spol2area(tutu)
plot(coincoin)

## gets the attributes
haha <- attpol2area(tutu)
area.plot(coincoin, values = haha$Waechter)


#########################################
##
## Conversion ltraj -> SpatialPointsDataFrame
##

data(puechcirc)
plot(puechcirc)

toto <- ltraj2spdf(puechcirc)
plot(toto)



#########################################
##
## Conversion ltraj -> SpatialLinesDataFrame
##

toto <- ltraj2sldf(puechcirc)
plot(toto)


}

## End(Not run)
```

kernelbb *Estimation of Kernel Brownian Bridge Home-Range*

Description

kernelbb is used to estimate the utilization distribution of an animal using the brownian bridge approach of the kernel method (for autocorrelated relocations; Bullard 1991, Horne et al. 2007). liker can be used to find the maximum likelihood estimation of the parameter sig1, using the approach defined in Horne et al. 2007 (see Details).

Usage

```
kernelbb(tr, sig1, sig2, grid = 40, same4all = FALSE, byburst = FALSE,
         extent = 0.5, nalpha = 25)

liker(tr, rangesig1, sig2, le = 1000,
      byburst = FALSE, plotit = TRUE)

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

Arguments

tr	an object of class ltraj of type II (time recorded), regular or not (see help(as.ltraj)).
sig1	first smoothing parameter for the brownian bridge method (related to the speed of the animals; it can be estimated by the function liker).
sig2	second smoothing parameter for the brownian bridge method (related to the imprecision of the relocations, supposed known).
grid	a number giving the size of the grid on which the UD should be estimated. Alternatively, this parameter may be an object of class asc, or a list of objects of class asc, with named elements corresponding to each level of the factor id
same4all	logical. If TRUE, the same grid is used for all animals. If FALSE, one grid per animal is used
byburst	logical. Whether the brownian bridge estimation should be done by burst.
extent	a value indicating the extent of the grid used for the estimation (the extent of the grid on the abscissa is equal to (min(xy[,1]) + extent * diff(range(xy[,1])))).
nalpha	a parameter used internally to compute the integral of the Brownian bridge. The integral is computed by cutting each step built by two relocations into nalpha sub-intervals.
rangesig1	the range of possible values of sig1 within which the likelihood should be maximized.
le	The number of values of sig1 tested within the specified range.
plotit	logical. Whether the results of the function should be plotted.
x	an object of class khr returned by kernelbb.
...	additionnal parameters to be passed to the generic functions print

Details

The function `kernelbb` uses the brownian bridge approach to estimate the Utilization Distribution of an animal with serial autocorrelation of the relocations (Bullard 1991, Horne et al. 2007). Instead of simply smoothing the relocation pattern (which is the case for the function `kernelUD`), it takes into account the fact that between two successive relocations r1 and r2, the animal has moved through a continuous path, which is not necessarily linear. A brownian bridge estimates the density of probability that this path passed through any point of the study area, given that the animal was located at the point r1 at time t1 and at the point r2 at time t2, with a certain amount of inaccuracy (controled by the parameter sig2, see Examples). Brownian bridges are placed over the different sections of the trajectory, and these functions are then summed over the area. The brownian bridge approach therefore smoothes a trajectory.

The brownian bridge estimation relies on two smoothing parameters, `sig1` and `sig2`. The parameter `sig1` is related to the speed of the animal, and describes how far from the line joining two successive relocations the animal can go during one time unit (here the time is measured in second). The function `liker` can be used to estimate this value using the maximum likelihood approach described in Horne et al. (2007). The larger this parameter is, and the more wiggly the trajectory is likely to be. The parameter `sig2` is equivalent to the parameter `h` of the classical kernel method: it is related to the inaccuracy of the relocations, and is supposed known (See examples for an illustration of the smoothing parameters).

The functions `getvolumeUD` and `getverticeshr` can then be used to conpute the home ranges (see `kernelbb`). More generally, more details on the generic parameters of `kernelbb` can be found on the help page of `kernelUD`.

Value

An object of class `khr`, subclass `kbbhrud`. This list has one component per animal (or per burst, depending on the value of the parameter `perburst`). Each component is itself a list, with the following sub-components:

`UD`	an object of class `asc`, with the values of density probability in each cell of the grid
`h`	a vector containing the values of sig1 and sig2.
`locs`	The relocations used in the estimation procedure.
`hmeth`	a character string "bb" (not used)

`liker` returns an object of class `liker`, with one component per animal (or per burst, depending on the value of the parameter `perburst`), containing the value of (i) optimized sig1, (ii) sig2, and (iii) a data frame named "cv" with the tested values of sig1 and the corresponding log-likelihood.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Bullard, F. (1991) *Estimating the home range of an animal: a Brownian bridge approach.* Master of Science, University of North Carolina, Chapel Hill.

Horne, J.S., Garton, E.O., Krone, S.M. and Lewis, J.S. (2007) Analyzing animal movements using brownian bridge. *Ecology*, **in press**.

See Also

`as.ltraj` for further information concerning objects of class `ltraj`. `kernelUD` for the classical kernel estimation. `asc` for additionnal informations on objects of class `asc`, `mcp` for estimation of home ranges using the minimum convex polygon, and for help on the function `plot.hrsize`. `kver` for information on objects of class `kver`

Examples

```
## Not run:

#########################################################
#########################################################
#########################################################
###
###        Example of a typical case study
###       with the brownian bridge approach
###

## Load the data
data(puechcirc)
x <- puechcirc[1]

## Field studies have shown that the mean standard deviation (relocations
## as a sample of the actual position of the animal) is equal to 58
## meters on these data (Maillard, 1996, p. 63). Therefore
sig2 <- 58

## Find the maximum likelihood estimation of the parameter sig1
## First, try to find it between 10 and 100 meters
liker(x, sig2 = 58, rangesig1 = c(10, 100))

## Wow! we expected a too large standard deviation! Try again between
## 1 and 10 meters:
liker(x, sig2 = 58, rangesig1 = c(1, 10))

## So that sig1 = 6.23

## Now, estimate the brownian bridge
tata <- kernelbb(x, sig1 = 6.23, sig2 = 58, grid = 50)
image(tata, addpoints=FALSE)

## OK, now look at the home range
image(tata[[1]]$UD)
```

```
contour(getvolumeUD(tata)[[1]]$UD, level=95,
        add=TRUE, col="red", lwd=2)





#########################################################
#########################################################
#########################################################
###
###      Comparison of the brownian bridge approach
###            with the classical approach
###


## Take an illustrative example: we simulate a trajectory
set.seed(2098)
pts1 <- data.frame(x = rnorm(25, mean = 4.5, sd = 0.05),
                   y = rnorm(25, mean = 4.5, sd = 0.05))
pts1b <- data.frame(x = rnorm(25, mean = 4.5, sd = 0.05),
                    y = rnorm(25, mean = 4.5, sd = 0.05))
pts2 <- data.frame(x = rnorm(25, mean = 4, sd = 0.05),
                   y = rnorm(25, mean = 4, sd = 0.05))
pts3 <- data.frame(x = rnorm(25, mean = 5, sd = 0.05),
                   y = rnorm(25, mean = 4, sd = 0.05))
pts3b <- data.frame(x = rnorm(25, mean = 5, sd = 0.05),
                    y = rnorm(25, mean = 4, sd = 0.05))
pts2b <- data.frame(x = rnorm(25, mean = 4, sd = 0.05),
                    y = rnorm(25, mean = 4, sd = 0.05))
pts <- do.call("rbind", lapply(1:25, function(i) {
          rbind(pts1[i,], pts1b[i,], pts2[i,], pts3[i,],
                pts3b[i,], pts2b[i,])
}))
dat <- 1:150
class(dat) <- c("POSIXct","POSIXt")
x <- as.ltraj(pts, date=dat, id = rep("A", 150))

## See the trajectory:
plot(x)


## Now, we suppose that there is a precision of 0.05
## on the relocations
sig2 <- 0.05
## and that sig1=0.1
sig1 <- 0.1

## Now fits the brownian bridge home range
(kbb <- kernelbb(x, sig1 = sig1,
                 sig2 = sig2))
```

```
## Now fits the classical kernel home range
(kud <- kernelUD(pts))


###### The results

opar <- par(mfrow=c(2,2), mar=c(0.1,0.1,2,0.1))
plot(pts, pch=16, axes=FALSE, main="The relocation pattern", asp=1)
box()
plot(x, axes=FALSE, main="The trajectory")
box()
image(kud[[1]]$UD, axes=FALSE, main="Classical kernel home range")
contour(getvolumeUD(kud)[[1]]$UD, lev=95, col="red", add=TRUE)
box()
image(kbb[[1]]$UD, axes=FALSE, main="Brownian bridge home range")
contour(getvolumeUD(kbb)[[1]]$UD, lev=95, col="red", add=TRUE)
box()
par(opar)



###############################################
###############################################
###############################################
###
###         Image of a brownian bridge.
###          Fit with two relocations
###


xx <- c(0,1)
yy <- c(0,1)
date <- c(0,1)
class(date) <- c("POSIXt", "POSIXct")
tr <- as.ltraj(data.frame(x = xx,y = yy), date, id="a")

## Use of different smoothing parameters
sig1 <- c(0.05, 0.1, 0.2, 0.4, 0.6)
sig2 <- c(0.05, 0.1, 0.2, 0.5, 0.7)

y <- list()
for (i in 1:5) {
  for (j in 1:5) {
     k <- paste("s1=", sig1[i], ", s2=", sig2[j], sep = "")
     y[[k]]<-kernelbb(tr, sig1[i], sig2[j])[[1]]$UD
   }
 }

## Displays the results
opar <- par(mar = c(0,0,2,0), mfrow = c(5,5))
foo <- function(x)
   {
     image(y[[x]], main = names(y)[x], axes = F)
     points(tr[[1]][,c("x","y")], pch = 16, col = "red")
   }
lapply(1:length(y), foo)

par(opar)



## End(Not run)
```

`kernelkc` *Kernel Smoothing in Space and Time of the Animals' Use of Space*

Description

These functions estimate the utilization distribution (UD) in space and time of animals monitored using radio-telemetry, using the product kernel estimator advocated by Keating and Cherry (2009).

Note that this approach has also been useful for the analysis of recoveries in programs involving ringed birds (Calenge et al. in prep., see section examples below).

`kernelkc` estimate the UD of several animals from an object of class `ltraj`.
`kernelkcbase` estimate one UD from a data frame with three columns indicating the spatial coordinates and associated timing.
`exwc` allows to search for the best value of the time smoothing parameter in the case where the time is considered as a circular variable (see details).
`getvolumeUDk` and `getvolumeUDs` provide utilities for home-range estimation.
`getverticeshrk` and `getverticeshrs` store the home range contours.

Usage

```
kernelkc(tr, h, tcalc, t0, grid = 40, circular = FALSE,
         cycle = 24 * 3600, same4all = FALSE,
         byburst = FALSE, extent = 0.5)

kernelkcbase(xyt, h, tcalc, t0, grid=40, circular=FALSE,
             cycle=24*3600, extent=0.5)

getvolumeUDk(kc)
getvolumeUDs(asc)

getverticeshrk(kc, lev = 95)
getverticeshrs(ascv, lev = 95)

exwc(hv)
```

Arguments

`tr`	an object of class `ltraj`
`xyt`	a data frame with three columns indicating the x and y coordinates, as well as the timing of the relocations.
`kc`	an object of class `kernelkco` returned by the function `kernelkc`
`asc`	an object of class `asc` returned by the function `kernelkcbase`
`ascv`	an object of class `asc` returned by the function `getvolumeUDs`
`lev`	the percentage level for home range contour estimation.
`h`	a numeric vector with three elements indicating the value of the smoothing parameters: the first and second elements are the smoothing parameters of the X and Y coordinates respectively, the third element is the smoothing parameter for the time dimension. If `circular=TRUE` it should be a smoothing parameter in the interval 0-1 (see details). If `circular=FALSE` this smoothing parameter should be given in seconds.
`tcalc`	the time at which the UD is to be estimated
`t0`	if `circular=TRUE`, this parameter indicates the time at which the time cycle begins (see examples).
`grid`	a number giving the size of the grid on which the UD should be estimated. Alternatively, this parameter may be an object of class `asc`, or a list of objects of class `asc`, with named elements corresponding to each level of the burst/id
`circular`	logical. Indicates whether the time should be considered as a circular variable (e.g., the 31th december 2007 is considered to be one day before the 1st january 2007) or not (e.g., the 31th december 2007 is considered to be one year after the 1st january 2007).
`cycle`	if `circular=TRUE`, the duration of the time cycle. for `kernelkc`, it should be given in seconds, and for `kernelkcbase`, in the units of the data (the units of the third column of `xyt`).
`same4all`	logical. If `TRUE`, the same grid is used for all levels of id/burst. If `FALSE`, one grid per id/burst is used.
`byburst`	logical. Indicates whether one UD should be estimated by burst of `tr`, or whether the data should be pooled across all bursts of each value of id in `tr`
`extent`	a value indicating the extent of the grid used for the estimation (the extent of the grid on the abscissa is equal to `(min(xy[,1]) + extent * diff(range(xy[,1])))`.
`hv`	a value of smoothing parameter for the time dimension.
`...`	additional arguments to be passed to the function `contour`.

Details

Keating and Cherry (in press) advocated the estimation of the UD in time and space using the product kernel estimator. These functions implement exactly this methodology.

For the spatial coordinates, the implemented kernel function is the biweight kernel.

Two possible approaches are possible to manage the time in the estimation process: (i) the time may be considered as a linear variable (e.g., the 31th december 2007 is considered to be one day before the 1st january 2007), or (ii) the time may be considered as a circular variable (e.g., the 31th december 2007 is considered to be one year after the 1st january 2007).

If the time is considered as a linear variable, the kernel function used in the estimation process is the biweight kernel. If the time is considered as a circular variable, the implemented kernel is the wrapped Cauchy distribution (as in the article of Keating and Cherry). In this latter case, the smoothing parameter should be chosen in the interval 0-1, with a value of 1 corresponding to a stronger smoothing.

These functions can only be used on objects of class "ltraj", but the estimation of the UD in time and space is also possible with other types of data (see the help page of `kernelkcbase`). Note that both `kernelkc` and `kernelkcbase` return conditional probability density function (pdf), i.e. the pdf to relocate an animal at a place, given that it has been relocated at time `tcalc` (i.e. the volume under the UD estimated at time `tcalc` is equal to 1 whatever `tcalc`).

The functions `getvolumeUDk` and `getvolumeUDs` modifies the UDs, so that the contour of the UD displayed by the function `contour` corresponds to the different percentage levels of home-range estimation (see examples).

The function `exwc` draws a graph of the wrapped Cauchy distribution for the chosen `h` parameter (for circular time), so that it is possible to make one's mind concerning the weight that can be given to the neighbouring points of a given time point. Note that although Keating and Cherry (in press) advocated the use of an automatic algorithm to select "optimal" values for the smoothing parameter, it is not implemented in adehabitat. Indeed, different smoothing parameters may allow to identify patterns at different scales, and we encourage the user to try several values before subjectively choosing the value which allows to more clearly identify the patterns of the UD.

Value

`kernelkc` returns a list of class "`kernelkco`" containing objects of class `asc`, mapping one estimate of the UD per burst or id (depending on the value of the parameter `byburst`).

`getvolumekc` returns a list of class "`kernelkcov`" containing objects of class `asc`, mapping the volume under the UD for each burst or id.

`kernelkcbase` returns an object of class "asc" mapping the estimated UD. The volume under the UD can be computed using the function `getvolumeUDs`.

`getverticeshrk` stores the home range contour as objects of class `area` in a list of class `kver`, with one component per burst/id.

`getverticeshrs` stores the home range contour as an object of class `area`.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Keating, K. and Cherry, S. (in press) Modeling utilization distributions in space and time. *Ecology*, in press.

Calenge, C., Guillemain, M., Gauthier-Clerc, M. and Simon, G. in prep. A new exploratory approach to the study of the spatio-temporal distribution of ring recoveries - the example of Teal (Anas crecca) ringed in Camargue, Southern France.

See Also

`as.ltraj` for additional information on objects of class `ltraj`, `kernelUD` for the "classical" kernel home range estimates.

Examples

```
## Not run:


################################################
##
## Illustrates the analysis of recoveries of
## ringed data

data(teal)
head(teal)

## compute the sequence of dates at which the
## probability density function (pdf) of recoveries is to be estimated

vv <- seq(min(teal$date), max(teal$date), length=50)
head(vv)

## The package "maps" should be installed for the example below
library(maps)


re <- lapply(1:length(vv), function(i) {

             ## Estimate the pdf. We choose a smoothing parameter of
             ## 2 degrees of lat-long for X and Y coordinates,
             ## and of 2 months for the time
             uu <- kernelkcbase(teal, c(2.5,2.5,2*30*24*3600), tcalc =
                                vv[i], grid=100, extent=0.1)

             ## now, we show the result
             ## potentially, we could type
             ##
             ## jpeg(paste("prdefu", i, ".jpg", sep=""))
             ##
             ## to store the figures in a file, and then to build a
             ## movie with the resulting files:
             ##

             image(uu, main=vv[i])

             ## highlight the area on which there is a probability
             ## equal to 0.95 to recover a bird
             hh <- getvolumeUDs(uu)

             contour(hh, levels=95, col="red",
                     drawlabels=FALSE, add=TRUE, lwd=2)

             ## The map:
             map(xlim=c(-20,70), ylim=c(30,80), add=TRUE)

             ## and if we had typed jpeg(...) before, we have to type
             ## dev.off()
             ## to close the device. When we have finished this loop
             ## We could combine the resulting files with imagemagick
             ## (windows) or mencoder (linux)
             })




################################################
##
## Illustrates how to explore the UD in time and
## space with the bear data

data(bear)

## compute the sequence of dates at which the UD is to be
## estimated
vv <- seq(min(bear[[1]]$date), max(bear[[1]]$date), length=50)
head(vv)

## estimates the UD at each time point
re <- lapply(1:length(vv), function(i) {

             ## estimate the UD. We choose a smoothing parameter of
             ## 1000 meters for X and Y coordinates, and of 72 hours
             ## for the time (after a visual exploration)
             uu <- kernelkc(bear, h = c(1000,1000,72*3600),
                            tcalc= vv[i])

             ## now, we show the result
             ## potentially, we could type
             ##
             ## jpeg(paste("UD", i, ".jpg", sep=""))
             ##
             ## to store the figures in a file, and then to build a
             ## movie with the resulting files:
             ##
             image(uu[[1]], main=vv[i])

             ## highlight the 95 percent home range
             hh <- getvolumeUDk(uu)

             contour(hh[[1]], levels=95, col="red",
                     drawlabels=FALSE, add=TRUE)

             ## and if we had typed jpeg(...) before, we have to type
             ## dev.off()
             ## to close the device. When we have finished this loop
             ## We could combine the resulting files with imagemagick
             ## (windows) or mencoder (linux)
             })


## Or, just show the home range:
re <- lapply(1:length(vv), function(i) {

             uu <- kernelkc(bear, h = c(1000,1000,72*3600),
                            tcalc= vv[i])

             pc <- getverticeshrk(uu, lev=95)
             plot(pc, xlim=c(510000, 530000),
                  ylim=c(6810000, 6825000), main=vv[i])
             })








##################################################
##
## Example with several wild boars (linear time)

## load wild boar data
data(puechcirc)


## keep only the first two circuits:
puech <- puechcirc[1:2]


## Now load te map of the elevation
data(puechabon)
elevation <- getkasc(puechabon$kasc, 1)
```

```
## compute the time point at which the UD is to be estimated
vv <- seq(min(puechcirc[[2]]$date), max(puechcirc[[2]]$date),
          length=50)


## The estimate the UD
re <- lapply(1:length(vv),
             function(i) {

                 ## We choose a smoothing parameter of 300 meters for
                 ## the x and y coordinates and of one hour for the time
                 ## (but try to play with these smoothing parameters)

                 uu <- kernelkc(puechcirc, h=c(300,300,3600),
                                tcalc = vv[i], same4all=TRUE,
                                extent=0.1)

                 ## show the elevation
                 image(elevation, main=vv[i],
                       xlim=c(698000,704000),
                       ylim=c(3156000,3160000))

                 ## and the UD, with contour lines
                 colo <- c("red","blue")
                 lapply(1:length(uu), function(i) {
                    contour(uu[[i]], col=colo[i], add=TRUE)
                 })

                 ## the blue contour lines show the UD of the mother and
                 ## the red ones correspond to her son. Adult wild boars
                 ## are known to be more "shy" that the youger ones.
                 ## Here, the low elevation corresponds to crop area
                 ## (vineyards). The young boar is the first and the
                 ## last in the crops
             })







##################################################
##
## Example with the bear, to illustrate (circular time)

data(bear)

## We consider a time cycle of 24 hours.
## the following vector contains the time points on the
## time circle at which the UD is to be estimated (note that
## the time is given in seconds)
vv <- seq(0, 24*3600-1, length=40)


## for each time point:
re <- lapply(1:length(vv),
             function(i) {

                 ## Estimation of the UD for the bear. We choose
                 ## a smoothing parameter of 1000 meters for the spatial
                 ## coordinates and a smoothing parameter equal to 0.2
                 ## for the time. We set the beginning of the time
                 ## cycle at midnight (no particular reason, just to
                 ## illustrate the function). So we pass, as t0, any
                 ## object of class POSIXct corresponding t a date at
                 ## this hour, for example the 12/25/2012 at 00H00
                 t0 <- as.POSIXct("2012-12-25 00:00")
                 uu <- kernelkc(bear, h=c(1000,1000,0.2), cycle=24*3600,
                                tcalc=vv[i], t0=t0, circular=TRUE)

                 ## shows the results
                 ## first compute the hour for the title
                 hour <- paste(floor(vv[i]/3600), "hours",
                               floor((vv[i]%%3600)/60), "min")

                 ## compute the 95% home range
                 pc <- getverticeshrk(uu, lev=95)
                 plot(pc, xlim=c(510000, 530000),
                      ylim=c(6810000, 6825000), main=hour)

                 ## compute the 50% home range
                 pc <- getverticeshrk(uu, lev=50)
                 plot(pc, add=TRUE, colpol="blue")

             })
 ## Now, each home range computed at a given time point corresponds to
 ## the area used by the animal at this time period.  We may for example
 ## try to identify the main difference in habitat composition of the
 ## home-range between different time, to identify differences in
 ## habitat use between different time of the day. We do not do it here
 ## (lack of example data)








##################################################
```

```
##
## Example of the use of the function kernelkcbase and
## related functions


## load the data
data(puechabon)
locs <- puechabon$locs

## keeps only the wild boar Chou
locs <- locs[locs$Name=="Jean",]

## compute the number of days since the beginning
## of the monitoring
dd <- cumsum(c(0, diff(strptime(locs$Date, "%y%m%d"))))
dd

## compute xyt. Note that t is here the number of
## days since the beginning of the monitoring (it
## is not an object of class POSIXt, but it may be)
xyt <- data.frame(locs[,c("X", "Y")], dd)

## Now compute the time points at which the UD is to be estimated:
vv <- 1:61

## and finally, show the UD changed with time:
re <- lapply(1:length(vv),
             function(i) {
                 ud <- kernelkcbase(xyt, h=c(300,300,20),
                                    tcalc=vv[i], grid=100)
                 image(ud, main=vv[i])
                 contour(getvolumeUDs(ud), level=95,
                         col="red", lwd=2, add=TRUE)

                 ## Just to slow down the process
                 Sys.sleep(0.2)
                 })




## End(Not run)
```

`kerneloverlap` *Spatial Interaction between Animals Monitored Using Radio-Tracking*

Description

These functions implements all the indices of kernel home-range overlap reviewed by Fieberg and Kochanny (2005). `kerneloverlap` computes these indices from a set of relocations, whereas `kerneloverlaphr` computes these indices from an object containing the utilization distributions of the animals.

Usage

```
kerneloverlap(xy, id = NULL,
              method = c("HR", "PHR", "VI", "BA", "UDOI", "HD"),
              lev = 95, conditional = FALSE, ...)

kerneloverlaphr(x, method = c("HR", "PHR", "VI", "BA", "UDOI", "HD"),
                lev=95, conditional=FALSE)
```

Arguments

`xy`	a data frame with two columns (x and y coordinates of the animal relocations)
`id`	a factor giving the animals identity associated to `xy`
`x`	an object of class `khrud` containing the utilization distributions of several animals, and returned by the function `kernelUD`.
`method`	the desired method for the estimation of overlap (see details)
`lev`	the level of the home range estimation
`conditional`	logical. If `TRUE`, the function sets to 0 the pixels of the grid over which the UD is estimated, outside the home range of the animal estimated at a level of probability equal to `lev`. Note that this argument has no effect when `meth="HR"`.
`...`	additional arguments to be passed to the function `kernelUD` for the kernel estimation of the utilization distribution.

Details

Fieberg and Kochanny (2005) made an extensive review of the indices of overlap between utilization distributions (UD) of two animals. The function `kerneloverlap` implements these indices. The argument `method` allows to choose an index

The choice `method="HR"` computes the proportion of the home range of one animal covered by the home range of another one, i.e.:

$$HR_{i,j} = A_{i,j}/A_i$$

, where $A_{i,j}$ is the area of the intersection between the two home ranges and A_i is the area of the home range of the animal i.

The choice `method="PHR"` computes the volume under the UD of the animal j that is inside the home range of the animal i (i.e., the probability to find the animal j in the home range of i). That is:

$$PHR_{i,j} = \int \int_{A_i} UD_j(x,y)dxdy$$

where $UD_j(x,y)$ is the value of the utilization distribution of the animal j at the point x,y.

The choice `method="VI"` computes the volume of the intersection between the two UD, i.e.:

$$VI = \int_x \int_y min(UD_i(x,y), UD_j(x,y))dxdy$$

Other choices rely on the computation of the joint distribution of the two animals under the hypothesis of independence UD[i](x,y) * UD[j](x,y).

The choice `method="BA"` computes the Bhattacharyya's affinity

$$BA = \int_x \int_y \sqrt{UD_i(x,y)} \times \sqrt{UD_j(x,y)}$$

The choice `method="UDOI"` computes a measure similar to the Hurlbert index of niche overlap:

$$UDOI = A_{i,j} \int_x \int_y UD_i(x,y) \times UD_j(x,y)$$

The choice `method="HD"` computes the Hellinger's distance:

$$HD = \int_x \int_y ((\sqrt{U}D_i(x,y) - \sqrt{U}D_j(x,y))^2 dxdy)^{1/2}$$

Value

A matrix giving the value of indices of overlap for all pairs of animals.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`, based on a work of John Fieberg

References

Fieberg, J. and Kochanny, C.O. (2005) Quantifying home-range overlap: the importance of the utilization distribution. *Journal of Wildlife Management*, **69**, 1346–1359.

See Also

`kernelUD` for additional information on kernel estimation of home ranges

Examples

```
## Not run:
data(puechabon)

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="VI", conditional=TRUE)

## Identical to
kud <- kernelUD(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
                grid=200, same4all=TRUE)
kerneloverlaphr(kud, meth="VI", conditional=TRUE)



## Other indices:

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="HR")

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="PHR")

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="BA")

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="UDOI")

kerneloverlap(puechabon$locs[,c("X","Y")], puechabon$locs$Name,
              grid=200, meth="HD")

## End(Not run)
```

---

kernelUD *Estimation of Kernel Home-Range*

Description

kernelUD is used to estimate the utilization distribution (UD) of animals monitored by radio-tracking, with the classical kernel method.
getvolumeUD and kernel.area provide utilities for home-range size estimation.
getverticeshr stores the home range contour as objects of class area in a list of class kver, with one component per animal.

Usage

```
kernelUD(xy, id = NULL, h = "href", grid = 40, same4all = FALSE,
         hlim = c(0.1, 1.5), kern = c("bivnorm", "epa"), extent = 0.5)
```

```
## S3 method for class 'khr'
print(x, ...)
## S3 method for class 'khr'
image(x, axes = FALSE, mar = c(0,0,2,0),
          addcontour = TRUE, addpoints = TRUE, ...)
plotLSCV(x)
getvolumeUD(x)
kernel.area(xy, id, h = "href", grid = 40,
            same4all = FALSE, hlim = c(0.1,1.5), kern = "bivnorm",
            levels = seq(20,95, by = 5),
            unin = c("m", "km"),
            unout = c("ha", "km2", "m2"), extent = 0.5)
getverticeshr(x, lev = 95)
```

Arguments

`xy`	a data frame with two columns (x and y coordinates of the animal relocations)
`id`	an optional factor giving the animals identity associated to `xy`
`h`	a character string or a number. If `h` is set to `"href"`, the ad hoc method is used for the smoothing parameter (see details). If `h` is set to `"LSCV"`, the least-square cross validation method is used. Note that `"LSCV"` is not available if `kern = "epa"`. Alternatively, `h` may be set to any given numeric value
`grid`	a number giving the size of the grid on which the UD should be estimated. Alternatively, this parameter may be an object of class `asc`, or a list of objects of class `asc`, with named elements corresponding to each level of the factor id (see examples)
`same4all`	logical. If `TRUE`, the same grid is used for all animals. If `FALSE`, one grid per animal is used
`hlim`	a numeric vector of length two. If `h = "LSCV"`, the function minimizes the cross-validation criterion for values of h ranging from `hlim[1]*href` to `hlim[2]*href`, where `href` is the smoothing parameter computed with the ad hoc method (see below)
`kern`	a character string. If `"bivnorm"`, a bivariate normal kernel is used. If `"epa"`, an Epanechnikov kernel is used.
`extent`	a value indicating the extent of the grid used for the estimation (the extent of the grid on the abscissa is equal to `(min(xy[,1]) + extent * diff(range(xy[,1])))`).
`x`	an object of class `khr` returned by `kernelUD`.
`axes`	logical. Whether the axes are to be plotted
`mar`	the margin parameter (see `help(par)`)
`addcontour`	logical. If `TRUE`, contours are drawn on the graphics
`addpoints`	logical. If `TRUE`, the animal relocations are drawn on the graphics
`levels`	a vector of percentage levels for home-range size estimation
`unin`	the units of the relocations coordinates. Either `"m"` for meters (default) or `"km"` for kilometers

`unout`	the units of the output areas. Either `"m2"` for square meters, `"km2"` for square kilometers or `"ha"` for hectares (default)
`lev`	the percentage level for home range contour estimation.
`...`	additionnal parameters to be passed to the generic functions `print` and `image`

Details

The Utilization Distribution (UD) is the bivariate function giving the probability density that an animal is found at a point according to its geographical coordinates. Using this model, one can define the home range as the minimum area in which an animal has some specified probability of being located. The functions used here correspond to the approach described in Worton (1995).

The kernel method has been recommended by many authors for the estimation of the utilization distribution (e.g. Worton, 1989, 1995). The default method for the estimation of the smoothing parameter is the *ad hoc* method, i.e. for a bivariate normal kernel

$$h = \sigma n^{-\frac{1}{6}}$$

where

$$\sigma = 0.5(\sigma(x) + \sigma(y))$$

which supposes that the UD is bivariate normal. If an Epanechnikov kernel is used, this value is multiplied by 1.77 (Silverman, 1986, p. 86). Alternatively, the smoothing parameter h may be computed by Least Square Cross Validation (LSCV). The estimated value then minimizes the Mean Integrated Square Error (MISE), i.e. the difference in volume between the true UD and the estimated UD. Note that the cross-validation criterion cannot be minimized in some cases. According to Seaman and Powell (1998) *"This is a difficult problem that has not been worked out by statistical theoreticians, so no definitive response is available at this time"* (see Seaman and Powell, 1998 for further details and tricky solutions). `plotLSCV` allows to have a diagnostic of the success of minimization of the cross validation criterion (i.e. to know whether the minimum of the CV criterion occurs within the scanned range). Finally, the UD is then estimated over a grid.

The default kernel is the bivariate normal kernel, but the Epanechnikov kernel, which requires less computer time is also available for the estimation of the UD.

The function `getvolumeUD` modifies the UD component of the object passed as argument, so that the contour of the UD displayed by the functions `contour` and `image.khr` corresponds to the different percentage levels of home-range estimation (see examples). In addition, this function is used in the function `kernel.area`, to compute the home-range size. Note, that the function `plot.hrsize` (see the help page of this function) can be used to display the home-range size estimated at various levels.

Value

The class `khr` is a class grouping three sub-classes, `khrud`, `kbbhrud` and `khrudvol`: `kernelUD` returns a list of the class `khrud`. This list has one component per animal (named as the levels of argument `id`). Each component is itself a list, with the following sub-components:

`UD`	an object of class `asc`, with the values of density probability in each cell of the grid
`h`	if `LSCV` is not used, the value of the smoothing parameter. if `LSCV` is used, a list with three components:

- `CV` the results of the cross-validation procedure. The first column contains the sequence of values tested for the smoothing parameter, and the second column contains the value of the cross-validation criterion.
- `convergence` TRUE if the LSCV succeeds (i.e. if the optimum smoothing parameter have been found by the procedure), FALSE otherwise.
- `h` the value of the smoothing parameter used in UD estimation.

`locs`	The relocations used in the estimation procedure.
`hmeth`	The argument `h` of the function kernelUD

`getvolumeUD` returns a list of class `khrvol`, with the same components as lists of class `khrud`.

`kernel.area` returns a data frame of subclass `hrsize`, with one column per animal and one row per level of estimation of the home range.

`getverticeshr` returns an object of class `kver`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Silverman, B.W. (1986) *Density estimation for statistics and data analysis*. London: Chapman \& Hall.

Worton, B.J. (1989) Kernel methods for estimating the utilization dirstibution in home-range studies. *Ecology*, **70**, 164–168.

Worton, B.J. (1995) Using Monte Carlo simulation to evaluate kernel-based home range estimators. *Journal of Wildlife Management*, **59**,794–800.

Seaman, D.E. and Powell, R.A. (1998) *Kernel home range estimation program (kernelhr)*. Documentation of the program.

See Also

`asc` for additionnal informations on objects of class `asc`, `mcp` for estimation of home ranges using the minimum convex polygon, and for help on the function `plot.hrsize`. `kver` for information on objects of class `kver`, `kernelbb` for an alternative approach of the kernel estimation for trajectory data.

Examples

```
data(puechabon)
loc <- puechabon$locs[, c("X", "Y")]
id <- puechabon$locs[, "Name"]

## Estimation of UD for the four animals
(ud <- kernelUD(loc, id))

image(ud) ## Note that the contours
          ## corresponds to values of probability density
udvol <- getvolumeUD(ud)
image(udvol)
## Here, the contour corresponds to the
## home ranges estimated at different probability
## levels (i.e. the contour 90 corresponds to the 90 percent
## kernel home-range)
## udvol describes, for each cell of the grid,
## the smaller home-range to which it belongs

## Calculation of the 95 percent home range
ver <- getverticeshr(ud, 95)
elev <- getkasc(puechabon$kasc, "Elevation") # Map of the area
image(elev)
plot(ver, add=TRUE)
legend(696500, 3166000, legend = names(ver), fill = rainbow(4))


## Example of estimation using LSCV
udbis <- kernelUD(loc, id, h = "LSCV")
image(udbis)

## Compare the estimation with ad hoc and LSCV method
## for the smoothing parameter
(cuicui1 <- kernel.area(loc, id)) ## ad hoc
plot(cuicui1)
(cuicui2 <- kernel.area(loc, id, h = "LSCV")) ## LSCV
plot(cuicui2)

## Diagnostic of the cross-validation
plotLSCV(udbis)


## Use of the same4all argument: the same grid
## is used for all animals
udbis <- kernelUD(loc, id, same4all = TRUE)
image(udbis)

## Estimation of the UD on a map
## (e.g. for subsequent analyses on habitat selection)
elev <- getkasc(puechabon$kasc, "Elevation")
opar <- par(mfrow = c(2, 2), mar = c(0, 0, 2, 0))
```

```
cont <- getcontour(elev)

for (i in 1:length(udbis)) {
   image(elev, main = names(udbis)[i], axes = FALSE)
   points(udbis[[i]]$locs, pch = 21, bg = "white", col = "black")
}


## Measures the UD in each pixel of the map
udbis <- kernelUD(loc, id, grid = elev)
opar <- par(mfrow = c(2, 2), mar = c(0, 0, 2, 0))
for (i in 1:length(udbis)) {
  image(udbis[[i]]$UD, main = names(udbis)[i], axes = FALSE)
  box()
  polygon(cont[, 2:3])
}
par(opar)




## Estimation of the UD with a list of objects of class "asc" passed as
## argument grid (useful for large datasets)

## For example, consider the following limits:
lim <- rbind(c(697901,701061,3160198,3162604),
             c(698936,701089,3159969,3162518),
             c(698461,701928,3157362,3160427),
             c(698265,701369,3157219,3162661))

gro <- lapply(1:4, function(i) {
              subsetmap(elev, xlim = lim[i,1:2], ylim=lim[i,3:4])
})
names(gro) <- levels(id)

## show the data:
opar <- par(mfrow=c(2,2), mar=c(0.1,0.1,2,0.1))
lapply(1:4, function(i) {
  image(gro[[i]], main=names(gro)[i], axes=FALSE)
  points(loc[id==names(gro)[i],])
  box()
})
gro

## The map has been subset to fit the relocations.
## Now, estimate the UD:
ud.one.per.grid <- kernelUD(loc, id, grid = gro)
image(ud.one.per.grid)


## The UD can then be matched to habitat maps
```

kselect — *K-Select Analysis: a Method to Analyse the Habitat Selection by Animals*

Description

Performs a multivariate analysis of ecological data (K-select analysis).

Usage

```
kselect(dudi, factor, weight, scannf = TRUE, nf = 2, ewa = FALSE)
## S3 method for class 'kselect'
print(x, ...)
## S3 method for class 'kselect'
kplot(object, xax = 1, yax = 2, csub = 2, possub = c("topleft",
              "bottomleft", "bottomright", "topright"),
              addval = TRUE, cpoint = 1, csize = 1, clegend = 2, ...)
## S3 method for class 'kselect'
hist(x, xax = 1, mar=c(0.1,0.1,0.1,0.1),
             ncell=TRUE, csub=2,
             possub=c("bottomleft", "topleft",
                      "bottomright", "topright"),
             ncla=15, ...)
## S3 method for class 'kselect'
plot(x, xax = 1, yax = 2, ...)
```

Arguments

`dudi`	an object of class `dudi`
`factor`	a factor with the same length as `nrow(dudi$tab)`
`weight`	a numeric vector of integer values giving the weight associated to the rows of `dudi$tab`
`scannf`	logical. Whether the eigenvalues bar plot should be displayed
`nf`	if `scannf = FALSE`, an integer indicating the number of kept axes
`ewa`	logical. If `TRUE`, uniform weights are given to all animals in the analysis. If `FALSE`, animal weights are given by the proportion of relocations of each animal (i.e. an animal with 10 relocations has a weight 10 times lower than an animal with 100 relocations)
`x, object`	an object of class `kselect`
`xax`	the column number for the x-axis
`yax`	the column number for the y-axis
`addval`	logical. If `TRUE`, the frequency of the relocations per animal is displayed (see examples)

`cpoint`	the size of the points (if 0, the points where no relocations are found are not displayed)
`mar`	the margin parameter (see `help(par)`)
`ncell`	logical. If `TRUE`, the histogram shows the distribution of the cells of the raster map where at least one relocation is found. If `FALSE`, the histogram shows the distribution of the relocations
`csub`	the character size for the legend, used with `par("cex")*csub`
`csize`	the size coefficient for the points
`clegend`	the character size for the legend used by `par("cex")*clegend`
`possub`	a character string indicating the sub-title position (`"topleft", "topright", "bottomleft", "bottomr`
`ncla`	the number of classes of the histograms
`...`	additional arguments to be passed to the generic function `histniche`, `print` or, in the case of `plot.kselect`, `s.distri`

Details

The K-select analysis is intended for hindcasting studies of habitat selection by animals using radio-tracking data. Each habitat variable defines one dimension in the ecological space. For each animal, the difference between the vector of average available habitat conditions and the vector of average used conditions defines the marginality vector. Its size is proportional to the importance of habitat selection, and its direction indicates which variables are selected. By performing a non-centered principal component analysis of the table containing the coordinates of the marginality vectors of each animal (row) on the habitat variables (column), the K-select analysis returns a linear combination of habitat variables for which the average marginality is greatest. It is a synthesis of variables which contributes the most to the habitat selection. As with principal component analysis, the biological significance of the factorial axes is deduced from the loading of variables.

`plot.kselect` returns a summary of the analysis: it displays (i) a graph of the correlations between the principal axes of the PCA of the objects of class `dudi` passed as argument and the factorial axes of the K-select analysis; (ii) a graph giving the scores of the habitat variables on the factorial axes of the K-select analysis; (iii) the barplot of the eigenvalues of the analysis (each eigenvalue measure the mean marginality explained by the axis; (iv) the projection of the non-recentred marginality vectors on the factorial plane (the origin of the arrow indicates the average available habitat conditions, and the end of the arrow indicates the average used conditions); (v) the projection of the resource units available to each animal on the first factorial plane and (vi) the coordinates of the recentred marginality vectors (i.e. recentred so that they have a common origin) on the first factorial plane.

`kplot.kselect` returns one graph per animal showing the projections of the available resource units on the factorial plane, as well as their use by the animal. `hist.kselect` does the same thing, but on one dimension instead of two.

Value

`kselect` returns a list of the class `kselect` and `dudi` (see `dudi`).

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Calenge, C., Dufour, A.B. and Maillard, D. (2005) K-select analysis: a new method to analyse habitat selection in radio-tracking studies. *Ecological modelling*, **186**, 143–153.

See Also

sahrlocs2kselect for conversion of objects class sahrlocs to objects suitable for a K-select analysis, s.distri, and dudi for class dudi.

Examples

```
## Not run:
## Loads the data
data(puechabon)
sahr <- puechabon$sahr

## prepares the data for the kselect analysis
x <- sahrlocs2kselect(sahr)
tab <- x$tab

## Example of analysis with two variables: the slope and the elevation.
## Have a look at the use and availability of the two variables
## for the 4 animals
tab <- tab[,((names(tab) == "Slope")|(names(tab) == "Elevation"))]
tab <- scale(tab)
tmp <- split.data.frame(tab, x$factor)
wg <- split(x$weight, x$factor)
opar <- par(mfrow = n2mfrow(nlevels(x$factor)))
for (i in names(tmp))
  s.distri(scale(tmp[[i]]), wg[[i]])
par(opar)

## We call a new graphic window
x11()
## A K-select analysis
acp <- dudi.pca(tab, scannf = FALSE, nf = 2)
kn <- kselect(acp, x$factor, x$weight,
 scannf = FALSE, nf = 2)

# use of the generic function scatter
scatter(kn)

# Displays the first factorial plane
kplot(kn)
kplot(kn, cellipse = 0, cpoint = 0)
kplot(kn, addval = FALSE, cstar = 0)
```

```
# this factorial plane can be compared with
# the other graph to see the rotation proposed by
# the analysis
graphics.off()

# Displays the first factorial axis
hist(kn)

# Displays the second factorial axis
hist(kn, xax = 2)

# Summary of the analysis
plot(kn)

## End(Not run)
```

---

kver *Handling of Objects of Class kver*

Description

Objects of class `kver` are created by `getverticeshr` and `getverticesclusthr`: they contain the vertices of the home range estimated using either nearest neighbour clustering or kernel estimation. An object of class `kver` is list of data frames of class `area`.

`plot.kver` is used to display the home-range contours defined in an object of clss `kver`.

`kver.rast` is used to rasterize the home ranges.

`kver2shapefile` is used to convert the home ranges into a shapefile object (for exportation to a GIS).

Usage

```
kver.rast(kv, asc)
kver2shapefile(kv, which = names(kv))
## S3 method for class 'kver'
plot(x, which = names(x), colpol = rainbow(length(which)),
          colborder = rep("black", length(which)), lwd = 2,
          add = FALSE, ...)
```

Arguments

`kv,x`	an object of class `kver`
`asc`	a matrix of class `asc`
`which`	a vector of character indicating the polygons to be plotted

`colpol`	a vector of the color for filling the polygon. The default, `NA`, is to leave polygons unfilled
`colborder`	a vector of the color to draw the border (black, by default). Use `border = NA` to omit borders
`lwd`	the border width, a positive number
`add`	logical. If `TRUE`, the polygons are added to a previous plot
`...`	additional arguments to be passed to the function `plot.area`

Value

`kver.rast` returns an object of class `asc`.
`kver2shapefile` returns a shapefile object (can be exported using the function `write.shapefile` of the package `shapefiles`.

Warning

`kver2shapefile` requires the packages `shapefiles`.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

See Also

`getverticesclusthr` and `getverticeshr` for functions creating this class of objects. `area` for more information on objects of class `area`. `asc` for more information on objects of class `asc`. `write.shapefile` for exportation of shape files.

Examples

```
data(puechabon)
lo<-puechabon$locs[,c("X","Y")]

## Home Range Estimation
res <- clusthr(lo, puechabon$locs$Name)

## gets the vertices
vec <- getverticesclusthr(res)
plot(vec)
image(kver.rast(vec, getkasc(puechabon$kasc,1)))
```

labcon *Labelling Connected Features*

Description

This function attributes unique labels to pixels belonging to connected features on a map of class asc.

Usage

```
labcon(x)
```

Arguments

x	an object of class asc

Value

Returns a matrix of class asc, of type "factor", with a number of levels equals to the number of connected components

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

asc for further information on the class asc

Examples

```
data(puechabon)
hr <- getsahrlocs(puechabon$sahr, "hr")
u <- getkasc(hr, "Jean")
image(u)

## numbering of the connected components
p <- labcon(u)
nlevels(p)
image(p)

##  stores the first component
c1 <- p
c1[c1 != 1] <- NA
image(c1)

##  stores the second component
c2 <- p
c2[c2 != 2] <- NA
image(c2)
```

`lowres` *Reducing the Resolution of a Map*

Description

`lowres` is a generic function, having methods for the classes `asc` and `kasc`. It is used to reduce the resolution of the maps.

Usage

```
lowres(x, np = 2, ...)
```

Arguments

`x`	an object of class `asc` or `kasc`
`np`	a number giving the number of pixels to merge together (see below)
`...`	further arguments passed to or from other methods

Details

The function merges together squares of `np * np` pixels. For maps of type `"numeric"` (see `help(asc)`), the function averages the value of the variable. For maps of type `"factor"`, the function gives the most frequent level in the square of `np * np` pixels. When several levels are equally represented in the square of `np * np` pixels, the function randomly samples one of these levels.

Value

Returns an object of class `asc` or `kasc`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc` and `kasc` for further information on objects of class `asc` and `kasc` respectively.

Examples

```
data(puechabon)
kasc <- puechabon$kasc

## The initial image
image(kasc)

## The transformed image
m <- lowres(kasc, np = 4)
image(m)
```

`lynxjura` *Monitoring of Lynx*

Description

This data set stores the results of the monitoring of lynx in the French Jura between 1980 and 1999. These data have been collected by the Lynx Network of the french wildlife management office (Office national de la chasse et de la faune sauvage).

Usage

```
data(lynxjura)
```

Format

The list `lynxjura` has two components:

- `map`an object of class `kasc` (see `help(kasc)`) that describes several variables on the study area: `forets` is the density of forests, `hydro` is the density of rivers, `routes` is the density of roads and `artif` is the distance from urbanized areas.
- `locs`a data frame containing the locations of presence indices of the lynx. `X` and `Y` are the x and y coordinates, `Date` is the date of the collection of the indice and `Type` represents the type of data (`C`: alive lynx captured, `D`: attacks on livestock, `E`: prints or tracks, `F`: feces, `J`: hairs, `L`: corpse of lynx, `O`: sightings and `P`: attacks on wild prey).

Source

Vandel, J.M. (2001) *Repartition du Lynx (Lynx lynx) en France (Massif Alpin, Jurassien et Vosgien). Methodologie d'etude et statut actuel.* Ecole Pratique des Haute Etudes de Montpellier II: Dissertation.

`madifa` *The MADIFA: a Factorial Decomposition of the Mahalanobis Distances*

Description

The MADIFA allows a factorial decomposition of the Mahalanobis distances. This method is presented here in the framework of niche-environment studies.
`predict.madifa` allows the computation of the Mahalanobis Distances based on a restricted number of factorial axes.
All other functions allow various graphical displays of the results of the MADIFA.

Usage

```
madifa(dudi, pr, scannf = TRUE, nf = 2)
## S3 method for class 'madifa'
print(x, ...)
## S3 method for class 'madifa'
scatter(x, xax = 1, yax = 2, pts = FALSE, percent = 95,
               clabel = 1, side = c("top", "bottom", "none"),
               Adensity, Udensity, Aangle, Uangle, Aborder,
               Uborder, Acol, Ucol, Alty,
               Ulty, Abg, Ubg, Ainch, Uinch, ...)
## S3 method for class 'madifa'
hist(x, scores = TRUE, type = c("h", "l"), adjust = 1, Acol,
            Ucol, Aborder, Uborder, Alwd = 1, Ulwd = 1, ...)
## S3 method for class 'madifa'
predict(object, index, attr, nf, ...)
s.madifa(x, xax = 1, yax = 2, cgrid = 1, clab = 1, ...)
## S3 method for class 'madifa'
plot(x, index, attr, xax = 1, yax = 2, cont = FALSE, ...)
```

Arguments

`dudi`	a duality diagram, an object of class `dudi`
`pr`	a vector giving the utilization weights associated to each unit
`scannf`	logical. Whether the eigenvalues barplot should be displayed
`nf`	an integer indicating the number of kept factorial axes
`x,object`	an object of class `madifa`
`xax`	the column number for the x-axis
`yax`	the column number for the y-axis
`pts`	logical. Whether the points should be drawn. If `FALSE`, minimum convex polygons are displayed
`percent`	100 minus the proportion of outliers to be excluded from the computation of the minimum convex polygons
`clabel`	a character size for the columns
`side`	if `"top"`, the legend of the kept axis is upside, if `"bottom"` it is downside, if `"none"` no legend
`Adensity`	the density of shading lines, in lines per inch, for the available pixels polygon. See `polygon` for more details
`Udensity`	the density of shading lines, in lines per inch, for the used pixels polygon. See `polygon` for more details
`Aangle`	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the available pixels polygon
`Uangle`	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the used pixels polygon

`Aborder`	the color for drawing the border of the available pixels polygon (or of the bars of the histogram) . See `polygon` for more details
`Uborder`	the color for drawing the border of the used pixels polygon (or of the bars of the histogram). See `polygon` for more details
`Acol`	the color for filling the available pixels polygon. if `pts == FALSE`, the color for the points corresponding to available pixels
`Ucol`	the color for filling the used pixels polygon. if `pts == FALSE`, the color for the points corresponding to used pixels
`Alty`	the line type for the available pixels polygon, as in `par`
`Ulty`	the line type for the used pixels polygon, as in `par`
`Abg`	if `pts == TRUE`, background color for open plot symbols of available pixels
`Ubg`	if `pts == TRUE`, background color for open plot symbols of used pixels
`Ainch`	if `pts == TRUE`, heigth in inches of the available pixels
`Uinch`	if `pts == TRUE`, heigth in inches of the largest used pixels
`scores`	logical. If `TRUE`, the histograms display the row scores of the MADIFA. If `FALSE`, they display the niche on the environmental variables (in this case, this is equivalent to `histniche`)
`type`	what type of plot should be drawn. Possible types are: * `"h"` for histograms, * `"l"` for kernel density estimates (see `?density`). By default, `type = "h"` is used. If `type = "l"` is used, the position of the mean of each distribution is indicated by dotted lines
`adjust`	if `type = "l"`, a parameter used to control the bandwidth of the density estimates (see `?density`)
`Alwd`	if `type = "l"`, the line width of the kernel density estimates of the available pixels
`Ulwd`	if `type = "l"`, the line width of the kernel density estimates of the used pixels
`cgrid`	a character size, parameter used with par("cex")* `cgrid` to indicate the mesh of the grid
`clab`	if not NULL, a character size for the labels, used with `par("cex")*clab`
`index`	an integer vector giving the position of the rows of `tab` in the initial object of class `kasc`
`attr`	an object of class `kasc` or `mapattr`
`cont`	logical. Whether contour lines should be added to the maps
`...`	additional arguments to be passed to the functions `print`, `scatter`, and `plot`

Details

The Mahalanobis distances are often used in the context of niche-environment studies (Clark et al. 1993, see the function `mahasuhab`). Each environmental variable defines a dimension in a multidimensionnal space, namely the ecological space. The Mahalanobis distance between any resource unit in this space (e.g. the point defined by the values of environmental variables in a pixel of a raster map) and the centroid of the niche (the distribution of used resource units) can be used to give a value of eccentricity to this point.

For a given distribution of available resource units, for which a measure of Mahalanobis distances is desired, the MADIFA (MAhalanobis DIstances Factor Analysis) partitions the ecological space into a set of axes, so that the first axes maximises the average proportion of their squared Mahalanobis distances. Note that the sum of the squared scores of any resource unit on all the axes of the analysis is equal to the squared Mahalanobis distances for this resource unit. Thus, the MADIFA partitions the Mahalanobis distances into several axes of biological meaning (see examples). `predict.madifa` allows to compute approximate Mahalanobis distances from the axes of the MADIFA.

`plot.madifa` returns a graphical summary of the analysis: it returns graphs of (i) the eigenvalues of the analysis (each eigenvalue measures the average Mahalanobis distance explained by each factorial axis); (ii) the scores of the habitat variables (i.e. the coefficients associated to each environmental variable in the linear combination defining the axes) - note that as the ecological space is distorted to "sphericize" the niche, the factorial axes are no longer orthogonals, and the scores of the variables are distributed within an ellipsoid instead of an hypersphere of radius equal to one in classical PCA. The limits of this ellipsoid is displayed on this graph, to see the amount of distortion done by the analysis (further research needs yet to be done on this graph); (iii) The projection of the available and used points on the factorial plane of the MADIFA; (iv) The map of the Mahalanobis distances computed from the original environmental variables; (v) the map of the approximated Mahalanobis distances computed from the two axes displayed in this plot; the correlations between the original environmental variables and the factorial axes; (v) the map of the first factorial axis and (vi) the map of the second factorial axis.

`hist.madifa` returns a graph of the niche and the available resource units on the factorial axes of the analysis.

Value

`madifa` returns a list of class `madifa` containing the following components:

`call`	original call.
`tab`	a data frame with n rows and p columns.
`pr`	a vector of length n containing the number of points in each pixel of the map.
`nf`	the number of kept factorial axes.
`eig`	a vector with all the eigenvalues of the analysis.
`lw`	row weights, a vector with n components.
`li`	row coordinates, data frame with n rows and nf columns.
`l1`	row normed coordinates, data frame with n rows and nf columns.
`cw`	column weights, a vector with p components.
`co`	column coordinates, data frame with p rows and nf columns.
`mahasu`	a vector of length n containing the squared Mahalanobis distances for the n units.

`cor`	the correlation between the MADIFA axes and the original variable

`predict.madifa` returns a matrix of class `asc`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Clark, J.D., Dunn, J.E. and Smith, K.G. (1993) A multivariate model of female black bear habitat use for a geographic information system. *Journal of Wildlife Management*, **57**, 519–526.

Calenge, C., Darmon, G., Basille, M., Loison, A. and Jullien J.M. (2008) The factorial decomposition of the Mahalanobis distances in habitat selection studies. *Ecology*, **89**, 555–566.

See Also

`mahasuhab` for a detailed description of the Mahalanobis distances, `enfa` and `gnesfa` for closely related methods.

Examples

```
## Not run:

data(bauges)

map <- bauges$kasc
locs <- bauges$loc

## We prepare the data for the MADIFA
(datamad1 <- data2enfa(map, locs))

## We then perform the PCA before the MADIFA
pc <- dudi.pca(datamad1$tab, scannf = FALSE)
(mad <- madifa(pc, datamad1$pr, nf=7,
               scannf = FALSE))

##########################################
##                                      ##
## Graphical exploration of the MADIFA ##
##                                      ##
##########################################

hist(mad)

plot(mad, datamad1$index, datamad1$attr)

## this plot represents:
##  - the eigenvalues diagram
##  - the scores of the columns on the axes
##  - a graph of the niche in the available space
##  - a map of the Mahalanobis distances computed
##    using all environmental variables
##  - a map of the Mahalanobis distances computed
##    using the two factorial axes used in the
##    previous graphs
##  - the correlation between habitat variables
##    and factorial axes
##  - the geographical maps of the two
##    factorial axes

## predict with just the first axis
plot(sqrt(predict(mad, datamad1$index, datamad1$attr, 1)))



#########################################
##                                     ##
## Mathematical properties of  MADIFA  ##
##                                     ##
#########################################

## mad$li is equal to mad$l1, up to a constant (mad$l1 is normed)
plot(mad$li[,1],mad$l1[,1])

## This constant is the square root of the corresponding eigenvalue:
## the variance of mad$l1 is equal to the eigenvalue
apply(mad$l1,2,function(x) sum(x^2))/nrow(mad$li)

## the variance of mad$l1 weighted by pr is equal to 1
apply(mad$l1,2,function(x) sum(mad$pr*x^2)/sum(mad$pr))

## Therefore, the eigenvalues are equal to the average of Mahalanobis
## distance for the available resource units on each axis
mean(mahalanobis(matrix(mad$l1[,1], ncol=1), 0, 1))
mad$eig[1]

## Computation of the Mahalanobis distances
ma1 <- c(mahasuhab(map, locs))[datamad1$index]

## The sum of squared scores for a given Resource unit is equal to the
## Mahalanobis distances
ma2 <- apply(mad$l1,1, function(x) sum(x^2))
plot(ma1,ma2)


## End(Not run)
```

mahasuhab *Habitat Suitability Mapping with Mahalanobis Distances.*

Description

This function computes the habitat suitability map of an area for a species, given a set of locations of the species occurences (Clark et al. 1993). This function is to be used in habitat selection studies, when animals are not identified.

Usage

```
mahasuhab(kasc, pts, type = c("distance", "probability"))
```

Arguments

`kasc`	a raster map of class `kasc`
`pts`	a data frame with two columns, giving the coordinates of the species locations
`type`	a character string. Whether the raw `"distance"` should be returned, or rather the `"probability"` (see details).

Details

Let assume that a set of locations of the species on an area is available (gathered on transects, or during the monitoring of the population, etc.). If we assume that the probability of detecting an individual is independent from the habitat variables, then we can consider that the habitat found at these sites reflects the habitat use by the animals.

The Mahalanobis distances method has become more and more popular during the past few years to derive habitat suitability maps. The niche of a species is defined as the probability density function of presence of a species in the multidimensionnal space defined by the habitat variables. If this function can be assumed to be multivariate normal, then the mean vector of this distribution corresponds to the optimum for the species.

The function `mahasuhab` first computes this mean vector as well as the variance-covariance matrix of the niche density function, based on the value of habitat variables in the sample of locations. Then, the *squared* Mahalanobis distance from this optimum is computed for each pixel of the map. Thus, the smaller this squared distance is for a given pixel, and the better is the habitat in this pixel.

Assuming multivariate normality, squared Mahalanobis distances are approximately distributed as Chi-square with n-1 degrees of freedom, where n equals the number of habitat characteristics. If the argument `type = "probability"`, maps of these p-values are returned by the function. As such these are the probabilities of a larger squared Mahalanobis distance than that observed when x is sampled from the niche.

Value

Returns a raster map of class `asc`.

Note

The computation of the squared Mahalanobis distances inverts the variance-covariance matrix of the niche density function (see ?mahalanobis). It is therefore important that the habitat variables considered are not too correlated among each other. When the habitat variables are too correlated, the variance-covariance matrix is singular and cannot be inverted.

Note also that it is recommended to scale the variables before the computation, so that they all have the same variance, and therefore the same weight in the analysis (see examples below).

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Clark, J.D., Dunn, J.E. and Smith, K.G. (1993) A multivariate model of female black bear habitat use for a geographic information system. *Journal of Wildlife Management*, **57**, 519–526.

See Also

asc for further information on objects of class asc, kasc for additional information on objects of class kasc, domain for another method of habitat suitability mapping, and mahalanobis for information on the computation of Mahalanobis distances.

Examples

```
## loads the data
data(lynxjura)
ka <- lynxjura$map
lo <- lynxjura$locs[,1:2]

## We first scale the maps
df <- kasc2df(ka)
pc <- dudi.pca(df$tab, scannf=FALSE)
tab <- pc$tab
ka <- df2kasc(tab, df$index, ka)

## habitat suitability mapping
hsm <- mahasuhab(ka, lo, type = "probability")
plot(hsm, main = "Habitat suitability map for the Lynx",
     plot.axes = { points(lo, pch = 16, cex=0.5)})
```

---

managNAkasc *"Cleaning" Objects of Class 'kasc'*

Description

An object of class `kasc` stores several maps in a data frame (one column per variable, and one row per pixel of the raster map). However, the features mapped are rarely rectangle-shaped, whereas the map are inevitably rectangles. Therefore, a lot of pixels of the maps do not contain data. The pixels of the map that do not contain data are `NA` in this data frame.
It often occurs that several variables are not mapped for exactly the same area (that is, some pixels are `NA` for some variables, and not for others). `managNAkasc` will set to `NA` all pixels that are not mapped for all variables.

Usage

```
managNAkasc(x)
```

Arguments

`x`	an object of class `kasc`

Value

Returns an object of class `kasc`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` for additional information on objects of class `kasc`.

`mcp` *Estimation of the Home Range Using the Minimum Convex Polygon Estimator*

Description

`mcp` computes the home range of several animals using the Minimum Convex Polygon estimator.
`mcp.area` is used for home-range size estimation.
`plot.hrsize` is used to display the home-range size estimated at various levels.

Usage

```
mcp(xy, id, percent = 95)
mcp.area(xy, id, percent = seq(20,100, by = 5),
         unin = c("m", "km"),
         unout = c("ha", "km2", "m2"), plotit = TRUE)
## S3 method for class 'hrsize'
plot(x, ...)
```

Arguments

`xy`	a data frame with two columns containing the coordinates of the relocations of the monitored animals
`id`	a factor giving the identity of the animal for each relocation
`percent`	100 minus the proportion of outliers to be excluded from the computation
`unin`	the units of the relocations coordinates. Either `"m"` (default) for meters or `"km"` for kilometers
`unout`	the units of the output areas. Either `"m2"` for square meters, `"km2"` for square kilometers or `"ha"` for hectares (default)
`plotit`	logical. Whether the plot should be drawn.
`x`	an objet of class `hrsize` returned by the function `mcp.area`, or `kernel.area` (see `kernelUD()`)
`...`	additionnal arguments to be passed to the function `plot`

Details

This function computes the Minimum Convex Polygon estimation after the removal of `(100 minus percent)` percent of the relocations the farthest away from the barycenter of the home range (computed by the arithmetic mean of the coordinates of the relocations for each animal).

Value

`mcp` returns an object of class `area`, with one polygon per level of the factor ID.
`mcp.area` returns a data frame of class `hrsize`, with one column per animal and one row per level of estimation of the home range.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Mohr, C.O. (1947) Table of equivalent populations of north american small mammals. *The American Midland Naturalist*, **37**, 223-249.

See Also

`chull`, `plot.area` to have a graphical display of the home ranges, `area` for additionnal information on the class `area`, and `area2dxf` for further exportation toward a GIS. `s.chull` for another way to display MCP.

Examples

```
data(puechabon)
locs <-  puechabon$locs

cp <- mcp(locs[,4:5], locs[,1])

## Plot the home ranges
opar <- par(mar = c(0,0,0,0))
area.plot(cp)

## ... And the relocations
points(locs[,4:5], pch = 16, col = as.numeric(locs[,1]))
par(opar)

## Computation of the home-range size:
cuicui1 <- mcp.area(locs[,4:5], locs[,1])
plot(cuicui1)
```

`mcp.rast` *Converts a Polygon to Raster*

Description

`mcp.rast` converts a polygon map to a raster map of class `asc`.

Usage

```
mcp.rast(poly, w, border=c("include", "exclude"))
```

Arguments

`poly`	a data frame with 2 columns giving the coordinates of a polygon object
`w`	an object of class `kasc`, or of class `asc`
`border`	a character string indicating what happens when the center of the pixel is located exactly on the limit of the polygon (`"include"` indicates that the pixel is considered to be inside the polygon).

Details

The raster map is needed to pass the format for the output raster object to the function.

Value

Returns an object of class `asc`.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

See Also

hr.rast

Examples

```
data(puechabon)
toto <- puechabon$kasc
loc <- puechabon$locs

## gets the coordinates of the relocations for the wild boar #1
wb1 <- loc[loc$Name == "Chou",]
wb1 <- cbind(wb1$X, wb1$Y)
nbpol <- chull(wb1)
xycoord <- wb1[nbpol,]

## rasterization of wb1
tutu <- mcp.rast(xycoord, toto)
image(tutu)

polygon(xycoord, lwd = 2)
```

`mindistkeep` *Detecting Absence of Movement in an Object of Class 'ltraj'*

Description

Objects of class `ltraj` are often created with data collected using some form of telemetry (radio-tracking, G.P.S., etc.). However, the relocations of the monitored animals are always somewhat imprecise. The function mindistkeep considers that when the distance between two successive relocations is lower than a given threshold distance, the animal actually does not move.

Usage

```
mindistkeep(x, threshold)
```

Arguments

`x`	An object of class `ltraj`
`threshold`	The minimum distance under which is is considered that the animal does not move

Value

An object of class `ltraj`

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

ltraj

Examples

```
data(puechcirc)
plot(puechcirc)

i <- mindistkeep(puechcirc, 10)
plot(i)
```

---

modpartltraj *Segmentation of a trajectory based on Markov models*

Description

These functions partition a trajectory into several segments corresponding to different behaviours of the animal.
modpartltraj is used to generate the models to which the trajectory is compared.
bestpartmod is used to compute the optimal number of segments of the partition.
partmod.ltraj is used to partition the trajectory into npart segments. plot.partltraj can be used to plot the results.

Usage

```
modpartltraj(tr, limod)
## S3 method for class 'modpartltraj'
print(x, ...)

bestpartmod(mods, Km = 30, plotit = TRUE,
            correction = TRUE, nrep = 100)

partmod.ltraj(tr, npart, mods, na.manage = c("prop.move","locf"))
## S3 method for class 'partltraj'
print(x, ...)
## S3 method for class 'partltraj'
plot(x, col, addpoints = TRUE, lwd = 2, ...)
```

Arguments

tr	an object of class ltraj containing only one trajectory (one burst of relocation)
limod	a list of syntactically correct R expression giving the models for the trajectory, implying one or several elements in tr (see details and examples)
x, mods	an object of class modpartltraj (for print.modpartltraj), partltraj (for print.partltraj and plot.partltraj) returned respectively by the function genmod.crw and partmod.ltraj

`na.manage`	a character string indicating what should be done with the missing values located between two segments. With `"locf"`, the missing values are added at the end of the first segment. With `"prop.move"`, the missing values are distributed at the end of the first and the beginning of the second segment. The proportion of missing values added at the end of the first segment correspond the relative proportion of "internal" missing values found within the segments predicted by the model used to predict the first segment.
`npart`	the number of partitions of the trajectory
`Km`	the maximum number of partitions of the trajectory
`plotit`	logical. Whether the results should be plotted.
`correction`	logical. Whether the log-likelihood should be corrected (see details).
`nrep`	logical. The number of Monte Carlo simulations used to correct the log-likelihood for each number of segments.
`col`	the colors to be used for the models
`addpoints`	logical. Whether the relocations should be added to the graph
`lwd`	the line width
`...`	additional arguments to be passed to other functions

Details

A trajectory is made of successive steps traveled by an organism in the geographical space. These steps (the line connecting two successive relocations) can be described by a certain number of descriptive parameters (relative angles between successive steps, length of the step, etc.). One aim of the trajectory analysis is to identify the structure of the trajectory, i.e. the parts of the trajectory where the steps have homogeneous properties. Indeed, an animal may have a wide variety of behaviours (feeding, traveling, escape from a predator, etc.). As a result, partitioning a trajectory occupies a central place in trajectory analysis.

These functions are to be used to partition a trajectory based on Markov models of animal movements. For example, one may suppose that a normal distribution generated the step lengths, with a different mean for each type of behaviour. These models and the value of their parameters are supposed a priori by the analyst. These functions allow, based on these a priori models, to find both the number and the limits of the segments building up the trajectory (see examples). Any model can be supposed for any parameter of the steps (the distance, relative angles, etc.), provided that the model is Markovian.

The rationale behind this algorithm is the following. First, the user should propose a set of model describing the movements of the animals, in the different segments of the trajectory. For example, the user may define two models of normal distribution for the step length, with means equal to 10 meters (i.e. a trajectory with relatively small steps) and 100 meters (i.e. a trajectory with longer step lengths). For a given step of the trajectory, it is possible to compute the probability density that the step has been generated by each model of the set. The function `modpartltraj` computes the matrix containing the probability densities associated to each step (rows), under each model of the set (columns). This matrix is of class `modpartltraj`.

Then, the user can estimate the optimal number of segments in the trajectory, given the set of a priori models, using the function `bestpartmod`, taking as argument the matrix of class `modpartltraj`. If `correction = FALSE`, this function returns the log of the probability (log-likelihood) that the trajectory is actually made of `K` segments, with each one described by one model. The resulting graph can be used to choose an optimal number of segment for the partition. Note that Gueguen (2007) noted that this algorithm tends to overestimate the number of segments in a trajectory. He proposed to correct this estimation using Monte Carlo simulations of the independence of the steps within the trajectory. At each step of the randomization process, the order of the rows of the matrix is randomized, and the curve of log-likelihood is computed for each number of segments, for the randomized trajectory. Then, the observed log-likelihood is corrected by these simulations: for a given number of segments, the corrected log-likelihood is equal to the observed log-likelihood minus the simulated log-likelihood. Because there is a large number of simulations of the independence, a distribution of corrected log-likelihoods is available for each number of segments. The "best" number of segments is the one for which the median of the distribution of corrected log-likelihood is maximum.

Finally, once the optimal number of segments `npart` has been chosen, the function `partmod.ltraj` can be used to compute the partition.

The mathematical rationale underlying these two functions is the following: given an optimal k-partition of the trajectory, if the ith step of the trajectory belongs to the segment k predicted by the model d, then either the relocation (i-1) belongs to the same segment, in which case the segment containing (i-1) is predicted by d, or the relocation (i-1) belongs to another segment, and the other (k-1) segments together constitute an optimal (k-1) partition of the trajectory 1-(i-1). These two probabilities are computed recursively by the functions from the matrix of class `partmodltraj`, observing that the probability of a 1-partition of the trajectory from 1 to i described by the model m (i.e. only one segment describing the trajectory) is simply the product of the probability densities of the steps from 1 to i under the model m. Further details can be found in Calenge et al. (in prep), and in Gueguen (2001, 2007).

Value

`partmodltraj` returns a matrix of class `partmodltraj` containing the probability densities of the steps of the trajectory (rows) for each model (columns).

`bestpartmod` returns a list with two elements: (i) the element `mk` is a vector containing the values of the log-probabilities for each number of segments (varying from 1 to `Km`), and (ii) the element `correction` contains either `"none"` or a matrix containing the corrected log-likelihood for each number of segments (rows) and each simulation of the independence (column).

`partmod.ltraj` returns a list of class `partltraj` with the following components: `ltraj` is an object of class `ltraj` containing the segmented trajectory (one burst of relocations per segment of the partition); `stats` is a list containing the following elements:

`locs`	The number ID of the relocations starting the segments (except the last one which ends the last segment)
`Mk`	The value of the cumulative log-probability for the Partition (i.e. the log-probability

	associated to a K-partition is equal to the log-probability associated to the (K-1)-partition plus the log-probability associated to the Kth segment)
`mod`	The number ID of the model chosen for each segment
`which.mod`	the name of the model chosen for each segment

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Calenge, C., Gueguen, L., Royer, M. and Dray, S. (in prep.) Partitioning the trajectory of an animal with Markov models.

Gueguen, L. (2001) Segmentation by maximal predictive partitioning according to composition biases. Pp 32–44 in: Gascuel, O. and Sagot, M.F. (Eds.), *Computational Biology*, LNCS, 2066.

Gueguen, L. (in prep.) Computing the probability of sequence segmentation under Markov models.

See Also

`ltraj`

Examples

```
## Not run:
## Example on the porpoise
data(porpoise)

## Keep the first porpoise
gus <- porpoise[1]
plot(gus)

## First test the independence of the step length
indmove(gus)
## There is a lack of independence between successive distances

## plots the distance according to the date
plotltr(gus, "dist")

## One supposes that the distance has been generated
## by normal distribution, with different means for the
## different behaviours
## The means of the normal distribution range from 0 to
## 130000. We suppose a standard deviation equal to 5000:

tested.means <- round(seq(0, 130000, length = 10), 0)
(limod <- as.list(paste("dnorm(dist, mean =",
                  tested.means,
                  ", sd = 5000)")))
```

```
## Build the probability matrix
mod <- modpartltraj(gus, limod)

## computes the corrected log-likelihood for each
## number of segments
bestpartmod(mod)

## The best number of segments is 4. Compute the partition:
(pm <- partmod.ltraj(gus, 4, mod))
plot(pm)


## Shows the partition on the distances:
plotltr(gus, "dist")

lapply(1:length(pm$ltraj), function(i) {
   lines(pm$ltraj[[i]]$date, rep(tested.means[pm$stats$mod[i]],
         nrow(pm$ltraj[[i]])),
         col=c("red","green","blue")[as.numeric(factor(pm$stats$mod))[i]],
         lwd=2)
})


## Computes the residuals of the partition
res <- unlist(lapply(1:length(pm$ltraj), function(i) {
   pm$ltraj[[i]]$dist - rep(tested.means[pm$stats$mod[i]],
         nrow(pm$ltraj[[i]]))
}))

plot(res, ty = "l")

## Test of independence of the residuals of the partition:
wawotest(res)


## End(Not run)
```

morphology — *Morphology: Erosion or Dilatation of Features on a Raster Map*

Description

morphology performs morphological operations on images of class asc.

Usage

```
morphology(x, operation = c("erode", "dilate"), nt = 5)
```

Arguments

`x`	a matrix of class `asc`
`operation`	a character string indicating the operation to be processed: either `"erode"` or `"dilate"`
`nt`	the number of times that the operation should be processed

Value

Returns a matrix of class `asc`, containing 1 when the pixel belong to one feature of the image and `NA` otherwise (see examples).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`asc` for further information on objects of class `asc`.

Examples

```
data(puechabon)
a <- getkasc(puechabon$kasc,"Elevation")

## dilatation
toto1 <- morphology(a, operation = "dilate", nt = 1)
toto2 <- morphology(a, operation = "dilate", nt = 2)
toto3 <- morphology(a, operation = "dilate", nt = 3)
toto5 <- morphology(a, operation = "dilate", nt = 5)
colo <- grey((1:5)/6)
image(toto5, col = colo[1])
image(toto3, add = TRUE, col = colo[2])
image(toto2, add = TRUE, col = colo[3])
image(toto1, add = TRUE, col = colo[4])
image(a, add = TRUE)

## erosion
colo <- grey((1:20)/21)
image(a, col = 1)
for (i in 1:19) {
  toto <- morphology(a, operation = "erode", nt = i)
  image(toto, add = TRUE, col = colo[i])
}
```

`mouflon` *GPS Monitoring of One Mouflon in the Caroux Mountain*

Description

This dataset is an object of class "ltraj" (regular trajectory, relocations every 20 minutes) containing the GPS relocations of one mouflon during two week-ends in the Caroux mountain (South of France).

Usage

```
data(mouflon)
```

Source

Office national de la chasse et de la faune sauvage, CNERA Faune de Montagne, 95 rue Pierre Flourens, 34000 Montpellier, France.

Examples

```
data(mouflon)
plot(mouflon)
```

`niche.test` *Monte-Carlo Test on Parameters of the Ecological Niche*

Description

`niche.test` tests for the significance of two parameters of the ecological niche of a species (marginality and tolerance), using Monte-Carlo methods. This is a bivariate test.

Usage

```
niche.test(kasc, points, nrep = 999, o.include = TRUE, ...)
```

Arguments

`kasc`	a raster map of class `kasc`
`points`	a data frame with two columns, giving the coordinates of the species locations
`nrep`	the number of permutations
`o.include`	logical, passed to `biv.test`. If `TRUE`, the origin is included in the plot
`...`	further arguments passed to `biv.test`

Details

niche.test tests the significance of two parameters describing the ecological niche: the marginality (squared length of the vector linking the average available habitat conditions to the average used habitat conditions in the ecological space defined by the habitat variables), and the tolerance (inertia of the niche in the ecological space, i.e. the sum over all variables of the variance of used pixels).

At each step of the randomisation procedure, the test randomly allocates the n points in the pixels of the map. The marginality and the tolerance are then recomputed on this randomised data set.

Actual values are compared to random values with the help of the function biv.test.

Value

Returns a list containing the following components:

dfxy	a data frame with the randomized values of marginality (first column) and tolerance (second column).
obs	the actual value of marginality and tolerance.

Warning

biv.test uses the function kde2d of the package MASS.

Author(s)

Mathieu Basille <basille@ase-research.org>
Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

biv.test for more details on bivariate tests. histniche for the histograms of the variables of the niche.

Examples

```
## Not run:
data(lynxjura)

## We keep only "wild" indices.
tmp=lynxjura$loc[,4]!="D"
niche=niche.test(lynxjura$map,
                 lynxjura$locs[tmp, c("X", "Y")],
                 side = "bottom")
names(niche)

## End(Not run)
```

`offsetdate` *Date Handling in an Object of Class 'ltraj'*

Description

This functions allows to set an offset value from the date in an object of class `ltraj` of type II (time recorded).

Usage

```
offsetdate(ltraj, offset, units = c("sec", "min", "hour", "day"))
```

Arguments

`ltraj`	an object of class `ltraj` of type II (time recorded)
`offset`	a numeric value indicating the offset to be deducted from the date
`units`	a character string indicating the time units for `offset`

Details

The use of offset is a convenient way to define reference dates in an object of class `ltraj`. For example, if the animal is monitored every night, from 18H00 to 06H00, the fact that the beginning and the end of the monitoring do not correspond to the same day may cause difficulties to handle the trajectory. Though these difficulties are not unsurmountable, it is often convenient to deduct an offset to the trajectory, so that the first relocation is collected at 0H and the last one at 12H00 the same day (i.e., in this example, an offset of 18 hours).

Value

an object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information on objects of class `ltraj`

Examples

```
data(puechcirc)

plotltr(puechcirc, "dt")

toto <- offsetdate(puechcirc, 17, "hour")

plotltr(puechcirc, "dt")
```

perarea — *Compute Areas and Perimeters of Objects of Class "area"*

Description

`perarea` computes the perimeters of polygons in objects of class `area`.
`ararea` computes the areas of polygons in objects of class `area`.

Usage

```
perarea(x)
ararea(x)
```

Arguments

`x`	object of class `area`

Value

a vector.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`area`, `plot.area`

Examples

```

data(puechabon)
locs <-  puechabon$locs

cp <- mcp(locs[,4:5], locs[,1])

perarea(cp)

ararea(cp)
```

pheasant *Radio-Tracking of Pheasants*

Description

This data set describes the use and availability of 5 habitat types for 13 pheasants monitored using radio-tracking.

Usage

```
data(pheasant)
```

Format

This list has three components:

`studyarea` a data frame giving the proportion of each habitat type (columns) on the study area. These habitat types are `Scrub`, `Broadleaf`, `Coniferous`, `Grassland` and `Crop`. These proportions are repeated by rows, for all animals (rows)

`mcp` a data frame giving the proportion of each habitat type (columns) in the home range of each animal (rows)

`locs` a data frame giving the proportion of relocations of each animal (rows) reported in 3 of the 5 habitat types (columns). Indeed, `Coniferous` and `Crops` were not used by most of the animals.

Source

Aebischer, N. J., Robertson, P. A. and Kenward, R. E. (1993) Compositional analysis of habitat use from animal radiotracking data. *Ecology*, **74**, 1313–1325.

plot.area *Graphical Display of Objects of Class "area"*

Description

`plot.area` allows a graphical display of objects of class `"area"`.

Usage

```
## S3 method for class 'area'
plot(x, which = levels(x[,1]),
          colpol = rep("green", nlevels(x[, 1])),
          colborder = rep("black", nlevels(x[, 1])),
          lwd = 2, add = FALSE, ...)
```

Arguments

`x`	an object of class `"area"`
`which`	a vector of character indicating the polygons to be plotted
`colpol`	a vector of the color for filling the polygon. The default, `NA`, is to leave polygons unfilled
`colborder`	a vector of the color to draw the border. The default. Use `border = NA` to omit borders
`lwd`	the border width, a **positive** number
`add`	logical. if `TRUE`, the polygons are added to a previous plot
`...`	additional arguments to be passed to the generic function `plot`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`area.plot` for another way to display objects of class `"area"`

Examples

```
## Loading the relocations of wild boars
## monitored using radio-tracking
data(puechabon)
locs <-  puechabon$locs
el <- getkasc(puechabon$kasc, "Elevation")

## Estimation of the MCP home ranges of the animals
cp <- mcp(locs[,4:5], locs[,1])

## Use of plot.area to display the results
plot(cp)

## different colors:
color <- c("red", "blue", "green", "yellow")
lev <- levels(cp[,1])
image(el)
plot(cp, colpol = color, add = TRUE)
legend(697198, 3165529, legend = lev, fill = color)

## or:
image(el)
plot(cp, colborder = color, colpol = NA, add = TRUE)
legend(697198, 3165529, legend = lev, fill = color)

## plots one animal
image(el)
plot(cp, which = "Brock", add = TRUE)
```

plot.ltraj	*Graphical Display of an Object of Class "ltraj"*

Description

plot.ltraj allows various graphical displays of the trajectories.

Usage

```
## S3 method for class 'ltraj'
plot(x, id = unique(unlist(lapply(x, attr, which = "id"))),
           burst = unlist(lapply(x, attr, which = "burst")), asc = NULL,
           area = NULL, xlim = NULL, ylim = NULL, colasc =
           gray((240:1)/256), colpol = "green",  addpoints = TRUE,
           addlines = TRUE, perani = TRUE, final = TRUE, ...)
```

Arguments

x	an object of class ltraj
id	a character vector containing the identity of the individuals of interest
burst	a character vector containing the burst levels of interest
asc	an object of class asc
area	an object of class area
xlim	the ranges to be encompassed by the x axis
ylim	the ranges to be encompassed by the y axis
colasc	a character vector giving the colors of the map of class asc
colpol	a character vector giving the colors of the polygon contour map, when area is not NULL
addpoints	logical. If TRUE, points corresponding to each relocation are drawn
addlines	logical. If TRUE, points corresponding to each relocation are drawn
perani	logical. If TRUE, one plot is drawn for each value of id, and the several bursts are superposed on the same plot for a given animal. If FALSE, one plot is drawn for each value of burst
final	logical. If TRUE, the initial and final relocations of each burst are indicated in blue and red, respectively
...	arguments to be passed to the generic function plot

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

For further information on the class ltraj, ltraj.

Examples

```
data(puechcirc)

plot(puechcirc)
plot(puechcirc, perani = FALSE)
plot(puechcirc, id = "JE93", perani = FALSE)

data(puechabon)
ele <- getkasc(puechabon$kasc, "Elevation")
plot(puechcirc, perani = FALSE, asc = ele)

cont <- getcontour(ele)
plot(puechcirc, area = cont)
```

`plot.sahrlocs` *Exploratory Analysis of Habitat Selection*

Description

`plot.sahrlocs` applies the function `widesII` or `widesIII` for each variable in the object of class `sahrlocs`, and the results are stored in a list. Then, the function `plot.wi` is applied to each component of the list. This allows to investigate habitat selection by animals at several scales for design II and III data.

Usage

```
## S3 method for class 'sahrlocs'
plot(x, ani = names(x$hr), var = names(x$sa),
              type = c("hr.in.sa", "locs.in.hr", "locs.in.sa"),
              ncla = 4, ylog = FALSE, caxis = 0.7, clab = 0.7,
              errbar = c("SE", "CI"), alpha = 0.05, draw = TRUE, ...)
```

Arguments

`x`	an object of class `sahrlocs`
`ani`	a character vector. This vector contains the names of the animals in `x` for which habitat selection should be displayed. At least two animals are required
`var`	a character vector. This vector contains the names of the variables in `x` for which habitat selection should be displayed
`type`	a character string. Type of habitat selection that should be investigated. If `"hr.in.sa"`, the selection of the home-range within the study area is displayed. If `"locs.in.sa"`, the selection of the relocations within the study area is displayed. If `"locs.in.hr"`, the selection of the relocations within the home range is displayed

`ncla`	numeric variables are converted to factors. This parameter controls the number of classes of these factors
`ylog`	logical. If `TRUE`, the selection ratios are plotted on a log scale
`caxis`	character size on axes to be passed to `par("cex.axis")`
`clab`	character size of axes labels to be passed to `par("cex.lab")`
`errbar`	a character string. Type of error bars: either `"CI"` for confidence intervals or `"SE"` for standard errors
`alpha`	the alpha-level for the tests
`draw`	logical. If `FALSE`, no plot is drawn
`...`	further arguments to be passed to the function `plot.wi`

Value

The function returns a list of objects of class `wiII` or `wiIII` (one component per animal).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Manly B.F.J., McDonald L.L., Thomas, D.L., McDonald, T.L. & Erickson, W.P. (2003) *Resource selection by animals - Statistical design and Analysis for field studies. Second edition.* London: Kluwer academic publishers.

See Also

`widesII` and `widesIII` for further information on objects of class `wiII` and `wiIII`, `as.sahrlocs` for further information on objects of class `sahrlocs`.

Examples

```
data(puechabon)
sahr <- puechabon$sahr
toto <- plot(sahr)








### Note that the wild boars named Brock and Calou
### have only one herbaceous cover class available
### (the second one), and they use it exclusively.
### So they have identical
### selection ratios (that's why the curve of Brock
### does not appear: it is hidden behind the curve
### of the boar named Calou).
```

```
toto
toto$Elevation
```

plotltr — *Changes in Traject Parameters Over Time*

Description

This function allows a graphical examination of the changes in descriptive parameters in objects of class `ltraj`

Usage

```
plotltr(x, which = "dist", ...)
```

Arguments

`x`	An object of class `ltraj`
`which`	a character string giving any syntactically correct R expression implying the descriptive elements in `x`.
`...`	additional parameters to be passed to the generic function `plot`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information about objects of class `ltraj`, and `sliwinltr` for a sliding window smoothing

Examples

```
data(puechcirc)

plotltr(puechcirc, "cos(rel.angle)")
plotltr(puechcirc, "dist")
plotltr(puechcirc, "dx")
```

porpoise *Argos monitoring of Porpoise Movements*

Description

This data set contains the relocations of 3 porpoises

Usage

```
data(porpoise)
```

Format

This data set is a regular object of class `ltraj` (i.e. constant time lag of 24H).

Details

The coordinates are given in meters (UTM - zone 19).

Source

http://whale.wheelock.edu/

Examples

```
data(porpoise)

plot(porpoise)
```

predict.enfa *Habitat Suitability Maps Built from the ENFA*

Description

`predict.enfa` computes habitat suitability maps using the Ecological-Niche Factor Analysis and the Mahalanobis distances method.

Usage

```
## S3 method for class 'enfa'
predict(object, index, attr, nf, ...)
```

Arguments

`object`	an object of class `enfa`
`index`	an integer vector giving the position of the rows of `tab` in the initial object of class `kasc`.
`attr`	an object of class `kasc` or `mapattr`.
`nf`	the number of axes of specialization kept for the predictions. By default, all axes kept in `object` are used
`...`	further arguments passed to or from other methods

Details

The predictions are based on the position of the niche defined by the ENFA within the multidimensional space of environmental variables. The ENFA produces row coordinates for each pixel, which are used with the function `mahalanobis`. For each pixel, this function computes the Mahalanobis distances from the barycentre of the niche.

Actually, the function `predict.enfa` is identical to the function `mahasuhab`, except that the habitat suitability map is computed using the axes of the ENFA, instead of the raw data.

Note that the MADIFA allows a more consistent factorial decomposition of the Mahalanobis distances.

Value

Returns a raster map of class `kasc`.

Author(s)

Mathieu Basille `<basille@ase-research.org>`

References

Clark, J.D., Dunn, J.E. and Smith, K.G. (1993) A multivariate model of female black bear habitat use for a geographic information system. *Journal of Wildlife Management*, **57**, 519–526.

Hirzel, A.H., Hausser, J., Chessel, D. & Perrin, N. (2002) Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? *Ecology*, **83**, 2027–2036.

See Also

`mahalanobis` for information on the computation of Mahalanobis distances. `mahasuhab` for more details on the computation of habitat suitability maps using the Mahalanobis distances. `madifa` for a more consistent factorial decomposition of the Mahalanobis distances

Examples

```
## Not run:
data(lynxjura)

map <- lynxjura$map

## We keep only "wild" indices.
tmp <- lynxjura$loc[,4] != "D"
locs <- lynxjura$locs[tmp, c("X","Y")]
dataenfa1 <- data2enfa(map, locs)

(enfa1 <- enfa(dudi.pca(dataenfa1$tab, scannf=FALSE),
               dataenfa1$pr, scannf = FALSE))

## Compute the prediction
pred <- predict(enfa1, dataenfa1$index, dataenfa1$attr)
image(pred)
contour(pred, col="green", add=T)
points(locs, col = "red", pch = 16)
## Lighter areas are the most preferred areas

## End(Not run)
```

`puech` *Radio-Tracking Data of Wild Boar (2)*

Description

This data set stores the results of the monitoring of 6 wild boar at Puechabon (Mediterranean habitat, South of France). These data have been collected by Daniel Maillard (Office national de la chasse et de la faune sauvage).

Usage

```
data(puech)
```

Details

The list `puech` has two components:
`puech$relocations` is a data frame containing the relocations of the wild boar resting sites in summer. It contains the coordinates of the relocations and the name of the corresponding wild boar.
`puech$map` is a list of objects of class `asc` (see `help(asc)`) that describe nine environmental variables on the study area (the elevation, the tree cover, the shrub cover, the distance to recreational trails, the distance to crops, the distance to water points, the grass cover, the slope and the sunshine). Note that both the maps and the relocations have been slightly "damaged" to preserve the copyright on the data.

References

Maillard, D. (1996). *Occupation et utilisation de la garrigue et du vignoble mediterraneens par le Sanglier*. Universite d'Aix-Marseille III: PhD thesis.

---

puechabon *Radio-Tracking Data of Wild Boar*

Description

This data set stores the results of the monitoring of 4 wild boars at Puechabon (Mediterranean habitat, South of France). These data have been collected by Daniel Maillard (Office national de la chasse et de la faune sauvage).

Usage

```
data(puechabon)
```

Details

The list `puechabon` has three components:
`puechabon$kasc` is an object of class `kasc` (see `help(kasc)`) that describes several variables on the study area.
`locs` is a data frame of the relocations of the wild boar resting sites in summer. Information on wild boars is provided by factors `Name`, `Sex`, `Age`.
`sahr` is the associated object of class `sahrlocs`. The home ranges were estimated with buffers including all pixels within 500 m of a boar relocation.

References

Maillard, D. (1996). *Occupation et utilisation de la garrigue et du vignoble mediterraneens par le Sanglier*. Universite d'Aix-Marseille III: PhD thesis.

---

puechcirc *Movements of wild boars tracked at Puechabon*

Description

This data set is an object of class `ltraj`, giving the results of the monitoring of 2 wild boars by radio-tracking at Puechabon (Mediterranean habitat, South of France). These data have been collected by Daniel Maillard (Office national de la chasse et de la faune sauvage), and correspond to the activity period of the wild boar (during the night, when the animals forage. The data set `puechabon` describes the resting sites).

Usage

```
data(puechcirc)
```

Format

This object has, in total, 204 relocations distributed among two animals and three bursts of relocations (`CH930803`, `CH930824`, `CH930827`, and `JE930827`).

Source

Maillard, D. (1996). *Occupation et utilisation de la garrigue et du vignoble mediterraneens par le Sanglier*. Universite d'Aix-Marseille III: PhD thesis.

`puechdesIII` *Habitat Selection by the Wild Boar at Puechabon*

Description

This data set contains two data frames describing the use and the availability of 3 elevation classes for 6 wild boars (*Sus scrofa* L.) monitored using radio-tracking at Puechabon (South of France). These data have been collected by Daniel Maillard (Office national de la chasse et de la faune sauvage).

Usage

```
data(puechdesIII)
```

Details

The list `puechdesIII` has two components:
The data frame `used` describes the number of telemetry relocations for each of the 6 animals in each of the three elevation classes.
The data frame `available` describes a sample of random points placed in the areas available to these wild boars (a buffer area of 200 m around the relocations).

Source

Maillard, D. (1996) *Occupation et utilisation de la garrigue et du vignoble mediterraneens par le Sanglier*. Universite d'Aix-Marseille III: PhD thesis.

qqchi *Quantile-Quantile Plots for Trajectories of Class 'ltraj'*

Description

The functions allow the examination of the distribution of trajectories descriptors (see Details).

Usage

```
## Chi distribution of the increment length / sqrt(dt)
qqchi(y, ...)

## Default S3 method:
qqchi(y, df = 2, ylim, main = "Chi Q-Q Plot",
              xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
              plot.it = TRUE, datax = FALSE, ...)

## S3 method for class 'ltraj'
qqchi(y, xlab = "Theoretical Quantiles",
            ylab = "Sample Quantiles (Distances)", ...)

## Normal Distribution of dx/sqrt(dt) or dy/sqrt(dt)
## S3 method for class 'ltraj'
qqnorm(y, which=c("dx","dy"), ...)
```

Arguments

`y`	a vector containing the data sample for `qqchi.default`. an object of class `ltraj` for other functions.
`df`	the number of degrees of freedom of the Chi distribution (default to 2).
`xlab, ylab, main`	plot labels.
`plot.it`	logical. Should the result be plotted?
`datax`	logical. Should data values be on the x-axis?
`which`	a character string indicating the component (dx or dy) to be examined.
`ylim, ...`	graphical parameters.

Details

Among the numerous statistics that can be used to describe the movements of an animal, the length of the increment between two successive relocations is very common. This increment can be described by a vector `i = c(dx, dy)`. Under the hypothesis of a Brownian motion, dx and dy should be normally distributed with mean = 0 and variance = dt (where dt is the time interval between the two relocations). Therefore, `dx/sqrt(dt)` and `dy/sqrt(dt)` should be normally distributed with mean = 0 and variance = 1. The function `qqnorm.ltraj` performs a quantile-quantile plot of `dx/sqrt(dt)` or `dy/sqrt(dt)` vs. a normal distribution to verify wether the Brownian motion assumption is correct.

Furthermore, the quantity `(dx^2 + dy^2)/dt` should be distributed according to a Chi-squared distribution with two degrees of freedom. Thus, the quantity `distance / sqrt(dt)` should be distributed according to a Chi distribution with two degrees of freedom (where `distance` is the distance between the two relocations). The function `qqchi.ltraj` performs quantile-quantile plot of `distance/sqrt(dt)` vs. a Chi distribution to verify wether the Brownian motion assumption is correct.

Value

for functions dealing with objects of class `ltraj`, a list with components being themselves lists, with components:

`x`	The x coordinates of the points that were/would be plotted
`y`	The original `y` vector, i.e., the corresponding y coordinates including 'NA's.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`chi`, `qqplot`, `ltraj`.

Examples

```
## Example with an Arithmetic Brownian Process
toto <- simm.mba(1:500, sig = diag(c(5, 5)))
qqnorm(toto, "dx")
qqnorm(toto, "dy")
qqchi(toto)

## Example of wild boar
data(puechcirc)
qqnorm(puechcirc, "dx")
qqnorm(puechcirc, "dy")
qqchi(puechcirc)
```

`rand.kselect` *Test of the Third-Order Habitat Selection*

Description

`rand.kselect` tests whether the marginality vector of animals is significantly larger than what is expected under the hypothesis of random habitat use (third-order habitat selection: selection by the animals of the relocations within their home range; the habitat availability is measured for each animal). The effect of each variable on individual marginality is also tested. Finally, the pertinence of a K-select analysis is also tested. This is a randomisation test. The alpha-level of the tests is ajusted using the Bonferroni inequality.

Usage

```
rand.kselect(dudi, factor, weight, nrep = 200, alpha = 0.05, ewa = FALSE)
## S3 method for class 'rand.kselect'
print(x, ...)
```

Arguments

`dudi`	an object of class `dudi`.
`factor`	a factor defining the animals identity
`weight`	a weight vector of integer values (number of relocations counted in each resource unit in row of the object dudi)
`nrep`	the number of repetitions of the test
`alpha`	the alpha level for the tests.
`ewa`	logical. If `TRUE`, uniform weights are given to all animals in the analysis. If `FALSE`, animal weights are given by the proportion of relocations of each animal (i.e. an animal with 10 relocations has a weight 10 times lower than an animal with 100 relocations)
`x`	an object of the class `rand.kselect`.
`...`	further arguments to be passed to the generic function `print`

Details

This test is carried out by simulating a random use of space by animals. `rand.kselect` is closely related to the function `kselect` (same arguments).

At each step of the randomisation procedure, and for each animal, the test randomly allocates the nk relocations (where nk is the sum of the weight vector for the animal k) in the Ik pixels available to this animal (where Ik is the length of the weight vector for animal k).

The length of the marginality vector is recomputed at each step of the randomisation procedure and for each animal. The effect of each variable on the use of pixels by each animal is measured by the criterion "(average habitat variable j used by animal i) minus (average habitat variable j available to animal i)". Finally the value of the first eigenvalue of the K-select analysis provides a criterion to test the pertinence of the K-select analysis.

All these values are then compared to the observed values to assess the significance of theses effects.

Value

Returns an object of class `rand.kselect`. This list has three components:

`global`	a vector of length 2 giving the results of the randomisation procedure for the first eigenvalue of the K-select analysis.
`marg`	a matrix giving the significance of the marginality of each animal.
`per.ind`	a list giving the results of the randomisation test for the coordinates of the marginality vector for each animal on each habitat variable.
`alpha`	the alpha level of the tests.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Calenge, C., Dufour, A.B. and Maillard, D. (*submitted*). K-select analysis, a new method to analyse habitat selection in radio-tracking studies.

See Also

`kselect` to perform a K-select analysis.

Examples

```
## Not run:
## Loads the data
data(puechabon)
sahr <- puechabon$sahr

## prepares the data for the kselect analysis: only two variables are kept
x <- sahrlocs2kselect(sahr)
tab <- x$tab
dud <- dudi.mix(tab, scannf = FALSE, nf = 2)

## the randomisation tests
## be patient, this can be very long on some machines
(te <- rand.kselect(dud, x$factor, x$weight, nrep = 50))


## End(Not run)
```

randtest.enfa *Randomisation Test for the Ecological Niche Factor Analysis*

Description

`randtest.enfa` performs a randomisation test for the Ecological Niche Factor analysis (ENFA).

Usage

```
## S3 method for class 'enfa'
randtest(xtest, nrepet = 999, ...)
```

Arguments

`xtest`	an object of class `enfa`
`nrepet`	the number of iterations for the randomisation test
`...`	further arguments to be passed to the generic function `randtest`

Details

This test is carried out by simulating a random distribution of the species occurrences in the pixels of a map.

At each step of the randomisation procedure, the test randomly allocates the nk occurrences (where nk is the sum of the occurrence vector `pr` of the object of class `enfa`) in the Ik pixels of the focus area (where Ik is the length of this occurrence vector).

At each step of the procedure, the first eigenvalue of the ENFA performed on the randomised data set is recomputed. This value provides a criterion to test the pertinence of the ENFA analysis.

Value

returns a list of class `randtest`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Manly, B.F.J. (1997) *Randomization, Bootstrap and Monte Carlo Methods in Biology*. London: Chapman & Hall.

Hirzel, A.H., Hausser, J., Chessel, D. and Perrin, N. (2002) Ecological-niche factor analysis: How to compute habitat suitability maps without absence data? *Ecology*, **83**, 2027–2036.

See Also

`enfa`

Examples

```
## Not run:
data(chamois)
locs <- chamois$locs
map <- chamois$map
map$Vegetation <- NULL
da <- data2enfa(map, locs)
en <- enfa(dudi.pca(da$tab, scannf=FALSE), da$pr, scannf = FALSE)
(tutu <- randtest(en, nrepet = 100))
plot(tutu)

## End(Not run)
```

`redisltraj` *Rediscretization of a Trajectory With Regular Step Length*

Description

This functions rediscretizes one or several trajectories in an object of class `ltraj`.

Usage

```
redisltraj(l, u, burst = NULL, samplex0 = FALSE, addbit = FALSE,
            nnew = 5)
```

Arguments

`l`	an object of class `ltraj`
`u`	the new step length
`burst`	The burst identity of trajectories to be rediscretized.
`samplex0`	Whether the first relocation of the trajectory should be sampled
`addbit`	logical. Whether the line segment linking the last relocation of the rediscretized trajectory and the last relocation of the raw trajectory should be added to the result (can be useful for computation of fractal dimension)
`nnew`	optionnally, you may specify the maximum ratio between number of relocations of the new trajectory. If not specified, this maximum is equal to 5 times the number of relocations of the raw trajectory.

Details

The rediscretization of trajectory has been advocated by several authors in the literature (Turchin 1998, Bovet & Benhamou 1988). It is also the first step of the computation of the fractal dimension of the path (Sugihara & May 1990).

Value

An object of class `"ltraj"`

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Bovet, P., & Benhamou, S. (1988) Spatial analysis of animal's movements using a correlated random walk model. *Journal of Theoretical Biology* **131**: 419–433.

Turchin, P. (1998) *Quantitative analysis of movement*, Sunderland, MA.

Sugihara, G., & May, R. (1990) Applications of fractals in Ecology. *Trends in Ecology and Evolution* **5**: 79–86.

See Also

`ltraj` for further information on objects of class `ltraj`

Examples

```
data(puechcirc)

puechcirc

## before rediscretization
plot(puechcirc, perani = FALSE)

## after rediscretization
toto <- redisltraj(puechcirc, 100)
plot(toto, perani = FALSE)
```

runsNAltraj *Highlighting the Patterns in Missing Values in Trajects*

Description

`runsNAltraj` performs a runs test to detect any autocorrelation in the location of missing relocations, for each burst of an object of class `ltraj`.

`summaryNAltraj` returns a summary of the number and proportion of missing values for each burst of an object of class `ltraj`.

`plotNAltraj` plots the missing values in an object of class `ltraj` against the time.

Usage

```
runsNAltraj(x, nrep = 500, plotit = TRUE, ...)

summaryNAltraj(x)

plotNAltraj(x, ...)
```

Arguments

`x`	An object of class `ltraj`
`nrep`	Number of randomisations
`plotit`	logical. Whether the results should be plotted on a graph
`...`	Further arguments to be passed to the generic function `plot`

Value

`runsNAltraj` returns a list of objects of class `randtest` (if a burst does not contain any missing value, the corresponding component is NULL).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information about objects of class `ltraj`, `setNA` for additional information about missing values in such objects

Examples

```
## Two relocations are theoretically separated by
## 10 minutes (600 seconds)
data(puechcirc)
puechcirc

## plot the missing values
plotNAltraj(puechcirc)


## Test for an autocorrelation pattern in the missing values
(runsNAltraj(puechcirc))
```

`rupicabau` *GPS Monitoring of One Chamois in the Bauges Mountains*

Description

This dataset is an object of class "ltraj" (regular trajectory, relocations every 20 minutes) containing the GPS relocations of two chamois during one day in the Bauges mountain (French Alps).

Usage

```
data(rupicabau)
```

Source

Office national de la chasse et de la faune sauvage, CNERA Faune de Montagne, 95 rue Pierre Flourens, 34000 Montpellier, France.

Examples

```
data(rupicabau)
plot(rupicabau)
```

`sahrlocs2kselect` *Preparation of K-Select Analysis*

Description

`sahrlocs2kselect` converts an object of class sahrlocs into a list that contains the arguments needed for a kselect analysis (see `help(kselect)`).

Usage

```
sahrlocs2kselect(sahr)
```

Arguments

`sahr`	an object of class `sahrlocs`

Value

Returns a list with the following components:

`tab`	a data frame giving the habitat composition of the home range of animals
`factor`	a factor giving for each row of `tab` the name of the corresponding animal
`weight`	a numeric vector giving for each row of `tab` the number of relocations numbered in this cell of the raster map

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Calenge, C., Dufour, A.B. and Maillard, D. (*submitted*). K-select analysis, a new method to analyse habitat selection in radio-tracking studies.

See Also

as.sahrlocs for additional information on the objects of class sahrlocs, kselect for a K-select analysis, and kasc2df for additional information on the index component of the output.

Examples

```
data(puechabon)
sahr <- puechabon$sahr
s <- sahrlocs2kselect(sahr)

# 1. PCA of the dataset
pc <- dudi.mix(s$tab, scannf = FALSE)

# 2. k-select analysis
kn <- kselect(pc, s$factor, s$weight, scannf = FALSE)


scatter(kn)
```

sahrlocs2niche *OMI Analysis of Radio-Tracking Data*

Description

sahrlocs2niche converts an object of class sahrlocs into a list that contains the arguments needed for a niche analysis (function niche in package ade4).

Usage

```
sahrlocs2niche(x, ani = names(x$hr), var = names(x$sa), used = c("hr",
"locs"))
```

Arguments

x	an object of class sahrlocs.
ani	a character vector giving the name of the animals in the analyses
var	a character vector giving the name of the variables in the analyses

`used`	a character string. If `"hr"`, the cells of the raster map that are considered used by the animals are taken from the `hr` component of the object `sahrlocs`, if `"locs"`, the cells of the raster map that are considered used by the animals are taken from the `locs` component (see `as.sahrlocs`))

Value

Returns a list with three components:

`index`	a vector of integer giving the index of the rows of `x$sa` kept for the analysis (this component may then be used with the function `df2kasc`).
`tab`	the table on which the `dudi` analysis will be processed.
`y`	a table giving the weights of the niche analysis.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

References

Doledec, S., Chessel, D. and Gimaret, C. (2000) Niche separation in community analysis: a new method. *Ecology*, **81**, 2914–1927.

See Also

`as.sahrlocs` for additional information on the objects of class `sahrlocs`, `niche` for an OMI analysis, and `kasc2df` for additional information on the `index` component of the output.

Examples

```
data(puechabon)
sahr <- puechabon$sahr

s <- sahrlocs2niche(sahr, used = "locs")


# 1. dudi.mix of the dataset
pc <- dudi.mix(s$tab, scannf = FALSE)

# 2. niche analysis
n <- niche(pc, s$y, scannf = FALSE)

plot(n)
```

`scatter.enfa`	*Scatter Plot of the Results of the ENFA*

Description

Performs the scatter diagrams of objects of class `enfa`.

Usage

```
## S3 method for class 'enfa'
scatter(x, xax = 1, yax = 2, pts = FALSE, nc = TRUE,
             percent = 95, clabel = 1, side = c("top", "bottom", "none"),
             Adensity, Udensity, Aangle, Uangle, Aborder, Uborder,
             Acol, Ucol, Alty, Ulty, Abg, Ubg, Ainch, Uinch, ...)
```

Arguments

`x`	an object of class `enfa`
`xax`	the column number for the x-axis
`yax`	the column number for the y-axis
`pts`	logical. Whether the points should be drawn. If `FALSE`, minimum convex polygons are displayed
`nc`	whether or not the niche center should be displayed
`percent`	100 minus the proportion of outliers to be excluded from the computation of the minimum convex polygons
`clabel`	a character size for the columns
`side`	if `"top"`, the legend of the kept axis is upside, if `"bottom"` it is downside, if `"none"` no legend
`Adensity`	the density of shading lines, in lines per inch, for the available pixels polygon. See `polygon` for more details
`Udensity`	the density of shading lines, in lines per inch, for the used pixels polygon. See `polygon` for more details
`Aangle`	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the available pixels polygon
`Uangle`	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the used pixels polygon
`Aborder`	the color for drawing the border of the available pixels polygon. See `polygon` for more details
`Uborder`	the color for drawing the border of the used pixels polygon. See `polygon` for more details
`Acol`	the color for filling the available pixels polygon. if `pts == FALSE`, the color for the points corresponding to available pixels

`Ucol`	the color for filling the used pixels polygon. if `pts == FALSE`, the color for the points corresponding to used pixels
`Alty`	the line type for the available pixels polygon, as in `par`
`Ulty`	the line type for the used pixels polygon, as in `par`
`Abg`	if `pts == TRUE`, background color for open plot symbols of available pixels
`Ubg`	if `pts == TRUE`, background color for open plot symbols of used pixels
`Ainch`	if `pts == TRUE`, heigth in inches of the available pixels
`Uinch`	if `pts == TRUE`, heigth in inches of the largest used pixels
`...`	further arguments passed to or from other methods

Details

`scatter.enfa` displays a factorial map of pixels, as well as the projection of the vectors of the canonical basis multiplied by a constant of rescaling. The kept axes for the plot are specified in a corner.

Author(s)

Mathieu Basille <`basille@ase-research.org`>

References

Basille, M., Calenge, C., Marboutin, E., Andersen, R. & Gaillard, J.M. (2008) Assessing habitat selection using multivariate statistics: Some refinements of the ecological-niche factor analysis. *Ecological Modelling*, **211**, 233–240.

See Also

`enfa`, `scatter`

Examples

```
data(lynxjura)

map <- lynxjura$map

## We keep only "wild" indices.
tmp <- lynxjura$locs[,4]!="D"
locs <- lynxjura$locs[tmp, c("X","Y")]

## We perform a square root transformation
## of the variable to normalize it
map[,4] <- sqrt(map[,4])

## We perform the ENFA
tmp <- data2enfa(map, locs[tmp, c("X","Y")])
(enfa1 <- enfa(dudi.pca(tmp$tab, scannf=FALSE),
               tmp$pr, scannf = FALSE))
scatter(enfa1)
```

scatterniche *Display the Niche in the Ecological Space*

Description

scatterniche displays the niche in the Ecological space (multidimensional space defined by habitat variables).

Usage

```
scatterniche(x, pr, xax = 1, yax = 2, pts = FALSE,
             percent = 95, clabel = 1,
             side = c("top", "bottom", "none"),
             Adensity, Udensity, Aangle, Uangle, Aborder,
             Uborder, Acol, Ucol, Alty, Ulty, Abg,
             Ubg, Ainch, Uinch, ...)
```

Arguments

x	a data frame giving the value of environmental variables (columns) in resource units (rows, e.g. the pixels of a raster map)
pr	a vector giving the utilisation weight for each resource unit
xax	the column number for the x-axis
yax	the column number for the y-axis
pts	logical. Whether the points should be drawn. If FALSE, minimum convex polygons are displayed
percent	100 minus the proportion of outliers to be excluded from the computation of the minimum convex polygons
clabel	a character size for the columns
side	if "top", the legend of the kept axis is upside, if "bottom" it is downside, if "none" no legend
Adensity	the density of shading lines, in lines per inch, for the available pixels polygon. See polygon for more details
Udensity	the density of shading lines, in lines per inch, for the used pixels polygon. See polygon for more details
Aangle	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the available pixels polygon
Uangle	the slope of shading lines, given as an angle in degrees (counter-clockwise), for the used pixels polygon
Aborder	the color to draw the border of the available pixels polygon. See polygon for more details
Uborder	the color to draw the border of the used pixels polygon. See polygon for more details

Acol	the color for filling the available pixels polygon. if pts==FALSE, the color for the points corresponding to available pixels
Ucol	the color for filling the used pixels polygon. if pts==FALSE, the color for the points corresponding to used pixels
Alty	the line type for the available pixels polygon, as in par.
Ulty	the line type for the used pixels polygon, as in par.
Abg	if pts==TRUE, background color for open plot symbols of available pixels
Ubg	if pts==TRUE, background color for open plot symbols of used pixels
Ainch	if pts==TRUE, heigth in inches of the available pixels
Uinch	if pts==TRUE, heigth in inches of the largest used pixels
...	further arguments passed to or from other methods

Author(s)

Mathieu Basille <basille@ase-research.org>
Clement Calenge <clement.calenge@oncfs.gouv.fr>

Examples

```
data(chamois)
cp <- count.points(chamois$locs, chamois$map)
chamois$map
li <- kasc2df(chamois$map)
cpi <- c(cp)[li$index]
## we focus on the distance to ecotone and on the slope,
## after centring and scaling (with the help of a PCA)
scatterniche(dudi.pca(li$tab[,2:3], scannf=FALSE)$tab, cpi)
scatterniche(dudi.pca(li$tab[,2:3], scannf=FALSE)$tab, cpi, pts=TRUE)
```

schoener *Compute Schoener's ratio*

Description

schoener computes the Schoener's ratio on radio-tracking data.
schoener.rtest performs a randomization test of the equality of the Schoener's ratio to 2

Usage

```
schoener(tr, keep, byburst = TRUE)
schoener.rtest(tr, keep, byburst = TRUE, nrep = 500)
```

Arguments

`tr`	an object of class `traj`
`keep`	a vector of length 2, giving the lower and the upper bound of the time interval for which a pair of relocations is considered in the computation of t\^2 (see details). These values are given in seconds.
`byburst`	logical. If `TRUE`, the Schoener's ratio is computed by burst. If `FALSE`, the ratio is computed by animal.
`nrep`	the number of randomisations of the test.

Details

The Schoener's ratio is a measure of time-autocorrelation in the data. This ratio is computed as the squared mean distance between "neighbour" relocations (t\^2) divided by the squared mean distance between the relocations and their barycenter (r\^2). The theoretical value of this ratio under the hypothesis of independance of the relocations is 2.

Swihart and Slade (1985) consider as neighbour two successive relocations. However, the Schoener's ratio computed in this way makes sense biologically only if the relocations are equally spaced in time. However, as indicated by these authors, "such a data set probably is the exception rather than the rule because many problems may arise in taking a locational reading at a specified time".

In this function, we define as "neighbour" two relocations (not necessarily successive relocations) separated by a time interval comprised within the bounds specified in the vector `keep` (in seconds). For example, if `keep = c(60, 300)`, all relocations separated by a time interval comprised between 1 and 5 minutes are considered in the computation. Thus, the total number of pairs of relocations m taken into account in the computation may be larger than the number of relocations n (m can be at most equal to n*(n-1)/2).

Value

returns an object of class `schoener`.

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

References

Schoener, T.W. (1981) An empirically based estimate of home range. *Theoretical Population Biology*, **20**, 281–325.

Swihart, R.K. and Slade, N.A. (1985) Testing for independence of observations in animal movements. *Ecology*, **66**, 1176–1184.

Solow, A.R. (1989) A randomization test for independence of animal locations. *Ecology*, **70**, 1546–1549.

Examples

```
data(puechcirc)
puechcirc <- ltraj2traj(puechcirc)
puechcirc$date[1:10]

## Relocations are taken every 10 minutes
## For example we consider relocations as
## neighbour when they are separated by a time
## interval comprised between 5 and 15 minutes
schoener(puechcirc, keep = c(5*60, 15*60))
## Not run:
schoener.rtest(puechcirc, keep = c(5*60, 15*60))

## End(Not run)
```

`set.limits` *Define the Same Time Limits for several Bursts in a Regular Trajectory*

Description

This function sets the same time limits for several bursts in a regular trajectory.

Usage

```
set.limits(ltraj, begin, dur, pattern,
           units = c("sec", "min", "hour", "day"),
           tz = "", ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`begin`	a character string which is used to determine the time of beginning of the study period (see below)
`dur`	the duration of the study period
`pattern`	a character string indicating the conversion specifications for `begin` (see below)
`units`	a character string indicating the time units of `dur`
`tz`	A timezone specification to be used for the conversion of `begin`. System-specific, but `""` is the current time zone, and `"GMT"` is UTC (see `help(strptime)`)
`...`	additional arguments to be passed to other functions

Details

Some studies are intended to compare regular trajectories of the same duration collected at different period. For example, the aim may be to identify the differences/similarities between different days (each one corresponding to a burst of relocation) in the pattern of movements of an animal between 05H00 and 08H00, with a time lag of 5 minutes. In such cases, it is often convenient that the relocations of the bursts are paired (e.g. the fifth relocation correspond to the position of the animal at 5H30 for all bursts).

The function `set.limits` is intended to ensure that the time of beginning, the end, and the duration of the trajectory is the same for all bursts of the object `ltraj`. If relocations are collected outside the limits, they are removed. If the actual time limits of the burst cover a shorter period than those specified, missing values are added.

Note that "time of beginning" is not a synonym for "date". That is, two trajectories of the same animal, both beginning at 05H00 and ending at 08H00, have the same time of beginning, but are necessarily not sampled on the same day, which implies that they correspond to different dates. For this reason, the time of beginning is indicated to the function `set.limits` by a character string, and the parameter `pattern` should indicate the conversion specifications. These conversions specifications are widely documented on the help page of the function `strptime`. For example, to indicate that the trajectory begins at 5H00, the value for `begin` should be `"05:00"` and the value for `pattern` should be `"%H:%M"`. If the trajectory should begin on january 10th, the value for `begin` should be `"01:10"` and `pattern` should be `"%m:%d"`. Note that the only conversion specifications allowed in this function are `%S` (seconds), `%M` (minutes), `%H` (hours), `%d` (day), `%m` (month), `%Y` (year with century), `%w` (weekday), and `%j` (yearday). See `help(strptime)` for additional information on these convention specifications.

Value

an object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information on objects of class `ltraj`, `sett0` for additional information on regular trajectories, and `sd2df` for additionnal information about regular trajectories of the same duration. See also `strptime` for further information about conversion specifications for dates.

Examples

```
## load data on the ibex
data(ibex)
ibex

## The monitoring of the 4 ibex should start and end at the same time
## define the time limits
```

```
ib2 <- set.limits(ibex, begin="2003-06-01 00:00", dur=14,
                  units="day", pattern="%Y-%m-%d %H:%M")
ib2
is.sd(ib2)

## All the trajectories cover the same study period
## Relocations are collected at the same time. This dataset can now be
## used for studies of interactions between animals
```

---

setmask *Applies a Mask on Objects of Class 'asc' or 'kasc'*

Description

Applies a mask on objects of class asc or kasc. In other words, the function creates an object of class asc or kasc, with NA for all pixels NA on the masking map.

Usage

```
setmask(x, mask)
```

Arguments

x	an object of class asc or kasc
mask	an object of class asc

Value

Returns an object of class asc or kasc

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

asc for additionnal information on objects of class asc.

Examples

```
data(puechabon)
kasc <- puechabon$kasc
image(kasc)
elev <- getkasc(kasc, "Elevation")
slope <- getkasc(kasc, "Slope")
```

```
## ma is the mask: only areas with elevation > 250 m
## are kept
ma <- elev
ma[ma < 250] <- NA
ma <- getascattr(elev, ma)
image(ma)

## The mask is applied on maps of slope
slp <- setmask(slope, ma)
image(slp)

## The mask is applied on all maps in kasc
im <- setmask(kasc, ma)
image(im)
```

---

setNA *Place Missing Values in Objects of Class 'ltraj'*

Description

This function places missing values in an (approximately) regular trajectory, when a relocation should have been collected, but is actually missing.

Usage

```
setNA(ltraj, date.ref, dt, tol = dt/10,
      units = c("sec", "min", "hour", "day"), ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`date.ref`	an object of class `POSIXt` (see below)
`dt`	the time lag between relocations
`tol`	the tolerance, which measures the imprecision in the timing of data collection (see below)
`units`	a character string indicating the time units for `dt` and `tol`
`...`	additional arguments to be passed to the function `rec`

Details

During the field study, the collection of the relocations of a trajectory may sometimes fail, which results into missing values. The class `ltraj` deal with these missing values, so that it is recommended to store the missing values in the data *before* the creation of the object of class `ltraj`. For example, GPS collars often fail to locate the animal, so that the GPS data imported within R contain missing values. It is recommended to *not remove* these missing values.

However, sometimes, the data come without any information concerning the location of these missing values. If the trajectory is approximately regular (i.e. approximately constant time lag), it is possible to determine where these missing values should occur in the object of class `ltraj`. This is the role of the function `setNA`.

The relocations in the object of class `ltraj` may not have been collected at exactly identical time lag (e.g. a relocation is collected at 17H57 instead of 18H00). The function `setNA` requires that the imprecision in the timing is at most equal to `tol`. Because of this imprecision, it is necessary to pass a reference date as argument to the function `setNA`. This reference date is used to determine at which time the missing values should be placed.

The reference date is chosen so that the rest of the division of (date.relocations - reference.date) by the time lag `dt` is equal to zero. For example, if it is known that one of the relocations of the trajectory has been collected on January 16th 1996 at 18H00, and if the theoretical time lag between two relocations is of one hour, the date of reference could be (for example) the August 1st 2007 at 05H00, because these two dates are separated by an exact number of hours (i.e. an exact number of `dt`). Therefore, any date fulfilling this condition could be passed as reference date. Alternatively, the August 1st 2007 at 05H30 is an uncorrect reference date, because the number of hours separating these two dates is not an integer.

Value

An object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information about objects of class `ltraj`. `sett0` (especially the examples of this help page) and `is.regular` for additional information about regular trajectories.

Examples

```
data(porpoise)
foc <- porpoise[1]

## the list foc does not contain any missing value:
foc
plotNAltraj(foc)

## we remove the second to tenth relocation
foc[[1]] <- foc[[1]][-c(2:10),]
foc <- rec(foc)

## The missing values are not visible:
foc
plotNAltraj(foc)
```

```
## The porpoise is located once a day.
## We use the first relocation as the reference date
foc2 <- setNA(foc, foc[[1]]$date[1], 24*3600)

## Missing values are now present
foc2
plotNAltraj(foc2)
```

---

sett0 — *Round the Timing of Collection of Relocations to Obtain Regular Trajectory*

Description

This function rounds the timing of collection of relocations in an object of class `ltraj` to obtain a regular trajectory, based on a reference date.

Usage

```
sett0(ltraj, date.ref, dt, correction.xy = c("none", "cs"),
      tol = dt/10, units = c("sec", "min", "hour", "day"), ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`date.ref`	an object of class `POSIXt` containing either one reference date (the same for all animals) or `n` reference dates, where `n` is the number of bursts in `ltraj` (see below)
`dt`	the time lag between relocations
`correction.xy`	the correction for the coordinates. `"none"` (default), does not performs any correction. `"cs"` performs a correction based on the hypothesis that the animal moves at constant speed (see below).
`tol`	the tolerance, which measures the imprecision in the timing of data collection (see below)
`units`	the time units for `dt` and `tol`
`...`	additional arguments to be passed to the function `rec`

Details

Trajectories are stored in adehabitat as lists of "bursts" of successive relocations with the timing of relocation. Regular trajectories are characterized by a constant time lag `dt` between successive relocations (don't mix animals located every 10 minutes and animals located every day in a regular trajectory).

However, in many cases, the actual time lag in the data may not be equal to the theoretical time lag `dt`: there may be some negligible imprecision in the time of collection of the data (e.g. an error of a few seconds on a time lag of one hour).

But many functions of `adehabitat` require exact regular trajectories. `sett0` allows to round the date so that all the successive relocations are separated exactly by `dt`. The function `sett0` requires that the imprecision is at most equal to `tol`. To proceed, it is necessary to pass a reference date as argument.

The reference date is chosen so that the rest of the division of (date.relocations - reference.date) by `dt` is equal to zero. For example, if it is known that one of the relocations of the trajectory should have been collected on January 16th 1996 at 18H00, and if the theoretical time lag between two relocations is of one hour, the date of reference could be (for example) the August 1st 2007 at 05H00, because these two dates are separated by an exact number of hours. Alternatively, the August 1st 2007 at 05H30 is an uncorrect reference date, because the number of hours separating these two dates is not an integer.

Note that this rounding adds an error on the relocation. For example, the position of a moving animal at 17H57 is not the same as its position at 18H00. If the time imprecision in the data collection is negligible (e.g. a few seconds, while `dt` is equal to an hour), this "noise" in the relocations can be ignored, but if it is more important, a correction on the relocation is needed. The function `sett0` may correct the relocations based on the hypothesis of constant speed (which is not necessarily biologically relevant, see examples).

Note finally that missing values can be present in the trajectory. Indeed, there are modes of data collection that fail to locate the animal at some dates. These failures should appear as missing values in the regular trajectory. It is often convenient to use the function `setNA` before the function `sett0` to set the missing values in a (nearly) regular trajectory.

Value

an object of class `ltraj` containing a regular trajectory.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information on objects of class `ltraj`, `is.regular` for regular trajectories, `setNA` to place missing values in the trajectory and `cutltraj` to cut a trajectory into several bursts based on a criteria.

Examples

```
## Not run:
###########################################################################
##
```

```
##
## Transform a GPS monitoring on 4 ibex into a regular trajectory
##

data(ibexraw)
is.regular(ibexraw)

## the data are not regular: see the distribution of dt (in hours)
## according to the date

plotltr(ibexraw, "dt/3600")

## The relocations have been collected every 4 hours, and there are some
## missing data

## The reference date: the hour should be exact (i.e. minutes=0):
refda <- strptime("00:00", "%H:%M")
refda

## Set the missing values
ib2 <- setNA(ibexraw, refda, 4, units = "hour")

## now, look at dt for the bursts:
plotltr(ib2, "dt")

## dt is nearly regular: round the date:

ib3 <- sett0(ib2, refda, 4, units = "hour")

plotltr(ib3, "dt")
is.regular(ib3)

## ib3 is now regular

## End(Not run)
```

---

simm.bb *Brownian bridge motion*

Description

This function simulates a brownian bridge motion

Usage

```
simm.bb(date = 1:100, begin = c(0, 0), end = begin, id = "A1", burst = id)
```

Arguments

`date`	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class `POSIXct`
`begin`	a vector of length 2 giving the x and y coordinates of the location beginning of the trajectory
`end`	a vector of length 2 giving the x and y coordinates of the location ending the trajectory
`id`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
`burst`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)

Value

An object of class `ltraj`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>
Stephane Dray <`dray@biomserv.univ-lyon1.fr`>
Manuela Royer <`royer@biomserv.univ-lyon1.fr`>
Daniel Chessel <`chessel@biomserv.univ-lyon1.fr`>

See Also

`ltraj`, `hbrown`

Examples

```
plot(simm.bb(1:1000, end=c(100,100)), addpoints = FALSE)
```

`simm.brown` *Simulate a Bivariate Brownian Motion*

Description

This function simulates a Bivariate Brownian Motion.

Usage

```
simm.brown(date = 1:100, x0 = c(0, 0), h = 1, id = "A1", burst = id)
```

Arguments

`date`	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class `POSIXct`
`x0`	a vector of length 2 containing the coordinates of the startpoint of the trajectory
`h`	Scaling parameter for the brownian motion (larger values give smaller dispersion)
`id`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
`burst`	a character string indicating the identity of the simulated burst (see `help(ltraj)`)

Details

A bivariate Brownian motion can be described by a vector `B2(t) = (Bx(t), By(t))`, where `Bx` and `By` are unidimensional Brownian motions. Let `F(t)` the set of all possible realisations of the process `(B2(s), 0 < s < t)`. `F(t)` therefore corresponds to the known information at time `t`. The properties of the bivariate Brownian motion are therefore the following: (i) `B2(0)= c(0,0)` (no uncertainty at time `t = 0`); (ii) `B2(t) - B2(s)` is independent of `F(s)` (the next increment does not depend on the present or past location); (iii) `B2(t) - B2(s)` follows a bivariate normal distribution with mean `c(0,0)` and with variance equal to `(t-s)`.

Note that for a given parameter `h`, the process `1/h * B2(      t * h^2 )` is a Brownian motion. The function `simm.brown` simulates the process `B2(t * h^2)`. Note that the function `hbrown` allows the estimation of this scaling factor from data.

Value

An object of class `ltraj`

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>
Stephane Dray <dray@biomserv.univ-lyon1.fr>
Manuela Royer <royer@biomserv.univ-lyon1.fr>
Daniel Chessel <chessel@biomserv.univ-lyon1.fr>

References

~put references to the literature/web site here ~

See Also

`ltraj`, `hbrown`

Examples

```
plot(simm.brown(1:1000), addpoints = FALSE)

## Note the difference in dispersion:
plot(simm.brown(1:1000, h = 4), addpoints = FALSE)
```

simm.crw *Simulation of a Correlated Random Walk*

Description

This function simulates a correlated random walk

Usage

```
simm.crw(date=1:100, h = 1, r = 0,
         x0=c(0,0), id="A1", burst=id,
         typeII=TRUE)
```

Arguments

date	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class `POSIXct`. *Note that the time lag between two relocations should be constant* (regular trajectories required)
h	the scaling parameter for the movement length
r	The concentration parameter for wrapped normal distribution of turning angles
x0	a vector of length 2 containing the coordinates of the startpoint of the trajectory
id	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
burst	a character string indicating the identity of the simulated burst (see `help(ltraj)`)
typeII	logical. Whether the simulated trajectory should be of type II (`TRUE`, time recorded) or not (`FALSE`, time not recorded). See `help(ltraj)`.

Details

Since the seminal paper of Kareiva and Shigesada (1983), most biologists describe the trajectories of an animal with the help of two distributions: the distribution of distances between successive relocations, and the distribution of turning angles between successive moves (relative angles in the class `ltraj`). The CRW is built iteratively. At each step of the simulation process, the orientation of the move is drawn from a wrapped normal distribution (with concentration parameter `r`). The length of the move is drawn from a chi distribution, multiplied by `h * sqrt(dt)`. `h` is a scale parameter (the same as in the function `simm.brown()`, and the distribution is multiplied by sqrt(t) to make it similar to the discretized Brownian motion if `r == 0`.

Value

an object of class `ltraj`

Note

This function requires the package `CircStats`.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>
Stephane Dray <dray@biomserv.univ-lyon1.fr>
Manuela Royer <royer@biomserv.univ-lyon1.fr>
Daniel Chessel <chessel@biomserv.univ-lyon1.fr>

References

Kareiva, P. M. & Shigesada, N. (1983) Analysing insect movement as a correlated random walk. *Oecologia*, **56**: 234–238.

See Also

chi, rwrpnorm, simm.brown, ltraj, simm.crw, simm.mba

Examples

```
set.seed(876)
u <- simm.crw(1:500, r = 0.99, burst = "r = 0.99")
v <- simm.crw(1:500, r = 0.9, burst = "r = 0.9", h = 2)
w <- simm.crw(1:500, r = 0.6, burst = "r = 0.6", h = 5)
x <- simm.crw(1:500, r = 0, burst = "r = 0 (Uncorrelated random walk)",
              h = 0.1)
z <- c(u, v, w, x)
plot(z, addpoints = FALSE, perani = FALSE)
```

---

simm.levy *Simulates a Levy Walk*

Description

This function simulates a Levy walk

Usage

```
simm.levy(date = 1:500, mu = 2, l0 = 1, x0 = c(0, 0),
          id = "A1", burst = id, typeII = TRUE)
```

Arguments

date	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class POSIXct. *Note that the time lag between two relocations should be constant* (regular trajectories required)
mu	The exponent of the Levy distribution
l0	The minimum length of a step

`x0`	a vector of length 2 containing the coordinates of the startpoint of the trajectory
`id`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
`burst`	a character string indicating the identity of the simulated burst (see `help(ltraj)`)
`typeII`	logical. Whether the simulated trajectory should be of type II (`TRUE`, time recorded) or not (`FALSE`, time not recorded). See `help(ltraj)`.

Details

This function simulates a Levy flight with exponent `mu`. This is done by sampling a random relative angle from a uniform distribution (-pi, pi) for each step, and a step length generated by `dt * (l0 * (runif(1)^(1/(1 - mu))))`

Value

an object of class `ltraj`

Author(s)

Clement Calenge `<clement.calenge@oncfs.gouv.fr>`

References

Bartumeus, F., da Luz, M.G.E., Viswanathan, G.M. Catalan, J. (2005) Animal search strategies: a quantitative random-walk analysis. *Ecology*, **86**: 3078–3087.

See Also

`chi`, `rwrpnorm`, `simm.brown`, `ltraj`, `simm.crw`, `simm.mba`, `simm.levy`

Examples

```
set.seed(411)
w <- simm.levy(1:500, mu = 1.5, burst = "mu = 1.5")
u <- simm.levy(1:500, mu = 2, burst = "mu = 2")
v <- simm.levy(1:500, mu = 2.5, burst = "mu = 2.5")
x <- simm.levy(1:500, mu = 3, burst = "mu = 3")
par(mfrow=c(2,2))
lapply(list(w,u,v,x), plot, perani=FALSE)
```

simm.mba *Simulation of an Arithmetic Brownian Motion*

Description

This function simulates an Arithmetic Brownian Motion.

Usage

```
simm.mba(date = 1:100, x0 = c(0, 0), mu = c(0, 0),
         sigma = diag(2), id = "A1", burst = id)
```

Arguments

`date`	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class `POSIXct`
`x0`	a vector of length 2 containing the coordinates of the startpoint of the trajectory
`mu`	a vector of length 2 describing the drift of the movement
`sigma`	a 2*2 positive definite matrix
`id`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
`burst`	a character string indicating the identity of the simulated burst (see `help(ltraj)`)

Details

The arithmetic Brownian motion (Brillinger et al. 2002) can be described by the stochastic differential equation:

$$d\mathbf{z}(t) = \mu dt + \mathbf{\Sigma} d\mathbf{B}2(t)$$

Coordinates of the animal at time t are contained in the vector `z(t)`. `dz = c(dx, dy)` is the increment of the movement during dt. `dB2(t)` is a bivariate brownian Motion (see `?simm.brown`). The vector `mu` measures the drift of the motion. The matrix `Sigma` controls for perturbations due to the random noise modeled by the Brownian motion. It can also be used to take into account a potential correlation between the components dx and dy of the animal moves during dt (see Examples).

Value

An object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>
Stephane Dray <`dray@biomserv.univ-lyon1.fr`>
Manuela Royer <`royer@biomserv.univ-lyon1.fr`>
Daniel Chessel <`chessel@biomserv.univ-lyon1.fr`>

References

Brillinger, D.R., Preisler, H.K., Ager, A.A. Kie, J.G. & Stewart, B.S. (2002) Employing stochastic differential equations to model wildlife motion. *Bulletin of the Brazilian Mathematical Society* **33**: 385–408.

See Also

simm.brown, ltraj, simm.crw, simm.mou

Examples

```
set.seed(253)
u <- simm.mba(1:1000, sigma = diag(c(4,4)),
              burst = "Brownian motion")
v <- simm.mba(1:1000, sigma = matrix(c(2,-0.8,-0.8,2), ncol = 2),
              burst = "cov(x,y) > 0")
w <- simm.mba(1:1000, mu = c(0.1,0), burst = "drift > 0")
x <- simm.mba(1:1000, mu = c(0.1,0),
              sigma = matrix(c(2, -0.8, -0.8, 2), ncol=2),
              burst = "Drift and cov(x,y) > 0")
z <- c(u, v, w, x)
plot(z, addpoints = FALSE, perani = FALSE)
```

simm.mou *Simulation of a Bivariate Ornstein-Uhlenbeck Process*

Description

This function simulates a bivariate Ornstein-Uhlenbeck process for animal movement.

Usage

```
simm.mou(date = 1:100, b = c(0, 0),
         a = diag(0.5, 2), x0 = b,
         sigma = diag(2), id = "A1",
         burst = id)
```

Arguments

date	a vector indicating the date (in seconds) at which relocations should be simulated. This vector can be of class POSIXct
b	a vector of length 2 containing the coordinates of the attraction point
a	a 2*2 matrix
x0	a vector of length 2 containing the coordinates of the startpoint of the trajectory
sigma	a 2*2 positive definite matrix

`id`	a character string indicating the identity of the simulated animal (see `help(ltraj)`)
`burst`	a character string indicating the identity of the simulated burst (see `help(ltraj)`)

Details

The Ornstein-Uhlenbeck process can be used to take into account an "attraction point" into the animal movements (Dunn and Gipson 1977). This process can be simulated using the stochastic differential equation:

$$d\mathbf{z} = \mathbf{a}(\mathbf{b} - \mathbf{z}(t))dt + \mathbf{\Sigma} d\mathbf{B2}(\mathbf{t})$$

The vector `b` contains the coordinates of the attraction point. The matrix `a` (2 rows and 2 columns) contains coefficients controlling the force of the attraction. The matrix `Sigma` controls the noise added to the movement (see `?simm.mba` for details on this matrix).

Value

An object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>
Stephane Dray <`dray@biomserv.univ-lyon1.fr`>
Manuela Royer <`royer@biomserv.univ-lyon1.fr`>
Daniel Chessel <`chessel@biomserv.univ-lyon1.fr`>

References

Dunn, J.E., & Gipson, P.S. (1977) Analysis of radio telemetry data in studies of home range. *Biometrics* **33**: 85–101.

See Also

`simm.brown`, `ltraj`, `simm.crw`, `simm.mba`

Examples

```
set.seed(253)
u <- simm.mou(1:50, burst="Start at the attraction point")
v <- simm.mou(1:50, x0=c(-3,3),
              burst="Start elsewhere")
w <- simm.mou(1:50, a=diag(c(0.5,0.1)), x0=c(-3,3),
              burst="Variable attraction")
x <- simm.mou(1:50, a=diag(c(0.1,0.5)), x0=c(-3,7),
              burst="Both")
z <- c(u,v,w,x)

plot(z, addpoints = FALSE, perani = FALSE)
```

sliwinltr *Apply a Function on an Object of Class "ltraj", Using a Sliding Window*

Description

This function applies a function on an object of class "ltraj", using a sliding window.

Usage

```
sliwinltr(ltraj, fun, step, type = c("locs", "time"),
          units = c("sec", "min", "hour", "day"),
          plotit = TRUE, ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`fun`	the function to be applied, implying at least one of the descriptive parameters in the object of class `ltraj` (see below)
`step`	the half-width of the sliding window. If `type=="locs"`, it is a number of relocations. If `type=="time"` it is a number described by `units`
`type`	character string. If `type == "locs"`, `step` describes a number of relocations: if `type == "time"`, `step` describes a time lag.
`units`	if `type == "time"`, the time units described by `step`. Ignored otherwise
`plotit`	logical. Whether the result should be plotted
`...`	additional arguments to be passed to the function `rec`

Details

An object of class ltraj is a list with one component per burst of relocations. The function `fun` is applied to each burst of relocations. This burst of relocations should be refered as `x` in `fun`. For example, to compute the mean of the distance between successive relocations, the function `fun` is equal to `function(x) mean(x$dist)`.

Do not forget that some of the descriptive parameters in the object `ltraj` may contain missing values (see `help(ltraj)`). The function should therefore specify how to manage these missing values.

Value

If `type=="locs"`, a list with one component per burst of relocation containing the smoothed values for each relocation.

If `type=="locs"`, a list with one component per burst of relocation. Each component is a data frame containing the time and the corresponding smoothed values for each date.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

ltraj for additional information about objects of class ltraj

Examples

```
data(capreotf)

## computes the average speed of the roe deer in a moving window of width
## equal to 60 minutes
toto <- sliwinltr(capreotf, function(x) mean(x$dist/x$dt, na.rm = TRUE),
                  step = 30, type = "time", units = "min")

## zoom before the peak
head(toto[[1]])
plot(toto[[1]][1:538,], ty="l")
```

---

speed — *Computes the Speed Between Successive Relocations of an Animal - Deprecated*

Description

speed measures the speed between successive relocations of animals, using objects of class traj.

Usage

```
speed(x, id = levels(x$id), burst = levels(x$burst), date = NULL,
      units = c("seconds", "hours", "days"))
```

Arguments

x	an object of class traj
id	a character vector giving the identity of the animals for which the speed is to be computed
burst	a character vector giving the identity of the circuits for which the speed is to be computed (see traj)
date	a vector of class POSIXct of length 2 (beginning, end) delimiting the period of interest
units	a character string. It determines how the speeds are computed. For example, if the coordinates are given in meters, and if units = "seconds", speeds are returned in meters per second.

Value

Returns a data frame with the following components:

`id`	the identity of the animal
`x`	if the speed is computed between the relocation 1 and 2, the x coordinate of the relocation 1.
`y`	if the speed is computed between the relocation 1 and 2, the y coordinate of the relocation 1.
`date`	a vector of class `POSIXct`, giving the date of relocation 1.
`burst`	the identity of the circuit
`sp.x`	the computed speed of the animal in the x direction
`sp.y`	the computed speed of the animal in the y direction
`speed`	the computed speed of the animal on the plane.
`dt`	the duration between the two relocations (in the units given by the parameter `units`).

Note

The function `speed` is deprecated. The class `ltraj` computes the speeds automatically (see `ltraj`).

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`traj`

Examples

```
## Not run:
#### Computes the speed for each wild boar
#### monitored at Puechabon
data(puechcirc)
puechcirc <- ltraj2traj(puechcirc)
puechcirc

plot(puechcirc)
sp <- speed(puechcirc)
sp[1:4,]

## End(Not run)
```

`squirrel` *Radio-Tracking Data of Squirrels*

Description

This data set describes the use and availability of 5 habitat types for 17 squirrels monitored using radio-tracking. See also the dataset `squirreloc`.

Usage

```
data(squirrel)
```

Format

This list has three components:

`studyarea` a data frame giving the proportion of each habitat type (columns) on the study area. These proportions are repeated by rows, for all animals

`mcp` a data frame giving the proportion of each habitat type (columns) in the home range of each animal (rows)

`locs` a data frame giving the proportion of relocations of each animal (rows) reported in each habitat type (columns).

Source

Aebischer, N. J., Robertson, P. A. and Kenward, R. E. (1993) Compositional analysis of habitat use from animal radiotracking data. *Ecology*, **74**, 1313–1325.

`squirreloc` *Radio-tracking of squirrels*

Description

This data set contains the trajectories of 15 radio-monitored squirrels, as well as the vector maps of habitat composition.

Usage

```
data(squirreloc)
```

Format

This data set is a list of two objects:

- locsis a data frame containing the relocations of 15 squirrels (with three columns: `id, x, y`
- mapis an object of class `area` containing the habitat composition of the area. The habitat types and colour coding are stored in the attribute "info" of this object

Details

The dataset squirreloc comes from the Ranges VI software. It has been used to illustrate the compositional analysis (see ?compana) and the eigenanalysis of selection ratios (see ?eisera). See also the dataset squirrel.

Source

Kenward, R.E., South, A.B. and Walls, S.S. (2003). Ranges6 v1.2 : For the analysis of tracking and location data. Online manual. Anatrack Ltd. Wareham, UK. ISBN 0-9546327-0-2.

Examples

```
data(squirreloc)


## habitat:
are <- squirreloc$map
co <- attr(are, "info")
plot(are, colp = co[,2])

legend(-10, 210, unique(co[,1]),
       unique(co[,2]), bg="white")



## relocations
locs <- squirreloc$locs
li <- split(locs[,2:3], locs[,1])
opar <- par(mfrow=n2mfrow(length(li)), mar=c(0,0,2,0))
lapply(1:length(li), function(i) {
plot(are, colp = co[,2], main=names(li)[i], axes=FALSE)
points(li[[i]], pch=16, cex=1.5)
box()
})
par(opar)
```

---

storemapattr *Store Attributes of Maps of Class asc and kasc*

Description

storemapattr stores attributes of maps of class asc and kasc in an object of class mapattr.

Usage

```
storemapattr(x)
```

Arguments

`x`	an object of class `asc` or `kasc`

Details

This function is essentially used by programmers in functions dealing with maps of class `asc` or `kasc`. The function `getXYcoords` is an example of function using `storemapattr`.

Value

Returns an object of class `mapattr`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`kasc` and `asc` for additional information on the classes `kasc` and `asc`.

Examples

```
data(puechabon)
(kasc <- puechabon$kasc)

(toto <- storemapattr(kasc))
```

`subsample` *Subsample a Trajectory*

Description

This function subsamples a regular trajectory (i.e. changes the time lag between successive relocations).

Usage

```
subsample(ltraj, dt, nlo = 1,
          units = c("sec", "min", "hour", "day"), ...)
```

Arguments

`ltraj`	an object of class `ltraj`
`dt`	numeric value. The new time lag (should be a multiple of the time lag in `ltraj`)
`nlo`	an integer, or a vector of integers (with length equal to the number of bursts in `ltraj`), indicating the position of the first location of the new bursts in the old bursts. For example, if the previous time lag is equal to 300 seconds and the new time lag is 900 seconds, the new bursts may begin at the first, second or third relocations of the old bursts in `ltraj`.
`units`	character string. The time units of `dt`
`...`	additional arguments to be passed to other functions

Value

An object of class `ltraj`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information on objects of class `ltraj`, `is.regular` for regular trajectories.

Examples

```
data(capreotf)
plot(capreotf)

toto <- subsample(capreotf, dt = 900)
plot(toto)
```

`subsetmap` *Storing a Part of a Map*

Description

`subsetmap` is a generic function. It has methods for the classes `asc` and `kasc`. It is used to store a part of any given map into an other object.

Usage

```
subsetmap(x, xlim = NULL, ylim = NULL, ...)
```

Arguments

`x`	an object of class `asc` or `kasc`
`xlim`	numerical vector of length 2. The x limits of the rectangle including the new map
`ylim`	numerical vector of length 2. The y limits of the rectangle including the new map
`...`	further arguments passed to or from other methods

Details

If `xlim` or `ylim` are not provided, the function asks the user to click on the map to delimit the lower left corner and the higher right corner of the new map (see Examples).

Value

Returns an object of class `asc` or `kasc`

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>, improvements by Jon Olav Vik

See Also

`asc`, `kasc`

Examples

```
data(puechabon)
kasc <- puechabon$kasc
el <- getkasc(kasc, "Elevation")

## limits of the new map:
xl <- c(701561, 704017)
yl <- c(3160560, 3162343)

## computation of the new map:
su <- subsetmap(el, xlim = xl, ylim = yl)

## Display
opar <- par(mar = c(0,0,0,0))
layout(matrix(c(1,1,1,1,1,1,1,1,2), byrow = TRUE, ncol = 3))
image(el, axes = FALSE)
polygon(c(xl[1], xl[2], xl[2], xl[1]),
        c(yl[1], yl[1], yl[2], yl[2]))
image(su, axes = FALSE)
box()

par(opar)
par(mfrow = c(1,1))
```

```
### Gets this part for the whole kasc object
m <- subsetmap(kasc, xlim = xl, ylim = yl)
image(m)


## Not run: 
 ## Interactive example
 su <- subsetmap(kasc)

 image(su)

## End(Not run)
```

---

teal *Teal (Anas crecca) Ring Recovery Dataset*

Description

This dataset describes the location and date of recovery of 800 teal ringed in Camargue, southern France between January 1952 and February 1978 using standard dabbling duck funnel traps hidden in the vegetation.

Usage

```
data(teal)
```

Format

The following variables are given for each recovery:

`x` a numeric vector giving the longitude of the recovery

`y` a numeric vector giving the latitude of the recovery

`date` a vector of class POSIXct containing the date of recovery. Actually, only the day and month have been indicated. The year of recovery has been set to 1900 or 1901, and should not be taken into account in the analysis.

Details

The Camargue teal ringing program led to the recovery of 9,114 teals after the ringing of 59,187 birds. These 800 recoveries of this dataset are a subsample of the 4,652 birds recovered during the first year following ringing. Note that both the coordinates and the date have been jittered to preserve copyright on the data.

Source

La Tour du Valat. A research centre for the conservation of Mediterranean wetlands. Le Sambuc - 13200 Arles, France. http://en.tourduvalat.org/

Examples

```
data(teal)

plot(teal[,1:2], asp=1,
     xlab="longitude", ylab="latitude",
     main="Capture site (red) and recoveries")

points(attr(teal, "CaptureSite"), pch=16,
       cex=2, col="red")
```

`trajdyn` *Interactive Display of Objects of Class 'ltraj'*

Description

This function provides an interactive version of `plot.ltraj`, for the exploration of objects of class `ltraj`.

Usage

```
trajdyn(x, burst = attr(x[[1]], "burst"), hscale = 1, vscale = 1,
        recycle = TRUE, display = c("guess", "windows", "tk"), ...)
```

Arguments

`x`	an object of class `ltraj`
`burst`	a character string indicating the burst identity to explore
`hscale`	passed to tkrplot
`vscale`	passed to tkrplot
`recycle`	logical. Whether the trajectory should be recycled at the end of the display
`display`	type of display. The default `guess` uses a windows graphics device if `getOption('device')=='windows` otherwise it uses tk (requiring the `tkrplot` package).
`...`	additional arguments to be passed to the function `plot.ltraj`.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for further information on the class `ltraj`, and `plot.ltraj` for information on arguments that can be passed to this function.

Examples

```
## Not run:

## Without map
data(puechcirc)
trajdyn(puechcirc)


## With map
data(puechabon)
aa <- getkasc(puechabon$kasc, 1)
trajdyn(puechcirc, asc = aa)


## End(Not run)
```

---

typeII2typeI *Change the Type of a Trajectory*

Description

This function transforms a trajectory of type II (time recorded) into a trajectory of type I (time not recorded).

Usage

```
typeII2typeI(x)
```

Arguments

x	a object of class "ltraj" of type II

Value

An object of class "ltraj"

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

See Also

as.ltraj for additional information on the objects of class "ltraj"

Examples

```
data(puechcirc)
puechcirc
typeII2typeI(puechcirc)
```

vanoise — *Habitat Use by Three Species of Galliformes in the Vanoise National Parc*

Description

This data frame describes the habitat use by the Black Grouse (*Tetrao tetrix*), the Rock Partridge (*Alectoris graeca*) and the Rock Ptarmigan (*Lagopus mutus*), in the Vanoise National Park (French Alps), from 1990 to 2000.

Usage

```
data(vanoise)
```

Format

This data frame has 3110 rows and eight columns describing the habitat composition for each occurrence of three species of Galliformes. For each located occurrence (in rows), the employees of the national park have noted: the species, the elevation (in metres), the aspect (8 classes), the habitat type (7 categories, `FR` means "fallen rocks") and the date (season, day, month and year).

Source

Calenge, C., Martinot, J.P. and Lebreton, P. (2003) Ecological niche separation among mountain Galliformes in the Vanoise National Parc. *Game and Wildlife Science*, 20, 259-285.

wawotest — *Wald-Wolfowitz Test of Randomness*

Description

The function `wawotest.default` performs a Wald Wolfowitz test of the random distribution of the values in a vector. The function `wawotest.ltraj` performs this tests for the descriptive parameters `dx`, `dy` and `dist` in an object of class `ltraj`. The function `wawotest` is generic.

Usage

```
wawotest(x, ...)
## Default S3 method:
wawotest(x, alter = c("greater", "less"), ...)
## S3 method for class 'ltraj'
wawotest(x, ...)
```

Arguments

`x`	for `wawotest.default`
`alter`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two-sided"
`...`	additional arguments to be passed to other functions

Details

The statistic of the test is equal to A = sum(y(i) y(i+1)), with y(N+1) = y(1). Under the hypothesis of a random distribution of the values in the vector, this statistic is normally distributed, with theoretical means and variances given in Wald & Wolfowitz (1943).

Value

`wawotest.default` returns a vector containing the value of the statistic (`a`), its esperance (`ea`), its variance (`va`), the normed statistic (`za`) and the P-value. `wawotest.ltraj` returns a table giving these values for the descriptive parameters of the trajectory.

Author(s)

Stephane Dray <`dray@biomserv.univ-lyon1.fr`>

References

Wald, A. & Wolfowitz, J. (1943) An exact test for randomness in the non-parametric case based on serial correlation. *Ann. Math. Statist.*, **14**, 378–388.

See Also

`indmove` and `runsNAltraj` for other tests of independence to be used with objects of class "ltraj"

Examples

```
data(puechcirc)
puechcirc
wawotest(puechcirc)
```

`whale` *Argos Monitoring of Whale Movement*

Description

This data set contains the relocations of one right whale.

Usage

```
data(whale)
```

Format

This data set is a regular object of class `ltraj` (i.e. constant time lag of 24H00)

Details

The coordinates are given in decimal degrees (Longitude - latitude).

Source

http://whale.wheelock.edu/Welcome.html

Examples

```
data(whale)

plot(whale)
```

`which.ltraj` *Identify Relocations Fullfilling a Condition in an Object of Class "ltraj"*

Description

This function identifies the relocations fullfilling a condition in an object of class `ltraj`.

Usage

```
which.ltraj(ltraj, expr)
```

Arguments

`ltraj`	an object of class `ltraj`
`expr`	a character string giving any syntactically correct R logical expression implying the descriptive elements in `ltraj`

Value

A data frame giving the ID, the Bursts and the relocations for which the condition described by `expr` is verified.

Author(s)

Clement Calenge <`clement.calenge@oncfs.gouv.fr`>

See Also

`ltraj` for additional information about objects of class `ltraj`

Examples

```
data(puechcirc)
puechcirc

## Identifies the relocations for which time lag is
## upper than one hour
which.ltraj(puechcirc, "dt>3600")
puechcirc[burst="CH930824"][[1]][27:28,]


## Identifies the speed between successive
## relocations upper than 0.8 meters/second
which.ltraj(puechcirc, "dist/dt > 0.8")

## This is the case for example for the
## relocations #28, 58, 59 and 60 of "CH930824"
puechcirc[burst="CH930824"][[1]][c(28,58,59,60),]
```

`wi` *Computation of Selection Ratios for Habitat Selection Studies.*

Description

These functions compute the resource selection ratios (wi) for design I, II and III data types, with resources defined by several categories. Basic tests are also provided.

Usage

```
widesI(u, a, avknown = TRUE, alpha = 0.05)
widesII(u, a, avknown = TRUE, alpha = 0.05)
widesIII(u, a, avknown = TRUE, alpha = 0.05)
## S3 method for class 'wiI'
print(x, ...)
## S3 method for class 'wiII'
print(x, ...)
## S3 method for class 'wiIII'
print(x, ...)
## S3 method for class 'wi'
plot(x, caxis = 0.7, clab = 1, ylog = FALSE, errbar = c("CI", "SE"),
        main = "Manly selectivity measure", noorder = TRUE, ...)
```

Arguments

`u`	for `widesI`, a vector with named elements describing the sample of used resource units. For `widesII` and `widesIII` a matrix or a data frame giving the number of used resource units for each animal (in rows) in each resource category (in columns)
`a`	for `widesI` and `widesII`, a vector with named elements describing the sample or the proportion of available resource units. For `widesIII` a matrix or a data frame giving the number or the proportion of available resource units for each animal (in rows) in each resource category (in columns)
`avknown`	logical. `TRUE` if the available proportions are known, and `FALSE` if they are estimated
`alpha`	the threshold value for the tests and confidence intervals
`x`	an object of class `wi`
`caxis`	character size on axes to be passed to `par("cex.axis")`
`clab`	character size of axes labels to be passed to `par("cex.lab")`
`ylog`	logical. If `TRUE`, the selection ratios are plotted on a log scale
`errbar`	a character string. Type of error bars: either `"CI"` for confidence intervals or `"SE"` for standard errors
`main`	a character string. The title of the graph
`noorder`	logical. If `TRUE`, the habitat categories are ordered on the graph in decreasing order of their preference. If FALSE, they are not ordered (i.e. they are in the same order as the columns in `used` and `available`
`...`	additionnal arguments to be passed to the function `plot`

Details

`widesI` may be used to explore resource selection by animals, when designs I are involved (habitat use and availability are measured at the population level - animals are not identified). The function tests habitat selection with the Khi2 of Pearson and log-likelihood Khi2 (recommended, see Manly et al. 2003). The Manly selectivity measure (selection ratio = used/available) is computed, the preference / avoidance is tested for each habitat, and the differences between selection ratios are computed and tested.

`widesII` computes the selection ratios with design II data (same availability for all animals, but use measured for each one). Tests of identical habitat use for all animals, and of habitat selection are also provided.

`widesIII` computes the selection ratios for design III data (when the use and the availability are measured for each animal - see examples on the wild boar below). Habitat selection is tested using a Chi-square for each animal, and the overall habitat selection is also tested.

Note that all these methods rely on the following hypotheses: (i) independence between animals, and (ii) all animals are selecting habitat in the same way (in addition to "traditional" hypotheses in these kinds of studies: no territoriality, all animals having equal access to all available resource units, etc., see Manly et al. 2002 for further details).

Value

widesI returns a list of the class wiI. widesII returns a list of class wiII. widesIII returns a list of class wiIII. These objects are all inheriting from the class wi. They have the following components:

used.prop	the proportion of use for each resource type.
avail.prop	the proportion of available resources.
wi	the Manly selectivity measure (selection ratio: used/available).
se.wi	the standard error of the selection ratios.
comparisons	a list with the following components: diffwi a matrix with the differences of the selection ratios for each pair of resource type. ICdiffupper a matrix containing the upper limit of confidence interval on the differences of the selection ratios between each pair of resource type. ICdifflower a matrix containing the lower limit of confidence interval on the differences of the selection ratios between each pair of resource type. signif the ranking matrix, with the sign of the differences between the resource type in row and the resource type in column. When the difference is significant, the sign is tripled.
profile	the profile of preferences: resource types are sorted so that the left type is the most preferred and the right type is the most avoided. Habitats for which the selection ratios are not significant are connected by a line.
alpha	the parameter alpha of this function.
avknown	the parameter avknown of this function.
se.used	only for designs I, the standard error of the proportion of use.
se.avail	only for designs I, the standard error of the available proportion.
chisquwi	only for designs I, the results of Chi-Square tests of the hypothesis that the selection ratios are in average equals to zero.
Bi	only for designs I, equals to wi/sum(wi).
Khi2P	only for designs I, test of random resource use (Pearson statistic).
Khi2L	For designs I, test of random resource use (Log-likelihood statistic). For design III, global test of random resource use (Log-likelihood statistic)
Khi2L1	only for designs II, test of identical use of habitat by all animals.
Khi2L2	only for designs II, test of overall habitat selection.
Khi2L2MinusL1	only for designs II, test of hypothesis that animals are on average using resources in proportion to availability, irrespective of whether they are the same or not (= Khi2L2 - Khi2L1).
wij	only for designs II and III, a matrix with the selection ratios for all animals and all resource categories.
ICwiupper	only for designs II and III, the upper limit of the confidence intervals on the selection ratios.
ICwilower	only for designs II and III, the lower limit of the confidence intervals on the selection ratios.
Khi2Lj	only for designs III, the test of habitat selection for each animal.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>

References

Manly B.F.J., McDonald L.L., Thomas, D.L., McDonald, T.L. & Erickson, W.P. (2003) *Resource selection by animals - Statistical design and Analysis for field studies. Second edition* London: Kluwer academic publishers.

Thomas D. L. and Taylor E. J. (1990) Study designs and tests for comparing resource use and availability. *Journal of Wildlife Management*, **54**, 322–330.

See Also

compana for compositional analysis, and eisera to perform an eigenanalysis of selection ratios.

Examples

```
############################
## Example of moose (Manly et al., 2003, p.52)
## Known available proportions on design I data
moose.avail <- c(0.34, 0.101, 0.104, 0.455)
moose.used <- c(25, 22, 30, 40)
names(moose.used) <- c("InBurnInterior",
                     "InBurnEdge",
                     "OutOfBurnEdge",
                     "OutOfBurnFurther")
names(moose.avail) <- names(moose.used)
## Computation of wi
(wiRatio <- widesI(moose.used, moose.avail))

## plot the values of the selection ratios
opar <- par(mfrow=c(2,2))
plot(wiRatio)

par(opar)




############################
## Example of Elk (Manly et al., 2003, p.62)
## Estimated available proportions on design I data
elk.avail <- c(15, 61, 84, 40)
elk.used <- c(3, 90, 181, 51)
names(elk.used) <- c("0%", "1-25%", "26-75%", ">75%")
names(elk.avail) <- names(elk.used)
## Computation of wi
(wiRatio <- widesI(elk.used, elk.avail, avknown=FALSE))

## plot the values of the selection ratios
```

```
opar <- par(mfrow=c(2,2))
plot(wiRatio)

par(opar)




############################
## Example of Bighorn (Manly et al., 2003, p.67)
## Known available proportions on design II data
data(bighorn)
## Computation of wi
(wi <- widesII(bighorn$used, bighorn$availT, alpha = 0.1))

## plot the values of the selection ratios
opar <- par(mfrow=c(2,2))
plot(wi)



############################
## Example of Bighorn (Manly et al., 2003, p.74)
## Estimated available proportions on design II data
## Computation of wi
(wi <- widesII(bighorn$used, bighorn$availE, avknown = FALSE, alpha = 0.1))

## plot the values of the selection ratios
plot(wi)

par(opar)


############################
## Example of Wild boar
## Estimated available proportions on design III data
data(puechdesIII)
used <- puechdesIII$used
available <- puechdesIII$available

## calculation of the selectio ratios
## with sampled availability
(i <- widesIII(used,available, avknown = FALSE, alpha = 0.1))

opar <- par(mfrow = c(2,2))
plot(i)

par(opar)
```

Index