7: Data Wrangling

Environmental Data Analytics | Kateri Salk

Spring 2020

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

```
getwd()
```

```
## [1] "/Users/ks501/Documents/GithubRepos/Environmental_Data_Analytics_2020"
```

```
library(plyr)
library(tidyverse)
library(lubridate)
NTL.phys.data.PeterPaul <- read.csv("./Data/Processed/NTL-LTER_Lake_ChemistryPhysics_PeterPaul_Processed
NTL.nutrient.data <- read.csv("./Data/Raw/NTL-LTER_Lake_Nutrients_Raw.csv")
```

Review of basic exploration and wrangling

```
# Data summaries for physical data
colnames(NTL.phys.data.PeterPaul)
```

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

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

```
## [1] 21613    12
```

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

```
## 'data.frame':    21613 obs. of  12 variables:
##  $ lakeid        : Factor w/ 2 levels "L","R": 1 1 1 1 1 1 1 1 1 1 ...
##  $ lakename      : Factor w/ 2 levels "Paul Lake","Peter Lake": 1 1 1 1 1 1 1 1 1 1 ...
##  $ year4         : int  1984 1984 1984 1984 1984 1984 1984 1984 1984 1984 ...
##  $ daynum        : int  148 148 148 148 148 148 148 148 148 148 ...
##  $ month         : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ sampledate    : Factor w/ 967 levels "1984-05-27","1984-05-28",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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
```

```
summary(NTL.phys.data.PeterPaul$comments)
```

```
## DO Probe bad - Doesn't go to zero     DO taken with Jones Lab Meter
##                               132                               112
##                              NA's
##                             21369
```

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

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

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

# Select Peter and Paul Lakes from the nutrient dataset
NTL.nutrient.data.PeterPaul <- filter(NTL.nutrient.data, lakename == "Paul Lake" | lakename == "Peter La

# Data summaries for nutrient data
colnames(NTL.nutrient.data.PeterPaul)
```

```
##  [1] "lakeid"     "lakename"   "year4"      "daynum"     "sampledate"
##  [6] "depth_id"   "depth"      "tn_ug"      "tp_ug"      "nh34"
## [11] "no23"       "po4"        "comments"
```

```
dim(NTL.nutrient.data.PeterPaul)
```

```
## [1] 2770   13
```

```
str(NTL.nutrient.data.PeterPaul)
```

```
## 'data.frame':    2770 obs. of  13 variables:
##  $ lakeid    : Factor w/ 26 levels "B","Berg","Bolg",..: 13 13 13 13 13 13 18 18 18 18 ...
##  $ lakename  : Factor w/ 26 levels "Bergner Lake",..: 16 16 16 16 16 16 17 17 17 17 ...
##  $ year4     : int  1991 1991 1991 1991 1991 1991 1991 1991 1991 1991 ...
##  $ daynum    : int  140 140 140 140 140 140 140 140 140 140 ...
##  $ sampledate: Factor w/ 1294 levels "4/14/12","4/19/12",..: 31 31 31 31 31 31 31 31 31 31 ...
##  $ depth_id  : int  1 2 3 4 5 6 1 2 3 4 ...
##  $ depth     : num  0 0.85 1.75 3 4 6 0 1 2.25 3.5 ...
##  $ tn_ug     : num  538 285 399 453 363 583 352 356 364 582 ...
##  $ tp_ug     : num  25 14 14 14 13 37 11 15 28 14 ...
##  $ nh34      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ no23      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ po4       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ comments  : Factor w/ 3 levels "","sample missing",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(NTL.nutrient.data.PeterPaul$lakename)
```

```
##       Bergner Lake            Bog Pot         Bolger Bog         Brown Lake
##                  0                  0                  0                  0
##  Central Long Lake      Crampton Lake      Cranberry Bog     East Long Lake
##                  0                  0                  0                  0
##            Eds Bog Forest Service Bog   Hummingbird Lake        Inkpot Lake
##                  0                  0                  0                  0
##      Kickapoo Lake        Morris Lake     North Gate Bog          Paul Lake
```

```
##                  0                  0                  0               1383
##         Peter Lake          Plum Lake     Raspberry Lake    Reddington Lake
##               1387                  0                  0                  0
##         Roach Lake         Tender Bog    Tenderfoot Lake       Tuesday Lake
##                  0                  0                  0                  0
##          Ward Lake     West Long Lake
##                  0                  0
```

```
# Notice that other lake names didn't go away, even though they have zero values
NTL.nutrient.data.PeterPaul <- droplevels(NTL.nutrient.data.PeterPaul)
summary(NTL.nutrient.data.PeterPaul$lakename)
```

```
##  Paul Lake Peter Lake
##       1383       1387
```

```
summary(NTL.nutrient.data.PeterPaul$comments)
```

```
##
## 2770
```

```
class(NTL.nutrient.data.PeterPaul$sampledate)
```

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

```
NTL.nutrient.data.PeterPaul$sampledate <- as.Date(NTL.nutrient.data.PeterPaul$sampledate, format = "%m/%

NTL.nutrient.data.PeterPaul <-
  NTL.nutrient.data.PeterPaul %>% #
  mutate(month = month(sampledate)) %>% #
  select(lakeid:daynum, month, sampledate:comments) %>% #
  drop_na(depth)

# Save processed nutrient file
write.csv(NTL.nutrient.data.PeterPaul, row.names = FALSE,
          file = "./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaul_Processed.csv")

# Remove columns that are not of interest for analysis
NTL.phys.data.PeterPaul.subset <- select(NTL.phys.data.PeterPaul,
                                         lakename:irradianceDeck)

NTL.nutrient.data.PeterPaul.subset <- select(NTL.nutrient.data.PeterPaul,
                                             lakename, year4, daynum, month, sampledate, depth:po4)

# write a more succinct line of code to subset the nutrient dataset.
```

Gather and Spread

For most situations, data analysis works best when you have organized your data into a tidy dataset. A tidy dataset is defined as: * Each variable is a column * Each row is an observation * Each value is in its own cell

However, there may be situations where we want to reshape our dataset, for example if we want to facet numerical data points by measurement type (more on this in the data visualization unit). We can program this reshaping in a few short lines of code using the package `tidyr`, which is conveniently included in the `tidyverse` package.

Note: `tidyr` is moving away from `gather` and `spread` and toward `pivot_longer` and `pivot_wider`, respectively. Note that the latter functions are only available on the newest version of `tidyr`, so we are using

`spread` and `gather` today to ensure compatibility. `gather` and `spread` are not going away, but they are not under active development.

```
# Gather nutrient data into one column
NTL.nutrient.data.PeterPaul.gathered <- gather(NTL.nutrient.data.PeterPaul.subset, "nutrient", "concentr
NTL.nutrient.data.PeterPaul.gathered <- subset(NTL.nutrient.data.PeterPaul.gathered, !is.na(concentrati
count(NTL.nutrient.data.PeterPaul.gathered, nutrient)
```

```
## # A tibble: 5 x 2
##   nutrient     n
##   <chr>    <int>
## 1 nh34      1130
## 2 no23      1161
## 3 po4       1186
## 4 tn_ug     1425
## 5 tp_ug     2279
```

```
write.csv(NTL.nutrient.data.PeterPaul.gathered, row.names = FALSE,
          file = "./Data/Processed/NTL-LTER_Lake_Nutrients_PeterPaulGathered_Processed.csv")

# Spread nutrient data into separate columns
NTL.nutrient.data.PeterPaul.spread <- spread(NTL.nutrient.data.PeterPaul.gathered, nutrient, concentrati

# Split components of cells into multiple columns
# Opposite of 'separate' is 'unite'
NTL.nutrient.data.PeterPaul.dates <- separate(NTL.nutrient.data.PeterPaul.subset, sampledate, c("Y", "m

# I recommend using lubridate rather than separate and unite.
```

Combining multiple datasets

Join

In many cases, we will want to combine datasets into one dataset. If all column names match, the data frames can be combined with the `rbind` function. If some column names match and some column names don't match, we can combine the data frames using a "join" function according to common conditions that exist in the matching columns. We will demonstrate this with the NTL-LTER physical and nutrient datasets, where we have specific instances when physical and nutrient data were collected on the same date, at the same lake, and at the same depth.

In dplyr, there are several types of join functions:

- `inner_join`: return rows in x where there are matching values in y, and all columns in x and y (mutating join).
- `semi_join`: return all rows from x where there are matching values in y, keeping just columns from x (filtering join).
- `left_join`: return all rows from x, and all columns from x and y (mutating join).
- `anti_join`: return all rows from x where there are *not* matching values in y, keeping just columns from x (filtering join).
- `full_join`: return all rows and all columns from x and y. Returns NA for missing values (mutating join).

Let's say we want to generate a new dataset that contains all possible physical and chemical data for Peter and Paul Lakes. In this case, we want to do a full join.

```
NTL.phys.nutrient.data.PeterPaul <- full_join(NTL.phys.data.PeterPaul.subset, NTL.nutrient.data.PeterPau
```

```
## Joining, by = c("lakename", "year4", "daynum", "month", "sampledate", "depth")
```

```
write.csv(NTL.phys.nutrient.data.PeterPaul, row.names = FALSE,
          file ="./Data/Processed/NTL-LTER_Lake_Chemistry_Nutrients_PeterPaul_Processed.csv")
```

rbind

The Niwot Ridge litter dataset, when downloaded from NEON, comes packaged with each month as a different .csv file. If we want to analyze the dataset as a single data frame, we need to combine each of these files.

```
Litter.June2016 <- read.csv("./Data/Raw/NIWO_Litter/NEON_NIWO_Litter_massdata_2016-06_raw.csv")
Litter.July2016 <- read.csv("./Data/Raw/NIWO_Litter/NEON_NIWO_Litter_massdata_2016-07_raw.csv")
Litter.August2016 <- read.csv("./Data/Raw/NIWO_Litter/NEON_NIWO_Litter_massdata_2016-08_raw.csv")

Litter.2019 <- rbind(Litter.June2016, Litter.July2016, Litter.August2016)
```

However, there are 20 months in this dataset, so importing all these files individually would be tedious to code. Here is a more efficient way to import and combine all files.

```
LitterFiles = list.files(path = "./Data/Raw/NIWO_Litter/", pattern="*.csv", full.names=TRUE)
LitterFiles
```

```
##  [1] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-06_raw.csv"
##  [2] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-07_raw.csv"
##  [3] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-08_raw.csv"
##  [4] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-09_raw.csv"
##  [5] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-10_raw.csv"
##  [6] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2016-11_raw.csv"
##  [7] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2017-07_raw.csv"
##  [8] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2017-08_raw.csv"
##  [9] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2017-10_raw.csv"
## [10] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2017-11_raw.csv"
## [11] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-06_raw.csv"
## [12] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-07_raw.csv"
## [13] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-08_raw.csv"
## [14] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-09_raw.csv"
## [15] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-10_raw.csv"
## [16] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2018-11_raw.csv"
## [17] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2019-06_raw.csv"
## [18] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2019-07_raw.csv"
## [19] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2019-08_raw.csv"
## [20] "./Data/Raw/NIWO_Litter//NEON_NIWO_Litter_massdata_2019-09_raw.csv"
```

```
Litter <- LitterFiles %>%
  ldply(read.csv)
```

We also have information about individual traps, including the location and type of landcover. Let's join these two datasets. Note that "siteID", "plotID" and "trapID" exist in both datasets, and we can join them by these conditions. Notice the dimensions of the final dataset.

```
Trap <- read.csv("./Data/Raw/NEON_NIWO_Litter_trapdata_raw.csv")
LitterTrap <- left_join(Litter, Trap, by = c("siteID", "plotID", "trapID"))
```

```
## Warning: Column `plotID` joining factors with different levels, coercing to
```

```
## character vector
```

```
## Warning: Column `trapID` joining factors with different levels, coercing to
## character vector
```

```
dim(Litter)
```

```
## [1] 1692   19
```

```
dim(Trap)
```

```
## [1] 24 23
```

```
dim(LitterTrap)
```

```
## [1] 1692   39
```

```
LitterTrap <- LitterTrap %>%
  select(plotID:trapID, collectDate, functionalGroup:qaDryMass, subplotID:geodeticDatum)

write.csv(LitterTrap, row.names = FALSE,
          file ="./Data/Processed/NEON_NIWO_Litter_mass_trap_Processed.csv")
```

Split-Apply-Combine

dplyr functionality, combined with the pipes operator, allows us to split datasets according to groupings (function: `group_by`), then run operations on those groupings and return the output of those operations. There is a lot of flexibility in this approach, but we will illustrate just one example today.

```
NTL.PeterPaul.summaries <-
  NTL.phys.nutrient.data.PeterPaul %>%
  filter(depth == 0) %>%
  group_by(lakename, month) %>%
  filter(!is.na(temperature_C) & !is.na(tn_ug) & !is.na(tp_ug)) %>%
  summarise(meantemp = mean(temperature_C),
            sdtemp = sd(temperature_C),
            meanTN = mean(tn_ug),
            sdTN = sd(tn_ug),
            meanTP = mean(tp_ug),
            sdTP = sd(tp_ug))

write.csv(NTL.PeterPaul.summaries, row.names = FALSE,
          file ="./Data/Processed/NTL-LTER_Lake_Summaries_PeterPaul_Processed.csv")
```

Alternative Methods for Data Wrangling

If you want to iteratively perform operations on your data, there exist several options. We have demonstrated the pipe as one option. Additional options include the `apply` function (https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/apply) and `for` loops (https://swcarpentry.github.io/r-novice-inflammation/15-supp-loops-in-depth/). These options are good options as well (again, multiple ways to get to the same outcome). A word of caution: loops are slow. This may not make a difference for small datasets, but small time additions will make a difference with large datasets.