Package 'PhysMove'

August 21, 2026

Type Package

Title Quantifying Animal Movement and Space-Use Patterns with Statistical Physics

Version 1.2.5

Date 2026-08-05

Description Provides tools to analyse animal movement and space-use patterns from telemetry data using methods derived from statistical physics. Methods span displacement-based approaches, distribution fitting, space-use metrics (including the influence of correlations on space-use), network-based community detection, and measures of entropy and predictability. The package enables characterisation of these patterns across spatial and temporal scales, including variation within and among individuals (inter- and intraspecific analyses). Outputs include interpretable metrics and visualisations to support ecological analysis and the investigation of fundamental movement processes. For applications of these methods in ecological studies see Rodríguez et al. (2017) <doi:10.1038/s41598-017-00165-0> and Sequeira et al. (2018) <doi:10.1073/pnas.1716137115>.

License GPL (>= 3)

URL https://github.com/HannahCalich/PhysMove

BugReports https://github.com/HannahCalich/PhysMove/issues

Depends R (>= 4.4)

Imports broom (>= 1.0.5), ggplot2 (>= 3.4.2), graphics, grDevices, grid, methods, poweRlaw (>= 1.0.0), RColorBrewer (>= 1.1-3), rlang (>= 1.1.1), rootSolve (>= 1.8.2.3), scales (>= 1.2.1), sf (>= 1.0-16), stats, utils

Suggests emln (>= 1.0), infomapecology (>= 2.0.0), kableExtra (>= 1.3.4), knitr (>= 1.43), maps (>= 3.4.2), officedown (>= 0.3.0), rmarkdown (>= 2.22), spelling (>= 2.2.1), testthat (>= 3.1.9)

VignetteBuilder knitr

Additional_repositories https://HannahCalich.github.io/drat

Config/Needs/check rcmdcheck

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

NeedsCompilation no

Author Hannah J. Calich [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7541-9584>),
Jorge Rodríguez [aut] (ORCID: <https://orcid.org/0000-0001-6593-5032>),
Víctor Eguíluz [aut] (ORCID: <https://orcid.org/0000-0003-1133-1289>),
Ana M. M. Sequeira [aut] (ORCID:
<https://orcid.org/0000-0001-6906-799X>)

Maintainer Hannah J. Calich <hannah.calich@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 07:50:08 UTC

Contents

angleList . . . 3
angleListAll . . . 3
calcDisp . . . 4
checkTracks . . . 5
communityMap . . . 6
compDist . . . 7
disp . . . 7
dispAll . . . 8
distResultsAll . . . 8
distResultsExp . . . 9
distResultsTrunc . . . 9
entropy . . . 10
entropyResults . . . 11
fitDist . . . 11
gyrationRad . . . 12
infomapCommunities . . . 13
infomapResult . . . 14
occupancy . . . 14
occupancyResults . . . 15
plotAngles . . . 16
plotDispPDF . . . 17
plotDist . . . 18
plotPDF . . . 19
plotRandomTracks . . . 20

plotTracks 21
predictability 22
randomise 23
randomResults 24
rms 24
tracks 25
tracksCRW 26
turningAngles 27

Index **29**

`angleList` *Example output from turningAngles()*

Description

Example output from the `turningAngles` function. Angles calculated using the `tracks` dataset and turningAngles() with max_hr=24. Note that the turningAngles() default is normally max_hr=240; however, this creates an unnecessarily large file for an example dataset so max_hr=24 was used instead.

Usage

```
angleList
```

Format

list

`angleListAll` *Example output from turningAngles()*

Description

Example output from the `turningAngles` function. Angles calculated using the `tracks` dataset and turningAngles() default parameters.

Usage

```
angleListAll
```

Format

list

`calcDisp` *Calculate displacements*

Description

This function allows you to calculate the displacement distances travelled by individuals over set time windows.

Usage

```
calcDisp(
  species_df,
  min_hr = 24,
  max_hr = 240,
  interval_hr = 24,
  range_hr = 6,
  verbose = FALSE
)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each individual (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following yyyy-mm-dd hh:mm:ss. See attached sample data `tracks`.
`min_hr`	Minimum number of hours to consider for calculations. Default is 24 hours.
`max_hr`	Maximum number of hours to consider for calculations. Default is 240 hours.
`interval_hr`	Time interval (in hours) used to set intervals between min_hr and max_hr. Default is 24 hours.
`range_hr`	Range (in hours) converts interval_hr into a time window (interval_hr +/- range_hr) so the code can identify location estimates that are close to, but not exactly separated by the interval_hr input value. If multiple location estimates fall within this time window the location estimate closest to the interval_hr input value will be used for calculations. For example, if interval_hr = 24 and range = 6, the algorithm will search for locations spaced 18 to 30 hours apart. Default is 6.
`verbose`	Logical. If TRUE, informative messages describing the number of displacements identified for each time window are displayed. Default is FALSE.

Value

A list containing the displacements in km recorded for each time window. Each list element corresponds with the time windows set (i.e., the first list element is the first time window).

Examples

```
calcDisp(tracks, min_hr=24, max_hr=240, interval_hr=24, range_hr=6, verbose=TRUE)
```

`checkTracks` *Check data format This function checks the format of telemetry data prior to running PhysMove metrics.*

Description

Check data format This function checks the format of telemetry data prior to running PhysMove metrics.

Usage

```
checkTracks(species_df, verbose = TRUE)
```

Arguments

`species_df`	A data frame containing telemetry data with columns named ref, lon, lat, and day.
`verbose`	Logical. If TRUE feedback is provided on your dataset. Default is TRUE.

Details

The columns must be formatted as follows: ref: numeric ID for each individual. lon and lat: numeric longitude and latitude in decimal degrees. day: POSIXct datetime values. Datetime format:

Value

Invisibly returns an integer error count (0 if no issues found), in addition to printing diagnostic messages/warnings.

Examples

```
checkTracks(tracks)
```

communityMap *Map Infomap communities*

Description

This function allows you to create a map of the level 1 Infomap communities calculated using the infomapCommunities function. To map only a selection of the communities use the subset_communities parameter.

Usage

```
communityMap(infomap_output, subset_communities, colours = "Dark2")
```

Arguments

infomap_output	Output list from the infomapCommunities function, from which the Infomap monolayer object is extracted internally
subset_communities	Concatenated vector of level 1 communities to be mapped. For example, subset_communities=c(1,2,3) will plot level 1 communities 1, 2, and 3. This parameter is particularly useful if Infomap has identified many communities and they are difficult to distinguish in the map All communities are included by default.
colours	Colour(s) for each community in the map. Valid input options include: base R (grDevices) color pallets (e.g., colours=rainbow), RColorBrewer palettes (e.g., colours="Dark2"), and colour names or hex numbers (e.g.,colours=c("darkred", "#4682B4", "#00008B", "darkgreen")). Note that grDevices color pallets do not use quotations. If the palette does not have enough distinct colours to match the communities being plotted the function will automatically create a continuous pallet with the colours provided. Default is "Dark2".

Value

A map illustrating level 1 Infomap communities.

Examples

```
communityMap(infomapResult)
```

`compDist` *Identify the best-fit distribution for data*

Description

This function allows you to determine if a power law, exponential, or log-normal distribution best-fit a probability density function of the input data using weighted Akaike Information Criterion (AIC). These fits use input data in conjunction with dmin and parameter values that were previously calculated with the `fitDist` function. By default, this function will calculate AICc scores (AIC scores corrected for small sample sizes) if n/K is <= 40 for the largest value of K, where n = sample size (nTail) and K = number of parameters in the model (see Burnham and Anderson (2004) for further details, <doi:10.1177/0049124104268644>). However, if force_AICc = TRUE AICc scores will be calculated regardless of n/K.

Usage

```
compDist(input, distResults, force_AICc = FALSE)
```

Arguments

`input`	List of values used to evaluate and compare distribution fits generated by fitDist()
`distResults`	List output from the `fitDist` function containing a dataframe of fit results (element 1) and a normalisation record (element 2)
`force_AICc`	Force function to calculate AICc scores instead of AIC scores when n/K is > 40. Default is FALSE.

Value

A data frame with that contains the summary statistics for each distribution fit (from the `fitDist` function) as well as the AICc/AIC scores and weighted AICc/AIC scores (wAICc/wAIC) for each distribution fit.

Examples

```
compDist(disp, distResultsAll, force_AICc=FALSE)
```

`disp` *Example output from calcDisp()*

Description

Example output from the `calcDisp` function. Displacements calculated using the `tracks` dataset and calcDisp() with max_hr=24. Note that the calcDisp() default is normally max_hr=240; however, this creates an unnecessarily large file for an example dataset so max_hr=24 was used instead.

Usage

```
disp
```

Format

list

---

dispAll *Example output from calcDisp()*

Description

Example output from the calcDisp function. Displacements calculated using the tracks dataset and calcDisp() with default parameters.

Usage

```
dispAll
```

Format

list

---

distResultsAll *Example output from fitDist()*

Description

Example output from the fitDist function. Full distribution fits calculated using the tracks dataset and fitDist() with full=TRUE. The first list element contains a data frame of distribution results (dmin, parameters, etc) and the second list element is a record of if the data were normalized or not, which is needed for compDist and plotDist.

Usage

```
distResultsAll
```

Format

list

distResultsExp *Example output from fitDist() using the best-fit dmin for an exponential distribution*

Description

Example output from the fitDist function. Distribution fits calculated using the tracks dataset and fitDist() with set_dmin=1.649160, which is the best-fit dmin for an exponential distribution for the tracks dataset. The first list element contains a data frame of distribution results (dmin, parameters, etc) and the second list element is a record of if the data were normalized or not, which is needed for compDist and plotDist.

Usage

```
distResultsExp
```

Format

list

distResultsTrunc *Example output from fitDist()*

Description

Example output from the fitDist function. Full distribution fits calculated using the tracks dataset and fitDist() with full=FALSE. The first list element contains a data frame of distribution results (dmin, parameters, etc) and the second list element is a record of if the data were normalized or not, which is needed for compDist and plotDist.

Usage

```
distResultsTrunc
```

Format

list

`entropy` *Entropy of trajectories*

Description

This function calculates the normalised entropy of individual trajectories based on the probability distribution of location observations across grid cells. Normalised entropy scores are calculated by dividing individual entropy scores by the log number of cells each trajectory visited, providing insight to how ordered or disordered the trajectories were. Values close to 1 indicate high entropy (disordered trajectories), while values closer to 0 indicate low entropy (ordered trajectories). A pdf plot of the normalized entropy values can be created with the `plotPDF` function.

Usage

```
entropy(species_df, gridCell = 0.25, histPlot = TRUE)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data `tracks`.
`gridCell`	Grid cell size in degrees. Default is 0.25.
`histPlot`	Plot a histogram of the normalised entropy values. Default is TRUE.

Value

Data frame of the normalised entropy values for each trajectory as well as the individual entropy values (not normalised) and the number of cells each trajectory visited. If histPlot=TRUE a histogram of the normalised entropy scores is created.

Examples

```
entropy(tracks, gridCell=0.25, histPlot=TRUE)
```

entropyResults *Example output from entropy()*

Description

Example output from the entropy function. Results calculated using the tracks dataset with entropy() default parameters

Usage

```
entropyResults
```

Format

data.frame

fitDist *Fit distributions to data*

Description

This function allows you to fit power law, exponential, or lognormal distributions to a list of values (e.g., displacement data).

Usage

```
fitDist(
  input,
  dist = c("pl", "exp", "lnorm"),
  set_dmin = NULL,
  full = FALSE,
  normalise = TRUE
)
```

Arguments

input	List of values used to fit the specified distributions; values are combined (and optionally normalised) prior to fitting.
dist	Continuous distributions that will be fit to the data. Possible values are power law ("pl"), exponential ("exp"), or lognormal ("lnorm"). Default is dist=c("pl","exp","lnorm").
set_dmin	To limit the fitted distribution to values above a specified value. If your data are going to be normalised this value will have to be a normalised value as well. Default is NULL.
full	To fit the distributions to the full range of data. Default is FALSE.
normalise	Normalises the input values by dividing each input value by the mean of its corresponding time window; normalise = TRUE is required if working with data calculated over multiple time windows.

Value

A list including a data frame of summary statistics for each distribution fit (first list element). Results data frame includes the distribution name, dmin (minimum value used to fit each distribution), parameter 1 (alpha, lambda, mu) and parameter 2 (NA, NA, sigma) for pl, exp, and lnorm distributions respectively, and nTail (the number of data points greater than or equal to dmin). A logical argument indicating if data were normalised is exported as the second list element because this information is needed for the `compDist` and `plotDist` functions.

Examples

```
fitDist(disp, dist=c("pl","exp","lnorm"), full=TRUE)
```

---

gyrationRad *Gyration Radius*

Description

This function calculates the gyration radius of individual trajectories. A pdf plot of the gyration radius values can be created with the `plotPDF` function.

Usage

```
gyrationRad(species_df, map = TRUE, mapCol = c("Black", "Red"), verbose = TRUE)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data `tracks`.
`map`	Create a map illustrating the gyration radius of each trajectory. Default is TRUE.
`mapCol`	Colours for points and gyration radii on map, respectively. Default is c("Black","Red").
`verbose`	Logical. If TRUE, an informative message is displayed when map features cannot be added to the plot. Default is TRUE.

Details

Data frame must also be sorted by ref and then day within each ref, see `checkTracks` for details

Value

A data frame containing the unique trajectory identifier (ref), the mean location (longitude and latitude), and the gyration radius (rG, in km) for each trajectory. If map = TRUE, a map of the gyration radius results is also produced

Examples

```
gyrationRad(tracks, map = TRUE, mapCol = c("Black","Red"))
```

`infomapCommunities` *Identify Infomap communities and create a transition probability matrix*

Description

This function uses the network community detection Infomap to identify Infomap communities based on a transition probability matrix (tpm), which summarizes the probability of individuals moving from one grid cell to another over a set time window. This function assumes directed movement, allows for self-links (where an individual stays in the same cell over time), and uses a tpm in link list format to create an Infomap 'monolayer_object'. Note that if warnings appear about columns or rows summing to 0 this simply means an individual moved into a cell and did not leave, which is a valid movement and not cause for alarm.

Usage

```
infomapCommunities(
  species_df,
  gridCell = 0.25,
  hours = 24,
  range_hr = 6,
  tpm = FALSE
)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data `tracks`.
`gridCell`	Grid cell size in degrees. Default is 0.25.
`hours`	Identify locations separated by this number of hours for movement calculations. Default is 24.
`range_hr`	Range (in hours) converts the hours parameter into a time window (hours +/- range_hr) so the code can identify location estimates that are close to, but not exactly separated by a set number of hours. If multiple location estimates fall within this time window the location estimate closest to the set hours input value will be used for calculations. For example, if hours = 24 and range = 6, the algorithm will search for locations spaced 18 to 30 hours apart. Default is 6.
`tpm`	Export the transition probability matrix in link list format. If tpm=TRUE, a 'TransitionProbabilityMatrix' data frame will be automatically returned as the second element of the output list. Default is FALSE.

Details

Please note: to run this function you must first download the infomapecology and emln R packages from GitHub and install the stand-alone Infomap file. For details please see: https://ecological-complexity-lab.github.io/infomap_ecology_package/installation To learn more about Infomap please visit: https://www.mapequation.org/

Example: library (infomapecology) infomapCommunities(tracks, gridCell=0.25, hours=24, range_hr=6, tpm=FALSE)

Value

A list where element 1 ('infomap_object') contains the Infomap results summarising the hierarchical structure of communities. If tpm = TRUE, element 2 ('tpm') contains the transition probability matrix used to construct the network. The transition probability matrix is returned in link list format (origin node, destination node, and transition probability).

---

infomapResult *Example output from infomapCommunities()*

Description

Example output from the `infomapCommunities` function. Results calculated using the `tracks` dataset and infomapCommunities() default parameters. The first list element contains the Infomap results (i.e., an Infomap monolayer object) and the second list element (if tpm = TRUE) includes the transition probability matrix (tpm).

Usage

```
infomapResult
```

Format

list

---

occupancy *Occupancy*

Description

This function allows you to calculate the spatial occupancy patterns of location estimates and create a map. A pdf plot of the occupancy values can be created with the `plotPDF` function.

Usage

```
occupancy(
  species_df,
  gridCell = 0.25,
  map = TRUE,
  colGrad = c("blue", "lightblue", "red")
)
```

Arguments

species_df	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data tracks.
gridCell	Grid cell size in degrees. Default is 0.25.
map	Create a map illustrating where occupancy occurs. Default is TRUE.
colGrad	Colour gradient for occupancy map that illustrates low, moderate, and high occupancy, respectively (applied to ggplot2::scale_fill_gradientn). Default is colGrad=c("blue", "lightblue","red").

Value

A data frame including occupancy values and corresponding locations (provided as the centre values of each grid cell). If map = TRUE, a map is created.

Examples

```
occupancy(tracks, gridCell=0.25, map=TRUE, colGrad=c("blue", "lightblue", "red"))
```

---

occupancyResults *Example output from occupancy()*

Description

Example output from the occupancy function. Results calculated using the tracks dataset and occupancy() default parameters.

Usage

```
occupancyResults
```

Format

data.frame

plotAngles *Plot turning angles with a circle plot*

Description

This function allows you to create a circle plot illustrating the frequency of turning angles from the turningAngles function.

Usage

```
plotAngles(angleList, timePlot = "all", colours = rainbow, legend = TRUE)
```

Arguments

angleList	List of angles calculated with the turningAngles function.
timePlot	Plot angles from all time windows or only plot angles from one specific time window. For example, timePlot=1 will only plot angles from the first time window while timePlot="all" will plot angles from all time windows. Default is timePlot="all".
colours	Colour(s) for lines in circle plot. Valid input options include: base R (grDevices) colour pallets (e.g., colours=rainbow), RColorBrewer palettes (e.g., colours="Dark2"), and colour names or hex numbers (e.g.,colours=c("darkred", "#4682B4", "#00008B", "darkgreen")). Note that grDevices colour pallets are functions and do not use quotations. If the palette does not have enough distinct colours to match the lines being plotted the function will automatically create a continuous pallet with the colours provided. Default is rainbow.
legend	Add a legend to the circle plot. Default is TRUE.

Value

Produces a circle plot of the angles calculated with the turningAngles function. Invisibly returns a data frame containing the values used to generate the plot, including time windows, frequencies, and angles. Assign the output to an object to access these data.

Examples

```
plotAngles(angleList, timePlot="all", colours=rainbow, legend=TRUE)
```

plotDispPDF *Create probability density function (PDF) plots of displacements*

Description

This function allows you to plot probability density functions (pdfs) of displacements. Displacements must be in list format where each list element corresponds to displacements calculated over a specific time window, which is the default output format from the calcDisp function.

Usage

```
plotDispPDF(displacements, normalise = TRUE, colours = rainbow, legend = TRUE)
```

Arguments

displacements	Displacements in list format (e.g., the output from calcDisp).
normalise	Normalise the displacements by the mean displacement for each time window. Default is TRUE.
colours	Colour(s) for plot points. Valid input options include: base R (grDevices) colour pallets (e.g., colours=rainbow), RColorBrewer palettes (e.g., colours="Dark2"), and colour names or hex numbers (e.g.,colours=c("darkred", "#4682B4", "#00008B", "darkgreen")). Note that grDevices colour pallets do not use quotations. If the palette does not have enough distinct colours to match the communities being plotted the function will automatically create a continuous pallet with the colours provided. Default is rainbow.
legend	Add legend with legend=TRUE. Default is TRUE.

Value

Produces probability density function (pdf) plots of displacements. Invisibly returns a data frame containing the calculated pdf values, corresponding displacements, and time window identifiers used to generate the plot. Assign the output to an object to access these data.

Examples

```
plotDispPDF(disp)
```

plotDist *Plot best-fit distributions to complementary cumulative distribution function (ccdf) of data*

Description

This function allows you to plot a complementary cumulative distribution function (ccdf) of values with fit lines based on distribution fits calculated with the `fitDist` function.

Usage

```
plotDist(
  input,
  distResults,
  fitLines = TRUE,
  setDist = NULL,
  colours = c("red", "gold2", "blue"),
  legend = TRUE,
  label = NULL
)
```

Arguments

`input`	List of values used to generate the ccdf plot (corresponding to values used in fitDist).
`distResults`	List output from the `fitDist` function containing a dataframe of fit results (list element 1) and a normalisation record (list element 2)
`fitLines`	Add fit lines based on the parameters calculated with the `fitDist` function. Default is TRUE.
`setDist`	Plot a subset of lines for each distribution fit calculated with the `fitDist` function (e.g., setDist=c("pl","exp")) Options include "pl", "exp", and "lnorm". The lines will be drawn in order from "pl", then "exp", then "lnorm" (when applicable). By default all lines are plotted. Default is NULL.
`colours`	Colours for each fit line. Valid input options include colour names or hex numbers. Default is colours=c("red","gold2","blue").
`legend`	Add legend with legend=TRUE. Default is TRUE.
`label`	X axis label. Note that "Normalised" will automatically be added if distributions were fit to normalised data. Default is NULL and will result in x-axis label of "input data".

Value

Produces a complementary cumulative distribution function (ccdf) plot of the input data with optional fit lines. Invisibly returns a data frame containing the sorted input values and corresponding ccdf values used to generate the plot. Assign the output to an object to access these data.

Examples

```
plotDist(disp, distResultsExp)
```

`plotPDF` *Plot a probability density function*

Description

This function allows you to plot a probability density function (pdf).

Usage

```
plotPDF(result, desc = NULL, nBins)
```

Arguments

`result`	Data used to create plot.
`desc`	Description of input data. This parameter is used to determine how the data are plotted and to assign appropriate x and y plot labels. Valid input options include: "occupancy" (e.g., from the `occupancy` function), "gyrationRad" (e.g., from the `gyrationRad` function), "entropy" (e.g., from the `entropy` function), "predictability" (from the `predictability` function), and NULL. Occupancy pdfs are created on a log-log scale due to the nature of the data while the other desc types are created on a standard xy plot (to plot occupancy on a standard xy scale leave desc as default). Default is NULL.
`nBins`	Number of bins used to calculate the pdf plot (e.g., nBins=25). By default, if desc="occupancy" the code will use 20 log-sized bins (due to the nature of the data) else the number of bins is determined by the range of the data. If the input values range from 0 to 1 (e.g., entropy or predictability results) the code will use 40 bins by default. If the input results fall outside the 0 to 1 range (e.g., gyration radius results) the code will use 15 bins by default.

Value

Produces a probability density function (pdf) plot of the input data. Invisibly returns a data frame containing the values used to generate the plot. Assign the output to an object to access these data.

Examples

```
plotPDF(occupancyResults$Occupancy, desc="occupancy")
```

plotRandomTracks *Plot Randomised Tracks*

Description

Plot locations from original and reshuffled tracks using RandomisedLat and RandomisedLong outputs from the randomise function

Usage

```
plotRandomTracks(
  species_df,
  ref = NULL,
  randomResults,
  numPlot = 1:5,
  colours = c("black", "grey70"),
  tracks = TRUE,
  startCol = "red",
  endCol = "blue",
  legend = TRUE
)
```

Arguments

species_df	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data tracks.
ref	Reference number of track from species_df to plot.
randomResults	Result from randomise function. Note that the attached example dataset randomResults was generated with randTrack=1 (to keep the example data small), so it only contains 1 randomised track; set numPlot=1 when using it.
numPlot	Number of randomised tracks to plot. The randomised tracks were consecutively numbered from 1 to however many you set in the randomise function. The input value can either be any of these individual numbers (e.g., 23), or a range of numbers (e.g., 1:10), which will plot all of the random tracks created within the range. Default is 1:5. Must not exceed the number of randomised tracks available in randomResults (e.g., numPlot=1 if randomResults was generated with randTrack=1).
colours	Colours to plot points from original and randomised tracks, respectively. Default is colours=c("black","grey70").
tracks	Add track lines to the plot. Default is TRUE.

startCol	Colour for origin location. startCol=NULL will cause the symbology of the origin location to match the symbology of the rest of the original track. Default is startCol="red".
endCol	Colour for destination location. endCol=NULL will cause the symbology of the destination location to match the symbology of the rest of the original track. Default is endCol="blue".
legend	Add legend with legend=TRUE (default).

Value

Produces a plot showing the original and randomised track locations. Invisibly returns a data frame containing the randomised track data used to create the plot. Assign the output to an object to access these data.

Examples

```
plotRandomTracks(tracks, ref=1, randomResults=randomResults, numPlot=1)
```

---

plotTracks *Plot Tracks*

Description

Plot species' location estimates and tracks.

Usage

```
plotTracks(species_df, ref = NULL, tracks = TRUE, colours = rainbow)
```

Arguments

species_df	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data tracks.
ref	Reference number of track from species_df to plot, options include an individual number (ref=1) or a range of numbers (ref=1:10). By default all unique reference numbers are plotted. Default is NULL.
tracks	Add track lines to the plot. Default is TRUE.
colours	Colour(s) for plot points. Valid input options include: base R (grDevices) color pallets (e.g., colours=rainbow), RColorBrewer palettes (e.g., colours="Dark2"), and colour names or hex numbers (e.g.,colours=c("darkred", "#4682B4", "#00008B", "darkgreen")). Note that grDevices colour pallets do not use quotations. If the palette does not have enough distinct colours to match the communities being plotted the function will automatically create a continuous pallet with the colours provided. Default is rainbow.

Value

Map of location estimates and tracks (if tracks=TRUE).

Examples

```
plotTracks(tracks, ref=NULL, tracks=TRUE, colours=rainbow)
```

`predictability` *Predictability of trajectories*

Description

This function allows you to calculate the limit of predictability for each trajectory based on each individual's entropy. This function requires 'indivEntropy', 'cellsVisited', and 'normalisedEntropy' from the `entropy` function. A pdf plot of the predictability values can be created with the `plotPDF` function.

Usage

```
predictability(species_df, entropyResults, startVal = NULL, histPlot = TRUE)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data `tracks`.
`entropyResults`	Data frame of results output from the `entropy` function.
`startVal`	Optional starting value used to find a root value for the limit of predictability equation. If NULL (default), the starting value is automatically determined from the normalised entropy for each individual. The function will iteratively decrease the starting value by 0.01 until an acceptable root within (0,1) is found.
`histPlot`	Plot a histogram of the limit of predictability scores. Default is TRUE.

Value

Limit of predictability values for each trajectory. If histPlot=TRUE a histogram of the limit of predictability scores is created.

Examples

```
predictability(tracks, entropyResults, startVal = NULL, histPlot=TRUE)
```

`randomise` *Randomise tracks*

Description

This function allows you to investigate the influence spatial and/or temporal correlations may have on an individuals' space use patterns. This is done by maintaining the origin and end location of each track, randomizing the order displacements occurred in between these points, and calculating how many grid cells the original and randomised tracks visited, which summarizes their space use.

Usage

```
randomise(species_df, randTrack = 100, gridCell = 0.25, plot = TRUE, lm = TRUE)
```

Arguments

`species_df`	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data `tracks`.
`randTrack`	Number of randomised tracks per individual. Default is 100.
`gridCell`	Grid cell size in degrees. Default is 0.25.
`plot`	Plot the number of cells visited in the original track versus the average number of cells visited in the reshuffled tracks. Default is TRUE.
`lm`	Calculate a linear regression to examine the relationship between the average number of cells visited by the Randomised tracks (target variable) and the number of cells visited in the original tracks (predictor variable). If plot = TRUE this parameter adds a solid black fit line to the data points and a black dashed line, which represents a 1:1 relationship. The slope of the fit line can be determined by typing 'randomResultslmestimate[2]'. Default is TRUE.

Value

List containing a dataframe of results (list element 1), the randomised longitude and latitude values (list elements 2 and 3, respectively), which are needed for the `plotRandomTracks` function, and if lm = TRUE, the results of the linear model are output (list element 4). The dataframe of results includes columns for the number of cells visited by each original track and the average number of cells visited by the Randomised tracks for each ref. Lastly, if plot = TRUE, a plot illustrating the number of cells visited by the original and randomised tracks is created, and if lm = TRUE, a fit line and reference line are added to the plot.

Examples

```
randomise(tracks, randTrack=100, gridCell=0.25, plot=TRUE, lm=TRUE)
```

`randomResults` *Example output from randomise()*

Description

Example output from the `randomise` function. Results calculated using the `tracks` dataset and randomise() with randTrack=1. Note that the randomise() default is normally randTrack=100; however, this creates an unnecessarily large file for an example dataset so randTrack=1 was used instead.

Usage

```
randomResults
```

Format

list

`rms` *Root-Mean-Square of Displacements*

Description

This function allows you to calculate root-mean-square displacements and plot them as a function of time

Usage

```
rms(
  species_df,
  timeUnit = "days",
  wBins = 1.1,
  plot = TRUE,
  lm = TRUE,
  strict = TRUE,
  verbose = TRUE
)
```

Arguments

species_df	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' Data frame must also be sorted by ref and then day within each ref, see checkTracks for details.
timeUnit	Unit used to calculate time between locations (e.g., "secs", "mins", "hours", "days"). Default is "days".
wBins	Bin width refers to the size of the time bins used to calculate how frequently displacements occurred. Default is 1.1
plot	Plot the root-mean-square and mean displacements against their corresponding time periods. Default is TRUE.
lm	Calculate a linear regression to examine the relationship between the root-mean-square displacement values (target variable) and time (predictor variable) and add fit line to the plot (if plot=TRUE). Default is TRUE.
strict	If TRUE, abort with a detailed error when invalid time/displacement pairs are found; if FALSE, warn and proceed using only valid pairs (up to 25 examples shown). Default is TRUE.
verbose	Logical. If TRUE, progress messages are displayed during calculations, including completion updates and the estimated scaling exponent. Default is TRUE.

Value

List containing a dataframe of results (list element 1) and the results of the linear model (if lm = TRUE, list element 2). The results dataframe includes the 'timeWindows' in log-sized bins along with their corresponding 'meanDisplacements' and 'rmsDisplacements' (root-mean-square displacements). If plot = TRUE, a plot of the mean displacement values and the root-mean-square displacement values against their corresponding time period is created, and if lm=TRUE, a fit line and reference line are added to the plot.#'

Examples

```
rms(tracks, timeUnit="days", wBins=1.1, plot=TRUE, lm=TRUE, verbose=TRUE)
```

---

tracks *Sample location data to demonstrate PhysMove functions*

Description

A data frame containing location data from 25 sample trajectories.

Usage

```
tracks
```

Format

A data frame with 15623 rows and 4 variables:

ref trajectory ID number as an integer

lon longitude of each position estimate in decimal degrees

lat latitude of each position estimate in decimal degrees

day datetime stamp for each location estimate in POSIXct format

---

tracksCRW *Sample location data to demonstrate PhysMove functions*

Description

A data frame containing location data from 25 sample trajectories that were developed to follow a correlated random walk model using the aniMotum R package.

Usage

```
tracksCRW
```

Format

A data frame with 15623 rows and 4 variables:

ref trajectory ID number as an integer

lon longitude of each position estimate in decimal degrees

lat latitude of each position estimate in decimal degrees

day datetime stamp for each location estimate in POSIXct format

Details

Reference: https://ianjonsen.r-universe.dev/aniMotum

turningAngles *Calculate turning angles from trajectories*

Description

This function allows you to calculate turning angles between sets of three consecutive location estimates separated by set time window(s).

Usage

```
turningAngles(
  species_df,
  min_hr = 24,
  max_hr = 240,
  interval_hr = 24,
  range_hr = 6,
  histPlot = c(TRUE, "all"),
  verbose = TRUE
)
```

Arguments

species_df	A data frame containing location data in rows. Columns have the following headers: "ref", "lon", "lat", "day". "ref" is the unique id number for each animal (e.g., their satellite tag number formatted as an integer), "lon" and "lat" are the longitude and latitude of each position estimate in decimal degrees in numeric format, "day" is the datetime stamp for each location estimate in POSIXct format following ' See attached sample data tracks.
min_hr	Minimum number of hours to consider for calculations. Default is 24 hours (i.e., 1 day).
max_hr	Maximum number of hours to consider for calculations. Default is 240 hours (i.e., 10 days).
interval_hr	Time interval (in hours) used to set intervals between min_hr and max_hr. Default is 24 hours (i.e., 1 day).
range_hr	Range (in hours) converts interval_hr into a time window (interval_hr +/- range_hr) so the code can identify location estimates that are close to, but not exactly separated by the interval_hr input value. If multiple location estimates fall within this time window the location estimate closest to the interval_hr input value will be used for calculations. For example, if interval_hr = 24 and range_hr = 6, the algorithm will search for locations spaced 18 to 30 hours apart. Default for range_hr is 6.
histPlot	Plot a histogram showing the frequency of turning angles from all time windows combined (default) or one specific time period. For example, histPlot=c(TRUE,1) to plot only the first time period. Default is histPlot=c(TRUE, "all").
verbose	Logical. If TRUE, progress messages are displayed during calculations. Default is TRUE.

Value

List of turning angles for each time window, the name of each list element corresponds with a time window in days. If histPlot = TRUE, a histogram of results is created.

Examples

```
turningAngles(tracks, min_hr=24, max_hr=240, interval_hr=24,
              range_hr=6, histPlot=c(FALSE, "all"), verbose=TRUE)
```

Index

∗ **datasets**
  angleList, 3
  angleListAll, 3
  disp, 7
  dispAll, 8
  distResultsAll, 8
  distResultsExp, 9
  distResultsTrunc, 9
  entropyResults, 11
  infomapResult, 14
  occupancyResults, 15
  randomResults, 24
  tracks, 25
  tracksCRW, 26

angleList, 3
angleListAll, 3

calcDisp, 4, *7, 8*, *17*
checkTracks, 5, *12*, *25*
communityMap, 6
compDist, 7, *8, 9*, *12*

disp, 7
dispAll, 8
distResultsAll, 8
distResultsExp, 9
distResultsTrunc, 9

entropy, 10, *11*, *19*, *22*
entropyResults, 11

fitDist, *7–9*, 11, *18*

gyrationRad, 12, *19*

infomapCommunities, *6*, 13, *14*
infomapResult, 14

occupancy, 14, *15*, *19*
occupancyResults, 15

plotAngles, 16
plotDispPDF, 17
plotDist, *8, 9*, *12*, 18
plotPDF, *10*, *12*, *14*, 19, *22*
plotRandomTracks, 20, *23*
plotTracks, 21
predictability, *19*, 22

randomise, *20*, 23, *24*
randomResults, *20*, 24
rms, 24

tracks, *3, 4*, *7–15*, *20–24*, 25, *27*
tracksCRW, 26
turningAngles, *3*, *16*, 27