Assignment 4: Data Wrangling (Fall 2024)

Aishwarya Patankar

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
#Installing the packages tidyverse, lubridate and here
#install.packages("tidyverse") : Line is hashed out while knitting file
#install.packages("lubridate")  Line is hashed out while knitting file
#install.packages("here")  Line is hashed out while knitting file

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

#1b
#Checking the working directory
getwd()
```

```
## [1] "/Users/aishwaryapatankar/Documents/Duke University/Spring2025/ENERGY872/EDA_Spring2025"
```

```r
#1c
Ozone2018 <- read.csv(
  file = here("./Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

Ozone2019 <- read.csv(
  file = here("./Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

PM2018 <- read.csv(
  file = here("./Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

PM2019 <- read.csv(
  file = here("./Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

#2
dim(Ozone2018)
```

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

```r
dim(Ozone2019)
```

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

```r
dim(PM2018)
```

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

```r
dim(PM2019)
```

```
## [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, we observe that all datasets have 20 columns but different number of rows. Ozone2018 has 9737 rows, Ozone2019 has 10592 rows, PM2018 has 8983 rows and PM2019 has 8581 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".

```r
#3
#Checking column names in all 4 datasheets
colnames(Ozone2018)
```

```
##  [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
colnames(Ozone2019)
```

```
##  [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
colnames(PM2018)
```

```
##  [1] "Date"                           "Source"
##  [3] "Site.ID"                        "POC"
```

```
##  [5] "Daily.Mean.PM2.5.Concentration" "UNITS"
##  [7] "DAILY_AQI_VALUE"                "Site.Name"
##  [9] "DAILY_OBS_COUNT"                "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"             "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"                      "CBSA_NAME"
## [15] "STATE_CODE"                     "STATE"
## [17] "COUNTY_CODE"                    "COUNTY"
## [19] "SITE_LATITUDE"                  "SITE_LONGITUDE"
```

```
colnames(PM2019)
```

```
##  [1] "Date"                           "Source"
##  [3] "Site.ID"                        "POC"
##  [5] "Daily.Mean.PM2.5.Concentration" "UNITS"
##  [7] "DAILY_AQI_VALUE"                "Site.Name"
##  [9] "DAILY_OBS_COUNT"                "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"             "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"                      "CBSA_NAME"
## [15] "STATE_CODE"                     "STATE"
## [17] "COUNTY_CODE"                    "COUNTY"
## [19] "SITE_LATITUDE"                  "SITE_LONGITUDE"
```

```
#Using str to check the format in which the Date column is
str(Ozone2018$Date)
```

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

```
str(Ozone2019$Date)
```

```
##  Factor w/ 365 levels "01/01/2019","01/02/2019",..: 1 2 3 4 5 6 7 8 9 10 ...
```

```
str(PM2018$Date)
```

```
##  Factor w/ 365 levels "01/01/2018","01/02/2018",..: 2 5 8 11 14 17 20 23 26 29 ...
```

```
str(PM2019$Date)
```

```
##  Factor w/ 365 levels "01/01/2019","01/02/2019",..: 3 6 9 12 15 18 21 24 27 30 ...
```

```
# Converting Date Column in the four datasheets into date object
Ozone2018$Date <- as.Date(Ozone2018$Date, format = "%m/%d/%Y")
Ozone2019$Date <- as.Date(Ozone2019$Date, format = "%m/%d/%Y")
PM2018$Date <- as.Date(PM2018$Date, format = "%m/%d/%Y")
PM2019$Date <- as.Date(PM2019$Date, format = "%m/%d/%Y")

#Checking Conversion to Date Object
class(Ozone2018$Date)
```

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

```
class(Ozone2019$Date)
```

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

```
class(PM2018$Date)
```

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

```
class(PM2019$Date)
```

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

```
#4 Selecting Columns
Ozone2018filtered <- select(Ozone2018,Date, DAILY_AQI_VALUE, Site.Name,AQS_PARAMETER_DESC, COUNTY, SITE_
Ozone2019filtered <- select(Ozone2019,Date, DAILY_AQI_VALUE, Site.Name,AQS_PARAMETER_DESC, COUNTY, SITE_
PM2018filtered <- select(PM2018,Date, DAILY_AQI_VALUE, Site.Name,AQS_PARAMETER_DESC, COUNTY, SITE_LATITU
PM2019filtered <- select(PM2019,Date, DAILY_AQI_VALUE, Site.Name,AQS_PARAMETER_DESC, COUNTY, SITE_LATITU

#5 Making cells in the PM sheets column AQS_PARAMETER_DESC to "PM2.5
PM2018filtered$AQS_PARAMETER_DESC <- "PM2.5"
PM2019filtered$AQS_PARAMETER_DESC <- "PM2.5"

#6
#Saving the four processed datasets
write.csv(Ozone2018filtered, file = "/Users/aishwaryapatankar/Documents/Duke University/Spring2025/ENERG

write.csv(Ozone2019filtered, file = "/Users/aishwaryapatankar/Documents/Duke University/Spring2025/ENERG

write.csv(PM2018filtered, file = "/Users/aishwaryapatankar/Documents/Duke University/Spring2025/ENERGY87

write.csv(PM2019filtered, file = "/Users/aishwaryapatankar/Documents/Duke University/Spring2025/ENERGY87
```

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
#Confirming Column Names are Same
colnames(Ozone2018filtered)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(Ozone2019filtered)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(PM2018filtered)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(PM2019filtered)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
#Joining dataframes using rbind
AirQualityDataJoined <- rbind(Ozone2018filtered, Ozone2019filtered, PM2018filtered, PM2019filtered)

#8 Using the pipe command for filtering sites, grouping,
#summarizing mean and creating month and year columns
AirQualityDataJoined_processed <-
  AirQualityDataJoined %>%
  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(meanAQIvalue = mean(DAILY_AQI_VALUE),
            meanLatitude = mean(SITE_LATITUDE),
            meanLongitude = mean(SITE_LONGITUDE),
            .groups = "drop") %>% #Ungrouping after summarizing to avoid error
  mutate(month=month(Date), year=year(Date))


#9
#Spreading dataset 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
AirQualityDataJoined_processed_spread <- pivot_wider(AirQualityDataJoined_processed,names_from = AQS_PA

#10
#Checking Dimensions of New Dataset
dim(AirQualityDataJoined_processed_spread)
```

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

```
#11
#Saving New Processed File
write.csv(AirQualityDataJoined_processed_spread, file = "/Users/aishwaryapatankar/Documents/Duke Univers
```

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 Creating Summary dataframe
SummaryDf <-
  AirQualityDataJoined_processed_spread%>%
  group_by(Site.Name, month, year)%>%
  summarise(meanOzone = mean(Ozone),
            meanPM = mean(PM2.5),
            .groups = "drop")%>% #Ungrouping after summarizing to avoid error
  drop_na(meanOzone)
#13
dim(SummaryDf)
```

```
## [1] 182   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: We use drop_na rather than na.omit because na.omit cause the removal of all rows with NA values in them. In case of using na.omit we lose data of sites where only Ozone values are available.