Assignment 4: Data Wrangling (Spring 2026)

Lexi Nelson

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on Data Wrangling.
Do not use any AI tools in completing this assignment.

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 is tidy and does not extend off the page in the PDF**.
7. **Push your completed RMD to your GitHub account**

Set up your session

1a. Load the `tidyverse` 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
#load tidyverse and here packages
#they are already downloaded so I can just use library() command
library(tidyverse)
library(here)

#1b
#check what R sees as the working directory
getwd() #this returns the same path as here()
```

```
## [1] "/home/guest/ENV872/EDE_Spring2026/Assignments"
```

```
#1c
#read in raw data files
#I coped the two data files from the EDE_Spring2026 raw data folder into a Data
#folder under my Assignments folder. I did this so that R could access the data
#in my working directory using "here".
EPAair_O3_2018 <- read.csv(
  file = here("./Data/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

EPAair_O3_2019 <- read.csv(
  file = here("./Data/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

EPAair_PM25_2018 <- read.csv(
  file = here("./Data/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

EPAair_PM25_2019 <- read.csv(
  file = here("./Data/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

#2
#add code to reveal the dimensions of the four datasets
dim(EPAair_O3_2018) #9737 rows by 20 columns
```

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

```
dim(EPAair_O3_2019) #10592 rows by 20 columns
```

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

```
dim(EPAair_PM25_2018) #8983 rows by 20 columns
```

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

```
dim(EPAair_PM25_2019) #8581 rows by 20 columns
```

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

```
#datasets have same number of columns but unique record counts
```

TIP: All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern?

Wrangle individual datasets to create processed files.

3. Change any date columns to be date objects.
4. Create new dataframes with just 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 cell values 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 any date columns to be date objects
#first check class of data columns
class(EPAair_O3_2018$Date) #factor
```

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

```
# Format as date
EPAair_O3_2018$Date <- mdy(EPAair_O3_2018$Date)
class(EPAair_O3_2018$Date) #now it is a date
```

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

```
class(EPAair_O3_2019$Date) #factor
```

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

```
# Format as date
EPAair_O3_2019$Date <- mdy(EPAair_O3_2019$Date)
class(EPAair_O3_2019$Date) #now it is a date
```

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

```
class(EPAair_PM25_2018$Date) #factor
```

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

```
# Format as date
EPAair_PM25_2018$Date <- mdy(EPAair_PM25_2018$Date)
class(EPAair_PM25_2018$Date) #now it is a date
```

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

```
class(EPAair_PM25_2019$Date) #factor
```

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

```
# Format as date
EPAair_PM25_2019$Date <- mdy(EPAair_PM25_2019$Date)
class(EPAair_PM25_2019$Date) #now it is a date
```

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

```r
#4
#create new dataframes with only selected few columns
#will also rename these dataframes since shorter names are easier to work with
O3_2018 <- select(EPAair_O3_2018, Date, DAILY_AQI_VALUE,
                  Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE,
                  SITE_LONGITUDE)
dim(O3_2018) #confirm there are now 7 columns in this dataframe
```

```
## [1] 9737    7
```

```r
O3_2019 <- select(EPAair_O3_2019, Date, DAILY_AQI_VALUE, Site.Name,
                  AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

PM25_2018 <- select(EPAair_PM25_2018, Date, DAILY_AQI_VALUE, Site.Name,
                    AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

PM25_2019 <- select(EPAair_PM25_2019, Date, DAILY_AQI_VALUE, Site.Name,
                    AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)


#5
#for the PM25 datasets, fill all cells in AQS_PARAMETER_DESC with "PM2.5"
PM25_2018$AQS_PARAMETER_DESC <- "PM2.5"
PM25_2019$AQS_PARAMETER_DESC <- "PM2.5"


#6
#Save all four processed datasets in the Processed folder
#first I had to create a folder in Assignments -> Data called Processed
#use row.names = FALSE since rows labels are just numbers, not meaningful
write.csv(
  O3_2018,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_O3_NC2018_processed.csv"))

write.csv(
  O3_2019,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_O3_NC2019_processed.csv"))

write.csv(
  PM25_2018,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_PM25_NC2018_processed.csv"))

write.csv(
  PM25_2019,
  row.names = FALSE,
  file = here("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. Use code to display the dimensions of the combined dataset.
9. 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"
- 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 new 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.

10. 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.
11. Call up the dimensions of your new tidy dataset.
12. Save your processed dataset with the following file name: "EPAair_O3_PM25_NC1819_Processed.csv"

```
#7
#combine the 4 datasets with rbind.
#This works because column names are identical
O3_PM_combined <- rbind(O3_2018, O3_2019, PM25_2018, PM25_2019)

#8
#display the dimensions of the combined dataset.
dim(O3_PM_combined) #37,893 rows by 7 columns
```

```
## [1] 37893     7
```

```
#9
#Wrangle dataset and save as new dataframe
#will need lubridate package so load that in
library(lubridate)
O3_PM_combined_wrangled <-
  O3_PM_combined %>% #include only sites that dataframes have in common
  mutate(Month = month(Date)) %>% #parse date column, add year & month columns
  mutate(Year = year(Date)) %>%
  filter(Site.Name %in% c("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")) %>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY, Month, Year) %>%
  #ensure new columns are included by adding them to group_by
  summarise(meanAQI = mean(DAILY_AQI_VALUE),
            mean_latitude = mean(SITE_LATITUDE),
            mean_longitude = mean(SITE_LONGITUDE))
```

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

```
#Use of split/apply/combine to generate daily means

#10
#Spread your datasets such that AQI values for ozone and PM2.5 are in separate columns
O3_PM_combined_wrangled_spread <- pivot_wider(O3_PM_combined_wrangled,
                                              names_from = AQS_PARAMETER_DESC,
                                              values_from = meanAQI)

#11
#find dimensions of new dataset
dim(O3_PM_combined_wrangled_spread) #8976 rows by 9 columns
```

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

```
#12
#save processed dataset
write.csv(
  O3_PM_combined,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv.csv"))
```

Generate summary tables

13. Use the split-apply-combine strategy to generate a summary data frame. Data should be split into groups by site, month, and year. Then compute the mean AQI values for ozone and PM2.5 for each group. Finally, add a pipe to remove instances where the 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.

14. Call up the dimensions of the summary dataset.

```
#13
#create a summary dataset and save as new dataframe
O3_PM_combined_wrangled_spread_summary <-
  O3_PM_combined_wrangled_spread %>%
  group_by(Site.Name, Month, Year) %>% #split into groups by site, month, year
  #compute mean AQI values
  summarise(mean_Ozone = mean(Ozone),
            mean_PM2.5 = mean(PM2.5)) %>%
  drop_na(mean_Ozone) #remove instances where mean ozone values not available
```

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

```
#Use of split/apply/combine

#14
#find dimensions of new dataset
dim(O3_PM_combined_wrangled_spread_summary) #182 rows by 9 columns
```

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

15. Why did we use the function `drop_na` rather than `na.omit`? Hint: replace `drop_na` with `na.omit` in part 13 and observe what happens with the dimensions of the summary date frame.

    Answer: Using na.omit in place of drop_na cut the dataset down to 101 rows. #I observed that all NA values had been dropped in the dataset including PM25 #values, rather than just the instances of ozone being NA which is what we #wanted to drop #Therefore we use drop_na so we can selectively drop NA values in one column #rather than the entire dataframe.

16. Stage, commit, and push your Assignment to your GitHub account. Provide a link to your repository below.

    Github repository URL: https://github.com/lexiwn17/EDE_Spring2026.git