Assignment 4: Data Wrangling (Fall 2024)

Ana Andino

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
getwd()
```

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

```
#1c

#Read in the data
EPA.2018.data <- read.csv(
  file= here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
```

```r
  stringsAsFactors = TRUE
)

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

EPA.PM25.2018 <- read.csv(
  file = here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE
)

EPA.PM25.2019 <- read.csv(
  file = here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE
)

#2
dim(EPA.2018.data)
```

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

```r
dim(EPA.2019.data)
```

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

```r
dim(EPA.PM25.2019)
```

```
## [1] 8581   20
```

```r
dim(EPA.PM25.2018)
```

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

```r
#another way of doing it with a shorter code:
epa.datasets <- list(EPA.PM25.2019, EPA.PM25.2018, EPA.2018.data, EPA.2019.data)
lapply(epa.datasets, dim)
```

```
## [[1]]
## [1] 8581   20
##
## [[2]]
## [1] 8983   20
##
## [[3]]
## [1] 9737   20
##
## [[4]]
## [1] 10592    20
```

> All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern? They do. All of them have 20 columns but they all differ in row dimensions.

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 Change date format
class(EPA.2018.data$Date)
```

```
## [1] "factor"
```

```
class(EPA.PM25.2019$Date)
```

```
## [1] "factor"
```

```
#All four datasets have Dates as factors. This cold also be checked looking
#at the environment

EPA.2018.data$Date <- as.Date(EPA.2018.data$Date, format = "%m/%d/%Y")
EPA.2019.data$Date <- as.Date(EPA.2019.data$Date, format = "%m/%d/%Y")
EPA.PM25.2018$Date <- as.Date(EPA.PM25.2018$Date, format = "%m/%d/%Y")
EPA.PM25.2019$Date <- as.Date(EPA.PM25.2019$Date, format = "%m/%d/%Y")
#CHECK FORMAT
class(EPA.2018.data$Date)
```

```
## [1] "Date"
```

```
class(EPA.2019.data$Date)
```

```
## [1] "Date"
```

```
class(EPA.PM25.2018$Date)
```

```
## [1] "Date"
```

```
class(EPA.PM25.2019$Date)
```

```
## [1] "Date"
```

```
#In a shorter way:
epa.datasets <- lapply(epa.datasets, function(x) {
  x$Date <- as.Date(x$Date, format = "%m/%d/%Y")
  return(x)
})
```

```r
#4 Selecting columns
EPA.2018.data.select <- select(
  EPA.2018.data, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC,
  COUNTY:SITE_LONGITUDE
  )

EPA.2019.data.select <- select(
  EPA.2019.data, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC,
  COUNTY:SITE_LONGITUDE
  )

EPA.PM25.2018.select <- select(
  EPA.PM25.2018, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC,
  COUNTY:SITE_LONGITUDE
  )

EPA.PM25.2019.select <- select(
  EPA.PM25.2019, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC,
  COUNTY:SITE_LONGITUDE
  )


#5 Changing row values

EPA.PM25.2018$AQS_PARAMETER_DESC <- "PM2.5"
EPA.PM25.2019$AQS_PARAMETER_DESC <- "PM2.5"

#FOR THE PROCESSED DATASETS:
EPA.PM25.2018.select$AQS_PARAMETER_DESC <- "PM2.5"
EPA.PM25.2019.select$AQS_PARAMETER_DESC <- "PM2.5"

#6 Save new datasets.
write.csv(EPA.2018.data.select, row.names = FALSE,
          file = "Data/Processed/EPAair_O3_NC2018_Processed.csv")
write.csv(EPA.2019.data.select, row.names = FALSE,
          file = "Data/Processed/EPAair_O3_NC2019_Processed.csv")
write.csv(EPA.PM25.2018.select, row.names = FALSE,
          file = "Data/Processed/EPAair_PM25_NC2018_Processed.csv")
write.csv(EPA.PM25.2019.select, 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 combine datasets
EPA.combined <- rbind(
  EPA.2018.data.select, EPA.2019.data.select,
  EPA.PM25.2018.select, EPA.PM25.2019.select)


#8
#Figuring out the common sites by myself#
common_sites <- Reduce(intersect, list(
  EPA.2018.data.select$Site.Name,
  EPA.2019.data.select$Site.Name,
  EPA.PM25.2018.select$Site.Name,
  EPA.PM25.2019.select$Site.Name
))
print(common_sites)
```

```
##  [1] "Linville Falls"      "Durham Armory"       "Leggett"
##  [4] "Hattie Avenue"       "Clemmons Middle"     "Mendenhall School"
##  [7] "Frying Pan Mountain" "West Johnston Co."   "Garinger High School"
## [10] "Castle Hayne"        "Pitt Agri. Center"   "Bryson City"
## [13] ""                    "Millbrook School"
```

```
#load packages
library(tidyverse)
library(dplyr)
EPA.combined.processed <-
  EPA.combined %>%
  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) %>%
```

```r
    summarise(meanAQI = mean(DAILY_AQI_VALUE, na.rm = TRUE),
              meanLAT= mean(SITE_LATITUDE, na.rm = TRUE),
              meanLON = mean(SITE_LONGITUDE, na.rm = TRUE))%>%
                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.
```

```r
#The result is a matriz 14,752 x 9

#9  Spread your data sets

EPA.combined.wide <- pivot_wider(EPA.combined.processed,
                                 names_from = AQS_PARAMETER_DESC,
                                 values_from = meanAQI
                                 )

#10 Dimensions of new dataset
dim(EPA.combined.wide)
```

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

```r
#The dimensions are 8,976 x 9

#11 Save

write.csv(EPA.combined.wide, 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.

```r
#12
EPA.summaries <-
  EPA.combined.wide %>%
  group_by(Site.Name, Month, Year)%>%
  summarise(
    mean_ozone = mean(Ozone, na.rm = TRUE),
    mean_PM2.5 = mean(PM2.5, na.rm = TRUE)) %>%
  drop_na(mean_ozone)
```

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

```r
#13

dim(EPA.summaries)
```

```
## [1] 239   5
```

```r
#The dimensions are 239 x 5


EPA.summaries <-
  EPA.combined.wide %>%
  group_by(Site.Name, Month, Year)%>%
  summarise(
    mean_ozone = mean(Ozone, na.rm = TRUE),
    mean_PM2.5 = mean(PM2.5, na.rm = TRUE)) %>%
  na.omit(mean_ozone)
```

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

```r
dim(EPA.summaries)
```

```
## [1] 223   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 you use na.omit fucntion, there are even less rows because drop.na only drops rows based on the missing values in the column I specify which in this case is Mean_Ozone. However, na.omit, drops rows in any column that contain NA values which would give a smaller data frame as in this case.