Package 'disperseR'

November 28, 2026

Title Classification of Animal Movements

Version 0.1.0

Description Tools to segment and classify animal movement trajectories, with a focus on long-range movements. Tracks can be regularised with continuous-time movement models from 'ctmm' and classified into behavioural states with hidden Markov models from 'momentuHMM'. Area-restricted phases (ARP) are then identified with density-based spatial clustering ('dbscan'), linked through time, and the movements between them are classified according to the spatial network of previously visited areas (unanchored movement, loops, network expansion and internal transit).

License MIT + file LICENSE

URL https://github.com/cherraiz/disperseR

BugReports https://github.com/cherraiz/disperseR/issues

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

Depends R (>= 3.5)

Imports dplyr (>= 1.1.0), tidyr, sf, ctmm, momentuHMM, dbscan, purrr, igraph, tibble, ggplot2 (>= 3.5.0), rlang

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Cesar Herraiz [aut, cre] (ORCID: <https://orcid.org/0000-0002-4770-2013>)

Maintainer Cesar Herraiz <c.herraizfer@gmail.com>

Repository CRAN

Date/Publication 2026-11-28 14:30:08 UTC

Contents

class.mov . . . 2
divide.track.by.gap . . . 4
draw.linked.clusters . . . 4
draw.step.densities . . . 5
draw.timeline . . . 6
draw.track . . . 7
fitHMM.wrap . . . 9
get.arp . . . 10
get.epsilon . . . 12
link.arp . . . 13
phase.summary . . . 14
print.disperseR . . . 15
resample.ctmm . . . 16
sim_states . . . 17
sim_tracks . . . 18
transfer.disperseR . . . 19

Index **21**

`class.mov` *Classify movement phases*

Description

Classifies movement trajectories into distinct functional phases (i.e., ARP, Type 0, Type 1, Type 2, Type 3) based on Area-Restricted Phase (ARP) patches, spatial links, and site fidelity, based on network complexity.

Usage

```
class.mov(
  data,
  id.col = "ID",
  time.col = "time",
  establish = FALSE,
  establish.days = NULL,
  arp.links = "arp"
)
```

Arguments

`data`	An object of class `disperseR` containing movement data and linked ARP clusters.
`id.col`	Character. Name of the column containing individual identifiers. Default is "ID".

`time.col` Character. Name of the column containing timestamp or date-time values. Default is `"time"`.

`establish` Logical. If `TRUE`, applies site fidelity corrections requiring a minimum duration at a ARP site to consider it established. Default is `FALSE`.

`establish.days` Numeric. Minimum duration threshold in days for an ARP cluster to be considered an established site. Required if `establish = TRUE`. The duration of a stay goes from the first to the last location of consecutive ARP blocks joined by `link.arp()`, including the linking locations between them, so a linked ARP made of several short visits counts as one stay.

`arp.links` Character. Method for evaluating links between ARPs. Either `"arp"` or `"mcp"` (Minimum Convex Polygon). Default is `"arp"`.

- `"arp"`: Classifies locations sequence of ARP cluster links as ARPs.
- `"mcp"`: Calculates a Minimum Convex Polygon (MCP) around the points of the preceding ARP cluster. It evaluates if the sequence of ARP cluster links geometrically falls within this polygon to refine the classification. If so, they are considered ARPs. Otherwise they are considered Type 1. Requires `x` and `y` coordinate columns.

Details

The movement phases follow a 0-3 hierarchy based on spatial network memory:

- **Type 0:** Unanchored movement. Occurs before the first ARP is established.
- **Type 1:** Loop / Local fidelity. Origin and destination ARPs are identical.
- **Type 2:** Network expansion. Movement from a known ARP to a newly discovered ARP.
- **Type 3:** Internal transit. Movement between two different, previously known ARPs.

Value

An object of class `disperseR` with an updated dataset containing the classified `phase` and `cluster_established` columns.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
linked <- link.arp(arp, thres = 24 * 3600)

mov <- class.mov(linked)
table(mov$data$ID, mov$data$phase)

# Require at least 3 days in an ARP for it to count as established
mov_est <- class.mov(linked, establish = TRUE, establish.days = 3)
table(mov_est$data$ID, mov_est$data$phase)
```

divide.track.by.gap *Split trajectory by time gap*

Description

Splits individual tracking trajectories into multiple segments if the time difference between consecutive locations exceeds a specified threshold.

Usage

```
divide.track.by.gap(data, ids, time.col, gap.days)
```

Arguments

data	A data.frame containing the tracking data.
ids	Character. The name of the column containing individual identifiers.
time.col	Character. The name of the column containing timestamps (must be POSIXct or Date).
gap.days	Numeric or named list. The maximum allowed gap in days before splitting the track. Can be a single numeric value for all individuals, or a named list specifying different thresholds per individual.

Value

A data.frame identical to the input but with an additional column id_split representing the new segmented identifiers.

Examples

```
# Split tracks wherever two consecutive fixes are more than 3 hours apart
split_trk <- divide.track.by.gap(sim_tracks, ids = "ID",
                                 time.col = "timestamp", gap.days = 3 / 24)
table(split_trk$id_split)
```

draw.linked.clusters *Plot linked Area-Restricted Phase (ARP) clusters*

Description

Generates a spatial plot of Area-Restricted Phase (ARP) clusters and the movement links between them. It visualizes ARPs as convex hulls, highlights ARP points, and draws the network edges connecting them.

Usage

```
draw.linked.clusters(data, ids = NULL, grid_spacing = NULL, id.col = "ID")
```

Arguments

data	An object of class disperseR containing movement data with ARP classifications.
ids	Character vector. A selection of individual IDs to plot. If NULL (default), plots all individuals.
grid_spacing	Numeric. Spacing for background reference grid lines. If NULL (default), no grid lines are drawn.
id.col	Character. Name of the column containing individual identifiers. Default is "ID".

Value

A ggplot object showing the spatial distribution and network of ARPs.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
linked <- link.arp(arp, thres = 24 * 3600)
draw.linked.clusters(linked)
draw.linked.clusters(linked, ids = "M01", grid_spacing = 5000)
```

---

draw.step.densities *Plot Step Length Densities by Behavioral State*

Description

Visualizes the density distribution of step lengths for different behavioral states using ggplot2. It can plot the distributions for a single individual or for the entire dataset, and optionally overlay a calculated threshold.

Usage

```
draw.step.densities(
  data,
  ind = NULL,
  ids = NULL,
  state.col = "state",
  step.col = "step",
  threshold = NULL,
  ...
)
```

Arguments

data	A data.frame or an object of class disperseR containing movement data.
ind	Character or Numeric. The specific individual ID to plot. If NULL, densities for all individuals are plotted combined. Default is NULL.
ids	Character. The name of the column containing individual identifiers. Required if ind is provided. Default is NULL.
state.col	Character. The name of the column containing the behavioral states. Default is "state".
step.col	Character. The name of the column containing the step lengths. Default is "step".
threshold	Numeric. An optional threshold value (e.g., ϵ threshold) to plot as a vertical dashed line. Default is NULL.
...	Additional arguments passed to or from other methods.

Value

A ggplot object showing the density curves.

Examples

```
eps <- get.epsilon(sim_states)
draw.step.densities(sim_states, threshold = eps)
draw.step.densities(sim_states, ind = "M01", ids = "ID")
```

---

draw.timeline *Plot the timeline of movement phases*

Description

Draws, for each individual, the sequence of movement phases (or any other categorical variable) along time. Each location is drawn as a bar that lasts until the next location, so monitoring gaps appear as empty spaces.

Usage

```
draw.timeline(
  data,
  phase.col = "phase",
  id.col = "ID",
  time.col = "time",
  max.gap = 3,
  palette = NULL
)
```

Arguments

data	A data.frame or an object of class disperseR, usually the output of class.mov.
phase.col	Character. Name of the column to display. Default is "phase".
id.col	Character. Name of the column containing individual identifiers. Default is "ID".
time.col	Character. Name of the column containing timestamps (POSIXct). Default is "time".
max.gap	Numeric. Intervals between consecutive locations longer than max.gap times the median interval of the individual are left empty. Default is 3.
palette	Optional named character vector of colours. If NULL, a default palette is used for the phase column and the default ggplot2 palette otherwise.

Value

A ggplot object.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
mov <- class.mov(link.arp(arp, thres = 24 * 3600))
draw.timeline(mov)

# Behavioural states instead of phases
draw.timeline(sim_states, phase.col = "state")
```

---

draw.track *Plot movement tracks*

Description

Draws the trajectory of each individual, with locations optionally coloured by any column of the data, such as the movement phase from class.mov, the behavioural state from fitHMM.wrap or the cluster_id from get.arp.

Usage

```
draw.track(
  data,
  colour.col = NULL,
  ids = NULL,
  id.col = "ID",
  time.col = "time",
  coord.names = c("x", "y"),
  colour.path = FALSE,
  km = FALSE,
```

```
  palette = NULL,
  point.size = 0.6
)
```

Arguments

data	A data.frame or an object of class disperseR.
colour.col	Character. Optional name of the column used to colour the locations. Default is NULL (no colour).
ids	Character vector. Optional selection of individuals to plot. Default is NULL (all individuals).
id.col	Character. Name of the column containing individual identifiers. Default is "ID".
time.col	Character. Name of the column containing timestamps, used to order the locations. Default is "time".
coord.names	Character vector of length 2 with the names of the X and Y coordinate columns. Default is c("x", "y").
colour.path	Logical. If TRUE, the path between locations is also coloured by colour.col; otherwise it is drawn in grey. Default is FALSE.
km	Logical. If TRUE, projected coordinates in metres are shown in kilometres. Default is FALSE.
palette	Optional named character vector of colours for the values of colour.col. If NULL, a default palette is used for the phase column and the default ggplot2 palette otherwise.
point.size	Numeric. Size of the location points. Default is 0.6.

Details

With one individual the plot keeps the true aspect ratio (coord_equal). With several individuals each one is drawn in its own panel with free scales.

Value

A ggplot object.

Examples

```
draw.track(sim_tracks, time.col = "timestamp", km = TRUE)

eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
mov <- class.mov(link.arp(arp, thres = 24 * 3600))
draw.track(mov, colour.col = "phase", km = TRUE)
draw.track(mov, colour.col = "phase", ids = "M01", colour.path = TRUE, km = TRUE)
```

`fitHMM.wrap` *Fit Hidden Markov Models for Movement Data Classification*

Description

Wrapper around `momentuHMM` to fit 2-state or 3-state Hidden Markov Models (HMM) to animal movement trajectories. Supports global, individual-level, or group-level model fitting.

Usage

```
fitHMM.wrap(
  data,
  ids = NULL,
  coordNames = c("x", "y"),
  coordtype = "UTM",
  mean.step.1 = NULL,
  sd.step.1 = NULL,
  mean.step.2 = NULL,
  sd.step.2 = NULL,
  ang.con.1 = NULL,
  ang.con.2 = NULL,
  stationary = FALSE,
  mode = "global",
  group.col = NULL,
  ...
)
```

Arguments

`data`	A data.frame containing spatial tracking trajectories.
`ids`	Character. Column name containing individual identifiers. Default is NULL.
`coordNames`	Character vector of length 2 specifying the X and Y coordinate column names. Default is `c("x", "y")`.
`coordtype`	Character. Coordinate type passed to `momentuHMM::prepData` (e.g., "UTM" or "LL"). Default is "UTM".
`mean.step.1`	Numeric. Initial mean step length for state 1 (Transit).
`sd.step.1`	Numeric. Initial standard deviation of step length for state 1 (Transit).
`mean.step.2`	Numeric. Initial mean step length for state 2 (Encamped).
`sd.step.2`	Numeric. Initial standard deviation of step length for state 2 (Encamped).
`ang.con.1`	Numeric. Initial concentration parameter for turning angle in state 1 (Transit).
`ang.con.2`	Numeric. Initial concentration parameter for turning angle in state 2 (Encamped).
`stationary`	Logical. If `TRUE`, fits a 3-state model (including a stationary state) and re-maps state 3 to state 2 post-estimation. Default is `FALSE`.

mode	Character. Fitting mode: "individual" (one model per ID), "groups" (one model per defined group), or "global" (single model for all data). Default is "global".
group.col	Character. Column name identifying groups when mode = "groups". Default is NULL.
...	Additional arguments passed to fitHMM.

Details

Step lengths are modeled using a gamma distribution and turning angles using a wrapped Cauchy distribution. In 2-state models, state 1 corresponds to Transit and state 2 to Encamped. Although the 2-state model generally performs well, in datasets lacking clear transit movements and containing abundant stationary periods, movement classification may be biased towards identifying stationary movements as Encamped and all other movements as Transit. In such cases, a 3-state model including a stationary state can improve state discrimination. The stationary state is initialized with a step-length prior equal to one tenth of the Encamped state (mean.step.2) prior from the 2-state model.

Value

An object of class disperseR containing:

data	A data.frame with movement parameters and Viterbi-decoded state assignments.
models	A list of fitted momentuHMM model objects.

Examples

```
trk <- sim_tracks[sim_tracks$ID == "F02", c("ID", "x", "y")]
hmm <- fitHMM.wrap(trk, ids = "ID",
                   mean.step.1 = 600, sd.step.1 = 400,
                   mean.step.2 = 80, sd.step.2 = 50,
                   ang.con.1 = 0.7, ang.con.2 = 0.1)
table(hmm$data$state)
```

---

get.arp *Identify Area-Restricted Phase (ARP) clusters*

Description

Identifies spatial Area-Restricted Phase (ARP) clusters using DBSCAN based on spatial proximity, duration, and state classification proportions.

Usage

```
get.arp(
  data,
  ids = "ID",
  epsilon,
  minPts = 3,
  time.col = "time",
  state.col = "state",
  pct.state1 = NULL,
  delta.days = NULL,
  group.col = NULL
)
```

Arguments

data	A data.frame or an object of class disperseR.
ids	Character. Name of the column containing individual identifiers. Default is "ID".
epsilon	Numeric or named list. Spatial distance threshold for DBSCAN. Can be a single value, or a named list per individual or group.
minPts	Integer. Minimum number of points required to form a cluster in DBSCAN. Default is 3.
time.col	Character. Name of the column containing timestamps (must be POSIXct). Default is "time".
state.col	Character. Name of the column containing behavioral or movement state classifications. Default is "state".
pct.state1	Numeric. Maximum proportion threshold of state 1 points allowed in a cluster. If not provided (NULL), this filter is not applied and a warning is triggered.
delta.days	Numeric. Minimum duration threshold in days for a cluster to be classified as a ARP area. Duration is computed from the first to the last location assigned to the cluster, regardless of intermediate locations. If not provided (NULL), this filter is not applied and a warning is triggered.
group.col	Character. Optional column name used for grouping when epsilon is provided as a named list.

Value

An object of class disperseR containing the updated dataset with cluster assignments, a list of cluster summaries, and list of filtered ARP clusters.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
arp$arp_clusters
```

`get.epsilon` *Calculate ϵ threshold for spatial clustering*

Description

Computes the threshold ϵ (epsilon) based on step lengths. The threshold can be calculated using density curve intersections (requires a behavioral state column with exactly two states), or simply by calculating the overall mean or median of the step lengths. The calculation can be applied globally, per individual, or per group.

Usage

```
get.epsilon(
  data,
  state.col = "state",
  step.col = "step",
  mode = "global",
  method = "density",
  ids = NULL,
  group.col = NULL
)
```

Arguments

`data`	A data.frame or an object of class `disperseR` containing movement data.
`state.col`	Character. The name of the column containing the behavioral states. Only used and required if `method = "density"`. Must contain exactly two unique states for the evaluated subset. Default is `"state"`.
`step.col`	Character. The name of the column containing the step lengths. Default is `"step"`.
`mode`	Character. The level at which to compute the threshold: `"global"`, `"individual"`, or `"groups"`. Default is `"global"`.
`method`	Character. The method used to calculate the threshold: `"density"` (intersection of density curves for two states), `"mean"`, or `"median"`. Default is `"density"`.
`ids`	Character. The name of the column containing individual identifiers. Required if `mode = "individual"`.
`group.col`	Character. The name of the column containing group identifiers. Required if `mode = "groups"`.

Value

A numeric value representing the threshold if `mode = "global"`, or a named list of numeric thresholds if `mode = "individual"` or `mode = "groups"`.

Examples

```
# Intersection of the step-length densities of both states
get.epsilon(sim_states)

# One threshold per individual, using the median step length
get.epsilon(sim_states, mode = "individual", ids = "ID", method = "median")
```

`link.arp` *Link Area-Restricted Phase (ARP) Clusters*

Description

Merges temporally close Area-Restricted Phase (ARP) clusters and movement gaps based on a specified temporal threshold using graph components from `igraph`.

Usage

```
link.arp(data, thres = 0, time.col = "time", id.col = "ID")
```

Arguments

`data`	An object of class `disperseR` containing movement data and ARP cluster definitions.
`thres`	Numeric. Temporal gap threshold (in seconds) below which adjacent ARP clusters will be linked. Default is 0.
`time.col`	Character. Name of the column containing timestamp or date-time values. Default is `"time"`.
`id.col`	Character. Name of the column containing individual identifiers. Default is `"ID"`.

Value

An updated object of class `disperseR` containing linked cluster IDs and associated graph network structures under `cluster_graphs`.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
# Link ARP clusters separated by less than 24 hours
linked <- link.arp(arp, thres = 24 * 3600)
table(linked$data$ID, linked$data$linked_clusters)
```

`phase.summary` *Summarise movement phases*

Description

Builds a table with one row per movement phase of each individual, as classified by `class.mov`: its start, end, duration, number of locations, distance travelled, net displacement and, for movements, the ARPs where they start and end.

Usage

```
phase.summary(
  data,
  id.col = "ID",
  time.col = "time",
  phase.col = "phase",
  coord.names = c("x", "y")
)
```

Arguments

`data`	An object of class `disperseR` returned by `class.mov`, or a data.frame with the same columns.
`id.col`	Character. Name of the column containing individual identifiers. Default is "ID".
`time.col`	Character. Name of the column containing timestamps (POSIXct). Default is "time".
`phase.col`	Character. Name of the column containing the movement phases. Default is "phase".
`coord.names`	Character vector of length 2 with the names of the X and Y coordinate columns (projected, in metres). Default is `c("x", "y")`.

Details

Consecutive locations with the same phase form one row. Consecutive ARP locations belonging to different linked clusters (`linked_clusters`) form separate rows. `distance_km` is the sum of the distances between consecutive locations of the phase, and `displacement_km` the straight-line distance between its first and last location. `from` and `to` give the ARP (`linked_clusters`) before and after each phase.

Value

A data.frame with columns `ID` (named as `id.col`), `n_phase`, `phase`, `arp`, `from`, `to`, `start`, `end`, `duration_days`, `n_locations`, `distance_km` and `displacement_km`.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
mov <- class.mov(link.arp(arp, thres = 24 * 3600))
ph <- phase.summary(mov)
head(ph)

# Total distance travelled outside ARPs per individual
tapply(ph$distance_km[ph$phase != "ARP"], ph$ID[ph$phase != "ARP"], sum)
```

`print.disperseR` *Print a disperseR object*

Description

Prints a short summary of a `disperseR` object: number of individuals and locations, the steps of the workflow already applied, and the number of locations in each movement phase when `class.mov` has been run.

Usage

```
## S3 method for class 'disperseR'
print(x, id.col = "ID", ...)
```

Arguments

`x`	An object of class `disperseR`.
`id.col`	Character. Name of the column containing individual identifiers. Default is "ID".
`...`	Not used.

Value

`x`, invisibly.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
arp
class.mov(link.arp(arp, thres = 24 * 3600))
```

`resample.ctmm` *Resample tracking data using Continuous-Time Movement Models (CTMM)*

Description

Fits continuous-time movement models to individual tracking data and predicts locations at regular time intervals. It handles coordinate reprojection, model selection, and retains static individual traits.

Usage

```
resample.ctmm(
  dat,
  tint,
  ids = "ID",
  timestamp = "timestamp",
  x = "x",
  y = "y",
  proj.in,
  proj.out = proj.in,
  static.cols = NULL,
  verbose = TRUE
)
```

Arguments

`dat`	A data.frame containing the tracking data.
`tint`	Numeric. The desired time interval for resampling, in seconds.
`ids`	Character. The name of the column containing individual identifiers. Default is "ID".
`timestamp`	Character. The name of the column containing timestamps. Default is "timestamp".
`x`	Character. The name of the column containing the X coordinate or Longitude. Default is "x".
`y`	Character. The name of the column containing the Y coordinate or Latitude. Default is "y".
`proj.in`	An object of class `crs` (from `sf`), an EPSG code, or a PROJ4 string representing the input coordinate reference system.
`proj.out`	An object of class `crs` (from `sf`), an EPSG code, or a PROJ4 string representing the output coordinate reference system. Must be in meters. Defaults to `proj.in`.
`static.cols`	Character vector. Optional names of columns containing static individual information (e.g., sex, species) to append to the resampled dataset. Default is NULL.
`verbose`	Logical. If TRUE, prints progress and model summaries to the console. Default is TRUE.

Value

A data.frame with the resampled trajectories, including predicted X and Y coordinates, time in POSIXct format, time in numeric seconds, and any specified static columns.

Examples

```
# Regularise the first 150 fixes of one individual to hourly locations
trk <- sim_tracks[sim_tracks$ID == "F02", ][1:150, ]
reg <- resample.ctmm(trk, tint = 3600, ids = "ID", timestamp = "timestamp",
                     proj.in = 25830, static.cols = "sex", verbose = FALSE)
head(reg)
```

---

sim_states *Simulated dispersal tracks with behavioural states*

Description

`sim_tracks` split at monitoring gaps longer than one day with `divide.track.by.gap`, regularised to hourly locations with `resample.ctmm`, and classified by segment with a hidden Markov model using `fitHMM.wrap` with `stationary = TRUE` (the stationary state, mostly daytime rest, is merged into the encamped state).

Usage

```
sim_states
```

Format

A data.frame with 10 columns:

ID Individual identifier.

id_split Track segment identifier (individual and segment number).

time Location time (POSIXct, UTC).

time_s Location time in seconds since 1970-01-01.

x, y Easting and northing (m, EPSG:25830).

step Step length (m).

angle Turning angle (radians).

state Viterbi-decoded state (1 = Transit, 2 = Encamped).

sex Static individual attribute.

Source

Simulated; see `data-raw/simulate_tracks.R` in the package sources.

`sim_tracks` *Simulated dispersal tracks*

Description

Simulated GPS tracks of five dispersing individuals with hourly fixes, timing jitter and about 10\ resting during the day, also while dispersing); directed movements are biased correlated random walks, and some of them include short stopovers (0.5-1.5 days). Coordinates are in ETRS89 / UTM zone 30N (EPSG:25830), in metres.

- `"M01"`: settles in an area made of two nearby patches, shuttling between them, moves to a second area with a stopover on the way, makes a looping excursion and returns to the first area (with another stopover).
- `"F02"`: settles in one area, makes a 2.5-day stay and settles in two more areas, with a stopover before the last one.
- `"M03"`: settles in one area and makes a long transit with a monitoring gap of about two days.
- `"F04"`: home range made of two nearby patches, with two looping excursions.
- `"F05"`: explores with stopovers, settles briefly, departs again and the tracking ends while it is still moving.

Usage

```
sim_tracks
```

Format

A data.frame with 6 columns:

ID Individual identifier.

timestamp Fix time (POSIXct, UTC).

x, y Easting and northing (m).

sex Static individual attribute.

behaviour Simulated behaviour: `"stay"` (area-restricted stay, including trips between nearby patches), `"stopover"` or `"travel"` (directed movement, including its daytime rests).

Source

Simulated; see `data-raw/simulate_tracks.R` in the package sources.

See Also

`sim_states`

`transfer.disperseR` *Transfer classification data from disperseR object to a base dataset*

Description

Merges behavioral or movement phase classifications generated in a disperseR object back into an external base data.frame using temporal or spatial matching methods.

Usage

```
transfer.disperseR(
  base.df,
  disperseR.obj,
  id.col = "name",
  time.col = "timestamp",
  method = "temporal",
  coord.names = c("x", "y"),
  max.diff = NULL
)
```

Arguments

`base.df`	A data.frame containing the target dataset where variables will be transferred.
`disperseR.obj`	An object of class `disperseR` containing the analyzed data.
`id.col`	Character. Column name in `base.df` representing individual identifiers.
`time.col`	Character. Column name in `base.df` representing timestamps.
`method`	Character. The matching method to use for transferring data. Choose one of: • `"temporal"`: Matches each row to the closest point in time. • `"spatial"`: Matches each row to the closest point in space using minimum Euclidean distance. • `"forward"`: Matches each row to the nearest future location in the disperseR object. • `"backward"`: Matches each row to the nearest past location in the disperseR object]. • `"movement"`: Uses temporal matching but applies a specific criteria for phase transitions. If the target point falls between two identical phases, it assigns that phase. If it falls exactly between a ARP (Area-Restricted Phase) and a movement phase, it prioritizes assigning the movement phase. • `"arp"`: Uses temporal matching but applies a specific criteria for phase transitions. If the target point falls between two identical phases, it assigns that phase. If it falls exactly between a ARP (Area-Restricted Phase) and a movement phase, it prioritizes assigning the ARP phase.
`coord.names`	Character vector of length 2. Names of the X and Y coordinate columns in `base.df`, required if `method = "spatial"`.
`max.diff`	Numeric. Optional. Maximum allowed time difference (in seconds) for a match to be considered valid.

Value

A data.frame based on `base.df` with appended classification columns from the `disperseR` object.

Examples

```
eps <- get.epsilon(sim_states)
arp <- get.arp(sim_states, epsilon = eps, delta.days = 2, pct.state1 = 0.2)
mov <- class.mov(link.arp(arp, thres = 24 * 3600))

# Transfer phases back to the original (irregular) fixes
out <- transfer.disperseR(sim_tracks, mov, id.col = "ID",
                          time.col = "timestamp", max.diff = 3600)
table(out$ID, out$phase)
```

Index

∗ **datasets**
sim_states, 17
sim_tracks, 18

class.mov, 2, *7*, *14, 15*

divide.track.by.gap, 4, *17*
draw.linked.clusters, 4
draw.step.densities, 5
draw.timeline, 6
draw.track, 7

fitHMM.wrap, *7*, 9, *17*

get.arp, *7*, 10
get.epsilon, 12

link.arp, *3*, 13

phase.summary, 14
print.disperseR, 15

resample.ctmm, 16, *17*

sim_states, 17, *18*
sim_tracks, *17*, 18

transfer.disperseR, 19