Package 'SNMA'

February 4, 2026

Type Package

Title Stream Network Movement Analyses

Version 0.1.5

Date 2026-01-24

Maintainer Donald T. McKnight <donald.mcknight@my.jcu.edu.au>

Description Calculating home ranges and movements of animals in complex stream environments is often challenging, and standard home range estimators do not apply. This package provides a series of tools for assessing movements in a stream network, such as calculating the total length of stream used, distances between points, and movement patterns over time. See Vignette for additional details. This package was originally released on 'GitHub' under the name 'SNM'. SNMA was developed for analyses in McKnight et al. (2025) <doi:10.3354/esr01442> which contains additional examples and information.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports sf,geosphere,igraph

Suggests ggplot2, ggnewscale, knitr,rmarkdown

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

Author Donald T. McKnight [aut, cre] (affiliation: Savanna Field Station)

Repository CRAN

Date/Publication 2026-02-04 18:10:13 UTC

Contents

`animal.points` *Example animal location data*

Description

Hypothetical data set with the GPS coordinates for two turtles

Usage

```
animal.points
```

Format

A data frame with 11 rows and 5 columns:

lon.raw Longitude in decimal degrees

lat.raw Latitude in decimal degrees

lon.shifted Longitude shifted to line up with stream points to illustrate how the functions work

lat.shifted Latitude shifted to line up with stream points to illustrate how the functions work

id Turtle IDs

date.time Dates and times of points in POSIX format

point Names of each coordinate (optional)

Source

Fictional example generated to demonstrate this package

`calc.stream.dist` *Calculate distance along a stream*

Description

Calculate the shortest distance along a stream network between a pair of points

Usage

```
calc.stream.dist(p1, p2, data)
```

Arguments

`p1`	(numeric vector) Longitude and latitude (in that order) of first point
`p2`	(numeric vector) Longitude and latitude (in that order) of second point
`data`	(list) Output from `prep.data`.

Details

This is the primary function for calculating the shortest distance between two points along the stream network.

Value

(numeric) Distance (in meters) between the two points

Examples

```
data(stream.line)
data(nodes)
data(animal.points)

network.20 <- prep.data(
  l = stream.line,
  freq = 20,
  nodes = nodes,
  lon.name = "lon",
  lat.name = "lat",
  node.name = "id"
)

calc.stream.dist(p1 = c(-88.99845, 17.17668),
 p2 = c(-88.99838, 17.17710),
 data=network.20)
```

`dist.over.time` *Distance moved over time intervals*

Description

For a series of time intervals, calculate the distance moved between each point and the previous point matching the given time interval.

Usage

```
dist.over.time(
  data,
  coords,
  lon.name = "lon",
  lat.name = "lat",
  id.name = "id",
  date.time.name = "date.time",
  units = "days",
  time.diff = 1,
  diff.max = NULL,
  sensitivity.min = 0.1,
  sensitivity.max = 0.1,
  sensitivity.change = 0,
  custom.times = NULL,
  custom.sensitivity = NULL
)
```

Arguments

`data`	(list) Output from `prep.data`.
`coords`	(data.frame) Data frame containing coordinates for animal locations. It must include a column of animal IDs, columns of latitude and longitude (in decimal degrees projected to the same system as the rest of the data), and a column of dates/times (in POSIX format). All other columns will be ignored.
`lon.name`	(character) For the `coords` object, the name of the column of longitudes (in quotes). Default = "lon"
`lat.name`	(character) For the `coords` object, the name of the column of latitudes (in quotes). Default = "lat"
`id.name`	(character) For the `coords` object, the name of the column of animal identities (in quotes). Default = "id"
`date.time.name`	(character) For the `coords` object, the name of the column of dates and times (in a POSIX format). Default = "date.time"
`units`	(character) One of "secs", "mins", "hours", or "days" specifying the units for all time and sensitivity inputs and outputs

`time.diff` (numeric) The amount of time to separate each analysis of distance (i.e., a time interval based on `units`). Default = 1

`diff.max` (numeric) The maximum time interval to analyze. If NULL, it will use the maximum possible interval. Default = `NULL`

`sensitivity.min` (numeric) The minimum (starting) buffer (+/-) around `time.diff` (i.e., the buffer when `time.diff` = 1). Default = 0.1

`sensitivity.max` (numeric) The maximum buffer (+/-) around `time.diff`. Default = 0.1

`sensitivity.change` (numeric) The amount that the sensitivity should change (both + and -) each time interval. Default = 0

`custom.times` (numeric vector) Optional vector of custom time intervals to analyze (overrides time.diff and diff.max)

`custom.sensitivity` (numeric vector) Optional vector of custom sensitivities to apply to time intervals (overrides all other sensitivity arguments)

Details

This function calculates movement over given intervals of time. For each coordinate (going from oldest to newest) it looks for matches to previous coordinates where the time difference between them matches the specified interval.

`time.diff`, `diff.max`, and `units` specify the time intervals. `time.diff` specifies the amount of additional time in each sequential interval, with `diff.max` specifying the largest time interval. For example, if `time.diff` = 6, `diff.max`, = 24 and `units` = "hours" movement will be calculated over 6 hours, 12 hours, 18 hours, and 24 hours.

Points can be used for multiple time intervals. For example, if `time.diff` = 1 and `units` = "days" and an animal was tracked at 12:00 on the 1st, 2nd, 3rd, 5th, and 6th of January, the following pairs of points would be used for each time interval:

- 1 day interval = days 1&2, 2&3, 5&6
- 2 day interval = days 1&3, 3&5
- 3 day interval = days 2&5, 3&6
- 4 day interval = days 1&5, 2&6
- 5 day interval = days 1&6

`sensitivity.min`, `sensitivity.max`, and `sensitivity.change` control a buffer of time (+/-) around each interval. This allows for situations such as radio telemetry where points will rarely match an exact time interval. For example, if the time interval is 24 hours with a 1 hour (+/-) buffer, then for each point, it will look for corresponding points 23-25 hours previously.

In many cases, it may be desirable to increase the buffer slightly as the size of the time intervals increase. The `sensitivity.max` and `sensitivity.change` arguments control this. `sensitivity.max` specifies the maximum value a buffer can have, and `sensitivity.change` controls the amount of buffer increase for new time interval.

Example: If:

- time.diff = 1
- diff.max = 6
- units = "days"
- sensitivity.min = 0.1
- sensitivity.max = 0.5
- sensitivity.change = 0.1

The function will use intervals of 1, 2, 3, 4, 5, 6 days, and the corresponding buffers will be +/- 0.1, 0.2, 0.3, 0.4, 0.5, 0.5 days.

Note that the units are the same for all entries, and the buffer caps at +/- 0.5 because of diff.max, Thus, for a given coordinate, it will look for corresponding coordinates 0.9-1.1, 1.8-2.2, 2.7-3.3, 3.6-4.4, 4.5-5.5, and 5.5-6.5 days previously.

To remove the buffer entirely, set sensitivity.min = 0, sensitivity.max = 0, and sensitivity.change = 0

To set a fixed buffer, set sensitivity.min and sensitivity.max to the same value and set sensitivity.change = 0

custom.times and custom.sensitivity allow users to supply vectors of times and sensitivities that change at irregular intervals (e.g., 1, 3, 5, 10 days). These arguments override other time and sensitivity arguments.

Note: The default sensitivity values are largely arbitrary, and you should carefully select your own based on your study questions and system.

Note: For a given coordinate and time interval, only one match will be allowed with a previous coordinate. If multiple matches are available within a time interval, only the one that is closest to the specified interval will be retained (in the case of ties, the first is arbitrarily retained). For example, if the time interval is 1 day with a +/- .5 buffer and the following points are available: 2023-01-01 12:00:00, 2023-01-01 13:00:00, and 2023-01-02 12:00:00, then when calculating the distance for 2023-01-02 12:00:00, it will only calculate the distance to the point at 2023-01-01 12:00:00 because it was closer to the specified time interval of 1 day.

Value

A data frame containing the outputs for each point at each time interval.

Columns lat,lon,id, and date.time are for the ending position for the interval analyzed

Column dist = distance moved (m) during a given interval (during the actual.diff prior to the specified point)

Column actual.diff = the exact time between the ending position and the previous point to which it was paired (in the time units specified by units)

Column time.diff = the time interval specified by time.diff (in the time units specified by units)

Column sensitivity = the time buffer (+/-) around the time.diff for the given point (in the time units specified by units)

Note: Although each endpoint will only be matched with one previous point for a given time interval, there may be multiple different end points within that time interval. For example, if time.diff

= 1, sensitivity.min = .1, units = "days", and for a given animal, there were points at "2023-06-01 12:00:00", "2023-06-02 12:00:00" and "2023-06-02 12:30:00", the results will include a row with "2023-06-02 12:00:00" as the endpoint and a row with "2023-06-02 12:30:00" as the endpoint (both showing the distance from "2023-06-01 12:00:00"). Depending on the data set and questions being asked, that may create undesirable pseudoreplication and you may need to filter the results based on date or some other criterion.

Note: When reading the time.diff column, remember that points are not necessarily sequential. Thus, days = 1, 2, 3 does not necessarily indicate movement over days 1, 2, and 3 of the study, rather it is movement over that many days, regardless of when in the study the pairs of points occurred. See the vignette for suggestions on visualizing and analyzing these data.

Examples

```
data(stream.line)
data(nodes)
data(animal.points)

network.20 <- prep.data(
  l = stream.line,
  freq = 20,
  nodes = nodes,
  lon.name = "lon",
  lat.name = "lat",
  node.name = "id"
)

dist.results <- dist.over.time(data=network.20,
  coords=animal.points,
  lon.name="lon.raw",
  lat.name="lat.raw",
  id.name="id",
  date.time.name="date.time",
  units="days",
  time.diff=1,
  diff.max=NULL,
  sensitivity.min=0,
  sensitivity.max=0,
  sensitivity.change=0)
```

---

movements *Calculate movements*

Description

Calculate movements in a stream network for multiple individuals

Usage

```
movements(
  data = NULL,
  space.use = TRUE,
  from.previous = TRUE,
  cumulative = TRUE,
  downstream.node = NULL,
  coords,
  lon.name = "lon",
  lat.name = "lat",
  id.name = "id",
  date.time.name = NULL
)
```

Arguments

data	(list) Output from prep.data.
space.use	(logical) If TRUE (default), total space use (range) will be calculated up to and including each point
from.previous	(logical) If TRUE (default), the distance from the previous point will be calculated.
cumulative	(logical) If TRUE (default), the cumulative distance moved up to and including each point will be calculated.
downstream.node	(integer) The ID (number) of the node representing the furthest point downstream in your network (the root). If included, the distance from that point and the direction of movement (upstream or downstream) will be calculated per point.
coords	(data.frame) Data frame containing coordinates for animal locations. At a minimum, it must include a column of animal IDs and columns of latitude and longitude (in decimal degrees projected to the same system as the rest of the data). It can optionally include a date.time column (highly recommended) containing the date and time of each point (in POSIX format). If the date.time column is present, the data will be sorted by that. Otherwise, ensure the data are sorted from oldest to newest. Other columns of metadata can be included and will be returned.
lon.name	(character) For the coords object, the name of the column of longitudes (in quotes). Default = "lon"
lat.name	(character) For the coords object, the name of the column of latitudes (in quotes). Default = "lat"
id.name	(character) For the coords object, the name of the column of animal identities (in quotes). Default = "id"
date.time.name	(character: optional) For the coords object, the name of the column of dates and times (in a POSIX format). If included, the data will be sorted from oldest to newest (otherwise sort before running), and the time difference between consecutive points will be returned. Default = NULL

Details

It takes a data frame of coordinates and calculates the specified summary statistics for each individual. All results are in meters. Input data should be sorted from oldest point to newest, unless a date.time column is included, in which case the sorting will be done internally.

`space.use` If `TRUE`, for each point, it will calculate the total space use up to and including the given point. This includes all branches of the network covered by the points, but no portion is measured more than once. It is simply the total area of the stream (expressed as linear meters) visited by the animal. Note that there may occasionally be a very slight difference between this metric and the other metrics when they describe the same space. This is caused by a slight misalignment between a node and the end of a segment, but usually will be trivial.

`from.previous` If `TRUE`, for each point, it calculates the distance between that point and the previous point (following the stream network). This calculation is required for the `cummualtive` calculations and outputs will be returned anytime `cummualtive == T`, even if `from.previous == F`.

`cumulative` If `TRUE`, for each point, it calculates the total distance moved up to and including that point. This differs from `space.use` because stretches of river get included each time they are visited, so the cumulative distance increases anytime the animal moves.

`downstream.node` If this is included, for each point, it calculates the current distance from the point furthest downstream. This can be a useful way to visualize movements over time by referencing a fixed point (the furthest downstream). It also returns a column showing the direction of movement. Keep in mind that on branched rivers, these results may be somewhat misleading. If, for example, a stream forks 10 m upstream, and an animal moves from 5 m up the right fork then backtracks on goes 6 m up the left fork, the distances from the furthest downstream point will be 15 and 16 (respectively) even though the animal moved 5 m downstream the left fork before moving 6 m up the other. Likewise, the direction would be scored as "upstream" even though it first moved downstream. The direction is based simply on whether the shortest distance to the furthest downstream point increased or decreased.

Note: This function does not incorporate the amount of time between points. Bear that in mind when interpreting data collected over uneven time intervals.

Example: If an individual starts 100 meters upstream from the furthest downstream point in your network and moves 10 m upstream from day 1 to 2, another 5 m upstream from day 2 to 3, then 4 m downstream from day 3 to 4, then doesn't move between days 4 and 5, it will return the following:

space.use	dist.from.prev	cumulative.dist	dist.from.downstream	direction
NA	NA	NA	100	NA
10	10	10	110	upstream
15	5	15	115	upstream
15	4	19	111	downstream
15	0	19	111	no.change

The first row is NA in most cases because there are no previous points to make comparisons. The `dist.from.prev` column is simply the distances described in the example. The `space.use` column shows increasing total space use over time. Notice that it does not change on days 4 and 5 because the animal visits areas where it had already been recorded. In contrast, the `cumulative.dist` increased on day 4 because it is simply the sum of all distances. Most values remained the same on the last day because there was no movement (as reflected by the `dist.from.prev` and `direction` columns).

Value

Data frame including all original columns, plus a columns of summary data. Column order may have changed. All measurements are in linear meters.

Column `space.use` = result of `space.use = TRUE`

Column `dist.from.prev` = result of `from.previous = TRUE`

Column `cumulative.dist` = result of `cumulative = TRUE`

Column `dist.from.downstream` = result of `downstream.node`

Column `direction` = result of `downstream.node` (direction)

Column `time.diff` = returned if a date.time column is included. Shows the elapsed time (in hours) between consecutive points

Examples

```
data(stream.line)
data(nodes)
data(animal.points)

network.20 <- prep.data(
  l = stream.line,
  freq = 20,
  nodes = nodes,
  lon.name = "lon",
  lat.name = "lat",
  node.name = "id"
)

move.results <- movements(data=network.20,
  space.use=TRUE,
  from.previous=TRUE,
  cumulative=TRUE,
  downstream.node=1,
  coords=animal.points,
  lon.name="lon.raw",
  lat.name="lat.raw",
  id.name="id",
  date.time.name="date.time")
```

---

nodes *Example node data*

Description

Hypothetical node data set

Usage

```
nodes
```

Format

A data frame with 8 rows and 3 columns:

lon Longitude in decimal degrees

lat Latitude in decimal degrees

id Node names (must be integers) ...

Source

Fictional example generated to demonstrate this package

`prep.data` *Prepare data*

Description

Wrapper function to prepare stream data for analyses

Usage

```
prep.data(
  l,
  freq = 1,
  nodes,
  lon.name = "lon",
  lat.name = "lat",
  node.name = "id"
)
```

Arguments

`l`	(shapefile) A projected polyline loaded with st_read(). Should include an id variable specifying different segments of the stream (this should be included even if there is only a single segment)
`freq`	(numeric) The distance (in meters) to add points. Default = 1
`nodes`	(data.frame) A data frame of coordinates for the nodes where stream segments meet or end. Should contain columns for latitude, longitude, and the identity of each node (**identities must be integers and go from 1 to number of nodes)**
`lon.name`	(character) Name of the column of longitudes (in quotes). Default = "lon"
`lat.name`	(character) Name of the column of latitudes (in quotes). Default = "lat"
`node.name`	(character) Name of the column of segment identities (in quotes). Default = "id"

Details

Takes a shapefile (polyline) of the stream (including all segments (i.e., edges), with each segment having a different ID) as well as a data frame of node locations. It then adds points at the specified interval (to increase accuracy) and does a series of calculations to build reference tables of distances along segments and among nodes.

Segments (edges) are defined as any section between two nodes with no nodes between them. Nodes occur at each start, end, or intersection of segments.

This is the only function needed to prepare data for analyses such as `calc.stream.dist`, `movements`, and `dist.over.time`

Note: All data MUST be in the same projected system. These functions are intended for data formatted in decimal degrees.

Do not use braided streams. If a stream is braided (i.e., channels split then reconnect forming islands) the calculations will be incorrect. If you are working in a braided system, but all of your known movements do not include the braids, simply use a shapefile that does not include the braids (e.g., a study that only tracks in the main channel and largest side channels, without tracking in braided channels)

Value

A list containing five objects:

`nodes` = (data.frame) Coordinates for the nodes where stream segments meet or end.

`increased.line` = (data.frame) Coordinates describing the center line of the stream with an increased frequency of points

`dps.segments` = (data.frame) Total distances of each segment and the nodes defining that segment

`dps.distances` = (list) List of data frames (one data frame per segment) with the distances between each consecutive pair of points

`node.distances` = (data.frame) Minimum distance between each pair of nodes (not simply nodes on a given segment)

Examples

```
data(stream.line)
data(nodes)
network.20 <- prep.data(
  l = stream.line,
  freq = 20,
  nodes = nodes,
  lon.name = "lon",
  lat.name = "lat",
  node.name = "id"
)
```

stream.boundaries *Example stream outline*

Description

Hypothetical outline of example stream (polygon shapefile). Not actually needed for functions.

Usage

```
stream.boundaries
```

Format

An sf data frame with 1 observation of 2 variables:

id NA

geometry shapefile geometry

Source

Fictional example generated to demonstrate this package

stream.line *Example stream network shapefile*

Description

Shapefile (polyline) for a hypothetical stream network

Usage

```
stream.line
```

Format

An "sf" data frame with 7 observations (stream segments) and 2 variables:

id id of each stream segement

geometry sf geometry data for each stream segement

Source

Fictional example generated to demonstrate this package. Shapefile was created in QGIS and imported via st_read()

Index