[PD-05-015] R for Geospatial Analysis and Mapping

Abstract

R is a programming language as well as a computing environment to perform a wide variety of data analysis, statistics, and visualization. One of the reasons for the popularity of R is that it embraces open, transparent scholarship and reproducible research. It is possible to combine content and code in one document, so data, analysis, and graphs are tied together into one narrative, which can be shared with others to recreate analyses and reevaluate interpretations. Different from tools like ArcGIS or QGIS that are specifically built for spatial data, GIS functionality is just one of many things R offers. And while users of dedicated GIS tools typically interact with the software via a point-and-click graphical interface, R requires command-line scripting. Many R users today rely on RStudio, an integrated development environment (IDE) that facilitates the writing of R code and comes with a series of convenient features, like integrated help, data viewer, code completion, and syntax coloring. By using R Markdown, a particular flavor of the Markdown language, RStudio also makes it particularly easy to create documents that embed and execute R code snippets within a text and to render both, static documents (like PDF), as well as interactive html pages, a feature particularly useful for exploratory GIS work and mapping.

Keywords: open source, programming, R, spatial analysis

Author & citation

Engel, C. (2019). R for Geospatial Analysis and Mapping. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2019 Edition), John P. Wilson (Ed.). DOI:10.22224/gistbok/2019.1.3.

Explanation

1. Definitions

Markdown: A plain text formatting syntax and a software tool that converts the plain text formatting to HTML to be used in a website.

simple features: A standardized way to encode geographic vector data. It specifies a common storage and access model. A simple feature is defined as having both, spatial and non-spatial attributes.

2. Introduction

R is a programming language as well as a computing environment to perform a wide variety of data analysis, statistics, and visualization. Supported by the non-profit R Foundation for Statistical Computing and a large user community it has enjoyed increasing popularity over the last decade. One of its strengths is its modularity. So called “packages” can be downloaded and installed individually to extend the R base functionality. There are 1000s of such packages for many different special areas of interest, among them a wide array of geospatial packages for spatial data processing, analysis, and visualization. The official repository for R packages is the Comprehensive R Archive Network (CRAN), and it is where all of the GIS packages discussed in this entry can be found.

Different from tools like ArcGIS or QGIS that are specifically built for spatial data, GIS functionality is just one of many things R offers. And while users of dedicated GIS tools typically interact with the software via a point-and-click graphical interface, R requires command-line scripting.

One of the reasons for the popularity of R is that it embraces open, transparent scholarship and reproducible research. It is possible to combine content and code in one document, so data, analysis, and graphs are tied together into one narrative, which can be shared with others to recreate analyses and reevaluate interpretations. This makes research transparent, particularly as an increasing number of journals and funding agencies expect research to be reproducible.

Many R users today rely on RStudio, an integrated development environment (IDE) that facilitates the writing of R code and comes with a series of convenient features, like integrated help, data viewer, code completion, and syntax coloring. By using R Markdown, a particular flavor of the **Markdown** language, RStudio also makes it particularly easy to create documents that embed and execute R code snippets within a text and to render both, static documents (like PDF), as well as interactive html pages, a feature particularly useful for exploratory GIS work and mapping.

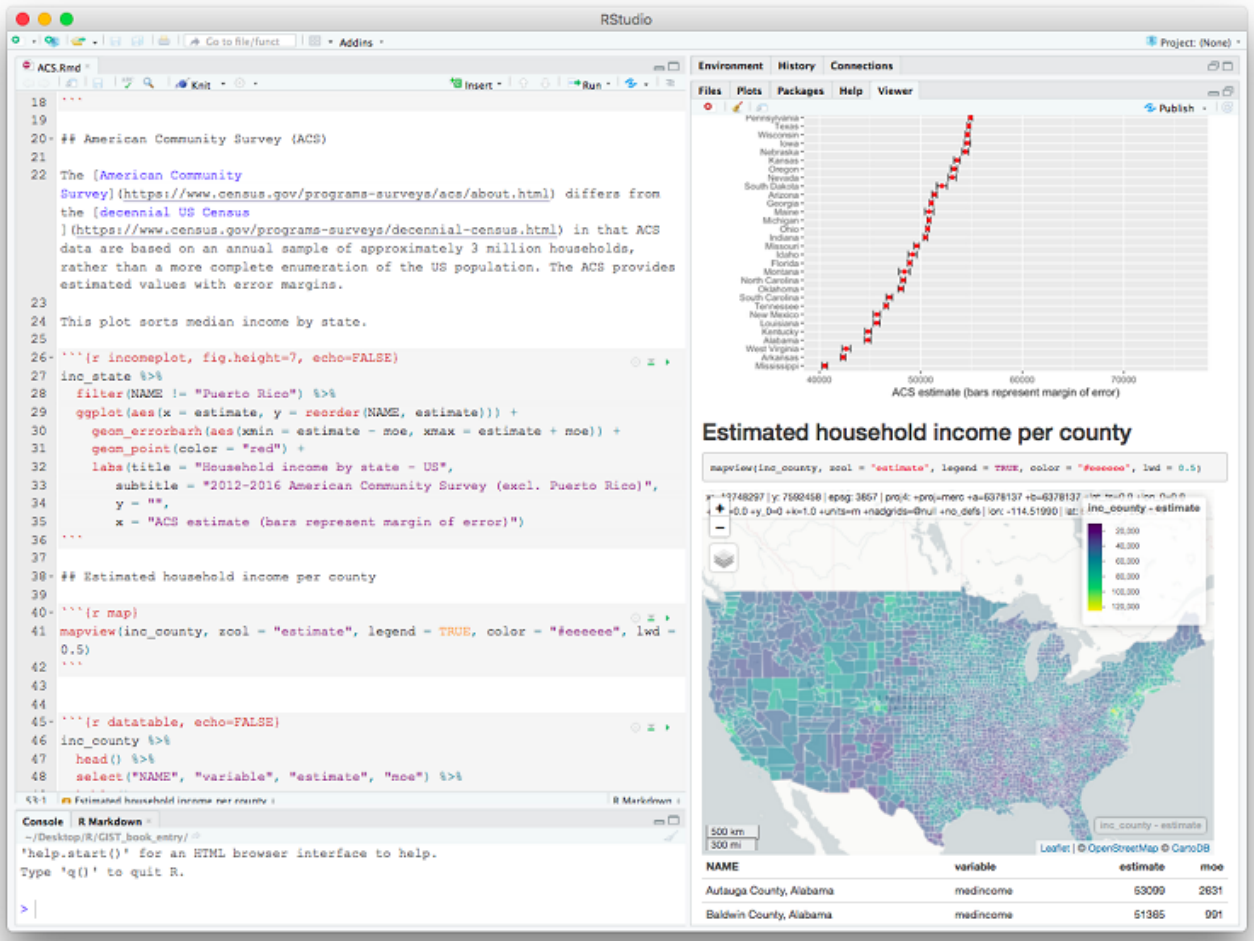

Figure 1. Example of R Studio in use.

Both R and RStudio are open source and freely available for a variety of operating systems, including Windows, macOS, and Linux. An alternative free version of R (R Open) is distributed by Microsoft with performance enhancements supporting parallel computing and code reproducibility.

3. GIS Objects in R

3.1 Vector Objects

There are currently two main approaches used in R to handle geographic vector data. They are implemented in two packages, sp (Bivand et al 2013) and sf (Pebesma 2018). The sp package is historically the older one. Its development began in the early 2000s in an attempt to standardize how spatial data would be treated in R and to allow for better interoperability between different analysis packages that use spatial data. The package (first released on CRAN in 2005) provides classes and methods to create points, lines, and polygons. (sp also implements raster grids, though they are rarely used today, particularly since a dedicated raster package was released in 2010.)

The implementation of spatial objects in sp follows a more “object oriented” thinking.

Spatial objects in sp have “slots” that hold different kinds of information, most fundamentally the coordinates of the bounding box and the Coordinate Reference System (CRS). This basic structure is then extended, depending on the particulars of the spatial object (point, line, or polygon), to hold geographic coordinates, a unique ID for each element, information about plotting order, if appropriate, and more. There is also a “data” slot for attribute data.

In addition, the sp package contains some functions to manipulate with sp objects, for example to retrieve the bounding box or to set or retrieve the CRS. Hundreds of R packages today refer to the spatial data types that are implemented in sp.

The more recent sf package was first released on CRAN in late 2016. It implements a formal standard called “**Simple Features**” that specifies a particular storage and access model for spatial vector geometries. This standard has been adopted widely, not only by spatial databases such as PostGIS, but also by open standard formats such as GeoJSON. sf implements this standard natively in R, so data are structured and conceptualized very different from sp objects.

sf objects are stored as a table (in R called a “data frame”), where each row represents one feature and columns store its attributes. One special column in the table is designated to hold the geographic coordinates. Each cell in this particular “geometry” column contains a list with the coordinate pairs that correspond to the feature in that row. Additional information, like the CRS, bounding box, and geometry type are stored as metadata to the data frame.

Since sf spatial objects are essentially R data frames with some special properties and data frames are very common and widely used in R, it is possible to take advantage of a series of functions that R already has available to work with data frames.

sp and sf are not the only way spatial vector objects are conceptualized in R. Other spatial packages may use their own class definitions for spatial data, for example the spatstat package, which focuses on point pattern analysis, and defines its own (ppp - point pattern) object. There are functions available that can convert from one spatial object type to another.

3.2 Raster Objects

The most common package to handle spatial raster data in R is called raster. Because of their regular structure, raster objects can be constructed, if a series of parameters are given, including the number of columns and rows (dimension), the cell size (resolution), the bounding box (extent), and the CRS, as well as cell values, if any. Like sp objects, raster objects in R store this information in slots.

With increasing use of high quality, high resolution, - and thus large - raster files, working with such files can be a challenge. Files are typically loaded into computer memory when they are processed, which can be a limiting factor, particularly in a desktop setting, so operations can become very slow or even impossible. The raster package is designed and written in a way that accessing and processing very large rasters is possible without the need to load the entire file into memory. It can work directly with raster datasets that are stored on disk and only loads information about the structure of the data into memory. This information is then used during computational operations to process the data in chunks.

The raster package can work with both single and multilayer raster files, such as multi-band satellite images or a time series of temperature values for a given geographic area. For higher-dimensional datasets, for example time series of multi-band satellite images, the newly released stars package is available, which targets more general raster (and vector) data cubes, and attempts to strongly integrate with the sf package.

3.3 Reading and Writing Spatial Data

Like many open source GIS tools, R relies on the Geospatial Abstraction Library (GDAL) to read and write spatial data. This means that a large variety of spatial formats, including GeoJSON, KML, ESRI Shapefiles, GeoTiffs, and many more, can be read into R, and similarly to be written out to disk. sf directly links to GDAL, while sp requires a separate package. The rgdal package, which provides bindings for GDAL, is most commonly used. It is also possible to directly read from and write to spatial databases such as PostGIS, and the sf package for example, provides functionality to do that.

A series of R packages link to external repositories for spatial data and allow to download data directly into R objects from there. Examples are rnaturalearth for the Natural Earth public domain map dataset, getlandsat for the ongoing collection of satellite imagery produced by Landsat 8, or osmdata to download and import OpenStreetMap data.

Instead of reading data into R there are also R packages that contain spatial datasets. These packages can be a helpful start for the beginner as they often demonstrate use cases. The spData package, for example, contains a diverse array of data stored in a range of file formats.

4. Geoprocessing and Spatial Statistics

Geoprocessing of sp objects relies on the (separate) rgeos package, which serves as an interface to topology functions for sp objects using the GEOS library, an open source C++ library for topology operations on geometries. The sf package includes numerous geoprocessing functions, as it links to GEOS directly. These operations include manipulations of spatial properties, e.g. topological relationships, spatial joins, aggregation or extraction based on location, distance measurements, and geometric operations, like buffers or centroids.

The already mentioned raster package includes raster operations for local, zonal, focal, and global statistics. It implements functions to perform raster data manipulations that are common in GIS. This includes not only map algebra, but also image transformations, like reprojection, change of resolution, cropping, resamplig, reclassifications, and more.

For statistical analysis, a variety of R packages are available. Just to list a few, spdep provides functions to analyze spatial dependence, for example Moran’s I test for residual spatial autocorrelation; gstat provides basic functionality for univariable and multivariable geostatistical analysis, including variogram modelling, kriging and many more (Pebesma, 2004); spatstat is a quite comprehensive package for analyzing spatial point patterns (Baddeley et al 2015). It contains thousands of functions for exploratory data analysis, model-fitting, simulation, spatial sampling, model diagnostics, formal inference, and plotting.

5. Making Maps in R

R has become increasingly versatile when it comes to mapping. It is possible to deliver a wide variety of geographical maps, from static, publication quality printed maps to highly interactive webmaps. Many of the already mentioned packages, like sp, sf, spatstat, or raster, include their particular plotting function which is tailored to the respective spatial objects the package uses. Among the earliest dedicated mapping packages to produce vector maps is the maps package, which also includes built-in map data, a convenient way for the novice to get started.

A widely used alternative is ggplot2, a powerful plotting package for R. It is not specifically geared towards mapping, but it has a large user community and is frequently used to produce high quality maps. ggplot works with the notion of 'layers' that can be added one by one, which allows users to superimpose either different visualizations of one dataset (e.g. a scatterplot and a fitted line) or different datasets (like different layers of the same geographic area). Due to its popularity, ggplot has a series of add on packages. A mapping package to be used in conjunction with ggplot is ggmap, which follows the layer-syntax of ggplot and facilitates the downloading of tiled basemaps from different services, like Google Maps and Openstreet Maps. There also are a few drawbacks. In order to plot sp objects with ggplot they need to be converted in a prior step, however ggplot (ggplot2 version 3.0.0 and above) can read and plot the sf format directly. ggplot also does not natively support raster objects, so they need to be rendered, too. rasterVis for example provides a wrapper function to plot raster objects with ggplot2.

One of the most widely used open-source libraries to generate webmaps is Leaflet, which is written in the JavaScript language. Typically, webmaps are composed of tiled basemaps with added vector type overlays and/or place markers and have interactive capabilities, like panning and zooming. The leaflet R package wraps the JavaScript library and makes it easy to create and display these webmaps in R.

The leaflet package is very flexible and allows for many elements of the map to be fine tuned and adjusted. Several R packages are built on top of the leaflet package, taking care of some of these settings, for example by assuming a default layout for the legend, so less coding is necessary to produce a map with interactive capabilities. Examples are the packages mapview and tmap, good entry points for the beginner. It is also easily possible to create animated maps (e.g. for easy visualization of spatiotemporal sequences).

Finally, R's shiny framework provides great flexibility to generate maps that are truly interactive, as it can take user input in multiple forms (through sliders, check boxes, drop down selections, and more) process it computationally (for example recompute a regression model with changed input values) and return output in various formats, including visualizations and maps.

6. Conclusion

From its original implementation as statistical programming language, R has come a long way and has become very versatile in supporting spatial analysis. New and improved GIS

packages are developed continuously, and consequently some packages are more mature than others. Like users of any open software, R users face the challenge of keeping their code up-to-date with new releases and new and updated packages, though there are strategies to mitigate this, for example the checkpoint package by Microsoft. Computer memory also can pose a limit, but there are constant efforts to improve performance and the possibility to connect to external databases can alleviate some of this. R integrates well with other open source GIS tools like PostGIS, SAGA, QGIS and GRASS.

So called “task views,” created and maintained by R community members, provide an overview of R packages that are relevant for a particular topic. The task view on “Analysis of Spatial Data” is a very useful starting place for an overview of the range of spatial R packages available. Many packages include so called “vignettes.” These are often very helpful mini-tutorials that demonstrate and explain practical uses of the software around selected problems. Listed below are the Packages referenced earlier, ones that are commonly used when undertaking spatial analysis and mapping projects.

Table 1. A partial list of R Packages relevant for spatial analysis & mapping. All packages are available for download from the Comprehensive R Archive Network (CRAN)

Package	Description
sp	classes and methods for spatial data
sf	simple features for R
raster	geographic data analysis and modeling
stars	scalable, spatiotemporal tidy arrays
rgdal	bindings for the "geospatial" data abstraction library
rgeos	interface to Geometry Engine Open Source (GEOS)
spData	datasets for spatial analysis
osmdata	import OSM data as simple features or spatial objects
getLandsat	get Landset 8 data from Amazon public data sets
rnaturalearth	world map data from Natural Earth
spatstat	spatial point pattern analysis, model-fitting, simulation tests
gstat	spatial and spatiotemporal geostatistical modeling, prediction, and simulation
spdep	spatial dependence: weighting schemes, statistics, and models
maps	draw geographical maps
ggplot2	data visualizations using the grammar of graphics
ggmap	spatial visualization with ggplot2
rasterVis	visualization methods for raster data
leaflet	create interactive web maps with the JavaScript "Leaflet" library
mapview	interactive viewing of spatial data in R
tmap	thematic maps
shiny	web application framework for R
checkpoint	install packages from snapshots on the checkpoint server for reproducibility

Particularly for those who are new to scripting and programming, there is a learning curve, but it pays off to be able to automate tasks and to create reproducible code.

References

Baddeley, A., Rubak, E., & Turner, R. (2016). Spatial point patterns: methodology and applications with R. Boca Raton; London; New York: CRC Press, Taylor & Francis Group.

Bivand, R., Pebesma, E. J., & Gómez-Rubio, V. (2013). Applied spatial data analysis with R (Second edition). New York: Springer.

Pebesma, E. J. (2004). Multivariable geostatistics in S: the gstat package. Computers & Geosciences, 30(7), 683–691.

Pebesma, E. J. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10(1), 439-446.