S4 Classes and Methods for Spatiotemporal Data: the `spt` Package

J. Blair Christian*

15 October 2009

Contents

Introduction

Note: This vignette was directly copied from the sp package, then edited. Hopefully they will see this as flattery...

The `spt` package provides S4 classes and methods for dealing with spatiotemporal data in S (R and S-Plus[1]). The spatiotemporal data structures implemented currently include points, with support for lines, polygons and grids coming in the next release, 0.01-02. We have chosen to use S4 classes and methods style (Chambers, 1998) to allow validation of objects created. Although we mainly aim at using spatiotemporal data in the geographical (two-dimensional) domain, the data structures that have a straightforward implementation in higher dimensions (points, grids) do allow this.

The motivation to write this package was born from my work at the EPA as a postdoc along with the SAMSI program on Spatiotemporal analysis where I had the privilege to be a party crasher. At this time, there are no packages with broad support for spatiotemporal data, however there are various associated

The package is available on R-forge. From the package home page, http://spt.r-forge.r-project.org, a graph gallery with R code, and the development source tree are available.

This vignette describes the initial classes, methods and functions provided by `spt`. Instead of manipulating the class slots (components) directly, we provide methods and functions to create the classes from elementary types such as data.frames or vectors and to convert them back to any of these types. Also, coercion (type casting) from one class to the other is provided sparsely, where relevant. If something isn't provide, please request it from the author(s).

Package `spt` is installed by

R code

```
> install.packages("spt", repos = "http://R-forge.R-project.org")
```

Currently, I put a lot of dependencies and they're not all necessary now (especially the suggested packages).

Package `spt` is loaded by

R code

```
> library(spt)
```

*US EPA, Research Triangle Park, NC, `blair.christian@gmail.com`

[1] our primary efforts target R; depending on the needs, we will address S-Plus as well

Spatiotemporal Data Classes

Spatiotemporal data is a complicated affair. There are many, many types of spatiotemporal data and many, many uses of it. I do not presuppose to help all users with all applications. The current efforts only pertain to my current work with air pollution data, monitoring sites located at lat/long locations (x, y), producing an observation such as the measurement of ozone in ppm at regular time intervals (eg hours, days, etc), usually which have some relation to the regulatory standard (I have the outline of functions to calculate the primary and secondary ozone standard, for example). These stations eventually produce flat (eg text) files which are 4 columns (lat, long, time, obs) and n_{obs} rows.

Spatiotemporal Classes

The basic structure of spatiotemporal classes consists of 3 core classes and 1 optional class. The core classes are for storing the temporal part of the data, the spatial part of the data, and the observations. These are stored somewhat independently, with a unique identifier for the time and spatial location providing the map between the classes, in the spirit of a relational database. This has several advantages. First, the bulky temporal and spatial parts are reduced to their unique elements, reducing storage space and speeding up computation by performing operations on the unique elements, then duplicating those results as needed. Second, it allows the data to be stored in different places. This is not currently implemented, but it would allow the spatial parts to be stored using, say, the `SQLiteMap` package for the spatial part.

The current class is `SpatialTemporalDataFrame`. However, only point data is currently (v 0.01-01) supported.

Currently, this class contains objects of the other classes. From a big picture view, I don't know if it's better to use `setClassUnion` instead.

Temporal Classes

Currently, the temporal class is called `stTemporal`. It consists of a pairlist (but it's currently 2 vectors, the integer unique temporal IDs, and the timeDate time stamps.

The format is very important to ensure that the timedates are read correctly. Please visit the help page for `strptime` for the complete list.

Example: create, get,

R code

```
> tmp <- stTemporal(timeVals = c("2008-09-29", "2009-01-14", "2012-12-12"),
+     format = "%Y-%m-%d")
> tmp
> getTid(tmp)
```

output

```
[1] 1 2 3
```

R code

```
> getTimedatestamps(tmp)
```

output

```
[1] "2008-09-29" "2009-01-14" "2012-12-12"
```

Spatial Classes

Currently, the spatial class is called `stSpatial`. It consists of 2 objects, the integer unique spatial IDs, and a `Spatial` object, currently only `SpatialPoints` is supported.

Please visit the vignett for the `sp` package for the complete list of items to be supported in version 1. Note that none of the data containing classes will be supported here since we have our own data classes.

Currently I inherit the spatial class, and I'm not sure what the implications are for that (there's a big difference whether you contain an object of a class or whether you extend a class).

Example: create, get,

R code

```
> n <- 10
> d <- 2
> crd <- matrix(runif(n * d), n, d)
> bbox <- cbind(apply(crd, 2, min), apply(crd, 2, max))
> colnames(bbox) <- c("min", "max")
> tmp <- new("stSpatialPoints", s.id = as.integer(1:n), coords = crd,
+     bbox = bbox, proj4string = CRS(as.character(NA)))
> tmp
> getSid(tmp)
```

output

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

R code

```
> getSpatialPoints(tmp)
```

output

```
SpatialPoints:
           [,1]      [,2]
 [1,] 0.6739453 0.3021973
 [2,] 0.9619853 0.6170330
 [3,] 0.9228386 0.9050982
 [4,] 0.7747456 0.8476483
 [5,] 0.7248062 0.4579902
 [6,] 0.7413480 0.3863993
 [7,] 0.6302933 0.6443511
 [8,] 0.1903071 0.7170724
 [9,] 0.6977443 0.1980982
[10,] 0.3703263 0.2772779
Coordinate Reference System (CRS) arguments: NA
```

Data Classes

The main data class is called `stDataFrame`

R code

```
> tmp <- new("stDataFrame", s.id = as.integer(1:3), t.id = as.integer(1:3),
+     df = data.frame(s.id = as.integer(1:3), t.id = as.integer(1:3),
+         x1 = runif(3), x2 = rnorm(3)))
> tmp
```

output

```
A dataframe with 3 observations and 2 data columns with
 3 unique timedates and
 3 unique locations.
```

Metadata Classes

Originally I had put in a metadata class to provide information about the data: units, collection data, url, citation info, etc, but I have removed it until the package is more mature unless there is rampant support.

Methods

There are methods for: dist, join, subset, apply, but I don't have time to fully go into them yet (short on time)....

Quick Tour

R code
```
> pollutant <- "Oz"
```

output
```
[1] "Oz"
```

R code
```
> state <- c("SC", "NC")
```

output
```
[1] "SC" "NC"
```

R code
```
> startDate <- "2007-01-01"
```

output
```
[1] "2007-01-01"
```

R code
```
> endDate <- "2007-01-30"
```

output
```
[1] "2007-01-30"
```

R code
```
> tmp <- getEPAAirExplorerData(pollutant, state, startDate, endDate)
```

output
```
[1] "getting data for state SC ..."
[1] "getting data for state SC ..."
[1] "getting data for state NC ..."
A dataframe with 554 observations and 3 data columns with
 30 unique timedates and
 11 unique locations.
```

R code
```
> tmp
```

output
```
A dataframe with 554 observations and 3 data columns with
 30 unique timedates and
 11 unique locations.
```

R code
```
> plot(tmp)
```

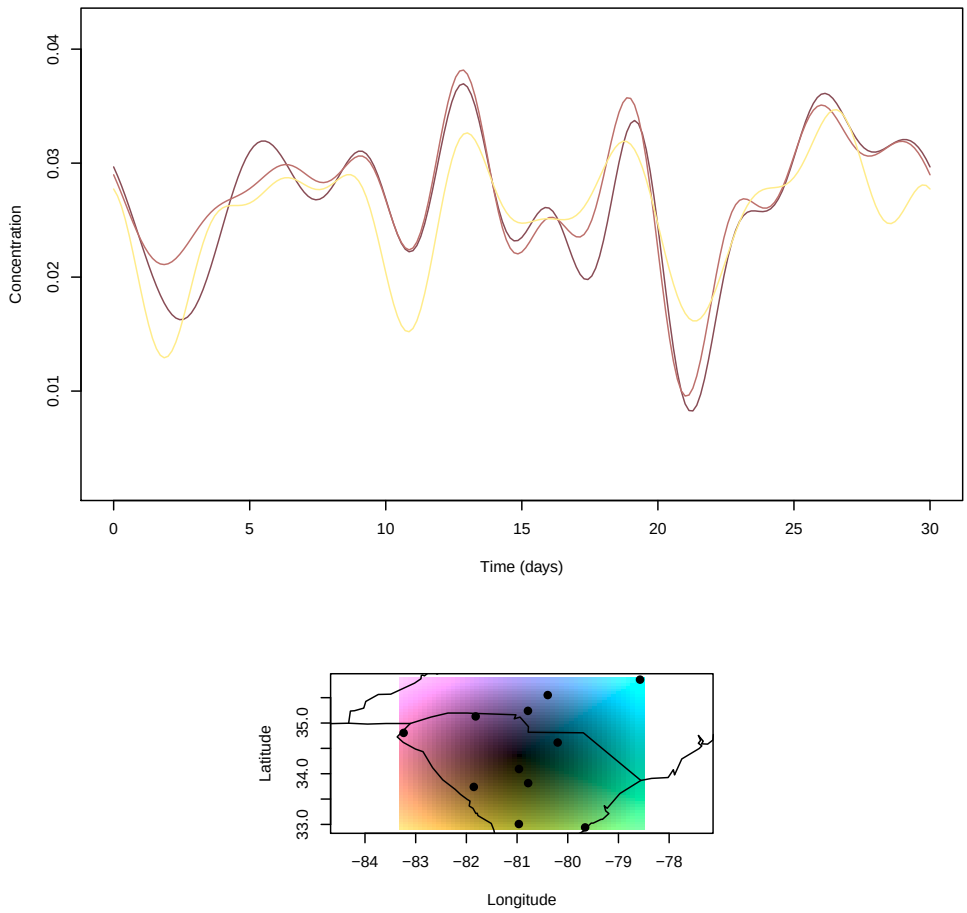

Figure 1: Top: Plots of ozone concentrations at the 3 stations with no missing data; Bottom: legend, color of top time plots corresponds to spatial location in legend

Performance

This code will be in need of constant performance work. I have tested the `SpatialTemporalDataFrame` with a maxtrix of about 3.5 million observations, with about 4,400 unique times and 900 unique locations. The original text file was about 240 Mb, it took aboud 10 min to create the R object (with match and spfrtTime taking the most- I should be able to increase the speed by removing match and replacing it with some unique/duplicated work), and the resulting saved Rdata file was about 140 Mb. This data set has lat, long, time, obs1, obs2. I used Rprof and calls to Sys.time() (verbose option inside constructor) to get data.

Wishlist

No ambitious project would be complete without feature requests.

Proposed Version 1.0 Features

- Support for Spatiotemporal data (points), in version 0.01-01 (creation, accessing, subsetting, joining, applying,
- Support for Spatiotemporal data (grids, lines, polygons)
- Data retrieval from website forms from http://www.epa.gov/cgi-bin/htmSQL/mxplorer/ (working, 0.01-01) and http://www.avianknowledge.net/akntools/download.
- RUnit is used to maintain quality control by providing a testing framework. (v0.01-01). Can be provided upon request until I put it into the test dir of the package. (haven't figured out yet).
- interfaces to `spBayes` and `INLA` packages
- periodicity (as in the xts library context)

Proposed Version 2.0 Features

- Support for larger data sets (both on the storage/access side as well as the computation side) On the storage side, perhaps database storage for some/all obs/objects, and on the computation side perhaps a way to pass to C or C++.
- More data access capabilities
- interfaces to `openair` and `trip` packages
- support to coerce timeDate to class xts

References

Chambers, J.M., 1998, Programming with data, a guide to the S language. Springer, New York.