Assignment 4: Data Wrangling (Fall 2024)

Chrissie Pantoja

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.

```
# Load necessary packages
library(tidyverse) #for data manipulation and visualization: filter, select, mutate, summarize, etc.
library(lubridate) # for date and time manipulation: partse_date_time()
library(here) # for project-oriented file path management: here()
```

1b. Check your working directory.

```
# Check the current working directory
getwd()
```

```
## [1] "/Users/chrissiepantoja/Library/CloudStorage/OneDrive-DukeUniversity/PHD DUKE/1 COURSES/3 FALL S
```

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).

```
# Read in the raw data files
file1 <- read.csv(here("Data/Raw/EPAair_O3_NC2018_raw.csv"), stringsAsFactors = TRUE)
file2 <- read.csv(here("Data/Raw/EPAair_O3_NC2019_raw.csv"), stringsAsFactors = TRUE)
file3 <- read.csv(here("Data/Raw/EPAair_PM25_NC2018_raw.csv"), stringsAsFactors = TRUE)
file4 <- read.csv(here("Data/Raw/EPAair_PM25_NC2019_raw.csv"), stringsAsFactors = TRUE)
```

2. Add the appropriate code to reveal the dimensions of the four datasets.

```
# Display the dimensions of each dataset
cat("Dimensions of file1: ", dim(file1)[1], " rows and ", dim(file1)[2], " columns\n")
```

```
## Dimensions of file1:  9737  rows and  20  columns
```

```
cat("Dimensions of file2: ", dim(file2)[1], " rows and ", dim(file2)[2], " columns\n")
```

```
## Dimensions of file2:  10592  rows and  20  columns
```

```
cat("Dimensions of file3: ", dim(file3)[1], " rows and ", dim(file3)[2], " columns\n")
```

```
## Dimensions of file3:  8983  rows and  20  columns
```

```
cat("Dimensions of file4: ", dim(file4)[1], " rows and ", dim(file4)[2], " columns\n")
```

```
## Dimensions of file4:  8581  rows and  20  columns
```

All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern? Yes, they follow this pattern.

Wrangle individual datasets to create processed files.

3. Change the Date columns to be date objects.

```
# Function to convert date strings to Date format (MM-DD-YYY)

file1$Date <- as.Date(file1$Date, format = "%m/%d/%Y")
file2$Date <- as.Date(file2$Date, format = "%m/%d/%Y")
file3$Date <- as.Date(file3$Date, format = "%m/%d/%Y")
file4$Date <- as.Date(file4$Date, format = "%m/%d/%Y")
```

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

```
# Select specific columns with the correct column names
file1.subsample <- select(file1, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATI
file2.subsample <- select(file2, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATI
file3.subsample <- select(file3, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATI
file4.subsample <- select(file4, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATI
```

5. For the PM2.5 datasets, fill all cells in AQS_PARAMETER_DESC with "PM2.5" (all cells in this column should be identical).

```
# Fill AQS_PARAMETER_DESC with "PM2.5" for the PM2.5 datasets
file3.subsample$AQS_PARAMETER_DESC <- "PM2.5"
file4.subsample$AQS_PARAMETER_DESC <- "PM2.5"
```

6. Save all four processed datasets in the Processed folder. Use the same file names as the raw files but replace "raw" with "processed".

```
# Save processed datasets
write_csv(file1.subsample, here("Data/Processed/EPAair_O3_NC2018_processed.csv"))
write_csv(file2.subsample, here("Data/Processed/EPAair_O3_NC2019_processed.csv"))
write_csv(file3.subsample, here("Data/Processed/EPAair_PM25_NC2018_processed.csv"))
write_csv(file4.subsample, 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.

```
# Ensure column names are identical
colnames(file1.subsample)
```

```
## [1] "Date"               "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(file2.subsample)
```

```
## [1] "Date"               "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(file3.subsample)
```

```
## [1] "Date"               "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(file4.subsample)
```

```
## [1] "Date"               "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
# Combine the datasets
combined_data <- rbind(file1.subsample, file2.subsample, file3.subsample, file4.subsample)
```

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.

```
#Checking the common sites
common_sites <- Reduce(intersect, list(file1.subsample$Site.Name, file2.subsample$Site.Name, file3.subsa
#print(common_sites)

# Wrangling dataset
df_processed <- combined_data %>%
  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_DESC, COUNTY) %>%
  summarise(
    meanAQI = mean(DAILY_AQI_VALUE, na.rm = TRUE),
    meanLat = mean(SITE_LATITUDE, na.rm = TRUE),
    meanLong = mean(SITE_LONGITUDE, na.rm = TRUE),
    .groups = 'drop'  # This avoids the warning about grouped output
  ) %>%
  mutate(
    Month = month(Date),
    Year = year(Date)
  )

# Print the dimensions of the processed data frame
cat("The dataset has", dim(df_processed)[1], "rows and", dim(df_processed)[2], "columns.\n")
```

```
## The dataset has 14752 rows and 9 columns.
```

```
# Print the processed data frame to verify
print(df_processed)
```

```
## # A tibble: 14,752 x 9
##    Date       Site.Name AQS_PARAMETER_DESC COUNTY meanAQI meanLat meanLong Month
##    <date>     <fct>     <fct>              <fct>    <dbl>   <dbl>    <dbl> <dbl>
##  1 2018-01-01 Bryson C~ PM2.5              Swain       35    35.4    -83.4     1
##  2 2018-01-01 Castle H~ PM2.5              New H~      13    34.4    -77.8     1
##  3 2018-01-01 Clemmons~ PM2.5              Forsy~      24    36.0    -80.3     1
##  4 2018-01-01 Durham A~ PM2.5              Durham      31    36.0    -78.9     1
##  5 2018-01-01 Garinger~ Ozone              Meckl~      32    35.2    -80.8     1
##  6 2018-01-01 Garinger~ PM2.5              Meckl~      20    35.2    -80.8     1
##  7 2018-01-01 Hattie A~ PM2.5              Forsy~      22    36.1    -80.2     1
##  8 2018-01-01 Leggett   PM2.5              Edgec~      14    36.0    -77.6     1
##  9 2018-01-01 Millbroo~ Ozone              Wake        34    35.9    -78.6     1
## 10 2018-01-01 Millbroo~ PM2.5              Wake        28    35.9    -78.6     1
## # i 14,742 more rows
## # i 1 more variable: Year <dbl>
```

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.

```
wide_df <- pivot_wider(df_processed,
    names_from = AQS_PARAMETER_DESC,
    values_from = meanAQI,
  )

print(wide_df)
```

```
## # A tibble: 8,976 x 9
##    Date       Site.Name          COUNTY meanLat meanLong Month  Year PM2.5 Ozone
##    <date>     <fct>              <fct>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 2018-01-01 Bryson City        Swain     35.4    -83.4     1  2018    35    NA
##  2 2018-01-01 Castle Hayne       New H~    34.4    -77.8     1  2018    13    NA
##  3 2018-01-01 Clemmons Middle    Forsy~    36.0    -80.3     1  2018    24    NA
##  4 2018-01-01 Durham Armory      Durham    36.0    -78.9     1  2018    31    NA
##  5 2018-01-01 Garinger High Sch~ Meckl~    35.2    -80.8     1  2018    20    32
##  6 2018-01-01 Hattie Avenue      Forsy~    36.1    -80.2     1  2018    22    NA
##  7 2018-01-01 Leggett            Edgec~    36.0    -77.6     1  2018    14    NA
##  8 2018-01-01 Millbrook School   Wake      35.9    -78.6     1  2018    28    34
##  9 2018-01-01 Pitt Agri. Center  Pitt      35.6    -77.4     1  2018    15    NA
## 10 2018-01-01 West Johnston Co.  Johns~    35.6    -78.5     1  2018    24    NA
## # i 8,966 more rows
```

10. Call up the dimensions of your new tidy dataset.

```
# Print the dimensions
cat("The new dataset has", dim(wide_df)[1], "rows and", dim(wide_df)[2], "columns.\n")
```

```
## The new dataset has 8976 rows and 9 columns.
```

11. Save your processed dataset with the following file name: “EPAair_O3_PM25_NC1819_Processed.csv”

```
write.csv(wide_df, 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.

```
# Generate the summary data frame
summary1_df <- wide_df %>%
  group_by(Site.Name, Month, Year) %>%
  summarize(
    mean_AQI_ozone = mean(Ozone, na.rm = TRUE),
    mean_AQI_PM2.5 = mean(PM2.5, na.rm = TRUE)) %>%
  drop_na(mean_AQI_ozone)
```

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

```
print(summary1_df)
```

```
## # A tibble: 239 x 5
## # Groups:   Site.Name, Month [127]
##    Site.Name   Month  Year mean_AQI_ozone mean_AQI_PM2.5
##    <fct>       <dbl> <dbl>          <dbl>          <dbl>
##  1 Bryson City     2  2019           32.4           33.0
##  2 Bryson City     3  2018           41.6           34.7
##  3 Bryson City     3  2019           42.5           32.5
##  4 Bryson City     4  2018           44.5           28.2
##  5 Bryson City     4  2019           45.4           26.7
##  6 Bryson City     5  2018           35.9           33.5
##  7 Bryson City     5  2019           39.6           31.8
##  8 Bryson City     6  2018           37.8           25.1
##  9 Bryson City     6  2019           34.0           31.0
## 10 Bryson City     7  2018           34.6           34.3
## # i 229 more rows
```

```
# Generate the summary data frame
summary2_df <- wide_df %>%
  group_by(Site.Name, Month, Year) %>%
  summarize(
    mean_AQI_ozone = mean(Ozone, na.rm = TRUE),
    mean_AQI_PM2.5 = mean(PM2.5, na.rm = TRUE)) %>%
  na.omit(mean_AQI_ozone)
```

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

```
print(summary2_df)
```

```
## # A tibble: 223 x 5
## # Groups:   Site.Name, Month [127]
##    Site.Name   Month  Year mean_AQI_ozone mean_AQI_PM2.5
##    <fct>       <dbl> <dbl>          <dbl>          <dbl>
##  1 Bryson City     2  2019           32.4           33.0
##  2 Bryson City     3  2018           41.6           34.7
##  3 Bryson City     3  2019           42.5           32.5
##  4 Bryson City     4  2018           44.5           28.2
##  5 Bryson City     4  2019           45.4           26.7
##  6 Bryson City     5  2018           35.9           33.5
##  7 Bryson City     5  2019           39.6           31.8
##  8 Bryson City     6  2018           37.8           25.1
##  9 Bryson City     6  2019           34.0           31.0
## 10 Bryson City     7  2018           34.6           34.3
## # i 213 more rows
```

```
# Print the dimensions
cat("The dataset with na.omit has", dim(summary2_df)[1], "rows and", dim(summary2_df)[2], "columns.\n")
```

```
## The dataset with na.omit has 223 rows and 5 columns.
```

13. Call up the dimensions of the summary dataset.

```
# Print the dimensions
cat("The dataset has", dim(summary1_df)[1], "rows and", dim(summary1_df)[2], "columns.\n")
```

```
## The dataset has 239 rows and 5 columns.
```

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: Using drop_na(mean_AQI_ozone) removes only rows with missing mean_AQI_ozone values, preserving rows with valid mean_AQI_PM2.5 values. In contrast, na.omit removes any row with NA in any column, leading to more rows being dropped (i.e., 16 rows dropped in this exercise).