Package ‘wildlifeDI’

June 16, 2021

Type     Package
Title    Calculate Indices of Dynamic Interaction for Wildlife Tracking Data
Version  0.4.1
Description Dynamic interaction refers to spatial-temporal associations in the movements of two (or more) animals. This package provides tools for calculating a suite of indices used for quantifying dynamic interaction with wildlife telemetry data. For more information on each of the methods employed see the references within. The package (as of version >= 0.3) also has new tools for automating contact analysis in large tracking datasets. The package draws heavily on the classes and methods developed in the 'adehabitat' packages.

URL      https://github.com/jedalong/wildlifeDI
Depends  R (>= 3.5.0)
Suggests adehabitatHR, ggplot2, GGally, igraph, network, knitr, rmarkdown, nlme
Imports   adehabitatLT, dplyr, graphics, rgeos, sf, sp, stats
License  GPL-3
VignetteBuilder knitr
RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation no
Author   Jed Long [aut, cre] (<https://orcid.org/0000-0002-2815-0399>)
Maintainer Jed Long <jed.long@uwo.ca>
Repository CRAN
Date/Publication 2021-06-16 15:20:02 UTC

R topics documented:

wildlifeDI-package ........................................... 2
Ca ................................................................. 3
wildlifeDI-package

wildlifeDI: Calculate Indices of Dynamic Interaction for Wildlife Tracking Data

Description

Dynamic interaction refers to spatial-temporal associations in the movements of two (or more) animals. This package provides tools for calculating a suite of indices used for quantifying dynamic interaction with wildlife telemetry data. For more information on each of the methods employed see the references within. The package (as of version 0.3) also has new tools for automating contact analysis in large tracking datasets. The package draws heavily on the classes and methods developed in the 'adehabitat' packages.

Details

The package wildlifeDI allows users to compute a number of currently available indices of dynamic interaction useful for wildlife telemetry studies. The currently available methods include:

- Prox - Proximity analysis (Bertrand et al. 1996)
• Ca - Coefficient of Association (Bauman 1998)
• Don - Doncaster’s measure of dynamic interaction (Doncaster 1990)
• Lixn - Minta’s measures of spatial-temporal interaction (Minta 1992)
• Cs - Coefficient of Sociality (Kenward et al. 1993)
• HAI - Half-weight Association Index (Atwood and Weeks Jr. 2003)
• Cr - Correlation coefficient (Shirabe 2006)
• DI - Dynamic interaction index (Long and Nelson 2013)
• IAB - Interaction statistic (Benhamou et al. 2014)

The package `wildlifeDI` also provides useful functionality for identifying which fixes are temporally simultaneous, required for many of the above methods, using the function `GetSimultaneous`, along with other functions for exploring spatial-temporal interactions patterns in wildlife telemetry data.

When citing this package please see citation (`'wildlifeDI`), also please cite the appropriate papers associated with individual methods being used.

The functions in `wildlifeDI` utilize the `ltraj` objects from the package `adehabitat`. For more information on objects of this type see `help(ltraj)`.

**Author(s)**

Jed Long

---

**Ca**

*Coefficient of Association*

**Description**

This function measures the dynamic interaction between two moving objects following the methods first described by Cole (1949), and more recently employed by Bauman (1998).

**Usage**

`Ca(traj1, traj2, tc = 0, dc = 50)`

**Arguments**

- **traj1**: an object of the class `ltraj` which contains the time-stamped movement fixes of the first object. Note this object must be a type II `ltraj` object. For more information on objects of this type see `help(ltraj)`.
- **traj2**: same as `traj1`.
- **tc**: temporal tolerance limit (in seconds) for defining when two fixes are simultaneous or together. Parameter passed to function `GetSimultaneous`.
- **dc**: distance tolerance limit (in appropriate units) for defining when two fixes are spatially together.
Details

This function can be used to calculate the Cole (1949) measure of dynamic interaction between two animals. Termed a coefficient of association, the Ca statistic tests the number of fixes the animals are observed together against the total number of fixes following:

\[ Ca = \frac{2AB}{A + B} \]

where \( A \) (respectively \( B \)) is the number of times animal 1 (resp. 2) are observed, and \( AB \) is the number of times the two animals are observed together. Several works, including Bauman (1998) have suggested that \( Ca > 0.5 \) indicates affiliation or fidelity, while \( Ca < 0.5 \) indicates no association between the two animals. Note that this function calls `GetSimultaneous` to identify the temporal component of identifying when fixes together.

Value

This function returns a numeric result of the Ca statistic.

References


See Also

`GetSimultaneous`, `Prox`, `HAI`

Examples

```r
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes, dc = 50 meters
Ca(deer37, deer38, tc = 7.5*60, dc = 50)
```

Description

The function `checkTO` is a simple function for identifying if, and for how long, two telemetry datasets overlap (temporally) with each other. The function returns a list with three pieces of information: a logical variable indicating if the two trajectories overlap temporally, and timings of the beginning and end of the overlap period.
Usage
checkTO(traj1, traj2)

Arguments
traj1
an object of the class \texttt{ltraj} which contains the time-stamped movement fixes of the first object. Note this object must be a type II \texttt{ltraj} object. For more information on objects of this type see \texttt{help(ltraj)}.

traj2
same as traj1.

Details
The function \texttt{checkTO} can be used to identify if, when, and for how long two telemetry datasets overlap temporally.

Value
A list of with three pieces of information, whether the two trajectories overlap ($\text{T0}$) a logical vector, the beginning ($\text{T0start}$), and end ($\text{T0end}$) of the overlap period, stored as \texttt{POSIX} objects.

See Also
GetSimultaneous, GetTO

Examples
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
spts <- checkTO(deer37, deer38)

\begin{verbatim}
conContext
\end{verbatim}

\textit{Examine context associated with contact phases}

Description
Extracts the variables associated with \texttt{var} before, during, and after contact phases, based on some specified time-lag.

Usage
\begin{verbatim}
conContext(
ltraj,
var = "dist",
def = "all",
idcol = "burst",
nrand = 0,
)\end{verbatim}
nlag = 0,
lag = 0,
gap = 0,
phaid
)

Arguments

ltraj an object of the class ltraj which should be output from the function conPhase.
var name(s) (as character) of columns (possibly from infolocs) to keep for contextual analysis.
def how to define the point-of-contact. The default is to define it as all fixes in a phase def = 'all', alternatively contacts can be defined as a single point along the phase defined as one of: 'first', 'last', 'minDist', 'minTime', which corresponds to the first fix in the contact phase, the last fix in the contact phase, the fix with the minimum time difference and the fix with the closest contact distance.
idcol column id associated with IDs of individuals, default is the 'burst'.
nrand number of random fixes to be selected (default = 0).
nlag number of lags to compute in the before and after phases of a contact. If lag = 0 then only contacts are used.
lag time (in seconds) for defining the lags in before and after periods of a contact.
gap time (in seconds) for excluding the lags in before and after periods of a contact.
phaid (optional) id(s) of the contact phase upon which to examine (default is all).

Details

This function is used following the conPhase function. One should choose how to define the contact point (i.e., the parameter contact) depending on the research question. In most typical cases (with regular interval tracking data) the lag time should be set to the tracking interval and the gap should be set to 1/2 the tracking interval.

Value

A dataframe that can be used to examine behaviour/context before, during, and after contact phases.

See Also

conPhase

Examples

## Not run:
data(does)
doecons <- conProcess(does,tc=15*60,dc=50)
doephas <- conPhase(doecons,pc=60*60)
cc <- conContext(var='dist',def='first',nlag=3,lag=30*60,gap=15*60)
conDisplacement

head(cc)

## End(Not run)

---

**Description**

Calculate the net-displacement (distance) of fixes before and after a contact from that contact point.

**Usage**

```
conDisplacement(ltraj, def = "all", idcol = "burst")
```

**Arguments**

- `ltraj` an object of the class `ltraj` which should be output from the function `conPhase`.
- `def` how to define the point-of-contact. The default is to define it as all fixes in a phase type = 'all', alternatively contacts can be defined as a single point along the phase defined as one of: 'first', 'last', 'minDist', 'minTime', which corresponds to the first fix in the contact phase, the last fix in the contact phase, the fix with the minimum time difference and the fix with the closest contact distance.
- `idcol` column id associated with IDs of individuals, default is the 'burst'

**Details**

This function is used to compute the net displacement away from contacts by an animal before and after a contact phase. Net displacement represents an important contextual variable, related to the mobility of the individual.

**Value**

An ltraj object with a new 'displacement' column in infolocs.

**See Also**

conPhase, conContext
Examples

```r
## Not run:
data(does)
doecons <- conProcess(does, tc=15*60, dc=50)
doecons <- conProcess(does, tc=15*60, dc=50)
disp_f <- conDisplacement(doecons, def='first')
disp_1 <- conDisplacement(doecons, def='last')

## End(Not run)
```

---

**conMatrix**

**Description**

Create a matrix that can be used for social network analysis.

**Usage**

`conMatrix(mtraj, idcol = "burst", output = "count")`

**Arguments**

- `mtraj`: an object of the class `ltraj` which is output from the function `conProcess`.
- `idcol`: column id associated with IDs of the individuals, default is the 'burst'
- `output`: ('count' or 'rate') whether to compute the counts of contacts in the contact matrix (default) or the contact-rate.

**Details**

This function is used to calculate the contact rates between individuals and output them in the form of a matrix. NOTE: This function is designed to be used only when a single ltraj object is input into the conProcess function. I.e., single-species contact networks.

**Value**

A matrix, with the contact rates between individuals.

**See Also**

`conProcess`, `conPairs`
Examples

```r
## Not run:
data(does)
doecons <- conProcess(does, tc=15*60, dc=50)
domat <- conMatrix(doecons)
domat_rate <- conMatrix(doecons, output='rate')

## End(Not run)
```

---

**conPairs**

Identify contact pairs

**Description**

Create a dataframe where each row represents a single contact pair.

**Usage**

```r
conPairs(ltraj)
```

**Arguments**

- `ltraj`: an object of the class `ltraj` which is output from the function `conProcess` or `conPhase`.

**Details**

This function is used to extract contact pairs following use of the `conProcess` or `conPhase` function. The returned dataframe has two new columns: `contact_orig_rowid` - the original row id of that particular fix, and `contact_pair_id` - a unique identifier to show which two fixes are represented by a pair of contacts. The number of unique pairs of contacts is then the highest number in this column, and will be equal to half the number of rows in the data frame.

**Value**

A data frame, where each row represents one of the two fixes in each unique contact pair.

**See Also**

`conProcess`, `conPhase`
conPhase

Process contact phases

Description
Computes phases where contacts occur based on a temporal tolerance.

Usage
conPhase(ltraj, pc = 0, idcol = "burst")

Arguments
- ltraj: an object of the class ltraj which is output from the function conProcess.
- pc: time (in seconds) to allow for which to combine contact events (see details).
- idcol: column used to identify individuals (default is the burst)

Details
This function is used following the conProcess function to arrange contacts into phases where continuous contact occurs (based on the user-defined time threshold pc). The idea is that we can consider a phase to be a continuous contact event (based on dc see conProcess) as long as the contact is only interrupted for no more than pc time units.

Value
An ltraj object with new column contact_pha.

See Also
conProcess, conSpatial, conTemporal, conSummary
### conProcess

**Description**

This function performs basic contact analysis between individuals in a group of tracked animals, or between two different groups of tracked animals.

**Usage**

```r
conProcess(mtraj1, mtraj2, dc = 0, tc = 0, idcol1 = "burst", idcol2)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtraj1</td>
<td>an object of the class ltraj which contains the time-stamped movement fixes of the first group of individuals. Each individual should be stored with a unique 'id'. (see ?as.ltraj)</td>
</tr>
<tr>
<td>mtraj2</td>
<td>(optional) same as mtraj1, but for the second group of individuals.</td>
</tr>
<tr>
<td>dc</td>
<td>distance tolerance limit (in appropriate units) for defining when two fixes are spatially together.</td>
</tr>
<tr>
<td>tc</td>
<td>time threshold for determining simultaneous fixes – see function: GetSimultaneous.</td>
</tr>
<tr>
<td>idcol1</td>
<td>column id associated with IDs of the first group of individuals, default is the 'burst'.</td>
</tr>
<tr>
<td>idcol2</td>
<td>(optional) column id associated with IDs of the second group of individuals.</td>
</tr>
</tbody>
</table>

**Details**

This function can be used to identify the nature of contacts in space and time between individuals in one or two groups.

**Value**

This function returns the object mtraj1 with three additional fields: contact - the number of contacts associated with each given fix. contact_id - the id(s) of the individual(s) associated with those contacts. contact_d - the distance (in the same units as mtraj1) at which the contacts occur. Note that if more than one contact occurs at a given time, the contact_id and contact_d fields will be a concatenated list of the contact IDs and distances.
conSpatial

Mapping wildlife contacts

Description

The function conSpatial is a simple function for mapping where wildlife contacts occur on the landscape with wildlife telemetry data.

Usage

conSpatial(ltraj, type = "point", def = "all")

Arguments

ltraj  
an object of the class ltraj which should be output from the function conPhase.

type  
one of (‘point’ - the default or ‘line’). Whether to generate contacts as points or phases as lines, points are the default, but lines can be useful for plotting and exploratory analysis. NOTE: if type = 'line' only those contact phases with at least two contact points are returned. So it is useful to use this in combination with contact points.

def  
if type = 'point' one of (‘all’, ‘phase’, ‘first’, ‘last’, ’minDist’, ’minTime’) which defines how contacts are to be mapped using all or part of a contact phase. (see Details)

Details

The function conSpatial can be used to map where contacts occur on the landscape, contacts being defined spatially based on a distance threshold dc and temporally based on the time threshold tc – see the function GetSimultaneous. The location of the contact can be calculated in a number of ways, and represented as points for each contact, or as line grouped by the contact phases. Which contacts to map can be defined in a number of ways using the def parameter:

i) def = ‘all’ (the default) all fixes where column contacts = 1 are returned in the sf object;

Examples

## Not run:
data(does)
doecons <- conProcess(does,tc=15*60,dc=50)

## End(Not run)

See Also

GetSimultaneous, dcPlot, conPhase, conSummary
defined;
iii) def = 'first' the first location fix of each phase is returned;
iv) def = 'last' the last location fix of each phase is returned;
v) def = 'minDist' the location fix of each phase which has the minimal contact distance is returned;
v) def = 'minTime' the location fix of each phase with the minimal time difference with contact fixes is returned;

Value

An sf object containing the locations/paths of the contacts. The time of the contact is stored in the attributes of the output object, along with the actual distance between fixes. The lines object contains attributes of the time of contact, and the min, max, and mean distance apart along a line segment.

See Also

conProcess, conPhase

Examples

```r
## Not run:
data(does)
doecons <- conProcess(does,tc=15*60,dc=50)
doephas <- conPhase(doecons,pc=60*60)
pts <- conSpatial(doephas)
plot(pts[['contact_pha']])
lns <- conSpatial(doephas,type='line')
plot(lns[['contact_pha']])
## End(Not run)
```

---

**conSummary**

*Summarize contacts and phases*

**Description**

Computes some basic summary statistics from a contact analysis.

**Usage**

```r
conSummary(ltraj)
```

**Arguments**

- **ltraj** an object of the class ltraj which should be output from the function conPhase.
Details

This function is used following the `conPhase` function. It computes the following summary statistics from the contact analysis: - total number of fixes in the dataset - total number of fixes deemed a contact - number of contact phases - longest phase duration - mean phase duration - median phase duration - no. of phase where the duration is only one fix (i.e., instantaneous contacts)

Value

A dataframe that can be used to summarize contact phases.

See Also

`conPhase`

Examples

```r
## Not run:
data(does)
doecons <- conProcess(does, tc=15*60, dc=50)
doephas <- conPhase(doecons, pc=60*60)
conSummary(doephas)
## End(Not run)
```

Description

Create a summary dataframe of the timing and and duration of contact phases.

Usage

```r
conTemporal(traj, units = "auto")
```

Arguments

- `traj`: an object of the class `ltraj` which is output from the function `conPhase`.
- `units`: units of duration e.g., 'mins' (see `difftime`).

Details

This function is used to calculate the start and end times of contact phases, and their duration following use of the `conPhase` function.

Value

A data frame, with the time and duration attributes associated with contact phases.
See Also
conPhase

Examples
## Not run:
data(does)
doecons <- conProcess(does, tc=15*60, dc=50)
doephases <- conPhase(doecons, pc=60*60)
conTemporal(doephases)
## End(Not run)

---

Cr  Movement Correlation Coefficient

Description
The function `Cr` computes the correlation statistic for movement data as presented in the paper by Shirabe (2006). The statistic is essentially a Pearson product-moment correlation statistic formulated for use with movement data.

Usage
```r
Cr(traj1, traj2, tc = 0)
```

Arguments
- `traj1`: an object of the class `ltraj` which contains the time-stamped movement fixes of the first object. Note this object must be a type II `ltraj` object. For more information on objects of this type see `help(ltraj)`.
- `traj2`: same as `traj1`.
- `tc`: time threshold for determining simultaneous fixes – see function: `GetSimultaneous`.

Details
The function `Cr` can be used to measure the level of dynamic interaction (termed correlation) between a pair of simultaneously moving objects. The statistic is sensitive to interaction in both movement direction (azimuth) and displacement, but is unable to disentangle the effects of these components.

Value
This function returns the Shirabe (2006) correlation statistic for two moving objects.
References

See Also
GetSimultaneous, DI

Examples
```
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes
Cr(deer37, deer38, tc = 7.5*60)
```

---

### Cs

**Coefficient of Sociality**

**Description**
The function Cs computes the coefficient of sociality between two moving objects following the methods outlined by Kenward et al. (1993). It also uses a signed Wilcoxon-rank test to test for significance.

**Usage**

```
Cs(traj1, traj2, tc = 0)
```

**Arguments**

- **traj1**: an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).
- **traj2**: same as traj1.
- **tc**: time threshold for determining simultaneous fixes – see function: GetSimultaneous.

**Details**
This function can be used to calculate the Kenward et al. (1993) coefficient of sociality (Cs) between two animals. The Cs statistic tests the observed mean distance between simultaneous fixes against that expected by the overall distribution of distances between all fixes.

\[
Cs = \frac{D_E - D_O}{D_O + D_E}
\]
Where $D_O$ is the mean observed distance between simultaneous fixes, and $D_E$ is the mean expected distance between all fixes. Kenward et al. (1993) propose $C_s$ as a useful metric for exploring attraction or avoidance behaviour. Values for $C_s$ closer to 1 indicate attraction, while values for $C_s$ closer to -1 indicate avoidance. Values of $C_s$ near 0 indicate that the two animals’ movements have no influence on one another.

Further, the difference between the observed and expected distances are compared using a paired signed-rank test (both one-sided tests, indicative of attraction or avoidance). See the function `GetSimultaneous` for details on how simultaneous fixes are determined from two trajectories.

**Value**

This function returns a list of objects representing the calculated values from the $C_s$ statistic and associated $p$-values from the signed rank test.

- **Do** – The mean distance of simultaneous fixes.
- **De** – The mean expected distance, from all fixes.
- **Cs** – The coefficient of sociality, see **Details**.
- **p.Attract** – One sided $p$-value from signed rank test, testing for attraction.
- **p.Avoid** – One sided $p$-value from signed rank test, testing for avoidance.

**References**


**See Also**

`GetSimultaneous`

**Examples**

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes
Cs(deer37, deer38, tc = 7.5*60)
Usage

dcPlot(mtraj1, mtraj2, tc = 0, idcol1 = "burst", idcol2, histplot = TRUE, dmax)

Arguments

mtraj1: an object of the class ltraj which contains the time-stamped movement fixes of the first group of individuals. Each individual should be stored with a unique 'id'. (see ?as.ltraj)

mtraj2: (optional) same as mtraj1, but for the second group of individuals.

tc: time threshold for determining simultaneous fixes – see function: GetSimultaneous.

idcol1: column id associated with IDs of the first group of individuals, default is the 'burst'.

idcol2: (optional) column id associated with IDs of the second group of individuals.

histplot: (logical) whether to output a histogram, along with a list of the natural breaks in the histogram (histplot = TRUE) or the dataframe of all paired distances used to construct the histogram (histplot=FALSE) to be used for further analysis.

histplot

dmax: (optional) distance value to 'cut-off' the distance histogram.

Details

The dcPlot function can be used to study the frequency distribution of pairwise distances between individual in a large telemetry dataset. It can be applied to a single group (if mtraj2 is ignored) or two-groups of individuals. The code attempts to find natural breaks (local minima) in the frequency histogram using an approach based on the peaks function attributed to B. Ripley (see https://stackoverflow.com/questions/6324354/add-a-curve-that-fits-the-peaks-from-a-plot-in-r). This tool is meant to be used for exploratory data analysis.

Value

If histplot = TRUE a list of the natural breaks (local minima) identified from the frequency histogram and a plot of the frequency histogram. If histplot = FALSE a dataframe containing all the pairwise and simultaneous distances between all individuals in the trajectory dataset.

See Also

GetSimultaneous, conProcess, Prox, Don, IAB

Examples

## Not run:
data(does)
dcPlot(does,tc=15*60,dmax=1000)

## End(Not run)
Description

GPS telemetry data for two male deer during a one-week period in March 2005. The two deer form a male bachelor group, making them an interesting case study for studying dynamic interaction patterns. The data are a subset of the data used as a case study in Long et al. (2014).

Format

An \texttt{ltraj} object with two bursts, representing the two different individual deer:

- Deer no. 37 containing 551 fixes.
- Deer no. 38 containing 567 fixes.

Details

The deer data are stored as a single \texttt{ltraj} object; two bursts contain the fixes for two individuals (deer37 and deer 38). GPS fixes were attempted at a regular sampling frequency of 15 minutes. For more information on these data how the deer data was collected or for citation please see the papers Webb et al. (2009, 2010).

References


Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
plot(deer37)
plot(deer38)
**DI**  
*Dynamic interaction index*

**Description**

The function DI measures dynamic interaction between two moving objects. It calculates the local level di statistic for movement displacement, direction, and overall. DI can compute time- and/or distance-based weighting schemes following Long and Nelson (2013).

**Usage**

```r
DI(traj1, traj2, tc = 0, local = FALSE, rand = 99, alpha = 1)
```

**Arguments**

- `traj1` an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II traj object. For more information on objects of this type see `help(ltraj)`.
- `traj2` same as `traj1`.
- `tc` time threshold for determining simultaneous fixes – see function: GetSimultaneous.
- `local` logical value indicating whether a dataframe (`local = TRUE`) containing the IAB index for each simultaneous fix should be returned (with a local permutation test), or (if `local = FALSE` - the default) the global index along with associated global permutation test.
- `rand` number of permutations to use in the local permutation test.
- `alpha` value for the $\alpha$ parameter in the formula for $di_d$ (default = 1).

**Details**

This function can be used for calculating the dynamic interaction (DI) statistic as described in Long and Nelson (2013). The DI statistic can be used to measure the local level of dynamic interaction between two moving objects. Specifically, it measures dynamic interaction in movement direction and displacement.

**Value**

If `local=FALSE` (the default) DI returns the numeric value of the DI index (along with $DI_{theta}$ and $DI_d$), and the associated p-value from a permutation test (see IAB). If `local=TRUE` DI returns a data frame that contains the localized di values (see Long and Nelson 2013). The columns for di, $di_theta$, and $di_d$ represent dynamic interaction overall, in direction (azimuth), and in displacement, respectively for each segment. A localized p-value for a one sided test for positive interaction (and z-score) is computed based on rand permutations of the segments. The pkey columns can be used to match the simultaneous segments to the original trajectory (see IAB).
does

References


See Also

GetSimultaneous, Cr, IAB

Examples

data(deer)
deer37 <- deer[1]  
deer38 <- deer[2]  
#tc = 7.5 minutes  
DI(deer37, deer38, tc = 7.5*60)  
df <- DI(deer37, deer38, tc = 7.5*60, local = TRUE)

---

does                GPS tracking data of female white-tailed deer

Description

GPS telemetry data for 8 does during month of May in 2011.

Format

An *ltraj* object with where bursts represent different individual deer.

Details

The doe data are stored as a single 'ltraj' object; each burst represents an individual.

Examples

data(does)  
plot(does)
Don

Doncaster’s measure of dynamic interaction

Description

The function Don measures the dynamic interaction between two moving objects following the methods outlined by Doncaster (1990).

Usage

Don(traj1, traj2, tc = 0, dc = 50, plot = TRUE)

Arguments

- traj1: an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).
- traj2: same as traj1.
- tc: time threshold for determining simultaneous fixes – see function: GetSimultaneous.
- dc: distance tolerance limit (in appropriate units) for defining when two fixes are spatially together.
- plot: logical, whether or not to plot the Doncaster plot. Default = TRUE.

Details

This function can be used to compute the Doncaster (1990) methods for measuring dynamic interaction between two objects. The Doncaster method tests the proportion of simultaneous fixes that are below dc against that which would be expected based on the distribution of distances between all fixes.

Value

This function first returns a plot, for distance values ranging from 0 to the maximum distance separating two fixes, of the observed proportion of simultaneous fixes below each distance value. The expected values based on all fixes are also included. Second, a list is returned that contains the contingency table of simultaneous fixes (paired) and non-paired fixes below and above dc, along with the associated p-value from the Chi-squared test.

- conTable – contingency table showing frequency of paired and non-paired fixes above and below dc.
- p.value – p-value from the Chi-squared test of conTable.

References

**FilterTraj**

Filter trajectory based on conditions

**Description**

The function `FilterTraj` is a function for extracting portions of a trajectory based on some filter criteria.

**Usage**

```r
FilterTraj(traj, type = "attribute", filter = NA)
```

**Arguments**

- `traj`: an object of the class `ltraj` which contains the time-stamped movement fixes of the object. Note this object must be a type II `ltraj` object. For more information on objects of this type see `help(ltraj)`.
- `type`: The type of filter to apply (one of 'attribute', 'temporal', 'tod').
- `filter`: The filter criteria (see details).

**Details**

The function `FilterTraj` can be used to extract a portion of a trajectory based on temporal criteria (such as time-of-day or a temporal window) or based on attributes of the trajectory. In the case of attribute filtering, the criteria can be any of the default attributes of an `ltraj` object, or based on additional attributes stored alongside the trajectory. The `filter` parameter is a flexible way to define the criteria for filtering. The `filter` object can be one of:
- POSIX class objects (list of length=2) for filtering based on some temporal window (see examples);
- character object (list of length=2) for filtering based on time-of-day (see examples);
- character object containing the attribute filter criteria as a logical expression (see examples and `?subset`).

**NOTE:** When using `FilterTraj` be very careful about using the output `ltraj` object in further analysis. When a new `ltraj` object is created, all the movement parameters (e.g., dist, dt) are re-calculated and thus not necessarily valid. Therefore, subsequent analysis should ideally focus only on the raw fix information (i.e., x, y, date).

---

**Examples**

```r
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes, dc = 50 meters
Don(deer37, deer38, tc = 7.5*60, dc = 50)
```
Value

A ltraj object with only those fixes satisfying the filter criteria.

See Also

GetTO

Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]

#Temporal Filter
t1 <- as.POSIXct(strptime('2005-03-09 00:00:00', format='%Y-%m-%d %H:%M:%S'))
t2 <- as.POSIXct(strptime('2005-03-11 00:00:00', format='%Y-%m-%d %H:%M:%S'))
twin <- c(t1, t2)
x <- FilterTraj(deer37, type='temporal', filter=twin)

#tod Filter
tod <- c('06:00:00', '10:00:00')
x <- FilterTraj(deer37, type='tod', filter=tod)

#attribute Filter
q <- 'dist > 100'
x <- FilterTraj(deer37, type='attribute', filter=q)

q <- 'dist > 100 & rel.angle < 1'
x <- FilterTraj(deer37, type='attribute', filter=q)

GetSimultaneous

Identify simultaneous fixes between trajectories

Description

The function GetSimultaneous identifies and extracts simultaneous fixes, within a given tolerance limit, between two movement datasets.

Usage

GetSimultaneous(traj1, traj2, tc = 0)
GetTO

Arguments

traj1 an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).

traj2 same as traj1.

tc time threshold for determining simultaneous fixes. For simplicity, tc is always taken in seconds.

Details

This function is used to determine the simultaneous fixes between two movement datasets facilitating further analysis.

Value

A single ltraj object containing two bursts, representing the two original ltraj objects, each containing only those fixes that are deemed simultaneous.

See Also

GetTO

Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes
trajs <- GetSimultaneous(deer37, deer38, tc = 7.5*60)
deer37 <- trajs[1]
deer38 <- trajs[2]

GetTO

Get period where two tracks overlap

Description

The function GetTemporalOverlap identifies and extracts parts of a trajectory that overlap in time with another trajectory.

Usage

GetTO(traj1, traj2, tc = 0)
Arguments

traj1  an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).

traj2  same as traj1.

tc  time threshold for considering if fixes are in the overlap period.

Details

This function is used to determine the fixes that overlap in time between two trajectories.

Value

A single ltraj object containing two bursts, representing the two original ltraj objects, each containing only those fixes that are overlap in time.

See Also

checkTO

Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
trajs <- GetTO(deer37, deer38)
deer37 <- trajs[1]
deer38 <- trajs[2]

HAI  Half-weight Association Index

Description

This function computes the Half-weight Association Index for examining the presence of dynamic interaction in wildlife telemetry studies. This implementation follows that outlined in the paper Atwood and Weeks (2003).

Usage

HAI(traj1, traj2, OZ, tc = 0, dc = 50)
**Arguments**

- **traj1**: an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).
- **traj2**: same as traj1.
- **OZ**: spatial polygon sf object associated with the home range (or some other form of) spatial overlap between traj1 and traj2. Required to be an sp polygon object.
- **tc**: time threshold for determining simultaneous fixes – see function: GetSimultaneous.
- **dc**: distance tolerance limit (in appropriate units) for defining when two fixes are spatially together.

**Details**

This function can be used to test for the presence of dynamic interaction within the shared area (often termed the overlap zone) of the two animals home ranges. Specifically, HAI is calculated in identical fashion to that for Ca, but considers only those fixes in the shared area. Typically, the overlap zone (OZ) is easily obtained by taking the spatial intersection of two polygon home ranges.

**Value**

This function returns the numeric value of the HAI statistic. Values near 1 indicate attraction within the shared home range area, while values near 0 indicate avoidance within this shared area.

**References**


**See Also**

GetSimultaneous, Ca

**Examples**

```r
## Not run:
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
library(adehabitatHR)
library(sp)
library(rgeos)
# use minimum convex polygon for demonstration...
hr37 <- mcp(SpatialPoints(Id(deer37)[,1:2]))
hr38 <- mcp(SpatialPoints(Id(deer38)[,1:2]))
OZ <- gIntersection(hr37,hr38)
#tc = 7.5 minutes, dc = 50 meters
```
The function `IAB` computes the IAB index following the methods described in the paper by Benhamou et al. (2014). It facilitates global analysis, with the significance testing procedure described in the paper, but also a local level output, to explore the IAB statistic through time.

**Usage**

```r
IAB(traj1, traj2, tc = 0, dc = 50, local = FALSE, rand = 99)
```

**Arguments**

- `traj1`: an object of the class `ltraj` which contains the time-stamped movement fixes of the first object. Note this object must be a type II `ltraj` object. For more information on objects of this type see `help(ltraj)`.
- `traj2`: same as `traj1`.
- `tc`: time threshold for determining simultaneous fixes – see function: `GetSimultaneous`.
- `dc`: critical distance where the IAB function will show maximum slope – see Benhamou et al. (2014) for more advice on selecting this parameter.
- `local`: logical value indicating whether a dataframe (`local = TRUE`) containing the IAB index for each simultaneous fix should be returned (with a local permutation test), or (if `local = FALSE` - the default) the global index along with associated global permutation test.
- `rand`: number of permutations to use in the local permutation test.

**Details**

The function `IAB` can be used to test for direct interaction in wildlife telemetry data and affords a novel significance testing procedure that takes into account the serially correlated structure of telemetry data. Specifically, it computes an index analogous to the Bhattacharyya coefficient between the potential influence domains of two animals. Like the other indices, IAB is dependent on the selection of an appropriate value for `dc` (which is termed $\Delta$ in the article). The dc parameter here is not a threshold distance, but rather the distance at which the function shows maximum slope (see Benhamou et al. 2014).

The significance testing procedure uses a wrapped shifting method in order to maintain the serially correlated structure of the data. At each shift, a sample value of IAB (termed MAB) is computed in order to generate a distribution of values to test against (for more information see Benhamou et al. 2014). Here a local version of this statistical testing procedure is implemented by taking `rand` samples of the $(n^2 - n)$ permutations of unpaired fixes. The p-values are computed following Benhamou et al. (2014), z-scores are calculated based on the mean and standard deviation of this hypothetical distribution.
Value

If local=FALSE (the default) IAB returns the numeric value of the IAB index and the associated p-values for one-sided tests for attraction or avoidance. If local=TRUE IAB returns a dataframe (containing the date/times of all simultaneous fixes (NOTE: times are associated with traj1), along with the distance between fixes at each time, and the IAB index value for each simultaneous fix. A localized p-value (.pa signifies the test for attraction and pb the test for avoidance) and z-score is computed based on rand permutations of the fixes. The pkey columns can be used to match the simultaneous fixes to the original trajectory.

References


See Also

GetSimultaneous, DI, Prox

Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes, dc = 50 meters
IAB(deer37, deer38, tc=7.5*60, dc=50)
df <- IAB(deer37, deer38, tc=7.5*60, dc=50, local=TRUE)

Lixn

*Minta’s Spatial-temporal interaction statistics*

Description

The function Lixn measures dynamic interaction between two animals following the methods outlined by Minta (1992).

Usage

Lixn(traj1, traj2, method = "spatial", tc = 0, hr1, hr2, OZ = NULL)

Arguments

traj1 an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).

traj2 same as traj1.
method

method for computing the marginal distribution from which expected values are computed. If method = "spatial", the marginal values are calculated based on areas of the shared and unshared portions of the home ranges. If method = "frequency", the marginal values are calculated based on the number of all fixes within the shared and unshared portions of the home ranges – see Details.

tc

time threshold for determining simultaneous fixes – see function: GetSimultaneous.

hr1

(– required if method = 'spatial') home range polygon associated with traj1. Must be an object that coerces to class SpatialPolygons*.

hr2

(– required if method = 'spatial') same as hr1, but for traj2.

OZ

(– required if method = 'frequency') shared area polygon associated with spatial use overlap between traj1 and traj2. Must be an object that coerces to class SpatialPolygons*.

Details

The function Lixn can be used to calculate the Minta (1992) measures of dynamic interaction between two animals. The Minta statistic tests how the two animals simultaneously utilize an area shared between the two individuals. Three coefficients are produced \( L_{AA} \), \( L_{BB} \), and \( L_{ixn} \). Each of these statistics are based on a contingency table that compares the observed frequency of those fixes that are simultaneous and within/outside the shared area to expectations based on area overlap proportions (if method="spatial") or expectations derived from all fixes (if method="frequency") – see Minta (1992) for more details. A Chi-squared statistic can then be used to examine the significance between the observed and expected use of the shared area.

Minta (1992) suggests the following interpretations of the coefficients. When \( L_{AA} \) is near 0, the first animal’s use of the shared area is random (or as expected). When \( L_{AA} > 0 \) it signifies spatial attraction to the shared area, or greater than expected use. When \( L_{AA} < 0 \) it signifies spatial avoidance of the shared area, or less than expected use. Interpretation of \( L_{BB} \) is the same as for \( L_{AA} \) with respect to the second animal. \( L_{ixn} \) tells us far more about the nature of the interaction between the two individuals. As \( L_{ixn} \) nears 0, both animals use the shared area randomly, with regards to the other animal. If \( L_{ixn} > 0 \) the animals use the shared area more simultaneously, whereas if \( L_{ixn} < 0 \) it is an indication of solitary use, or avoidance. This is why \( L_{ixn} \) is termed the temporal interaction coefficient. A Chi-squared test can be used to identify the significance of the \( L_{AA} \), \( L_{BB} \), and \( L_{ixn} \) values.

NOTES:
1. With modern telemetry datasets, where home ranges are readily estimated, choosing method = 'spatial' is most appropriate.
2. When the home ranges do not overlap the Lixn statistic is not defined and the function returns a string of NA's.
3. When one home range completely encloses another the Lixn statistic is not defined and the function returns a string of NA's and 'ContainsB' (or 'ContainsB') under the p.IXN result.
4. Further to points 2 and 3, the Lixn statistic is not appropriate in situations where the overlap area is either very large or very small relative to either home range (i.e., a situation with almost complete enclosure or virtually no overlap). Thus, it is advised that Lixn be used only in situations where there are suitable marginal areas for areaA, areaB, and areaAB – see Minta (1992).
Value

This function returns a list of objects representing the calculated values from the Minta statistic and associated $p$-values from the Chi-squared test.

- pTable – contingency table showing marginal probabilities of expected use based on the selection of the method parameter.
- nTable – contingency table showing observed frequency of use of the shared area based on simultaneous fixes.
- oTable – the odds for each cell in the contingency table.
- Laa – the calculated value of the $L_{AA}$ statistic
- p.AA – the associated $p$-value
- Lbb – the calculated value of the $L_{BB}$ statistic
- p.BB – the associated $p$-value
- Lixn – the calculated value of the $L_{ixn}$ statistic
- p.IXN – the associated $p$-value

References


See Also

GetSimultaneous

Examples

```r
## Not run:
data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
library(adehabitatHR)
library(sp)
# use minimum convex polygon for demonstration...
hr37 <- mcp(SpatialPoints(ld(deer37)[,1:2]))
hr38 <- mcp(SpatialPoints(ld(deer38)[,1:2]))
# tc = 7.5 minutes, dc = 50 meters
Lixn(deer37, deer38, method='spatial', tc=7.5*60, hr1=hr37, hr2=hr38)

## End(Not run)
```
Description

The function \texttt{ltraj2sf} is a simple function for converting \texttt{ltraj} to \texttt{sf} objects.

Usage

\texttt{ltraj2sf(traj, type = \texttt{"point"})}

Arguments

\begin{itemize}
\item \texttt{traj} \texttt{an object of the class \texttt{ltraj} which contains the time-stamped movement fixes of the object. For more information on objects of this type see help(ltraj).}
\item \texttt{type} \texttt{One of \texttt{"point"} (the default) or \texttt{"line"}.}
\end{itemize}

Details

The function \texttt{ltraj2sf} can be used to convert an \texttt{ltraj} object into an \texttt{sf} spatial object (either as points or lines).

Value

\texttt{A \texttt{sf} object either points or lines.}

See Also

\texttt{conSpatial}

Examples

\begin{verbatim}
data(deer)
#points
deer_pt <- ltraj2sf(deer)
plot(deer_pt[\'id\'])

#lines
deer_ln <- ltraj2sf(deer)
plot(deer_ln[\'id\'])
\end{verbatim}
 mockhunt

**Description**

Data output from the function `conContext` showing contact events between mock-hunters and male white-tailed deer during the hunting seasons of 2008 and 2009 in Oklahoma.

**Format**

An `ltraj` object with where bursts represent different individual deer.

**Details**

The mockhunt data are stored as a single `dataframe` object; the columns represent environmental attributes associated with contact events.

**References**


**Examples**

```r
data(mockhunt)
head(mockhunt)
```

---

**Prox**

**Proximity Index**

**Description**

The function `Prox` simply computes the proportion of (simultaneous) fixes that are proximal, based on some spatial threshold – `dc` (Bertrand et al. 1996). It also facilitates local-level proximity analysis.

**Usage**

```r
Prox(traj1, traj2, tc = 0, dc = 50, local = FALSE, GetSimultaneous = TRUE)
```
Arguments

traj1  an object of the class ltraj which contains the time-stamped movement fixes of the first object. Note this object must be a type II ltraj object. For more information on objects of this type see help(ltraj).

traj2  same as traj1.

tc  time threshold for determining simultaneous fixes – see function: GetSimultaneous.

dc  distance tolerance limit (in appropriate units) for defining when two fixes are spatially together.

local  logical value indicating whether or not a dataframe, containing the distance between each simultaneous fix, should be returned.

GetSimultaneous  logical value indicating whether proximity analysis is based on simultaneous fixes (if TRUE the default) – see function GetSimultaneous or (if FALSE) a one-way mapping from traj1 to traj2 is used.

Details

The function Prox can be used to test for the presence of attraction (via proximity) in wildlife telemetry data. Prox is simply the proportion of simultaneous fixes within the threshold distance – dc. The local output (dataframe) can be useful for examining variation in proximity through time.

Value

If local=FALSE (the default) Prox returns the numeric value of the Prox index. If local=TRUE Prox returns a dataframe containing the date/times of all simultaneous fixes from traj1, and in the case of GetSimultaneous = FALSE the time of the fixes that were deemed simultaneous in traj2. If GetSimultaneous = TRUE (the default) the Prox considers only the simultaneous fixes, as defined in GetSimultaneous. If FALSE Prox considers all the fixes in traj1 relative to traj2. The latter functionality is useful when the time between fixes for one trajectory (traj1) is much shorter than the second trajectory.

References


See Also

GetSimultaneous, contacts

Examples

data(deer)
deer37 <- deer[1]
deer38 <- deer[2]
#tc = 7.5 minutes, dc = 50 meters
Prox(deer37, deer38, tc=7.5*60, dc=50)
df <- Prox(deer37, deer38, tc=7.5*60, dc=50, local=TRUE)
Index

* contacts
  conContext, 5
  conDisplacement, 7
  conMatrix, 8
  conPairs, 9
  conPhase, 10
  conProcess, 11
  conSpatial, 12
  conSummary, 13
  conTemporal, 14

* datasets
  deer, 19
  does, 21
  mockhunt, 33

* indices
  Ca, 3
  Cr, 15
  Cs, 16
  DI, 20
  Don, 22
  HAI, 26
  IAB, 28
  Lixn, 29
  Prox, 33

* processing
  checkTO, 4
  FilterTraj, 23
  GetSimultaneous, 24
  GetTO, 25
  ltraj2sf, 32

Ca, 3
checkTO, 4
conContext, 5
conDisplacement, 7
conMatrix, 8
conPairs, 9
conPhase, 10
conProcess, 11
conSpatial, 12
conSummary, 13
conTemporal, 14
Cr, 15
Cs, 16
dcPlot, 17
deer, 19
DI, 20
does, 21
Don, 22
FilterTraj, 23
GetSimultaneous, 24
GetTO, 25
HAI, 26
IAB, 28
Lixn, 29
ltraj2sf, 32
mockhunt, 33
Prox, 33
wildlifeDI-package, 2

36