4: Part 1 - Data Wrangling

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

Spring 2023

Objectives

1. Describe the usefulness of data wrangling and its place in the data pipeline
2. Wrangle datasets with dplyr functions
3. Apply data wrangling skills to a real-world example dataset

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:

```
getwd()
```

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

```
#install.packages(tidyverse)
library(tidyverse)
#install.packages(lubridate)
library(lubridate)
NTL.phys.data <- read.csv("./Data/Raw/NTL-LTER_Lake_ChemistryPhysics_Raw.csv", stringsAsFactors = TRUE)

colnames(NTL.phys.data)
```

```
##  [1] "lakeid"          "lakename"        "year4"           "daynum"
##  [5] "sampledate"      "depth"           "temperature_C"   "dissolvedOxygen"
##  [9] "irradianceWater" "irradianceDeck"  "comments"
```

```
head(NTL.phys.data)
```

```
##   lakeid  lakename year4 daynum sampledate depth temperature_C dissolvedOxygen
## 1      L Paul Lake  1984    148    5/27/84  0.00          14.5             9.5
## 2      L Paul Lake  1984    148    5/27/84  0.25            NA              NA
```

```
## 3      L Paul Lake  1984    148    5/27/84  0.50            NA              NA
## 4      L Paul Lake  1984    148    5/27/84  0.75            NA              NA
## 5      L Paul Lake  1984    148    5/27/84  1.00          14.5             8.8
## 6      L Paul Lake  1984    148    5/27/84  1.50            NA              NA
##   irradianceWater irradianceDeck comments
## 1            1750           1620     <NA>
## 2            1550           1620     <NA>
## 3            1150           1620     <NA>
## 4             975           1620     <NA>
## 5             870           1620     <NA>
## 6             610           1620     <NA>
```

```
summary(NTL.phys.data)
```

```
##      lakeid                lakename         year4          daynum
##  R      :11288   Peter Lake    :11288   Min.   :1984   Min.   : 55.0
##  L      :10325   Paul Lake     :10325   1st Qu.:1991   1st Qu.:166.0
##  T      : 6107   Tuesday Lake  : 6107   Median :1997   Median :194.0
##  W      : 4188   West Long Lake: 4188   Mean   :1999   Mean   :194.3
##  E      : 3905   East Long Lake: 3905   3rd Qu.:2006   3rd Qu.:222.0
##  M      : 1234   Crampton Lake : 1234   Max.   :2016   Max.   :307.0
##  (Other): 1567   (Other)       : 1567
##    sampledate        depth       temperature_C   dissolvedOxygen
##  5/17/94:   84   Min.   : 0.00   Min.   : 0.30   Min.   :  0.00
##  9/5/90 :   64   1st Qu.: 1.50   1st Qu.: 5.30   1st Qu.:  0.30
##  10/1/07:   61   Median : 4.00   Median : 9.30   Median :  5.60
##  9/10/90:   61   Mean   : 4.39   Mean   :11.81   Mean   :  4.97
##  5/10/87:   60   3rd Qu.: 6.50   3rd Qu.:18.70   3rd Qu.:  8.40
##  5/9/88 :   60   Max.   :20.00   Max.   :34.10   Max.   :802.00
##  (Other):38224                   NA's   :3858    NA's   :4039
##  irradianceWater     irradianceDeck                                  comments
##  Min.   :   -0.337   Min.   :   1.5   DO Probe bad - Doesn't go to zero:  206
##  1st Qu.:   14.000   1st Qu.: 353.0   DO taken with Jones Lab Meter    :  162
##  Median :   65.000   Median : 747.0   NA's                             :38246
##  Mean   :  210.242   Mean   : 720.5
##  3rd Qu.:  265.000   3rd Qu.:1042.0
##  Max.   :24108.000   Max.   :8532.0
##  NA's   :14287       NA's   :15419
```

```
str(NTL.phys.data)
```

```
## 'data.frame':    38614 obs. of  11 variables:
##  $ lakeid         : Factor w/ 9 levels "C","E","H","L",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ lakename       : Factor w/ 9 levels "Central Long Lake",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ year4          : int  1984 1984 1984 1984 1984 1984 1984 1984 1984 1984 ...
##  $ daynum         : int  148 148 148 148 148 148 148 148 148 148 ...
##  $ sampledate     : Factor w/ 1712 levels "10/1/07","10/1/93",..: 134 134 134 134 134 134 134 134 134
##  $ depth          : num  0 0.25 0.5 0.75 1 1.5 2 3 4 5 ...
##  $ temperature_C  : num  14.5 NA NA NA 14.5 NA 14.2 11 7 6.1 ...
##  $ dissolvedOxygen: num  9.5 NA NA NA 8.8 NA 8.6 11.5 11.9 2.5 ...
##  $ irradianceWater: num  1750 1550 1150 975 870 610 420 220 100 34 ...
##  $ irradianceDeck : num  1620 1620 1620 1620 1620 1620 1620 1620 1620 1620 ...
##  $ comments       : Factor w/ 2 levels "DO Probe bad - Doesn't go to zero",..: NA NA NA NA NA NA NA
```

```
dim(NTL.phys.data)
```

```
## [1] 38614    11
```

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

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

```
# Format sampledate as date
NTL.phys.data$sampledate <- as.Date(NTL.phys.data$sampledate, format = "%m/%d/%y")
```

Data Wrangling

Data wrangling extends data exploration: it allows you to process data in ways that are useful for you. An important part of data wrangling is creating *tidy datasets*, with the following rules:

1. Each variable has its own column
2. Each observation has its own row
3. Each value has its own cell

What is the best way to wrangle data? There are multiple ways to arrive at a specific outcome in R, and we will illustrate some of those approaches. Your goal should be to write the simplest code that will get you to your desired outcome. However, there is sometimes a trade-off of the opportunity cost to learn a new formulation of code and the time it takes to write complex code that you already know. Remember that the best code is one that is easy to understand for yourself and your collaborators. Remember to comment your code, use informative names for variables and functions, and use reproducible methods to arrive at your output.

Dplyr Wrangling Functions

`dplyr` is a package in R that includes functions for data manipulation (i.e., data wrangling or data munging). `dplyr` is included in the tidyverse package, so you should already have it installed on your machine. The functions act as verbs for data wrangling processes. For more information, run this line of code:

```
vignette("dplyr")
```

```
## starting httpd help server ... done
```

Filter

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

Here are the relevant commands used in the `filter` function. Add some notes to designate what these commands mean. `==` filter comparing a value to a specific value `!=` compare if values are different `<` `<=` `>` `>=` `&` checking multiple conditions `|`checking if one or the other is true

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

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

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

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

```
# matrix filtering
NTL.phys.data.surface1 <- NTL.phys.data[NTL.phys.data$depth == 0,]

# dplyr filtering
NTL.phys.data.surface2 <- filter(NTL.phys.data, depth == 0)
NTL.phys.data.surface3 <- filter(NTL.phys.data, depth < 0.25)

# Did the methods arrive at the same result?
head(NTL.phys.data.surface1)
```

```
##    lakeid     lakename year4 daynum sampledate depth temperature_C
## 1       L    Paul Lake  1984    148 1984-05-27     0          14.5
## 18      R   Peter Lake  1984    149 1984-05-28     0          14.8
## 40      T Tuesday Lake  1984    150 1984-05-29     0          15.0
## 56      L    Paul Lake  1984    155 1984-06-03     0          18.8
## 72      R   Peter Lake  1984    156 1984-06-04     0          18.8
## 90      T Tuesday Lake  1984    157 1984-06-05     0          21.0
##    dissolvedOxygen irradianceWater irradianceDeck comments
## 1              9.5            1750           1620     <NA>
## 18             9.2            1630           1540     <NA>
## 40             9.5            1850           1960     <NA>
## 56             8.0            1100           1050     <NA>
## 72             9.0             275            275     <NA>
## 90             8.4            1200           1200     <NA>
```

```
dim(NTL.phys.data.surface1)
```

```
## [1] 1902   11
```

```
head(NTL.phys.data.surface2)
```

```
##   lakeid     lakename year4 daynum sampledate depth temperature_C
## 1      L    Paul Lake  1984    148 1984-05-27     0          14.5
## 2      R   Peter Lake  1984    149 1984-05-28     0          14.8
## 3      T Tuesday Lake  1984    150 1984-05-29     0          15.0
## 4      L    Paul Lake  1984    155 1984-06-03     0          18.8
## 5      R   Peter Lake  1984    156 1984-06-04     0          18.8
## 6      T Tuesday Lake  1984    157 1984-06-05     0          21.0
##   dissolvedOxygen irradianceWater irradianceDeck comments
## 1             9.5            1750           1620     <NA>
## 2             9.2            1630           1540     <NA>
## 3             9.5            1850           1960     <NA>
## 4             8.0            1100           1050     <NA>
## 5             9.0             275            275     <NA>
## 6             8.4            1200           1200     <NA>
```

```
dim(NTL.phys.data.surface2)
```

```
## [1] 1902   11
```

```
head(NTL.phys.data.surface3)
```

```
##   lakeid     lakename year4 daynum sampledate depth temperature_C
## 1      L    Paul Lake  1984    148 1984-05-27     0          14.5
## 2      R   Peter Lake  1984    149 1984-05-28     0          14.8
## 3      T Tuesday Lake  1984    150 1984-05-29     0          15.0
## 4      L    Paul Lake  1984    155 1984-06-03     0          18.8
## 5      R   Peter Lake  1984    156 1984-06-04     0          18.8
## 6      T Tuesday Lake  1984    157 1984-06-05     0          21.0
##   dissolvedOxygen irradianceWater irradianceDeck comments
## 1             9.5            1750           1620     <NA>
## 2             9.2            1630           1540     <NA>
## 3             9.5            1850           1960     <NA>
## 4             8.0            1100           1050     <NA>
## 5             9.0             275            275     <NA>
## 6             8.4            1200           1200     <NA>
```

```
dim(NTL.phys.data.surface3)
```

```
## [1] 1902   11
```

```
# 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.PeterPaul1 <- filter(NTL.phys.data, lakename == "Paul Lake" | lakename == "Peter Lake")
NTL.phys.data.PeterPaul2 <- filter(NTL.phys.data, lakename != "Central Long Lake" &
                                     lakename != "Crampton Lake" & lakename != "East Long Lake" &
                                     lakename != "Hummingbird Lake" & lakename != "Tuesday Lake" &
                                     lakename != "Ward Lake" & lakename != "West Long Lake")
NTL.phys.data.PeterPaul3 <- 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.JunethruOctober1 <- filter(NTL.phys.data, daynum > 151 & daynum < 305)
NTL.phys.data.JunethruOctober2 <- filter(NTL.phys.data, daynum > 151, daynum < 305)
NTL.phys.data.JunethruOctober3 <- filter(NTL.phys.data, daynum >= 152 & daynum <= 304)
NTL.phys.data.JunethruOctober4 <- filter(NTL.phys.data, daynum %in% c(152:304))

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

```
## [1] "integer"
```

```
# Exercise:
# filter NTL.phys.data for Tuesday Lake from 1990 through 1999.
```

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

Answer:

Arrange

Arranging allows us to change the order of rows in our dataset. By default, the arrange function will arrange rows in ascending order.

```
NTL.phys.data.depth.ascending <- arrange(NTL.phys.data, depth)
NTL.phys.data.depth.descending <- arrange(NTL.phys.data, desc(depth))

# Exercise:
# Arrange NTL.phys.data by temperature, in descending order.
# Which dates, lakes, and depths have the highest temperatures?
```

Select

Selecting allows us to choose certain columns (variables) in our dataset.

```
NTL.phys.data.temps <- select(NTL.phys.data, lakename, sampledate:temperature_C)
```

Mutate

Mutating allows us to add new columns that are functions of existing columns. Operations include addition, subtraction, multiplication, division, log, and other functions.

```
NTL.phys.data.temps <- mutate(NTL.phys.data.temps, temperature_F = (temperature_C*9/5) + 32)
```

Lubridate

A package that makes coercing date much easier is `lubridate`. A guide to the package can be found at https://lubridate.tidyverse.org/. The cheat sheet within that web page is excellent too. This package can do many things (hint: look into this package if you are having unique date-type issues), but today we will be using two of its functions for our NTL dataset.

```
# add a month column to the dataset
NTL.phys.data.PeterPaul1 <- mutate(NTL.phys.data.PeterPaul1, month = month(sampledate))

# reorder columns to put month with the rest of the date variables
NTL.phys.data.PeterPaul1 <- select(NTL.phys.data.PeterPaul1, lakeid:daynum, month, sampledate:comments)

# find out the start and end dates of the dataset
interval(NTL.phys.data.PeterPaul1$sampledate[1], NTL.phys.data.PeterPaul1$sampledate[21613])
```

```
## [1] 1984-05-27 UTC--2016-08-16 UTC
```

```
interval(first(NTL.phys.data.PeterPaul1$sampledate), last(NTL.phys.data.PeterPaul1$sampledate))
```

```
## [1] 1984-05-27 UTC--2016-08-16 UTC
```

Pipes

Sometimes we will want to perform multiple functions on a single dataset on our way to creating a processed dataset. We could do this in a series of subsequent functions or create a custom function. However, there is another method to do this that looks cleaner and is easier to read. This method is called a pipe. 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:

```
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)
```

Notice that we did not place the dataset name inside the wrangling function but rather at the beginning.

Saving processed datasets

```
write.csv(NTL.phys.data.PeterPaul1, row.names = FALSE, file = "./Data/Processed/NTL-LTER_Lake_Chemistry
```

Closing Discussion

When we wrangle a raw dataset into a processed dataset, we create a code file that contains only the wrangling code. We then save the processed dataset as a new spreadsheet and then create a separate code file to analyze and visualize the dataset. Why do we keep the wrangling code separate from the analysis code?