Package 'adehabitatHR'

November 28, 2017

Version 0.4.15

Date 2017-11-28

Depends R (>= 3.0.1), sp, methods, deldir, ade4, adehabitatMA, adehabitatLT

Suggests maptools, tkrplot, MASS, rgeos

Imports graphics, grDevices, stats

Title Home Range Estimation

Author Clement Calenge, contributions from Scott Fortmann-Roe

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

Description A collection of tools for the estimation of animals home range.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-11-28 22:07:15 UTC

R topics documented:

BRB *Utilization Distribution of an Animal Based on Biased Random Bridges*

Description

This function estimates the utilization distribution of one/several animals using a biased random bridge approach (Benhamou and Cornelis 2010, Benhamou 2011). This function also allows the decomposition of the utilization distribution into (i) an intensity distribution reflecting the average time spent by the animal in the habitat patches, and (ii) a recursion distribution reflecting the number of visits of the animal in the habitat patches (Benhamou and Riotte-Lambert, 2012).

Usage

```
BRB(ltr, D, Tmax, Lmin, hmin, type=c("UD","ID", "RD"), radius = NULL,
    maxt = NULL, filtershort = TRUE, habitat = NULL, activity = NULL,
    grid = 200, b=FALSE, same4all=FALSE, extent=0.5, tau = NULL,
    boundary=NULL)

BRB.D(ltr, Tmax = NULL, Lmin = NULL, habitat = NULL, activity = NULL)

BRB.likD(ltr, Dr=c(0.1,100),
         Tmax = NULL, Lmin = NULL,
         habitat = NULL, activity = NULL)
```

Arguments

ltr	an object of class ltraj.
D	a number corresponding to the diffusion parameter (in squared "units" per second, where "units" denote the units of the relocation coordinates) used for the estimation. Alternatively this parameter may be an object of class DBRB returned by the function BRB.D.
Dr	a vector of length two giving the lower and upper limits of the diffusion coefficient, within which the maximum likelihood could be found.
Tmax	the maximum duration (in seconds) allowed for a step built by successive relocations. All steps characterized by a duration dt greater than Tmax are not taken into account in the calculations.
Lmin	the minimum distance (in units of the coordinates) between successive relocations, defining intensive use or resting (See details). The distance should be specified in units of the relocation coordinates (i.e. in metres if they are specified in metres).
hmin	The minimum smoothing parameter (in units of the relocations coordinates), applied to all recorded relocations. See details for a description of this parameter.

type	The type of distribution expected by the user: "UD" returns the utilization distribution using the approach described by Benhamou and Cornelis (2010). "ID" returns the intensity distribution described in Benhamou and Riotte-Lambert (2012), i.e. a distribution reflecting the average time spent in habitat patches. "RD" returns the recursion distribution described in Benhamou and Riotte-Lambert (2012), i.e. a distribution reflecting the number of visits in habitat patches.
radius	If type = "ID" or "RD", the radius of the patches (in units of the relocation coordinates) used in the calculation of the residence time or the number of visits. If NULL, the radius is set to 3*hmin.
maxt	If type = "ID" or "RD", maximum time threshold (in seconds) that the animal is allowed to spend outside the patch before that we consider that the animal actually left the patch (see ?residenceTime).
filtershort	logical indicating the behaviour of the function when the length of a step is lower than Lmin (see details). It must be set to TRUE if track segments shorter than Lmin are assumed to correspond to resting periods, which thereby will be filtered out systematically, or to FALSE to take these short track segments into account when not associated to resting (animals active without moving more than Lmin). They then will be given a null diffusion coefficient.
habitat	optionally, an object of class SpatialPixelsDataFrame with one column describing the habitat type on the area.
activity	optionally, a character indicating the name of the variable in the infolocs component of ltr indicating the proportion of time between relocation i and relocation i+1 during which the animal was active (Users of adehabitatLT prior to version 0.3 should read the section Warning below).
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 SpatialPixels, or a list of objects of class SpatialPixels with as many elements as there are bursts in ltr.
b	logical specifying how relocation and movement variances are combined. If TRUE, the relocation variance progressively merges with the movement component; if FALSE, the relocation variance has a constant weight (see Benhamou, 2011).
same4all	logical. if TRUE, the same grid is used for the estimation of all bursts. If FALSE, one grid is used per 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])))).
tau	interpolation time (tau, in seconds). Defaults to tmin/10, where tmin is the minimum duration of a step in ltr.
boundary	If, not NULL, an object inheriting the class SpatialLines defining a barrier that cannot be crossed by the animals. There are constraints on the shape of the barrier that depend on the smoothing parameter (***see details***)

Details

The function BRB uses the biased random bridge approach to estimate the Utilization Distribution of an animal with serial autocorrelation of the relocations. This approach is similar to the Brownian

bridge approach (see ?kernelbb), with several noticeable improvements. Actually, the Brownian bridge approach supposes that the animal movement is random and purely diffusive between two successive relocations: it is supposed that the animal moves in a purely random fashion from the starting relocation and reaches the next relocation randomly. The BRB approach goes further by adding an advection component (i.e., a "drift") to the purely diffusive movement: is is supposed that the animal movement is governed by a drift component (a general tendency to move in the direction of the next relocation) and a diffusion component (tendency to move in other directions than the direction of the drift).

The BRB approach is based on the biased random walk model. This model is the following: at a given time t, the speed of the animal is drawn from a probability density function (pdf) and the angle between the step and the east direction is drawn from a circular pdf with given mean angle and concentration parameters. A biased random walk occurs when this angular distribution is not uniform (i.e. there is a preferred direction of movement). Now, consider two successive relocations r1 = (x1, y1) and r2 = (x2, y2) collected respectively at times t1 and t2. The aim of the Biased Random Bridges approach is to estimate the pdf that the animal is located at a given place r = (x,y) at time ti (with t1 < ti < t2), given that it is located at r1,r2 at times t1,t2, and given that the animal moves according a biased random walk with an advection component determined by r1 and r2.

Benhamou (2011) proposed an approximation for this pdf, noting that it can be approximated by a circular bivariate normal distribution with mean location corresponding to (x1 + pi*(x2-x1), y1 + pi*(y2-y1)), where pi = (ti-t1)/(t2-t1). The variance-covariance matrix of this distribution is diagonal, with both diagonal elements corresponding to the diffusion coefficient D. This coefficient D can be estimated using the plug-in method, using the function BRB.D (for details, see Benhamou, 2011). Note that the diffusion parameter D can be estimated for each habitat type is a habitat map is available. Note that the function BRB.likD can be used alternatively to estimate the diffusion coefficient using the maximum likelihood method.

An important aspect of the BRB approach is that the drift component is allowed to change in direction and strength from one step to the other, but should remain constant during each of them. For this reason, it is required to set an upper time threshold Tmax. Steps characterized by a longer duration are not taken into account into the estimation of the pdf. This upper threshold should be based on biological grounds.

As for the Brownian bridge approach, this conditional pdf based on biased random walks takes an infinite value at times ti = t1 and ti = t2 (because, at these times, the relocation of the animal is known exactly). Benhamou proposed to circumvent this drawback by considering that the true relocation of the animal at times t1 and t2 is not known exactly. He noted: "a GPS fix should be considered a punctual sample of the possible locations at which the animal may be observed at that time, given its current motivational state and history. Even if the recording noise is low, the relocation variance should therefore be large enough to encompass potential locations occurring in the same habitat patch as the recorded location". He proposed two ways to include this "relocation uncertainty" component in the pdf: (i) either the relocation variance progressively merges with the movement component, (ii) or the relocation variance has a constant weight. This is controlled by the parameter b of the function. In both cases, the minimum uncertainty over the relocation of an animal is observed for ti = t1 or t2. This minimum standard deviation corresponds to the parameter hmin. According to Benhamou and Cornelis, "hmin must be at least equal to the standard deviation of the localization errors and also must integrate uncertainty of the habitat map when UDs are computed for habitat preference analyses. Beyond these technical constraints, hmin also should incorporate a random component inherent to animal behavior because any recorded location, even if accurately recorded and plotted on a reliable map, is just a punctual sample of possible locations at which the

animal may be found at that time, given its current motivational state and history. Consequently, hmin should be large enough to encompass potential locations occurring in the same habitat patch as the recorded location".

Practically, the BRB approach can be carried out with the help of the movement-based kernel density estimation (MKDE) developed by Benhamou and Cornelis (2010). This method consists in dividing each step i into Ti/tau intervals, where Ti is the duration of the step (in seconds) and tau is the interpolation time (in seconds). A kernel density estimation is then used to estimate the required pdf, with a smoothing parameter varying with each interpolated location ri and corresponding to: hi^2 = hmin^2 + 4pi(1-pi)(hmax^2 - hmin^2)Ti/Tmax. In this equation, hmax^2 corresponds to hmin^2+ D*Tmax/2 if b is FALSE and to D*Tmax/2 otherwise. Note that this smoothing parameter may be a function of the habitat type where the interpolated relocation occurs if the diffusion parameters are available for each habitat types.

The special case where a given step covers a distance lower than Lmin merits further details. When the parameter filtershort = TRUE, it is always assumed that the animal was resting at this time, and this step is filtered out before the estimation. When the parameter filtershort = FALSE, this assumption is not made. In this case, the behaviour of the function depends on the availability of a variable measuring the activity of the animal (when the name of the variable containing the ai in the infolocs component is passed as the parameter activity; see ?infolocs for additionnal information on this component). If the animal was active during the step, the smoothing parameter hi is set to hmin for this step. This procedure allows to give more weight to the immediate surroundings of this relocation (indicating an intensive use of these immediate surroundings). If the animal was inactive, then the animal was resting and the step is filtered out before the estimation. Note however that activity value may sometimes be relatively high while the animal is resting, e.g. if disturbed by flies, possibly requiring manual correction of activity values based on the distance moved between relocations).

If no activity variable is available and filtershort = FALSE, it is always suppose that the animal was active between the two relocations, the step is not filtered out and the smoothing parameter hi is set to hmin for this step.

The parameter boundary allows to define a barrier that cannot be crossed by the animals. When this parameter is set, the method described by Benhamou and Cornelis (2010) for correcting boundary biases is used. The boundary can possibly be defined by several nonconnected lines, each one being built by several connected segments. Note that there are constraints on these segments (not all kinds of boundary can be defined): (i) each segment length should at least be equal to 3*h (the size of "internal lane" according to the terminology of Benhamou and Cornelis), (ii) the angle between two line segments should be greater that pi/2 or lower that -pi/2. The UD of all the pixels located within a band defined by the boundary and with a width equal to 6*h ("external lane") is set to zero.

Benhamou and Riotte-Lambert (2012) showed that the space use at any given location, as estimated by this approach, can be seen as the product between the mean residence time per visit times the number of visits of the location. They proposed an approach allowing the decomposition of the UD into two components: (i) the intensity distribution reflecting this average residence time and (ii) a recursion distribution reflecting the number of visits. This function allows to estimate these two components, by setting the argument type to "ID" and "RD" respectively.

Note that all the methods available to deal with objects of class estUDm are available to deal with the results of the function BRB (see ?kernelUD).

Value

BRB returns an object of class estUDm when the UD is estimated for several animals, and estUD when only one animal is studied.

BRB.D and BRB.likD returns a list of class DBRB, with one component per burst containing a data frame with the diffusion parameters.

Warning

Users of the version 0.2 of adehabitatHR should be careful that there was a slight inconsistency in the package design: whereas all the parameters characterizing the steps in an object of class "ltraj" (e.g. dist, dx, dy) describe the step between relocations i and i+1, it was expected for BRB that the activity described the proportion of activity time between relocation i-1 and i. This inconsistency has now been corrected since version 0.3.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>, based on a C translation of the Pascal source code of the program provided by Simon Benhamou.

References

Benhamou, S. (2011) Dynamic approach to space and habitat use based on biased random bridges *PLOS One*, **6**, 1–8.

Benhamou, S. and Cornelis, D. (2010) Incorporating Movement Behavior and Barriers to Improve Biological Relevance of Kernel Home Range Space Use Estimates. *Journal of Wildlife Management*, **74**, 1353–1360.

Benhamou, S. and Riotte-Lambert, L. (2012) Beyond the Utilization Distribution: Identifying home range areas that are intensively exploited or repeatedly visited. *Ecological Modelling*, **227**, 112–116.

See Also

kernelbb for the Brownian bridge kernel estimation, kernelUD and estUD-class for additional information about objects of class estUDm and estUD, infolocs for additional information about the infolocs component, as.ltraj for additional information about the class ltraj.

Examples

```
## Example dataset used by Benhamou (2011)
data(buffalo)

## The trajectory:
buffalo$traj

## The habitat map:
buffalo$habitat

## Show the dataset
```

```
plot(buffalo$traj, spixdf = buffalo$habitat)

## Estimate the diffusion component for each habitat type
## Using the plug-in method
vv <- BRB.D(buffalo$traj, Tmax = 180*60, Lmin = 50,
            habitat = buffalo$habitat, activ = "act")

vv

## Note that the values are given here as m^2/s, whereas
## they are given as m^2/min in Benhamou (2011). The
## values in m^2 per min are:
vv[[1]][,2]*60

## Approximately the same values, with slight differences due to
## differences in the way the program of Benhamou (2011) and the present
## one deal with the relocations occurring on the boundary between two
## different habitat types
## Note that an alternative estimation of the Diffusion coefficient
## could be found using maximum likelihood
vv2 <- BRB.likD(buffalo$traj, Tmax = 180*60, Lmin = 50,
                habitat = buffalo$habitat, activ = "act")
vv2
vv[[1]][,2]*60


## Estimation of the UD with the same parameters as those chosen by
## Benhamou (2011)
ud <- BRB(buffalo$traj, D = vv, Tmax = 180*60, tau = 300, Lmin = 50, hmin=100,
          habitat = buffalo$habitat, activity = "act", grid = 50, b=0,
          same4all=FALSE, extent=0.5)
ud

## Show the UD.
image(ud)

## Not run:
## Example of the decomposition of the UD into a recursion distribution
## and a intensity distribution (Benhamou and Riotte-Lambert 2012).
##
## 1. Intensity Distribution using the same parameters as Benhamou and
## Riotte-Lambert (2012)

id <- BRB(buffalo$traj, D = 440/60, Tmax = 3*3600, Lmin = 50, type = "ID",
          hmin=100, radius = 300, maxt = 2*3600, activity="act", filtershort=FALSE,
          grid = 200, extent=0.1)

rd <- BRB(buffalo$traj, D = 440/60, Tmax = 3*3600, Lmin = 50, type = "RD",
          hmin=100, radius = 300, maxt = 2*3600, activity="act", filtershort=FALSE,
          grid = 200, extent=0.1)

ud <- BRB(buffalo$traj, D = 440/60, Tmax = 3*3600, Lmin = 50,
          hmin=100, radius = 300, maxt = 2*3600, activity="act", filtershort=FALSE,
```

```
          grid = 200, extent=0.1)

par(mfrow = c(2,2), mar=c(0,0,2,0))
image(getvolumeUD(id))
title("ID")
image(getvolumeUD(rd))
title("RD")
image(getvolumeUD(ud))
title("UD")

## End(Not run)
```

CharHull *Estimation of the Home Range by Delaunay Triangulation method*

Description

The function CharHull implements the method developed by Downs and Horner (2009) for the home range estimation.

Usage

```
CharHull(xy, unin = c("m", "km"),
         unout = c("ha", "m2", "km2"),
         duplicates = c("random", "remove"), amount = NULL)
```

Arguments

xy	an object inheriting the class SpatialPoints containing the x and y coordinates of the relocations of the animal. If xy inherits the class SpatialPointsDataFrame, it should contain only one column (factor) corresponding to the identity of the animals for each relocation.
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)
duplicates	a setting to determine how duplicated points are handled. If "random" the duplicated points are slightly moved randomly. If "remove" the duplicated points are removed.
amount	if duplicates == random, this parameter controls the amount of noise added to the data (see the help page of jitter for additional information on this parameter).

Details

This method consists in the computation of the Delaunay triangulation of the set of relocations. Then, the triangles are ordered from the smallest to the largest. It is possible to select a given percentage of the smallest triangles (measured by their area) as the home-range estimation. The contour can be extracted with the function `getverticeshr`

Value

an object of the class `MCHu`

Note

This function relies on the package `deldir`.

Author(s)

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

References

Downs J.A. and Horner, M.W. (2009) A Characteristic-Hull Based Method for Home Range Estimation. *Transactions in GIS*, **13**, 527–537.

See Also

`MCHu` for further information on the class `MCHu`, and `SpatialPolygonsDataFrame-class` for additional information on this class. See `getverticeshr` to extract a given home range contour.

Examples

```
## Not run:
data(puechabonsp)
lo<-puechabonsp$relocs[,1]

## Home Range Estimation
res <- CharHull(lo)

## Displays the home range
plot(res)

## Computes the home range size
MCHu2hrsize(res)

## Computes the 95 percent home range
ver <- getverticeshr(res)
ver
plot(ver)

## End(Not run)
```

clusthr *Estimation of the Home Range by Single-Linkage Cluster Analysis*

Description

clusthr allows the estimation of the home range by single-linkage cluster analysis (see details).

Usage

```
clusthr(xy, unin = c("m", "km"),
        unout = c("ha", "m2", "km2"),
        duplicates=c("random","remove"), amount = NULL)
```

Arguments

xy	an object inheriting the class SpatialPoints containing the x and y relocations of the animal. If xy inherits the class SpatialPointsDataFrame, it should contain only one column (factor) corresponding to the identity of the animals for each relocation.
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)
duplicates	a setting to determine how duplicated points are handled. If "random" the duplicated points are slightly moved randomly. If "remove" the duplicated points are removed.
amount	if duplicates == random, this parameter controls the amount of noise added to the data (see the help page of jitter for additional information on this parameter).

Details

This method estimates home range using the single-linkage cluster analysis modified 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.

Note that a given home-range contour can be extracted using the function getverticeshr.

Value

The function clusthr returns either objects of class SpatialPolygonsDataFrame (if the relocations of only one animals are passed as the xy argument) or a list of SpatialPolygonsDataFrame of class MCHu – Multiple Convex Hull (if the relocations of several animals are passed as the xy argument).

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

MCHu for further information on the class MCHu, and SpatialPolygonsDataFrame-class for additional information on this class. See getverticeshr to extract a given home range contour.

Examples

```
data(puechabonsp)
lo<-puechabonsp$relocs[,1]

## Home Range Estimation
res <- clusthr(lo)

## Displays the home range
plot(res)

## Computes the home range size
MCHu2hrsize(res)

## get the 95 percent home range:
plot(getverticeshr(res, percent=95))
```

---

estUD-class *Class "estUD": Storing Utilization Distributions in R*

Description

This class is an extension of the class SpatialPixelsDataFrame of the package sp, and is designed to store the utilization distribution of an animal

Objects from the Class

Objects of class "estUD" can be created using the functions kernelUD and getvolumeUD.

Slots

h: Object of class "list" containing all information concerning the smoothing parameters used in the estimation process

vol: Object of class "logical" indicating whether the mapped values coorespond to the UD or to the volume under the UD (see ?kernelUD)

data: Object of class "data.frame" containing the values of the UD

Extends

Class "SpatialPixelsDataFrame", directly.

Methods

coerce signature(from = "estUD", to = "data.frame"): converts the object into a data frame

show signature(object = "estUD"): printing method of the object

Author(s)

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

See Also

SpatialPixelsDataFrame for additional information about this class, and kernelUD for additional information about the methods generating such objects.

Examples

```
## load the data
data(puechabonsp)

## estimate one UD for each animal
jj <- kernelUD(puechabonsp$relocs[,1])
image(jj)
jj


## Consider the first animal
jj[[1]]
class(jj[[1]])
image(jj[[1]])
```

findmax *Find Local Maxima on a Map of Class 'SpatialPixelsDataFrame'*

Description

findmax finds the local maxima on a map of class SpatialPixelsDataFrame.

Usage

```
findmax(x)
```

Arguments

x	a map of class SpatialPixelsDataFrame with one column

Details

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

Value

an object of class SpatialPoints containing the x and y coordinates of the local maxima.

Author(s)

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

See Also

SpatialPixelsDataFrame-class for additionnal information on objects of class SpatialPixelsDataFrame.

Examples

```
data(puechabonsp)

## estimates the UD
kud <- kernelUD(puechabonsp$relocs[,1])

## displays the maximum
image(kud[[1]])
points(findmax(kud[[1]]))
```

getverticeshr *Extract the home-range contour of one or several animals*

Description

These functions allow the extraction of the home-range contours computed using various methods (kernel home range, cluster home range, etc.)

Usage

```
getverticeshr(x, percent = 95, ...)
## S3 method for class 'estUD'
getverticeshr(x, percent = 95, ida = NULL, unin = c("m", "km"),
                               unout = c("ha", "km2", "m2"),
                               standardize = FALSE, ...)
## S3 method for class 'estUDm'
getverticeshr(x, percent = 95, whi = names(x),
                               unin = c("m", "km"),
                               unout = c("ha", "km2", "m2"),
                               standardize = FALSE, ...)
## S3 method for class 'MCHu'
getverticeshr(x, percent = 95, whi = names(x), ...)
## Default S3 method:
getverticeshr(x, percent = 95, ...)
```

Arguments

x	For getverticeshr.estUD, an object of class estUD. For getverticeshr.estUDm, an object of class estUDm. For getverticeshr.MCHu, an object of class MCHu.
percent	a single value giving the percentage level for home-range estimation
ida	a character string indicating the id of the polygons corresponding to the home range in the resulting SpatialPolygonsDataFrame (see the help page of SpatialPolygonsDataFrame). By default it is set to "homerange"
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)
whi	a vector of character strings indicating which animals should be returned.
standardize	a logical value indicating whether the UD should be standardized over the area of interest, so that the volume under the UD and *over the area* is equal to 1..
...	Additional arguments to be passed to and from other methods

Value

An object of class SpatialPolygonsDataFrame containing the selected home range contours of the animals.

Note

The function `getverticeshr.default` is present for compatibility purposes. Its use generates an error.

Author(s)

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

See Also

`kernelUD`, `kernelbb` or `kernelkc` for methods generating objects of classes `estUD` and `estUDm`, `clusthr`, `LoCoH.a` and `CharHull` for methods generating objects of class `MCHu`.

Examples

```
### Example with a kernel home range
data(puechabonsp)
loc <- puechabonsp$relocs

## have a look at the data
head(as.data.frame(loc))
## the first column of this data frame is the ID

## Estimation of UD for the four animals
(ud <- kernelUD(loc[,1]))

## Calculates the home range contour

ver <- getverticeshr(ud, percent=95)
ver
plot(ver)

## Example with a cluster home range
clu <- clusthr(loc[,1])
ver2 <- getverticeshr(clu, percent=95)
ver2
plot(ver2)
```

---

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(ltr, 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

ltr,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 SpatialPixels, or a list of objects of class SpatialPixels, 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 `estUDm`

`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. , `mcp` for estimation of home ranges using the minimum convex polygon, and for help on the function `plot.hrsize`.

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.
liker(x, sig2 = 58, rangesig1 = c(10, 100))

## Wow! we expected a too large standard deviation! Try again between
## 1 and 10:
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 = 100)
image(tata)


## OK, now look at the home range
image(tata)
plot(getverticeshr(tata, 95), add=TRUE, 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
coordinates(pts) <- c("x","y")
(kud <- kernelUD(pts))


###### The results

opar <- par(mfrow=c(2,2), mar=c(0.1,0.1,2,0.1))
plot(pts, pch=16)
title(main="The relocation pattern")
box()
plot(x, axes=FALSE, main="The trajectory")
box()
image(kud)
title(main="Classical kernel home range")
plot(getverticeshr(kud, 95), add=TRUE)
box()
image(kbb)
title(main="Brownian bridge kernel home range")
plot(getverticeshr(kbb, 95), 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])
   }
 }

## Displays the results
opar <- par(mar = c(0,0,2,0), mfrow = c(5,5))
foo <- function(x)
   {
     image(y[[x]])
     title(main = names(y)[x])
     points(tr[[1]][,c("x","y")], pch = 16)
   }
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. 2010, 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).

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)

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.
`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 `SpatialPixels`. In addition, for the function `kernelkc` this parameter can be a list of objects of class `SpatialPixels`, 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 (2009) 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 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 (2009) advocated the use of an automatic algorithm to select "optimal" values for the smoothing parameter, it is not implemented in adehabitatHR. 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 "estUDm" containing objects of class estUD, mapping one estimate of the UD per burst or id (depending on the value of the parameter byburst).

kernelkcbase returns an object of class "estUD" mapping the estimated UD.

Author(s)

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

References

Keating, K. and Cherry, S. (2009) Modeling utilization distributions in space and time. *Ecology*, **90**: 1971–1980.

Calenge, C., Guillemain, M., Gauthier-Clerc, M. and Simon, G. 2010. 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. *Journal of Ornithology*, **151**, 945–950.

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, col=grey(seq(1,0, length=8)))
              title(main=vv[i])

              ## highlight the area on which there is a probability
              ## equal to 0.95 to recover a bird
              ## ****warning! The argument standardize=TRUE should
              ## be passed, because the UD is defined in space and
              ## time, and because we estimate the UD just in space
              plot(getverticeshr(uu, 95, standardize=TRUE), add=TRUE,
                   border="red", 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], grid=100)

              ## 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, col=grey(seq(1,0,length=10)))
              title(main=vv[i])

              ## highlight the 95 percent home range
              ## we set standardize = TRUE because we want to estimate
              ## the home range in space from a UD estimated in space and
              ## time
              plot(getverticeshr(uu, 95, standardize=TRUE), lwd=2,
                   border="red", 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 <- getverticeshr(uu, 95, standardize=TRUE)
              plot(pc, xlim=c(510000, 530000),
                   ylim=c(6810000, 6825000))
              title(main=vv[i])
              })









################################################
##
## Example with several wild boars (linear time)

## load wild boar data
data(puechcirc)


## keep only the first two circuits:
puechc <- puechcirc[1:2]


## Now load the map of the elevation
data(puechabonsp)
```

```
## 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(puechabonsp$map,
                       xlim=c(698000,704000),
                       ylim=c(3156000,3160000))
                 title(main=vv[i])

                 ## and the UD, with contour lines
                 colo <- c("green","blue")
                 lapply(1:length(uu), function(i) {
                        contour(as(uu[[i]],"SpatialPixelsDataFrame"),
                                add=TRUE, col=colo[i])
                 })

                 ## 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 <- getverticeshr(uu, 95, standardize=TRUE)
                 plot(pc, xlim=c(510000, 530000),
                      ylim=c(6810000, 6825000))
                 title(main=hour)

                 ## compute the 50% home range
                 pc <- getverticeshr(uu, 50, standardize=TRUE)
                 plot(pc, add=TRUE, col="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(puechabonsp)
locs <- puechabonsp$relocs

## keeps only the wild boar Jean
locs <- locs[slot(locs, "data")[,1]=="Jean",]

## compute the number of days since the beginning
## of the monitoring
dd <- cumsum(c(0, diff(strptime(slot(locs, "data")[,4], "%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(as.data.frame(coordinates(locs)), 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])
                 plot(getverticeshr(ud, 95, standardize=TRUE),
                       border="red", lwd=2, add=TRUE)

                 ## Just to slow down the process
                 Sys.sleep(0.2)
                 })



## End(Not run)
```

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, method = c("HR", "PHR", "VI", "BA", "UDOI",
              "HD"), percent = 95, conditional = FALSE, ...)

kerneloverlaphr(x, method = c("HR", "PHR", "VI", "BA", "UDOI", "HD"),
                percent = 95, conditional = FALSE, ...)
```

Arguments

`xy`	an object of class `SpatialPointsDataFrame` containing only one column (which is a factor indicating the identity associated to the relocations))
`x`	an object of class `estUDm` containing several home-ranges for which the overlap is to be calculated
`method`	the desired method for the estimation of overlap (see details)
`percent`	the percentage 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 `percent`. 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(puechabonsp)

kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="VI", conditional=TRUE)

## Identical to
```

```
kud <- kernelUD(puechabonsp$relocs[,1],
                grid=200, same4all=TRUE)
kerneloverlaphr(kud, meth="VI", conditional=TRUE)

## other indices
kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="HR")

kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="PHR")

kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="BA")

kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="UDOI")

kerneloverlap(puechabonsp$relocs[,1],
              grid=200, meth="HD")

## End(Not run)
```

---

kernelUD *Estimation of Kernel Home-Range*

Description

The function kernelUD estimates the UD of one or several animals.

plotLSCV allows to explore the results of the least-square cross-validation algorithm used to find the best smoothing value.

image allows a graphical display of the estimates.

getvolumeUD and kernel.area provide utilities for home range and home-range size estimation.

getverticeshr stores the home range contour as an object of class SpatialPolygonsDataFrame (package sp), with one row per animal.

estUDm2spixdf can be used to convert the result into an object of class SpatialPixelsDataFrame

as.data.frame.estUD can be used to convert an object of class estUD as a data frame.

Usage

```
kernelUD(xy, h = "href", grid = 60,
         same4all = FALSE, hlim = c(0.1, 1.5),
         kern = c("bivnorm", "epa"), extent = 1,
         boundary = NULL)

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

```
## S3 method for class 'estUD'
image(x, ...)

## S3 method for class 'estUDm'
image(x, ...)

## S3 method for class 'estUD'
as.data.frame(x, row.names, optional, ...)

plotLSCV(x)

getvolumeUD(x, standardize = FALSE)

kernel.area(x, percent = seq(20, 95, by = 5),
            unin = c("m", "km"),
            unout = c("ha", "km2", "m2"), standardize = FALSE)

estUDm2spixdf(x)
```

Arguments

xy	An object inheriting the class SpatialPoints containing the x and y relocations of the animal. If xy inherits the class SpatialPointsDataFrame, it should contain only one column (factor) corresponding to the identity of the animals for each relocation.
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 inheriting the class SpatialPixels, that will be used for all animals. For the function kernelUD, it may in addition be a list of objects of class SpatialPixels, with named elements corresponding to each level of the factor id.
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 controlling the extent of the grid used for the estimation (the extent of the grid on the abscissa is equal to (min(abscissa.relocations) + extent * diff(range(abscissa.relocations))), and similarly for the ordinate).
same4all	logical. If TRUE, the same grid is used for all animals. If FALSE, one grid per animal is used. Note that when same4all = TRUE, the grid used for the es-

	timation is calculated by the function (so that the parameter grid cannot be a SpatialPixels object).
boundary	If, not NULL, an object inheriting the class SpatialLines defining a barrier that cannot be crossed by the animals. There are constraints on the shape of the barrier that depend on the smoothing parameter h (***see details***)
x	an object of class estUD (UD for one animal) or estUDm (UD for several animals). For the function estUDm2spixdf, an object of class estUDm only. For the function as.data.frame.estUD, an object of class estUD only.
percent	for kernel.area, a vector of percentage levels for home-range size estimation. For getverticeshr, a single value giving the percentage level for home-range estimation.
standardize	a logical value indicating whether the UD should be standardized over the area of interest, so that the volume under the UD and *over the area* is equal to 1.
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)
row.names	unused argument here
optional	unused argument here
...	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

$$\sigma^2 = 0.5(var(x) + var(y))$$

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: that the pixel values of the resulting object are equal to the percentage of the smallest home range containing this pixel. 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.

The parameter `boundary` allows to define a barrier that cannot be crossed by the animals. When this parameter is set, the method described by Benhamou and Cornelis (2010) for correcting boundary biases is used. The boundary can possibly be defined by several nonconnected lines, each one being built by several connected segments. Note that there are constraints on these segments (not all kinds of boundary can be defined): (i) each segment length should at least be equal to `3*h` (the size of "internal lane" according to the terminology of Benhamou and Cornelis), (ii) the angle between two line segments should be greater that `pi/2` or lower that `-pi/2`. The UD of all the pixels located within a band defined by the boundary and with a width equal to `6*h` ("external lane") is set to zero.

Value

The function `kernelUD` returns either: (i) an object belonging to the S4 class `estUD` (see `?estUD-class`) when the object `xy` passed as argument contains the relocations of only one animal (i.e., belong to the class `SpatialPoints`), or (ii) a list of elements of class `estUD` when the object `xy` passed as argument contains the relocations of several animals (i.e., belong to the class `SpatialPointsDataFrame`).

The function `getvolumeUD` returns an object of the same class as the object passed as argument (`estUD` or `estUDm`).

`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 `SpatialPolygonsDataFrame`.

`estUDm2spixdf` returns an object of class `SpatialPixelsDataFrame`.

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 distribution 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.

Benhamou, S. and Cornelis, D. (2010) Incorporating Movement Behavior and Barriers to Improve Biological Relevance of Kernel Home Range Space Use Estimates. *Journal of Wildlife Management*, **74**, 1353–1360.

See Also

mcp for help on the function plot.hrsize.

Examples

```
## Load the data
data(puechabonsp)
loc <- puechabonsp$relocs

## have a look at the data
head(as.data.frame(loc))
## the first column of this data frame is the ID


## Estimation of UD for the four animals
(ud <- kernelUD(loc[,1]))

## The UD of the four animals
image(ud)

## Calculation of the 95 percent home range
ver <- getverticeshr(ud, 95)

## and display on an elevation map:
elev <- puechabonsp$map
image(elev, 1)
plot(ver, add=TRUE, col=rainbow(4))
legend(699000, 3165000, legend = names(ud), fill = rainbow(4))

## Example of estimation using LSCV
udbis <- kernelUD(loc[,1], h = "LSCV")
image(udbis)


## Compare the estimation with ad hoc and LSCV method
## for the smoothing parameter
(cuicui1 <- kernel.area(ud)) ## ad hoc
plot(cuicui1)
(cuicui2 <- kernel.area(udbis)) ## LSCV
plot(cuicui2)

## Diagnostic of the cross-validation
plotLSCV(udbis)



## Use of the same4all argument: the same grid
## is used for all animals
## BTW, we indicate a grid with a fine resolution:
udbis <- kernelUD(loc[,1], same4all = TRUE, grid = 100)
image(udbis)
```

```
## Estimation of the UD on a map
## (e.g. for subsequent analyses on habitat selection)
## Measures the UD in each pixel of the map
udbis <- kernelUD(loc[,1], grid = elev)
image(udbis)


########################################
##
## Estimating the UD with the presence of a barrier
## The boars are located on the plateau of Puechabon (near
## Montpellier, France), and their movements are limited by the
## Herault river.

## We first map the elevation:
image(elev)

## Then, we used the function locator() to identify the limits of the
## segments of this barrier. BEWARE! The boundary should satisfy the two
## constraints: (i) segment length > 3*h, (ii) no angle lower than pi/2
## between successive segments. We choose a smoothing parameter of 100
## m, so that no segment length should be less than 300 m.
## Because the resolution of the map is 100 m, this means that no
## segment should cover less than 3 pixels. We have used the function
## locator() to digitize this barrier and then the function dput to
## have the following limits:

bound <- structure(list(x = c(701751.385381925, 701019.24105475,
                        700739.303517889,
                        700071.760160759, 699522.651915378,
                        698887.40904327, 698510.570051342,
                        698262.932999504, 697843.026694212,
                        698058.363261028),
                        y = c(3161824.03387414,
                        3161824.03387414, 3161446.96718494,
                        3161770.16720425, 3161479.28718687,
                        3161231.50050539, 3161037.5804938,
                        3160294.22044937, 3159389.26039528,
                        3157482.3802813)), .Names = c("x", "y"))

lines(bound, lwd=3)

## We convert bound to SpatialLines:
bound <- do.call("cbind",bound)
Slo1 <- Line(bound)
Sli1 <- Lines(list(Slo1), ID="frontier1")
barrier <- SpatialLines(list(Sli1))

## estimation of the UD
kud <- kernelUD(loc[,1], h=100, grid=100, boundary=barrier)
```

```
## Result:
image(kud)

## Have a closer look to Calou:
kud2 <- kud[[2]]
image(kud2, col=grey(seq(1,0,length=15)))
title(main="Home range of Calou")
points(loc[slot(loc,"data")[,1]=="Calou",], pch=3, col="blue")
plot(getverticeshr(kud2, 95), add=TRUE, lwd=2)
lines(barrier, col="red", lwd=3)
```

---

kver2spol *Conversion of old classes from adehabitat to classes from adehabitatHR*

Description

These functions convert home ranges available in adehabitat toward classes available in the package adehabitatHR.

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

khr2estUDm converts an object of class khr (kernel UD) into an object of class estUDm.

Usage

```
kver2spol(kv)
khr2estUDm(x)
```

Arguments

kv	an object of class kver.
x	an object of class khr.

Author(s)

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

LoCoH *Estimating LoCoH home ranges*

Description

The functions computes the home range of one or several animals using the LoCoH family of methods.

The functions `LoCoH.k`, `LoCoH.r`, and `LoCoH.a` implement the k-LoCoH, r-LoCoH, and a-LoCoH respectively (Getz et al. 2007).

The functions `LoCoH.k.area`, `LoCoH.r.area`, and `LoCoH.a.area` compute the curve showing the relationships between the home-range size (computed to a specified percent) and the k, r or a parameters respectively.

Usage

```
LoCoH.k(xy, k=5, unin = c("m", "km"),
        unout = c("ha", "m2", "km2"),
        duplicates=c("random","remove"), amount = NULL)

LoCoH.r(xy, r, unin = c("m", "km"),
        unout = c("ha", "m2", "km2"),
        duplicates=c("random","remove"), amount = NULL)

LoCoH.a(xy, a, unin = c("m", "km"),
        unout = c("ha", "m2", "km2"),
        duplicates=c("random","remove"), amount = NULL)



LoCoH.k.area(xy, krange, percent=100, unin = c("m", "km"),
             unout = c("ha", "m2", "km2"),
             duplicates=c("random","remove"), amount = NULL)

LoCoH.r.area(xy, rrange, percent=100, unin = c("m", "km"),
             unout = c("ha", "m2", "km2"),
             duplicates=c("random","remove"), amount = NULL)

LoCoH.a.area(xy, arange, percent=100, unin = c("m", "km"),
             unout = c("ha", "m2", "km2"),
             duplicates=c("random","remove"), amount = NULL)
```

Arguments

`xy`	An object inheriting the class `SpatialPoints` containing the x and y relocations of the animal. If `xy` inherits the class `SpatialPointsDataFrame`, it should con-

	tain only one column (a factor) corresponding to the identity of the animals for each relocation.
k	numeric. The number of nearest neighbors minus one out of which to create convex hulls
r	numeric. The convex hulls are created out of all points within r distance from the root points
a	numeric. Create convex hulls from the maximum number of nearest neighbors such that the sum of their distances is less than or equal to this parameter
unin	the units of the relocations coordinates. Either "m" 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
duplicates	a setting to determine how duplicated points are handled. If "random" the duplicated points are slightly moved randomly. If "remove" the duplicated points are removed.
amount	if duplicates == random, this parameter controls the amount of noise added to the data (see the help page of jitter for additional information on this parameter).
krange	a vector containing the values of k for which the home range size is to be estimated.
arange	a vector containing the values of k for which the home range size is to be estimated.
rrange	a vector containing the values of k for which the home range size is to be estimated.
percent	the percentage level of the home range. For the function plot.LoCoH, this value could also be the character string "all", indicating that all the polygons are to be displayed.

Value

The functions LoCoH.* return either objects of class SpatialPolygonsDataFrame (if the relocations of only one animals are passed as the xy argument) or a list of SpatialPolygonsDataFrame (if the relocations of several animals are passed as the xy argument).

The functions LoCoH.*.area return invisibly either a vector (if the relocations of only one animals are passed as the xy argument) or a data frame containing the home-range sizes for various values of k, r (rows) for the different animals (columns).

Note

These functions rely on the packages rgeos, gpclib, and maptools.

The LoCoH family of methods for locating Utilization Distributions consists of three algorithms: Fixed k LoCoH, Fixed r LoCoH, and Adaptive LoCoH. All the algorithms work by constructing a small convex hull for each relocation, and then incrementally merging the hulls together from smallest to largest into isopleths. The 10% isopleth contains 10% of the points and represents a higher utilization than the 100% isopleth that contains all the points.

Fixed k LoCoH: Also known as k-NNCH, Fixed k LoCoH is described in Getz and Willmers (2004). The convex hull for each point is constructed from the (k-1) nearest neighbors to that point. Hulls are merged together from smallest to largest based on the area of the hull.

Fixed r LoCoH: In this case, hulls are created from all points within `r` distance of the root point. When merging hulls, the hulls are primarily sorted by the value of k generated for each hull (the number of points contained in the hull), and secondly by the area of the hull.

Adaptive LoCoH: Here, hulls are created out of the maximum nearest neighbors such that the sum of the distances from the nearest neighbors is less than or equal to d. Use the same hull sorting as Fixed r LoCoH.

Fixed r LoCoH and Adaptive LoCoH are discussed in Getz et al (2007).

All of these algorithms can take a significant amount of time. Time taken increases exponentially with the size of the data set.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>
with contributions from Scott Fortmann-Roe <scottfr@gmail.com>

References

Getz, W.M. & Wilmers, C.C. (2004). A local nearest-neighbor convex-hull construction of home ranges and utilization distributions. *Ecography*, **27**, 489–505.

Getz, W.M., Fortmann-Roe, S.B, Lyons, A., Ryan, S., Cross, P. (2007). LoCoH methods for the construction of home ranges and utilization distributions. *PLoS ONE*, **2**: 1–11.

See Also

`MCHu`, `getverticeshr`.

Examples

```
## Not run:

## Load the data
data(puechabonsp)

## The relocations:
locs <- puechabonsp$relocs
locsdf <- as.data.frame(locs)
head(locsdf)

## Shows the relocations
plot(locs, col=as.numeric(locsdf[,1]))


## Examinates the changes in home-range size for various values of k
## Be patient! the algorithm can be very long
ar <- LoCoH.k.area(locs[,1], k=c(8:13))
```

```
## 12 points seems to be a good choice (rough asymptote for all animals)
## the k-LoCoH method:
nn <- LoCoH.k(locs[,1], k=12)

## Graphical display of the results
plot(nn, border=NA)

## the object nn is a list of objects of class
## SpatialPolygonsDataFrame
length(nn)
names(nn)
class(nn[[1]])

## shows the content of the object for the first animal
as.data.frame(nn[[1]])

## The 95% home range is the smallest area for which the
## proportion of relocations included is larger or equal
## to 95% In this case, it is the 22th row of the
## SpatialPolygonsDataFrame.
## The area covered by the home range is for this first animal
## equal to 22.87 ha.

## shows this area:
plot(nn[[1]][11,])


## rasterization of the home ranges:
## use the map of the area:
image(puechabonsp$map)
ras <- MCHu.rast(nn, puechabonsp$map, percent=100)
opar <- par(mfrow=c(2,2))
lapply(1:4, function(i) { image(ras,i); box()})
par(opar)


## r-LoCoH and a-LoCoH can be applied similarly


## End(Not run)
```

MCHu *The Class "MCHu": Managing Home Ranges Built by Multiple Convex Hulls*

Description

The class "MCHu" is designed to store home ranges built by multiple convex hulls, for example built using the single-linkage cluster algorithm (function clusterhr) or the LoCoH (e.g. function LoCoH.k).

The function plot.MCHu allows to graphically display the home-ranges.

MCHu.rast allows to compute a raster map of the home ranges.
MCHu2hrsize allows to compute the home range size for specified percentage levels for the home range (see help(plot.hrsize)).

spoldf2MCHu allows to convert a SpatialPolygonsDataFrame storing home ranges built by multiple convex hulls into an object of class "MCHu".

Usage

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

## S3 method for class 'MCHu'
plot(x, percent="all", points=NULL, ...)

MCHu.rast(x, spdf, percent=100)

MCHu2hrsize(x, percent=seq(20,100, by=10), plotit=TRUE)

spoldf2MCHu(spdf, nam="a")
```

Arguments

x	an object of class MCHu.
spdf	an object of class SpatialPixelsDataFrame.
points	an object of class SpatialPoints or SpatialPointsDataFrame with one column (a factor storing the identity of the animal for each relocation), containing the relocations of the animal(s).
percent	the percentage level of the home range. For the function plot.MCHu, this value could also be the character string "all", indicating that all the polygons are to be displayed.
plotit	a logical value indicating whether the results should be plotted.
nam	the name of the animal to be used in the object of class "MCHu".
...	additional arguments to be passed to the functions print and plot.

Details

The class "MCHu" is basically a list of objects of class SpatialPolygonsDataFrame, with one data frame per animal.

Value

The function MCHu.rast returns an object of class SpatialPixelsDataFrame.

The function MCHu2hrsize returns an object of class hrsize (see ?mcp.area).

Author(s)

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

See Also

clusthr and LoCoH for home range estimation methods returning this class of objects.

Examples

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

## The relocations:
locs <- puechabonsp$relocs
locsdf <- as.data.frame(locs)
head(locsdf)

## Shows the relocations
plot(locs, col=as.numeric(locsdf[,1]))


## 12 points seems to be a good choice (rough asymptote for all animals)
## the k-LoCoH method:
nn <- LoCoH.k(locs[,1], k=12)

## Graphical display of the results
plot(nn, border=NA)

## Rasterize the home range on the elevation map:
image(puechabonsp$map)
(oo <- MCHu.rast(nn, puechabonsp$map))
image(oo)


## End(Not run)
```

---

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.
hr.rast is used to rasterize a minimum convex polygon.
plot.hrsize is used to display the home-range size estimated at various levels.

Usage

```
mcp(xy, percent=95, unin = c("m", "km"),
    unout = c("ha", "km2", "m2"))

mcp.area(xy, percent = seq(20,100, by = 5),
         unin = c("m", "km"),
         unout = c("ha", "km2", "m2"), plotit = TRUE)

hr.rast(mcp, w)

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

Arguments

xy	An object inheriting the class SpatialPoints containing the x and y relocations of the animal. If xy inherits the class SpatialPointsDataFrame, it should contain only one column (a factor) corresponding to the identity of the animals for each relocation.
percent	A single number for the function mcp and a vector for the function mcp.area: 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())
mcp	an objet of class SpatialPolygons returned by the function mcp.
w	an objet of class SpatialPixelsDataFrame used as a reference for the rasterization.
...	additional 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 centroid 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 SpatialPolygonsDataFrame, in which the first column contains the ID of the animals, and the second contains the home range size.

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.

hr.rast returns an object of class SpatialPixelsDataFrame.

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, SpatialPolygonsDataFrame-class for additionnal information on the class SpatialPolygonsDataFrame.

Examples

```
data(puechabonsp)
rel <-  puechabonsp$relocs


## estimates the MCP
cp <- mcp(rel[,1])

## The home-range size
as.data.frame(cp)

## Plot the home ranges
plot(cp)

## ... And the relocations
plot(rel, col=as.data.frame(rel)[,1], add=TRUE)


## Computation of the home-range size:
cuicui1 <- mcp.area(rel[,1])

## Rasterization
ii <- hr.rast(cp, puechabonsp$map)

opar <- par(mfrow=c(2,2))
lapply(1:4, function(i) {image(ii, i); box()})
par(opar)
```

Index