Spatiotemporal Data Representation in R

Lorena A. Santos[1], **Karine R. Ferreira**[1], **Gilberto R. Queiroz** [1], **Lubia Vinhas**[1]

[1]National Institute for Space Research
Av. dos Astronautas, 1758,
12227-010 - São José dos Campos (SP) - Brazil

{lorena.santos,karine.ferreira,gilberto.queiroz,lubia.vinhas}@inpe.br

Abstract. *Recent advances in devices that collect geospatial information have produced massive spatiotemporal data sets. Earth observation and GPS satellites, sensor networks and mobile gadgets are examples of technologies that have created large data sets with better spatial and temporal resolution than ever. This scenario brings a challenge for Geoinformatics: we need software tools to represent, process and analyze these large data sets efficiently. R is a environment widely used for data analysis. In this work, we present a study of spatiotemporal data representation in R. We evaluate R packages to access and create three spatiotemporal data types as different views on the same observation set: time series, trajectories and coverage.*

1. Introduction

Recently, the amount of devices that collect geospatial information has greatly increased. Earth observation and GPS satellites, sensor networks and mobile gadgets are examples of technologies that have created large data sets with better spatial and temporal resolution than ever. This technological advance brings many challenges for Geoinformatics. We need novel software tools to represent, process and analyze big spatiotemporal data sets efficiently.

In Geoinformatics, spatiotemporal data representation is an open issue. Spatial information is represented following well-established models and concepts. This includes the dichotomy between object-based and field-based models [Galton 2004]. Examples of long-standing concepts are vector and raster data structures, topological operators, spatial indexing, and spatial joins [RIGAUX et al. 2002]. Most existing GIS and spatial database systems, such as PostGIS and Oracle Spatial, are grounded on these concepts. However, there is no consensus on how to represent spatiotemporal information in computational systems.

Many existing proposals of spatiotemporal data models focus on representing the evolution of objects and fields over time. Some proposals are specific for discrete changes in objects [Worboys 1994] [Hornsby and Egenhofer 2000], others for moving objects [Guting and Schneider 2005] [ISO 2008] and still others for fields or coverage [Liu et al. 2008] [OGC 2006]. To properly capture changes in the world, representing evolution of objects and fields over time is not enough. We also need to represent events and relationships between events and objects explicitly [Worboys 2005]. Events are occurrents [Galton and Mizoguchi 2009]. They are individual happenings with definite beginnings and ends. The demand for models that describe events has encouraged recent research on spatiotemporal data modeling [Galton and Mizoguchi 2009].

R is a software tool widely used for data analysis [R Development Core Team 2011]. It provides a broad variety of statistical methods (time-series analysis, classification and clustering) and a high-level programming environment and language suitable for fast developing new algorithms. R is extended via packages. Although there are many packages for spatial data handling and analyzing, few of them can properly deal with the temporal dimension of spatial data.

This paper presents a study of spatiotemporal data handling in R. We evaluate R packages for spatiotemporal data access and representation. To guide this evaluation, we consider the spatiotemporal data types proposed by Ferreira et al. (2014). They propose a data model that represents objects and fields that change over time as well as events. Based on this model, we describe in this work how to load and create three spatiotemporal data types in R as different views on the same observation set: *time series*, *trajectory* and *coverage*.

2. An Observation-based Model for Spatiotemporal Data

Ferreira et al. (2014) propose a data model for spatiotemporal data and specify it using an algebraic formalism. Algebras describe data types and their operations in a formal way, independently of programming languages. The proposed algebra is extensible, defining data types as building blocks for other types. It takes observations as basic units for spatiotemporal data representation and allows users to create different views on the same observation set, meeting application needs.

Observations are our means to assess spatiotemporal phenomena in the real world. Recent research draws attention to the importance of using observations as a basis for designing geospatial applications [Kuhn 2009]. The proposed model defines three spatiotemporal data types as abstractions built on *observations*: *time series*, *trajectory* and *coverage*. A *time series* represents the variation of a property over time in a fixed location. A *trajectory* represents how locations or boundaries of an object change over time. A *coverage* represents the variation of a property in a spatial extent at a time. We also define an auxiliary type called *coverage series* that represents a time-ordered set of coverages that have the same boundary. Using these types, we can represent objects and fields that change over time as well as *events*.

2.1. Different Views on the Same Observation Set

Figure 1 shows an example of observations collected by five moving objects. Each observation is represented as a tuple in the form (id, x, y, t, p), where id is the object identification, x and y are spatial locations, t is time and p is a property value collected in the spatial local x and y and in time t. In this example, the property collected is air pollution.

On these observations, we can create different views depending on the kinds of analysis we want to perform on them. Each view is materialized as a data type. Figure 2 illustrates trajectory and coverage instances built on the same observation set shown in Figure 1. For example, to analyze how the objects move over time and space, we create an instance of the *trajectory* data type for each object. Each trajectory instance contains observations of an specific object. To analyze how the air pollution varies in a region, we create instances of the *coverage* data type. Each coverage instance contains observations in a specific period, mixing observations of different objects. To analyze how the air

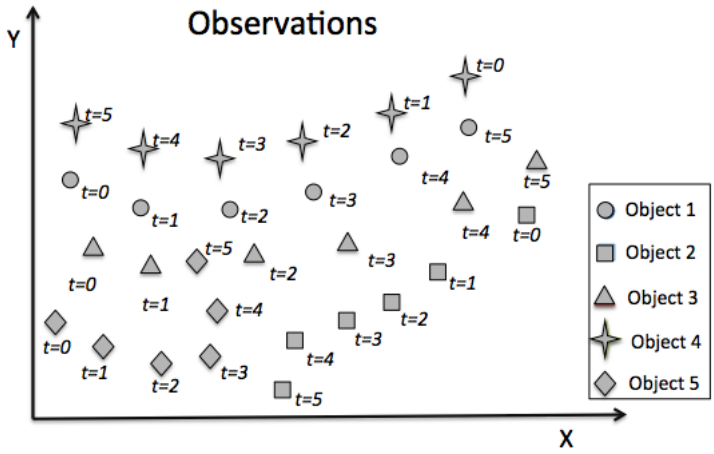

Figura 1. Spatiotemporal observations

pollution varies in a given spatial location over time, we can create an instance of *time series* data type.

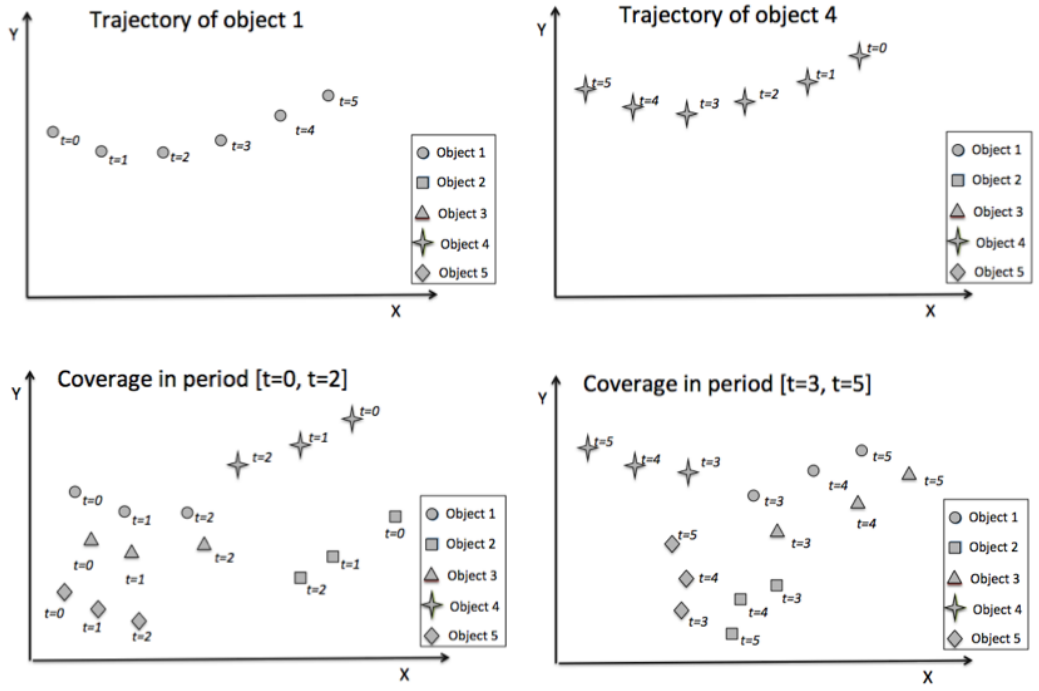

Figura 2. Different data types built on the same observation set

In the proposed model, each spatiotemporal data type instance, *time series*, *trajectory* and *coverage*, has an interpolation function, called interpolator, suitable for it. A time series has an interpolator that estimates property values in non-measured times. A trajectory has an interpolator able to estimate locations in non-observed times. A coverage has

an interpolator to estimate property values in non-observed locations.

The capability of creating different views or data types on the same observation set is an important requirement for spatiotemporal data representation. In this work, we evaluate how to do this using R. We use equivalent R data types to represent observation sets and to create trajectory, time series and coverage from these sets.

3. Spatiotemporal Data Representation in R

In this section, we describe a set of R packages for spatiotemporal data representation and access. Packages for data access are `Rgdal` [Bivand et al. 2013a], `Rpostgres` [Conway et al. 2008] and `Rodbc` [Ripley and Lapsley 2016]. Packages with data types that can be used to represent spatiotemporal data are `spacetime` [Pebesma 2012], `xst` [Ryan and Ulrich 2012], `trajectories` [Pebesma and Klus 2015] and `raster` [Hijmans 2016]. Furthermore, we evaluate some R package that provide interpolation functions that are crucial for creating spatiotemporal data type, such as `gstat` [Pebesma and Graeler 2016].

Figure 3 shows a diagram with these packages and the relationships among them.

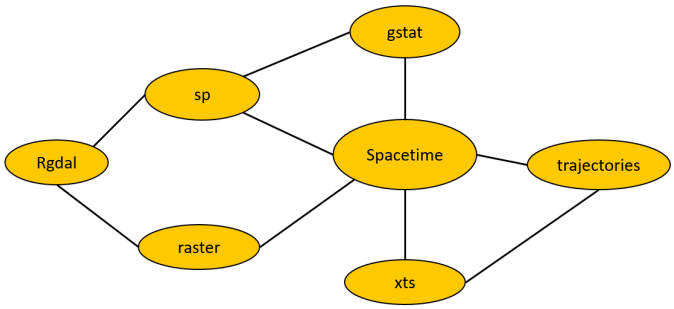

Figura 3. Relationships between packages

Spatiotemporal data can be obtained from different data source, such as, database (e.g. Postgis), data files (e.g. shapefiles and geotif raster files) and web services [Ferreira et al. 2015]. In R, there are packages that can access data from distinct type of source. However, these packages do not work directly with the concept of spatiotemporal data.

The `Rgdal` [Bivand et al. 2013a] package allows a broader range of spatial data sources [Lovelace and Cheshire 2014], such as shapefiles, raster data files and database systems. `RODBC` and `Rpostgres` packages allow to create SQL queries in R for accessing data in database systems, but they do not deal with spatial data.

The `spacetime` package contains a set of base classes for spatiotemporal data representation that are widely used for other R packages for spatiotemporal analyses. It is built upon the classes and methods for spatial data from package `sp` and for time series data from package `xts` [Pebesma 2012]. The `xts` package was chosen due to its support

to represent several types of date and time. Moreover, it extends functionality of `zoo` package, that has good tools for aggregation over time [Zeileis and Grothendieck 2005].

Each spatial data type from `sp` package has two slots that contain bound box, a matrix of numerical coordinates and other slots that contain a `CRS` class object defining the coordinate reference system [Bivand et al. 2013b]. The spatial data can be a particular spatial point, line, polygon or set of polygons, or a pixel (grid or raster cell) [Pebesma 2012].

The `spacetime` package classes are shown in Figure 4. Spatial Full Grid (`STF`) and Sparse Grid (`STS`) have the same general layout, with observations on a space time grid. The main difference between both is that `STF` stores the full grid, all observations of all space time points, while `STS` only stores non-missing valued observations. Examples that can be represent by these classes are: time sequences of satellite imagery and measuring air quality every hour [Pebesma 2016].

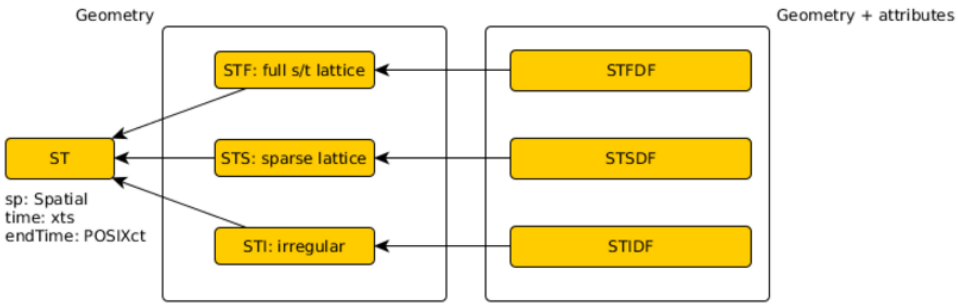

Figura 4. Classes for spatiotemporal data in package spacetime. [Pebesma 2016]

Irregular Grid (`STI`) represent a layout where for each spatial data a time point is stored. An example that can be represent for `STI` is measurement from mobile sensors. All classes presented derive from a base class,`ST`. This class is virtual, which is not represent actual data. The ST class derives two order classes: a spatio-temporal geometries and a augment class with actual data, in the form of data frame.

The objects from `xts` and `sp` packages are not used to store property values. For purely temporal information `xts` is used, and for purely spatial information `sp` objects is used. Then, it is necessary to use a `data.frame` to combine, space, time and property values. It represents the data as rectangle of rows containing observations on columns of property values [Bivand et al. 2013b].

The raster data set is composed by multiple layers, hence, the `raster` package has two classes for work with multi-layer data ,`rasterStack` and `rasterBrick` . The principal difference between these classes is that a `rasterBrick` can only be linked to a single file, while `rasterStack` can be formed from separate files and/or from few layers from a single file [Hijmans 2016]. Each raster layer in the stack or brick needs to be in the same projection, spatial extent and resolution. In other cases, such as, reading satellite images, we do not use `Rgdal` package, the `raster` package can be used directly using the function `raster()`. To represent spatiotemporal data raster, from a collection raster, we add a time for each layer using the `setZ` function from `raster` package, then

we can coerce these data to spatiotemporal type from `spacetime` package.

`Trajectories` package provides three data types to represent trajectories, `Track`, `Tracks` and `TracksCollection`, based on the `STIDF` type. The class `Track` represents a single trajectory followed by a person, animal or object. `Tracks` embodies a collection of trajectories followed by a single person, animal or object. The class `TracksCollection` represents a collection of trajectories followed by different persons, animals or objects. Besides that, this package provides a set of operations over trajectories, such as computing of trajectories STBox and calculating distances between two tracks.

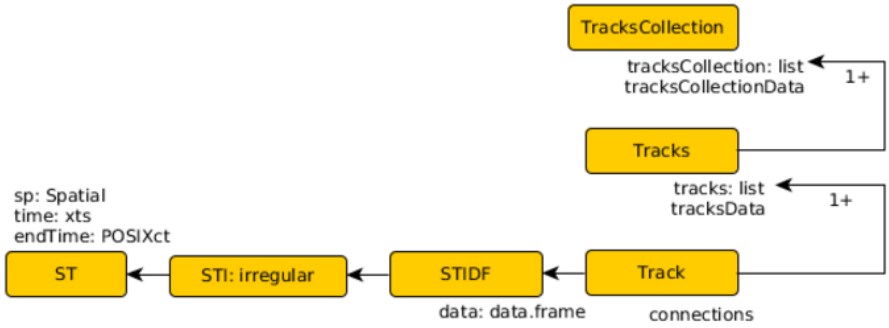

Figura 5. Classes for spatiotemporal data as trajectories. [Pebesma 2016]

The package `gstat` provides spatial and spatiotemporal interpolation functions. Types derived from `sp` and `spacetime` packages can be utilized in this package.

4. Case Study

In this section, we present the use of the R packages listed in the previous section approaching the data types proposed by [Ferreira et al. 2014]. We performed a case study using observations of vessels around the Brazilian coast. These observations are stored in a PostGIS database and contain trajectories of 993 vessels collected during 4 years, from 2008 to 2011. The Figure 6 presents the observations of all vessels.

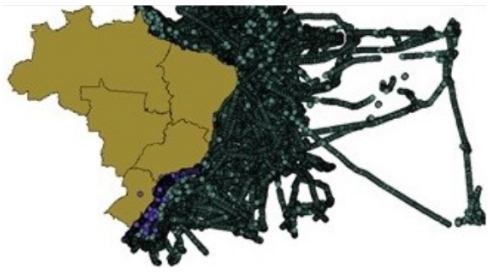

Figura 6. All Observation Set ploted against the map of Brazil

4.1. Observation Set

In this case study, we selected a small subset, containing trajectories of 166 vessels. We filtered data temporally obtaining trajectories only one day and spatially for locations in Rio de Janeiro State. Our subset is shown in Figure 7.

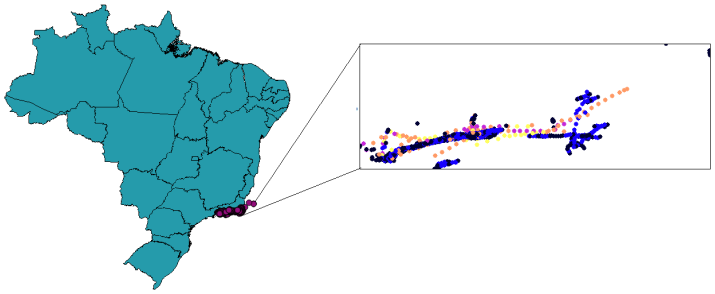

Figura 7. Filtered Data

The filtered vessel observations are stored in table of PostGIS database, where each row contains an observation of a vessel. Each observations contains the vessel id (integer type), time (timestamp type), spatial location (geometry type) and velocity of each vessel (numeric type). The table format is shown in Figure 8.

id integer	datahora timestamp without time zone	ponto geometry(Geometry,4326)	velocity numeric
583	2010-06-17 05:08:35	0101000020E6100000BBB88D06F08E45C0371AC05B201137C0	0.408763472410365
583	2010-06-17 06:08:05	0101000020E6100000A2B437F8C29445C0F90FE9B7AF1337C0	1.35374278395259
583	2010-06-17 07:08:43	0101000020E6100000287E8CB96B9145C0FBCBEEC9C31237C0	0.751117663040952
583	2010-06-17 08:08:10	0101000020E61000007958A835CD8B45C0569FABADD80F37C0	1.31993573146551
583	2010-06-17 09:09:25	0101000020E61000000BB5A679C79145C08A1F63EE5A1237C0	1.34078049066134
583	2010-06-17 10:09:45	0101000020E610000054742497FF9045C052B81E85EB1137C0	0.181341567758016

Figura 8. Observation set

In this case study, we used `RODBC` to connect in the database and filter the observation data using SQL language inside R environment. This this step is shown in Figure 9.

```
library(RODBC)

con <- odbcConnect("PostgreSQL30")

query <- "select id, datahora, velocity, st_x(ponto) x, st_y(ponto) y
          from onedayVelocidadeAll
          order by datahora"


vesselsObs <- sqlQuery(con, query)
```

Figura 9. Acessing data from Postgis

`RODBC` does not work with spatial data. Therefore, coordinates data is returned as numeric type and so must be converted to spatial type of R. A matrix is created with coordinate values and then we transform this matrix in a `SpatialPoint` in R format with a coordinate reference system. For temporal data, we do not need realize changes, because the data was returned in `POSIXct` type. This type is standard way of representing time in R [Wuertz et al. 2015], and also in `spacetime` package.

Finally we created a `data.frame` with only one column, containing velocity data, named `VelocityDF`. Combining the spatial, temporal and data frame objects we can apply in a spacetime class. In this case we apply `STIDF` , because our observations are irregular data. Figure 10 shows how these data are prepared in R.

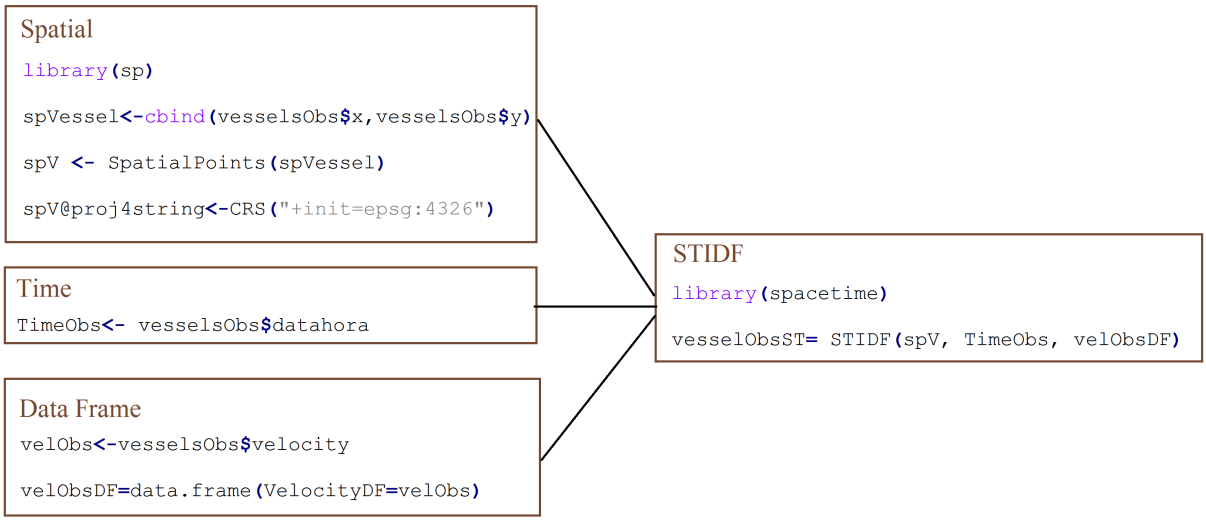

Figura 10. Creating STIDF object

4.2. Trajectories and Time Series

Two vessels from the observation set was selected. We created two objects of type `STIDF`, one for each vessel, `vesselST1` and `vesselST2`. Then, we created two instances of `Track` type from `trajectories` package; each one to represent a trajectory of an object. The R code for this step is shown in Figure. 11 and the trajectories generate are show in Figure 12.

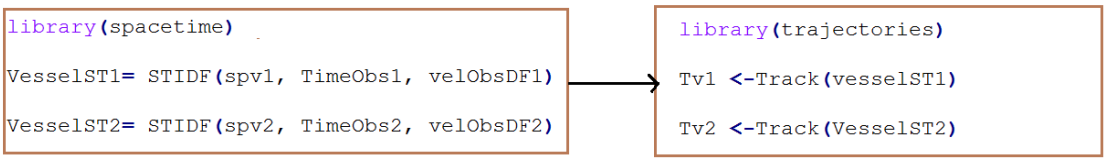

Figura 11. Creating Trajectories from STIDF

Here, we can also analyze how two vessels are moving together. The `trajectories` has a function to calculate distances between two tracks using the method `compare`. This function returns a object of type `difftrack`. From this object we can obtain the distance between trajectories over time and create a time series using the `xts` type. This time series represents the distance variation between two trajectories over time. The Figure 13 shows the time series and how it is generated.

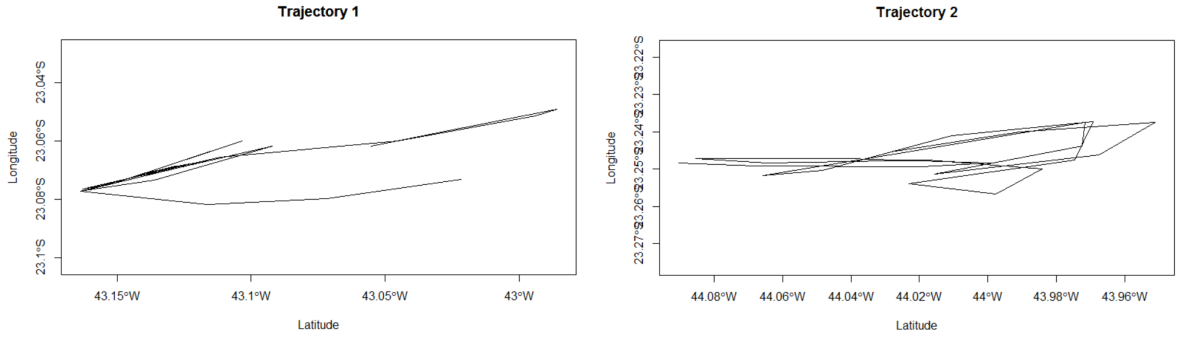

Figura 12. Visualizing spatiotemporal data as Trajectories

```
library(xts)

trajCmp<-compare(Tv1,Tv2)

timeCmp<-trajCmp@conns1$time

dist1<-trajCmp@conns1$dists

distTS<-xts(dist1,timeCmp)
```

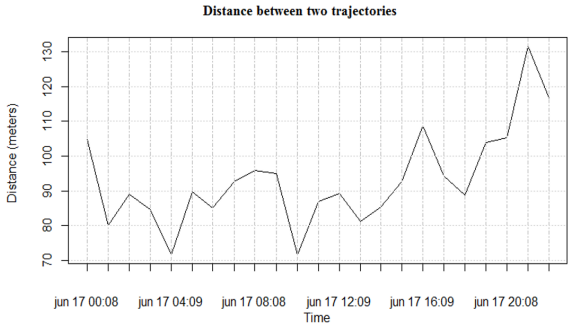

Figura 13. Time Series: distance variation over time

4.3. Coverage

To represent the velocity variation over time and space, we create coverages. We put together observations obtained by vessels during one day and produced a coverage that investigate how velocity varies within the boundary delimited by spatiotemporal bounding box object created previously. This kind of view on the vessel observations allows users to identify possible regions where vessels are fishing. In fishing areas, vessels have low velocity.

Our observations are discrete. They need to be combined with interpolation functions to estimate values in non-observed spatial locations [Ferreira et al. 2014]. We used a spatial interpolation and created a grid where every cell has a velocity value.

To create a grid, we constructed a `GridTopoloy` object using the bounding box of our subset region and a cell size of $0.01°$ in each direction. From the `GridTopoloy` object, we constructed the `SpatialGrid` object and associated a coordinate reference system (`CRS`) to it. Figure 14 shows the code used to create this grid.

We created a coverage for each hour of a day, using the Inverse Distance Weighted Interpolator (IDW) interpolation function. In this case, each coverage contains the observations of a specific hour, mixing observations of different vessels. The type returned by

the IDW interpolator is `SpatialGridDataframe`. This type is typical for representation raster GIS [Bivand et al. 2013b]. Using `Rgdal`, each grid can be saved as Tif format. Figure 15 shows the code to obtain the time interval from `STIDF` object to be spatially interpolated over the grid created earlier. This step was realized 24 times, for each hour.

```
csN <- c(0.01, 0.01)

ccN <- vesselObsST@sp@bbox[,1] + (csN/2)

cdN <- ceiling(diff(t(vesselObsST@sp@bbox))/csN)

gridVessel <- SpatialGrid(GridTopology(cellcentre.offset = ccN,
                                cellsize = csN, cells.dim = cdN))

proj4string(gridVessel)<-CRS("+init=epsg:4326")
```

Figura 14. Creating a spatiotemporal grid

```
t0 <- coredata(naviosST@time['2010-06-17 00:00:00/2010-06-17 01:00:00'] )

vesselt0<- idw(naviosST@data[t0[1]:t0[length(t0)],]~ 1,
               naviosST@sp[t0[1]:t0[length(t0)],], gridNavio, idp = 2.5)
```

Figura 15. Inverse Distance Weighted Interpolator

Once all grids were created, we read the 24 grids using `stack()` function from `raster` package and put these raster grids together using `stack::raster` function. Two stacks are generated, one containing all raster grids during 00:00 until 12:00 and another containing all raster grids during 12:00pm until 23:59pm. Furthermore, each raster pushed in the stack was associated to time interval using `setZ()` function from `raster` package. These raster stacks contain spatial and temporal data. They can be converted to the spatiotemporal data type `STFDF` of the `spacetime` package. The Figure 16 shows the code that describes this step for one period.

The Figure 17 shows the spatiotemporal raster grids to represent coverages in R. These coverages represent how the velocity varies over time within a specific region of the ocean. Looking these coverages, we can visually identify areas where the velocity is low. Such areas can be possible regions where vessels are fishing. Thus, using coverage types built on the vessel observations, we can extract information about the variation of velocity over time and space, using spatiotemporal data mining techniques.

It is important to note that the interval between 12:00 and 15:00, figures 17, contains large regions where the vessel velocity is low. We contrast with red circles regions where velocity is lower than others. Visually, we observed that these regions have low velocity in all time intervals since 12:00 until 23:59 pm.

```
library(raster)

s12<-stack(vessel2)

period_12_24<-seq(
from=as.POSIXct("2010-06-17 12:00", tz="UTC"),
to=as.POSIXct("2010-06-17 23:00", tz="UTC"),
by="hour"
)

rasterCover12_24<-raster::stack(s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23)

rasterST12_24 <- setZ(rasterCover12_24, period_12_24)

names(rasterST12_24) <- format(period_12_24, '%a_%Y%m%d')

STRaster12_24 <- as(rasterST12_24, "STFDF")
```

Figura 16. Creating spatiotemporal raster grids to represent coverages

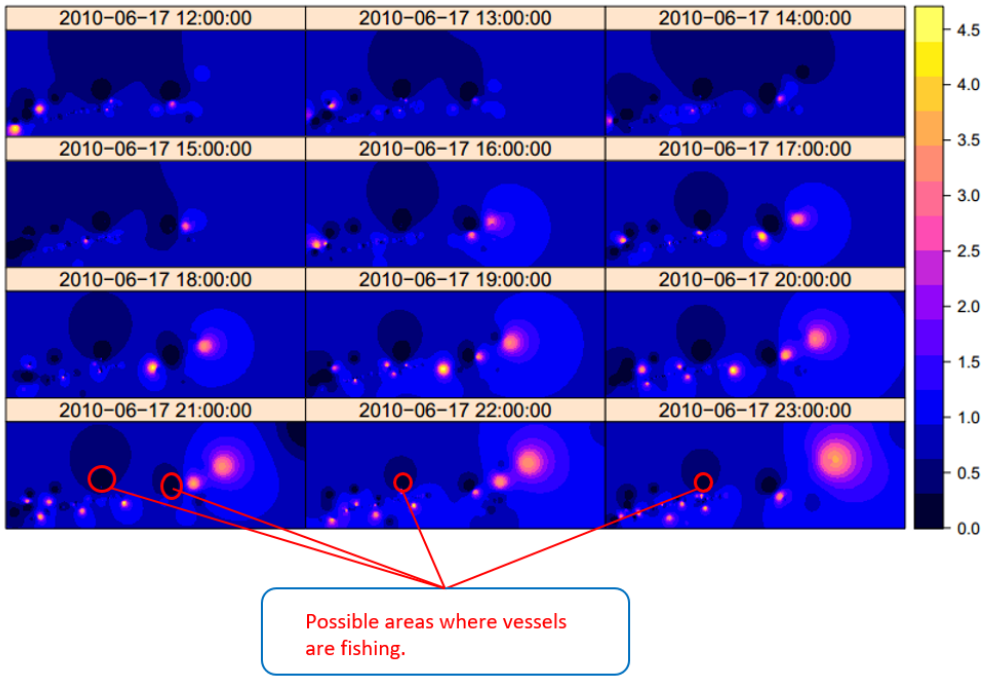

Figura 17. Velocity variation during 12:00 until 23:59

5. Evaluation and Final remarks

Although there are conceptual differences between the classes provided by R and the data types proposed by Ferreira et al. (2014), it is possible to use existing R classes to represent such data types. In summary, Table 1 lists the R packages and their classes that can be used to represent spatiotemporal data.

The data types proposed by Ferreira et al. (2014) have interpolation functions, called interpolator, associated to their instances. In R packages, the interpolation functions

are not directly associated to their objects. Thus, we have to use interpolation functions provided by other packages to estimate values in non-observed locations and times. For example, to create the coverages shown Figure 16, we used the IDW interpolation function from `gstat` package.

We can conclude that the `spacetime` package plays a key role in spatiotemporal data representation in R. It provides base data types, `STIDF`, `STFDF` and `STSDF`, that are used for many other packages to represent and analyze spatiotemporal data. For example, the packages `trajectories` defines their classes to represent trajectories based on the `STIDF` type. The package `gstat` provides spatiotemporal interpolation functions, such as spatiotemporal kriging, based on the `STFDF` class.

Spatiotemporal Data Type	Package	Class
Observation	spacetime	STIDF
Time series	xts	xts
Trajectory	trajectories	Track
Coverage	raster	setZ
	spacetime	STFDF

Tabela 1. R Packages and their classes to represent spatiotemporal data

Using existing R packages, such `RODBC` and `Rgdal`, we can access data from different data sources. Furthermore we can load subsets of data using queries in the case of vector data or creating several stacks of raster data to be processed in batches avoid overhead in memory. However, they do not deal with spatiotemporal data.

When we load the observation set using the `RODBC` package, spatial data sets are loaded as textual or numerical types. Then, it is necessary to convert these data types to spatial type. Using the `Rgdal` package, we can access the observation set as spatial data types. Neither of them can access spatiotemporal data directly. It is necessary to prepare spatial and temporal data separately and then construct spatiotemporal data types.

A disadvantage of not having packages that directly access spatiotemporal data sets, such as trajectory and coverage, is that we can not filter such data sets properly. For example, we can not load from data sources to R classes only the trajectories whose spatiotemporal bounding boxes intersect a given box. Or, we can not load from data sources to R classes only a part of coverages based on a spatiotemporal restriction. These filters are important because R has limitations of memory on handling large objects. According to Kane et al. [Kane et al. 2013], R is not well-suited for working with data structures larger than about 10-20% of a computer RAM memory. Thus, it is necessary to handle data sets by parts in R, using filters to restrict the amount of data in memory.

As future work, we intend to use spatiotemporal data mining algorithms in R to

analyze the coverages showed in Figure 16. The goal is to identify regions where vessel velocities are low, that is, regions where vessels are probably fishing.

Referências

Bivand, R., Keitt, T., and Rowlingson, B. (2013a). Rgdal: Bindings for the geospatial data abstraction library. r package version 0.8-10.

Bivand, R. S., Pebesma, E., and Gomez-Rubio, V. (2013b). *Applied spatial data analysis with R, Second edition*. Springer, NY.

Conway, J., Eddelbuettel, D., Nishiyama, T., Prayaga, S. K., and Tiffin, N. (2008). Rpostgresql: R interface to the postgresql database system.

Ferreira, K. R., Camara, G., and Monteiro, A. M. V. (2014). An algebra for spatiotemporal data from observations to events. *Transactions in GIS*, 18(2):253–269.

Ferreira, K. R., de Oliveira, A. G., Monteiro, A. M. V., and de Almeida, D. B. (2015). Temporal GIS and spatiotemporal data sources. In *XVI Brazilian Symposium on GeoInformatics(GEOINFO), Campos do Jordão, São Paulo, Brazil, November 29 - December 2, 2015.*, pages 1–13.

Galton, A. (2004). Fields and objects in space, time, and space-time. *Spatial cognition and computation*, 4(1):39–68.

Galton, A. and Mizoguchi, R. (2009). The water falls but the waterfall does not fall: New perspectives on objects, processes and events. *Applied Ontology*, 4(2):71–107.

Guting, R. H. and Schneider, M. (2005). *Moving Objects Databases*. Morgan Kaufmann.

Hijmans, R. J. (2016). Introduction to the raster package.

Hornsby, K. and Egenhofer, M. (2000). Identity-based change: a foundation for spatio-temporal knowledge representation. *International Journal of Geographical Information Science*, 14(3):207–224.

ISO (2008). Geographic information - schema for moving features. ISO 19141:2008, International Organization for Standardization, Geneva, Switzerland.

Kane, M. J., Emerson, J., and Weston, S. (2013). Scalable strategies for computing with massive data. *Journal of Statistical Software*, 55(14):1–19.

Kuhn, W. (2009). *A Functional Ontology of Observation and Measurement*, pages 26–43. Springer Berlin Heidelberg, Berlin, Heidelberg.

Liu, Y., Goodchild, M. F., Guo, Q., Tian, Y., and Wu, L. (2008). Towards a general field model and its order in GIS. *International Journal of Geographical Information Science*, 22(6):623–643.

Lovelace, R. and Cheshire, J. (2014). Introduction to visualising spatial data in R. *National Centre for Research Methods Working Papers*, 14(03).

OGC (2006). Opengis implementation specification for geographic information - simple feature access-part 1:common architecture. Technical Report 19141:2008, OPEN GEOSPATIAL CONSORTIUM, Geneva, Switzerland.

Pebesma, E. (2012). spacetime: Spatio-temporal data in R. *Journal of Statistical Software*, 51(7):1–30.

Pebesma, E. (2016). Handling and analyzing spatial, spatiotemporal and movement data.

Pebesma, E. and Graeler, B. (2016). gstat: Spatial and spatio-temporal geostatistical modelling, prediction and simulation.

Pebesma, E. and Klus, B. (2015). Analysing trajectory data in R.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

RIGAUX, P., SCHOLL, M., and VOISARD (2002). *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ripley, B. and Lapsley, M. (2016). Rodbc: Odbc database access.

Ryan, J. and Ulrich, J. (2012). xts: extensible time series. r package version 0.8-6.

Worboys, M. F. (1994). A unified model for spatial and temporal information. *Comput. J.*, 37(1):36–34.

Worboys, M. F. (2005). Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science*, 19(1):1–28.

Wuertz, D., Setz, T., Chalabi, Y., and Byers, M. M. J. W. (2015). Package timedate.

Zeileis, A. and Grothendieck, G. (2005). zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27.