Assignment 4: Data Wrangling (Fall 2024)

Mazhar Bhuyan

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
library(tidyverse)
library(lubridate)
library(here)

#1b
getwd()
```

```
## [1] "/home/guest/EDA_Spring2025"
```

```
#1c

file_path_EPA1 <- here("Data", "Raw", "EPAair_O3_NC2018_raw.csv")
file_path_EPA2 <- here("Data", "Raw", "EPAair_O3_NC2019_raw.csv")
file_path_EPA3 <- here("Data", "Raw", "EPAair_PM25_NC2018_raw.csv")
```

```r
file_path_EPA4 <- here("Data", "Raw", "EPAair_PM25_NC2019_raw.csv")

EPA_Oz_2018 <- read.csv(file_path_EPA1,
                         stringsAsFactors = TRUE)

EPA_Oz_2019 <- read.csv(file_path_EPA2,
                         stringsAsFactors = TRUE)

EPA_PM25_2018 <- read.csv(file_path_EPA3,
                         stringsAsFactors = TRUE)

EPA_PM25_2019 <- read.csv(file_path_EPA4,
                         stringsAsFactors = TRUE)
#2

dim(EPA_Oz_2018)
```

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

```r
dim(EPA_Oz_2019)
```

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

```r
dim(EPA_PM25_2018)
```

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

```r
dim(EPA_PM25_2019)
```

```
## [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 contain 20 variables, i.e. columns each. But obs are different

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".

```r
#3

glimpse(EPA_Oz_2018$Date)
```

```
##  Factor w/ 364 levels "01/01/2018","01/02/2018",..: 60 61 62 63 64 65 66 67 68 69 ...
```

```r
EPA_Oz_2018$Date <- mdy(EPA_Oz_2018$Date)
EPA_Oz_2019$Date <- mdy(EPA_Oz_2019$Date)
EPA_PM25_2018$Date <- mdy(EPA_PM25_2018$Date)
EPA_PM25_2019$Date <- mdy(EPA_PM25_2019$Date)

#4

EPA_Oz_2018_Processed <-EPA_Oz_2018 %>%
                          select(Date,
                           DAILY_AQI_VALUE,
                           Site.Name,
                           AQS_PARAMETER_DESC,
                           COUNTY,
                           SITE_LATITUDE,
                           SITE_LONGITUDE)

EPA_Oz_2019_Processed <- EPA_Oz_2019 %>%
                          select(Date,
                                DAILY_AQI_VALUE,
                                Site.Name,
                                AQS_PARAMETER_DESC,
                                COUNTY,
                                SITE_LATITUDE,
                                SITE_LONGITUDE)

EPA_PM25_2018_Processed <- EPA_PM25_2018 %>%
                            select(Date,
                                   DAILY_AQI_VALUE,
                                   Site.Name,
                                   AQS_PARAMETER_DESC,
                                   COUNTY,
                                   SITE_LATITUDE,
                                   SITE_LONGITUDE)

EPA_PM25_2019_Processed <- EPA_PM25_2019 %>%
                            select(Date,
                                   DAILY_AQI_VALUE,
                                   Site.Name,
                                   AQS_PARAMETER_DESC,
                                   COUNTY,
                                   SITE_LATITUDE,
                                   SITE_LONGITUDE)
```

```r
#5
#[kept the previous filename]
EPA_PM25_2018_Processed <- EPA_PM25_2018 %>%
                            select(Date,
                                   DAILY_AQI_VALUE,
                                   Site.Name,
                                   AQS_PARAMETER_DESC,
                                   COUNTY,
                                   SITE_LATITUDE,
                                   SITE_LONGITUDE) %>%
                            mutate(AQS_PARAMETER_DESC = "PM2.5")


EPA_PM25_2019_Processed <- EPA_PM25_2019 %>%
                            select(Date,
                                   DAILY_AQI_VALUE,
                                   Site.Name,
                                   AQS_PARAMETER_DESC,
                                   COUNTY,
                                   SITE_LATITUDE,
                                   SITE_LONGITUDE) %>%
                            mutate(AQS_PARAMETER_DESC = "PM2.5")

#6

write.csv(EPA_Oz_2018_Processed,
          file = here("Data","Processed","EPAair_O3_NC2018_processed.csv"),
          row.names = FALSE)

write.csv(EPA_Oz_2019_Processed,
          file = here("Data","Processed","EPAair_O3_NC2019_processed.csv"),
          row.names = FALSE)

write.csv(EPA_PM25_2018_Processed,
          file = here("Data","Processed", "EPAair_PM25_NC2018_processed.csv"),
          row.names = FALSE)
write.csv(EPA_PM25_2019_Processed,
          file = here("Data","Processed","EPAair_PM25_NC2019_processed.csv"),
          row.names = FALSE)
```

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
#Combining by rows. We can also use "bind_rows"

EPA_combined <- rbind(EPA_Oz_2018_Processed,
                      EPA_Oz_2019_Processed,
                      EPA_PM25_2018_Processed,
                      EPA_PM25_2019_Processed)
dim(EPA_combined)
```

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

```
#8
common_sites <- 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")

#"is.na" or "na.rm" which one to use. I prefer na.rm since it does not remove any rows but omits NA whi

EPA_common_sites <- EPA_combined %>%
  filter(Site.Name %in% common_sites) %>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarise(
    Mean_AQI = mean(DAILY_AQI_VALUE, na.rm = TRUE),
    Mean_Latitude = mean(SITE_LATITUDE, na.rm = TRUE),
    Mean_Longitude = mean(SITE_LONGITUDE, na.rm = TRUE)) %>%
    mutate(
    Month = month(Date, label = TRUE),
    Year = year(Date)
  )
```

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

```r
# Count the total number of missing values in the data frame

#sum(is.na(EPA_combined)) #no missing value in combined dataset

dim(EPA_common_sites)
```

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

```r
#9

EPAair_O3_PM25_NC1819_Processed <- EPA_common_sites %>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC,
    values_from = Mean_AQI)
#10

dim(EPAair_O3_PM25_NC1819_Processed)
```

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

```r
#11
write.csv(EPAair_O3_PM25_NC1819_Processed,
          file = here("Data","Processed","EPAair_O3_PM25_NC1819_Processed.csv"),
          row.names = FALSE)
```

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.

```r
#12

 EPA_summary <- EPAair_O3_PM25_NC1819_Processed %>%
  group_by(Site.Name, Month, Year) %>%
  summarise(
    Mean_AQI_Ozone = mean(Ozone, na.rm = TRUE),
    Mean_AQI_PM25 = mean(PM2.5, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  drop_na(Mean_AQI_Ozone)

summary(EPA_summary)
```

```
##                  Site.Name      Month         Year      Mean_AQI_Ozone
##  Garinger High School: 24   Mar    :26   Min.   :2018   Min.   :23.90
##  Millbrook School    : 24   Apr    :26   1st Qu.:2018   1st Qu.:34.55
##  Clemmons Middle     : 18   May    :26   Median :2019   Median :41.61
```

```
##  Durham Armory        : 18   Jun    :26   Mean   :2019   Mean   :40.57
##  Frying Pan Mountain : 18   Jul    :26   3rd Qu.:2019   3rd Qu.:45.46
##  Leggett              : 18   Aug    :26   Max.   :2019   Max.   :59.23
##  (Other)              :119   (Other):83
##  Mean_AQI_PM25
##  Min.   : 1.778
##  1st Qu.:25.516
##  Median :31.935
##  Mean   :30.148
##  3rd Qu.:36.014
##  Max.   :44.600
##  NA's   :16
```

```r
#13

dim(EPA_summary)
```

```
## [1] 239   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.

```r
EPA_summary_na_omit <- EPAair_O3_PM25_NC1819_Processed %>%
  group_by(Site.Name, Month, Year) %>%
  summarise(
    Mean_AQI_Ozone = mean(Ozone, na.rm = TRUE),
    Mean_AQI_PM25 = mean(PM2.5, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  na.omit()
dim(EPA_summary_na_omit)
```

```
## [1] 223   5
```

Answer:

The na.omit() function removed 16 observations where it found NA values, as it is not selective about which columns to check. In contrast, when we applied drop_na(), we observed that, for example, the Frying Pan Mountain site had NA values specifically for AQS_PM25. However, when na.omit() was used, it removed these observations entirely.

Removing observations from a dataset requires careful investigation, as it can significantly impact the analysis and outcomes. This is why we chose to use drop_na() instead of na.omit(), allowing for a more targeted approach to handling missing values without unnecessarily discarding data.