Weather Patterns X COVID-19

Final Project Documentation

Uhuru Kamau, Shuyang Lu, Yifan Zhao, Chenhao Zhao, Zhaofeng Liu

Contents


```r
# Load Required Packages
library(tidyverse)
library(kableExtra)
library(readr)
library(gridExtra)
library(knitr)
```

Project Workflow

Right Side

```r
knitr::include_graphics(path = "images/QBS181_1.png")
```

Figure 1: QBS181 Final Project: WorkFlow (Part A)

Left Side

```r
knitr::include_graphics(path = "images/QBS181_2.png")
```

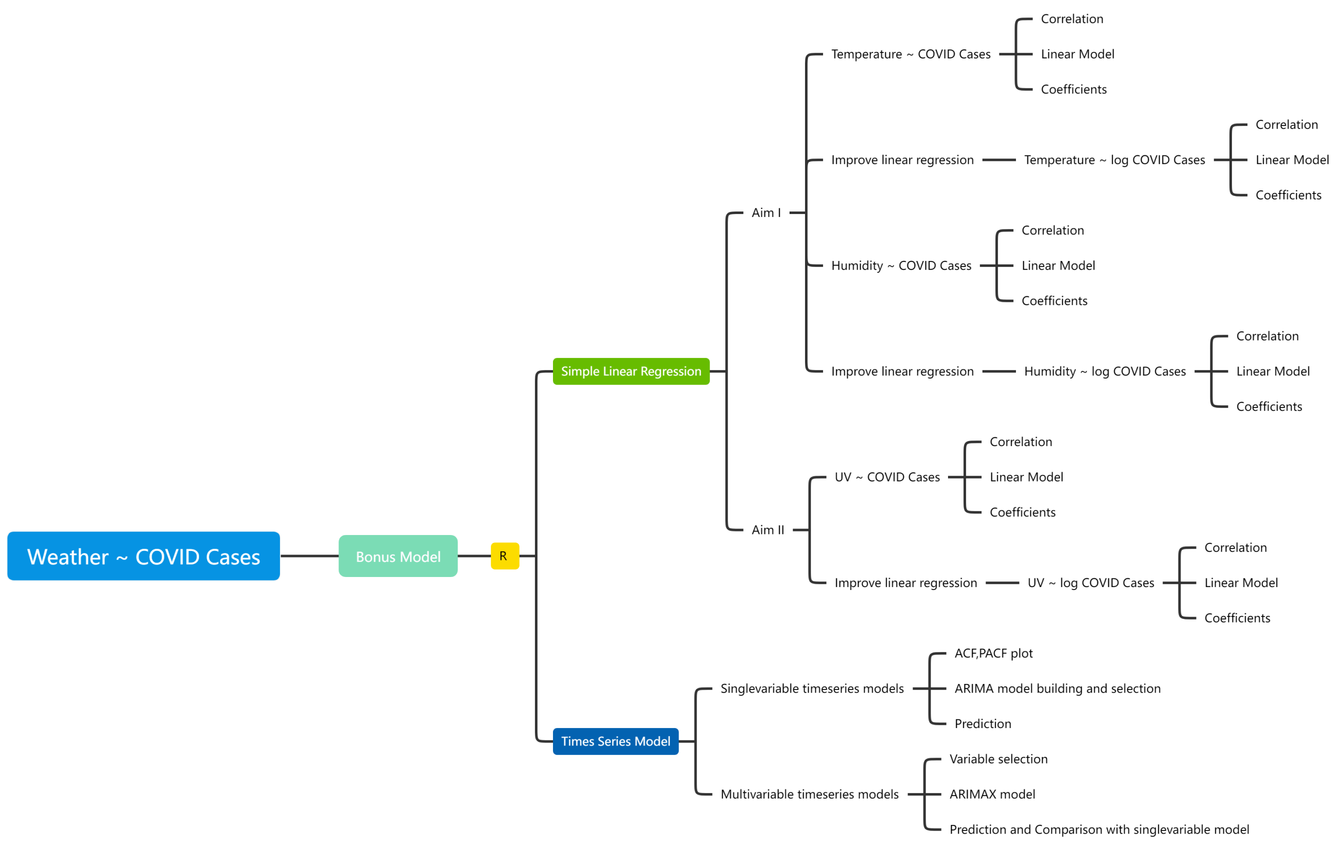


Figure 2: QBS181 Final Project: WorkFlow (Part B)

Data Acquisition

1. New York City COVID-19 Data Archive

- Source: NYC OpenData
- Acquisation Method
  - Download .csv file
- Purpose:
  - We will use this time series data to track changes in the incidence of COVID-19.

2. New York City Weather Data

- Source: Weather Underground - Weather Archive
- Acquisition Method
  - Webscraping/ API Tool
- Purpose:
  - Merge time series weather data with timeseries Covid-19 data and investigate potential associations

3. Daily UV Index Scores - New York City

- Source: Central New York's Live Weather Source
- Acquisition Method
  - UV index values are presented as tables (see figure)
  - Copy tables and paste into Microsoft Excel
  - Save as .csv file
- Purpose
  - Sunlight and Vitamin-D absorbtion
    * It is generally accepted that there is a positive association between exposure to sunlight and absorbtion of vitamin-D.
    * It is also generally accepted that there is a positive association between vitamin-D absorbtion and immune system capacity.
  - We will us UV-Index as a proxy for exposure to sunlight at the population level and test for associations between UV Index and the incidence of Covid-19.

Relational Schema

```r
knitr::include_graphics(path = "images/Relational_Schema.png")
```

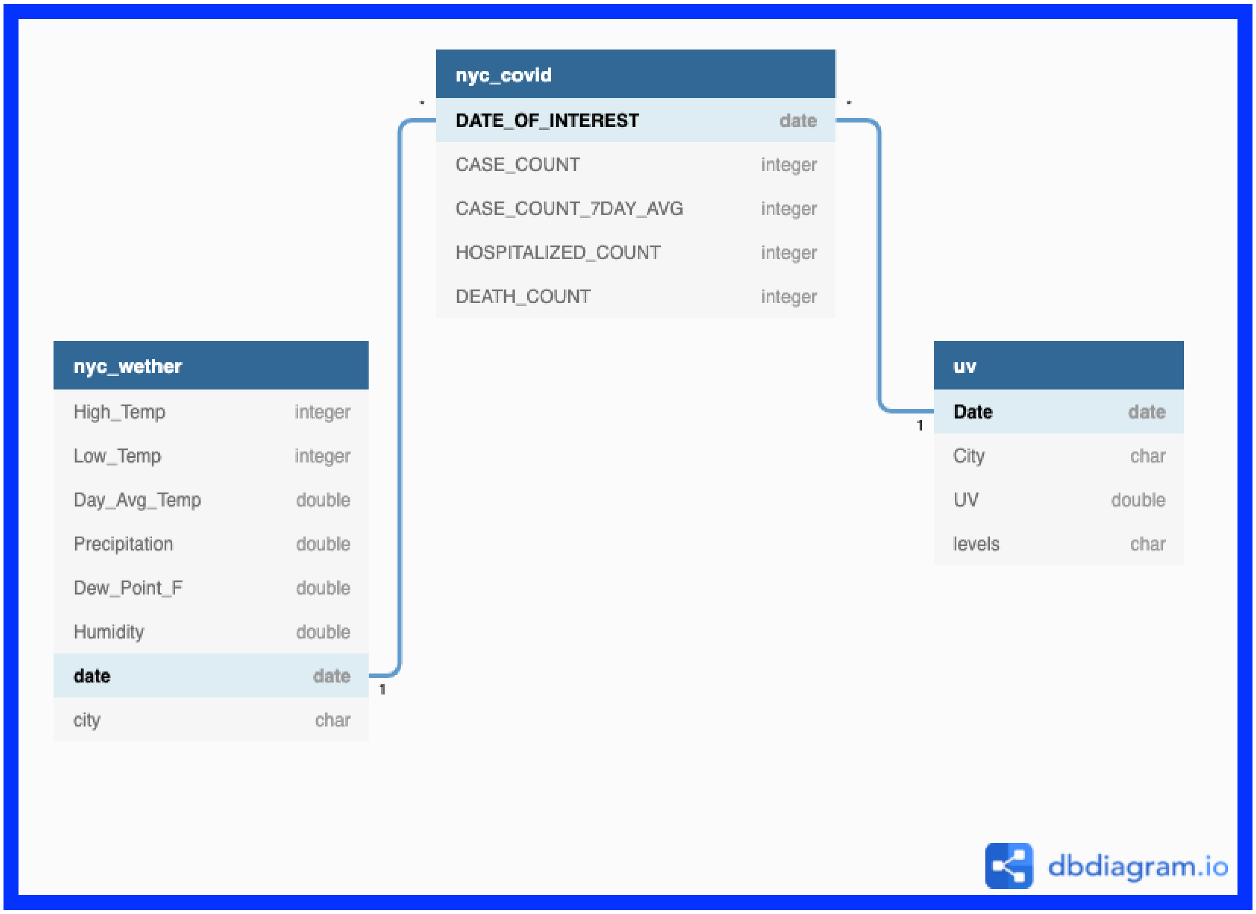


Figure 3: Highlighting the Keys to our Relational Database

Data Cleaning

1. New York City COVID-19 Data Archive

A. Read in the Covid-19 Date Frame

```
covid_df <- read.csv("data/Raw Data/nyc_covid19_data/NYC_Covid_Data_raw.csv")
```

2. New York City Weather Data

A. Read-in File from Raw Data File

- The raw file has an issue with the column headers.
  - Several Headers include symbols that don't work with the interpretor
    - eg. $Low\_Temp(^oF)$, $High\_Temp(^oF)$
- Solution: Update column names while reading in the file!

```
# vector with acceptable column names
headers <- c("High.Temp", "Low.Temp", "Avg.Temp", "Precip", "Dew.Point", "Humidity",
    "Date", "City")

# read in the raw data file
weather.raw <- read.csv(file = "data/Raw Data/nyc_weather_raw.csv", header = TRUE,
    col.names = headers)
```

B. Format all observations of the "Date" Variable

- Variable is of class "character" by default

```
class(weather.raw$Date)
```

```
RESULT [1] "character"
```

- Reclassify the variable as a "Date"

```
weather.clean <- weather.raw %>%
    mutate(Date = as.Date(Date, "%m/%d/%Y"))
```

- Outcome:

```
class(weather.clean$Date)
```

```
RESULT [1] "Date"
```

C. Missing Data

- Since we intend to do a time series, we need to identify any missing dates in the "date" column.
  - We will do this using a CUSTOM FUNCTION!

```r
# Custom function to find the missing date in the date column
find.missing.dates <- function(d) {
    date_range <- seq(min(d), max(d), by = 1)
    date_range[!date_range %in% d]
}
```

- Use the custom function to identify missing dates in our NYC Weather df

```r
# Display the missing dates
date.missing = c()
date.missing <- find.missing.dates(weather.clean$Date)
print(date.missing)
```

```
RESULT [1] "2020-11-08"
```

D. Replace Missing Values

- Method: Fill the missing data by averaging the former 6 days' data

```r
# Find the index of the day before '2020-11-08'
weather.clean$Date <- as.character(weather.clean$Date)
id.missing.date = which(weather.clean$Date == "2020-11-07") + 1
```

- Build a custom function to fill the missing data
  - Approach: use the average of the previous six days

```r
# Custom function to fill the missing data by averaging the former 6 days' data
fill.missing.values <- function(df, newrow.id) {
    newrow <- list()
    value <- c()
    first.row = newrow.id - 6
    last.row = newrow.id - 1
    col.num = ncol(df) - 2
    for (i in 1:col.num) {
        subs <- weather.clean[first.row:last.row, i]  # Create a new subset for each column
        value <- mean(subs)  # Calculate the mean
        newrow <- append(newrow, value)
    }
    return(newrow)
}
```

- Use the custom function to fill in the values of the missing row

```r
# Fill the missing values in the missing row
missing.row <- fill.missing.values(weather.clean, id.missing.date)
missing.row <- append(missing.row, "2020-11-08")
missing.row <- append(missing.row, "new york city")
```

- Build another custom function to insert the row into the df

```r
# Custom function to insert the new row
insertRow <- function(existingDF, newrow, r) {
    existingDF[seq(r + 1, nrow(existingDF) + 1), ] <- existingDF[seq(r, nrow(existingDF)),
        ]
    existingDF[r, ] <- newrow
    existingDF
}
```

- Insert the imputed value into the df!

```r
# Insert the missing row and store it into a new df
weather.clean <- insertRow(weather.clean, missing.row, id.missing.date)
```

E. Remove the "City" Variable

- Every observation is is "new york city"
  - This variable is effectively just clutter.

```r
weather.clean = weather.raw %>%
    select(-City)
```

F. Display

```r
kable(x = weather.clean[1:5, ], digits = 2, align = "c")
```

High.Temp	Low.Temp	Avg.Temp	Precip	Dew.Point	Humidity	Date
44	26	35.46	0.00	13.67	41.83	3/1/2020
56	38	48.17	0.00	30.46	51.12	3/2/2020
58	48	52.41	0.01	44.59	75.47	3/3/2020
57	46	50.52	0.28	28.52	44.76	3/4/2020
52	40	44.75	0.00	25.38	48.50	3/5/2020

G. Write the Processed Data to a new .csv file

```r
write.csv(x = weather.clean, file = "data/Processed Data/nyc_weather.csv")
```

H. In Excel - Add a "Season" variable to the .csv generated in the prior section

- Open File in Microsoft Excel
  - Summarise the process
- Save the file as "nyc_clean_weather_add_season.csv"
  - Push to Github

I. Read in "nyc_clean_weather_add_season.csv" and reformat the "date" variable

- Read in the file

```r
# vector with acceptable column names
headers <- c("High.Temp", "Low.Temp", "Avg.Temp", "Precip", "Dew.Point", "Humidity",
    "date", "City", "Month", "season")
add_season_nyc_weather <- read.csv(file = "data/nyc_clean_weather_add_season.csv",
    header = T, col.names = headers) %>%
    select(-City)
```

- Reformat the "date" variable

```r
# change date type from ymd to mdy
add_season_nyc_weather$date <- format(as.Date(add_season_nyc_weather$date, "%Y/%m/%d"),
    "%m/%d/%Y")
```

J. Display

```r
knitr::kable(x = add_season_nyc_weather[1:5, ], align = "c")
```

High.Temp	Low.Temp	Avg.Temp	Precip	Dew.Point	Humidity	date	Month	season
44	26	35.46	0.00	13.67	41.83333	03/01/2020	3	spring
56	38	48.17	0.00	30.46	51.12500	03/02/2020	3	spring
58	48	52.41	0.01	44.59	75.47059	03/03/2020	3	spring
57	46	50.52	0.28	28.52	44.76000	03/04/2020	3	spring
52	40	44.75	0.00	25.38	48.50000	03/05/2020	3	spring

3. Daily UV Index Scores - New York City

A. Read in the csv file that we built in excel

```r
nyc.uv = read.csv(file = "data/Raw Data/nyc_uv_raw.csv", header = T)
```

B. Remove the "City" Variable

- Every observation is "New York City" so this variable is effectively clutter.

```r
nyc.uv <- nyc.uv %>%
    select(-City)
```

C. Display

```r
knitr::kable(x = nyc.uv[1:5, ], align = "c")
```

Date	UV
3/1/2020	0.8
3/2/2020	1.3
3/3/2020	1.3
3/4/2020	0.8
3/5/2020	2.1

Data Wrangling and Objectives

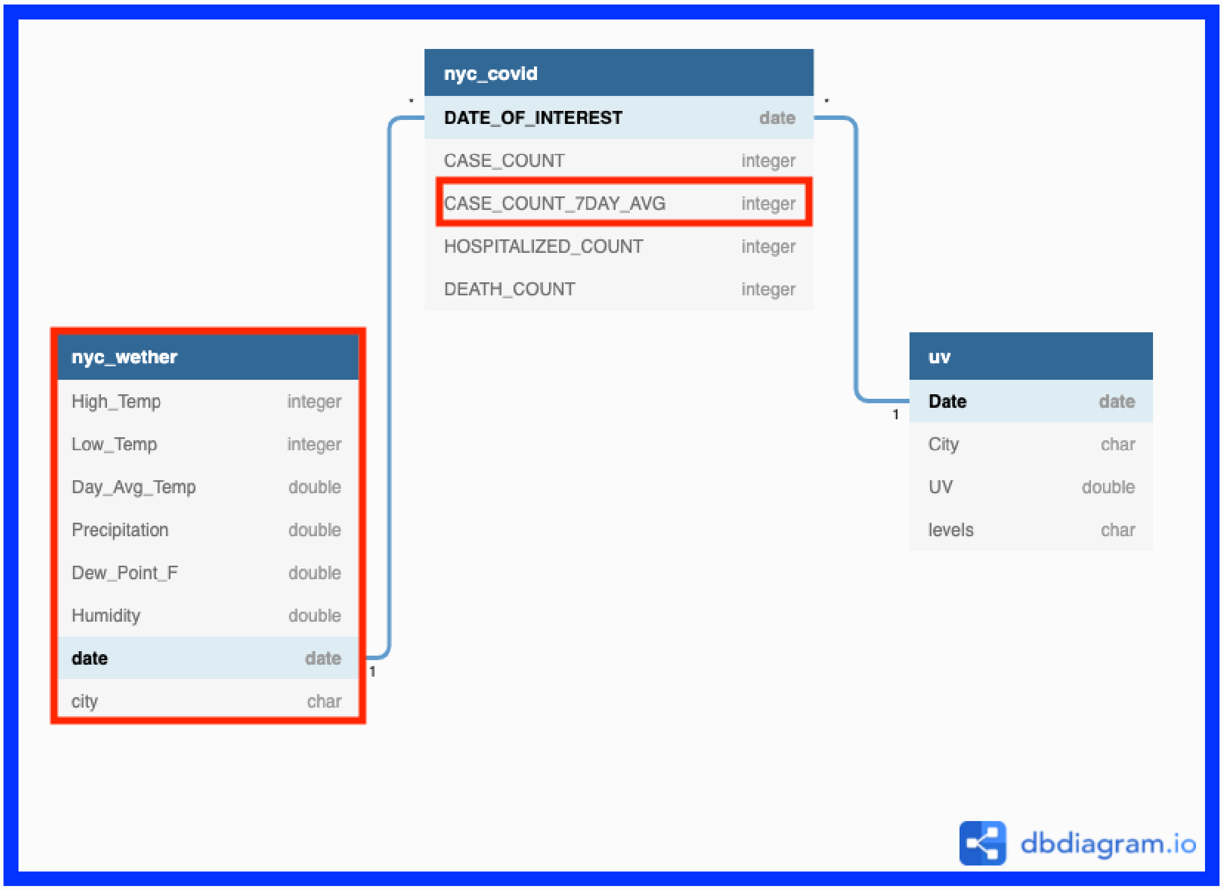
1. Is there a Difference in Number of Cases Observed in the Summer vs in the Winter?

A. Add the "CASE_COUNT_7DAY_AVG" variable from the covid df to the add_season_nyc_weather df

- Visual Aid

```r
knitr::include_graphics(path = "images/weather_and_7day_avg.png")
```

\begin{figure}

{

}

\caption{CASE_COUNT_7DAY_AVG variable from the covid df to the add_season_nyc_weather df}
\end{figure}

- Combine

```r
add_season_test <- add_season_nyc_weather
rownames(covid_df) <- covid_df$DATE_OF_INTEREST
add_season_test$Case_Count_7Day_Avg <- covid_df[, "CASE_COUNT_7DAY_AVG"]
```

B. Create a df with ONLY the summer and winter observations

```r
sum_win <- add_season_test %>%
    filter(season == "summer" | season == "winter")
```

C. Run a Wilcox Test to test the difference in incidence (summer vs winter)

```r
wilcox.test(sum_win[which(sum_win$season == "summer"), ]$Case_Count_7Day_Avg, sum_win[which(sum_win$sea
    "winter"), ]$Case_Count_7Day_Avg)
```

```
RESULT
RESULT  Wilcoxon rank sum test with continuity correction
RESULT
RESULT data:  sum_win[which(sum_win$season == "summer"), ]$Case_Count_7Day_Avg and sum_win[which(sum_wi
RESULT W = 0, p-value < 2.2e-16
RESULT alternative hypothesis: true location shift is not equal to 0
```

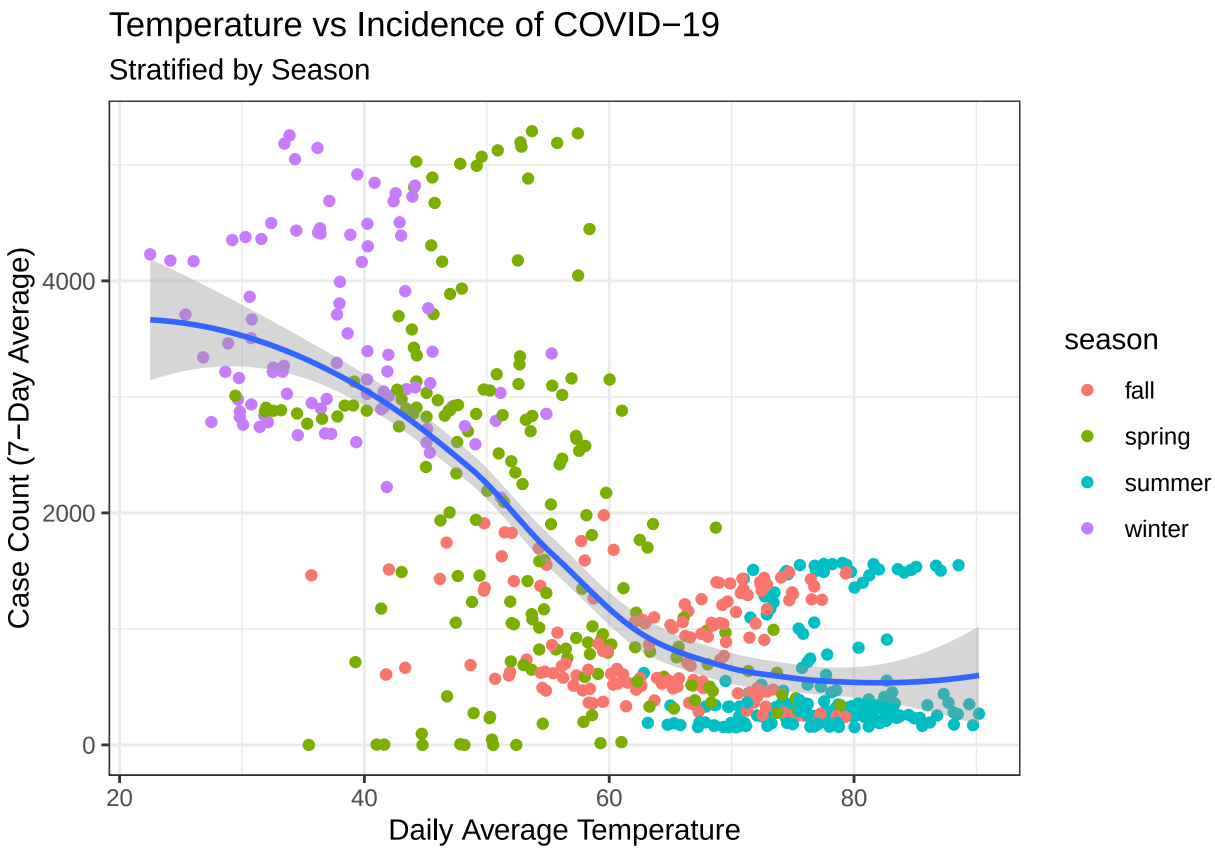
- We can reject the null hypothesis that there is no difference in incidence.

Is there an Association between temperature and Incidence?

- Scatter Plot

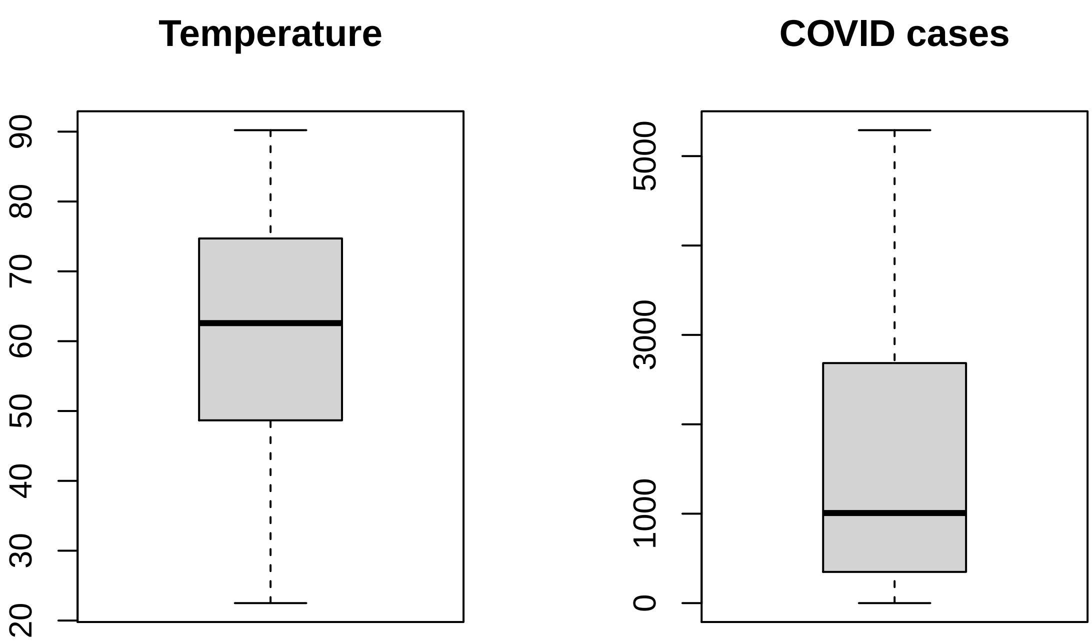
```r
ggplot(data = add_season_test) + geom_point(mapping = aes(x = Avg.Temp, y = Case_Count_7Day_Avg,
    color = season)) + labs(x = "Daily Average Temperature", y = "Case Count (7-Day Average)",
    title = "Temperature vs Incidence of COVID-19", subtitle = "Stratified by Season") +
    theme_bw() + geom_smooth(mapping = aes(x = Avg.Temp, y = Case_Count_7Day_Avg))
```

```
RESULT `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- Boxplot

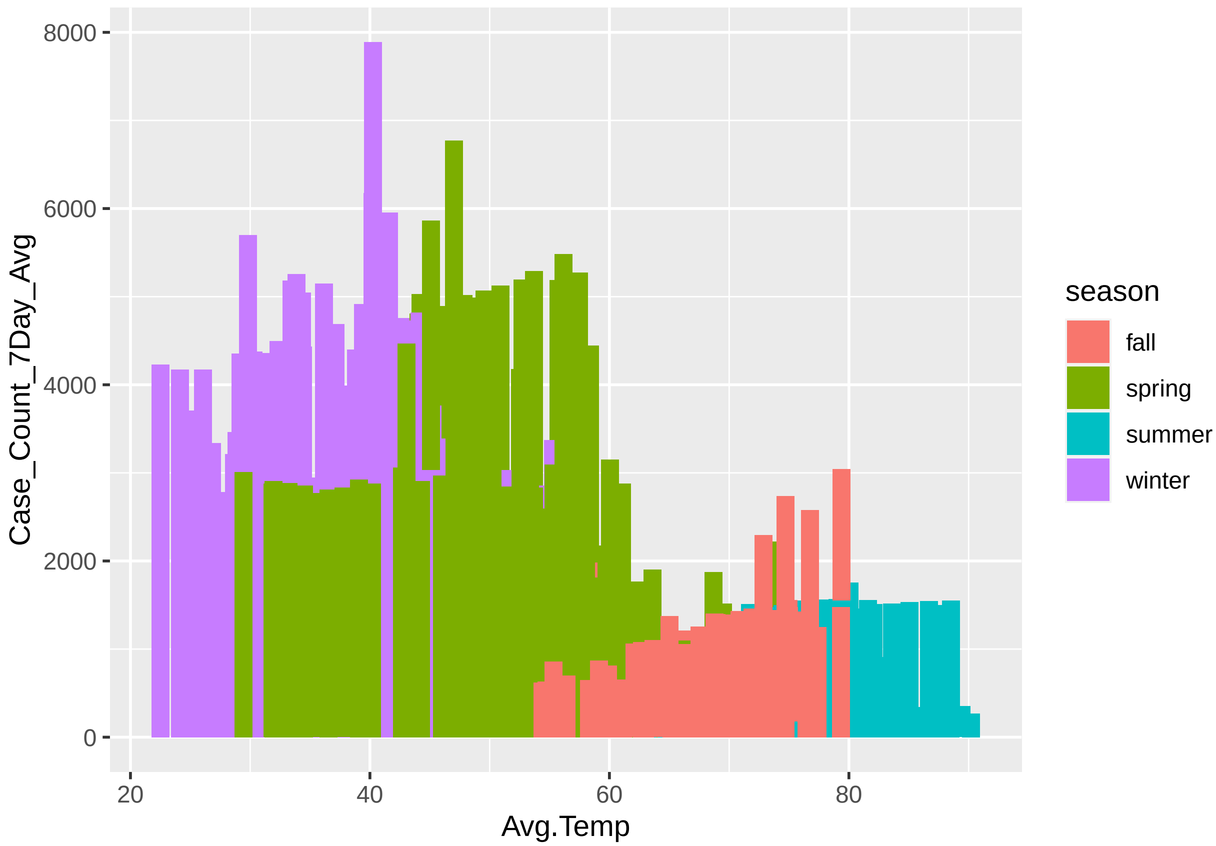
```
par(mfrow = c(1, 2))  # divide graph area in 2 columns
boxplot(add_season_test$Avg.Temp, main = "Temperature")
boxplot(add_season_test$Case_Count_7Day_Avg, main = "COVID cases")
```



- Bar Graph

```
ggplot(data = add_season_test) + geom_col(mapping = aes(x = Avg.Temp, y = Case_Count_7Day_Avg,
    fill = season), width = 1.5)
```

```
RESULT Warning: position_stack requires non-overlapping x intervals
```



- Correlation between Temperature and Case Count

```
cor(add_season_test$Avg.Temp, add_season_test$Case_Count_7Day_Avg)
```

```
RESULT [1] -0.7175508
```

- Single Variable Linear Regression Model Temperature ~ Cases

```
linearMod <- lm(add_season_test$Avg.Temp ~ add_season_test$Case_Count_7Day_Avg, data = add_season_test)
print(linearMod)
```

```
RESULT
RESULT Call:
RESULT lm(formula = add_season_test$Avg.Temp ~ add_season_test$Case_Count_7Day_Avg,
RESULT     data = add_season_test)
RESULT
RESULT Coefficients:
RESULT                          (Intercept)  add_season_test$Case_Count_7Day_Avg
RESULT                            73.416364                            -0.008214
```

- View Linear Model Coefficients

```
summary(linearMod)$coefficients
```

```
RESULT                                          Estimate    Std. Error    t value
RESULT (Intercept)                           73.416363791 0.6608480427 111.09417
RESULT add_season_test$Case_Count_7Day_Avg -0.008214267 0.0003233623 -25.40268
RESULT                                          Pr(>|t|)
RESULT (Intercept)                           0.000000e+00
RESULT add_season_test$Case_Count_7Day_Avg 1.417949e-97
```

- Improve Linear Model By using the Log incident case values

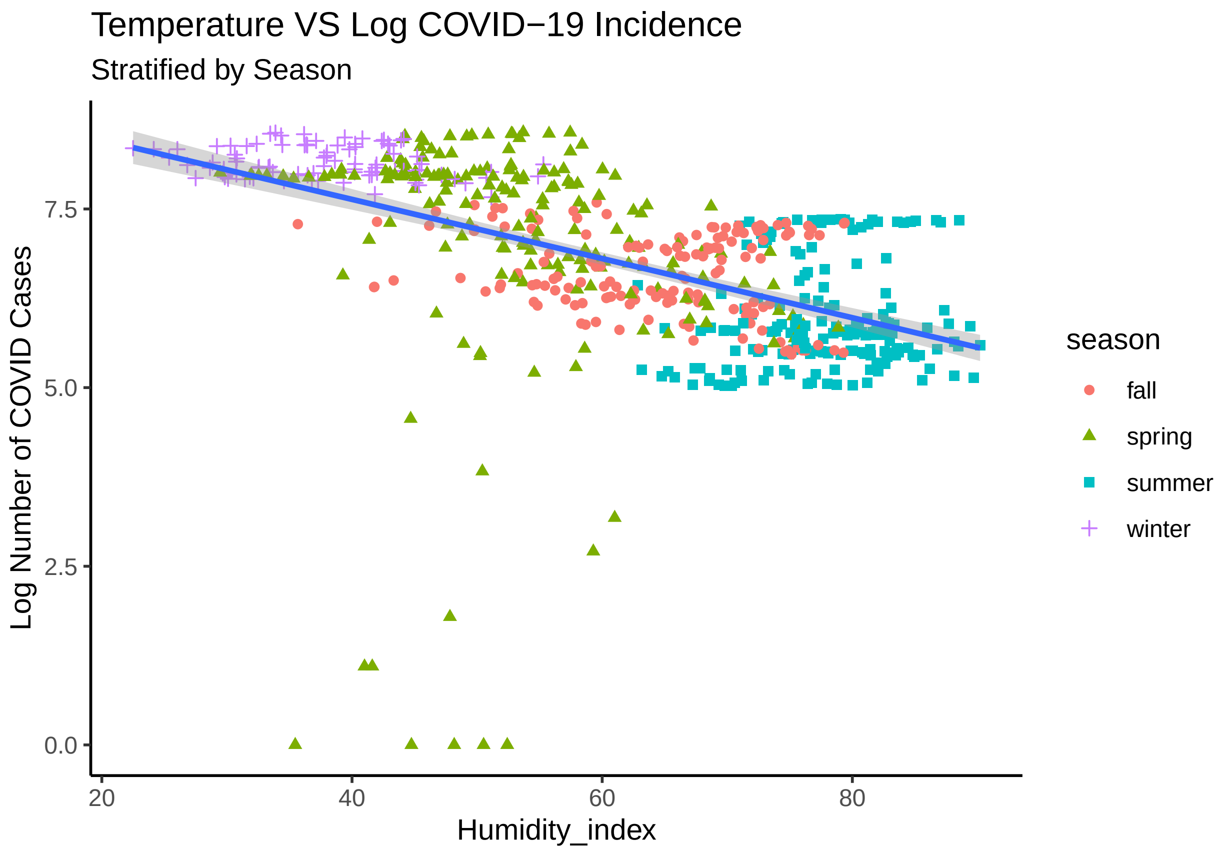
```
# log cases
add_season_test$log_cases <- log(add_season_test$Case_Count_7Day_Avg)
for (i in 1:nrow(add_season_test)) {
    if (add_season_test$Case_Count_7Day_Avg[i] == 0) {
        add_season_test$log_cases[i] = 0
    }
}
```

- Scatter plot of temp~ new log cases

```
## test log cases
ggplot(data = add_season_test) + geom_point(mapping = aes(x = Avg.Temp, y = log_cases,
    color = season, shape = season)) + labs(x = "Humidity_index", y = "Log Number of COVID Cases",
    title = "Temperature VS Log COVID-19 Incidence", subtitle = "Stratified by Season") +
    theme_classic() + geom_smooth(mapping = aes(x = Avg.Temp, y = log_cases), method = "lm",
    inherits.aes = F)
```

```
RESULT Warning: Ignoring unknown parameters: inherits.aes
```

```
RESULT ‘geom_smooth()‘ using formula ’y ~ x’
```



Temperature VS Log COVID−19 Incidence
Stratified by Season
Log Number of COVID Cases
7.5
5.0
2.5
0.0
20
40
60
80
Humidity_index
season
fall
spring
summer
winter