Assignment 4: Data Wrangling (Fall 2024)

Fiona Price

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. Load the `tidyverse`,  `lubridate`, and `here` packages into your session.
library(tidyverse)
library(lubridate)
library(here)

#1b. Check working directory.
here()
```

```
## [1] "/home/guest/ede_fall2024"
```

```r
#1c. Read in all four raw data files associated with the EPA Air dataset, being sure to set string colu
#Check to see what the four files are.
filenames <- dir(here('Data','Raw'),pattern = ("EPAair*"))
print(filenames)
```

```
## [1] "EPAair_O3_NC2018_raw.csv"   "EPAair_O3_NC2019_raw.csv"
## [3] "EPAair_PM25_NC2018_raw.csv" "EPAair_PM25_NC2019_raw.csv"
```

```
#Load in the O3_2018 data.
EPAair_O3_2018 <- read.csv(
   file=here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = TRUE
)

#Load in the O3_2019 data.
EPAair_O3_2019 <- read.csv(
   file=here("Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = TRUE
)

#Load in the PM25_2018 data.
EPAair_PM25_2018 <- read.csv(
   file=here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors = TRUE
)

#Load in the PM25_2019 data.
EPAair_PM25_2019 <- read.csv(
   file=here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = TRUE
)

#2. Check the dimensions of the datasets.
dim(EPAair_O3_2018)
```

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

```
#The dimensions are 9737 rows and 20 columns.

dim(EPAair_O3_2019)
```

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

```
#The dimensions are 10592 rows and 20 columns.

dim(EPAair_PM25_2018)
```

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

```
#The dimensions are 8983 rows and 20 columns.

dim(EPAair_PM25_2019)
```

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

```
#The dimensions are 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, my datasets follow this pattern. See above comments for specific numbers 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
#Start with EPAair_03_2018.
EPAair_03_2018$Date <- as.Date(EPAair_03_2018$Date, format = "%m/%d/%Y")
#Check the class.
class(EPAair_03_2018$Date)
```

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

```
#Change the date format of EPAair_03_2019.
EPAair_03_2019$Date <- as.Date(EPAair_03_2019$Date, format = "%m/%d/%Y")
#Check the class.
class(EPAair_03_2019$Date)
```

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

```
#Change the date format of EPAair_PM25_2018.
EPAair_PM25_2018$Date <- as.Date(EPAair_PM25_2018$Date, format = "%m/%d/%Y")
#Check the class.
class(EPAair_PM25_2018$Date)
```

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

```
#Change the date format of EPAair_PM25_2019.
EPAair_PM25_2019$Date <- as.Date(EPAair_PM25_2019$Date, format = "%m/%d/%Y")
#Check the class.
class(EPAair_PM25_2019$Date)
```

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

```r
#4. Select the following columns: Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LA

#Start with EPAair_O3_2018.
EPAair_O3_2018_processed <- EPAair_O3_2018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

#Select the columns for EPAair_O3_2019.
EPAair_O3_2019_processed <- EPAair_O3_2019 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

#Select the columns for EPAair_PM25_2018.
EPAair_PM25_2018_processed <- EPAair_PM25_2018 %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SITE_LONGITUDE)

#Select the columns for EPAair_PM25_2019.
EPAair_PM25_2019_processed <- EPAair_PM25_2019 %>%
  select(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".
#Use the mutate function to do this.
EPAair_PM25_2018_processed <- EPAair_PM25_2018_processed %>% mutate(AQS_PARAMETER_DESC = "PM2.5")

EPAair_PM25_2019_processed <- EPAair_PM25_2019_processed %>% mutate(AQS_PARAMETER_DESC = "PM2.5")


#6. Save all four processed datasets in the Processed folder. Use the same file names as the raw files
#Save the EPAair_O3_2018_processed file.
write.csv(
  EPAair_O3_2018_processed,
  file = here("./Data/Processed/EPAair_O3_NC2018_Processed.csv"),
  row.names=FALSE)

#Save the EPAair_O3_2019_processed file.
write.csv(
  EPAair_O3_2019_processed,
  file = here("./Data/Processed/EPAair_O3_NC2019_Processed.csv"),
  row.names=FALSE)

#Save the EPAair_PM25_2018_processed file.
write.csv(
  EPAair_PM25_2018_processed,
  file = here("./Data/Processed/EPAair_PM25_2018_processed.csv"),
  row.names=FALSE)

#Save the EPAair_PM25_2019_processed file.
write.csv(
  EPAair_PM25_2019_processed,
  file = here("./Data/Processed/EPAair_PM25_2019_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. Use `rbind` to combine the datasets.
EPAair <- rbind(EPAair_O3_2018_processed, EPAair_O3_2019_processed,
                EPAair_PM25_2018_processed, EPAair_PM25_2019_processed)

#8.
#Check the intersection
common_elements <- Reduce(intersect, list(EPAair_O3_2018_processed$Site.Name,
                                          EPAair_O3_2019_processed$Site.Name,
                                          EPAair_PM25_2018_processed$Site.Name,
                                          EPAair_PM25_2019_processed$Site.Name))
common_elements
```

```
##  [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"
```

```
#Create list of desired sites (note: I am doing this before I wrangle the
#dataset to make that step cleaner).

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")

#Load the `lubridate` package (this will be used for parsing the date column).
library(lubridate)

EPAair <- EPAair %>%
  filter(Site.Name %in% sites) %>% #filters using the list I created above
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  summarise(meanAQI = mean(DAILY_AQI_VALUE),
            meanLat = mean(SITE_LATITUDE),
            meanLong = mean(SITE_LONGITUDE)) %>% #grouping and generating daily
  #means
  mutate(month = month(Date),
         year = year(Date)) #parsing the date column into month and year
```

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

```
#Check the dimensions of this dataset.
dim(EPAair)
```

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

```
#9. Spread your datasets such that AQI values for ozone and PM2.5 are in separate columns.
#Use the `pivot_wider` function to do this.
EPAair_tidy <- pivot_wider(EPAair, names_from = AQS_PARAMETER_DESC,
                      values_from = meanAQI)

#10. Find dimensions of new tidy datset.
dim(EPAair_tidy)
```

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

```
#There are 8976 rows and 9 columns.

#11. Save new tidy dataset.
write.csv(
  EPAair_tidy,
  file = here("./Data/Processed/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. Generate a summary data from of mean AQI values for ozone and PM2.5. Remove
#instances where mean ozone values are not available.
EPAair_summary <- EPAair_tidy %>%
  group_by(Site.Name, month, year) %>%
  summarize(meanOzone = mean(Ozone),
            meanPM2.5 = mean(PM2.5)) %>%
  drop_na(meanOzone)
```

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

```
#13. Find the dimensions of this summary dataset.
dim(EPAair_summary)
```

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

```
#There are 182 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: `na.omit` removes any rows from the dataframe that contain NAs. `drop_na`, however, just removes instances of NAs in the meanOzone column, but keeps NAs in the PM2.5 column. Essentially, `na.omit` got rid of more information than we wanted to get rid of here, because we only wanted to get rid of NAs in the ozone column, not the PM2.5 column.