Package 'move'

August 23, 2016

Type Package

Title Visualizing and Analyzing Animal Track Data

Version 2.1.0

Date 2016-08-22

Author Bart Kranstauber <bart.kranstauber@ieu.uzh.ch>, Marco Smolla <marco.smolla@postgrad.manchester.ac.uk>

Maintainer Bart Kranstauber <bart.kranstauber@ieu.uzh.ch>

Description Contains functions to access movement data stored in 'movebank.org' as well as tools to visualize and statistically analyze animal movement data, among others functions to calculate dynamic Brownian Bridge Movement Models. Move helps addressing movement ecology questions.

License GPL (>= 3)

URL http://computational-ecology.com/main-move.html

BugReports https://gitlab.com/bartk/move/issues

LazyLoad yes

LazyData yes

LazyDataCompression xz

Depends geosphere (>= 1.4-3), methods, sp, raster (>= 2.4-15), rgdal, R (>= 2.15.0)

Suggests adehabitatHR, adehabitatLT, circular, ggmap, mapproj, maptools, testthat

Imports httr, Rcpp

LinkingTo Rcpp

SystemRequirements C++11

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-08-23 15:25:13

R topics documented:

`move-package` *An overview of the functions in this package*

Description

move is a package that contains functions to access movement data stored at www.movebank.org as well as tools to visualize and statistically analyse animal movement data. Move addresses movement ecological questions.

Details

The package implements classes for movement data and supports

- Creation of Move objects (see Move-class) representing animals and their track
- Calculation of utilization distributions using the dynamic Brownian bridge Movement Model
- Plotting tracks, utilization distributions and contours

- Access to raster, n.col, projection and coordinates
- Different CRS projection methods such as longlat or aeqd

I. Creating Move objects

Move objects can be created from files with the function:

`move` To create an object containing one animal track
`moveStack` To create an object containing multiple move objects
`getMovebankData` To create a Move or a MoveStack object with data from Movebank

II. Calculation of the utilization distribution

With the function below the dynamic Brownian Bridge Movement Model calculates the utilization density from a Move object:

`brownian.bridge.dyn` To calculate the utilization density

III. Accessing values

`coordinates`	Track-coordinates of the Move Object
`as.data.frame`	A data.frame with the important data of the Move Object
`n.locs`	The number of locations
`timeLag`	The time lags between the locations
`projection`	The projection method of the track/raster

IV. Plotting data

The track or the utilization distribution can be plotted with the following functions:

`plot`	plots the utilization distribution with fixed width and height ratio (see DBBMM-class) or the track (see Move-class)
`image`	plots the utilization distribution fitted to the window
`contour`	adds the contours of utilization distribution to a plot

Author(s)

Bart Kranstauber, Marco Smolla

Maintainer: Bart Kranstauber <bart.kranstauber@uni-konstanz.de>, Marco Smolla < marco.smolla@postgrad.manch

References

Move package vignette

move on CRAN

`.UD-class` *The UD class*

Description

This Class represents a simple abstraction of the utilization distribution, UD, where all probabilities necessarily sum to one. It is exported for experienced user to program against.

Slots

crs part of the Raster-class

data part of the Raster-class

extent part of the Raster-class

file part of the Raster-class

history part of the Raster-class

names part of the Raster-class

legend part of the Raster-class

method stores the method that was used to calculate the utilization distribution (UD), e.g. dynamic Brwonian Bridge

ncols part of the Raster-class

nrows part of the Raster-class

rotated part of the Raster-class

rotation part of the Raster-class

title part of the Raster-class

z part of the Raster-class

Author(s)

Bart Kranstauber

.unUsedRecords *.unUsedRecords and .unUsedRecordsStack class*

Description

The class .unUsedRecords and .unUsedRecordsStack is mostly an internal class that is made public to make inheritance easier. It is a basal class that stores unused records.

Slots

timestampsUnUsedRecords unused timestamps

sensorUnUsedRecords unused sensor information

dataUnUsedRecords further unused data

Author(s)

Marco Smolla

angle *angle information from a track or track stack*

Description

This function returns a summary about angle related measurements of a track or track stack. These are: average azimuth, variance of azimuth, standard error of azimuth.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
angle(x)
  ## S4 method for signature '.MoveTrackStack'
angle(x)
  ## S4 method for signature '.MoveTrackSingle'
angleSummary(x)
  ## S4 method for signature '.MoveTrackStack'
angleSummary(x)
```

Arguments

x	Move or MoveStack object

Value

Angles in degrees

Author(s)

Marco Smolla

Examples

```
## Not run:
data(leroy)
data(fishers)
  angle(leroy) #angles from a Move object
  angle(fishers) #angles from a MoveStack object
#  angleSummary(leroy) # summary of angle measures of a Move object
#  angleSummary(fishers) # summary of angle measures of a MoveStack object

## End(Not run)# failed on cran build
```

---

as.data.frame *Return a Data Frame*

Description

Function to create a data.frame with the information of a spatial data frame contained in the Move object.

Usage

```
  ## S4 method for signature 'Move'
as.data.frame(x,...)
```

Arguments

x	an object of the Move-class
...	additional arguments to be passed to or from methods

Details

as.data.frame extracts the sdf argument from a Move object (see Move-class)

Author(s)

Marco Smolla

Examples

```
## create a move object
data(leroy)
data <- leroy[99:150,]

## returns a data.frame with all information stored in the spatial data frame of the move object
head(df <- as.data.frame(data))
```

`brownian.bridge.dyn` *Creates a DBBMM object*

Description

The brownian.bridge.dyn function uses a Move object (see `Move-class`) to calculate the utilization distribution, UD, of the given track. It uses the dynamic Brownian Bridge Movement Model (dBBMM) to do so. The dBBMM has the advantage over the other Brownian Bridge Movement Model that changes in behavior are accounted for. It does so by using the behavioral change point analysis in a sliding window. For details see references.

Usage

```
brownian.bridge.dyn(object, raster, dimSize, location.error,
               margin=11, window.size=31, ext=.3, bbox=NA,...)
```

Arguments

`object`	an object of the Move-class
`raster`	a RasterLayer object or numeric value. A numeric value for `raster` is interpreted as the resolution of the square raster cells (in map units); the according raster will be calculated internally. If a RasterLayer is provided the `brownian.bridge.dyn` starts to calculate the UD based on that raster.
`dimSize`	numeric. dimSize is only used if `raster` is not set. `dimSize` is interpreted as the number of cells along the largest dimension of the track. The according raster will be calculated internally. Default is 10
`location.error`	single numeric value or vector of the length of coordinates that describes the error of the location (sender/receiver) system in map units, or a character string with the name of the column containing the location error.
`margin`	The margin used for the behavioral change point analysis.
`window.size`	The size of the moving window along the track. Larger windows provide more stable/accurate estimates of the brownian motion variance but are less well able to capture more frequent changes in behavior.
`ext`	Describes the amount of extension of the bounding box around the animal track. It can be numeric (same extension into all four directions), vector of two (first x, then y directional extension) or vector of four (xmin, xmax, ymin, ymax extension). Default is .25 (extends the bounding box by 25%). Only considered in combination with a numeric raster argument or the dimSize argument.
`bbox`	vector with 4 numbers defining a bounding box for the raster
`...`	for additional arguments, for example burstType which is a character vector with the name of burst type for which the UD needs to be calculates in case a bursted brownian bridge is calculated. Also the argument `verbose=FALSE` can be used to suppress printing messages about the computational size

Details

There are four ways to launch the brownian.bridge.dyn function which are as follows:

1. Use a raster
A RasterLayer object is set for the raster argument which is then used to calculate the UD.

2. Set the cell size
To set the cell size, set a numeric value for the raster argument without providing dimSize. The numeric raster argument is used as the cell sizes of the raster.

3. Set the number of cells (col/row)
To set the number of cells along the largest dimension a numeric dimSize argument can be set.

4. Using default raster
When there are no values set, the default raster value is used to calculate and create a RasterLayer object, which is returned to the same function. Note: depending on the size of the area of interest, the default cell size value can result in a large number of cells which may take a very long time to calculate!

The function prints an estimate of the size of the computational task ahead. This can give an indication of how long the computation is going to take. It should scale roughly linearly with the duration of the computations. In our experience 10e9 takes about a minute with an average laptop.

There is one further argument that can be given: `time.step`. It correspond to the size of the timer intervals taken for every integration step (in minutes). If left `NULL` 15 steps are taken in the shortest time interval.

Note

Note that the first few and last few segments of the trajectory are omitted in the calculation of the UD since a lower number of estimates for the Brownian motion variance are obtained for those segments.

Thanks to Ryan Nielson for making the BBMM package that served as an example for early versions of this code.

Author(s)

Bart Kranstauber, Marco Smolla

References

Kranstauber, B., Kays, R., LaPoint, S. D., Wikelski, M. and Safi, K. (2012), A dynamic Brownian bridge movement model to estimate utilization distributions for heterogeneous animal movement. Journal of Animal Ecology. doi: 10.1111/j.1365-2656.2012.01955.x

Examples

```
## create a move object
data(leroy)
## change projection method to aeqd and center the coordinate system to the track
```

```
data2 <- spTransform(leroy[30:90,], CRSobj="+proj=aeqd +ellps=WGS84", center=TRUE)

## create a DBBMM object
dbbmm <- brownian.bridge.dyn(object=data2, location.error=12, dimSize=155, ext=.45,
     time.step=19.1, margin=15)
```

`brownian.motion.variance.dyn` *Calculates the dynamic brownian motion variance*

Description

A function to calculate the dynamic brownian motion variance for a movement track. It can be used by advanced programmers to program against.

Usage

```
## S4 method for signature '.MoveTrackSingle,numeric,numeric,numeric'
brownian.motion.variance.dyn(object, location.error, window.size, margin)
```

Arguments

`object`	An object of the Move-class, that can be used for variance calculation. It needs to be in a flat coordinate system.
`location.error`	A numeric vector with the location error.
`window.size`	The window size used for the variance calculation.
`margin`	The margin size used for variance calculation.

Value

An object of the type dBMvariance is returned

Author(s)

Bart Kranstauber

References

Kranstauber, B., Kays, R., LaPoint, S. D., Wikelski, M. and Safi, K. (2012), A dynamic Brownian bridge movement model to estimate utilization distributions for heterogeneous animal movement. Journal of Animal Ecology. doi: 10.1111/j.1365-2656.2012.01955.x

See Also

`brownian.bridge.dyn`

Examples

```
data(leroy)
data2 <- spTransform(leroy[1:80,], CRSobj="+proj=aeqd +ellps=WGS84", center=TRUE)
err<-rep(23.5,n.locs(data2))
dBMvar <- brownian.motion.variance.dyn(data2, location.error=err, margin=13, window.siz=31)
dBMvar
```

`burst` *Bursting a track*

Description

Bursting a track by specified variable

Usage

```
  ## S4 method for signature 'Move,factor'
burst(x, f, ...)
```

Arguments

`x`	a Move object
`f`	a character, factor, or numeric that indicates how to burst the coordinates of a Move object. It must be one shorter than the number of locations, because there are always one less segments of a track than coordinates
`...`	not used

Details

The burst function bursts (divides) a track in segments that are specified by the burstIDs (e.g. behavioral annotations). It allows to investigate different parts of a track according to supplied variables like day and night, movement and rest, and so on.

Author(s)

Marco Smolla

Examples

```
data(leroy)
behav <- c(rep(1:4,each=200), rep(5, 118))
testb <- burst(x=leroy, f=behav)
plot(testb)
```

burstId *Returns burstId*

Description

Obtain a factor returning the ids of behavioral categorization per segment

Usage

```
  ## S4 method for signature 'MoveBurst'
burstId(x)
  ## S4 replacement method for signature '.MoveTrackSingleBurst,factor'
burstId(x)<-value
```

Arguments

x	a MoveBurst object
value	Replacement values for the burst factor, either a factor or a character

Value

Returns a factor indicating the categorization

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
burstTrack <- burst(leroy,months(timestamps(leroy))[-1])
burstId(burstTrack)
```

citations *Extract the citation of a Move or MoveStack object*

Description

The citations method returns or sets the citation of a track from a Move or MovesStack object.

Usage

```
  ## S4 method for signature '.MoveGeneral'
citations(obj)
  ## S4 replacement method for signature '.MoveGeneral'
citations(obj) <- value
```

Arguments

obj	Move or MoveStack object
value	citation from class character

Author(s)

Marco Smolla

Examples

```
data(leroy)
citations(leroy) #get the citation from a Move object
citations(leroy) <- "No paper available" #change the citation and set it for a Move object
data(fishers)
citations(fishers) #get the citation from a MoveStack object
citations(fishers) <- "Nothing to cite" #change the citation and set it for a MoveStack object
```

---

contour *Contour plot*

Description

Contour plot of a RasterLayer from a DBBMM object.

Usage

```
  ## S4 method for signature '.UD'
contour(x, ...)
  ## S4 method for signature '.UDStack'
contour(x, ...)
```

Arguments

x	an object of the DBBMM-class or DBBMMStack-class
...	additional arguments like levels and nlevels, see details

Details

The contour function creates a shape of the area in which the animal can be found by a certain probability (i.e. the 90% contour describes the area in which the animal can be found with the 90% probability). One or several probabilities can be set with `levels` (numeric or vector of values between 0 and 1). If no value is set all contour lines are returned. You can also use `nlevel` to set a number of fixed distance levels.
To change parameters of the contour or line plotting use the usual parameters of the `plot` function (like lwd, lty, and so on). You can also add the contour lines to a plot by adding `add = TRUE`.

Author(s)

Marco Smolla

Examples

```
data(leroydbbmm)
## to add a 50% and 95% contour to a plot from DBBMM object dbbmm
plot(leroydbbmm)
contour(leroydbbmm, levels=c(.5,.95), add=TRUE)
```

---

coordinates *Extract the track coordinates from a Move/MoveStack object*

Description

The coordinates method extracts the coordinates of a track.

Usage

```
  ## S4 method for signature 'Move'
coordinates(obj,...)
```

Arguments

obj	A valid Move or MoveStack object
...	Additional arguments, see Details

Details

Returns a matrix with the coordinates of the track in a Move or MoveStack object.

Author(s)

Marco Smolla

Examples

```
## create a move object
data(leroy)
## extract the coordinates
coords <- coordinates(leroy)
```

corridor *Corridor*

Description

Corridor identifies movement track segments whose attributes suggest corridor use behavior

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
corridor(x,speedProp=.75, circProp=.25, plot=FALSE, ...)
  ## S4 method for signature '.MoveTrackStack'
corridor(x,speedProp=.75, circProp=.25, plot=FALSE, ...)
```

Arguments

x	Move or MoveStack
speedProp	numeric between 0 and 1, defines the proportion of speeds which are high enough to be a valid corridor point
circProp	numeric between 0 and 1, defines the proportion of trajectories that are low enough to be a valid corridor point
plot	logical, if TRUE the track is plotted together with dots that indicate corridor points (color scale indicates how many corridor points are near by, less: blue, many: pink)
...	additional arguments like levels and nlevels, see details

Details

The corridor function uses the attributes of a movement step to identify movement steps that exhibit corridor use behavior. For each segment, the speed and the azimuth are calculated and assigned to the segment midpoint. A circular buffer is created around the midpoint of each segment whose radius is equal to half the segment length. The segment azimuth (180 >= azimuth > -180) is then converted into a new unit (the 'pseudo-azimuth' 0 <= 360). Subsequent, the circular variance of the pseudo-azimuths of all segment midpoints that fall within the circular buffer are calculated. This identifies segments that are near parallel segments. Next, it is determined whether a segment's speed is higher than `speedProp` (by default the upper 25% speeds) and its circular variance is lower than `circProp` (by default the lower 25% of all variances). Segment midpoints that meet both of these requirements are considered as a 'corridor' point, all others are considered 'non-corridor' points. Finally, a corridor point is determined to be within a true corridor if within its buffer there are more 'corridor' points than 'non-corridor' points.

Value

The function returns a MoveBurst object or a list of MoveBurst objects (if a MoveStack is supplied). The MoveBurst dateframe stores the following information:
- segment midpoint

- speed
- azimuth
- pseudo-azimuth
- circular variance

The object is bursted by the factor that indicates whether a coordinate belongs to a corridor segment or not.

Note

The default values for the `speedProp` and `circProp` can be changed as per the users discretion using the according argument. If the result of the function is assigned to a variable a MoveBurst object is returned (see Value).

Author(s)

Marco Smolla

References

LaPoint, S., Gallery, P., Wikelski, M. and Kays, R. (2013), Animal Behavior, Cost-based Corridor Models, and Real Corridors. Landscape Ecology. doi:10.1007/s10980-013-9910-0.

Examples

```
data(leroy)
  tmp <- corridor(leroy, plot=TRUE)
  head(tmp) #if assigned to a variable, the coordinates are exported
  data(fishers)
  stacktmp <- corridor(fishers[c(1:400,sum(n.locs(fishers))-(400:1)),]) #working with a stack
```

DBBMM-class *The DBBMM class*

Description

The DBBMM object is created within the brownian.bridge.dyn function from a Move object. It includes among others a raster object and probabilities.

Slots

DBMvar Object of class `"dBMvarianceTmp"`: includes the window.size, margin, means, in.windows, break.list, and points of interest

crs part of the Raster-class

ext the extension factor set by the user

data part of the Raster-class

extent part of the Raster-class

file part of the Raster-class

history part of the Raster-class

legend part of the Raster-class

method stores the method that was used to calculate the utilization distribution (UD), e.g. dynamic Brownian Bridge

ncols part of the Raster-class

nrows part of the Raster-class

rotated part of the Raster-class

rotation part of the Raster-class

title part of the Raster-class

z part of the Raster-class

Methods

`contour signature(object = "DBBMM")`: adds a contour line to a plot

`image signature(object = "DBBMM")`: plots the raster from a DBBMM object with fixed cell size ratio

`plot signature(object = "DBBMM")`: plots the raster from a DBBMM object with re-size insensitive proportions

`proj4string signature(object = "DBBMM")`: extracts the projection method of the raster stored within the DBBMM object

`raster signature(object = "DBBMM")`: extracts the raster from the DBBMM object

`outerProbability signature(object = "DBBMM")`: calculates the animal occurrence probabilities at the border of the raster

Author(s)

Marco Smolla

DBBMMStack-class *The DBBMMStack class*

Description

The DBBMMStack object is created within the brownian.bridge.dyn function from a Move object. It includes among others a raster object and probabilities.

Slots

DBMvar Object of class "dBMvariance": includes the break.list and points of interest

crs part of the Raster-class

ext the extension factor set by the user

extent part of the Raster-class

filename part of the Raster-class

layers part of the Raster-class

method the method that was used to calculate the utilization distribution, e.g. dynamic Brwonian Bridge

ncols part of the Raster-class

nrows part of the Raster-class

rotated part of the Raster-class

rotation part of the Raster-class

title part of the Raster-class

z part of the Raster-class

Methods

contour signature(object = "DBBMMStack"): adds a contour line to a plot

image signature(object = "DBBMMStack"): plots the raster from a DBBMMStack object with fixed cell size ratio

plot signature(object = "DBBMMStack"): plots the raster from a DBBMMStack object with re-size insensitive proportions

proj4string signature(object = "DBBMMStack"): extracts the projection method of the raster stored within the DBBMMStack object

raster signature(object = "DBBMMStack"): extracts the raster from the DBBMMStack object

outerProbability signature(object = "DBBMMStack"): calculates the animal occurrence probabilities at the border of the raster

Author(s)

Marco Smolla

---

dBGBvariance-class *Class to store the orthogonal and parallel variance*

Description

This class stores the orthogonal and parellel variances calculated with the dynBGBvariance function.

dBMvarianceTmp *dBMvarianceTmp class*

Description

The class dBMvarianceTmp is mostly an internal class that is made public to make inheritance easier. It is a basal class that stores results of the dbbmm

Slots

window.size The window size used for dbbmm calculation

margin The margin used for dbbmm calculation

means ...

in.windows ...

interest ...

break.list ...

Author(s)

Marco Smolla

distance *distance information from a track or track stack*

Description

DistanceSummary returns a summary of distance related measurements of a track or track stack, or for the distance function the distance between locations.

Usage

```
  ## S4 method for signature '.MoveTrackSingle,missing'
distance(x)
  ## S4 method for signature '.MoveTrackStack,missing'
distance(x)
  ## S4 method for signature '.MoveTrackSingle'
distanceSummary(x)
  ## S4 method for signature '.MoveTrackStack'
distanceSummary(x)
```

Arguments

x	Move or MoveStack object

Value

All values are returned in meters if the projection of the coordinates is longlat, otherwise their in map units mostly meters as well. For longlat distance on a sphere is calculated using the ellipsoid else on a plane using Pythagoras. Check and set the projection of your Move or MoveStack object using the proj4string() function.

Author(s)

Marco Smolla

Examples

```
data(fishers)
data(leroy)
distance(leroy) #distances from a Move object
distance(fishers) #distances from a MoveStack object
#distanceSummary(leroy) # summary of distance measures of a Move object
#distanceSummary(fishers) # summary of distance measures of a MoveStack object
```

---

dynBGB *Calculation of the dynamic Bivariate Gausian Bridge*

Description

This function creates a utilization distribution according to the Bivariate Gaussian Bridge model. It returns an object of the class dynBGB-class.

Arguments

`move`	the move object or variance object used for calculating the ud if a `.MoveTrackSingle` object is supplied this is converted into a `dBGBvariance` object using the `dynBGBvariance` function
`raster`	either the raster used for UD calculation or the resolution of the raster used for UD calculation
`locErr`	the location errors used for the calculation

Author(s)

Bart Kranstauber

References

Kranstauber, B., Safi, K., Bartumeus, F.. (2014), Bivariate Gaussian bridges: directional factorization of diffusion in Brownian bridge models. Movement Ecology 2:5. doi:10.1186/2051-3933-2-5.

Examples

```
data(leroy)
leroy <- leroy[230:265,]

## change projection method to aeqd and center the coordinate system to the track
dataAeqd <- spTransform(leroy, CRSobj="+proj=aeqd +ellps=WGS84", center=TRUE)

dBGB <- dynBGB(dataAeqd, locErr=9, raster=10, ext=2.15, windowSize=31, timeStep=6, margin=15)
plot(dBGB, col=hsv(sqrt(1:700/1000)))
lines(dataAeqd)
```

---

dynBGB-class *dynBGB class*

Description

This class stores the utilization density calculated using dynamic Bivariate Gausian Briges. It is an extention of the .UD class.

Author(s)

Bart Kranstauber

See Also

.UD

---

dynBGBvariance *calculate variance for a track using a running window*

Description

the function uses `windowApply` with the `BGBvarbreak` function in order to implement a dynaminc calculation of the variance

emd *Earth movers distance*

Description

Calculates the disimilairty, measured as the earth movers distance.

Usage

```
  ## S4 method for signature 'SpatialPoints,SpatialPoints'
emd(x,y, gc = FALSE, threshold = NULL,...)
  ## S4 method for signature 'RasterLayer,RasterLayer'
emd(x,y, ...)
```

Arguments

x	A Raster, Raster stack/brick or SpatialPoints object, it is also possible to provide UD objects. In case of spatial points data frame the first columns of data is used as weights. In case of SpatialPoints all points are weighted equally.
y	A Raster or SpatialPoints.
gc	True if great circle distances should be used.
threshold	The maximal distance over which locations are compaired.
...	Currently not used

Value

An dist object is returned

Author(s)

Bart Kranstauber

References

Kranstauber, B., Smolla, M., Safi,K., Similarity in spatial utilization distributions measured by the Earth Mover's Distance.

Examples

```
data(dbbmmstack)
values(dbbmmstack)[values(getVolumeUD(dbbmmstack))>.999999]<-0
stk<-(dbbmmstack/cellStats(dbbmmstack,sum))
emd(stk[[1]],stk[[2]])
emd(stk)
emd(stk, threshold=10000)
x<-SpatialPointsDataFrame(cbind(c(1:3,5),2), data=data.frame(rep(.25,4)))
y<-SpatialPointsDataFrame(coordinates(x), data.frame(c(0,.5,.5,0)))
emd(x,y)
emd(x,y,threshold=.1)
```

equalProj *Checks projections for being equal*

Description

Checks whether all objects of a list are in the same projection

Usage

```
  ## S4 method for signature 'list'
equalProj(x)
```

Arguments

x	a list of projected objects, like DBBMM or Raster objects, returning the projection string with with the function proj4string

Details

equalProj checks for equal projections using the function of identicalCRS from the package sp. It returns true if none of the objects have a proj4 string.

Author(s)

Bart Kranstauber

Examples

```
data(ricky)
data(leroy)
data(leroydbbmm)
equalProj(list(leroydbbmm,leroydbbmm))
  equalProj(list(leroy,leroydbbmm))
  equalProj(list(leroy,ricky))
```

fishers *A MoveStack*

Description

An MoveStack consisting of two animals, Leroy and Ricky.T

Usage

```
data(fishers)
```

Format

An object of the class MoveStack

Source

https://www.datarepository.movebank.org/handle/10255/move.330

References

LaPoint, Scott, Paul Gallery, Martin Wikelski, and Roland Kays. Animal Behavior, Cost-Based Corridor Models, and Real Corridors. Landscape Ecology 28, 8: 1615???30. doi:10.1007/s10980-013-9910-0.

Examples

```
data(fishers)
```

---

getMotionVariance *Returns the estimated motion variance*

Description

This function returns from an object where it has been calculated before

Usage

```
getMotionVariance(x,...)
```

Arguments

x	A variance object or an UD object calculated using the dynamic Bivariate Gaussian Bridges or dynamic Brownian Bridges
...	Currently un used

Author(s)

Bart Kranstauber

See Also

brownian.bridge.dyn, dynBGB

Examples

```
data(leroydbbmm)
data(dbbmmstack)
getMotionVariance(leroydbbmm)
  getMotionVariance(dbbmmstack)
```

getMovebank *Creates an URL to download Data from Movebank*

Description

An enhanced function to download data from Movebank by manually building an URL. This function should only be used by advanced programmers.

Usage

```
getMovebank(entity_type, login, ...)
```

Arguments

entity_type	the entity_type of the data source
login	a MovebankLogin, if empty you'll be asked to enter your username or password
...	passing on additional arguments

Author(s)

Marco Smolla

getMovebankAnimals *Animals, tags and IDs in a Movebank study*

Description

Returns the animals, their tags and IDs from a Movebank study

Usage

```
getMovebankAnimals(study, login)
```

Arguments

study	a character string (study name) or the numeric study ID as it is stored on Movebank
login	an object of the MovebankLogin-class, if empty you'll be asked to enter your username or password

Details

getMovebankAnimals belongs to the Movebank browsing functions and returns a data.frame that includes the animalID, animalName, id, sensor_type_id and tag_id from the requested study.

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

See Also

movebankLogin

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankAnimals(study=82207, login=login)

## End(Not run)
```

---

getMovebankData *Download data from Movebank*

Description

getMovebankData downloads the location and timestamp columns of a study stored in Movebank

Usage

```
getMovebankData(study, animalName, login, ...)
```

Arguments

study	character, full name of the study, as stored on Movebank
animalName	character, single or a vector, with the name of the individuals as stored on Movebank
login	a MovebankLogin, if empty you'll be asked to enter your username or password
...	passing on additional arguments

Details

getMovebankData belongs to the Movebank browsing functions and returns a Move object from studies with only on animal or MoveStack object for studies with multiple animals. If only a single or several particular animals of a study should be downloaded, a character vector can be provided for the animalName argument.
Remember that you need an account at Movebank.org, see movebankLogin.

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

It his possible to add the argument `removeDuplictedTimestamps=TRUE` and set it to true which allows you delete the duplicated timestamps, it is strongly advised not to use this option because there is no control over which records are removed. Its better to edit the records in movebank and mark the appropriate records as outliers.

Author(s)

Marco Smolla

See Also

`movebankLogin`

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankData(study="BCI Ocelot", login=login)
#returns a MoveStack object from the specified study
getMovebankData(study="BCI Agouti", login=login)
#returns a Move object (there is only one individual in this study)
getMovebankData(study=123413, animalName=c("Mancha","Yara"), login=login)
#returns a MoveStack with two individuals

## End(Not run)
```

`getMovebankID` *Study ID*

Description

Returns the numeric study ID that corresponds to the character study name stored on Movebank

Usage

```
getMovebankID(study,login)
```

Arguments

`study`	character, full name of the study, as stored on Movebank
`login`	an object of the `MovebankLogin-class`, if empty you'll be asked to enter your username or password

Details

getMovebankID belongs to the Movebank browsing functions and returns the ID of a study as it is stored on Movebank.org.

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

See Also

movebankLogin

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankID(study="BCI Ocelot", login=login)

## End(Not run)
```

---

getMovebankSensors *Information about Movebank sensors*

Description

getMovebankSensors returns information about sensors in a study.

Usage

```
getMovebankSensors(study, login)
```

Arguments

study	a character string (study name) or the numeric study ID as it is stored on Movebank
login	an object of the MovebankLogin-class, if empty you'll be asked to enter your username or password

Details

getMovebankSensors belongs to the Movebank browsing functions and returns either information about all sensor types that are available on Movebank (if the study argument is missing) or the sensor IDs corresponding to to the animal IDs in a specific study.

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

See Also

movebankLogin

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankSensors(study=123413, login=login)

## End(Not run)
```

---

getMovebankSensorsAttributes

Available sensor attributes

Description

This function returns all attributes of the sensors of the requested study.

Usage

```
getMovebankSensorsAttributes(study, login)
```

Arguments

study	a character string (study name) or the numeric study ID as it is stored on Movebank
login	an object of the MovebankLogin-class, if empty you'll be asked to enter your username or password

Details

getMovebankSensorAttributes belongs to the Movebank browsing functions and returns the attributes of the sensors of a study, i.e. what is the sensor id and which data types are stored for this sensor (e.g. GPS sensors store longitude and latitude locations, and timestamps and have 673 as their ID on Movebank).

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

See Also

movebankLogin

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankSensorsAttributes(study=123413, login=login)

## End(Not run)
```

---

getMovebankStudies *All studies on Movebank*

Description

Returns all studies available on Movebank

Usage

```
getMovebankStudies(login)
```

Arguments

login	an object of the MovebankLogin-class, if empty you'll be asked to enter your username or password

Details

getMovebankStudies belongs to the Movebank browsing functions and returns a data.frame of all studies available on Movebank.

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankStudies(login=login)

## End(Not run)
```

`getMovebankStudy` *Returns study information*

Description

`getMovebankStudy` belongs to the Movebank browsing functions and returns information about the requested study like the authors of that study, licence type, citation and more.

Usage

```
getMovebankStudy(study, login)
```

Arguments

`study`	a character string (study name) or the numeric study ID as it is stored on Movebank
`login`	an object of the `MovebankLogin-class`, if empty you'll be asked to enter your username or password

Note

See the 'browseMovebank' vignette (move website download section) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

Examples

```
## Not run:
#obtain a login
login<-movebankLogin()
getMovebankStudy(study="BCI Agouti", login=login)

## End(Not run)
```

getVolumeUD *Modify a UD raster*

Description

Modifies the UD raster: that the cell values of the resulting raster are equal to the percentage of the smallest home range containing this cell.

Usage

```
  ## S4 method for signature '.UD'
getVolumeUD(x, ...)
```

Arguments

x	one or several objects of the DBBMM-class or an DBBMMStack-class object
...	additional Raster or DBBMM objects

Author(s)

Marco Smolla

See Also

raster2contour

Examples

```
data(leroydbbmm)
data(dbbmmstack)
getVolumeUD(leroydbbmm) #for a single object
getVolumeUD(dbbmmstack)
getVolumeUD(leroydbbmm, leroydbbmm, leroydbbmm) #for several objects

plot(getVolumeUD(leroydbbmm))
```

hrBootstrap *Calculates and plot the Minimum Convex Polygon for a track*

Description

The hrBootstrap function calculates the 0, 25, 50, 75, 100% percentile of the Minimum Convex Polygon area by step wise (logarithmic) increasing the number of samples per calculation. For every step this calculation is repeated `rep` times with random coordinates from the track. For example it calculates 100 times the mcp area from 3 random locations and store the area. In the next step it calculates it from 5 random locations and so on. The returned graph shows the 5 percentiles of the area sizes. The dashed line indicates the real mcp area size of all locations.

Usage

```
  ## S4 method for signature 'SpatialPoints'
hrBootstrap(x, rep=100, plot=TRUE, level=95, levelMax=100, unin='km', unout='m2', ...)
  ## S4 method for signature '.MoveTrackStack'
hrBootstrap(x, rep=100, plot=TRUE, level=95, levelMax=100, unin="km", unout="m2", ...)
```

Arguments

x	Move, MoveStack, or SpatialPoints object
rep	numeric value for the number of repetitions per sample size, default is 100
plot	logical value that indicates whether the graph is plotted or not, default is TRUE
level	the percentage of coordinates taken into account for the MCP area size calculation, default is 95 (95% of all coordinates are taken into account)
levelMax	the percentage of coordinates taken into account for the maximum MCP area size calculation (horizontal line in the plot)
unin	units form the input values (can be 'm' or 'km')
unout	units for the output values (can be 'm2', 'km2', or 'ha')
...	a Move or MoveStack object

Details

The hrBootstrap function passes values (samples of the track) on to the function mcp that is part of the adehbitatHR package. See the help function of mcp for more information about input and output units.

Value

The values are returned in a data.frame with the units indicated by unout.

Note

Plots for MoveStacks are plotted one after another, and not side by side.

Author(s)

Marco Smolla

Examples

```
m <- move(x=rnorm(70), y=rnorm(70), time=as.POSIXct(1:70, origin="1970-1-1"),
  proj=CRS("+proj=aeqd +ellps=WGS84"))
hrBootstrap(m,rep=5, level=99, unout="m2", plot=TRUE) #for a Move object
  m2 <- move(x=rnorm(70), y=rnorm(70), time=as.POSIXct(1:70, origin="1970-1-1"),
  proj=CRS("+proj=aeqd +ellps=WGS84"))
mstack <- moveStack(list(m,m2))
hrBootstrap(mstack,rep=5, unout="m2", plot=FALSE) #for a MoveStack object

hrBootstrap(as(m,"SpatialPoints"),rep=5, unout="m2", plot=TRUE) #for a SpatialPoints object
```

idData *Functions for dealing with the idData*

Description

This function returns or replaces the idData which is the data per individual.

Usage

```
  ## S4 replacement method for signature '.MoveTrack,missing,missing,data.frame'
idData(x,i,j) <- value
  ## S4 replacement method for signature '.MoveTrack,ANY,ANY,ANY'
idData(x,i,j) <- value
  ## S4 method for signature '.MoveTrack'
idData(x,i,j,...)
```

Arguments

x	Move, MoveBurst or MoveStack object
i	Selection of the rows
j	Selection for the columns
value	Replacement values for the selected idData
...	Other arguments to the data frame subseting such as drop=F

Value

Either the idData data frame or the modified move object

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
idData(leroy)
```

interpolateTime *Interpolate a trajectory*

Description

This function allows to interpolate trajectories. It does this on the basis of a simple interpolation, depending on the spaceMethod that is specified.

Usage

```
interpolateTime(x, time, spaceMethod=c('euclidean','greatcircle','rhumbline'),...)
```

Arguments

x	an object of the Move-class
time	either timestamps, or a number of locations or time interval that is used to generate a sequence of timestamps using the seq function
spaceMethod	a character that indicates the interpolation function to be used
...	other arguments currently not used

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
plot(interpolateTime(leroy[1:200,], 700,'gr'))
```

leroy *GPS track data from a fisher*

Description

This file includes spatial data from a fisher (*Martes pennanti*). It can be used to test the different functions from the move package.
These location data were collected via a 105g GPS tracking collar (manufactured by E-obs GmbH) and programmed to record the animal's location every 15 minutes, continuously. The collar was deployed from 10 February 2009 through 04 March 2009 on an adult, resident, male fisher, in New York, USA (see References). The data usage is permitted for exploratory purposes. For other purposes please get in contact.

Usage

```
data("leroy")
```

Format

An object of the class move

Author(s)

Scott LaPoint

Source

https://www.datarepository.movebank.org/handle/10255/move.330

References

For more information, contact Scott LaPoint <sdlapoint@gmail.com>

Examples

```
## create a Move object from the data set
data <- move(system.file("extdata","leroy.csv.gz",package="move"))
plot(data)
data(leroy)
```

---

lines *Plotting a track as lines*

Description

Function for plotting a recorded track from a Move object as lines

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
lines(x,...)
  ## S4 method for signature '.MoveTrackStack'
lines(x,col=NA,...)
  ## S4 method for signature '.MoveTrackSingleBurst'
lines(x,col=NA,...)
```

Arguments

x	Move object
col	a vector of colors (same length as the number of individual for a stack or number of burst levels or segments for a burst object, or one)
...	arguments to be passed on, e.g. lty or lwd.

Author(s)

Marco Smolla

See Also

points plot

Examples

```
data(leroy)
data(leroydbbmm)
plot(leroydbbmm)
lines(spTransform(leroy, center=TRUE), col=3) # add a track from a Move object to a plot
```

---

move *Create a Move object*

Description

The move method creates Move or MoveStack object from Movebank or other (compressed) csv files, also zip files from the environmental annotation tool can be loaded. If you use your own data you need to set the projection method with the 'proj' argument and specify in which column of your data the function finds locations and timestamps.

Usage

```
## S4 method for signature 'connection,missing,missing,missing,missing'
move(x, removeDuplicatedTimestamps=F, ...)
## S4 method for signature 'numeric,numeric,POSIXct,data.frame,CRS'
move(x, y, time, data, proj, sensor='unknown',animal='unnamed',...)
```

Arguments

x	Full path to the file location, OR a vector with x coordinates if non-Movebank data are provided (e.g. data$x). Alternatively if no move other arguments are provided an ltraj object
y	vector of y coordinates if non-Movebank data are provided
time	column indicator for non-Movebank data for the time stamps, with POSIXct conversion, i.e. as.POSIXct(data$timestamp, format="%Y-%m-%d %H:%M:%S", tz="UTC")
data	Optional extra data associated with the relocation, if empty it is filled with the coordinates and timestamps
proj	projection method for non-Movebank data; requires a valid CRS (see CRS-class) object, like CRS("+proj=longlat +ellps=WGS84"); default is NA

`sensor` sensor name, either single character or a vector with length of the number of coordinates

`animal` animal ID or name, either single character or a vector with length of the number of coordinates

`removeDuplicatedTimestamps`
It his possible to add the argument removeDuplicatedTimestamps and set it to true which allows you delete the duplicated timestamps, it is strongly advised not to use this option because there is no control over which records are removed. Its better to edit the records in movebank.

`...` Additional arguments

Details

The easiest way to import data is to download the study you are interested in from www.movebank.org. Set the file path as the x argument of the move function. The function detects whether there are single or multiple individuals in this file and automatically creates either a Move or MoveStack object. Another way is to read in your data using `read.csv`. Then the columns with the x and y coordinates, and the timestamp, as well as the whole data.frame of the imported data are given to the `move` function. Again the function detects whether to return a Move or a MoveStack object

Note

The imported data set is checked whether it is in a Movebank format. If this is not the case, you have to use the alternative import for non-Movebank data (see above). Because the SpatialPointsDataFrame function that creates the spatial data frame of the `Move` object can not process NA location values, all rows with NA locations are stored as unused records.

If the data include double timestamps check your data for validity. You may want to consider a function to delete double timestamps, like: `data <- data[which(!duplicated(data$timestamp)), ]` or use the removeDuplicatedTimestamps argument but this does no attempt to retain the most likely location.

Due to convention all names are turned into 'good names' which means, without spaces ('Ricky T' becomes 'Ricky.T').

Author(s)

Marco Smolla \& Bart Kranstauber

Examples

```
## create a move object from a Movebank csv file
filePath<-system.file("extdata","leroy.csv.gz",package="move")
data <- move(filePath)

## create a move object from non-Movebank data
file <- read.table(filePath,
   header=TRUE, sep=",", dec=".")
data <- move(x=file$location.long, y=file$location.lat,
     time=as.POSIXct(file$timestamp,
     format="%Y-%m-%d %H:%M:%S", tz="UTC"),
```

```
data=file, proj=CRS("+proj=longlat +ellps=WGS84"), animal="Leroy")
```

`Move-class` *The Move class*

Description

The `Move` object contains at least time and coordinate information of an animal. It can contain further data that are individual to the animal, e.g. the sex or age. These data are stored in the idData data.frame. If the object was created with the Movebank browsing functions it does also contain the study name, licence and citation information.

Slots

idData Object of class `"data.frame"`: additional (one row) data;

dateCreation Object of class `"numeric"`: time stamp when the file was downloaded;

study Object of class `"character"`: name of the study;

citation Object of class `"character"`: how to cite the study;

license Object of class `"character"`: the license under which the data were published;

timesMissedFixes Object of class `"POSIXct"`: stores the timestamps of lines of the data set that were removed because they included NA locations

bbox belongs to the SpatialPointsDataFrame

coords coordinates of the track, belongs to the SpatialPointsDataFrame

coords.nrs belongs to the SpatialPointsDataFrame

data additional data of that object that is stored in the SpatialPointsDataFrame

proj4string projection of the coordinates

timestamps timestamps according to the coordinates

Methods

`move` `signature(object = "Move")`: creates a Move object

`getMovebankData` `signature(object = "character")`: creates a Move object by accessing Movebank

`spTransform` `signature(object = "Move")`: transforms coordinates to a different projection method

`show` `signature(object = "Move")`: prints a summary of all data stored in the Move object

`as.data.frame` `signature(object = "Move")`: extracts the spatial data frame

`coordinates` `signature(object = "Move")`: extracts the coordinates only from the Move object

`proj4string` `signature(object = "Move")`: extracts the projection method from the Move object

`time.lag` `signature(object = "Move")`: calculates time lags between coordinates

`n.locs` `signature(object = "Move")`: calculates number of locations

`plot` `signature(object = "Move")`: plots the track of the animal

Note

.MoveTrack and .MoveTrackSingle are also exported for other people to program against

Author(s)

Marco Smolla

`move2ade` *Convert a Move or MoveStack object to adehabitat compatible object*

Description

Convert a Move or MoveStack object to adehabitat compatible object. This is necessary because Move and MoveStack objects are not inherited by the object class that is typically used by the adehabitat package. Therefore, the `move2ade` function allows to use functions of the adehabitatHR package with objects that were originally created with the Move package.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
move2ade(x)
  ## S4 method for signature '.MoveTrackStack'
move2ade(x)
```

Arguments

`x`	a Move or MoveStack object

Value

The returned object is from SpatialPointsDataFrame with the animal name (or 'unnamed') stored in the data slot of the SpatialPointsDataFrame.

Author(s)

Marco Smolla

Examples

```
data(fishers)
data(leroy)
move2ade(leroy) #for a Move object
move2ade(fishers) #for a MoveStack object
```

movebankLogin *Login into Movebank*

Description

Creates an object that can be used with all Movebank browsing functions.

Usage

```
## S4 method for signature 'character,character'
movebankLogin(username,password)
```

Arguments

username	Your Movebank username
password	Your Movebank password

Details

Use this function to login to Movebank. After you logged in, you can use the Movebank browsing functions from the move package.

Note

If you do not have the RCurl package installed movebankLogin will store your username and password in the object you assign it to. Furthermore, if you have no RCurl a http connection is used to retrieve data from Movebank instead of the more secure https protocol.

Author(s)

Marco Smolla

Examples

```
## Not run:
##first create the login object
login <- movebankLogin(username="xxx", password="zzz")

##and than use it with Movebank browsing functions
getMovebankStudies(login)

## End(Not run)
```

MovebankLogin-class *The MovebankLogin class*

Description

The MovebankLogin object is needed for every Movebank browsing function. Alternatively, one can also chose to enter the username and password every time one uses one of the browsing functions. The object is inhereted from an `httr request` obnject.

Methods

`movebankLogin signature(object = "character")`: creates a MovebankLogin object

Author(s)

Bart Kranstauber \& Marco Smolla

MoveBurst *MoveBurst class*

Description

The class MoveBurst, used for storing information about one individual with assignments of segments. Every segment between to locations has a class for example behavioural category.

Slots

bbox ...
burstId ...
citation ...
coords ...
coords.nrs ...
data ...
dateCreation ...
idData ...
license ...
proj4string ...
sensor ...
study ...
timesMissedFixes ...
timestamps ...

Author(s)

Marco Smolla

moveStack *Creating a MoveStack*

Description

Stacks a list of Move objects

Usage

```
  ## S4 method for signature 'list'
moveStack(x)
```

Arguments

x	a list of Move objects

Details

This function stacks single Move objects to a MoveStack object.

Note

All animal names are converted into 'good names' which means, that spaces are replaced with points and duplicated names get an individual number added. For example:
'Leroy, Leroy' -> adding number to duplicated names ->'Leroy, Leroy.1'
'Ricky T' -> replacing spaces -> 'Ricky.T'

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(ricky)
l <- list(ricky[200:270,], leroy[200:270,])
moveStack(l)
```

`MoveStack-class` *The MoveStack class*

Description

The MoveStack object is created within the brownian.bridge.dyn function from a Move object. It includes among others a raster object and probabilities.

Slots

citation Object of class `"dBMvariance"`: includes the break.list and points of interest, ...;

bbox belongs to the SpatialPointsDataFrame

coords the extension factor set by the user

coords.nrs belongs to the SpatialPointsDataFrame

data additional data of that object that is stored in the SpatialPointsDataFrame

dateCreation date and timestamp when this object was created

idData additional data to all individuals

license the license terms of the used track material

proj4string projection of all coordinates

study name of the study

timestamps timestamps according to the coordinates

trackId a vector that indicates, which data, coordinates and timestamps belong to an individual

Methods

Methods defined with class "MoveStack" in the signature:

`[` `signature(x="MoveStack")`: select subset ...

`getMovebankData` `signature(object = "character")`: creates a MoveStack object by accessing Movebank

Author(s)

Marco Smolla

n.indiv *Extract the number of individuals of a Move or MoveStack object*

Description

The n.indiv function returns the number of individuals from a Move or MovesStack object.

Usage

```
  ## S4 method for signature 'Move'
n.indiv(obj)
  ## S4 method for signature '.MoveTrackStack'
n.indiv(obj)
```

Arguments

obj	Move or MoveStack object

Details

The function returns 1 for objects of the class Move, and number of individuals for a moveStack

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
n.indiv(leroy)
data(fishers)
n.indiv(fishers)
```

n.locs *Extract the number of locations of a Move or MoveStack object*

Description

The n.locs method returns the number of locations of a track from a Move or MovesStack object.

Usage

```
  ## S4 method for signature 'SpatialPointsDataFrame'
n.locs(obj)
  ## S4 method for signature '.MoveTrackStack'
n.locs(obj)
```

Arguments

obj	Move or MoveStack object

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(fishers)

n.locs(leroy) #number of locations of Move object
n.locs(fishers) #number of locations of MoveStack object
```

---

outerProbability *Calculates the probabilities at the edges of a raster*

Description

The outerProbability method calculates the summed probability of the cells at the border of a raster

Usage

```
  ## S4 method for signature 'RasterLayer'
outerProbability(raster,border,...)
  ## S4 method for signature 'DBBMMStack'
outerProbability(raster,border,...)
```

Arguments

raster	a RasterLayer or DBBMMStack object that has values for the raster cells
border	numeric from 0 to 1; ratio of the number of columns at the border relative to the whole raster from which the probabilities should be summed up; default is 10% (0.1)
...	hand over information

Details

The function returns the summed probability at the border (e.g. the outer 10% of the cells) of a raster. This value can be used as an indicator whether the extent of the used raster is to small for the UD calculation and therefore too much probabilities are not calculated because they are outside the raster.

Value

numeric value for a single DBBMM object, or a list of numeric values for a DBBMMStack

Author(s)

Marco Smolla

Examples

```
data(leroydbbmm)
#calculate the probabilities of 20% of the raster at the border from a DBBMM
outerProbability(leroydbbmm, border=.2)

#calculate the probabilities of 50% of the raster at the border from a DBBMMStack
outerProbability(leroydbbmm, border=.5)
```

---

plot *Plotting track or raster*

Description

Function for plotting a recorded track from a Move object or the probability values from a DBBMM object

Usage

```
## S4 method for signature '.MoveTrackSingle,missing'
plot(x, y,asp=1, ...)
## S4 method for signature '.MoveTrackStack,missing'
plot(x, y, type="p",asp=1,  ...)
## S4 method for signature '.MoveTrackSingleBurst,missing'
plot(x, y, type="p",asp=1,  ...)
```

Arguments

x	Move, MoveStack, MoveBurst or DBBMM object
y	unused variable (listed for compatibility reasons)
type	defines the type of the plot (e.g. 'l', 'p', 'b', 'o')
asp	defines the aspect ratio of the plot generally 1 makes most / only sense since then x and y dimenstions are the same
...	arguments to be passed to methods, such as graphical parameters, and the logical add argument (see par)

Details

If x is a Move, MoveStack object a track is plotted with points and lines. The track can be added to another plot with the add = TRUE.
If x is a MoveBurst object colored lines (according to the burstID) are plotted if the type is set to 'l'. By default it is 'p' which plots the real coordinates of the Move object as points. If x is a DBBMM object its raster object is plotted with the corresponding cell values. Unlike the image function, the cell size ratio keeps the same when the plot window is re-sized.

Note

Have a look on the proportion of the graphic device when printing a track or raster. The plot function does not use equal sized units on both axes.

Author(s)

Marco Smolla

See Also

points lines

Examples

```
data(leroy)
data(fishers)
plot(leroy) # plot a Move object
plot(leroy, type="o", col=3)
plot(fishers, col=c(3,5), lwd=3) # plot a MoveStack object
plot(fishers, type="l", col=c(3,5), lwd=3)
data(dbbmmstack)
data(leroydbbmm)
plot(leroydbbmm) # plot the raster of a DBBMM object
plot(dbbmmstack) # plot the raster of a DBBMMStack object
```

---

plotBursts *Plotting the centroids of a track*

Description

The plotBursts function plots bursted Move objects (see ?burst for how to create a bursted Move Object). For every single burst a circle is plotted at the center of every segment of a track. A segment represents consecutive coordinates that belong to a single burst. The properties of the plotted circles (size and color) can represent different information (see Details).

Usage

```
  ## S4 method for signature 'list'
plotBursts(object, add=TRUE,
      sizeFUN=function(x) {as.numeric(diff(range(timestamps(x))),
      units ="mins")},
      col = NA, breaks = 3, ...)
  ## S4 method for signature '.MoveTrackSingleBurst'
plotBursts(object, add=TRUE,
       sizeFUN=function(x) {
       as.numeric(diff(range(timestamps(x))),
       units ="mins")
       },
       col = NA, breaks = 3, ...)
```

Arguments

object	a SpatialPointsDataFrame or a list of these that include coordinates and color, and size of the centroid indicators
add	logical, if FALSE a new plot is generated, default is TRUE
sizeFUN	a function to calculate the size of the plotted circles (see details)
breaks	how many size classes should the circles have, default is 3
col	a vector of color codes with the same length as the burstID. By default the standard colors from 1:8 are used. If there are more than 8 burstIDs the colors are recycled
...	additional plot attributes

Details

Circle colors correspond to burstIDs. The size of the circles is defined by a function. By default this function calculates the relative duration of a segment compared to the whole duration. The function can be adjusted to use a different measure using the sizeFUN argument.
Note, it is possible to define break points by using the breaks argument.

Author(s)

Marco Smolla

Examples

```
data(leroy)
behav <- c(rep(1:4,each=200), rep(5, 118))
testb <- burst(leroy, f=behav)
plot(coordinates(leroy),type="l")
plotBursts(testb, breaks=3, add=TRUE, pch=19)
plotBursts(testb, breaks=5, add=FALSE, pch=19)

##plotBursts(leroy_b, breaks=c(-Inf,4000,6000, Inf), add=FALSE, pch=19)

#bursting track by realtive segment length
plotBursts(object=testb, breaks=3, sizeFUN=function(x) sum(seglength(x)), pch=19, add=FALSE)
```

---

points *Plotting the points of a track*

Description

Function for plotting a recorded track from a Move object as points.

Usage

```
## S4 method for signature '.MoveTrackSingle'
points(x,...)
## S4 method for signature '.MoveTrackStack'
points(x,col=NA,...)
## S4 method for signature '.MoveTrackSingleBurst'
points(x,...)
```

Arguments

x	Move or a DBBMM object
col	a vector of colors (same length as the number of objects)
...	arguments to be passed on, e.g. col for color, or add to add the points to a plot.

Author(s)

Marco Smolla

See Also

plot lines

Examples

```
data(leroydbbmm)
data(leroy)
plot(leroydbbmm)
points(spTransform(leroy, center=TRUE), col=3) # add a track from a Move object to a plot
```

---

raster *Extract raster from DBBMM*

Description

Extracts the RasterLayer from a DBBMM and DBBMMStack object.

Usage

```
  ## S4 method for signature 'DBBMM'
raster(x)
  ## S4 method for signature 'DBBMMStack'
raster(x)
```

Arguments

x	a DBBMM or DBBMMStack object

Details

The raster function extracts the raster object from a DBBMM or DBBMMStack object.

Value

An object from class RasterLayer is returned.

Author(s)

Marco Smolla

Examples

```
data(leroydbbmm)
data(dbbmmstack)
raster(leroydbbmm) #returns the raster of a DBBMM object
raster(dbbmmstack) # returns the raster of a DBBMMStack object
plot(raster2contour(leroydbbmm, levels=c(.5,.9)), col=c(5,4)) # plot the raster lines of a DBBMM
```

---

raster2contour *Convert raster to contour lines*

Description

The function converts a ud(stack) object to a SpatialLinesDataFrame.

Usage

```
## S4 method for signature '.UD'
raster2contour(x, ...)
## S4 method for signature '.UDStack'
raster2contour(x, ...)
```

Arguments

x	a DBBMM or DBBMMStack object, that includesa raster object
...	additional arguments that are passed on from other functions, most important being level which determines the contour level

Details

The raster2contour function creates a SpatialLinesDataFrame from a given raster or DBBMM object. This allows to re-project the contours to different projections.

Author(s)

Marco Smolla

See Also

getVolumeUD

Examples

```
data(leroydbbmm)
data(dbbmmstack)
raster2contour(leroydbbmm)
raster2contour(dbbmmstack)
raster2contour(dbbmmstack, level=c(.5,.95))
```

---

ricky *GPS track data from a fisher*

Description

This file includes spatial data from a fisher (*Martes pennanti*). It can be used to test the different functions from the move package.
These location data were collected via a GPS tracking collar (manufactured by E-obs GmbH) and programmed to record locations depending on the animal's behaviour (up to a one location every two minutes). The collar was deployed on an adult, resident, male fisher, in New York, USA (see References). The data usage is permitted for exploratory purposes. For other purposes please get in contact.

Author(s)

Scott LaPoint

References

For more information, contact Scott LaPoint <sdlapoint@gmail.com>

Examples

```
## create a Move object from the data set
data(ricky)
plot(ricky)
```

searchMovebankStudies *Search for a study*

Description

Searches for a study within Movebank

Usage

```
searchMovebankStudies(x, login)
```

Arguments

x	a character string to search within the Movebank study names
login	an object of the MovebankLogin-class, if empty you'll be asked to enter your username or password

Details

The search function searches explicitly for the entered phrase. If you for example type 'Goose' it will not show you studies including 'goose'. So rather search for 'oose' to find both.

Value

The function returns a character vector of study names.

Note

See the 'browseMovebank' vignette (link) for more information about security and how to use Movebank from within R.

Author(s)

Marco Smolla

Examples

```
## Not run:
searchMovebankStudies("MPIO", login=login) #returns all studies that include this exact term

## End(Not run)
```

seglength *Calculates the segment length of a track*

Description

Calculates the segment length of a track

Usage

```
  ## S4 method for signature 'SpatialPointsDataFrame'
seglength(x)
```

Arguments

x	a SpatialPointsDataFrame, like a Move or MoveStack object

Details

The seglength function calculates the distances between point 1 and point 2, point 2 and point 3, ...

Value

If the SpatialPointsDataFrame is projected in longitude latitude coordinates, the returned values are in meters. If not, the distances are Euclidean distances in map units. Distances are calculates with the `pointDistance` from the package raster. Note that in stacks distances are not split between animals.

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(fishers)
head(seglength(leroy)) #Move object in longlat projection
head(seglength(fishers)) #MoveStack object in aeqd projection
```

sensor *Extract the sensor of a Move or MoveStack object*

Description

The sensor method returns or sets the sensor of a track from a Move or MovesStack object.

Usage

```
## S4 method for signature '.MoveTrack'
sensor(this,...)
## S4 method for signature '.unUsedRecords'
sensor(this,...)
```

Arguments

this	Move or MoveStack object
...	Curently not used

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
sensor(leroy) #get the sensor from a Move object
data(fishers)
sensor(fishers) #get the sensor from a MoveStack object
```

show *Show a Move object*

Description

Displays a summary of a Move object.

Usage

```
  ## S4 method for signature 'Move'
show(object)
  ## S4 method for signature 'MoveStack'
show(object)
```

Arguments

object	a Move or MoveStack object

Details

The show function displays a summary of a Move object. This includes:

- animal ID
- species name
- study name
- number of track points
- receiver type
- projection method
- date of file creation
- the first three lines of the spatial data frame
- study citation
- data license
- number of omitted locations due to NAs in the dataset

If the imported data are not from the Movebank database Animal, Species, nPoints, Receiver, and Study are not shown.

Author(s)

Marco Smolla

---

speed *speed information from a track or track stack*

Description

This function returns a summary of speed related measurements of a Move or MoveStack object.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
speed(x)
  ## S4 method for signature '.MoveTrackStack'
speed(x)
  ## S4 method for signature '.MoveTrackSingle'
speedSummary(x)
  ## S4 method for signature '.MoveTrackStack'
speedSummary(x)
```

Arguments

x	Move or MoveStack object

Value

The function returns the speed in m/s if the track is longlat else map units per second. In most cases the map units will also be meters.

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(fishers)

speed(leroy) #speeds from a Move object
speed(fishers) #speeds from a MoveStack object
#speedSummary(leroy) # summary of speed measures of a Move object
#speedSummary(fishers) # summary of speed measures of a MoveStack object
```

---

split *Splitting a MoveStack*

Description

Splitting MoveStack in a list of Move objects

Usage

```
  ## S4 method for signature 'MoveStack,missing'
split(x, f, drop=FALSE, ...)
```

Arguments

x	MoveStack or MoveBurst object
f	not needed
drop	not needed
...	not needed

Details

A MoveStack is split into a list of Move objects by the track IDs of the given MoveStack. For stacking moveStack can be used. Also a MoveBurst object can be split this will lead to seperate list elements for every burst, every location where the burst is switched will then be recycled.

Author(s)

Marco Smolla

Examples

```
data(dbbmmstack)
data(fishers)
split(fishers) #splitting a MoveStack
split(dbbmmstack) #splitting a DBBMMStack
```

---

spTransform *Transform Move object projection method*

Description

The spTransform function transforms the coordinates stored in the Move object from the default longlat coordinates to the default aeqd (Azimuthal Equi-distance) projection or a different projection.

Usage

```
## S4 method for signature 'Move,character'
spTransform(x,CRSobj,center=FALSE)
## S4 method for signature 'Move,missing'
spTransform(x,center=FALSE,...)
```

Arguments

x	a Move or a MoveStack object
CRSobj	a CRS like character that describes the projection method to which the coordinates should be transformed, if missing "+proj=aequd" is used as default value
center	logical, if TRUE the center of the coordinate system is the center of the track; FALSE is default
...	for additional arguments

Details

The spTransform function transforms the coordinates of a Move object by default from "+proj=longlat" to "+proj=aeqd". In this format the coordinates can be used by the brownian.bridge.dyn function.
If center is TRUE the center of the coordinate system is set to the center of the track.

Author(s)

Marco Smolla

Examples

```
## create a Move object
data(leroy)
## transform the Move object by default into "+aeqd" projection method
## and center the coordinate system
spTransform(leroy, center=TRUE)

## transform the Move object into another projection method, like mollweide
spTransform(leroy, CRSobj="+proj=moll +ellps=WGS84")

##check projection method
proj4string(leroy)
```

---

subset-method *Returns a single object from a MoveStack*

Description

The subset function can be used to return a subset of a Move(Stack) object.

Usage

```
  ## S4 method for signature 'MoveStack,ANY,ANY'
x[i]
  ## S4 method for signature 'MoveStack,character,missing'
x[[i]]
```

Arguments

x	MoveStack object
i	numeric, character or logical vector for individuals in a stack or a set of locations

Details

The double square bracket method is used for sub setting a stack to a single move object according to the individual name or return a stack of multiple individuals.

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
leroy[1:20,]
leroy[c(TRUE,FALSE),]
data(fishers)
fishers[1:300,]
fishers[1]
fishers[['Ricky.T']]
fishers[[2]]
fishers[[c(TRUE,FALSE)]]
```

---

summary *A summary of a DBBMM or DBBMMStack, Move or MoveStack object*

Description

Summarizes the information of the raster from a DBBMM or DBBMMStack, or the data from a Move or MoveStack object

Usage

```
  ## S4 method for signature '.UD'
summary(object)
  ## S4 method for signature '.UDStack'
summary(object)
  ## S4 method for signature '.MoveTrackSingle'
summary(object)
  ## S4 method for signature '.MoveTrackStack'
summary(object)
```

Arguments

object	a DBBMM/DBBMMStack, Move/MoveStack object

Details

Returns the projection, extent, and maximum and minimum values of the raster stored within the DBBMM or DBBMMStack object. If the object is a Move or MoveStack object the functions distance, time, speed, and angle is called and a list with the summarized values is returned.

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(fishers)
data(leroydbbmm)
  summary(leroy) # summary of angle measures of a Move object
  summary(fishers) # summary of angle measures of a MoveStack object
  summary(leroydbbmm) # summary of a DBBMM object
```

`time.lag` *Calculates the time lags between the coordinates*

Description

The time.lag function calculates the time lags between locations.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
time.lag(x,...)
  ## S4 method for signature '.MoveTrackStack'
time.lag(x,units, ...)
```

Arguments

`x`	a Move or MoveStack object
`units`	The units used for the conversion, they need to be specified for a stack to ensure similair units between individuals
`...`	further arguments

Details

This function is now replaced by timeLag

Author(s)

Marco Smolla

timeLag *Calculates the time lags between the coordinates*

Description

The timeLag function calculates the time lags between locations.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
timeLag(x,...)
  ## S4 method for signature '.MoveTrackStack'
timeLag(x,units, ...)
```

Arguments

x	a Move or MoveStack object
units	The units used for the conversion, they need to be specified for a stack to ensure similair units between individuals
...	further arguments

Details

Optionally the argument units can be passed on to ensure the time lag in a certain unit, this is especially useful in case a move stack is bursted. For more information on the units argument see the help of difftime.

Value

The function returns a numeric vector with the time lags.

Author(s)

Bart Kranstauber

Examples

```
data(fishers)
data(leroy)

head(timeLag(leroy,units="mins")) #calculate timelags for a Move object
head(timeLag(fishers,units="mins")) #calculate timelags for a MoveStack object
head(timeLag(fishers, units="hours")) #calculate timelags in different units
```

timestamps *Extract the timestamps of a Move or MoveStack object*

Description

The timestamps method returns or sets the timestamps of a track from a Move or MovesStack object.

Usage

```
## S4 method for signature '.MoveTrackSingle'
timestamps(this)
## S4 method for signature '.MoveTrack'
timestamps(this)
## S4 replacement method for signature '.MoveTrack'
timestamps(this) <- value
```

Arguments

this	Move or MoveStack object
value	timestamps from class POSIXct

Author(s)

Marco Smolla

Examples

```
data(leroy)
data(fishers)

  timestamps(leroy) #get the timestamps from a Move object
  timestamps(fishers) #get the timestamps from a MoveStack object
  #change the timestamps and set it for a Move object
  timestamps(leroy) <- timestamps(leroy)+60
  #change the timestamps and set it for a MoveStack object
  timestamps(fishers) <- timestamps(fishers)+60
```

timeSummary *time information from a track or track stack*

Description

This function returns a summary about time related measurements of a Move or MoveStack object.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
timeSummary(x, units="hours")
  ## S4 method for signature '.MoveTrackStack'
timeSummary(x, units="hours")
```

Arguments

x	Move or MoveStack object
units	defines the output uni, 'secs', 'mins', 'hours', 'days', ...

Author(s)

Marco Smolla

Examples

```
data(fishers)
data(leroy)
timeLag(leroy)
timeLag(fishers)
timeSummary(leroy) # summary of time measures of a Move object
timeSummary(fishers, units="hours") # summary of time measures of a MoveStack object
```

---

trackId *Returns trackId*

Description

Obtain a factor returning the ids of individuals per location

Usage

```
  ## S4 method for signature 'MoveStack'
trackId(x)
```

Arguments

x	a MoveStack object

Value

Returns a factor indicating to which individuals locations belong in a stack.

Author(s)

Bart Kranstauber

Examples

```
data(fishers)
  trackId(fishers)
```

`turnAngleGc` *Calculates turning angles on great circle tracks*

Description

This function returns the turn angels of a great circle track.

Usage

```
  ## S4 method for signature '.MoveTrackSingle'
turnAngleGc(x)
```

Arguments

`x`	Move object, in long lat projection

Details

On great circle tracks the bearing of arrival on a point is not the same as witch the previous point was left. This function returns the difference between these bearings between -180 and 180. The bearings are calculated using the functions bearing and finalBearing of the geosphere package.

Value

The function returns the turning angles in degrees.

Author(s)

Bart Kranstauber

Examples

```
data(leroy)
turnAngleGc(leroy)
```

UDStack *Creating UDStack objects*

Description

The function enables the easy generation of .UDStacks, which is forexample useful for using other UD function such as `getVolumeUD`.

Usage

```
UDStack(x,...)
```

Arguments

x	A list or raster brick/stack that needs to be converted to a .UDStack object. Alternative a burst stack, in this case the layers are standardized.
...	Currently not used

Value

An UDStack

Author(s)

Bart Kranstauber

Examples

```
data(dbbmmstack)
stk<-as(dbbmmstack,"RasterStack")
UDStack(stk)
lst<-split(dbbmmstack)
UDStack(lst)
```

unUsedRecords<- *Creates unused records data*

Description

A function that assigns locations as unused or returns the unUsedRecords part of the object it could for example be used to remove locations test locations from a track.

Usage

```
  ## S4 replacement method for signature '.MoveTrackSingle,logical'
unUsedRecords(obj) <- value
  ## S4 replacement method for signature '.MoveTrackStack,logical'
unUsedRecords(obj) <- value
  ## S4 method for signature '.unUsedRecords'
unUsedRecords(obj,...)
  ## S4 method for signature '.unUsedRecordsStack'
unUsedRecords(obj,...)
```

Arguments

obj	a Move or MoveStack object
value	A logical vector of the same length as the number of locations
...	Not used at the moment

Author(s)

Marco Smolla

Examples

```
data(leroy)
par(mfrow=2:1)
plot(leroy, type='b')
unUsedRecords(leroy)<-as.logical((1:n.locs(leroy))%%2)
plot(leroy, type='b')
```

---

utilization density data

Dynamic brownian bridges

Description

Utilization densities calculated with `brownian.bridge.dyn` to examplify functions.

Usage

```
data("leroydbbmm")
```

Details

see createRDataFile.R in inst/extdata for the exact calculation

Examples

```
data(dbbmmstack)
data(leroydbbmm)
leroydbbmm
```

Index