Assignment 4: Data Wrangling (Spring 2026)

Aadya Shukla

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 Loading packages
library(tidyverse)
library(here)

#1b Checking working directory
here()
```

```
## [1] "/home/guest/872L/EDE_Spring2026"
```

```
#1c Reading in files using read.csv
EPA.Ozone2018 <- read.csv(
  here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE
```

```
)
EPA.Ozone2019 <- read.csv(
  here('Data/Raw/EPAair_O3_NC2019_raw.csv'),
  stringsAsFactors = TRUE
)
EPA.PM.2018 <- read.csv(
  here('Data/Raw/EPAair_PM25_NC2018_raw.csv'),
  stringsAsFactors = TRUE
)
EPA.PM.2019 <- read.csv(
  here('Data/Raw/EPAair_PM25_NC2019_raw.csv'),
  stringsAsFactors = TRUE
)

#2 Checking dimensions
dim(EPA.Ozone2018) #9,737 rows and 20 columns
```

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

```
dim(EPA.Ozone2019) #10,592 rows and 20 columns
```

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

```
dim(EPA.PM.2018) #8,983 rows and 20 columns
```

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

```
dim(EPA.PM.2019) #8,581 rows and 20 columns
```

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

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".

```r
#3 Using mdy as the order follows month-day-year
EPA.Ozone2018$Date <- mdy(EPA.Ozone2018$Date)
EPA.Ozone2019$Date <- mdy(EPA.Ozone2019$Date)
EPA.PM.2018$Date <- mdy(EPA.PM.2018$Date)
EPA.PM.2019$Date <- mdy(EPA.PM.2019$Date)

#4 Using select to remove unnecessary columns
EPA.Ozone2018.Processed <- select(
  EPA.Ozone2018, Date, DAILY_AQI_VALUE, Site.Name,
  AQS_PARAMETER_DESC, COUNTY:SITE_LONGITUDE)

EPA.Ozone2019.Processed <- select(
  EPA.Ozone2019, Date, DAILY_AQI_VALUE, Site.Name,
  AQS_PARAMETER_DESC, COUNTY:SITE_LONGITUDE)

EPA.PM.2018.Processed <- select(
  EPA.PM.2018, Date, DAILY_AQI_VALUE, Site.Name,
  AQS_PARAMETER_DESC, COUNTY:SITE_LONGITUDE)

EPA.PM.2019.Processed <- select(
  EPA.PM.2019, Date, DAILY_AQI_VALUE, Site.Name,
  AQS_PARAMETER_DESC, COUNTY:SITE_LONGITUDE)

#5 Using mutate to assign column values to PM2.5
EPA.PM.2018.Processed <- mutate(
  EPA.PM.2018.Processed, AQS_PARAMETER_DESC = "PM2.5")

EPA.PM.2019.Processed <- mutate(
  EPA.PM.2019.Processed, AQS_PARAMETER_DESC = "PM2.5")

#6 Saving datasets as processed csv files using write.csv
write.csv(
  EPA.Ozone2018.Processed,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_O3_NC2018_processed.csv"))

write.csv(
  EPA.Ozone2019.Processed,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_O3_NC2019_processed.csv"))

write.csv(
  EPA.PM.2018.Processed,
  row.names = FALSE,
  file = here("Data/Processed/EPAair_PM25_NC2018_processed.csv"))

write.csv(
  EPA.PM.2019.Processed,
  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 Using rbind to combine the 4 processed datasets
EPA.data <- rbind(
  EPA.Ozone2018.Processed,EPA.Ozone2019.Processed,
  EPA.PM.2018.Processed,EPA.PM.2019.Processed)

#8 Dimensions
dim(EPA.data) #37,893 rows and 7 columns
```

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

```
#9 Filtered rows, grouped values, calculated mean, and added 2 columns for Month & Year
EPA.data.edit <- EPA.data %>%
  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) %>%
  summarize(
    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.
```

```
#10 Transformed row values into columns
EPA.data.processed <- EPA.data.edit %>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC,
    values_from = meanAQI,
  )

#11 Dimensions
dim(EPA.data.processed) #8,976 rows and 9 columns
```

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

```
#12 Saving processed files
write.csv(
  EPA.data.processed,
  row.names = FALSE,
  here("Data/Processed/EPAair_O3_PM25_NC1819_Processed.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 Grouped values, calculated mean AQIs, removed NAs from ozone column
EPA.data.summary <- EPA.data.processed %>%
  group_by(Site.Name,Month,Year) %>%
  summarize(
    meanOzone = mean(Ozone),
    meanPM2.5 = mean(PM2.5),
  ) %>%
  drop_na(meanOzone)
```

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

```
#14 Dimensions
dim(EPA.data.summary) # 182 rows and 5 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: The na.omit function is a very crude way of removing NA values as it removes any and every row that has NAs. In our case, we are only removing NA values from the ozone column, not from the PM 2.5 column. This is easily achieved using drop_na. However, the na.omit removed rows that had NAs in the PM2.5 column as well, resulting in the deletion of 80 more columns than necessary.

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

    Github repository URL: https://github.com/aadya0126/EDE_Spring2026