Interactive data manipulation in mapview

01 Jul 2016

Applicants: Tim Appelhans, Florian Detsch, Christoph Reudenbach

Environmental Informatics Marburg, Faculty of Geography, University of Marburg, Germany; tim.appelhans@staff.uni-marburg.de

Supporting authors: Edzer Pebesma, Kenton Russell, Michael Sumner

R has become very popular in the geospatial community as it provides a very powerful environment for geospatial analysis with well designed classes for spatial data represenatation. Packages sp and raster form the basis and there is active development for integrating modern approaches such as simple features or approaches focused on tabular representation such as spbabel. These classes enable leveraging of R's infinite arsenal of statistical algorithms and thus provide researchers and analysts with more powerful tools than any commercially available geospatial software. Data visualisation has always been one of R's strengths in comparison to other data analysis platforms. This is, however, only partially true for geospatial analysis tasks. Geospatial data visualisation is two-fold.

1. **Cartographic presentation of results:**
   In most cases geospatial analysis will end with the production of one or more maps showing the intended outcome in a spatially explicit way. There are many packages that provide suitable functionality for creating static maps for spatial data in R. Depending on the discipline and the intended outcome users can choose from a wide variety of packages for final presentation grade map production (such as GEOmap mainly for geological mapping, rworldmap for mapping of global data, ggmap to plot spatial data on top of background maps or tmap for thematic mapping to only name a few; many more packages are mentioned in the CRAN Task View: Analysis of Spatial Data). Other packages provide bridges to external software, such as plotKML to view spatial and spatio-temporal data in Google Earth or gmt which provides a bridge to the General Mapping Tools. More recently, interactive (web-) mapping has become popular and several packages provide options for this. The most popular is without doubt leaflet which uses htmlwidgets to access the leafletjs JavaScript library from R. Interactivity such as zooming, panning and querying of attributes is very desirable as geospatial data is by definition multi-dimensional and it is very common to perform analyses across a range of spatial scales where details matter as much as broad patterns. Such interactive features become even more important for quick inspections during the analysis workflow.

2. **Visual exploration during workflow:**
   An integral part of any spatial data analysis workflow is visual inspection of intermediate analysis results. Virtually every open or closed source geospatial software provides an interactive data visualisation module where the visualisation is updated throughout each of the steps in the analysis. These visual modules usually provide easy access to the data through layered plots that can be changed and explored with one or two mouse clicks. This makes visual exploration of complex spatial data easy and quick. Such quick visual inspection capabilities are currently lacking in R. Package mapview was created to fill this gap. It is in large parts built on top of the `leaflet` package and enables interactive visualisation of multiple layers with one or only a few lines of code, e.g. `mapview(x, burst = TRUE) + y` will visualise all attributes of spatial object `x` as individual layers and additionally add a layer of the locations of object `y`. Furthermore, all attributes of `x` and `y` are fully queryable via popups (see `?mapview` for examples). As such, it provides an interactive version of `plot()` or `spplot()` for spatial data. Spatial data can quickly grow to large proportions of many tens of thousands of features, so `mapview` also includes some special functionality to cope with such amounts. `leaflet` based maps can become rather unresponsive when data sets are "big". Furthermore, at least in the development version, `mapview` provides functionality to view spatial data with arbitrary projection, which is necessary in many applications, e.g. for people working in Arctic environments.

So far, we have outlined why visualisation is an important part of geospatial data analysis and have shown several possiblities how it can be achieved.

A further aspect of geospatial analysis, which is implemented in virtually all open or closed source geospatial software suites is interactive data manipulation. This is what this application is intended to address.

The problem

The problems we will solve are:

1. With very few exceptions, there is currently no way to manipulate spatial data in an interactive manner in R. One noteworthy exception is `drawExtent()` in the `raster` package which lets the user select a geographic sub-region of a given Raster* object by clicking (twice) on a static plot of the visualised layer and saving the resultant extent or subset in a new object (if desired). Such operations are standard spatial tasks and are part of all standard spatial toolboxes. Having to select from a static plot is not optimal as the region of interest may be small in comparison to the dimension of the original layer and thus precise queries may not be possible. Having interactive capabilities to first navigate to the desired region and then perform some interaction with the intended layer is much more desirable.
2. `htmlwidgets` based interactive visualisation packages such as `leaflet` and `mapview` provide great possibilities for data exploration across spatial scales, however, thus far this is a one-way ticket. There is currently no straight-forward way to get data from the JavaScript side back into R. The most suitable (and popular) way to achieve such reactive two-way communication is via package shiny (see e.g. here, here and here). These solutions are highly optimised for the individual task and focus of the respective app. A generalised framework is likely beyond the cpapbilities of `shiny`. In addition, most apps implemented in `shiny` do not pass back the manipulated data into the users environment as a modified or new object.
3. The JavaScript `leafletjs` offers plugins/extensions to interact with rendered map content. As an example the plugin Leaflet.draw shall be mentioned here.
4. There are tools that provide reactive capabilities on the JavaScript side such as mobx. There are also first implementations regarding R integration of such tools, e.g. crosstalk. The latter can easily be used with the new flexdashboards but, again, this is focused on inter-htmlwidget communication and there are thus far no capabilities to pass the selected data back to the users R environment.
5. Data exchange between `R` and `JavaScript` is readily available through packages such as jasonlite or geojsonio so that transfer between the environments can easily be achieved.

Thus, in this application we propose to implement two-way data exchange between R and JavaScript to enable easy and user-friendly interaction with spatial data utilising the above mentioned tools. Such functionality is useful in general, but for spatial data it is especially desirable as pattern based decisions/selections are very common and form an integral part of many spatial analyses.

Which users will benefit from solving these problems?
It is expected that the proposed two-way data exchange and reactive capabilities will enhance spatial analysis experience for all users working with geospatial datasets. With the implementation of simple features in R, this group is likely to grow as it will make geospatial data accessible to a wider audience. In addition, two-way data exchange with `htmlwidgets` is an unresolved frontier. Iterating, testing, and working toward a stable generalized approach will help all `htmlwidgets` authors and users.

Furthermore, we intend to implement the two-way data chain so that it will integrate with the highly popular maggittr package meaning that chains such as `x %>% doSomething() %>% mapviewToInteractivelySelectStuff() %>% doMoreComputation() %>% visualiseWithLeaflet()` will work with `x` being a spatial object.

In general, we are aiming for an unopiniated framework that is not limited to `mapview`, but will be beneficial to all `htmlwidgets` based visualisation tools so that the developed functionality will benefit the general community by providing interactive reactivity for a wide range of end users.

Given that the use of geospatial data is generally expected to grow in the future (e.g. via Google's location history) we anticipate that intuitive (interactive) data manipulation will in particular aid users that are non-experts in the domain of spatial data analysis.

The plan

We want to solve the problem by carrying out the following steps (M1 refers to month 1):

1. develop an R package that enables general two-way (spatial) data exchange between R and JavaScript content (M1-6)
2. implement relevant JavaScript plugins to provide user-friendly spatial data manipulation/interaction on the JavaScript side (M1-6)
3. streamline current `mapview` functionality so that it provides a generalised framework that is completely compatible with `leaflet` via a common `htmlwidgets` infrastructure (M3-9)
4. ensure that the proposed two-way data flow is compatible with all relevant spatial classes in R, including `sp`, `raster` and the currently developed `simple features` (M3-9)
5. write user-oriented tutorial vignettes showing how to use the two-way data chain (M6-12)
6. Collect and process community feed back (M6-12).

Failure modes and recovery plan:

1. Adoption of different standards for two-way communication for `htmlwidgets` after development. **Recovery plan:** Smooth transition to adopt new standard for `mapview`.
2. Unforeseen problems with `leafletjs` or other open source JavaScript technologies on which we rely. **Recovery plan:** Try to (i) develop own JavaScript plugins to solve issues embedded in `mapview`, (ii) implement reactivity on top of different platform such as Open Layers.
3. Inability to find suitable student assistant to complete the task. **Recovery plan:** Hire professional developper to implement core functionality and continue development on this basis.

How can the ISC help

The following table contains the cost items.

Item	Cost
Employ a student assistant for one year (10 hrs/week)	€ 6500
Present the results at UseR! 2017	€ 1500
Total:	€ 8000 (9100 USD)

Dissemination

Development will take place on github, information will be shared and reactions and contributions invited through r-sig-geo, as well as StackOverflow and GIS StackExchange. The project will use a standard open source license such as GPLv3 or MIT for maximum dissemination. The work will be published in blogposts (quarterly), announced on r-sig-geo (3300 subscribers), and intermediary results will be presented at UseR! 2017. The final result will be published in a paper either submitted to The R Journal or to the Journal of Statistical Software; this paper will be available before publication as a package vignette.