Assignment 4: Data Wrangling

Enikoe Bihari

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on Data Wrangling

Directions

1. Change "Student Name" on line 3 (above) with your name.
2. Work through the steps, **creating code and output** that fulfill each instruction.
3. Be sure to **answer the questions** in this assignment document.
4. When you have completed the assignment, **Knit** the text and code into a single PDF file.
5. After Knitting, submit the completed exercise (PDF file) to the dropbox in Sakai. Add your last name into the file name (e.g., "Fay_A04_DataWrangling.Rmd") prior to submission.

The completed exercise is due on Monday, Feb 7 @ 7:00pm.

Set up your session

1. Check your working directory, load the `tidyverse` and `lubridate` packages, and upload all four raw data files associated with the EPA Air dataset. See the README file for the EPA air datasets for more information (especially if you have not worked with air quality data previously).
2. Explore the dimensions, column names, and structure of the datasets.

```
#1
# check working directory
getwd()
```

```
## [1] "C:/Users/eniko/Documents/Duuuuuke/2021-22/Data Analytics/Environmental_Data_Analytics_2022/Assi
```

```
# load packages
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
## Warning: package 'ggplot2' was built under R version 4.0.5
## Warning: package 'tibble' was built under R version 4.0.5
## Warning: package 'tidyr' was built under R version 4.0.5
## Warning: package 'readr' was built under R version 4.0.5
## Warning: package 'dplyr' was built under R version 4.0.5
## Warning: package 'forcats' was built under R version 4.0.5
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```r
library(plyr)

o3.2018 <- read.csv("../Data/Raw/EPAair_O3_NC2018_raw.csv",
                    stringsAsFactors = T)
o3.2019 <- read.csv("../Data/Raw/EPAair_O3_NC2019_raw.csv",
                    stringsAsFactors = T)
pm25.2018 <- read.csv("../Data/Raw/EPAair_PM25_NC2018_raw.csv",
                    stringsAsFactors = T)
pm25.2019 <- read.csv("../Data/Raw/EPAair_PM25_NC2019_raw.csv",
                    stringsAsFactors = T)

#2
# explore the data sets
dim(o3.2018)
```

```
## [1] 9737   20
```

```r
colnames(o3.2018)
```

```
##  [1] "Date"
##  [2] "Source"
##  [3] "Site.ID"
##  [4] "POC"
##  [5] "Daily.Max.8.hour.Ozone.Concentration"
##  [6] "UNITS"
##  [7] "DAILY_AQI_VALUE"
##  [8] "Site.Name"
##  [9] "DAILY_OBS_COUNT"
## [10] "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"
## [12] "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"
## [14] "CBSA_NAME"
## [15] "STATE_CODE"
## [16] "STATE"
## [17] "COUNTY_CODE"
## [18] "COUNTY"
## [19] "SITE_LATITUDE"
## [20] "SITE_LONGITUDE"
```

```r
str(o3.2018)
```

```
## 'data.frame':    9737 obs. of  20 variables:
##  $ Date                                : Factor w/ 364 levels "01/01/2018","01/02/2018",..: 60 61 62
##  $ Source                              : Factor w/ 1 level "AQS": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Site.ID                             : int  370030005 370030005 370030005 370030005 370030005 3700
##  $ POC                                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Daily.Max.8.hour.Ozone.Concentration: num  0.043 0.046 0.047 0.049 0.047 0.03 0.036 0.044 0.049 0
##  $ UNITS                               : Factor w/ 1 level "ppm": 1 1 1 1 1 1 1 1 1 1 ...
##  $ DAILY_AQI_VALUE                     : int  40 43 44 45 44 28 33 41 45 40 ...
##  $ Site.Name                           : Factor w/ 40 levels "","Beaufort",..: 35 35 35 35 35 35 35 3
##  $ DAILY_OBS_COUNT                     : int  17 17 17 17 17 17 17 17 17 17 ...
##  $ PERCENT_COMPLETE                    : num  100 100 100 100 100 100 100 100 100 100 ...
##  $ AQS_PARAMETER_CODE                  : int  44201 44201 44201 44201 44201 44201 44201 44201 44201 4
##  $ AQS_PARAMETER_DESC                  : Factor w/ 1 level "Ozone": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ CBSA_CODE                          : int  25860 25860 25860 25860 25860 25860 25860 25860 25860 2
##  $ CBSA_NAME                          : Factor w/ 17 levels "","Asheville, NC",..: 9 9 9 9 9 9 9 9 9
##  $ STATE_CODE                         : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ STATE                              : Factor w/ 1 level "North Carolina": 1 1 1 1 1 1 1 1 1 1 ...
##  $ COUNTY_CODE                        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ COUNTY                             : Factor w/ 32 levels "Alexander","Avery",..: 1 1 1 1 1 1 1 1
##  $ SITE_LATITUDE                      : num  35.9 35.9 35.9 35.9 35.9 ...
##  $ SITE_LONGITUDE                     : num  -81.2 -81.2 -81.2 -81.2 -81.2 ...
```

Wrangle individual datasets to create processed files.

3. Change date to a date object
4. Select the following columns: Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE
5. For the PM2.5 datasets, fill all cells in AQS_PARAMETER_DESC with "PM2.5" (all cells in this column should be identical).
6. Save all four processed datasets in the Processed folder. Use the same file names as the raw files but replace "raw" with "processed".

```
#3
# change the date to date object
o3.2018$Date <- as.Date(o3.2018$Date,
                        format = "%m/%d/%Y")
o3.2019$Date <- as.Date(o3.2019$Date,
                        format = "%m/%d/%Y")
pm25.2018$Date <- as.Date(pm25.2018$Date,
                          format = "%m/%d/%Y")
pm25.2019$Date <- as.Date(pm25.2019$Date,
                          format = "%m/%d/%Y")
class(o3.2018$Date)
```

```
## [1] "Date"
```

```
#4
# select necessary columns
o3.2018.subset <- select(o3.2018, Date,
                         DAILY_AQI_VALUE,
                         Site.Name,
                         AQS_PARAMETER_DESC,
                         COUNTY,
                         SITE_LATITUDE,
                         SITE_LONGITUDE)

o3.2019.subset <- select(o3.2019,
                         Date,
                         DAILY_AQI_VALUE,
                         Site.Name,
                         AQS_PARAMETER_DESC,
                         COUNTY,
                         SITE_LATITUDE,
                         SITE_LONGITUDE)

pm25.2018.subset <- select(pm25.2018,
                           Date,
                           DAILY_AQI_VALUE,
```

```
                            Site.Name,
                            AQS_PARAMETER_DESC,
                            COUNTY,
                            SITE_LATITUDE,
                            SITE_LONGITUDE)

pm25.2019.subset <- select(pm25.2019,
                            Date,
                            DAILY_AQI_VALUE,
                            Site.Name,
                            AQS_PARAMETER_DESC,
                            COUNTY,
                            SITE_LATITUDE,
                            SITE_LONGITUDE)

#5
# fill in cells
pm25.2019.subset$AQS_PARAMETER_DESC = 'PM2.5'
pm25.2018.subset$AQS_PARAMETER_DESC = 'PM2.5'

#6
# create csv files
write.csv(o3.2019.subset,
          row.names = FALSE,
          file ="../Data/Processed/EPAair_O3_NC2019_processed.csv")
write.csv(o3.2018.subset,
          row.names = FALSE,
          file ="../Data/Processed/EPAair_O3_NC2018_processed.csv")
write.csv(pm25.2018.subset,
          row.names = FALSE,
          file ="../Data/Processed/EPAair_pm25_NC2018_processed.csv")
write.csv(pm25.2019.subset,
          row.names = FALSE,
          file ="../Data/Processed/EPAair_pm25_NC2019_processed.csv")
```

Combine datasets

7. Combine the four datasets with `rbind`. Make sure your column names are identical prior to running this code.
8. Wrangle your new dataset with a pipe function (%>%) so that it fills the following conditions:

- Filter records to include just the sites that the four data frames have in common: "Linville Falls", "Durham Armory", "Leggett", "Hattie Avenue", "Clemmons Middle", "Mendenhall School", "Frying Pan Mountain", "West Johnston Co.", "Garinger High School", "Castle Hayne", "Pitt Agri. Center", "Bryson City", "Millbrook School". (The `intersect` function can figure out common factor levels if we didn't give you this list...)
- Some sites have multiple measurements per day. Use the split-apply-combine strategy to generate daily means: group by date, site, aqs parameter, and county. Take the mean of the AQI value, latitude, and longitude.
- Add columns for "Month" and "Year" by parsing your "Date" column (hint: `lubridate` package)
- Hint: the dimensions of this dataset should be 14,752 x 9.

9. Spread your datasets such that AQI values for ozone and PM2.5 are in separate columns. Each location on a specific date should now occupy only one row.

10. Call up the dimensions of your new tidy dataset.
11. Save your processed dataset with the following file name: "EPAair_O3_PM25_NC2122_Processed.csv"

```
#7
# combine data sets
EPAair = rbind(o3.2018.subset,
               o3.2019.subset,
               pm25.2018.subset,
               pm25.2019.subset)

#8
# find common sites
common.sites.o3 = intersect(o3.2018.subset$Site.Name,
                            o3.2019.subset$Site.Name)
common.sites.pm25 = intersect(pm25.2018.subset$Site.Name,
                            pm25.2019.subset$Site.Name)
common.sites = intersect(common.sites.o3,common.sites.pm25)
common.sites
```

```
##  [1] "Linville Falls"       "Durham Armory"        "Leggett"
##  [4] "Hattie Avenue"        "Clemmons Middle"      "Mendenhall School"
##  [7] "Frying Pan Mountain"  "West Johnston Co."    "Garinger High School"
## [10] "Castle Hayne"         "Pitt Agri. Center"    "Bryson City"
## [13] ""                     "Millbrook School"
```

```
# colnames(EPAair)

# filter/group data
EPAair.filter = EPAair %>%
  filter(Site.Name %in% common.sites & Site.Name != "") %>%
  group_by(Date,
           Site.Name,
           AQS_PARAMETER_DESC,
           COUNTY) %>%
  summarise_at(vars(DAILY_AQI_VALUE,
                    SITE_LATITUDE,
                    SITE_LONGITUDE),
               funs(mean(.))) %>%
  # I have no idea why this doesn't work so I googled the workaround above (?)
  # summarise(meanAQI = mean(DAILY_AQI_VALUE),
  #        meanLat = mean(SITE_LATITUDE),
  #        meanLong = mean(SITE_LONGITUDE)) %>%
  mutate(Month = month(Date)) %>% mutate(Year = year(Date))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
```

```
#9
# spread the data
EPAair.tidy <- pivot_wider(EPAair.filter,
                           names_from = AQS_PARAMETER_DESC,
                           values_from = DAILY_AQI_VALUE)

#10
# get dimensions
dim(EPAair.tidy)
```

```
## [1] 8976    9
```

```
#11
# write to a csv file
write.csv(EPAair.tidy,
          row.names = FALSE,
          file ="../Data/Processed/EPAair_O3_PM25_NC2122_Processed.csv")
```

Generate summary tables

12.

a. Use the split-apply-combine strategy to generate a summary data frame from your results from Step 9 above. Data should be grouped by site, month, and year. Generate the mean AQI values for ozone and PM2.5 for each group.
b. BONUS: Add a piped statement to 12a that removes rows where both mean ozone and mean PM2.5 have missing values.

13. Call up the dimensions of the summary dataset.

```
#12(a,b)
# get data summary
EPAair.summary = EPAair.tidy %>%
  group_by(Site.Name,
           Month,
           Year) %>%
  summarise_at(vars(Ozone,
                    PM2.5),
               funs(mean(.))) %>%
  # I have no idea why this doesn't work so I googled the workaround above (?)
  # summarise(meano3 = mean(Ozone),
  #           meanPM2.5 = mean(PM2.5)) %>%
  filter(!is.na(Ozone) & !is.na(PM2.5))
  # drop_na(Ozone, PM2.5) %>%
EPAair.summary
```

```
## # A tibble: 101 x 5
## # Groups:   Site.Name, Month [74]
##    Site.Name    Month  Year Ozone PM2.5
##    <fct>        <dbl> <dbl> <dbl> <dbl>
##  1 Bryson City      3  2018  41.6  34.7
##  2 Bryson City      4  2018  44.5  28.2
##  3 Bryson City      4  2019  45.4  26.7
##  4 Bryson City      7  2019  30.4  33.6
##  5 Bryson City      9  2018  25.4  25.1
##  6 Bryson City     10  2018  31    31.3
```

```
##  7 Castle Hayne     4  2018  48.7  14.9
##  8 Castle Hayne     4  2019  45.1  14.3
##  9 Castle Hayne     5  2019  42.8  16.5
## 10 Castle Hayne     7  2018  36.5  15.5
## # ... with 91 more rows
```

```
#13
# get dimensions
dim(EPAair.summary)
```

```
## [1] 101   5
```

14. Why did we use the function `drop_na` rather than `na.omit`?

    Answer: I used the `!is.na()` function here because it can be used to identify NAs in specific columns; it gives a boolean output so it can be combined with `&` and `|` as many times as necessary to get the exact end result you want. Similarly, `drop_na()` can be used to drop rows that have NAs in just specific columns. However, `na.omit()` drops all rows that have NAs in any column.