Assignment 4: Data Wrangling

Annabelle White

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
pkgs <- c("tidyverse", "lubridate", "here") # I hate repeating the same command
lapply(pkgs, library, character.only=TRUE) # so I looked up how to load them all at once
```

```
## [[1]]
##  [1] "forcats"   "stringr"   "dplyr"     "purrr"     "readr"     "tidyr"
##  [7] "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"  "grDevices"
## [13] "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"
## [13] "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
##
## [[3]]
##  [1] "here"       "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"      "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [13] "graphics"   "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
#1b
here()
```

```
## [1] "C:/Users/ardwh/OneDrive/Documents/NSOE/env872/EDA-Spring2023"
```

```
#1c
# This could be done much more efficiently
# But I'm not yet sure how to ensure stringsAsFactors that way
O3_2018.raw <- read.csv(
  file = here('Data','Raw','EPAair_O3_NC2018_raw.csv'),
  stringsAsFactors = T
)
O3_2019.raw <- read.csv(
  file = here('Data','Raw','EPAair_O3_NC2019_raw.csv'),
  stringsAsFactors = T
)
PM25_2018.raw <- read.csv(
  file = here('Data','Raw','EPAair_PM25_NC2018_raw.csv'),
  stringsAsFactors = T
)
PM25_2019.raw <- read.csv(
  file = here('Data','Raw','EPAair_PM25_NC2019_raw.csv'),
  stringsAsFactors = T
)

#2
# I tried using lapply() to glimpse each one but it went ballistic on me
# so I guess I'll just do it one by one
glimpse(O3_2018.raw)
```

```
## 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~
```

```
glimpse(O3_2019.raw)
```

```
## 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~
```

```
glimpse(PM25_2018.raw)
```

```
## 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~
```

```
glimpse(PM25_2019.raw)
```

```
## 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".

#"'{r Wrangle individual datasets}

```
# I detested doing this step-by-step so I made pipes for each dataset
# This could be further refined by making an iterative process

O3_2018.processed <-
  O3_2018.raw %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE) %>% #4
  mutate(Date = mdy(Date)) #3
write.csv(O3_2018.processed, row.names = FALSE,
          file = here('Data','Processed',
                      'EPAair_O3_NC2018_processed.csv')) #6
```

```
O3_2019.processed <-
  O3_2019.raw %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE) %>% #4
  mutate(Date = mdy(Date)) #3
write.csv(O3_2019.processed, row.names = FALSE,
          file = here('Data','Processed',
                      'EPAair_O3_NC2019_processed.csv')) #6

PM25_2018.processed <-
  PM25_2018.raw %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE) %>% #4
  mutate(Date = mdy(Date)) %>% #3
  mutate(AQS_PARAMETER_DESC = "PM2.5") #5
write.csv(PM25_2018.processed, row.names = FALSE,
          file = here('Data','Processed',
                      'EPAair_PM25_NC2018_processed.csv')) #6

PM25_2019.processed <-
  PM25_2019.raw %>%
  select(Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY,
         SITE_LATITUDE, SITE_LONGITUDE) %>% #4
  mutate(Date = mdy(Date)) %>% #3
  mutate(AQS_PARAMETER_DESC = "PM2.5") #5
write.csv(PM25_2019.processed, row.names = FALSE,
          file = here('Data','Processed',
                      'EPAair_PM25_NC2019_processed.csv')) #6
```

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
EPA.air <- rbind(O3_2018.processed,
                 O3_2019.processed,
                 PM25_2018.processed,
                 PM25_2019.processed) # Join the four dataframes

#8
EPA.air.processed <-
  EPA.air %>%
  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")) %>%
  # You gave me the list of site names so I'm just going to use it
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>% # Group by fields
  summarise(AQI.Mean = mean(DAILY_AQI_VALUE), # Find means of these fields
            Latitude.Mean = mean(SITE_LATITUDE), # Though it seems foolish
            Longitude.Mean = mean(SITE_LONGITUDE)) %>%
  mutate(Month = lubridate::month(Date),
         Year = lubridate::year(Date)) # Add new month & year columns
```

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

```
# I tried for HOURS to use intersect to cleverly filter the site names
# It does not work that way with joined dataframes
# And joining, then splitting, then joining again feels incredibly redundant
# I hate being redundant with my code
# So I'm just going to do it my way and you can take off points if you like

#9
EPA.air.processed <- pivot_wider(EPA.air.processed,
                                 names_from = AQS_PARAMETER_DESC,
                                 values_from = AQI.Mean)
# This splits AQI.Mean into values for ozone and particulate matter,
# based on the corresponding values in AQS_PARAMETER_DESC.
# This flattens each distinct measurement into just one row! Handy!

#10
dim(EPA.air.processed) # Shows dimensions of dataset: 8976 x 9
```

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

```
#11
write.csv(EPA.air.processed, row.names = FALSE,
          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

EPA.air.summary <-
  EPA.air.processed %>%
  group_by(Site.Name, Month, Year) %>% # Group by fields
  summarise(Ozone.Mean = mean(Ozone), # Find means of these fields
            PM2.5.Mean = mean(PM2.5))  %>%
  drop_na(Ozone.Mean)
```

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

```
#13
dim(EPA.air.summary) # Shows dimensions: 182 x 5
```

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

```
# This is a drastic reduction in dimensions
# But looking at EPA.air.processed shows that the majority of ozone measurements were NA
# So this is to be expected
```

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

    Answer: The drop_na function is set up to work with tidyverse and allows us to select which variables to drop NA values from. Since we only care about removing NAs from ozone but not PM in this case, it's useful to be able to specify them. The na.omit function would have dropped all rows with NA in any column.