Finding and accessing environmental data in R

ENS-215

18-Feb-2022

In today's lecture we are going to learn how to make use of a number of R packages that allow you to directly query and access data from some notable environmental databases. These data access packages are incredibly powerful as they allow you to streamline your workflow by combining the data identification and acquisition steps directly with your data analysis. They also provide a highly efficient and reproducible approach to data identification and acquisition - allowing you to easily adapt or expand your research question with minimal additional effort.

There are many packages that have been developed for accessing data through R, including a wide range of packages for accessing environmental datasets. Today we will look at a few notable packages and in the coming weeks we will explore some additional ones. The packages below highlight a number of notable research-quality datasets that are likely to be useful in many of your class projects, theses, and future research/work.

Let's load in a few packages that we will use today. It is likely that many of you may not yet have installed the `leaflet` package. If so, you should go ahead and do so before proceeding. FYI, the leaflet package allows you to make interactive maps. You will learn how to do this in the upcoming weeks, however a few of the packages used today will automatically generate leaflet maps and so you will need this package for those maps to be automatically generated.

```r
library(tidyverse)
library(lubridate)
library(leaflet)
```

Climate data

In much of our environmental research we need to use climate/meteorological data. While there are a host of organization/agencies that provide freely available research quality datasets, locating and accessing the data can often be time-consuming and difficult. Furthermore, once you've located the data (generally through the website of the agency/organization collecting the data) you typically need to download the file (e.g., csv, Excel), load it into R, and reformat the data for your analysis. The R packages below allow you to quickly locate the data of interest and directly load it into R in a format that is ready to use in R - typically all within a single function call!

GSODR package: access daily meteorological data from NOAA

The **Global Surface Summary of the Day - GSOD** is a dataset from the National Oceanic and Atmospheric Administration (NOAA) that provides daily average values for a range of meteorological variables (e.g., air temperature, precipitation, wind speed, barometric pressure) at thousands of locations around the globe. These daily average data are computed hourly observations at meteorological stations (i.e., they are actual observations and not model/simulation outputs) and the data goes back upwards of 90 years at some locations. FYI, the hourly data is available through the Integrated Surface Hourly (ISH) dataset from NOAA, which we will learn how to access later in today's lecture.

We will learn how to use the `GSODR` package to directly access this data.

First let's load in the package (FYI, you will need to install the package first if you have not installed it before)

```r
library(GSODR)
```

```
## Warning: package 'GSODR' was built under R version 4.0.5
```

The package comes with a dataset that has all of the available meteorological stations and their location information/details (e.g., lat/long, country, state, period of data coverage). We will load in this data below.

```r
load(system.file("extdata", "isd_history.rda", package = "GSODR"))
```

Take a look at the `isd_history` data frame to familiarize yourself with what it contains. You'll see that you could use this information to easily identify sites of interest to your particular research question. Below you can see a map of all of the available meteorological stations in the GSOD database.

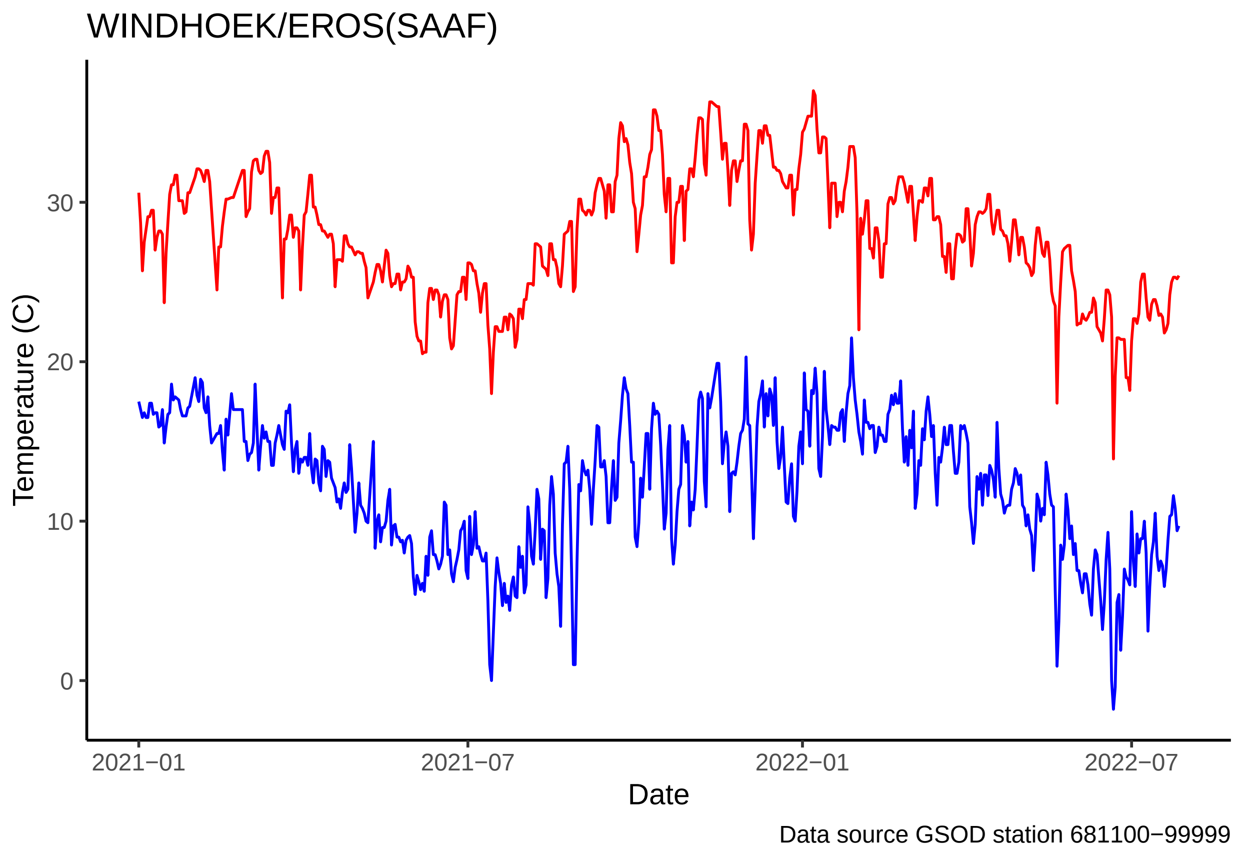
If you identified a station(s) of interest you can download the data using the `get_GSOD()` function. Let's try this out for Windhoek, which is the capital of the country of Namibia (located in SW Africa). To download data for a given station, you need to supply the station ID (STNID) and the years of interest. You can locate the STNID in the `isd_history` data frame.

```r
met_df <- get_GSOD(years = 2021:2022, station = "681100-99999")
```

Take a look at the `met_df` data frame. The description of the variables can be found here.

Let's plot the a time-series of the daily minimum and maximum air temperatures from 2021-2022 at the selected met station.

```r
met_df %>%
  ggplot() +
  geom_line(aes(x = YEARMODA, y = MIN), color = "blue") +
  geom_line(aes(x = YEARMODA, y = MAX), color = "red") +

  labs(title = met_df$NAME[1],
       x = "Date",
       y = "Temperature (C)",
       caption = paste("Data source GSOD station", met_df$STNID[1])
       ) +

  theme_classic()
```



Oftentimes you have a study location/area of interest and you would like to identify any/all meteorological stations that exist nearby your site. The `GSODR` package has a really helpful function, `nearest_stations()` that does this. You need to supply the lat/long of the area to query and the radial distance (in kilometers) from your search point. The function will then return a list of the STNID with the station info for all stations that fall within your search radius.

Let's test this out by searching for all of the stations within 25 km of Schenectady.

```
schdy_met_stations <- nearest_stations(LAT = 42.81,
                                       LON = -73.94,
                                       distance = 25)



schdy_met_stations
```

```
## [1] "744994-99999" "999999-14735" "725180-14735"
```

We can see that there are 3 meteorological stations within 25 km of Schenectady.

- Download the data from 2010-2015 for the stations nearby Schenectady

- Try downloading meteorological data for a location you are interested in. Create some figures and/or tables that provide insight into that locations meteorological conditions.

worldmet: access hourly meteorological data from NOAA

In the section above, we learned how to use the `GSODR` package to retrieve daily meteorological data from thousands of meteorological stations around the globe. Now we will learn how to access the hourly data from NOAA's Integrated Surface Database using the `worldmet` package. Note that this package is very similar to

GSODR and in fact they access data from the same set of stations (with GSODR providing daily summaries and worldmet providing access to the hourly data).

Let's load in the worldmet package. If you haven't yet installed worldmet you will need to go ahead and do so first.

```
library(worldmet)
```

```
## Warning: package 'worldmet' was built under R version 4.0.5
```

The worldmet package is pretty straightforward to use - you simply need to specify the station you would like access and the year(s) you would like to download the data for.

We can use the getMeta() function to obtain a data frame containing information (e.g., station ID, name, lat/long,...) about each of the stations that have data available for download.

```
worldmet_site_df <- getMeta(plot = F)
```

Take a look at the data frame you just obtained and you will see that the code column contains the station ID that is unique to each site. The data frame also has a bunch of additional useful information about each site, including the time period over which data is available.

Note that I set in the function call above I set the plot argument to false. If you set it equal to true (T), then an interactive map will be generated showing the location of all the sites. To have the map show up when you knit your document you will also need to set the returnMap argument to T. Let's make a map of all of the available sites.

```
getMeta(plot = T, returnMap = T)
```

Ok, now we can go ahead and use the importNOAA() function to download some hourly meteorological data. This function requires two arguments. The code argument is where you specify the station ID. The year argument is where you specify the year(s) you would like to download. To specify multiple years you would set the years argument as follows years = 2010:2022, which in that case would download hourly data from 2010 through 2022.

Let's go ahead and download hourly data for Albany International Airport (station code: 725180-14735) for 2021 through the most recent reported measurement (~ 1 or 2 days prior to today).

```
worldmet_data <- importNOAA(code = "725180-14735",
                                     year = 2021:2022)
```

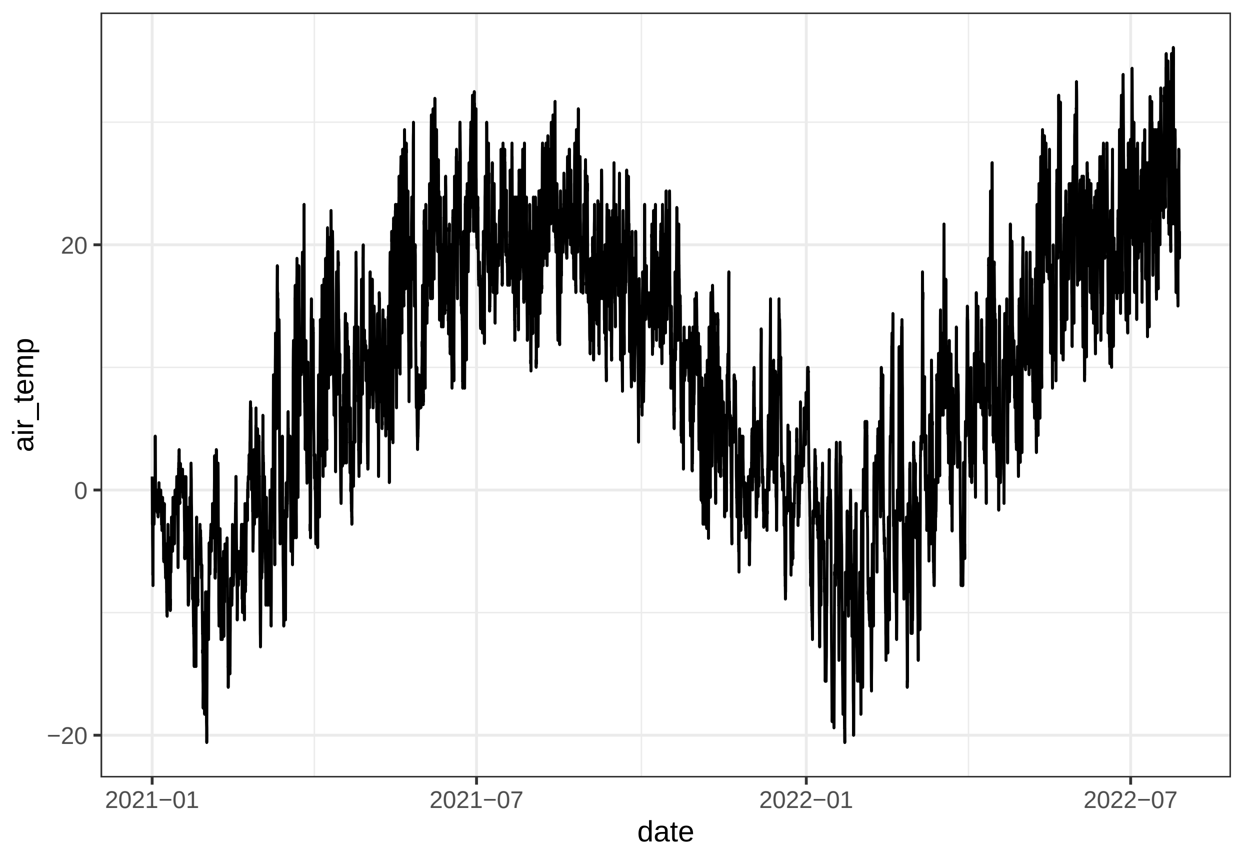
Descriptions of all the variables and their units are available here

Important: All data are reported in Greenwich Meantime (GMT) time zone format. This means that you may want to convert the time data to the stations local time zone.

Let's quickly plot some of the data to see how it looks.

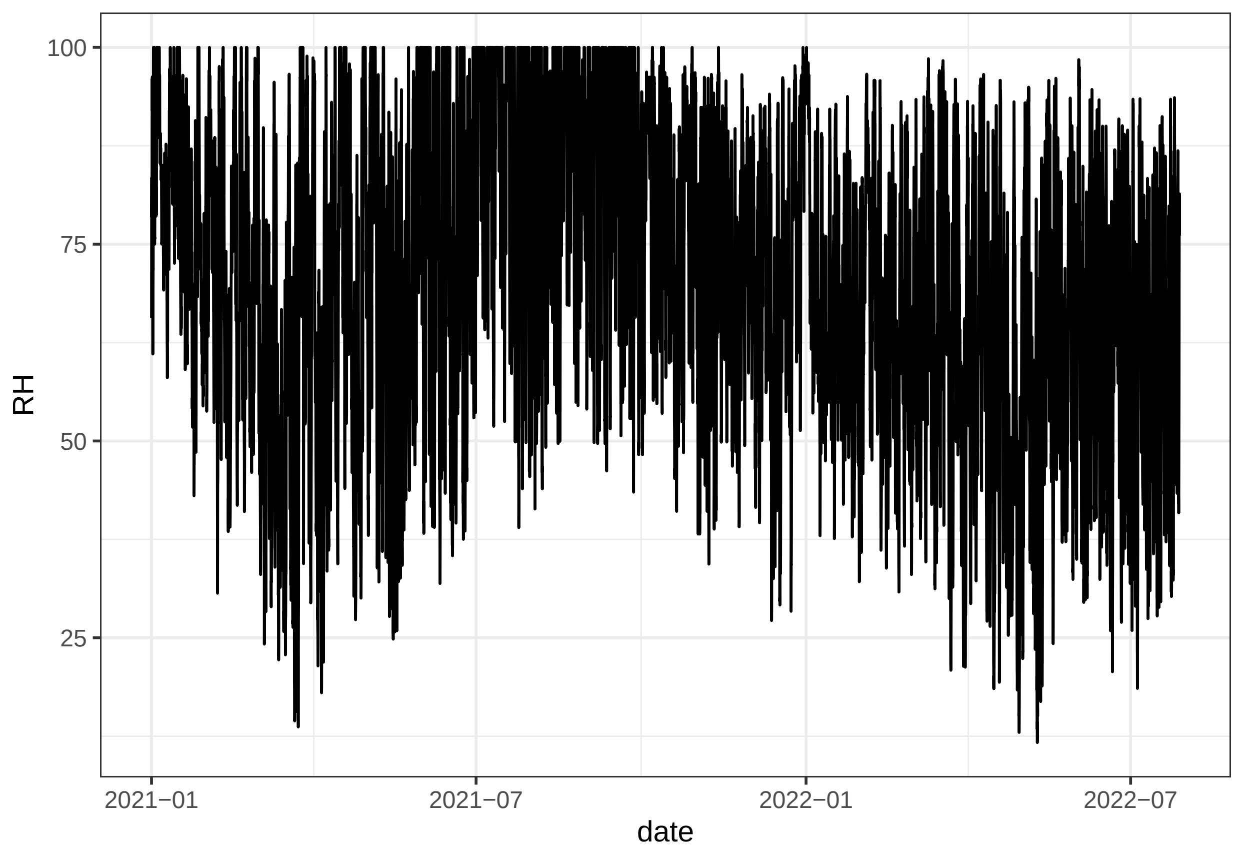
Hourly temperature data

```
worldmet_data %>%
  ggplot(aes(x = date, y = air_temp)) +
  geom_line() +

  theme_bw()
```



Hourly relative humidity data

```
worldmet_data %>%
  ggplot(aes(x = date, y = RH)) +
  geom_line() +

  theme_bw()
```



Hydrological data using the `dataRetrieval` package

The USGS National Water Information Service (NWIS) is an incredible database containing physical and chemical data on the streams/rivers, lakes, and groundwater for nearly 2 million locations across the US and US territories. From NWIS you can download data related to thousands of different parameters including streamflow, groundwater levels, and a wide range of water chemistry and physical conditions.

This data can be queried and retrieved using the `dataRetrieval` package that was developed and is maintained by the USGS. Given the richness and volume of the data there are many functions and features within this package and some knowledge of how the available data is particularly helpful when using `dataRetrieval`. Today we'll learn how to download daily streamflow data and also learn the basics of downloading water quality data. For those interested in learning more I am helping to provide additional guidance and you can also explore the excellent background and tutorial articles that have been put together by the developers of the package.

When querying the NWIS database to find and/or download available data you generally need to specify a parameter (i.e., what was being measured) and a location/region of interest.

If you already know the USGS site number (the unique numeric identifier the USGS assigns to sampling sites) and you know that the parameter of interest was measured at the site you can proceed to the data acquisition step. However, many times we don't know what/if available sites with the desired data exist and we need to first identify those sites. Once you've identified the sites with the desired data, you can then directly acquire the data for the site(s) of interest.

Let's work through an example where we locate sites in Schenectady county that have mean daily streamflow and/or gauge height (i.e., stream water level) data. Note that we could look for other parameters (e.g., turbidity, concentration of chloride, pH,. . .) if we were interested in those measurements.

First let's load in the `dataRetrieval` package. If you have never used this package before you will need to install it before proceeding.

```
library(dataRetrieval)
```

```
## Warning: package 'dataRetrieval' was built under R version 4.0.5
```

Identify sites with desired data

We are going to query the NWIS database for sites within Schenectady county (`countyCd = "36093"`) that have streamflow (00060) and/or gauge height (i.e., water level) (00065) data.

Note that every US county has a FIPS county code, which is a unique numeric code that the US has assigned to each county. You can find a list of all the county codes here.

You can also find a list of all of the parameter codes used by the USGS here.

You'll also note that we specified `service = "dv"`, this means to query for sites that have daily data reported.

```
sites_schdy <- whatNWISsites(countyCd = "36093",
                             parameterCd = c("00060","00065"),
                             service = "dv"
                             )



sites_schdy
```

```
##   agency_cd  site_no                                   station_nm
## 1      USGS 01351500                 SCHOHARIE CREEK AT BURTONSVILLE NY
## 2      USGS 01351450                    SCHOHARIE CREEK AT ESPERANCE NY
## 3      USGS 01354230    MOHAWK RIVER AT LOCK 9 AT ROTTERDAM JUNCTION NY
## 4      USGS 01354330         MOHAWK RIVER AT LOCK 8 NEAR SCHENECTADY NY
## 5      USGS 01354490                     MOHAWK RIVER AT SCHENECTADY NY
## 6      USGS 01354500 MOHAWK RIVER AT FREEMAN'S BRIDGE AT SCHENECTADY NY
## 7      USGS 01356190                 LISHA KILL NORTHWEST OF NISKAYUNA NY
## 8      USGS 01356200                          LISHA KILL AT NISKAYUNA NY
##   site_tp_cd dec_lat_va dec_long_va colocated           queryTime
## 1         ST   42.80003   -74.26278     FALSE 2022-08-01 14:38:41
## 2         ST   42.76058   -74.25478     FALSE 2022-08-01 14:38:41
## 3         ST   42.87828   -74.04075     FALSE 2022-08-01 14:38:41
## 4         ST   42.82816   -73.99037     FALSE 2022-08-01 14:38:41
## 5         ST   42.81980   -73.94929     FALSE 2022-08-01 14:38:41
## 6         ST   42.83053   -73.93075     FALSE 2022-08-01 14:38:41
## 7         ST   42.78361   -73.85667     FALSE 2022-08-01 14:38:41
## 8         ST   42.79174   -73.84734     FALSE 2022-08-01 14:38:41
```

You can see that there are 8 sites that meet the search criteria. You'll also note that under the `site_tp_cd` column you see *ST*, which stand for stream. If you queried for a parameter (e.g., water temperature) that could also apply to other site types, then you would likely see other `site_tp_cd` values listed too (e.g., *GW*).

Now let's get information about the available data (e.g., the number of observations, the length of the data record).

```
sites_what_data <- whatNWISdata(siteNumber = sites_schdy$site_no,
                                service = "dv",
                                parameterCd = c("00060","00065"),
                                statCd = "00003")


sites_what_data
```

```
##     agency_cd  site_no                                     station_nm
## 2        USGS 01351450                   SCHOHARIE CREEK AT ESPERANCE NY
## 347      USGS 01351500                SCHOHARIE CREEK AT BURTONSVILLE NY
## 615      USGS 01354230    MOHAWK RIVER AT LOCK 9 AT ROTTERDAM JUNCTION NY
## 619      USGS 01354330         MOHAWK RIVER AT LOCK 8 NEAR SCHENECTADY NY
## 742      USGS 01354500 MOHAWK RIVER AT FREEMAN'S BRIDGE AT SCHENECTADY NY
## 750      USGS 01356190               LISHA KILL NORTHWEST OF NISKAYUNA NY
##     site_tp_cd dec_lat_va dec_long_va coord_acy_cd dec_coord_datum_cd alt_va
## 2           ST   42.76058   -74.25478            1              NAD83 551.31
## 347         ST   42.80003   -74.26278            1              NAD83 507.44
## 615         ST   42.87828   -74.04075            1              NAD83     NA
## 619         ST   42.82816   -73.99037            H              NAD83 199.46
## 742         ST   42.83053   -73.93075            1              NAD83 199.46
## 750         ST   42.78361   -73.85667            S              NAD83 238.78
##     alt_acy_va alt_datum_cd   huc_cd data_type_cd parm_cd stat_cd  ts_id
## 2         0.01       NAVD88 02020005           dv   00065   00003 237107
## 347       0.01       NAVD88 02020005           dv   00060   00003 105043
## 615         NA         <NA> 02020004           dv   00065   00003 265679
## 619       0.01       NAVD88 02020004           dv   00065   00003 105047
## 742       0.01       NAVD88 02020004           dv   00060   00003 292954
## 750       0.01       NGVD29 02020004           dv   00060   00003 105052
##     loc_web_ds medium_grp_cd parm_grp_cd   srs_id access_cd begin_date
## 2         <NA>           wat        <NA> 17164583         0 2017-12-03
## 347       <NA>           wat        <NA>  1645423         0 1939-10-01
## 615       <NA>           wat        <NA> 17164583         0 2019-12-12
## 619       <NA>           wat        <NA> 17164583         0 2011-12-14
## 742       <NA>           wat        <NA>  1645423         0 2011-08-06
## 750       <NA>           wat        <NA>  1645423         0 1993-08-26
##       end_date count_nu
## 2   2022-07-31     1681
## 347 2022-07-31    30255
## 615 2022-07-31      907
## 619 2022-07-31     3874
## 742 2022-07-31     4013
## 750 2012-09-29     5878
```

In the above function call you see that we set the argument `statCd = "00003"`. This specifies the statistic to download. The code `00003` means to obtain the average value. We need to specify this since we are downloading daily values and since most sites take sub-hourly readings we need to tell the function what type of daily value to return (e.g., minimum measurement, maximum, average,…). If you want to obtain a different statistic than the average you can simply change the statistic code.

You can find a list of USGS statistic codes here.

We can see that we have information about the data records for the sites with available gauge height (parm_cd = 00065) and streamflow (parm_cd = 00060). Below is a map of these sites.

Based on this information you might choose to acquire the data for some or all of these sites. Since there are only a few sites, let's go ahead and download the water gauge height and streamflow data for all of these sites.

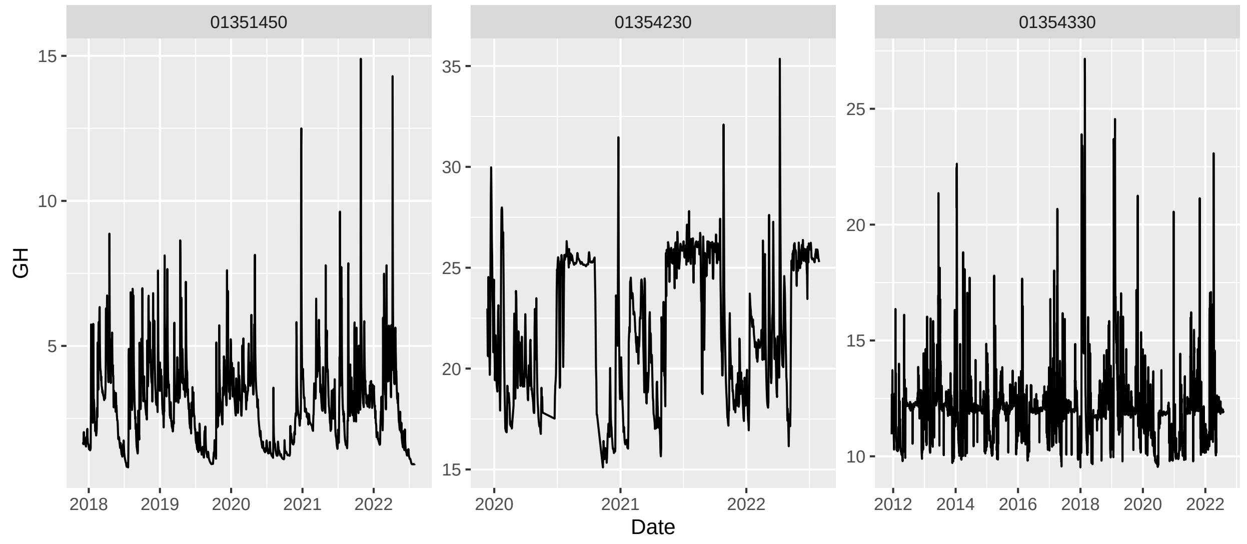
```
df_stream_data <- readNWISdv(siteNumbers = sites_what_data$site_no,
                             parameterCd = c("00060","00065"),
                             statCd = "00003")
```

When you download the data it will have column names that refer to parameter codes and other data descriptors that are not particularly human readable. Let's use the `renameNWISColumns()` function, which renames the columns with more human readable names.

```
df_stream_data <- df_stream_data %>%
  renameNWISColumns()
```

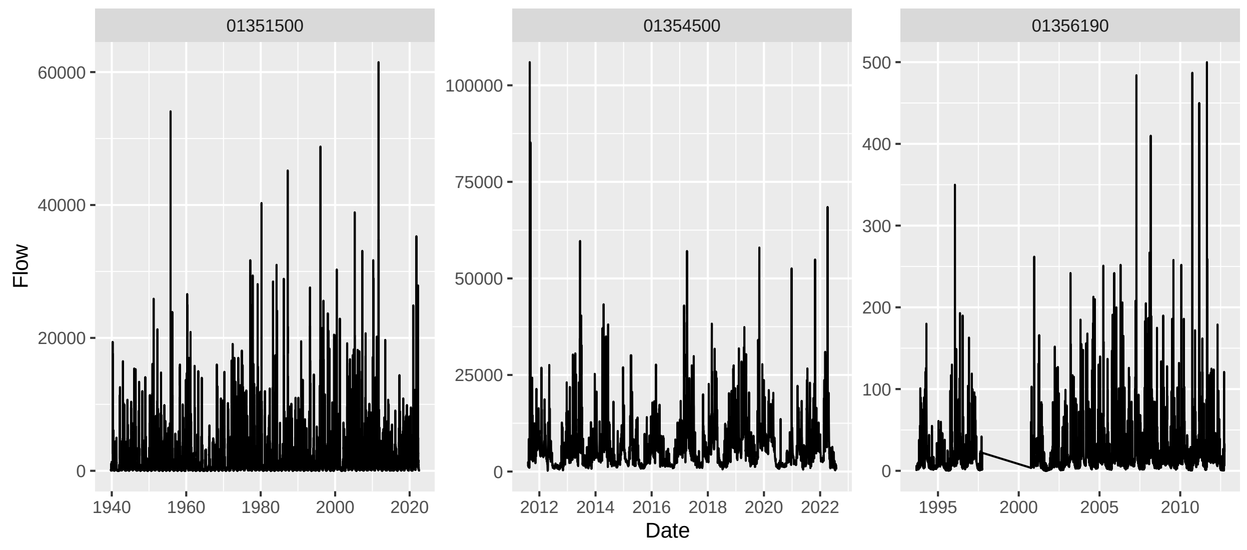
Now let's plot a time-series of the gauge heights for each of the sites.

```
df_stream_data %>%
  filter(!is.na(GH)) %>% # remove NA gauge heights

  ggplot(aes(x =Date, y = GH)) +
  geom_line() +

  facet_wrap(~ site_no, scales = "free")
```



Now let's plot a time-series of the streamflow for each of the sites.

```
df_stream_data %>%
  filter(!is.na(Flow)) %>%

  ggplot(aes(x =Date, y = Flow)) +
  geom_line() +

  facet_wrap(~ site_no, scales = "free")
```



Water quality example

The NWIS database has over 4.4 million water quality measurements across hundreds of thousands of locations throughout the US and US territories. We can use the `dataRetrieval` package to query and access this data. As noted earlier, the `dataRetrieval` package and understanding NWIS data can take a bit of time to learn, though we'll work through an example to highlight just how powerful `dataRetrieval` is for obtaining research grade water quality measurements.

The key first step is to specify what water quality parameter(s) you are interested in obtaining. You can find the list of the available USGS parameter codes here.

You can also obtain a data frame of the USGS parameter codes directly in R, by calling `parameterCdFile`. Let's do that now.

```
usgs_parm_cd_df <- parameterCdFile
```

Take a look at this data frame. You'll see that there are more than 16,000 parameter codes! You'll see that each parameter code is assigned to one 16 different groups (e.g., physical, microbiological, nutrient,...). This makes it easier to located parameters by their group/type. You'll also see that the full name and the units associated with each parameter are reported in the data frame.

Let's go ahead and obtain data for **Nitrate, water, filtered, milligrams per liter as nitrogen** which is parameter code (00618).

Since there are likely 10's or hundreds of thousands on nitrate measurements in the NWIS database we are going to limit our query by location and by date.

Let's use the `whatNWISdata()` function to find out what data is available for nitrate (see the `parameterCd` argument) in New York state (see the `stateCd` argument). We will limit our query to sites that have at least some measurements that were collected after 1-Jan-2015 (see the `startDate` argument). Note that the sites can have data collected before that date, however they must have at least one measurement after that date in order to satisfy our search criteria.

Note that we are filtering the data so that we only keep groundwater sites.

```
pCode <- c("00618")

NY_NO3 <- whatNWISdata(stateCd="NY",
                       parameterCd=pCode,
                       startDate = "2015-01-01"
```

```
                     ) %>%

  filter(site_tp_cd == "GW")
```

Take a look at the `NY_NO3` data frame. You'll see that this data frame has information on all of the available data based on the above query. You'll see that each row represents a site with data. The last three columns of this data frame are particularly useful - they contain the date of the first (`begin_date`), last (`end_date`) measurement, and the number of total measurements for that site (`count_nu`). You'll see that some sites have multiple measurements, indicating that samples were collected on a number of different dates/times.

Now, we will use the `readNWISqw()` function to acquire the data for the sites we have identified.

To do this we need to specify a site(s) to obtain. We do this by providing the `siteNumbers` argument with a vector of sites numbers.

We also need to specify the time period for the measurements we would like to obtain. We will limit ourselves to only downloading measurements that were taken from 1-Jan-2015 onward (see the `startDate` argument).

We also need to specify the parameter to download. Since we are interested in nitrate data we set the `parameterCd` argument to be equal to the nitrate parameter code of interest (i.e., 00618).

```
NY_NO3_recent_data <- readNWISqw(siteNumbers = NY_NO3$site_no,
                                 startDate = "2015-01-01",
                                 parameterCd = pCode)
```

```
## Warning: NWIS qw web services are being retired.
## Please see vignette('qwdata_changes', package = 'dataRetrieval')
## for more information.
## https://cran.r-project.org/web/packages/dataRetrieval/vignettes/qwdata_changes.html
```

Now we've downloaded the data - take a look at the data frame. You'll see the measurement values in the `result_va` column.

Recall that some sites have multiple measurements (since measurements were taken on a number of different dates/times). Let's summarize our data so that we can get the maximum nitrate concentration observed at each of the sites.

To do this we are going to group by `site_no` (i.e., group by the different sampling sites/locations) and then summarize the nitrate measurements to take the maximum value observed at each site.

```
NY_NO3_recent_data_summary <- NY_NO3_recent_data %>%
  group_by(site_no) %>%
  summarize(NO3_max = max(result_va, na.rm = T)) %>%

  left_join(NY_NO3) # joining our water quality data with the data frame that contains site information
```

```
## Joining, by = "site_no"
```

Take a look at this data frame to make sure you understand the steps above and what we have done.

Below is a map showing the data that we just summarized.

We will learn how to make these types of maps in upcoming lectures.

Exercises

Continue to experiment with the packages that we have learned how to use today. Think about how you might apply these packages to your final projects. Some possible things to explore include:

- Try obtaining daily or hourly meteorological data for site(s) that you are interested in.

- Compare meteorological conditions between sites of interest.
- Perform some of the time-series visualization/analysis approaches to the meteorological data.
- Use the `dataRetrieval` package to obtain data for additional parameters and sites from the NWIS data base. Spend some time exploring this data.