Assignment 2

Conor Heffron (23211267)

Task 1: Manipulation

1. Load the dataset dublin-bikes.txt, save it as a tibble and give meaningful names to the variables related to the weather.

```r
# Load dataset
library(readr)
dublin_bikes <- read_delim("./dublin-bikes-v2.txt", delim = "\t", escape_double = FALSE, c

# Save as tibble
library(tibble)
db_tib = as_tibble(dublin_bikes)

# Print first 10 rows
head(db_tib, 10)
```

```
# A tibble: 10 x 12
   Time     rain  temp  wdsp clamt Clontarf - James Lar~1 Clontarf - Pebble Be~2
   <chr>   <dbl> <dbl> <dbl> <dbl>                  <dbl>                  <dbl>
 1 2022-0~     0  13       6     6                      6                      8
 2 2022-0~     0  13.6     7     6                      1                      2
 3 2022-0~     0  14       6     6                      1                      2
 4 2022-0~     0  14.4     5     6                      0                      0
 5 2022-0~     0  14.4     5     7                      1                      3
 6 2022-0~     0  13.5     6     7                     21                     26
 7 2022-0~     0  14.2     6     7                     30                     44
 8 2022-0~     0  15       8     7                     89                    111
 9 2022-0~     0  15.5     9     7                    123                    170
10 2022-0~     0  16.4    11     5                     67                     90
# i abbreviated names: 1: `Clontarf - James Larkin Rd`,
#   2: `Clontarf - Pebble Beach Carpark`
# i 5 more variables: `Griffith Avenue (Clare Rd Side)` <dbl>,
#   `Griffith Avenue (Lane Side)` <dbl>, `Grove Road Totem` <dbl>,
#   `Richmond Street Cyclists 1` <dbl>, `Richmond Street Cyclists 2` <dbl>
```

2. What is the size (number of rows and columns) this dataset? Write some code to check that the variable Time is stored using an appropriate class for a date, and the other variables are numeric, fix them if they aren't.

- 8760 rows and 12 columns

```r
# Get dimensions of dublin bikes tibble (8760 rows and 12 columns)
dim(db_tib)
```

```
[1] 8760   12
```

```r
# Display structure of time column
str(db_tib["Time"])
```

```
tibble [8,760 x 1] (S3: tbl_df/tbl/data.frame)
 $ Time: chr [1:8760] "2022-09-01T00:00:00Z" "2022-09-01T01:00:00Z" "2022-09-01T02:00:00Z" "2
```

- Time column is character string instead of date/time value

```r
# Reformat time variable / column to POSIXct type
tz_chars <- c("T", "Z")
for (ch in tz_chars)
  db_tib["Time"] <- lapply(db_tib["Time"], function(x) gsub(ch, " ", x))
db_tib[['Time']] <- as.POSIXct(db_tib[['Time']], format = "%Y-%m-%d %H:%M:%S")

# Display structure of Dublin bikes tibble
str(db_tib)
```

```
tibble [8,760 x 12] (S3: tbl_df/tbl/data.frame)
 $ Time                          : POSIXct[1:8760], format: "2022-09-01 00:00:00" "2022-09-0
 $ rain                          : num [1:8760] 0 0 0 0 0 0 0 0 0 0 ...
 $ temp                          : num [1:8760] 13 13.6 14 14.4 14.4 13.5 14.2 15 15.5 16.4
 $ wdsp                          : num [1:8760] 6 7 6 5 5 6 6 8 9 11 ...
 $ clamt                         : num [1:8760] 6 6 6 6 7 7 7 7 7 5 ...
 $ Clontarf - James Larkin Rd    : num [1:8760] 6 1 1 0 1 21 30 89 123 67 ...
 $ Clontarf - Pebble Beach Carpark: num [1:8760] 8 2 2 0 3 26 44 111 170 90 ...
 $ Griffith Avenue (Clare Rd Side): num [1:8760] 0 0 0 0 0 0 0 0 0 0 ...
 $ Griffith Avenue (Lane Side)    : num [1:8760] 0 0 0 0 0 0 0 0 0 0 ...
 $ Grove Road Totem               : num [1:8760] 33 8 5 6 2 39 132 324 619 287 ...
 $ Richmond Street Cyclists 1     : num [1:8760] 25 3 7 7 2 2 9 43 81 42 ...
 $ Richmond Street Cyclists 2     : num [1:8760] 8 1 6 3 2 9 40 88 228 153 ...
```

3. Convert the variable containing the cloud amount information into an ordered factor. Print the levels and the output of a check to confirm it's ordered.

```
# clamt cloud amount (okta):
# - 0 oktas represents the complete absence of cloud
# - 1 okta represents a cloud amount of 1 eighth or less, but not zero
# - 7 oktas represents a cloud amount of 7 eighths or more, but not full cloud cover
# - 8 oktas represents full cloud cover with no breaks
# - 9 oktas represents sky obscured by fog or other meteorological phenomena

# Convert the variable containing the cloud amount information into an ordered factor
start <- min(db_tib$clamt)
stop <- max(db_tib$clamt)+1
db_tib$clamt <- factor(db_tib$clamt, levels = start:stop, ordered = TRUE)
```

```
# Print the levels and the output of a check to confirm it's ordered.
print(levels(db_tib$clamt))
```

```
 [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```
print(paste("db_tib$clamt ordered? ", is.ordered(db_tib$clamt)))
```

```
[1] "db_tib$clamt ordered?  TRUE"
```

4. Split the information in the column Time into two columns: one containing the date (i.e. date only, no time), and the other the hour. Check that there are 24 hours for each date, and that there are 365 different dates.

```
db_tib$Date <- as.Date(db_tib$Time)
db_tib$Hour <- format(as.POSIXct(db_tib$Time), format = "%H:%M:%S")
```

```
# Load dplyr
library(dplyr)
```

```
Attaching package: 'dplyr'


The following objects are masked from 'package:stats':

    filter, lag


The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
# do count by date
db_tib %>%
  count(Date)
```

```
# A tibble: 367 x 2
   Date           n
   <date>     <int>
 1 2022-08-31     1
 2 2022-09-01    24
 3 2022-09-02    24
 4 2022-09-03    24
 5 2022-09-04    24
 6 2022-09-05    24
 7 2022-09-06    24
 8 2022-09-07    24
 9 2022-09-08    24
10 2022-09-09    24
# i 357 more rows
```

```r
# Note: There was an NA value for date with count 1 introduced by v2 file

# Omit rows with NA in any column of data frame
db_tib <- na.omit(db_tib)

# do unique count on Date only column
# (366 and not 355 because data is inclusive of 31st of August in 2022 and 2023)
length(unique(db_tib$Date))
```

```
[1] 366
```

5. Add two columns one containing the day of the week and the other the month. Check that these two columns are ordered factors.

```r
# Extract day of week and month columns from Time variable
db_tib$day_of_week <- weekdays(db_tib$Date)
db_tib$month <- months(db_tib$Date)
```

```r
# Convert day of week variable to ordered factor
db_tib$day_of_week <- factor(db_tib$day_of_week, levels = unique(weekdays(db_tib$Date)), o
print(paste("db_tib$day_of_week ordered? ", is.ordered(db_tib$day_of_week)))
```

```
[1] "db_tib$day_of_week ordered?  TRUE"
```

```r
# Convert month variable to ordered factor
db_tib$month <- factor(db_tib$month, levels = unique(months(db_tib$Date)), ordered = TRUE)
print(paste("db_tib$month? ", is.ordered(db_tib$month)))
```

```
[1] "db_tib$month?  TRUE"
```

```r
# Check tibble structure and variable types
str(db_tib)
```

```
tibble [8,727 x 16] (S3: tbl_df/tbl/data.frame)
 $ Time                              : POSIXct[1:8727], format: "2022-09-01 00:00:00" "2022-09-0
 $ rain                              : num [1:8727] 0 0 0 0 0 0 0 0 0 0 ...
 $ temp                              : num [1:8727] 13 13.6 14 14.4 14.4 13.5 14.2 15 15.5 16.4
 $ wdsp                              : num [1:8727] 6 7 6 5 5 6 6 8 9 11 ...
 $ clamt                             : Ord.factor w/ 10 levels "0"<"1"<"2"<"3"<..: 7 7 7 7 8 8
 $ Clontarf - James Larkin Rd        : num [1:8727] 6 1 1 0 1 21 30 89 123 67 ...
 $ Clontarf - Pebble Beach Carpark: num [1:8727] 8 2 2 0 3 26 44 111 170 90 ...
 $ Griffith Avenue (Clare Rd Side): num [1:8727] 0 0 0 0 0 0 0 0 0 0 ...
 $ Griffith Avenue (Lane Side)     : num [1:8727] 0 0 0 0 0 0 0 0 0 0 ...
 $ Grove Road Totem                  : num [1:8727] 33 8 5 6 2 39 132 324 619 287 ...
 $ Richmond Street Cyclists 1      : num [1:8727] 25 3 7 7 2 2 9 43 81 42 ...
 $ Richmond Street Cyclists 2      : num [1:8727] 8 1 6 3 2 9 40 88 228 153 ...
 $ Date                              : Date[1:8727], format: "2022-08-31" "2022-09-01" ...
 $ Hour                              : chr [1:8727] "00:00:00" "01:00:00" "02:00:00" "03:00:00"
 $ day_of_week                       : Ord.factor w/ 7 levels "Wednesday"<"Thursday"<..: 1 2 2
 $ month                             : Ord.factor w/ 12 levels "August"<"September"<..: 1 2 2 2
 - attr(*, "na.action")= 'omit' Named int [1:33] 1035 1036 1037 1038 1039 1040 1041 1042 104
  ..- attr(*, "names")= chr [1:33] "1035" "1036" "1037" "1038" ...
```

6. Remove the column Time and use dplyr::relocate() to put the new columns with the date, hour, day of the week, and month as the first four columns of the dataset.

```
# Drop Time column
db_tib <- select(db_tib, -Time)
```

```
# relocate Date, Hour, day_of_week, month to the front of tibble
db_tib <- db_tib %>%
  relocate(Date, Hour, day_of_week, month)

# Sanity check of tibble head data
head(db_tib)
```

```
# A tibble: 6 x 15
  Date       Hour     day_of_week month      rain  temp  wdsp clamt
  <date>     <chr>    <ord>       <ord>     <dbl> <dbl> <dbl> <ord>
1 2022-08-31 00:00:00 Wednesday   August        0  13        6 6
2 2022-09-01 01:00:00 Thursday    September     0  13.6      7 6
3 2022-09-01 02:00:00 Thursday    September     0  14        6 6
4 2022-09-01 03:00:00 Thursday    September     0  14.4      5 6
5 2022-09-01 04:00:00 Thursday    September     0  14.4      5 7
6 2022-09-01 05:00:00 Thursday    September     0  13.5      6 7
# i 7 more variables: `Clontarf - James Larkin Rd` <dbl>,
#   `Clontarf - Pebble Beach Carpark` <dbl>,
#   `Griffith Avenue (Clare Rd Side)` <dbl>,
#   `Griffith Avenue (Lane Side)` <dbl>, `Grove Road Totem` <dbl>,
#   `Richmond Street Cyclists 1` <dbl>, `Richmond Street Cyclists 2` <dbl>
```

```
tail(db_tib)
```

```
# A tibble: 6 x 15
  Date       Hour     day_of_week month   rain  temp  wdsp clamt
  <date>     <chr>    <ord>       <ord>  <dbl> <dbl> <dbl> <ord>
1 2023-08-31 18:00:00 Thursday    August   0    15.3     6 7
2 2023-08-31 19:00:00 Thursday    August   0    14.9     7 7
3 2023-08-31 20:00:00 Thursday    August   0    14.6     5 7
4 2023-08-31 21:00:00 Thursday    August   0    14.7     4 7
```

```
5 2023-08-31 22:00:00 Thursday    August   0.4  14.5     2 7
6 2023-08-31 23:00:00 Thursday    August   0.1  14.5     3 7
# i 7 more variables: `Clontarf - James Larkin Rd` <dbl>,
#   `Clontarf - Pebble Beach Carpark` <dbl>,
#   `Griffith Avenue (Clare Rd Side)` <dbl>,
#   `Griffith Avenue (Lane Side)` <dbl>, `Grove Road Totem` <dbl>,
#   `Richmond Street Cyclists 1` <dbl>, `Richmond Street Cyclists 2` <dbl>
```

Task 2: Analysis

1. Use functions from base R to compute which month had in total the highest and the lowest Precipitation Amount.

```r
# base r aggregate function
df_rain <- aggregate(as.numeric(db_tib$rain), by=list(Category=db_tib$month), FUN=sum)
```

```r
# Highest precipitation amount
df_rain[which.max(df_rain$x),]
```

```
   Category     x
12     July 149.2
```

```r
# Lowest precipitation amount
df_rain[which.min(df_rain$x),]
```

```
  Category    x
7 February 16.2
```

```r
# Via dplyr in one shot
df_rain2 <- db_tib %>%
  group_by(month) %>%
  summarise(rain_sum = sum(as.numeric(rain))) %>%
  arrange(desc(rain_sum)) %>%
  filter(row_number()==1 | row_number()==n())

df_rain2
```

```
# A tibble: 2 x 2
  month    rain_sum
  <ord>       <dbl>
1 July        149.
2 February     16.2
```

2. Use ggplot2 to create a time series plot of the maximum and minimum daily temperatures. The two time series must be on the same plot.

```
# Add maximum and minimum temperature columns
temps_df <- db_tib %>%
  group_by(day_of_week) %>%
  mutate(max_temp = max(temp), min_temp = min(temp))

temps_df
```

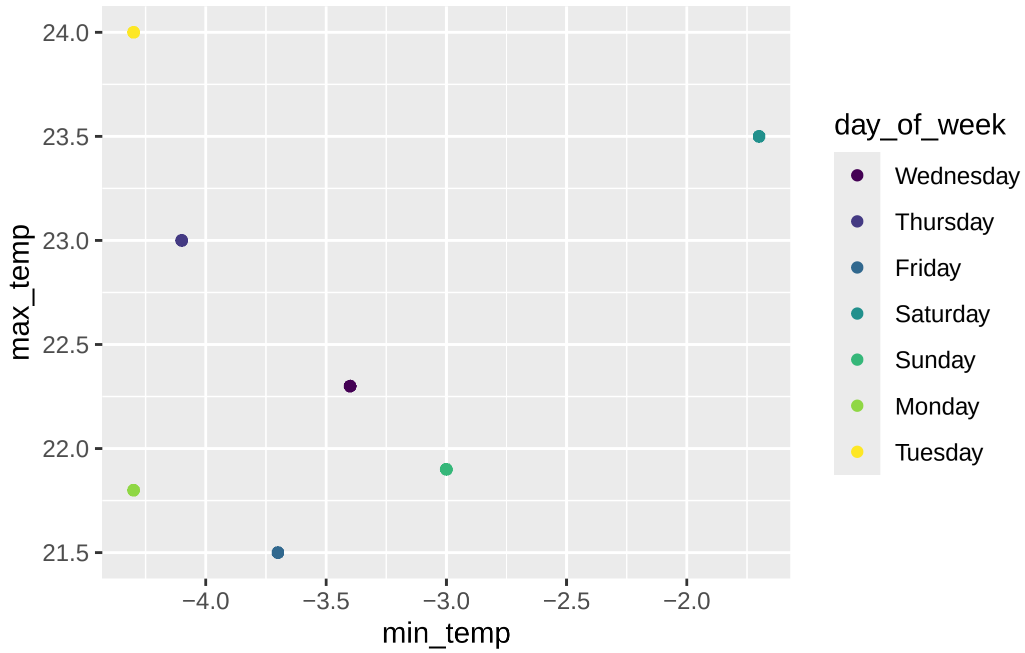
```
# A tibble: 8,727 x 17
# Groups:   day_of_week [7]
   Date       Hour     day_of_week month      rain  temp  wdsp clamt
   <date>     <chr>    <ord>       <ord>     <dbl> <dbl> <dbl> <ord>
 1 2022-08-31 00:00:00 Wednesday   August        0  13       6 6
 2 2022-09-01 01:00:00 Thursday    September     0  13.6     7 6
 3 2022-09-01 02:00:00 Thursday    September     0  14       6 6
 4 2022-09-01 03:00:00 Thursday    September     0  14.4     5 6
 5 2022-09-01 04:00:00 Thursday    September     0  14.4     5 7
 6 2022-09-01 05:00:00 Thursday    September     0  13.5     6 7
 7 2022-09-01 06:00:00 Thursday    September     0  14.2     6 7
 8 2022-09-01 07:00:00 Thursday    September     0  15       8 7
 9 2022-09-01 08:00:00 Thursday    September     0  15.5     9 7
10 2022-09-01 09:00:00 Thursday    September     0  16.4    11 5
# i 8,717 more rows
# i 9 more variables: `Clontarf - James Larkin Rd` <dbl>,
#   `Clontarf - Pebble Beach Carpark` <dbl>,
#   `Griffith Avenue (Clare Rd Side)` <dbl>,
#   `Griffith Avenue (Lane Side)` <dbl>, `Grove Road Totem` <dbl>,
#   `Richmond Street Cyclists 1` <dbl>, `Richmond Street Cyclists 2` <dbl>,
#   max_temp <dbl>, min_temp <dbl>
```

```
# Create and display plot
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.2
```

```
temps_plot <- ggplot(temps_df, aes(x = min_temp, y = max_temp, color = day_of_week))

temps_plot + geom_point()
```



3. Check if, according to this dataset, there has been on average more rain during the weekend (Sat-Sun) with respect to weekdays (Mon-Fri).

```
# Create weekdays vector
weekdays_v <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")

# calculate rainfall for weekdays
rain_weekdays <- db_tib %>%
  group_by(day_of_week) %>%
  filter(day_of_week %in% weekdays_v) %>%
  summarise(weekdays_rainfall = sum(rain), weekdays_mean_rainfall = mean(rain))

# Initialize weekends vector
weekends_v <- c("Saturday", "Sunday")

# Calculate rainfall for weekends
```

```
rain_weekends <- db_tib %>%
  group_by(day_of_week) %>%
  filter(day_of_week %in% weekends_v) %>%
  summarise(weekends_rainfall = sum(rain), weekends_mean_rainfall = mean(rain))
```

```
# Weekdays rainfall
sum(rain_weekdays$weekdays_rainfall)
```

```
[1] 597.3
```

```
# Weekends rainfall
sum(rain_weekends$weekends_rainfall)
```

```
[1] 350.3
```

```
# Check rain totals by comparing to original data set rain variable
total_rainfall <- sum(db_tib$rain)
weekdays_rainfall <- sum(rain_weekdays$weekdays_rainfall)
weekends_rainfall <- sum(rain_weekends$weekends_rainfall)
round(total_rainfall, 2) == round(weekends_rainfall + weekdays_rainfall, 2)
```

```
[1] TRUE
```

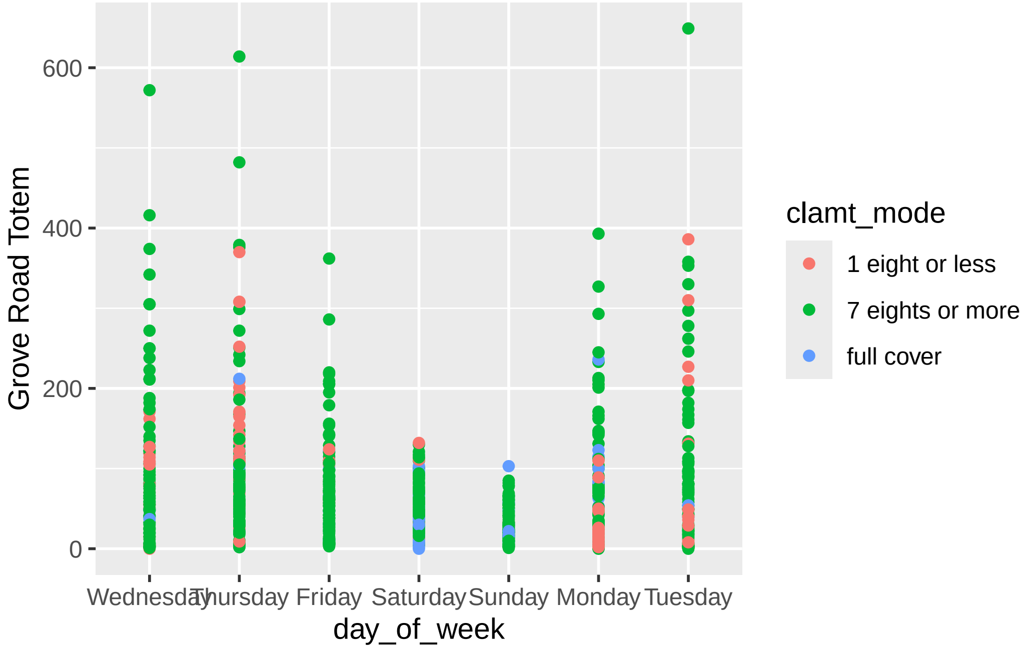
```
# Check there has been on average more rain during the weekend (Sat-Sun) with
# respect to weekdays (Mon-Fri)
mean(rain_weekends$weekends_mean_rainfall) > mean(rain_weekdays$weekdays_mean_rainfall)
```

```
[1] TRUE
```

4. Focus on the data for one month of the year of your choice, create a plot of the daily traffic volume in a locations of your choice, and the mode of the Cloud amount each day. Comment on your findings. Notice that there isn't a built-in function to calculate the mode in R. The mode is defined as the most frequently occurring value in the set of observations.

```
# Map clamt to mode label
db_dec <- db_tib %>%
  filter(month == "December") %>%
  mutate(clamt_mode = case_when(clamt == 0 ~ "absence",
                                clamt == 1 ~ "1 eight or less",
                                clamt %in% 2:7 ~ "7 eights or more",
                                clamt == 8 ~ "full cover",
                                clamt == 9 ~ "foggy", .default = "NA"))

dec_plot <- ggplot(db_dec, aes(y = `Grove Road Totem`, x = day_of_week, color = clamt_mode
dec_plot + geom_point()
```



Task 3: Creativity

- Do something interesting with these data! Create two plots or two tables or one plot and one table showing something we have not discovered above already and outline your findings.

```
# Get mean, median and standard deviation of 'mean hourly wind speed (kt)'
db_tib %>%
  summarise(mean_wind = mean(wdsp), median_wind = median(wdsp), sd_wind = sd(wdsp))
```

```
# A tibble: 1 x 3
  mean_wind median_wind sd_wind
      <dbl>       <dbl>   <dbl>
1      8.96           9    3.93
```

- Mean / Median around 9 and standard deviation is close to 4 overall.
- Lets look closer at this data per month beyond December.

```
# Extract year variable
db_tib$year <- as.numeric(format(db_tib$Date,'%Y'))

# Get mean, median and standard deviation of 'mean hourly wind speed (kt)' per year and mo
db_wind <- db_tib %>%
  group_by(year, month) %>%
  summarise(mean_wind = mean(wdsp), median_wind = median(wdsp), sd_wind = sd(wdsp))
```

```
`summarise()` has grouped output by 'year'. You can override using the
`.groups` argument.
```

```
# Sort by standard deviation
db_wind %>% arrange(desc(sd_wind))
```

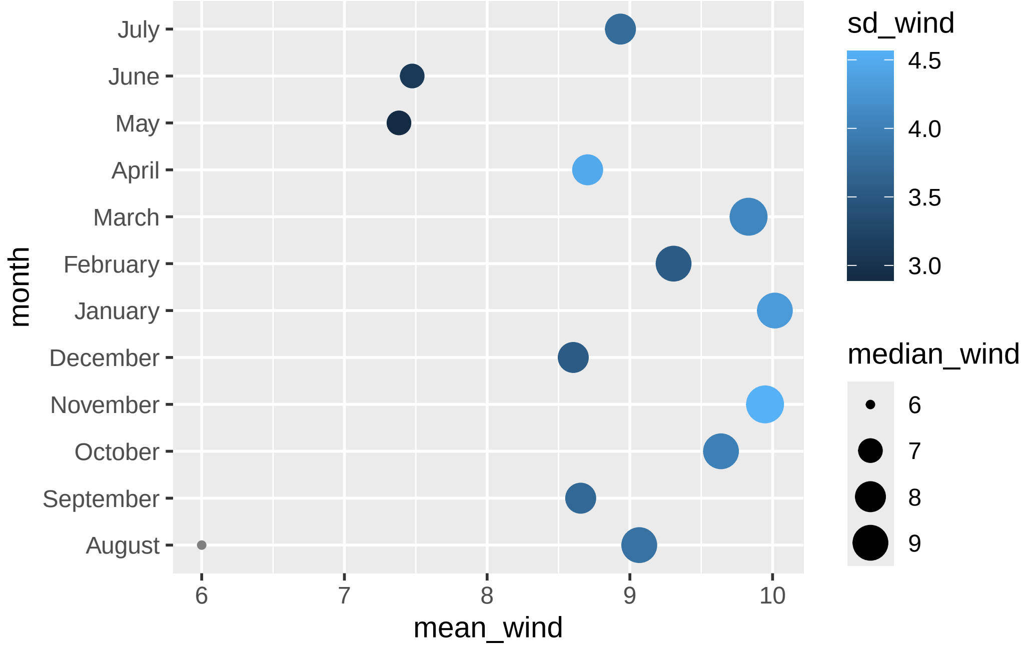
```
# A tibble: 13 x 5
# Groups:   year [2]
    year month     mean_wind median_wind sd_wind
   <dbl> <ord>         <dbl>       <dbl>   <dbl>
 1  2022 November       9.95         9.5    4.57
 2  2023 April          8.70         8      4.48
 3  2023 January       10.0          9      4.31
 4  2023 March          9.83         9.5    4.08
 5  2022 October        9.64         9      3.99
 6  2023 August         9.07         9      3.83
 7  2023 July           8.93         8      3.75
 8  2022 September      8.66         8      3.70
 9  2022 December       8.60         8      3.55
10  2023 February       9.31         9      3.55
11  2023 June           7.48         7      3.09
12  2023 May            7.38         7      2.89
13  2022 August         6            6     NA
```

- Notice that the greatest standard deviation in mean wind speed was in November 2022.

- However, January 2023 had the highest mean wind speed (only month to hit double digits for abs value).
- I think this is interesting to note that November 22 was more unpredictable but January was in fact windier on average.

```r
library(ggplot2)

# Creat plot for wind aggregattions
ggplot(db_wind, aes(x = month, y = mean_wind, color = sd_wind, size = median_wind)) +
  geom_point()  +
  coord_flip()
```


July
June
May
April
March
February
January
December
November
October
September
August
month
mean_wind
sd_wind
median_wind