A Proposal for Spatiotemporal Data Retrieving in R

Rodrigo S. S. Adeu[1], Karine R. Ferreira[1], Pedro R. Andrade[1]

[1]National Institute for Space Research
Av. dos Astronautas, 1758,
12.227-010 - São José dos Campos (SP) - Brazil

`rodrigo.sales@embraer.com.br,{karine.ferreira,pedro.andrade}@inpe.br`

Abstract. *Despite the development of R packages for spatial features and temporal data, developers still facing a lack on spatiotemporal data access and spatiotemporal data manipulation. Translating spatiotemporal data retrieved from different data sources into high level R classes is a common problem into developer's life. This paper describes an ongoing proposal to provide an abstraction layer for spatiotemporal data retrieving in R environment.*

1. Introduction

Spatiotemporal data is everywhere, being gathering from different devices such as Earth Observation and GPS satellites, sensor networks and mobile gadgets [Monteiro et al. 2016]. As the number, volume and resolution of spatiotemporal datasets increase, traditional methods for dealing with such data are becoming overwhelmed [Cheng et al. 2013]. This scenario brings a challenge for Geoinformatics: we need software tools to represent, process and analyse these large data set efficiently [Santos et al. 2016]. In this context, the use of R environment has been used as an alternative to data analysis. We now describe an on going proposal for access, retrieving and usage for spatiotemporal data access in R, starting by a R package review.

2. R Environment for Spatial, Temporal and Spatiotemporal Data

R environment is an integrated suite of software facilities for data manipulation, calculation and graphical display, and can be easily extended via packages [R Core Team 2011]. Packages provide a mechanism for loading optional code, data and documentation as needed. This extensions provides extra functionalities not included on the original R environment.

When looking at spatial data representation, Open Geospatial Consortium (OGC) has specified a standard that describes the common architecture for simple feature geometry used by Geographic Information Systems [OGC 2006]. Simple features refers to a formal standard (ISO 19125-1:2004) that describes how objects in the real world can be represented in computers, with emphasis on the spatial geometry of these objects. Package `sf` represents simple features as native R objects using simple data structures (S3 classes, lists, matrix, vector) [Pebesma et al. 2018 a.].

The challenges related to time data representation are also well explored on R environment. There are lots of packages and initiatives for time modelling on R. For instance, the `xts` package, developed motivated by the ability to improve performance by imposing reasonable constraints, while providing a truly time-based structure [Ryan and Ulrich 2017].

But developers are facing a very different scenario when trying to represent spatiotemporal data. The lack of a standard for spatiotemporal data modelling avoid the development of reusable solutions and common packages. For raster data representation, stars package [Pebesma et al. 2018 b.] has addressed the challenge to represent dense arrays, with space and time being array dimensions. In this paper, we present a proposal for an implementation of a R package, using a spatiotemporal model representation based on observations for vector data. This model defines three data types as abstractions built on observations: time series, trajectory, and coverage. Using these types, we can create different views on the same observation set, meeting application needs [Ferreira el al. 2014].

2. Observation Based Model

The proposed model starts with observations, which are our means to assess spatiotemporal phenomena in the real world [Kuhn, 2009]. The model defines three data types as abstractions built on observations: time series, trajectory, and coverage. A time series represents the variation of a property over time. It is obtained from observations that measure values at controlled times 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. Using a UML class diagram, we can represent a Time Serie, a Trajectory and a Coverage as subclasses from the superclass Observation.

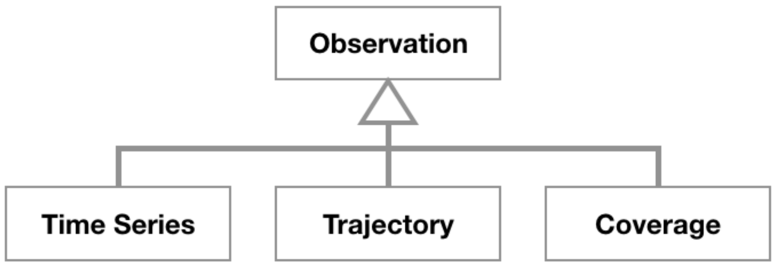


Figure 1. Simplified Observation Based Model Class Diagram.

We'll now look at the actual R packages that can be used to retrieve the observation data and represent them into high level classes.

2. R Packages for SpatioTemporal Data Usage

As data sources are constantly collecting information, and there is no international standard and widely accepted spatiotemporal data model with well-developed theories and technologies [Hall and Leahy 2008], is very common to find lots of different formats for storing and sharing this data. GeoJSON, KML, PostgreSQL/ Postgis databases are a few examples of supported formats. In order to simplify the interface to the most common formats, `rgdal` package provides bindings to Geospatial

Data Abstraction Library (GDAL) and access to projection/transformation operations from the PROJ.4 library [Bivand et al. 2017].

The second important functionality is to provide high level classes to map spatiotemporal objects. This feature allow developers to increase abstraction and improve reuse of source codes, focusing on data analysis and business logic instead of low level details. In this paper, we use a data model proposed by Ferreira el al. (2014) that takes observations as the basic unit for spatiotemporal data representation. For each data type, an R package can be used to address specific details and characteristics of this abstractions. For *time series*, the package `xts` [Ryan and Ulrich 2017] can be used. For *trajectory*, the package `trajectories` [Pebesma and Klus 2015] can be used. And for *coverage*, the package `spacetime` [Pebesma 2012] can be used. Figure 2 shows this packages and the relationships between them, including the package `sp` [Pebesma and Bivand 2005] used to represent spatial data when this packages were released. We also include the package `sf`, [Pebesma et al. 2018 a.], that should be used for representing spatial geometry objects following the OGC standard. Nowadays, package `sf` is not compatible with `spacetime` and `trajectories` packages.

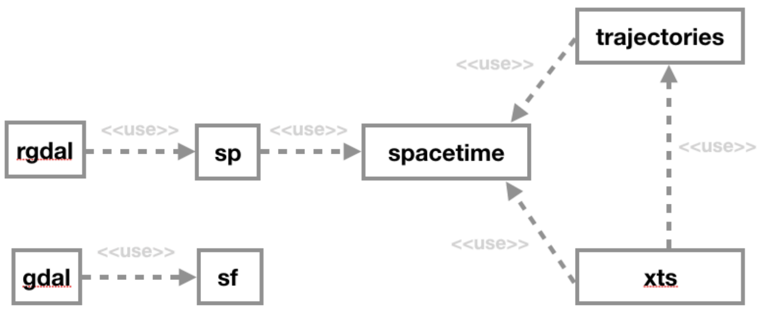


Figure 2. Possible R packages and relationships

In order to represent the observation based model classes as high level classes in R, and fulfil them with observation data, we need to specify a software architecture to simplify the data access and the translation to the high level classes. On the next session, we'll start to detail a proposal of architecture for a R package.

3. Software Architecture

When looking at a software architecture, this packages can be grouped into two different layers, encapsulating important functionalities to developers. The first one called *Data Access Layer* is responsible for dealing with different data sources, access the raw data, and provide this data to developers. Package `rgdal` is included in this layer. The second layer is called *Data Usage Layer* and is responsible for providing high level classes for developer usage. Packages `xts`, `trajectories` and `spacetime` are included in this layer. Developers are responsible for dealing with this two layers, and translate or adapt outcoming data from *Data Access Layer* into incoming data for *Data Usage Layer.* We represent that with a *Middleware Layer*.

Nowadays, software developers spend undesirable time writing code inside *Data Access Layer* and *Middleware Layer.* As data sources changes, *Data Access Layer* code that retrieve this data must be rewritten. *Middleware Layer* code that translate this data to high level classes must also be rewritten. In an ideal scenario, developers should be focused on business logic and data analysis, writing code for *Data Usage Layer*. This software architecture is showed on Figure 3.

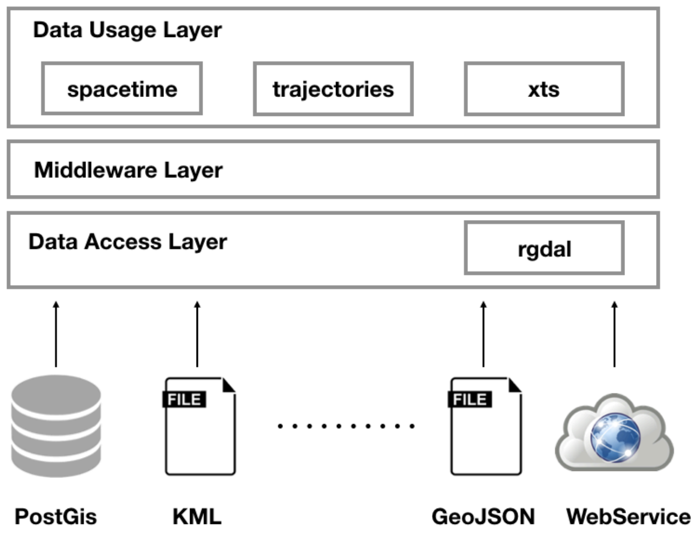


Figure 3. Common software architecture

In this paper we describe a proposal for an R package that abstracts *Data Access Layer* and *Middleware Layer*. By using intermediate classes, we intend to change characteristics from this layers using reconfigurable objects, instead or rewriting code.

4. Proposed R Package Internal Architecture

Internally, the proposed R package implements design patterns and architectural solutions to encapsulate the internal structure and provide a reusable piece of software. The driver for the package design was implement configurable objects with a neutral interface, allowing developers to use the package independent of the data source and the data set internal structure. The goal is to provide a solution to allow developers to focus only on data usage with high level classes. A proposed class diagram is showed on Figure 4.

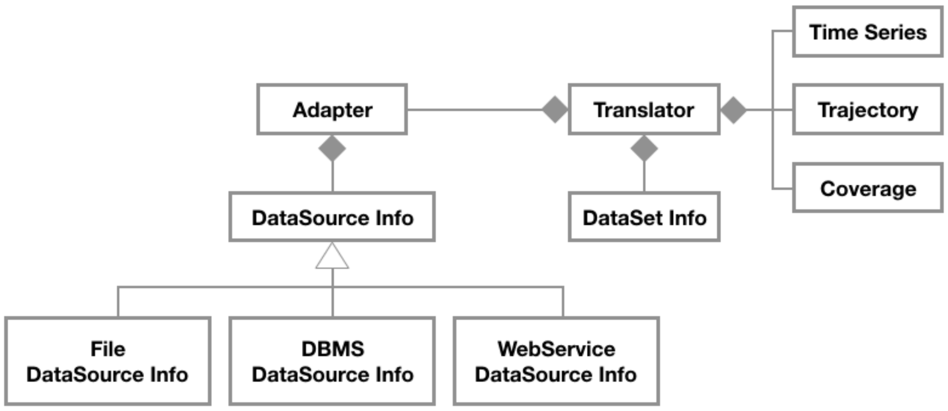
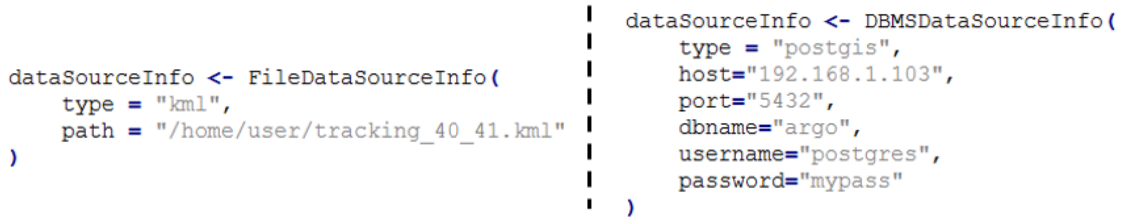


Figure 4. Proposed package internal architecture

In order to specify information about data sources, there are two different `DataSourceInfo` specific classes, each one for a specific domain. `FileDataSourceInfo` class is used to access data stored on files, while `DBMSDataSourceInfo` is used to access data stored on Data Base Management Systems. Details of each data source, for example file path for Kml files, or username and password for PostGis connections, are encapsulated on each class. As an example, Figure 5 shows the creation of `DataSourceInfo` objects for Kml and Postgis.



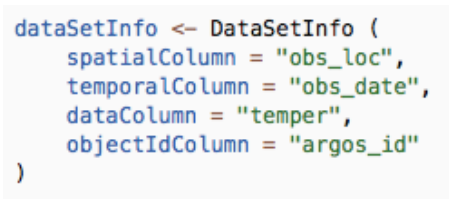
```
dataSourceInfo <- FileDataSourceInfo(
    type = "kml",
    path = "/home/user/tracking_40_41.kml"
)
```

```
dataSourceInfo <- DBMSDataSourceInfo(
    type = "postgis",
    host="192.168.1.103",
    port="5432",
    dbname="argo",
    username="postgres",
    password="mypass"
)
```

Figure 5. Creating configuration objects for Kml on left and Postgis on right.

The basic `DataSourceInfo` class acts as an interface for specific data source information classes, providing standardised access to data sources.

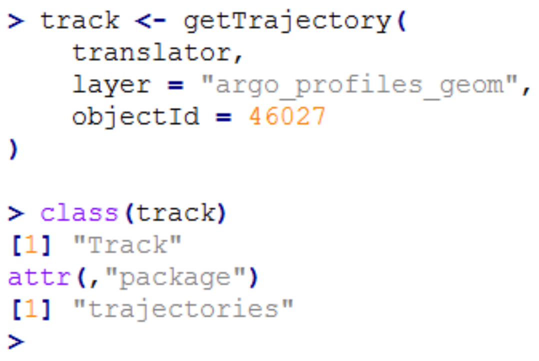
In order to specify information about the data set internal structure, an object of `DataSetInfo` class must be instantiated. This object provide details about dataset internal structure, like spatial columns, temporal columns and measure columns. An example of `DataSetInfo` object is showed on Figure 6.



```
dataSetInfo <- DataSetInfo (
    spatialColumn = "obs_loc",
    temporalColumn = "obs_date",
    dataColumn = "temper",
    objectIdColumn = "argos_id"
)
```

Figure 6. DataSetInfo object for Argo Data

The `Translator` class is responsible for implement an adapter, receive data from it, translate it into high level classes (Time Series, Trajectory and Coverage) and return this object to the developer. This class is also responsible for receiving columns that need to be loaded, and insert the correct values inside the object. Figure 7 illustrates the use of the `Translator` class to retrieve a `track` object, representing a trajectory from trajectories package.



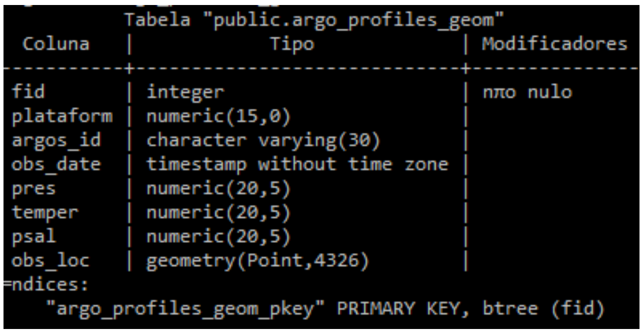
```
> track <- getTrajectory(
    translator,
    layer = "argo_profiles_geom",
    objectId = 46027
)

> class(track)
[1] "Track"
attr(,"package")
[1] "trajectories"
>
```

Figure 7. Translator class returning a Trajectory object

Using this packages, developers should only set a `DataSourceInfo` class specific for the desired data source, and specify the columns that the class `Translator` needs to retrieve using a `DataSetInfo` class. `Translator` class will use `Adapter` class to retrieve this data and return a high level class, containing the desired data to the developer.

4. Case Study

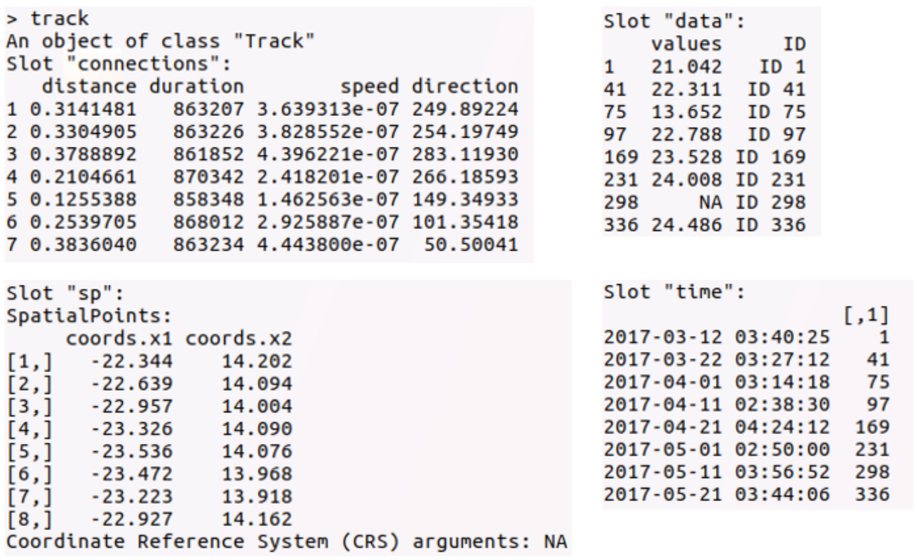
To evaluate the capabilities of the proposed abstraction layer, data from Argos Project were used. Argos float technology is recognised within the climate community as a huge step in the climate observing system, providing essential observations of the ocean down to 2000 metres [Argos 2015]. This data was stored inside a PostGis database, and the internal structure of the table `argo_profiles_geom` is showed on Figure 8.



```
            Tabela "public.argo_profiles_geom"
 Coluna    |            Tipo             | Modificadores
-----------+-----------------------------+---------------
fid        | integer                     | nπo nulo
plataform  | numeric(15,0)               |
argos_id   | character varying(30)       |
obs_date   | timestamp without time zone |
pres       | numeric(20,5)               |
temper     | numeric(20,5)               |
psal       | numeric(20,5)               |
obs_loc    | geometry(Point,4326)        |
=ndices:
    "argo_profiles_geom_pkey" PRIMARY KEY, btree (fid)
```

Figure 8. Argo_profiles_geom table detailing

Using the `DataSourceInfo` showed on right side of Figure 4, `DataSetInfo` showed on Figure 5, and instantiating objects from `Adapter` and `Translator` classes, we were able to use the R command provided on Figure 7 and retrieve temperature measurements from float 46027. The most relevant slots of the returned object are showed on Figure 9. All specific methods from class `Track` are also available.



```
> track
An object of class "Track"
Slot "connections":
   distance duration        speed direction
1 0.3141481   863207 3.639313e-07 249.89224
2 0.3304905   863226 3.828552e-07 254.19749
3 0.3788892   861852 4.396221e-07 283.11930
4 0.2104661   870342 2.418201e-07 266.18593
5 0.1255388   858348 1.462563e-07 149.34933
6 0.2539705   868012 2.925887e-07 101.35418
7 0.3836040   863234 4.443800e-07  50.50041
```

```
Slot "data":
    values     ID
1   21.042   ID 1
41  22.311  ID 41
75  13.652  ID 75
97  22.788  ID 97
169 23.528 ID 169
231 24.008 ID 231
298     NA ID 298
336 24.486 ID 336
```

```
Slot "sp":
SpatialPoints:
     coords.x1 coords.x2
[1,]   -22.344    14.202
[2,]   -22.639    14.094
[3,]   -22.957    14.004
[4,]   -23.326    14.090
[5,]   -23.536    14.076
[6,]   -23.472    13.968
[7,]   -23.223    13.918
[8,]   -22.927    14.162
Coordinate Reference System (CRS) arguments: NA
```

```
Slot "time":
                    [,1]
2017-03-12 03:40:25    1
2017-03-22 03:27:12   41
2017-04-01 03:14:18   75
2017-04-11 02:38:30   97
2017-04-21 04:24:12  169
2017-05-01 02:50:00  231
2017-05-11 03:56:52  298
2017-05-21 03:44:06  336
```

Figure 9. Examples of track object slots

5. Further Developments

As this proposal, and the development of this R package still ongoing, there are lots of open points and definitions that need to be finalised. For further developments we would like suggest improvements on `Translator` class capabilities. As there are many date/time formats and no standardisation, an important modification on this class should be the use of regular expressions on date/time pattern recognition. `DataSetInfo` and `DataSourceInfo` classes could also be improved using RDF to describe the data, instead of source codes.

An important feature that could also be added is the event retrieving functionality. If the user adds some restriction (eg. a time slice, an object id or a bounding box), the package should be able to return an event, included on that restriction. In the above example, an event could be: two touching floats, or a float registering a high temperature increasing in a small amount of time.

To pack all this features, and comply with the OGC simple feature standard, we propose the creation of a temporal simple feature standard. This proposal shall include all abstractions to retrieve the data from multiple data sources, and translate it to high level classes, using the observation based model as a driver. The development of an R package encapsulating all this concepts can be used as a proof of concept of this temporal simple feature proposal.

References

Argos. 2015. International Argo Program: A Revolution in Climate Research. Argos Forum #81. p. 20. Available at: http://www.argos-system.org/wp-content/uploads/2016/08/r1580_f34_Argos-Forum-81_En.pdf

Bivand, R., Keitt, T., Rowlingson B. (2017). Rgdal: Bindings for the Geospatial Data Abstraction Library. R package version 1.2-8. https://cran.r-project.org/package=rgdal

Cheng, T., Haworth, J., Anbaroglu, B., Tanaksaranond, G., Wang, J. (2013). Spatiotemporal Data Mining. Handbook of Regional Science, p. 1173-1193.

Ferreira, K. R., Camara, G., Monteiro, A. V. M. (2014). An Algebra for Spatiotemporal Data from Observations to Events. Transactions in GIS, p. 253-269.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN 0-201-63361-2.

Hall, B. and Leahy, M. G. (2008). Open Source Approaches in Spatial Data Handling. Springer. p. 126-128. ISBN 978-1-84996-094-6.

Kuhn, W. A functional ontology of observation and measurement. In: International Conference on GeoSpatial Semantics (GeoS 2009), 2009, Mexico City, Mexico. Proceedings... Spring Verlag, 2009. Lecture Notes in Computer Science.

Monteiro, D. V., Ferreira, K. R., Santos, R., Andrade, P. R. (2016). Extending R for Big Trajectory Data Access.

OPEN GEOSPATIAL CONSORTIUM (OGC): OpenGIS Implementation Specification for Geographic Information – Simple Feature Access - Part 1: Common architecture. Reference number: OGC 06-103r3. Version: 1.2.0. Report. Available at <http://www.opengeospatial.org>. 2006.

Pebesma, E. (2012). spacetime: Spatio-Temporal Data in R. Journal of Statistical Software, 51(7), 1-30. URL http://www.jstatsoft.org/v51/i07/.

Pebesma, E. and Bivand, R. (2005). Classes and methods for spatial data in R. R News 5 (2), https://cran.r-project.org/doc/Rnews/.

Pebesma, E. and Klus, B. (2015). trajectories: Classes and Methods for Trajectory Data. R package version 0.1-4. https://CRAN.R-project.org/package=trajectories

Pebesma, E., Bivand, R., Cook I., Keitt T., Sumner, M., Lovelace R., Wickham H., Ooms J., Racine E., (2018 a.). “sf: Simple Features for R”. R package version 0.6-3. https://CRAN.R-project.org/package=sf.

Pebesma, E., Summer, M., Racine, E., Fantini, A., (2018 b.) “stars: Scalable, Spatiotemporal Tidy Arrays for R”. R package version 0.1-1. https://CRAN.R-project.org/package=stars

R Core Team (2016). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. ISBN 3-900051-07-0. http://r-project.org/about.html.

Ryan, J. A. and Ulrich, J. M. (2017). xts: eXtensible Time Series. R package version 0.10-0. https://CRAN.R-project.org/package=xts

Santos, L. A., Ferreira, K. R., Queiroz, G. R., Vinhas, L. (2016) Spatiotemporal Data Representation in R. XVII GEOINFO, p. 178-191.