Lab 4: Data Wrangling

Environmental Data Analytics | John Fay and Luana Lima | Developed by Kateri Salk

Spring 2023

Objectives

1- Answer questions on M3/A3 2- Answer questions on M4 3- Practice wrangling datasets with dplyr functions

Set up your session

Today we will work with a dataset from the North Temperate Lakes Long-Term Ecological Research Station. The NTL-LTER is located in the boreal zone in northern Wisconsin, USA. We will use the chemical and physical limnology dataset, running from 1984-2016.

Opening discussion: why might we be interested in long-term observations of temperature, oxygen, and light in lakes?

> Add notes here:

```
#Install packages
library(tidyverse)
library(lubridate)
library(here) #The here package allows for better control of relative paths

#Ensure that "here" points to your project folder
here()
```

```
## [1] "/home/guest/EDA/EDA-Spring2023"
```

```
#Read in the data
NTL.phys.data <- read.csv(
  file=here("Data/Raw/NTL-LTER_Lake_ChemistryPhysics_Raw.csv"),
  stringsAsFactors = TRUE
)

#Show the datatype of the 'sampledate' column
str(NTL.phys.data$sampledate)
```

```
##  Factor w/ 1712 levels "10/1/07","10/1/93",..: 134 134 134 134 134 134 134 134 134 134 ...
```

```
#Alternatively, use the tidyverse/dplyr "glimpse" function
glimpse(NTL.phys.data$sampledate)
```

```
##  Factor w/ 1712 levels "10/1/07","10/1/93",..: 134 134 134 134 134 134 134 134 134 134 ...
```

```
# Change sampledate values into date objects
NTL.phys.data$sampledate <- mdy(NTL.phys.data$sampledate)
```

Filter

Filtering allows us to choose certain rows (observations) in our dataset.

```
# note the data types of these two columns
class(NTL.phys.data$lakeid)
```

```
## [1] "factor"
```

```
class(NTL.phys.data$depth)
```

```
## [1] "numeric"
```

```
# dplyr filtering
NTL.phys.data.surface <- filter(NTL.phys.data, depth == 0)

# Choose multiple conditions to filter
summary(NTL.phys.data$lakename)
```

```
## Central Long Lake     Crampton Lake    East Long Lake  Hummingbird Lake 
##               539              1234              3905               430 
##         Paul Lake        Peter Lake      Tuesday Lake         Ward Lake 
##             10325             11288              6107               598 
##    West Long Lake 
##              4188
```

```
NTL.phys.data.PeterPaul <-
  filter(NTL.phys.data, lakename %in% c("Paul Lake", "Peter Lake"))

# Choose a range of conditions of a numeric or integer variable
summary(NTL.phys.data$daynum)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    55.0   166.0   194.0   194.3   222.0   307.0
```

```
NTL.phys.data.JunethruOctober <- filter(NTL.phys.data, daynum %in% c(152:304))
```

```
# Exercise 1:
# filter NTL.phys.data for the year 1999
#we have year that will tell you what year it will be recorded, so you can choose how to filter, she wi
# what code do you need to use, based on the class of the variable?
NTL.phys.data1999 <- filter(NTL.phys.data, year4 == 1999)
```

```
#1898 rows

# Exercise 2:
# filter NTL.phys.data for Tuesday Lake from 1990 through 1999.
#you also want to filter 1990 through 1999 and these belong to different columns in your dataset
NTL.phys.data1990 <- filter(NTL.phys.data, lakename=="Tuesday Lake" & year4 >= 1990 &
                              year4 <=1999)
#1971 rows
```

Question: Why don't we filter using row numbers?

Answer:

Pipes

Pipe is another method to wrangle datasets that looks cleaner and is easier to read. We designate a pipe with %>%. A good way to think about the function of a pipe is with the word "then."

Let's say we want to take our raw dataset (NTL.phys.data), *then* filter the data for Peter and Paul lakes, *then* select temperature and observation information, and *then* add a column for temperature in Fahrenheit:

```
#Example using pipes to wrangle data
#if we are doing things within a pipe, then you dont need to use the first R command because function f
#use a pipe to do the same thing as above, but you want July only

NTL.phys.data.processed <-
  NTL.phys.data %>%
  filter(lakename == "Paul Lake" | lakename == "Peter Lake") %>%
  select(lakename, sampledate:temperature_C) %>%
  mutate(temperature_F = (temperature_C*9/5) + 32)

#Exercise 3: Using a pipe filter NTL.phys.data for Tuesday Lake from 1990
# through 1999 only for July.
#since you are in the pipe you dont need the name of the file again
#you can also mutate it
#command shift M is a shortcut for the pipe symbol
#the summary function just gives you summary stats, but summarize will perform a specific one

NTL.phys.data.Ex3 <-
  NTL.phys.data %>%
  filter(lakename == "Tuesday Lake" & year4 >= 1990 &
           year4 <= 1999 & month (sampledate) == 7)

#537 observations


#Exercise 4: Using the data from part 3, a pipe, and the summarize() function
# find the mean surface temperature (hints: you will need to add another
# filter for depth==0). Make sure you eliminate NAs before computing the means.
```

```
NTL.phys.data.Ex4 <-
  NTL.phys.data.Ex3 %>%
  filter(depth ==0) %>%
  drop_na(temperature_C) %>%
  summarize(mean_temp <- mean(temperature_C))

#how do you see what the mean is?
```

Gather and Spread

For gather we will use `pivot_longer` and for spread we will use `pivot_wider`.

```
#Exercise 5: Gather irradiance data (measured in the water column and measured
#  on the deck of the sampling boat) into one column using pivot_longer. Name
#  the new column holding the irradiance type as "Irradiance_Type", and name the
#  new column holding the irradiance values as "Irradiance_Value".
#you need to provide the column names that you want to transform into one
NTL.phys.data.Ex5 <-
  NTL.phys.data %>%
  pivot_longer(irradianceWater:irradianceDeck,
             names_to = "irradiance",
             values_to = "radiation") #you can name it whatever, but you need to pick a new name
#Exercise 6: Spread temperatureC into more than one column based on the depth.
#you want to make it wider, so when you create a wider data frame, each element in the depth will becom
#31k observations
#you are generating a lot of NAs

NTL.phys.data.Ex6 <-
  NTL.phys.data %>%
  pivot_wider(names_from = "depth",
            values_from = "temperature_C")
```