

Filtering water quality dataset

Amber Lee

6/15/2021

Unfinished

Is it okay if, after filtering/processing the original data, we only have 105,267 rows? We are only keeping about half of the original data...

Summary

One important goal before we interpolate missing values is to clean the data. Cleaning includes filtering for relevant rows and selecting the relevant variables. This will decide on the scope of our analysis. We will also see that removing duplicate rows will reduce the missing values (that aren't actually missing), thus making the interpolation step significantly easier.

In cleaning the LTRM Water Quality dataset, we encounter the following questions:

- 1) Which variables (columns) are the most important for us to keep (out of the 133 total variables)?
- 2) Why do duplicate rows happen, and how do we deal with them?
- 3) Which samples (rows) are high enough quality for us to keep? (This involves the QF codes.) We want to set low quality data to NA for interpolation later.

The LTRM Water Quality dataset has duplicate rows for the same SHEETBAR, which is problematic because SHEETBAR is a unique identifier for a water data sheet (sample at a date, time, and location).

Load libraries

```
library(tidyverse)
library(ggplot2)
library(lubridate)
library(corrplot)
library(RColorBrewer)
library(kableExtra)
```

Read data

```
# set working directory to source file location
# setwd("~/Documents/GitHub/UMR-TDA-2021")

water20 <- read.csv(file = "../LTRM data/ltrm_water_data_lat_long.csv")
```

1. These are the important variables

```
water_var <- c('TN', 'TP', 'TEMP', 'DO', 'TURB',  
              'COND', 'VEL', 'SS', 'WDP', 'CHLcal', 'SECCHI')  
  
waterQF_var <- paste(water_var, "QF", sep = "")  
  
identifier_var <- c('SHEETBAR', 'DATE', 'LATITUDE', 'LONGITUDE', 'FLDNUM', 'STRATUM', 'LOCATCD')  
  
waterQF_var <- waterQF_var[waterQF_var != "WDPQF" &  
                           waterQF_var != "CHLcalQF"]  
  
waterQF_var <- c(waterQF_var, "ZMAXQF")
```

We decided that the 11 continuous variables of importance were: total nitrogen, total phosphorous, temperature, dissolved oxygen, turbidity, water condition, velocity, suspended solids, water depth, chlorophyll-a, and Secchi distance.

In addition, we will want to include the QA/QC codes, along with identifier variables like SHEETBAR and date.

Lastly, we manually edit these variable strings because:

- The water depth variable is WDP, but the corresponding quality factor is ZMAXQF rather than WDPQF.
- CHLcal, calibrated fluorometric chlorophyll a, does not have a corresponding quality factor code. According to the metadata: "CHLcal is generated by calibration of fluorometric chlorophyll readings (CHLF) to season and year specific measurements of spectrophotometric chlorophyll (CHLS). Data from sites where CHLS and CHLF are both collected are used to build river-specific calibration curves for these data. Values are corrected for pheophytin. Units are micrograms per liter."

2. Investigating duplicate SHEETBARs in the data

At the same sheetbar, multiple samples can be taken at different water depths.

There are 204305 total rows in the LTRM water quality dataset. Of these rows, there are 156474 distinct SHEETBAR codes.

We visualize duplicates as follows. To identify the duplicate rows, we count the number of occurrences of each unique SHEETBAR value in the dataset. Then, we can calculate and plot the distribution of SHEETBAR duplicates.

```
duplicates <- water20 %>%  
  select(SHEETBAR) %>%  
  group_by(SHEETBAR) %>%  
  summarize(count = n())  
  
duplicates %>% head()
```

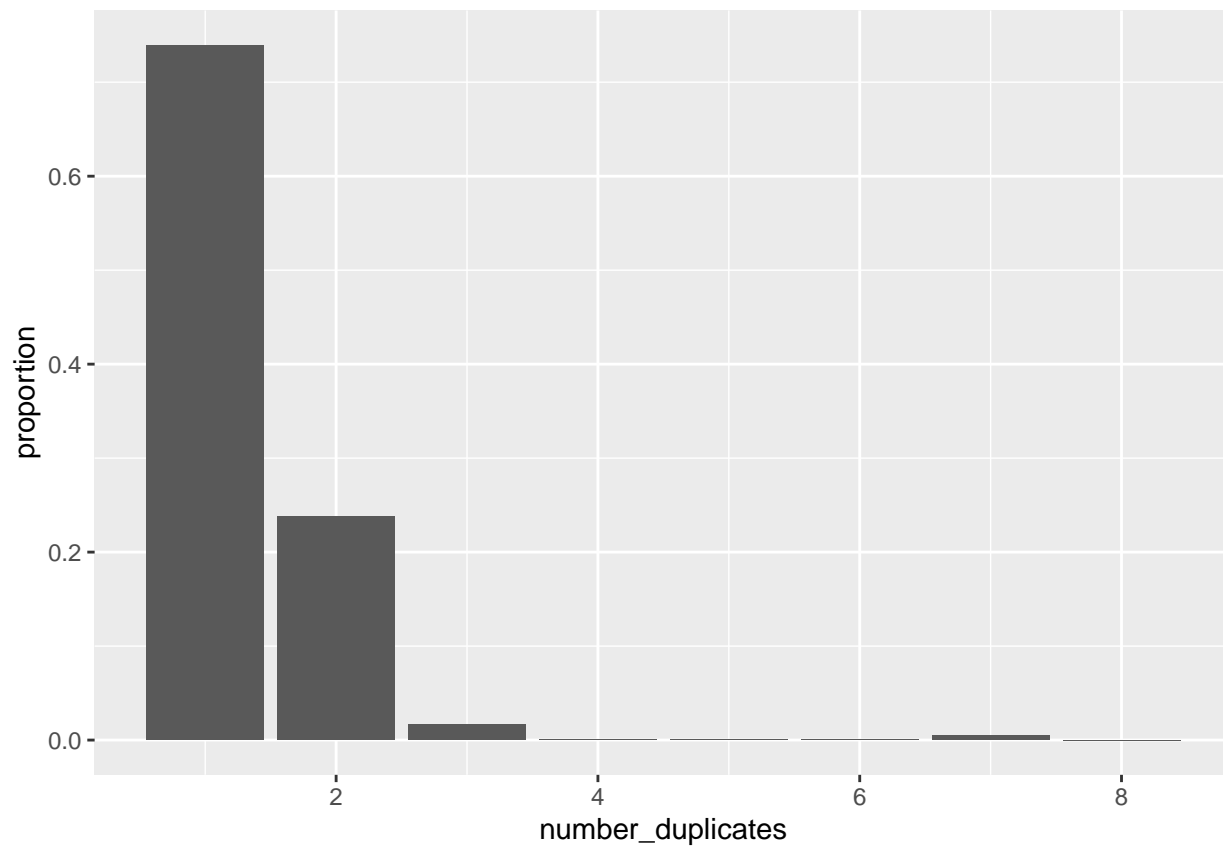
```
## # A tibble: 6 x 2  
##   SHEETBAR count  
##   <int> <int>  
## 1 -4604348     1  
## 2 -4604347     2  
## 3 -4604346     2  
## 4 -4604345     2
```

```
## 5 -4604344      1
## 6 -4604343      2

count_n_duplicates <- function(n, df) {
  return((df %>% filter(count == n) %>% dim())[1]/156474) #156k distinct sheetbars
}

count_duplicates <- data.frame(proportion = sapply(1:8, count_n_duplicates,
                                                  duplicates),
                              number_duplicates = 1:8)

ggplot(count_duplicates, aes(x = number_duplicates, y = proportion)) +
  geom_bar(stat = "identity")
```



```
count_duplicates %>% kbl(booktabs = T)
```

proportion	number_duplicates
0.7393497	1
0.2377072	2
0.0166481	3
0.0006199	4
0.0002365	5
0.0007605	6
0.0046781	7
0.0000000	8

SHEETBAR	Z	CALCZCD	DO	TP	TN
-4604347	0.2	SF	7.4	0.265	7.49
-4604347	4.8	BT	7.5	NA	NA

SHEETBAR	Z	CALCZCD	DO	TP	TN
41015929	0.2	SF	14.4	0.075	2.557
41015929	1.0	OT	14.8	NA	NA
41015929	2.0	OT	14.9	NA	NA
41015929	3.0	OT	15.0	NA	NA
41015929	4.0	OT	15.2	NA	NA
41015929	4.6	BT	15.2	NA	NA

The proportion of SHEETBARs with at least one duplicated row is 0.2606503 (representing about 47,000 rows). When do duplicated rows occur?

We look at two SHEETBARs with duplicated rows.

```
water20 %>%
  filter(SHEETBAR == -4604347) %>%
  select(SHEETBAR, Z, CALCZCD, DO, TP, TN) %>%
  kbl(booktabs = T) %>%
  kable_styling(latex_options = "striped")
```

```
water20 %>%
  filter(SHEETBAR == 41015929 ) %>%
  select(SHEETBAR, Z, CALCZCD, DO, TP, TN) %>%
  kbl(booktabs = T) %>%
  kable_styling(latex_options = "striped")
```

Here, we see that TP and TN, total phosphorous and total nitrogen, are measured only at the surface level (when CALCZCD == "SF"). The variable CALCZCD is a categorical variable with levels surface, middle, bottom, and other. It is calculated with the sample depth and the total water depth (of the river site).

In contrast, dissolved oxygen DO is measured at various depths (denoted by Z) because different parts of the water column have different levels of DO. It would be inappropriate to average the dissolved oxygen levels because they were taken at different sample depths.

Thus, this missing values of TP and TN are occurring at different sample depths at the same sampling site. These missing values aren't *really* missing values; they would be redundant to interpolate.

We can reasonably keep only the samples taken at the surface level

We decided to filter for rows that were labelled as surface level, CALCZCD == "SF". Implicitly, this filtering step removes samples for which the sample depth is missing.

```
table(water20$CALCZCD)
```

```
##
##          BT          MD          OT          SF
##  8436  34991   2971  10736  147171
```

More than 70% of the samples were taken on the surface level. Of these measurements taken at the surface level, the eleven important continuous variables (in `water_var`) were recorded with a recording rate of at least 50%. This is a good sanity check because TN and TP are never recorded in the middle and bottom water

depths. we checked to see the recording rate of the eleven important variables and found recording rates greater than 50%.

```
supply(water20 %>%
  filter(CALCZCD == "SF") %>%
  select(all_of(water_var)),
  function(x) sum(is.na(x)/length(x)))
```

```
##          TN          TP          TEMP          DO          TURB          COND          VEL
## 0.49246115 0.50078480 0.01157837 0.01390220 0.01440501 0.01510488 0.39597475
##          SS          WDP          CHLcal          SECCHI
## 0.17570717 0.07648925 0.26196058 0.09751921
```

What if CALCZCD is missing?

There are about 8000 samples with missing CALCZCD. However, we will see this is okay because when CALCZCD is missing, nearly all of our 11 continuous variables are missing too. These samples are not particularly useful.

```
(water20 %>%
  filter(CALCZCD == "") %>%
  dim())[1]
```

```
## [1] 8436
```

```
supply((water20 %>%
  filter(CALCZCD == "") %>%
  select(all_of(water_var))), function(x) sum(is.na(x)/length(x)))
```

```
##          TN          TP          TEMP          DO          TURB          COND          VEL          SS
## 0.9998815 0.9998815 0.9992888 0.9992888 0.9996444 0.9992888 0.9997629 0.9996444
##          WDP          CHLcal          SECCHI
## 0.8488620 0.9996444 0.9665718
```

When CALCZCD isn't recorded, the rest of the variables aren't recorded. Since these samples represent 0.041 percent of the original data, we decide that it is okay to exclude these observations.

(Sidenote) Water column variables

What is the difference between WDP, ZMAX, CALCZCD, SAMPZCD? (Could be answered later because not all these variables will be used in our analysis)

Filter for surface level observations

This eliminates most of the duplicates. We will return to the duplicates discussion later.

```
water20 <- water20 %>%
  filter(CALCZCD == "SF") %>%
  select(all_of(c(identifier_var, water_var, waterQF_var)))
```

3. QA/QC filtering

We have 11 continuous water quality variables. They have corresponding QA/QC codes which describe the quality of the data collected. *For the collected data are too compromised, we set their value to NA to interpolate later.* The low quality data values will not be used in our subsequent analysis.

The QA/QC codes that we do not want are as follows:

- 7 of them, TURBQF, TEMPQF, DOQF, VELQF, ZMAXQF, SECCHIQF, and CONDQF, will be filtered out if they are "A" or "0" (in Python, pandas reads each column as characters and integers, so 0 must also be filtered for if using pandas. In R, the entire column is converted to character type).
- 3 of them, TNQF, TPQF, and SSQF, will be filtered out if they are 8 or 64.
- 1 of them, CHLcal, doesn't have a corresponding QF code. TODO

The code works as follows:

1. Define two character vectors, one for which QF codes of "A" and "0" are problematic; one for which QF codes of 8 and 64 are problematic.
2. Create a temporary index `tmp_idx` for each row in the dataset because, as we know, SHEETBARs are not unique identifiers!
3. For the entire water dataset, do the following:
 - a. Define two functions that appropriately set the water variable values to NA. There are two functions because there are two QF code rules ("A" and "0"; 8 and 64). Each function will iterate through pairs of columns (of the variable and corresponding QF code variable) to set the variable column to NA.
 - b. Join the pairs of columns (by `tmp_idx`). This join happens twice, once for each function.
 - c. Join the two dataframes (by `tmp_idx`).
 - d. Join with the entire water dataset to get the `identifier_var`.

```
qf_A0 <- c("TURBQF", "TEMPQF", "DOQF", "VELQF", "ZMAXQF",  
          "SECCHIQF", "CONDQF")  
  
qf_864 <- c("TNQF", "TPQF", "SSQF")  
  
water20$tmp_idx <- 1:nrow(water20)
```

make function

```
qfcodes_setNA <- function(qf_A0, qf_864, water_df,  
                          identifier_var, water_var, waterQF_var){  
  # qf_A0 is a character vector of the variable qf names for which  
  # A, 0 qf codes are bad  
  
  # qf_864 is a character vector of the variable qf names for which  
  # 8, 64 qf codes are bad  
  
  # water_df is the entire water_df  
  
  # last 3 variables are just for naming  
  
  remove_A0 <- function(qf_str, df){  
    # remove the QF at the end  
    var_str <- substr(qf_str, 1, nchar(qf_str)-2)  
  }
```

```

if (var_str == "ZMAX") { var_str <- "WDP"}

df <- df %>%
  # !!sym is for non standard evaluation
  mutate (!!sym(var_str) := case_when (!!sym(qf_str) == "A" ~ NA_real_,
                                       # specify the type of NA correctly
                                       !!sym(qf_str) == "0" ~ NA_real_,
                                       TRUE ~ !!sym(var_str))) %>%
  # other QF code values are fine, keep data
  select(all_of(c("tmp_idx", var_str, qf_str))) # keep sheetbar for joins

return(df)
}

remove_864 <- function(qf_str, df){

  # remove the QF at the end
  var_str <- substr(qf_str, 1, nchar(qf_str)-2)

  df <- df %>%
    mutate (!!sym(var_str) := case_when (!!sym(qf_str) == 8 ~ NA_real_,
                                         # use correct NA type and !!sym for NSE
                                         !!sym(qf_str) == 64 ~ NA_real_,
                                         TRUE ~ !!sym(var_str))) %>%
    # remaining QF code values are fine
    select(all_of(c("tmp_idx", var_str, qf_str))) # sheetbars for joining
}

remove_A0_df <- bind_cols(lapply(qf_A0, remove_A0, water_df)) %>%
  rename(tmp_idx = `tmp_idx...1`) %>%
  select(!contains("..."))

remove_864_df <- bind_cols(lapply(qf_864, remove_864, water_df)) %>%
  rename(tmp_idx = `tmp_idx...1`) %>%
  select(!contains("..."))

fixedqf_df <- inner_join(remove_864_df, remove_A0_df, by = "tmp_idx") %>%
  inner_join(water_df, by = c("tmp_idx", waterQF_var)) %>%
  # this causes duplicate columns for water_var, which we do want
  # because water_df has the old (wrong) values for water_var
  select(!contains(".y")) %>% # remove the old (wrong) values from water_df
  rename_with(~ gsub(".x", "", .), contains(".x")) %>% # rename
  select(all_of(c(identifier_var, water_var, waterQF_var))) # reorder columns

return(fixedqf_df)
}

qfwater20 <- qfcodes_setNA(qf_A0, qf_864, water20,
                          identifier_var, water_var, waterQF_var)

```

check that qfwater20 is working as we want.

This test function iterates through each variable (and its corresponding QF code):

1. `badqf_length` is the number of rows with bad qf codes
2. `badqf_sum` is the sum of NA values among the bad qf codes
3. We want `badqf_length == badqf_sum`.

```
qfcodes_check <- function(qf_str, df, badqf){

  var_str <- substr(qf_str, 1, nchar(qf_str)-2)
  if (var_str == "ZMAX") {var_str = "WDP"}

  badqf_length <- df %>%
    filter(!sym(qf_str) == badqf[1] | !sym(qf_str) == badqf[2]) %>%
    pull(!sym(var_str)) %>%
    length()

  badqf_sum <- df %>%
    filter(!sym(qf_str) == badqf[1] | !sym(qf_str) == badqf[2]) %>%
    pull(!sym(var_str)) %>%
    is.na(.) %>%
    sum()

  if (badqf_length != badqf_sum) {return(qf_str)}
  return(badqf_length == badqf_sum)
}

c(unlist(lapply(qf_A0, qfcodes_check, qfwater20, c("A", "O"))),
  unlist(lapply(qf_864, qfcodes_check, qfwater20, c(8, 64))))

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE

Yes!!! It works!!

# write.csv(filterwater, "../LTRM data/water_data_filtered.csv", row.names = FALSE)
```