Package 'recurse'

July 23, 2025

Type Package

Title Computes Revisitation Metrics for Trajectory Data

Version 1.4.0

Date 2024-07-17

Author Chloe Bracis [aut, cre]

Maintainer Chloe Bracis <cbracis@uw.edu>

Description Computes revisitation metrics for trajectory data, such as the number of revisitations for each location as well as the time spent for that visit and the time since the previous visit. Also includes functions to plot data.

License MIT + file LICENSE

Imports Rcpp (>= 0.12.7)

LinkingTo Rcpp

Suggests testthat, circular, prevR, scales, fields, methods, move, move2, knitr, rmarkdown, sf

LazyData true

RoxygenNote 7.3.1

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-07-18 10:10:02 UTC

Contents

`.calculateCrossingPercentageCmplx` *Calculates percentage of trajectory segment within circle*

Description

Calculates the percentage of a segment that lies within a circle for a point A inside the circle and point B outside the circle for a circle with center C and radius R.

Usage

```
.calculateCrossingPercentageCmplx(Cz, Az, Bz, R)
```

Arguments

`Cz`	circle center
`Az`	point 1
`Bz`	point 2
`R`	radius

`calculateIntervalResidenceTime` *Calculates residence time within user-specified breaks*

Description

Using the results from `getRecursions` or `getRecursionsAtLocations`, calculates the residence time during user-specified intervals (rather than the entire trajectory period) in the radius around each location.

Usage

```
calculateIntervalResidenceTime(x, breaks, labels = NULL)
```

Arguments

x	recurse object returned from call to getRecursions or getRecursionsAtLocations with verbose = TRUE
breaks	vector of POSIX datetimes describing the interval boundaries
labels	(optional) vector or names for the intervals

Details

When recursions are calculated, the residence time in the radius around each location is also calculated. This method allows the user to post-process the results from calculating recursions to calculate residence time over user-specified intervals, rather than the entire trajectory. This allows the calculation of residence time on biologically relevant scales, such as seasons, and in cases where large gaps between visits (e.g., a seasonal migrant) may make splitting up the residence time preferable.

Note that care should be taken to use the same time zone when specifying the break points as used in the datetime for the movement trajectory.

Value

A matrix of residence times where the columns are the coordinate indices of the locations (either movement trajectory locations or user-specified locations) and the rows are the time intervals.

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

getRecursions, getRecursionsAtLocations

Examples

```
data(martin)
revisits = getRecursions(martin, radius = 1)
breaks = strptime(c("2000-01-01 00:00:00", "2000-01-15 00:00:00", "2000-02-01 00:00:00"),
format = "")
intervalResTime = calculateIntervalResidenceTime(revisits, breaks)
```

---

drawCircle *Draws a circle*

Description

Draws a circle in data coordinates, so it will be a circle if the aspect ratio of the plot is 1, or else it will be appear as an ellipse.

Usage

```
drawCircle(x, y, radius, nv = 100, border = NULL, col = NA, lty = 1, lwd = 1)
```

Arguments

x	x-coordinate of circle center
y	y-coordinate of circle center
radius	radius of circle
nv	how many plotted segments
border	polygon border
col	line color
lty	line type
lwd	line width

Details

This function is useful to display a representative circle with the specified radius on a plot of revisits.

Value

invisibly, the x and y points of the drawn circle

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

plot.recurse

Examples

```
data(martin)
revisits = getRecursions(martin, radius = 1)
plot(revisits, martin, legendPos = c(10, -15))
drawCircle(10, -10, 1)
```

getRecursions *Calculates recursion information from the trajectory*

Description

A circle of radius R is drawn around each point in the trajectory. The number of revisits is calculated as the number of segments of the trajectory passing through that circle.

Usage

```
getRecursions(
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'data.frame'
getRecursions(
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'Move'
getRecursions(
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'move2'
getRecursions(
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'MoveStack'
getRecursions(
```

```
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)
```

Arguments

x	Either a data frame, move2, Move-class, or MoveStack object. For a data frame, the trajectory data with four columns (the x-coordinate, the y-coordinate, the datetime, and the animal id).
radius	numeric radius to use in units of the (x,y) location data to detect recursions.
threshold	a time difference (in units timeunits) to ignore excursions outside the radius. Defaults to 0.
timeunits	character string specifying units to calculate time differences in for the time spans inside the radius and since the visit in revisitStats. Defaults to hours.
verbose	TRUE to output complete information (can be large for large input data frames) or FALSE to output basic information.

Details

For each point in the trajectory, a circle of radius R is drawn around that point. Then the number of segments of the trajectory passing through that circle is counted. This is the number of revisits, so each point will have at least one revisit (the initial visit). For each revisit, the time spent inside the circle is calculated, as well as the time since the last visit (NA for the first visit). In order to calculate the time values, the crossing time of the radius is calculated by assuming linear movement at a constant speed between the points inside and outside the circle.

Projection. Consider the projection used. Since segments are counted passing through circles drawn around points, an equal area projection would ensure similar size comparisons (e.g., spTransform).

Either single or multiple individuals are supported, but be aware that this function will be slow with large amounts of data (e.g. millions of points), so consider pre-specifying the locations (getRecursionsAtLocations) or use clustering. Multiple individuals are handled via the id column of the data.frame or using a move2 or MoveStack object.

Value

A list with several components, revisits and residenceTime are vectors of the same length as the x dataframe. revisits is the number of revisits for each location, where 1 means that there were no revisits, only the initial visit. residenceTime is the total time spent withing the radius. radius is the specified radius used for all the calculations. timeunits is the specified time units used to specify timespans.

When verbose = TRUE, additional information is also returned, dists and revisitStats. Next, dists gives the distance matrix between all locations. Finally, revisitStats gives further statistics on each visit. These are calculated per location (i.e., no aggregation of nearby points is performed), and give the index and location of the point of the track at the center of the radius, the radius entrance and exit time of the track for that visit, how much time was spent inside the radius, and how long since the last visit (NA for the first visit).

Methods (by class)

- `getRecursions(data.frame)`: Get recursions for a data.frame object consisting of columns x, y, datetime, and id
- `getRecursions(Move)`: Get recursions for a Move-class object
- `getRecursions(move2)`: Get recursions for a move2 object (for details see `vignette("programming_move2_object" package = "move2")`)
- `getRecursions(MoveStack)`: Get recursions for a MoveStack object

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

`getRecursionsAtLocations`

Examples

```
data(martin)
revisits = getRecursions(martin, radius = 1)
plot(revisits, martin, legendPos = c(10, -15))
drawCircle(10, -10, 1)
```

`getRecursions3D` *Calculates recursion information from the 3D trajectory*

Description

A sphere of radius R is drawn around each point in the trajectory. The number of revisits is calculated as the number of segments of the trajectory passing through that sphere.

Usage

```
getRecursions3D(
  x,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'data.frame'
getRecursions3D(
  x,
  radius,
```

```
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)
```

Arguments

x	A data frame with the trajectory data in five columns (the x-coordinate, the y-coordinate, the z-coordinate, the datetime, and the animal id).
radius	numeric radius to use in units of the (x,y,z) location data to detect recursions.
threshold	a time difference (in units timeunits) to ignore excursions outside the radius. Defaults to 0.
timeunits	character string specifying units to calculate time differences in for the time spans inside the radius and since the visit in revisitStats. Defaults to hours.
verbose	TRUE to output complete information (can be large for large input data frames) or FALSE to output basic information.

Details

For each point in the trajectory, a sphere of radius R is drawn around that point. Then the number of segments of the trajectory passing through that sphere is counted. This is the number of revisits, so each point will have at least one revisit (the initial visit). For each revisit, the time spent inside the sphere is calculated, as well as the time since the last visit (NA for the first visit). In order to calculate the time values, the crossing time of the radius is calculated by assuming linear movement at a constant speed between the points inside and outside the sphere.

Projection. Consider the projection used. Since segments are counted passing through spheres drawn around points, an equal area projection would ensure similar size comparisons (e.g., spTransform).

Either single or multiple individuals are supported, but be aware that this function will be slow with large amounts of data (e.g. millions of points), so consider pre-specifying the locations (getRecursionsAtLocations) or use clustering. Multiple individuals are handled via the id column of the data.frame.

Value

A list with several components, revisits and residenceTime are vectors of the same length as the x dataframe. revisits is the number of revisits for each location, where 1 means that there were no revisits, only the initial visit. residenceTime is the total time spent withing the radius. radius is the specified radius used for all the calculations. timeunits is the specified time units used to specify timespans.

When verbose = TRUE, additional information is also returned, dists and revisitStats. Next, dists gives the distance matrix between all locations. Finally, revisitStats gives further statistics on each visit. These are calculated per location (i.e., no aggregation of nearby points is performed), and give the index and location of the point of the track at the center of the radius, the radius entrance and exit time of the track for that visit, how much time was spent inside the radius, and how long since the last visit (NA for the first visit).

Methods (by class)

- `getRecursions3D(data.frame)`: Get recursions for a data.frame object consisting of columns x, y, z, datetime, and id

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

`getRecursions3DAtLocations`

Examples

```
data(martin)
revisits = getRecursions(martin, radius = 1)
plot(revisits, martin, legendPos = c(10, -15))
drawCircle(10, -10, 1)
```

---

getRecursions3DAtLocations

Calculates recursion information from the 3D trajectory for specific locations

---

Description

A sphere of radius R is drawn around each specified location. The number of revisits is calculated as the number of segments of the trajectory passing through that sphere.

Usage

```
getRecursions3DAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'data.frame'
getRecursions3DAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
```

```
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)
```

Arguments

x	A data frame with the trajectory data with five columns (the x-coordinate, the y-coordinate, the z-coordinate, the datetime, and the animal id).
locations	A data frame with x and y locations at which to calculate the recursions.
radius	numeric radius to use in units of the (x,y,z) location data to detect recursions.
threshold	a time difference (in units timeunits) to ignore excursions outside the radius. Defaults to 0.
timeunits	character string specifying units to calculate time differences in for the time spans inside the radius and since the visit in revisitStats. Defaults to hours.
verbose	TRUE to output complete information (can be large for large input data frames) or FALSE to output basic information.

Details

For specified location, a sphere of radius R is drawn around that point. This method differs from getRecursions in that only specified locations are used, rather than all points in the trajectory. Then the number of segments of the trajectory passing through that sphere is counted. This is the number of revisits to that location. For each revisit, the time spent inside the sphere is calculated, as well as the time since the last visit (NA for the first visit). In order to calculate the time values, the crossing time of the radius is calculated by assuming linear movement at a constant speed between the points inside and outside the sphere.

Projection. Consider the projection used. Since segments are counted passing through spheres drawn around points, an equal area projection would ensure similar size comparisons (e.g., spTransform).

Either single or multiple individuals are supported, but be aware that this function will be slow with large amounts of data (e.g. millions of points), so consider pre-specifying the locations (getRecursionsAtLocations) or use clustering. Multiple individuals are handled via the id column of the data.frame.

Value

A list with several components, revisits and residenceTime are vectors of the same length as the x dataframe. revisits is the number of revisits for each location, where 1 means that there were no revisits, only the initial visit. residenceTime is the total time spent withing the radius. radius is the specified radius used for all the calculations. timeunits is the specified time units used to specify timespans.

When verbose = TRUE, additional information is also returned, dists and revisitStats. Next, dists gives the distance matrix between all locations. Finally, revisitStats gives further statistics on each visit. These are calculated per location (i.e., no aggregation of nearby points is performed), and give the index and location of the point of the track at the center of the radius, the radius entrance and exit time of the track for that visit, how much time was spent inside the radius, and how long since the last visit (NA for the first visit).

Methods (by class)

- `getRecursions3DAtLocations(data.frame)`: Get recursions at specified locations for a data.frame object

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

`getRecursions`

Examples

```
data(martin)
locations = data.frame(x = c(-10, 0, 20), y = c(5, 0, 0))
revisits = getRecursionsAtLocations(martin, locations, radius = 1)
plot(revisits, locations, legendPos = c(10, -15),
     alpha = 1, pch = 17, xlim = range(martin$x), ylim = range(martin$y))
points(martin$x, martin$y, pch = ".", col = "gray50")
drawCircle(10, -10, 1)
```

`getRecursionsAtLocations` *Calculates recursion information from the trajectory for specific locations*

Description

A circle of radius R is drawn around each specified location. The number of revisits is calculated as the number of segments of the trajectory passing through that circle.

Usage

```
getRecursionsAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'data.frame'
getRecursionsAtLocations(
  x,
  locations,
```

```
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'Move'
getRecursionsAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'move2'
getRecursionsAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'MoveStack'
getRecursionsAtLocations(
  x,
  locations,
  radius,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)
```

Arguments

x	Either a data frame, move2, Move-class, or MoveStack object. For a data frame, the trajectory data with four columns (the x-coordinate, the y-coordinate, the datetime, and the animal id).
locations	A data frame with x and y locations at which to calculate the recursions.
radius	numeric radius to use in units of the (x,y) location data to detect recursions.
threshold	a time difference (in units `timeunits`) to ignore excursions outside the radius. Defaults to 0.
timeunits	character string specifying units to calculate time differences in for the time spans inside the radius and since the visit in `revisitStats`. Defaults to hours.

verbose — TRUE to output complete information (can be large for large input data frames) or FALSE to output basic information.

Details

For specified location, a circle of radius R is drawn around that point. This method differs from getRecursions in that only specified locations are used, rather than all points in the trajectory. Then the number of segments of the trajectory passing through that circle is counted. This is the number of revisits to that location. For each revisit, the time spent inside the circle is calculated, as well as the time since the last visit (NA for the first visit). In order to calculate the time values, the crossing time of the radius is calculated by assuming linear movement at a constant speed between the points inside and outside the circle.

Projection. Consider the projection used. Since segments are counted passing through circles drawn around points, an equal area projection would ensure similar size comparisons (e.g., spTransform).

Either single or multiple individuals are supported, but be aware that this function will be slow with large amounts of data (e.g. millions of points), so consider pre-specifying the locations (getRecursionsAtLocations) or use clustering. Multiple individuals are handled via the id column of the data.frame or using a move2 or MoveStack object.

Value

A list with several components, revisits and residenceTime are vectors of the same length as the x dataframe. revisits is the number of revisits for each location, where 1 means that there were no revisits, only the initial visit. residenceTime is the total time spent withing the radius. radius is the specified radius used for all the calculations. timeunits is the specified time units used to specify timespans.

When verbose = TRUE, additional information is also returned, dists and revisitStats. Next, dists gives the distance matrix between all locations. Finally, revisitStats gives further statistics on each visit. These are calculated per location (i.e., no aggregation of nearby points is performed), and give the index and location of the point of the track at the center of the radius, the radius entrance and exit time of the track for that visit, how much time was spent inside the radius, and how long since the last visit (NA for the first visit).

Methods (by class)

- getRecursionsAtLocations(data.frame): Get recursions at specified locations for a data.frame object
- getRecursionsAtLocations(Move): Get recursions at specified locations for a Move-class object
- getRecursionsAtLocations(move2): Get recursions at specified locations for a move2 object (for details see vignette("programming_move2_object", package = "move2"))
- getRecursionsAtLocations(MoveStack): Get recursions at specified locations for a MoveStack object

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

getRecursions

Examples

```
data(martin)
locations = data.frame(x = c(-10, 0, 20), y = c(5, 0, 0))
revisits = getRecursionsAtLocations(martin, locations, radius = 1)
plot(revisits, locations, legendPos = c(10, -15),
     alpha = 1, pch = 17, xlim = range(martin$x), ylim = range(martin$y))
points(martin$x, martin$y, pch = ".", col = "gray50")
drawCircle(10, -10, 1)
```

---

getRecursionsInPolygon.Move

Calculates recursion information from the trajectory inside a polygon

---

Description

The number of revisits to a polygon is calculated as the number of segments of the trajectory passing through the polygon.

Usage

```
## S3 method for class 'Move'
getRecursionsInPolygon(
  trajectory,
  polygon,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

getRecursionsInPolygon(
  trajectory,
  polygon,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'data.frame'
getRecursionsInPolygon(
  trajectory,
  polygon,
  threshold = 0,
```

```
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'move2'
getRecursionsInPolygon(
  trajectory,
  polygon,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)

## S3 method for class 'MoveStack'
getRecursionsInPolygon(
  trajectory,
  polygon,
  threshold = 0,
  timeunits = c("hours", "secs", "mins", "days"),
  verbose = TRUE
)
```

Arguments

`trajectory`	Either a data frame, move2, Move-class, or MoveStack object. For a data frame, the trajectory data with four columns (the x-coordinate, the y-coordinate, the datetime, and the animal id)
`polygon`	A st_polygon object with a single convex polygon.
`threshold`	A time difference (in units `timeunits`) to ignore excursions outside the radius. Defaults to 0.
`timeunits`	Character string specifying units to calculate time differences in for the time spans inside the radius and since the visit in `revisitStats`. Defaults to hours.
`verbose`	`TRUE` to output complete information (can be large for large input data frames) or `FALSE` to output basic information.

Details

The number of segments of the trajectory passing through the polygon is counted as the number of revisits. For each revisit, the time spent inside the polygon is calculated, as well as the time since the last visit (NA for the first visit). In order to calculate the time values, the crossing time of the polygon is calculated by assuming linear movement at a constant speed between the points inside and outside the polygon. Note the polygon must be convex as described in further detail below.

Projection. Consider the projection used. Since segments are counted passing through the polygon, an equal area projection would ensure similar size comparisons. A geographic projection is not appropriate. The projection for the polygon and the trajectory must be the same.

Polygon. The polygon must be specified as a st_polygon object. It should consist of a single polygon (i.e. st_geometry_type = POLYGON). It should further be convex, though this requirement

is not enforced, calculations for non-convex polygons will not necessarily be accurate. It may be advantageous to simplify complex geometry in order to shorten the time to run. If it is necessary to use a non-convex polygon, one approach would be to split it into convex pieces that can be run one-by-one. However, some visits would then be double-counted and would need to be combined back together based on the entrance/exit times and sequence of trajectory locations. Multiple polygons would need to be handled with multiple calls with the output then concatenated together.

Either single or multiple individuals are supported, but be aware that this function will be slow with large amounts of data (e.g. millions of points). Multiple individuals are handled via the `id` column of the data.frame.

Value

A list with several components. `revisits` is the number of revisits to the polygon. `residenceTime` is the total time spent withing the polygon. `radius` is NA in the case of polygons. `timeunits` is the specified time units used to specify timespans.

When `verbose = TRUE`, additional information is also returned in `revisitStats`. Next, `dists` gives the distance matrix between all locations. Finally, `revisitStats` gives further statistics on each visit. These are calculated per location (i.e., no aggregation of nearby points is performed), and give the index and location of the point of the track at the center of the radius (NA and 1 in the case of polygons), the radius entrance and exit time of the track for that visit, how much time was spent inside the radius, and how long since the last visit (`NA` for the first visit).

Methods (by class)

- `getRecursionsInPolygon(Move)`: Get recursions in polygon for a Move-class trajectory
- `getRecursionsInPolygon(data.frame)`: Get recursions inside a polygon for a trajectory data.frame object consisting of columns x, y, datetime, and id
- `getRecursionsInPolygon(move2)`: Get recursions in polygon for a move2 object (for details see `vignette("programming_move2_object", package = "move2")`)
- `getRecursionsInPolygon(MoveStack)`: Get recursions in polygon for a MoveStack trajectory

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

`getRecursions`

Examples

```
if (requireNamespace("sf"))
{
    data(track)
    poly = sf::st_polygon(list(cbind(c(4,6,6,3,4), c(1,2,4,3,1))))
    poly = sf::st_sfc(poly, crs = "EPSG:3410")
    revisits = getRecursionsInPolygon(track, poly)
}
```

martin *Sample trajectory (martin).*

Description

A dataset containing a sample trajectory with revisits.

Usage

```
data(martin)
```

Format

A data frame with 600 rows and 4 columns

Details

- x. x-coordinate
- y. y-coordinate
- t. time
- id. identifier

plot.recurse *Calculates recursion information from the trajectory*

Description

Plots a trajectory color coded by number of revisits to each point.

Usage

```
## S3 method for class 'recurse'
plot(x, xyt, ..., col, alpha = 1, legendPos = NULL)
```

Arguments

x	recurse object returned from call to getRecursions
xyt	data.frame of x, y, t, and id representing the xy-coordinates and the time (same as call to getRecursions)
...	additional arguments to plot
col	optional vector of colors as long as the maximum number of revisits to color code trajectory points
alpha	optional alpha value for color transparency between 0 and 1
legendPos	a vector of length 2 with the x and y coordinate of the center of the legend in user coordinates

Details

This method allows the user to visually represent the number of revisitations by location. The size of the circle of radius R can be added to the plot with `drawCircle`.

Value

the plot

Author(s)

Chloe Bracis <cbracis@uw.edu>

See Also

`getRecursions`, `getRecursionsAtLocations`, `drawCircle`

Examples

```
data(martin)
revisits = getRecursions(martin, radius = 1)
plot(revisits, martin, legendPos = c(10, -15))
drawCircle(10, -10, 1)
```

---

recurse *Computes revisitation metrics for trajectory data*

Description

Computes revisitation metrics for trajectory data, such as the number of revisitations for each location as well as the time spent for that visit and the time since the previous visit. Also includes functions to plot data.

Details

The function `getRecursions` computes the revisit metrics, which can be plotted with `plot.recurse`. Alternatively, `getRecursionsAtLocations` computes revisit metrics for specified locations, rather than all locations in the movement trajectory.

Author(s)

Chloe Bracis <cbracis@uw.edu>

track *Sample trajectory (track).*

Description

A dataset containing a sample trajectory with revisits.

Usage

```
data(track)
```

Format

A data frame with 100 rows and 4 columns

Details

- x. x-coordinate
- y. y-coordinate
- t. time
- id. identifier

wren *Sample trajectory (wren).*

Description

A dataset containing a sample trajectory with revisits.

Usage

```
data(wren)
```

Format

A data frame with 600 rows and 4 columns

Details

- x. x-coordinate
- y. y-coordinate
- t. time
- id. identifier

Index