Package ‘movementAnalysis’

April 18, 2013

Version 0.6
Date 2013/04/18
Depends R (>= 2.14.0), adehabitat, adehabitatLT, sp, maptools, lmomco
Suggests OpenCL (>= 0.1.3.1)
Title Analysis of trajectory data using linear or Brownian motion model
Author Stef Sijben
Description Analysis of trajectory data. Contains functions for detecting movement patterns using different movement models, including the linear motion model and the Brownian bridge motion model (BBMM).
Maintainer Stef Sijben <s.m.a.sijben@student.tue.nl>
License GPL (>= 3)

R topics documented:

barplot.encounterDuration .................................................. 2
bbtraj ................................................................. 3
burst-statistics .......................................................... 5
contourPolygons .......................................................... 6
direction ............................................................... 7
distance ............................................................... 8
encounterDistribution ..................................................... 10
encounterDuration ......................................................... 11
monkey.data ............................................................. 13
monkey.tr ............................................................. 13
plot.utilizationDistribution ................................................ 14
position ............................................................... 15
speed ................................................................. 16
speedDistribution ........................................................ 17
utilizationDistribution .................................................... 18
velocity ............................................................... 20

Index 21
barplot.encounterDuration

*Draw a barplot representing the duration of encounters.*

**Description**

Given the result from a call to `encounterDuration`, this function draws a barplot. The plot contains one group of bars for each pair of IDs or bursts in the data (depending on the type of data). Each group contains a bar for each model for which the encounter duration was computed.

**Usage**

```r
## S3 method for class 'encounterDuration'
barplot(height, col = NULL, units="auto", ...)
```

**Arguments**

- `height`: The object containing the duration of encounters.
- `col`: The colours to use in the plot. By default, takes colours with equally spaced hues in the `hcl` space.
- `units`: The unit used on the y axis of the plot. Must be a unit accepted by `difftime`.
- `...`: Other graphical parameters, e.g. from `barplot.default`.

**Details**

If `height` is an array, i.e. it was the result from a call to `encounterDuration` with `byburst=FALSE`, the name displayed under each pair of bars are the names of the IDs involved, separated by a hyphen. Otherwise, you may specify the labels yourself in the row names of `height` (which is a `data.frame`). If all labels are numeric, this function replaces them by the names of the bursts involved, separated by a hyphen.

**Value**

The return value is the same as what `barplot.default` returns, i.e. a vector or matrix indicating the coordinates of bar midpoints.

**Examples**

```r
data(" vervet_monkeys", package="movementAnalysis")
ed <- encounterDuration(monkey.tr, 1/zero.noslash/zero.noslash, byburst=TRUE)
barplot(ed)
```
bbtraj

Class for storing Brownian bridge trajectories

Description

The class bbtraj stores trajectories of animals, similar to the class ltraj in package adehabitatLT. The difference is that a bbtraj object also stores the variances used when dealing with Brownian bridges. When converting a data set to an object of this class (using the function as.bbtraj) the most likely value for the diffusion coefficient is calculated and stored in the resulting object. na.omit filters the missing measurements from a trajectory, removing bursts that are empty after this filtering. All information that is derived from multiple measurements is recomputed. bbfilterNA is a deprecated alias for na.omit.

Usage

as.bbtraj(xys, date, id, burst = id, typeII = TRUE,
      slsp = c("remove", "missing"),
      infolocs = data.frame(pkey = paste(id, date, sep=".")),
      make.regular=FALSE)
diffusionCoefficient(tr, byburst = FALSE, nsteps = 1/zero.noslash/zero.noslash/zero.noslash)
## S3 method for class 'bbtraj'
na.omit(object, slsp=c("remove", "missing"), ...)
## S3 method for class 'bbtraj'
split(x, f=as.factor(sapply(x, function(b) { attr(b, "id") })), drop = FALSE, ...)

Arguments

xys a data.frame containing the x and y coordinates and variance 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 (for details, see as.ltraj)
infolocs If not NULL, a data.frame containing extra information about the relocations.
make.regular Either FALSE, or a list specifying the arguments to setNA and set0.
tr A trajectory of class ltraj or bbtraj
byburst A logical indicating whether a different diffusion coefficient should be estimated for each burst.
nsteps The number of different values for the diffusion coefficient to test. The range in which the value can be is determined automatically.

object The trajectory from which to filter missing values.

... Further arguments that might be required by specialized methods.

x The trajectory to split.

f A factor providing the destination group for each burst.

drop Logical indicating if levels that do not occur should be dropped.

Details

If make.regular is not FALSE, it must be a list containing parameters accepted by setNA and sett0, by the same names or in the proper order. as these functions are used to make the resulting trajectory regular.

diffusionCoefficient can be used to estimate the diffusion coefficient for an already existing bbtraj object. When calling as.bbtraj, it is invoked automatically. However, one might want to re-estimate the diffusion coefficient after modifying some values or constructing a trajectory from scratch.

Value

An object of class bbtraj is a list with one component per burst of relocations. Each component is a data frame with two attributes: the attribute "id" indicates the identity of the animal, and the attribute "burst" indicates the identity of the burst. Each data frame stores the following columns:

x the x coordinate for each relocation

y the y coordinate for each relocation

diff.coeff The diffusion coefficient for the move. This has no meaning if the next relocation is missing, but usually it is set to the same value for many relocations.

loc.var The variance of the measured position

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.

diffusionCoefficient returns a vector containing the optimal values for each ID or burst.
References


See Also

`ltraj`

Examples

```r
data("vervet_monkeys", package="movementAnalysis")
monkey.data

tr <- as.bbtraj(data.frame(x=monkey.data$X, y=monkey.data$Y, var=monkey.data$StdDev^2),
date=monkey.data$DateTime, id=monkey.data$GroupID, burst=monkey.data$GroupDayNo)
tr
is.regular(tr)

# Now, make the trajectory regular while constructing it
tr.reg <- as.bbtraj(data.frame(x=monkey.data$X, y=monkey.data$Y, var=monkey.data$StdDev^2),
date=monkey.data$DateTime, id=monkey.data$GroupID, burst=monkey.data$GroupDayNo,
    make.regular=list(date.ref=strptime("/zero.noslash/zero.noslash:/zero.noslash/zero.noslash", "%H:%M"),
    dt=1, units="hour"))
is.regular(tr.reg)

# Extract a single burst from the trajectory
tr["BD_1417"]
```

burst-statistics

Collect various statistics about individual bursts

Description

These functions compute a certain statistic of every burst in a given trajectory. The statistics are net displacement distance, direction of net displacement and day journey length (D JL), which is the total length of all relocations.

Usage

- `displacement.distance(tr)`
- `displacement.direction(tr, units = c("radian", "degree", "compass"))`
- `djl(tr)`

Arguments

- **tr** The trajectory object
- **units** The units in which to return the result
contourPolygons

Details
All functions use the linear motion model to compute the statistics.
displacement.direction can return its result in different units. The default is in radians from the positive $x$ axis, but it can also report degrees from this axis or compass heading (i.e. degrees from north, measured clockwise).

Value
A vector containing the requested statistic for each burst, indexed by the burst name.

See Also
bbtraj

Examples
data("vervet_monkeys", package="movementAnalysis")
djl(monkey.tr)
displacement.direction(monkey.tr, units="compass")

contourPolygons

Description
Given a utilization distribution, compute the polygons that contain a certain fraction of the UD.

Usage
contourPolygons(ud, levels = 0.99)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ud</td>
<td>The utilization distribution.</td>
</tr>
<tr>
<td>levels</td>
<td>The level(s) at which to compute contours.</td>
</tr>
</tbody>
</table>

Value
A list of SpatialPolygons as defined in package sp. Each element of the list represents the result for one utilization distribution from ud. If multiple levels are requested, these are separate Polygons in one element of the result list.

See Also
utilizationDistribution
direction

Examples

```r
data("vervet_monkeys", package="movementAnalysis")
ud <- utilizationDistribution(monkey.tr)
cp <- contourPolygons(ud, levels=c(0.95, 0.99))
str(cp[[1]])

# Draw the home range for the first animal
plot(cp[[1]], col=1:2)
```

---

direction

Compute the (distribution of) movement direction.

Description

Direction (in the linear motion model) and density of the direction distribution (in the BBMM) over specific time intervals.

Usage

```r
direction(tr, time, time.scale = NA)
direction(d, tr, time, time.scale = NA)
```

Arguments

- `tr` The trajectory object
- `time` A vector of times at which to report the direction
- `time.scale` The length of the intervals over which to report the direction
- `d` The direction for which to compute the density
- `lower` The lower end of the interval for the CDF

Details

The direction at a time \( t \) is defined as the direction of the net displacement over the time interval \([t - \text{time.scale}/2, t + \text{time.scale}/2]\). Directions are in radians from the positive \( x \) axis.

Since direction has a circular distribution, the distribution function has to be computed with respect to some base value. Using the `lower` argument, one can change this base value. Thus, when calling `ddirection(x, ..., lower=1)`, the density function will be integrated from 1 to \( x \). The latter may be a vector.

Value

direction returns a matrix indexed by the ID in the trajectory and time. ddirection returns an array indexed by ID, the requested direction and time.
References

See Also
velocity, bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the (linear model) direction over two half-hour intervals
d <- direction(monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")), 1800)
d

# Compute the density of four directions over the same time intervals
ddirection(d, monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")), 1800)

Description
Density, distribution function, quantile function for the distances between entities at the given time(s), using the Brownian bridge movement model. Distance between entities and whether encounters occur in the linear movement model.

Usage

distance(tr, time)

ddistance(d, tr, time)
pdistance(d, tr, time)
qdistance(p, tr, time)

encounter(d, tr, time)
encounterIntervals(d, tr)

Arguments

d Vector of distances
p Vector of probabilities
tr The trajectory object
time Vector of times
**distance**

**Details**

encounterIntervals accepts only a single distance.

**Value**

distance returns a three dimensional array indexed by two IDs and the time, each value representing the distance in the linear movement model between these two IDs at that time.

encounter returns whether an encounter occurs for each value of \(d\) and for each of the requested times. The value is computed between each pair of IDs in the trajectory. This means that the result is a 4 dimensional array, indexed by the IDs involved, the value of \(d\) and the time.

encounterIntervals computes the time intervals in which there is an encounter between each pair of IDs. In the linear model this can be solved analytically, which is often faster than requesting whether an encounter occurs for many different times.

distance computes the density, pdistance the distribution function and qdistance evaluates the quantile function for the requested parameters. pdistance is equivalent to the probability that an encounter occurs, given a threshold distance.

The functions return the requested values for each value of the first parameter (\(d\) or \(p\)) and for each of the requested times. The value is computed between each pair of IDs in the trajectory.

This means that the result is a 4 dimensional array, indexed by the IDs involved, the value of \(d\) or \(p\) and the time.

**See Also**

bbtraj

**Examples**

data("vervet_monkeys", package="movementAnalysis")

# Compute the distance at two distinct times
distance(monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")))

# See whether an encounter occurs at the 50m or 100m thresholds
encounter(c(10, 100), monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")))

# Compute when there are encounters
ei <- encounterIntervals(100, monkey.tr)
ei[["BD","NH"]]

# Compute the 5th and 95th percentile of the distance at two distinct times
qdistance(c(0.05, 0.95), monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")))
encounterDistribution  Spatial distribution of encounters

Description
Computes the expected duration of encounters at each location for every pair of IDs.

Usage
encounterDistribution(tr, threshold, grid=NULL, timestepSize = 60, xC=NULL, yC=NULL, grid.dim = 100, grid.pad = 0.2)

Arguments
- **tr**: The trajectory for which to compute the UD
- **threshold**: The maximum distance at which an encounter occurs
- **grid**: a grid of class asc
- **timestepSize**: The difference between consecutive time steps, in seconds
- **xC**: The x coordinates of the vertical grid lines
- **yC**: The y coordinates of the horizontal grid lines
- **grid.dim**: If all of grid, xC and yC are omitted, this specifies the dimension of the shortest axis of the grid used.
- **grid.pad**: If the grid is automatically generated, its range is the range of the relocations extended grid.pad times the length of the range on each side.

Details
One can specify the grid in three ways:

- If grid is set, the coordinates of the grid lines are derived from there.
- These coordinates can also be specified in the parameters xC and yC.
- Otherwise, the grid is determined automatically. The grid then ranges over a bounding box of all measurements, extended on each side by a specified fraction of the range. The number of grid cells can be controlled via grid.dim. The amount by which the grid is extended is controlled via grid.pad, which may be a vector specifying the extension on top, right, bottom, left respectively. It is recycled as usual.

Value
The return value is a list, indexed by two IDs. If grid is given, each element of the result list is an object of class asc, representing the same grid. Otherwise, each element of the list is a matrix, indexed by the coordinates specified in xC and yC.

Element result["id1", "id2"] of the result represents the distribution of the position of id1 while it had encounters with id2. This is not the same as result["id2", "id1"], since that is the distribution of id2’s location during its encounters with id1.
The diagonal entries of the result list contain the utilization distribution of each ID, since an entity is always at a distance 0 from itself.

Warning

There seems to be some problem with the result being transposed, this needs further investigation. Until then, you can plot the transpose of the result using `image(t(ud["BD","NH"]))`.

Note

The `image` function has ugly colours, use the `col` attribute to define a better colour map.

Also note that this function may take a rather long time to complete, so please be patient, specify a sufficiently small grid or use a larger time step.

Examples

```r
# Define grid lines: equally spaced between the min and max coordinate in monkey.tr
xlim <- range(unlist(sapply(monkey.tr, function(b) { b$x })), na.rm=TRUE)
xc <- seq(xlim[1], xlim[2], length.out=20)

ylim <- range(unlist(sapply(monkey.tr, function(b) { b$y })), na.rm=TRUE)
yc <- seq(ylim[1], ylim[2], length.out=20)

# Compute the UD and plot the result for one ID
ud <- encounterDistribution(monkey.tr, 1/zero.noslash/zero.noslash, xc=xc, yc=yc)
image(ud["BD","NH"])```

Description

Computes the duration of encounters between each pair of groups over the whole measurement period of a trajectory. The user can select what movement model(s) to apply and whether the result should be on the level of bursts or IDs. `encounterDurationById` returns the duration of encounters for each pair of IDs, given the duration of encounters for each pair of bursts. This allows to obtain both types of result without recomputing everything.

Usage

```r
encounterDuration(tr, threshold, model = c("BBMM","linear"), byburst = FALSE, timestepSize = 60)
encounterDurationById(encounterDurationByBurst)```
encounterDuration

Arguments

tr The trajectory to analyze
threshold The maximum distance at which an encounter is detected
model The movement models for which to compute the encounter duration
byburst If TRUE, the result contains durations for pairs of bursts. If FALSE, the result contains encounter durations between pairs of IDs.
timestepSize If a movement model requires numerical integration in the time dimension, this is the size of each time step.

encounterDurationByBurst

The result of calling encounterDuration with byburst=TRUE.

Details

Since the duration of encounters is a random variable in the Brownian bridge movement model, this function cannot give exact results there. Instead, it reports the expected duration of encounters in the BBMM.

If you already have the encounter duration between bursts and you also want the encounter duration between IDs, you should use encounterDurationById instead of calling encounterDuration again with different parameters. They give identical results, but the former is much faster since it does not recompute all relevant durations.

Value

If byburst == FALSE, the result is a data.frame with one row for each interesting pair of bursts. A pair of bursts is interesting if they overlap in time, since otherwise the duration is always zero. The result contains the following fields:

id1,id2 The IDs of the bursts involved
burst1,burst2 The names of the bursts involved

In addition there is one column for each model requested, named after the model. These columns are of type difftime, with unit secs. They contain the encounter duration according to that model.

If byburst == TRUE, the result is a 3 dimensional array, indexed by the two IDs and the movement model. The entries in the array are again of class difftime with unit secs.

See Also

bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")
ed <- encounterDuration(monkey.tr, 1/zero.noslash/zero.noslash, byburst=TRUE)
barplot(ed)
barplot(encounterDurationById(ed))
monkey.data

# This one has the same result, but needs to recomputes everything
barplot(encounterDuration(monkey.tr, 100))

---

**monkey.data**

**Description**

This data set contains GPS data obtained from two vervet monkeys over two days. The data is sampled at one sample per hour from 5:00 to 19:00, although there are missing values.

**Usage**

```r
data(vervet_monkeys)
```

**Format**

A data.frame containing the raw data. The fields are:

- **GroupID** The identifier of the monkey.
- **GroupDayNo** The identifier of the monkey and the day the measurement was obtained.
- **DateTime** The time of the measurement.
- **X** X coordinate of the location.
- **Y** Y coordinate of the location.
- **StdDev** The standard deviation of the location error.

**Examples**

```r
data(vervet_monkeys)
head(monkey.data)
```

---

**monkey.tr**

**Description**

This data set contains GPS data obtained from two vervet monkeys over two days. The data is sampled at one sample per hour from 5:00 to 19:00, although there are missing values.

**Usage**

```r
data(vervet_monkeys)
```
**Format**

An object of type `bbtraj` created from the data in `monkey.data`, containing four bursts, one for each `(monkey, day)` pair.

**Examples**

```r
data( vervet_monkeys)
monkey.tr
plot(monkey.tr)
```

---

**plot.utilizationDistribution**

*Plot or draw contours for utilization distributions.*

---

**Description**

Plots the given utilization distribution, either by applying a colourmap to the values or by drawing contours where the value passes certain levels.

**Usage**

```r
## S3 method for class 'utilizationDistribution'
plot(x, col = hcl(1:length(x) * 360/length(x), 50, 70),
     xlim = NULL, ylim = NULL, add = FALSE, ...)
## S3 method for class 'utilizationDistribution'
contour(x, levels = 0.99, col = hcl(1:length(x) * 360/length(x), l=35),
     xlim = NULL, ylim = NULL, labels = levels, add = FALSE, ...)
```

**Arguments**

- `x` The utilization distribution to plot.
- `col` The colours to use in the plot. May be a vector of colours or a list containing the colour scheme for each ID.
- `xlim` The minimum and maximum x coordinate to plot.
- `ylim` The minimum and maximum y coordinate to plot.
- `add` Logical specifying if plot should be added to an already existing plot; defaults to FALSE.
- `levels` The level(s) at which to draw contours.
- `labels` The labels to print with the contours. Use `drawlabels=FALSE` to hide them altogether.
- `...` Other graphical parameters, e.g. from `plot.default` or `contour.default`. 
Details

If \( \text{col} \) is a vector, it is assumed to contain a colour code for each grid to be plotted. The colour code for each value of the UD is derived by varying the opacity of the provided colour. If the length of \( \text{col} \) does not match the number of UDs to plot, it is recycled as usual.

If \( \text{col} \) is a list, it is assumed to contain one element for each UD to plot, specifying exactly which colours to use in that plot.

See Also

utilizationDistribution, SpatialPolygons

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the UD and plot the result for one ID
ud <- utilizationDistribution(monkey.tr)
plot(ud)

# Specify nicer colours
plot(ud, col=c("red","green"))
# Add a contour to the existing plot in the same colours
contour(ud, col=c("red","green"), add=TRUE)
Value

An array indexed by ID, parameter name, time.

See Also

bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")

position(monkey.tr, as.POSIXct(c("2011-01-19 08:30:00")))

# , , 1295422200
# # x y var
# # BD 322487.6 6900484 1133.653
# # NH 323012.9 6900720 1192.534

| speed | Get statistics about speed |

Description

Get speed over any time intervals in the linear model. Get density, distribution function, quantiles and expected value in BBMM.

Usage

speed(tr, time, time.scale = NA)
mu.speed(tr, time, time.scale = NA)

pspeed(v, tr, time, time.scale = NA)
dspeed(v, tr, time, time.scale = NA)
qspeed(p, tr, time, time.scale = NA)

Arguments

v Vector of speeds
p Vector of probabilities
tr The trajectory object
time Vector of times
time.scale The length of the intervals over which to report the speed or direction

Details

The speed at a time \( t \) is defined as the absolute value of the \texttt{velocity} over the time interval \([t - time.scale/2, t + time.scale/2]\).
speedDistribution

Value

speed and mu.speed return a matrix indexed by the ID in the trajectory and time. dspeed, pspeed and qspeed return an array indexed by ID, the requested speed or probability and time.

See Also

velocity, bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the (linear model) speed over two half-hour intervals
v <- speed(monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")), 1800)
v

# Compute the density of four speeds over the same time intervals
ds(speed(v, monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")), 1800)

---

speedDistribution Compute average speed at each location

Description

For each grid cell, this function computes the average speed over time conditioned on the trajectory being in that cell.

Usage

speedDistribution(tr, grid = NULL, timestepSize = 60, time.scale = timestepSize, xc = NULL, yc = NULL, grid.dim = NULL, grid.pad = NULL)

Arguments

tr The trajectory for which to compute the UD
grid a grid of class asc
timestepSize The difference between consecutive time steps, in seconds.
time.scale The length of the intervals over which to compute the speed.
xc The x coordinates of the vertical grid lines
yc The y coordinates of the horizontal grid lines
grid.dim If all of grid, xc and yc are omitted, this specifies the dimension of the shortest axis of the grid used.
grid.pad If the grid is automatically generated, its range is the range of the relocations extended grid.pad times the length of the range on each side.
utilizationDistribution

Value

Returns a list, indexed by the IDs in tr. If grid is given, each element of the result list is an object of class asc, representing the same grid. Otherwise, each element of the list is a matrix, indexed by the coordinates specified in xc and yc, or by the coordinates computed from tr.

See Also

velocity, speed, bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the speed distribution and plot the result
sd <- speedDistribution(monkey.tr)
image(sd[[1]]) # Plot only the distribution of the first entity

utilizationDistribution

Compute utilization distribution for a trajectory

Description

This function computes the utilization distribution (UD) for each of the IDs present in the trajectory. The user may specify the grid on which the UD is evaluated and may also specify the size of the time step in the numerical integration.

Usage

utilizationDistribution(tr, grid=NULL, timestepSize = 6/zero.noslash, xc=NULL, yc=NULL, grid.dim = 1/zero.noslash/zero.noslash, grid.pad = /zero.noslash.2, byburst = FALSE, cutoff.level = 1)

utilizationDistribution.byID(ud)

Arguments

tr          The trajectory for which to compute the UD
grid       a grid of class asc
timestepSize The difference between consecutive time steps, in seconds
xc          The x coordinates of the vertical grid lines
yc          The y coordinates of the horizontal grid lines
grid.dim    If all of grid, xc and yc are omitted, this specifies the dimension of the shortest axis of the grid used.
grid.pad    If the grid is automatically generated, its range is the range of the relocations extended grid.pad times the length of the range on each side.
byburst  Whether the result should be a UD for each burst.
cutoff.level  What fraction of the UD volume must be actually stored.
ud  The result of utilizationDistribution with byburst=TRUE

Details

One can specify the grid in three ways:

- If grid is set, the coordinates of the grid lines are derived from there.
- These coordinates can also be specified in the parameters xc and yc.
- Otherwise, the grid is determined automatically. The grid then ranges over a bounding box of all measurements, extended on each side by a specified fraction of the range. The number of grid cells can be controlled via grid.dim. The amount by which the grid is extended is controlled via grid.pad, which may be a vector specifying the extension on top,right,bottom,left respectively. It is recycled as usual.

If cutoff.level is set to a value less than 1, the result will not store a utilization distribution for the full grid, but instead an axis-aligned box that contains the full contour at the requested level.

When a UD is extracted from a list containing such partial UDs, the cells that were left out are restored and filled with zeros.

utilizationDistribution.byID can be used to convert the result of utilizationDistribution with byburst=TRUE to a result that is aggregated by ID.

Value

Returns a list, indexed by the IDs in tr. If grid is given, each element of the result list is an object of class asc, representing the same grid. Otherwise, each element of the list is a matrix, indexed by the coordinates specified in xc and yc.

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the UD and plot the result
ud <- utilizationDistribution(monkey.tr, byburst=TRUE)
plot(ud)

# Convert to one UD for each group
ud <- utilizationDistribution.byID(ud)
plot(ud)
velocity  

Compute distribution parameters for velocity

Description

The velocity over any time interval is a bivariate normal distribution. This function computes the parameters for this distribution. The velocity in the linear model is identical to the mean velocity, so ignore the variance in that case.

Usage

velocity(tr, time, time.scale = NA)

Arguments

- **tr**: The trajectory object
- **time**: Vector of times
- **time.scale**: The length of the intervals over which to report the direction

Details

The velocity at a time \( t \) is defined as the net rate of displacement over the time interval \([t - time.scale/2, t + time.scale/2]\), i.e.

\[
v(t, \Delta t) = \frac{x(t + \Delta t/2) - x(t - \Delta t/2)}{\Delta t}.
\]

Value

Returns a three dimensional array indexed by ID in the trajectory, variable name and time. The variables are named \( x \), \( y \) and \( var \) for the \( x \) and \( y \) components of the mean and the variance respectively.

See Also

bbtraj

Examples

data("vervet_monkeys", package="movementAnalysis")

# Compute the velocity distribution parameters over two half-hour intervals
velocity(monkey.tr, as.POSIXct(c("2011-01-18 15:15:15", "2011-01-19 16:30:00")), 1800)
Index

*Topic **datasets**
  monkey.data, 13
  monkey.tr, 13
  [.bbtraj (bbtraj), 3
  [<-.[.bbtraj (bbtraj), 3
  [[].utilizationDistribution
    (utilizationDistribution), 18
  as.bbtraj (bbtraj), 3
  as.ltraj, 3
  asc, 10, 17, 18
  barplot.default, 2
  barplot.encounterDuration, 2
  bbFilterNA (bbtraj), 3
  bbtraj, 3, 6, 8, 9, 12, 14–18, 20
  burst-statistics, 5
  contour.default, 14
  contour.utilizationDistribution
    (plot.utilizationDistribution), 14
  contourPolygons, 6
  ddirection (direction), 7
  ddistance (distance), 8
  difftime, 2, 12
  diffusionCoefficient (bbtraj), 3
  direction, 7
  displacement.direction
    (burst-statistics), 5
  displacement.distance
    (burst-statistics), 5
  distance, 8
  djl (burst-statistics), 5
  dspeed (speed), 16
  encounter (distance), 8
  encounterDistribution, 10
  encounterDuration, 2, 11
  encounterDurationById
    (encounterDuration), 11
  encounterIntervals (distance), 8
  graphical parameters, 2, 14
  hcl, 2
  image, 11
  ltraj, 3, 5
  monkey.data, 13, 14
  monkey.tr, 13
  mu.speed (speed), 16
  na.omit.bbtraj (bbtraj), 3
  pdirection (direction), 7
  pdistance (distance), 8
  plot.default, 14
  plot.utilizationDistribution, 14
  Polygons, 6
  position, 15
  POSIXct, 15
  pspeed (speed), 16
  qdistance (distance), 8
  qspeed (speed), 16
  setNA, 3
  sett0, 3
  SpatialPolygons, 6, 15
  speed, 16, 18
  speedDistribution, 17
  split.bbtraj (bbtraj), 3
  summary.bbtraj (bbtraj), 3
  utilizationDistribution, 6, 15, 18
  velocity, 8, 16–18, 20