`stars`: Scalable, spatiotemporal tidy arrays for R

Edzer Pebesma, Institute for Geoinformatics, Münster, Germany
Michael Sumner, Australian Antarctic Division, Hobart, Tasmania
Etienne B. Racine, Laval University, Québec, Canada

Summary

A lot of spatiotemporal data takes the form of dense, multidimensional arrays. Examples are

- population counts by region, year and age group
- weather data by variable, time step and sensor location
- satellite imagery, e.g. energy by spectral band, location (pixel) and time of collection
- climate model data, e.g. surface temperature by location, time and climate scenario.
- financial data, e.g. share price by company and time

Although such data *can* be represented in long tables, for larger datasets the array form is beneficial because it does not replicate dimension indexes, and the array form provides faster access by being implicitly indexed. R's native arrays have a number of limitations, they

- cannot handle heterogeneous data records (e.g. consisting of a `numeric`, a `logical` and a `Date`) like we typically have in data.frame's,
- can only deal with in-memory data, and
- do not handle spatial or temporal array dimensions.

This project will (i) implement a flexible and generic multidimensional array model for heterogeneous records that (ii) handles strong spatial and temporal referencing of array indexes, and (iii) scales from moderately sized *in- memory* data, to large *on-disk* data, and to massive data held on remote servers, while using a unified user interface that follows the tidy tools manifesto.

The Problem

How do we handle and analyze large amounts of spatially referenced time series data in R? How do we handle satellite imagery that don't fit on a local disc with R, or for which we need a small cluster to finish computation within acceptable time? How can we quickly and easily develop an analysis by testing it on a small portion of the spatiotemporal datasets, before deploying it on the massive data set? How can we use pipe-based workflows or dplyr-verbs on such data sets? How can we visualy explore high-dimensional raster data?

Today, people use R for large spatiotemporal data, but hit limits related to usability, user interface, and scalability. The r-sig-geo mailing list documents many of these cases.

Now that the simple features for R project has largely modernized the handling and analysis vector data (points, lines, polygons) in R in a tidyverse-friendly fashion, it is time for raster data to catch up. This proposal aims at spatiotemporal raster data, as well as time series of feature data.

Existing work

Base R supports n-dimensional homogeneous arrays of basic types (double, integer, logical, logical), in-memory. Package ff supports out-of-memory data structures, held on local disc, but no spatial or temporal references on dimensions.

Spatial packages include rgdal, which lets you read and write raster data, possibly piece-by-piece, in one of 142 different file formats. Package raster allows users to work with raster maps or stacks of them, where a stack can either refer to different bands (color) or different time steps, but not both. Package raster can iterate functions over large files on disc, and takes care of the caching, using either rgdal or ncdf4. Another package for reading and writing NetCDF is RNetCDF. Packages that are more dedicated to a single data source or type include include RStoolbox, MODIS, landsat, and hsdar; each of these relies on raster or rgdal for file-based I/O.

CRAN package spacetime provides heterogeneous records, using a `data.frame` for attributes. It keeps indexes for each record to a spatial geometry (grid cell/point/polygon) and time instance or period; it keeps all data in memory and builds on xts for temporal, and sp for spatial reference.

With support from the R Consortium, the Distributed Computing Working Group has started to develop an API for distributed computing; an initial version is available in package ddR. It aims at generic R data structures, and works towards relieving users from worrying that data is distributed.

Relevant work outside R includes

- GDAL, in particular gdal virtual tiles for building arbitrary large grid data sets from many individual files,
- SciDB, an open source array database which has no spatial or temporal capabilities, but a strongly scalable architecture, and extremely flexible array manipulation methods
- SciDB4geo, a SciDB Plugin for Managing Spatial and Temporal Reference Information of Arrays, and SciDB4gdal, a GDAL driver for SciDB arrays, two activities to make SciDB databases aware of space and time
- PostGIS Raster, a raster data extension of PostGIS
- Rasdaman, an array database dedicate to images, which is partially open source.

Since there is a definite trend that downloading Earth observation data is no longer feasible, we will have to work towards solutions where the data are accessed over a web service interface. Cloud services such as AWS are starting to give access to the large remote sensing imagery archives of e.g. Landsat, MODIS and Sentinel satellites.

The Plan:

We will develop an R package and container infrastructure that

- supports dense, n-dimensional arrays with heterogeneous records,
- supports flexible reference from array dimensions to space, where space can be gridded (2D/3D raster) or a set of simple features (1D, irregular)
- supports flexible reference from array dimensions to time (`POSIXct`, `Date`)
- supports regular arrays (fixed cell size / time step) as well as irregular arrays
- allows working in memory, on local disc, and on a remote computer (using a web service interface),
- allows R functions to be passed on to the web service back-end, and executed there in parallel,
- uses in-memory proxies to large arrays, allowing to work with the first n records before computations are carried out on the full arrays (similar to how dplyr does this)
- allows pipe-based workflows, using `data.frame`s, and dplyr-style verbs.

We will document the software and provide tutorials and reproducible data analysis examples using locally downloaded imagery, as well as scalable examples accessing larger ($>$ 1 Tb) datasets using docker container images.

We will document the RESTful API that connects the R client with the web service holding (and processing) the big Earth observation data.

We will also develop and discuss a migration path for the raster package (which has 43K lines of R, C and C++ code), and its functionality, into the new infrastructure.

We will publish the resulting products in an open access form, in the R journal, but also in a journal (or on a conference) more directed to the Earth observation community.

Timeline:

- Month 1-2: work out design, decide web service technology, basic web service API design
- Month 3-6: programming the R package, testing with smaller data sets
- Month 7-8: testing on larger datasets, develop test cases, deploy on docker containers
- Month 9-12: write tutorials, develop teaching material and reproducible examples
- Month 9-12: experiment with different back-ends: file-based, or database such as SciDB

Possible failure modes are:

- we can't get the RESTful API to work properly; solution path: ask the rOpenSci community for help (Scott Chamberlain, Jeroen Ooms)

- downloading large image sets is too cumbersome (slow) for the larger tutorial examples; solution path: deploy a test server for teaching/experimenting purposes in the Amazon cloud, where Landsat and Sentinel imagery is readily available

How Can The ISC Help:

We will use most funding to develop the R package and web service API. Total costs will be 10,000 USD, and breaks down in:

- workshop: travel costs for Etienne Racine and Michael Sumner to visit Muenster, or another venue where we can all meet (USD 2500).
- Programming, project communication: (USD 7000).
- Cloud deployment in the Amazon cloud (USD 500).

Dissemination:

We will regularly post blogs about the project on r-spatial.org, use twitter, post to r-sig-geo, stackoverflow, and communicate through github issues or gitter discussion. The project will live on GitHub, in the r-spatial organisation. We will work under a permissive open source license, probably LGPL-2.1. Pull requests will be encouraged. R consortium blogs will be provided at start and end. Publications in *the R Journal* and other scientific outlets are foreseen.