# Package: move (via r-universe)

November 25, 2024

**Type** Package

**Title** Visualizing and Analyzing Animal Track Data

**Version** 4.2.6

**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** https://bartk.gitlab.io/move/

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

**LazyLoad** yes

**LazyData** yes

**LazyDataCompression** xz

**Depends** geosphere (>= 1.4-3), methods, sp, raster (>= 3.6-14), R (>= 3.5.0)

**Suggests** adehabitatHR, adehabitatLT, markdown, rmarkdown, circular, ggmap, mapproj, testthat, knitr, ggplot2, leaflet, lubridate, ctmm, amt, bcpa, EMbC, solartime

**Imports** httr, memoise, terra, xml2, Rcpp

**LinkingTo** Rcpp

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Bart Kranstauber [aut, cre], Marco Smolla [aut], Anne K Scharf [aut]

**Maintainer** Bart Kranstauber <b.kranstauber@uva.nl>

**Repository** CRAN

**Date/Publication** 2024-11-20 16:40:02 UTC

**Config/pak/sysreqs** libgdal-dev gdal-bin libgeos-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev

# Contents

---

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

---

## Description

move is a package that contains functions to access movement data stored on Movebank as well as tools to visualize and statistically analyse animal movement data. Move addresses ecological questions regarding movement.

## 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

The dynamic Brownian Bridge Movement Model calculates the occurrences distribution of a given track

| | |
|---|---|
| `brownian.bridge.dyn` | To calculate the occurrences distribution |
| `getVolumeUD` | To calculate the Utilization distribution (UD) |

## III. Accessing values

| | |
|---|---|
| `bbox` | Bounding box of a Move* object |
| `coordinates` | Track-coordinates of a Move* object |
| `extent` | Extent of a Move* object |
| `namesIndiv` | Names of a Move* object |
| `n.locs` | The number of locations a Move* object |
| `projection` | The projection method of a Move* object or raster |
| `timeLag` | The time lags between the locations of a Move* object |
| `timestamps` | Track-timestamps of a Move* object |

## 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-clas |
| `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, Anne Scharf

Maintainer: Bart Kranstauber, Marco Smolla, Anne Scharf

## References

move on CRAN

---

`.UD-class` *The UD class*

---

## Description

The `.UD`, `.UDStack` and `.UDBurstStack` class represent a `raster` of a simple abstraction of the utilization distribution (UD) where all probabilities necessarily sum to one. A `.UDStack` object can be obtained with the function UDStack.

## 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 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 = ".UD"): adds a contour line to a plot, also for .UDStack

emd signature(object = ".UD"): quantifies similarity between utilization distributions, also for .UDStack

getVolumeUD signature(object = ".UD"): modifies the .UD/.UDStack raster

outerProbability signature(object = ".UD"): calculates the animal occurrence probabilities at the border of the raster (only for .UD class)

plot signature(object = ".UD"): plots the raster from a .UD/.UDStack object with re-size insensitive proportions

raster2contour signature(object = ".UD"): converts a raster to contour lines, also for .UDStack

show signature(object = ".UD"): displays summary the .UD/.UDStack object

summary signature(object = ".UD"): summarizes the information of the raster from a .UD/.UDStack object

subset signature(object = ".UD"): subsets the .UD/.UDStack object

split signature(object = ".UDStack"): splits a .UDStack into a list of .UD objects

## Note

A DBBMM and dynBGB object contains a .UD.
A DBBMMStack contains a .UDStack.
A DBBMMBurstStack contains a .UDBurstStack.
These objects can be used to program against.

## Author(s)

Bart Kranstauber & Anne Scharf

---

# .unUsedRecords-class *The .unUsedRecords and .unUsedRecordsStack class*

---

## Description

The .unUsedRecords and .unUsedRecordsStack object stores unused records in Move, MoveBurst and MoveStack objects and can be obtained with the function unUsedRecords.

## Slots

**trackIdUnUsedRecords** Object of class "factor": vector that indicates, which data, coordinates and timestamps of the unused records belong to each individual in a .unUsedRecordsStack object

**timestampsUnUsedRecords** Object of class "POSIXct": timestamps associated to the unused records.

**sensorUnUsedRecords** Object of class "factor": sensors used to record the unused records

**dataUnUsedRecords** Object of class "data.frame": data associated to the unused records

### Methods

Methods defined with class "MoveStack" in the signature:

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

`sensor` `signature(object = ".unUsedRecords")`: extracts the sensor(s) used to record the data

`timestamps` `signature(object = ".unUsedRecords")`: extracts returns or sets the timestamps

`trackId` `signature(object = ".unUsedRecordsStack")`: returning the Id of the individual per data point

### Author(s)

Marco Smolla & Anne Scharf

---

## angle *Headings of the segments of a movement track*

---

### Description

This function calculates the heading/azimuth/direction of movement of each segment between consecutive locations of a track.

### Usage

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

### Arguments

| | |
|---|---|
| x | a `move`, `moveStack` or `moveBurst` object |

### Details

Other terms for this measurment are azimuth or direction of travel/movement. The angles are relative to the North pole. The headings are calculated using the functions `bearing` of the geosphere package.

### Value

Angles in degrees (between -180 and 180). North is represented by 0.

If a `move` object is provided, a numeric vector one element shorter than the number of locations is obtained.
If a `moveStack` object is provided, a list with one element per individual containing a numeric vector one element shorter than the number of locations is obtained.

### Author(s)

Marco Smolla & Anne Scharf

### See Also

turnAngleGc

### Examples

```
## angles from a Move object
data(leroy)
head(angle(leroy))
# to add this information to the move object, a "NA" has to be assigened
# e.g. to the last location (it also could be assigend to the first location).
leroy$angles <- c(angle(leroy), NA)

## angles from a MoveStack object
data(fishers)
str(angle(fishers))
# to add this information to the moveStack object, a "NA" has to be assigened
# e.g. to the last location of each individual
fishers$angles <- unlist(lapply(angle(fishers), c, NA))
```

---

## as.data.frame *Returns a Data Frame*

---

### Description

Function to create a data.frame of a Move, dBMvariance, dBGBvariance, .unUsedRecords object.

### Usage

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

  ## S4 method for signature 'dBMvariance'
as.data.frame(x)

  ## S4 method for signature '.unUsedRecords'
as.data.frame(x)
  ## S4 method for signature '.unUsedRecordsStack'
as.data.frame(x)
```

## Arguments

| | |
|---|---|
| x | a move, moveStack, moveBurst, dBMvariance, dBMvarianceStack, dBMvarianceBurst, dBGBvariance,.unUsedRecords or .unUsedRecordsStack object |

## Details

Depending on the class of the object provided, the obtained data.frame contains the information contained in the slots:

- if class move: "timestamps", "idData", "sensor", "data", "coords".
- if class moveStack: "timestamps", "idData", "sensor", "data", "coords", "trackId".
- if class moveBurst: "timestamps", "idData", "sensor", "data", "coords", "burstId".
- if class dBMvariance: "timestamps", "sensor", "data", "coords", "window.size", "margin", "means", "in.windows", "interest".
- if class dBMvarianceStack: "timestamps", "sensor", "data", "coords", "window.size", "margin", "means", "in.windows", "interest","trackId".
- if class dBMvarianceBurst: "timestamps", "sensor", "data", "coords", "window.size", "margin", "means", "in.windows", "interest", "burstId".
- if class dBGBvariance: "timestamps", "sensor", "data", "coords", "paraSd", "orthSd", "margin", "windowSize".
- if class .unUsedRecords: "dataUnUsedRecords", "timestampsUnUsedRecords", "sensorUnUsedRecords".
- if class .unUsedRecordsStack: "trackIdUnUsedRecords", "dataUnUsedRecords", "timestampsUnUsedRecords", "sensorUnUsedRecords".

## Value

'data.frame'

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
## obtain data.frame from move object
data(leroy)
head(as.data.frame(leroy))

## obtain data.frame from moveStack object
data(fishers)
head(as.data.frame(fishers))

## obtain data.frame from .unUsedRecordsStack object
unUsedFishers <- unUsedRecords(fishers)
head(as.data.frame(unUsedFishers))
```

`brownian.bridge.dyn` *Calculates a dynamic Brownian Bridge*

## Description

This function uses a Move or MoveStack object to calculate the utilization distribution (UD) of the given track. It uses the dynamic Brownian Bridge Movement Model (dBBMM) to do so, having 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=1, dimSize=10, location.error,
                    margin=11, window.size=31, ext=.3, bbox=NA,...)
```

## Arguments

| | |
|---|---|
| `object` | a `move`, `moveStack`, `moveBurst`, `dBMvariance`, `dBMvarianceStack` object. This object must be in a projection different to longitude/latitude, use `spTransform` to transform your coordinates. |
| `raster` | a `RasterLayer` object or a numeric value. If a `RasterLayer` is provided the `brownian.bridge.dyn` starts to calculate the UD based on that raster. If a numeric value is provided it is interpreted as the resolution of the square raster cells (in map units); the according raster will be calculated internally. |
| `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. |
| `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 can be provided. |
| `margin` | The margin used for the behavioral change point analysis. This number has to be odd. |
| `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. This number has to be odd. |
| `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 .3 (extends the bounding box by 30%). 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. Only considered in combination with a numeric raster argument or the dimSize argument. |
| `...` | Additional arguments: |

`time.step` It correspond to the size of the time intervals taken for every integration step (in minutes) and thus specifies the temporal resolution of the numerical integration. If left `NULL` 15 steps are taken in the shortest time interval. See 'Details'. Optional.

`verbose` logical. default is TRUE; if FALSE printing messages about the computational size is suppressed. Optional.

`burstType` character vector with the name(s) of burstId(s) for which the UD should be calculated. This attribute can only be used if a `moveBurst` is provided in the `object` argument. Optional.

## Details

There are four ways to launch the `brownian.bridge.dyn function`:

1. Use a raster:
A RasterLayer object is set for the raster argument which is then used to calculate the UD.
(needed arguments: *object, raster(*`=RasterLayer`*), location.error, margin, window.size*; optional arguments: *time.step, verbose, burstType*)

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.
(needed arguments: *object, raster(*`=numeric`*), location.error, margin, window.size*; optional arguments: *ext, bbox, time.step, verbose, burstType*)

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.
(needed arguments: *object, dimSize, location.error, margin, window.size*; optional arguments: *ext, bbox, time.step, verbose, burstType*)

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 although changes in the setup mean the computational complexity still is calculated base on the extent but this is not informative any more on the computation time. It is only useful as a rough indication of calculation duration.

`time.step`. The default value is the shortest time interval divided by 15. This means, if there is a location recorded e.g. every 30 mins, the function divides each segment into 2 mins chunks upon which it does the calculation. If for some reason there is one time interval of 15 secs in the track, each segment of the track will be divided into 1secs chunks, increasing the calculation time immensely. Before calculating the DBBMM, use e.g. `min(timeLag(x=myMoveObject, units="mins"))` to check which is the duration of the shortest time interval of the track. If the

track contains time intervals much shorter than the scheduled on the tag, set the `time.step` e.g. to the scheduled time interval at which the tag was set, divided by 15.

## Value

'DBBMM' object, if `move` or `dBMvariance` object is provided
'DBBMMStack' object, if `moveStack` or `dBMvarianceStack` object is provided
'DBBMMBurstStack' object, if `moveBurst` object is provided

## 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 & Anne Scharf

## 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.motion.variance.dyn, getMotionVariance, getVolumeUD, contour, outerProbability, raster, raster2contour, dynBGB, dynBGBvariance

## 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=125, ext=1.2,
        time.step=2, margin=15)

plot(dbbmm)
```

`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 also 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` | a `move`, `moveStack` or `moveBurst` object can be used for variance calculation. This object must be in a flat coordinate system (projection different to longitude/latitude), use `spTransform` to transform your coordinates. |
| `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. |
| `window.size` | The size of the moving window along the track for the variance calculation. Larger windows provide more stable/accurate estimates of the brownian motion variance but are less well able to capture more frequent changes in behavior. This number has to be odd. |
| `margin` | The margin size used for variance calculation. This number has to be odd. |

## Value

'`dBMvariance`' object, if `move` object is provided
'`dBMvarianceStack`' object, if `moveStack` object is provided
'`dBMvarianceBurst`' object, if `moveBurst` object is provided

## Author(s)

Bart Kranstauber & Anne Scharf

## 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, dynBGBvariance

## 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 a specified variable

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a `move` object |
| `f` | a character, factor, or numeric vector 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 |
| `...` | Currently not implemented |

## Details

The burst function bursts (divides) a track into 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.

## Value

a '`moveBurst`' object

## Author(s)

Marco Smolla

## See Also

burstId, split, plotBursts

## Examples

```
data(leroy)
behav <- c(rep(c("B1","B2","B3","B2"),each=200), rep("B1", 118))
testb <- burst(x=leroy, f=behav)
plot(testb, type="l")
```

---

`burstId` *Returns or sets the burstId*

---

## Description

Obtain or set the ids of the behavioral categorization per segment of a MoveBurst object.

## 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 ids, either a factor or a character vector |

## Value

Returns a `factor` indicating the category of each segment.

## Author(s)

Bart Kranstauber & Anne Scharf

## See Also

burst

## Examples

```
data(leroy)
burstTrack <- burst(x=leroy,f=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 MoveStack object.

## Usage

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

## Arguments

| | |
|---|---|
| `obj` | a `move`, `moveStack` or `moveBurst` object |
| `value` | citation with class character |

## Value

character string of the citation

## Author(s)

Marco Smolla & Anne Scharf

## See Also

licenseTerms

## 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 or dynBGB object.

## Usage

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

## Arguments

| | |
|---|---|
| x | a DBBMM, DBBMMStack, dynBGB, .UD or .UDStack object |
| ... | additional arguments, like levels and nlevels, that can be passed to contour (graphics package). 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 of a DBBMM, dynBGB or .UD object by adding add = TRUE.

## Author(s)

Marco Smolla & Anne Scharf

## 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)
contour(leroydbbmm, levels=c(.5,.95))
```

## coordinates *Extract the track coordinates from a Move or MoveStack object*

### Description

The coordinates method extracts the coordinates of a track.

### Usage

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

### Arguments

| | |
|---|---|
| obj | a move, moveStack, moveBurst, dBMvariance, dBMvarianceBurst, dBMvarianceStack or dBGBvariance object |
| ... | Currently not implemented |

### Value

Returns a matrix with the coordinates of the track

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
## create a move object
data(leroy)
## extract the coordinates
head(coordinates(leroy))
```

## corridor *Corridor behavior identification*

### Description

This function identifies movement track segments whose attributes suggest corridor use behavior

### Usage

```
  ## S4 method for signature '.MoveTrackSingle'
corridor(x,speedProp=.75, circProp=.25, minNBsegments = 2, plot=FALSE, ...)
  ## S4 method for signature '.MoveTrackStack'
corridor(x,speedProp=.75, circProp=.25, minNBsegments = 2, plot=FALSE, ...)
```

## Arguments

| | |
|---|---|
| x | a move, moveStack or moveBurst object. |
| speedProp | numeric between 0 and 1, defines the proportion of speeds which are high enough to be a valid corridor point (default value selects speeds that are greater than 75 % of all speeds). |
| circProp | numeric between 0 and 1, defines the proportion of the circular variances that is low enough to be a valid corridor point. Low values of the circular variance indicate that the segments are (near) parallel (default value selects variances that are lower than 25 % of all variances). |
| minNBsegments | numeric equal or larger than 2 representing the minimum number of neiburing corridor segments that each corridor segments has to have (see Details). Default is 2. |
| plot | logical, if TRUE the track is plotted together with dots that indicate corridor points when a move object is provided (color scale indicates how many corridor points are near by, few: blue, many: pink). |
| ... | cex argument can be set specifying the size of the points when plot=TRUE. Optional. |

## Details

The corridor function uses the attributes of a movement step (segment) 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 converted into a new unit, the 'pseudo-azimuth' (0 <= 360), removing the directional information.
Subsequent, the circular variance of the pseudo-azimuths of all segment midpoints that fall within the circular buffer is calculated. Low values of the circular variance indicate that the segments are (near) parallel.
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 circular buffer there are more 'corridor' points than 'non-corridor' points.
The argument minNBsegments can be used to establish the minimum number of 'corridor' points that each circular buffer needs for the focal segment to be defined as a corridor. It is useful to exclude wrongly (visual inspection) identified corridors with only a few segments by increasing the value of minNBsegments. Note that when increasing the value of minNBsegments, only segments with enough corridor neighbors are classified as corridors and not all segments that visually seem to fit to be classified as corridors. To remove the wrongly classified corridors, a value of 3 or 4 is usually sufficient.

## Value

The function returns a moveBurst object or a list of moveBurst objects (if a MoveStack is supplied). The MoveBurst date.frame stores the following information:

- segment midpoint
- speed
- azimuth
- pseudo-azimuth
- circular variance

The object is bursted by the factor that indicates whether the segment belongs to a corridor segment or not, and is specified in the "burstId" slot.

## Note

The default values for the `speedProp` and `circProp` can be changed by the users discretion using the according argument.

## Author(s)

Marco Smolla & Anne Scharf

## 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

```
if(requireNamespace("circular") ){
## with a move object
  data(leroy)
  tmp <- corridor(leroy, plot=TRUE)
  plot(tmp, type="l", col=c("red","black")[c(tmp@burstId,NA)])

## with a moveStack object
  data(fishers)
  stacktmp <- corridor(fishers[c(1:400,sum(n.locs(fishers))-(400:1)),])
  plot(stacktmp[[2]], col=c("red","black")[stacktmp[[2]]@burstId])
  lines(stacktmp[[2]], col=c("red","black")[c(stacktmp[[2]]@burstId,NA)])
}
```

---

# DBBMM-class *The DBBMM class*

---

## Description

The `DBBMM` object is created within the brownian.bridge.dyn function from a `Move` or `dBMvariance` object. It contains a dBMvariance object and a `raster` with probabilities.

## Slots

**crs** part of the Raster-class

**data** part of the Raster-class

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

**ext** the extension factor set by the user

**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

`emd signature(object = "DBBMM")`: quantifies similarity between utilization distributions

`equalProj signature(object = "DBBMM")`: checks whether all objects of a list are in the same projection

`getMotionVariance signature(object = "DBBMM")`: extracts the estimated motion variance

`getVolumeUD signature(object = "DBBMM")`: modifies the UD raster

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

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

`raster2contour signature(object = "DBBMM")`: converts a raster to contour lines

`show signature(object = "DBBMM")`: displays summary the DBBMM object

`summary signature(object = "DBBMM")`: summarizes the information of the raster from a DBBMM object

`subset signature(object = "DBBMM")`: subsets the DBBMM object

## Note

The `DBBMM` object contains a `dBMvariance` and a `.UD` object which can be used to program against.

## Author(s)

Marco Smolla & Anne Scharf

DBBMMBurstStack-class *The DBBMMBurstStack class*

## Description

The DBBMMBurstStack object is created within the brownian.bridge.dyn function from a MoveBurst or dBMvarianceBurst object. It contains a dBMvarianceBurst object and a raster with probabilities.

## Slots

**crs** part of the Raster-class

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

**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

getMotionVariance signature(object = "DBBMMBurstStack"): extracts the estimated motion variance

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

show signature(object = "DBBMMBurstStack"): displays summary the DBBMMBurstStack object

subset signature(object = "DBBMMBurstStack"): subsets the DBBMMBurstStack object

UDStack signature(object = "DBBMMBurstStack"): creates UDStack objects

## Note

The DBBMMBurstStack object contains a dBMvarianceBurst and a .UDStack object which can be used to program against.

## Author(s)

Anne Scharf

---

`DBBMMStack-class` *The DBBMMStack class*

---

## Description

The `DBBMMStack` object is created within the brownian.bridge.dyn function from a `MoveStack` or `dBMvarianceStack` object. It contains a dBMvarianceStack object and a `raster` with probabilities.

## Slots

**crs** part of the Raster-class

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

**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

`emd` `signature(object = "DBBMMStack")`: quantifies similarity between utilization distributions

`equalProj` `signature(object = "DBBMMStack")`: checks whether all objects of a list are in the same projection

`getMotionVariance` `signature(object = "DBBMMStack")`: extracts the estimated motion variance

`getVolumeUD` `signature(object = "DBBMMStack")`: modifies the UD raster

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

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

raster2contour signature(object = "DBBMMStack"): converts a raster to contour lines

show signature(object = "DBBMMStack"): displays summary the DBBMMStack object

split signature(object = "DBBMMStack"): splits a DBBMMStack into a list of DBBMM objects

summary signature(object = "DBBMMStack"): summarizes the information of the raster from a DBBMMStack object

subset signature(object = "DBBMMStack"): subsets the DBBMMStack object

## Note

The DBBMMStack object contains a dBMvarianceStack and a .UDStack object which can be used to program against.

## Author(s)

Marco Smolla & Anne Scharf

---

# dBGBvariance-class *The dynBGBvariance class*

---

## Description

The dynBGBvariance object stores the orthogonal and parallel variances calculated by the dynamic Bivariate Gaussian Bridge model, and is created within the dynBGBvariance function from a Move object.

## Slots

**windowSize** The window size used for dynBGBvariance calculation

**margin** The margin used for dynBGB calculation

**paraSd** The standard deviation values of the parallel variance values

**orthSd** The standard deviation values of the orthogonal variance values

**nEstim** The number of windows each location was included in

**segInterest** Logical string, FALSE: segments have been omitted in the calculation since a lower number of estimates for variance are obtained for these segments. TRUE: segments included in the calculation

## Methods

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

coordinates signature(object = "dBGBvarianceTmp"): extracts the coordinates from the Move object contained in the dBGBvarianceTmp

dynBGB signature(object = "dBGBvarianceTmp"): calculates the utilization distribution (UD) of the given track using the dynamic Bivariate Gaussian Bridge model

`getMotionVariance signature(object = "dBGBvarianceTmp")`: extracts the estimated motion variance

`lines signature(object = "dBGBvarianceTmp")`: add lines of the track of the animal to a plot

`points signature(object = "dBGBvarianceTmp")`: add points of the track of the animal to a plot

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

`show signature(object = "dBGBvarianceTmp")`: displays summary the dBGBvarianceTmp object

`summary signature(object = "dBGBvarianceTmp")`: summarizes the information of the raster from a dBGBvarianceTmp object

`subset signature(object = "dBGBvarianceTmp")`: subsets the dBGBvarianceTmp object

## Note

The `dynBGBvariance` object contains a `.MoveTrackSingle` and a `dBGBvarianceTmp` object which can be used to program against.

## Author(s)

Bart Kranstauber & Anne Scharf

---

`dBMvariance` *The dBMvariance class*

---

## Description

The `dBMvariance` object is created within the brownian.motion.variance.dyn function from a `Move` object.
The `dBMvarianceStack` object is created when a `MoveStack` is the input object.
The `dBMvarianceBurst` object when the input is a `MoveBurst` object.
These objects contain the motion variance calculated by the dynamic Brownian Bridge Movement Model.

## Slots

**window.size** The window size used for dbbmm calculation

**margin** The margin used for dbbmm calculation

**means** The variance values

**in.windows** The number of windows each location was included in

**interest** Logical string, FALSE: segments have been omitted in the calculation since a lower number of estimates for variance are obtained for these segments. TRUE: segments included in the calculation

**break.list** list of the locations of breaks found

## Methods

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

`brownian.bridge.dyn` `signature(object = "dBMvarianceTmp")`: calculates the utilization distribution (UD) of the given track using the dynamic Brownian Bridge Movement Model

`coordinates` `signature(object = "dBMvarianceTmp")`: extracts the coordinates from the Move object contained in the dBMvarianceTmp

`getMotionVariance` `signature(object = "dBMvarianceTmp")`: extracts the estimated motion variance

`lines` `signature(object = "dBMvarianceTmp")`: add lines of the track of the animal to a plot

`points` `signature(object = "dBMvarianceTmp")`: add points of the track of the animal to a plot

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

`show` `signature(object = "dBMvarianceTmp")`: displays summary the dBMvarianceTmp object

`summary` `signature(object = "dBMvarianceTmp")`: summarizes the information of the raster from a dBMvarianceTmp object

`subset` `signature(object = "dBMvarianceTmp")`: subsets the dBMvarianceTmp object

## Note

The `dBMvariance` object contains a `.MoveTrackSingle` and a `dBMvarianceTmp` object.
The `dBMvarianceStack` object contains a `.MoveTrackStack` and a `dBMvarianceTmp` object.
The `dBMvarianceBurst` object contains a `.MoveTrackSingleBurst` and a `dBMvarianceTmp` object.
The class `dBMvarianceTmp` is mostly an internal class that is made public to make inheritance easier. These objects can be used to program against.

## Author(s)

Marco Smolla & Anne Scharf

---

# distance *Distance between the locations of a movement track*

---

## Description

Calculates the distance between the consecutive locations of a Move or MoveStack object.

## Usage

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

### Arguments

| | |
|---|---|
| x | a move, moveStack or moveBurst object |

### Details

pointDistance is used to calculate the distances.

### Value

Distance in map units.
If the projection of the coordinates is long/lat all values are returned in meters, otherwise in the map units of the projection of the move object. For long/lat distance is calculated on a sphere using the ellipsoid, for other projections the calculation is done on a plane using Pythagoras. Check and set the projection of your Move, MoveStack or MoveBurst object using the `proj4string()` function.

If a `move` or `moveBurst` object is provided, a numeric vector one element shorter than the number of locations is obtained.
If a `moveStack` object is provided, a list with one element per individual containing a numeric vector one element shorter than the number of locations is obtained

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
## distance from a Move object
data(leroy)
head(distance(leroy))
# to add this information to the move object, a "NA" has to be assigened
# e.g. distance is assigned to the first location of a segment
leroy$distance <- c(distance(leroy), NA)

## distance from a MoveStack object
data(fishers)
str(distance(fishers))
# to add this information to the moveStack object, a "NA" has to be assigened
# e.g. distance is assigned to the first location of a segment
fishers$distance <- unlist(lapply(distance(fishers), c, NA))
```

---

## duplicatedDataExample *Tracking data example with duplicated timestamps*

---

### Description

This file contains a data frame with fictional tracking data of two individuals, including duplicated timestamps. These data are used in the example of the function getDuplicatedTimestamps.

## Examples

```
data(duplicatedDataExample)
```

---

dynBGB *Calculation of the dynamic Bivariate Gausian Bridge*

---

## Description

This function creates a utilization distribution according to the Bivariate Gaussian Bridge model.

## Usage

```
## S4 method for signature 'dBGBvariance,RasterLayer,numeric'
dynBGB(move, raster, locErr, timeStep, ...)
## S4 method for signature '.MoveTrackSingle,RasterLayer,numeric'
dynBGB(move, raster, locErr, margin, windowSize, ...)
## S4 method for signature '.MoveTrackSingle,numeric,ANY'
dynBGB(move, raster, locErr, ext, ...)
## S4 method for signature '.MoveTrackSingle,missing,ANY'
dynBGB(move, raster, locErr, dimSize, ext, ...)
```

## Arguments

| | |
|---|---|
| move | a move or dBGBvariance object. This object must be in a projection different to longitude/latitude (one suitable for euclidean geometry), use spTransform to transform your coordinates. |
| raster | a RasterLayer object or a numeric value. If a RasterLayer is provided the dynBGB starts to calculate the UD based on that raster. If a numeric value is provided it is interpreted as the resolution of the square raster cells (in map units); the according raster will be calculated internally. |
| locErr | 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 can be provided. |
| timeStep | It correspond to the size of the timer intervals taken for every integration step (in minutes). If left NULL 20.1 steps are taken in the shortest time interval. See 'Details'. Optional. |
| margin | The margin used for the behavioral change point analysis. This number has to be odd. |
| windowSize | 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. This number has to be odd. |
| 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). Only considered in combination with a numeric raster argument or the dimSize argument. |

| | |
|---|---|
| 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. |
| ... | Currently no other arguments implemented. |

## Details

There are three ways to launch the dynBGB function:

1. Use a raster:
A RasterLayer object is set for the raster argument which is then used to calculate the UD.
(needed arguments: *move, raster(*=RasterLayer*), locErr, margin, windowSize*; optional arguments: *timeStep*)

2. Set the cell size
To set the cell size, set a numeric value for the raster argument which is used as the cell sizes of the raster.
(needed arguments: *move, raster(*=numeric*), locErr, margin, windowSize, ext*; optional arguments: *timeStep*)

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.
(needed arguments: *move, dimSize, locErr, margin, windowSize, ext*; optional arguments: *timeStep*)

timeStep. The default value is the shortest time interval divided by 20.1. This means, if there is a location recorded e.g. every 40 mins, the function divides each segment into 1.99 mins chunks upon which it does the calculation. If for some reason there is one time interval of 20 secs in the track, each segment of the track will be divided into 1secs chunks, increasing the calculation time immensely. Before calculating the DBBMM, use e.g. min(timeLag(x=myMoveObject, units="mins")) to check which is the duration of the shortest time interval of the track. If the track contains time intervals much shorter than the scheduled on the tag, set the timeStep e.g. to the scheduled time interval at which the tag was set, divided by 20.1.

## Value

It returns an object of the class dynBGB-class.

## Author(s)

Bart Kranstauber & Anne Scharf

## 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.

## See Also

dynBGBvariance, getMotionVariance, getVolumeUD, contour, outerProbability, raster, raster2contour, brownian.bridge.dyn, brownian.motion.variance.dyn

## 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=0.5, windowSize=31, margin=15, timeStep=15/20.1)
plot(dBGB, col=hsv(sqrt(1:700/1000)))
lines(dataAeqd)
```

---

# dynBGB-class *The dynBGB class*

---

## Description

The dynBGB object is created within the dynBGB function from a Move object. It contains a dBGB-variance object and a raster with probabilities.

## Slots

**crs** part of the Raster-class

**data** part of the Raster-class

**var** Object of class "dBGBvariance": includes the windowSize, margin, paraSd,orthSd, nEstim, segInterest

**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. dynBGB

**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 = "dynBGB"): adds a contour line to a plot

equalProj signature(object = "dynBGB"): checks whether all objects of a list are in the same projection

getMotionVariance signature(object = "dynBGB"): extracts the estimated motion variance

getVolumeUD signature(object = "dynBGB"): modifies the UD raster

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

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

raster2contour signature(object = "dynBGB"): converts a raster to contour lines

show signature(object = "dynBGB"): displays summary the dynBGB object

summary signature(object = "dynBGB"): summarizes the information of the raster from a dynBGB object

subset signature(object = "dynBGB"): subsets the dynBGB object

### Note

The dynBGB object contains a dBGBvariance and a .UD object which can be used to program against.

### Author(s)

Bart Kranstauber & Anne Scharf

### See Also

.UD

---

## dynBGBvariance *Calculates the Bivariate Gaussian Bridge motion variance*

---

### Description

A function to calculate the dynamic Bivariate Gaussian Bridge orthogonal and parallel variance for a movement track

### Usage

```
dynBGBvariance(move, locErr, margin, windowSize,...)
```

## Arguments

| | |
|---|---|
| move | a move object. This object must be in a projection different to longitude/latitude, use spTransform to transform your coordinates. |
| locErr | 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 can be provided. |
| margin | The margin used for the behavioral change point analysis. This number has to be odd. |
| windowSize | 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. This number has to be odd. |
| ... | Additional arguments |

## Details

The function uses windowApply with the BGBvarbreak function in order to implement a dynamic calculation of the variance

## Value

a dBGBvariance-class object

## Author(s)

Bart Kranstauber & Anne Scharf

## 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.

## See Also

dynBGB, brownian.motion.variance.dyn

## 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)

dBGBvar <- dynBGBvariance(dataAeqd, locErr=9, windowSize=31, margin=15)
dBGBvar
```

emd *Earth movers distance*

## Description

The earth mover's distance (EMD) quantifies similarity between utilization distributions by calculating the effort it takes to shape one utilization distribution landscape into another

## 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, RasterStack, RasterBrick, SpatialPoints, SpatialPointsDataFrame, DBBMM or DBBMMStack object. These objects can represent a probability surface, i.e. comparability is easiest when the sum of values is equal to 1. In the case of a SpatialPointsDataFrame the first column of the data is used as weights. In the case of SpatialPoints all points are weighted equally. |
| y | same class as provided in 'x', with the exception of RasterStack, RasterBrick and DBBMMStack, where in order to compare the utilization distributions stored within the layers of one object this argument can be left empty. Alternatively another set of rasters can be provided to compare with. |
| gc | logical, if FALSE (default) euclidean distances are calculated, if TRUE great circle distances will be used. See 'Details'. |
| threshold | numeric, the maximal distance (in map units) over which locations are compared. |
| ... | Currently not used |

## Details

For easy interpretation of the results the utilization distributions objects compared should represent a probability surface, i.e. the sum of their values is equal to 1. Nevertheless there is also the possibility to provide utilization distributions objects with the same volume, i.e. the sum of their values is equal to the same number. In the later case interpretation of the results is probably less intuitive.

Euclidean distances are suitable for most planar spatial projections, while great circle distances, calculated using the Haversine function, could be used to compare probability distributions stretching over larger geographical distances taking into account the spherical surface of the Earth.

The function can be optimized by omitting locations that have negligible contribution to the utilization density; for example, EMD can be calculated only for the cells within the 99.99% contour of the UD. This will maximally introduce a very small error in the EMD because only small amounts

of probability were omitted, but often, given the long tail of most UDs, many cells are omitted, which greatly reduces the complexity. See 'Examples'.

For more details of the method see 'References'.

## Value

An `matrix` of distances of the class 'dist'

## Author(s)

Bart Kranstauber & Anne Scharf

## References

Kranstauber, B., Smolla, M. and Safi, K. (2017), Similarity in spatial utilization distributions measured by the earth mover's distance. Methods Ecol Evol, 8: 155-160. doi:10.1111/2041-210X.12649

## Examples

```
## with a DBBMMStack object
  data(dbbmmstack)
  ## to optimize the calculation, the cells outside of the 99.99% UD contour
  # are removed by setting them to zero.
  values(dbbmmstack)[values(getVolumeUD(dbbmmstack))>.999999]<-0
  ## transform each layer to a probability surface (i.e. sum of their values is 1)
  stk<-(dbbmmstack/cellStats(dbbmmstack,sum))
  emd(stk[[1]],stk[[2]])
  emd(stk)
  emd(stk, threshold=10000)

## with a SpatiaPointsDataFrame
  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)

## with a DBBMMBurstStack object, to compare the utilization
# distributions of e.g. different behaviors
  data(leroy)
  leroyB <- burst(x=leroy,f=c(rep(c("Behav.1","Behav.2"),each=400),rep("Behav.1", 118)))
  leroyBp <- spTransform(leroyB, CRSobj="+proj=aeqd +ellps=WGS84", center=TRUE)
 leroyBdbb <- brownian.bridge.dyn(object=leroyBp[750:850], location.error=12, raster=600,
                                  ext=.45, time.step=15/15, margin=15)

  ## transform the DBBMMBurstStack into a UDStack
  leoryBud <- UDStack(leroyBdbb)
  ## to optimize the calculation, the cells outside of the 99.99% UD contour
  # are removed by setting them to zero.
  values(leoryBud)[values(getVolumeUD(leoryBud))>.999999]<-0
  ## transform each layer to a probability surface (i.e. sum of their values is 1)
  stk2<-(leoryBud/cellStats(leoryBud,sum))
```

```
emd(stk2)
```

---

`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 of class `raster`, `move`, `moveStack`, `moveBurst`, `DBBMM`, `DBBMMStack`, `DBBMMBurstStack`, `dynBGB` |

## 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 `proj4string`.

## Value

TRUE or FALSE
It returns TRUE if none of the objects have a `proj4string`.

## Author(s)

Bart Kranstauber & Anne Scharf

## Examples

```
data(fishers)
ricky<-fishers[['Ricky.T']]
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 (2013) Animal Behavior, Cost-Based Corridor Models, and Real Corridors. Landscape Ecology 28, 8: 1615-1630. doi:10.1007/s10980-013-9910-0.

### Examples

```
data(fishers)
```

## getDataRepositoryData *Download data from the Movebank Data Repository*

### Description

Download data from the Movebank Data Repository via DOI

### Usage

```
getDataRepositoryData(x, ...)
```

### Arguments

| | |
|---|---|
| x | character string of the DOI of data stored on the Movebank Data Repository |
| ... | Currently not used |

### Details

This function downloads data stored in the Movebank Data Repository via the DOI. The output is MoveStack object containing the location data from all available sensors in the study. The non-location senor data are stored in the `UnUsedRecords` slots. Datasets without location data are excluded.

If duplicated timestamps are present in the data, the first one is chosen by default. To use a more informed approach you can download the data of interest from the Movebank Data Repository, read it in with `read.csv` and use the function getDuplicatedTimestamps to located the duplicated timestamps and then decide which one to keep. And then use the function move to create a Move or MoveStack object from the cleaned .csv file.

### Value

move or moveStack object

### Note

Visit the dataset's repository page at http://dx.doi.org/<doi> for citations and a readme that might contain additional details needed to understand the data. If analyzing these published datasets, always consult the related papers and cite the paper and dataset. If preparing analysis for publication, also contact the data owner if possible for their contribution.

### Author(s)

Anne Scharf

### See Also

getMovebankData, getMovebankNonLocationData, getMovebank, move

### Examples

```
## Not run:
getDataRepositoryData("doi:10.5441/001/1.2k536j54")

## End(Not run)
```

---

## getDuplicatedTimestamps

*Identifies duplicated timestamps*

---

### Description

Identifies all pairs of duplicated timestamps within an individual and sensor type from data downloaded from Movebank or own data.

## Usage

```
  ## S4 method for signature 'character'
getDuplicatedTimestamps(x, ..., onlyVisible = TRUE)

  ## S4 method for signature 'factor'
getDuplicatedTimestamps(x, timestamps, sensorType, visible=NULL, ...)
```

## Arguments

| | |
|---|---|
| x | full path to the csv (or compressed) file location downloaded from a Movebank study, or to the zip file location downloaded from the EnvData tool in Movebank.<br>data.frame read into R from a csv file downloaded from Movebank, or downloaded with getMovebankLocationData.<br>factor containing the name(s) of the individual(s) if non-Movebank data are provided. |
| timestamps | vector containing timestamps with POSIXct conversion if non-Movebank data are provided,i.e. as.POSIXct(data$timestamp, format="%Y-%m-%d %H:%M:%S", tz="UTC") |
| sensorType | optional, character or vector of characters containing sensor type(s) if non-Movebank data are provided. |
| onlyVisible | logical, indicating if the visible column in the movebank data should be considered when the column visible is present, the default is to ignore all non-visible/outlier locations |
| visible | optional, a logical vector indicating the locations that should be considered. |
| ... | currently not implemented |

## Details

If own data (non-Movebank) are used, the vectors specified in "x", "timestamps" and optionally "visible" have to have the same length.

## Value

This function returns a list. The name of the list elements contains the individual's name, the timestamp that is duplicated and the sensor type (if provided). Each list element contains a vector with the corresponding row numbers where the duplicated timestamps are located in the table. If no duplicated timestamps are found NULL is returned.

## Author(s)

Anne Scharf

## Examples

```
data(duplicatedDataExample)
getDuplicatedTimestamps(x=as.factor(duplicatedDataExample$individual.id),
                        timestamps=as.POSIXct(duplicatedDataExample$timestamps,
```

```
            format="%Y-%m-%d %H:%M:%S", tz="UTC"),
                        sensorType=duplicatedDataExample$sensor.type)

filePath<-system.file("extdata","leroy.csv.gz",package="move")
getDuplicatedTimestamps(filePath)
```

---

`getMotionVariance` *Extracts the estimated motion variance*

---

## Description

This function returns the estimated motion variance calculated by the dynamic Bivariate Gaussian Bridges or dynamic Brownian Bridges

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a `DBBMM`, `DBBMMStack`, `DBBMMBurstStack`,`dBMvariance`, `dBMvarianceBurst`, `dBMvarianceStack`, `dynBGB` or `dBGBvariance` object |
| `...` | Currently not implemented |

## Value

- a `numeric` vector of variances if a DBBMM, DBBMMBurstStack, dBMvariance or dBMvarianceBurst object is provided
- a `list` of variances per individual if a DBBMMStack or dBMvarianceStack object is provided
- a `matrix` of the orthogonal and parallel variances if dynBGB or dBGBvariance object is provided

## Author(s)

Bart Kranstauber & Anne Scharf

## See Also

`brownian.bridge.dyn`, `dynBGB`, `brownian.motion.variance.dyn`, `dynBGBvariance`, `dBMvariance-class`, `dBGBvariance-class`

## Examples

```
data(leroydbbmm)
data(dbbmmstack)
getMotionVariance(leroydbbmm)[1:50] ## with a DBBMM object
str(getMotionVariance(dbbmmstack)) ## with a DBBMMStack object
```

getMovebank *Download Data from Movebank*

## Description

An enhanced function to download information of studies, animals, deployments and tags, and sensor measurements from Movebank. Many of the options of this function have been included as separate more user friendly functions listed in the *See Also* section below.

## Usage

```
## S4 method for signature 'character,MovebankLogin'
getMovebank(entity_type , login, ...)
```

## Arguments

| | |
|---|---|
| entity_type | character. The entity type to download from movebank, possible options are: "tag_type", "study", "tag", "individual", "deployment" or "event". See 'Details' for more information. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password. |
| ... | Arguments passed on to the Movebank API: |

i_am_owner logical. If TRUE all studies the user is a data manager for will be returned. Optional.

study_id numeric. It is the *Movebank ID* of the study. It can be obtained on the *Study Details* page on Movebank or with getMovebankID.

individual_id numeric. It is the internal individual Movebank identifier. A single individual or a vector of several individuals from the same study can be specified. It corresponds to the *id* values of getMovebank("individual", login, study_id). Optional.

deployment_id numeric. It is the deployment Movebank identifier. A single deployment or a vector of several deployments from the same study can be specified. It corresponds to the *id* values of getMovebank("deployment", login, study_id). Optional.

sensor_type_id numeric. It is the numeric id of the sensor type. A single sensor type or a vector of several sensor types can be specified. The corresponding numeric id for each sensor type can be found through getMovebank("tag_type", login). To obtain the sensor types available in the study use getMovebankSensors. Optional.

attributes character. A single attribute, a vector of attributes or "all" can be specified. Optional. See 'Details' for more information.

timestamp_start, timestamp_end character or POSIXct. Starting and/or ending timestamp to download the data for a specific time period. Timestamps have to be provided in format 'yyyyMMddHHmmssSSS'. If POSIXct then it is converted to character using UTC as a time zone, note that this can change the time. Optional.

## Details

- `getMovebank("tag_type", login)`: returns all sensor types in Movebank and their corresponding sensor id. See also getMovebankSensors.
- `getMovebank("study", login)`: returns all studies where the user has permission to see the data. You may have permission to see only the study details, view some or all tracks but not download data, or view and download some or all data. Also, there are studies that you do not have permission to see at all, these studies will not be included in this list. See also getMovebankStudies.
- `getMovebank("study", login, i_am_owner=T)`: returns all studies where the user is a data manager.
- `getMovebank("study", login, study_id)`: returns a summary of information about one or more studies. See also getMovebankStudy.
- `getMovebank("tag", login, study_id)`: returns tag reference information from a study. See also getMovebankReferenceTable.
- `getMovebank("individual", login, study_id)`: returns animal reference information from a study. See also getMovebankAnimals, getMovebankReferenceTable.
- `getMovebank("deployment", login, study_id)`: returns deployment reference information from a study. See also getMovebankReferenceTable.
- `getMovebank("event", login, study_id,...)`: returns the sensor measurements from a study. See also getMovebankData, getMovebankLocationData, getMovebankNonLocationData.

The default columns of `getMovebank("event", login,study_id,...)` are *timestamp, location_lat, location_long, individual_id, tag_id*. If the downloaded study only contains GPS data, these default columns are suitable, but for all other sensors, additional columns are required. The columns available vary among sensor type and tag manufacturer, the complete list of available attributes for a specific study can be obtained with: `getMovebankSensorsAttributes(study, login=login)`. If `attributes="all"` than all attributes that are present in the study will be downloaded.

The definitions of the content of the columns is detailed in the Attribute Dictionary on Movebank

## Value

'`data.frame`'

## Note

- 'id' in `getMovebank("study", login, ...)` are the values required in `study_id`
- 'id' in `getMovebank("individual", login, study_id)` are the values required in `individual_id`
- 'id' in `getMovebank("deployment", login, study_id)` are the values required in `deployment_id`
- 'id' in `getMovebank("tag_type", login)` are the values required in `sensor_type_id`
- 'id' in `getMovebank("tag", login, study_id)` corresponds to 'tag_id'

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

If the data include double timestamps you can use the `getDuplicatedTimestamps` function to identify them and decide which one to keep.

## Author(s)

Marco Smolla & Anne Scharf

## See Also

movebankLogin, getMovebankData, getMovebankLocationData, getMovebankNonLocationData, getMovebankReferenceTable, getMovebankAnimals, getMovebankID, getMovebankSensors, getMovebankSensorsAttributes, getMovebankStudies, getMovebankStudy, searchMovebankStudies

## Examples

```
## Not run:

## first create the login object
login <- movebankLogin()

anne## get Movebank ID from study
studyID <- getMovebankID(study="MPIAB white stork lifetime tracking data (2013-2014)",
                         login=login)
studyID2 <- getMovebankID(study="Ocelots on Barro Colorado Island, Panama", login=login)

## get a summary of information about the two studies
getMovebank("study", login=login, study_id=c(studyID,studyID2))

## get tag reference information from the study
head(getMovebank("tag", login=login, study_id=studyID))

## get animal reference information from the study
head(getMovebank("individual", login=login, study_id=studyID))

## get deployments reference information from the study
head(getMovebank("deployment", login=login, study_id=studyID))

## get the sensor measurements from the study
## find out which sensors were used in this study
unique(getMovebankSensors(study=studyID,login=login)$sensor_type_id)
## get movebank ID of one individual of this study
indID <- getMovebank("individual", login=login, study_id=studyID)$id[50]
## the correspondence table between the individual ID and the
## animal names can be obtained like this
head(getMovebank("individual", login=login,
                 study_id=studyID)[, c("id", "local_identifier")])

## get GPS and accelerometer data within a time period
## to download all attributes for all sensors included in the study
attrib <- "all"
## get measurments for a given time period, in this case for GPS and
## accelerometer, and between "2013-06-25 03:55:00.000" and "2013-06-26 10:25:00.000"
getMovebank("event", login=login, study_id=studyID, sensor_type_id=c(653,2365683),
            individual_id=indID, attributes=attrib, timestamp_start="20130625035500000",
            timestamp_end="20130626102500000 ")
```

```
## get all GPS data for 2 individuals
## get movebank ID of another individual of this study
indID2 <- getMovebank("individual", login=login, study_id=studyID)$id[35]
## get GPS measurements for these two individuals with all available attributes
head(storks <- getMovebank("event", login=login, study_id=studyID,
                           sensor_type_id=653, individual_id=c(indID,indID2),
                           attributes="all"))

## create moveStack
## get the names of the individuals as they appear on Movebank
(individualNames <- getMovebank("individual", login=login,
                                study_id=studyID)[c(35,50), c("id", "local_identifier")])
head(storks2 <- merge(storks,individualNames,by.x="individual_id", by.y="id"))

myMoveStack <- move(x=storks2$location_long, y=storks2$location_lat,
                    time=as.POSIXct(storks2$timestamp, format="
                    data=storks2,
                    proj=CRS("+proj=longlat +ellps=WGS84"),
                    animal=storks2$local_identifier)

plot(myMoveStack, type="l")

## End(Not run)
```

---

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

---

## Description

This function returns information of the animals, their tags and IDs from a Movebank study.

## Usage

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

## Arguments

| | |
|---|---|
| `study` | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with `getMovebankID`. |
| `login` | a `MovebankLogin` object, if empty you'll be asked to enter your username and password |

## Details

`getMovebankAnimals` belongs to the Movebank browsing functions and returns a `data.frame` from the requested study that includes among others the `individual_id`, `tag_id`, `deployment_id`, `sensor_type_id` which are the internal ids of Movebank, the `tag_local_identifier`, `local_identifier` which are the ids uploaded to Movebank by the user and other information if available as e.g. `death_comments`, `sex`, `individual_taxon_canonical_name`, etc.

## Value

'data.frame'

## Note

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

## Author(s)

Marco Smolla & Anne Scharf

## See Also

movebankLogin, getMovebankReferenceTable

## Examples

```
## Not run:

# obtain a login
login<-movebankLogin()
getMovebankAnimals(study=2950149, login=login)

## End(Not run)
```

---

# getMovebankData *Download data from Movebank as a Move object*

---

## Description

This function downloads the location data and timestamp of a study stored in Movebank as a move/moveStack object

## Usage

```
## S4 method for signature 'numeric,character,MovebankLogin'
getMovebankData(study, animalName, login, ...)

## S4 method for signature 'numeric,numeric,MovebankLogin'
getMovebankData(study, animalName, login,
                removeDuplicatedTimestamps=FALSE,
                includeExtraSensors=FALSE, deploymentAsIndividuals=FALSE,
                includeOutliers=FALSE,...)
```

## Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |
| animalName | character. Name of the individuals as stored on Movebank. A single individual or a vector of several individuals from the same study can be specified. Optional. |
| includeExtraSensors | logical; if TRUE data from non location sensors included in the study will be also downloaded, the data will automatically be stored in the unUsedRecords slot as they cannot produce locations. See 'Details'. |
| removeDuplicatedTimestamps | logical; if TRUE duplicated timestamps values will be removed. See 'Note'. |
| deploymentAsIndividuals | logical; if TRUE the deployments will be downloaded separately. See 'Details'. |
| includeOutliers | logical; if TRUE locations marked as outliers in Movebank will be included in the regular trajectory otherwise as unUsedRecords. See 'Details' |
| ... | Additional arguments passed on to the movebank API through getMovebank function: |
| | timestamp_start, timestamp_end character or POSIXct. Starting and/or ending timestamp to download the data for a specific time period. Timestamps have to be provided in format 'yyyyMMddHHmmssSSS'. If POSIXct then it is converted to character using UTC as a time zone, note that this can change the time. Optional. |

## Details

getMovebankData belongs to the Movebank browsing functions and returns a Move object from studies with only one animal or a MoveStack object for studies with multiple animals.
Remember that you need an account on Movebank, see movebankLogin.

*Attribute names:*
The definitions of the content of the columns within the @idData, @sensor, @data slots of the move or moveStack object is detailed in the Attribute Dictionary on Movebank

includeExtraSensors:
If this includeExtraSensors=TRUE the data of all non location sensors (e.g. acceleration, magnetometer, etc) available in the study will be downloaded and stored in the unUsedRecords slot. Data from a single or a set of non location sensors can be also downloaded as a data.frame with the function getMovebankNonLocationData

deploymentAsIndividuals:
If single individuals have several deployments, and these are wished to be downloaded separately, this can be done by setting deploymentAsIndividuals=TRUE. In this case the "@trackId" will contain the names of the deployments.

*idData:*
The idData slot contains only the information of the animals. To obtain information on tags, deployments and sensors of the study use the function getMovebankReferenceTable.
When `deploymentAsIndividuals=TRUE` than the idData slot contains the information of the deployments.

## Value

Object of class 'Move' or 'MoveStack'

## Note

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

`removeDuplicatedTimestamps`:
It is possible to set `removeDuplicatedTimestamps=TRUE` which allows you delete the duplicated timestamps in case your data set contains them. Using this argument will retain the first of multiple records with the same animal ID and timestamp, and remove any subsequent duplicates. In case you want to control which of the duplicate timestamps are kept and which are deleted, we recommend to download the data as a .csv file from Movebank or to use the function getMovebankLocationData, find the duplicates using e.g. getDuplicatedTimestamps, decide which of the duplicated timestamp to retain, and than create a move/moveStack object with the function move. Another option is to edit the records in movebank and mark the appropriate records as outliers.

*includeOutliers:*
In Movebank outliers can be marked manually or using filters, including a duplicate filter that flags duplicate records based on user-selected attributes, retaining the first record of each duplicate set that was imported to the study. When `includeOutliers=FALSE` (default) these records are automatically placed in the UnusedRecords slots. If `includeOutliers=TRUE` these records are included along all other locations. This option can be useful if the user wants to e.g. implement their own filter/algorithm to identify outliers. Entries that contain NAs in the coordinate columns will always be included in the UnusedRecords slots.

*Multiple sensors:*
The `getMovebankData` function downloads the data of all location sensors available in the study. If the study contains several location sensors, the resulting move/moveStack object can be separated into a move/moveStack object per sensor type:
`x[x@sensor=="z"]` where "x" is a Move or a MoveStack object, and "z" is the name of the sensor e.g. "GPS", "Radio Transmitter", etc.

*Downloading a study with many locations:*
If the study to be downloaded has many locations (probably in the order of 10s of millions), the download may take so long that the connection breaks, and the study cannot be downloaded. We recommend to download each individual separately to ensure a successfully download. See more details and examples in the 'browseMovebank' vignette.

## Author(s)

Marco Smolla & Anne Scharf

## See Also

movebankLogin,getMovebankLocationData, getMovebankNonLocationData

## Examples

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

# returns a MoveStack object from the specified study
getMovebankData(study="Ocelots on Barro Colorado Island, Panama", login=login)

# returns a Move object (there is only one individual in this study)
getMovebankData(study="Coatis on BCI Panama (data from Powell et al. 2017)", login=login)

# returns a MoveStack with two individuals
getMovebankData(study=123413, animalName=c("Mancha","Yara"), login=login)

# Get a specific timerange, eg: all positions untill "2003-05-06 19:45:10.000"
(ocelots <- getMovebankData(study=123413, animalName=c("Mancha","Yara"),
                            login=login, timestamp_end="20030506194510000"))

timestamps(ocelots)
## End(Not run)
```

---

# getMovebankID *Movebank Study ID*

---

## Description

This function returns the numeric Movebank ID of the study which 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 | a MovebankLogin object, if empty you'll be asked to enter your username and password |

## Details

getMovebankID belongs to the Movebank browsing functions and returns the Movebank ID of a study as it is stored on Movebank. This number can also be found on the *Study Details* page of the study on Movebank.

## Value

The function returns one 'numeric' value.

## Note

The character study name on Movebank can be potentially edited and changed at any time by the *Data Manager(s)*, whereas the *Movebank ID* is uniquely assigned to each study when it is uploaded to Movebank, and cannot be modified afterwards.

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

## Author(s)

Marco Smolla & Anne Scharf

## See Also

movebankLogin

## Examples

```
## Not run:

#obtain a login
login<-movebankLogin()
getMovebankID(study="Ocelots on Barro Colorado Island, Panama", login=login)

## End(Not run)
```

---

# getMovebankLocationData

*Download location data from Movebank as a table*

---

## Description

This function downloads the location data for one or several sensors of a study stored in Movebank.

## Usage

```
## S4 method for signature 'numeric,numeric,character,MovebankLogin'
getMovebankLocationData(study, sensorID, animalName, login, ...)

## S4 method for signature 'numeric,numeric,numeric,MovebankLogin'
getMovebankLocationData(study, sensorID, animalName, login,
                        includeOutliers=FALSE, underscoreToDots=TRUE, ...)
```

## Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |
| sensorID | character or numeric. Name or ID number of sensor(s) recording location data. A single sensor or a vector of sensors can be specified. If the argument is left empty data of all location sensors will be downloaded. Optional. See 'Details'. |
| animalName | character. Name of the individuals as stored on Movebank. A single individual or a vector of several individuals from the same study can be specified. If the argument is left empty data of all individuals will be downloaded. Optional. |
| includeOutliers | logical. If TRUE locations marked as outliers in Movebank will be included. Default is FALSE. |
| underscoreToDots | logical. Many of the functions in the *Move* package rely on the column names containing dots and not underscores. Default is TRUE. See 'Details'. |
| ... | Additional arguments passed on to the movebank API through getMovebank function: |
| | timestamp_start, timestamp_end character or POSIXct. Starting and/or ending timestamp to download the data for a specific time period. Timestamps have to be provided in format 'yyyyMMddHHmmssSSS'. If POSIXct then it is converted to character using UTC as a time zone, note that this can change the time. Optional. |

## Details

getMovebankLocationData belongs to the Movebank browsing functions and returns a data.frame with data from one or multiple location sensors from studies with one animal or multiple animals.
Remember that you need an account on Movebank, see movebankLogin.
Note that getMovebankData has also the option to download location data directly into a move/moveStack object.

*Attribute names:*
The definitions of the content of the columns of the returned data.frame is detailed in the Attribute Dictionary on Movebank. The attributes deployment_id, individual_id, tag_id, study_id correspond to the internal ids of Movebank.

sensorID:
See getMovebankSensors to obtain all available sensors of the study of interest. The valid names for this argument are those of the the columns "name" or "id" of the table returned by getMovebankSensors(login). The valid numeric Ids are also in the column "sensor_type_id" in the table returned for a specific study with getMovebankSensors(study,login). This function only accepts location sensors which are marked as "true"" in the "is_location_sensor" column of the table returned by getMovebankSensors(login).

underscoreToDots:
.csv files downloaded from the Movebank webpage contain dots in their column names, and .csv

files downloaded via the API (like in the case of this function) contain instead underscores in their column names. Many of the functions in the *Move* package were created based on the webpage csv downloaded data and rely on the column names with dots. If you would like to use function like e.g. `getDuplicatedTimestamps` or read in the csv file with `move` by stating the path to file, among others, than the column names have to be with dots.

*Downloading a study with many locations:*
If the study to be downloaded has many locations (probably in the order of 10s of millions), the download may take so long that the connection breaks, and the study cannot be downloaded. We recommend to download each individual separately to ensure a successfully download. See more details and examples in the 'browseMovebank' vignette.

## Value

'`data.frame`'

## Note

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

## Author(s)

Anne Scharf

## See Also

movebankLogin, getMovebankData, getMovebankNonLocationData

## Examples

```
## Not run:
## first create the login object
login <- movebankLogin()

## get GPS data for one individual
str(getMovebankLocationData(study=74496970, sensorID="GPS",
                            animalName="DER AR439", login=login))

## get GPS data for one individual after the "2013-07-12 06:50:07.000"
str(getMovebankLocationData(study=74496970, sensorID="GPS", animalName="DER AR439",
                            login=login, timestamp_start="20130712065007000"))

# get GPS data for all individuals of the study between
# the "2013-08-15 15:00:00.000" and "2013-08-15 15:01:00.000"
str(getMovebankLocationData(study=74496970, sensorID=653,
                            login=login, timestamp_start="20130815150000000",
                            timestamp_end="20130815150100000"))

## End(Not run)
```

getMovebankNonLocationData
*Download non-location data from Movebank*

## Description

This function downloads the non location data for one or several sensors of a study stored in Movebank

## Usage

```
## S4 method for signature 'numeric,numeric,character,MovebankLogin'
getMovebankNonLocationData(study, sensorID, animalName, login, ...)

## S4 method for signature 'numeric,numeric,numeric,MovebankLogin'
getMovebankNonLocationData(study, sensorID, animalName, login, ...)
```

## Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |
| sensorID | character or numeric. Name or ID number of sensor(s) recording non location data. A single sensor or a vector of sensors can be specified. If the argument is left empty data of all non location sensors will be downloaded. Optional. See 'Details'. |
| animalName | character. Name of the individuals as stored on Movebank. A single individual or a vector of several individuals from the same study can be specified. If the argument is left empty data of all individuals will be downloaded. Optional. |
| ... | Additional arguments passed on to the movebank API through getMovebank function: |
| | timestamp_start, timestamp_end character or POSIXct. Starting and/or ending timestamp to download the data for a specific time period. Timestamps have to be provided in format 'yyyyMMddHHmmssSSS'. If POSIXct then it is converted to character using UTC as a time zone, note that this can change the time. Optional. |

## Details

getMovebankNonLocationData belongs to the Movebank browsing functions and returns a data.frame with data from one or multiple non-location sensors from studies with one animal or multiple animals.
Remember that you need an account on Movebank, see movebankLogin.

Note that getMovebankData has also the option to download non location alongside with the location data.

*Attribute names:*
The definitions of the content of the columns of the returned `data.frame` is detailed in the Attribute Dictionary on Movebank

`sensorID`:
See getMovebankSensors to obtain all available sensors of the study of interest. The valid names for this argument are those of the the columns "name" or "id" of the table returned by `getMovebankSensors(login)`. The valid numeric Ids are also in the column "sensor_type_id" in the table returned for a specific study with `getMovebankSensors(study,login)`. This function only accepts non-location sensors which are marked as "false"" in the "is_location_sensor" column of the table returned by `getMovebankSensors(login)`.

## Value

'`data.frame`'

## Note

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

*Downloading a study with a lot of data:*
If the study to be downloaded has many locations (probably in the order of 10s of millions), the download may take so long that the connection breaks, and the study cannot be downloaded. We recommend to download each individual separately to ensure a successfully download. See more details and examples in the 'browseMovebank' vignette.

## Author(s)

Anne Scharf

## See Also

movebankLogin, getMovebankData, getMovebankLocationData

## Examples

```
## Not run:
## first create the login object
login <- movebankLogin()

## get acceleration data for one individual
str(getMovebankNonLocationData(study=74496970 , sensorID="Acceleration",
                               animalName="DER AR439", login=login))

## get acceleration data for one individual after the "2013-07-12 06:50:07.000"
str(getMovebankNonLocationData(study=74496970 , sensorID="Acceleration", animalName="DER AR439",
                               login=login, timestamp_start="20130712065007000"))

# get acceleration data for all individuals of the study between
```

```
# the "2013-08-15 15:00:00.000" and "2013-08-15 15:01:00.000"
str(getMovebankNonLocationData(study=74496970 , sensorID=2365683,
                               login=login, timestamp_start="20130815150000000",
                               timestamp_end="20130815150100000"))

## End(Not run)
```

---

`getMovebankReferenceTable` *Download all reference data of a Movebank study*

---

## Description

This function returns the information of the animals, tags, deployments and sensors from a Movebank study

## Usage

```
getMovebankReferenceTable(study, login, allAttributes = FALSE)
```

## Arguments

| | |
|---|---|
| `study` | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with `getMovebankID`. |
| `login` | a `MovebankLogin` object, if empty you'll be asked to enter your username and password |
| `allAttributes` | logical. If FALSE the output will only include the attributes that currently contain information in the study (default). If TRUE the output will include all attributes available on Movebank. |

## Details

`getMovebankReferenceTable` belongs to the Movebank browsing functions and returns a `data.frame` from the requested study, including all data provided by the user referring to the animals, tags and deployments. It also includes `animal_id`, `tag_id`, `deployment_id`, `sensor_type_id`, `study_id` which are the internal ids of Movebank. This table is equivalent to the table obtained on the Movebank webpage trough the option "Download Reference Data" of the study.

## Value

'`data.frame`'

## Note

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

## Author(s)

Anne Scharf

## See Also

movebankLogin

## Examples

```
## Not run:
# obtain a login
login <- movebankLogin()
getMovebankReferenceTable(study=74496970, login=login)[1:6,]
getMovebankReferenceTable(study=74496970, login=login, allAttributes=T)[1:6,]

## End(Not run)
```

---

# getMovebankSensors *Information about Movebank sensors*

---

## Description

This function returns the sensor types used in a Movebank study.

## Usage

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

## Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |

## Details

getMovebankSensors belongs to the Movebank browsing functions and returns the sensor type(s) used for each tag_id within the specified study.

If the study argument is missing, information about all sensor types available on Movebank and the correspondence between sensor_type_id and the sensor name is obtained.

## Value

'data.frame'

### Note

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

### Author(s)

Marco Smolla & Anne Scharf

### See Also

movebankLogin

### Examples

```
## Not run:

## obtain a login
login<-movebankLogin()

## obtain sensors types of each tag in the specified study
getMovebankSensors(study=2950149, login=login)

## obtain all sensors available on Movebank
getMovebankSensors(login=login)

## End(Not run)
```

---

## getMovebankSensorsAttributes

*Available attributes of Movebank sensors*

---

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |
| ... | Extra arguments passed on to the getMovebank function |

## 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 653 as their ID on Movebank).

The definition of each of the attributes is detailed in the Attribute Dictionary on Movebank

The correspondence between the sensor type and the sensor type id can be found with the function getMovebankSensors, leaving the study argument empty.

## Value

'data.frame'

## Note

See the 'browseMovebank' vignette 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=2950149, login=login)

## End(Not run)
```

---

# getMovebankStudies *All studies on Movebank*

---

## Description

This function returns all studies available on Movebank.

## Usage

```
getMovebankStudies(login)
```

### Arguments

| | |
|---|---|
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |

### Details

getMovebankStudies belongs to the Movebank browsing functions and returns all studies available on Movebank.

### Value

returns an object of class 'factor' with the names of all studies available on Movebank.

### Note

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

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
## Not run:

# obtain a login
login <- movebankLogin()
getMovebankStudies(login=login)

## End(Not run)
```

---

## getMovebankStudy *Returns information of a Movebank study*

---

### Description

This function returns information about the requested study as e.g. the authors of the study, licence type, citation and more.

### Usage

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

### Arguments

| | |
|---|---|
| study | character or numeric. Character: full name of the study, as stored on Movebank. Numeric: *Movebank ID* of the study which can be obtained on the *Study Details* page on Movebank or with getMovebankID. |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and password |

### Details

getMovebankStudy belongs to the Movebank browsing functions and returns a data.frame with information about the requested study (e.g.: authors of the study, licence type, citation, etc).

### Value

'data.frame'

### Note

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

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
## Not run:
# obtain a login
login<-movebankLogin()
getMovebankStudy(study="Coatis on BCI Panama (data from Powell et al. 2017)", login=login)

## End(Not run)
```

---

## getVolumeUD *Utilization distribution (UD)*

---

### Description

The UD represents the minimum area in which an animal has some specified probability of being located (Cumulative Distribution Function).

### Usage

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

## Arguments

| | |
|---|---|
| x | a DBBMM, DBBMMStack, dynBGB, .UD or .UDStack object |
| ... | when x is of class DBBMM or dynBGB, several objects of class DBBMM or dynBGB can be added (see 'Examples') |

## Value

'raster' or 'rasterStack'
If several objects are provided, a list of rasters is returned

## Note

To obtain this modified UD raster from a DBBMMBurstStack object, transform the object with the UDStack function into a '.UDStack' class, and than use the getVolumeUD function upon the obtained object.

## Author(s)

Marco Smolla & Anne Scharf

## See Also

raster2contour, contour, UDStack

## Examples

```
data(leroydbbmm)
data(dbbmmstack)
data(leroydbgb)
getVolumeUD(leroydbbmm) # for a DBBMM object
getVolumeUD(dbbmmstack) # for a DBBMMStack object
getVolumeUD(leroydbgb) # for a dynBGB object

getVolumeUD(leroydbbmm, leroydbgb) # for several objects

plot(getVolumeUD(leroydbbmm))

## e.g. select the raster corresponding to the 95% UD
leroyUD <- getVolumeUD(leroydbbmm)
leroyUD[leroyUD>0.95] <- NA
plot(leroyUD)
```

hrBootstrap *Calculates and plots the area of the Minimum Convex Polygon of a track*

## Description

The hrBootstrap function calculates the 0, 25, 50, 75, 100% percentile of the Minimum Convex Polygon (MCP) area by a logarithmic step wise increase of the number of samples per calculation. For every step this calculation is repeated `rep` times with random coordinates from the track.

## 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` | a `move`, `moveStack`, `moveBurst` 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 in each step, default is 95 (95% of all coordinates per step 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 from the input values (can be 'm' or 'km') |
| `unout` | units for the output values (can be 'm2', 'km2', or 'ha') |
| `...` | Currently not implemented |

## Details

The function calculates the 0, 25, 50, 75, 100% percentile of the Minimum Convex Polygon (MCP) area with a logarithmic step wise increase of 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 stores 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 (see 'Values'). The dot-dashed line indicates the real MCP area size of all locations.

The `hrBootstrap` function passes values (samples of the track) on to the function `mcp` that is part of the adehabitatHR package. See the help 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.

Plot legend:
- 0% percentile of mcp area: blue bottom line
- 25% percentile of mcp area: red bottom line
- 50% percentile of mcp area: black middle line
- 75% percentile of mcp area: red top line
- 100% percentile of mcp area: blue top line
- Real mcp area size of all locations: horizontal dot-dashed black line

The number of locations used in each step are printed in the console.

## Note

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

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
if(requireNamespace("adehabitatHR")){
## for a Move object
  m <- move(x=rnorm(55), y=rnorm(55), time=as.POSIXct(1:55, origin="1970-1-1"),
           proj=CRS("+proj=aeqd +ellps=WGS84"), animal='a')
  hrBootstrap(m,rep=5, level=99, unout="m2", plot=TRUE)

## for a MoveStack object
   m2 <- move(x=rnorm(30), y=rnorm(30), time=as.POSIXct(1:30, origin="1970-1-1"),
            proj=CRS("+proj=aeqd +ellps=WGS84"), animal='b')
  mstack <- moveStack(list(m[30:50,],m2))
  hrBootstrap(mstack,rep=5, unout="m2", plot=FALSE)

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

---

# idData *Obtain or replace the idData slot of a Move object*

---

## Description

This function returns or replaces the idData slot of a Move, MoveStack or MoveBurst object.

### Usage

```
  ## S4 method for signature '.MoveTrack'
idData(x,i,j,...)

  ## 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
```

### Arguments

| | |
|---|---|
| x | a move, moveStack or moveBurst object |
| i | Selection of the rows |
| j | Selection for the columns |
| value | Replacement values for the selected idData |
| ... | Other arguments to the data frame subsetting such as drop=F |

### Value

Either the idData data.frame or the modified move object

### Author(s)

Bart Kranstauber & Anne Scharf

### Examples

```
data(fishers)
idData(fishers)

## obtain e.g. only the tag and individual identifier columns
idData(fishers, j=c(6,7))
idData(fishers, j=c("tag.local.identifier", "individual.local.identifier"))
```

---

## 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 | a move or moveBurst object. |
| time | either a number of locations (class numeric), a time interval (class difftime) or a vector of timestamps (class POSIXct) at which the interpolation should be done. See 'Details'. |
| spaceMethod | a character that indicates the interpolation function (euclidean, great circle or along the rhumb line) to be used to generate the new locations. |
| ... | Currently not implemented. |

## Details

In the argument time:
- number of locations: refer the total number of locations that the resulting track will have distributed equally over time. E.g. if time=200, the resulting track will have 200 points interpolated at a constant time interval.
- time interval: refers to the time interval at which a location should be interpolated. E.g. if time=as.difftime(10, units="mins") a location will be interpolated every 10 mins.
- vector of timestamps: the timestamps of this vector have to be in ascending order, and within the time range of the track.

## Value

Move-class object of the interpolated locations.

## Author(s)

Bart Kranstauber & Anne Scharf

## Examples

```
data(leroy)
## providing the number of locations
plot(leroy[100:150,], col="red",pch=20)
points(mv <- interpolateTime(leroy[100:150,], time=500, spaceMethod='greatcircle'))

## providing a time interval
plot(leroy[100:150,], col="red",pch=20)
points(mv2 <- interpolateTime(leroy[100:150,], time=as.difftime(10, units="mins"),
                              spaceMethod='greatcircle'))

## providing a vector of timestamps
plot(leroy[100:150,], col="red",pch=20)
ts <- as.POSIXct(c("2009-02-13 10:00:00", "2009-02-13 12:00:00", "2009-02-13 14:00:00",
                   "2009-02-13 16:00:00","2009-02-13 18:00:00","2009-02-13 20:00:00",
                   "2009-02-13 22:00:00","2009-02-14 00:00:00","2009-02-14 02:00:00",
                   "2009-02-14 04:00:00","2009-02-14 06:00:00", "2009-02-14 08:00:00",
                   "2009-02-14 10:00:00"),  format="%Y-%m-%d %H:%M:%S", tz="UTC")
points(mv3 <- interpolateTime(leroy[100:150,], time=ts, spaceMethod='greatcircle'))
```

`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)
```

## leroydbgb *dynamic Bivariate Gausian Bridge example object*

### Description

Utilization densities calculated with dynBGB to exemplify functions.

### Usage

```
data("leroydbgb")
```

### Details

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

### Examples

```
data(leroydbgb)
leroydbgb
```

## licenseTerms *Extract the license terms of a Move or MoveStack object*

### Description

The licenseTerms method returns or sets the license terms of a track from a Move or MoveStack object.

### Usage

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

### Arguments

| | |
|---|---|
| obj | a move, moveStack or moveBurst object |
| value | license terms with class character |

### Value

character string of the license terms

### Author(s)

Anne Scharf

## See Also

citations

## Examples

```
data(leroy)
licenseTerms(leroy) #get the license from a Move object

## change the license and set it for a Move object
licenseTerms(leroy) <- "use of data only permitted after obtaining licence from the PI"

data(fishers)
licenseTerms(fishers) #get the license from a MoveStack object

## change the license and set it for a MoveStack object
licenseTerms(fishers) <- "use of data only permitted after obtaining licence from the PI"
```

---

# lines *Plotting the lines of a track*

---

## Description

Function for plotting a 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 | a move, moveStack, moveBurst, dBMvariance, dBMvarianceStack, dBMvarianceBurst or dBGBvariance object. |
| col | a vector of colors of the same length as the number of individual for a moveStack, or number of burst levels or of segments for a moveBurst object. If left empty the default 8 colors from R are used, which will be recycled if the object contains more individuals or burst levels (run palette() to obtain vector of default colors) |
| ... | arguments to be passed on, e.g. lty or lwd. |

## Author(s)

Marco Smolla & Anne Scharf

## See Also

points, plot

## Examples

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

## plot the points and lines of a moveStack
data(fishers)
plot(fishers, type='p',pch=4)
lines(fishers, col=3:4)
```

---

# move *Create a Move object*

---

## Description

This function creates Move or MoveStack object from a .csv file with location data downloaded from a Movebank study, from a zip file downloaded from the EnvData (environmental annotation tool) of a Movebank study, from a ltraj, telemetry, track_xyt, track or binClstPath object or from own data. If you use your own data you need to set the projection method with the 'proj' argument and specify which columns of your data contain the coordinates and timestamps.

## Usage

```
## S4 method for signature 'connection,missing,missing,missing,missing'
move(x, removeDuplicatedTimestamps=F, ...)

## S4 method for signature 'ltraj,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'telemetry,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'track_xyt,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'list,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'track,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'binClstPath,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'binClstStck,missing,missing,missing,missing'
move(x, y, time, data, proj,...)
## S4 method for signature 'data.frame,missing,missing,missing,missing'
```

```
move(x, y, time, data, proj,...)

## 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 csv (or compressed) file location downloaded from a Movebank study, OR to the zip file location downloaded from the EnvData tool in Movebank.<br>a ltraj object from the package adehabitatLT.<br>a telemetry object or list of telemetry objects from the package ctmm.<br>a track_xyt object from the package amt.<br>a track object from the package bcpa.<br>a binClstPath or a binClstStck object from the package EMbC.<br>a data.frame object downloaded from Movebank webpage or with getMovebankLocationData.<br>numeric vector with x coordinates if non-Movebank data are provided (e.g. data$x). |
| y | numeric vector with y coordinates if non-Movebank data are provided (e.g. data$y). |
| time | vector of time stamps with POSIXct conversion if non-Movebank data are provided, i.e. as.POSIXct(data$timestamp, format="%Y-%m-%d %H:%M:%S", tz="UTC") |
| data | extra data associated with the relocations, if empty it is filled with the coordinates and timestamps. Optional. |
| proj | projection method for non-Movebank data; requires a valid CRS (see CRS-class) object, e.g. CRS("+proj=longlat +ellps=WGS84"); default is NA. Optional. |
| sensor | Sensor name(s), either single character or a vector with length of the number of coordinates. If multiple sensors are provided this has to be done as a vector with the same length as the number of coordinates. Optional. |
| animal | animal ID(s) or name(s), either single character or a vector with length of the number of coordinates. If multiple individuals are provided this has to be done as a vector with the same length as the number of coordinates. Optional. |
| removeDuplicatedTimestamps | logical; if TRUE duplicated timestamps values will be removed. Only available when reading in data from movebank via path to a .csv file. Using this argument will retain the first of multiple records with the same animal ID and timestamp, and remove any subsequent duplicates. See 'Note'. |
| ... | Additional arguments |

## Details

The easiest way to import data is to download the study you are interested in from https://www.movebank.org and 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, MoveStack object. See the 'browseMovebank' vignette for more information on how to directly

download data from Movebank from within R.

Another way is to read in your data using read.csv. Then specify the arguments "x" and "y" the columns of your data containing the x and y coordinates, in the argument "time" the column containing the timestamp, optionally the columns containing the information of the sensor(s) used, the animal name(s) and the projection, as well as the whole data.frame of the imported data. If the argument "animal" is left empty or contains only the name of one animal the function will return a Move object. If the data contains multiple animal names the function will return a MoveStack object.

## Value

returns an object of class 'move' or 'moveStack'.

If data of Movebank are used, the definitions of the content of the columns within the @idData, @sensor, @data slots of the move or moveStack object is detailed in the Attribute Dictionary on Movebank

When the move or moveStack is created providing a path to a .csv or .zip file downloaded from Movebank the coordinates in the @coords slot are named "location.long" and "location.lat". When the move or moveStack is created by providing a data.frame, the coordinates in the @coords slot are named "coords.x1" and "coords.x2".

## Note

It is checked whether the imported data set (via file path) is in a Movebank format. If your data isn't in a Movebank format, you have to use the alternative import for non-Movebank data reading in your data using read.csv and specifying which columns contain the needed information (see 'Details').

*Locations with "NA":*
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 in the Move object.

*Duplicated timestamps:*
When you are importing data from movebank (via path to .csv or .zip file) you can also set the argument "removeDuplicatedTimestamps=TRUE", which allows you delete the duplicated timestamps in case your data set contains them. Using this argument will retain the first of multiple records with the same animal ID and timestamp, and remove any subsequent duplicates. In case you want to control which of the duplicate timestamps are kept and which are deleted, we recommend to download the data as a .csv file from Movebank or to use the function getMovebankLocationData, find the duplicates using e.g. getDuplicatedTimestamps, decide which of the duplicated timestamp to retain, and than create a move/moveStack object with the function move. Another option is to edit the records in movebank and mark the appropriate records as outliers.

*Naming:*
Due to convention all names are turned into 'good names' which means, without spaces ('Ricky T' becomes 'Ricky.T'), if names are numbers a "X" will be prepended ('12345' becomes 'X12345') and most symbols will be replaced by "." ('Ricky-T' becomes 'Ricky.T').

*Outliers:*
In Movebank outliers can be marked manually or using filters, including a duplicate filter that flags

duplicate records based on user-selected attributes, retaining the first record of each duplicate set that was imported to the study. These outliers will be marked with 'false' in the column 'visible', if data were downloaded including outliers.
When the move object is created by providing the path to the file downloaded from Movebank, the records marked as outliers are automatically placed in the UnusedRecords slots.
If these marked outliers want to be included in the move object, read in the data from the downloaded csv file from movebank with `read.csv`, set the marked outliers to 'true' in the column 'visible', or remove the column 'visible' from the data frame, save the table as a csv file and create the move object.

*Multiple sensors:*
If a move/moveStack object contains multiple sensors, this object can be separated into a move/moveStack object per sensor type:
`x[x@sensor=="z"]` where "x" is a Move or a MoveStack object, and "z" is the name of the sensor e.g. "GPS", "Radio Transmitter", etc.

*Telemetry object with error calibration:*
If the `telemetry` object (from ctmm) contains calibrated data, i.e. the GPS error has been calculated using the available tools in the ctmm package, the move object will contain an extra column in the data slot called `error.loc.mts` that will contain the error in meters for each location. This information can be used e.g. in the `location.error` argumet of the dynamic Brownian Bridge functions or the `locErr` argument of the Bivariate Gaussian Bridge functions.

*Providing a data.frame object:*
To read in a data.frame as a move/movestack object without specifying which arguments correspond to each argument, the data.frame is assumed to be downloaded from Movebank via the webpage or the getMovebankLocationData function. The function assumes a movebank format with the coordinates under the columns "location long" and "location lat" and projection lat/long; the timestamp under the column "timestamps" in the movebank format "%Y-%m-%d %H:%M:%S" and in timezone UTC; individual Id under the column "individual local identifier" and the sensor type under the column "sensor type".

## Author(s)

Marco Smolla, Bart Kranstauber & Anne Scharf

## 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", sensor="GPS")
plot(data, type="b", pch=20)

## if the data contain multiple individuals a moveStack will be created
fishersPath<-system.file("extdata","fishersSubset.csv.gz",package="move")
```

```
fishersSubset <- read.table(fishersPath, header=TRUE, sep=",", dec=".")
data2 <- move(x=fishersSubset$location.long, y=fishersSubset$location.lat,
          time=as.POSIXct(fishersSubset$timestamp,format="%Y-%m-%d %H:%M:%S", tz="UTC"),
             data=fishersSubset, proj=CRS("+proj=longlat +ellps=WGS84"),
             animal=fishersSubset$individual.local.identifier,
             sensor=fishersSubset$sensor)
plot(data2, type="b", pch=20, col=c("green","blue")[data2@idData$individual.local.identifier])
plot(data2[[2]], type="l")
```

---

`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 specific to the animal, e.g. the sex or age, which are stored in the idData slot `data.frame`. Any data associated to the coordinates are stored in the data slot `data.frame`. If the object was created with the Movebank browsing functions it also contains the study name, licence and citation information.
A `Move` object can be created with the functions move, getMovebankData or getDataRepository-Data.

## Slots

**bbox** belongs to the SpatialPointsDataFrame

**citation** Object of class `"character"`: how to cite the study, when Movebank data are used

**coords** coordinates of the track, belongs to the SpatialPointsDataFrame

**coords.nrs** belongs to the SpatialPointsDataFrame

**data** Object of class `"data.frame"`: additional data associated to the coordinates

**dataUnUsedRecords** Object of class `"data.frame"`: data associated to the unused records

**dateCreation** Object of class `"POSIXct"`: timestamp when the Move object was created

**idData** Object of class `"data.frame"`: additional (one row) data. These data contain information associated to the animal

**license** Object of class `"character"`: the license under which the data were published, when Movebank data are used

**proj4string** Object of class `"CRS"`: projection of the coordinates

**sensor** Object of class `"factor"`: sensors used to record the coordinates

**sensorUnUsedRecords** Object of class `"factor"`: sensors used to record the unused records

**study** Object of class `"character"`: name of the study, when Movebank data are used

**timestamps** Object of class `"POSIXct"`: timestamps associated to the coordinates

**timestampsUnUsedRecords** Object of class `"POSIXct"`: timestamps associated to the unused records, i.e. lines of the data that were removed because they included NA locations

## Methods

angle signature(object = "Move"): calculates angles between consecutive locations

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

brownian.bridge.dyn signature(object = "Move"): calculates the utilization distribution (UD) of the given track using the dynamic Brownian Bridge Movement Model

brownian.motion.variance.dyn signature(object = "Move"): calculates the motion variance of the dynamic Brownian Bridge Movement Model

burst signature(object = "Move"): bursts a track by a specified variable

citations signature(object = "Move"): extracts or sets the citation

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

corridor signature(object = "Move"): identifies track segments whose attributes suggest corridor use behavior

distance signature(object = "Move"): calculates distances between consecutive locations

dynBGB signature(object = "Move"): calculates the utilization distribution (UD) of the given track using the dynamic Bivariate Gaussian Bridge model

dynBGBvariance signature(object = "Move"): calculates the orthogonal and parallel motion variance of the dynamic Brownian Bridge Movement Model

equalProj signature(object = "Move"): checks whether all objects of a list are in the same projection

hrBootstrap signature(object = "Move"): calculates and plots the area of the Minimum Convex Polygon of a track

idData signature(object = "Move"): returns or replaces the idData slot

interpolateTime signature(object = "Move"): interpolates trajectories based on time

lines signature(object = "Move"): add lines of the track of the animal to a plot

move2ade signature(object = "Move"): converts to a adehabitat compatible object

moveStack signature(object = "Move"): stacks a list of Move objects

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

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

points signature(object = "Move"): add points of the track of the animal to a plot

seglength signature(object = "Move"): calculates the length of each segment of a track

sensor signature(object = "Move"): extracts the sensor(s) used to record the coordinates

show signature(object = "Move"): displays summary the Move object

speed signature(object = "Move"): calculates speed between consecutive locations

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

summary signature(object = "Move"): summarizes the information of Move object

subset signature(object = "Move"): subsets the Move object

timeLag signature(object = "Move"): calculates time lag between consecutive locations

timestamps signature(object = "Move"): gets the timestamps associated to the coordinates

turnAngleGc signature(object = "Move"): calculates angles between consecutive locations

unUsedRecords signature(object = "Move"): returns the unUsedRecords object containing the data of the unused records

### Note

The Move object contains a .MoveGeneral, .MoveTrack, .MoveTrackSingle and .unUsedRecords object which can be used to program against.

### Author(s)

Marco Smolla & Anne Scharf

---

## move2ade *Convert a Move or MoveStack object to a SpatialPointsDataFrame*

---

### 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, moveStack or moveBurst object |

### Details

There is also the possibility to convert between a ltraj object and a Move or MoveStack:
as(x,"ltraj") where "x" is a Move or MoveStack object
as(x,"Move") or as(x,"MoveStack") where "x" is a ltraj 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 & Anne Scharf

## 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.

If the function is left empty, you'll be requested to enter your username and password. This option is useful to keep Movebank login data confidential when R-code is shared.

## Value

'MovebankLogin'

## Note

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

## Author(s)

Marco Smolla & Anne Scharf

## Examples

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

## and than use it with Movebank browsing functions, e.g.
getMovebankStudies(login)

## End(Not run)
```

---

MovebankLogin-class *The MovebankLogin class*

---

## Description

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

## Methods

getMovebank `signature(object = "MovebankLogin")`: download data from Movebank

getMovebankAnimals `signature(object = "MovebankLogin")`: get animals, tags and IDs of a Movebank study

getMovebankData `signature(object = "MovebankLogin")`: download data from Movebank as a Move* object

getMovebankLocationData `signature(object = "MovebankLogin")`: download location data from Movebank as a table

getMovebankNonLocationData `signature(object = "MovebankLogin")`: download non-location data from Movebank as a table

getMovebankID `signature(object = "MovebankLogin")`: get study ID from Movebank

getMovebankSensors `signature(object = "MovebankLogin")`: get information about Movebank sensors

getMovebankSensorsAttributes `signature(object = "MovebankLogin")`: get available sensor attributes of a Movebank Study

getMovebankStudies `signature(object = "MovebankLogin")`: get all studies available on Movebank

getMovebankStudy `signature(object = "MovebankLogin")`: get information of a Movebank study

searchMovebankStudies `signature(object = "MovebankLogin")`: search for a Movebank study by key words

getMovebankReferenceTable signature(object = "MovebankLogin"): get all reference data of a Movebank study

show signature(object = "MovebankLogin"): shows user name and password contained in the MovebankLogin object

## Author(s)

Bart Kranstauber, Marco Smolla & Anne Scharf

---

MoveBurst *MoveBurst class*

---

## Description

The class MoveBurst is used to store the track of one individual with a categorical assignment to each segment. Every segment between two locations has a class of for example a behavioral category. A MoveBurst object is created with the functions burst and corridor.

## Slots

**bbox** See Move-class

**burstId** Id of the behavioral categorization assigned to each segment, one shorter then the number of locations.

**citation** See Move-class

**coords** See Move-class

**coords.nrs** See Move-class

**data** See Move-class

**dataUnUsedRecords** See Move-class

**dateCreation** See Move-class

**idData** See Move-class

**license** See Move-class

**proj4string** See Move-class

**sensor** See Move-class

**sensorUnUsedRecords** See Move-class

**study** See Move-class

**timestamps** See Move-class

**timestampsUnUsedRecords** See Move-class

## Methods

angle signature(object = "MoveBurst"): calculates angles between consecutive locations

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

brownian.bridge.dyn signature(object = "MoveBurst"): calculates the utilization distribution (UD) of the given track using the dynamic Brownian Bridge Movement Model

brownian.motion.variance.dyn signature(object = "MoveBurst"): calculates the motion variance of the dynamic Brownian Bridge Movement Model

burstId signature(object = "MoveBurst"): returns the Id of the behavioral categorization assigned to each segment

citations signature(object = "MoveBurst"): extracts or sets the citation

coordinates signature(object = "MoveBurst"): extracts the coordinates from the track

corridor signature(object = "MoveBurst"): identifies track segments whose attributes suggest corridor use behavior

distance signature(object = "MoveBurst"): calculates distances between consecutive locations

dynBGB signature(object = "MoveBurst"): calculates the utilization distribution (UD) of the given track using the dynamic Bivariate Gaussian Bridge model

dynBGBvariance signature(object = "MoveBurst"): calculates the orthogonal and parallel motion variance of the dynamic Brownian Bridge Movement Model

equalProj signature(object = "MoveBurst"): checks whether all objects of a list are in the same projection

hrBootstrap signature(object = "MoveBurst"): calculates and plots the area of the Minimum Convex Polygon of a track

idData signature(object = "MoveBurst"): returns or replaces the idData slot

interpolateTime signature(object = "MoveBurst"): interpolates trajectories based on time

lines signature(object = "MoveBurst"): add lines of the track of the animal to a plot

move2ade signature(object = "MoveBurst"): converts to a adehabitat compatible object

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

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

plotBursts signature(object = "MoveBurst"): plots the centroids of a bursted track

points signature(object = "MoveBurst"): add points of the track of the animal to a plot

seglength signature(object = "MoveBurst"): calculates the length of each segment of a track

sensor signature(object = "MoveBurst"): extracts the sensor(s) used to record the coordinates

show signature(object = "MoveBurst"): displays summary the MoveBurst object

speed signature(object = "MoveBurst"): calculates speed between consecutive locations

split signature(object = "MoveBurst"): splits a MoveBurst into a list of Move objects

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

summary signature(object = "MoveBurst"): summarizes the information of MoveBurst object

subset signature(object = "MoveBurst"): subsets the MoveBurst object

timeLag signature(object = "MoveBurst"): calculates time lag between consecutive locations

timestamps signature(object = "MoveBurst"): gets the timestamps associated to the coordinates

turnAngleGc signature(object = "MoveBurst"): calculates angles between consecutive locations

unUsedRecords signature(object = "MoveBurst"): returns the unUsedRecords object containing the data of the unused records

## Note

The MoveBurst object contains a .MoveGeneral, .MoveTrackSingleBurst and .unUsedRecords object which can be used to program against.

## Author(s)

Marco Smolla & Anne Scharf

---

moveStack *Creating a MoveStack*

---

## Description

Stacks a list of Move objects

## Usage

```
  ## S4 method for signature 'list'
moveStack(x, forceTz=NULL, ...)

  ## S4 method for signature 'Move'
moveStack(x, ..., forceTz=NULL)

  ## S4 method for signature 'MoveStack'
moveStack(x, ..., forceTz=NULL)
```

## Arguments

| | |
|---|---|
| x | a list of move or moveStack objects (or a combination of both). Timestamps of all objects have to be in the same time zone. |
| forceTz | The time zone, as a character, that the resulting moveStack object should have (see OlsonNames() for available time zones). If NULL the timestamps of the resulting moveStack will be in the time zone of the computer (see Sys.timezone()) |
| ... | Additional move or moveStack objects. |

### Details

This function stacks single Move or Movestacks objects to a MoveStack object.

### Value

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, Bart Kranstauber & Anne Scharf

### See Also

split

### Examples

```
data(leroy)
ricky<-move(system.file("extdata","ricky.csv.gz", package="move"))
leroy<-spTransform(leroy, crs(ricky))

## creating a moveStack from a list of move objects
l <- list(ricky[200:270,], leroy[200:270,])
moveStack(l)

## creating a moveStack from several move objects
moveStack(ricky[200:270,], leroy[200:270,], forceTz="UTC")

## creating a moveStack with the same time zone as input move objects
moveStack(ricky[200:270,], leroy[200:270,], forceTz=attr(timestamps(ricky),"tzone"))
```

---

## MoveStack-class *The MoveStack class*

---

### Description

The MoveStack object is a stack of move objects. A MoveStack object can be created with the functions move, moveStack, getMovebankData or getDataRepositoryData

## Slots

**bbox** belongs to the SpatialPointsDataFrame

**citation** Object of class "character": how to cite the study, when Movebank data are used

**coords** coordinates of the track, belongs to the SpatialPointsDataFrame

**coords.nrs** belongs to the SpatialPointsDataFrame

**data** Object of class "data.frame": additional data associated to the coordinates

**dataUnUsedRecords** Object of class "data.frame": data associated to the unused records

**dateCreation** Object of class "POSIXct": timestamp when the MoveStack object was created

**idData** Object of class "data.frame": additional (one row) data. These data contain information associated to the animal

**license** Object of class "character": the license under which the data were published, when Movebank data are used

**proj4string** Object of class "CRS": projection of the coordinates

**sensor** Object of class "factor": sensors used to record the coordinates

**sensorUnUsedRecords** Object of class "factor": sensors used to record the unused records

**study** Object of class "character": name of the study, when Movebank data are used

**timestamps** Object of class "POSIXct": timestamps associated to the coordinates

**timestampsUnUsedRecords** Object of class "POSIXct": timestamps associated to the unused records, i.e. lines of the data that were removed because they included NA locations

**trackId** Object of class "factor": vector that indicates, which data, coordinates and timestamps belong to each individual

**trackIdUnUsedRecords** Object of class "factor": vector that indicates, which data, coordinates and timestamps of the unused records belong to each individual

## Methods

angle signature(object = "MoveStack"): calculates angles between consecutive locations

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

brownian.bridge.dyn signature(object = "MoveStack"): calculates the utilization distribution (UD) of the given track using the dynamic Brownian Bridge Movement Model

brownian.motion.variance.dyn signature(object = "MoveStack"): calculates the motion variance of the dynamic Brownian Bridge Movement Model

citations signature(object = "MoveStack"): extracts or sets the citation

coordinates signature(object = "MoveStack"): extracts the coordinates from the track

corridor signature(object = "MoveStack"): identifies track segments whose attributes suggest corridor use behavior

distance signature(object = "MoveStack"): calculates distances between consecutive locations

equalProj signature(object = "MoveStack"): checks whether all objects of a list are in the same projection

hrBootstrap signature(object = "MoveStack"): calculates and plots the area of the Minimum Convex Polygon of a track

idData signature(object = "MoveStack"): returns or replaces the idData slot

lines signature(object = "MoveStack"): add lines of the track of the animals to a plot

move2ade signature(object = "MoveStack"): converts to a adehabitat compatible object

moveStack signature(object = "MoveStack"): stacks a list of MoveStack objects

n.indiv signature(object = "MoveStack"): returns the number of individuals

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

plot signature(object = "MoveStack"): plots the track of the animals

points signature(object = "MoveStack"): add points of the track of the animals to a plot

seglength signature(object = "MoveStack"): calculates the length of each segment of a track

sensor signature(object = "MoveStack"): extracts the sensor(s) used to record the coordinates

show signature(object = "MoveStack"): displays summary the MoveStack object

speed signature(object = "MoveStack"): calculates speed between consecutive locations

split signature(object = "MoveStack"): splits a MoveStack into a list of Move objects

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

summary signature(object = "MoveStack"): summarizes the information of MoveStack object

subset signature(object = "MoveStack"): subsets the MoveStack object

timeLag signature(object = "MoveStack"): calculates time lag between consecutive locations

timestamps signature(object = "MoveStack"): gets the timestamps associated to the coordinates

trackId signature(object = "MoveStack"): returning the Id of the individual per coordinate

turnAngleGc signature(object = "MoveStack"): calculates angles between consecutive locations

unUsedRecords signature(object = "MoveStack"): returns the unUsedRecordsStack object containing the data of the unused records

## Note

The MoveStack object contains a .MoveGeneral, .MoveTrackStack and .unUsedRecordsStack object which can be used to program against.

## Author(s)

Marco Smolla & Anne Scharf

## n.indiv *Extract the number of individuals of a MoveStack*

### Description

This function returns the number of individuals from a MoveStack object.

### Usage

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

### Arguments

| | |
|---|---|
| obj | a move or moveStack object |

### Value

Returns the number of individuals.
It will be always 1 for objects of the class Move

### 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

This function returns the number of locations of a track from a Move or MoveStack object.

### Usage

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

### Arguments

| | |
|---|---|
| obj | a move, moveStack or moveBurst object |

### Value

number of locations.
If a MoveStack is provided, the number of locations per individual is returned.

### Author(s)

Marco Smolla

### Examples

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

---

## namesIndiv *Extract the names of the individuals of a move or moveStack object*

---

### Description

This function returns the names of the individuals from a move or moveStack object.

### Usage

```
namesIndiv(obj)
```

### Arguments

| | |
|---|---|
| obj | a move or moveStack object |

### Value

Returns the name as a character for a move object, and the names as a character vector from a moveStack object.
If no name has been provided when creating the move object, "unnamed" will be returned.

### Author(s)

Anne Scharf

## Examples

```
data(leroy)
namesIndiv(leroy)

data(fishers)
namesIndiv(fishers)
```

---

`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`, `DBBMM`, `DBBMMStack`, `dynBGB` or `.UD` object |
| `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) |
| `...` | Currently not implemented |

## 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 or dynBGB object, or a list of numeric values for a DBBMM-Stack

## Author(s)

Marco Smolla & Anne Scharf

## 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 | a move, moveStack, moveBurst, DBBMM, DBBMMStack, DBBMMBurstStack, dynBGB, dBMvariance, dBMvarianceStack, dBMvarianceBurst, dBGBvariance, .UD, .UDStack or .UDBurstStack 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 sense since the x and y dimensions are the same |
| ... | arguments to be passed to methods, such as graphical parameters, and the logical add argument (see par). See 'Details' for col (color) options. |

## Details

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 coordinates of the Move object as points.
If x is a DBBMM, DBBMMStack, DBBMMBurstStack or dynBGB object its raster object is plotted with the corresponding cell values. Unlike the image function, it keeps the same cell size ratio when the plot window is re-sized.

In the argument col a vector of colors of the same length as the number of individual for a moveStack, or number of burst levels for a moveBurst object can be specified. If left empty the default 8 colors from R are used, which will be recycled if the object contains more individuals or burst levels (run palette() to obtain vector of default colors).

### 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 & Anne Scharf

### 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 bursted track*

---

### Description

The plotBursts function plots bursted Move objects (see burst for how to create a bursted Move object). The function plots a circle at the midpoint of each burst segment (consecutive coordinates that belong to a single burst).

### 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 moveBurst object or a list of moveBurst objects |
| 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 (see palette() to obtain vector of default colors). If there are more than 8 burstIDs the colors are recycled |
| ... | additional plot attributes |

### Details

sizeFUN
The color of the circles correspond to the burstIDs. The size of the cycles can have different meanings, depending on what function is defined. By default the size refers to the relative time of the burst segment compared to the whole track. This argument accepts any personalized function.

### Note

If a list of moveBurst objects is provided, the plots are plotted one after another, and not side by side.

### Author(s)

Marco Smolla & Anne Scharf

### 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, col=rainbow(5))

## plotting circle size of a moveBurst track by realtive segment length
plotBursts(object=testb, breaks=3, sizeFUN=function(x){sum(distance(x))}, pch=19, add=FALSE)
```

---

## points *Plotting the points of a track*

---

### Description

Function for plotting a 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 | a move, moveStack, moveBurst, dBMvariance, dBMvarianceStack, dBMvarianceBurst or dBGBvariance object. |
| col | a vector of colors of the same length as the number of individual for a moveStack, or number of burst levels for a moveBurst object. If left empty the default 8 colors from R are used, which will be recycled if the object contains more individuals or burst levels (run palette() to obtain vector of default colors) |
| ... | arguments to be passed on, e.g. col for color, or add to add the points to a plot. See ?par for options. |

## Author(s)

Marco Smolla & Anne Scharf

## See Also

plot, lines

## Examples

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

## plot a moveStack object
data(fishers)
plot(fishers, type="l")
points(fishers, col=3:4, pch=4)
```

---

# raster *Extract raster topology from DBBMM or dynBGB*

---

## Description

Extracts the RasterLayer topology from a DBBMM, DBBMMStack and dynBGB object.

### Usage

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

### Arguments

| | |
|---|---|
| x | a DBBMM, DBBMMStack or dynBGB object |

### Details

This function extracts the raster topology (i.e. without values) of the input object. DBBMM, DBBMMStack and dynBGB objects can be directly used in most raster functions but in case a raster with values needs to be extracted as(x,'RasterLayer') can be used.

### Value

An object from class RasterLayer is returned.

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
data(leroydbbmm)
data(dbbmmstack)
raster(leroydbbmm) #returns the raster topology of a DBBMM object
raster(dbbmmstack) # returns the raster topology of a DBBMMStack object
```

---

## raster2contour *Convert raster to contour lines*

---

### Description

The function converts a raster UD(stack) object to a SpatialLinesDataFrame. This allows to reproject the contours to different projections.

### Usage

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

## Arguments

| | |
|---|---|
| x | a DBBMM, DBBMMStack, dynBGB, .UD or .UDStack object |
| ... | additional arguments, like levels and nlevels, that can be passed to 'rasterToContour' function |

## 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.

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

## Value

'SpatialLinesDataFrame'

## Author(s)

Marco Smolla & Anne Scharf

## See Also

getVolumeUD, contour, outerProbability

## Examples

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

## from a DBBMM object
(cont1 <- raster2contour(leroydbbmm))
plot(cont1)

## from a dynBGB object
(cont2 <- raster2contour(leroydbgb, level=.95))
plot(cont2)

## from a DBBMMStack object
(cont3 <- raster2contour(dbbmmstack))
plot(cont3)
(cont4 <- raster2contour(dbbmmstack, level=c(.5,.95)))
plot(cont4)
```

searchMovebankStudies *Search for a study on Movebank*

## Description

This function searches for studies within Movebank by a specified keyword or phrase.

## Usage

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

## Arguments

| | |
|---|---|
| x | a character string to search within the Movebank study names |
| login | a MovebankLogin object, if empty you'll be asked to enter your username and 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 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()
# returns all studies that include this exact term: "MPIO"
searchMovebankStudies(x="MPIO", login=login)

## End(Not run)
```

`seglength` *Segment lengths of a track*

## Description

Calculates the length of each segment of a track

## Usage

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

## Arguments

| | |
|---|---|
| x | a Move, MoveStack or MoveBurst object |

## Details

The `seglength` function calculates the distances between point 1 and point 2, point 2 and point 3, and so on.
Distances are calculates with the pointDistance function from the package raster.

## Value

A numeric vector one element shorter than the number of locations is obtained. Note that in moveStacks distances are not split between animals (see 'Examples' on how to add the values to a moveStack).

Length in map units.
If the projection of the coordinates is longitude/latitude all values are returned in meters, otherwise it is the Euclidean distance in the map units of the projection of the move object. Check and set the projection of your Move, MoveStack or MoveBurst object using the `proj4string()` function.

## Author(s)

Marco Smolla

## Examples

```
## Not run:
## Move object in longlat projection
data(leroy)
head(seglength(leroy))
# to add this information to the move object, a "NA" has to be assigened
# e.g. to the last location (it also could be assigend to the first location).
leroy$segLength <- c(seglength(leroy), NA)

## MoveStack object in longlat projection
```

```
data(fishers)
head(seglength(fishers))
# to add this information to the moveStack object, a "NA" has to be assigened
# e.g. to the last location of each individual (it also could be assigend to the first location).
fishers$segLength <-  unlist(lapply(lapply(split(fishers),seglength),c, NA))

## End(Not run)
```

---

sensor *Extract the sensor of a Move unUsedRecords object*

---

## Description

Extracts the sensor(s) used to record the locations of a track from a Move or unUsedRecords object.

## Usage

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

## Arguments

| | |
|---|---|
| this | a move, moveStack, moveBurst, .unUsedRecords or .unUsedRecordsStack object |
| ... | Currently not used |

## Value

'factor' with the sensor(s) name(s).
Note that the returned vector for a MoveStack or .unUsedRecordsStack is not split between animals.

## Author(s)

Bart Kranstauber

## Examples

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

head(sensor(unUsedRecords(leroy))) ## get the sensor from the unused records of a Move object

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

show *Show a Move, DBBMM, dynBGB object*

## Description

Displays a summary of the input object.

## Usage

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

## Arguments

| | |
|---|---|
| object | a move, moveStack, moveBurst, DBBMM, DBBMMStack, DBBMMBurstStack, dBMvariance, dBMvarianceBurst, dBMvarianceStack, dynBGB, dBGBvariance, .UD, .UDStack, .UDBurstStack or movebankLogin object |

## Details

For Move, dBMvariance and dBGBvariance objects the function displays a summary including: 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, etc. If the imported data are not from the Movebank database Animal, Species, nPoints, Receiver, and Study are not shown.

For DBBMM, dynBGB or .UD objects a summary of the raster properties is shown.

For the movebankLogin object the username and password is shown.

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
data(leroy)
show(leroy) # show a move object
data(leroydbbmm)
show(leroydbbmm) # show DBBMM object
```

speed *Speed between the locations of a movement track*

## Description

This function returns the speed between consecutive locations of Move or MoveStack object.

## Usage

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

## Arguments

| | |
|---|---|
| x | a move, moveStack or moveBurst object |

## Value

Speed in map units/second.
If the projection of the coordinates is long/lat all values are returned in m/s, otherwise in the map units/second of the projection of the move object. Check and set the projection of your Move, MoveStack or MoveBurst object using the `proj4string()` function.

If a `move` or `moveBurst` object is provided, a numeric vector one element shorter than the number of locations is obtained.
If a `moveStack` object is provided, a list with one element per individual containing a numeric vector one element shorter than the number of locations is obtained.

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
# speeds from a Move object
data(leroy)
head(speed(leroy))
# to add this information to the move object, a "NA" has to be assigened
#  e.g. to the last location (it also could be assigend to the first location).
leroy$speed <- c(speed(leroy), NA)

## speeds from a MoveStack object
data(fishers)
str(speed(fishers))
# to add this information to the moveStack object, a "NA" has to be assigened
# e.g. to the last location of each individual (the speed belongs to the following segment).
fishers$speed <- unlist(lapply(speed(fishers),c, NA ))
```

split *Splitting a MoveStack, MoveBurst or DBBMMStack*

## Description

Splitting a MoveStack or MoveBurst into a list of Move objects. Splitting a DBBMMStack into a list of DBBMM objects.

## Usage

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

## Arguments

| | |
|---|---|
| x | a moveStack, moveBurst or DBBMMStack object |
| f | not needed |
| drop | not needed |
| ... | Currently not implemented |

## Details

A MoveStack is split into a list of Move objects by the trackId slot of the given MoveStack, obtaining one move object per unique trackId (usually corresponding to animal names). For staking this list of move objects use moveStack.

A MoveBurst object is split into a list of Move objects by the burstId slot of the given MoveBurst. One move object per burst (e.g. segment with given behavior) is obtained. Every location where the burst is switched will be recycled.

A DBBMMStack is split into a list of DBBMM objects by the trackId slot of the given DBBMMStack.

## Value

'list'

## Note

After splitting any object, the coordinates in the @coords slot in the resulting objects are named "coords.x1" and "coords.x2" (due to the usage of functions of other packages within this function).

## Author(s)

Marco Smolla & Anne Scharf

## Examples

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

## splitting a DBBMMStack
data(dbbmmstack)
split(dbbmmstack)

## splitting a MoveBurst
data(leroy)
behav <- c(rep(c("a","b","c","a"),each=200), rep("b", 118))
leroyBurst <- burst(x=leroy, f=behav)
split(leroyBurst)
```

---

`spTransform` *Transform projection of movement track*

---

## Description

The spTransform function transforms the coordinates stored in the Move object from the default long/lat coordinates to the default aeqd (Azimuthal Equi-distance) projection or a user defined 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`, `moveStack` or `moveBurst` object to be transformed |
| `CRSobj` | object of class CRS, or of class character in which case it is converted to CRS. Can be left empty if `center=TRUE` |
| `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.

### Value

same as input object with coordinates transformed to the new coordinate reference system.

### Author(s)

Marco Smolla & Anne Scharf

### 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 *Subset movement tracks*

---

### Description

Extraction of a subset of locations or individuals from a movement track.

### Usage

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

### Arguments

| | |
|---|---|
| x | a move, moveStack, moveBurst, DBBMM, DBBMMStack, DBBMMBurstStack, dynBGB, dBMvariance, dBMvarianceBurst, dBMvarianceStack or dBGBvariance object |
| i | numeric, character or logical vector for individuals in a stack or a set of locations |

### Details

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

## Value

same object class as the input containing the selected locations or individuals

## Author(s)

Bart Kranstauber & Anne Scharf

## Examples

```
## subseting a Move, MoveBurst, DBBMM, dBMvariance, dBMvarianceBurst,
## dBMvarianceStack or dBGBvariance object by locations
data(leroy)
leroy[1:20,] # subset to selected range of coordinates of a move objects
leroy[c(1,10,20),] # subset to selected coordinates of a move objects
leroy[c(TRUE,FALSE),] # subset to every second location
leroy[c(TRUE,FALSE,FALSE),] # subset to every third location

## subseting a moveStack, DBBMMStack or DBBMMBurstStack object,
## by locations
data(fishers)
# subset to selected range of coordinates of a moveStack objects. If the first individual contains
# more than, in this case 300, locations, only locations of the fist individual will be returned
fishers[1:300,]
fishers[1] # returns first location of first individual

## or individuals
fishers[['Ricky.T']] # returns move object of named individual
fishers[[c('Leroy','Ricky.T')]] # returns subseted moveStack only with the named individual
fishers[[2]] # returns move object of 2nd individual
fishers[[c(1,2)]]# returns subseted moveStack only with the selected individual

fishers[[c(TRUE,FALSE)]] # returnes move or moveStack object with those individuals that are 'TRUE'
```

---

summary *Summary of Move, DBBMM, dynBGB objects*

---

## Description

Summarizes the information contained in the input object

## Usage

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

### Arguments

| | |
|---|---|
| object | move, moveStack, moveBurst, DBBMM, DBBMMStack, DBBMMBurstStack, dynBGB, dBMvariance, dBMvarianceBurst, dBMvarianceStack,dBGBvariance, .UD, .UDStack or .UDBurstStack object |

### Details

Returns the projection, extent, and maximum and minimum values of the raster stored within the DBBMM, DBBMMStack, dynBGB, .UD, .UDStack or .UDBurstStack object. For the remaining objects it returns a summary of the data contained in the '@data' slot.

### Author(s)

Marco Smolla & Anne Scharf

### Examples

```
data(leroy)
summary(leroy) # summary of a move object
data(leroydbbmm)
summary(leroydbbmm) # summary of a DBBMM object
```

---

## thinTrackTime *Thinning trajectories to a specific time interval or distance.*

---

### Description

These functions thin trajectories, by selecting segments from the original track with a fixed time interval or distance. Finding all segments of a specific time interval might for example be useful for fitting step selection functions.

### Usage

```
thinTrackTime(x, interval = NA, tolerance = NA,
  criteria = c("closest", "first", "all"), ...)
thinDistanceAlongTrack(x, interval = NA, tolerance = NA,
  criteria = c("closest", "first", "all"), ...)
```

### Arguments

| | |
|---|---|
| x | a move object |
| interval | in thinTrackTime a object of class difftime specifying a time interval. See 'Examples'.<br>in thinDistanceAlongTrack a numeric value specifying a distance. The units will correspond to the map units. If the coordinates are in long/lat, than the value should be provided in meters. |

| | |
|---|---|
| tolerance | in thinTrackTime a object of class difftime specifying the tolerance of the specified interval. See 'Examples'.<br>in thinDistanceAlongTrack a numeric value specifying the tolerance of the specified interval |
| criteria | the criteria ("closest", "first" or "all") to be used when multiple solutions are available. Default is "closest". |
| ... | Currently not implemented. |

## Details

The functions search for consecutive segments with a cumulative sum of the time lag (or distance) corresponding to interval and tolerance values. From each selected chunk of the track, only the first and last location are kept in the new object, this new segment is labeled with "selected". The segments labeled as "notSelected" are those parts of the track that did not fulfill the indicated interval. A "notSelected" burst can correspond to multiple consecutive segments that have a larger timelag than the one specified, or a single large time gap that is present in the original data.
Note that in the case of thinDistanceAlongTrack, the distances between the locations in the new object do not represent the distance that the animal actually traveled, as the intermediate location are removed.

## Value

A MoveBurst object, with segments labeled either 'selected' or 'notSelected', only the selected segments match the criteria set in the function call.

A list of MoveBurst objects will all possible solutions if the criteria is set to "all".

## Note

This function finds the maximal number of segments that meet the criteria but does not ensure that the average matches the set interval.

## Author(s)

Bart Kranstauber & Anne Scharf

## See Also

interpolateTime

## Examples

```
data("leroy")
leroysub <- leroy[1:200]
### selecting those segments that have a time interval of 15mins pulsminus 5mins
thintime <- thinTrackTime(leroysub, interval = as.difftime(15, units='mins'),
                          tolerance = as.difftime(5, units='mins'))
summary(timeLag(thintime,"mins")[thintime@burstId=="selected"])

### selecting those segments that have a distance of 100m pulsminus 10m
thindist <- thinDistanceAlongTrack(leroysub, interval = 100, tolerance = 10)
```

```
summary(distance(thindist)[thindist@burstId=="selected"])
```

---

`timeLag` *Time lags between the locations of a movement track*

---

## Description

Calculates the time lags between consecutive locations of a track.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a `move`, `moveStack` or `moveBurst` object |
| `units` | The units used for the conversion (e.g. "secs", "mins", "hours", "days" or "weeks"). They should be specified for a moveStack to ensure the same units between individuals. Optional (but recommended). |
| `...` | Currently not implemented. |

## Details

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

## Value

Time lags in the specified units.

If a `move` or `moveBurst` object is provided, a numeric vector one element shorter than the number of locations is obtained.
If a `moveStack` object is provided, a list with one element per individual containing a numeric vector one element shorter than the number of locations is obtained.

## Author(s)

Bart Kranstauber & Anne Scharf

## Examples

```
## time lags from a Move object
data(leroy)
head(timeLag(leroy, units="hours"))
# to add this information to the move object, a "NA" has to be assigened
# e.g. to the first location (it also could be assigend to the first location).
leroy$timeLag <- c(timeLag(leroy, units="hours"), NA)

## time lags from a MoveStack object
data(fishers)
str(timeLag(fishers, units="mins"))
# to add this information to the moveStack object, a "NA" has to be assigened
# e.g. to the duration is assigned to the first location of each segment
fishers$timeLag <- unlist(lapply(timeLag(fishers, units="mins"),  c, NA))
```

---

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

---

## Description

The timestamps method returns or sets the timestamps of a track from a Move or MoveStack 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`, `moveStack`, `moveBurst`, `.unUsedRecords` or `.unUsedRecordsStack` object |
| `value` | timestamps from class POSIXct |

## Value

vector of class `POSIXct`.
Note that for moveStacks a single string is returned without splitting by individual (see 'Examples').

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
data(leroy)
data(fishers)

## get the timestamps from a Move object
head(timestamps(leroy))
## get the timestamps from a MoveStack object
head(timestamps(fishers))
## get the timestamps from a unUsedRecords object
head(timestamps(unUsedRecords(leroy)))

## get timestamps separatly for each individual from a MoveStack
str(lapply(split(fishers), timestamps))

## 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)+60.1
```

---

# trackId *Returns trackId*

---

## Description

Obtain the Id of the individual per location of a MoveStack or unUsedRecordsStack

## Usage

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

## Arguments

| | |
|---|---|
| x | a moveStack or .unUsedRecordsStack object |

## Value

Returns a factor indicating for each location to which individual it belongs.

## Author(s)

Bart Kranstauber

## Examples

```
data(fishers)
head(trackId(fishers))
head(trackId(unUsedRecords(fishers)))
```

turnAngleGc *Turning angles on great circle tracks*

## Description

This function returns the turning angles of a great circle track. This angle represents the relative angle between the consecutive segments.

## Usage

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

## Arguments

| | |
|---|---|
| x | a move, moveStack or moveBurst 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

Angles in degrees (between -180 and 180)

If a move or moveBurst object is provided, a numeric vector two elements shorter than the number of locations is obtained.
If a moveStack object is provided, a list with one element per individual containing a numeric vector two elements shorter than the number of locations is obtained.

## Author(s)

Bart Kranstauber & Anne Scharf

## See Also

angle

## Examples

```
## turnAngleGc from a Move object
data(leroy)
head(turnAngleGc(leroy))
# to add this information to the move object, a "NA" has to be assigened
# to the first and last location.
leroy$turnAngleGc <- c(NA, turnAngleGc(leroy), NA)
```

```
## turnAngleGc from a MoveStack object
data(fishers)
str(turnAngleGc(fishers))
# to add this information to the moveStack object, a "NA" has to be assigened
# to the first and last location of each individual
fishers$turnAngleGc <-unlist(lapply(turnAngleGc(fishers), function(x) c(NA, x, NA)))
```

---

UDStack *Creating UDStack objects*

---

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| x | A list of rasters, a rasterBrick, a rasterStack or a DBBMMBurstStack object that needs to be converted to a .UDStack object. |
| ... | Currently not used |

## Details

The values of a DBBMMBurstStack are standardized per raster layer.

## Value

An UDStack object

## Author(s)

Bart Kranstauber & Anne Scharf

## Examples

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

## transforming a DBBMMBurstStack into UDStack, e.g. to than
## use the "getVolumeUD" or "emd" function
data(leroy)
leroyB <- burst(x=leroy,f=c(rep(c("Behav.1","Behav.2"),each=400),rep("Behav.1", 118)))
```

```
leroyBdbb <- brownian.bridge.dyn(object=spTransform(leroyB[785:820], center=TRUE),
                                 location.error=12, dimSize=115, ext=.45,
                                 time.step=25/15, margin=15)
cellStats(leroyBdbb, sum)
leroyBud <- UDStack(leroyBdbb)
cellStats(leroyBud, sum)
```

---

unUsedRecords<- *Extracts or creates the unUsedRecords*

---

## Description

This function returns the unUsedRecords part of the move object or assigns locations as unused, this could for example be used to remove test locations from a track. unUsedRecords can include events with no locations, locations flagged as outliers, non-location sensor data when `includeExtraSensors` is set to TRUE in the `getMovebankData` function.

## Usage

```
 ## S4 method for signature '.unUsedRecords'
unUsedRecords(obj,...)
 ## S4 method for signature '.unUsedRecordsStack'
unUsedRecords(obj,...)

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

## Arguments

| | |
|---|---|
| `obj` | a `move`, `moveStack` or `moveBurst` object |
| `value` | A logical vector of the same length as the number of locations |
| `...` | Currently not implemented |

## Value

an `.unUsedRecords` or `.unUsedRecordsStack` object

## Author(s)

Marco Smolla & Anne Scharf

## Examples

```
data(leroy)
data(fishers)

## get unused records from a move or moveStack object
str(unUsedRecords(leroy)) # from a move object
str(unUsedRecords(fishers)) # from a moveStack object

## assign locations of a move object as unused record
par(mfrow=2:1)
plot(leroy, type='b')
# e.g. assign every second location as unused
unUsedRecords(leroy)<-as.logical((1:n.locs(leroy))%%2)
plot(leroy, type='b')

# e.g. assign first 20 locations as unused
data(leroy)
unUsedRecords(leroy)<- as.logical(c(rep("TRUE",20), rep("FALSE",n.locs(leroy)-20)))
```

---

# utilization density data

*Dynamic brownian bridges*

---

## Description

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

## Usage

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

## Details

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

## Examples

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

# Index