UK river flow data

Luke Shaw

2020-04-11

In this post I try out the recently published [winfapReader](ilapros.github.io/winfapReader/) R package for getting UK river flow data, and think about what I have learnt since my involvement in the project 3 years ago.

# Intro

Recently, out of the blue, I received a notification for this tweet:

Four years after the first script my first attempt to a public R package is ready to be tested by others: https://t.co/kPR2cSDysQ is an interface with the @UK\_NRFA extreme data locally or via their API. With thanks for the support/code to @clavitolo @mattfry\_ceh @lukefshaw

— Ilaria Prosdocimi (@ilapros) April 8, 2020

Dr Ilaria Prosdocimi was my final year project (i.e. dissertation) supervisor at the University of Bath, and I last worked on river flow data about 2 and a half years ago. The package update was a very pleasant surprise, and I am very grateful for being included as a contributor, as well as happy that code I wrote has been of use to somebody else!

# Project reflections

The full pdf of my final report is available [here](https://lukefshaw.netlify.com/20170508_final_year_project.pdf).

Here are some quick thoughts after re-reading that document for the first time since submission!

## Dates dates dates

I discovered then, and still love now, the “lubridate” package for *anything* involving dates.

## Data cleaning

The following quote rings very true in every data project I have worked on since:

I spent at least half of this project cleaning the data, involving removing years with incomplete data and checking that the data was consistent.

For this project, the data cleaning/validation work even worked its way into the title - “Assessing the reliability of high flows records”!

## More dates

* I’m reminded of the following lyric from the musical RENT:

How do you measure, measure a year

In my current job, the year can start in 3 places:

1. Academic year - Sep - Aug
2. Financial year - Apr - Mar
3. Calendar year - Jan - Dec

Re-reading my project made me laugh, as I had forgotten about the ‘Water year’, which is Oct - Sep. In fact things get even more complicated as the years don’t all change at midnight on the 01st of the month, like for the calendar year. As mentioned above; thank heavens for lubridate!

## Uncertainty over ‘old’ code

This is the first time code I’ve written is well and truly public, other than a package which is so specific as to only be of use to one person in the world - whoever is in the job role I was in when I made it. As such, it is somewhat daunting to think code I have written may be used by someone else. Looking back now at the code written then I think how I would do things differently, but realistically I’m not going to go back and re-write things, as that task will always fall towards the bottom of the pile and the current code works, after all. It would be too tempting to never put code online for fear it could be better, as it always could be better.

# Trying winfapReader

Here is a testcase of winfapReader looking at water gauging stations around Bristol, where I am currently based. I was inspired by [this blog](https://www.musclesofquartz.com/post/quick-review-of-winfapreader/) by Sean Turner.

## Set-up and location

library(winfapReader) # new package to be trialled  
library(rnrfa) # NRFA official package  
library(tidyverse) # for data wrangling  
library(knitr) # for table output formatting

The rnrfa package is on CRAN and usefule for finding which guaging stations exist. The NRFA also has a [search tool](https://nrfa.ceh.ac.uk/data/search) where you can manually find stations.

We’ll take the lat-long for the middle of Bristol, and define a bounding box for an area nearby

bbw <- 0.4 #bounding box width  
brist\_latlong <- c(51.4545, -2.5879)  
#get Bristol station data  
riv\_brist <- rnrfa::catalogue(bbox = list(lat\_min = brist\_latlong[1]-bbw/2,   
 lat\_max = brist\_latlong[1]+bbw/2,   
 lon\_min = brist\_latlong[2]-bbw/2,   
 lon\_max = brist\_latlong[2]+bbw/2))  
  
#only keep stations with highest quality data. An important step, and as my   
#final year project can attest there's lots to think about with data quality!  
riv\_brist <- riv\_brist %>% filter(`feh-pooling` == TRUE)  
  
knitr::kable(riv\_brist %>% select(id, name))

|  |  |
| --- | --- |
| id | name |
| 53004 | Chew at Compton Dando |
| 53006 | Frome (Bristol) at Frenchay |
| 53017 | Boyd at Bitton |
| 53026 | Frome (Bristol) at Frampton Cotterell |

The rnrfa::catalogue function also returns a wealth of data which I won’t look at here, but just to say there are 101 columns-worth of info per site!

5 stations are in the area, and one (id 52015 which is on the left of the image below) is deemed of not good enough quality to use here . So we’re left with 4 stations, shown in the image. The image is a screen grab from the NRFA [search tool](https://nrfa.ceh.ac.uk/data/search).

## Getting POT data

The winfapReader package gives access to 2 sets of data:

1. Annual Maxima
2. Peaks Over the Threshold (POT)

My project was around the second, so that’s the data I’ll download and look at here! In a sentence:

POT data are the values recorded anytime the river flow exceeds the threshold, set by the NRFA to give an average of between 3 and 5 peaks a year.

Of course there is more information on the NRFA website, and in the project text, if interested further. It’s interesting data.

With the winfapReader package, you can connect to NRFA’s API and download the data you want from only knowing the station ID. It outputs the data in a list of three, so I’ve made this little wrapper function to pull into a tibble:

reshape\_pot\_list\_to\_tbl\_df <- function(id){  
 ## takes the station id, gets POT data, and reshapes into tbl\_df  
   
 # use winfapReader package to get list  
 pot\_list <- winfapReader::get\_pot(id)  
   
 # join two table outputs  
 pot\_df <- pot\_list$tablePOT %>%   
 dplyr::left\_join(pot\_list$WaterYearInfo,by = 'WaterYear') %>%  
 tibble::as\_tibble()  
   
 # add in start and end dates but as NA vals for all other cols  
 # maybe not worth it? Only included so all get\_pot info kept  
 start\_end\_df <- tibble::tibble(Station = id,   
 Date = pot\_list$dateRange)  
 pot\_df <- pot\_df %>%   
 dplyr::bind\_rows(start\_end\_df) %>%  
 dplyr::arrange(Date)  
   
 return(pot\_df)  
}

Now we can download POT data in one line (though of course number of lines of code is not a great metric for code quality!):

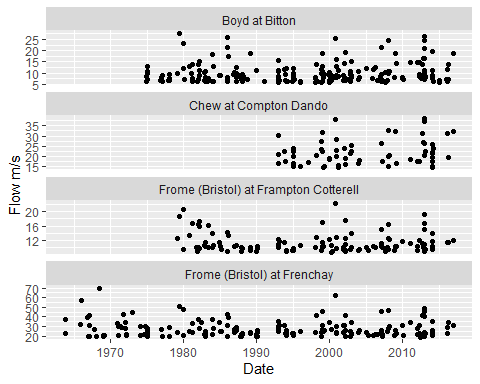
# all the heavy lifting happens in this line  
brist\_pot\_df <- purrr::map\_dfr(riv\_brist$id, reshape\_pot\_list\_to\_tbl\_df)   
# add the name of the station for plotting later  
brist\_pot\_df <- left\_join(brist\_pot\_df,   
 riv\_brist %>% select(id, name),  
 by = c("Station"="id"))  
  
knitr::kable(head(brist\_pot\_df))

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Station | Date | WaterYear | Flow | Stage | potPercComplete | potThreshold | name |
| 53004 | 1992-11-29 | NA | NA | NA | NA | NA | Chew at Compton Dando |
| 53004 | 1992-11-30 | 1992 | 30.1 | 3.874 | 83.56164 | 14.72 | Chew at Compton Dando |
| 53004 | 1992-12-02 | 1992 | 16.8 | 2.814 | 83.56164 | 14.72 | Chew at Compton Dando |
| 53004 | 1992-12-18 | 1992 | 21.0 | 3.262 | 83.56164 | 14.72 | Chew at Compton Dando |
| 53004 | 1993-10-13 | 1993 | 22.3 | 3.391 | 100.00000 | 14.72 | Chew at Compton Dando |
| 53004 | 1993-12-19 | 1993 | 17.4 | 2.888 | 100.00000 | 14.72 | Chew at Compton Dando |

## First plot

Here’s an initial plot of the data we’ve got;

#only take records above 75% completeness (also removes start/end dates)  
plot\_df <- brist\_pot\_df %>% filter(potPercComplete > 75)  
  
#plots away!  
ggplot(plot\_df, aes(x=Date, y = Flow)) +  
 geom\_point() +  
 facet\_wrap(~name, scales = "free\_y", ncol = 1) +  
 labs(y = "Flow m/s")



A few things spring to mind when looking at this plot:

1. The time period is different for different stations. Frome at Frenchay goes all the way back to the 1960s, whereas Chew at Compton Dando doesn’t start till the 1990s.
2. What happened around 2013? Seem to be lots of peaks?
3. The flow is different for different locations. The Frome at Frenchay has much more than at Frampton Cotterell, and a quick online search shows the Bradley Brook joins the River Frome between the two stations, so an increased flow makes sense. Always good to sense check the data.

So the initial plotting has given rise to a few interesting questions to be explored further.

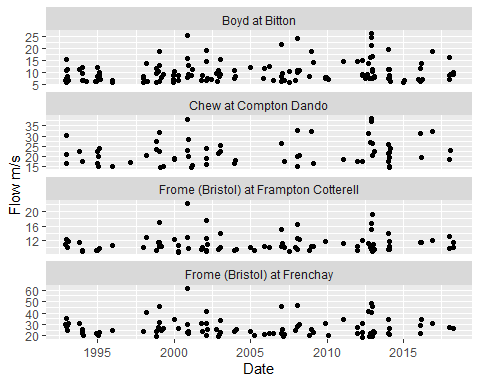
## Consistent time period

To compare data across stations, I’ll only keep data in which all 4 had data.

valid\_dates\_df <- brist\_pot\_df %>%   
 # the NA potThreshold columns is a proxy for start and end date as   
 # reshape\_pot\_list\_to\_tbl\_df joined the list outputs from winfapReader::get\_pot()  
 filter(is.na(potThreshold)) %>%   
 select(Station, Date) %>%   
 group\_by(Station) %>%   
 summarise(start = min(Date), end = max(Date)) %>%  
 ungroup() %>% pivot\_longer(cols = c("start","end")) %>%  
 # use the in-built Water Year finder function  
 mutate(wy = winfapReader::water\_year(value))  
  
#start and end dates for where all stations have data  
start\_wy = max(valid\_dates\_df$wy[valid\_dates\_df$name == "start"])  
end\_wy = min(valid\_dates\_df$wy[valid\_dates\_df$name == "end"])  
  
#make a cut version of the data set  
brist\_pot\_filt\_df <- brist\_pot\_df %>%   
 filter(WaterYear >= start\_wy, WaterYear <= end\_wy)

Re-do our plot to check:

ggplot(brist\_pot\_filt\_df %>% filter(!is.na(Date)), aes(x=Date, y = Flow)) +  
 geom\_point() +  
 facet\_wrap(~name, scales = "free\_y", ncol = 1) +  
 labs(y = "Flow m/s")



Looks good!

## When were the most POTs?

If we wanted to know when the highest flow occured each year, the POT data should not be the first port of call. The Annual Maximum data is for that purpose, and as measurements and calculations can differ unfortunately it’s not always as simple as taking the max POT value each year. Again, this was a large part of my project!

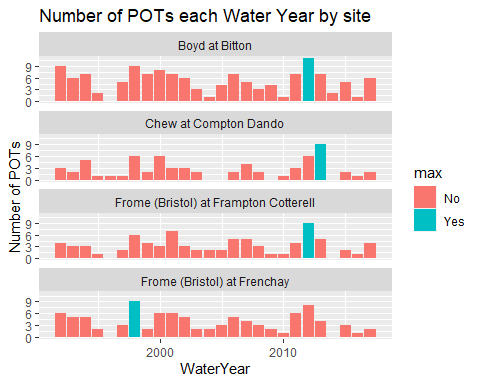
Instead, POT data has power in the number of peaks over the threshold. Lets see which water years had the most peaks by site. Will it be the same year for all?

#number of POTs each year  
pot\_each\_year <- brist\_pot\_df %>%  
 group\_by(Station, WaterYear, name) %>%  
 summarise(num\_pots = n()) %>%  
 ungroup()  
  
#max pots for each site where they all had data  
max\_pots <- pot\_each\_year %>%  
 filter(WaterYear >= start\_wy, WaterYear <= end\_wy) %>%  
 group\_by(Station) %>%  
 filter(num\_pots == max(num\_pots)) %>%  
 ungroup()  
  
knitr::kable(max\_pots)

|  |  |  |  |
| --- | --- | --- | --- |
| Station | WaterYear | name | num\_pots |
| 53004 | 2013 | Chew at Compton Dando | 9 |
| 53006 | 1998 | Frome (Bristol) at Frenchay | 9 |
| 53017 | 2012 | Boyd at Bitton | 11 |
| 53026 | 2012 | Frome (Bristol) at Frampton Cotterell | 9 |

Oh, so not all in the same year of 2013. We can do better at trying to see these results.

# wrangle the data before plotting  
pot\_each\_year\_filt\_max\_df <- pot\_each\_year %>%   
 filter(WaterYear >= start\_wy, WaterYear <= end\_wy) %>%  
 # left join onto max\_pots and create a column max which says if it was   
 # a max number of POT that year  
 left\_join(max\_pots %>% mutate(max='Yes') %>% select(Station, WaterYear, max),   
 by = c('Station','WaterYear')) %>%  
 mutate(max = ifelse(is.na(max),'No',max))  
  
ggplot(pot\_each\_year\_filt\_max\_df, aes(x = WaterYear, y = num\_pots, fill=max)) +  
 geom\_bar(stat='identity') +  
 facet\_wrap(~name, ncol = 1) +  
 labs(y = "Number of POTs", title = "Number of POTs each Water Year by site")



An interesting visualisation! Maybe not quite what we were expecting, but again it invites further questions about the data. Clearly only 2 sites, Boyd at Bitton and Frome at Frenchay, had most POTs in the same year - the 2012 Water Year. There was also lots of POTs in 1998, the most for Frome at Frenchay and second or third most for the other three sites.

This data is of course very rich - there’s more we could look at. Individual dates, times of year, following water flow (as opposed to picking an area), and much else too! That’s enough for me for today, though.

# Conclusion

The winfapReader package is very cool, and powerful when combined with the rnrfa one. I only looked at a specific area, and it was quick to get the data I wanted.

Thank you Dr Ilaria Proscdocimi. I am flattered to be included as a contributor to this work.