Assignment 4: Data Wrangling (Fall 2024)

Leah Li

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.

```r
#1a
#Load packages
library(tidyverse)
library(lubridate)
library(here)

#1b
#Check working directory
getwd()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```r
here()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```r
#1c
#Load data
EPA_Raw_O3_2018.data <- read.csv(
  file = here("./Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

EPA_Raw_O3_2019.data <- read.csv(
  file = here("./Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

EPA_Raw_PM_2018.data <- read.csv(
  file = here("./Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

EPA_Raw_PM_2019.data <- read.csv(
  file = here("./Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

#2
#Check Dimension
dim(EPA_Raw_O3_2018.data)
```

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

```r
dim(EPA_Raw_O3_2019.data)
```

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

```r
dim(EPA_Raw_PM_2018.data)
```

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

```r
dim(EPA_Raw_PM_2019.data)
```

```
## [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, my four dataset has different rows but the same number of columns (20 cols).

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
# Convert the "Date" column to a date object for each dataset
EPA_Raw_O3_2018.data$Date <- as.Date(EPA_Raw_O3_2018.data$Date, format = "%m/%d/%Y")
EPA_Raw_O3_2019.data$Date <- as.Date(EPA_Raw_O3_2019.data$Date, format = "%m/%d/%Y")
EPA_Raw_PM_2018.data$Date <- as.Date(EPA_Raw_PM_2018.data$Date, format = "%m/%d/%Y")
EPA_Raw_PM_2019.data$Date <- as.Date(EPA_Raw_PM_2019.data$Date, format = "%m/%d/%Y")

#4
# Selecting the desired columns from each dataset
EPA_O3_2018_processed <- EPA_Raw_O3_2018.data %>%
  select(Date, DAILY_AQI_VALUE, Site.Name,
         AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

EPA_O3_2019_processed <- EPA_Raw_O3_2019.data %>%
  select(Date, DAILY_AQI_VALUE, Site.Name,
         AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

EPA_PM_2018_processed <- EPA_Raw_PM_2018.data %>%
  select(Date, DAILY_AQI_VALUE, Site.Name,
         AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

EPA_PM_2019_processed <- EPA_Raw_PM_2019.data %>%
  select(Date, DAILY_AQI_VALUE, Site.Name,
         AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)



#5
# For EPA_PM_2018_selected.data, fill AQS_PARAMETER_DESC with "PM2.5"
EPA_PM_2018_processed$AQS_PARAMETER_DESC <- "PM2.5"

# For EPA_PM_2019_selected, fill AQS_PARAMETER_DESC with "PM2.5"
EPA_PM_2019_processed$AQS_PARAMETER_DESC <- "PM2.5"

#6
#Save files in Processed folder
write.csv(EPA_O3_2018_processed, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2018_processed.csv")

write.csv(EPA_O3_2019_processed, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2019_processed.csv")

write.csv(EPA_PM_2018_processed, row.names = FALSE,
          file = "./Data/Processed/EPAair_PM25_NC2018_processed.csv")

write.csv(EPA_PM_2019_processed, 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"

```r
#7
# Ensure column names are identical across all processed datasets
colnames(EPA_O3_2018_processed) <- colnames(EPA_PM_2018_processed)
colnames(EPA_O3_2019_processed) <- colnames(EPA_PM_2019_processed)

# Combine all four processed datasets using rbind
EPA_combined <- rbind(EPA_O3_2018_processed,
                      EPA_O3_2019_processed,
                      EPA_PM_2018_processed,
                      EPA_PM_2019_processed)



#8
# Wrangle and reshape the combined dataset
EPA_wrangled <- EPA_combined %>%
  # Filter out rows with missing Site.Name
  filter(!is.na(Site.Name)) %>%
  # Filter for the specific common sites directly in the pipe
  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, and County
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  # Take the mean of AQI value, latitude, and longitude
  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)
  ) %>%
  # Add Month and Year columns parsed from the Date
  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.
```

```
#9
# Pivot the wrangled dataset to have separate columns for Ozone and PM2.5 AQI values
EPA_spread <- EPA_wrangled %>%
  # Pivot the data so that Ozone and PM2.5 have separate AQI columns
  pivot_wider(names_from = AQS_PARAMETER_DESC, values_from = Mean_AQI, names_prefix = "AQI_") %>%
  # Rename columns for clarity
  rename(AQI_Ozone = AQI_Ozone, AQI_PM25 = AQI_PM2.5)



#10
dim(EPA_spread)
```

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

```
#11
write.csv(EPA_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
# Generate summary grouped by Site.Name, Month, and Year
summary_EPA <- EPA_spread %>%
  group_by(Site.Name, Month, Year) %>%
  # Calculate the mean AQI values for Ozone and PM2.5
  summarise(
    Mean_AQI_Ozone = mean(AQI_Ozone, na.rm = TRUE),
    Mean_AQI_PM25 = mean(AQI_PM25, na.rm = TRUE)
  ) %>%
  # Remove rows where Ozone mean values are missing
  drop_na(Mean_AQI_Ozone)
```

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

```
#13

dim(summary_EPA)
```

```
## [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.

    Answer: We used drop_na() instead of na.omit() because drop_na() allows for targeted removal of rows based on missing values in a specific column, in this case, the Ozone AQI values. This ensures that rows with missing Ozone values are dropped, while rows with missing PM2.5 values are retained, which is the desired behavior. In contrast, na.omit() removes rows with any missing values in the dataset, which would result in the unintended removal of rows with missing PM2.5 values.