Assignment 4: Data Wrangling (Fall 2024)

Jingze Dai

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 loading libraries

library(tidyverse)
library(lubridate)
library(here)

#1b

# checking the working directory
getwd()
```

```
## [1] "/home/guest/ENV872/EDE_Fall2024"
```

```r
# checking the top-level directory
here()
```

```
## [1] "/home/guest/ENV872/EDE_Fall2024"
```

```r
#1c importing datasets

# Ozone 2018 data
o3_2018 <- read.csv(file=here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
                      stringsAsFactors = TRUE)

# Ozone 2019 data
o3_2019 <- read.csv(file=here("Data/Raw/EPAair_O3_NC2019_raw.csv"),
                      stringsAsFactors = TRUE)

# PM2.5 2018 data
pm25_2018 <- read.csv(file=here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
                        stringsAsFactors = TRUE)

# PM2.5 2019 data
pm25_2019 <- read.csv(file=here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
                        stringsAsFactors = TRUE)

#2

# revealing dimensions of the datasets
dim(o3_2018)
```

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

```r
dim(o3_2019)
```

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

```r
dim(pm25_2018)
```

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

```r
dim(pm25_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, they all have 20 columns but different rows. O3 in 2018, O3 in 2019, PM2.5 in 2018 and PM2.5 in 2019 have 9737, 10592, 8983 and 8581 rows, respectively.

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".

```r
#3 changing date columns from factors to dates using mdy function
o3_2018$Date <- mdy(o3_2018$Date)
o3_2019$Date <- mdy(o3_2019$Date)
pm25_2018$Date <- mdy(pm25_2018$Date)
pm25_2019$Date <- mdy(pm25_2019$Date)

#4 selecting the targeted columns

selected_o3_2018 <- o3_2018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE)

selected_o3_2019 <- o3_2019 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE)

selected_pm25_2018 <- pm25_2018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE)

selected_pm25_2019 <- pm25_2019 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE)

#5 filling AQS_parameter_desc with PM2.5 for those dataframes

selected_pm25_2018 <- selected_pm25_2018 %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

selected_pm25_2019 <- selected_pm25_2019 %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

#6 saving dataframes as 'processed'

write.csv(selected_o3_2018, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2018_processed.csv")
write.csv(selected_o3_2019, row.names = FALSE,
          file = "./Data/Processed/EPAair_O3_NC2019_processed.csv")
write.csv(selected_pm25_2018, row.names = FALSE,
          file = "./Data/Processed/EPAair_PM25_NC2018_processed.csv")
write.csv(selected_pm25_2019, 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"

```
#7 combining datasets

# making sure the column names are identical
colnames(selected_o3_2018)
```

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

```
colnames(selected_o3_2019)
```

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

```
colnames(selected_pm25_2018)
```

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

```
colnames(selected_pm25_2019)
```

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

```r
# column names are the same, using rbind to combine datasets
EPAair_data <- rbind(selected_o3_2018, selected_o3_2019,
                     selected_pm25_2018, selected_pm25_2019)

# checking the summary of combined data
summary(EPAair_data)
```

```
##       Date            DAILY_AQI_VALUE                  Site.Name
##  Min.   :2018-01-01   Min.   :  0.00   Millbrook School    : 2169
##  1st Qu.:2018-06-27   1st Qu.: 27.00   Garinger High School: 1818
##  Median :2019-01-06   Median : 36.00   Hattie Avenue       : 1432
##  Mean   :2018-12-26   Mean   : 36.27   Durham Armory       : 1405
##  3rd Qu.:2019-06-23   3rd Qu.: 45.00   Pitt Agri. Center   : 1303
##  Max.   :2019-12-31   Max.   :136.00   Clemmons Middle     : 1261
##                                        (Other)             :28505
##  AQS_PARAMETER_DESC         COUNTY      SITE_LATITUDE   SITE_LONGITUDE
##  Ozone:20329        Mecklenburg: 3903   Min.   :34.36   Min.   :-83.80
##  PM2.5:17564        Forsyth    : 3175   1st Qu.:35.26   1st Qu.:-81.37
##                     Wake       : 2846   Median :35.64   Median :-80.23
##                     Cumberland : 1795   Mean   :35.62   Mean   :-80.21
##                     Haywood    : 1672   3rd Qu.:35.99   3rd Qu.:-78.77
##                     Swain      : 1628   Max.   :36.51   Max.   :-76.21
##                     (Other)    :22874
```

```r
# we can confirm that there are 20329 rows for ozone (10592 + 9737)
# there are 17564 rows for PM2.5 (8983 + 8581)

#8 wrangling the combined dataset

# creating the list of sites to filter instead of using intersect function
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")

# creating pipe function
EPAair_processed <- EPAair_data %>%
  # selecting 'Site.Name' that appears in the list
  filter(Site.Name %in% sites) %>%

  # grouping by 'Date' 'Site.Name' 'AQS_PARAMETER_DESC' 'COUNTY'
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%

  # calculating the mean of 'DAILY_AQI_VALUE' 'SITE_LATITUDE' 'SITE_LONGITUDE'
  summarise(
    DAILY_AQI_VALUE = mean(DAILY_AQI_VALUE),
    SITE_LATITUDE = mean(SITE_LATITUDE),
    SITE_LONGITUDE = mean(SITE_LONGITUDE)) %>%

  # adding month and year columns
  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.
```

```
# we can confirm that the dimension of the dataset is indeed 14,752 x 9
dim(EPAair_processed)
```

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

```
#9 spreading the dataset using pivot_wider function

EPAair_processed <- EPAair_processed %>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC,
    values_from = DAILY_AQI_VALUE)

#10 calling dimensions of new dataset

dim(EPAair_processed)
```

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

```
#11 exporting as processed data

write.csv(EPAair_processed, 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

summaries_data <- EPAair_processed %>%
  # grouping by 'Site.Name' 'Month' 'Year'
  group_by(Site.Name, Month, Year) %>%

  # calculating mean AQI values
  summarise(
    mean_Ozone_AQI = mean(Ozone),
    mean_PM2.5_AQI = mean(PM2.5)) %>%

  drop_na(mean_Ozone_AQI)
```

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

```
#13

dim(summaries_data)
```

```
## [1] 182   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.

```
summaries_2 <- EPAair_processed %>%
  # grouping by 'Site.Name' 'Month' 'Year'
  group_by(Site.Name, Month, Year) %>%

  # calculating mean AQI values
  summarise(
    mean_Ozone_AQI = mean(Ozone),
    mean_PM2.5_AQI = mean(PM2.5)) %>%

  na.omit(mean_Ozone_AQI)
```

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

```
dim(summaries_2)
```

```
## [1] 101   5
```

Answer: We can see that after using na.omit, the dimension of the summary data dropped from 182 rows to 101 rows. This is because that drop_na() function allows us to specify which columns to drop the missing values and it only removes empty rows in that specified column (i.e. O3 column in this case. However, the na.omit function removes row that contains missing value in any columns, and cannot selectively remove NA from specified columns, thus in this case, rows with NA in pm2.5 are also removed, resulting in data loss