# Assignment 4: Data Wrangling (Fall 2024)

Iman Byndloss

## OVERVIEW

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

## Directions

1. Rename this file `<FirstLast>_A04_DataWrangling.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. Ensure that code in code chunks does not extend off the page in the PDF.

## Set up your session

1a. Load the `tidyverse`, `lubridate`, and `here` packages into your session.

1b. Check your working directory.

1c. Read in all four raw data files associated with the EPA Air dataset, being sure to set string columns to be read in a factors. See the README file for the EPA air datasets for more information (especially if you have not worked with air quality data previously).

2. Add the appropriate code to reveal the dimensions of the four datasets.

```
#1a: Used library() to load relevant packages.
library(tidyverse)
library(lubridate)
library(here)

#1b: Used getwd() to check the working directory.
# Notably, here() only finds your files.
getwd()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```
#1c: Used read.csv() to assign each set of raw data as a dataset in RStudio.
# Included stringsAsFactors=TRUE.
# Based on README, O3 correlates to ozone; PM25 correlates to PM2.5
# (i.e., inhalable particulate matter <= 2.5 micrometers in diameter).
```

```
EPA.O3.NC2018 <- read.csv(
  file=here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE
)
EPA.O3.NC2019 <- read.csv(
  file=here("Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE
)
EPA.PM25.NC2018 <- read.csv(
  file=here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE
)
EPA.PM25.NC2019 <- read.csv(
  file=here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE
)

#2: Used dim() to check the dimensions for each dataset.
dim(EPA.O3.NC2018)
```

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

```
dim(EPA.O3.NC2019)
```

```
## [1] 10592    20
```

```
dim(EPA.PM25.NC2018)
```

```
## [1] 8983   20
```

```
dim(EPA.PM25.NC2019)
```

```
## [1] 8581   20
```

> All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern? Yes, all datasets have 20 columns but varying numbers of rows.

## Wrangle individual datasets to create processed files.

3. Change the Date columns to be date objects.
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: Used as.Date() to establish format for the "Date" column of each date frame.
# Ensured it was saved to the appropriate place with "<-".
EPA.O3.NC2018$Date <- as.Date(EPA.O3.NC2018$Date, format = "%m/%d/%Y")
EPA.O3.NC2019$Date <- as.Date(EPA.O3.NC2019$Date, format = "%m/%d/%Y")
EPA.PM25.NC2018$Date <- as.Date(EPA.PM25.NC2018$Date, format = "%m/%d/%Y")
EPA.PM25.NC2019$Date <- as.Date(EPA.PM25.NC2019$Date, format = "%m/%d/%Y")

#4: Used select() to choose the relevant columns from each data frame.
# Ensured it was saved to the appropriate place with "<-".
EPA.O3.NC2018 <- select(EPA.O3.NC2018, Date, DAILY_AQI_VALUE, Site.Name,
                        AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE,
                        SITE_LONGITUDE)
EPA.O3.NC2019 <- select(EPA.O3.NC2019, Date, DAILY_AQI_VALUE, Site.Name,
                        AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE,
                        SITE_LONGITUDE)
EPA.PM25.NC2018 <- select(EPA.PM25.NC2018, Date, DAILY_AQI_VALUE, Site.Name,
                          AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE,
                          SITE_LONGITUDE)
EPA.PM25.NC2019 <- select(EPA.PM25.NC2019, Date, DAILY_AQI_VALUE, Site.Name,
                          AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE,
                          SITE_LONGITUDE)

#5: Used "=" to refill the relevant "AQS_PARAMETER_DESC" columns with "PM2.5".
EPA.PM25.NC2018$AQS_PARAMETER_DESC = "PM2.5"
EPA.PM25.NC2019$AQS_PARAMETER_DESC = "PM2.5"

#6: Used write.csv() to save and rename all datasets into the processed folder.
# "row.names = FALSE" prevents the automatic inclusion of numbered rows.
write.csv(EPA.O3.NC2018,
          row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2018_processed.csv")
write.csv(EPA.O3.NC2019,
          row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2019_processed.csv")
write.csv(EPA.PM25.NC2018,
          row.names = FALSE,
          file = "./Data/Processed/EPAair_PM25_NC2018_processed.csv")
write.csv(EPA.O3.NC2019,
          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:

- Include only 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 function `intersect` can figure out common factor levels - but it will include sites with missing site information, which you don't want...)

- Some sites have multiple measurements per day. Use the split-apply-combine strategy to generate daily means: group by date, site name, 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_NC1819_Processed.csv"

```
#7: Used rbind() to combine the four data sets.
# Renamed the combined data set.
EPA.NC <- rbind(EPA.O3.NC2018, EPA.O3.NC2019, EPA.PM25.NC2018, EPA.PM25.NC2019)

#8: Used the pipe function %>% to do the following...
# Used filter() to include only common sites between the four data sets.
# Used the split-apply-combine strategy to generate daily means;
# this included using groupby() and summarise() with mean();
# Notably, the in-class example also included filter(!is.na() & !is.na()...) for
# all variables addressed in mean(). By default, the mean() function returns NA
# if there are any NA values in the column
# Added columns for "Month" and "Year" by parsing "Date" column with mutate().
EPA.NC.processed <-
  EPA.NC %>%
  filter(Site.Name=="Linville Falls" | Site.Name=="Durham Armory" |
           Site.Name=="Leggett" | Site.Name=="Hattie Avenue" |
           Site.Name=="Clemmons Middle" | Site.Name=="Mendenhall School" |
           Site.Name=="Frying Pan Mountain" | Site.Name=="West Johnston Co." |
           Site.Name=="Garinger High School" | Site.Name=="Castle Hayne" |
           Site.Name=="Pitt Agri. Center" | Site.Name=="Bryson City" |
           Site.Name=="Millbrook School") %>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarise(meanAQI = mean(DAILY_AQI_VALUE),
            meanLAT = mean(SITE_LATITUDE),
            meanLONG = mean(SITE_LONGITUDE)) %>%
  mutate(month=month(Date), year=year(Date))
```

```
## `summarise()` has grouped output by 'Date', 'Site.Name', 'AQS_PARAMETER_DESC'.
## You can override using the `.groups` argument.
```

```
# Checked the dimensions with dim().
dim(EPA.NC.processed)
```

```
## [1] 14752     9
```

```
#9: Used pivot_wider() to spread the processed data frame,
# so that O3 and PM2.5 were separate columns.
EPA.NC.spread <- pivot_wider(EPA.NC.processed,
                             names_from = AQS_PARAMETER_DESC,
                             values_from = meanAQI)

#10: Checked the dimensions with dim().
dim(EPA.NC.spread)
```

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

```
#11: Used write.csv() to save and rename the new data frame.
write.csv(EPA.NC.spread,
          row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv")
```

## Generate summary tables

12. Use the split-apply-combine strategy to generate a summary data frame. Data should be grouped by site, month, and year. Generate the mean AQI values for ozone and PM2.5 for each group. Then, add a pipe to remove instances where mean **ozone** values are not available (use the function `drop_na` in your pipe). It's ok to have missing mean PM2.5 values in this result.

13. Call up the dimensions of the summary dataset.

```
#12: Used the pipe function %>% to do the following...
# Used the split-apply-combine strategy to generate a summary data frame;
# Used group_by() to form groups based on site, month, and year;
# Used summarise() and mean() to generate the mean AQI values (ozone and PM2.5);
# Used drop_na() to remove instances where meanO3 are NA.
EPA.NC.summary <-
  EPA.NC.spread %>%
  group_by(Site.Name, month, year) %>%
  summarise(meanO3 = mean(Ozone),
            meanPM25 = mean(PM2.5)) %>%
  drop_na(meanO3)
```

```
## `summarise()` has grouped output by 'Site.Name', 'month'. You can override
## using the `.groups` argument.
```

```
# Copied the previous code but used na.omit() instead of drop_na().
EPA.NC.test <-
  EPA.NC.spread %>%
  group_by(Site.Name, month, year) %>%
  summarise(meanO3 = mean(Ozone),
            meanPM25 = mean(PM2.5)) %>%
  na.omit(meanO3)
```

```
## `summarise()` has grouped output by 'Site.Name', 'month'. You can override
## using the `.groups` argument.
```

```
#13: Checked the dimensions with dim().
dim(EPA.NC.summary)
```

```
## [1] 182   5
```

```
dim(EPA.NC.test)
```

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

14. Why did we use the function `drop_na` rather than `na.omit`? Hint: replace `drop_na` with `na.omit` in part 12 and observe what happens with the dimensions of the summary date frame.

    Answer: drop_na() drops rows where any column contains a missing value while na.omit() returns the object with incomplete cases removed. drop_na() resulted in the data frame retaining more rows (182), in contrast to the 101 rows when using na.omit(). Based on this difference (and closer examination of the code), it becomes clear that na.omit() removed all NA, not only those associated with the "meanO3" column.