Assignment 4: Data Wrangling (Fall 2024)

Jonathan Gilman

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.

```
#1a
#load packages
library(tidyverse)
library(lubridate)
library(here)

#1b
#check current working directory
here()
```

```
## [1] "/home/guest/EDE_Fall2024"
```

```
#1c
#upload datasets
# `EPAair_O3_NC2018_raw.csv`, `EPAair_O3_NC2019_raw.csv`, `EPAair_PM25_NC2018_raw.csv`, and `EPAair_PM2
```

```r
ozone_2018 <- read.csv(
  file = here('Data/Raw/EPAair_O3_NC2018_raw.csv'),
  stringsAsFactors = T
)
ozone_2019 <- read.csv(
  file = here('Data/Raw/EPAair_O3_NC2019_raw.csv'),
  stringsAsFactors = T
)
pm25_2018 <- read.csv(
  file = here('Data/Raw/EPAair_PM25_NC2018_raw.csv'),
  stringsAsFactors = T
)
pm25_2019 <- read.csv(
  file = here('Data/Raw/EPAair_PM25_NC2019_raw.csv'),
  stringsAsFactors = T
)

#2
# look at dimensions of the 4 df: # of rows, # of columns
dim(ozone_2018)
```

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

```r
dim(ozone_2019)
```

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

```r
dim(pm25_2018)
```

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

```r
dim(pm25_2019)
```

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

> All four datasets should have the same number of columns but unique record counts (rows). Do your datasets follow this pattern?

Yes, all four datasets here have 20 columns.

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: Change the Date columns to be date objects.
ozone_2018$Date <- as.Date(ozone_2018$Date, format = "%m/%d/%Y")
ozone_2019$Date <- as.Date(ozone_2019$Date, format = "%m/%d/%Y")
pm25_2018$Date <- as.Date(pm25_2018$Date, format = "%m/%d/%Y")
pm25_2019$Date <- as.Date(pm25_2019$Date, format = "%m/%d/%Y")


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

o18 <- select(ozone_2018, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, S
o19 <- select(ozone_2019, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, S
p18 <- select(pm25_2018, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SI
p19 <- select(pm25_2019, Date, DAILY_AQI_VALUE, Site.Name, AQS_PARAMETER_DESC, COUNTY, SITE_LATITUDE, SI

#5: For the PM2.5 datasets, fill all cells in AQS_PARAMETER_DESC with "PM2.5" (all cells in this column
library(dplyr)
p18 <- p18 %>% mutate(AQS_PARAMETER_DESC = "PM2.5")

p19 <- p19 %>% mutate(AQS_PARAMETER_DESC = "PM2.5")


#6: store processed files
#`EPAair_O3_NC2018_raw.csv`, `EPAair_O3_NC2019_raw.csv`, `EPAair_PM25_NC2018_raw.csv`, and `EPAair_PM25

write_csv(o18, here('Data/Processed/EPAair_O3_NC2018_processed.csv'))
write_csv(o19, here('Data/Processed/EPAair_O3_NC2019_processed.csv'))
write_csv(p18, here('Data/Processed/EPAair_PM25_NC2018_processed.csv'))
write_csv(p19, here('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 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: combine datasets
# check column names
colnames(o18)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(o19)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(p18)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
colnames(p19)
```

```
## [1] "Date"                "DAILY_AQI_VALUE"    "Site.Name"
## [4] "AQS_PARAMETER_DESC" "COUNTY"             "SITE_LATITUDE"
## [7] "SITE_LONGITUDE"
```

```
#combine datasets
epa_air <- rbind(o18,o19,p18,p19)
#epa_air

#8
#common sites
common <-  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" )

#wrangle with pipe
wrangled <- epa_air %>%
  # filter for common sites
  filter(Site.Name %in% common) %>%
  # group
```

```
  group_by(Date, Site.Name, AQS_PARAMETER_DESC, COUNTY) %>%
  # find mean of the AQI value, latitude, and longitude
  summarize(
    mean_AQI = mean(DAILY_AQI_VALUE, na.rm = TRUE),
    mean_lat = mean(SITE_LATITUDE, na.rm = TRUE),
    mean_long = mean(SITE_LONGITUDE, na.rm = TRUE),
    .groups = 'drop'
  ) %>%
  # add columns for "Month" and "Year" by parsing "Date" column
  mutate(Month = month(Date),
         Year = year(Date))

# check dimensions -- should be 14,752 x 9.
dim(wrangled)
```

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

```
#9
library(tidyr)

# pivot wider to separate AQI values for ozone and PM2.5
epa_air_wide <- wrangled %>%
  pivot_wider(
    names_from = AQS_PARAMETER_DESC,
    values_from = AQS_PARAMETER_DESC,
    values_fn = list(AQS_PARAMETER_DESC = ~ .[1]),
    values_fill = list(Ozone = NA, `PM2.5` = NA)
  )


#10: Call up the dimensions of your new tidy dataset
dim(epa_air_wide)
```

```
## [1] 14650    10
```

```
#11: Save your processed dataset with the following file name: "EPAair_O3_PM25_NC1819_Processed.csv
write_csv(epa_air_wide, 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
summary <- epa_air_wide %>%
  # ID month and year
  mutate(Month = month(Date),
```

```r
         Year = year(Date)) %>%
  # group by site, month, and year
  group_by(Site.Name, Month, Year) %>%
  # generate mean AQI values for each group
  summarize(
    mean_ozone = mean(mean_AQI[!is.na(Ozone)], na.rm = TRUE),
    mean_pm25 = mean(mean_AQI[!is.na(PM2.5)], na.rm = TRUE),
    .groups = 'drop') %>%
  # drop values where mean ozone values are not available
  drop_na(mean_ozone)

#13
# call up dimensions
dim(summary)
```

```
## [1] 239   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 the function `drop_na` instead of `na.omit` because `drop_na` allows us to target the mean_ozone column and remove rows where there is not a value. The function `na.omit` doesn't allow for that targeting, and just removes any row from any column where there is a missing value.