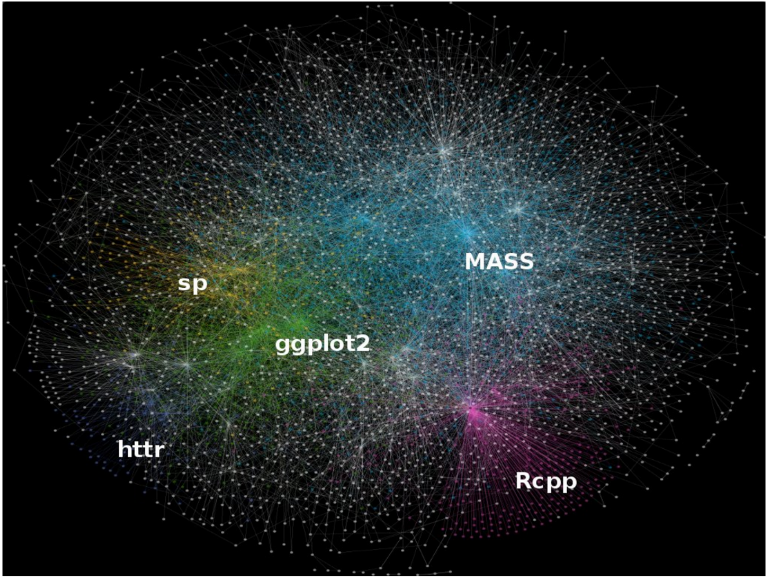
Using R for Large Spatiotemporal Data Sets

Edzer Pebesma

ifgi
Institute for Geoinformatics
University of Münster

EGU 2017, Session IE3.6, Tue, 25 apr 2017



Colin Gillespie @csgillespie · Apr 18

Updated #rstats dependencies map of CRAN (original by @RevoAndrie see blog.revolutionanalytics.com/2015/07/the-ne…)

pic.twitter.com/4hXpnu8O4A

7 23

How large are the datasets YOU analyse with R?

Current limitations:

- R: in-memory
- `raster`: on-disk
- `dplyr`: in-database

Sources of large geo-datasets:

- remote sensing (hyperspectral; time series)
- laser altimetry
- (climate) model results

Challenges:

- avoid download
- distributed storage & computing

Outlook to `http://r-spatial.org` developments:

1. `sf`: simple features for R (done)
   - handle geometry in a list-column in a `data.frame` or `tibble`
   - pipe-friendly, `dplyr` compat, tidyverse, `st_join` for spatial joins
   - merges vector capacity of `sp`, `rgdal`, and `rgeos`
   - coordinate conversions and transformations
   - `geom_sf` support in `ggplot2`
2. `stars`: spatiotemporal tidy arrays for R (still to do)
   - multidimensional arrays with space and time among its dimensions
   - e.g. non-raster time series, raster time series with multiple, or mixed attributes
   - extend on-disk to remote, in-cloud storage (API)
   - pipe-based, dplyr-style workflows
   - develop workflow on small samples, flexibly downsampling dimensions

Both projects enjoy support from the

R consortium

Why replace sp with sf?

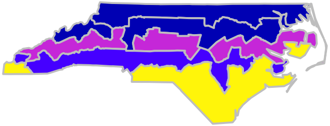
- sustainability: rewrite 15+ years code of sp, rgdal, rgeos
- use modern GDAL and GEOS libraries, and modern C++11 & Rcpp interfaces
- implementing simple features ISO standard:
  $\Rightarrow$ 1:1 mapping and round tripping between R and databases, geojson, LOD, etc.
- direct Well-Known Binary (WKB) read/write and S3 is much faster than `rgdal`!
- "tidy" spatial analysis:
  - `data.frame` based, easier to understand data structure
  - implements `dplyr` verbs, with sticky geometry
  - ggplot2 support by `geom_sf`
  - direct DBI inteface to spatial databases

sf examples

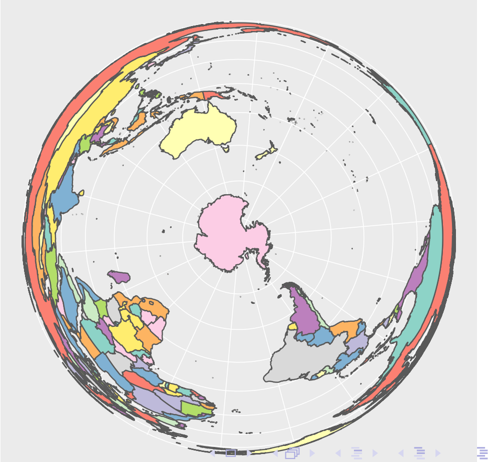
```
> library(sf)
> nc = read_sf(system.file("gpkg/nc.gpkg", package="sf"))
> library(units)
> (a <- nc %>% mutate(area = st_area(.)) %>%
+   group_by(group = c(rep(1:5, each = 20))) %>%
+   summarize(area = set_units(sum(area), km^2)))

Simple feature collection with 5 features and 2 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 yma
epsg (SRID):    4267
proj4string:    +proj=longlat +datum=NAD27 +no_defs
# A tibble: 5 × 3
  group          area              geom
  <int>        <units>  <simple_feature>
1     1 22022.91 km^2 <MULTIPOLYGON...>
2     2 22866.40 km^2 <MULTIPOLYGON...>
3     3 26491.78 km^2 <MULTIPOLYGON...>
4     4 24139.21 km^2 <MULTIPOLYGON...>
5     5 31511.46 km^2 <MULTIPOLYGON...>

> par(mar=rep(0,4))
> plot(a["area"], main = "", border = 'grey')
```



```
> library(ggplot2)
> library(maps)
> world1 <- sf::st_as_sf(map('world', plot = FALSE, fill = TRUE))
> laea <- "+proj=laea +y_0=0 +lon_0=155 +lat_0=-90 +ellps=WGS84"
> world2 <- sf::st_transform(world1, laea)
> fill = sf.colors(xc = factor(1:253))
> ggplot() + geom_sf(data = world2, fill = fill)
```



`stars`: spatiotemporal tidy arrays

Resolve R's native array limitations:

- 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, only character `dimnames`.

Resolve `raster`'s limitations:

- 2 dimensions are always space (=raster)
- cope with n-D dense arrays, rather than 3-D raster:
- stack is either multiple colors, or multiple times, not both
- brick or stack? low level details exposed to users
- handle I/O natively (not through rgdal)
- handle data sizes larger than those fitting on local disk
- S4

`stars`' approach: support

- array dimensions that *can* be space, time, can also be e.g. spectral, simulation, model
- e.g. arrays with features (1D) x time1 x time2 (2D), with time1 time of prediction and time2 forecast lag
- heterogeneous records, just like `data.frame`
- S3 `data.frame` objects that proxy real data, like `dplyr`'s database proxies
- proxy data are not the first *n* records, but a thinned (subsampled) version of the array, revealing large-scale structure
- dense, but both regular *and irregular* arrays
- distributed storage, distributed computing, using lazy, distributed evaluation
- fast development times by developing the workflow on subsampled proxy, when finished applying it to the full data
- pipe-based, `dplyr`-style verbs