Assignment 4: Data Wrangling (Fall 2024)

Rosie Wu

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
library(tidyverse)
library(lubridate)
library(here) #The here package allows for better control of relative paths

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

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

```
#1c read all excel files
EPAO3_2018 <- read.csv(
  file=here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE)
```

```
EPAO3_2019 <- read.csv(
  file=here("Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

EPAPM25_2018 <- read.csv(
  file=here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE)

EPAPM25_2019 <- read.csv(
  file=here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE)

#2 check for the dimensions of the data sets, which are rows*columns
dim(EPAO3_2018)
```

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

```
dim(EPAO3_2019)
```

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

```
dim(EPAPM25_2018)
```

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

```
dim(EPAPM25_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? Answer: They all have same number of columns, which is 20, while they all have different number of 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".

```
#3 converting all the Date columns to date objects formatting
EPAO3_2018$Date <- as.Date(EPAO3_2018$Date, format = "%m/%d/%Y")
EPAO3_2019$Date <- as.Date(EPAO3_2019$Date, format = "%m/%d/%Y")
EPAPM25_2018$Date <- as.Date(EPAPM25_2018$Date, format = "%m/%d/%Y")
```

```
EPAPM25_2019$Date <- as.Date(EPAPM25_2019$Date, format = "%m/%d/%Y")

#4 Select columns: Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
# SITE_LATITUDE, SITE_LONGITUDE

EPAO3_2018_q4 <- EPAO3_2018[c("Date", "DAILY_AQI_VALUE", "Site.Name",
            "AQS_PARAMETER_DESC", "COUNTY", "SITE_LATITUDE", "SITE_LONGITUDE")]
EPAO3_2019_q4 <- EPAO3_2019[c("Date", "DAILY_AQI_VALUE", "Site.Name",
            "AQS_PARAMETER_DESC", "COUNTY", "SITE_LATITUDE", "SITE_LONGITUDE")]
EPAPM25_2018_q4 <- EPAPM25_2018[c("Date", "DAILY_AQI_VALUE", "Site.Name",
            "AQS_PARAMETER_DESC", "COUNTY", "SITE_LATITUDE", "SITE_LONGITUDE")]
EPAPM25_2019_q4 <- EPAPM25_2019[c("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":
EPAPM25_2018_q4$AQS_PARAMETER_DESC <- "PM2.5"
EPAPM25_2019_q4$AQS_PARAMETER_DESC <- "PM2.5"

#6 Save all four processed datasets in the Processed folder, keep all files
target_directory <- "/home/guest/EDE_Fall2024/Data/Processed"
write.csv(EPAO3_2018_q4, file.path(target_directory,
                          "EPAair_O3_NC2018_processed.csv"), row.names = FALSE)
write.csv(EPAO3_2019_q4, file.path(target_directory,
                          "EPAair_O3_NC2019_processed.csv"), row.names = FALSE)
write.csv(EPAPM25_2018_q4, file.path(target_directory,
                         "EPAair_PM25_NC2018_processed.csv"), row.names = FALSE)
write.csv(EPAPM25_2019_q4, file.path(target_directory,
                         "EPAair_PM25_NC2019_processed.csv"), row.names = FALSE)
```

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 all of the four processed data sets, which all have 7 columns
combined_set = rbind(EPAO3_2018_q4, EPAO3_2019_q4, EPAPM25_2018_q4,
                     EPAPM25_2019_q4)

#8 filter names in the Site.Name column
combined_new_set <-
  filter(combined_set, 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")) %>%

  # 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.
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarize(
    mean_AQI = mean(DAILY_AQI_VALUE),
    mean_latitude = mean(SITE_LATITUDE),
    mean_longitude = mean(SITE_LONGITUDE)) %>%
  # Add columns for "Month" and "Year" by parsing your "Date" column
  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 Spread data sets so AQI values ozone and PM2.5 are in separate columns
# name: AQS names (PM2.5 and ozone), values are AQI daily
reshaped_set <- combined_new_set %>%
  pivot_wider(names_from =  AQS_PARAMETER_DESC,
    values_from = mean_AQI)

#10 show the dimension of the new dataset
dim(reshaped_set)
```

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

```
#11 save the current dataset to csv
write.csv(reshaped_set, file.path(target_directory,
                    "EPAair_O3_PM25_NC1819_Processed.csv"), row.names = FALSE)
```

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 summary datset: Data should be grouped by site, month, and year
# mean AQI values for ozone and PM2.5 for each group
# use drop_na to remove rows that "ozone" values are not available
summary_set <- reshaped_set %>%
  group_by(Year, Site.Name, Month) %>%
  summarize(mean_Ozone = mean(Ozone),
            mean_PM2.5 = mean(PM2.5)) %>%
  drop_na(mean_Ozone)
```

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

```
#13 Call dimensions of the summary dataset
dim(summary_set)
```

```
## [1] 182   5
```

```
# try na.omit function rather than drop_na to observe
summary_set_try <- reshaped_set %>%
  group_by(Year, Site.Name, Month) %>%
  summarize(mean_Ozone = mean(Ozone),
            mean_PM2.5 = mean(PM2.5)) %>%
  na.omit(mean_Ozone)
```

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

```
dim(summary_set_try)
```

```
## [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: After trying with na.omit and viewing the two summary datasets, it shows that drop_na for ozone will only drop rows with only "na" in Ozone column, whereas na.omit omits rows where there are NA present in general, including the PM2.5 column. Therefore, there are much less rows in the summary dataset after na.omit than the one after drop_na.