Assignment 4: Data Wrangling (Fall 2024)

Victoria Thompson

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

#1b
#check WD
getwd()
```

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

```
#1c
#read in datasets
  air2018 <- read.csv(
```

```r
  file = here('Data','Raw','EPAair_O3_NC2018_raw.csv'),
  stringsAsFactors = T
)
  air2019 <- read.csv(
  file = here('Data','Raw','EPAair_O3_NC2019_raw.csv'),
  stringsAsFactors = T
)
  pm252018 <- read.csv(
  file = here('Data','Raw','EPAair_PM25_NC2018_raw.csv'),
  stringsAsFactors = T
)
  pm252019 <- read.csv(
  file = here('Data','Raw','EPAair_PM25_NC2019_raw.csv'),
  stringsAsFactors = T
)

#2
#get dimensions and print
dimension_air2018 <- dim(air2018)
print(dimension_air2018)
```

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

```r
dimension_air2019 <- dim(air2019)
print(dimension_air2019)
```

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

```r
dimension_pm252018 <- dim(pm252018)
print(dimension_pm252018)
```

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

```r
dimension_pm252019 <- dim(pm252019)
print(dimension_pm252019)
```

```
## [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 a varying 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
#changing factors to dates
air2018$Date <- mdy(air2018$Date)
air2019$Date <- mdy(air2019$Date)
pm252018$Date <- mdy(pm252018$Date)
pm252019$Date <- mdy(pm252019$Date)

#confirming classes
class(air2018$Date)
```

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

```
class(air2019$Date)
```

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

```
class(pm252018$Date)
```

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

```
class(pm252019$Date)
```

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

```
#4
#select data
air2018.v2 <-
  air2018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

air2019.v2 <-
  air2019 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

pm252018.v2 <-
  pm252018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

pm252019.v2 <-
  pm252019 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

#5
#write over values so that all in the column are "PM2.5"
pm252018.v2$AQS_PARAMETER_DESC <- "PM2.5"
head(pm252018.v2$AQS_PARAMETER_DESC) #check to make sure values are all PM2.5
```

```
## [1] "PM2.5" "PM2.5" "PM2.5" "PM2.5" "PM2.5" "PM2.5"
```

```
pm252019.v2$AQS_PARAMETER_DESC <- "PM2.5"
head(pm252019.v2$AQS_PARAMETER_DESC) #check to make sure values are all PM2.5
```

```
## [1] "PM2.5" "PM2.5" "PM2.5" "PM2.5" "PM2.5" "PM2.5"
```

```
#6
#save csvs
  write.csv(air2018.v2,
  file = here('Data','Processed','EPAair_O3_NC2018_processed.csv'))

  write.csv(air2019.v2 ,
  file = here('Data','Processed','EPAair_O3_NC2019_processed.csv'))

  write.csv(pm252018.v2,
  file = here('Data','Processed','EPAair_PM25_NC2018_processed.csv'))

  write.csv(pm252019.v2,
  file = 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"

```r
#7
#using rbind to combine all datasets
alldata <- rbind(air2018.v2, air2019.v2, pm252018.v2,pm252019.v2)

#8

#storing shared sites
sharedsites <- 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"
)


# pipe function
airqualwrangled <- alldata %>%
  filter(Site.Name %in% sharedsites) %>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarise(meanAQI = mean(DAILY_AQI_VALUE),
            meanlat = mean(SITE_LATITUDE),
            meanlong = mean(SITE_LONGITUDE)) %>%
  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
# check dimensions of airqualwrangled to see if it matches 14,752 x 9
airqualwrangleddim <- dim(airqualwrangled)
print(airqualwrangleddim)
```

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

```r
#9

# spread data into separate columns using pivot_wider
airqualwrangled.spread <- pivot_wider(airqualwrangled, names_from = AQS_PARAMETER_DESC, values_from = me

#10
#check and print dimensions
airqualwrangled.spreadddim <- dim(airqualwrangled.spread)
print(airqualwrangled.spreadddim)
```

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

```r
#11
#save csv
write.csv(airqualwrangled.spread,
  file = 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.

```
#12
#split-apply-combine
airqualsummary <- airqualwrangled.spread %>%
  group_by(Site.Name, Month, Year) %>%
  summarise(meanAQIPM = mean(PM2.5),
            meanAQIOzone = mean(Ozone)) %>%
  drop_na(meanAQIOzone)
```

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

```
#13
#check and print dimensions
airqualsummarydim <- dim(airqualsummary)
print(airqualsummarydim)
```

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

```
#testing na.omit
airqualsummary2 <- airqualwrangled.spread %>%
 group_by(Site.Name, Month, Year) %>%
  summarise(meanAQIPM = mean(PM2.5),
            meanAQIOzone = mean(Ozone)) %>%
  na.omit(meanAQIOzone)
```

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

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: drop_na only removes rows within a set collumn that have an NA value. The collumn remains intact with all of the NA values removed. na.omit removes any rows that have at least one NA in any column. In context, this means that using na.omit would remove the entire row of data if an ozone collumn contains an NA.