Assignment 4: Data Wrangling (Fall 2024)

Nargis Taraki

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 I checked the work directory.
getwd()
```

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

```
#1c I read in all data

 EPAair_O3_NC2018_raw <- read.csv(here("./Data/Raw/EPAair_O3_NC2018_raw.csv"),
 stringsAsFactors = TRUE)
 EPAair_O3_NC2019_raw <- read.csv(here("./Data/Raw/EPAair_O3_NC2019_raw.csv"),
```

```
 stringsAsFactors = TRUE)
 EPAair_PM25_NC2018_raw <- read.csv(here
 ("./Data/Raw/EPAair_PM25_NC2018_raw.csv"), stringsAsFactors = TRUE)
 EPAair_PM25_NC2019_raw <- read.csv(here
 ("./Data/Raw/EPAair_PM25_NC2019_raw.csv"), stringsAsFactors = TRUE)

#2

dim(EPAair_O3_NC2018_raw)
```

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

```
dim(EPAair_O3_NC2019_raw)
```

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

```
dim(EPAair_PM25_NC2018_raw)
```

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

```
dim(EPAair_PM25_NC2019_raw)
```

```
## [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 all datasets have 20 colums each but the numbers of rows 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".

```
#3 Here I  Changed the Date column to Date objects

EPAair_O3_NC2018_raw$Date <- lubridate::mdy(EPAair_O3_NC2018_raw$Date)
EPAair_O3_NC2019_raw$Date <- lubridate::mdy(EPAair_O3_NC2019_raw$Date)
EPAair_PM25_NC2018_raw$Date <- lubridate::mdy(EPAair_PM25_NC2018_raw$Date)
EPAair_PM25_NC2019_raw$Date <- lubridate::mdy(EPAair_PM25_NC2019_raw$Date)

#4 I Selected the required columns
```

```
selected_columns <- c("Date", "DAILY_AQI_VALUE", "Site.Name", "AQS_PARAMETER_DESC", "COUNTY", "SITE_LAT

EPAair_O3_NC2018_processed <- EPAair_O3_NC2018_raw %>% select(all_of(selected_columns))
EPAair_O3_NC2019_processed <- EPAair_O3_NC2019_raw %>% select(all_of(selected_columns))
EPAair_PM25_NC2018_processed <- EPAair_PM25_NC2018_raw %>% select(all_of(selected_columns))
EPAair_PM25_NC2019_processed <- EPAair_PM25_NC2019_raw %>% select(all_of(selected_columns))


#5. Then Set all AQS_PARAMETER_DESC values to "PM2.5" for PM2.5 datasets

EPAair_PM25_NC2018_processed$AQS_PARAMETER_DESC <- "PM2.5"
EPAair_PM25_NC2019_processed$AQS_PARAMETER_DESC <- "PM2.5"

#6  And here I Saved the processed data sets

write.csv(EPAair_O3_NC2018_processed, here("./Data/Processed/EPAair_O3_NC2018_processed.csv"), row.name
write.csv(EPAair_O3_NC2019_processed, here("./Data/Processed/EPAair_O3_NC2019_processed.csv"), row.name
write.csv(EPAair_PM25_NC2018_processed, here("./Data/Processed/EPAair_PM25_NC2018_processed.csv"), row.
write.csv(EPAair_PM25_NC2019_processed, here("./Data/Processed/EPAair_PM25_NC2019_processed.csv"), row.
```

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 Here I combined the four datasets

combined_data <- bind_rows(
  EPAair_O3_NC2018_processed,
  EPAair_O3_NC2019_processed,
  EPAair_PM25_NC2018_processed,
  EPAair_PM25_NC2019_processed
)
#Beacuse I had problem in results of 10th question so i checked the dimensions of the combined data set

dim(combined_data)
```

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

```r
#8  Here i define the list of common sites.
library(dplyr)

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")

# I Wrangle the combined data here.

final_data <- combined_data %>%
  filter(Site.Name %in% common_sites) %>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarise(
    DAILY_AQI_VALUE = mean(DAILY_AQI_VALUE, na.rm = TRUE),
    SITE_LATITUDE = mean(SITE_LATITUDE, na.rm = TRUE),
    SITE_LONGITUDE = mean(SITE_LONGITUDE, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  mutate(
    Month = month(Date),
    Year = year(Date)
  )


#9 Spread the datasets

final_tidy_data <- final_data %>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC,
    values_from = DAILY_AQI_VALUE,
    values_fill = list(DAILY_AQI_VALUE = NA)
  )

#10
dim(final_tidy_data)
```

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

```r
#11
write.csv(final_tidy_data, here("./Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv"), row.names = FA
```

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 Here I group data  by site, month, and year, also calculated mean AQI for ozone and PM2.5

summary_data <- final_tidy_data %>%
  group_by(Site.Name, Year, Month) %>%
  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)


#13

dim(summary_data)
```

```
## [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 use drop_na instead of na.omit because drop_na selectively removes rows with missing values only in specified columns, allowing us to retain valuable data for other columns. In contrast, na.omit removes any row with at least one missing value across all columns, which can significantly reduce the dataset size and potentially discard useful information. This selective approach helps maintain a larger and more informative dataset for analysis, especially when some columns may have missing values that are not critical.