Package ‘track2KBA’

September 27, 2023

**Title**    Identifying Important Areas from Animal Tracking Data

**Version**  1.1.1

**URL** [https://github.com/BirdLifeInternational/track2kba](https://github.com/BirdLifeInternational/track2kba)

**Description** Functions for preparing and analyzing animal tracking data, with the intention of identifying areas which are potentially important at the population level and therefore of conservation interest. Areas identified using this package may be checked against global or regionally-defined criteria, such as those set by the Key Biodiversity Area program. The method published herein is described in full in Beal et al. 2021 [doi:10.1111/2041-210X.13713].

**Depends**  R (>= 2.10)

**License**  LGPL-3

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.2.3

**BugReports**  [https://github.com/BirdLifeInternational/track2kba/issues](https://github.com/BirdLifeInternational/track2kba/issues)

**Suggests**  adehabitatLT, doParallel, knitr, parallel, rmarkdown, tinytest

**VignetteBuilder**  knitr

**Imports**  adehabitatHR, dplyr, foreach, geosphere, ggplot2, lubridate, magrittr, maps, Matching, methods, move, purrr, raster (>= 3.6-3), rlang, sf (>= 0.7-4), sp (>= 1.5-0), tidyr

**NeedsCompilation**  no

**Author**  Martin Beal [aut, cre] (<https://orcid.org/0000-0003-1654-1410>), Steffen Oppel [aut] (<https://orcid.org/0000-0003-1654-1410>), Maria Dias [aut] (<https://orcid.org/0000-0002-7281-4391>), Mark Miller [ctb], Phillip Taylor [ctb], Virginia Morera-Pujol [ctb] (<https://orcid.org/0000-0001-6500-5548>), Elizabeth Pearmain [ctb] (<https://orcid.org/0000-0002-6600-1482>), Jonathan Handley [ctb] (<https://orcid.org/0000-0001-6468-338X>), Ben Lascelles [ctb]
boobies

St. Helena Masked Boobies

Description

A GPS tracking data set of Masked Boobies during incubation and chick-rearing at St. Helena Island. Formatted following BirdLife International’s Seabird Tracking Database standard http://www.seabirdtracking.org/. Data from Oppel et al. 2015.

Usage

boobies

Format

A data frame with 116355 obs. of 6 variables:

- **track_id**: Unique identifier code for each bird
- **date_gmt**: Character vector representing date (Greenwich Mean Time)
- **time_gmt**: Character vector representing time (Greenwich Mean Time)
- **longitude**: Longitudinal position of bird
- **latitude**: Latitudinal position of bird
- **lon_colony**: Longitudinal position of breeding colony
- **lat_colony**: Latitudinal position of breeding colony
estSpaceUse

Source


estSpaceUse

Estimate the space use of tracked animals using kernel utilization distribution

Description

estSpaceUse is a wrapper for kernelUD which estimates the utilization distribution (UD) of multiple individuals or tracks in a tracking dataset.

Usage

estSpaceUse(tracks, scale, levelUD, res = NULL, polyOut = FALSE)

Arguments

tracks SpatialPointsDataFrame. Must be projected into an equal-area coordinate system. If not, first run projectTracks.

scale numeric (in kilometers). The smoothing parameter ('H') used in the kernel density estimation, which defines the width of the normal distribution around each location. The findScale function can assist in finding sensible scales.

levelUD numeric (percent). Specify which utilization distribution contour at which to subset the polygon output. NOTE: This will only affect the output if polyOut=TRUE.

res numeric (in square kilometers). Grid cell resolution for kernel density estimation. Default is a grid of 500 cells, with spatial extent determined by the latitudinal and longitudinal extent of the data.

polyOut logical scalar (TRUE/FALSE). If TRUE then output will include a plot of individual UD polygons and a simple feature with kernel UD polygons for the level of levelUD. NOTE: creating polygons of UD is both computationally slow and prone to errors if the usage included in levelUD extends beyond the specified grid. In this case estSpaceUse will return only the estUDmobject and issue a warning.

Details

A utilization distribution will be calculated for each unique 'ID'. The data should be regularly sampled or interpolated (see adehabitatLT package for functions to this end).

If desired res results in memory-heavy grid (e.g. >10,000 cells) use polyOut = FALSE to speed things up.
Value

Returns an object of class estUDm which is essentially a list, with each item representing the utilization distribution of a level of 'ID'. Values in the output signify the usage probability per unit area for that individual in each grid cell. This can be converted into a SpatialPixelsDataFrame via the adehabitatHR::estUDm2spixdf function.

If polyOut=TRUE the output will be a list with two components: "KDE_Surface" is the estUDm object and UDPolygons is polygon object of class sf (Simple Features) with the UD contour for each individual at the specified levelUD.

If polyOut=TRUE but the polygon delineation in adehabitatHR::getverticeshr fails, output is an object of class estUDm and a warning will be issued.

See Also

formatFields, tripSplit, findScale

Examples

library(sf)
library(sp)
library(magrittr)

## make some play data
dataGroup <- data.frame(
    Longitude = c(1, 1.01, 1.02, 1.04, 1.05, 1.03, 1),
    Latitude = c(1, 1.01, 1.02, 1.03, 1.021, 1.01, 1),
    ID = rep("A", 7),
    DateTime = format(
        lubridate::ymd_hms("2021-01-01 00:00:00") +
        lubridate::hours(0:6)
        )
)

tracks <- sf::st_as_sf(
dataGroup, coords = c("Longitude", "Latitude"),
crs = 4326, agr = "constant") %>%
sf::st_transform(crs = 32631) %>%
sf::as_Spatial()

## estimate utilization distributions for each track
KDE <- estSpaceUse(tracks, scale=10, levelUD = 50)
findScale

Description

findScale takes a tracking data set and outputs a series of candidate smoothing parameter values. Additionally, it compares the scale of movement resolved by the sampling resolution of the data set, to a grid of desired resolution.

Usage

```r
findScale(
  tracks,
  scaleARS = TRUE,
  res = NULL,
  sumTrips = NULL,
  scalesFPT = NULL,
  peakWidth = 1,
  peakMethod = "first"
)
```

Arguments

- **tracks**: SpatialPointsDataFrame. Must be projected into an equal-area coordinate system; if not, first run `projectTracks`.
- **scaleARS**: logical scalar (TRUE/FALSE). Do you want to calculate the scale of area-restricted search using First Passage Time analysis? NOTE: does not allow for duplicate date-time stamps.
- **res**: numeric. The desired grid cell resolution (square kilometers) for subsequent kernel analysis (NOT performed in this function). If this is not specified, the scale of movement is compared to a 500-cell grid, with spatial extent determined by the latitudinal and longitudinal extent of the data.
- **sumTrips**: data.frame. Output of `tripSummary` function. If not specified, `tripSummary` will be called within the function.
- **scalesFPT**: numeric vector. Set of spatial scales at which to calculate First Passage Time. If not specified, the distribution of between-point distances will be used to derive a set.
- **peakWidth**: numeric. How many scale-steps either side of focal scale used to identify a peak. Default is 1, whereby a peak is defined as any scale at which the variance in log FPT increases from the previous scale, and decreases for the following one.
- **peakMethod**: character. Which method should be used to select the focal peak for each ID. Options are "first", "max", and "steep". "steep" is a value of scalesFPT at which the variance in log FPT changes most rapidly compared to the surrounding scale(s).

Details

The purpose of this function is to provide guidance regarding the two most sensitive steps in the track2KBA analysis: specification of the (1) smoothing parameter and the (2) grid cell size for kernel density estimation (KDE). Specifically, the goal is to allow for exploration of the effect of these parameters and their inter-relatedness, so that an informed decision may be made regarding their specification in subsequent track2KBA steps.
Kernel density estimation has been identified as particularly sensitive to the specification of the smoothing parameter (AKA bandwidth, or ‘H’ value), that is, the parameter that defines the width of the normal distribution around each location. Many techniques for identifying ‘optimal’ smoothing parameters have been proposed (see Gitzen, Millsapugh, and Kernohan for a classic review; see Fleming and Calabreses 2017 for a later implementation) and many of these techniques have their merits; however, in the track2KBA implementation of KDE we have opted for simplicity.

In the context of the track2KBA analysis, the smoothing parameter ought to represent the relevant scale at which the animal interacts with the environment. Therefore, when selecting a Scale value for subsequent analysis, the user must take into account the movement ecology of the study species. For species which use Area-Restricted Search (ARS) behavior when foraging, First Passage Time analysis may be used to identify the scale of interaction with the environment (Fauchald and Tveraa 2003), however not all species use ARS when foraging and therefore different techniques must be used.

What minimum spatial scales are detectable by the data also depends on the sampling resolution. Therefore, when applying First Passage Time analysis, findScale sets the range of scales at which movements are analyzed based on the distribution of forward, between-point displacements in the data.

The grid cell size also affects the output of kernel density-based space use analyses. Therefore, by specifying the res parameter you can check whether your desired grid cell size is reasonable, given the scale of movement resolved by your data.

Value

This function returns a one-row dataframe with the foraging range in the first column (i.e. ‘med_max_distance’) calculated by tripSummary, and the median step length (i.e. between point distance) for the data set. The subsequent columns contain various candidate smoothing parameter (‘h’) values calculated in the following ways:

1. ‘mag’ - log of the foraging range (i.e. median maximum trip distance)
2. ‘href’ - reference bandwidth a simple, data-driven method which takes into account the number of points, and the variance in X and Y directions.
\[\sqrt{(X+Y) \ast (n(-1/6))}; \text{where } X=\text{Longitude/Easting}, Y=\text{Latitude/Northing}, \text{and } n=\text{number of relocations}\]
3. ‘scaleARS’ - spatial scale of area-restricted Search behavior as estimated using First Passage Time analysis (see fpt)

If the scaleARS option is used, a diagnostic plot is shown which illustrates the change in variance of log-FPT values calculated at each FPT scale. Grey vertical lines indicate the peaks identified for each individual using peakMethod method chosen, and the red line is the median of these, and the resulting scaleARS in the output table.

All values are in kilometers.

Examples

```r
## make some play data
dataGroup <- data.frame(Longitude = c(1, 1.01, 1.02, 1.04, 1.05, 1),
                      Latitude = c(1, 1.01, 1.02, 1.03, 1.021, 1),
                      ID = rep("A", 6),
```
findSite

Delineating sites of potential importance to conservation

Description

findSite uses the core areas (based on utilization distributions) of individual animals to identify areas used regularly used by a significant portion of the local source population (i.e. the tracked population).

Usage

findSite(KDE, represent, popSize = NULL, levelUD, thresh, polyOut = FALSE)

Arguments

KDE estUDm or SpatialPixels/GridDataFrame. If estUDm, as created by `estSpaceUse` or `adehabitatHR::kernelUD`, if Spatial*, each column should correspond to the Utilization Distribution of a single individual or track.

represent Numeric (between 0-1). Output value provided by `repAssess` which assesses how representative the tracking data are for characterising the space use of the wider population.
**findSite**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>popSize</td>
<td>Numeric, the number of individuals breeding or residing at the origin location from where animals were tracked, quantifying the population that the tracking data represent. This number will be used to calculate how many animals use the delineated areas of aggregation. If no value for popSize is provided then output will be as the proportion of the population.</td>
</tr>
<tr>
<td>levelUD</td>
<td>Numeric (percentage). Specifies the quantile used for delineating the core use (or home range) areas of individuals based on the kernel density estimation (e.g. core area=50, home range=95).</td>
</tr>
<tr>
<td>thresh</td>
<td>Numeric (percentage). Threshold percentage of local source population needed to be found using a location for it to be considered part of a 'potentialSite'. Default is set based on degree of representativeness.</td>
</tr>
<tr>
<td>polyOut</td>
<td>Logical. (Default TRUE) Should the output be a polygon dataset (TRUE) or grid of animal densities (FALSE). See 'Value' below for more details.</td>
</tr>
</tbody>
</table>

**Details**

findSite estimates the proportion of the local source population using an area based on the proportion of overlap among individual core areas and the degree of representativeness as quantified by `repAssess`). This value is then compared to a threshold of importance (i.e. a certain population) to delineate areas as 'potentialSites'. Thresholds area either set automatically set on the representativeness of the sample (lower rep==higher threshold), or set manually by the user.

The areas identified are sites of ecological relevance to the populations, which may be significant for the wider region or entire species, which can be assessed using global (or regional) criteria, such as those of the Key Biodiversity Area program.

The KBA criteria for site assessment are published in the KBA standard, which may be found here: [http://www.keybiodiversityareas.org/](http://www.keybiodiversityareas.org/).

If grid used for estimating core areas (i.e. KDE) is very memory-heavy (e.g. >10,000 cells) use `polyOut = FALSE` to speed things up.

**Value**

if `polyOut = TRUE` function returns an object of class `sf` containing polygon data with three data columns: Column `N_IND` indicates the number of tracked individuals whose core use area (at levelUD) overlapped with this polygon.

Column `N_animals` estimates the number of animals from the represented population that predictably use the polygon area during the tracked season. If no value for (at popSize) is provided, this number is the proportion of the represented population using the area.

Column `potentialSite` indicates whether the polygon can be considered a potential Site (TRUE) or not (FALSE).

if `polyOut = FALSE` function returns a gridded surface of class `SpatialPixelsDataFrame`, with the same three aforementioned columns as cell values.

If `polyOut = TRUE` the user may choose to automatically produce a plot of the result using `plot=TRUE`. The map produced displays the areas which hold aggregations above a certain threshold proportion of the population. If there are no areas displayed on the map, then either the species doesn’t aggregate, the Scale is too small to identify aggregations in this species, or the tracked sample aren’t representative enough to meet the thresholds.
formatFields

Examples

KDE <- track2KBA::KDE_example

## identify potential sites
pot_site <- findSite(KDE, represent = 90, levelUD = 50)

---

**Format tracking data**

**Description**

formatFields formats the column names of a data frame so that they are accepted by track2KBA functions.

**Usage**

```r
formatFields(
  dataGroup,
  formatBL = FALSE,
  fieldID,
  fieldLat,
  fieldLon,
  fieldDateTime = NULL,
  fieldDate = NULL,
  fieldTime = NULL,
  formatDT = NULL,
  cleanDF = FALSE
)
```

**Arguments**

- `dataGroup`: data.frame or data.table.
- `formatBL`: logical. Is data set already in format of BirdLife Seabird tracking database? If so, indicate TRUE. `fieldID` must still be specified and other fields may be ignored. arguments.
- `fieldID`: character. Unique identifier; e.g. for individuals or dataGroup.
- `fieldLat`: numeric. Name of column corresponding to the LATITUDINAL positions.
- `fieldLon`: numeric. Name of column corresponding to the LONGITUDINAL positions.
- `fieldDateTime`: character. If existing, this is the name of the column corresponding to the combined DATE & TIME.
- `fieldDate`: character. Name of column corresponding to the DATE only.
- `fieldTime`: character. Name of column corresponding to the TIME only.
formatFields

formatDT character. What is the format of the data in your DateTime, Date, and Time columns (e.g. "ymd_HMS")? Specify the format following the standard in parse_date_time.

cleanDF logical scalar (T/F). Should columns which are non-essential for track2KBA analysis be removed from dataframe, or not? Removal will speed analysis up a bit.

Details

If data are already in format of BirdLife Seabird tracking database (http://www.seabirdtracking.org/), use formatBL = TRUE and formatting conversion will occur automatically. I.e., data have following columns: "latitude", "latitude", "date_gmt", "time_gmt". You must still specify the ID column as either the track or animal identifier.

By matching up the names of your existing columns with those recognized by track2KBA functions, formatFields re-formats the data frame, and converts the date/date-time fields into a single date-time field of class POSIXct.

If date-time is combined in a single column, please use fieldDateTime instead of fieldDate and fieldTime.

Value

Returns a data.frame, with 'ID', 'Latitude', 'Longitude', and 'DateTime' (class POSIXct) columns.

Examples

```r
## Load example dataset
tracks_raw <- track2KBA::boobies
## using data with user-custom format i.e. with separate Date and Time fields
tracks_formatted <- formatFields(
  dataGroup = tracks_raw,
  fieldID = "track_id",
  fieldLat = "latitude",
  fieldLon = "longitude",
  fieldDate = "date_gmt",
  fieldTime = "time_gmt"
)
## using data with only single Date field
tracks_formatted <- formatFields(
  dataGroup = tracks_raw,
  fieldID = "track_id",
  fieldLat = "latitude",
  fieldLon = "longitude",
  fieldDate = "date_gmt",
  formatDT = "ymd"
)
## Not run:
## if data were downloaded from Seabird Tracking Database
tracks_formatted <- formatFields(
  dataGroup = tracks_raw,
  formatBL = TRUE
)
```
indEffectTest

fieldID = "bird_id")
## End(Not run)

indEffectTest  Test site fidelity

Description

indEffectTest tests whether the variance in overlap between space use areas within a group (e.g. within individuals) is significant compared to between groups (e.g. between individuals).

Usage

indEffectTest(
  tracks,
  tripID,
  groupVar,
  plot = TRUE,
  method = c("HR", "PHR", "VI", "BA", "UDOI", "HD"),
  conditional = TRUE,
  levelUD = 50,
  scale,
  grid = 500,
  iterations = 1000
)

Arguments

tracks  SpatialPointsDataFrame. Must be projected into an equal-area coordinate system. If not, first run projectTracks.
tripID  character. Column in tracks corresponding to the within group ID (e.g. trip-individual combination)
groupVar  character. Column in tracks corresponding to the between group ID (e.g. individual or track)
plot  logical scalar (TRUE/FALSE). Do you want to output a boxplot of the result?
conditional  logical scalar (T/F). If TRUE, the function sets to 0 the pixels of the grid over which the UD is estimated, outside the home range of the animal estimated at a level of probability equal to percent. Note that this argument has no effect when method="HR" (from kerneloverlap).
levelUD  numeric. The desired contour level of the utilization distribution to be used in overlap estimation. NOTE: this is irrelevant if conditional=FALSE.
scale numeric (in kilometers). Smoothing (‘H’) parameter for kernel density estimation.

grid numeric or SpatialPixels. If numeric, specify the desired number of grid cells over which the utilization distributions will be estimated. A default grid of 500 cells is used.

iterations numeric. Indicate the desired number of Kolmogorov-Smirnov iterations to run. 500 is an advisable minimum for statistical rigor.

Details

This function works by producing kernel density areas at a desired contour level (i.e. UDLEv) for each level of tripID and estimating the degree of overlap between all pairwise comparisons using the desired overlap method. Then, comparisons are split into 'within' and 'between' groups, determined by the grouping variable (i.e. groupVar) argument.

If conditional=TRUE then the overlap estimates will range from 0 to levelUD (unless method= "HR").

Then, the empirical distribution of each group is compared in a bootstrapped Kolmogorov-Smirnov test, to check whether differences in the distributions are significant. If so, it indicates that individuals within the groupVar reuse sites more than expected by chance.

NOTE: Because indEffectTest relies on kerneloverlap to estimate overlap, it was not possible to implement a res argument as is done in other track2KBA functions. Therefore, it is advised to either leave the default of 500 cells, or ascertain the number of cells in the grid of chosen res from the output of estSpaceUse.

Value

indEffectTest returns a list containing three objects. In the first slot 'Overlap Matrix', the full matrix of overlap comparisons. In the 'Overlap' slot, a dataframe with a column identifying whether each overlap estimate corresponds to a within-group, or a between-group comparison. In the third slot 'Kolmogorov-Smirnov' is the test output of the Kolmogorov-Smirnov test, indicating the D parameter and significance estimates.

Examples

```r
tracks_raw <- track2KBA::boobies
## format data
tracksFormatted <- formatFields(
  dataGroup = tracks_raw,
  fieldID = "track_id",
  fieldLat = "latitude",
  fieldLon = "longitude",
  fieldDate = "date_gmt",
  fieldTime = "time_gmt"
)

colony <- data.frame(
  Longitude = tracksFormatted$Longitude[1],
  Latitude = tracksFormatted$Latitude[1]
)
## Split into trips
```
KDE_example

Trips <- tripSplit(tracks_formatted, 
  colony=colony, 
  innerBuff=2, 
  returnBuff=20, 
  duration=1, 
  nests = FALSE, 
  rmNonTrip = TRUE 
)

## project dataset
tracks_prj <- projectTracks(
  Trips, 
  projType = "azim", 
  custom = "TRUE"
)

## estimate fidelity of individuals across trips
result <- indEffectTest(
  tracks_prj, 
  tripID = "tripID", 
  groupVar = "ID", 
  scale = 30
)

---

**KDE_example**

Utilization distributions examples

**Description**

Four Utilization Distributions derived for four Masked Boobies using track2KBA::estSpaceUse and a scale parameter of 30 km.

**Usage**

KDE_example

**Format**

An "estUDm" object:

- **h** Smoothness parameter in meters
- **proj4string** Spatial projection...
mapKDE

Make simple maps of Kernel Density Estimates

Description

mapKDE uses output from estSpaceUse to create maps illustrating utilization distributions for each ID.

Usage

mapKDE(KDE, colony = NULL, show = TRUE)

Arguments

- **KDE**: Simple feature MULTIPOLYGON or estUDm object. Must be output of `estSpaceUse` function.
- **colony**: data.frame. Optional. ‘Latitude’ and ‘Longitude’ locations to display reference point of, for example, a breeding or tagging site.
- **show**: logical. show plot, or just save it. Note, saving plot only works for Simple Features input. Default is TRUE.

Details

If the input is simple features polygons, these will be displayed for all IDs on same map. If input estUDm utilization distribution surface, each ID level gets its own facet displaying the full UD.

Value

Returns a figure of either single map with all core ranges displayed together, or a series of facetted maps, each of which shows a utilization distribution corresponding to a level of ID in KDE.

See Also

- `estSpaceUse`

Examples

```r
## make some play data
dataGroup <- data.frame(Longitude = c(1, 1.01, 1.02, 1.04, 1.05, 1.03, 1),
Latitude = c(1, 1.01, 1.02, 1.03, 1.021, 1.01, 1),
ID = rep("A", 7),
DateTime = format(lubridate::ymd_hms("2021-01-01 00:00:00") +
lubridate::hours(0:6)
)

## project tracks
tracks <- projectTracks(dataGroup, projType = "azim", custom = TRUE)
```
## estimate utilization distributions for each track
KDE <- estSpaceUse(tracks, scale=10, levelUD = 50)
## map it
mapKDE(KDE)

---

**mapSite**

*Make simple maps of aggregation and important sites*

**Description**

`mapSite` uses output from `findSite` to create maps illustrating density of animals in space, and borders of potentially important areas for the population.

**Usage**

```r
mapSite(Site, colony = NULL, show = TRUE)
```

**Arguments**

- **Site**: Simple feature MULTIPOLYGON object or SpatialPixelsDataFrame. Must be output of `findSite` function.
- **colony**: data.frame. Optional. Must contain columns named 'Latitude' and 'Longitude', with coordinate locations to display reference point of, for example, a breeding or tagging site.
- **show**: logical. show plot, or just save it. Note, saving plot only works for Simple Features input. Default is TRUE.

**Details**

If the input is simple features polygons (i.e. `polyOut = TRUE` in `findSite`), areas which meet threshold of importance are displayed (in red) on top of of the estimated density of animals in space. Black borders are political and coastline borders. If there are no red borders areas displayed on the map, then either the species doesn’t aggregatee enough to meet the threshold, or the tracked sample aren’t representative enough to identify significant aggregations.

If input is SpatialPixelsDataFrame (i.e. `polyOut = FALSE` in `findSite`), a simple density surface map is plotted.

**Value**

Returns a figure of either single map with all core ranges displayed together, or a series of faceted maps, each of which shows a utilization distribution corresponding to a level of ID in `KDE`.

**See Also**

`estSpaceUse`
Examples

KDE <- track2KBA::KDE_example

## identify potential sites
pot_site <- findSite(KDE, represent = 90, levelUD = 50)
## Map it
mapSite(pot_site)

mapTrips
Make simple maps of foraging trips

Description
mapTrips uses output from tripSplit to create maps illustrating movements for each ID.

Usage
mapTrips(trips, colony, IDs = NULL, colorBy = c("complete", "trip"))

Arguments

- **trips** SpatialPointsDataFrame. Must be output of tripSplit function.
- **colony** data.frame. Containing 'Latitude' and 'Longitude' fields specifying the central location(s) from which trips begin. If more than one location, each row should correspond to an appropriate location (Lat/lon) for each ID value in trips.
- **IDs** numeric vector. Sequence of numeric indices for the IDs you wish to map. Max of 25.
- **colorBy** character string. Either "complete" if trips are to be coloured as complete or incomplete, or "trip" if trips are to be coloured by trip ID.

Details
This function only works with the output of tripSplit.

Value
Returns a figure of facetted maps, each of which corresponds to a level of ID in trips.

See Also
tripSplit
move2KBA

Examples

```r
## make some play data
dataGroup <- data.frame(Longitude = rep(c(1:10, 10:1), 2),
                        Latitude = rep(c(1:10, 10:1), 2),
                        ID = c(rep("A", 20), rep("B", 20)),
                        DateTime = format(
                                    lubridate::ymd_hms("2021-01-01 00:00:00") +
                                    lubridate::hours(0:19))
)
colony <- data.frame(  
    Longitude = dataGroup$Longitude[1], Latitude = dataGroup$Latitude[1]
)
Trips <- tripSplit(dataGroup,  
    colony=colony,  
    innerBuff=2,  
    returnBuff=20,  
    duration=1,  
    nests = FALSE,  
    rmNonTrip = TRUE
)
```

```r
## Visualize trips
mapTrips(Trips, colony) # add colony location to each facet
mapTrips(Trips, colony, colorBy = "trip") # color trips by their order
```

move2KBA

*Import Movebank data sets for track2KBA analysis*

Description

move2KBA imports data from Movebank repository and re-formats them to fit track2KBA functions.

Usage

```r
move2KBA(movebankID = NULL, user = NULL, password = NULL, filename = NULL)
```

Arguments

- **movebankID**: character or numeric. Character: full name of the study, as stored on Movebank. Numeric: Movebank ID of the study. Both can be obtained on the Study Details page on Movebank ([https://www.movebank.org](https://www.movebank.org)) or with `getMovebankID`.
- **user**: Username associated with your Movebank account.
- **password**: Password associated with your Movebank username.

Details

This is a wrapper function for functions in move package to import and format tracking data from Movebank. It also attains study site location data (lat/lons).
projectTracks

Description

projectTracks is a convenience function to project tracking data to a an equal-area projection for use in kernel density analysis.

Usage

projectTracks(dataGroup, projType, custom)

Arguments

dataGroup data.frame or SpatialPointsDataFrame. Tracking data, with fields as named by formatFields. Must contain 'Latitude' and 'Longitude' columns.

projType character. Select type of equal-area projection to use. Two options are available: 'cylin' projects to a World Cylindrical Equal Area projection, and 'azim' projects to a Lambert Azimuthal EA.

custom logical (TRUE/FALSE). Choose whether projection will use default centering parameters or whether to set projection center on centroid of latitude and longitude in dataGroup.

Input data can be tracks split into trips (i.e. output of tripSplit)

Value

Returns a list object of length two, containing tracking data (accessed using: dataset$data) and study site location information (accessed using: dataset$site).

See Also

getMovebankData for data download, getMovebank for study metadata, getMovebankID for getting study ID number

Examples

## Not run:

dataset <- move2KBA(movebankID=xxx, user="myusername", password="mypassword")

tracks <- dataset$data ## access tracking data

site   <- dataset$site   ## access study site coordinates

## End(Not run)
projectTracks

Details

Data are transformed to either a World Cylindrical Equal Area, or a Lambert equal-area projection. Cylindrical projections generally appear better for data that are distributed more along one axis, while azimuthal appear better for data that is distributed evenly within a radius. The most important thing is that the data are in an equal-area projection for Kernel Density Analysis (e.g. estSpaceUse).

If custom=TRUE, the projection will be centered on the data. This is particularly preferable for data that cross the international dateline, or near the poles. However, it is important to recognize that this projection is specific to inpute dataset (i.e. dataGroup) so if projectTracks is run again with even slightly different data, the projections will differ, which may cause issues down the line if merging spatial datasets again.

NOTE that these projections may not be the most appropriate for your data and it is almost certainly better to manually identify a projection appropriate for your study region. Custom projections are centered on the centroid of the tracking locations, which is biased for locations close to the poles. In this case it would be better identify an appropriate polar projection for your study area instead of relying on projectTracks. So it is not strictly necessary for projectTracks to be used in track2KBA analysis, what is important is that an equal-area projection of some kind is used when constructing utilization distributions.

Value

Returns a SpatialPointsDataFrame, which can be used for the following functions: findScale, estSpaceUse, indEffectTest, repAssess

See Also

tripSummary

Examples

dataGroup <- data.frame(Longitude = c(1, 2, 3, 1),
                         Latitude = c(1, 1, 2, 1),
                         ID = rep("A", 4),
                         DateTime = format(lubridate::ymd_hms("2021-01-01 00:00:00") +
                                            lubridate::hours(0:3)
                                   )
)

## project tracks
tracks_prj <- projectTracks(
dataGroup,
projType = "azim",
custom = "TRUE"
)
RepAssess estimates the degree to which the space use of a tracked sample of animals represents that of the larger population.

Usage

```r
repAssess(
  tracks,
  KDE = NULL,
  iteration = 1,
  levelUD,
  avgMethod = "mean",
  nCores = 1,
  bootTable = FALSE
)
```

Arguments

- **tracks**: SpatialPointsDataFrame of spatially projected animal relocations. Must include 'ID' field.
- **KDE**: Kernel Density Estimates for individual animals. Several input options: an estUDm, a SpatialPixels/GridDataFrame, or a RasterStack. If estUDm, must be as created by `estSpaceUse` or `adehabitatHR::kernelUD`, if Spatial* each column should correspond to the Utilization Distribution of a single individual or track. If a RasterStack, each layer must be an individual UD.
- **iteration**: numeric. Number of times to repeat sub-sampling procedure. The higher the iterations, the more robust the result.
- **levelUD**: numeric. Specify which contour of the utilization distribution (KDE) you wish to filter to (e.g. core area=50, home range=95).
- **avgMethod**: character. Choose whether to use the arithmetic or weighted mean when combining individual IDs. Options are: 'mean' arithmetic mean, or 'weighted', which weights each UD by the number of points per level of ID.
- **nCores**: numeric. The number of processing cores to use. For heavy operations, the higher the faster. NOTE: CRAN sets a maximum at 2 cores. If using the gitHub version of the package, this can be set to a maximum of one fewer than the maximum cores in your computer.
- **bootTable**: logical (TRUE/FALSE). If TRUE, output is a list, containing in the first slot the representativeness results summarized in a table, and in the second the full results of the iterated inclusion calculations.
Details

Representativeness is assessed by fitting statistical model to the relationship between sample size and inclusion rate. Incursion rate is the proportion of out-sample points included in in-sample space use areas.

First, the set of IDs is iteratively sub-sampled, and in each iteration a set of individual Utilization Distributions (UD, 'KDE' argument) are pooled and the points of the un-selected (out-sample) IDs are overlaid on the area ('levelUD') of the UD. The proportion of these outsaple points which overlap the pooled UD area is known as the inclusion rate, and represents an estimate of representativeness at each sample size. Then, a non-linear function is fit to the relationship between the inclusion rate and sample size (i.e. number of tracks/animals) in order to estimate the point at which the relationship reaches an asymptote (i.e. no more information added per new track). repAssess then estimates the representativeness of the sample by dividing the inclusion rate estimated at the maximum sample size minus 3 (for samples where n < 20), 2 (for samples < 50) or 1 (for sample >100) by this asymptote. The maximum sample size appearing in the plot will be different than the true 'n' of the dataset in order to account for the possible number of combinations of individuals, thereby ensuring a robust result. The maximum sample size reflects the number of KDEs, so if any ID has fewer than 5 points, this ID is omitted from the analysis. Finally, using this relationship, minimum representative sample sizes (70

repAssess accepts UDs calculated outside of track2KBA, if they have been converted to class RasterStack or SpatialPixelsDataFrame. However, one must make sure that the cell values represent continuous probability densities (i.e. values >=0 which integrate to 1 over the raster) and not not discrete probability masses (i.e. values >=0 which sum to 1), nor home range quantiles (i.e. 0-1, or 0-100 representing

When setting avgMethod care must be taken. If the number of points differ greatly among individuals and the UDs are calculated as classic KDEs (e.g. from estSpaceUse) then the weighted mean is likely the optimal way to pool individual UDs. However, if any other method (for example AKDE, auto-correlated KDE) was used to estimate UDs, then the arithmetic mean is the safer option.

NOTE: this function does not work with fewer than 4 IDs (tracks or individual animals).

Value

if bootTable=FALSE (the default) A single-row data.frame is returned, with columns 'SampleSize' signifying the sample size (i.e., number of KDEs)'out' signifying the percent representativeness of the sample,'type' is the type of asymptote value used to calculate the 'out' value, and 'asym' is the asymptote value used. If bootTable=TRUE, a list returned with above dataframe in first slot and full iteration results in second slot.

There are two potential values for 'type': 'asymptote' is the ideal, where the asymptote value is calculated from the parameter estimates of the successful nls model fit. 'inclusion' is used if the nls fails to converge, or if the fit model is flipped and the asymptote value is negative. In these cases, the mean inclusion rate is taken for the largest sample size.'Rep70' signifies the sample size which is ~70 representative, and 'Rep95' signifies the sample size which approaches the asymptote.

Examples

library(dplyr)
tracks_raw <- track2KBA::boobies
## format data
tripSplit

Split tracking data into trips

Description

tripSplit employs splitSingleID to split data from multiple individuals’ into discrete trips made from centrally-located places.

Usage

```
tripSplit(
  dataGroup, colony, innerBuff = NULL, returnBuff = NULL, duration = NULL, gapLimit = NULL, nests = FALSE, rmNonTrip = FALSE, verbose = TRUE
)
```

```
splitSingleID(
  Track, colony, innerBuff = 15, returnBuff = 45, duration = 12,
)```
Arguments

dataGroup data.frame. Must contain 'Latitude', 'Longitude', 'ID' and 'DateTime' columns (correct format may be assured using formatFields function).

colony data.frame. Containing 'Latitude' and 'Longitude' fields specifying the central location(s) from which trips begin. If data are from MoveBank this information may be extracted using the move2KBA function. If nests=TRUE, each row should correspond to an appropriate location (Lat/Lon) for each ID value in dataGroup.

innerBuff numeric (in kilometers). Indicate the distance that an animal must travel for the movement to be considered a trip. Note that this is also the metric that determines whether two subsequent trips are split - if your animal records locations > innerBuff (km) from its place of origin and no locations at the place of origin (e.g. for burrow-nesting species) then subsequent trips may be lumped into a single trip. Increase innerBuff to ensure correct splitting of trips.

returnBuff numeric (in kilometers). Indicate the proximity required for a trip to be considered as returning. This is useful for identifying incomplete trips (i.e. where storage/transmission failed during the trip).

duration numeric (in hours). The period of time that the animals must be at large for the movement to be considered a trip.

gapLimit numeric (in days). The period of time between points to be considered too large to be a contiguous tracking event. Can be used to ensure that deployments on the same animal in different years do not get combined into extra long trips. Defaults to one year.

nests logical scalar (TRUE/FALSE). Should the central place used in trip-splitting be specific to each ID? If so, each place must be matched with an 'ID' value in both dataGroup and colony objects.

rmNonTrip logical scalar (TRUE/FALSE). Should periods not associated with trips be filtered out? Note that this does not filter out the trip start and end points which fall within innerBuff. Defaults to FALSE.

verbose logical scalar (TRUE/FALSE). Should the function print messages when trips start outside the innerBuffer or doesn’t return to the 'colony'? Default is TRUE. tripSummary. Default is TRUE.

Track dataFrame.

Details

This function splits central place foraging animal movement data into individual trips away from a central location based on distance and time.

nests=TRUE may be used if it is desired, for example, to use specific nest locations instead of one central location for all individuals/dataGroup.
Value

Returns an un-projected (WGS84) SpatialPointsDataFrame, with the field 'tripID' added to identify each unique trip-ID combination. If rmNonTrip=TRUE, then output has been filtered of points deemed not associated with trip movements.

See Also

tripSummary, mapTrips

Examples

```r
## make some play data
dataGroup <- data.frame(Latitude = rep(c(1:10, 10:1), 2),
                      Longitude = rep(c(1:10, 10:1), 2),
                      ID = c(rep("A", 20), rep("B", 20)),
                      DateTime = format(lubridate::ymd_hms("2021-01-01 00:00:00") +
                                                  lubridate::hours(0:19))
)
colony <- data.frame(
  Longitude = dataGroup$Longitude[1],
  Latitude = dataGroup$Latitude[1]
)
## split tracks into trips
Trips <- tripSplit(dataGroup,
  colony=colony,
  innerBuff=2,
  returnBuff=20,
  duration=1,
  nests = FALSE,
  rmNonTrip = TRUE
)
```

---

**tripSummary** *Summary of trip movements*

Description

tripSummary provides a simple summary of foraging trip distances, durations, and directions performed by central place foraging animals.

Usage

```r
tripSummary(trips, colony = NULL, nests = FALSE, extraDist = FALSE)
```
Arguments

- **trips**: SpatialPointsDataFrame, as produced by `tripSplit`.
- **colony**: data.frame with 'Latitude' and 'Longitude' columns specifying the locations of the central place (e.g. breeding colony). If `nests=TRUE`, colony should have a third column, 'ID' with corresponding character values in the 'ID' field in `trips`.
- **nests**: logical scalar (TRUE/FALSE). Were central place (e.g. deployment) locations used in `tripSplit` specific to each unique 'ID'? If so, each place must be matched with an 'ID' value in both `trips` and `colony` objects.
- **extraDist**: logical scalar (TRUE/FALSE). If TRUE, the distance between the first and last points of each trip and the colony will be added to the 'total_dist' (total distance travelled) for each trip.

Details

`nests=T` may be used if it is desired, for example, to use specific nest locations instead of one central location for all individuals/track.

Value

Returns a tibble data.frame grouped by ID. Trip characteristics included are trip duration (in hours), maximum distance and cumulative distance travelled (in kilometers), direction (in degrees, measured from origin to furthest point of track), start and end times as well as a unique trip identifier ('tripID') for each trip performed by each individual in the data set. Distances are calculated on a great circle.

If the beginning of a track is starts out on a trip which is followed by only one point within `InnerBuff`, this is considered an 'incomplete' trip and will have an NA for duration. If an animal leaves on a trip but does not return within the `ReturnBuff` this will be also classified an 'incomplete trip'.

See Also

`tripSplit`

Examples

```r
## make some play data
dataGroup <- data.frame(Longitude = rep(c(1:10, 10:1), 2),
                        Latitude = rep(c(1:10, 10:1), 2),
                        ID = c(rep("A", 20), rep("B", 20)),
                        DateTime = format(lubridate::ymd_hms("2021-01-01 00:00:00") +
                                           lubridate::hours(0:19))
)

colony <- data.frame(
   Longitude = dataGroup$Longitude[1],
   Latitude = dataGroup$Latitude[1]
)
## split tracks into trips
trips <- tripSplit(dataGroup, colony=colony,
                   innerBuff = 1,
```
returnBuff = 1,
duration = 0.5,
rmNonTrip = FALSE
)
## summarise trip characteristics
sumTrips <- tripSummary(trips, colony)
Index

* datasets
  boobies, 2
  KDE_example, 13

boobies, 2

estSpaceUse, 3, 7, 12, 14, 15, 19–21

findScale, 3, 4, 4, 19
findSite, 7, 13
formatFields, 4, 9, 18, 23
fpt, 6

getMovebank, 18
getMovebankData, 18
getMovebankID, 17, 18
indEffectTest, 11, 19

KDE_example, 13
kerneloverlap, 11, 12
kernelUD, 3

mapKDE, 14
mapSite, 15
mapTrips, 16
move2KBA, 17, 23

parse_date_time, 10
projectTracks, 3, 5, 11, 18

repAssess, 7, 8, 19, 20, 21

splitSingleID(tripSplit), 22

tripSplit, 4, 16, 18, 22, 25
tripSummary, 5, 6, 19, 24