Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

trajectories: Classes and Methods for Trajectory Data

M. Mehdi Moradi
University of Jaume I

Edzer Pebesma
University of Münster

Jorge Mateu
University of Jaume I

Abstract

The package **trajectories** presents different classes and methods to handle and summarise trajectory data in R. It also provides model fitting to moving objects using time series modelling in the package **forecast**. Moreover, when dealing with a list of moving data that overlap in time, **trajectories** develops some statistical methodologies which provide users with some information about the behaviour of objects over time and with respect to each other. Using some methodologies for spatial point patterns, available in the package **spatstat**, we present an estimator for the intensity function of trajectory patterns which highlights the more visited streets, dense paths, etc. Based on second-order summary statistics for spatial point patterns, **trajectories** proposes a variability area which shows the variation of the type of interaction between moving objects over time. We also discuss the discrepancy between the estimated intensity and the expected intensity per area per time. Our methods are applied to a taxi trajectory data from Beijing, China.

Keywords: Chi-map, Distance, Intensity, R, Second-order characteristics, Spatio-Temporal, Taxi Movements, Trajectory.

1. Introduction

Modern data collection techniques allow tracking objects continuously. This means that we do not only know the current location of a moving object, but we also track the objects over time. A set of some tracks from different moving objects may be considered a trajectory pattern. Another example is that in which we record the location of a group of moving objects (e.g., cars, pedestrians, animals) within a regular/irregular time sequence. Güting and Schneider (2005) focused on moving objects databases and extended database technology to deal with moving objects. Challa, Morelande, Musicki, and Evans (2011) presented an introduction to the field of object tracking and provided solid foundation to the collection of diverse algorithms developed by academics, scientific researchers and engineers. Hanks,

Hooten, Alldredge *et al.* (2015) proposed a continuous-time, discrete-space (CTDS) model for animal movement. Niu, Blackwell, and Skarin (2016) considered a multivariate Ornstein Uhlenbeck diffusion process to model the movement of animals in continuous time. Russell, Hanks, and Haran (2016) introduced an approach that models dependent movement by augmenting a dynamic marginal movement model with a spatial point process interaction function within a weighted distribution framework. Hooten and Johnson (2017) presented a natural basis function approach to constructing appropriate covariance models for movement processes. There are already some R packages available in CRAN Task View: Handling and Analyzing Spatio-Temporal Data to handle moving objects. Most of them are focused on handling and analysing animal movements such as **adehabitatLT** (Calenge 2006), **tripEstimation** (Sumner, Wotherspoon, and Hindell 2009), **argosfilter** (Freitas 2012), **V-Track** (Campbell, Watts, Dwyer, and Franklin 2012), **animalTrack** (Farrell and Fuiman 2013), **BBMM** (Nielson, Sawyer, and McDonald 2013), **bcpa** (Gurarie 2014), **BayesianAnimalTracker** (Liu 2014), **TrackReconstruction** (Battaile 2014), **mkde** (Tracey, Sheppard, Zhu, Sinkovts, Chourasia, Lockwood, and Fisher 2014), **SimilarityMeasures** (Toohey 2015), **smam** (Yan and Pozdnyakov 2016), **trip**(Sumner 2016), **moveHMM** (Michelot, Langrock, and Patterson 2016), **FLightR** (Rakhimberdiev, Saveliev, Piersma, and Karagicheva 2017). In particular, the package **adehabitatLT** (Calenge 2006) provides tools to simulate trajectories using a Brownian motion, correlated Random walks and Levy walks. Toohey (2015) presented four different similarity measures in **SimilarityMeasures**. Michelot *et al.* (2016) in **moveHMM** provided animal movement modelling using hidden Markov models. Using multiple regression, **fishmove** (Radinger and Wolter 2014) provides functions to predict fish movement parameters. The R package **trackeR** (Frick and Kosmidis 2017) provides infrastructure for running and cycling Data.

However, to the best of our knowledge, R is still missing a complete set of generic data structures and methods to effectively analyse trajectories without being limited to a particular domain. Figure 1 shows routes passed by 5 different taxis on the 4th of Feb 2008 in Beijing, China. Each taxi has a different start/end time. To avoid having a complicated plot, we have only displayed 5 tracks; the entire dataset will be analysed in Section 5.

The entire dataset is stored in the R package **taxidata** and can be installed through the following code.

```
R>  # install.packages("taxidata",
R>  #  repos = "http://pebesma.staff.ifgi.de",type = "source")
```

Figure 1 is generated using the following lines of code.

```
R> library("trajectories")
R> # library("taxidata")
R> # Beijing <- taxidata
R> # Beijing <- Beijing[1:2000]
R> # Z <- lapply(X=1:length(Beijing), function(i){
R> #   q <-  cut(Beijing[[i]], "day", touch = F)
R> #   return(q@tracks[[3]])
R> # })
R> # plot(Z[[21]],xlim=c(420000,470000),ylim=c(4390000,4455000),lwd=2)
R> # plot(Z[[26]],add=T,col="orange",lwd=2)
R> # plot(Z[[20]],add=T,col=2,lwd=2)
```

```
R> # plot(Z[[12]],add=T,col=3,lwd=2)
R> # plot(Z[[15]],add=T,col=4,lwd=2)
```

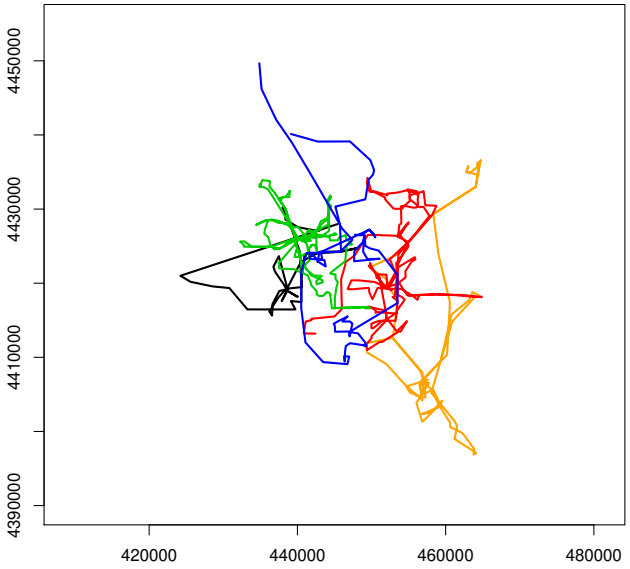

Figure 1: Trajectory pattern containing tracks of five taxis on the 4th of Feb 2008 in Beijing, China. Each color represents a different taxi track.

Studying the behaviour of moving objects over time and their interaction, either between objects or with environment, plays a crucial role in understanding how they use space and more importantly how they interact each other. Moving objects are moving within a particular area over time, thus a snapshot of a trajectory pattern might be seen as a spatial point pattern. This aspect then empowers us to study the behaviour of moving objects within space and over time. A set of locations, usually non-uniformly distributed within a certain region, can be considered as a realisation of a spatial point process. Analysis of spatial point processes has been widely discussed in the literature (Møller and Waagepetersen 2003; Illian, Penttinen, Stoyan, and Stoyan 2008; Diggle 2013; Baddeley, Rubak, and Turner 2015). The R package **spatstat** (Baddeley and Turner 2005; Baddeley *et al.* 2015) provides different tools for statistical analysis, model-fitting, simulation and tests on spatial point patterns. Diggle (2013) has broadly considered the details of spatio-temporal point processes. Application of such processes can be found in traffic management, geography, forestry, ecology, epidemiology, seismology, astronomy and criminology.

This paper describes a collection of tools provided by the R package **trajectories** to handle, simulate and statistically analyse movement data regardless of the domain, converting a trajectory pattern into a list of point patterns based on regular timestamps. We here propose different functions to analyse the behaviour of objects over time and how they use space and also how they interact with each other. The type of interaction between objects may vary over time. The effect of the environment on the type of interaction might also be of

interest. Therefore, using the literature of spatial point processes, the R package **trajectories** opens up a new way of thinking about trajectory datasets. We define different classes to handle trajectories, and different functions for simulating and performing exploratory data analysis. We also borrow first and second-order characteristics from the literature of spatial point processes and, by adapting them to trajectory patterns, we aim at highlighting the most frequently used routes within the studied area together with disclosing the type of interaction between the objects over time. Moreover, the **trajectories** package fits time series models to the spatial coordinates of a trajectory dataset.

The plan of the paper is the following. Section 2 presents some background and definitions. Section 3 describes different classes of trajectories and explains different methods to summarise trajectory data. Simulation and model fitting of trajectory data is described in Section 4. Section 5 develops some statistical methods to perform exploratory data analysis and they are demonstrated through the taxi trajectory data from Beijing, China. The paper ends with some final conclusions in Section 6.

2. Setup and definition

Spatial point locations are usually analysed through spatial point processes, see Møller and Waagepetersen (2003); Daley and Vere-Jones (2007); Baddeley *et al.* (2015). Such events might also be labelled with the temporal instant, demanding then a spatio-temporal analysis (Diggle 2013). Let X be a spatial point process in $\mathbb{R}^2$ with intensity function $\lambda(\cdot) > 0$ and second-moment intensity $\lambda^2(\cdot,\cdot)$. For any Borel-measurable real function $f(x)$ where $\int_{\mathbb{R}^2} |f(u)|\lambda(u)\mathrm{d}u < \infty$, it is satisfied

$$\mathbb{E}\left[\sum_{x\in X} f(x)\right] = \int_{\mathbb{R}^2} f(u)\lambda(u)\mathrm{d}u, \qquad u \in \mathbb{R}^2. \tag{1}$$

Equation 1 has been broadly used in the literature of spatial point processes and it is called *Campbell's formula*. Generally speaking, $\lambda(\cdot)$ is the expected number of points per unit area. Recall that if $\lambda(\cdot)$ is constant then X is homogeneous, otherwise X is called an inhomogeneous point process. One of the first and an important step in studying point processes is to investigate the intensity function. Estimating the intensity function of spatial point processes has been largely discussed (Diggle 1985; Jones 1993; Chiu, Stoyan, Kendall, and Mecke 2013). We denote a realisation of the point process X with n points as $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$; note that n is not fixed in advance. In point process analysis, we only consider a single realisation of the underlying point process, and then analyse the behaviour of that realisation. However, we might be able to collect the location of spatial events according to regular or irregular timestamps over time. We then have a series of points over time per single object which can be considered as a single track. For instance, recording the location of a taxi over particular times results in the route passed by that taxi. We here consider a *trajectory pattern* as a point pattern which lives in $\mathbb{R}^2$ but moves over time. This allows us to adapt the point process statistical methodology into the literature of trajectory patterns.

definicijatraj

Definition 2.1 *A trajectory pattern is a dataset which provides observed tracks (s_i) of a set of moving objects such as cars, humans, etc over a finite time period T. We denote a trajectory pattern consisted by $n > 0$ single tracks as $S = \{s_i : s_i \subset \mathbb{R}^2,\ i = 1, \ldots, n\}$, that is a countable set of tracks. Each s_i is itself a countable set of points, e.g., $s_i = \{x_1^{s_i}, x_2^{s_i} \cdots, x_{m_i}^{s_i}\}$ with*

$i = 1, \cdots, n$ where each s_i consists of $m_i < \infty$ points that are associated with an increasing set of time stamps $t_1, ..., t_{m_i}$, $t_j > t_{j+1} \forall j$.

We point out that the length of each of the tracks s_i is not necessarily the same for all tracks. In other words, they might have different start/end times. Each single track s_i represents the movement of a moving object within a finite time/area. It is usually supposed that locations of a moving object are recorded in regular timestamps. However, if timestamps are not regular, one can still interpolate the locations in regular timestamps. Therefore, each single track in S might be seen as a set of points corresponding to the considered timestamps. Discretising all tracks of S according to regular timestamps results in a list of point patterns (one per each time) which enables us to consider a trajectory pattern as a point pattern which is changing over time. Therefore, using point pattern methodology, one can study the behaviour of moving objects over time. For instance, the spatially varying distribution of objects and the type of interaction between them over time can be of interest. Assume that the trajectory pattern S is observed within the time period T, thus discretising T into a time sequence $\{t_i : t_i \subset T, i = 1, \ldots, k \quad \text{where} \quad t_i < t_j \quad \text{if} \quad i < j\}$ generates a collection of spatial point patterns, say, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ $(k > 1)$. Details are provided in Section 5. Nevertheless, one may still consider S as a point process on $\mathbb{R}^{2 \otimes k}$, where $\mathbb{R}^{2 \otimes k}$ means $\mathbb{R}^2 \times \cdots \times \mathbb{R}^2$ for k times when the length of timestamps is k. We do not discuss this other approach here.

3. Classes and methods

In this section, we review different classes of trajectories to handle movement data in R. These classes were initially defined in the R package **spacetime** (Pebesma 2012). Before moving into the details and start analysing trajectory patterns, we load the package with:

```
R> library("trajectories")
```

3.1. Track

The class '`Track`' represents a single track followed by a person, animal or an object. Instances of this class are meant to hold a series of consecutive location/timestamps that are not interrupted by another activity. The class contains five slots: @sp to store the spatial points, @time to store the corresponding 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 (covariate information) and @connections to keep a record of attribute data between points (e.g., distance, duration, speed and direction). A '`Track`' object can be created out of an '`STIDF`' (see Pebesma (2012)) object as follows

```
R> set.seed(10)
R> library("spacetime")
R> library("sp")
R> t0 = as.POSIXct(as.Date("2013-09-30",tz="CET"))
R> x = c(7,6,5,5,4,3,3)
R> y = c(7,7,6,5,5,6,7)
```

```
R> n = length(x)
R> t = t0 + cumsum(runif(n) * 60)
R> crs = CRS("+proj=longlat +ellps=WGS84") # longlat
R> stidf = STIDF(SpatialPoints(cbind(x,y),crs), t,
+                  data.frame(co2 = rnorm(n,mean = 10)))
R> A1 = Track(stidf)
R> A1

An object of class Track
7points
bbox:
  min max
x   3   7
y   5   7
Time period: [2013-09-30 02:00:30, 2013-09-30 02:02:31]
```

additionalda

Figure 2 shows the plot of track A1 passed by person A. By default, distance, duration, speed and direction are computed as the connections data. Optionally, a data frame containing additional connections data (covariates) and/or a custom function for calculating the data of segments between consecutive points can be passed.

Moreover, and using the function `as.Track`, one can create an object of class '`Track`' if spatial coordinates and corresponding times are provided. Additional information can also be passed to the function `as.Track` using an argument `covariate`.

```
R> x <- runif(10,0,1)
R> y <- runif(10,0,1)
R> date <- seq(as.POSIXct("2015-1-1 0:00"), as.POSIXct("2015-1-1 9:00"),
+                 by = "hour")
R> records <- as.data.frame(rpois(10,5))
R> as.Track(x,y,date,covariate = records)

An object of class Track
10points
bbox:
         min       max
x 0.23958913 0.8382877
y 0.09308813 0.9546536
Time period: [2015-01-01, 2015-01-01 09:00:00]
```

3.2. 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). An object of class '`Tracks`' can be created by:

```
R> plot(A1)
```

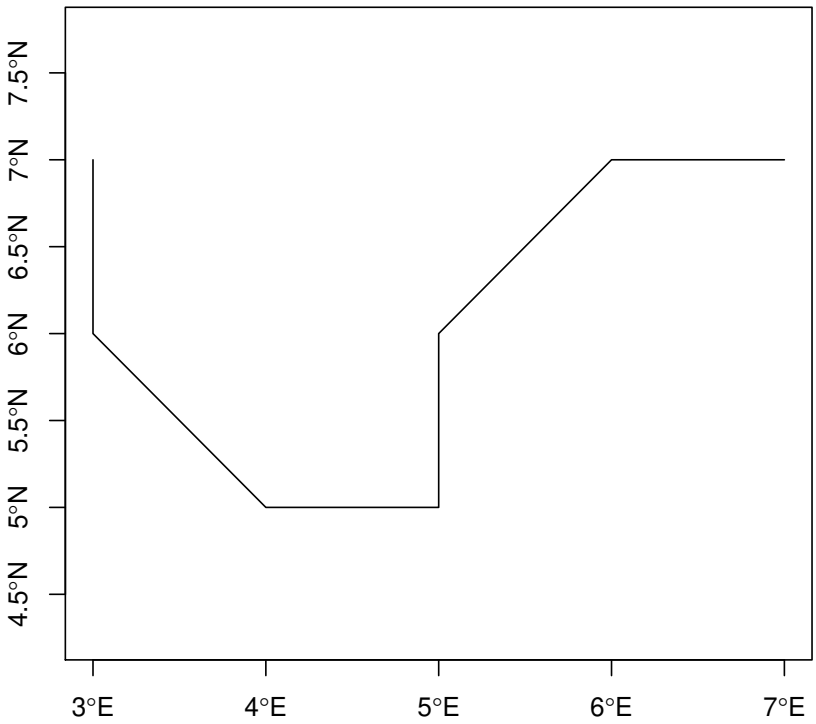

Figure 2: Single track A1 passed by person A.

```
R> x = c(7,6,6,7,7)
R> y = c(6,5,4,4,3)
R> n = length(x)
R> t = max(t) + cumsum(runif(n) * 60)
R> stidf = STIDF(SpatialPoints(cbind(x,y),crs), t,
+                  data.frame(co2 = rnorm(n,mean = 10)))
R> A2 = Track(stidf)
R> # Tracks for person A:
R> A = Tracks(list(A1=A1,A2=A2))
R> A

An object of class Tracks
2 tracks followed by a single object
```

where `A1` and `A2` are 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. As for the 'Track' method, a data frame and/or a custom function can be passed to expand the default data.

3.3. TracksCollection

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 per each object). A 'TracksCollection' object can be created by:

```
R> # person B, track 1:
R> x = c(2,2,1,1,2,3)
R> y = c(5,4,3,2,2,3)
R> n = length(x)
R> t = max(t) + cumsum(runif(n) * 60)
R> stidf = STIDF(SpatialPoints(cbind(x,y),crs), t,
+                  data.frame(co2 = rnorm(n,mean = 10)))
R> B1 = Track(stidf)
R> # person B, track 2:
R> x = c(3,3,4,3,3,4)
R> y = c(5,4,3,2,1,1)
R> n = length(x)
R> t = max(t) + cumsum(runif(n) * 60)
R> stidf = STIDF(SpatialPoints(cbind(x,y),crs), t,
+                  data.frame(co2 = rnorm(n,mean = 10)))
R> B2 = Track(stidf)
R> # Tracks for person B:
R> B = Tracks(list(B1=B1,B2=B2))
R> Tr = TracksCollection(list(A=A,B=B))
R> Tr

An object of class TracksCollection
2 collection of tracks followed by 2  object
```

where A and B are 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. As for the 'Track' and 'Tracks' methods outlined above, a data frame and/or a custom function can be passed to expand the default data.

3.4. segments

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

Figure 3 shows the classes and their connection. We point out that classes 'STIDF','STI' and 'ST' belong to the package **spacetime** inherently .

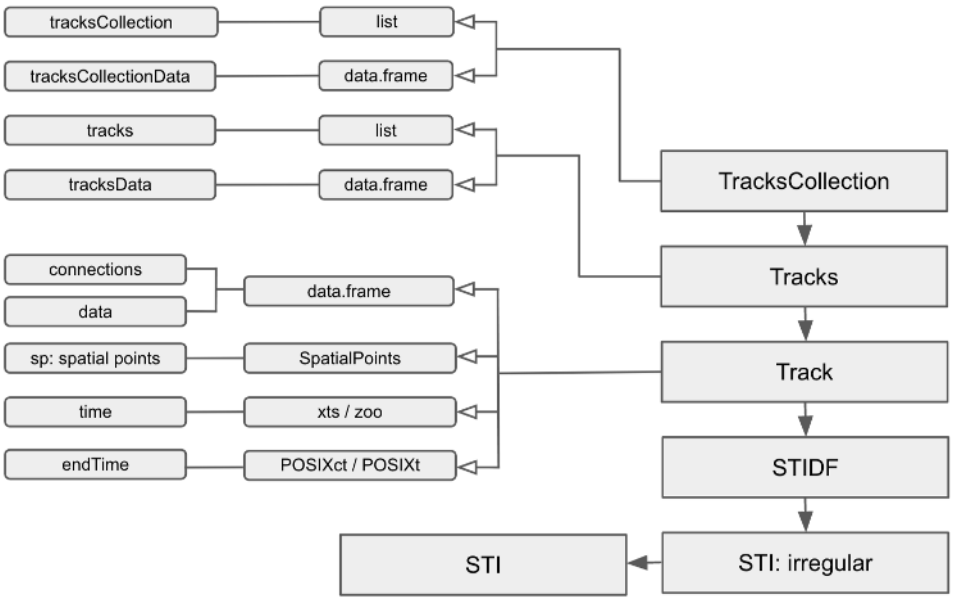

Figure 3: Classes for trajectory data in the package **trajectories**. Solid arrows denote inheritance. Arrows show the corresponding slot's class and slot's names are displayed using lines accordingly.

A wealth of methods have been implemented to cover the most frequently used use cases. Table ?? lists some of the methods applied to the objects of classes `Track`, `Tracks` and `TracksCollection`. Apart from those listed in Table ??, attribute data can be obtained or replaced by using `[]`, `[[]]`, `@` and `$`.

The use of some methods in Table ?? is shown in the following lines of code. Figure 4 shows the trellis plot of object `Tr` from class '`TracksCollection`' which is previously created. We point out that we have used slot `data` of the corresponding tracks as attributes to see their changes over time.

```
R> dim(A1)

geometries
         7

R> dim(B1)

geometries
         6

R> stbox(A1)

    x y                 time
min 3 5 2013-09-30 02:00:30
max 7 7 2013-09-30 02:02:31
```

tablica metoda

Method	Operation
`dim`	Returns the number of spatial points of any track
`summary`	Summarises the internal information
`proj4string`	Retrieves projection attributes
`coordinates`	Retrieves the coordinates of spatial locations
`coordnames`	Retrieves coordinate names of fixes
`bbox`	The box (window) which contains the objects
`stbox`	The spatio-temporal box (window) which contains the objects
`aggregate`	Spatially aggregate track properties (coercing fixes to points)
`compare`	Compares two '`Track`' objects: for the common time period
`dists`	Compares two '`Tracks`' using the mean distance, the Frechet distance,etc
`downsample`	Remove fixes from a '`Track`', starting with the most densely sampled ones
`frechetDist`	Compute Frechet distance between two '`Track`' objects
`stcube`	Draw a space-time cube
`stplot`	Create trellis plot for '`TracksCollection`' objects
`generalize`	Resample '`Track`' to lower freqency or minimal distance
`cut`	Obtain ranges of space and time coordinates

Table 1: Methods implemented in the package **trajectories** for objects from class '`Track`','`Tracks`' and '`TrackCollection`'.

```
R> downsample(A1,B1)

An object of class Track
6points
bbox:
  min max
x   3   7
y   5   7
Time period: [2013-09-30 02:00:30, 2013-09-30 02:02:31]
```

4. Simulation and model fitting

4.1. Trajectory simulation

Simulating trajectory patterns can be a useful tool to imitate true models and understand their behaviour. The package **trajectories** allows simulating tracks using `rTrack`, `rTracks`, `rTracksCollection` where `rTrack()` generates a single track, `rTracks()` simulates a collection of tracks assumed to be passed by a single object and `rTracksCollection` is used to simulate a set of tracks passed by different objects. By default, these functions do not consider any box (or window) for the track to be simulated in and consider `origin=c(0,0)` as the origin of the track. However, one can still restrict the track to a desirable closed box using the argument `bbox`. If `transform=TRUE` and no `bbox` is given, then `rTrack` transforms the track to the default box $[0,1] \times [0,1]$, where in this case the origin is a random point in the default

```
R> stplot(Tr, attr = "co2", arrows = TRUE, lwd = 3, by = "IDs",cex.axis=2)
```

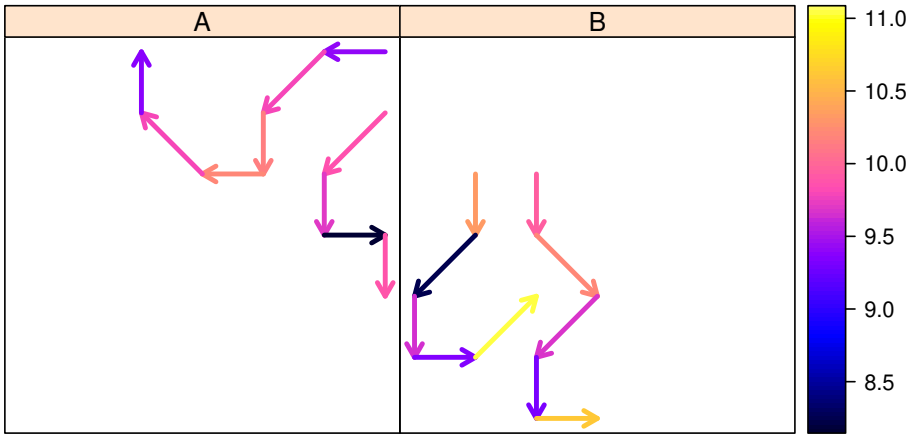

Figure 4: Co2 consumption over time.

box. If a default box `bbox` (e.g., `m` in the following example) is given and `transform=TRUE`, then `origin` is a random point in `bbox` and the final track is also transformed into `bbox`. The function `rTrack` simulates tracks with a predefined number of points per track (indicated as `n` in the code with default 100). However if `nrandom=TRUE` then it simulates a track with a random number of points based on a Poisson distribution with parameter `n`. An example of these functions is the following:

```
R> set.seed(10)
R> x <- rTrack();x

An object of class Track
100points
bbox:
        min       max
x -53.38677 -3.232289
y -53.08787 -1.594295
Time period: [1970-01-01, 1970-01-01 01:39:00]

R> y <- rTrack(transform = T);y

An object of class Track
100points
bbox:
  min max
x   0   1
```

```
y   0   1
Time period: [1970-01-01, 1970-01-01 01:39:00]

R> m <- matrix(c(0,10,0,10),nrow=2,byrow = T)
R> w <- rTrack(bbox = m,transform = T);w

An object of class Track
100points
bbox:
  min max
x   0  10
y   0  10
Time period: [1970-01-01, 1970-01-01 01:39:00]

R> z <- rTrack(bbox = m,transform = T,nrandom = T);z

An object of class Track
108points
bbox:
  min max
x   0  10
y   0  10
Time period: [1970-01-01, 1970-01-01 01:47:00]
```

Figure 5 shows four different random tracks: `x` is a random track with all defaults, `y` is a random track transformed to a unit box, `w` is a random track transformed to the box $[0, 10] \times [0, 10]$, and `z` is a simulated track in a same box as `w` but with a random number of points. The number of points in `w` is 100 whereas `z` is constituted by 108 points.

4.2. Model fitting

The behaviour of a track might also be studied using available tools for time series modelling. However, obtaining a proper model is extremely important as it highlights the underlying structure of the series, and the fitted model can be used for future forecasting. The R package **trajectories** can fit ARIMA models to movement data. Using R package **forecast**, the function `auto.arima.Track` fits arima models to the spatial coordinates of an object of class '`Track`'. Note this is applicable to individuals. See example below.

```
R> library("forecast")
R> data("A3")
R> auto.arima.Track(A3)

Arima model fitted to x-coordinate: ARIMA(0,2,1)
Arima model fitted to y-coordinate: ARIMA(1,1,1) with drift
```

5. Exploratory data analysis

```
R> par(mfrow=c(2,2),mar=rep(2.2,4))
R> plot(x,lwd=2,main="x");plot(y,lwd=2,main="y")
R> plot(w,lwd=2,main="w");plot(z,lwd=2,main="z")
```

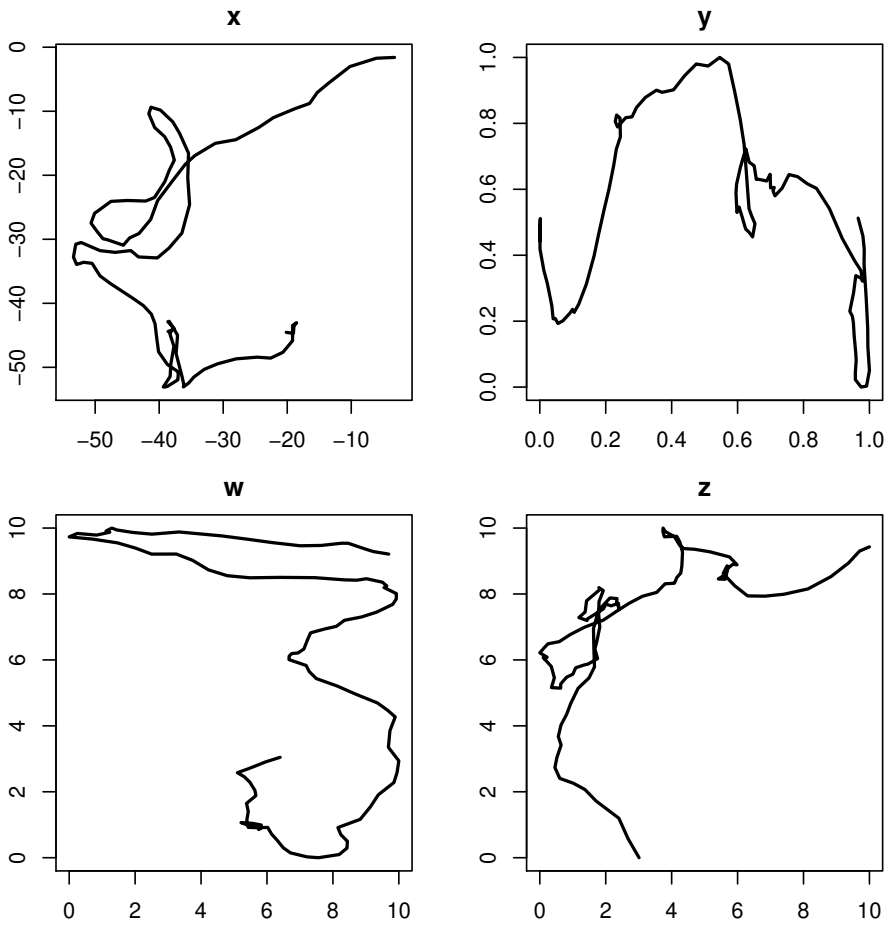

Figure 5: Simulated random tracks using `rTrack`. `x` is a random track with all defaults. `y` is a random track transformed to a unit box. `w` is a random track transformed to the box $[0, 10] \times [0, 10]$ and `z` is simulated in the same box as `w` but with a random number of points.

This section presents some statistical methods, implemented in the R package **trajectories**, to analyse the behaviour of trajectory patterns. A single plot of trajectories pattern might not display interesting information, and if the pattern contains too many tracks, it then needs some analysis to summarise and reveal concealed information. In particular, one may be interested in discovering the more visited streets within a city. Other interesting findings could be the type of interaction between moving objects over time. In short, having a trajectory pattern, we might be interested in answering the following questions:

1. How does the average distance between objects change over time?

2. How is the spatially varying distribution of objects?

3. How do moving objects interact with each other? Does their interaction vary over time?

4. How does the spatially varying distribution of objects vary over time?

5.1. Data

We considered a sample of the T-Drive trajectory dataset that contains one-week trajectories of 10357 taxis during the period of Feb. 2 to Feb. 8, 2008, within Beijing, China. T-drive is a smart driving direction services based on the global positioning system (GPS) trajectories of a large number of taxis. The GPS-equipped taxis are mobile sensors probing the traffic flows on road surfaces. So, the taxi trajectories contain the information of both human knowledge of experienced drivers and traffic patterns. The total number of points in this dataset is about 15 million and the total distance of the trajectories reaches up to 9 million kilometres. For more details about the data, see Yuan, Zheng, Zhang, Xie, Xie, Sun, and Huang (2010); Yuan, Zheng, Xie, and Sun (2011).

We here point out some useful information about the dataset:

- 21 taxis have no information recorded.
- Regardless of taxis with no information, there are 4694 taxis with less than 10 recorded locations in at least one day.
- There are tracks with some jumps to the outside of the studied area and it may be caused by lack of GPS accuracy so that wrong locations have been removed. These locations might later be recovered by interpolation.

We thus analyse the cleaned dataset which is based on moving data of 5642 taxis. The map of the studied area is displayed in Figure 6. It is seen that the metropolitan area of Beijing is almost located in the centre of the map while there are some other townships, airports in the countryside of Beijing.

In the following, we present the implemented methods in **trajectories** by applying them to the taxi movement dataset in Beijing, China.

5.2. Distance analysis

A simple way to get into the nature of movement data is to study the distance between objects. The function `dists` provides users with calculating the distance between a pair of objects of class 'Tracks'. This considers the distance between tracks when they overlap in time. The output is a matrix with distances between each pair of tracks or 'NA', if they do not overlap in time. A function to calculate distances can be passed to `dists`, such as `mean`, `sum`, `frechetDist`, etc.

```
R> ## create Tracks objects
R> # tracks1 <- Tracks(list(Beijing[[1]], Beijing[[2]]))
R> # tracks2 <- Tracks(list(Beijing[[2]], Beijing[[1]]))
R> # dists(tracks1, tracks2,mean)
```

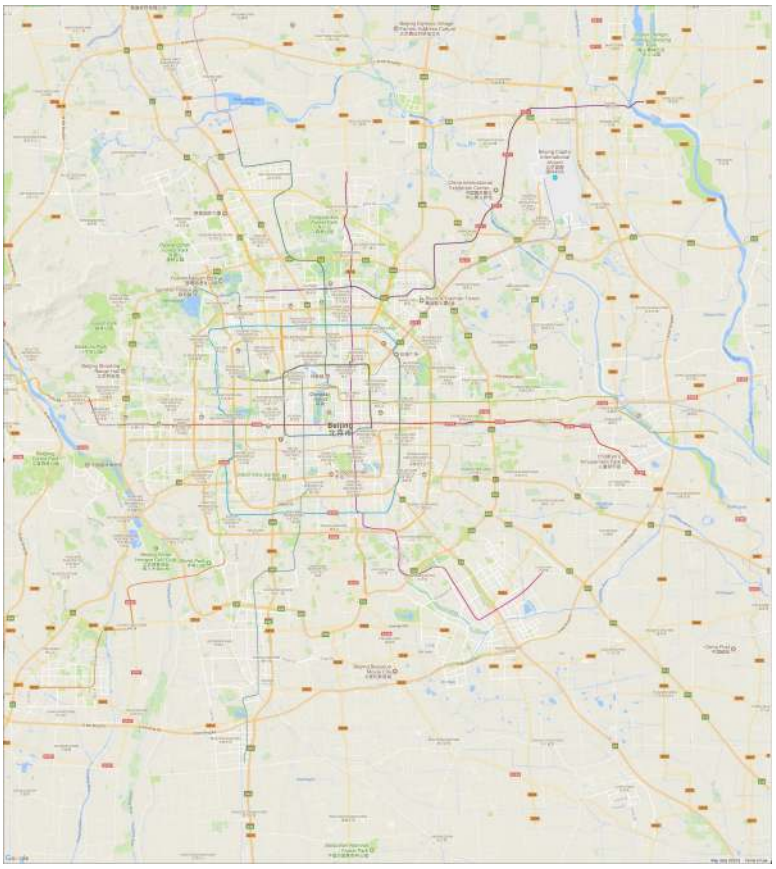

Figure 6: Map of the studied area in Beijing, China.

Average distance over time

studiyingp

The distance between objects over time might discover some interesting information. Studying pairwise distances over time can somehow reveal the type of interaction between objects. Having a pattern of tracks, we may be able to see how moving objects interact each other over time. Moreover, this can highlight the crowded hours within a particular period of time. We here propose to look at average pairwise distances over time. To do so, one can imitate the following steps:

interpolacija"reko

1. Based on the time range of all tracks s_i, create a regular time sequence.
2. Interpolate each track s_i based on the created time sequence. For this purpose, the function `reTrack` can be used. It reconstructs each track s_i according to a desirable time sequence.
3. Discretise the trajectory pattern S to a collection of point patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. Note that the number of points in each pattern might be different.
4. For each $\mathbf{x}_i, i = 1, \ldots, k$, calculate pairwise distances between all data points.
5. Report the average of pairwise distances per each time.

Steps above are implemented in the function `avedistTrack`. In order to use `avedistTrack`, we only need to specify the argument `timestamp`. It then returns the average distance between objects based on that timestamps.

```
R>  # par(mfrow=c(1,2))
R>  # meandist <- avedistTrack(Beijing,timestamp = "20 mins")
R>  # plot(meandist,type="l",lwd=2,cex.axis=1.7,cex.lab=1.7)
R>  # distinframe <- data.frame(tsq=attr(meandist,"tsq"),dist=meandist)
R>  # dist3rd <- distinframe[substr(distinframe$tsq,start = 1,stop=10)==
R>  #                              "2008-02-03",]
R>  # plot(dist3rd$tsq,dist3rd$dist,type="l",xlab="time",
R>  #       ylab="average distance",lwd=2,cex.axis=1.7,cex.lab=1.7)
```

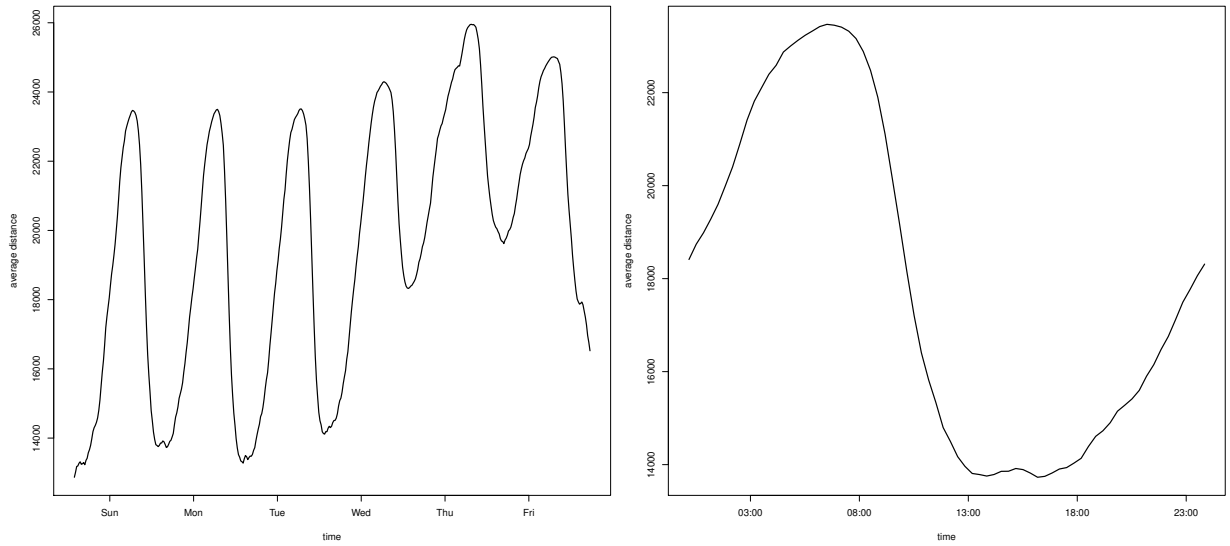

Figure 7: Average pairwise distance between taxis in Beijing, China. *Left*: Within the period $2-8$, Feb 2008. *Right*: During 3-rd of Feb 2008.

Figure 7 shows the average distance between taxis in Beijing. The left plot shows the average pairwise distance between taxis for all the week and it is easily seen that there is a daily trend. In order to see the more crowded hours within a day, we show the average pairwise distances during the 3-rd of Feb 2008 in the right plot in Figure 7. It can be seen that the crowded time comes between midday and $17:00$. Also, it shows how taxis are getting far from each other during night. Note that small average distances between taxis might be a sign of traffic during the corresponding hours. We point out that the reason of having larger average distance in the last two days might be the Chinese new year holidays.

As an alternative, one can use nearest neighbour distances instead of pairwise distances. But that might not distinguish patterns with different clusters, i.e., patterns with different sets of tracks concentrated in some particular subregions.

5.3. Movement smoothing

Aiming at analysing moving objects, it might be of interest to highlight the relationship of movement with space and time. This section is considered to perform smoothing over the length of movements per each consecutive time. Thus, it might reveal those areas with faster/slower movements. For this purpose, the function `Track.idw` performs inverse-distance weighted smoothing over a trajectory pattern by imitating the following steps:

1. Follow steps 1-3 in Section 5.2.
2. Using each consecutive point patterns, say $\mathbf{x}_o$ and $\mathbf{x}_d$, build $k-1$ segment patterns.
3. For each segment pattern, find the mid-point of segments, mark it with the length of the corresponding segment.
4. Using the marked mid-points, create $k-1$ marked point patterns where each mark represents the length of movement per location.
5. Apply the function `idw` from the package **spatstat** (Baddeley *et al.* 2015) to each marked point pattern where it does inverse-distance weighted smoothing.
6. Step 5 returns $k-1$ maps in which the average of them is the output of function `Track.idw`.

Mathematically speaking, if for each point pattern $\mathbf{x}_i$, data points $\{x_1, x_2, \ldots, x_{n_i}\}$ are marked by $\{l_1, l_2, \ldots, l_{n_i}\}$ then the smoothed value at an arbitrary location $u \in W$ is

$$\bar{g}(u) = \frac{1}{k-1}\left[\sum_{i=1}^{k-1} \frac{\sum_{j=1}^{n_i} w_j l_j}{\sum_{j=1}^{n_i} w_j}\right], \tag{2}$$

where

$$w_j = \frac{1}{(d(u, x_j))^p}, \tag{3}$$

in which d measures the distance between u and $x_j \in \mathbf{x}_i$, l_j is the corresponding mark of x_j that is the length of the corresponding segment to x_j, n_i is the number of points in the i-th pattern and p is an integer, being 2 as a default value. For details see Baddeley *et al.* (2015, Chapter 15).

In order to use `Track.idw`, we only need to set the argument `timestamp`. When collecting data, it may happen to record the location of objects which are actually stopped. Therefore and to not include them in the movement smoothing, there is an argument `epsilon` so that movements with length less than `epsilon` are not included in the computation. If no `epsilon` is set in the function, it then uses all segments. The following `code` generates an image of class '`im`' which reflects the smoothed movement of taxis per 20 minutes.

```
R>  # b <- Track.idw(Beijing,timestamp = "20 mins",epsilon=1000)
R>  # plot(b,main="",ribwid=0.04,ribsep=0.02,cex.axis=1.5)
```

koliko metaraprodju u 20

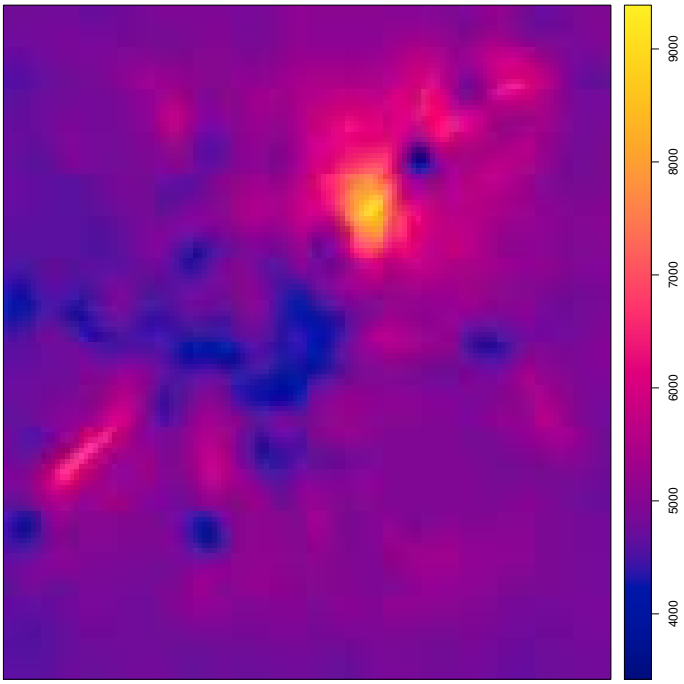

Figure 8: Movement smoothing for taxi data in Beijing, China based on `timestamp = "20 mins"` and movements with length longer than 1000 meters.

brzina kretanja

Figure 8 shows the movement smoothing for taxi data in Beijing according to the `timestamp= '20 mins"`. Here, we have not considered movements with length less than 1000 meters. In other words, we assume taxis with the length of movements less than 1000 meters per 20 minutes as stopped. This confirms that moving in the centre is slower than countryside/highways in Beijing, and in particular, it reveals some highways/freeways in which taxis are moving faster.

After smoothing the length of movements over space, we now turn to see the changes in the average of the length of movements over time. The function `avemove` measures the average length of movements passed by a collection of tracks based on a desirable timestamps. Now, we apply this to the taxi data in Beijing as follows

```
R>    # q <- avemove(Beijing,timestamp = "20 mins",epsilon=1000)
R>    # par(mfrow=c(1,2))
R>    # plot(q,type="l",lwd=2,cex.axis=1.7,cex.lab=1.7)
R>    # qdata <- data.frame(q,attr(q,"time"))
R>    # colnames(qdata) <- c("dist","startingtime")
R>    # q3rd <- qdata[substr(qdata$startingtime,start = 1,stop=10)=="2008-02-03",]
R>    # plot(q3rd$startingtime,q3rd$dist,type="l",xlab="time (hour)"
R>    #    ,ylab="average movement",lwd=2,cex.axis=1.7,cex.lab=1.7)
```

Figure 9 shows the average length of movements per 20 minutes by taxis in Beijing. The daily trend can be seen in the left plot, and the right plot shows that between midnight and early morning, the average length of movements is decreasing while from morning till noon there

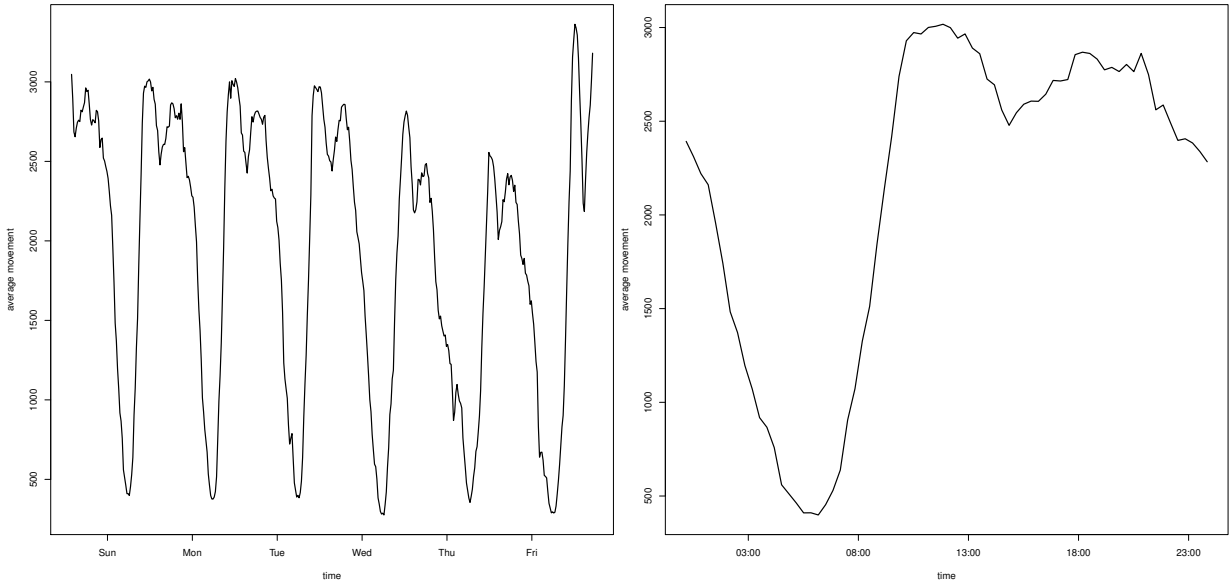

Figure 9: Average length of movements by taxis in Beijing, China versus time based on `timestamp = "20 mins"`, and movements with length longer than 1000 meters. *Left*: Within the period $2-8$ Feb 2008. *Right*: During the 3-rd of Feb 2008.

is an increase in the length of movements. In the afternoon, there can be seen a decrease in the average length of movements which might be caused by traffic.

intensityfu

5.4. Intensity function

An exploratory data analysis of point patterns often starts with estimating the intensity function $\lambda(\cdot)$ which reflects the mean number of points in different regions and may be seen as a "*heat-map*" for the events. A well-known method to estimate the intensity function $\lambda(\cdot)$ is by kernel smoothing. Diggle (1985) introduced a uniform edge-corrected kernel estimator

$$\widehat{\lambda}(u) = \frac{1}{c_W(u)} \sum_{i=1}^{n} \kappa(u - x_i), \qquad u \in W \tag{4}$$

and Jones (1993) proposed the alternative estimator

$$\widehat{\lambda}(u) = \sum_{i=1}^{n} \frac{\kappa(u - x_i)}{c_W(x_i)}, \qquad u \in W \tag{5}$$

where

$$c_W(u) = \int_W \kappa(u - v)\mathrm{d}v, \qquad u \in W \tag{6}$$

is the edge correction factor, and κ is a kernel function. We note that the estimator 4 is unbiased if the true intensity is uniform while the estimator 5 conserves mass, meaning that $\int_W \widehat{\lambda}(u)\mathrm{d}u = n$ where n is the number of points in the point pattern in question. Both estimators 4 and 5 can be computed using the R package **spatstat** for planar point patterns.

We here avoid mathematical definitions of a point process and assume that X is finite, i.e., any realisation is a point pattern with finite number of points, and the number of points in any subregion is a well-defined random variable (Baddeley *et al.* 2015). For technical details of point processes see Møller and Waagepetersen (2003); Daley and Vere-Jones (2007).

Being able to estimate the intensity function of planar spatial point patterns, we are here interested in estimating the intensity function for trajectory patterns. Such estimator can highlight well-traveled areas based on the tracks of moving objects (e.g., humans, cars) within the time period in question. We next propose an average intensity estimate using the following steps:

1. Follow steps $1-3$ as in Section 5.2.
2. For each $\mathbf{x}_i$, estimate the intensity function, say, $\lambda_i(\cdot)$.
3. The average over all estimated intensity functions $\widehat{\lambda}_i$ may be considered as an estimated intensity for the trajectory pattern S.

Mathematically speaking, and (for instance) using the intensity estimator 5, we propose

$$\widehat{\widehat{\lambda}}(u) = \frac{1}{k}\sum_{i=1}^{k}\widehat{\lambda}_i(u) = \frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{n_i}\frac{\kappa(u-x_j)}{c_W(x_j)}, \qquad u \in W, \tag{7}$$

as an estimator of the intensity of the trajectory pattern S. Intuitively, we interpret $\widehat{\widehat{\lambda}}(u)$ as the average expected number of points (objects) within the time period T in a small area around u. As a simple example, consider the movements of cars within a city in a particular day, the estimator 7 reveals the more dense streets, highways, freeways, etc in that day. Intensity estimator 7, possible to use both edge corrections, is implemented in the **trajectories** package using the function `density.list`. The function `density.list` builds the point patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ and pass them to the function `density.ppp` in the package **spatstat**.

We next turn to calculate the average estimated intensity of the taxi data using the estimator 7.

```
R>  # library("spatstat")
R>  # d <- density(Beijing,timestamp = "20 mins",bw.ppl)
R>  # par(mfrow=c(1,2),mar=rep(1,4))
R>  # plot(d,main="",ribwid=0.04,ribsep=0.02,cex.axis=1.7)
R>  # #focus on the center
R>  # w <- owin(c(440000,455000),c(4410000,4430000))
R>  # pps <- attr(d,"ppps")
R>  # npps <- lapply(X=1:length(pps),FUN = function(i){
R>  #   pps[[i]][w]
R>  # })
R>  #
R>  # centerimg <- lapply(X=1:length(npps),FUN = function(i){
```

```
R>  #   density(npps[[i]],bw.ppl(npps[[i]]))
R>  # })
R>  # fcenterimg <- Reduce("+",centerimg)/length(centerimg)
R>  #
R>  # plot(fcenterimg,main="",ribwid=0.04,ribsep=0.02,cex.axis=1.7)
```

Figure 10 shows the estimated intensity using the estimator 4 for both Beijing and its metropolitan area. The bandwidth has been selected using a likelihood cross-validation method and the function `bw.ppl` in **spatstat**. Other bandwidth selection methods can also be passed to `density.list`. Figure 10 highlights the most well-traveled areas in which those areas in the countryside with higher intensities (left plot) are some townships or airports. The right plot highlights the crowded routes within the centre of Beijing, China.

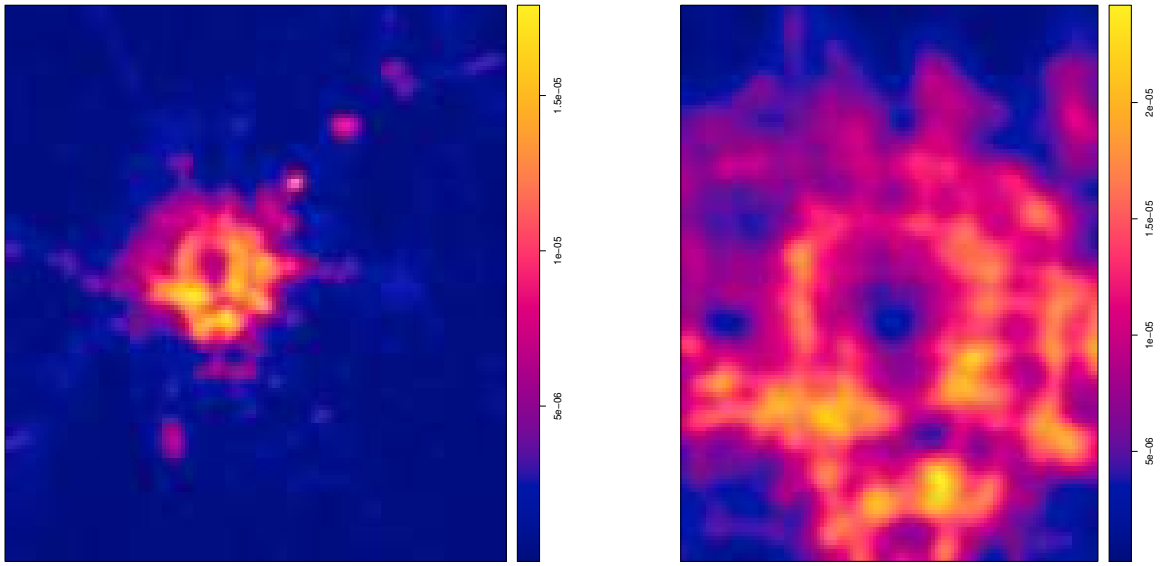

Figure 10: Estimated intensity function. Left: Beijing. Right: Beijing metropolitan area.

Voronoi esti

One may still think of adaptive intensity estimators such as the Voronoi estimator (Ord 1978; Barr and Schoenberg 2010). We here point out that as the estimator 7 is built based on an average of estimated intensities of a set of spatial point patterns, one can estimate each of $\lambda_i(\cdot)$ using adaptive estimators resulting in a final adaptive estimator for the corresponding trajectory pattern.

5.5. Chi maps

After discretising the trajectory pattern S to some point patterns and being able to estimate the individual intensity functions $\lambda_i(\cdot)$, one may think of discovering the areas with more/less events than the expected number. This motivates us to think of χ^2 statistics

$$\chi^2 = \frac{o - e}{\sqrt{e}}, \tag{8}$$

which measures the discrepancy between the expected number (e) and the observed number (o). This can be easily applied to the estimated intensity functions $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_k$ and in any time $t_i, i = 1, \ldots, k$, and as a result we can see where the estimated intensity differs from the expected intensity. For example, for a fixed time $t = t_1$,

$$e_{t_1}(u) = \frac{\sum_{i=1}^{k} \widehat{\lambda}_i(u) \sum_{v \in W} \widehat{\lambda}_1(v)}{\sum_{i=1}^{k} \sum_{v \in W} \widehat{\lambda}_i(v)}, \qquad u \in W,$$

is the expected intensity at time $t = t_1$ and location $u \in W$. Doing so for all $u \in W$ enables us to draw a map of χ^2 values in a fixed time. The resulting map discloses the areas where the estimated intensity differs from the expected intensity. The function `chimaps` generates a map based on a given timestamp and rank. The argument rank is a number between one and the length of the generated time sequence based on the given timestamp, and with default one.

The chi maps of the 3-rd of Feb based on three different ranks are displayed in Figure 11. Values of each pixel is calculated by equation 8. We show the chi maps for three different times during the day in which changes over time can be seen. The left plot of Figure 11 shows the chi map at `06:10:44` so that the estimated intensity is higher than the expected intensity in the countryside. The reason for this might be the movements from countryside to the city center in the early morning. The middle plot of Figure 11 shows that the estimated intensity in the city is higher than the expected intensity. This may be caused by heavier traffic in the city during the day than in the countryside. In the right plot of Figure 11, although the estimated intensity is still slightly higher than the expected one in the city, we can see that the χ^2 statistic 8 takes values around 0 almost everywhere at night. These three plots together confirm the changes in the values of the χ^2 statistic 8 over time so that the mass is moving to the city in the morning and goes away in the evening. This behaviour may be explained by the movements to the city in the morning and moving back to the countryside in the evening.

```
R>  # ch <- chimaps(Beijing,timestamp = "20 mins",rank = 1)
R>  # chall <- attr(ch,"ims")
R>  # minmax <- lapply(X=1:length(chall),function(i){
R>  #     return(list(min(chall[[i]]$v),max(chall[[i]]$v)))
R>  #   })
R>  # minmax <- do.call("rbind",minmax)
R>  # col5 <- colorRampPalette(c('blue','white','red'))
R>  # color_levels=200
R>  # par(mar=c(0,0,1,1))
R>  # par(mfrow=c(1,3))
R>  # plot(chall[[51]],zlim=c(-max(abs(unlist(minmax))),max(abs(unlist(minmax))))
R>  #      ,main="",ribwid=0.04,ribsep=0.02,
R>  #      col=col5(n=color_levels),cex.axis=1.7)
R>  # title(attr(ch,"timevec")[51],line = -10,cex.main=2)
R>  # plot(chall[[75]],zlim=c(-max(abs(unlist(minmax))),max(abs(unlist(minmax))))
R>  #      ,main="",ribwid=0.04,ribsep=0.02,
```

```
R>  #      col=col5(n=color_levels),cex.axis=1.7)
R>  # title(attr(ch,"timevec")[75],line = -10,cex.main=2)
R>  # plot(chall[[104]],zlim=c(-max(abs(unlist(minmax))),max(abs(unlist(minmax))))
R>  #      ,main="",ribwid=0.04,ribsep=0.02,
R>  #      col=col5(n=color_levels),cex.axis=1.7)
R>  # title(attr(ch,"timevec")[104],line = -10,cex.main=2)
```

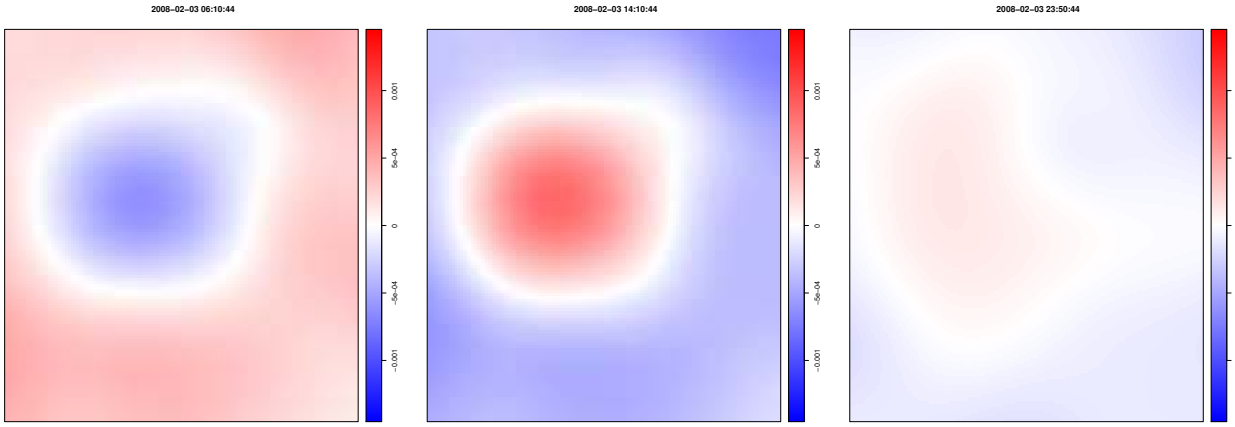

Figure 11: Chi maps. *Left*: in the morning, *Middle*: in the afternoon, *Right*: at night. Exact time is reported on top of each plot.

5.6. Second-order summary statistics

After discretising a trajectory pattern and estimating the intensity function of each single resulted point pattern, we now turn to look at the interaction between the moving objects over time. We are interested in distinguishing whether objects tend to move independently or they show some kind of dependence (e.g., clustering or inhibition). A common way in the point process literature is to use summary statistics such as K- and pair correlation functions (Ripley 1977; Baddeley, Møller, and Waagepetersen 2000; Baddeley *et al.* 2015). Pairwise distances are the hint here: if objects tend to be close to each other, then most of the pairwise distances are going to be small, and if they favour to stand far, then only a few of the pairwise distances are small (Baddeley *et al.* 2015). Baddeley *et al.* (2000) considered second-order intensity-reweighted stationary point processes, and defined the inhomogeneous $K-$function as

$$K_{inhom}(r) = \frac{1}{|B|}\mathbb{E}\sum_{x_i \in B}\sum_{x_j \neq x_i}\frac{\mathbf{1}\{\|x_i - x_j\| \leq r\}}{\lambda(x_i)\,\lambda(x_j)}, \qquad r \geq 0 \tag{9}$$

for any region $B \subset \mathbb{R}^2$ with area $|B| > 0$. The pair correlation function is also given by

$$g(u,v) = \frac{\lambda^2(u,v)}{\lambda(u)\lambda(v)}, \qquad u, v \in W. \tag{10}$$

We point out that for Poisson point processes $K_{inhom}(r) = \pi r^2 (g(r) = 1)$, and $K_{inhom}(r) > \pi r^2 (g(r) > 1)$ indicates clustering, while $K_{inhom}(r) < \pi r^2 (g(r) < 1)$ shows inhibition between

points. There is a close relationship between the K-function 9 and the pair correlation function 10 as

$$g(r) = \frac{K'_{inhom}(r)}{\pi r^2}, \qquad r > 0$$

where K'_{inhom} is the derivative of K_{inhom}. The plug-in estimator of the K-function 9 is of the form

$$\widehat{K}(r) = \frac{1}{|W|} \sum_i \sum_j \frac{\mathbf{1}\{d_{ij} \leq r\} e(x_i, x_j, r)}{\widehat{\lambda}(x_i)\, \widehat{\lambda}(x_j)}, \qquad r \geq 0 \tag{11}$$

where d_{ij} is the distance between x_i, x_j and $e(x_i, x_j, r)$ is an edge correction. For more details see Møller and Waagepetersen (2003, Chapter 4); Illian *et al.* (2008); Diggle (2013); Gabriel (2014); Baddeley *et al.* (2015, Chapter 7).

Summary statistics such as the K- and the pair correlation functions are used to analyse the type of interaction between points. Having this in mind, we next turn to use these functions in analysing the trajectory pattern S. Similar to our proposal for the intensity function and using summary statistics for point patterns, we here propose a variability area for the K- and the pair correlation functions as follows:

1. According to regular timestamps, discretise the trajectory pattern S and build the point patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$.
2. For all the resulted point patterns $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, estimate the K-function using 9.
3. From the estimated K-functions $\widehat{K}_1, \widehat{K}_2, \ldots, \widehat{K}_k$, build the pointwise variability area of the K-function, i.e., for each value of distance argument r, sort $\widehat{K}_i(r)$ and then take the lowest and highest value amongst all; doing so for a sequence of r results in a variability area for the K-function. This shows how the type of interaction between objects changes over time.

Note that, steps above can be applied to the pair correlation function as well.

In both functions above, users can take advantage of the bandwidth selection to first estimate the intensity function and then pass estimated intensities to the function `Kinhom` or `pcfinhom`. Default is to not pass any estimated intensity function to `Kinhom` or `pcfinhom` in which the intensity will be estimated using the 'leave-one-out' kernel smoother (Baddeley *et al.* 2000, 2015). Different edge corrections can be also passed to `Kinhom.Track` and `pcfinhom.Track`.

Finally and taking into account that estimated intensity in Figure 10 represent a non-unifrom distribution and/or clustering behaviour, we show the variability area of K-function and pair correlation function over time in Figure 12 (considering the "translate" correction, see Gabriel (2014)). The left plot displays the variation of K-function, showing that for small distances taxis tend to have a clustering behaviour while for larger distances they favour inhibition. The right plot of the variation of the pair correlation function also confirms the same behaviour. Due to the preference of moving within particular zones, K-function and pair correlation function might result as what is displayed in Figure 12. In other words, taxis might prefer to take passengers to close destinations within particular zones rather than further destinations.

Argument `q` is to use quantiles of the values of K-function (or pair correlation function) rather than using maximum and minimum. Default `q=0` uses maximum and minimum.

```
R>   # K <- Kinhom.Track(Beijing,correction = "translate",
R>   #                       timestamp = "20 mins",q=0)
R>   # par(mfrow=c(1,2),mar=rep(5,4))
R>   # plot(K,cex.axis=1.7,cex.lab=1.5,cex=2)
R>   # g <- pcfinhom.Track(Beijing,timestamp = "20 mins",q=0)
R>   # plot(g,cex.axis=1.7,cex.lab=1.5,cex=2)
```

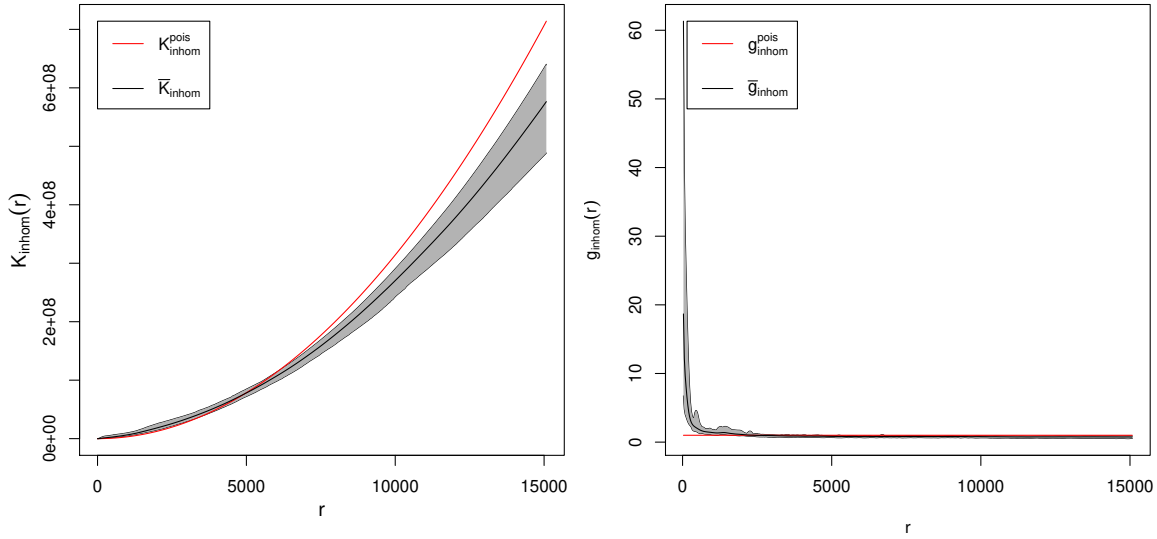

Figure 12: Variability area of second-order summary statistics for taxi data in Beijing, China. *Left*: K-function, *Right*: pair correlation function.

We point out that as this kind of discretising trajectory patterns results in a set of spatial point patterns, it may also drives us to consider them as replicated spatial point patterns. Therefore, one may be able to introduce a single K-function or pair correlation function for all patterns. Note that in this case, point patterns are not independent. For more details, see Diggle (2013, Section 5.4).

6. Summary and discussion

Analysing trajectory data is often a challenge as moving objects are living in space while moving in time. Such data might be analysed in space or time separately. However, one may look at trajectory patterns with a different perspective.

In this paper, we introduced a set of classes to handle trajectory data and defined a set of methods to summarise trajectory patterns. We have considered different classes to handle tracks passed by one object or a group of objects. We also aimed at statistically analysing trajectory data in space and see the changes in the behaviour of data over time. To do so, the literature of spatial point processes has been adapted to trajectory patterns. An average intensity function is defined for a trajectory pattern with tracks that overlap in time and

it nicely reveals the more visited spots, streets, highways, etc. Pairwise distances between different objects have been also considered in **trajectories** showing how moving objects interact over time. Moreover, we have implemented methods to smooth the length of movements over time which highlights where objects move faster/slower. In addition, the discrepancy between the estimated intensity and the expected intensity per location per time can be studied in **trajectories** which shows how the mass is moving within the studied area over time. We finally presented a variability area for second-order summary statistics such as K-function and pair correlation function which displays the type of interaction between objects over time, highlighting possible clustering or inhibition.

To perform an exploratory data analysis, we finally applied the developed methodologies to a taxi data from Beijing, China. The results nicely highlight the behaviour of taxis over time. We have recognized where the more visited spots are and also at what time data pattern is denser or objects get closer to each other. We smoothed the movements of taxis and discovered where and at what time they move faster. Not surprisingly, results confirm that moving within the city is slower than countryside or highways. Also, in the afternoon there is a decrease in the length of movements which might be affected by traffic. Chi maps have been studied showing how the mass is moving within the studied area. They display that objects are moving from countryside to the city in the morning making a heavy traffic in the city in the afternoon, as the estimated intensity is higher than the expected one. Chi maps also confirm that the discrepancy between the estimated intensity and the expected one is less during the evening than any other time. The variability area of K-function and pair correlation function show that for small distances, taxis tend to show a clustering behaviour while for longer distances they show repulsion.

We leave open the analysis of the effect of the environment on moving objects and the corresponding parametric analysis for future works.

Acknowledgments

M.M. Moradi gratefully acknowledges funding from the European union through the GEO-C project (H2020-MSCA-ITN-2014, Grant Agreement Number 642332, http:// www.geo-c.eu/. J. Mateu is partially funded by grants MTM2016-78917-R and P1-1B2015-40. Benedikt Klus, Benedikt Gräler and Nikolai Gorte are acknowledged for code contributions to the **spacetime** package.

References

Baddeley A, Møller J, Waagepetersen R (2000). "Non-and semi-parametric estimation of interaction in inhomogeneous point patterns." *Statistica Neerlandica*, **54**(3), 329–350.

Baddeley A, Rubak E, Turner R (2015). *Spatial Point Patterns: Methodology and Applications with R*. CRC Press.

Baddeley A, Turner R (2005). "**Spatstat**: an R package for analyzing spatial point patterns." *Journal of Statistical Software, Articles*, **12**(6), 1–42. ISSN 1548-7660. doi:10.18637/jss.v012.i06. URL https://www.jstatsoft.org/v012/i06.

Barr CD, Schoenberg FP (2010). "On the Voronoi estimator for the intensity of an inhomogeneous planar Poisson process." *Biometrika*, **97**(4), 977–984.

Battaile B (2014). ***TrackReconstruction**: Reconstruct animal tracks from magnetometer, accelerometer, depth and optional speed data, depth and optional speed data.* R package version 1.1, URL https://CRAN.R-project.org/package=TrackReconstruction.

Calenge C (2006). "The package **adehabitat** for the R software: a tool for the analysis of space and habitat use by animals." *Ecological modelling*, **197**(3), 516–519.

Campbell HA, Watts ME, Dwyer RG, Franklin CE (2012). "**V-Track**: software for analysing and visualising animal movement from acoustic telemetry detections." *Marine and Freshwater Research*, **63**(9), 815–820.

Challa S, Morelande M, Musicki D, Evans R (2011). *Fundamentals of Object Tracking*. Cambridge University Press.

Chiu SN, Stoyan D, Kendall WS, Mecke J (2013). *Stochastic Geometry and its Applications*. John Wiley & Sons.

Daley DJ, Vere-Jones D (2007). *An Introduction to The theory of Point Processes: Volume II: General Theory and Structure*. Springer-Verlag Science & Business Media.

Diggle P (1985). "A kernel method for smoothing point process data." *Applied Statistics*, **34**(2), 138–147.

Diggle P (2013). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. CRC Press.

Farrell E, Fuiman L (2013). ***animalTrack**: Animal track reconstruction for high frequency 2-dimensional (2D) or 3-dimensional (3D) movement data.* R package version 1.0.0, URL https://CRAN.R-project.org/package=animalTrack.

Freitas C (2012). ***argosfilter**: Argos locations filter.* R package version 0.63, URL https://CRAN.R-project.org/package=argosfilter.

Frick H, Kosmidis I (2017). "**trackeR**: Infrastructure for Running and Cycling Data from GPS-Enabled Tracking Devices in R." *Journal of Statistical Software, Articles*, **82**(7), 1–29. ISSN 1548-7660. doi:10.18637/jss.v082.i07. URL https://www.jstatsoft.org/v082/i07.

Gabriel E (2014). "Estimating second-order characteristics of inhomogeneous spatio-temporal point processes." *Methodology and Computing in Applied Probability*, **16**(2), 411–431.

Gurarie E (2014). ***bcpa**: Behavioral change point analysis of animal movement.* R package version 1.1, URL https://CRAN.R-project.org/package=bcpa.

Güting RH, Schneider M (2005). *Moving objects databases*. Elsevier.

Hanks EM, Hooten MB, Alldredge MW, *et al.* (2015). "Continuous-time discrete-space models for animal movement." *The Annals of Applied Statistics*, **9**(1), 145–165.

Hooten MB, Johnson DS (2017). "Basis function models for animal movement." *Journal of the American Statistical Association*, **112**(518), 578–589.

Illian J, Penttinen A, Stoyan H, Stoyan D (2008). *Statistical Analysis and Modelling of Spatial Point Patterns*. John Wiley & Sons.

Jones MC (1993). "Simple boundary correction for kernel density estimation." *Statistics and Computing*, **3**(3), 135–146.

Liu YS (2014). ***BayesianAnimalTracker**: Bayesian Melding of GPS and DR Path for Animal Tracking*. R package version 1.2, URL https://CRAN.R-project.org/package=BayesianAnimalTracker.

Michelot T, Langrock R, Patterson TA (2016). "**moveHMM**: an R package for the statistical modelling of animal movement data using hidden Markov models." *Methods in Ecology and Evolution*, **7**(11), 1308–1315.

Møller J, Waagepetersen R (2003). *Statistical Inference and Simulation for Spatial Point Processes*. CRC Press.

Nielson RM, Sawyer H, McDonald TL (2013). ***BBMM**: Brownian bridge movement model*. R package version 3.0, URL https://CRAN.R-project.org/package=BBMM.

Niu M, Blackwell PG, Skarin A (2016). "Modeling interdependent animal movement in continuous time." *Biometrics*, **72**(2), 315–324.

Ord J (1978). "How many trees in a forest?" *Mathematical Sciences*, **3**, 23–33.

Pebesma E (2012). "**spacetime**: Spatio-Temporal data in R." *Journal of Statistical Software, Articles*, **51**(7), 1–30. ISSN 1548-7660. doi:10.18637/jss.v051.i07. URL https://www.jstatsoft.org/v051/i07.

Radinger J, Wolter C (2014). "Patterns and predictors of fish dispersal in rivers." *Fish and Fisheries*. R package **fishmove**, version 0.3-3, URL http://dx.doi.org/10.1111/faf.12028.

Rakhimberdiev E, Saveliev A, Piersma T, Karagicheva J (2017). "**FLightR**: an R package for reconstructing animal paths from solar geolocation loggers." *Methods in Ecology and Evolution*, **8**(11), 1482–1487.

Ripley BD (1977). "Modelling spatial patterns." *Journal of the Royal Statistical Society. Series B (Methodological)*, **39**(2), 172–212.

Russell JC, Hanks EM, Haran M (2016). "Dynamic models of animal movement with spatial point process interactions." *Journal of Agricultural, Biological, and Environmental Statistics*, **21**(1), 22–40.

Sumner MD (2016). ***trip**: Tools for the Analysis of Animal Track Data*. R package version 1.5.0, URL https://CRAN.R-project.org/package=trip.

Sumner MD, Wotherspoon SJ, Hindell MA (2009). "Bayesian estimation of animal movement from archival and satellite tags." *PLoS One*, **4**(10), e7324. doi:https://doi.org/10.1371/journal.pone.0007324.

Toohey K (2015). ***SimilarityMeasures**: Trajectory Similarity Measures*. R package version 1.4, URL https://CRAN.R-project.org/package=SimilarityMeasures.

Tracey JA, Sheppard J, Zhu J, Sinkovts R, Chourasia A, Lockwood G, Fisher RN (2014). ***mkde**: 2D and 3D movement-based kernel density estimates (MKDEs).* R package version 0.1, URL https://CRAN.R-project.org/package=mkde.

Yan J, Pozdnyakov V (2016). ***smam**: Statistical Modeling of Animal Movements.* R package version 0.3-0, URL https://CRAN.R-project.org/package=smam.

Yuan J, Zheng Y, Xie X, Sun G (2011). "Driving with knowledge from the physical world." In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 316–324. ACM.

Yuan J, Zheng Y, Zhang C, Xie W, Xie X, Sun G, Huang Y (2010). "T-drive: driving directions based on taxi trajectories." In *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, pp. 99–108. ACM.

Affiliation:

M. Mehdi Moradi
Institute for New Imaging Technology
University of Jaume I
Avda. Sos Baynat, s/n
12071 Castellon, Spain
E-mail: moradi@uji.es

and

Edzer Pebesma
Institute for Geoinformatics
University of Münster
Weseler Strasse 253
48151 Münster, Germany
E-mail: edzer.pebesma@uni-muenster.de
URL: http://ifgi.uni-muenster.de/

52°North Initiative for Geospatial Open Source Software GmbH
Martin-Luther-King-Weg 24
48155 Muenster, Germany
URL: http://www.52north.org/

and

Jorge Mateu
Department of Mathematics
University of Jaume I
Avda. Sos Baynat, s/n
12071 Castellon, Spain
E-mail: mateu@uji.es
URL: https://www3.uji.es/~mateu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd