Geographic data analysis and geospatial web applications with R and ReactJS

Dr Layik Hama[*]*— University of Leeds, Leeds Institute for Data Analytics (LIDA)*
Dr Robin Lovelace[†]*— University of Leeds, Institute for Transport Studies (ITS) and Leeds Institute for Data Analytics (LIDA)*

2019-01-28

Summary

Data scientists, and R users in particular, tend to focus on 'back-end' data processing, with visual outputs usually generated statically. Recent developements have made it easier for R developers to deploy applications remotely, the `shiny` package being a notable example. However, such 'pure R' approaches have limitations, especially when it comes to geographic applications and provide little for people wanting to develop APIs. To overcome this issue we developed `geoplumber`, an R package that leaves the fron-tend to native JavaScript code and the back-end to R/databases. This paper describes the approach and highlights the possibilities with a real-world example.

Keywords: Web Applications with R, Rpackage, JavaScript, ReactJS, Web Mapping

1 Introduction

Scripting languages (Mazzoni et al. 2006) have been powering the web for a long time. The most widely used ones are not used for geospatial data processing, for example PHP (Zhao et al. 2012). We know that Python is also a scripting language and there is Flask (Grinberg 2018) enabling web applications in Python.

The statistical programming language R (R Core Team 2019) is a widely used language for data science and, with packages such as `sf` enabling geocomputation. Data scientists who use R and want to deploy their applications often wonder: how can I use R to deploy on the web?

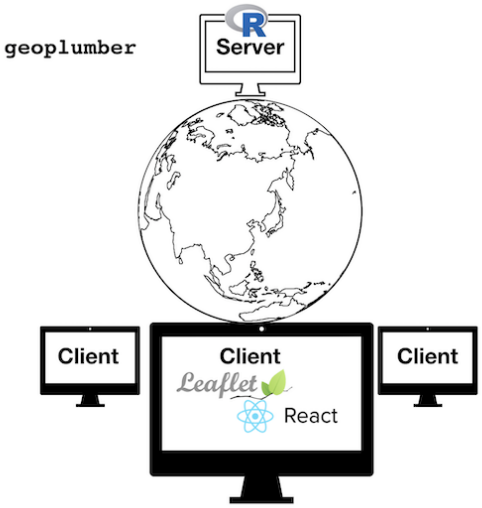


Figure 1: Running R as back-end and ReactJS + LeafletJS at the front-end of a web application.

[*]l.hama@leeds.ac.uk
[†]r.lovelace@leeds.ac.uk

We found out that there had already been some work in this regard, inspired the Python package Flask, called `plumber` (Trestle Technology, LLC 2018). After preliminary checks and checking the codebase, we decided that we could use `plumber` to power the project.

2 Approach: Combining technologies for data science and the web

Facebook developed ReactJS (Fedosejev 2015). The approach is designed for scalability and rapid front end development (Gackenheimer 2015). These advantages, plus prior experience with React, explain our choice of front-end framework, to be combined with `plumber` in the back-end. After developing an ad-hock web app based on this approach, we found there were advantages of generalising the approach. The R packaging system makes the approach accessible to other data scientists wanting a flexible, scalable yet lightweight front-end for their R applications, and encourages testing and feedback. `geoplumber`, which can be installed with the following command, was born.

2.1 Geospatial web applications

JavaScript has a rich set of front-end web mapping technologies, including LeafletJS and Mapbox JS. Since the emergence of WebGL, the possibilities for creating "web application frameworks" for displaying large geospatial datasets, have increased greatly.

The package is a combination of above technologies which are loosely coupled and could be used for non-geospatial purposes. This loose coupling is done with attention, making it accessible to developers who already use ReactJS tools, without needing to engage with the R ecosystem. `geoplumber` is compatible with the `npm` (Node Package Manager), and uses the Create-React-App (CRA) (Banks and Porcello 2017) NodeJS package for deployment.

2.2 Interactive data analysis

A focus of `geoplumber` is interactive data analysis. The package includes functions to support such tasks as adding React or generic JS code to a web application, taking advantage of React's modular design. The result is a "framework" where node packages can be defined for use in the front-end and data can be served from a flexible R-based back-end.

3 Use case: a web app for visualising road traffic casualties

A geoplumber app, as it stands, is a standard `npm` package generated by CRA. For the API, an `R` directory containing a `plumber.R` file is added, which is used by the underlying `plumber` package. To create a `geoplumber` app:

```
dir_name = "/tmp/gisruk19"
library(geoplumber)
gp_create(dir_name)
setwd(dir_name)
gp_build()
```

The directory and files structure of a `geoplumber` applications looks like this:

```
+- R/plumber.R       # back-end code
+- README.md
+- package.json      # npm package file
+- public            # public facing docs
+- src               # front-end JS code.
```

We can then do all our data processing straight from R and serve the data using API end-points. `plumber` works by adding tags in front of standard R functions. Lets get some data using `stats19` package.

```
library(stats19)
accidents = stats19::get_stats19(year = 2017, ask = FALSE)
accidents = dplyr::sample_n(accidents, 500)
accidents = stats19::format_sf(accidents, lonlat = TRUE)
```

```
accidents_geojson = geojsonsf::sf_geojson(accidents)
class(accidents)
```

```
## [1] "sf"         "tbl_df"     "tbl"        "data.frame"
```

```
class(accidents_geojson)
```

```
## [1] "geojson" "json"
```

For example, to geneate an end-point that returns an R object which contains JSON data in a parameter called `accidents_geojson`, we could write a function like this and add into a `geoplumber` app's `R/plumber.R` file:

```
#' @get /api/stats19                              # 1
all_geojson <- function(res){                    # 2
  res$headers$`Content-type` = "application/json"# 3
  res$body <- accidents_geojson                  # 4
  res
}
# copy above, run
# geoplumber::gp_endpoint_from_clip()
# to add it into your geoplumber app
```

In the line with `# 1` comment above, the `@get` part means `/api/stats19` is going to be a HTTP GET path. The function has a `response` parameter which can be modified and returned. In this case, we set the response `content-type` and also load the body of the response object with the JSON object to be returned. Therefore, `@get /api/stats19`, translates into `http://localhost:8000/api/stats19` which would returns the `accidents_geojson` object.

In this example, `stats19` (R. Lovelace et al. 2019) R package is used for data acquisition and processing. Using `stats19` we can get the crashes for years since 1979 from DfT (Department for Transport, United Kingdom). There are functions in `geoplumber` to work on the development and finally deploy our application. A screen shot of an example is shown in Table 1.

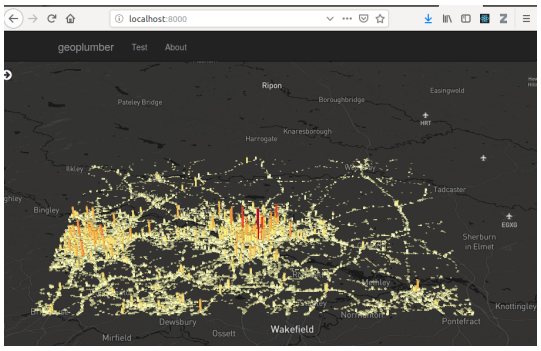
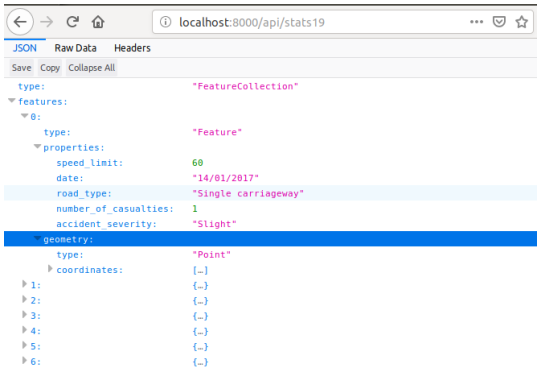
front-end view	The API serving the data
	

Table 1: A geoplumber app for road casualty data. On the left, customised front-end uses Uber's DeckGL (React). On the right the same data served from `/api/stats19` end-point in R.

We can also interactively see the `accidents` object and choose a column to filter using the API from the front-end:

```
geoplumber::gp_sf(accidents,
                  props_list =
                    list(accident_severity = unique(accidents$accident_severity)))
```

If you then visit `http://localhost:8000` on your broswer, it should give you something like Figure 2.

4 Geospatial Databases

Although currently the work in progress repository[1] does not include one, it is possible to add a database of choice to the stack. Due to the light weight Flask/plumber type of API frameworks, it is possible to make use of the full potential of R language. For example, to connect to a MySQL database running on a Linux machine, with username and password defined at the users `~/.my.cnf` file as per MySQL convenctions. We can then create API end-points that can connect to a MySQL instance with a `geoplumber` schema defined in it, using `RMySQL` and `DBI` packages as follows:

```
con <- DBI::dbConnect(RMySQL::MySQL(), group = "my-db")
# we can send SQL queries such as selecting a schema
DBI::dbGetQuery(conn = con, "use geoplumber;")
#> data frame with 0 columns and 0 rows
DBI::dbListTables(con)
#> character(0)
```

Add the output of the above into another end-point:

```
#' @get /api/tables
tables = function(res){
  # ...
  res$body <- tables_list_geojson
  # ...
}
```

[1] See: https://github.com/ATFutures/geoplumber

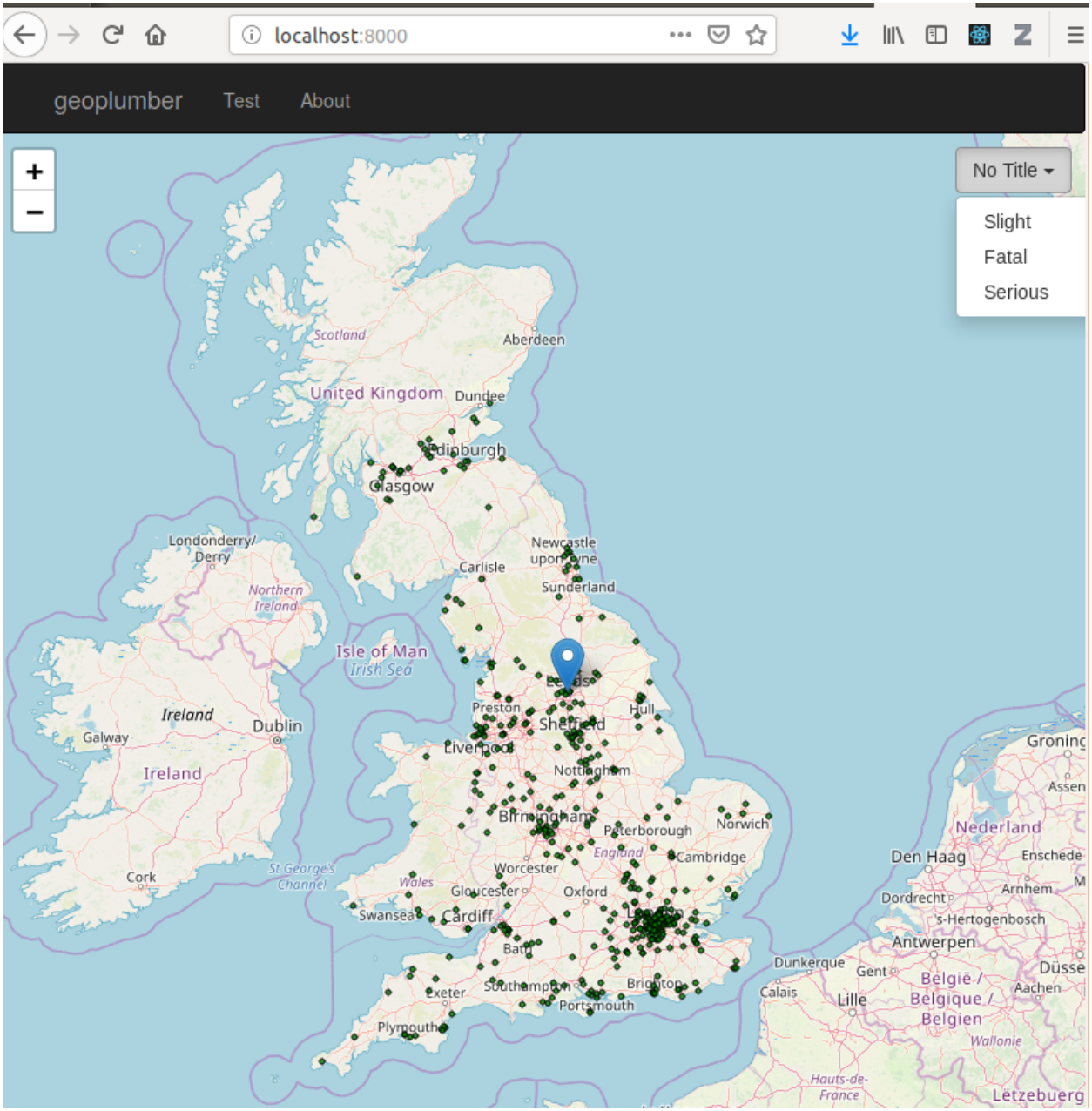


Figure 2: STATS19 sample 500 points across the UK for year 2017. With an interactive dropdown querying R code subsetting the 500 sample dataset.

5 Deployment

Standard deployment documentation[2] for `plumber` application is provided by the developers. In our small scale deployment experience, we have deployed two separate geoplumber apps using Docker virtualization technology, using reverse proxy from a standard Nginx HTTP server. The Dockerfile and other deployment details are available on a GitHub repository[3].

6 Acknowldgements

The work carried out is funded by Consumer Data Resarch Center (CDRC) at Leeds Institute for Data Analytics (LIDA) at University of Leeds. The open source ecosystem enables projects like Linux and can benefit geoplumber immensely. The GitHub repository on GitHub[4] lists all contributors to the package and interested parties are invited to collaborate further.

References

Banks, Alex, and Eve Porcello. 2017. *Learning React: Functional Web Development with React and Redux*. " O'Reilly Media, Inc."

Fedosejev, Artemij. 2015. *React. Js Essentials*. Packt Publishing Ltd.

Gackenheimer, Cory. 2015. "Introducing Flux: An Application Architecture for React." In *Introduction to React*, 87–106. Springer.

Grinberg, Miguel. 2018. *Flask Web Development: Developing Web Applications with Python*. " O'Reilly Media, Inc."

Lovelace, Robin, R Lovelace, M Morgan, L Hama, and M Padgham. 2019. "Stats19: A Package for Working with Open Road Crash Data." *Journal of Open Source Software*. doi:10.21105/joss.01181.

Mazzoni, Silvia, Frank McKenna, Michael H Scott, Gregory L Fenves, and others. 2006. "OpenSees Command Language Manual." *Pacific Earthquake Engineering Research (PEER) Center* 264.

R Core Team. 2019. "R: A Language and Environment for Statistical Computing." Vienna, Austria: R Foundation for Statistical Computing.

Trestle Technology, LLC. 2018. *Plumber: An Api Generator for R* (version 0.4.6). https://cran.r-project.org/web/packages/plumber/index.html.

Zhao, Haiping, Iain Proctor, Minghui Yang, Xin Qi, Mark Williams, Qi Gao, Guilherme Ottoni, et al. 2012. "The Hiphop Compiler for Php." In *ACM Sigplan Notices*, 47:575–86. 10. ACM.

[2]See https://www.rplumber.io/docs/hosting.html
[3]See https://github.com/ATFutures/activeTransportToolbox
[4]See https://github.com/ATFutures/geoplumber