Lab 4: Data Wrangling

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

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] "C:/Workspace/Teaching_F24/EDE_base"
```

```
#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. - The 1st parameter if the filter command is the dataframe we wish to filter. - The 2nd on is the **filter expression**: - `depth == 0` keeps rows with depth equal to zero (surface) - `lakename %in% c("Paul Lake", "Peter Lake")` keeps Paul & Peter lake rows - `daynum %in% c(152:304)` keeps rows with `daynum` values between 152 and 304

Enter these filter expressions below

```
# 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))
```

```r
# Exercise 1:
# filter NTL.phys.data for the year 1999
# what code do you need to use, based on the class of the variable?
Ex1 <- filter(NTL.phys.data, year4 == 1999)

# Exercise 2:
# filter NTL.phys.data for Tuesday Lake from 1990 through 1999.
Ex2 <- filter(
  NTL.phys.data,
  (lakename == 'Tuesday Lake') & (year4 %in% c(1990:1999))
)
```

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:

```r
#Example using pipes to wrangle data:
#Add pipes in the correct place below
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.
Ex3 <- NTL.phys.data %>%
  filter(lakename == 'Tuesday Lake')  %>%
  filter(year4 %in% c(1990:1999)) %>%
  filter(month(sampledate) == 7)

#Exercise 4: Using the data from part 3, a pipe, and the summarize() function,
# find the mean surface water temperature.
# (hint: you will need to filter for depth==0.25).
Ex4 <- Ex3 %>%
  filter(depth == 0) %>%
  drop_na(temperature_C) %>%
  summarize(
    mean_surface = mean(temperature_C)
  )
Ex4
```

```
##   mean_surface
## 1        22.46
```

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".
Ex5 <- NTL.phys.data %>%
  pivot_longer(
    cols = c(irradianceWater,irradianceDeck),
    names_to = "irradiance_type",
    values_to = "irradiance"
  )

#Exercise 6: Spread temperatureC into more than one column based on the depth.
Ex6 <- NTL.phys.data %>%
  pivot_wider(
    id_cols = sampledate,
    names_from = depth,
    values_from = temperature_C
  )
```

```
## Warning: Values from ‘temperature_C‘ are not uniquely identified; output will contain
## list-cols.
## * Use ‘values_fn = list‘ to suppress this warning.
## * Use ‘values_fn = {summary_fun}‘ to summarise duplicates.
## * Use the following dplyr code to identify duplicates.
##   {data} %>%
##   dplyr::group_by(sampledate, depth) %>%
##   dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
##   dplyr::filter(n > 1L)
```

```
# or
Ex6 <- NTL.phys.data %>%
  select(1:7) %>%  #To remove other measurement variables
  pivot_wider(
    names_from = depth,
    values_from = temperature_C
  )
```