rnrfa: An R package to Retrieve, Filter and Visualize Data from the UK National River Flow Archive

by Claudia Vitolo, Matthew Fry and Wouter Buytaert

Abstract The UK National River Flow Archive (NRFA) stores several types of hydrological data and metadata: daily river flow and catchment rainfall time series, gauging station and catchment information. Data are served through the NRFA web services via experimental RESTful APIs. Obtaining NRFA data can be unwieldy due to complexities in handling HTTP GET requests and parsing responses in JSON and XML formats. The **rnrfa** package provides a set of functions to programmatically access, filter, and visualize NRFA data using simple R syntax. This paper describes the structure of the **rnrfa** package, including examples using the main functions *gdf()* and *cmr()* for flow and rainfall data, respectively. Visualization examples are also provided with a Shiny web application and functions provided in the package. Although this package is regionally-specific, the general framework and structure could be applied to similar databases.

Introduction

The increasing volume of environmental data available online poses non trivial challenges for efficient storage, access and share of this information (Vitolo et al., 2015). An integrated and consistent use of data is achieved by extracting data directly from web services and processing them on-the-fly. This improves the flexibility of modelling applications allowing a more seamless workflow integration, and also avoids the need to store local copies that would need to be periodically updated, therefore reducing maintenance issues in the system.

In the hydrology domain, various data providers are adopting web services and data Application Programming Interfaces (APIs) to allow users a fast and efficient access to public datasets, such as the National River Flow Archive (NRFA) hosted by the Centre for Ecology and Hydrology in the United Kingdom. The NRFA is a primary source of information for hydrologists, modellers, researchers and practitioners operating on UK catchments. It stores several types of hydrological data and metadata: gauged daily flow and catchment mean rainfall time series as well as gauging station and catchment information. Data are typically served through the NRFA web services via a web-based graphical interface (http://nrfa.ceh.ac.uk/) and, more recently, via experimental RESTful APIs. REST (Representational State Transfer) is an architectural style that uses the HyperText Transfer Protocol (HTTP) to perform operations such as accessing resources on the web via a Uniform Resource Identifier (URI). In simple terms, the location of a NRFA dataset on the web is a unique string of characters that follows a pattern. This string is assembled using the rules described in the API documentation and can be tested by typing the string in the address bar of a web browser.

This paper describes the technical implementation of the **rnrfa**, a package developed in the open source environment for statistical computing R. The **rnrfa** package takes the complexities related to web development and data transfer away from the user providing a set of functions to programmatically access, filter, and visualize NRFA data using simple R syntax. Although the NRFA APIs are still in their infancy and prone to further consolidation and refinement, the experimental implementation of the **rnrfa** package can be used to test these data services and provide useful feedbacks to the provider.

The package is in line with a Virtual Observatory approach (Beven et al., 2012) as it can be used as backend tool to link data and models in a seamless fashion. It complements R's growing functionality in environmental web technologies (CRAN Task View on Web Technologies), amongst which are **rnoaa** (Chamberlain, 2015, interface to NOAA climate data API), **waterData** (Ryberg and Vecchia, 2014, interface to the U.S. Geological Survey daily hydrologic data services) and **RNCEP** (Kemp et al., 2011, interface to NASA NCEP weather data).

This paper first presents the NRFA archive, its web services and related APIs. We then illustrate the design and implementation of the **rnrfa** package, and how it can be used in synergy with existing R packages such as **shiny** (Chang et al., 2016), **leaflet** (Cheng and Xie, 2015), **rmarkdown**(Allaire et al., 2016), **DT** (Xie, 2015a), **dplyr** (Wickham and Francois, 2015) and **parallel** to generate interactive mapping applications, dynamic reports and big data analytics experiments.

NRFA web services

The NRFA web services allow to view, filter and download data via a graphical interface. This approach has a number of limitations. Firstly, time series of daily streamflow discharge and catchment rainfall can only be downloaded one at the time. Therefore, for large scale analyses, downloading datasets for hundreds of sites becomes a rather tedious task. Secondly, metadata can only be visualised (in table format) but not be downloaded. Metadata analyses may require copying and pasting large amounts of information introducing potential errors. Due to the above limitations, the NRFA is also accessible programmatically via a set of RESTful APIs. The API documentation is not in the public domain yet, therefore it must be considered experimental and subject to changes.

Station metadata (called *catalogue* hereafter) is available in JavaScript Object Notation (JSON) format. The catalogue contains a total of 18 attributes, which are listed in Table 1. The NRFA also provides time series of Gauged Daily Flow (*gdf*, in m^3/s) and Catchment Mean Rainfall (*cmr*, in mm per month), formatted in an XML variant called WaterML2. WaterML2 is an Open Geospatial Consortium (OGC) standard used worldwide to rigorously and unambiguously describe hydrological time series. It builds upon existing standards such as Observations & Measurements (Cox et al., 2011) for the metadata section and GML (GML, 2013) for the observed time series. It is typically defined as a "Collection" and made of five sections:

- The *metadata* section contains the document metadata (e.g the generation date, the version, the generation system and the list of profiles utilised).
- The *temporalExtent* contains a description of the time period for which there are recordings (with a time stamp of start and end date).
- The *localDictionary* is a gml-based dictionary which stores the identifier (e.g. United Kingdom National River Flow Archive) and two dictionary entries: the first one describes the type of measurement (e.g. Gauged Daily Flow) with details on the variable measured (e.g. flow), units (e.g. m3/s) and frequency of measurements (e.g. daily); the second entry describes the gauging site with details on ratings and their limitations.
- The *samplingFeatureMember* describes the monitoring point (e.g. vertical datum and time zone) and the station owner.
- The *observationMember* contains a set of nodes which schema is borrowed from the OGC Observation and Measurement standard. This section contains a gml-based identifier (station identification number) and additional information, such as *ObservationMetadata* (contact info, identification info, etc.), *phenomenonTime* (beginning and end of recordings), *ObservationProcess* (process type and reference). Finally the sub-section *result* contains the measurements in a gml-based format.

The nested structure of the WaterML2 files makes parsing of long time-series and related metadata relatively slow and complex. In order to improve access to NRFA's public data and metadata, we implemented a set of functions to assemble HTTP GET requests and parse XML/JSON responses from/to the catalogue and WaterML2 services using a simple R syntax.

Package availability and dependencies

The **rnrfa** is a package designed to extends basic R functionalities to interact with the NRFA. It depends on the following packages that should be installed beforehand: **cowplot** (Wilke, 2016), **plyr** (Wickham, 2011), **httr** (Wickham, 2016a), **xml2** (Wickham and Hester, 2016), **stringr** (Wickham, 2016b), **xts** (Ryan and Ulrich, 2014), **rjson** (Couture-Beil, 2014), **ggmap** (Kahle and Wickham, 2013), **ggplot2** (Wickham, 2009), **rgdal** (Bivand et al., 2015), **sp** (Pebesma and Bivand, 2005; Bivand et al., 2013) and **parallel**[1]. The stable version of the package is available on the Comprehensive R Archive Network repository and can be downloaded and installed by typing the following command in the R console:

```
> install.packages("rnrfa")
```

The development version is available from a GitHub repository and can be installed via devtools, using the following commands:

```
> install.packages("devtools")
> devtools::install_github("cvitolo/rnrfa")
```

The package is loaded using the following command:

[1]In order to run the examples in this manuscript, the following packages should also be installed: **shiny**, **leaflet**, **DT**, **ggrepel** (Slowikowski, 2016), **knitr** (Xie, 2016, 2015b, 2014) and **rmarkdown**.

Column number	Column name	Description
1	*id*	Station identification number
2	*name*	Name of the station
3	*location*	Area in which the station is located
4	*river*	Name of the river catchment
5	*stationDescription*	General station description, containing information on weirs, ratings, etc.
6	*catchmentDescription*	Information on topography, geology, land cover, etc.
7	*hydrometricArea*	UK hydrometric area identification number, the related map is based on the Surface Water Survey designed in the 1930s and available at http://www.ceh.ac.uk/data/nrfa/hydrometry/has.html),
8	*operator*	UK measuring authorities, the related map is available at http://www.ceh.ac.uk/data/nrfa/hydrometry/mas.html)
9	*haName*	Name of the hydrometric area
10	*gridReference*	The Ordnance Survey grid reference number
11	*stationType*	Type of station (e.g. flume, weir, etc.)
12	*catchmentArea*	Catchment area in Km^2
13	*gdfStart*	First year of monitoring
14	*gdfEnd*	Last year of monitoring
15	*farText*	Information on the regime (e.g. natural, regulated, etc.)
16	*categories*	Various tags (e.g. FEH_POOLING, FEH_QMED, HIFLOWS_INCLUDED)
17	*altitude*	Altitude measured in metres above Ordnance Datum or, in Northern Ireland, Malin Head.
18	*sensitivity*	Sensitivity index calculated as the percentage change in flow associated with a 10 mm increase in stage at the Q_{95} flow.

Table 1: Gauging station metadata, more detail here: http://www.ceh.ac.uk/data/nrfa/data/gauging_stations.html

```
> library(rnrfa)
```

The package is fully documented and additional sample applications are available on the dedicated web page. Feedbacks and contributions can be submitted through GitHub issue tracking system and pull requests respectively.

Design and implementation

In many hydrological analyses the importance of efficient data retrieval is often underestimated with the consequence of allocating more time to this first task then to the data processing and analysis of results. The **rnrfa** packages provides re-usable functions, based on a consistent syntax, that attempt to simplify data retrieval and makes it scalable to multiple data requests.

Catalogue metadata

The full list of gauging stations is in JSON format and can be retrieved using the function `catalogue()`, used with no inputs.

```
> allStations <- catalogue()
```

This converts the information into a data frame with one row per station and 18 columns (Table 1 contains a detailed description of the attributes). The reader should note that the server response includes the Ordnance Survey (OS) grid reference, not latitude and longitude coordinates. The `catalogue()` function converts the grid reference to latitude and longitude, then joins the coordinates to the data frame containing the list of stations.

The conversion is handled by the `osg_parse()` function which can transform OS grid references of different lengths to: a) latitude and longitude, in the WSGS84[2] coordinate system; b) easting and northing, in the BNG[3] coordinate system. This function accepts two arguments: `gridRef`, a character string containing the OS grid reference and `CoordSystem` that can be either "WGS84" (default) or "BNG". The code below shows how to convert an example OS grid reference, "NC581062", to the two types of coordinates.

```
# Option a: from OS grid reference to WGS84
> osg_parse(gridRef = "NC581062", CoordSystem = "WGS84")

# Option b: from OS grid reference to BNG
> osg_parse(gridRef = "NC581062", CoordSystem = "BNG")
```

Filtering stations

The `catalogue()` function provides 5 optional arguments that can be used to filter metadata based on various criteria. The argument `all`, for instance, is TRUE by default and forces all the metadata to be retrieved. If `all` is set to FALSE, the resulting data frame contains only the following columns: `id`, `name`, `river`, `catchmentArea`, `lat`, `lon`. This can be used, for instance, to print a concise version of the table to the screen.

At the time of writing, 1539 stations are monitored within NRFA. Very rarely the full set of stations is used. Depending on to the aim of the analysis, stations might need to be filtered based on a geographical bounding box, length of the recording period, thresholds, etc. Below are some examples showing how to filter stations based on one or multiple criteria.

Filtering based on a geographical bounding box
Stations can be filtered based on a bounding box thanks to the NRFA web-service and a specific functionality of its API. A bounding box should be defined as a list of four named elements (minimum longitude, minimum latitude, maximum longitude and maximum latitude) and passed as input to the `catalogue()` function using the argument `bbox`. The following example shows how to define a bounding box for the Plynlimon area (mid-Wales, United Kingdom), filter the related stations and map their location using the **ggmap** package. In Figure 1 the location of each station is shown as a red dot, while the name of the station is used as a label.

```
# Define a bounding box:
> bbox <- list(lonMin = -3.76, latMin = 52.43, lonMax = -3.67, latMax = 52.48)
```

[2] World Geodetic System 1984, EPSG code: 4326
[3] British/Irish National Grid, EPSG codes: 27700/29902

```
# Filter stations based on bounding box
> someStations <- catalogue(bbox)

# Map
> library(ggmap)
> library(ggrepel)
> m <- get_map(location = as.numeric(bbox), maptype = 'terrain')
> ggmap(m) + geom_point(data = someStations, aes(x = lon, y = lat),
                        col = "red", size = 3, alpha = 0.5) +
>            geom_text_repel(data = someStations, aes(x = lon, y = lat, label = name),
                             size = 3, col = "red")
```

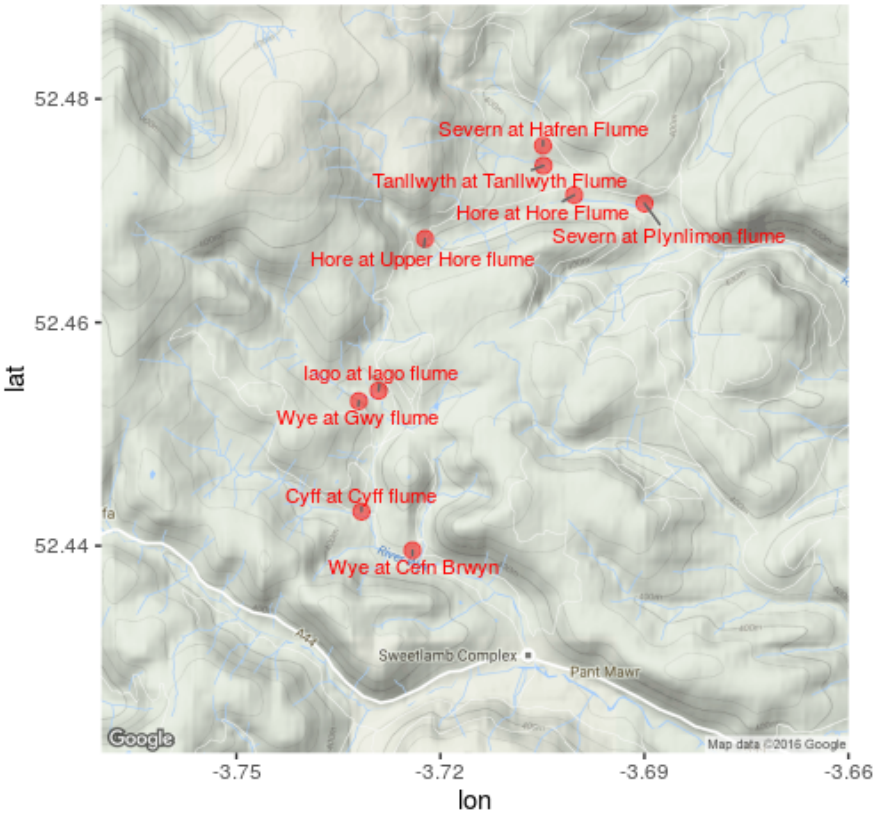

Figure 1: Map of Plynlimon area with NRFA selected gauging stations (red dots).

Filtering based on recording period
To calculate summary statistics, it is often useful to select only stations with at least x number of recording years. In the example below, we select only gauging stations with a minimum of 100 years of recordings, using the argument `minRec`. The result is a list of three stations, two of which are located in South England and one in Wales.

```
# Select stations with more than 100 years of recordings
> s100Y <- catalogue(minRec = 100, all = FALSE)

# Print s100Y to the screen
>         s100Y

     id     name                river   catchmentArea  lat      lon
636  38001  Lee at Feildes Weir Lee     1036           51.76334  0.01277874
665  39001  Thames at Kingston  Thames  9948           51.41501 -0.30887638
1130 55032  Elan at Caban Dam   Elan    184            52.26907 -3.57239164
```

Filtering based on metadata entries
It is also possible to filter stations based on a number of metadata entries using the arguments: `columnName` (name of the column to filter) and `columnValue` (string or numeric value to match or

compare). The function `catalogue()` looks for records containing the string `columnValue` in the column `columnName`. If `columnName` refers to a character field, the search is case sensitive and can be used to filter the stations based on the river name, catchment name, location and so on. In the example below we filter 34 stations falling within the Wye (Hereford) hydrometric area:

```
> stationsWye <- catalogue(columnName = "haName", columnValue = "Wye (Hereford)")
```

If `columnName` refers to a numeric field and columnValue contains special characters such as $>$, $<$, $\geq$ and $\leq$ followed by a number, stations are filtered using a threshold. For instance, there are 7 stations with drainage area smaller than 1 Km^2, which can be filtered using the command below:

```
> stations1KM <- catalogue(columnName = "catchmentArea", columnValue = "<1")
```

Combined filtering

Filtering capabilities can also be combined. In the example below we filter all the stations within the above defined bounding box that belong to the Wye (Hereford) hydrometric area and have a minimum of 50 years of recordings. The only station that satisfies all the criteria is the Wye at Cefn Brwyn.

```
> catalogue(bbox, columnName = "haName", columnValue = "Wye (Hereford)",
            minRec = 50, all = FALSE)

  id    name               river  catchmentArea  lat       lon
6 55008 Wye at Cefn Brwyn  Wye    10.6           52.43958  -3.724108
```

WaterML2 services

Once a certain number of stations are selected, time series of gauged daily flow and catchment mean rainfall data can be obtained by requesting access to the NRFA WaterML2 service using the functions `gdf()` and `cmr()`, respectively. These functions assemble and send data requests to the WaterML2 service, parse responses and convert them to a time series object (of class **xts**). They use the same syntax and require the following arguments:

- `id`, the station identification numbers. This can either be a single string or a character vector.
- `metadata`, a logical variable. If set to FALSE (default), metadata are not parsed. If it is set to TRUE, the result for a single station is a list of two named elements: `data` (time series) and `meta` (metadata).

When `gdf()` and `cmr()` are executed, the assembled data request is printed to the screen. This is very useful if the user wants to understand how the API works behind the scenes, but not when incorporating the code in automated scripts. Although the NRFA API documentation is not public yet, the patterns are simple and can be easily extrapolated running a few examples.

Get gauged daily flow

Raw flow data are typically measured in m^3/s, at 15-minute intervals. Data are first quality controlled, then the daily mean is calculated and stored in the NRFA public database. These data are typically collected for the monitoring of river networks but can also be used to calibrate hydrological models and build forecasting systems. The example below shows how to get the daily flow for the Tanllwyth at Tanllwyth Flume and the assembled data request (printed to the console).

```
> flow <- gdf(id = "54090")

http://nrfaapps.ceh.ac.uk/nrfa/xml/waterml2?db=nrfa_public&stn=54090&dt=gdf
```

The result is a time series (of class "xts"). No station-specific information is stored, because the argument `metadata` is set to FALSE by default. An xts object can be easily converted into a data frame object and exported to a text file (e.g. csv) for use in other modelling software, as demonstrated in the example below.

```
# Get gauged daily flow for station 54090
> flow <- gdf(id = "54090")

# Convert to csv
> write.csv(as.data.frame(flow), "flowDF.csv", quote = FALSE)
```

Get catchment mean rainfall

The main forcing input in any hydrological model is rainfall. In many cases it is important to calculate the average rainfall over a catchment, this is achieved by using geospatial interpolation methods or, more simplistically, calculating the weighted average using a number of weather stations within the catchment and/or in the nearby areas. The NRFA provides pre-calculated monthly catchment mean rainfall, measured in mm, for a number of UK catchments. As the calculation is consistent across catchments, these datasets are a valuable resource to ensure reproducibility of hydrological analyses. Similar to `gdf()`, the function `cmr()` allows users to retrieve the catchment mean rainfall data by specifying the argument `id`. The example below shows that, if we set the argument `metadata` to TRUE, we can use metadata to automatically populate title and labels in a plot, as in Figure 2. The reader should note that `rain$data` is an `xts` object, therefore `plot(rain$data)` uses the `plot.xts` method.

```
> rain <- cmr(id = "54090", metadata = TRUE)
> data <- rain$data
> meta <- rain$meta
> plot(data, main = paste(meta$variable, "-",
       meta$stationName), xlab = "", ylab = meta$units)
```

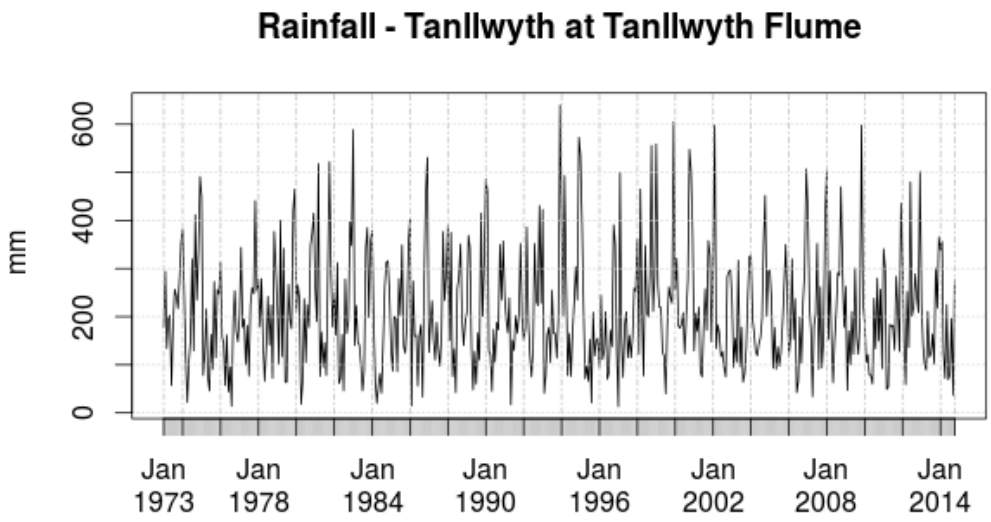

Figure 2: Monthly catchment mean rainfall for the Tanllwyth at Tanllwyth Flume catchment.

Station information consist of: the station name, location in latitude and longitude coordinates, the variable measured (i.e. rainfall), units (i.e. mm), aggregation function (i.e. accumulation), time step of recording (i.e. month) and time zone.

Convert and compare flow and rainfall for a given catchment

In the NRFA, flow and rainfall are stored in m^3/s and $mm/month$, respectively, therefore they are not directly comparable. However, given the catchment area (from the metadata catalogue), the flow can be easily converted into mm/day and then compared to the rainfall, for instance by plotting them on the same time line. Although the operations are trivial, it is a relatively lengthy procedure that can be simplified using the function `plot_rain_flow()`. This function uses the station `id` as input to request metadata as well as flow and rainfall time series for the given catchment, converts the flow from its original units to mm/day and then plots the converted flow and rainfall on two different y-axes so that they can be visually compared, as shown in Figure 3.

```
> plot_rain_flow(id = "54090")
```

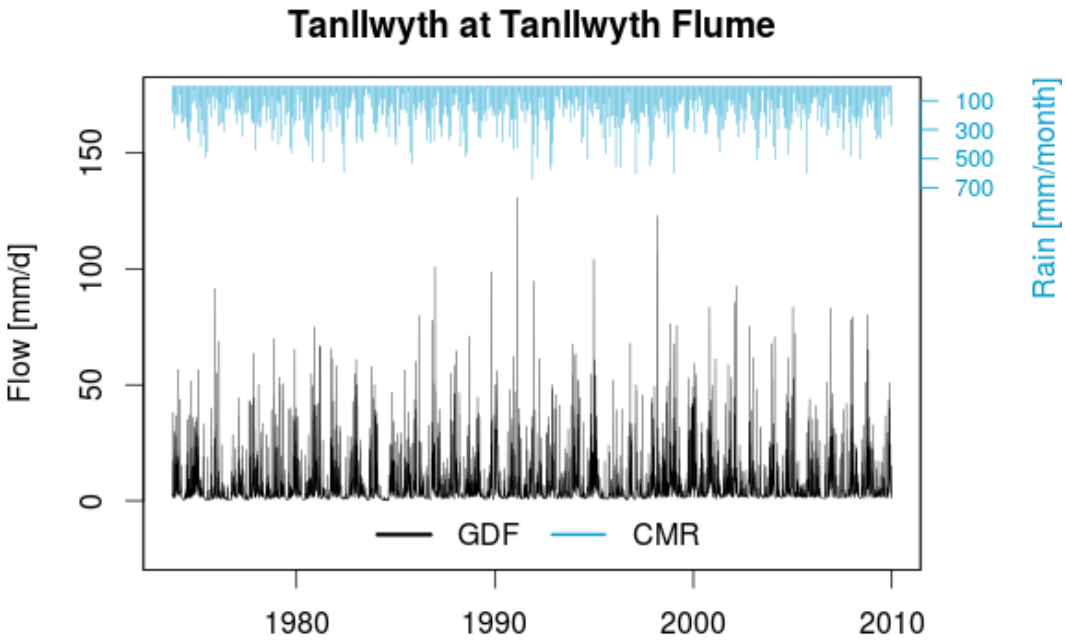

Figure 3: Monthly catchment mean rainfall and daily flow for the Tanllwyth at Tanllwyth Flume catchment.

Multiple sites

The package **rnrfa** is particularly useful for large scale data acquisition. If the `id` argument is a vector, the functions `gdf()` and `cmr()` can be used to sequentially fetch time series (meta)data from multiple sites. As the server can handle multiple requests, concurrent calls can be sent simultaneously using the **parallel** package. In order to send concurrent calls, a cluster object, created by the **parallel** package, should be passed to `gdf()` or `cmr()` using the argument `cl`. Below is a simple benchmark test in which we compare the processing time for collating flow time series data for the 9 stations in the Plynlimon area sending: a) 1 data request at the time and b) 9 simultaneous requests. The operations are repeated 10 times. The results are averaged and summarised in Table 2, which shows that (a) takes about 18 seconds, while (b) about 8 seconds. The reader should note that the time for retrieval does not reduce proportionally with the number of simultaneous requests because there is a limit in the number of calls the server can handle, which depends on the infrastructure and the number of incoming requests from other users.

```
> library(microbenchmark)
> library(parallel)

> cl <- makeCluster(getOption("cl.cores", 9))

> microbenchmark(
                 # sequential requests
                 gdf(id = someStations$id, metadata = FALSE, cl = NULL),

                 # concurrent requests
                 gdf(id = someStations$id, metadata = FALSE, cl = cl),

                 times = 10
  )

> stopCluster(cl)
```

Test	min	lq	mean	median	uq	max	neval
a	17.598647	17.95601	18.419300	18.355630	19.037328	19.16267	10
b	3.564888	8.91512	8.411546	9.504491	9.666812	10.58291	10

Table 2: Benchmark tests comparing retrieval time for sequential (a) and simultaneous calls to the server (b). Results show time in seconds, obtained by averaging over 10 repetitions using the **microbenchmark** package (Mersmann, 2015).

Some applications

The **rnrfa** package is an ideal building block for many scientific workflows but can also work as backend tool for a number of web applications, from interactive mapping and dynamic reports that improve reproducibility of analysis, to the integration into more sophisticated big data analytics experiments. This can be achieved thanks to the intrinsic interoperability of the R environment. Some example applications are given in the following sections.

Dynamic mapping and reporting application

Here we demonstrate the generation of a dynamic mapping and reporting application to summarise stations metadata and map the spatial distribution of the monitoring network for each operator. The user can select the name of the operator using a drop-down menu and the dynamic document automatically renders an interactive map showing a marker for each station in the network on top of a background map based on OpenStreetMap. Users can zoom in/out and navigate to a specific area. Finally, the user can click on a marker to read name and station identification number from a popup window. Figure 4 shows a screenshot of the web application. At the bottom of the page is a dynamic table that summarises the metadata associated with the selected stations in the network. The table can be filtered using an interactive search box. The textual content also updates automatically reporting the number of stations within the selected network. The web application depends on the following packages: **Rmarkdown**, **knitr**, **shiny**, **leaflet** and **DT** and its source code is available as gist at the following url: `https://gist.github.com/cvitolo/d5d46b5e8f3676013857`.

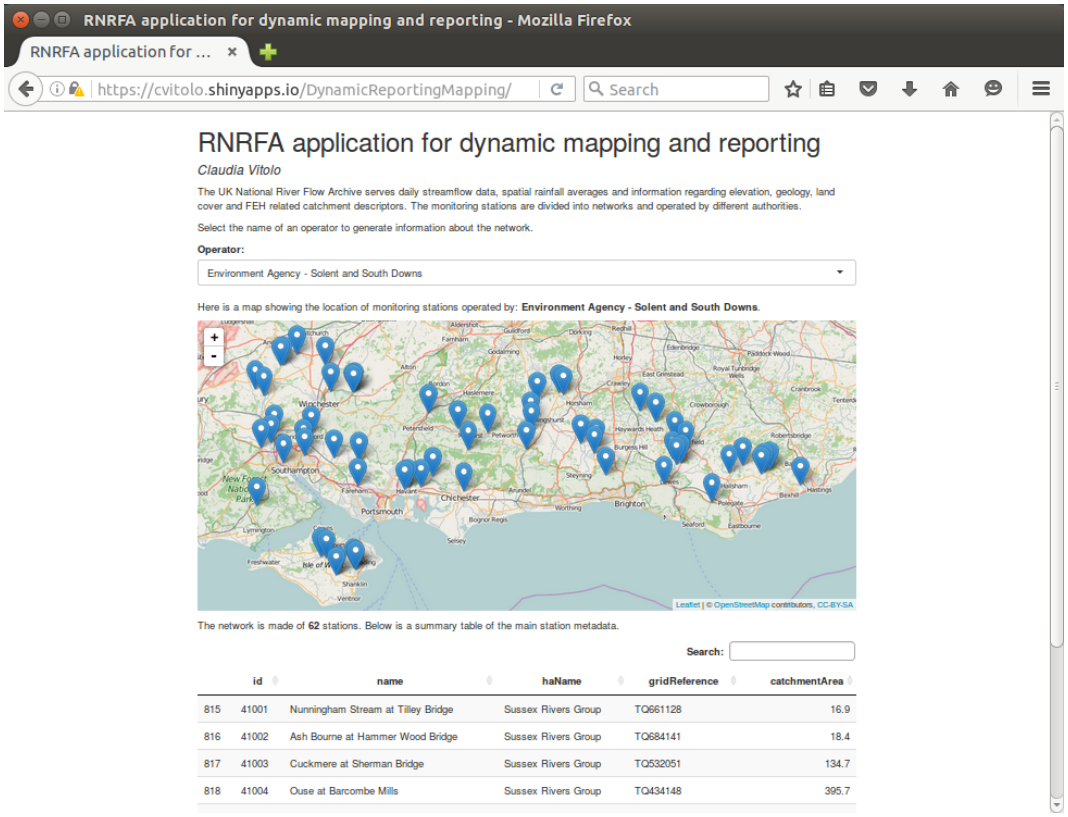

Figure 4: RNRFA application for dynamic reporting and mapping.

Geoprocessing based on user-defined areas

The NRFA web site does not allow users to execute geoprocessing tasks, for instance, to intersect the list of stations with user-defined or externally sourced areas. In some cases it might be of interest to explore the distribution of stations based on high-level administrative boundaries such as regions/countries. This is useful to understand whether there are differences in the reliability of the networks that can be explained by the different management approaches. Eurostat established a hierarchy of three levels of administrative divisions within each European country, called Nomenclature of Territorial Units for Statistics (NUTS)[4]. At the first level, UK is divided into 12 regions: Northern Ireland, Scotland, Wales and 9 English sub-regions (East Midlands, East of England, Greater London, North East, North West, South East, South West, West Midlands, Yorkshire and the Humber). Calculating, for instance, the number/density of stations by region is not possible using the NRFA website because the stations' metadata does not contain information on this type of administrative region and users cannot specify their own. However, these simple geoprocessing operations become relatively trivial using the **rnrfa** package.

The procedure consists of 4 steps:

- retrieve the list of NRFA stations (using the `catalogue()` function)
- load the NUTS (level 1) shapefile and reproject the polygon to the geographic coordinate system WGS84 (using the **rgdal** and **sp** packages)
- transform NRFA list of stations into a `SpatialPointDataFrame`
- spatially overlay NRFA stations (points) and NUTS1 regions (polygons)
- a new column, containing the name of the NUTS1 regions, is added to the list of NRFA stations.

The updated list of stations is included, as sample dataset, in the data folder of this package, under the name `stationSummary`. The table below summarises the number of stations per region, the area of each region (in Km2), and the density of stations (number of stations/Km2). The metadata can now be easily summarised by NUTS1 region, for instance the boxplot below shows the distribution of years of recording. Northern Ireland seem to have the youngest network, with recording years in the range [16, 44]. Only three regions have stations with more than 100 years of recordings: East of England, London and Wales. Scotland and Northern Ireland have the lowest density of gauging stations, while Greater London the highest. The code to reproduce this example is available as gist at the following url: https://gist.github.com/cvitolo/aa3bc6f08a8394f653442e276568f9b3.

	NUTS_ID	Region	# stations	Area (Km2)	Density
1	UKC	North East (England)	54.00	8601.77	0.006
2	UKD	North West (England)	137.00	14170.34	0.010
3	UKE	Yorkshire and the Humber	102.00	15418.70	0.007
4	UKF	East Midlands	101.00	15637.21	0.006
5	UKG	West Midlands	103.00	12999.97	0.008
6	UKH	East of England	149.00	19159.91	0.008
7	UKI	Greater London	36.00	1575.97	0.023
8	UKJ	South East (England)	169.00	19105.67	0.009
9	UKK	South West (England)	176.00	23912.24	0.007
10	UKL	Wales	132.00	20817.37	0.006
11	UKM	Scotland	324.00	78984.40	0.004
12	UKN	Northern Ireland	56.00	14175.46	0.004

[4]The Nomenclature of Territorial Units for Statistics (NUTS), is a standard for referencing the administrative divisions of European countries. There are three levels of NUTS and a shapefile is available from the Eurostat website (http://ec.europa.eu/eurostat/cache/GISCO/geodatafiles/NUTS_2013_01M_SH.zip)

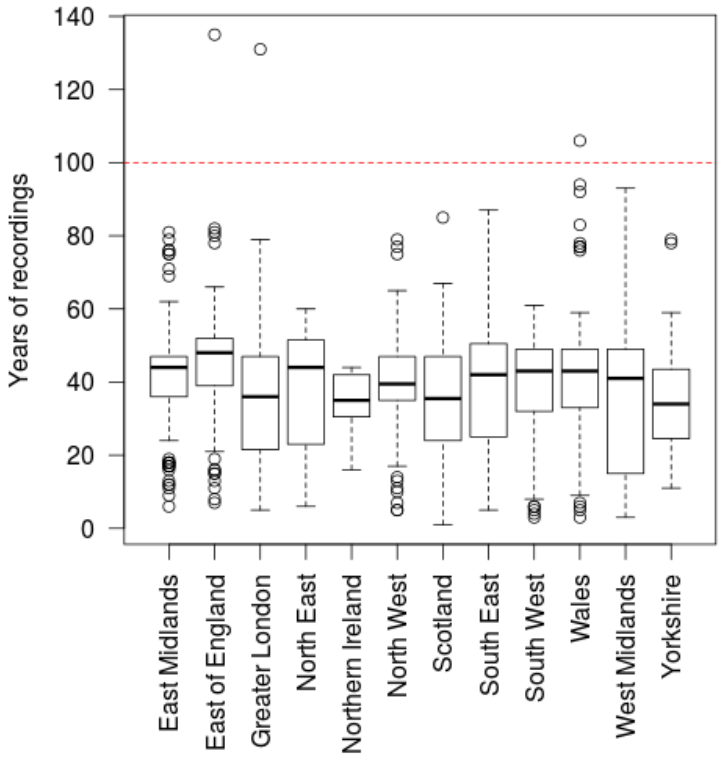

Figure 5: Distribution of recording years for NRFA stations by NUTS1 regions.

Big data analytics experiment

In the last few years, the UK MetOffice has reported "unusual warmth and lack of rainfall during March and April, particularly over England and Wales"[5]. Dry springs can affect water resources, because river flow below average translates, for instance, in reduced availability of drinking water. In this section we present a big data analytics experiment in which we try to understand if there is any evidence, in the NRFA data, that springs in the UK are becoming drier, both in terms of rainfall and river flow. This type of experiment consists of retrieving all the available rainfall and flow time series and find out, for each station, whether there is a increasing/decreasing trend.

Using NRFA website, the comparison of time series is only feasible for a limited number of sites. Time series should be first downloaded as text file and then compared manually. The biggest advantage of using the **rnrfa** package, instead, is that multiple downloads can be automated using a single line of code.

In this experiment we used a cluster of 64 cores to download and analyse all the time series available from the NRFA stations with more than 10 years of recordings. The time series were first downloaded, then summarised in terms of annual averages over the spring period. Seasonal averages can be calculated using the function `seasonal_averages()`, which takes as input a time series and a period of interest (e.g. spring) and calculates the related annual average. Using a very simplistic approach, a linear model was fit to the annual averages and the slope coefficient was used to estimate the trend. Negative slopes correspond to decreasing flow/rainfall, while positive slopes correspond to an increase of flow/rainfall over time. Once the fitted slope is calculated for each station, the results can be plotted using the function `plot_trend()`. Figures 6 and 7 show only the statistically significant trends for rainfall and flow respectively. Each figure is divided into two plots: plot A shows the spatial distribution of negative trends with red dots and positive trends with blue dots; plot B shows the variability of trends over NUTS1 regions. In the latter plot, outliers are removed by showing only values between the 5th and 95th quantiles. From a meteorological perspective (Figure 6), there are only positive statistically significant trends and Scotland shows the largest. In terms of hydrological responses (Figure 7), trends are more subtle as the interquartile range is concentrated around zero. The most extreme negative trends were found in Scotland and North East England.

The entire run took about 31 minutes, the code to reproduce this example is available as gist at the

[5] http://www.metoffice.gov.uk/climate/uk/interesting/2011_spring

following url: https://gist.github.com/cvitolo/612eb2ae9b47fe8f11a1ed8d06e3b434. There are certainly more rigorous methodologies to estimate seasonal trends. This experiment was just an attempt to demonstrate that the **rnrfa** can simplify large scale data acquisition tasks.

Figure 6: Map and boxplot of rainfall trend during spring.

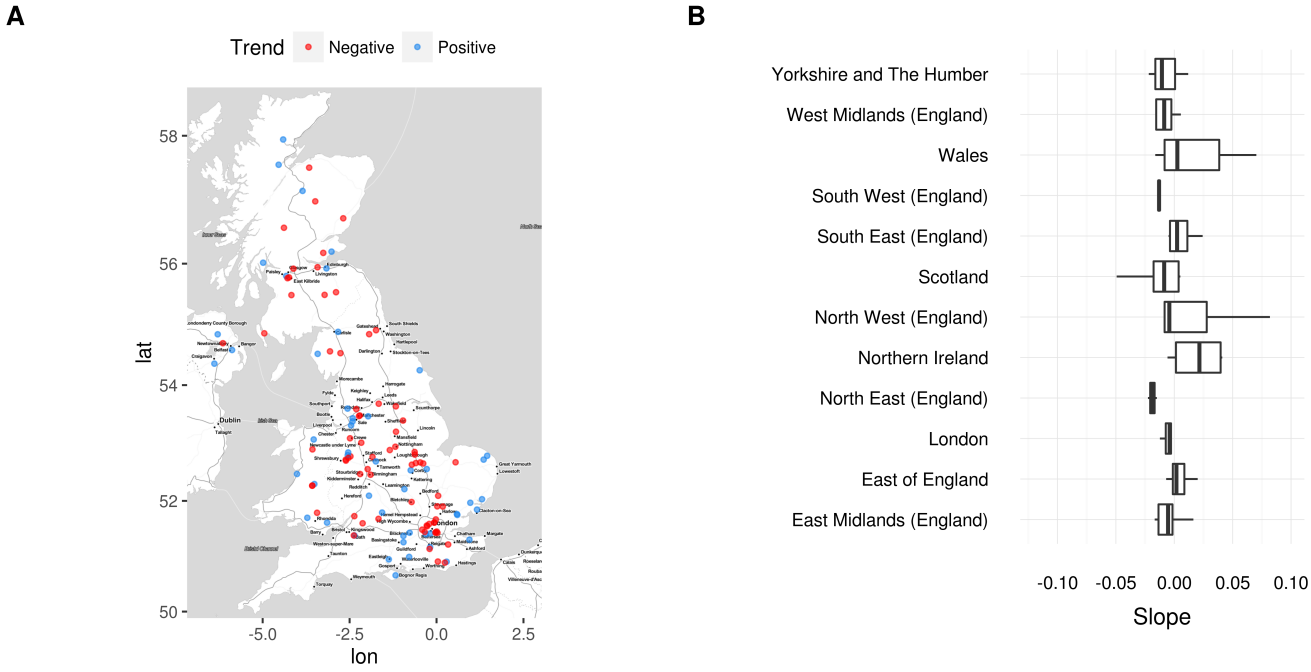

Figure 7: Map and boxplot of flow trend during spring.

A note on package usage

The **cranlogs** (Csardi, 2015) package provides an API interface to download logs from the RStudio CRAN Mirror which contain download counts from unique IP addresses and can be used as a proxy to estimate the volume of package users. By September 2016, the **rnrfa** package had been downloaded

from CRAN 6372 times, just from this mirror, following a trend very similar to the **waterData** package (see Figure 8). Being the RStudio mirror located in the US, it is expected that the download counts from UK mirrors could be even higher. We derive that this package is of interest for a large community of users, which gives us scope for future developments.

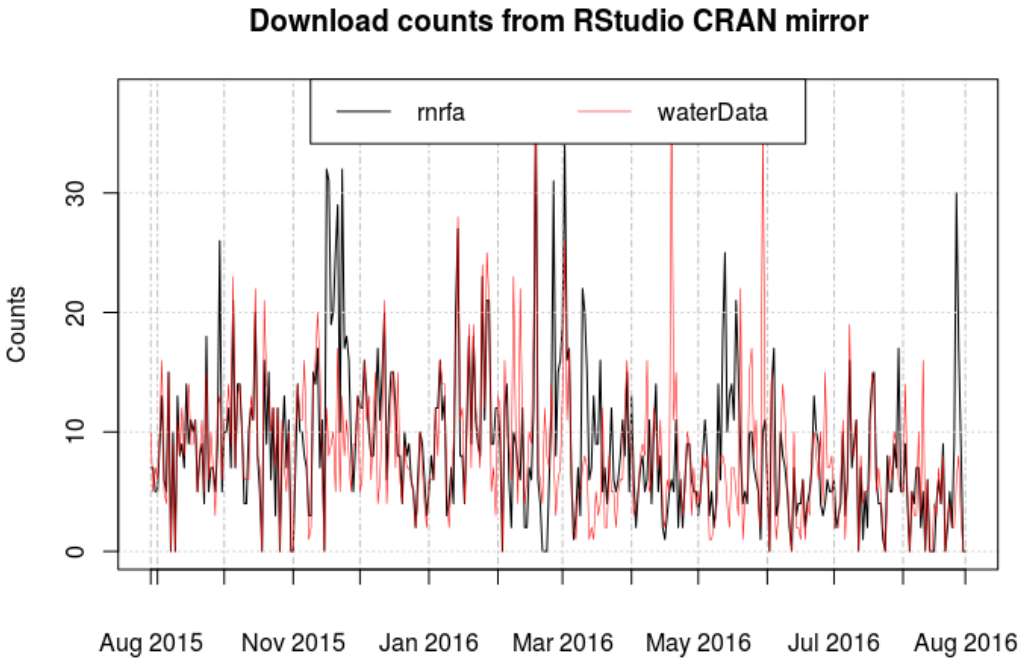

Figure 8: Comparison between **rnrfa** and **waterData** download counts from independent IP addresses.

Summary

This article describes the **rnrfa** package for interacting programmatically with the UK National River Flow Archive. It allows to access web resources such as the catalogue of stations metadata and the WaterML2 service to retrieve gauged daily flow and (monthly) catchment mean rainfall. The package provides functions to query the catalogue based on various criteria (e.g. geographical bounding box, minimum number of recording years, river catchment/hydrometric area/operators amongst many other options), retrieve and visualise flow and rainfall time series, convert coordinates and flow measurements, and plot basic seasonal trends grouped on user defined regions. Some of these capabilities are strongly linked to the particular content of the NRFA database and are not directly transferable/applicable to other data sources. However the `gdf()` and `cmr()` functions could be re-used, with minimal changes, to get data/metadata from other providers adopting the WaterML2 standard.

The package is a convenient stand alone application that allows NRFA users a more efficient access to the public database, compared to the web interface, e.g. the possibility to efficiently retrieve data from multiple sites. The **rnrfa** package can also be used as back-end tool for web applications. Amongst the existing R interfaces to data APIs, **rnrfa** follows a logic similar to **waterData**: sites are first identified through a catalogue, streamflow data are imported via the station identification number, then data are visualised and/or used in analyses. However, our package does not implement any function for data cleanup, because NRFA data are highly quality controlled. Users can currently take advantage of other packages such as **xts** to calculate aggregate variables, **evd** (Stephenson, 2002) for the analysis of extreme events, **outliers** (Komsta, 2011) to identify possible outliers and **sp** and **spacetime** (Pebesma, 2012; Bivand et al., 2013) for more advanced spatio-temporal processing.

In the future, we plan to implement additional processing functions (e.g. to compare gdf with flow in bankfull condition, highly important for flood frequency estimations). Further developments are also scheduled on the NRFA side to include Web Feature Service (WFS), Sensor Observation Services (SOS)

and updates to WaterML2 OGC standards. WFS layers can already be loaded and manipulated using **rgdal** (Bivand et al., 2015), while **sos4R** (Nüst, 2010) can be used as client for SOS.

Acknowledgments

This work was carried out when Claudia Vitolo was working at the Imperial College London and supported by the Natural Environment Research Council pilot Probability, Uncertainty & Risk in the Environment (PURE) NE/I004017/1.

Comments from two referees are gratefully acknowledged.

Bibliography

J. Allaire, J. Cheng, Y. Xie, J. McPherson, W. Chang, J. Allen, H. Wickham, A. Atkins, and R. Hyndman. *rmarkdown: Dynamic Documents for R*, 2016. URL https://CRAN.R-project.org/package=rmarkdown. R package version 0.9.2. [p1]

K. Beven, W. Buytaert, and L. Smith. On virtual observatories and modelled realities (or why discharge must be treated as a virtual variable). *HYDROLOGICAL PROCESSES*, 26:1906–1909, 2012. doi: 10.1002/hyp.9261. URL http://dx.doi.org/10.1002/hyp.9261. [p1]

R. Bivand, T. Keitt, and B. Rowlingson. *rgdal: Bindings for the Geospatial Data Abstraction Library*, 2015. URL https://CRAN.R-project.org/package=rgdal. R package version 1.1-3. [p2, 14]

R. S. Bivand, E. Pebesma, and V. Gomez-Rubio. *Applied spatial data analysis with R, Second edition*. Springer, NY, 2013. URL http://www.asdar-book.org/. [p2, 13]

S. Chamberlain. *rnoaa: 'NOAA' Weather Data from R*, 2015. URL https://CRAN.R-project.org/package=rnoaa. R package version 0.5.0. [p1]

W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2016. URL https://CRAN.R-project.org/package=shiny. R package version 0.13.2. [p1]

J. Cheng and Y. Xie. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2015. URL https://CRAN.R-project.org/package=leaflet. R package version 1.0.0. [p1]

A. Couture-Beil. *rjson: JSON for R*, 2014. URL https://CRAN.R-project.org/package=rjson. R package version 0.2.15. [p2]

S. Cox et al. Observations and measurements-xml implementation. *OGC document*, 2011. [p2]

G. Csardi. *cranlogs: Download Logs from the 'RStudio' 'CRAN' Mirror*, 2015. URL https://CRAN.R-project.org/package=cranlogs. R package version 2.1.0. [p12]

O. GML. Opengis geography markup language (gml) encoding standard. Technical report, Tech rep, open geospatial consortium. http://www. opengeospatial. org/standards/gml. Accessed May, 2013. [p2]

D. Kahle and H. Wickham. ggmap: Spatial visualization with ggplot2. *The R Journal*, 5(1):144–161, 2013. URL http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf. [p2]

M. U. Kemp, E. E. van Loon, J. Shamoun-Baranes, and W. Bouten. RNCEP: global weather and climate data at your fingertips. *Methods in Ecology and Evolution*, 3(1):65–70, jul 2011. doi: 10.1111/j.2041-210x.2011.00138.x. URL http://dx.doi.org/10.1111/j.2041-210x.2011.00138.x. [p1]

L. Komsta. *outliers: Tests for outliers*, 2011. URL https://CRAN.R-project.org/package=outliers. R package version 0.14. [p13]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-2.1. [p9]

D. Nüst. sos4R: An R client for the OGC Sensor Observation Service, 2010. URL http://www.nordholmen.net/sos4r/. [p14]

E. Pebesma. spacetime: Spatio-temporal data in R. *Journal of Statistical Software*, 51(7):1–30, 2012. URL http://www.jstatsoft.org/v51/i07/. [p13]

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL http://CRAN.R-project.org/doc/Rnews/. [p2]

J. A. Ryan and J. M. Ulrich. *xts: eXtensible Time Series*, 2014. URL https://CRAN.R-project.org/package=xts. R package version 0.9-7. [p2]

K. R. Ryberg and A. V. Vecchia. *waterData: An R Package for Retrieval, Analysis, and Anomaly Calculation of Daily Hydrologic Time Series Data*, 2014. URL https://CRAN.R-project.org/package=waterData. R package version 1.0.4. [p1]

K. Slowikowski. *ggrepel: Repulsive Text and Label Geoms for 'ggplot2'*, 2016. URL https://CRAN.R-project.org/package=ggrepel. R package version 0.5. [p2]

A. G. Stephenson. evd: Extreme value distributions. *R News*, 2(2):0, June 2002. URL http://CRAN.R-project.org/doc/Rnews/. [p13]

C. Vitolo, Y. Elkhatib, D. Reusser, C. J. Macleod, and W. Buytaert. Web technologies for environmental Big Data. *Environmental Modelling & Software*, 63:185–198, jan 2015. doi: 10.1016/j.envsoft.2014.10.007. URL http://dx.doi.org/10.1016/j.envsoft.2014.10.007. [p1]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p2]

H. Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1): 1–29, 2011. URL http://www.jstatsoft.org/v40/i01/. [p2]

H. Wickham. *httr: Tools for Working with URLs and HTTP*, 2016a. URL https://CRAN.R-project.org/package=httr. R package version 1.1.0. [p2]

H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2016b. URL https://CRAN.R-project.org/package=stringr. R package version 1.1.0. [p2]

H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2015. URL https://CRAN.R-project.org/package=dplyr. R package version 0.4.3. [p1]

H. Wickham and J. Hester. *xml2: Parse XML*, 2016. URL https://CRAN.R-project.org/package=xml2. R package version 1.0.0. [p2]

C. O. Wilke. *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'*, 2016. URL https://CRAN.R-project.org/package=cowplot. R package version 0.6.2. [p2]

Y. Xie. *Implementing Reproducible Computational Research*, chapter knitr: A Comprehensive Tool for Reproducible Research in R. Chapman and Hall/CRC, 2014. ISBN 978-1466561595. [p2]

Y. Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2015a. URL https://CRAN.R-project.org/package=DT. R package version 0.1. [p1]

Y. Xie. *Dynamic Documents with R and knitr. 2nd edition*. Chapman and Hall/CRC, 2015b. ISBN 978-1498716963. [p2]

Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2016. URL https://CRAN.R-project.org/package=knitr. R package version 1.14. [p2]

Claudia Vitolo
Forecast Department
European Centre for Medium-range Weather Forecast
Reading
United Kingdom
claudia.vitolo@gmail.com

Matthew Fry
Centre for Ecology and Hydrology
Wallingford
United Kingdom
mfry@ceh.ac.uk

Wouter Buytaert
Department of Civil and Environmental Engineering
Imperial College London
United Kingdom
w.buytaert@imperial.ac.uk