Assignment 4: Data Wrangling (Fall 2024)

Yufan Du

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 Load the packages
library(tidyverse)
library(lubridate)
library(here)

#1b  check working directory
here()
```

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

```
#1c
# Read datasets without converting to factors initially
epa_o3_2018 <- read_csv(here("Data", "Raw", "EPAair_O3_NC2018_raw.csv"))
epa_o3_2019 <- read_csv(here("Data", "Raw", "EPAair_O3_NC2019_raw.csv"))
epa_pm25_2018 <- read_csv(here("Data", "Raw", "EPAair_PM25_NC2018_raw.csv"))
```

```r
epa_pm25_2019 <- read_csv(here("Data", "Raw", "EPAair_PM25_NC2019_raw.csv"))

# Convert Date columns to date format (I Have to move 3 here, otherwise it keeps give me warning)
epa_o3_2018 <- epa_o3_2018 %>%
  mutate(Date = mdy(Date))

epa_o3_2019 <- epa_o3_2019 %>%
  mutate(Date = mdy(Date))

epa_pm25_2018 <- epa_pm25_2018 %>%
  mutate(Date = mdy(Date))

epa_pm25_2019 <- epa_pm25_2019 %>%
  mutate(Date = mdy(Date))

# Convert all other character columns to factors
epa_o3_2018 <- epa_o3_2018 %>%
  mutate(across(where(is.character), as.factor))

epa_o3_2019 <- epa_o3_2019 %>%
  mutate(across(where(is.character), as.factor))

epa_pm25_2018 <- epa_pm25_2018 %>%
  mutate(across(where(is.character), as.factor))

epa_pm25_2019 <- epa_pm25_2019 %>%
  mutate(across(where(is.character), as.factor))


#2  Display the dimensions (rows and columns) of each dataset
dim(epa_o3_2018)  # Dimensions of Ozone data for 2018
```

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

```r
dim(epa_o3_2019)  # Dimensions of Ozone data for 2019
```

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

```r
dim(epa_pm25_2018)  # Dimensions of PM2.5 data for 2018
```

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

```r
dim(epa_pm25_2019)  # Dimensions of PM2.5 data for 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, it does.

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 I moved it to 1c. Otherwise, I keep getting empty Data column.


#4 Select the required columns
epa_o3_2018 <- epa_o3_2018 %>%
  select(Date, DAILY_AQI_VALUE, `Site Name`, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

epa_o3_2019 <- epa_o3_2019 %>%
  select(Date, DAILY_AQI_VALUE, `Site Name`, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

epa_pm25_2018 <- epa_pm25_2018 %>%
  select(Date, DAILY_AQI_VALUE, `Site Name`, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

epa_pm25_2019 <- epa_pm25_2019 %>%
  select(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" for the PM2.5 datasets
epa_pm25_2018 <- epa_pm25_2018 %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

epa_pm25_2019 <- epa_pm25_2019 %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

#6 Save each dataset with the updated file names
write_csv(epa_o3_2018, here("Data","Processed", "EPAair_O3_NC2018_processed.csv"))
write_csv(epa_o3_2019, here("Data","Processed", "EPAair_O3_NC2019_processed.csv"))
write_csv(epa_pm25_2018, here("Data","Processed", "EPAair_PM25_NC2018_processed.csv"))
write_csv(epa_pm25_2019, here("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 the datasets with rbind
epa_combined <- rbind(epa_o3_2018, epa_o3_2019, epa_pm25_2018, epa_pm25_2019)

#8  Wrangle the dataset
epa_combined <- epa_combined %>%

  # Filter for the specific site names
  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 description, and county
  group_by(Date, `Site Name`, AQS_PARAMETER_DESC, COUNTY) %>%

  # Summarize to get the daily mean for AQI, latitude, and longitude
  summarize(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") %>%

  # Add Month and Year columns
  mutate(Month = month(Date),
         Year = year(Date)) %>%

  # Ungroup the data
  ungroup()

#check with the hint
dim(epa_combined)
```

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

```
#9 Spread the dataset
epa_tidy <- epa_combined %>%
  pivot_wider(names_from = AQS_PARAMETER_DESC, values_from = DAILY_AQI_VALUE)
```

```r
#10 Check the dimensions
dim(epa_tidy)
```

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

```r
#11 Save the processed dataset
write_csv(epa_tidy, here("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.

```r
#12 Generate a summary data frame, grouped by site, month, and year
summary_data <- epa_tidy %>%
  group_by(`Site Name`, Month, Year) %>%
  summarise(
    mean_Ozone = mean(Ozone, na.rm = TRUE),
    mean_PM25 = mean(PM2.5, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  drop_na(mean_Ozone)  # Remove instances where ozone mean values are missing
```

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

```r
#13 Check the dimensions of the summary dataset
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: When I replace `drop_na` with `na.omit`, the dimensions changed from "239 5" to "223 5". This is because `na.omit` ignore more value. Using drop_na(mean_AQI_Ozone) removes only rows where ozone data is missing, which is what we want. Using na.omit() will also remove rows with missing PM2.5 data, which is unnecessary and will result in fewer rows than intended.