Package: topdowntimeratio (via r-universe)

September 9, 2024

Type Package

Title Top-Down Time Ratio Segmentation for Coordinate Trajectories

Version 0.1.0

Description Data collected on movement behavior is often in the form of time- stamped latitude/longitude coordinates sampled from the underlying movement behavior. These data can be compressed into a set of segments via the Top- Down Time Ratio Segmentation method described in Meratnia and de By (2004) <doi:10.1007/978-3-540-24741-8_44> which, with some loss of information, can both reduce the size of the data as well as provide corrective smoothing mechanisms to help reduce the impact of measurement error. This is an improvement on the well-known Douglas-Peucker algorithm for segmentation that operates not on the basis of perpendicular distances. Top-Down Time Ratio segmentation allows for disparate sampling time intervals by calculating the distance between locations and segments with respect to time. Provided a trajectory with timestamps, tdtr() returns a set of straight- line segments that can represent the full trajectory. McCool, Lugtig, and Schouten (2022) <doi:10.1007/s11116-022-10328-2> describe this method as implemented here in more detail.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports data.table (>= 1.9.8), geodist (>= 0.0.4), lubridate, magrittr, stats

Depends R (>= 4.1)

RoxygenNote 7.2.1

Suggests spelling, testthat

Language en-US

Repository https://daniellemccool.r-universe.dev

RemoteUrl https://github.com/daniellemccool/topdowntimeratio

RemoteRef HEAD

RemoteSha 2c8148b162d46a41279a64d6273eb4fd91178887

Contents

`getSegments` *Get Segments*

Description

Extract segment info from the segmented data.table.

Usage

```
getSegments(
  data,
  coord.type = c("coordinate", "distance", "both"),
  group = FALSE
)
```

Arguments

`data`	data.table returned from function tdtr()
`coord.type`	return actual coordinates, relative distance, or both (see Details)
`group`	separate by group, default is FALSE

Details

Segment location information can be either in lat/lon coordinates, or expressed in terms of distance for a more anonymous presentation of small trajectories. (Full anonymity is not guaranteed as sufficiently long trajectories with small error parameters can provide enough data to match against a map.)

Value

data.table with segments only, containing information about the start and end locations, start and end time and distance covered by the segment

Examples

```
df <- data.frame(entity_id = rep(1, 12),
   timestamp = c(1, 2, 4, 10, 14, 18, 20, 21, 24, 25, 28, 29),
   lon = c(5.1299311, 5.129979, 5.129597, 5.130028, 5.130555, 5.131083,
           5.132101, 5.132704, 5.133326, 5.133904, 5.134746, 5.135613),
   lat = c(52.092839, 52.092827, 52.092571, 52.092292, 52.092076, 52.091821,
           52.091420, 52.091219, 52.091343, 52.091651, 52.092138, 52.092698))
# First generate segments
res30 <- tdtr(df,
     group_col = NULL,
     max_error = 30)
# Then extract a data.table of segments
getSegments(res30)

# Calculating distance instead of coordinates
segs <- getSegments(res30, coord.type = "distance")
segs
plot(c(0, 700), c(0, 200), col = "white",
     xlab = "East-West distance",
     ylab = "North-South distance")
with(segs,
     segments(seg_start_lon_dist, seg_start_lat_dist,
      seg_end_lon_dist, seg_end_lat_dist))
```

`getSegsExtra` *Get Segments with calculated data*

Description

This function calculates various segment-level metrics that require the raw data before returning a data.table with the segments and the calculated results. Calculates speed, bearing and radius of gyration information.

Usage

```
getSegsExtra(
  data,
  coord.type = c("coordinate", "distance", "both"),
  group = FALSE
)
```

Arguments

data	data.table returned from function /codetdtr
coord.type	return actual coordinates, relative distance, or both
group	Separate by group, default is FALSE

Value

data.table of segments, annotated with segment-level information on distance, mean and variance of immediate bearing difference, total bearing variance over the segment, mean, maximum and variance of calculated speed in meters per second, percentage of zero-speed entries, whether the segment consists of fewer than 3 locations, and the time-weighted radius of gyration.

Examples

```
df <- data.frame(entity_id = rep(1, 12),
   timestamp = c(1, 2, 4, 10, 14, 18, 20, 21, 24, 25, 28, 29),
   lon = c(5.1299311, 5.129979, 5.129597, 5.130028, 5.130555, 5.131083,
           5.132101, 5.132704, 5.133326, 5.133904, 5.134746, 5.135613),
   lat = c(52.092839, 52.092827, 52.092571, 52.092292, 52.092076, 52.091821,
           52.091420, 52.091219, 52.091343, 52.091651, 52.092138, 52.092698))
# First generate segments
res100 <- tdtr(df,
     group_col = NULL,
     max_error = 100)
# Then extract a data.table of segments
getSegsExtra(res100)
```

---

iterate *Perform one iteration of segmentation. Updates by reference and should be an internal function.*

Description

Perform one iteration of segmentation. Updates by reference and should be an internal function.

Usage

```
iterate(data, max_error)
```

Arguments

data	data.table that has been setup by setup
max_error	stopping criteria from tdtr

meanFilter *Mean filter*

Description

Mean filter

Usage

```
meanFilter(coord, n = 3)
```

Arguments

coord	A vector of coordinates over which to apply a mean filter
n	The number of values to average

Value

A vector of mean-averaged coordinates

medianFilter *Median filter*

Description

Median filter

Usage

```
medianFilter(coord, n = 3)
```

Arguments

coord	A vector of coordinates over which to apply a mean filter
n	The number of values to average (best when odd-numbered)

Value

A vector of median-averaged coordinates

`radiusOfGyrationDT` *Radius of Gyration*

Description

Calculates the time-weighted radius of Gyration provided a data.table containing latitude, longitude and a timestamp. This is the root-mean-square time-weighted average of all locations. Weighting by time is provided to adjust for unequal frequency of data collection.

Usage

```
radiusOfGyrationDT(lat_col, lon_col, timestamp, dist_measure = "geodesic")
```

Arguments

`lat_col`	Time-ordered vector of latitudes
`lon_col`	Time-ordered vector of longitudes
`timestamp`	Timestamps associated with the latitude/longitude pairs
`dist_measure`	Passed through to geodist::geodist_vec, One of "haversine" "vincenty", "geodesic", or "cheap" specifying desired method of geodesic distance calculation.

Details

Time-weighted RoG is defined as

$$\sqrt{\frac{\sum_i w_j \times dist([\overline{lon}, \overline{lat}], [lon_j, lat_j])}{\sum_i w_j}}$$

Where

$$\overline{lon} = \frac{\sum_j w_j lon_j}{\sum_j w_j} \quad \text{and} \quad \overline{lat} = \frac{\sum_j w_j lat_j}{\sum_j w_j}$$

And the weighting element w_j represents half the time interval during which a location was recorded

$$w_j = \frac{t_{j+1} - t_{j-1}}{2}$$

Value

Time-weighted radius of gyration

Examples

```
# Inside a data.table
dt <- data.table::data.table(
  lat = c(1, 1, 1, 1, 1),
  lon = c(1, 1.5, 4, 1.5, 2),
  timestamp = c(100, 200, 300, 600, 900)
```

```
)
dt[, radiusOfGyrationDT(lat, lon, timestamp)]
# As vectors
radiusOfGyrationDT(
  c(1, 1, 1, 1, 1),
  c(1, 1.5, 4, 1.5, 2),
  c(100, 200, 300, 600, 900)
  )
```

`setup` *Set up a data.table for iterative segmentation*

Description

Set up a data.table for iterative segmentation

Usage

```
setup(data)
```

Arguments

`data`	A data.frame or data.table containing lat, lon and timestamp

Value

A data.table with numeric timestamp, and an initial segment

`splitDiffTime` *Split the difference when it comes to difftimes*

Description

Averages out time differences between successive locations. This is useful in calculating the time-weighted Radius of Gyration, as it provides a method of using both the first and last locations. This assumes that the location is measured at a given time period and will account for half of the time difference occurring between this location and the one immediately preceding, as well as half the time difference occurring between this location and the one immediately following.

Usage

```
splitDiffTime(timestamp)
```

Arguments

`timestamp`	a duration, period, difftime or interval

Value

the averaged difftime of same length

`tdtr` *Perform Top-Down Time Ratio segmentation*

Description

Perform Top-Down Time Ratio segmentation

Usage

```
tdtr(
  data,
  col_names = list(entity_id_col = "entity_id", timestamp_col = "timestamp", latitude_col
    = "lat", longitude_col = "lon"),
  group_col = "state_id",
  max_segs = 5000,
  n_segs = max_segs,
  max_error = 200,
  add_iterations = FALSE
)
```

Arguments

`data`	is a data.frame or data.table with timestamp, lat and lon
`col_names`	named list with existing column names for timestamp, latitude and longitude column (these are changed to 'timestamp', 'lat' and 'lon' respectively)
`group_col`	NULL for no grouping, or string column name representing a grouping in the data where initial segments will be drawn.
`max_segs`	with maximum number of segments allowed, default is 5000
`n_segs`	used to generate a specific number of segments
`max_error`	used as stopping criteria, default is 200
`add_iterations`	Add iterations to previous `tdtr` run

Value

data.table with segment information

Examples

```
df <- data.frame(person = rep(1, 12),
   time = c(1, 2, 4, 10, 14, 18, 20, 21, 24, 25, 28, 29),
   longitude = c(5.1299311, 5.129979, 5.129597, 5.130028, 5.130555, 5.131083,
           5.132101, 5.132704, 5.133326, 5.133904, 5.134746, 5.135613),
   lat = c(52.092839, 52.092827, 52.092571, 52.092292, 52.092076, 52.091821,
           52.091420, 52.091219, 52.091343, 52.091651, 52.092138, 52.092698))
# Generate segments under a max error of 100m
res100 <- tdtr(df,
     col_names = list(entity_id_col = "person",
                      timestamp_col = "time",
                      latitude_col = "lat",
                      longitude_col = "longitude"),
     group_col = NULL,
     max_error = 100)
# Generate segments under a max error of 30m
res30 <- tdtr(df,
     col_names = list(entity_id_col = "person",
                      timestamp_col = "time",
                      latitude_col = "lat",
                      longitude_col = "longitude"),
     group_col = NULL,
     max_error = 30)
plot(df$lon, df$lat)
segments(res100$seg_start_lon, res100$seg_start_lat,
         res100$seg_end_lon, res100$seg_end_lat, col = "blue")
segments(res30$seg_start_lon, res30$seg_start_lat,
         res30$seg_end_lon, res30$seg_end_lat, col = "red")
```

Index