Assignment 4: Data Wrangling

Elsie Liu #3

OVERVIEW

This exercise accompanies the lessons in Environmental Data Analytics on Data Wrangling

Set up your session

1. Check your working directory, load the `tidyverse` and `lubridate` packages, and upload all four raw data files associated with the EPA Air dataset. See the README file for the EPA air datasets for more information (especially if you have not worked with air quality data previously).
2. Explore the dimensions, column names, and structure of the datasets.

```
#1 pre
library(plyr)
library(tidyverse)
library(lubridate)
getwd()
```

```
## [1] "/Users/Lsy/Box Sync/Duke/spring2022/872-EDA/GitRepository/Environmental_Data_Analytics_2022/Ass
```

```
airo318<- read.csv("../Data/Raw/EPAair_O3_NC2018_raw.csv")
airo319<- read.csv("../Data/Raw/EPAair_O3_NC2019_raw.csv")
airpm18<- read.csv("../Data/Raw/EPAair_PM25_NC2018_raw.csv")
airpm19<- read.csv("../Data/Raw/EPAair_PM25_NC2019_raw.csv")

#2 explore
dim(airo318)
```

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

```
colnames(airo318)
```

```
##  [1] "Date"
##  [2] "Source"
##  [3] "Site.ID"
##  [4] "POC"
##  [5] "Daily.Max.8.hour.Ozone.Concentration"
##  [6] "UNITS"
##  [7] "DAILY_AQI_VALUE"
##  [8] "Site.Name"
##  [9] "DAILY_OBS_COUNT"
## [10] "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"
## [12] "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"
## [14] "CBSA_NAME"
## [15] "STATE_CODE"
```

```
## [16] "STATE"
## [17] "COUNTY_CODE"
## [18] "COUNTY"
## [19] "SITE_LATITUDE"
## [20] "SITE_LONGITUDE"
```

```
dim(airpm18)
```

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

```
colnames(airo319)
```

```
##  [1] "Date"
##  [2] "Source"
##  [3] "Site.ID"
##  [4] "POC"
##  [5] "Daily.Max.8.hour.Ozone.Concentration"
##  [6] "UNITS"
##  [7] "DAILY_AQI_VALUE"
##  [8] "Site.Name"
##  [9] "DAILY_OBS_COUNT"
## [10] "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"
## [12] "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"
## [14] "CBSA_NAME"
## [15] "STATE_CODE"
## [16] "STATE"
## [17] "COUNTY_CODE"
## [18] "COUNTY"
## [19] "SITE_LATITUDE"
## [20] "SITE_LONGITUDE"
```

```
dim(airpm19)
```

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

```
colnames(airpm18)
```

```
##  [1] "Date"                           "Source"
##  [3] "Site.ID"                        "POC"
##  [5] "Daily.Mean.PM2.5.Concentration" "UNITS"
##  [7] "DAILY_AQI_VALUE"                "Site.Name"
##  [9] "DAILY_OBS_COUNT"                "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"             "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"                      "CBSA_NAME"
## [15] "STATE_CODE"                     "STATE"
## [17] "COUNTY_CODE"                    "COUNTY"
## [19] "SITE_LATITUDE"                  "SITE_LONGITUDE"
```

```
dim(airo319)
```

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

```
colnames(airpm19)
```

```
##  [1] "Date"                           "Source"
##  [3] "Site.ID"                        "POC"
##  [5] "Daily.Mean.PM2.5.Concentration" "UNITS"
```

```
##  [7] "DAILY_AQI_VALUE"               "Site.Name"
##  [9] "DAILY_OBS_COUNT"               "PERCENT_COMPLETE"
## [11] "AQS_PARAMETER_CODE"            "AQS_PARAMETER_DESC"
## [13] "CBSA_CODE"                     "CBSA_NAME"
## [15] "STATE_CODE"                    "STATE"
## [17] "COUNTY_CODE"                   "COUNTY"
## [19] "SITE_LATITUDE"                 "SITE_LONGITUDE"
```

Wrangle individual datasets to create processed files.

3. Change date to a date object
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 coerce date format
airo318$Date<- as.Date(airo318$Date,format = "%m/%d/%Y")
airo319$Date<- as.Date(airo319$Date,format = "%m/%d/%Y")
airpm18$Date<- as.Date(airpm18$Date,format = "%m/%d/%Y")
airpm19$Date<- as.Date(airpm19$Date,format = "%m/%d/%Y")

#4 select columns
sub_airo318 <- select(airo318, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITU
sub_airo319 <- select(airo319, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITU
sub_airpm18 <- select(airpm18, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITU
sub_airpm19 <- select(airpm19, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITU

#5 fill AQS_PARAMETER_DESC with PM2.5
sub_airpm18$AQS_PARAMETER_DESC <- "PM2.5"
sub_airpm19$AQS_PARAMETER_DESC <- "PM2.5"

#6 save to preocessed folder
write.csv(x = sub_airo318, file = "../Data/Processed/EPAair_O3_NC2018_processed.csv", row.names = F)
write.csv(x = sub_airo319, file = "../Data/Processed/EPAair_O3_NC2019_processed.csv", row.names = F)
write.csv(x = sub_airpm18, file = "../Data/Processed/EPAair_PM25_NC2018_processed.csv", row.names = F)
write.csv(x = sub_airpm19, file = "../Data/Processed/EPAair_PM25_NC2019_processed.csv", row.names = F)
```

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:

- Filter records to include just the 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 `intersect` function can figure out common factor levels if we didn't give you this list...)
- Some sites have multiple measurements per day. Use the split-apply-combine strategy to generate daily means: group by date, site, 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_NC2122_Processed.csv"

```
#7 Combine the four datasets
air.all <- rbind(sub_airo318,sub_airo319,sub_airpm18,sub_airpm19)

#8 Wrangle new dataset with a pipe function
colnames(air.all) <- c("Date", "AQI", "Site", "AQSParameter", "County","latitude","longitude")

air.sum <-
  air.all %>%
  filter(Site %in% c("Linville Falls", "Durham Armory", "Leggett", "Hattie Avenue", "Clemmons Middle",
  group_by(Date, Site, AQSParameter, County) %>%
  summarise(meanAQI = mean(AQI),
            meanLat = mean(latitude),
            meanLong = mean(longitude)) %>%
  mutate(Month = month(Date)) %>%
  mutate(Year = year(Date))
```

```
## `summarise()` has grouped output by 'Date', 'Site', 'AQSParameter'. You can override using the `.grou
```

```
dim(air.sum)
```

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

```
#9 spread ozone and pm2.5
air.tidy <- pivot_wider(data = air.sum, names_from = AQSParameter, values_from = meanAQI)

#10 dimension
dim(air.tidy)
```

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

```
#11 save csv file
write.csv(air.tidy, "../Data/Processed/EPAair_O3_PM25_NC2122_Processed.csv",row.names = F)
```

Generate summary tables

12a. Use the split-apply-combine strategy to generate a summary data frame from your results from Step 9 above. Data should be grouped by site, month, and year. Generate the mean AQI values for ozone and PM2.5 for each group. 12b. BONUS: Add a piped statement to 12a that removes rows where both mean ozone and mean PM2.5 have missing values. 13. Call up the dimensions of the summary dataset.

```
#12(a,b)
#a. generate summary data frame
air.wrangle.fin <-
  air.tidy %>%
  group_by(Site, Month, Year) %>%
  summarise(meanPM25 = mean(PM2.5),
            meanO3 = mean(Ozone))
```

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

```
#b. remove rows with NAs in both ozone and pm2.5
air.wrangle.fin <- filter(air.wrangle.fin, !is.na(meanPM25) | !is.na(meanO3))
```

```r
#13 dimension
dim(air.wrangle.fin)
```

```
## [1] 292   5
```

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

    Answer: "na.omit" would removes NA from the whole dataframe, drop_na removes NA based on each specific column.