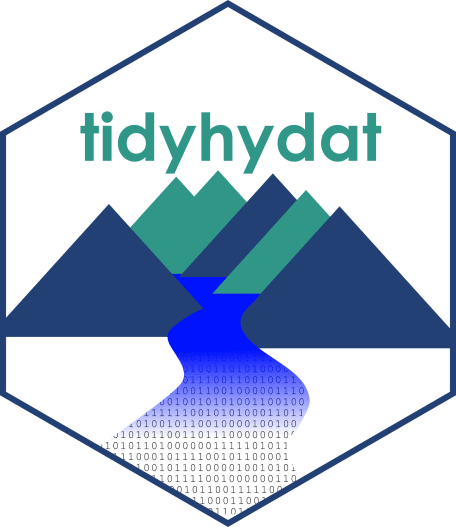
One of the best things about learning R is that no matter your skill level, there is always someone who can benefit from your experience. Topics in R ranging from complicated machine learning approaches to calculating a mean all find their relevant audiences. This is particularly true when writing R packages. With an ever evolving R package development landscape (R, GitHub, external data, CRAN, continuous integration, users), there is a strong possibility that you will be taken into regions of the R world that you never knew existed. More experienced developers may not get stuck in these regions and therefore not think to shine a light on them. It is the objective of this post to explore some of those regions in the R world that were [highlighted for me when the tidyhydat package was reviewed by rOpenSci](https://github.com/ropensci/onboarding/issues/152).



tidyhydat is a new rOpenSci R package that provides a standard method of accessing [Environment and Climate Change Canada’s](https://www.canada.ca/en/environment-climate-change.html) [Water Survey of Canada](https://wateroffice.ec.gc.ca/index_e.html) data sources (WSC) using a consistent and easy to use interface that employs [tidy data principles](https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html). I developed the package because I was repeating the [same steps](https://twitter.com/drob/status/611885584584441856?ref_src=twsrc%5Etfw&ref_url=http%3A%2F%2Ftinyheero.github.io%2Fjekyll%2Fupdate%2F2015%2F07%2F26%2Fmaking-your-first-R-package.html) to [import and tidy](http://r4ds.had.co.nz/explore-intro.html) WSC data in R. Many useRs will read this post and think “I’d know how to do that”. To you I tip my hat. To those who read this and think “huh, never knew that”, you are the target audience. In this post, I’ll focus on 5 things that I learned during the tidyhydat rOpenSci review, provide a brief introduction on how the package works and provide an example use case.

tidyhydat can be installed from CRAN by the usual manner:

install.packages("tidyhydat")

**Five things**

**1. Documentation**

A second and third set of eyes, in the form of rOpenSci reviewers, on the tidyhydat documentation added to the quality of the package immensely. It is so easy to write documentation that makes sense when you are writing the code, and makes zero sense to anyone else. Having reviewers review documentation and politely tap you on the shoulder to say “this is barely English” is the clear line between a package that is useful to only you and a package that is useful to others. From adding examples to each and every function, to suggestions that units be included in function documentation, all provide a more clear pathway for a potential new user to navigate the package.

**2. Where to store external data**

The single biggest feature of tidyhydat that caused the most headaches and learning opportunities for me is the fact that much of its capabilities rely on an external .sqlite3 data: HYDAT. HYDAT is the WSC’s quarterly published [archive of hydrometric data](http://collaboration.cmc.ec.gc.ca/cmc/hydrometrics/www/). Some stations data range extends into the 19^th^ century. It is, however, rather messy data and in a format (.sqlite3) that is likely unfamiliar to many useRs accustomed to flat data files like .csv, .txt or .xlsx.

tidyhydat seamlessly removes the need for a user to ever deal with the .sqlite3 file directly except running a function to download HYDAT upon installation:

download\_hydat()

Well, at least it does now. Previously, I had devised some convoluted solution that involved downloading the data to a specific directory then setting that directory in your .Renviron file. That is when one of the reviewers suggested the wonderfully simple rappirs package. As is so often the case, my internal dialogue of “wow, this is a cool package!” was followed by “and of course Hadley Wickham wrote it”. (Author’s note: *does he sleep?*). Here is the description of the [rappdirs](https://cran.r-project.org/package=rappdirs) package:

An easy way to determine which directories on the users computer you should use to save data, caches and logs.

This is exactly what I needed. Via rappdirs::user\_data\_dir() I was able to specify a directory to download the database that is OS-independent then tell every hy\_\* function to look for HYDAT in that directory.

**3. Prefix-style function naming**

In addition to interacting with HYDAT, tidyhydat also provides a series of functions to import real-time hydrometric data into R. Initially, to distinguish real-time functions from those that access HYDAT, I employed a confusing scheme where HYDAT functions were capitalized (DLY\_FLOWS) and real-time data functions were presented as lowercase (download\_realtime). Luckily another helpful suggestion by a reviewer was to employ consistent prefix-style naming rather than capitalization to accomplish this. Along with a slight function name change, DLY\_FLOWS() became hy\_daily\_flows(). This small change actually solves several problems at once:

* Prefixes act as sort of a package specific namespace helping to avoid any conflicts with other packages.
* Prefixes facilitate auto-completion because users only have to type hy\_ (+ TAB) at which point they can choose the function they would like to use.
* Enable a user to immediately know if a function accesses HYDAT. This unified interface provides clarity and consistency.

**4. Vignettes that use external data on CRAN**

One reviewer suggested that a vignette which worked through a complete analysis may be useful to a new user. This was a great idea. One problem: Since the package was destined for CRAN it needed to meet all the associated requirements. With respect to long-form documentation CRAN does not build vignettes and only makes sure they are actually runnable; it uses what you’ve built locally. To run through an example analysis, I needed to access the HYDAT database which is not shipped to CRAN. [This discussion](https://community.rstudio.com/t/precompiling-vignette-with-devtools/1583/6) provided a delightfully simple solution to my problem. The solution involved setting a [global knitr option](https://raw.githubusercontent.com/ropensci/tidyhydat/master/vignettes/tidyhydat_an_introduction.Rmd) for eval as TRUE or FALSE depending on the presence of an environment variable. Locally when I build the vignettes, the environment variable is present so eval=TRUE. On CRAN, the environment variable is absent for eval=FALSE. Therefore the [vignette code becomes runnable](https://cran.r-project.org/web/packages/tidyhydat/vignettes/tidyhydat_an_introduction.html) on CRAN.

**5. An exercise in reproducible science.**

tidyhydat was originally developed to save time accessing data. Instead of frequently opening database connections, making approximately the same query, forgetting to close the database connection, trying to reopen it, cursing R and generally blaming all the wrong people, I developed a package. However, after using the package it has become clear that it has a much deeper and useful feature — it is a tool that facilitates reproducible science and can provide reproducible workflows that support science-based decisions around water flow. It is now possible to set up code-based workflows that are re-runnable with more current data, different stations or regions over a range of time scales. The importance of this cannot be overstated and will be the focus of the remainder of this post.

**A reproducible work flow**

To illustrate a reproducible workflow facilitated by tidyhydat and made possible by the capabilities of R, we can ask a simple question related to a hydrometric station: how unique is this station compared to others in the same watershed? In this example we will consider the FRASER RIVER AT MISSION station. Records extend over many years and we can compare an overlapping record using correlation to develop a measure of **uniqueness** for this station. But first we need to engage in some preamble:

**Required packages**

A portion of the package audience is likely those with little or no experience in R so an example of the type of analysis possible with tidyhydat is helpful. For our purposes here, we need to install a whole other slew of packages. The packages needed for this analysis mostly fall into either the tidyverse collection of packages or the igraph family of packages. Several of these packages (dplyr, tidyr) were already installed alongside tidyhydat. The remaining packages can be installed using:

install.packages(c("purrr","corrr","igraph","ggraph","ggmap"))

then load the required packages:

library(tidyhydat)

library(dplyr)

library(tidyr)

library(corrr)

library(igraph)

library(ggraph)

library(ggmap)

**Using tidyhydat**

The first step when using tidyhydat will often start with the hy\_stations() function which holds metadata for all stations (active or discontinued) in HYDAT. WSC has a standard 7 digit station number format which contains hierarchical watershed information. The FRASER RIVER AT MISSION (08MH024) is located in the sub-sub-drainage Lower Fraser – Nahatlatch (08MH), in the sub-drainage Lower Fraser (08M) in the Pacific Ocean drainage (08). We can leverage that embedded watershed information and generate a list of station ID’s for all stations within our example watershed the Lower Fraser – Nahatlatch (08MH). An English description of the code below would follow as “Extract all stations from HYDAT then extract the first four digits from STATION\_NUMBER then filter for some sub-sub-drainage, then filter the stations for those that are active, then extract the station number as a vector”:

daily\_flows\_boxed <- hy\_stations() %>%

mutate(SUB\_SUB\_DRAINAGE\_AREA\_CD = substr(STATION\_NUMBER, 1,4)) %>%

filter(SUB\_SUB\_DRAINAGE\_AREA\_CD == "08MH") %>%

filter(HYD\_STATUS == "ACTIVE") %>%

pull(STATION\_NUMBER)

daily\_flows\_boxed

## [1] "08MH001" "08MH002" "08MH005" "08MH006" "08MH016" "08MH024" "08MH029"

## [8] "08MH035" "08MH056" "08MH076" "08MH090" "08MH098" "08MH103" "08MH126"

## [15] "08MH141" "08MH147" "08MH149" "08MH152" "08MH153" "08MH155" "08MH156"

## [22] "08MH166" "08MH167" "08MH168"

To move from the station vector to daily flow information we simply add another tidyhydat function hy\_daily\_flows() to the pipe:

daily\_flows\_boxed <- hy\_stations() %>%

mutate(SUB\_SUB\_DRAINAGE\_AREA\_CD = substr(STATION\_NUMBER, 1,4)) %>%

filter(SUB\_SUB\_DRAINAGE\_AREA\_CD == "08MH") %>%

filter(HYD\_STATUS == "ACTIVE") %>%

pull(STATION\_NUMBER) %>%

hy\_daily\_flows()

daily\_flows\_boxed

## # A tibble: 320,307 x 5

## STATION\_NUMBER Date Parameter Value Symbol

##

## 1 08MH005 1911-10-01 Flow NA

## 2 08MH006 1911-10-01 Flow NA

## 3 08MH005 1911-10-02 Flow NA

## 4 08MH006 1911-10-02 Flow NA

## 5 08MH005 1911-10-03 Flow NA

## 6 08MH006 1911-10-03 Flow NA

## 7 08MH005 1911-10-04 Flow NA

## 8 08MH006 1911-10-04 Flow NA

## 9 08MH005 1911-10-05 Flow NA

## 10 08MH006 1911-10-05 Flow NA

## # ... with 320,297 more rows

Now that we have all our hydrometric data tidy and in R, I’ll outline a cool example of what you can do with it.

**Correlate all stations in Fraser River – Lower Nahatlatch (08MH)**

Remember, we are trying to evaluate the uniqueness of the FRASER RIVER AT MISSION via correlation as compared to other stations in the watershed. We can utilize the excellent [corrr package](https://cran.r-project.org/package=corrr), specifically correlate() and stretch(), to calculate the correlation between each station all in a pipe. This series of piped functions takes all the daily flow data (daily\_flows\_boxed), converts it to long format, removes the columns we don’t want, correlates each column, stretches the data into long form and gets rid of correlations that results in NA’s:

cor\_df <- daily\_flows\_boxed %>%

spread(STATION\_NUMBER, Value) %>%

select(-Date, -Symbol, -Parameter) %>%

correlate() %>%

stretch() %>%

filter(!is.na(r))

**igraph munging**

At this point we need to begin using the [igraph](https://cran.r-project.org/package=igraph) package for network analysis. Our first step is to make a graph for all the correlations then construct a subgraph for only the 08MH024 station:

## Convert to an igraph object for plotting

graph\_correlation <- graph\_from\_data\_frame(cor\_df)

## Subset for those edges from 08MH024

graph\_sub <- subgraph.edges(graph\_correlation, E(graph\_correlation)[inc('08MH024')])

**Create plot**

Finally, our last step is to plot each point in a spatially accurate manner and apply the results from the igraph object. The plotting step graphs the station location against satellite imagery using the [ggmap](https://cran.r-project.org/package=ggmap) and [ggraph](https://cran.r-project.org/package=ggraph) packages and connect stations with coloured lines that reflect the strength of the correlation:

## Get the lat long data, isolate the coordinates and rename to x and y

latlong\_layout\_ggmap <- hy\_stations(station\_number = V(graph\_sub)$name) %>%

select(LONGITUDE, LATITUDE) %>%

rename(x = LONGITUDE, y = LATITUDE)

## Create the spatial layout based on latitude and longitude.

spatial\_layout\_ggmap <- create\_layout(graph = graph\_sub,

layout = "manual",

node.positions = latlong\_layout\_ggmap)

## Acquire the static map

map <- get\_map(c(min(latlong\_layout\_ggmap$x),

min(latlong\_layout\_ggmap$y),

max(latlong\_layout\_ggmap$x),

max(latlong\_layout\_ggmap$y)),

source = "google", maptype = 'satellite', zoom = 9)

## Plot the igraph object

ggmap(map, base\_layer = ggraph(spatial\_layout\_ggmap), device = "extent") +

geom\_edge\_link(aes(color = r), edge\_width = 1) +

guides(edge\_alpha = "none") +

scale\_edge\_colour\_gradientn(name = "Correlation \nCoefficient", colours = rainbow(8)) +

geom\_node\_point(colour = "white", size = 4) +

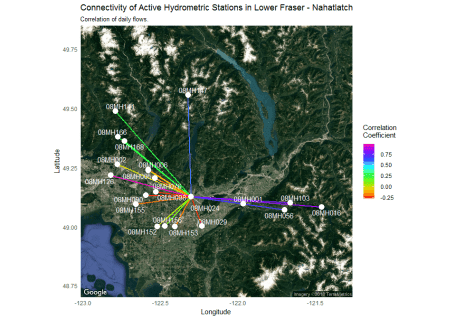
geom\_node\_text(aes(label = name), repel = TRUE, colour = "white") +

theme\_minimal() +

labs(y = "Latitude", x = "Longitude",

title = "Connectivity of Active Hydrometric Stations in Lower Fraser - Nahatlatch",

subtitle = "Correlation of daily flows.")



**What does the plot tell us**

Our original objective was to assess if the 08MH024 station was a **unique** station. This type of plot provides a clear pattern of hydrologic linkages. Flows at 08MH024 are more similar to stations at higher elevations (e.g. 08MH016) and less similar to lower elevation stations (e.g. 08MH002) demonstrating a spatial pattern of uniqueness. This likely reflects the importance of upstream hydrologic processes. The Fraser River, the largest river in British Columbia drains a huge area (228000 square km) where the flows in the majority of contributing upstream tributaries are driven by snow melt. This station, however, is located near the coastal environment where river flow is driven more by rainfall. This helps us connect upstream and downstream processes and evaluates the relative contributions to flow in the main river. A visual tool, such as the one illustrated here, provides a clear example of using the capabilities of R after bringing in a rich hydrometric data set, and highlights a reproducible hydrological workflow in R.

**Acknowledgements**

I would like to thank rOpenSci for the opportunity to have the tidyhydat package reviewed. From the package review side of things, I would like to thank [Scott Chamberlain](https://github.com/sckott), [Laura DeCicco](https://github.com/ldecicco-USGS) and [Luke Winslow](https://github.com/lawinslow) for their time and efforts in improving the package. Hopefully this post has outlined the ways in which their contributions were useful. From the being awesome and helpful side of things, I also say thanks to [Stefanie Butland](https://twitter.com/StefanieButland?lang=en) for helping shepherd this package to rOpenSci. More generally tidyhydat offers another piece of evidence that the rOpenSci package review process consistently yields better results and returns more user friendly products that facilitate open science.