Analysing Trajectory Data in R

Benedikt Klus Edzer Pebesma

August 19, 2015

Contents

1 Introduction

This vignette describes classes and methods which have initially been added to package *spacetime* [3, 1] and moved to the new package *trajectories*. They aim to improve the overall handling of trajectory data in R. To date, R is missing a complete set of generic data structures and methods to effectively analyse trajectories without being limited to a particular domain. One of the more comprehensive works dedicated to trajectories is the one of Calenge which he released as package *adehabitatLT* on CRAN during his PhD, but which is restricted to animal trajectory analysis [2]. The classes and methods presented below are an attempt to address the lack of a broader range of utilities to deal with trajectory data in R and integrate tightly with the classes and methods provided by package *spacetime*. To start trajectory'ing, load the package (and optionally its examples) with:

```
> library("spacetime")
> library("trajectories")
> example("Track")
```

2 Classes

The update implements four data classes for trajectory representation: `Track`, `Tracks`, `TracksCollection` and `segments`. The first three build upon class `STIDF`, whereas `segments` is based on `data.frame`. The classes and their instantiation options are subject of the following subsections.

2.1 Track

The class `Track` represents a single track followed by a person, animal or object. Instances of this class are meant to hold a series of consecutive location/time stamps that are not interrupted by another activity. The class contains five slots, four of which are inherited from class `STIDF`: `@sp` to store the geometry, `@time` to store the time, `@endtime` to store the end time when having generalised line geometries with one value per attribute for a set of points (otherwise, defaults to the time defined in `@time`), `@data` to store the attributes and `@connections` to keep a record of attribute data between points (e.g., distance, duration and speed). A `Track` object can be created out of an `STIDF` object like so:

```
> Track(stidf)
```

By default, distance, duration, speed and direction are computed as the connections data. Optionally, a data frame containing additional connections data and/or a custom function for calculating the data of segments between consecutive points can be passed. Please refer to the documentation for detailed information.

2.2 Tracks

The class `Tracks` embodies a collection of tracks followed by a single person, animal or object. The class contains two slots: `@tracks` to store the tracks as objects of class `Track` and `@tracksData` to hold a summary record for each particular track (e.g., minimum and maximum time, total distance and average speed). A `Tracks` object can be created by:

```
> Tracks(list(A1 = A1, A2 = A2))
```

... with `A1` and `A2` being instances of class `Track`. By default, the minimum and maximum coordinates and time, the total number of geometries, the total distance as well as the average speed are computed as the summary information data. Same to the `Track` method presented above, a data frame and/or a custom function can be passed to expand the default data.

2.3 TracksCollection

Finally, the class `TracksCollection` represents a collection of tracks followed by many persons, animals or objects. The class contains two slots: `@tracksCollection` to store the tracks as objects of class `Tracks` and `@tracksCollectionData` to hold summary information about each particular person, animal or object (e.g., the total number of tracks). A `TracksCollection` object can be created by:

```
> TracksCollection(list(A = A, B = B))
```

... with `A` and `B` being objects of class `Tracks`. By default, the total number of tracks as well as the minimum and maximum coordinates and time are computed as the summary information data. Same to the `Track` and `Tracks` methods outlined above, a data frame and/or a custom function can be passed to expand the default data.

2.4 segments

The class `segments` was written to provide a data structure for storing all the segments of a track with a segment representing the line between two consecutive points.

3 Methods

A wealth of methods has been implemented to cover the most frequently used use cases. The methods are presented along with illustrative examples in the following subsections.

3.1 Utility

The update implements the base methods `dim` and `summary` to retrieve the dimension and get summaries of `Track`, `Tracks` and `TracksCollection` objects.

```
> dim(Tr)

       IDs     tracks geometries 
         2          4         24 

> summary(Tr)

Object of class TracksCollection
 with Dimensions (IDs, tracks, geometries): (2, 4, 24)
[[stbox]]
    x y                time
min 1 1 2013-09-30 01:00:12
max 7 7 2013-09-30 01:11:10
[[Spatial:]]
Object of class SpatialPoints
Coordinates:
  min max
x   1   7
y   1   7
Is projected: FALSE 
proj4string : [+proj=longlat +ellps=WGS84]
Number of points: 24
[[Temporal:]]
     Index                       timeIndex    
 Min.   :2013-09-30 01:00:12   Min.   :1.000  
 1st Qu.:2013-09-30 01:02:45   1st Qu.:2.000  
 Median :2013-09-30 01:05:45   Median :3.500  
 Mean   :2013-09-30 01:05:36   Mean   :3.542  
 3rd Qu.:2013-09-30 01:08:45   3rd Qu.:5.000  
 Max.   :2013-09-30 01:11:10   Max.   :7.000  
[[Data attributes:]]
      co2          
 Min.   :-1.9482  
 1st Qu.:-0.8099  
 Median :-0.4393  
 Mean   :-0.1873  
 3rd Qu.: 0.4970  
 Max.   : 1.9839  
[[Connections:]]
    distance          duration           speed          direction    
 Min.   :110387   Min.   : 2.858   Min.   : 2184   Min.   :  0.0  
 1st Qu.:110394   1st Qu.:18.598   1st Qu.: 3111   1st Qu.:123.8  
```

```
 Median :110974   Median :27.577   Median : 4745   Median :180.0
 Mean   :126695   Mean   :28.328   Mean   : 7506   Mean   :173.2
 3rd Qu.:156416   3rd Qu.:38.105   3rd Qu.: 6607   3rd Qu.:224.9
 Max.   :156696   Max.   :58.901   Max.   :38628   Max.   :315.2
```

Furthermore, the methods `proj4string`, `coordinates`, `coordnames` and `bbox` of package `sp` [4, 1] have been implemented to get back the same results for trajectories.

```
> proj4string(B)

[1] "+proj=longlat +ellps=WGS84"

> coordinates(A1)

     x y
[1,] 7 7
[2,] 6 7
[3,] 5 6
[4,] 5 5
[5,] 4 5
[6,] 3 6
[7,] 3 7

> coordnames(Tr)

[1] "x" "y"

> bbox(A)

  min max
x   3   7
y   3   7
```

`spacetime` has been added a slightly modified version of the `bbox` method which does not constrain to space, but also considers time. Compare ...

```
> bbox(Tr)

  min max
x   1   7
y   1   7
```

... to:

```
> stbox(Tr)

    x y                time
min 1 1 2013-09-30 01:00:12
max 7 7 2013-09-30 01:11:10
```

3.2 Selection

Retrieving and replacing attribute data of `Track`, `Tracks` and `TracksCollection` objects can be obtained by using one of the base methods `[`, `[[`, `$`, `[[<-` and `$<-`. Although one may access the attributes through the slots directly, it is highly recommended not to do so, but use the appropriate selection method. The following code snippet showcases the broad range of selection options:

Select the first two `Tracks` objects of a `TracksCollection`, return an object of class `TracksCollection`:

```
> class(Tr[1:2])
> dim(Tr[1:2])
```

Select the second `Tracks` object of a tracks collection. Returns an object of class `Tracks`:

```
> class(Tr[2])

[1] "Tracks"
attr(,"package")
[1] "trajectories"

> dim(Tr[2])

    tracks geometries
         2         12
```

Select the first track of the second `Tracks` object of a `TracksCollection`. Returns an object of class `Track`:

```
> class(Tr[2][1])

[1] "Track"
attr(,"package")
[1] "trajectories"

> dim(Tr[2][1])

geometries
         6
```

Select tracks 1 and 2 of the first `Tracks` object as well as track 2 of the second `Tracks` object of a `TracksCollection`, return an object of class `TracksCollection`.

```
> class(Tr[list(1:2, 2)])

[1] "TracksCollection"
attr(,"package")
[1] "trajectories"
```

```
> dim(Tr[list(1:2, 2)])

       IDs     tracks geometries
         2          3         18
```

Select any tracks of a tracks collection that intersect `Spatial` object `Muenster`.

```
> Tr[Muenster]
```

Select attribute `co2` of a `TracksCollection`, either by

```
> class(Tr[["co2"]])

[1] "numeric"

> length(Tr[["co2"]])

[1] 24
```

or by

```
> class(Tr$co2)

[1] "numeric"

> length(Tr$co2)

[1] 24
```

Add or replace an attribute of a tracks collection, by

```
> Tr[["distance"]] = Tr[["distance"]] * 1000
```

or by

```
> Tr$distance = Tr$distance * 1000
```

3.3 Coercion

The implementation comes with a wealth of coercion methods, wich allow for converting objects of class `Track`, `Tracks` and `TracksCollection` to a variety of other classes. All available options are documented in table 1.

3.4 Plotting

Tracks can be plotted using either the `plot`, the `stplot` or the `stcube` method. While the first two give two-dimensional plots, which greatly fulfill their purpose, the latter one facilitates decent space-time cube representations of tracks, which leverage the third dimension. Figure 1 shows the spatial distribution of a tracks collection, whereas figure 2 depicts the CO2 consumption over time for one and the same object.

Class	Track	Tracks	TracksCollection
segments	Yes	Yes	Yes
data.frame	Yes	Yes	Yes
xts	Yes	Yes	Yes
Spatial	Yes	Yes	Yes
Line	Yes	No	No
Lines	Yes	Yes	No
SpatialLines	Yes	Yes	Yes
SpatialPointsDataFrame	Yes	Yes	Yes
SpatialLinesDataFrame	No	Yes	Yes
STIDF	Yes	Yes	Yes

Table 1: Available Coercion Options

3.5 Analysis

The update implements the methods `over` and `aggregate` for `Track`, `Tracks` and `TracksCollection` objects to provide the same functionality as is provided by packages `sp` and `spacetime`. In addition, a further method has been added to allow for generalising tracks by either space, time or a freely selectable number of segments. The points of a segment are wrapped up in a `SpatialLines` object with `time` and `endTime` reflecting the start and end time of the segment. The attributes are aggregated per segment. The following code snippet depicts the main options:

```
> # Generalise a track into 5 minute intervals. Use max() as the
> # aggregation function.
> generalize(B, max, timeInterval = "2 min")
> # Generalise a track into 200 distance units (usually metres).
> generalize(A2, distance = 200)
> # Generalise a track into n segments with each segment consisting of
> # two points.
> generalize(Tr, min, n = 2)
> # Simplify the given geometries using the Douglas-Peucker algorithm
> # with tolerance value 2.
> generalize(A, timeInterval = "3 min", tol = 2)
> # Keep the middle point of each segment rather than generalising to
> # objects of class "SpatialLines".
> generalize(A1, n = 3, toPoints = TRUE)
```

4 Demo

The package ships with two demos looking at trajectories while using two different datasets. The *Tracks* demo is based on the Geolife GPS trajectory dataset, which emerged from the (Microsoft Research Asia) Geolife project

```
> plot(Tr, col = 2, axes = TRUE)
```

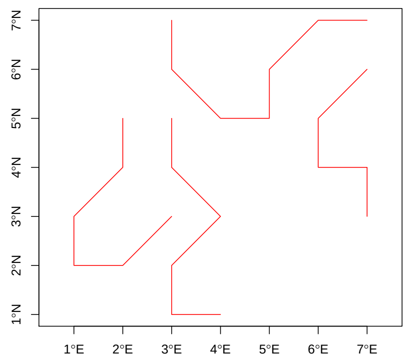

Figure 1: Spatial Distribution of a Tracks Collection

[5, 7, 6]. The *stcube* demo, instead, uses trajectories of the enviroCar project at `www.envirocar.org` and plots them in a space-time cube. The demos can be loaded as follows:

```
> demo("Track")
> demo("stcube")
```

Below, a small snippet of the *stcube* demo is shown:
A space-time cube can be shown by either

```
> A3 = importEnviroCar("528cf1a3e4b0a727145df093")
> stcube(A3, showMap = TRUE, col = "red")
```

or, avoiding loading the data from the web site

```
> data(A3)
> stcube(A3, showMap = TRUE, col = "red")
```

References

[1] R. S. Bivand, E. Pebesma, and V. Gómez-Rubio. *Applied Spatial Data Analysis with R*. Springer, 2013.

[2] C. Calenge. The package adehabitat for the R software: A tool for the analysis of space and habitat use by animals. *Ecological Modelling*, 197:1035, 2006.

```
> stplot(Tr, attr = "co2", arrows = TRUE, lwd = 3, by = "IDs")
```

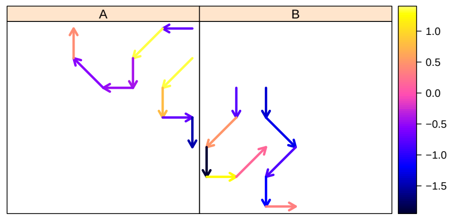

Figure 2: CO2 Consumption Over Time

[3] E. Pebesma. spacetime: Spatio-Temporal Data in R. *Journal of Statistical Software*, 51(7):1–30, 2012.

[4] Edzer J. Pebesma and Roger S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, 2005.

[5] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Ma. Understanding Mobility Based on GPS Data. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 312–321, 2008.

[6] Y. Zheng, X. Xie, and W. Ma. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[7] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, pages 791–800, 2009.