Assignment 4: Data Wrangling (Fall 2024)

Ayden Schirmacher

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.

```
remove(list=ls())

#1a loading library packages
library(tidyverse) #these packages are already installed, so we just have to
#read them into the library
library(lubridate)
library(here)

#1b checking workspace
getwd()
```

```
## [1] "/home/guest/EDA_Spring 2025"
```

```
#1c importing files as dataframes
pm25_nc2019<-read.csv(
  file = here("Data/Raw/EPAair_PM25_NC2019_raw.csv"),
  stringsAsFactors = T)

pm25_nc2018<-read.csv(
  file=here("Data/Raw/EPAair_PM25_NC2018_raw.csv"),
  stringsAsFactors=T)

o3_nc2019<-read.csv(
  file=here("Data/Raw/EPAair_O3_NC2019_raw.csv"),
  stringsAsFactors = T)

o3_nc2018<-read.csv(
  file=here("Data/Raw/EPAair_O3_NC2018_raw.csv"),
  stringsAsFactors = T)

#2 checking the structure of the dataset, number of observations and variables
dim(pm25_nc2018)
```

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

```
dim(pm25_nc2019)
```

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

```
dim(o3_nc2018)
```

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

```
dim(o3_nc2019)
```

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

All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern? >Answer: Yes, each of the datasets has a unique number of observations of the same number of variables (20 variables). PM25_NC2018 has 8,983 observations; PM25_NC2019 has 8,581 observations; O3_NC2018 has 9,737 observations; and O3_NC2019 has 10,592 observations.

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 setting the date columns to be in date data format
pm25_nc2018$Date<-as.Date(pm25_nc2018$Date, format="%m/%d/%Y")
pm25_nc2019$Date<-as.Date(pm25_nc2019$Date, format="%m/%d/%Y")
o3_nc2018$Date<-as.Date(o3_nc2018$Date, format="%m/%d/%Y")
o3_nc2019$Date<-as.Date(o3_nc2019$Date, format="%m/%d/%Y")

#4 selecting required columns
pm25_nc2018_select<-select(pm25_nc2018,
              Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
              SITE_LATITUDE, SITE_LONGITUDE)
pm25_nc2019_select<-select(pm25_nc2019,
              Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
              SITE_LATITUDE, SITE_LONGITUDE)
o3_nc2018_select<-select(o3_nc2018,
              Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
              SITE_LATITUDE, SITE_LONGITUDE)
o3_nc2019_select<-select(o3_nc2019,
              Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
              SITE_LATITUDE, SITE_LONGITUDE)

#5 assigning value of "PM2.5" to the AQS column
pm25_nc2018_select <- pm25_nc2018_select %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

pm25_nc2019_select <- pm25_nc2019_select %>%
  mutate(AQS_PARAMETER_DESC = "PM2.5")

#6 creating csv files from processed data
write.csv(pm25_nc2018_select, row.names = FALSE,
          file="./Data/Processed/EPAair_PM25_NC2018_processed.csv")
write.csv(pm25_nc2019_select, row.names = FALSE,
          file="./Data/Processed/EPAair_PM25_NC2019_processed.csv")
write.csv(o3_nc2018_select, row.names = FALSE,
          file="./Data/Processed/EPAair_O3_NC2018_processed.csv")
write.csv(o3_nc2019_select, row.names = FALSE,
          file="./Data/Processed/EPAair_O3_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 processed data
combined_all<-rbind(pm25_nc2018_select, pm25_nc2019_select,
                    o3_nc2018_select, o3_nc2019_select)

#8 finding the intersecting site names
common_names <- Reduce(
  intersect,
  list(
    pm25_nc2018_select$Site.Name,
    pm25_nc2019_select$Site.Name,
    o3_nc2018_select$Site.Name,
    o3_nc2019_select$Site.Name))

common_names <- common_names[common_names != ""]
print(common_names)
```

```
##  [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"
```

```
#selecting desired information from processed combined data, removing NA,
#and calculating mean values
combined_all_select <- combined_all %>%
  filter(Site.Name %in% common_names)%>%
  filter(!is.na(Site.Name))%>%
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY)%>%
  summarise(mean_aqi = mean(DAILY_AQI_VALUE),
            mean_lat = mean(SITE_LATITUDE),
            mean_long = 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
dim(combined_all_select)
```

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

```r
#9 making ozone and pm2.5 into their own columns
combined_all_select_wider<-combined_all_select%>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC, #categorical column of ozone and PM2.5
    values_from = mean_aqi) #values column

#10 structure of dataset
dim(combined_all_select_wider)
```

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

```r
#11 saving new combined processed data as a csv fil
write.csv(combined_all_select_wider, 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 making table for average ozone and PM2.5 grouped by site, month, and year
#while removing non available data.
summary_df<-combined_all_select_wider%>%
  group_by(Site.Name, month, year)%>%
  summarise(
    mean_aqi_ozone=mean(Ozone),
    mean_aqi_pm25=mean(PM2.5))%>%
  drop_na(mean_aqi_ozone)
```

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

```r
#13 summary data frame structure
dim(summary_df)
```

```
## [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.

    Answer: We used `drop_na` because it is specifically designed to drop rows with missing values in specific columns. Meanwhile, `na.omit` removes any row in a data frame that contains NA values across all columns and does not give the flexibility to specify which columns to check for missing data. Therefore, in this case it is more appropriate to use drop.na since we can specify the column we are concerned with and maintain a substantial dataset.