Package 'adehabitatLT'

January 28, 2018

Version 0.3.23

Date 2018-01-28

Depends R (>= 2.10.0), sp, methods, ade4, adehabitatMA, CircStats, stats

Suggests maptools, tkrplot, MASS

Imports graphics, grDevices, utils

Title Analysis of Animal Movements

Author Clement Calenge, contributions from Stephane Dray and Manuela Royer

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

Description A collection of tools for the analysis of animal movements.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-01-28 16:12:50 UTC

R topics documented:

acfdist.ltraj *Compute correlogram for angular and linear descriptors of a movement path*

Description

The functions `acfdist.ltraj` and `acfang.ltraj` compute (and by default plot) a correlogram-like function .

Usage

```
acfdist.ltraj(x, which = c("dist", "dx", "dy"), nrep = 999, lag = 1,
              plot = TRUE, xlab = "Lag", ylab = "autocorrelation")

acfang.ltraj(x, which = c("absolute", "relative"), nrep = 999, lag = 1,
             plot = TRUE, xlab = "Lag", ylab = "autocorrelation")
```

Arguments

`x`	an object of the class `ltraj`
`which`	to select on which parameter the autocorrelation should be computed (see details).
`nrep`	the number of repetitions used to test the significance of autocorrelation for each lag value.
`lag`	maximum lag at which to calculate the autocorrelation. Default is 1.
`plot`	logical. If 'TRUE' (the default) the autocorrelation is plotted.
`xlab`	a title for the x axis
`ylab`	a title for the y axis

Details

The function `acfdist.ltraj` is used to compute a correlogram for linear descriptors and `acfang.ltraj` for angular descriptors (see `as.ltraj` for a description of these descriptors).

Statistics used are defined in Dray et al. (in press). They are based on squared differences between successive values. For angular descriptors, the statistic is based on the chord distance.

In the case of missing data, the computation of the correlograms is restricted to the pairs of successive observed data and only observed data are permuted (i.e. the structure of the missing data is kept constant under permutation).

The grey area represents a 95 % interval obtained after permutation of the data. If the observed data is outside this region, it is considered as significant and represetend by a black symbol. Note that no multiple-comparison adjustement is performed.

Value

A list of matrices. Each matrix corresponds to a 'burst'. The matrix contains for each lag value (column), the values of autocorrelation (observed, and the 2.5 %, 50 % and 97.5 % quantiles of for the set of `nrep` permutations of values).

Author(s)

Stephane Dray <dray@biomserv.univ-lyon1.fr>

References

Dray, S., Royer-Carenzi, M. and Calenge, C. The exploratory analysis of autocorrelation in animal movement studies. *Ecological Research*, in press.

Calenge, C., Dray, S. and Royer-Carenzi, M. (2009) The concept of animals trajectories from a data analysis perspective. *Ecological Informatics*, **4**,34–41.

See Also

`as.ltraj` for additional information on the class `ltraj`, `wawotest` for a simple test of the autocorrelation of the descriptive parameters on the trajectory.

Examples

```
## Not run:
data(puechcirc)
puechcirc
acfang.ltraj(puechcirc, lag=5)
acfdist.ltraj(puechcirc, lag=5)

## 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)
```

`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.

`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"),
         infolocs = data.frame(pkey = paste(id, date, sep="."),
                               row.names=row.names(xy)),
         proj4string = CRS())
## S3 method for class 'ltraj'
print(x, ...)
## S3 method for class 'ltraj'
summary(object, ...)
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)
infolocs	if not NULL, a data frame containing additionnal information on the relocations (e.g., precision). By default, a primary key is generated.
proj4string	an object of class CRS storing the projection information of the relocations.
...	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), as well as optionally metadata on the relocations (precision, etc.).

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`. Note that the time zone of dates in objects of type II should be the same for all bursts (this is checked by the functions of adehabitatLT).

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 adehabitatLT 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`.

One word now about the attribute `infolocs` of the object of class `ltraj`. This attribute is intended to store metadata concerning the relocations building the trajectory (the precision of the relocations, the value of environmental variables at this place, etc.). There are constraints on the structure of this attribute. Although any variable can be stored in this attribute, it is required that: (i) all the relocations take a value (or a missing value) for all variables, (ii) all the variables are measured (or correspond to missing values) for all bursts and ids. This means for example that the function `c.ltraj` cannot be used to combine an object where only variable "A" is stored as metadata and an object where only variable "B" is stored as metadata. The function `removeinfo` can be used to remove this attribute. Note also that the data.frames in the list infolocs should have the same row.names as the corresponding elements in the object of class `"ltraj"`.

Finally, note that an object of class `ltraj` *can* have an attribute named `"proj4string"`, storing the projection information of the object. The package adehabitatLT does not manage projection

information, but this attribute can be useful when an object of class ltraj is converted to other classes (in particular spatial classes). Note that this attribute can be NULL (identical to a NA CRS).

Value

summary.ltraj returns a data frame.
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 and one optionnal attribute: the attribute "id" indicates the identity of the animal, and the attribute "burst" indicates the identity of the burst. An optional attribute "infolocs" contains any additional information desired by the user (precision, etc.). Each main 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.

Author(s)

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

References

Calenge, C., Dray, S. and Royer, M. (2009) The concept of animals' trajectories from a data analysis perspective. *Ecological Informatics*, **4**: 34–41.

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.

Examples

```
data(puechabonsp)
locs <- puechabonsp$relocs
head(locs)
xy <- coordinates(locs)
df <- as.data.frame(locs)
id <- df[,1]

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

(litrI <- as.ltraj(xy, id = id, 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(df$Date)
da <- as.POSIXct(strptime(as.character(df$Date),"%y%m%d", tz="Europe/Paris"))


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

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

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

## 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]])
```

```
#####################################################
##
## Example of a trajectory of type II (time recorded)
## with an infolocs attribute:

data(capreochiz)
head(capreochiz)

## Create an object of class "ltraj"
cap <- as.ltraj(xy = capreochiz[,c("x","y")], date = capreochiz$date,
                id = "Roe.Deer", typeII = TRUE,
                infolocs = capreochiz[,4:8])
cap
```

---

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)
```

buffalo *GPS monitoring of a buffalo*

Description

This data set contains the relocations of an African buffalo (Syncerus caffer) monitored in the W National Park (Niger) by D. Cornelis, as well as the habitat map of the study area.

Usage

```
data(buffalo)
```

Format

This dataset is a list containing an object of class `ltraj` and a SpatialPixelsDataFrame.

Details

The "infolocs" component of the ltraj stores the proportion of the time duration between relocation i-1 and relocation i during which the animal was active.

Source

Cornelis D., Benhamou S., Janeau G., Morellet N., Ouedraogo M. & de Vissher M.-N. (submitted). The spatiotemporal segregation of limiting resources shapes space use patterns of West African savanna buffalo. Journal of Mammalogy.

Examples

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

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

Description

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

Usage

```
burst(ltraj)
burst(ltraj) <- value
id(ltraj)
id(ltraj) <- value
infolocs(ltraj, which)
infolocs(ltraj) <- value
removeinfo(ltraj)
```

Arguments

ltraj	an object of class ltraj
value	for the assignment functions burst and id, a character vector of up to the same length as ltraj. For infolocs a list of data frames of the same length of ltraj (with each component having the same number of rows as the corresponding element in ltraj).
which	an optional character vector containing the names of the variables in the infolocs attribute to be returned

Details

The functions id, burst and infolocs are accessor functions, and id<- and burst<- are replacement function. removeinfo removes the attribute infolocs from the object ltraj (see the help page of as.ltraj).

Value

For id and burst, a character vector of the same length as ltraj. For infolocs, the data frame containing the information on the relocations. removeinfo returns an object of class 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

## example of an object with an attribute "infolocs"
data(capreochiz)
head(capreochiz)

## Create an object of class "ltraj"
cap <- as.ltraj(xy = capreochiz[,c("x","y")], date = capreochiz$date,
                id = "Roe.Deer", typeII = TRUE,
                infolocs = capreochiz[,4:8])
cap
cap2 <- removeinfo(cap)
cap2

infolocs(cap)
```

`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 data.frame containing the relocations and metadata on these relocations (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)

head(capreochiz)
```

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)
```

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, pchi gives the distribution function, qchi gives the quantile function, and rchi 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)
```

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.

The criterion used to cut the trajectory may imply any of the parameters describing a trajectory in the object ltraj (e.g., "dt", "dist", "dx", etc. see the help page of as.ltraj), as well as any variable stored in the attribute "infolocs" of the object.

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, FALSE 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

```
## Not run:
#######################################################
##
## 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, "UTC")
     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

## 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`.

When replacement is done, it is required that `value` and `x` have the same variables in attribute `infolocs` (i.e., both contain the same variables or both do not contain any variable, see the help page of `as.ltraj`)

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
```

---

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). The corresponding metadata in the attribute `infolocs` are also returned.

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", tz="Europe/Paris"))
tutu <- gdltraj(puechcirc, min = lim[1],
                max = lim[2], type="POSIXct")
plot(tutu, perani = FALSE)
```

---

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.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, or the variables in the optional attribute infolocs .
...	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")
```

---

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)
```

`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

```
## Not run:
## theoretical independence between
br <- simm.brown(1:1000)
testang.ltraj(br)
testdist.ltraj(br)

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

```
## Not run:
## 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, "UTC")
     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")



## End(Not run)
```

`lavielle` *Segmentation of a time series using the method of Lavielle (1999, 2005)*

Description

These functions allow to perform a non-parametric segmentation of a time series using the penalized contrast method of Lavielle (1999, 2005). The function `lavielle` computes the contrast matrix (i.e., the matrix used to segment the series) either from a series of observations or from an animal trajectory. The function `chooseseg` can be used to estimate the number of segments building up the trajectory. The function `findpath` can be used to find the limits of the segments (see Details).

Usage

```
lavielle(x, ...)

## Default S3 method:
lavielle(x, Lmin, Kmax, ld = 1,
                          type = c("mean", "var", "meanvar"), ...)

## S3 method for class 'ltraj'
lavielle(x, Lmin, Kmax, ld = 1, which = "dist",
                         type = c("mean", "var", "meanvar"), ...)

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

chooseseg(lav, S = 0.75, output = c("full","opt"),
          draw = TRUE)

findpath(lav, K, plotit = TRUE)
```

Arguments

x	for lavielle.default, a vector containing the successive observations building up the series. For lavielle.ltraj, an object of class ltraj.
Lmin	an integer value indicating the minimum number of observations in each segment. Should be a multiple of ld.
Kmax	an integer value indicating the maximum number of segments expected in the series
ld	an integer value indicating the resolution for the calculation of the contrast function. The contrast function will be evaluated for segments containing the observations c(1:ld), c(1:(2*ld)), c(1:(3*ld)), and all segments will necessarily contain a multiple of ld observations. Note that ld should be set to values greater than 1 if memory problem occur
type	the type of contrast function to be used to segment the series (see Details)
which	a character string giving any syntactically correct R expression implying the descriptive elements in x or the variables in the optional attribute infolocs.
lav	an object of class "lavielle"
S	a value indicating the threshold in the second derivative of the contrast function
output	type of output expected (see the section value)
draw	a logical value indicating whether the decrease in the contrast function should be plotted
K	The number of segments
plotit	a logical value indicating whether the segmentation should be plotted
...	additional arguments to be passed from or to other functions

Details

The method of Lavielle (1999, 2005) per se finds the best segmentation of a time series, given that it is built by K segments. It searches the segmentation for which a contrast function (measuring the contrast between the actual series and the segmented series) is minimized. Different contrast functions are available measuring different aspects of the variation of the series from one segment to the next: when type = "mean", we suppose that only the mean of the segments varies between segments; when type = "var", we suppose that only the variance of the segments varies between segments; when type = "meanvar", we suppose that both the mean and the variance varies between segments. It is required to specify a value for the minimum number of observations Lmin in a segment, as well as the maximum number of segments Kmax in the series.

There are several approaches to estimate the best number of segments K to partition the time series. One possible approach is the graphical examination of the decrease of the contrast function with the number of segments. In theory, there should be a clear "break" in the decrease of this function after the optimal value of K. Lavielle (2005) suggested an alternative way to estimate automatically the optimal number of segments, also relying on the presence of a "break" in the decrease of the contrast function. He proposed to choose the last value of K for which the second derivative of a standardized constrast function is greater than a threshold S (see Lavielle, 2005 for details). Based on numerical experiments, he proposed to choose the value S = 0.75. Note, however, that for short time series (i.e. less than 500 observations) some simulations indicated that this value may not be optimal and may depend on the value of Kmax, so that the graphical method is maybe more appropriate.

Value

The function lavielle.default returns a list of class lavielle, with an attribute "typeseg" set to "default". This list contains the following elements:

contmat	The contrast matrix
sumcont	The optimal contrast
matpath	The matrix of the paths from the first to the last observation
Kmax	The maximum number of segments
Lmin	The minimum number of observations in a segment
ld	the value of the resolution ld
series	The time series

The function lavielle.ltraj also returns a list of class lavielle, with an attribute "typeseg" set to "ltraj".

The function chooseseg returns the optimal number of segments when output = "opt", and a dataframe containing the value of the contrast function Jk and of the second derivative D of the standardized contrast function for each possible value of K, if output = "full".

The function findpath return a list containing vectors giving the index of the first and last observations in each segment, when the object of class "lavielle" passed as argument is characterized by an attribute "typeseg" set to "default". When the attribute "typeseg" is set to "ltraj", this function returns an object of class ltraj where each burst correspond to a segment.

Note

The contrast matrix is a matrix of size `n*n` (with `n` the number of observations in the series). If `n` is large, memory problems may occur. In this case, setting `ld` to a value greater than one will allow to reduce the size of this matrix (i.e. it will be of size `k*k`, where `k = floor(n/ld)`). However, this will also reduce the resolution of the segmentation, so that the segment limits will be less precisely estimated.

Author(s)

Clement Calenge <clement.calenge@oncfs.gouv.fr>. The code is a C translation based on the Matlab code of M. Lavielle

References

Lavielle, M. (1999) Detection of multiple changes in a sequence of dependent variables. *Stochastic Processes and their Applications*, **83**: 79–102.

Lavielle, M. (2005) Using penalized contrasts for the change-point problem. Report number 5339, Institut national de recherche en informatique et en automatique.

Examples

```
################################################
##
## A simulated series

set.seed(129)
seri <- c(rnorm(100), rnorm(100, mean=2),
          rnorm(100), rnorm(100, mean=-3),
          rnorm(100), rnorm(100, mean=2))
plot(seri, ty="l", xlab="time", ylab="Series")

## Segmentation:
(l <- lavielle(seri, Lmin=10, Kmax=20))

## choose the number of segments
chooseseg(l)

## There is a clear break in the
## decrease of the contrast function after K = 6
## Moreover, Jk(6) >> 0.75 and Jk(7) << 0.75
## We choose 6 segments:
fp <- findpath(l, 6)
fp

## This list gives the limits of the segments
## for example, to get the first segment:
seg <- 1
firstseg <- seri[fp[[seg]][1]:fp[[seg]][2]]

################################################
```

```
##
## Now, changes of variance

## A simulated series
set.seed(129)
seri <- c(rnorm(100), rnorm(100, sd=2),
          rnorm(100), rnorm(100, sd=3),
          rnorm(100), rnorm(100, sd=2))
plot(seri, ty="l", xlab="time", ylab="Series")

## Segmentation:
(l <- lavielle(seri, Lmin=10, Kmax=20, type="var"))

## choose the number of segments
chooseseg(l)

## There is a clear break in the
## decrease of the contrast function after K = 6
## Moreover, Jk(6) >> 0.75 and Jk(7) << 0.75
## We choose 6 segments:
fp <- findpath(l, 6)
fp

## This list gives the limits of the segments
## for example, to get the first segment:
seg <- 1
firstseg <- seri[fp[[seg]][1]:fp[[seg]][1]]

################################################
##
## Example of segmentation of a trajectory

## Show the trajectory
data(porpoise)
gus <- porpoise[1]
plot(gus)

## Show the changes in the distance between
## successive relocations with the time
plotltr(gus, "dist")

## Segmentation of the trajectory based on these distances
lav <- lavielle(gus, Lmin=2, Kmax=20)

## Choose the number of segments
chooseseg(lav)
## 4 segments seem a good choice

## Show the partition
kk <- findpath(lav, 4)
plot(kk)
```

ld *Quick Conversion of Objects of Class ltraj from and to Dataframes*

Description

The two functions ld and dl are useful to quickly convert objects of class ltraj from and to dataframes.

Usage

```
ld(ltraj)
dl(x, proj4string=CRS())
```

Arguments

ltraj	an object of class ltraj
x	an object of class data.frame, containing at least columns named x,y,date.
proj4string	a valid CRS object containing the projection information.

Details

The function ld concatenates all bursts in an object of class ltraj, adds two columns named id and burst, and, when it is present, also adds the variables in the infolocs component.

The function dl creates an object of class ltraj from a data.frame. If no column named id exists, a random ID is generated. If no column named burst exists, the ID is used as burst. The columns named "dx", "dy", "dist", "dt", "R2n", "abs.angle" and "rel.angle" are recomputed by the function (see ?as.ltraj). Additional columns are used as the infolocs component.

Value

ld returns an object of class data.frame.
dl returns an object of class ltraj.

Author(s)

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

See Also

as.ltraj for additional information about objects of class ltraj

Examples

```
data(puechcirc)
puechcirc ## class ltraj
uu <- ld(puechcirc)
head(uu)
dl(uu)
```

`ltraj2spdf` *Conversion of the class "ltraj" to the package "sp"*

Description

These functions convert the class "ltraj" available in adehabitatLT toward classes available in the package sp.

`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`.

Usage

```
ltraj2spdf(ltr)
ltraj2sldf(ltr, byid = FALSE)
```

Arguments

`ltr`	an object of class `ltraj`.
`byid`	logical. If `TRUE`, one objects of class `Lines` correspond to one animal. if `FALSE`, one object of class `Lines` correspond to one burst.

Author(s)

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

See Also

`ltraj` for objects of class `ltraj`.

Examples

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



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

data(puechcirc)
plot(puechcirc)

toto <- ltraj2spdf(puechcirc)
plot(toto)
```

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

toto <- ltraj2sldf(puechcirc)
plot(toto)


}

## End(Not run)
```

`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 (and replaces the coordinates of relocation i+1 by the coordinates of relocation i).

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 (2009) 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 Gueguen (2001, 2009).

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. (unpublished) 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. (2009) Computing the likelihood of sequence segmentation under Markov modelling. Arxiv preprint arXiv:0911.3070.

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)
```

---

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)
```

`na.omit.ltraj` *Removes the missing values in a trajectory*

Description

`na.omit.ltraj` can be used to remove missing relocations from a trajectory.

Usage

```
## S3 method for class 'ltraj'
na.omit(object, ...)
```

Arguments

`object`	an object of class `ltraj`
`...`	additionnal arguments to be passed to or from other methods

Value

An object of class `ltraj`

Author(s)

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

See Also

`setNA` to place the missing values in the trajectory

Examples

```
data(puechcirc)
puechcirc

na.omit(puechcirc)
```

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")
```

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")), spixdf = NULL,
           spoldf = NULL, xlim = NULL, ylim = NULL, colspixdf =
           gray((240:1)/256), colspoldf = "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
spixdf	an object of class SpatialPixelsDataFrame
spoldf	an object of class SpatialPolygons
xlim	the ranges to be encompassed by the x axis
ylim	the ranges to be encompassed by the y axis
colspixdf	a character vector giving the colors of the map spixdf
colspoldf	a character vector giving the colors of the polygon contour map, when spoldf 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(puechabonsp)
plot(puechcirc, perani = FALSE, spixdf = puechabonsp$map[,1])

cont <- getcontour(puechabonsp$map[,1])
plot(puechcirc, spoldf = cont)
```

`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", pch = 16, cex = 0.7, addlines = TRUE,
        addpoints = TRUE,...)
```

Arguments

`x`	An object of class `ltraj`
`which`	a character string giving any syntactically correct R expression implying the descriptive elements in `x` or the variables in the optional attribute `infolocs`.
`pch`	the type of points on the plot (see `help(par)`).
`cex`	the size of points on the plot (see `help(par)`).
`addlines`	logical. Indicates whether lines should be added to the plot.
`addpoints`	logical. Indicates whether points should be added to the plot.
`...`	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)
```

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 `puechabonsp` in the package `adehabitatMA` 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.

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)
```

---

rasterize.ltraj *Rasterize a Trajectory*

Description

The function `rasterize.ltraj` allows to rasterize a trajectory.

Usage

```
rasterize.ltraj(ltr, map)
```

Arguments

`ltr`	An object of class `ltraj`
`map`	An object inheriting the class `SpatialPixels`

Value

A list of objects of class `SpatialPointsDataFrame`, with one component per burst in the object of class `ltraj`. Each object contains the coordinates of the pixels of the maps traversed by the trajectory. The number of the step that traverse each pixel is indicated.

Author(s)

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

See Also

as.ltraj for additional information about objects of class `ltraj`

Examples

```
data(puechabonsp)
data(puechcirc)

## Show the trajectories on the map
plot(puechcirc, spixdf = puechabonsp$map)

## rasterize the trajectories
ii <- rasterize.ltraj(puechcirc, puechabonsp$map)

## show, e.g. the first rasterized trajectory
tr1 <- ii[[1]]
head(tr1)
plot(tr1)

## so, for example, to see the pixels traversed by the third step of the
##  trajectory
points(tr1[tr1[[1]]==3,], col="red")

## So, if we want to calculate the mean elevation for each step:
mel <- over(tr1, puechabonsp$map)
mo <- tapply(mel[[1]], tr1[[1]], mean)
plot(mo, ty="l")

## It is clear that elevation decreases at the middle of the monitoring
## and increases again at the end (the animal sleeps on the plateau
## and goes down in the vineyards during the night).

## Now define an infolocs component in puechcirc corresponding to the
## mean elevation:

val <- lapply(1:length(ii), function(i) {

    ## get the rasterized trajectory
    tr <- ii[[i]]

    ## get the pixels of the map
    mel <- over(tr, puechabonsp$map)

    ## calculate the mean elevation
    mo <- tapply(mel[[1]], tr[[1]], mean)

    ## prepare the output
    elev <- rep(NA, nrow(puechcirc[[i]]))

    ## place the average values at the right place
    ## names(mo) contains the step number (i.e. relocation
    ## number +1)
    elev[as.numeric(names(mo))+1] <- mo

    ## Checks that the row.names are the same for
```

```
    ## the result and the ltraj component
    df <- data.frame(elevation = elev)
    row.names(df) <- row.names(puechcirc[[i]])

    return(df)
})

## define the infolocs component
infolocs(puechcirc) <- val

## and draw the trajectory
plotltr(puechcirc, "elevation")
```

`redisltraj` *Rediscretization of a Trajectory With Regular Step Length or Duration*

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, type = c("space", "time"))
```

Arguments

`l`	an object of class `ltraj`
`u`	the new step length in units of the coordinates or step duration in seconds
`burst`	The burst identity of trajectories to be rediscretized.
`samplex0`	Whether the first relocation of the trajectory should be sampled
`addbit`	logical. When `type="space"`, 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.
`type`	a character string indicating whether the step duration (`"time"`) or length (`"space"` should be constant

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).

When `type="time"`, a linear interpolation is performed to find new relocations separated by the given time lag.

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

```
####################################
##
## Example of space rediscretization

data(puechcirc)

puechcirc

## before rediscretization
plot(puechcirc, perani = FALSE)

## after rediscretization
toto <- redisltraj(puechcirc, 100)
plot(toto, perani = FALSE)


####################################
##
## Example of time rediscretization

data(buffalo)
```

```
tr <- buffalo$traj

## Show the time lag before rediscretization
plotltr(tr, "dt")

## Rediscretization every 1800 seconds
tr <- redisltraj(tr, 1800, type="time")

## Show the time lag after rediscretization
plotltr(tr, "dt")
```

`residenceTime` *Trajectory Analysis using the Residence Time Method*

Description

These functions can be used to apply the residence time method (Barraquand and Benhamou, 2008).

Usage

```
residenceTime(lt, radius, maxt, addinfo = FALSE,
              units = c("seconds", "hours", "days"))

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

## S3 method for class 'resiti'
plot(x, addpoints = FALSE, addlines = TRUE, ...)
```

Arguments

`lt`	an object of class `ltraj`
`radius`	the radius of the patch (in units of the coordinates)
`maxt`	maximum time threshold that the animal is allowed to spend outside the patch before that we consider that the animal actually left the patch (see Details)
`addinfo`	logical value. If `TRUE`, then the residence time method is added as a variable in the `infolocs` component of the object `lt`. If `FALSE` this function returns an object of class `resiti`
`units`	a character string indicating the time units of `maxt`
`x`	an object of class `"resiti"`
`addpoints`	logical. Whether points should be added to the plot.
`addlines`	logical. Whether lines should be added to the plot.
`...`	additionnal arguments to be passed to or from other methods

Details

Barraquand and Benhamou (2008) proposed a new approach to identify the places where the animals spend the most of their time, relying on the calculation of their residence time in the various places where they have been relocated. This approach is similar to the first passage time method: for a given value of `radius` and for a given relocation, the first passage time is defined as the time required by the animal to pass through a circle of given radius centred on the relocation (see the help page of the function `fpt` for additional details). The residence time associated to a given relocation corresponds to the first passage time calculated at this place plus the passage times that occurred in this circle before or after the current relocation, *given* that the animal did not spent a time greater than `maxt` before reentering the circle (see Barraquand and Benhamou, 2008, for details). It is therefore computed by determining the various times at which the path intersects the perimeter of the circle centred on the current relocation, both forward and backward, and then by summing the durations associated with the various portions of the path occurring within the circle. The graphical examination of the changes with time allow to identify the dates and places where the animal spent most of its time.

A partitionning method can be used to segment the series formed by the residence time into homogeneous segments. Barraquand and Benhamou (2008) propose the method of Lavielle (1999, 2005). See the function `lavielle` for details about this method.

Value

If `addinfo = FALSE`, the function `residenceTime` returns a list of class `"resiti"` where each element corresponds to a burst of the object `lt`. Each element is a `data.frame` with two columns: the date and the residence time associated with the date.

Author(s)

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

References

Barraquand, F. and Benhamou, S. (2008) Animal movement in heterogeneous landscapes: identifying profitable places and homogeneous movement bouts. *Ecology*, **89**, 3336–3348.

See Also

`lavielle` for the partitionning of the trajectory based on the residence time.

Examples

```
## Not run:
data(albatross)
ltr <- albatross[1]

## show the distances between successive relocations as a function
## of date
plotltr(ltr)

## focus on the first period
```

```
ltr <- gdltraj(ltr, as.POSIXct("2001-12-15", tz="UTC"),
               as.POSIXct("2003-01-10", tz="UTC"))

plot(ltr)
## We identify places that seem to be a patch and, with locator,
## we measure approximately their size.
## The approximate patch radius can be set equal to 100 km as a first try

plotltr(ltr, "dt")
## As a first try, we could set maxt equal to 15000 seconds, i.e.
## approximately 4 hours

## calculation of the residence time
res <- residenceTime(ltr, radius = 100000, maxt=4, units="hour")
plot(res)

## There seems to be about 10 segments. Let us try the method
## of Lavielle (1999, 2005) to segment this series:
## First calculate again the residence time as the infolocs attribute
## of the trajectory
res <- residenceTime(ltr, radius = 100000, maxt=4, addinfo = TRUE, units="hour")
res

## Note that the residence time is now an attribute of the infolocs
## component of res


## Now, use the Lavielle method, with Kmax set to 2-3 times the
## "optimal" number of segments, assessed visually according
## to the recommendations of Barraquand and Benhamou (2008)
## We set the minimum number of relocations in each segment to
## 10 observations (given that the relocations were theoretically
## taken every hour, this defines a patch as a place where the animal
## stays at least 10 hours: this also defines the scale of our study)

ii <- lavielle(res, which="RT.100000", Kmax=20, Lmin=10)

## Both the graphical method and the automated method to choose
## the optimal number of segments indicate 4 segments
## (see ?lavielle for a description of these methods):

chooseseg(ii)

## We identify the 4 segments: the method of Lavielle seems to do a good
## job:
(pa <- findpath(ii, 4))

## and we plot this partition:
plot(pa, perani=FALSE)


## Now, we could try a study at a smaller scale (patch = 50km):
res <- residenceTime(ltr, radius = 50000, maxt=4, addinfo = TRUE,
```

```
                     units="hour")
ii <- lavielle(res, which="RT.50000", Kmax=20, Lmin=10)

## 5 segments seem a good choice:
chooseseg(ii)

## There is more noise in the residence time, but
## the partition is still pretty clear:
(pa <- findpath(ii, 5))

## show the partition:
plot(pa, perani = FALSE)


## Now try at a larger scale (patch size=250 km)
res <- residenceTime(ltr, radius = 250000, maxt=4, addinfo = TRUE,
                     units="hour")
ii <- lavielle(res, which="RT.250000", Kmax=15, Lmin=10)

## 5 segments seem a good choice again:
chooseseg(ii)

## There is more noise in the residence time, but
## the partition is still pretty clear:
(pa <- findpath(ii, 5))

## show the partition:
plot(pa, perani = FALSE)


## End(Not run)
```

`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

Details

The statistics used here for the test is the number of runs in the sequence of relocations. For example, the sequence reloc-NA-NA-reloc-reloc-reloc-NA-NA-NA-reloc contains 5 runs, 3 runs of successful relocations and 2 runs of missing values. Under the hypothesis of random distribution of the missing values in the sequence, the theoretical expectation and standard deviation of the number of runs is known. The runs test is a randomization test that compares the standardized value of the number of runs (i.e. (value-expectation)/(standard deviation)) to the distribution of values obtained after randomizing the distribution of the NA in the sequence. Thus, a negative value of this standardized number of runs indicates that the missing values tend to be clustered together in the sequence.

Value

runsNAltraj returns a list of objects of class randtest (if a burst does not contain any missing value, the corresponding component is NULL).

Note

In the versions of adehabitatLT prior to 0.3.21, a bug occurred in the calculation of the P-value (the test actually presented the value 1-P). This bug is now corrected.

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)
```

`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 (and so is the corresponding metadata in the attribute infolocs). If the actual time limits of the burst cover a shorter period than those specified, missing values are added to the trajectory (and in the corresponding metadata in the attribute infolocs).

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", tz="Europe/Paris")
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
```

---

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 placement 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` (and in the optional attribute `infolocs`). 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 2017 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 adehabitatLT 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 adehabitatLT 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 2017 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", tz="Europe/Paris")
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, proj4string=CRS())
```

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))
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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,
           proj4string=CRS())
```

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))
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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, proj4string=CRS())
```

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).
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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,
          proj4string=CRS())
```

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).
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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,
         proj4string=CRS())
```

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))
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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, proj4string=CRS())
```

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))
proj4string	a valid CRS object containing the projection information (see ?CRS from the package sp).

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

```
## Not run:
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")

## End(Not run)
```

---

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)
```

---

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")
```

---

testNM *Null Model Approach for Animal Movement Analysis*

Description

The functions `NMs.*` allow to define "single null models" (see details). The function `NMs2NMm` can be used on an object of class `"NMs"` to define "multiple null models". The function `testNM` can be used to simulate the defined null models.

Usage

```
NMs.randomCRW(ltraj, rangles = TRUE, rdist = TRUE, fixedStart = TRUE,
              x0 = NULL, rx = NULL, ry = NULL, treatment.func = NULL,
              treatment.par = NULL, constraint.func = NULL,
              constraint.par = NULL, nrep = 999)

NMs.randomShiftRotation(ltraj, rshift = TRUE, rrot = TRUE,
                        rx = NULL, ry = NULL, treatment.func = NULL,
                        treatment.par = NULL, constraint.func = NULL,
                        constraint.par = NULL, nrep = 999)


NMs.CRW(N = 1, nlocs = 100, rho = 0, h = 1, x0 = c(0,0),
        treatment.func = NULL,
        treatment.par = NULL, constraint.func = NULL,
        constraint.par = NULL, nrep = 999)


NMs.randomCs(ltraj, Cs = NULL, rDistCs = TRUE,
             rAngleCs = TRUE,
             rCentroidAngle = TRUE, rCs = TRUE,
             newCs = NULL, newDistances = NULL,
```

```
             treatment.func = NULL, treatment.par = NULL,
             constraint.func = NULL, constraint.par = NULL,
             nrep=999)

NMs2NMm(NMs, treatment.func = NULL,
        treatment.par = NULL, constraint.func = NULL,
        constraint.par = NULL, nrep = 999)

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

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

testNM(NM, count = TRUE)
```

Arguments

ltraj	an object of class "ltraj"
rangles	logical. Whether the turning angles should be randomized.
rdist	logical. Whether the distances between successive relocations should be randomized.
fixedStart	logical. If TRUE, the first location of the randomized trajectories corresponds to x0. If FALSE, the first location (x, y) is sampled in the interval (rx, ry)
x0	a vector of length 2 giving the x and y coordinates of the first relocations. If NULL and fixedStart=TRUE, the first location of the trajectory corresponds to the first location of the actual trajectory
rx	a vector of length 2 giving the x coordinates of the bounding box where the first location of the trajectory should be sampled
ry	a vector of length 2 giving the range (min, max) of the y coordinates of the bounding box where the first location of the trajectory should be sampled
treatment.func	any function taking two arguments x and par, where x is the trajectory generated by the function, and par can be any R object (e.g. a list containing the parameters needed by the function. Note that the argument par should be present in the function definition even if it is not needed in the function). See details and examples. If NULL, a function is defined internally that simply returns the raw trajectory simulated by the model
treatment.par	the R object that will be passed as an argument to the parameter par of the function treatment.func
constraint.func	any function taking two arguments x and par, where x is the trajectory generated by the function, and par can be any R object (e.g. a list containing the parameters needed by the function). **This function should necessarily return a logical value** (See details and examples). If NULL, a function is defined internally that always returns TRUE
constraint.par	The R object that will be passed as an argument to the parameter par of the function constraint.func

nrep	The number of repetitions of the null model
rshift	logical. Whether the trajectory should be shifted over the study area.
rrot	logical. Whether the trajectory should be rotated around its barycentre.
N	The number of animals to simulate.
nlocs	The number of relocations building up each trajectory
rho	The concentration parameter for wrapped normal distribution of turning angles (see ?simm.crw)
h	the scaling parameter for the movement length (see ?simm.crw)
Cs	a list of vectors of length 2. Each vector should contain the x and y coordinates of the capture sites of the animals in ltraj. This list should therefore have the same number of elements as ltraj
rDistCs	logical. Whether the distances between the barycentre of the trajectories and the corresponding capture sites should be randomized among trajectories
rAngleCs	logical. Whether the angle between the east direction and the line connecting the capture site and the barycentre of the trajectory should be drawn from an uniform distribution.
rCentroidAngle	logical. Whether the trajectory should be randomly rotated around its barycentre.
rCs	logical. Whether the trajectory should be randomly associated to a new capture site.
newCs	a list of vectors of length 2. Each vector should contain the x and y coordinates of new capture sites. If NULL and rcs=TRUE, the new capture sites are sampled in cs
newDistances	a vector of new distances that will be used to define the distances between capture sites and centroid of simulated trajectories if rDistCs=TRUE
NM,x	a null model of class "NMs" or "NMm"
NMs	a null model of class "NMs"
count	whether the iterations should be displayed
...	additionnal arguments to be passed to or from other methods

Details

The null model approach has been considered as a useful approach in many fields of ecology to study biological processes. According to Gotelli and Graves (1996), "A null model is a pattern-generating model that is based on randomization of ecological data or random sampling from a known or imagined distribution. The null model is designed with respect to some ecological or evolutionary process of interest. Certain elements of the data are held constant, and others are allowed to vary stochastically to create new assemblage patterns. The randomization is designed to produce a pattern that would be expected in the absence of a particular ecological mechanism".

This approach can be very useful to test hypotheses related to animal movements. The package adehabitatLT propose several general null models that can be used to test biological hypotheses. For example, imagine that we want to test the hypothesis that no habitat selection occurs when the animal moves. The shape of the trajectory, under this hypothesis would be the pure result

of changing activity (moving, foraging, resting). Therefore, a possible approach to test whether habitat selection actually occurs would be to randomly rotate the trajectory around its barycentre and shifting it over the study area. The function NMs.randomShiftRotation can be used to define such a model. It is possible to constrain the randomization by defining a "constraint function" (e.g. to keep only the randomized trajectories satisfying a given criterion). It is required to specify a "treatment function" (i.e., a function that will be applied to each randomized trajectory). Once the null model has been defined, it is then possible to perform the randomizations using the function testNM. We give below the details concerning the available null models, as well as the constraint and the treatment functions.

First, two types of null models can be defined: single (NMs) and multiple (NMm) null models. Consider an object of class "ltraj" containing N bursts. With NMs, the treatment function will be applied to each randomized burst of relocations. Thus, for example, if nrep repetitions of the null model are required, nrep repetitions of the null models will be carried out for each burst separately. The treatment function will be applied on each randomized burst. With NMm, for each repetition, N randomized bursts of relocations are generated. The treatment is then applied, for each repetition, to the whole set of N randomized bursts. Thus, NMs are useful to test hypothesis on each trajectory separately (e.g. individual habitat selection), whereas NMm are useful to test hypotheses relative to the whole set of animals stored in an object of class ltraj (e.g. interactions between animals). The only current way to define an object of class NMm is to first define an object of class NMs and then use the function NMs2NMm to indicate the treatment function that should be applied on the whole set of trajectories.

The constraint function should be user-defined. It should return a logical value indicating whether the constraint is satisfied. With NMs, this function should take only two parameters: x and par. The argument x is a data frame with three columns describing a trajectory (the X and Y coordinates, and the date as a vector of class "POSIXct"), and the argument par can be any R object required by the constraint function (e.g. if the constraint to keep 80% of the relocations of the randomized trajectories within a given habitat type, the parameter par can be a raster map, or a list of raster maps). With NMm, this function should also take only the two parameters x and par. The argument par can be any R object required by the constraint function. However, when "NMm" are defined, the argument x of the constraint function should be an object of class "ltraj". If the function NMs2NMm is used to define the object of class "NMm", two types of constraint can therefore be defined: at the individual level (in the function NMs.*) and for the whole set of animals (in the function NMs2NMm). In this case, some constraints will be satisfied at the individual level, and others at the scale of the whole set of animals. If no constraint function is defined by the user, a constraint function always returning TRUE is automatically defined internally.

The treatment function can be any function defined by the user, but should take two arguments x and par, identical to those passed to the constraint function (i.e., x should be a data frame with three columns for NMs and an object of class "ltraj" for NMm. Note that only one treatment function can be applied to the randomization: if NMs2NMm is used to define an object of class NMm, the treatment function defined in the function NMs.* will be ignored, and only the treatment function defined in the function NMs2NMm will be taken into account. If no treatment function is defined by the user, a treatment function will be defined internally, simply returning the randomized trajectory (i.e. a data.frame with three columns for NMs, and an object of class ltraj for NMm).

We now describe the list of available null models:

NMs.CRW: this model is a purely parametric model. It simulates a correlated random walk with specified parameters (see ?simm.crw for a complete description of this model).

NMs.randomCRW: this model also simulates a correlated random walk, but the distributions of the

turning angles and/or distances between successive relocations are derived from the trajectories passed as arguments. It is possible to randomize the turning angles, the distances between successive relocations, or both (default).

`NMs.randomShiftRotation`: this model randomly rotates the trajectory around its barycentre and randomly shifts it over the study area (but does not change its shape). The function allows for a random rotation, a random shift or both operations (default).

`NMs.randomCs`: this model is similar to the previous one: it keeps the shape of the trajectory unchanged. However, it randomizes the position of the trajectories with respect to a set of capture sites (it can be used to take into account the fact that a the home range of a sedentary animal captured at a given place is likely to be close to this place). First a capture site may be randomly drawn from a list of capture sites (either the actual capture sites or a list passed by the user). Then, the angle between the east direction and the line connecting the capture site of the animal and the barycentre of its trajectory is randomly drawn from a uniform distribution. Then, the distance between this barycentre and the capture site is randomly drawn from the observed distribution of distances between capture sites and trajectory barycentre (or from a set of distances passed as argument). Finally, the trajectory is randomly rotated around its barycentre.

Value

For objects of class `"NMs"` a list of `N` elements (where `N` is the number of trajectories in the object of class `"ltraj"` passed as argument, with each element is a list storing the `nrep` results of the treatment function applied to each randomized trajectory.

For objects of class `"NMm"`, a list of `nrep` elements, each element storing the result of the treatment function applied to each set of `N` randomized trajectories.

Author(s)

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

References

Gotelli, N. and Graves, G. (1996) *Null models in Ecology*. Smithsonian Institution Press.

Richard, E., Calenge, C., Said, S., Hamann, J.L. and Gaillard, J.M. (2012) Studying spatial interactions between sympatric populations of large herbivores: a null model approach. *Ecography*, in press.

See Also

`as.ltraj` for additional information about objects of class `"ltraj"`

Examples

```
## Not run:
########################################
##
## example using NMs.randomShiftRotation

## first load the data:
data(puechcirc)
```

```
data(puechabonsp)
map <- puechabonsp$map

## Consider the first animal
## on an elevation map
anim1 <- puechcirc[1]
plot(anim1, spixdf=map[,1])

## We define a very simple treatment function
## for a NMs model: it just plots the randomized trajectory
## over the study area
## As required, the function takes two arguments:
## x is a data.frame storing a randomized trajectory (three
## columns: the x, y coordinates and the date)
## par contains the map of the study area

myfunc <- function(x, par)
{
   par(mar = c(0,0,0,0))
   ## first plot the map
   image(par)

   ## then add the trajectory
   lines(x[,1], x[,2], lwd=2)
}

## Then we define the null model
##
## We define the range of the study area where the trajectory
## will be shifted:
rxy <- apply(coordinates(map),2,range)
rxy

## We define the null model with 9 repetitions
nmo <- NMs.randomShiftRotation(na.omit(anim1), rshift = TRUE, rrot = TRUE,
                               rx = rxy[,1], ry = rxy[,2], treatment.func = myfunc,
                               treatment.par = map, nrep=9)

## Then apply the null model
par(mfrow = c(3,3))
tmp <- testNM(nmo)

## You may try variations, by setting rshift or rrot to FALSE, to see
## the differences

## Note that some of the randomized trajectories are located outside the
## study area, although all barycentres are located within the X and Y
## limits of this study area.
## We may define a constraint function returning TRUE only if all
## relocations are located within the study area

## again, note that the two parameters are x and par
consfun <- function(x, par)
```

```
{
   ## first convert x to the class SpatialPointsDataFrame
   coordinates(x) <- x[,1:2]

   ## then use the function over from the package sp
   ## to check whether all points in x are located inside
   ## the study area
   ov <- over(x, geometry(map))
   return(all(!is.na(ov)))
}

## Now fit again the null model under these constraints:
nmo2 <- NMs.randomShiftRotation(na.omit(anim1), rshift = TRUE, rrot = TRUE,
                                rx = rxy[,1], ry = rxy[,2], treatment.func = myfunc,
                                treatment.par = map,
                                constraint.func = consfun,
                                constraint.par = map, nrep=9)

## Then apply the null model
par(mfrow = c(3,3))
tmp <- testNM(nmo2)

## all the relocations are now inside the study area.


########################################
##
## example using NMs.randomCRW


## We generate correlated random walks with the same starting
## point as the original trajectory, the same turning angle
## distribution, and the same distance between relocation
## distribution. We use the same constraint function as previously
## (all relocations falling within the study area), and we
## use the same treatment function as previously (just plot
## the result).
mo <- NMs.randomCRW(na.omit(anim1), rangles=TRUE, rdist=TRUE,
                    treatment.func = myfunc,
                    treatment.par = map, constraint.func=consfun,
                    constraint.par = map, nrep=9)

par(mfrow = c(3,3))
tmp <- testNM(mo)

## Now, try a different treatment function: e.g. measure
## the distance between the first and last relocation,
## to test whether the animal is performing a return trip
myfunc2 <- function(x, par)
{
    sqrt(sum(unlist(x[1,1:2] - x[nrow(x),1:2])^2))
}
```

```
## Now fit again the null model with this new treatment and 499 repetitions:
mo2 <- NMs.randomCRW(na.omit(anim1), rangles=TRUE, rdist=TRUE,
                     treatment.func = myfunc2,
                     treatment.par = map, constraint.func=consfun,
                     constraint.par = map, nrep=499)

## Then apply the null model
set.seed(298) ## to make the calculation reproducible
rand <- testNM(mo2)

## rand is a list with one element (there is one trajectory in anim1).
length(rand[[1]])

## The first element of rand is a list of length 499 (there are 499
## randomizations).
head(rand[[1]])

## unlist this list:
rand2 <- unlist(rand[[1]])

## calculate the observed average elevation:
obs <- myfunc2(na.omit(anim1)[[1]][,1:3], map)

## and performs a randomization test:
(rt <- as.randtest(rand2, obs, alter="less"))
plot(rt)

## Comparing to a model where the animal is moving randomly, and based
## on the chosen criterion (distance between the first and last
## relocation), we can see that the distance between the first and last
## relocation is rarely observed.  It seems to indicate that the animal
## tends to perform a loop.



########################################
##
## example using NMs2NMm

## Given the previous results, we may try to see if all
## the trajectories in puechcirc are characterized by return
## trips
## We need a NMm approach. Because we have 3 burst in puechcirc
## we need a summary criterion. For example, the mean
## distance between the first and last relocation.

## We program a treatment function: it also takes two arguments, but x
## is now an object of class "ltraj" !
## par is needed, but will not be used in the function

myfunm <- function(x, par)
{
   di <- unlist(lapply(x, function(y) {
```

```
      sqrt(sum(unlist(y[1,1:2] - y[nrow(y),1:2])^2))
   }))
   return(mean(di))
}

## Now, prepare the NMs object: we do not indicate any treatment
## function (it would not be taken into account when NMs would be
## transformed to NMm).  However, we keep the constraint function
## the simulated trajectories should fall within the study area
mo2s <- NMs.randomCRW(na.omit(puechcirc), constraint.func=consfun,
                      constraint.par = map)

## We convert this object to NMm, and we pass the treatment function
mo2m <- NMs2NMm(mo2s, treatment.func = myfunm, nrep=499)

## and we fit the model
set.seed(908)
resu <- testNM(mo2m)

## We calculate the observed mean distance between the
## first and last relocation
obs <- myfunm(na.omit(puechcirc))

## and performs a randomization test:
(rt <- as.randtest(unlist(resu), obs, alter="less"))
plot(rt)

## The test is no longer significant


########################################
##
## example using NMs.randomCs

## Consider this sample of 5 capture sites:
cs <- list(c(701184, 3161020), c(700164, 3160473),
           c(698797, 3159908), c(699034, 3158559),
           c(701020, 3159489))
image(map)
lapply(cs, function(x) points(x[1], x[2], pch=16))

## Consider this sample of distances:
dist <- c(100, 200, 150)

## change the treatment function so that the capture sites are showed as
## well. Now, par is a list with two elements: the first one is the map
## and the second one is the list of capture sites

myfunc <- function(x, par)
{
   par(mar = c(0,0,0,0))
   ## first plot the map
   image(par[[1]])
```

```
   lapply(par[[2]], function(x) points(x[1], x[2], pch=16))

   ## then add the trajectory
   lines(x[,1], x[,2], lwd=2)
}


## Now define the null model, with the same constraints
## and treatment as before
mod <- NMs.randomCs(na.omit(anim1), newCs=cs, newDistances=dist,
                    treatment.func=myfunc, treatment.par=list(map, cs),
                    constraint.func=consfun, constraint.par=map,
                    nrep=9)

## apply the null model
par(mfrow = c(3,3))
tmp <- testNM(mod)

## End(Not run)
```

`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(puechabonsp)
trajdyn(puechcirc, spixdf = puechabonsp$map)


## 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)
```

---

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, a vector containing the successive observations building the series. For wawotest.ltraj, an object of class ltraj.
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` (or the name of a variable in the optional attribute `infolocs`)

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),]
```

Index