# Assignment 4: Data Wrangling

Nagarajan Vaidya Subramanian

Spring 2023

## 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.

The completed exercise is due on Friday, Feb 20th @ 5:00pm.

## 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. Apply the `glimpse()` function to reveal the dimensions, column names, and structure of each dataset.

```
# 1a
library(tidyverse)
library(lubridate)
# install.packages('here')
library(here)

# 1b
getwd()

## [1] "X:/ENV 872 Environmental Data Analytics/Git_codes/EDA-Spring2023"

# 1c
o3_2018 <- read.csv("./Data/Raw/EPAair_O3_NC2018_raw.csv", stringsAsFactors =
TRUE)
```

```
o3_2019 <- read.csv("./Data/Raw/EPAair_O3_NC2019_raw.csv", stringsAsFactors =
TRUE)
pm25_2018 <- read.csv("./Data/Raw/EPAair_PM25_NC2018_raw.csv",
stringsAsFactors = TRUE)
pm25_2019 <- read.csv("./Data/Raw/EPAair_PM25_NC2019_raw.csv",
stringsAsFactors = TRUE)

# 2
print("Ozone raw data from 2018")

## [1] "Ozone raw data from 2018"

glimpse(o3_2018)

## Rows: 9,737
## Columns: 20
## $ Date                                    <fct> 03/01/2018, 03/02/2018,
03/03/201…
## $ Source                                  <fct> AQS, AQS, AQS, AQS, AQS, AQS,
AQS…
## $ Site.ID                                 <int> 370030005, 370030005,
370030005, …
## $ POC                                     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ Daily.Max.8.hour.Ozone.Concentration <dbl> 0.043, 0.046, 0.047, 0.049,
0.047…
## $ UNITS                                   <fct> ppm, ppm, ppm, ppm, ppm, ppm,
ppm…
## $ DAILY_AQI_VALUE                         <int> 40, 43, 44, 45, 44, 28, 33,
41, 4…
## $ Site.Name                               <fct> Taylorsville Liledoun,
Taylorsvil…
## $ DAILY_OBS_COUNT                         <int> 17, 17, 17, 17, 17, 17, 17,
17, 1…
## $ PERCENT_COMPLETE                        <dbl> 100, 100, 100, 100, 100, 100,
100…
## $ AQS_PARAMETER_CODE                      <int> 44201, 44201, 44201, 44201,
44201…
## $ AQS_PARAMETER_DESC                      <fct> Ozone, Ozone, Ozone, Ozone,
Ozone…
## $ CBSA_CODE                               <int> 25860, 25860, 25860, 25860,
25860…
## $ CBSA_NAME                               <fct> "Hickory-Lenoir-Morganton,
NC", "…
## $ STATE_CODE                              <int> 37, 37, 37, 37, 37, 37, 37,
37, 3…
## $ STATE                                   <fct> North Carolina, North
Carolina, N…
## $ COUNTY_CODE                             <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, …
```

```
## $ COUNTY                                    <fct> Alexander, Alexander,
Alexander, …
## $ SITE_LATITUDE                             <dbl> 35.9138, 35.9138, 35.9138,
35.913…
## $ SITE_LONGITUDE                            <dbl> -81.191, -81.191, -81.191, -
81.19…

print("\n Ozone raw data from 2019")

## [1] "\n Ozone raw data from 2019"

glimpse(o3_2019)

## Rows: 10,592
## Columns: 20
## $ Date                                      <fct> 01/01/2019, 01/02/2019,
01/03/201…
## $ Source                                    <fct> AirNow, AirNow, AirNow,
AirNow, A…
## $ Site.ID                                   <int> 370030005, 370030005,
370030005, …
## $ POC                                       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ Daily.Max.8.hour.Ozone.Concentration <dbl> 0.029, 0.018, 0.016, 0.022,
0.037…
## $ UNITS                                     <fct> ppm, ppm, ppm, ppm, ppm, ppm,
ppm…
## $ DAILY_AQI_VALUE                           <int> 27, 17, 15, 20, 34, 34, 27,
35, 3…
## $ Site.Name                                 <fct> Taylorsville Liledoun,
Taylorsvil…
## $ DAILY_OBS_COUNT                           <int> 24, 24, 24, 24, 24, 24, 24,
24, 2…
## $ PERCENT_COMPLETE                          <dbl> 100, 100, 100, 100, 100, 100,
100…
## $ AQS_PARAMETER_CODE                        <int> 44201, 44201, 44201, 44201,
44201…
## $ AQS_PARAMETER_DESC                        <fct> Ozone, Ozone, Ozone, Ozone,
Ozone…
## $ CBSA_CODE                                 <int> 25860, 25860, 25860, 25860,
25860…
## $ CBSA_NAME                                 <fct> "Hickory-Lenoir-Morganton,
NC", "…
## $ STATE_CODE                                <int> 37, 37, 37, 37, 37, 37, 37,
37, 3…
## $ STATE                                     <fct> North Carolina, North
Carolina, N…
## $ COUNTY_CODE                               <int> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, …
## $ COUNTY                                    <fct> Alexander, Alexander,
Alexander, …
```

```
## $ SITE_LATITUDE                          <dbl> 35.9138, 35.9138, 35.9138,
35.913…
## $ SITE_LONGITUDE                         <dbl> -81.191, -81.191, -81.191, -
81.19…

print("\n PM2.5 data from 2018")

## [1] "\n PM2.5 data from 2018"

glimpse(pm25_2018)

## Rows: 8,983
## Columns: 20
## $ Date                             <fct> 01/02/2018, 01/05/2018, 01/08/2018,
01/…
## $ Source                           <fct> AQS, AQS, AQS, AQS, AQS, AQS, AQS,
AQS,…
## $ Site.ID                          <int> 370110002, 370110002, 370110002,
370110…
## $ POC                              <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ Daily.Mean.PM2.5.Concentration <dbl> 2.9, 3.7, 5.3, 0.8, 2.5, 4.5, 1.8,
2.5,…
## $ UNITS                            <fct> ug/m3 LC, ug/m3 LC, ug/m3 LC, ug/m3
LC,…
## $ DAILY_AQI_VALUE                  <int> 12, 15, 22, 3, 10, 19, 8, 10, 18,
7, 24…
## $ Site.Name                        <fct> Linville Falls, Linville Falls,
Linvill…
## $ DAILY_OBS_COUNT                  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ PERCENT_COMPLETE                 <dbl> 100, 100, 100, 100, 100, 100, 100,
100,…
## $ AQS_PARAMETER_CODE               <int> 88502, 88502, 88502, 88502, 88502,
8850…
## $ AQS_PARAMETER_DESC               <fct> Acceptable PM2.5 AQI & Speciation
Mass,…
## $ CBSA_CODE                        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA,…
## $ CBSA_NAME                        <fct> "", "", "", "", "", "", "", "", "",
"",…
## $ STATE_CODE                       <int> 37, 37, 37, 37, 37, 37, 37, 37, 37,
37,…
## $ STATE                            <fct> North Carolina, North Carolina,
North C…
## $ COUNTY_CODE                      <int> 11, 11, 11, 11, 11, 11, 11, 11, 11,
11,…
## $ COUNTY                           <fct> Avery, Avery, Avery, Avery, Avery,
Aver…
## $ SITE_LATITUDE                    <dbl> 35.97235, 35.97235, 35.97235,
35.97235,…
```

```
## $ SITE_LONGITUDE                  <dbl> -81.93307, -81.93307, -81.93307, -
81.93…

print("\n PM2.5 data from 2019")

## [1] "\n PM2.5 data from 2019"

glimpse(pm25_2019)

## Rows: 8,581
## Columns: 20
## $ Date                            <fct> 01/03/2019, 01/06/2019, 01/09/2019,
01/…
## $ Source                          <fct> AQS, AQS, AQS, AQS, AQS, AQS, AQS,
AQS,…
## $ Site.ID                         <int> 370110002, 370110002, 370110002,
370110…
## $ POC                             <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ Daily.Mean.PM2.5.Concentration <dbl> 1.6, 1.0, 1.3, 6.3, 2.6, 1.2, 1.5,
1.5,…
## $ UNITS                           <fct> ug/m3 LC, ug/m3 LC, ug/m3 LC, ug/m3
LC,…
## $ DAILY_AQI_VALUE                 <int> 7, 4, 5, 26, 11, 5, 6, 6, 15, 7,
14, 20…
## $ Site.Name                       <fct> Linville Falls, Linville Falls,
Linvill…
## $ DAILY_OBS_COUNT                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, …
## $ PERCENT_COMPLETE                <dbl> 100, 100, 100, 100, 100, 100, 100,
100,…
## $ AQS_PARAMETER_CODE              <int> 88502, 88502, 88502, 88502, 88502,
8850…
## $ AQS_PARAMETER_DESC              <fct> Acceptable PM2.5 AQI & Speciation
Mass,…
## $ CBSA_CODE                       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA,
NA,…
## $ CBSA_NAME                       <fct> "", "", "", "", "", "", "", "", "",
"",…
## $ STATE_CODE                      <int> 37, 37, 37, 37, 37, 37, 37, 37, 37,
37,…
## $ STATE                           <fct> North Carolina, North Carolina,
North C…
## $ COUNTY_CODE                     <int> 11, 11, 11, 11, 11, 11, 11, 11, 11,
11,…
## $ COUNTY                          <fct> Avery, Avery, Avery, Avery, Avery,
Aver…
## $ SITE_LATITUDE                   <dbl> 35.97235, 35.97235, 35.97235,
35.97235,…
## $ SITE_LONGITUDE                  <dbl> -81.93307, -81.93307, -81.93307, -
81.93…
```

## Wrangle individual datasets to create processed files.

3. Change 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
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
o3_2018_processed <- o3_2018 %>%
    select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
SITE_LATITUDE,
        SITE_LONGITUDE)

o3_2019_processed <- o3_2019 %>%
    select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
SITE_LATITUDE,
        SITE_LONGITUDE)

pm25_2018_processed <- pm25_2018 %>%
    select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
SITE_LATITUDE,
        SITE_LONGITUDE)

pm25_2019_processed <- pm25_2019 %>%
    select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
SITE_LATITUDE,
        SITE_LONGITUDE)

# 5
pm25_2018_processed$AQS_PARAMETER_DESC = "PM2.5"

pm25_2019_processed$AQS_PARAMETER_DESC = "PM2.5"

# 6
write.csv(o3_2018_processed, file =
"./Data/Processed/EPAair_O3_NC2018_processed.csv")
write.csv(o3_2019_processed, file =
"./Data/Processed/EPAair_O3_NC2019_processed.csv")
```

```
write.csv(pm25_2018_processed, file =
"./Data/Processed/EPAair_PM25_NC2018_processed.csv")
write.csv(pm25_2019_processed, 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 all 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...)

- 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
colnames(o3_2018_processed) == colnames(o3_2019_processed)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

colnames(o3_2018_processed) == colnames(pm25_2018_processed)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

colnames(pm25_2018_processed) == colnames(pm25_2019_processed)

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

aqi <- rbind(o3_2018_processed, o3_2019_processed, pm25_2018_processed,
pm25_2019_processed)
```

```r
# 8 creating a vector of common site names sitenames <-
# intersect(pm25_2018_processed$Site.Name, o3_2018_processed$Site.Name)
# sitenames <- intersect(pm25_2019_processed$Site.Name, sitenames) sitenames
<-
# intersect(o3_2019_processed$Site.Name, sitenames)

# creating a vector of site names to use while filtering our dataframe by
site
sitenames <- 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")

# grouping by date, site name, AQS parameter, and county; then taking the
mean
# of the AQI value, latitude, and longitude; then removing the original
# latitude and longitude columns.
aqi_new <- aqi %>%
    filter(aqi$Site.Name %in% sitenames) %>%
    group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
    mutate(mean_aqi = mean(DAILY_AQI_VALUE), mean_lat = mean(SITE_LATITUDE),
mean_long = mean(SITE_LONGITUDE),
        Month = month(Date), Year = year(Date)) %>%
    select(Date, Month, Year, Site.Name, AQS_PARAMETER_DESC, COUNTY,
mean_aqi, mean_lat,
        mean_long)



# 9
aqi_spread <- aqi_new %>%
    group_by(Date, Site.Name, COUNTY, Month, Year) %>%
    pivot_wider(names_from = AQS_PARAMETER_DESC, values_from = mean_aqi)

## Warning: Values from `mean_aqi` are not uniquely identified; output will
contain
## list-cols.
## • Use `values_fn = list` to suppress this warning.
## • Use `values_fn = {summary_fun}` to summarise duplicates.
## • Use the following dplyr code to identify duplicates.
##   {data} %>%
##   dplyr::group_by(Date, Month, Year, Site.Name, COUNTY, mean_lat,
mean_long,
##   AQS_PARAMETER_DESC) %>%
##   dplyr::summarise(n = dplyr::n(), .groups = "drop") %>%
##   dplyr::filter(n > 1L)

# 10
dim(aqi_spread)
```

```
## [1] 8976    9

# 11 write.csv(aqi_spread,
# './Data/Processed/EPAair_O3_PM25_NC1819_Processed.csv') commented out
because
# it throws an error that i cant seem to fix
```

## 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
aqi_summary <- aqi_spread %>%
    group_by(Site.Name, Month, Year) %>%
    mutate(mean_Ozone = mean(Ozone), mean_pm25 = mean(PM2.5)) %>%
    drop_na(mean_Ozone)

## Warning: There were 616 warnings in `mutate()`.
## The first warning was:
## ℹ In argument: `mean_Ozone = mean(Ozone)`.
## ℹ In group 1: `Site.Name = Bryson City`, `Month = 1`, `Year = 2018`.
## Caused by warning in `mean.default()`:
## ! argument is not numeric or logical: returning NA
## ℹ Run
]8;;ide:run:dplyr::last_dplyr_warnings()dplyr::last_dplyr_warnings()]8;; to
see the 615 remaining warnings.

# 13
dim(aqi_summary)

## [1]  0 11
```

14. Why did we use the function `drop_na` rather than `na.omit`?

    Answer: The drop_na function is a port of the dropna function from Python. It removes rows containing missing values in the column we specify. Whereas na.omit does not have that level of granuality; instead it will remove a row if there is an NA in *any* column. Because we want to retain data points with missing values of PM2.5 concentration, we are using the drop_na function.