R-global: analysing spatial data globally

Edzer Pebesma

2019-03-18

Project team

Edzer Pebesma, Roger Bivand, Michael Sumner, Ege Rubak and Jeroen Ooms

The Problem

Most spatial data today comes with *geodetic* coordinates, expressed as degrees longitude and latitude, associated with some reference ellipsoid (typically WGS84), denoting locations on an ellipsoidal body. Most of the R software used to analyse these data assumes that locations denote two-dimensional *Cartesian* coordinates, coordinates on a flat 2-D space, i.e. that the data are projected. Projection distorts areas, distances and/or directions, and does this more so when the the geographical area considered is larger. Ideally, one would analyse the data as they are, points on an ellipsoid, only project when needed (i.e. when plotting), choosing appropriate projections by default.

This problem affects all data scientists who are faced with spatial data that come with geodetic coordinates (degrees longitude/latitude). In particular it affects those who are not well versed in the problem of choosing a projection appropriate to their problem. We think that this is the case for the majority of data scientists today.

More in detail, we will demonstrate two simple problems. In the first, we show that a line crossing the dateline seems to not cross it, but rather span the entire Earth:

```
> library(sf)
Linking to GEOS 3.7.0, GDAL 2.3.2, PROJ 5.2.0
> line = st_as_sfc("LINESTRING(-179 50, 179 50)", crs = 4326) # crossing dateline
> dateline = st_as_sfc("LINESTRING(180 0, 180 90)", crs = 4326)
> st_intersects(line, dateline) # do they cross? No:
although coordinates are longitude/latitude, st_intersects assumes that they are planar
Sparse geometry binary predicate list of length 1, where the predicate was `intersects'
 1: (empty)
> st_bbox(line) # ... but does span the entire Earth:
xmin ymin xmax ymax
-179   50  179   50
```

In another problem, a polygon spanning the North Pole does not include the pole:

```
> pol = st_as_sfc("POLYGON((0 80, 120 80, 240 80, 0 80))", crs = 4326)
> st_bbox(pol) # a line, essentially
xmin ymin xmax ymax
   0   80  240   80
> pt = st_as_sfc("POINT(0 90)", crs = 4326)
> st_intersects(pol, pt) # polygon does not include pole? No:
although coordinates are longitude/latitude, st_intersects assumes that they are planar
Sparse geometry binary predicate list of length 1, where the predicate was `intersects'
 1: (empty)
```

In both cases warnings are emitted, but answers are plainly wrong.

Further problems, connected to this are:

- an essential feature of current handling of *all* spatial data is the *bounding box*, which is computed by finding the minimum and maximum geodetic coordinate values; on a sphere this box does not yield a region that contains the geometry. The natural definition of a region on a sphere is that of a *cap*, defined by a center point and a radius (angle);
- having a good measure for the bounding *cap* is needed to inform the PROJ 6 and the upcoming PROJ 7 library to choose ellipses for more accurate reprojection;
- all spatial indexes now provided by `sf` and `stars` assume two-dimensional Cartesian coordinates; this also misses connectedness across the antimeridian (180 degree West) and the poles;
- `sf::st_graticule`, used by all plotting functions including `ggplot2::geom_sf` is now a fragile function that uses a lot of heuristics; with proper geodetic geometry operations it can be implemented robustly.

Previous attempts to this problem include:

- package `sf` warns the user when it makes the assumption that *geodetic* coordinates are taken as *Cartesian* coordinates; for small areas in many cases this is acceptable, but the software doesn't help distinguishing these cases from those where it is not acceptable;
- package `lwgeom` extends the `sf` package with functions found in the geometry engine empowering PostGIS (liblwgeom), and provides several functions that correctly compute ellipsoidal measures (e.g. for distance, area); it is used by `sf` when needed. Liblwgeom is an incomplete implementation of spherical geometries that does not provide e.g. geodetic unions, intersections or buffers.

This problem should be tackled because a majority of spatial data users

- now get a warning too often, will start ignoring it, also when it should not be ignored;
- are unlikely to properly teach themselves the reprojections currently needed to tackle these problems;
- will find that modern systems including BigQuery GIS, Google Maps or Google Earth Engine have resolved this problem.

It should also be tackled because it *can* be tackled: the library empowering these modern systems, `S2 Geometry`, is open source, well maintained and documented, and can be ported to R.

The proposal

Overview

R-global will complement the modern R-spatial stack (packages `sf` for points/lines/polygons and `stars` for raster data) to correctly handle datasets with geodetic coordinates (degrees longitude/latitude) and no longer require the conversion to appropriate projections for this. This affects the analysis of all datasets that are global as well as local datasets that cover the North or South poles or the antimeridian (the -180 degree meridian). It will also give better results for datasets spanning larger areas, and pave the way for plotting defaults that do not have latitude and longitude as perpendicular axes (but e.g. azimuthal perspective or orthographic projections, those that give the illusion of looking at the Earth from space).

It will do so by developing an R package that uses the `S2 Geometry` open source library to provide all geodetic geometry functions required for this. Packages `sf` and `stars` will use this package whenever required.

Detail

The minimum viable product is an R package that

- uses the `S2 Geometry` library to provide functions for `st_intersects`, `st_contains`, `st_intersection`, `st_union` and all other binary predicates and geometry operations currently provided by `sf` and available in `S2 Geometry`, for simple feature geometries having geodetic coordinates,
- provides an alternative to the handling of bounding boxes for data with geodetic coordinates by using caps (a center point and a radius).

The architecture will be kept simple:

- C++ functions (using `Rcpp`) will be written to convert between the `S2 Geometry` data structures and R simple feature geometries
- `Rcpp` will be used to write wrapper functions around the `S2 Geometry` classes and methods, such that they work on simple feature geometries
- `sf` and `stars` functions will use the new package when it is available, or otherwise emit warnings when appropriate (as they now do)
- `sf::st_graticule` will use the new functionality when available

Users of `sf`, `stars` (now at 110K and 10K downloads per month, respectively) and downstream packages will automatically adopt the new functionality once the new package is on CRAN and binaries are available for all platforms (OSX, Windows).

Project plan

Start-up phase

The project will run from Jul 1st, 2019 to Apr 1st, 2020. Licence will be all Apache 2.0 (The `S2 Geometry` library is also licenced under Apache 2.0).

Regular, quarterly reports will be given to the R Consortium. The user community will be informed by a series of blogs on the r-spatial.org blog:

- one blog outlining the project's ambitions
- at least one blog on progress, halfway the realisation, at moments when more user involvement is sought
- a final blog on what has been realised when the project has finished

Technical delivery

The project will deliver an R package that contains the `S2 Geometry` C++ library, and interfaces it to the modern stack of spatial packages (`sf`, `stars` and downstream, including `ggplot::geom_sf`).

Other aspects

Blog posts on r-spatial.org will announce and report on progress. Attention to the project will be drawn on Twitter, in particular when new blog posts are published on r-spatial.org. Progress will be presented at the UseR! 2020 meeting in Saint Louis or at the 2020 rstudio::conf meeting in San Francisco.

Requirements

The problem is very clear and evident; the solution path is too. There are no external requirements that have to be fulfilled to make this project actually happen.

People

The actual programming will be done by Edzer Pebesma. Roger Bivand and Michael Sumner will play an advisory role, will be shared in discussions about the API to choose, and will be asked to test.

Processes

A code of conduct, similar to the one currently in place for package `sf` will be used. The code will be developed under the Apache 2.0 license. The s2geometry library is also available under the same license.

Funding

For the realisation of this project, US $ 10,000 is requested.

Summary

The costs breakdown is as follows:

- writing conversion functions from S2geometry classes to and from simple features: $ 2500
- interfacing S2geometry functions to R methods: $ 2000
- R support for spatial indexing on the spere: $ 2000
- writing geodetic coordinate support for `stars` regular, rectilinear, and curvilinear grids: $ 2500
- writing documentation (pkgdown site, vignettes) $ 1000

Success

The project is succesful if an R package is available on CRAN (source, OSX and Windows binaries) that

- provides comprehensive geodetic geometry operations, addressing the flaws stated in the "problem" section
- provides spatial indexes on the sphere to users of the R packages `sf` and `stars`
- is well documented, and blogs on r-spatial.org and tweets have informed users about it

Definition of done

The project is finished when an R package is on CRAN that provides:

- conversion functions from `S2 Geometry` classes to and from simple features
- interfaces for R methods to `S2 Geometry` geodetic geometry functions
- R support for spatial indexes on the spere
- geodetic coordinate support for `stars` regular, rectilinear, and curvilinear grids
- documentation in the form of package vignettes

and when the promised blogs have appeared on r-spatial.org.

Measuring success

- The blogs on r-spatial will indicate the project progress
- The R package will be developed on a GitHub repository under the r-spatial organisation, so that issues and commit messages can be tracked to see the project status
- Regular reports to the R consortium will be provided

Future work

We believe that this proposal is an important step in the direction of a modern, coherent and robust ecosystem of R packages for spatial data science. Follow-up steps to which this proposal will positively contribute include:

- improved support for coordinate reference system (following the new features in PROJ 7.0.0 and support of the WKT2 standard by GDAL 2.5.0);
- improved plotting capabilities with more generic and flexible, potentially automatic handling of projections chosen on the fly (e.g. by a virtual globe interface to massive image collections, similar to Google Earth Engine);
- various spatial analysis packages will be able to benefit from the functionality of the new package, including `spatstat`, `gstat` and many others.

Key risks

Risks that we can think of now include:

- The `S2 Geometry` library has dependencies that are difficult to port to CRAN Windows or OSX
- Mapping simple features to `S2 Geometry` data structures will impose difficulties

Both risks should be low-moderate: The s2 package currently on CRAN is a proof of concept for the first, the second one should be overcome by reading documentation and asking questions on the appropriate google group for S2geometry users/developers.