HW1-Trinath Sai Subhash Reddy-Pittala

Subhash

2023-03-05

```
library(formatR)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggfortify)
library(nycflights13)
library(modelr)
library(magick)
```

```
## Linking to ImageMagick 6.9.12.3
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```
library(haven)
library(dplyr)
library(ISLR)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
knitr::opts_chunk$set(width = 60, warning = FALSE, message = FALSE, tidy.opts=list(width.cutoff=60),tidy
#knitr::opts_chunk$set(echo = TRUE)
```

R Working Environment

Please load all the packages used in the following R chunk before the function sessionInfo(). This is done using the library() function. e.g., library(tidyverse)

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.9.1    ISLR_1.4           haven_2.5.1        magick_2.7.3
##  [5] modelr_0.1.10      nycflights13_1.0.2 ggfortify_0.4.15   GGally_2.1.2
##  [9] forcats_0.5.2      stringr_1.5.0      dplyr_1.0.10       purrr_1.0.1
## [13] readr_2.1.3        tidyr_1.3.0        tibble_3.1.8       ggplot2_3.4.0
## [17] tidyverse_1.3.2    formatR_1.14
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.10          assertthat_0.2.1    digest_0.6.31
##  [4] utf8_1.2.2           R6_2.5.1            cellranger_1.1.0
##  [7] plyr_1.8.8           backports_1.4.1     reprex_2.0.2
## [10] evaluate_0.20        httr_1.4.4          pillar_1.8.1
## [13] rlang_1.0.6          googlesheets4_1.0.1 readxl_1.4.1
## [16] rstudioapi_0.14      rmarkdown_2.20      googledrive_2.0.0
## [19] munsell_0.5.0        broom_1.0.3         compiler_4.2.2
## [22] xfun_0.36            pkgconfig_2.0.3     htmltools_0.5.4
## [25] tidyselect_1.2.0     gridExtra_2.3       reshape_0.8.9
## [28] fansi_1.0.4          crayon_1.5.2        tzdb_0.3.0
## [31] dbplyr_2.3.0         withr_2.5.0         grid_4.2.2
## [34] jsonlite_1.8.4       gtable_0.3.1        lifecycle_1.0.3
## [37] DBI_1.1.3            magrittr_2.0.3      scales_1.2.1
## [40] cli_3.6.0            stringi_1.7.12      fs_1.6.0
## [43] xml2_1.3.3           ellipsis_0.3.2      generics_0.1.3
## [46] vctrs_0.5.2          RColorBrewer_1.1-3  tools_4.2.2
## [49] glue_1.6.2           hms_1.1.2           fastmap_1.1.0
## [52] yaml_2.3.7           timechange_0.2.0    colorspace_2.1-0
## [55] gargle_1.2.1         rvest_1.0.3         knitr_1.42
```

Working with data

1. The data set rnf6080.dat records hourly rainfall at a certain location in Canada, every day from 1960 to 1980.

   a. Load the data set into R and make it a data frame called rain.df. What command did you use?

```
rain.df <- read.table("rnf6080.dat")
head(rain.df)
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
## 1 60  4  1  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 3 60  4  3  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 4 60  4  4  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 5 60  4  5  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 6 60  4  6  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
##   V22 V23 V24 V25 V26 V27
## 1   0   0   0   0   0   0
## 2   0   0   0   0   0   0
## 3   0   0   0   0   0   0
## 4   0   0   0   0   0   0
## 5   0   0   0   0   0   0
## 6   0   0   0   0   0   0
```

   b. How many rows and columns does rain.df have? How do you know? (If there are not 5070 rows and 27 columns, you did something wrong in the first part of the problem.)

```
nrow(rain.df)
```

```
## [1] 5070
```

```
ncol(rain.df)
```

```
## [1] 27
```

   c. What command would you use to get the names of the columns of rain.df? What are those names?

```
colnames(rain.df)
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12"
## [13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24"
## [25] "V25" "V26" "V27"
```

   d. What command would you use to get the value at row 2, column 4? What is the value?

```
rain.df[2, 4]
```

```
## [1] 0
```

   e. What command would you use to display the whole second row? What is the content of that row?

```
rain.df[2, ]
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
## 2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
##   V22 V23 V24 V25 V26 V27
## 2   0   0   0   0   0   0
```

f. What does the following command do?

```
names(rain.df) <- c("year", "month", "day", seq(0, 23))
```

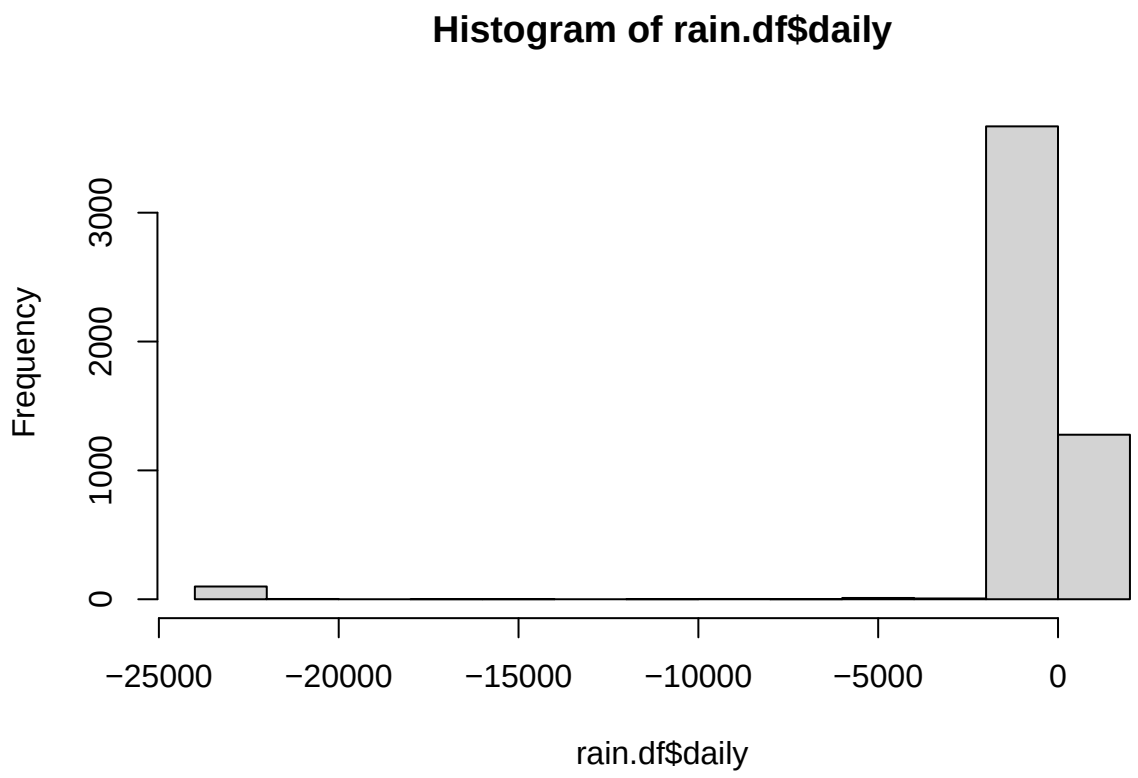
It renames column names to Year Month Day and 0 to 23 for the rest.

g. Create a new column called daily, which is the sum of the 24 hourly columns.

```
rain.df <- mutate(rain.df, daily = `0` + `1` + `2` + `3` + `4` +
    `5` + `6` + `7` + `8` + `9` + `10` + `11` + `12` + `13` +
    `14` + `15` + `16` + `17` + `18` + `19` + `20` + `21` + `22` +
    `23`)
```

h. Give the command you would use to create a histogram of the daily rainfall amounts. Please make sure to attach your figures in your .pdf report.

```
hist(rain.df$daily)
```



i. Explain why that histogram above cannot possibly be right.

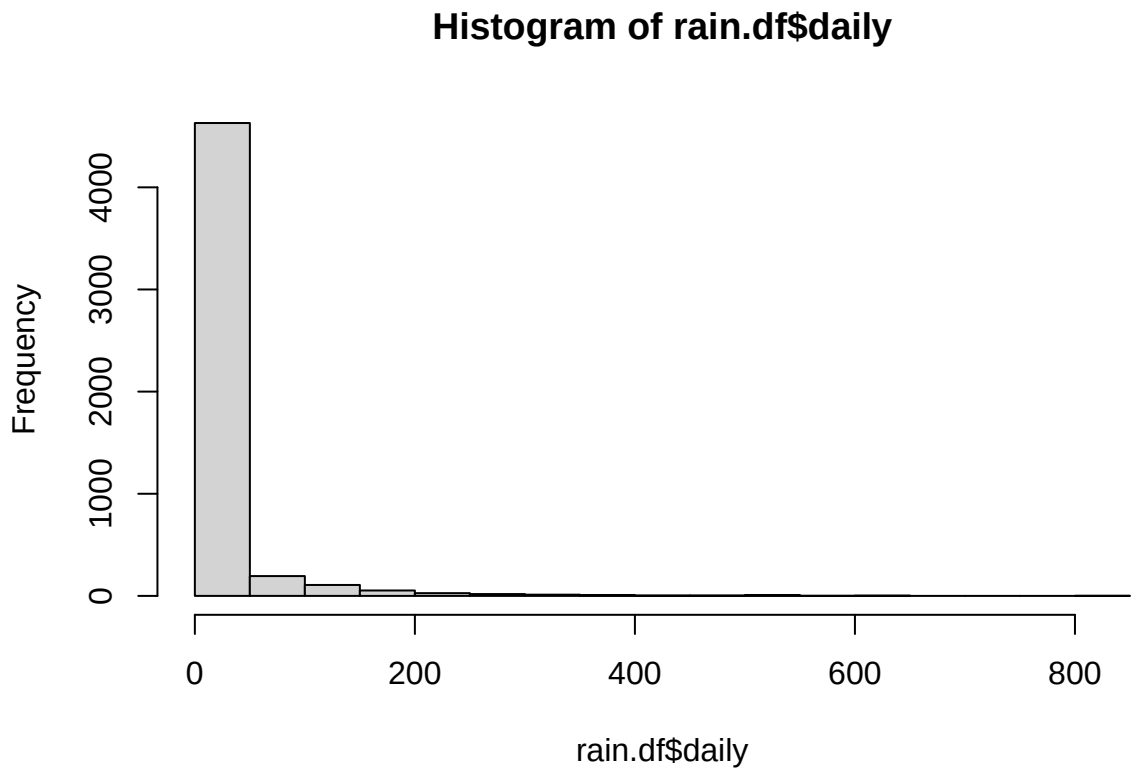
It is not correct because rainfall cant be negative. This is occuring due to NA being replaced as -999 in data.

j. Give the command you would use to fix the data frame.

```
rain.df[rain.df == -999] <- NA
rain.df$daily <- c(rowSums(rain.df[, 4:27], na.rm = TRUE))
```

k. Create a corrected histogram and again include it as part of your submitted report. Explain why it is more reasonable than the previous histogram.

```
hist(rain.df$daily)
```



This is more reasonable graph because it is showing the right skewed normal distribution without any obvious abnormalities like negative rainfall.

Data types

2. Make sure your answers to different parts of this problem are compatible with each other.

a. For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

```
x <- c("5", "12", "7")
max(x)
sort(x)
sum(x)
```

There are errors, because we are intializing the numbers to be character strings.

max(x) finds max in numeric vector. On character vector it returns character with highest index.

sort(x) sort in lexograpical order in character vector. So result would be {“12”, “5”, “7”}

sum(x) will throw error because of invalid type of argument.

b. For the next two commands, either explain their results, or why they should produce errors.

```
y <- c("5", 7, 12)
y[2] + y[3]
```

We are creating a vector y with three elements, where the first element is a character string, and rest numeric.

y[2] + y[3] will throw an error since y[1] is character string.

c. For the next two commands, either explain their results, or why they should produce errors.

```
z <- data.frame(z1 = "5", z2 = 7, z3 = 12)
z[1, 2] + z[1, 3]
```

```
## [1] 19
```

We are creating a data frame z with three columns and one row. The first column is a character string, and rest numeric.

z[1,2] + z[1,3] result 19, unlike above case in this case each vector is taken column wise so errors pop up.

Linear algebra

3. Consider the linear system AX = b. Please answer the following questions

a. Write an R function called my_solver() such that given inputs A and b, the function my_solver() returns the solution of the linear system

```
my_solver <- function(A, b) {
    X <- solve(A, b)
    return(X)
}
```

b. Run the following code to get A and b.

```
n <- 100
set.seed(123)
A <- rWishart(1, 150, diag(n))[, , 1]
b <- rnorm(n, 1)
X <- my_solver(A, b)
```

Verifying

```
AX = A %*% X
AX <- as.numeric(AX)
all.equal(AX, b, tolerance = 1e-06)
```

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

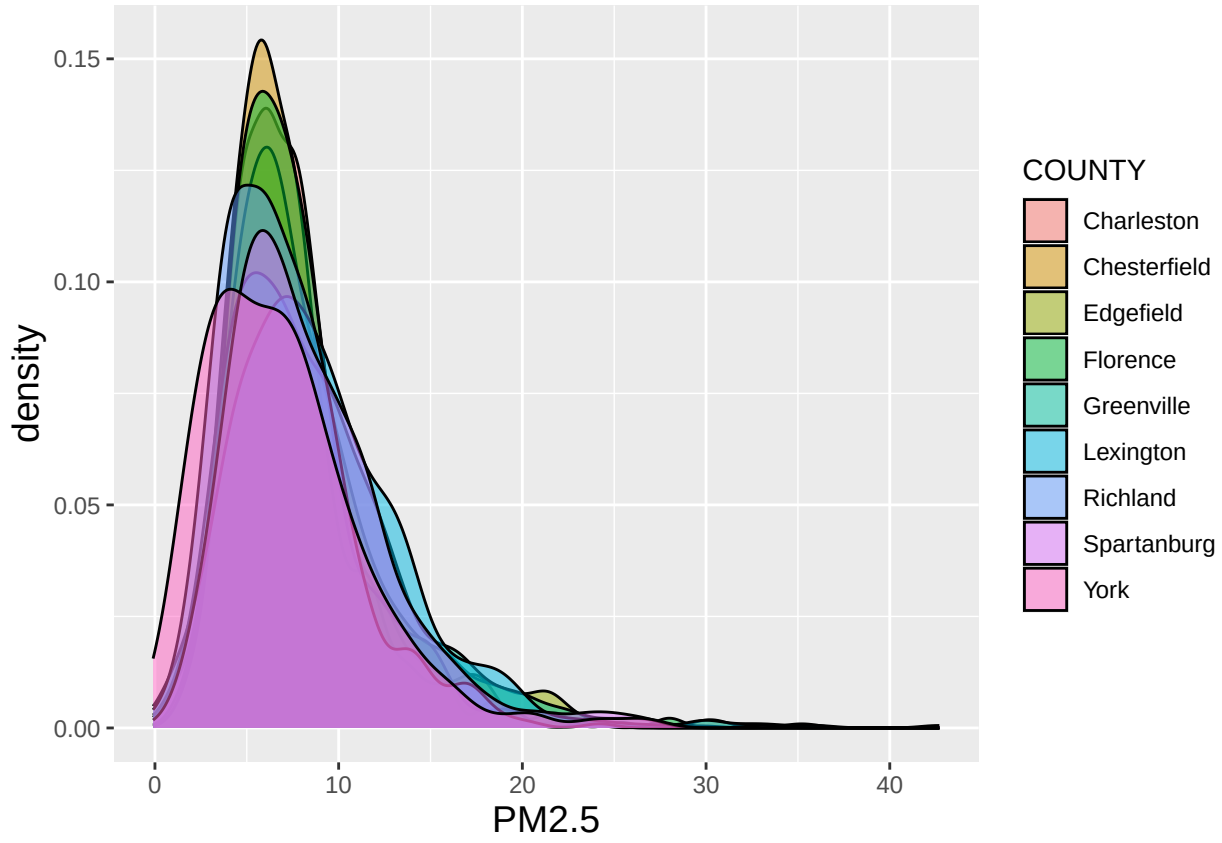
Working with ggplot2

4. EPA monitors Air Quality data across the entire U.S. The file "AQSdata.csv" contains daily PM 2.5 concentrations and other information. Please make the following questions using the ggplot() function for plotting. In addition make sure that all the x-axis and y-axis labels have 14 font size.

a. Read the data file "AQSdata.csv" into R.

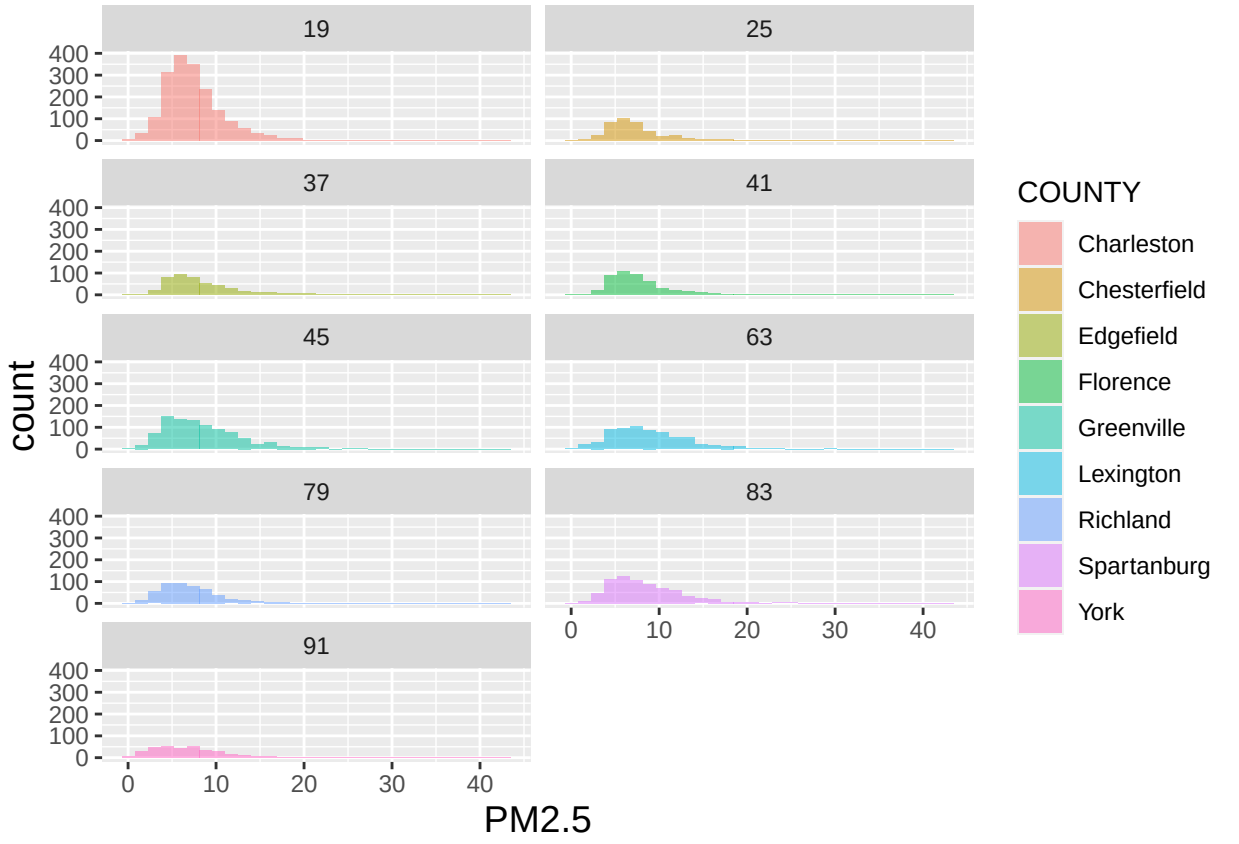
b. Generate density plots of PM2.5 concentrations grouped by County in one single panel, where each density should have its own color. What do you find from the figure?

```
ggplot(AQSdata, aes(x = PM2.5, fill = COUNTY)) + geom_density(alpha = 0.5) +
    theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14))
```



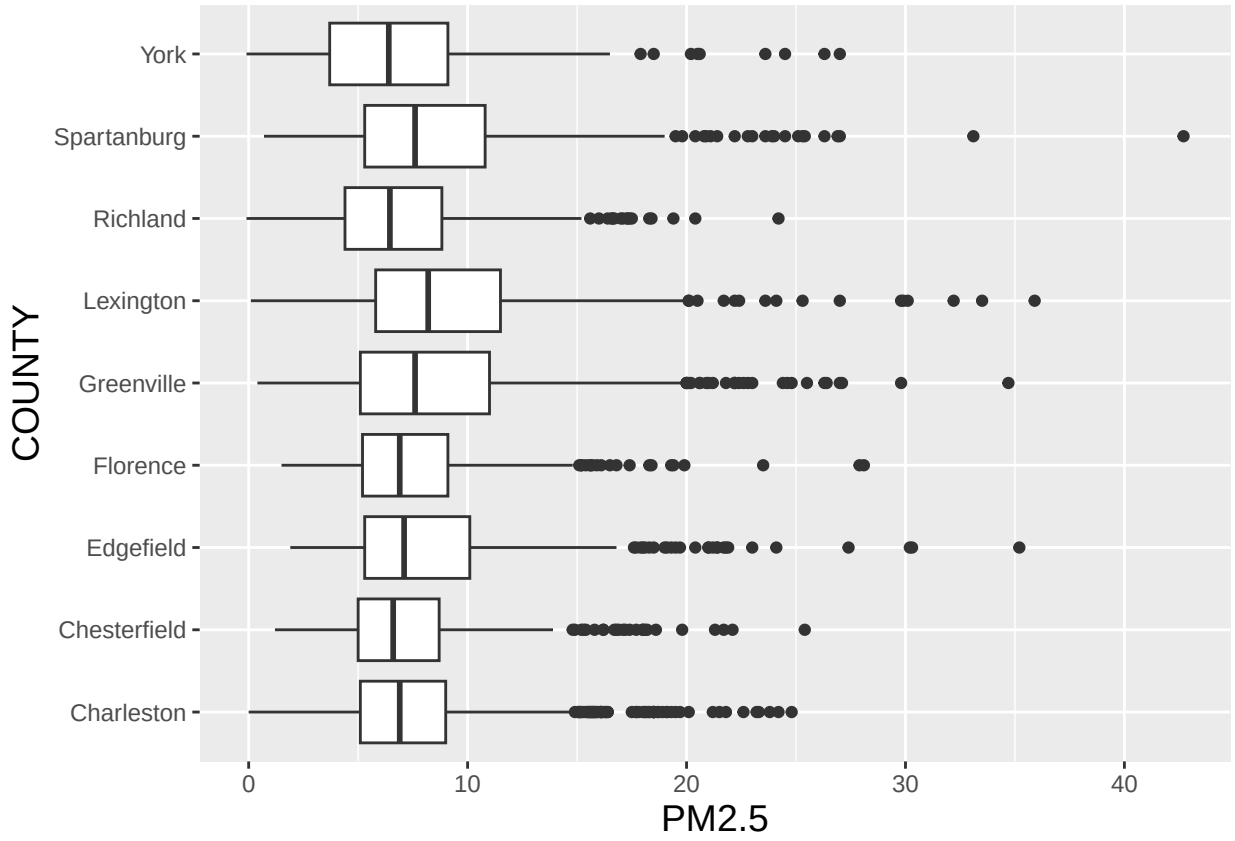
c. Plot histograms of PM2.5 concentrations across different counties with one panel for one histogram.

```
ggplot(AQSdata, aes(x = PM2.5, fill = COUNTY)) + geom_histogram(alpha = 0.5) +
    theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14)) +
    facet_wrap(~COUNTY_CODE, ncol = 2)
```



d. Generate boxplots of PM2.5 concentrations by County. What would you say about the distributions?

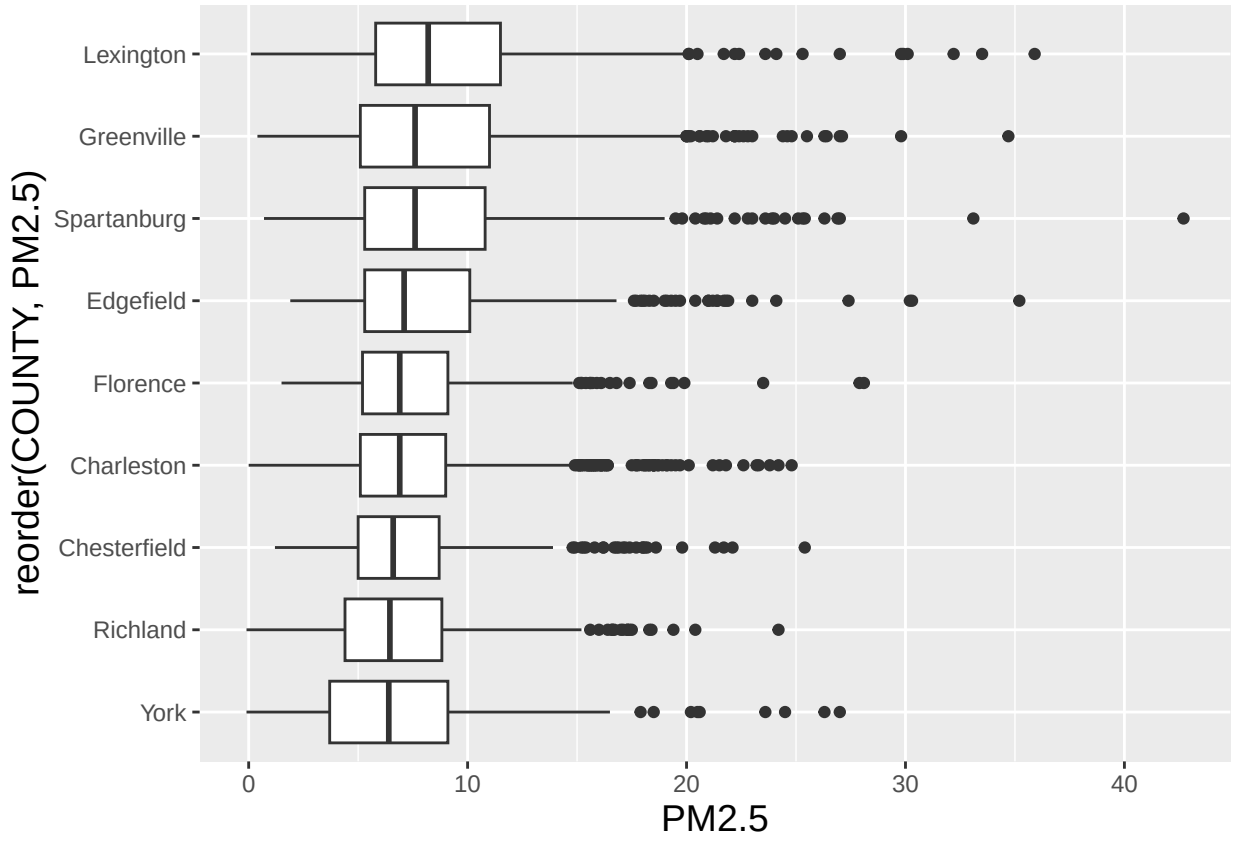
```
# Fill = COUNTY for Legend
ggplot(AQSdata, aes(x = PM2.5, y = COUNTY)) + geom_boxplot() +
    theme(axis.title.x = element_text(size = 14), axis.title.y = element_text(size = 14))
```



e. Reorder the boxplots above by the median value of PM2.5 concentrations.

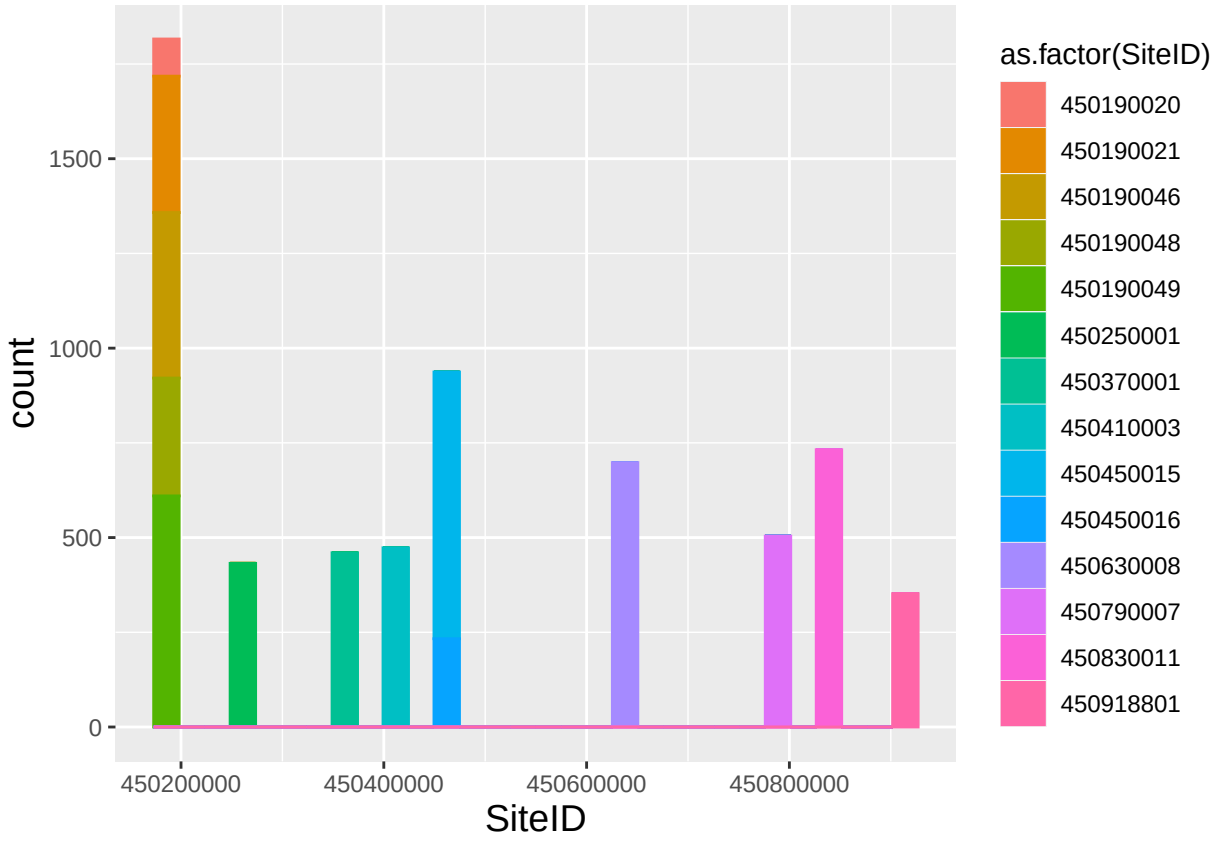
```
aqs_data_median <- aggregate(PM2.5 ~ COUNTY, data = AQSdata,
    median)
aqs_data_median$County <- reorder(aqs_data_median$COUNTY, aqs_data_median$PM2.5)

ggplot(AQSdata, aes(x = reorder(COUNTY, PM2.5), y = PM2.5)) +
    geom_boxplot() + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + coord_flip()
```



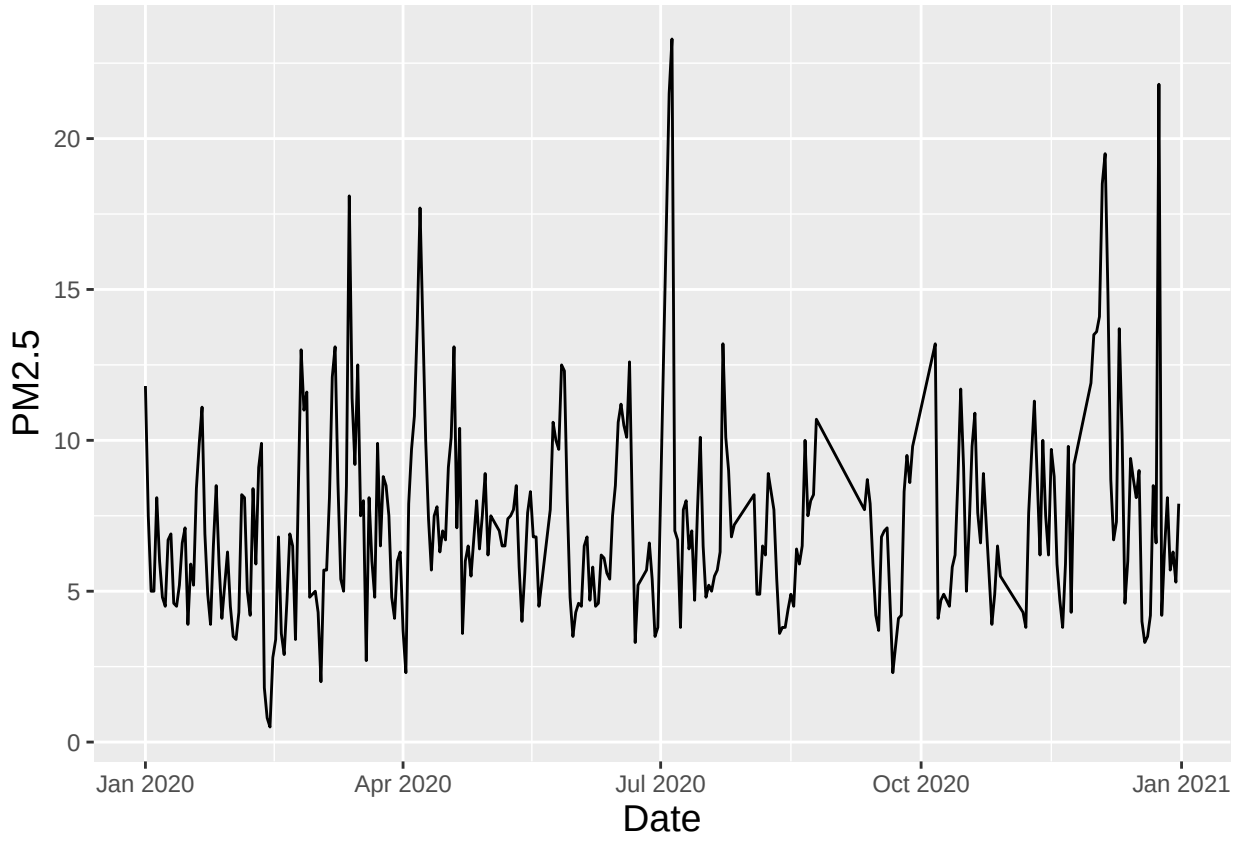
f. Converting the Site ID to a factor and plot the histogram grouped by Site ID.

```
ggplot(AQSdata, aes(x = SiteID, color = as.factor(SiteID), fill = as.factor(SiteID))) +
    geom_histogram() + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14))
```



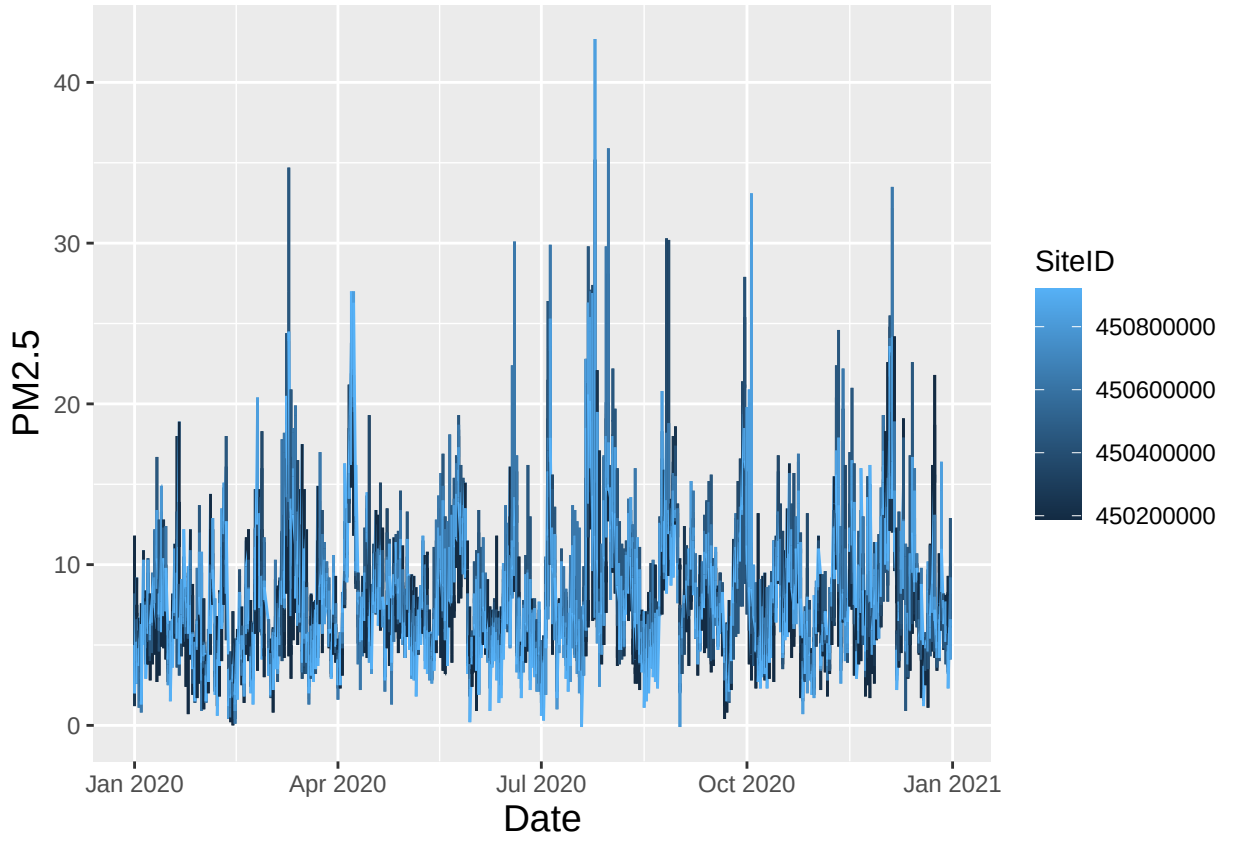
g. Generate the time series plot for the monitoring Site ID 450190048.

```
AQSdata_site <- AQSdata[AQSdata$SiteID == "450190048", ]
ggplot(AQSdata_site, aes(x = as.Date(Date, "%m/%d/%y"), y = PM2.5)) +
    geom_line() + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + xlab("Date")
```



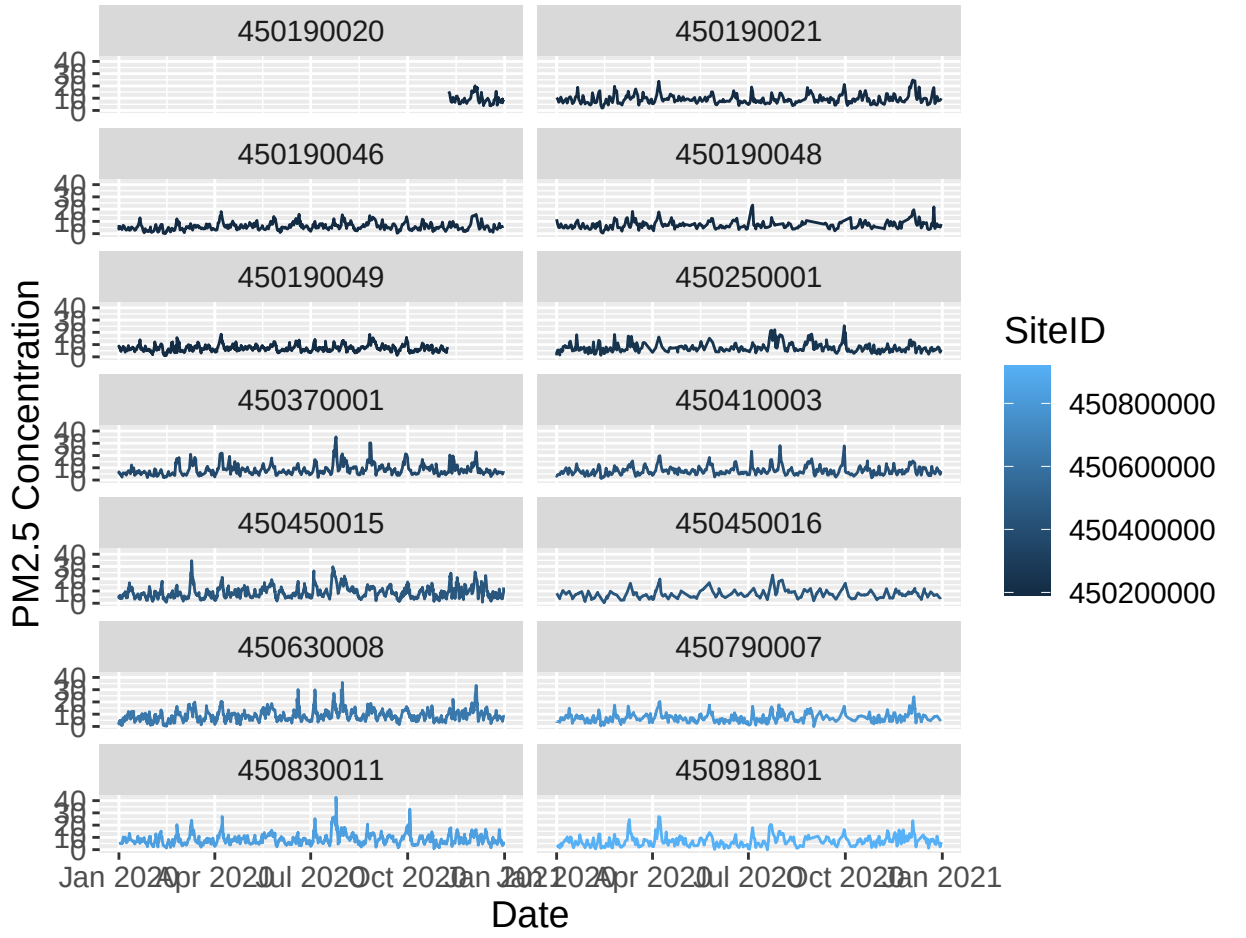
h. Plot time series of PM2.5 concentrations for all monitoring sites in one panel, where each site has its own color

```
ggplot(AQSdata, aes(x = as.Date(Date, "%m/%d/%y"), y = PM2.5,
    color = SiteID)) + geom_line() + theme(axis.title.x = element_text(size = 14),
    axis.title.y = element_text(size = 14)) + xlab("Date")
```



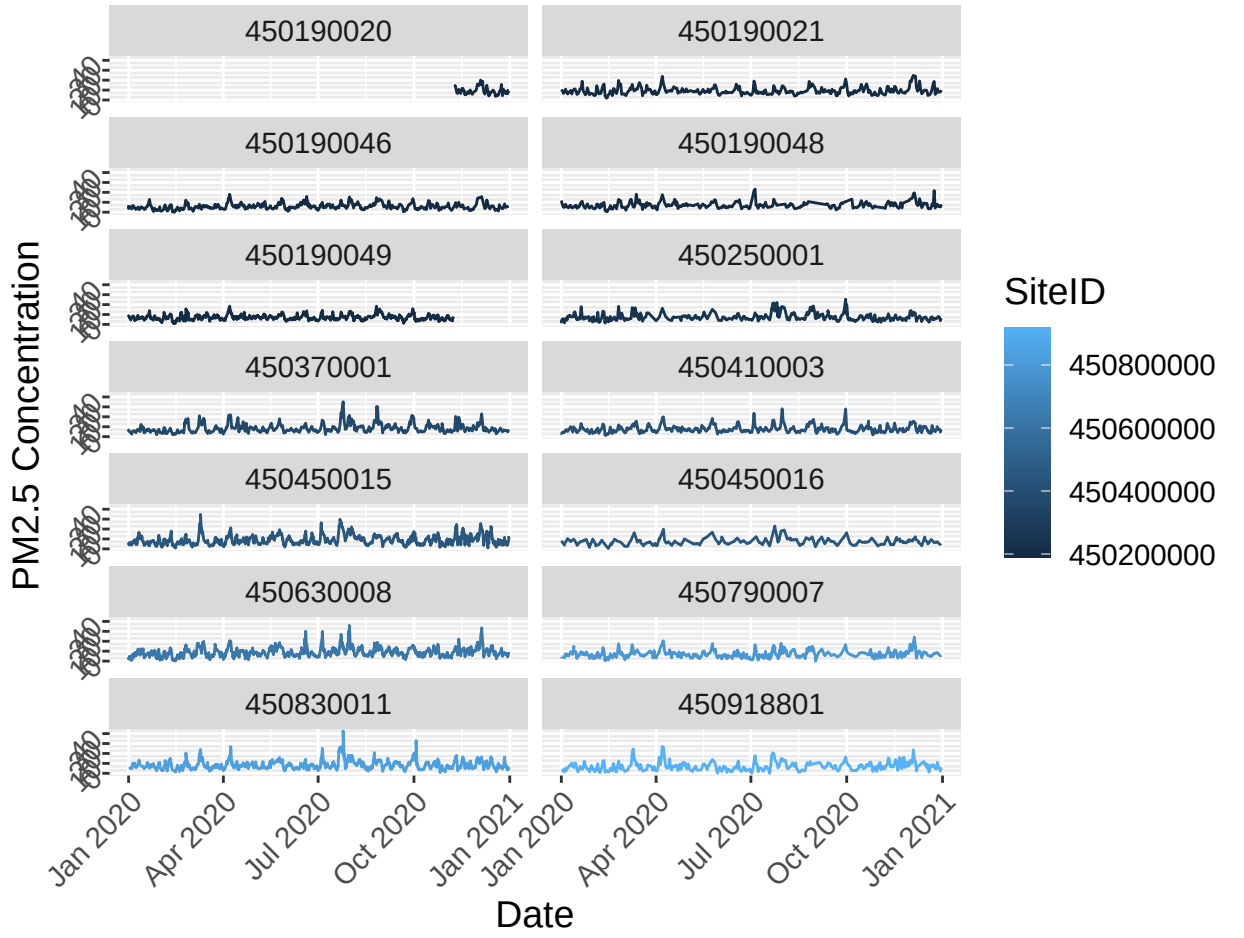
i. Plot time series of PM2.5 concentrations across all monitoring sites in multiple panels, where one panel only has one site, and each row only has two panels.

```
ggplot(AQSdata, aes(x = as.Date(Date, "%m/%d/%y"), y = PM2.5,
    color = SiteID)) + geom_line() + theme(text = element_text(size = 14)) +
    xlab("Date") + ylab("PM2.5 Concentration") + facet_wrap(~SiteID,
    ncol = 2)
```



j. In the time series plot, there seems to be not enough space to hold the x-axis labels. One way to avoid this is to rotate the axis labels. Please rotate all the time labels 45 degree.

```
ggplot(AQSdata, aes(x = as.Date(Date, "%m/%d/%y"), y = PM2.5,
    color = SiteID)) + geom_line() + theme(text = element_text(size = 14)) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    theme(axis.text.y = element_text(angle = 45, hjust = 1)) +
    xlab("Date") + ylab("PM2.5 Concentration") + facet_wrap(~SiteID,
    ncol = 2)
```



Working with dplyr

5. Continuing working with the above PM 2.5 data.

a. Filter all the observations in the county Greenville. How many observations are there?

```
greenville_data <- AQSdata %>%
    filter(COUNTY == "Greenville")

nrow(greenville_data)
```

```
## [1] 937
```

b. Filter all the observations in Greenville in August 2021

```
# Convert date to date format
AQSdata$Date <- mdy(AQSdata$Date)

# Filter all the observations in Greenville in August 2021
greenville_aug_2021 <- AQSdata %>%
```

```
    filter(COUNTY == "Greenville") %>%
    filter(month(Date) == 8 & year(Date) == 2021)

# Check number of observations
nrow(greenville_aug_2021)
```

```
## [1] 82
```

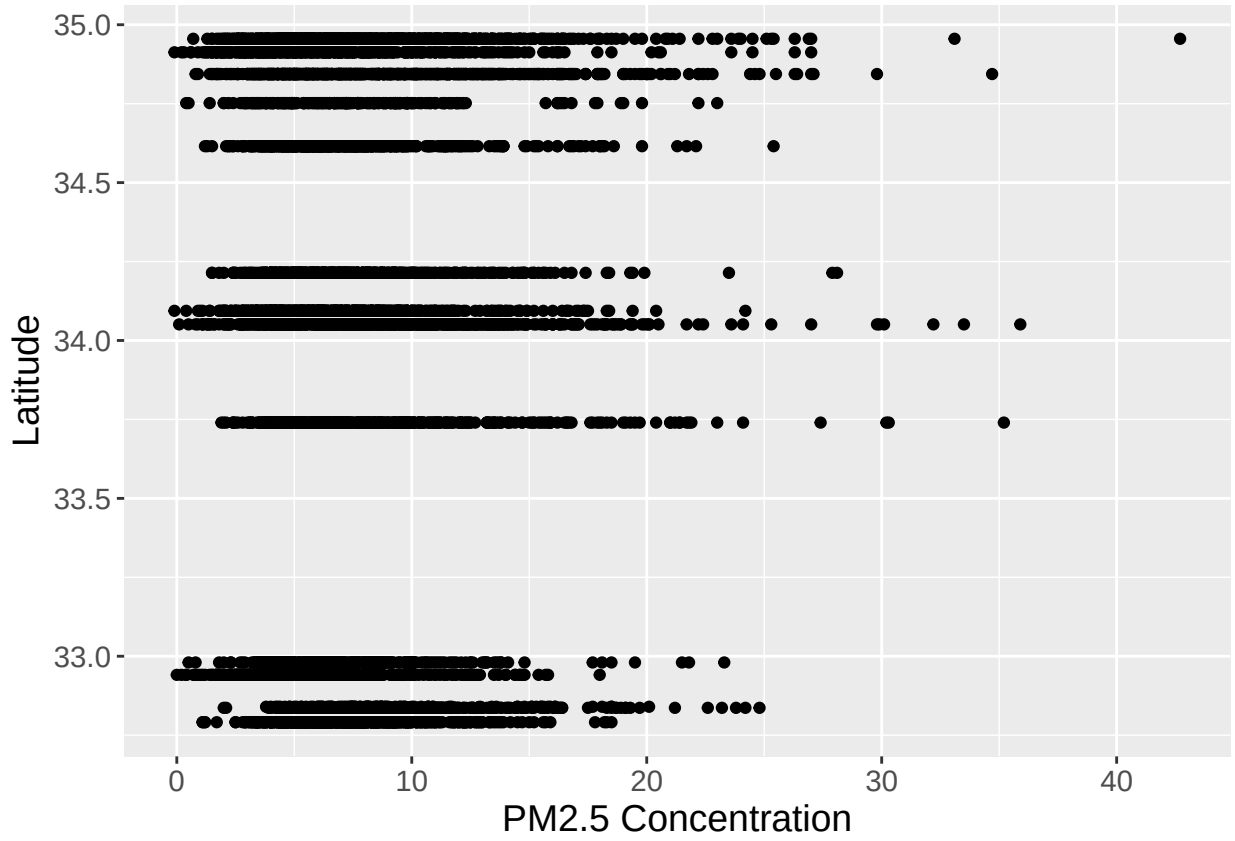
c. Filter all the observations in Greenville in August 2021 and select the variables PM2.5 concentrations, Date, latitude and longitude of sites

```
greenville_aug_2021_vars <- greenville_aug_2021 %>%
    select(Date, PM2.5, SITE_LATITUDE, SITE_LONGITUDE)
head(greenville_aug_2021_vars)
```

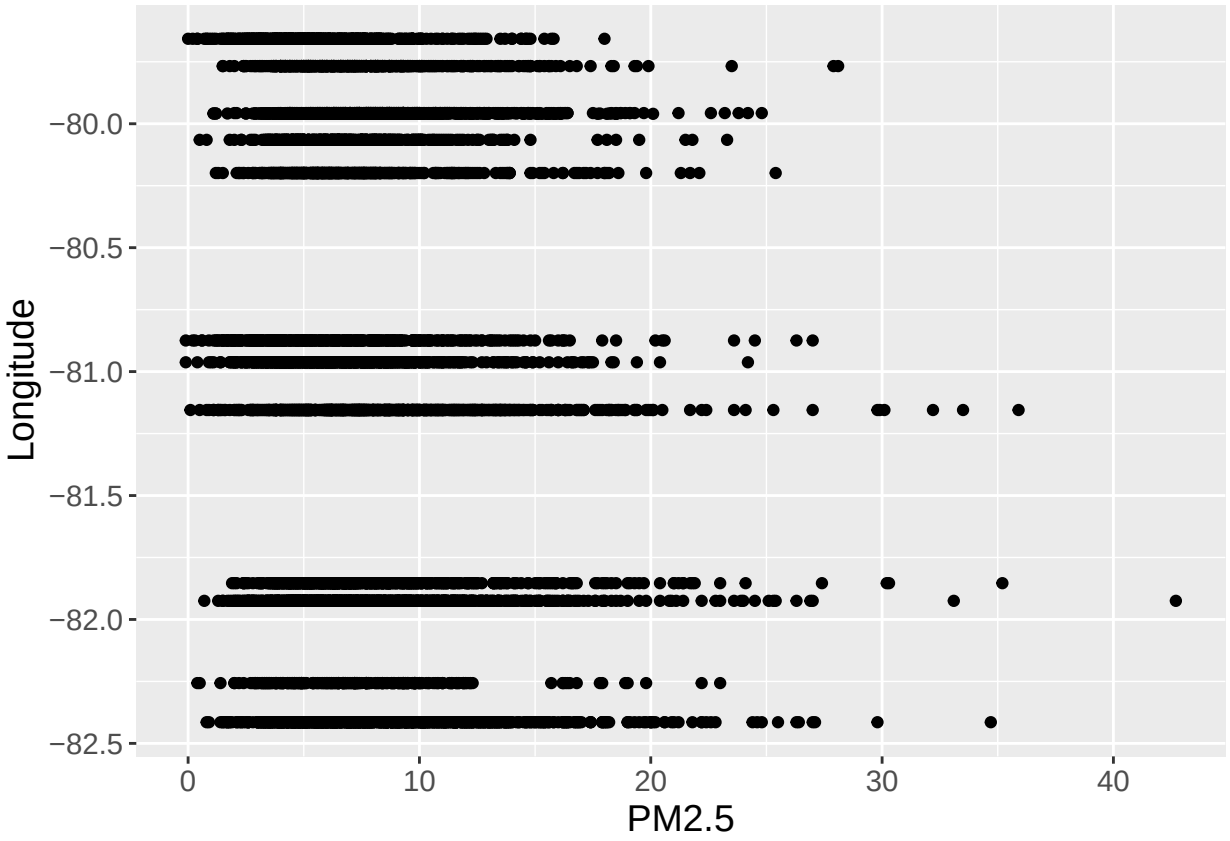
```
##         Date PM2.5 SITE_LATITUDE SITE_LONGITUDE
## 1 2021-08-01  13.8       34.8439      -82.41458
## 2 2021-08-02  19.0       34.8439      -82.41458
## 3 2021-08-03  16.9       34.8439      -82.41458
## 4 2021-08-04  15.6       34.8439      -82.41458
## 5 2021-08-05  11.0       34.8439      -82.41458
## 6 2021-08-06  10.3       34.8439      -82.41458
```

d. Generate scatter plot of PM2.5 against latitude and longitude in two different panels

```
ggplot(AQSdata, aes(x = SITE_LATITUDE, y = PM2.5)) + geom_point() +
    theme(text = element_text(size = 14)) + xlab("Latitude") +
    ylab("PM2.5 Concentration") + coord_flip()
```




```
ggplot(AQSdata, aes(x = SITE_LONGITUDE, y = PM2.5)) + geom_point() +
    theme(text = element_text(size = 14)) + xlab("Longitude") +
    ylab("PM2.5") + coord_flip()
```



About Assignment

6.

a. What is the point of reproducible code?

The point of reproducible code is to make our life easier. Suppose having a certain set of libraries and formatting types in .rmd file helps a lot compared to selecting those every single time.

b. Given an example of why making your code reproducible is important for you to know in this class and moving forward.

As explained in former question, reproducibility helps save time by helping us focus on data rather than worrying about whether a similar dataset is loaded correctly or not.

c. On a scale of 1 (easy) – 10 (hard), how hard was this assignment. If this assignment was hard (> 5), please state in one sentence what you struggled with.

7

```
rain.df <- mutate(rain.df, daily = `0` + `1` + `2` + `3` + `4` +
    `5` + `6` + `7` + `8` + `9` + `10` + `11` + `12` + `13` +
    `14` + `15` + `16` + `17` + `18` + `19` + `20` + `21` + `22` +
    `23`)
```

To find out about how to give numeric column names to a function like above took me about 1 hour of Googling and Stackoverflow.