Assignment 4: Data Wrangling (Fall 2024)

Asreeta Ushasri

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 Install packages tidyverse, lubridate, here.

#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("here")

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

#1b Check working directory

getwd()
```

```
## [1] "/home/guest/AU_EDE_Fall2024"
```

```
#1c Add four data sets to global environment.

O3_NC2018 <-
  read.csv(
file =
  here("./Data/Raw/EPAair_O3_NC2018_raw.csv"),
stringsAsFactors = TRUE)

O3_NC2019 <-
  read.csv(
file =
  here("./Data/Raw/EPAair_O3_NC2019_raw.csv"),
stringsAsFactors = TRUE)

PM25_NC2018 <-
  read.csv(
file =
  here("./Data/Raw/EPAair_PM25_NC2018_raw.csv"),
stringsAsFactors = TRUE)

PM25_NC2019 <-
  read.csv(
file =
  here("./Data/Raw/EPAair_PM25_NC2019_raw.csv"),
stringsAsFactors = TRUE)

#2 Check the dimensions of each data set.

dim(O3_NC2018)
```

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

```
dim(O3_NC2019)
```

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

```
dim(PM25_NC2018)
```

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

```
dim(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?

> Answer: Yes, these data sets all have 20 columns. However, the number of rows is different for each data set.

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 Check the format of the date columns again

#view(O3_NC2018)
#view(O3_NC2019)
#view(PM25_NC2018)
#view(PM25_NC2019)

#Use the as.Date function to change to date objects.

O3_NC2018$Date <- as.Date(O3_NC2018$Date,
                          format = "%m/%d/%Y")
O3_NC2019$Date <- as.Date(O3_NC2019$Date,
                          format = "%m/%d/%Y")
PM25_NC2018$Date <- as.Date(PM25_NC2018$Date,
                            format = "%m/%d/%Y")
PM25_NC2019$Date <- as.Date(PM25_NC2019$Date,
                            format = "%m/%d/%Y")


#4 Select columns in table, use select function

O3_NC2018 <- select(O3_NC2018, Date, DAILY_AQI_VALUE, Site.Name,
                    AQS_PARAMETER_DESC, COUNTY,
                    SITE_LATITUDE, SITE_LONGITUDE)

O3_NC2019 <- select(O3_NC2019, Date, DAILY_AQI_VALUE, Site.Name,
                    AQS_PARAMETER_DESC, COUNTY,
                    SITE_LATITUDE, SITE_LONGITUDE)

PM25_NC2018 <- select(PM25_NC2018, Date, DAILY_AQI_VALUE, Site.Name,
                      AQS_PARAMETER_DESC, COUNTY,
                      SITE_LATITUDE, SITE_LONGITUDE)

PM25_NC2019 <- select(PM25_NC2019, Date, DAILY_AQI_VALUE, Site.Name,
                      AQS_PARAMETER_DESC, COUNTY,
                      SITE_LATITUDE, SITE_LONGITUDE)


#5 Fill all cells in AQS_PARAMETER_DESC with "PM2.5"

#First, view the tables with the updated columns
```

```
#view(PM25_NC2018)
#view(PM25_NC2019)

#Now, use the mutate function to change the cell values.

PM25_NC2018 <- mutate(PM25_NC2018,
                      AQS_PARAMETER_DESC = "PM2.5")

PM25_NC2019 <- mutate(PM25_NC2019,
                      AQS_PARAMETER_DESC = "PM2.5")

#View the data again to make sure the cells have changed correctly.

#view(PM25_NC2018)
#view(PM25_NC2019)


#6 Save all four processed datasets in the Processed folder.
#Use the write.csv function.

write.csv(O3_NC2018, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2018_Processed.csv")

write.csv(O3_NC2019, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2019_Processed.csv")

write.csv(PM25_NC2018, row.names = FALSE,
          file = "./Data/Processed/EPAair_PM25_NC2018_Processed.csv")

write.csv(PM25_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 Combine data frames
#First, read in processed data

O3_NC2018_Processed <-
  read.csv( "./Data/Processed/EPAair_O3_NC2018_Processed.csv")

O3_NC2019_Processed <-
  read.csv( "./Data/Processed/EPAair_O3_NC2019_Processed.csv")

PM25_NC2018_Processed <-
  read.csv( "./Data/Processed/EPAair_PM25_NC2018_Processed.csv")

PM25_NC2019_Processed <-
  read.csv( "./Data/Processed/EPAair_PM25_NC2019_Processed.csv")

#Check if columns are identical

#view(O3_NC2018_Processed)
#view(O3_NC2019_Processed)
#view(PM25_NC2018_Processed)
#view(PM25_NC2019_Processed)

#Then, use rbind to combine data frames

O3_PM25_NC1819 <- rbind(O3_NC2018_Processed, O3_NC2019_Processed,
                        PM25_NC2018_Processed, PM25_NC2019_Processed)

#8 Wrangle your new dataset with a pipe function (%>%)
#It should 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"

O3_PM25_NC1819_Sites <-
  O3_PM25_NC1819 %>%
  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(Mean.DAILY.AQI = 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'.
## You can override using the `.groups` argument.
```

```
O3_PM25_NC1819_Sites <- mutate(O3_PM25_NC1819_Sites, Month = month(Date))

O3_PM25_NC1819_Sites <- mutate(O3_PM25_NC1819_Sites, Year = year(Date))

#Check data frame using view and dim

#view(O3_PM25_NC1819_Sites)
dim(O3_PM25_NC1819_Sites)
```

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

```
#9 Use the pivot wider operation to spread data.

O3_PM25_NC1819_Sites_Spread <-
  pivot_wider(O3_PM25_NC1819_Sites, names_from = AQS_PARAMETER_DESC,
              values_from = Mean.DAILY.AQI)

#Check the dataset using view

#view(O3_PM25_NC1819_Sites_Spread)


#10 Determine dimensions of new sheet

dim(O3_PM25_NC1819_Sites_Spread)
```

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

```
#11 Save processed data as csv

write.csv(O3_PM25_NC1819_Sites_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 Split, Apply, Combine Strategy

O3_PM25_NC1819_Processed <-
  O3_PM25_NC1819_Sites_Spread %>%
    group_by(Site.Name, Month, Year) %>%
    summarise(PM2.5 = mean(PM2.5),
              Ozone = mean(Ozone)) %>%
     drop_na(PM2.5)
```

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

```
#view(O3_PM25_NC1819_Processed)

#13 Call upon the dimensions of this data set.

dim(O3_PM25_NC1819_Processed)
```

```
## [1] 211   5
```

```
#14 Compare drop_na to na.omit

O3_PM25_NC1819_Processed_Trial <-
  O3_PM25_NC1819_Sites_Spread %>%
    group_by(Site.Name, Month, Year) %>%
    summarise(PM2.5 = mean(PM2.5),
              Ozone = mean(Ozone)) %>%
     na.omit(PM2.5)
```

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

```
dim(O3_PM25_NC1819_Processed_Trial)
```

```
## [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: The drop_na function restricts N/A in the PM2.5 column only, giving 211 rows. The na.omit function omits all N/A values from PM2.5 and Ozone, which leaves only 101 rows. In this case, we want to omit only the N/A from the PM2.5 column, but we want to continue to include the N/A from the Ozone column. The operation drop_na allows us to drop values from one specific column, and maintain the N/A values in the other column.