

## Table of Contents

1.0	Introduction.....	2
2.0	Content.....	3
2.1	Data Import and Pre-processing.....	3
2.2	Question 1: How to predict raining day? .....	6
2.2.1	Analysis 1: Find the rain distribution by month. ....	7
2.2.2	Analysis 2: Find the total rainfall by month. ....	10
2.2.3	Analysis 3: Compare the status of 9am cloud during raining day with non-raining day.....	12
2.2.4	Analysis 4: Find the frequency of raining according to temperature at 3pm.....	15
2.2.5	Analysis 5: Find the relationship between humidity and RainToday.....	18
2.2.6	Analysis 6: Find the relationship between humidity at 9am and evaporation based on RainToday.....	21
2.2.7	Analysis 7: Find the relationship between sunshine and evaporation. ....	23
2.2.8	Analysis 8: Find the relationship between sunshine and cloud at 3pm. ....	26
2.2.9	Analysis 9: Status of the amount of rainfall for the whole year. ....	28
2.2.10	Analysis 10: Mean of weather data group by months. ....	30
2.3	Question 2: When is the best time for outdoor activities? .....	32
2.3.1	Analysis 11: Find the daily temperature average using Minimum Temperature and Maximum Temperature. ....	33
2.3.2	Analysis 12: Compare the number of hot days and cool days based on Temperature at 3pm.....	35
2.3.3	Analysis 13: Find the median and mean of temperature at 9am according to RainToday categorized by cloud at 9am.....	37
2.3.4	Analysis 14: Find the relationship among Temp9am, Temp3pm, average_temp ..	39
2.4	Question 3: How to avoid storm in the U.S.? .....	42
2.4.1	Analysis 15: Find the status of wind for the whole year.....	43
2.4.2	Analysis 16: Find the relationship of wind direction and wind speed at 3pm.....	48

2.5 Others.....	50
2.5.1 Analysis 17: Correlation relationship among the weather data. ....	50
2.5.2 Analysis 18: Find the density of each weather data by entering the choice. ....	52
2.5.3 Analysis 19: Write new CSV files into new folder.....	55
3.0 Extra Feature .....	56
3.1 Bar Diagram with Two Bars .....	56
3.2 Melt Function.....	58
3.3 Donut Chart.....	59
3.4 3D Chart with RGL.....	64
3.5 Wind Rose.....	66
3.6 Violin Chart .....	68
3.7 Correlation among Weather Data .....	71
3.8 Geom_hline and geom_vline .....	72
4.0 Conclusion .....	74
5.0 Reference .....	75

## 1.0 Introduction

The dataset of daily weather data is provided by U.S. Weather Database. It contains 22 columns and 366 rows that cover the various aspects, including temperature, air pressure, humidity, evaporation, wind speed, and others. The purpose of this assignment is to create a meaningful analysis for representing the overall weather trend in the U.S. that helps in decision making.

The techniques that used in the data analysis include Data Exploration, Data Manipulation, Data Visualization, and Data Transformation. Data Exploration is the critical step for studying the data in a broader view of essential information, while data visualization transforms the information from texture to picture. Data visualization focuses on generating crucial diagrams and plots that help understand the complex relationship among data (U.C. Business Analytics R Programming Guide, 2020).

Data manipulation, including subsetting, sorting, reordering, and others, will make the untidy data easier to work with (DataFlair, 2019). After manipulated the original data, it will be able to do further analysis and plot a new diagram based on the updated data. RStudio is the powerful IDE used to complete the R language coding for the data analysis in this assignment. It makes data visualization more accessible and the view of created objects more convenient (Kim Love, 2019).

## 2.0 Content

### 2.1 Data Import and Pre-processing

#### Load Library

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(openair)
library(plotrix)
library(reshape2)
library(rgl)
library(corrplot)
library(hrbrthemes)
```

Figure 1: Load Libraries

#### Import Data

```
weather_data = read.csv( "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\weather.csv",
                        header = TRUE, stringsAsFactors = TRUE)
```

Figure 2: Import Dataset from csv file

The import data is saved as a data frame called “weather\_data”. When importing the data, header is also imported (header = TRUE), and all the character values are changed to factor values (stringsAsFactors = TRUE).

## Data Pre-processing

```
# Create a new data frame (Date) and combine with Weather Data
Date = seq(as.Date("2020-01-01"), length = 366, by = "days")
weather_data = cbind(Date, weather_data)

# Mutate (add "Month" column)
weather_data = mutate(weather_data, Month = format(Date, "%m"))
weather_data = weather_data %>% relocate(Month, .after = Date)

# View the data in table format
View(weather_data)

# Check data type
str(weather_data)

# Correct data type
weather_data$Month <- factor(weather_data$Month)

# Summary for weather data
summary(weather_data)
```

Figure 3: Code for Data Pre-processing

Assumption of this dataset is U.S. weather data in the year 2020 from 1<sup>st</sup> January to 31<sup>st</sup> December. Before the data analysis, date data (“Date”) has been created in a new data frame and combined with the original weather data frame. Month data (“Month”) also added into the original weather data frame using “mutate” technique.

```
'data.frame': 366 obs. of 24 variables:
 $ Date      : Date, format: "2020-01-01" "2020-01-02" "2020-01-03" "2020-01-04" ...
 $ Month     : chr "01" "01" "01" "01" ...
 $ MinTemp   : num 8 14 13.7 13.3 7.6 6.2 6.1 8.3 8.8 8.4 ...
 $ MaxTemp   : num 24.3 26.9 23.4 15.5 16.1 16.9 18.2 17 19.5 22.8 ...
 $ Rainfall  : num 0 3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 ...
 $ Evaporation : num 3.4 4.4 5.8 7.2 5.6 5.8 4.2 5.6 4 5.4 ...
 $ Sunshine  : num 6.3 9.7 3.3 9.1 10.6 8.2 8.4 4.6 4.1 7.7 ...
 $ WindGustDir : Factor w/ 16 levels "E","ENE","ESE",...: 8 2 8 8 11 10 10 1 9 1 ...
 $ WindGustSpeed: int 30 39 85 54 50 44 43 41 48 31 ...
 $ WindDir9am : Factor w/ 16 levels "E","ENE","ESE",...: 13 1 4 15 11 10 10 10 1 9 ...
 $ WindDir3pm  : Factor w/ 16 levels "E","ENE","ESE",...: 8 14 6 14 3 1 3 1 2 3 ...
 $ WindSpeed9am : int 6 4 6 30 20 20 19 11 19 7 ...
 $ WindSpeed3pm : int 20 17 6 24 28 24 26 24 17 6 ...
 $ Humidity9am : int 68 80 82 62 68 70 63 65 70 82 ...
 $ Humidity3pm : int 29 36 69 56 49 57 47 57 48 32 ...
 $ Pressure9am : num 1020 1012 1010 1006 1018 ...
 $ Pressure3pm : num 1015 1008 1007 1007 1018 ...
 $ Cloud9am    : int 7 5 8 2 7 7 4 6 7 7 ...
 $ Cloud3pm    : int 7 3 7 7 7 5 6 7 7 1 ...
 $ Temp9am     : num 14.4 17.5 15.4 13.5 11.1 10.9 12.4 12.1 14.1 13.3 ...
 $ Temp3pm     : num 23.6 25.7 20.2 14.1 15.4 14.8 17.3 15.5 18.9 21.7 ...
 $ RainToday   : Factor w/ 2 levels "No","Yes": 1 2 2 2 2 1 1 1 1 2 ...
 $ RISK_MM     : num 3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 0 ...
 $ RainTomorrow : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 1 1 2 1 ...
```

Figure 4: Data Type before Data Correction

It is a must to check the data type (str (weather\_data)) of all the values in the data frame to ensure all the data are stored with numerical or categorical variables. If there is incorrect data type such as “Month”, it is needed to change the values to factors for storing as categorical variables.

Date	Month	MinTemp	MaxTemp	Rainfall	Evaporation	
Min. :2020-01-01	01 : 31	Min. : -5.300	Min. : 7.60	Min. : 0.000	Min. : 0.200	
1st Qu.:2020-04-01	03 : 31	1st Qu.: 2.300	1st Qu.:15.03	1st Qu.: 0.000	1st Qu.: 2.200	
Median :2020-07-01	05 : 31	Median : 7.450	Median :19.65	Median : 0.000	Median : 4.200	
Mean :2020-07-01	07 : 31	Mean : 7.266	Mean :20.55	Mean : 1.428	Mean : 4.522	
3rd Qu.:2020-09-30	08 : 31	3rd Qu.:12.500	3rd Qu.:25.50	3rd Qu.: 0.200	3rd Qu.: 6.400	
Max. :2020-12-31	10 : 31	Max. :20.900	Max. :35.80	Max. :39.800	Max. :13.800	
(Other):180						
Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm
Min. : 0.000	NW : 73	Min. :13.00	SE : 47	NW : 61	Min. : 0.000	Min. : 0.00
1st Qu.: 5.950	NNW : 44	1st Qu.:31.00	SSE : 40	WNW : 61	1st Qu.: 6.000	1st Qu.:11.00
Median : 8.600	E : 37	Median :39.00	NNW : 36	NNW : 47	Median : 7.000	Median :17.00
Mean : 7.909	WNW : 35	Mean :39.84	N : 31	N : 30	Mean : 9.652	Mean :17.99
3rd Qu.:10.500	ENE : 30	3rd Qu.:46.00	NW : 30	ESE : 27	3rd Qu.:13.000	3rd Qu.:24.00
Max. :13.600	(Other):144	Max. :98.00	(Other):151	(Other):139	Max. :41.000	Max. :52.00
NA's :3	NA's : 3	NA's :2	NA's : 31	NA's : 1	NA's :7	
Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	
Min. :36.00	Min. :13.00	Min. : 996.5	Min. : 996.8	Min. :0.000	Min. :0.000	
1st Qu.:64.00	1st Qu.:32.25	1st Qu.:1015.4	1st Qu.:1012.8	1st Qu.:1.000	1st Qu.:1.000	
Median :72.00	Median :43.00	Median :1020.1	Median :1017.4	Median :3.500	Median :4.000	
Mean :72.04	Mean :44.52	Mean :1019.7	Mean :1016.8	Mean :3.891	Mean :4.025	
3rd Qu.:81.00	3rd Qu.:55.00	3rd Qu.:1024.5	3rd Qu.:1021.5	3rd Qu.:7.000	3rd Qu.:7.000	
Max. :99.00	Max. :96.00	Max. :1035.7	Max. :1033.2	Max. :8.000	Max. :8.000	
Temp9am	Temp3pm	RainToday	RISK_MM	RainTomorrow		
Min. : 0.100	Min. : 5.10	No :300	Min. : 0.000	No :300		
1st Qu.: 7.625	1st Qu.:14.15	Yes: 66	1st Qu.: 0.000	Yes: 66		
Median :12.550	Median :18.55		Median : 0.000			
Mean :12.358	Mean :19.23		Mean : 1.428			
3rd Qu.:17.000	3rd Qu.:24.00		3rd Qu.: 0.200			
Max. :24.700	Max. :34.50		Max. :39.800			

Figure 5: Data Summary after Data Correction

It is essential to know the summary of the dataset (summary (weather\_data)) as it provides the sum of each categorical variables and the summarised information of each continuous variables.

## 2.2 Question 1: How to predict raining day?

It is a common question in our daily life. Weather bureau makes daily reports on the weather conditions and predict the weather for the upcoming days. Human always concerns about raining days as sometimes it will be a resistance for their project progresses. Hence, to prevent any emergency, it becomes a challenge to overcome them with accurate weather forecast.

### Data Exploration

Code:

```
raining_day <- filter(weather_data,RainToday == "Yes")
nrow(raining_day)

raining_day [(raining_day$RainToday == "Yes") & (raining_day$RainTomorrow == "Yes"),1]
```

Figure 6: Code of Question 1 Data Exploration

Output:

```
> raining_day <- filter(weather_data,RainToday == "Yes")
> nrow(raining_day)
[1] 66
> raining_day [(raining_day$RainToday == "Yes") & (raining_day$RainTomorrow == "Yes"),1]
[1] "2020-01-02" "2020-01-03" "2020-01-04" "2020-01-31" "2020-02-15" "2020-02-19" "2020-02-20" "2020-02-21"
[9] "2020-03-20" "2020-04-01" "2020-04-05" "2020-04-13" "2020-04-22" "2020-05-25" "2020-06-13" "2020-07-26"
[17] "2020-08-10" "2020-10-31" "2020-11-14" "2020-11-15" "2020-12-04"
```

Figure 7: Output of Question 1 Data Exploration

From the output above, it shows the raining days in the year 2020 are 66 days. It also means the non-raining days are 300 days as there are 366 columns in this dataset. Besides, there are total 21 days which rain today and also rain tomorrow, and the longest rally is 4 days, which is from 2<sup>nd</sup> January to 5<sup>th</sup> January.

### 2.2.1 Analysis 1: Find the rain distribution by month.

This analysis is used to find the rain distribution according to month. It will show a pie chart with different colour to represent each month. The main purpose of this analysis is to show the months with the most raining days and the least raining days.

Code:

```
Yes_No = data.frame(weather_data %>% group_by(Month) %>% count(RainToday) %>% spread(RainToday,n))
Yes_No['No'] = NULL
names(Yes_No)[2] <- "Num_RainingDay"

Total_Yes = 0
for (i in Yes_No$Num_RainingDay){
  Total_Yes = Total_Yes + i
}
Yes_No=cbind(Yes_No, Percentage = c(round ((Yes_No$Num_RainingDay / Total_Yes) * 100, digits = 2)))

pie(Yes_No$Percentage, paste0(Yes_No$Percentage, "%"), radius = 1,
    main = "Rain Distribution(%)",
    col = c("#3e8f4f", "#649a4f", "#86a453", "#a6ad5b", "#c4b667", "#e1bf76",
            "#fcc889", "#fdb680", "#fca47b", "#f8917a", "#f37f7c", "#ea6e80"),
    clockwise = TRUE)
legend("right", legend = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                            "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),
    fill = c("#3e8f4f", "#649a4f", "#86a453", "#a6ad5b", "#c4b667", "#e1bf76",
            "#fcc889", "#fdb680", "#fca47b", "#f8917a", "#f37f7c", "#ea6e80"))
```

Figure 8: Code of Analysis 1

Number of raining days has been calculated (count (RainToday)) according to each month (group\_by (Month)). After that, the data stored in another data frame called “Yes\_No”, and arranged as months (spread (RainToday, n)). All the mentioned commands are combined and executed in one line code using piping technique (%>%). There are three columns in Yes\_No data frame, which are shown as the following figure:

	Month	No	Yes
1	01	23	8
2	02	18	11
3	03	27	4
4	04	20	10
5	05	28	3
6	06	26	4
7	07	28	3
8	08	26	5
9	09	25	5
10	10	27	4
11	11	25	5
12	12	27	4

Figure 9: Yes\_No Data Frame before Correcting Data.

From the above figure, the total of raining days and non-raining days for each month has been shown. To plot pie chart for the raining days only, the “No” column is deleted (Yes\_No['No'] = NULL) and the “Yes” column is renamed as “Num\_RainingDay” as it will



be easier to recognize the data. Percentage of rain distribution for each month is calculated using for loop and stored as “Percentage” column. The final variables that stored in “Yes\_No” data frame is shown as the diagram below:

	Month	Num_RainingDay	Percentage
1	01	8	12.12
2	02	11	16.67
3	03	4	6.06
4	04	10	15.15
5	05	3	4.55
6	06	4	6.06
7	07	3	4.55
8	08	5	7.58
9	09	5	7.58
10	10	4	6.06
11	11	5	7.58
12	12	4	6.06

Figure 10: Yes\_No Data Frame after Correcting Data.

The radius of the plotted pie chart is 1. The pie chart is also given a title “Rain Distribution (%)” and different colour for each month. All the month names are also shown as an explanatory note using legend function.

Output:

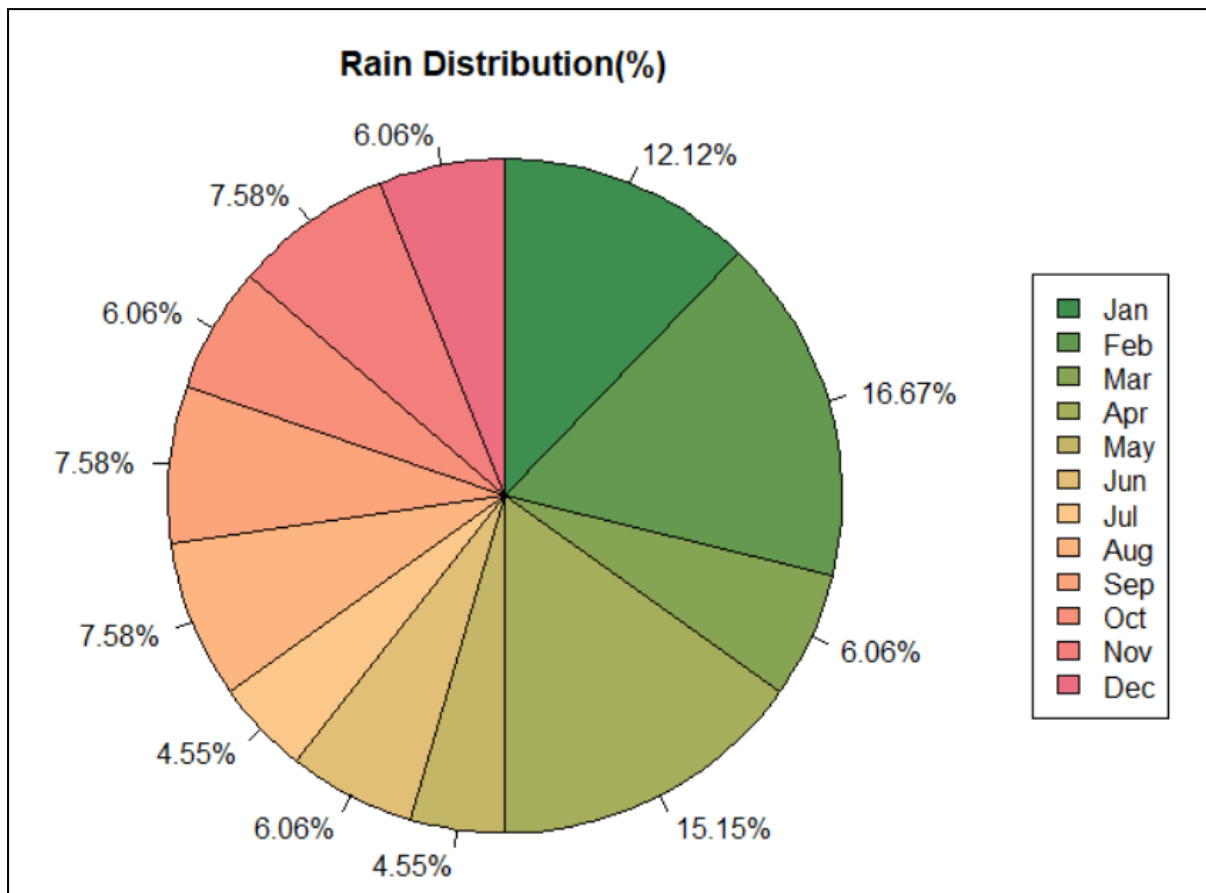


Figure 11: Data Visualization of Analysis 1

From this analysis, the raining season is distributed in January to April which contributes about 50% of the whole pie chart. February has the most raining days, which is 16.67%. On the other hand, May and July both possess the least raining days, which is only 4.55% for each month.

This analysis will help the following analysis as it allows us to study the regular pattern for the raining season or the month which has less rain distribution.

### 2.2.2 Analysis 2: Find the total rainfall by month.

This analysis is used to the amount of rainfall for each month. It will show a simple combination of line and point graph with “Month” as x-axis and “Rainfall” as y-axis. The main purpose of doing this analysis is to find the month which has the most rainfall or the least rainfall.

Code:

```
Total_rainfall = aggregate(Rainfall ~ Month, weather_data, sum)
Total_rainfall$Month <- as.integer(Total_rainfall$Month)

ggplot(Total_rainfall, aes(x = Month,y = Rainfall))+
  geom_line(color = "grey")+
  geom_point(shape = 21, color = "black",fill = "#69b3a2", size = 6)+
  ggtitle("Total Rainfall(mm) in 2020") +
  scale_x_continuous(breaks = seq(0,12,1),labels = function(x) month.abb[x]))
```

Figure 12: Code of Analysis 2

The amount of rainfall is calculated using aggregate function group by months. The calculated data is stored in another data frame called “Total\_rainfall” with the “Month” variables. The data frame is shown as following diagram:

	Month	Rainfall
1	1	118.0
2	2	78.4
3	3	38.6
4	4	69.8
5	5	30.0
6	6	17.4
7	7	12.8
8	8	23.8
9	9	42.0
10	10	17.6
11	11	40.8
12	12	33.6

Figure 13: Total\_rainfall data frame

Before plotting the diagram, the “Month” values are changed as integer values for showing the month names in the x-axis. The plotted diagram is given a title “Total Rainfall(mm) in 2020” and the x-axis is shown as month abbreviation using scale\_x\_continuous function.

Output:

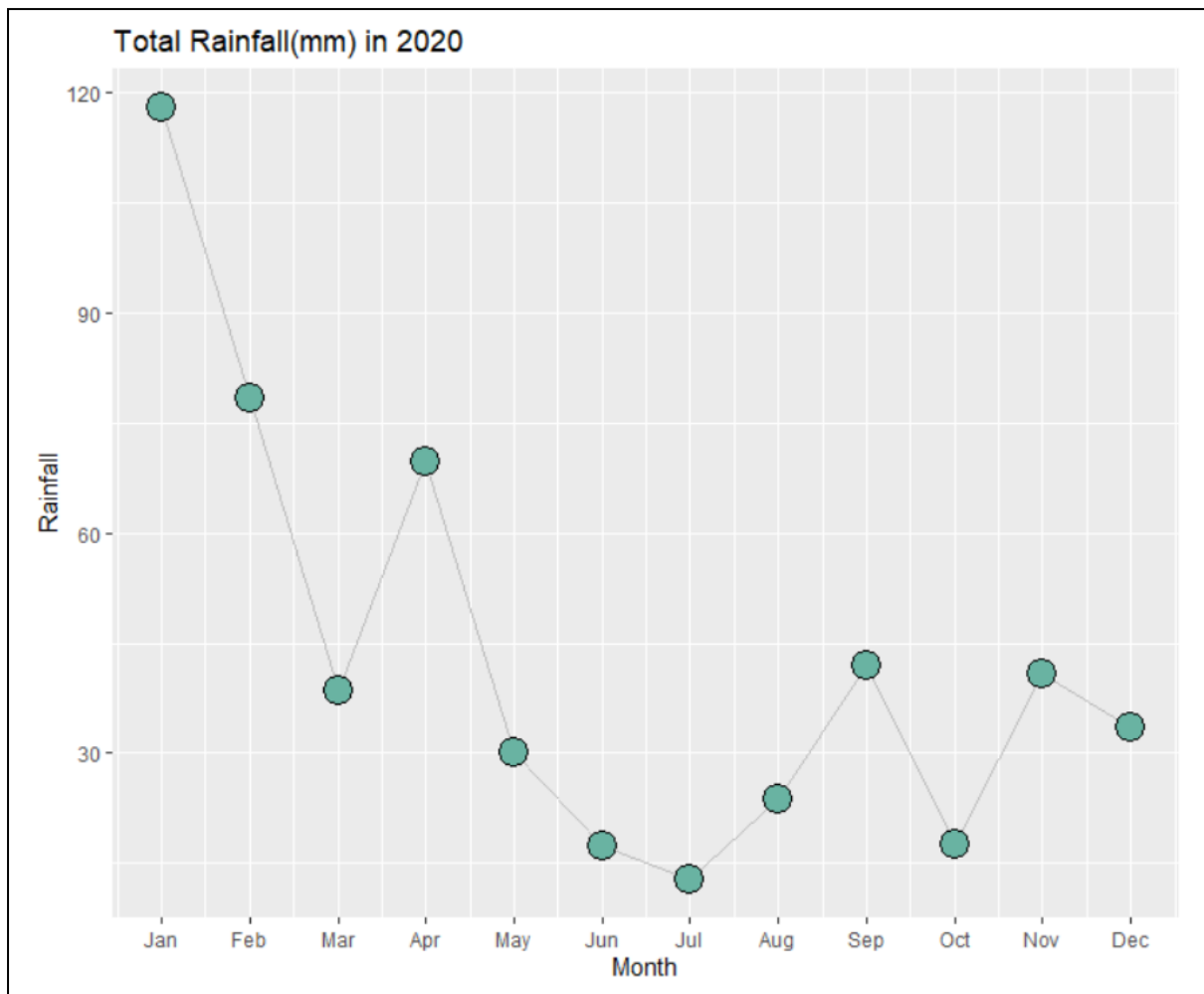


Figure 14: Data Visualization of Analysis 2

The diagram above shows that the month with the most amount of rainfall is January, which is nearly 120mm and the least amount of rainfall is in July, which is less than 15mm.

With this analysis, the following analysing process can be focused on January and July to find the reason for the distribution of rainfall amount.

### 2.2.3 Analysis 3: Compare the status of 9am cloud during raining day with non-raining day.

This analysis is used to find the cloud status at 9am. It will show a bar graph that indicate the frequency of cloud9am according to raining and non-raining day.

Code:

```
Cloud_Data <- data.frame(weather_data %>% group_by(Cloud9am) %>% count(RainToday))
head(Cloud_Data)
colnames(Cloud_Data)[3] <- "Frequency"

ggplot(Cloud_Data, aes(x=Cloud9am, y=Frequency, fill=RainToday)) +
  geom_bar(stat='identity', alpha = 0.5, position='dodge') +
  geom_text(aes(label = Frequency),
            position = position_dodge(width = 1),
            vjust = -0.5, size = 3) +
  xlab("Cloud9am(OKtas)") +
  scale_x_continuous(breaks = seq(0,8,1)) +
  scale_y_continuous(breaks = seq(0,100,20))
```

Figure 15: Code of Analysis 3

The used technique before the data visualization is same with the technique used in analysis 1. The Cloud 9am data is extracted from the original weather data and stored in “Cloud\_Data” data frame in the form of sum according to the status of “RainToday”. Before plotting the bar graph, the “n” column is renamed as “Frequency” for the ease of reviewing the data.

	Cloud9am	RainToday	Frequency
1	0	No	33
2	1	No	103
3	1	Yes	13
4	2	No	12
5	2	Yes	5
6	3	No	13
7	3	Yes	4
8	4	No	9
9	5	No	16
10	5	Yes	7
11	6	No	23
12	6	Yes	2
13	7	No	70
14	7	Yes	16
15	8	No	21

Figure 16: Cloud\_Data Data Frame

The plotted bar diagram is come with the label for the frequency of each Cloud9am bar using geom\_text command. Besides, the number of x-axis is increasing by 1 from 0 to 8 (scale\_x\_continuous), and the number of y-axis is increasing by 20 from 0 to 100

(scale\_y\_continuous). The purpose of using the customized x-axis and y-axis is to have a clearer vision for the label of the diagram.

Output:

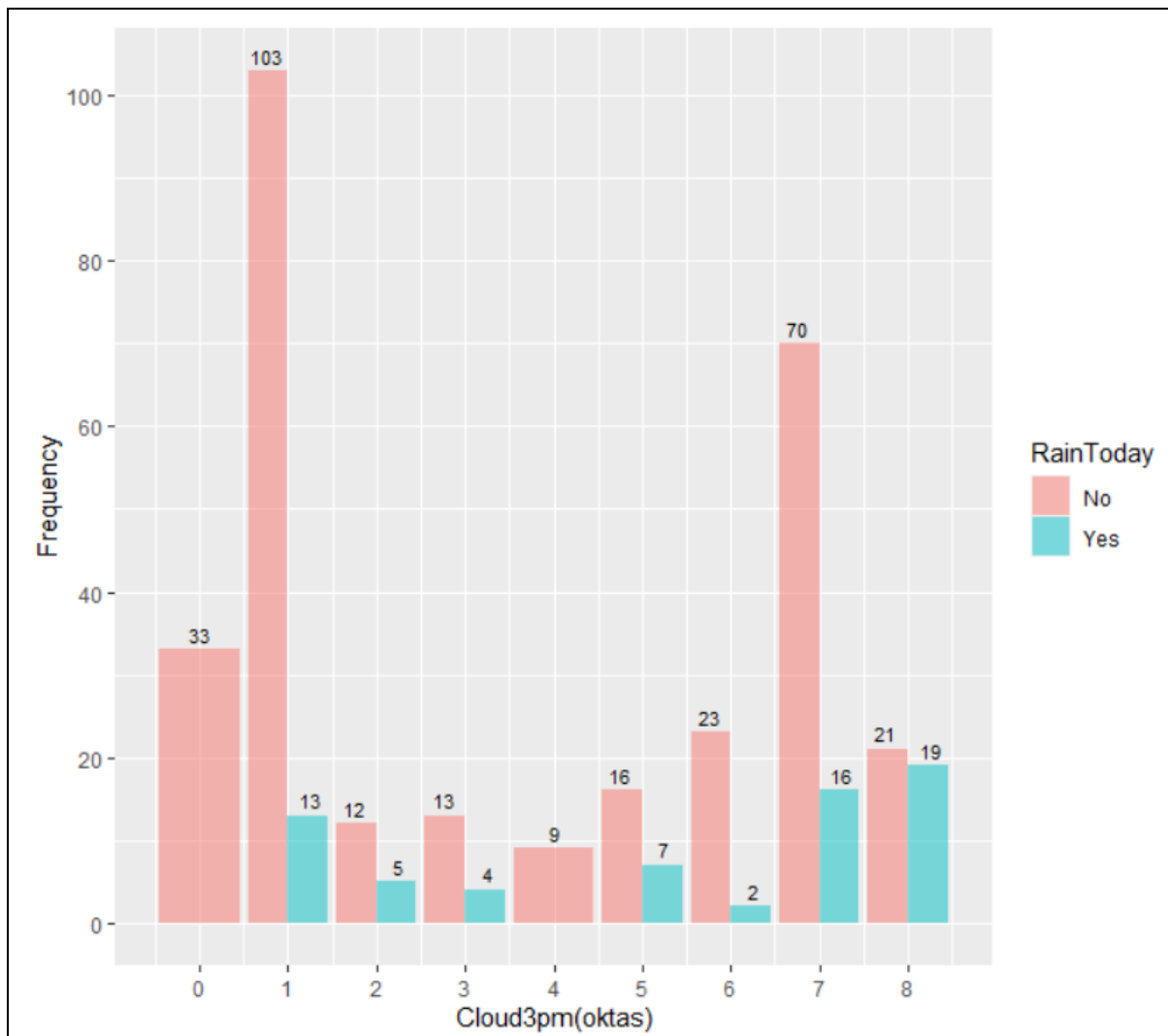


Figure 17: Data Visualization of Analysis 3

From this analysis, it shows that there are not any raining days when the cloud9am is 0 okta and 4 oktas. Most of the raining days occurs when status of cloud at 9am is 7 oktas and 8 oktas. Compared among the non-raining days, cloud9am at 1 okta and 7 oktas takes up a large slice in the frequency.

Overall, when the status of cloud9am is over or equal to 7 oktas, the probability of raining is the highest. Besides, there is no possible to rain when the cloud9am is equal to 0 okta. From the research from SciJinks, the main reason for raining is the clouds that made of water droplets. The sea water or the ice will evaporate and become water droplets. When water droplets condense on each other, it will form cloud. As the water droplets suspended in cloud

become heavy, it will fall down and become rain (What Makes It Rain? | NOAA SciJinks – All About Weather, 2021).

This analysis is basically used to find the relationship between cloud and raining day. Theoretically, the more cloud indicates the more probability of raining. From the result obtained from the given dataset, it did not show a strong positive correlation between the Cloud9am and RainToday as the raining data is not enough, but it is reasonable there is no rain when the cloud9am is 0 okta and most of the raining day happens when the cloud9am is 8 oktas. Hence, the assumption that the more cloud will result in higher probability of raining can be made.

#### 2.2.4 Analysis 4: Find the frequency of raining according to temperature at 3pm.

The aim of having this analysis is to find the probability of raining according to temperature at 3pm. This analysis will use histogram with different colour based on the frequency of raining days.

Code:

```
Rain_Temp <- filter(weather_data, RainToday == "Yes")
nrow(Rain_Temp)

ggplot(Rain_Temp, aes(x = Temp3pm)) +
  geom_histogram(color="orange", aes(fill = ..count..)) +
  ggtitle("Temperature(°C) at 3pm during Raining Day") +
  scale_fill_gradient("Count", low = "green", high = "red") +
  scale_x_continuous(breaks = seq(5,40,2))
```

Figure 18: Code of Analysis 4

Firstly, the “Rain\_Temp” data frame is used to store the raining day data only using filter command. The visualization is using histogram and plot the frequency of raining according to Temp3pm with different colour (color="orange", aes (fill = ..count..)). Green colour indicates low frequency and red colour indicates high frequency. The graph is given a title “Temperature(°C) at 3pm during Raining Day”, and the number of x-axis is increasing 2 from 5 to 40.



Output:

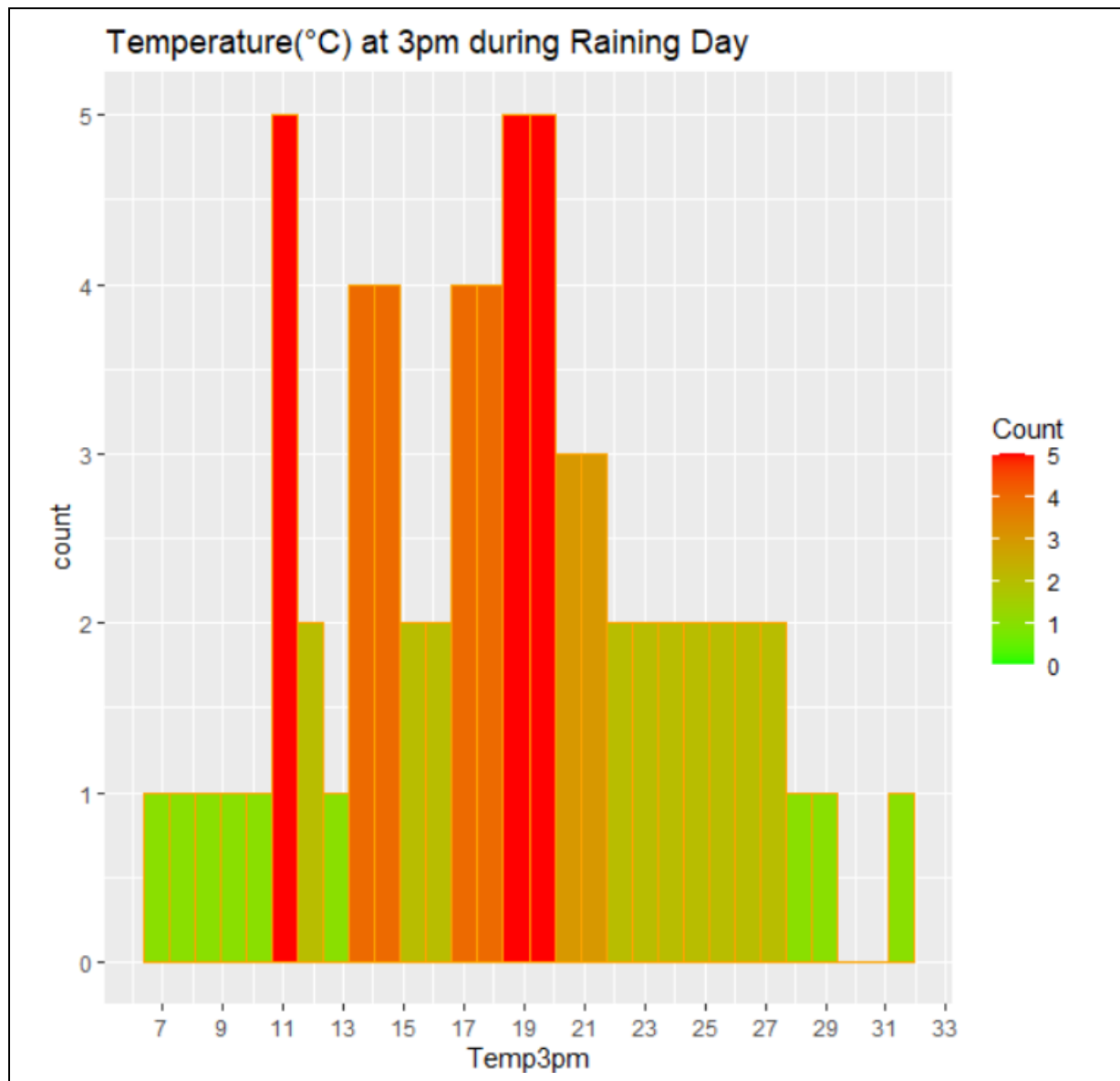


Figure 19: Data Visualization of Analysis 4

From the diagram above, there is a high probability of raining when the temperature at 3pm is around 11°C, from 13°C to 15°C, and from 17°C to 19°C. There will be no raining day when the temperature is about 30°C and over 32°C.

The academic literature on the relationship between rainfall and temperature has focused on the provision of the higher temperature indicates lower rainfall. The literature found out the negative correlation between the rainfall and the temperature in South Africa (Tinyiko R. Nkuna and John O. Odiyo, 2011). The range of temperature in South Africa is from 7°C to 27°C. Most of the raining days happens in July and August which possess the least average temperature, around 12.75°C (Climatestotravel, 2017).

Cape Town - Average temperatures						
Month	Min (°C)	Max (°C)	Mean (°C)	Min (°F)	Max (°F)	Mean (°F)
January	16	26	21	61	79	69.8
February	16	27	21.5	61	81	70.7
March	14	25	19.5	57	77	67.1
April	12	23	17.5	54	73	63.5
May	9	20	14.5	48	68	58.1
June	8	18	13	46	64	55.4
July	7	18	12.5	45	64	54.5
August	8	18	13	46	64	55.4
September	9	19	14	48	66	57.2
October	11	21	16	52	70	60.8
November	13	24	18.5	55	75	65.3
December	15	25	20	59	77	68
Year	11.5	22	16.7	52.7	71.5	62

Figure 20: Average Temperature in South Africa (Climatestotravel, 2017).

Cape Town - Average precipitation			
Month	Millimeters	Inches	Days
January	12	0.5	2
February	8	0.3	2
March	17	0.7	3
April	45	1.8	6
May	85	3.3	9
June	80	3.1	9
July	85	3.3	10
August	70	2.8	10
September	45	1.8	7
October	30	1.2	5
November	17	0.7	3
December	11	0.4	2
Year	505	19.9	68

Figure 21: Average Precipitation in South Africa (Climatestotravel, 2017).

The highest frequency is only 5 for a single temperature. As the given dataset is not enough for proving the statement (lower temperature higher rainfall), the plotted diagram slightly indicates the negative correlation between RainToday and Temp3pm. Hence, it can be assumed that there will be a high probability of raining when the temperature is low.

### 2.2.5 Analysis 5: Find the relationship between humidity and RainToday.

The relationship between humidity and rain status is sketched as box plot. The main purpose of doing this analysis is to find the range of humidity during the raining day.

Code:

```
humidity_data <- subset(weather_data, select = c(Humidity9am, Humidity3pm, RainToday))
humidity_data = melt(humidity_data)

ggplot(humidity_data, aes(x = variable, y = value, color=RainToday)) +
  geom_boxplot(alpha=0.3) +
  facet_wrap(~RainToday) +
  xlab("Humidity") +
  ylab("Value(%)") +
  scale_y_continuous(breaks = seq(0,100,20))
```

Figure 22: Code of Analysis 5

To find the relationship between humidity and RainToday, data such as Humidity9am, Humidity3pm and RainToday, is retrieved from the original weather data and stored in a new data frame, called “humidity\_data” using subset command. An extra feature (melt command) is used in this analysis and will be explain in extra feature part.

The plotted diagram is divided into two categories according to the status of raining of that day(facet\_wrap(~RainToday)). Red colour indicates the non-raining day and blue colour indicates the raining day.

Output:

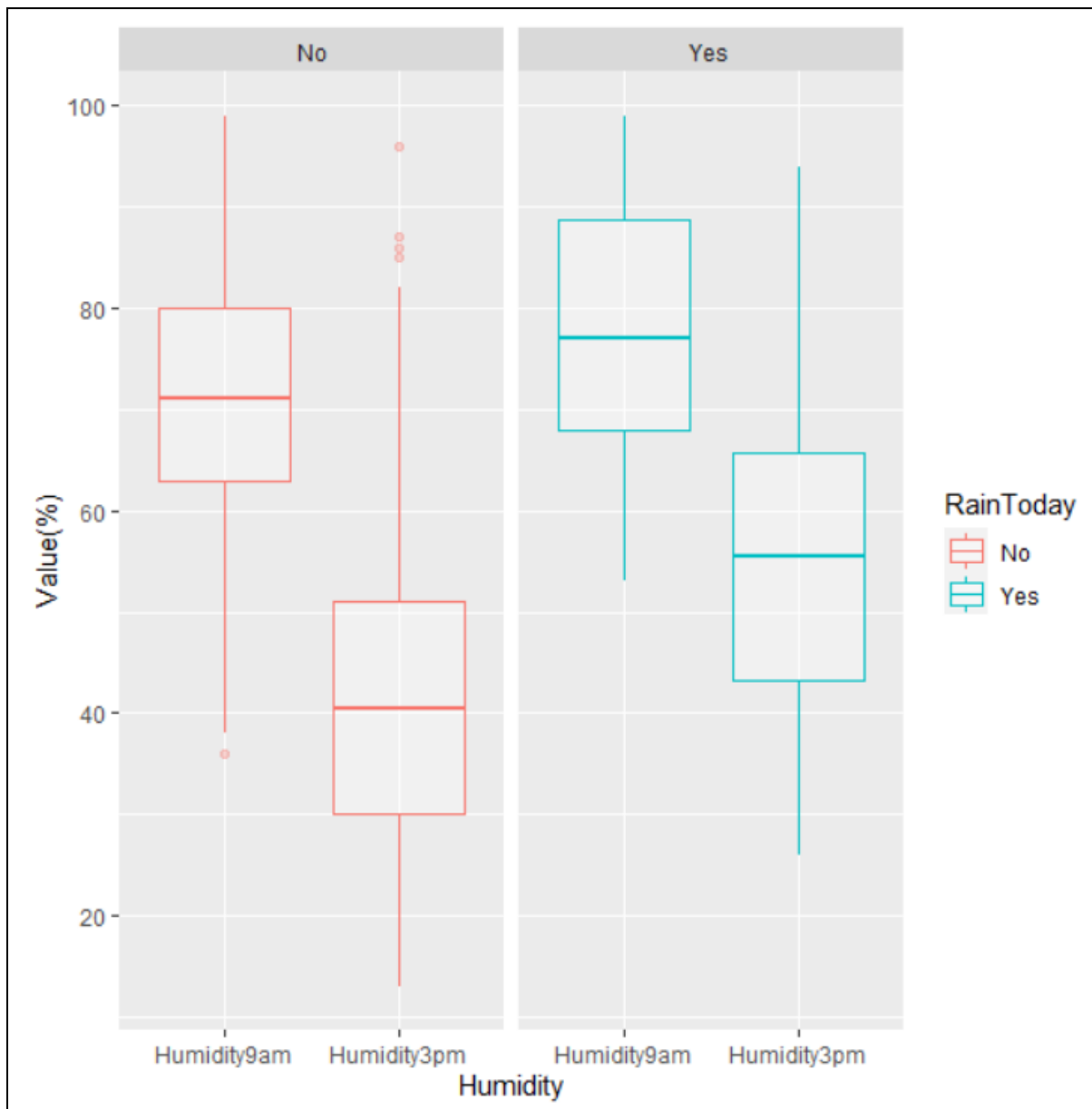


Figure 23: Data Visualization of Analysis 5

From the data visualization above, there are strong signs that the humidity at 9am and 3pm during raining days are higher than the humidity during non-raining days. During raining days, the median of the Humidity9am is between 75% and 80%, while median of the Humidity3pm is around 55%. Besides, the box plot also shows that the humidity at 9am is higher than 3pm.

The result of this analysis is proven by Smart Frog (2016), which found that relative humidity in the air is affected by rain. As the humidity rate is 100% when raining, and the humidity rate will be reduced as time goes on. The longer the raining, the higher the humidity

rate as air will continuously draw the water (How Rain and Humidity Connected? | Smart Fog, 2016).

Therefore, a small conclusion can be made that the more raining days will bring more humidity in that month.

2.2.6 Analysis 6: Find the relationship between humidity at 9am and evaporation based on RainToday.

To find the status of raining day, the relationship between humidity at 9am and evaporation will indirectly indicate the relationship between evaporation and rain. This analysis will be plotted in scatter plot.

Code:

```
ggplot(weather_data, aes(Evaporation, Humidity9am)) +  
  geom_point(aes(color=RainToday)) +  
  ggtitle("HUMIDITY9AM(%) VS EVAPORATION(mm) based on RainToday") +  
  stat_smooth(method=lm)
```

Figure 24: Code of Analysis 6

The point graph will be given a title “HUMIDITY9AM(%) VS EVAPORATION(mm) based on RainToday”. Colour of the point will be altered according to the status of the RainToday. Red colour indicates non-raining days, while blue colour indicates raining days. Besides, an average line will also be included in the diagram to provide a more apparent vision when observing the patterns of the two variables (stat\_smooth(method=lm)).

Output:

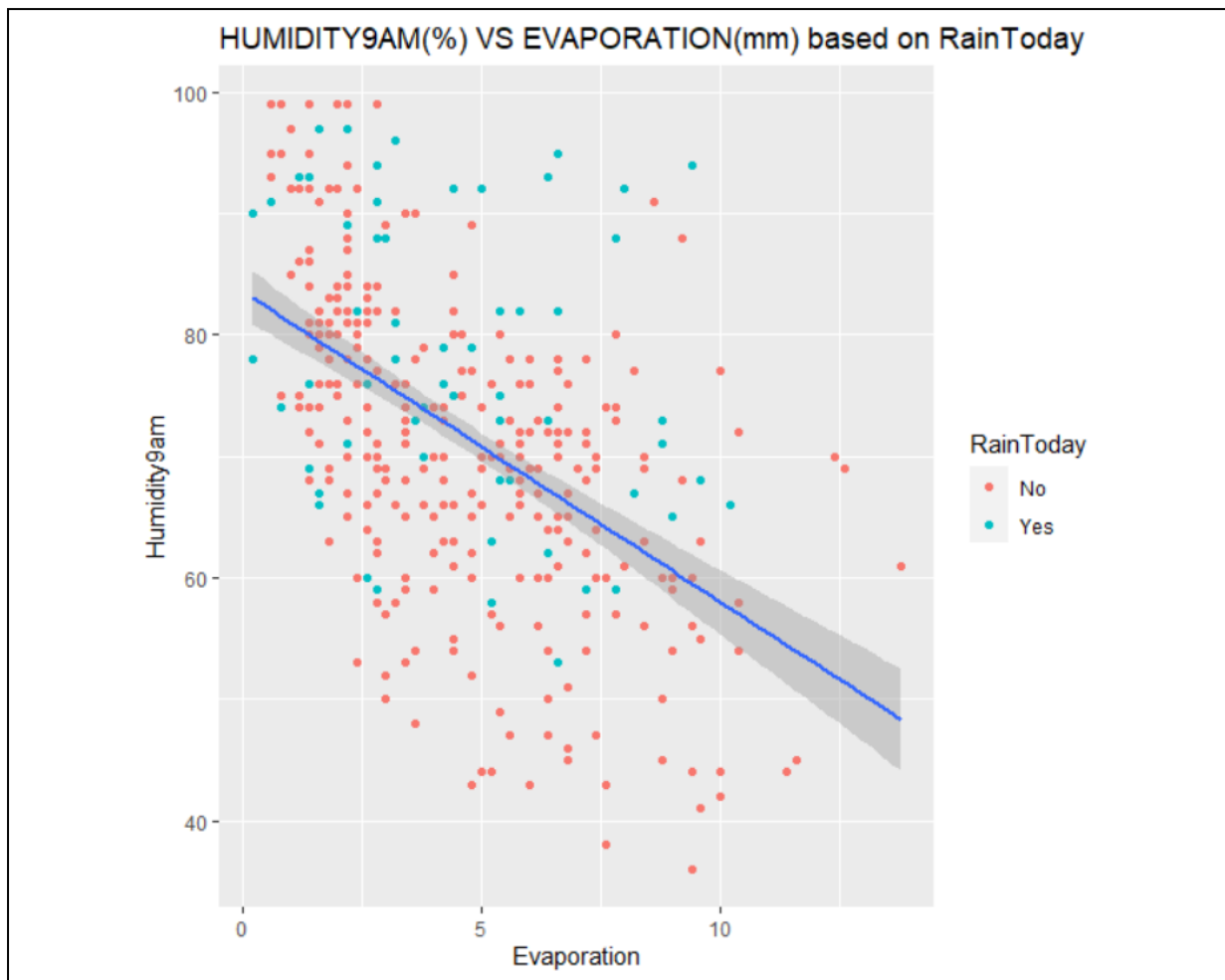


Figure 25: Data Visualization of Analysis 6

It is obvious to find that the lower humidity rate will cause the higher evaporation rate. The result of this analysis is because the air will become drier when the relative humidity in the air becomes lower. Hence, there will be less evaporation occur (Evaporation and Climate, n.d.).

Consequently, in a month, the more raining days, the more the relative humidity, the less the evaporation.

### 2.2.7 Analysis 7: Find the relationship between sunshine and evaporation.

After knowing the relationship among humidity, evaporation, and rain, it is required to find the relationship between sunshine and evaporation in order to find the detail of weather data during raining day. The data visualization in this analysis is an advanced version for the scatter plot that used in analysis 6.

Code:

```
Sun_Eva <- subset(weather_data, select= c("Sunshine","Evaporation"))
summary(Sun_Eva,maxsum = 6)
Sun_Eva = na.omit(Sun_Eva)

ggplot(Sun_Eva, aes(x=Sunshine, y=Evaporation)) +
  geom_hline(yintercept = median(Sun_Eva$Evaporation), size= 1.5, color=("red")) +
  geom_vline(xintercept = median(Sun_Eva$Sunshine), size= 1.5, color=("yellow")) +
  geom_point(aes(color=Sunshine)) +
  geom_smooth(formula = y ~ x, method = 'loess', color = "black", size = 1, se = FALSE) .
  xlab("Sunshine(hrs/day)") + ylab("Evaporation(mm)") +
  scale_x_continuous(breaks = seq(0,15,2)) +
  scale_y_continuous(breaks = seq(0,15,2))
```

Figure 26: Code of Analysis 7

“Sunshine” and “Evaporation” values are taken out from original dataset and stored in “Sun\_Eva” data frame. The summary that includes minimum, maximum, median, and other values is shown as the following diagram:

Sunshine	Evaporation
Min. : 0.000	Min. : 0.200
1st Qu.: 5.950	1st Qu.: 2.200
Median : 8.600	Median : 4.200
Mean : 7.909	Mean : 4.522
3rd Qu.:10.500	3rd Qu.: 6.400
Max. :13.600	Max. :13.800
NA's :3	

Figure 27: Summary for Sun\_Eva data frame

The diagram above shows that there are three NA values in the sunshine variables. Hence, the NA values are omitted to avoid the influence for the quality of the plotted diagram (na.omit()).

The advanced scattered plot will include a vertical line with yellow colour that indicates the median for the sunshine variables (geom\_vline()) and a horizontal line with red colour that indicates the median for the evaporation variables (geom\_hline()). The method used to draw the relation between the sunshine duration and evaporation is local regression (“loess”). Besides, the grey area that shows the confidence interval is also omitted to ensure a clearer vision (se = FALSE).



Output:

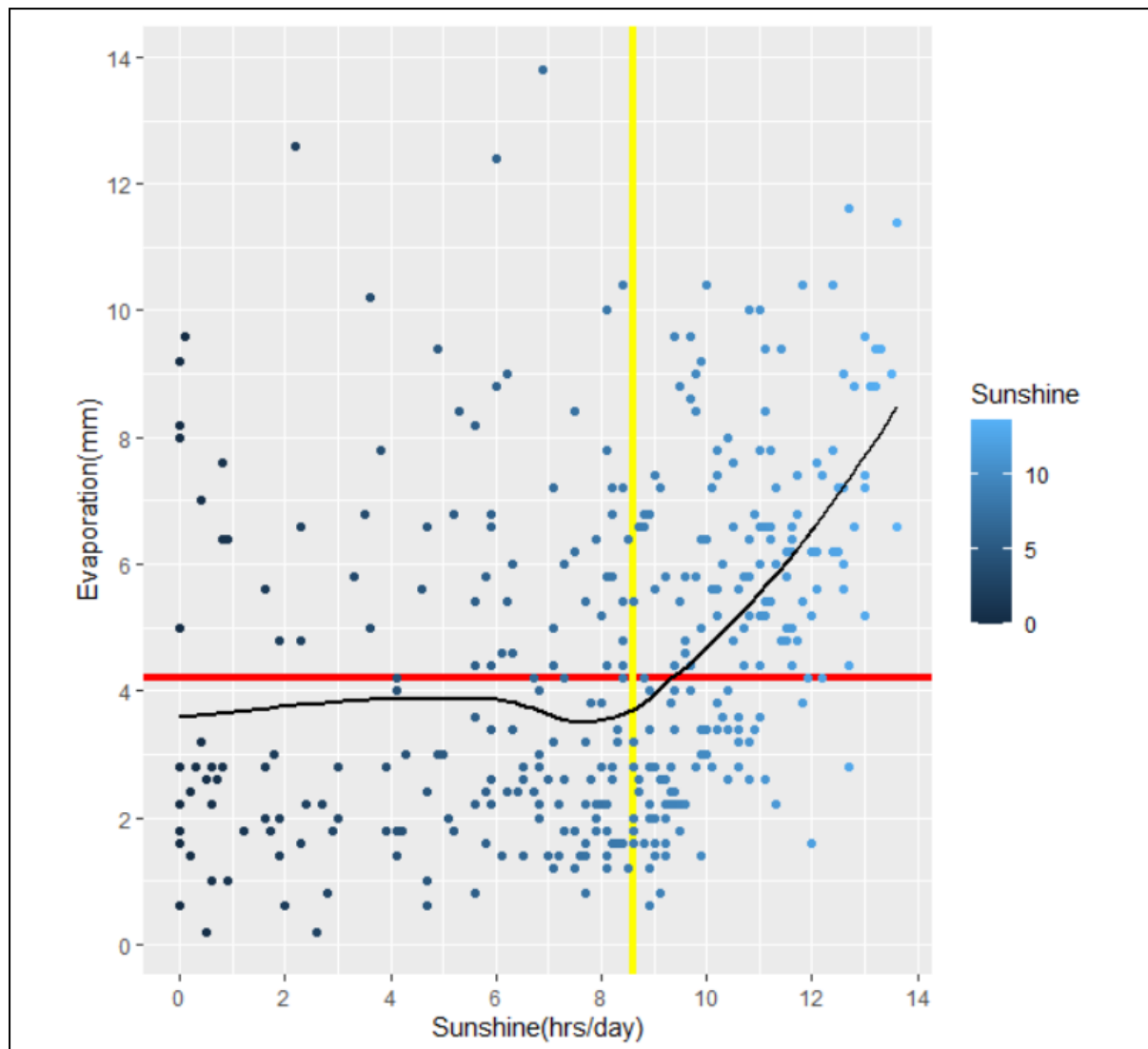


Figure 28: Data Visualization of Analysis 7

This analysis shows that the median for the sunshine is between 8 hrs/day and 9 hrs/day (8.6hrs/day). The median for the evaporation is slightly more than 4mm (4.2mm). Apart from that, the black line also shows that the sunshine is positive correlated with evaporation.

The longer the sunshine duration in a day indicates the more sunlight and the more solar radiation. As the solar radiation will increases the temperature, the energy content in a water molecule will also raise up. The higher energy content will cause the pushing of water molecule surface, letting them easier to diffuse into the air. This will form an evaporation process in the nature (Claude E. Boyd, 2012). The relationship between temperature and saturation vapor pressure is shown as the following diagram:

Temperature (° C)	↕	Saturation Vapor Pressure (mm Hg)
0		4.579
5		6.543
10		9.209
15		12.788
20		17.535
25		22.377
30		31.695
35		42.175
40		55.324

Figure 29: Relationship between temperature and saturation vapor pressure (Claude E. Boyd, 2012).

Hence, theoretically, the longer the daytime sunshine duration, the higher the evaporation in a day.

### 2.2.8 Analysis 8: Find the relationship between sunshine and cloud at 3pm.

The sunshine is mainly affected by cloud as they are incompatible. This analysis will be shown in combination of line and point graph.

Code:

```
weather_data %>% group_by(Cloud3pm) %>% na.omit(Sunshine) %>%  
  summarise(Total = n(), SunshineToCloud3pm = sum (Sunshine) / Total) %>%  
  ggplot(aes(x = Cloud3pm, y = SunshineToCloud3pm)) +  
  geom_point(color="violetred") +  
  geom_line(color="blue") + xlab("Cloud3pm(oktas)") +  
  geom_text(aes(label = round(SunshineToCloud3pm,2)),hjust=0, vjust=0)
```

Figure 30: Code of Analysis 8

All the commands in this analysis are completed in an execution using piping feature. Before drawing the diagram, data pre-processing has been done to transform the data into more understandable format. The original data is grouped by Cloud3pm. Sum of each cloud status is stored as a column and the result of sunshine divided by total of cloud status is also stored as another column using summarise command. Point in the graph is violet red colour, while the line is blue colour. Besides, labels for each of the point is shown in the plotted graph with the value using geom\_text.

```
# A tibble: 9 x 3  
  Cloud3pm Total SunshineToCloud3pm  
    <int> <int>          <dbl>  
1         0      6          10.9  
2         1     96          10.2  
3         2     34           9.73  
4         3     20           9.58  
5         4     20           9.16  
6         5     25           8.54  
7         6     29           7.57  
8         7     75           5.30  
9         8     23           2.20
```

Figure 31: Value for data visualization

Output:

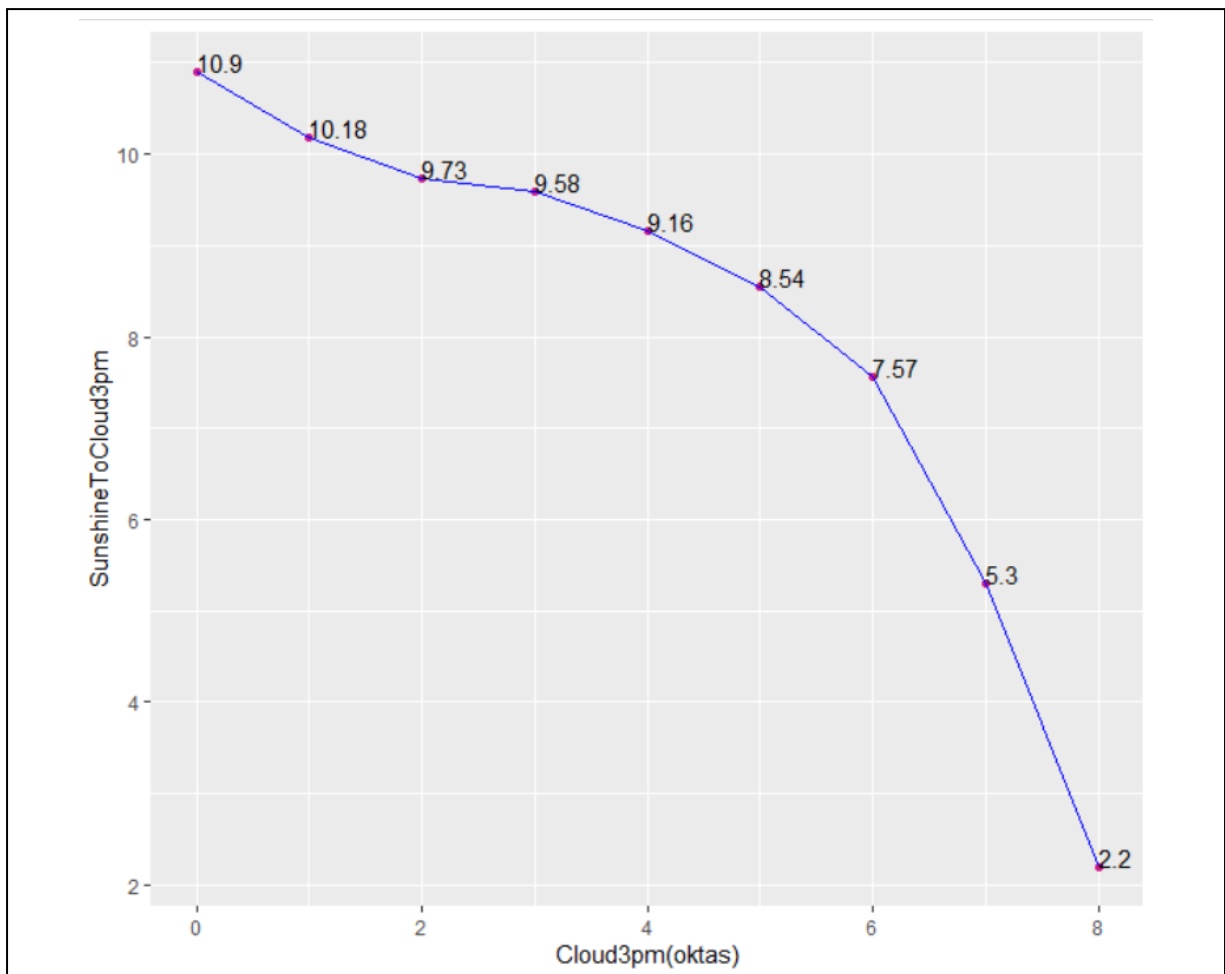


Figure 32: Data Visualization of Analysis 8

In this data visualization, the point indicates the ratio of sunshine for each cloud status at 3pm. It shows the highest ratio of sunshine when the Cloud3pm is 0 okta, which is 10.9. The lowest ratio happens when the Cloud3pm is 8 oktas, which is only 2.2.

The result of the is proven by Dorota Matuszko (2011), he found out the coefficient of determination between mean daytime cloudiness and mean daytime sunshine duration is 0.98. It means the longer sunshine duration can be explained by cloudiness as longest duration of sunshine is recorded when there is less cloud in the sky (Matuszko, 2011). However, the research also stated that other factors such as temperature will also affect the sunshine duration.

Hence, from this analysis, it totally shows a strong negative correlation between the sunshine duration and the cloud at 3pm, the more the cloud, the less the sunshine duration in a day.

### 2.2.9 Analysis 9: Status of the amount of rainfall for the whole year.

This analysis is used to divide the amount of rainfall into “High”, “Moderate”, or “Low”. To do this analysis, “raining\_day” data frame is taken from the data exploration that completed before the data analysis which contains only the raining weather dataset. The data visualization for this analysis will be a 3D pie chart.

Code:

```
Up_Line <- mean(raining_day$Rainfall) * 1.1
Down_Line <- mean(raining_day$Rainfall) * 0.9
raining_day <- mutate(raining_day, RainStatus = ifelse(Rainfall > Up_Line, "High",
                                                       ifelse(Rainfall < Down_Line, "Low", "Moderate")))

names(raining_day)
value <- c()
value[1] <- nrow(filter(raining_day, RainStatus == "High"))
value[2] <- nrow(filter(raining_day, RainStatus == "Moderate"))
value[3] <- nrow(filter(raining_day, RainStatus == "Low"))
name <- c("High", "Moderate", "Low")
pie3D(value, radius = 0.9, labels = name,
      explode=0.1, theta = 0.8, labelcex = 1.2,
      main = "Amount of rainfall in 2020",
      col=c("brown", "#ddaa00", "pink"))
```

Figure 33: Code of Analysis 9

Before plotting the diagram, data pre-processing is needed to be done for the convenience of analyse the data. Moderate amount of rainfall is distributed between up line which is 110% of the mean of rainfall and down line which is 90% of the mean of rainfall. High amount of rainfall is the amount that over the moderate amount of rainfall, and low amount of rainfall is the amount that less than the moderate amount of rainfall. The distribution of the status of rainfall is completed using the ifelse command. The amount of each status of rainfall is calculated using nrow function.

Output:

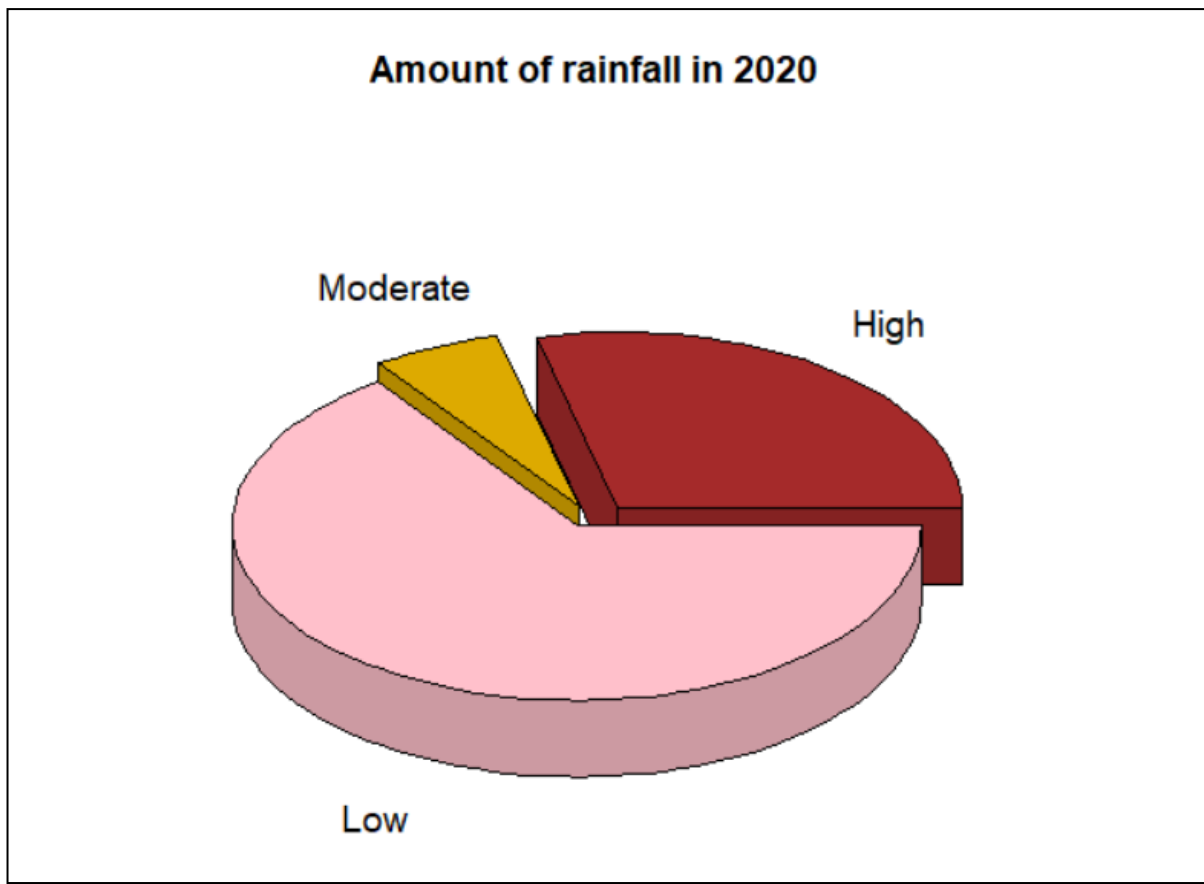


Figure 34: Data Visualization of Analysis 9

The plotted diagram shows that low amount of rainfall possesses a large slice compared to moderate and high amount of rainfall in the year 2020.

### 2.2.10 Analysis 10: Mean of weather data group by months.

This analysis is extended from analysis 1 and 2 to find the general pattern of each weather data for the raining days and non-raining days.

Code:

```
data_mean <- subset(weather_data, select=c(Month, Cloud9am, Cloud3pm, Temp3pm, Humidity9am,
                                           Humidity3pm, Evaporation, Sunshine))
data_mean <- na.omit(data_mean)

data_mean %>% melt(id = c("Month")) %>%
  dcast(Month ~ variable, fun.aggregate = mean) %>%
  format(., digits = 2) %>%
  cbind(., Rainfall = Total_rainfall$Rainfall, Num_RainingDay = Yes_No$Num_RainingDay)
```

Figure 35: Code of Analysis 10

To find the mean of each weather data grouped by months, data such as cloudiness, temperature, humidity, evaporation, and sunshine are taken out from the original dataset and stored in a new data frame, called “data\_mean” using subset command. As summary in data pre-processing has shown that there are several NA values in the sunshine, “na.omit” command has to be executed to remove the NA values in the new dataset.

After all the variables grouped by months(melt), dcast command allows the dataset to find the mean of the values according to their groups. All the values stored in the dataset have to be formatted as two digits. To have a better view of the dataset, the new dataset (“data\_mean”) is combined with the dataset made in analysis 1 and 2.

Output:

	Month	Cloud9am	Cloud3pm	Temp3pm	Humidity9am	Humidity3pm	Evaporation	Sunshine	Rainfall	Num_RainingDay
1	01	4.5	5.0	23	70	44	5.8	7.6	118.0	8
2	02	5.1	4.9	23	70	49	5.7	8.4	78.4	11
3	03	4.0	3.8	28	65	39	8.0	9.4	38.6	4
4	04	4.6	4.8	23	73	46	6.0	8.1	69.8	10
5	05	2.9	3.7	25	71	36	5.9	9.0	30.0	3
6	06	3.1	3.8	18	75	42	3.7	8.0	17.4	4
7	07	3.4	2.9	16	81	47	2.3	7.5	12.8	3
8	08	5.1	4.4	13	84	59	1.7	5.2	23.8	5
9	09	4.0	4.5	11	84	57	1.6	5.9	42.0	5
10	10	3.9	4.4	11	70	50	2.7	7.2	17.6	4
11	11	3.3	2.9	17	61	36	4.6	8.7	40.8	5
12	12	2.6	3.2	22	62	30	6.2	9.9	33.6	4

Figure 36: Output of Analysis 10

From all the analysis that made in question 1, we can conclude that the more cloud and lower temperature will result in higher probability of raining. The more raining days in a month will cause high relative humidity, less evaporation, less daytime sunshine duration, and more cloud.

As January occupies the most amount of rainfall and February possesses the most raining days, their cloud9am and cloud4pm are considered high compared to other months. However, temp3pm is slightly high compared to other months, and humidity at 9am and 3pm, evaporation, and sunshine are at moderate level, which do not completely follow the result of the analysis.

From analysis 1, May and July have the least raining days. Besides, from analysis 2, July also possesses the least amount of rainfall. The cloudiness at 9am and 3pm in May and July are lower than the cloudiness in the first half year. Temperature at 3pm in May is considered high that causes the low number of raining days but temperature in July is at moderate level. The inconsequence also shows in humidity at 9am and 3pm as humidity in May and July is also considered about average among all the months.

In short, the data in this dataset only slightly tallies with the result made from the data analysis. As the given data for doing this analysis is not enough, a research in deeper and wider has to be done to complement the flaw of the analysis to ensure a more precise weather forecast.



### 2.3 Question 2: When is the best time for outdoor activities?

Weather is one of the factors that will affect physical activities of human. It is proven by the research that investigated the correlation between time spent outdoors and certain weather conditions, such as temperature, humidity, and wind speed. Among all the variables, temperature is the most influential elements that will affect the outdoor activities (Al-Mohannadi and Qatar, 2006). The relationship between temperature and physical activity of residents in a community is shown as the following diagram:

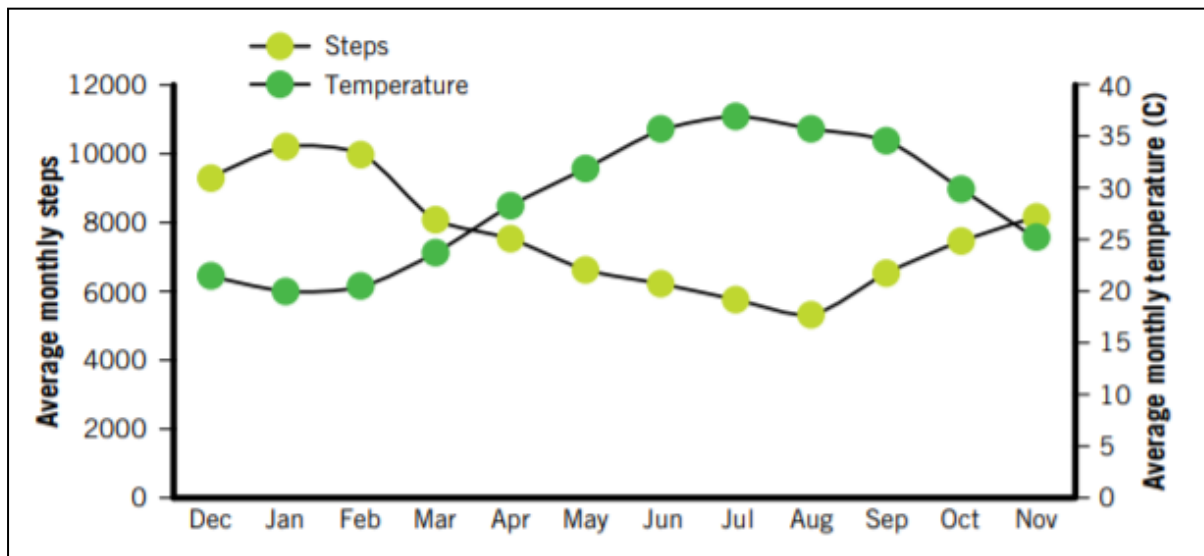


Figure 37: Relationship between temperature and outdoor activity (Al-Mohannadi and Qatar, 2006)

From the diagram above, the temperature in a month is strongly negative correlated with the number of steps. Hence, in this question, data analysis on the specific weather data that is temperature will be made to show the best time for outdoor activities.

### 2.3.1 Analysis 11: Find the daily temperature average using Minimum Temperature and Maximum Temperature.

At start, the analysis about average temperature in a day has to be made to show the status of temperature in the year 2020.

Code:

```
average_temp = (weather_data$MinTemp + weather_data$MaxTemp) / 2
ggplot(weather_data, aes(x = Date, y = average_temp)) +
  geom_line(alpha = 0.3) +
  geom_hline(yintercept = c(max(average_temp), min(average_temp)), size = 1.5, color = "red") +
  stat_smooth(formula = y ~ x, se = FALSE, color = "black", method = 'loess') +
  ggtitle("Average Temperature(°C) Recorded in USA") +
  xlab("Month") + ylab("Average Temperature") +
  scale_x_date(date_breaks = "1 month", date_labels = "%b") +
  scale_y_continuous(breaks = seq(0, 28, 4)) + expand_limits(y = 0)
```

Figure 38: Code of Analysis 11

The technique that used to plot the diagram is similar with the technique used in analysis 7. The point graph has been changed to line graph in this diagram to show the temperature trend in this year. The actual average of temperature has been plotted in blur line, while the clearer black line shows the temperature trends in this year.

Output:

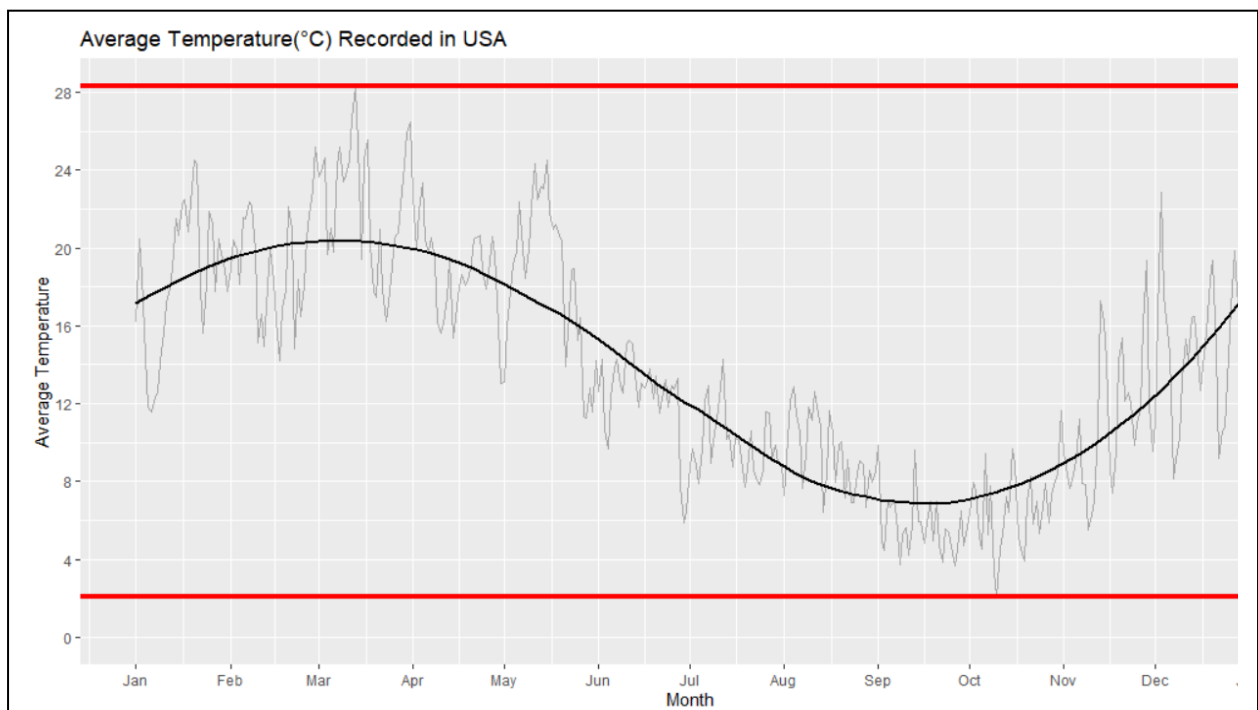


Figure 39: Data Visualization of Analysis 11

The diagram above is like a sine graph. The hot season is distributed in the months from February to May, while the cool season is distributed in the months from August to November.

Besides, the upper red line indicates the maximum average temperature which is nearly  $28^{\circ}\text{C}$ , and the lower red line indicates the minimum average temperature which is nearly  $2^{\circ}\text{C}$ .

Hence, from the analysis of average temperature using the maximum and minimum temperature of a day, it is obvious to show that second half of the year (July to December) is a better time for traveling compared to the first half year (January to June).

### 2.3.2 Analysis 12: Compare the number of hot days and cool days based on Temperature at 3pm.

Code:

```
Hot_Cool = select(weather_data, Temp3pm) %>%
  mutate(TempStatus = ifelse(Temp3pm > 25, "Hot", "Cool")) %>%
  count(TempStatus=="Hot")
colnames(Hot_Cool) = c("TempStatus", "Frequency")
Hot_Cool[1,1] = "Cool"
Hot_Cool[2,1] = "Hot"

Hot_Cool$fraction <- Hot_Cool$Frequency / sum(Hot_Cool$Frequency)
Hot_Cool$ymax <- cumsum(Hot_Cool$fraction)
Hot_Cool$ymin <- c(0, head(Hot_Cool$ymax, n=-1))
Hot_Cool$labelPosition <- (Hot_Cool$ymax + Hot_Cool$ymin) / 2
Hot_Cool$label <- paste0(Hot_Cool$TempStatus, "\n value: ", Hot_Cool$Frequency)

ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus)) +
  geom_rect() +
  geom_label(x=3.5, aes(y=labelPosition, label=label), size=5) +
  ggtitle("Number of hot days and cool days") +
  scale_fill_brewer(palette=1) +
  coord_polar(theta="y") +
  xlim(c(2, 4)) +
  theme_void() +
  theme(plot.title = element_text(hjust=0.5), legend.position = "none")
```

Figure 40: Code of Analysis 12

To find the number of hot and cool days, temperature at 3pm in the year 2020 has to be taken out from the original dataset and stored in a new data frame, called “Hot\_Cool”. The temperature at 3pm is divided into two categories that are “Hot” and “Cold”. According to Think Metric (2021), hot days indicates the temperature is over 25°C and the rest indicates the cool days. In this data pre-processing, piping technique is used again to combine all the commands.

Apart of the frequency, to plot a donut chart, the fraction, y-axis maximum and y-axis minimum of each variables are needed to be found out. Finally, the “Hot\_Cool” data frame used for data visualization is shown at the following diagram:

	TempStatus	Frequency	fraction	ymax	ymin	labelPosition	label
1	Cool	289	0.7896175	0.7896175	0.0000000	0.3948087	Cool\n value: 289
2	Hot	77	0.2103825	1.0000000	0.7896175	0.8948087	Hot\n value: 77

Figure 41: Hot\_Cool data frame

Output:

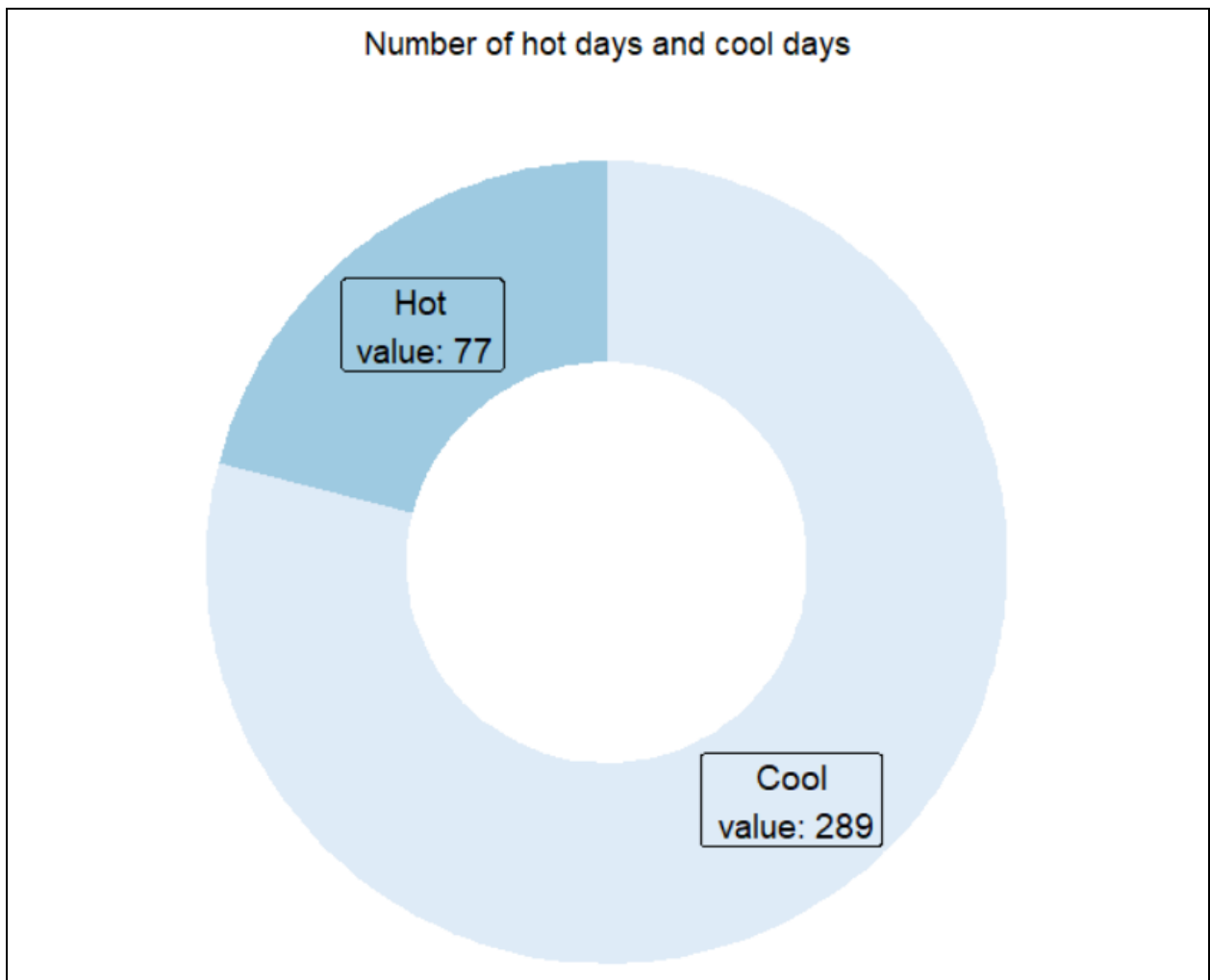


Figure 42: Data Visualization of Analysis 12

From the plotted donut diagram, the number of cool days is more than the number of hot days in U.S. as hot has a value of 77 and cool has a value of 289. Hence, U.S. residents have more opportunities to have outdoor activities and interact with each other in the year 2020.

### 2.3.3 Analysis 13: Find the median and mean of temperature at 9am according to RainToday categorized by cloud at 9am

Code:

```
ggplot(weather_data, aes(x=RainToday, y=Temp9am, fill=RainToday)) +  
  geom_boxplot() +  
  stat_summary(fun=mean, geom="point", shape=20, size=5, color="red", fill="red") +  
  facet_wrap(~Cloud9am, scale="free") +  
  scale_fill_brewer(palette="RdBu")
```

Figure 43: Code of Analysis 13

This analysis is using multiple box plots that grouped by the Cloud9am to show the relationship among cloud, temperature and rain. 1st quartile, median, and the 3<sup>rd</sup> quartile of temperature will be shown by the box plot, while the mean of temperature will be shown with a red dot.

Output:

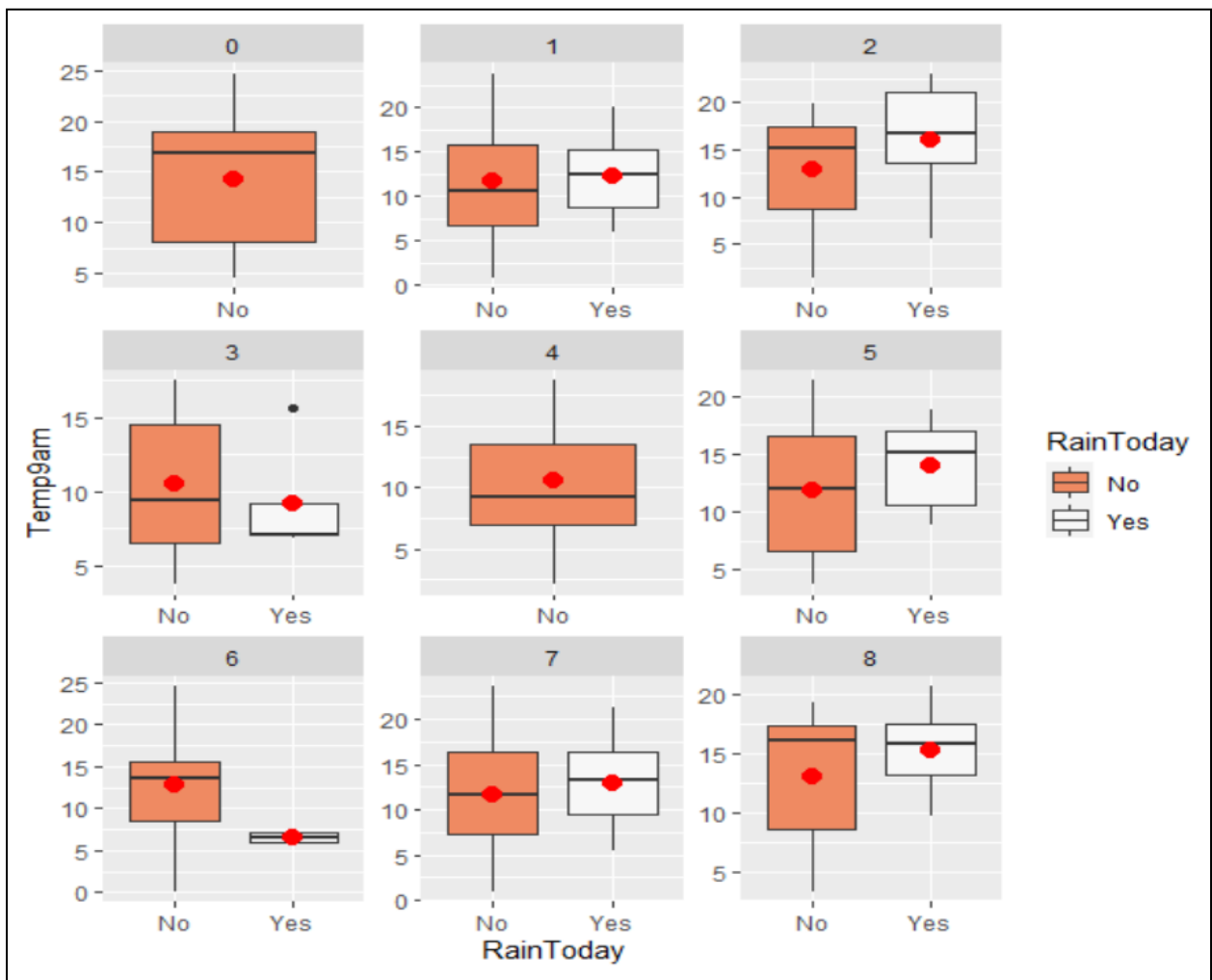


Figure 44: Data Visualization of Analysis 13

From the result of analysis 3, we know that there will be no raining when the cloud is 0okta or 4oktas. In this data visualization, for the 2oktas cloud, the median and mean of temperature at 9am are the highest compared to other cloud status. The lowest median and mean of temperature at 9am are shown in the 6oktas cloud during raining day.

In this data analysis, it shows inconsistence of temperature at 9am based on the rain status grouped by cloud at 9am. At specific cloud status, temperature during non-raining days will be higher than temperature during raining days. However, it will be exactly opposite point of view when observing other cloud status.

#### 2.3.4 Analysis 14: Find the relationship among Temp9am, Temp3pm, average\_temp

In this analysis, a 3D box chart will be plotted to show the temperature status. It reuses the average temperature of minimum and maximum temperature from analysis 11.

Code:

```
Temp_Data <- subset(weather_data, select = c(Temp9am,Temp3pm,RainToday))
mycolors <- c('royalblue1', 'darkcyan')
Temp_Data$color <- mycolors[ as.numeric(Temp_Data$RainToday) ]

plot3d(
  x=Temp_Data$Temp9am, y=Temp_Data$Temp3pm, z=average_temp,
  col = Temp_Data$color,
  type = 's',
  radius = 1,
  xlab="Temp9am", ylab="Temp3pm", zlab="Avg Temp")

bgplot3d({
  plot.new()
  title(main = '3D Diagram for Temperature', line = 3)
})
```

Figure 45: Code of Analysis 14

There will be two types of points in the 3D diagram. Royal blue colour will represent the non-raining day, while dark cyan colour will represent the raining day. X-axis is the temperature at 9am, y-axis is the temperature at 3pm, and the z-axis is the average of minimum and maximum temperature. The 3D box diagram is plotted using plot3d command, while title is given for the diagram using bgplot3d command.



Output:

### 3D Diagram for Temperature

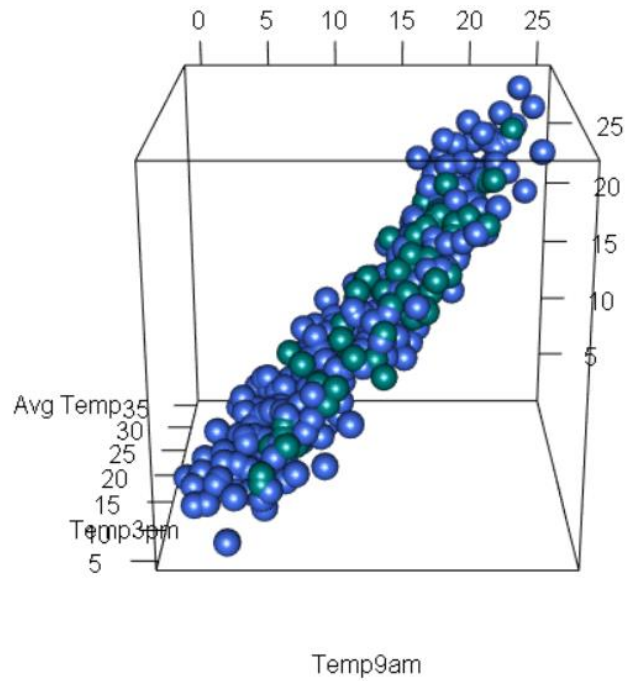


Figure 46: Data Visualization of Analysis 14

### 3D Diagram for Temperature

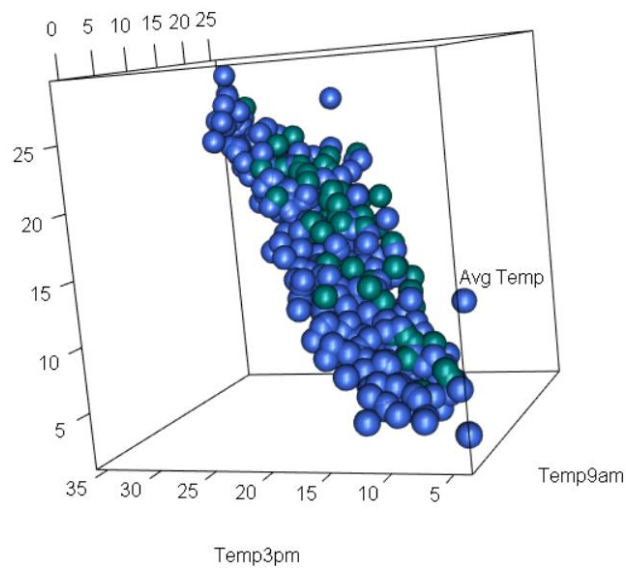


Figure 47: Adjusted Data Visualization of Analysis 14

Figure 46 is the original 3D box plot, while figure 47 is the adjusted 3D box plot as it will provide a clearer vision for the x-axis, y-axis, and z-axis. There is an obvious distinction between the number of raining and non-raining days. The number of royal blue colour points are more than the number of dark cyan colour points.

Besides, the average temperature and temperature at 9am and 3pm are strongly positive correlated with each other.

## 2.4 Question 3: How to avoid storm in the U.S.?

The number of tropical storms in a year keeps increasing years by years. A case-study approach was adopted to show that average number of tropical storms from the year 1966 to the year 2009 is 11 times per year. However, in recent year, from the year 2000 to the year 2014, the average number has been increased to 15 tropical storms per year (Hurricanes and Climate Change | Center for Climate and Energy Solutions, 2020). The diagram below shows the number of storms bases on the year:

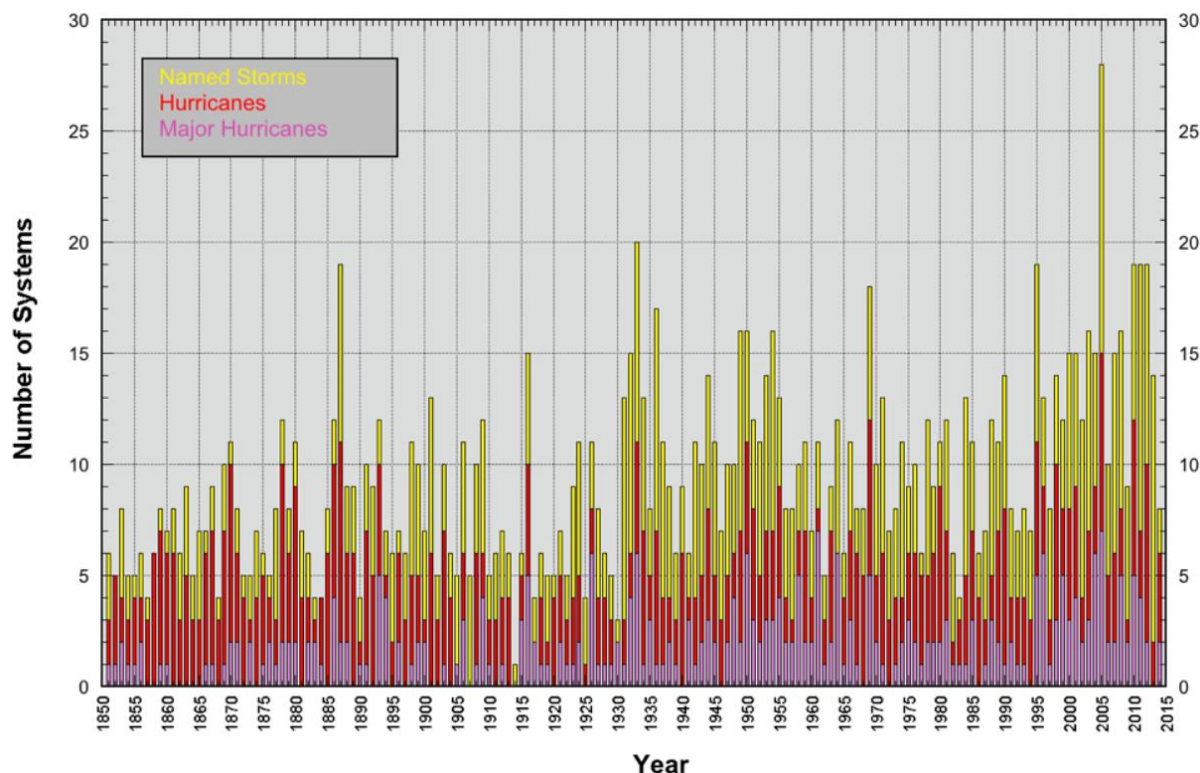


Figure 48: Number of different types of storm per year from 1850 to 2015  
(Hurricanes and Climate Change | Center for Climate and Energy Solutions, 2020)

The main reason for the increasing number of storms is the problem of global warming. As global temperature increased annually, the number and strength of the storm also increased (Bob Berwyn, 2020). Hence, it is a must to include the research of predicting the storm and its impacts to reduce the damage taken by the storms.

### 2.4.1 Analysis 15: Find the status of wind for the whole year.

The first analysis on this question was designed to find out the frequency and strength of wind speed according to its direction.

Code:

```
wind_data <- subset(weather_data, select = c(WindGustDir, WindGustSpeed, WindDir9am,
                                             WindSpeed9am, WindDir3pm, WindSpeed3pm))
class(wind_data$WindGustDir)
dir <- setNames( seq(0, 337.5 , by = 22.5),
                c("N", "NNE", "NE", "ENE", "E", "ESE", "SE", "SSE",
                  "S", "SSW", "SW", "WSW", "W", "WNW", "NW", "NNW"))
wind_data$WindGustDir = dir[as.character(wind_data$WindGustDir)]
wind_data$WindDir9am = dir[as.character(wind_data$WindDir9am)]
wind_data$WindDir3pm = dir[as.character(wind_data$WindDir3pm)]
class(wind_data$WindGustDir)

wind_analysis <- function(WindSpeed, WindDir){
  return(windRose(wind_data, WindSpeed, WindDir,
                  breaks = c(0,5,10,15,20,25,30,35,40,100),
                  angle = 22.5 ,
                  auto.text = FALSE,
                  paddle = FALSE,
                  annotate = FALSE,
                  grid.line = 5,
                  key = list(labels = c("0 - 5",
                                         "5 - 10",
                                         "10 - 15",
                                         "15 - 20",
                                         "20 - 25",
                                         "25 - 30",
                                         "30 - 35",
                                         "35 - 40",
                                         ">40")),
                  key.footer = "WSP (km/h)",
                  key.position = "right",
                  par.settings = list(axis.line = list(col = "lightgray"),
                                      col = c("#4f4f4f", "#0a7cb9", "#f9be00", "#ff7f2f", "#d7153a"))))
}
wind_analysis("WindGustSpeed", "WindGustDir")
wind_analysis("WindSpeed9am", "WindDir9am")
wind_analysis("WindSpeed3pm", "WindDir3pm")
```

Figure 49: Code of Analysis 15

Before plotting the diagram, wind data such as WinGustDir, WindGustSpeed, WindDir9am, WindSpeed9am, WindDir3pm, and WindSpeed3pm are taken from the original dataset and stored in a new data frame, called “wind\_data”. As the values stored in the wind direction columns are factors, it is necessary to change the wind direction variables to numeric data according to their directions. It is because the windRose function only accepts the numeric values of wind direction as input.

As there are three types of wind status graphs have to be plotted, wind analysis has been stored as a function with wind speed and wind direction parameters. It will reduce the number of duplicated codes to enhance the quality of R programming. The windRose function will be explained in the extra feature part.

Output:

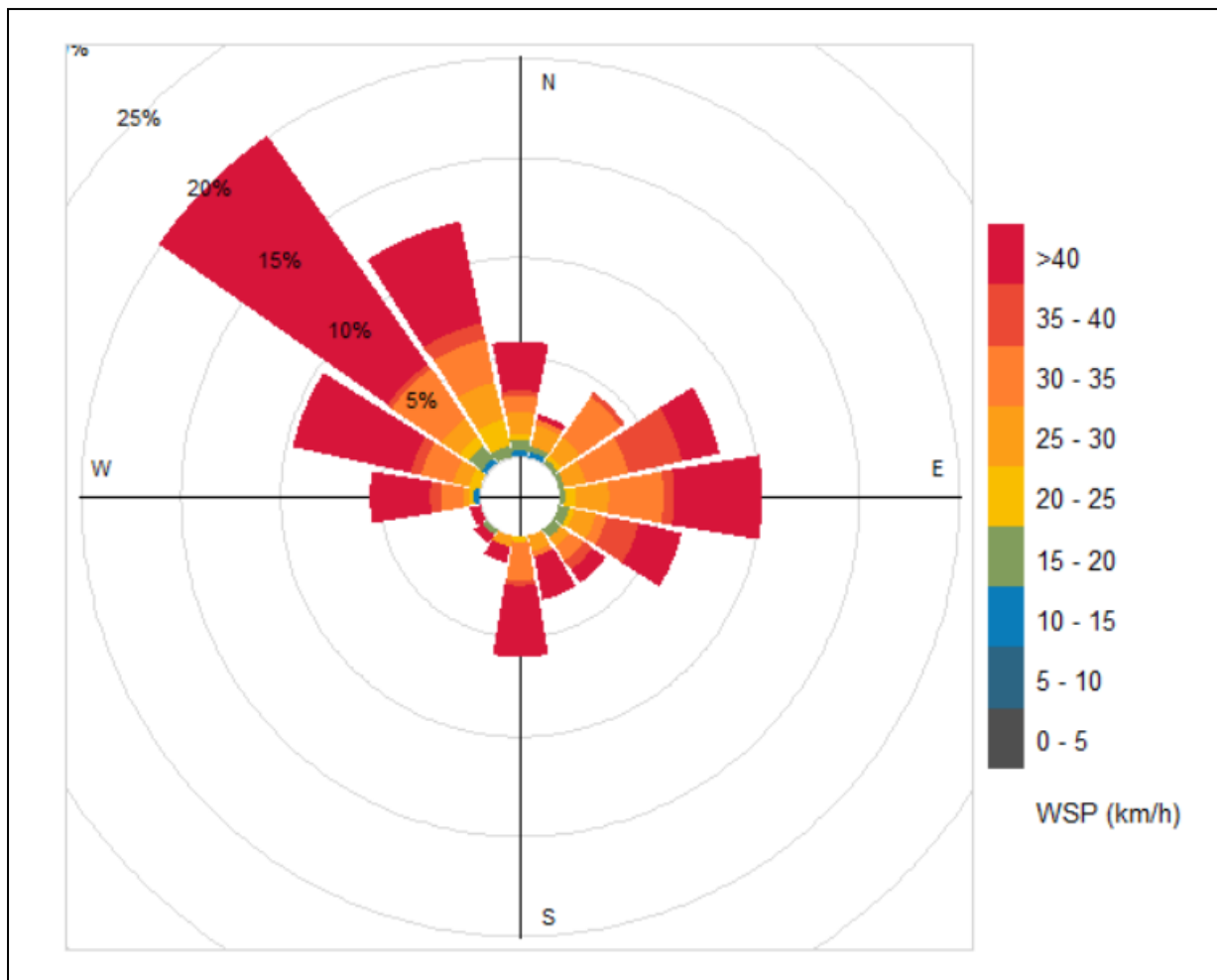


Figure 50: Wind Status of Wind Gust

The diagram above shows the status of wind gust in U.S. in the year 2020. Most of the wind gust is over 40km/h, and most of the wind is at the direction of “NW” which possesses 20% of the wind gust. Least of the wind gust blows at the direction of “WS”, “WSW”, and “SSW”.

Wind gust is a sudden and brief increase of the speed of the wind. At the ground, wind gust is caused by three mechanisms such as turbulence of air friction, wind shear, and heat of solar energy. It will affect the wind to change its speed in a sudden. Usually, the wind gust will remain less than 20 seconds (Stephanie Sigafos, 2012).

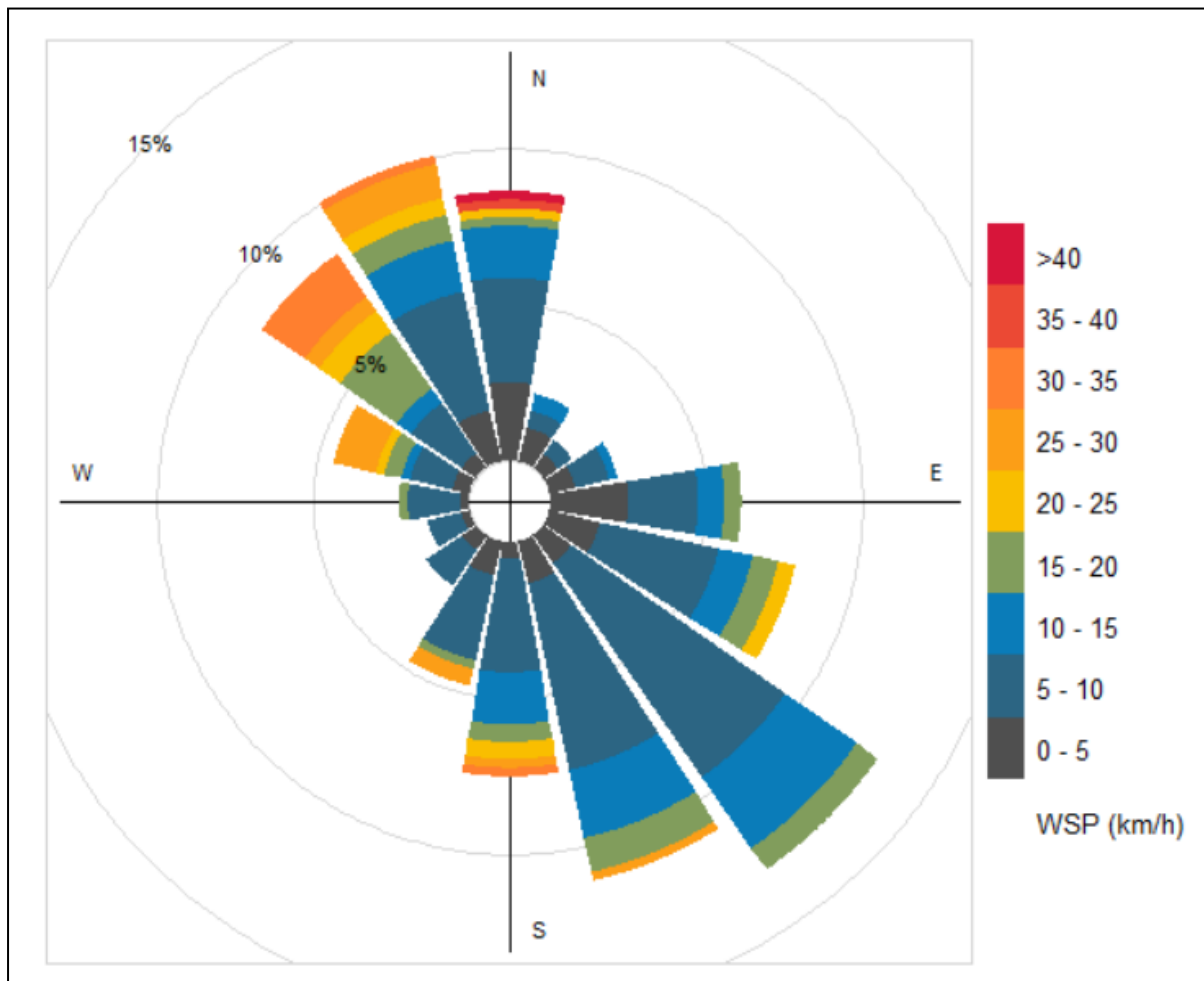


Figure 51: Wind Status of Wind at 9am

The strongest wind speed recorded at 9am is from the “N” wind direction, which is over 40 km/h. However, most of the wind speed is in the range between 5km/h and 10km/h, which indicates light breeze. The following diagram will show the types of wind and its wind speed:

Beaufort Number	Wind Speed (miles/hour)	Wind Speed (km/hour)	Wind Speed (knots)	Description	Wind Effects on Land
0	<1	<1	<1	Calm	Calm. Smoke rises vertically.
1	1-3	1-5	1-3	Light Air	Wind motion visible in smoke.
2	4-7	6-11	4-6	Light Breeze	Wind felt on exposed skin. Leaves rustle.
3	8-12	12-19	7-12	Gentle Breeze	Leaves and smaller twigs in constant motion.
4	13-18	20-28	11-16	Moderate Breeze	Dust and loose paper are raised. Small branches begin to move.
5	19-24	29-38	17-21	Fresh Breeze	Small trees begin to sway.
6	25-31	39-49	22-27	Strong Breeze	Large branches are in motion. Whistling is heard in overhead wires. Umbrella use is difficult.
7	32-38	50-61	28-33	Near Gale	Whole trees in motion. Some difficulty experienced walking into the wind.
8	39-46	62-74	34-40	Gale	Twigs and small branches break from trees. Cars veer on road.
9	47-54	75-88	41-47	Strong Gale	Larger branches break from trees. Light structural damage.
10	55-63	89-102	48-55	Storm	Trees broken and uprooted. Considerable structural damage.
11	64-72	103-117	56-63	Violent Storm	Widespread damage to structures and vegetation.
12	> 73	> 117	> 64	Hurricane	Considerable and widespread damage to structures and vegetation. Violence.

Figure 52: Types of wind according to its speed (Beaufort Scale on Wind Speeds, 2019)

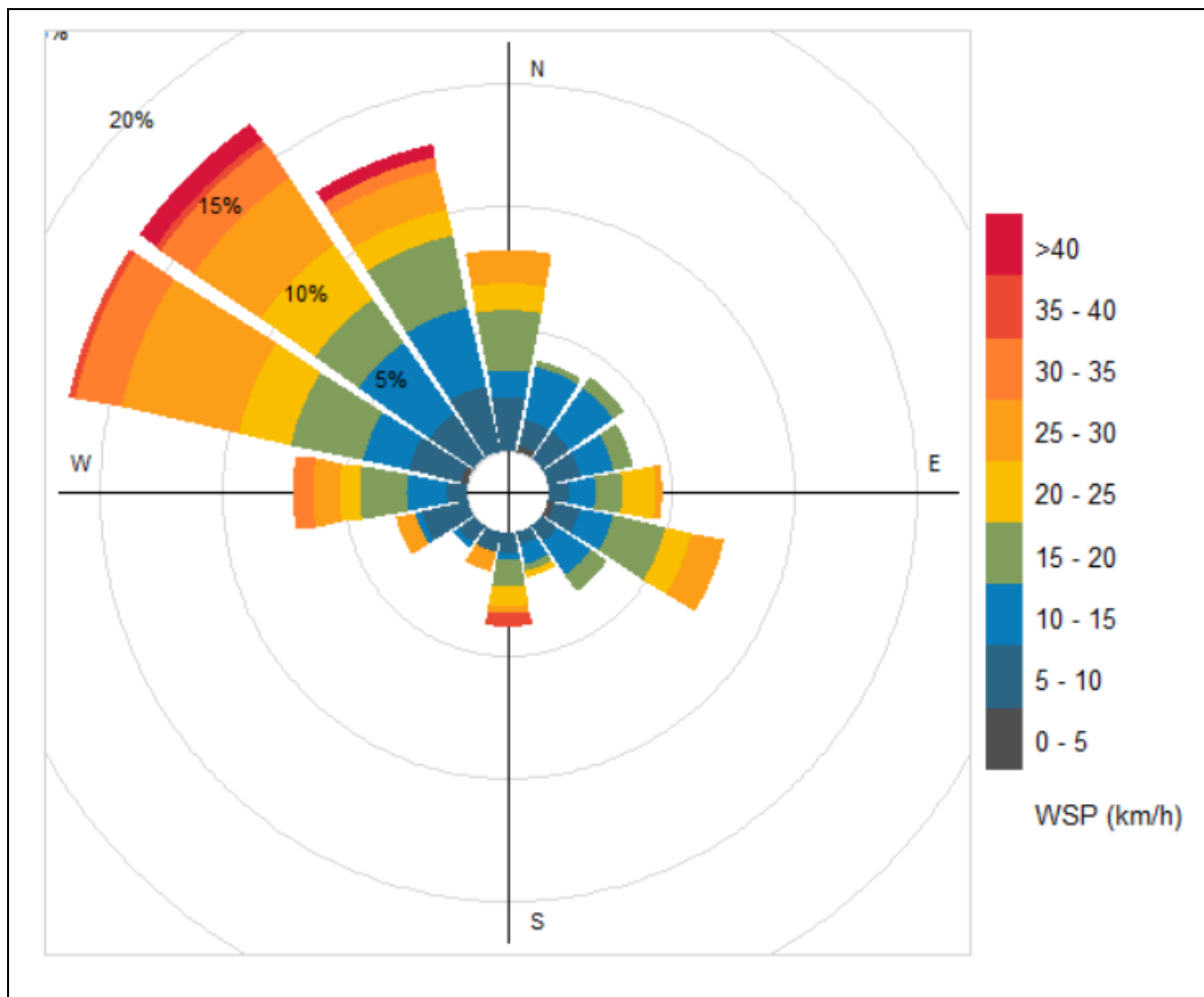


Figure 53: Wind Status of Wind at 3pm

From the plotted diagram above, “NW”, “NNW” and “WNW” directions will receive more frequent wind compared to other directions. Wind speed at 3pm is stronger than the wind at 9am. Most of the time, the wind speed at 3pm is over 15km/h, which will cause leaves and smaller twigs start moving. The frequency of the wind at 3pm over 40km/h is also higher than the wind at 9am. This wind is considered as strong breeze that will cause the passerby difficult to hold the umbrella and the larger branches of a tree will continue shaking.

In short, most of the time, U.S. residents will experience calmer wind instead of frequent and strong wind in the year 2020. They will just encounter a sudden and brief wind gust that will not longer than 20 seconds.



### 2.4.2 Analysis 16: Find the relationship of wind direction and wind speed at 3pm.

After found out the general status of the wind, it is necessary to investigate the wind in a more detail in the following analysis.

Code:

```
wind_data1 <- subset(weather_data, select = c(WindDir3pm, WindSpeed3pm))
summary(wind_data1)
wind_data1 <- na.omit(wind_data1)

ggplot(wind_data1, aes(x=WindDir3pm, y= WindSpeed3pm)) +
  geom_violin(alpha=0.5, color="gray") +
  geom_jitter(alpha=0.5, aes(color=WindDir3pm), position = position_jitter(width = 0.1)) +
  coord_flip()
```

Figure 54: Code of Analysis 16

At the data pre-processing stages that made before the data analysis, it shows a NA value in the WindDir3pm. Hence, the deletion of that NA value has to be done to enhance the quality of the data analysis.

The technique that used to plot the following diagram is violin chart. The detail of violin chart will be discussed at the extra feature part.

Output:

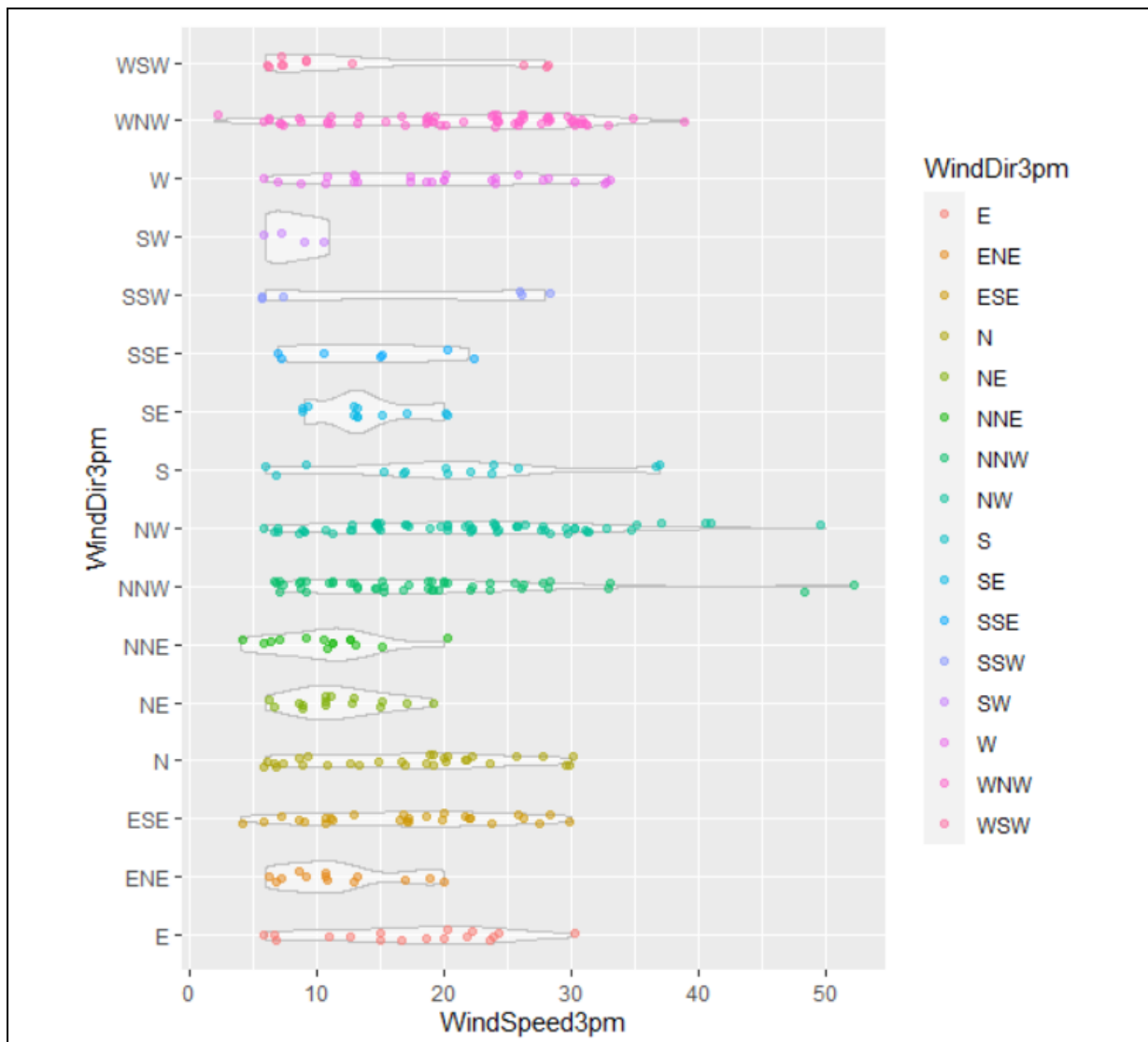


Figure 55: Data Visualization of Analysis 16

There is only a wind that over 50km/h at the direction of “NNW”. Most of the wind remains the speed between 5km/h and 30km/h. The frequency of wind that less than 5km/h is lesser than the frequency of wind over 30km/h.

## 2.5 Others

### 2.5.1 Analysis 17: Correlation relationship among the weather data.

Code:

```
new_data <- select(weather_data, -c("Date", "Month", "WindGustDir", "WindDir9am",  
                                   "WindDir3pm", "RainToday", "RainTomorrow"))  
summary(new_data)  
new_data <- na.omit(new_data)  
  
factor_data <- names(which(sapply(new_data, class) == "factor"))  
numeric_data <- setdiff(colnames(new_data), factor_data)  
numeric_data <- setdiff(numeric_data, "RainTomorrow")  
numeric_data_mat <- as.matrix(new_data[, numeric_data, drop=FALSE])  
numeric_data_cor <- cor(numeric_data_mat)  
corrplot(numeric_data_cor)
```

Figure 56: Code of Analysis 17

Output:

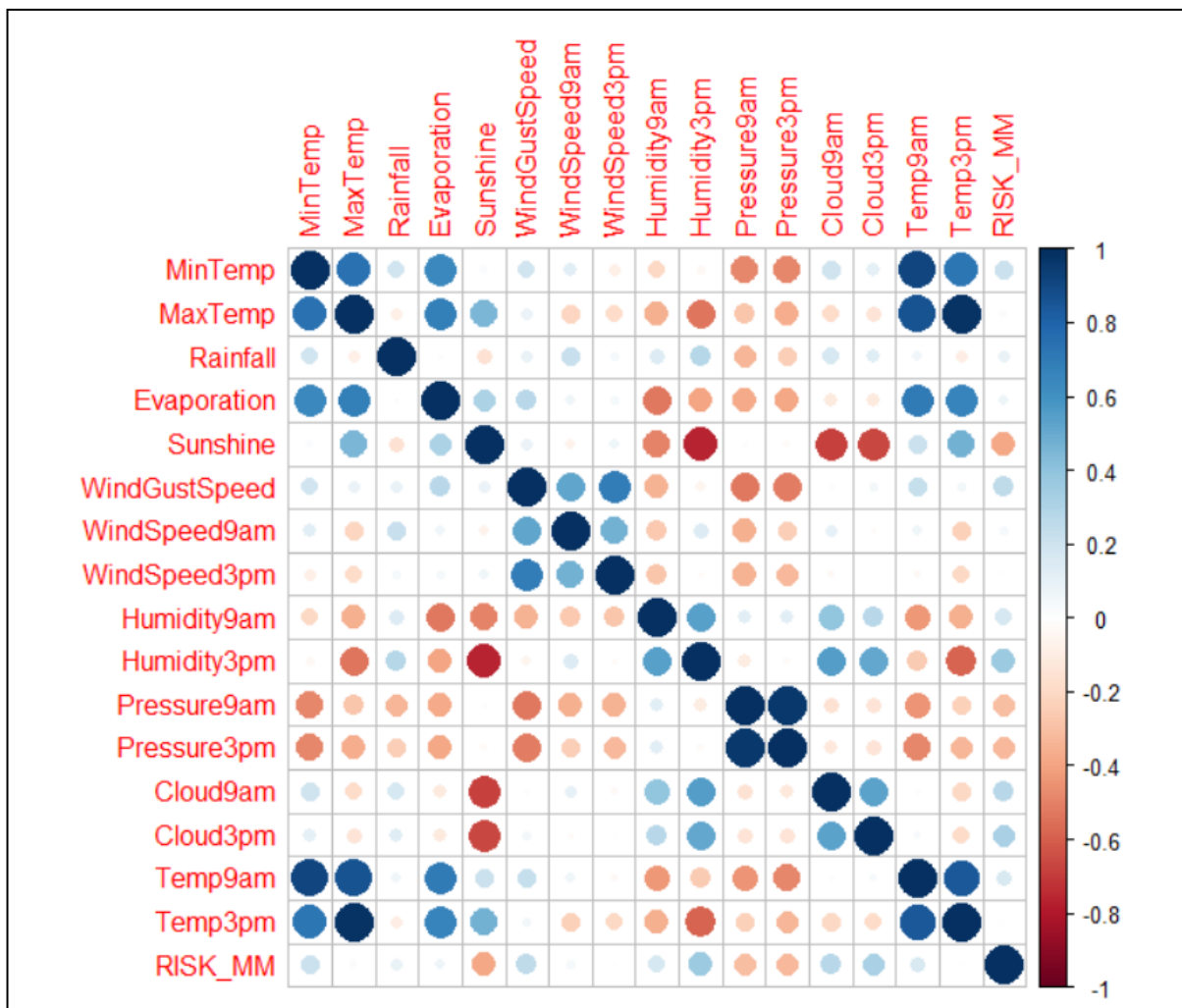


Figure 57: Data Visualization of Analysis 17

This analysis provides a data visualization for the relationship among the variables in the weather data set. The used data in this analysis includes only continuous variables instead of categorical variables. The results of this analysis show:

- Evaporation is strongly positive correlated with temperature.
- Sunshine is strongly negative correlated with humidity and cloud.
- Humidity is strongly positive correlated with cloud.
- Wind speed is moderately negative correlated with pressure.
- Pressure is moderately negative correlated with temperature.
- RISK\_MM is slightly positive correlated with humidity and cloud.

### 2.5.2 Analysis 18: Find the density of each weather data by entering the choice.

Code:

```
op1 <- ggplot(weather_data, aes(Sunshine, group=RainTomorrow, fill=RainTomorrow)) +  
  geom_density(adjust=1.5, alpha=.4) +  
  theme_ipsum()  
op2 <- ggplot(weather_data, aes(Temp9am, group=RainTomorrow, fill=RainTomorrow)) +  
  geom_density(adjust=1.5, alpha=.4) +  
  theme_ipsum()  
op3 <- ggplot(weather_data, aes(Cloud9am, group=RainTomorrow, fill=RainTomorrow)) +  
  geom_density(adjust=1.5, alpha=.4) +  
  theme_ipsum()  
op4 <- ggplot(weather_data, aes(Pressure9am, group=RainTomorrow, fill=RainTomorrow)) +  
  geom_density(adjust=1.5, alpha=.4) +  
  theme_ipsum()  
op5 <- ggplot(weather_data, aes(Humidity9am, group=RainTomorrow, fill=RainTomorrow)) +  
  geom_density(adjust=1.5, alpha=.4) +  
  theme_ipsum()  
  
input <- function(){  
  message("Density:\n(A)Sunshine\n(B)Temp9am\n(C)Cloud9am\n(D)Pressure9am\n(E)Humidity9am".  
  option = (readline(prompt = "Option: "))  
  result = switch(option,  
    A = op1,  
    B = op2,  
    C = op3,  
    D = op4,  
    E = op5,  
    "Error")  
  if(result!="Error"){  
    print(result)  
  }else{  
    print("Invalid Option")  
  }  
}  
input()
```

Figure 58: Code of Analysis 18

Output:

```
Density:  
(A)Sunshine  
(B)Temp9am  
(C)Cloud9am  
(D)Pressure9am  
(E)Humidity9am  
Option:
```

Figure 59: Output of Analysis 18

The output of analysis 18 is required user input to choose among “A”, “B”, “C”, “D” or “E”. Each of the choice will lead to different data visualization that shows the density of the variables.

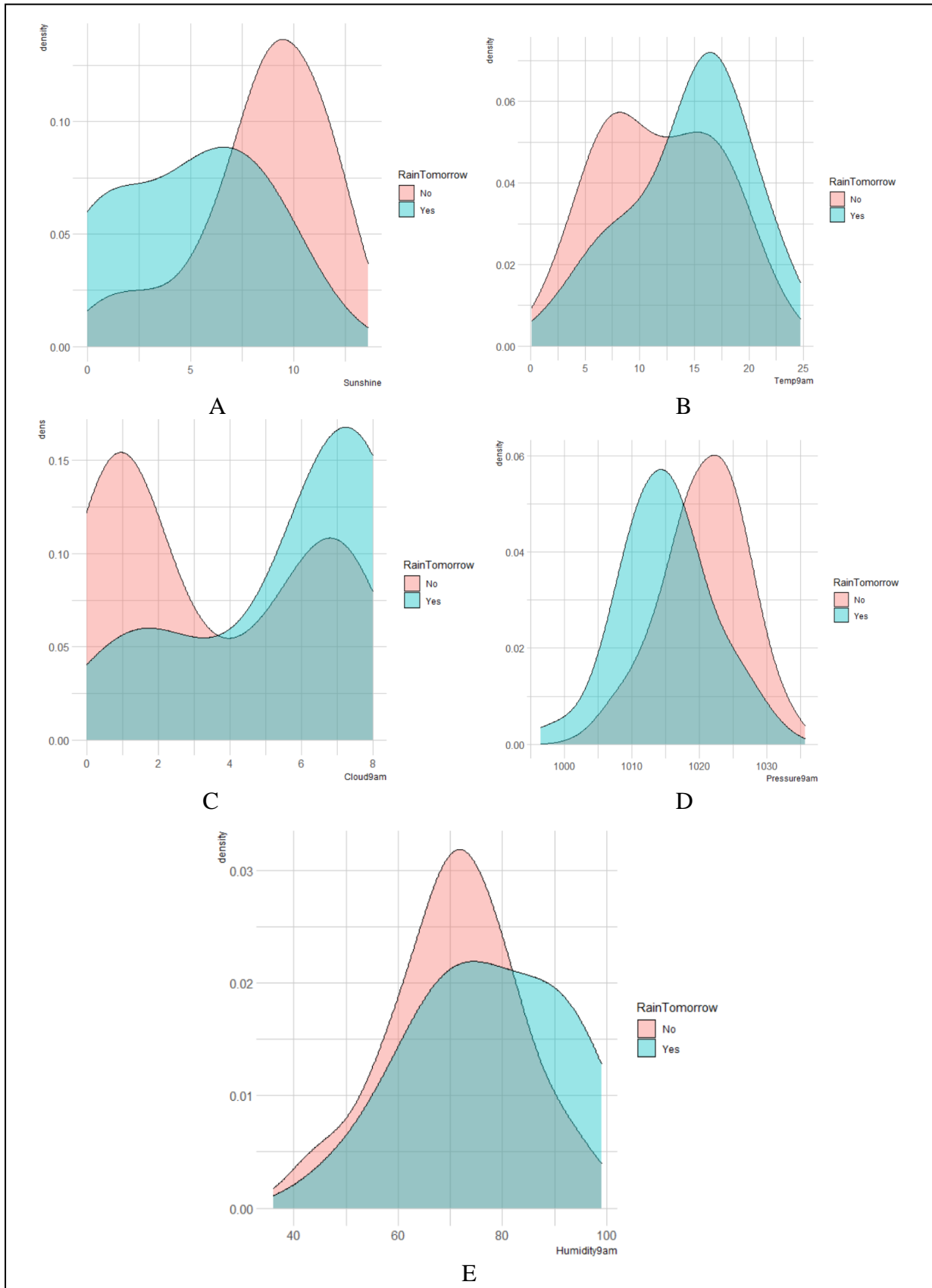


Figure 60: Combination of Data Visualization of Analysis 18.  
 (A) Sunshine. (B) Temp9am. (C) Cloud9am. (D) Pressure9am. (E) Humidity9am.

The density plot shows a kernel density the represents the distribution of a group of numeric values. For example, figure 60A found that most of the sunshine distribute at around 10 during non-raining day. Figure 60B shows that most of the temperate at 9am distribute between 15 and 17.5 during raining day. Figure 60C displays the distribution of values of cloud at 9am is between 6 and 8 during raining days.

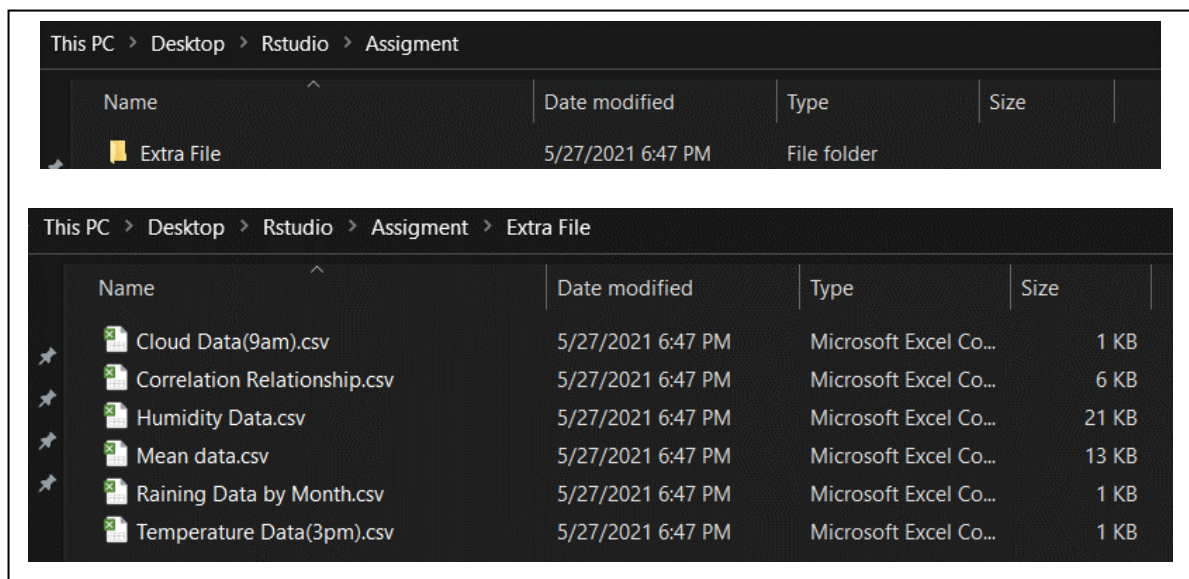
### 2.5.3 Analysis 19: Write new CSV files into new folder.

Code:

```
# Write new CSV file into new folder
dir.create("C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File")
# Data from Analysis 1
Total_rainfall$Month = NULL
Rain_Data <- cbind(Yes_No, Total_rainfall)
write.csv(Rain_Data, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Raining Data by Month.csv")
# Data from Analysis 3
write.csv(Cloud_Data, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Cloud Data(9am).csv")
# Data from Analysis 5
write.csv(humidity_data, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Humidity Data.csv")
# Data from Analysis 10
write.csv(data_mean, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Mean data.csv")
# Data from Analysis 12
write.csv(Hot_Cool, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Temperature Data(3pm).csv")
# Data from Analysis 17
write.csv(numeric_data_cor, "C:\\Users\\MSI\\Desktop\\Rstudio\\Assignment\\Extra File\\Correlation Relationship.csv")
```

Figure 61: Code of Analysis 19

Output:



The screenshot shows two views of a Windows File Explorer window. The top view shows the 'Assignment' folder containing an 'Extra File' folder. The bottom view shows the contents of the 'Extra File' folder, which includes six CSV files: 'Cloud Data(9am).csv', 'Correlation Relationship.csv', 'Humidity Data.csv', 'Mean data.csv', 'Raining Data by Month.csv', and 'Temperature Data(3pm).csv'. All files were modified on 5/27/2021 at 6:47 PM.

This PC > Desktop > Rstudio > Assignment				
Name	Date modified	Type	Size	
Extra File	5/27/2021 6:47 PM	File folder		

This PC > Desktop > Rstudio > Assignment > Extra File				
Name	Date modified	Type	Size	
Cloud Data(9am).csv	5/27/2021 6:47 PM	Microsoft Excel Co...	1 KB	
Correlation Relationship.csv	5/27/2021 6:47 PM	Microsoft Excel Co...	6 KB	
Humidity Data.csv	5/27/2021 6:47 PM	Microsoft Excel Co...	21 KB	
Mean data.csv	5/27/2021 6:47 PM	Microsoft Excel Co...	13 KB	
Raining Data by Month.csv	5/27/2021 6:47 PM	Microsoft Excel Co...	1 KB	
Temperature Data(3pm).csv	5/27/2021 6:47 PM	Microsoft Excel Co...	1 KB	

Figure 62: Result of Analysis 19

This analysis is used to create a new file (dir.create) and stores all the created data frame into several new csv files.



## 3.0 Extra Feature

### 3.1 Bar Diagram with Two Bars

This technique is used in analysis 3. “Geom\_bar” command allows us to plot a bar box. By default, position in “geom\_bar” equals to stack. Hence, multiple variables will be automatically stacked together, like the following diagram:

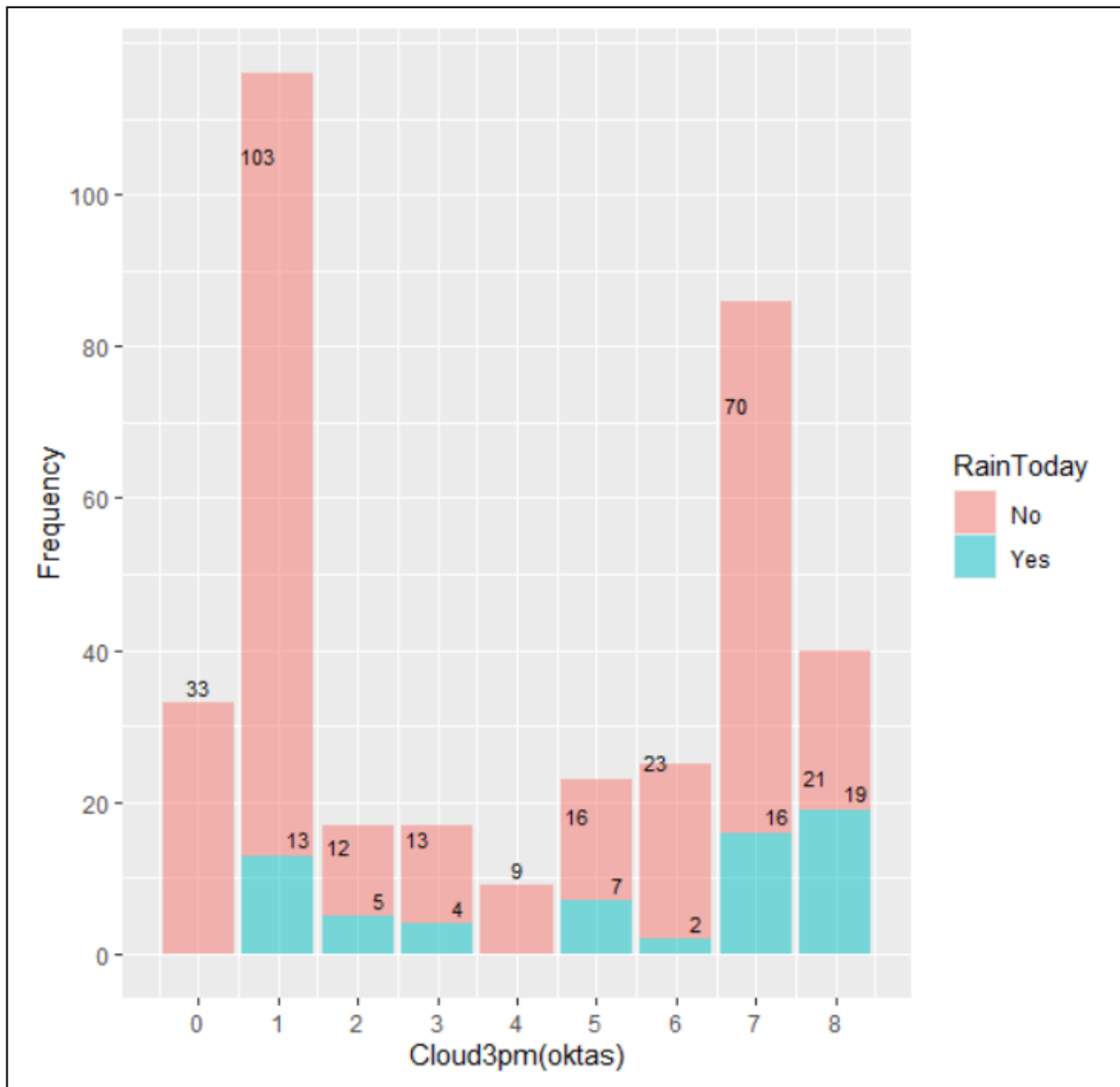


Figure 63: Box Plot before Changing the “Position”

For enhancing the quality of the data visualization, “position='dodge'” in the “geom\_bar” command will allow the values in each bar dodged side-to-side, like the following diagram:

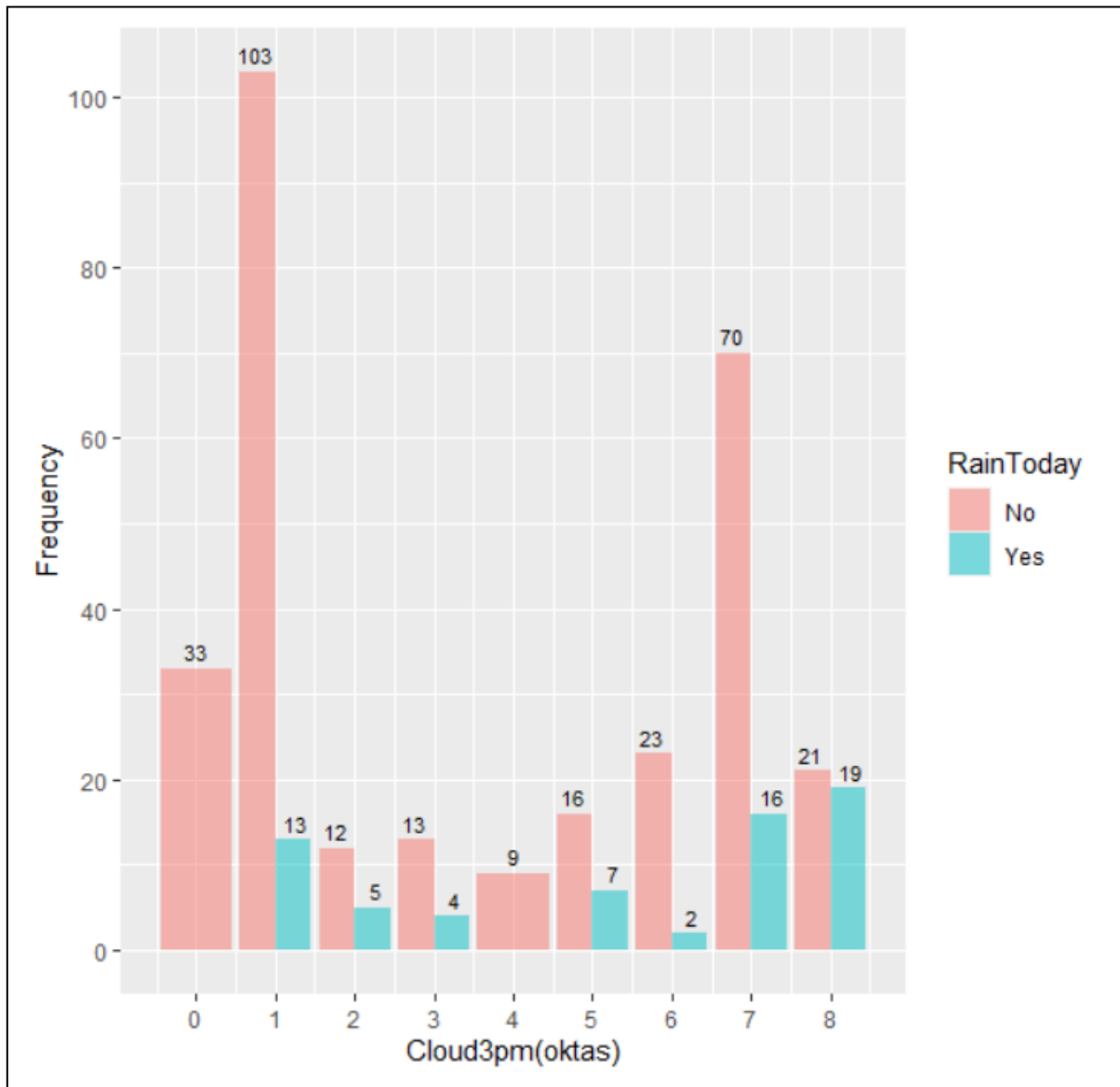


Figure 64: Box Plot after Changing the “Position”

Figure 64 is an advanced version of figure 63 as it could aid the eyes for viewing the bar and also the labels of each bar.

### 3.2 Melt Function

This technique is used in analysis 5 and 10. The melt feature requires the installation of “reshape2” library. Melt will set the data to melt. For example, use of “humidity\_data = melt(humidity\_data)” will be shown as below:

	Humidity9am	Humidity3pm	RainToday
1	68	29	No
2	80	36	Yes
3	82	69	Yes
4	62	56	Yes
5	68	49	Yes
6	70	57	No
7	63	47	No
8	65	57	No
9	70	48	No
10	82	32	Yes

Figure 65: Data Frame before Using Melt Function

	RainToday	variable	value
1	No	Humidity9am	68
2	Yes	Humidity9am	80
3	Yes	Humidity9am	82
4	Yes	Humidity9am	62
5	Yes	Humidity9am	68
6	No	Humidity9am	70
7	No	Humidity9am	63
8	No	Humidity9am	65
9	No	Humidity9am	70
10	Yes	Humidity9am	82

Figure 66: Data Frame after Using Melt Function

In this example, melt command will take “RainToday” as ID and get “Humidity9am” and “Humidity3pm” as variables. And the last column will be the value of the variable.

### 3.3 Donut Chart

This technique is used in analysis 12. Process of plotting a donut diagram:

1. Having a set dataset which stores only numeric variables with entities.
2. Translate the numeric values to proportion.
3. Display the data in a rectangle plot using `geom_rect()`.

**Code:**

```
ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus)) +  
  geom_rect()
```

Figure 67: Code of using `geom_rect()`

**Output:**

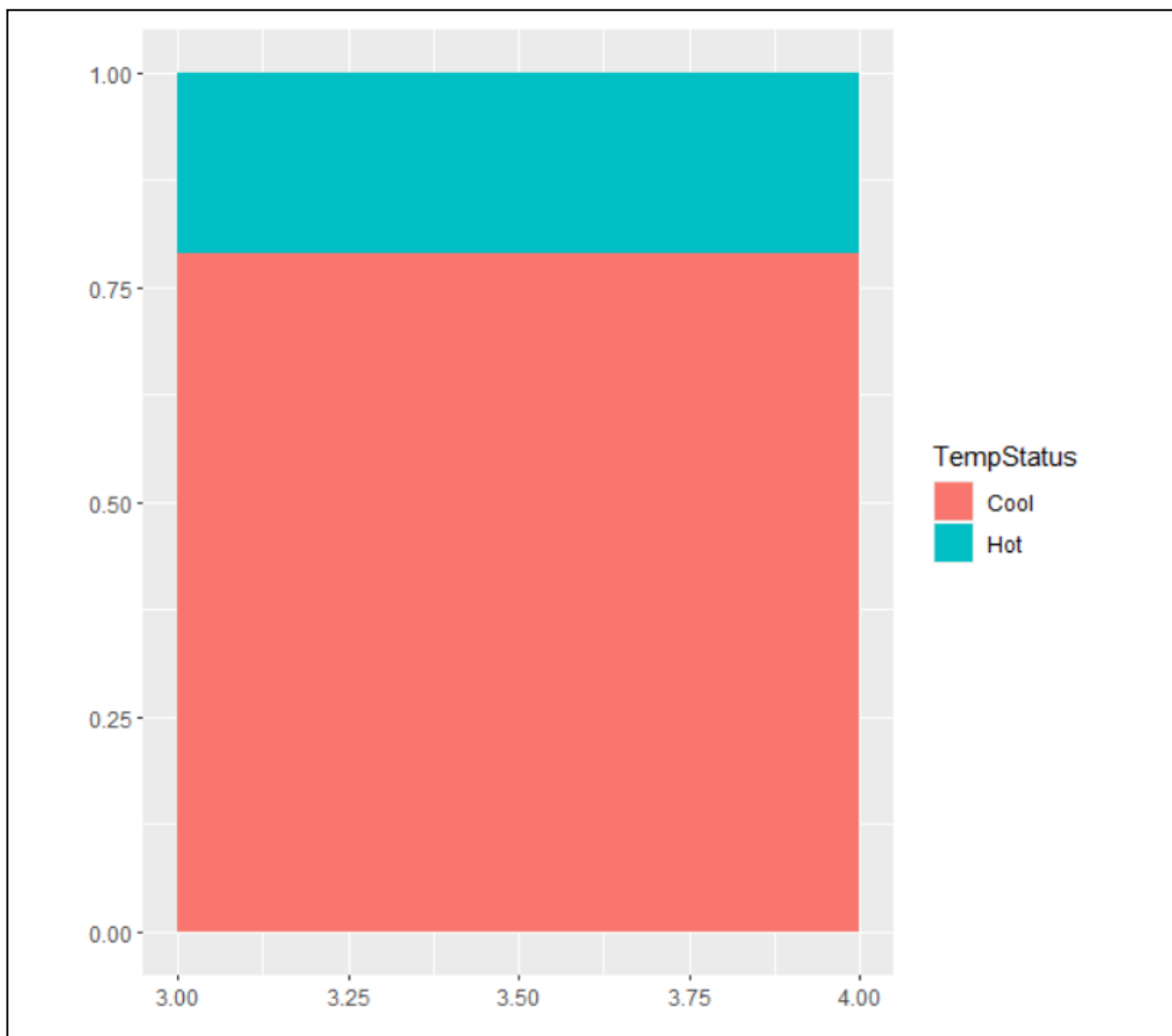


Figure 68: Output of using `geom_rect()`

4. Change the rectangle plot to a pie plot using `coord_polar()`.

**Code:**

```
ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus)) +  
  geom_rect() +  
  coord_polar(theta="y")
```

Figure 69: Code of using `coord_polar()`

**Output:**

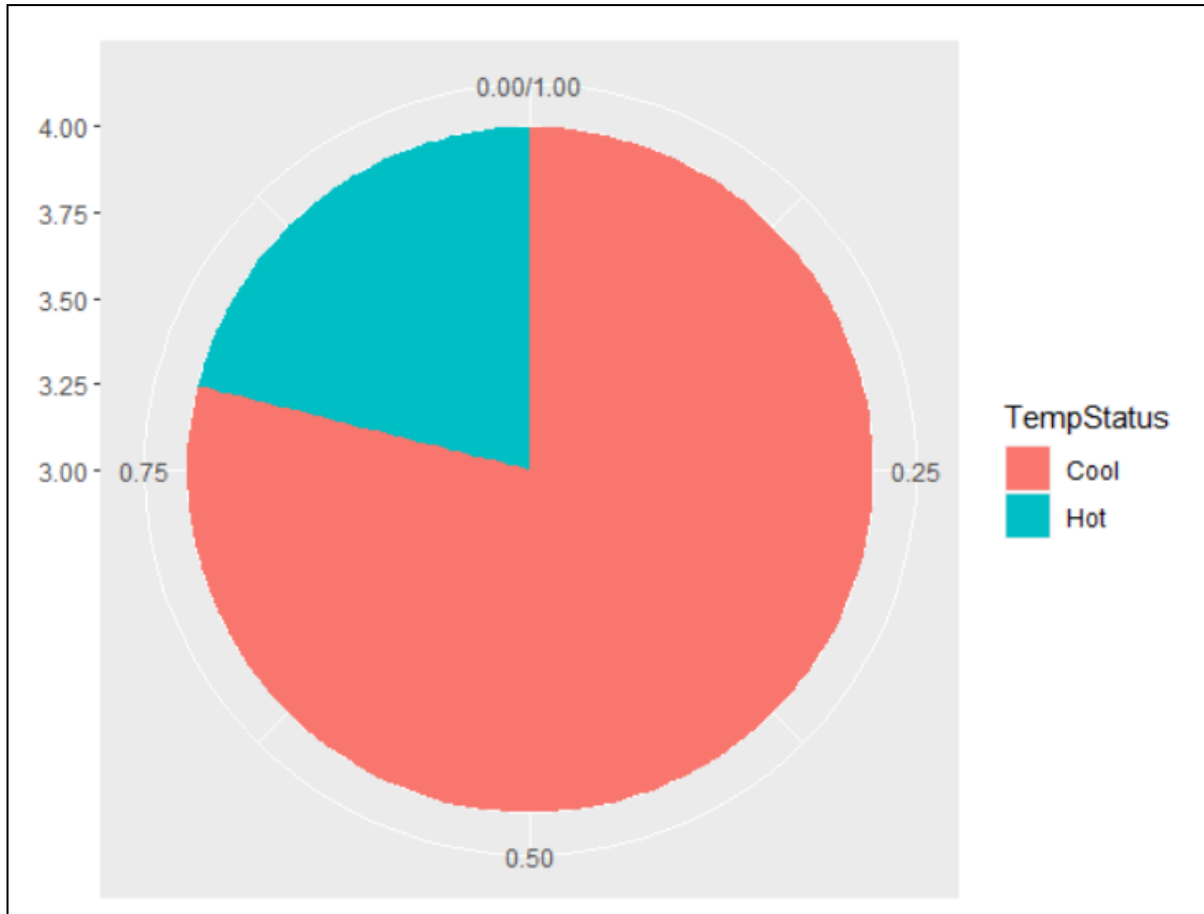


Figure 70: Output of using `coord_polar()`

5. Add empty circle in the middle of the plot using `xlim()` to form a donut chart.

**Code:**

```
ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus)) +  
  geom_rect() +  
  coord_polar(theta="y") +  
  xlim(c(2, 4))
```

Figure 71: Code of using `xlim()`

**Output:**

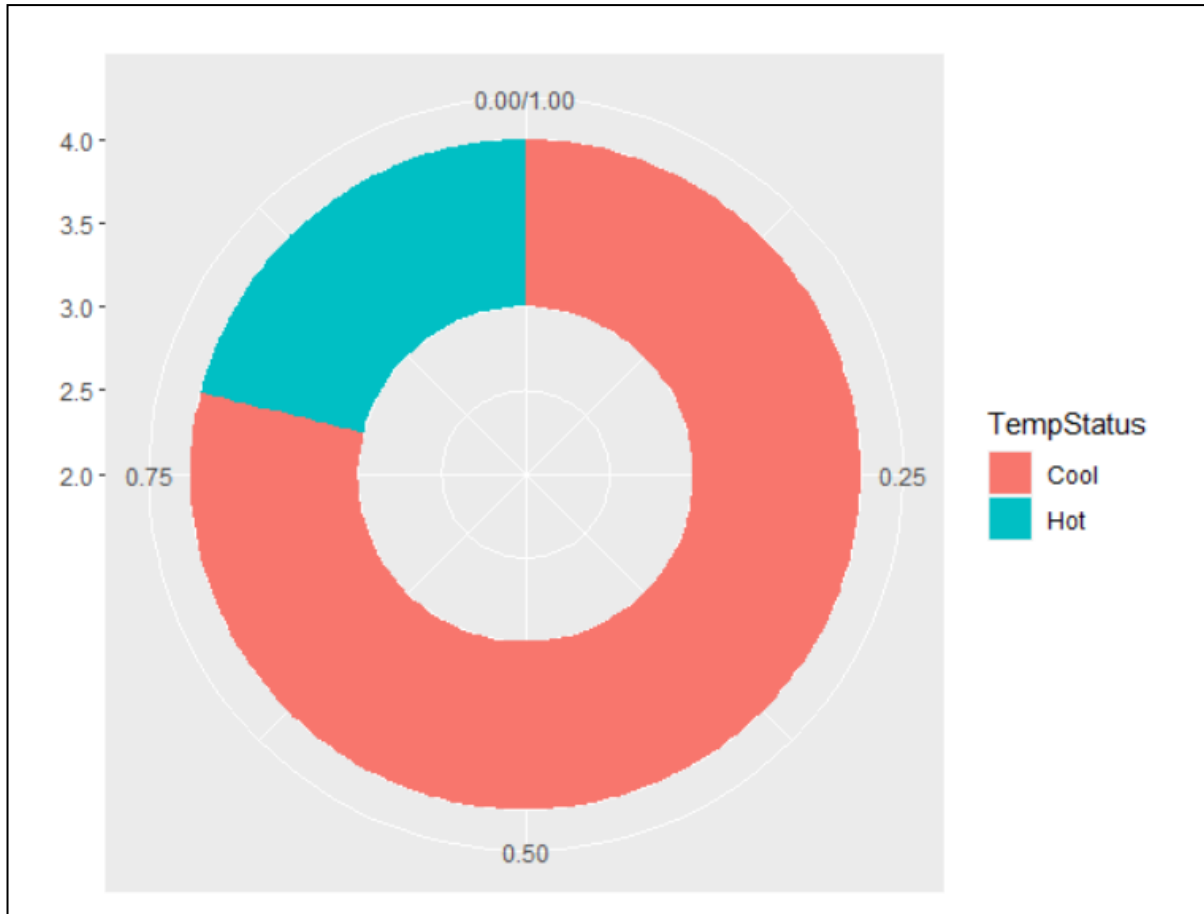


Figure 72: Output of using `xlim()`

6. Remove the unnecessary background, axis, and labels using `theme_void()`.

**Code:**

```
ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus))  
  geom_rect() +  
  coord_polar(theta="y") +  
  xlim(c(2, 4)) +  
  theme_void()
```

Figure 73: Code of using `theme_void()`

**Output:**

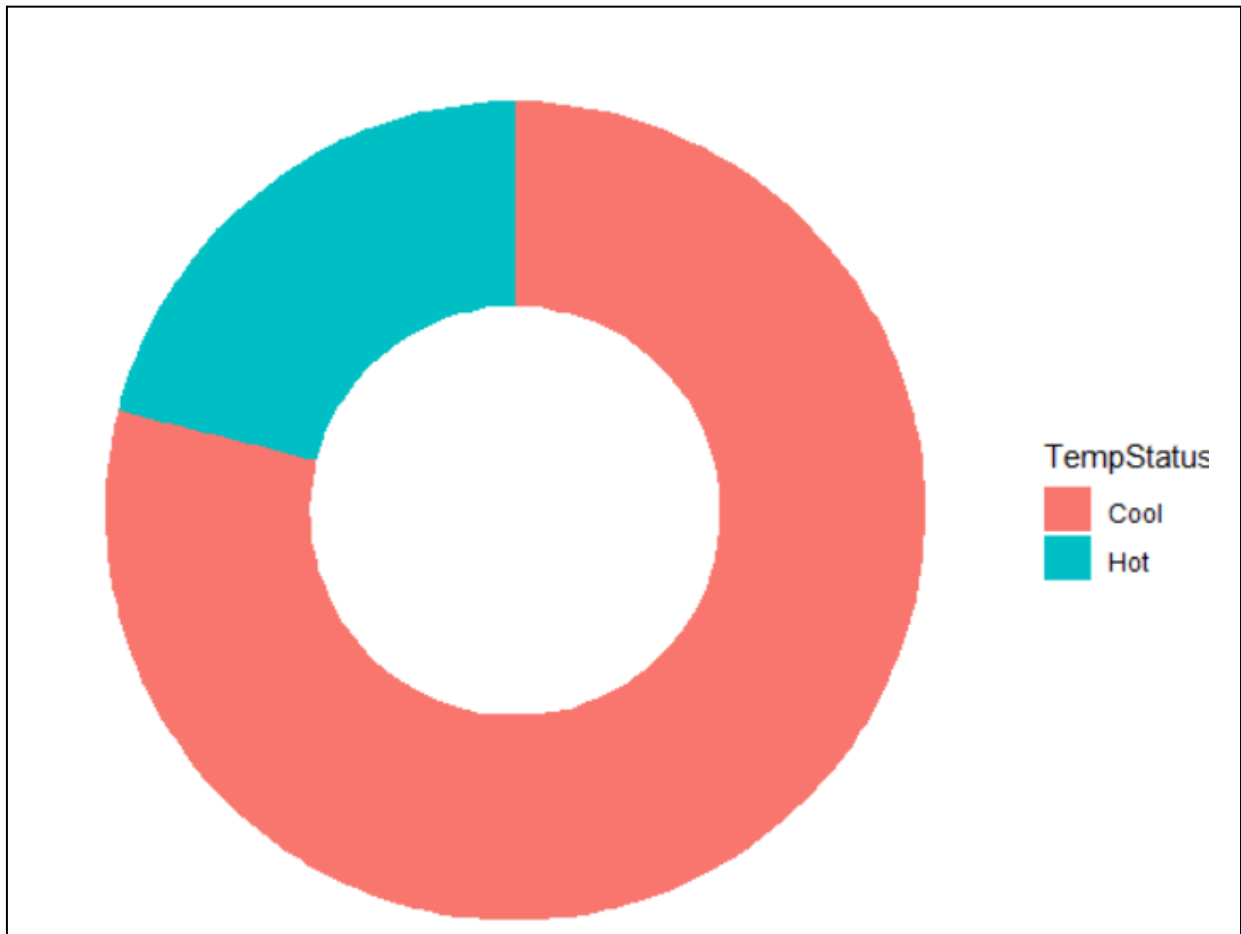


Figure 74: Output of using `theme_void()`

7. Change the colour of donut using `scale_fill_brewer()`. Add labels into the donut using `geom_label()`. Give a title to the donut using `ggtitle()`.

**Code:**

```
ggplot(Hot_Cool, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=TempStatus)) +  
  geom_rect() +  
  coord_polar(theta="y") +  
  xlim(c(2, 4)) +  
  theme_void() +  
  theme(plot.title = element_text(hjust=0.5), legend.position = "none") +  
  geom_label(x=3.5, aes(y=labelPosition, label=label), size=5) +  
  scale_fill_brewer(palette=1) +  
  ggtitle("Number of hot days and cool days")
```

Figure 75: Code of final donut chart

**Output:**

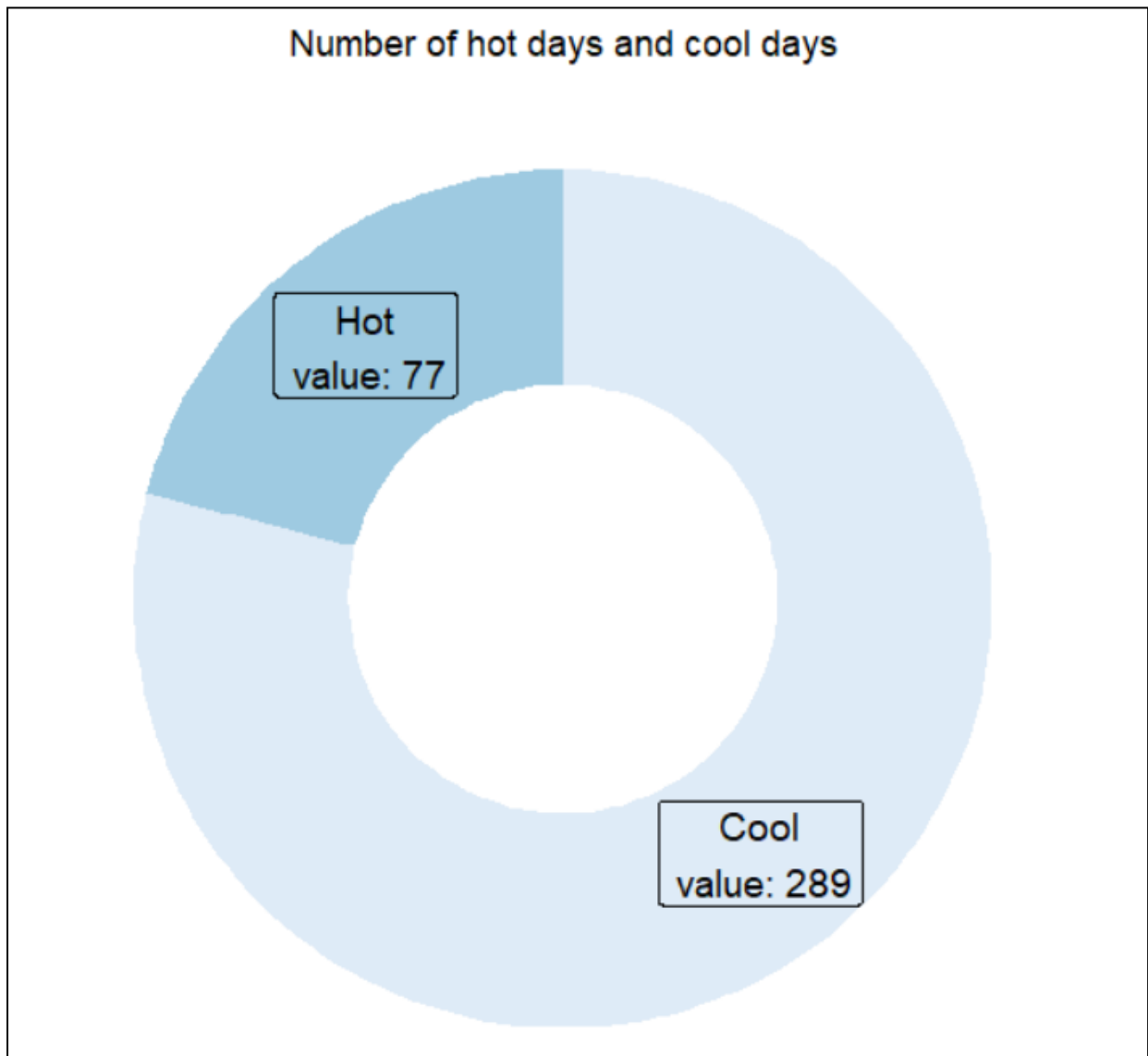


Figure 76: Output of final donut chart



### 3.4 3D Chart with RGL

This technique is used in analysis 14. The 3D box chart needs a library, called “rgl”.  
Process of plotting 3D box plot:

1. Having three different variables from dataset for x-axis, y-axis, and z-axis.
2. Plotting a 3D box diagram without title using plot3d().

#### Code:

```
plot3d(  
  x=Temp_Data$Temp9am, y=Temp_Data$Temp3pm, z=average_temp,  
  col = Temp_Data$color,  
  type = 's',  
  radius = 1,  
  xlab="Temp9am", ylab="Temp3pm", zlab="Avg Temp")
```

Figure 77: Code of Plotting 3D Chart without Title

#### Output:

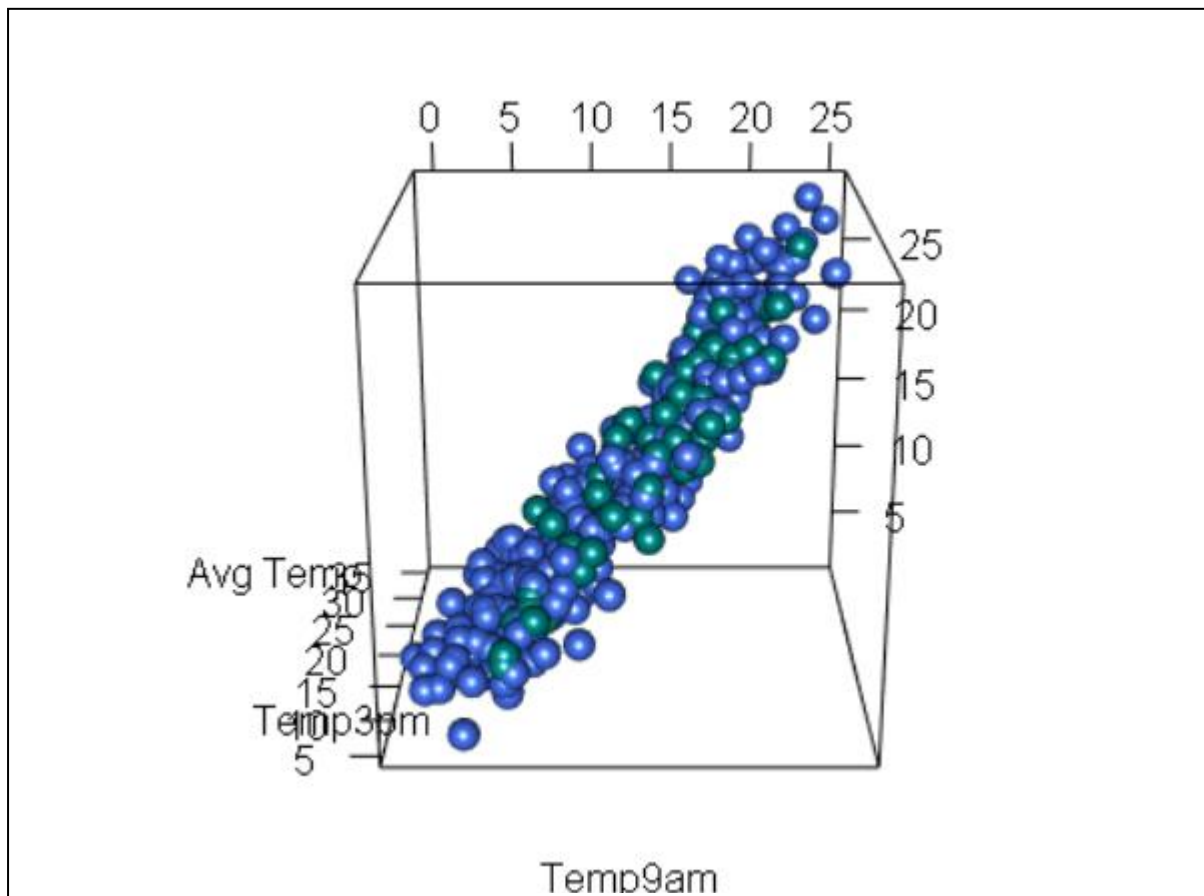


Figure 78: Output of Plotting 3D Chart without Title

3. Assigning title to 3D Chart using `bgplot3d()`.

**Code:**

```
bgplot3d({  
  plot.new()  
  title(main = '3D Diagram for Temperature', line = 3)  
})
```

Figure 79: Code of Plotting 3D Chart with Title

**Output:**

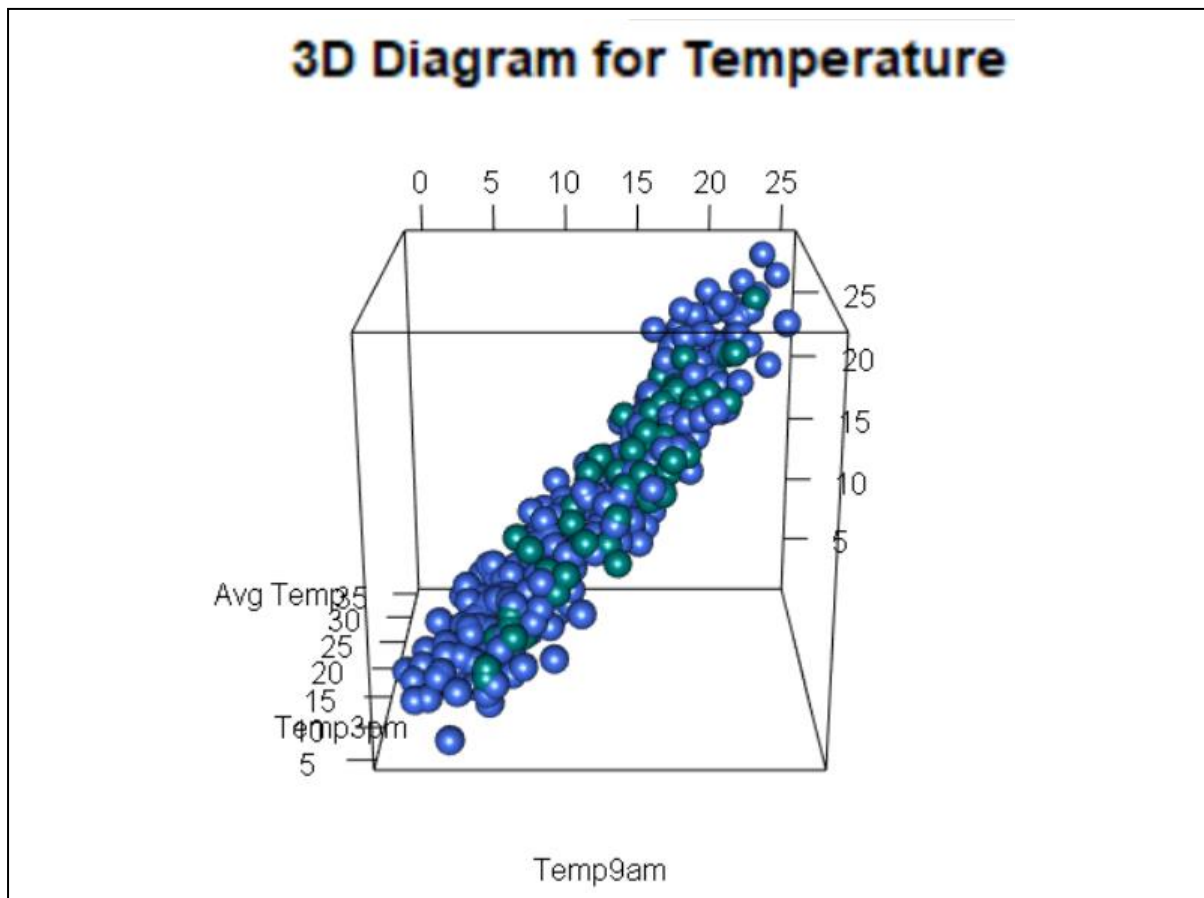


Figure 80: Output of Plotting 3D Chart with Title

### 3.5 Wind Rose

This technique is used in analysis 15. It requires library “openair” which could provide air quality data analysis. The used technique “windRose” is a traditional plot that will show the wind speed and its direction in a circle graph.

#### Code:

```
windRose(wind_data, "windGustSpeed", "windGustDir",
  breaks = c(0,5,10,15,20,25,30,35,40,100),
  angle = 22.5 ,
  auto.text = FALSE,
  paddle = FALSE,
  annotate = FALSE,
  grid.line = 5,
  key = list(labels = c("0 - 5",
                        "5 - 10",
                        "10 - 15",
                        "15 - 20",
                        "20 - 25",
                        "25 - 30",
                        "30 - 35",
                        "35 - 40",
                        ">40")),
  key.footer = "WSP (km/h)",
  key.position = "right",
  par.settings = list(axis.line = list(col = "lightgray")),
  col = c("#4f4f4f", "#0a7cb9", "#f9be00", "#ff7f2f", "#d7153a"))
```

Figure 81: Code of Plotting Wind Rose Diagram

“windRose” command requires the wind speed and wind direction in the same data frame. Wind direction has to be stored in numeric data. The default angle is 30. Besides, default paddle is true and will show wedge style wind rose diagram.

**Output:**

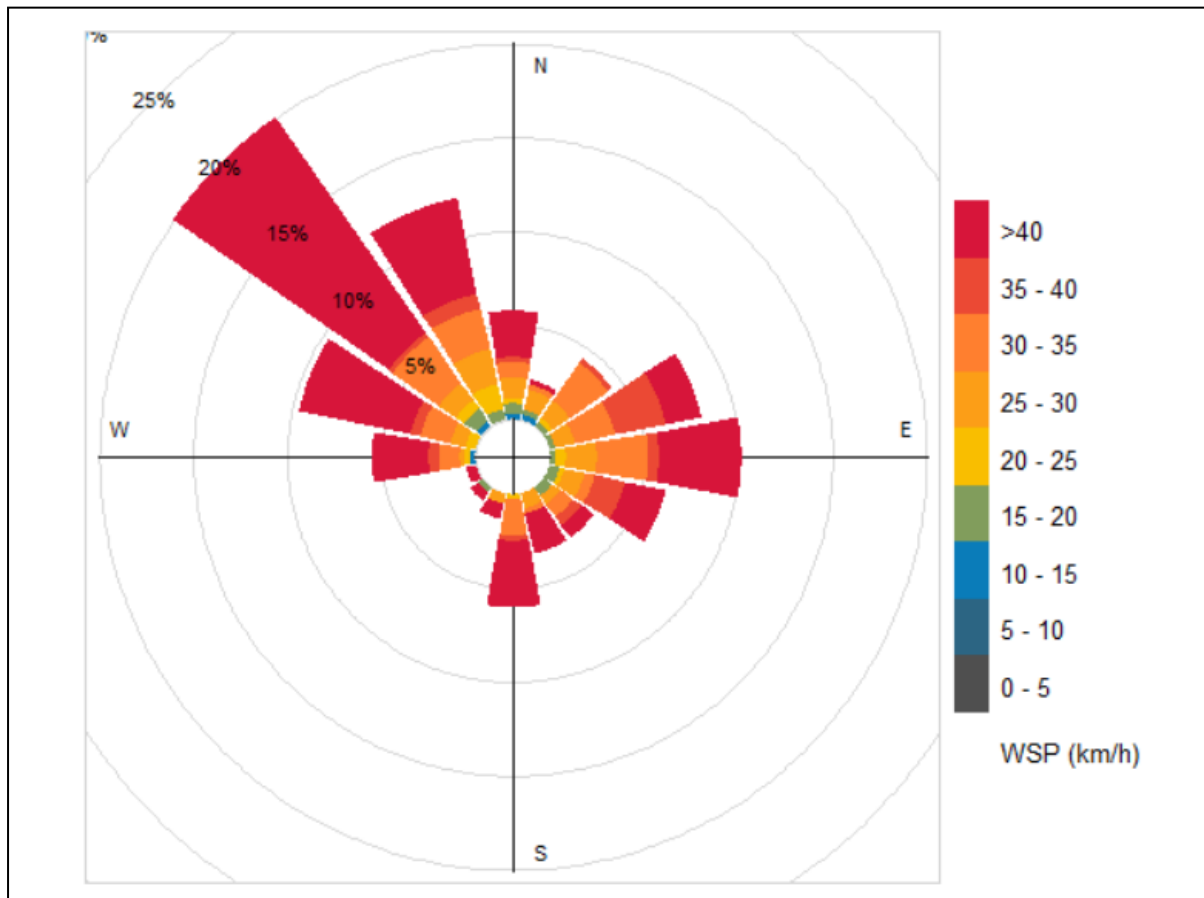


Figure 82: Output of Plotting Wind Rose Diagram

### 3.6 Violin Chart

This technique is used in analysis 16. Process of making violin chart:

1. Having a dataset with two variables.
2. Plotting a grey area using `geom_violin()`.

**Code:**

```
ggplot(wind_data1,aes(x=windDir3pm,y= windSpeed3pm)) +  
  geom_violin(alpha=0.5, color="gray")
```

Figure 83: Code of using `geom_violin()`

**Output:**

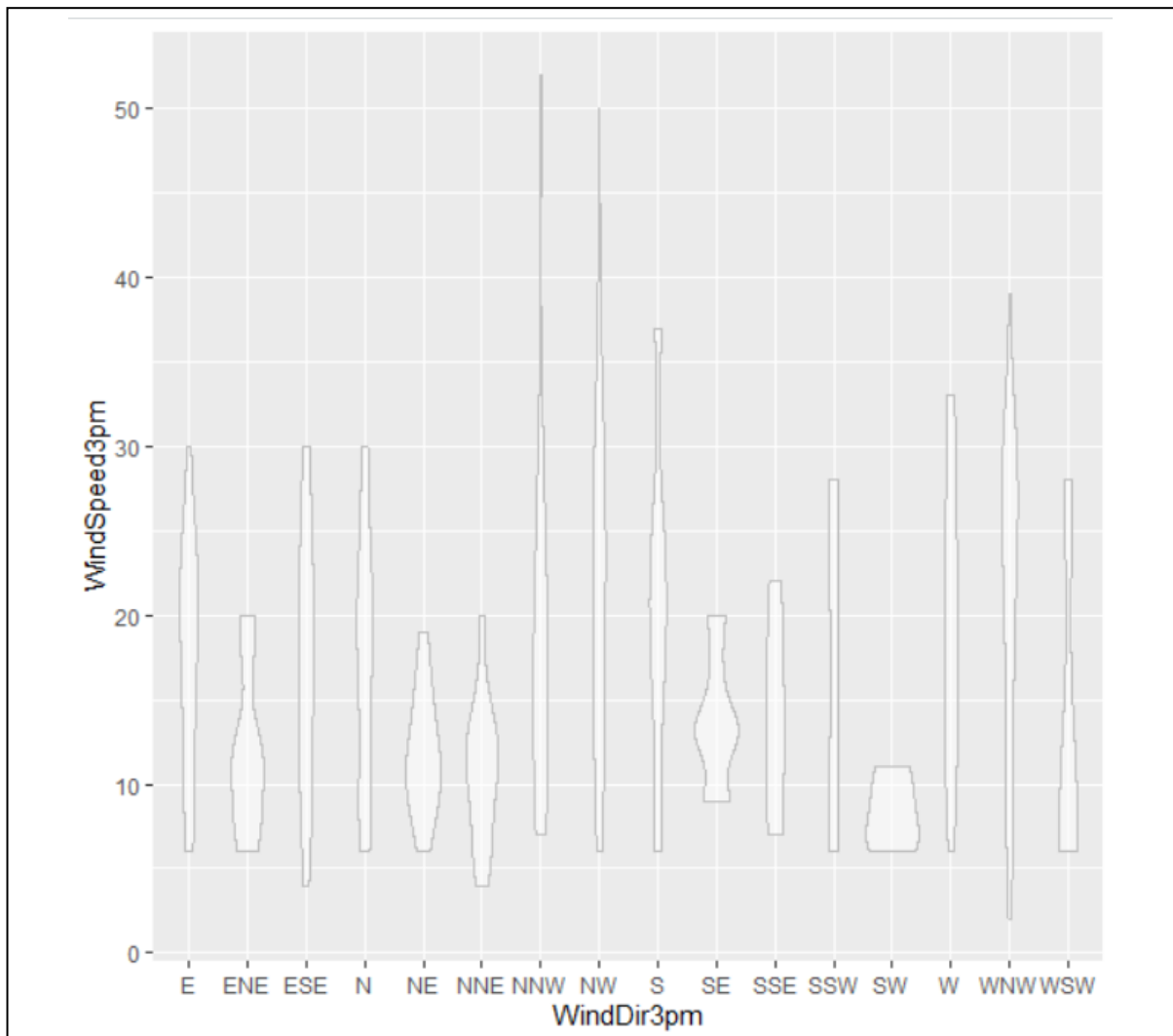


Figure 84: Output of using `geom_violin()`

3. Adding points at the grey area using `geom_jitter()` or `geom_point()`.

**Code:**

```
ggplot(wind_data1, aes(x=windDir3pm, y= windSpeed3pm)) +  
  geom_violin(alpha=0.5, color="gray")+  
  geom_jitter(alpha=0.5, aes(color=windDir3pm), position = position_jitter(width = 0.1))
```

Figure 85: Code of using `geom_jitter()`

**Output:**

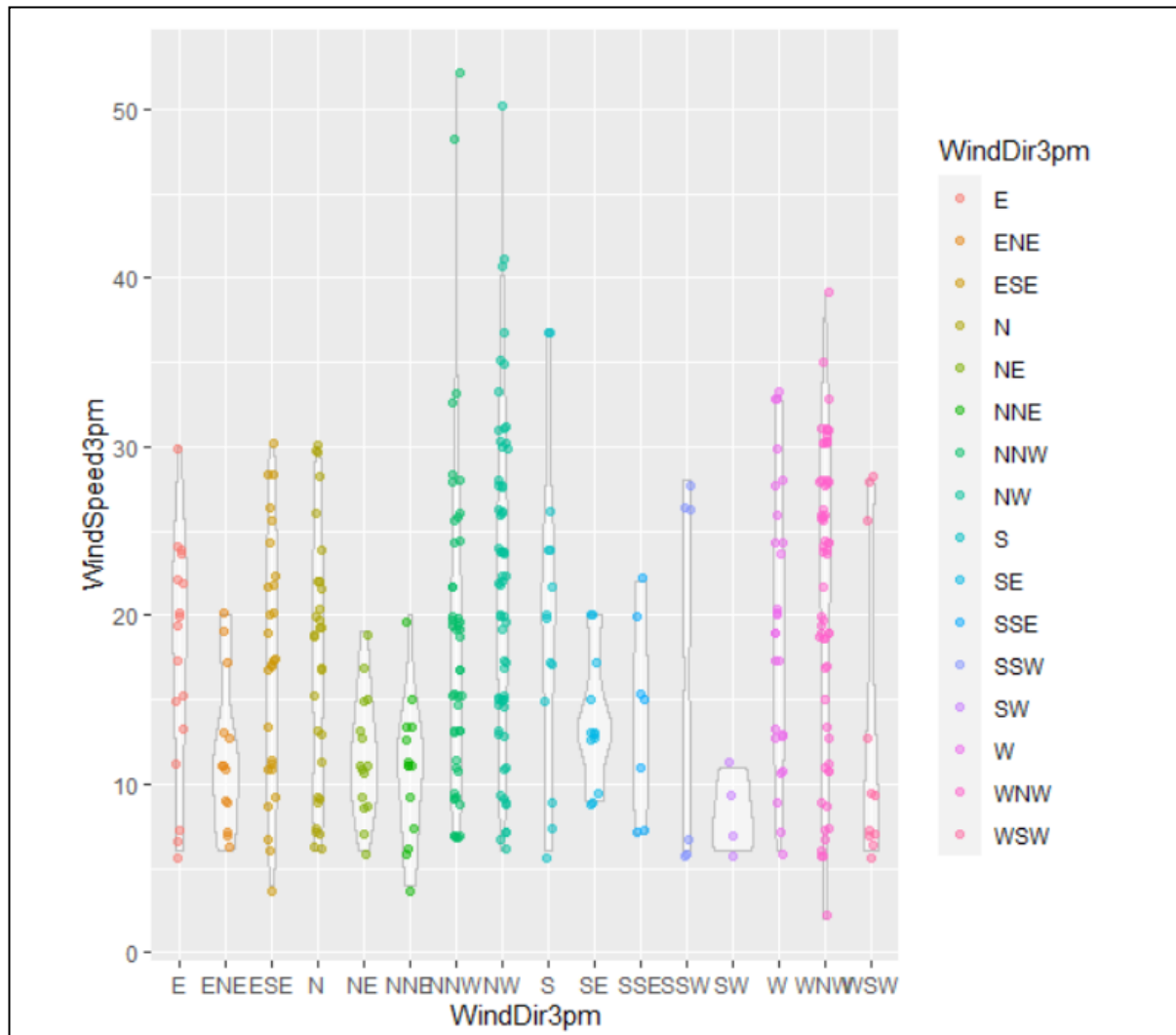


Figure 86: Output of using `geom_jitter()`

4. Changing the view from vertical to horizontal using coord\_flip().

**Code:**

```
ggplot(wind_data1, aes(x=WindDir3pm, y= windSpeed3pm)) +  
  geom_violin(alpha=0.5, color="gray")+  
  geom_jitter(alpha=0.5, aes(color=WindDir3pm), position = position_jitter(width = 0.1)) +  
  coord_flip()
```

Figure 87: Code of using coord\_flip()

**Output:**

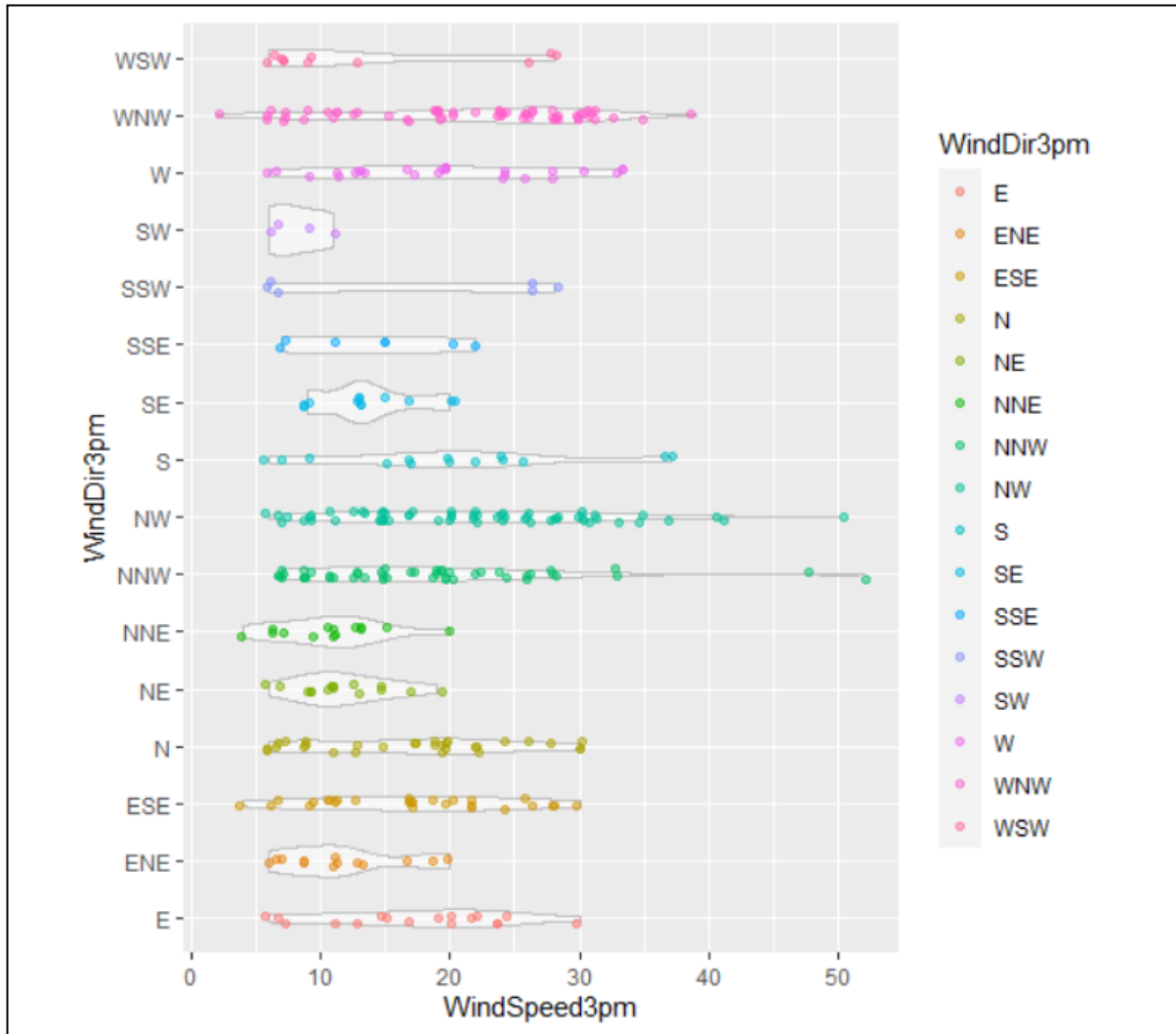


Figure 88: Output of using coord\_flip()

### 3.7 Correlation among Weather Data

This technique is used in analysis 17. It needs a new package, called “corrplot”. “corrplot” allows to plot a correlogram, which is a graph of correlation matrix. It could only plot the correlation between numeric data. Hence, in the original weather dataset, we could only use numeric variables and non-NA values.

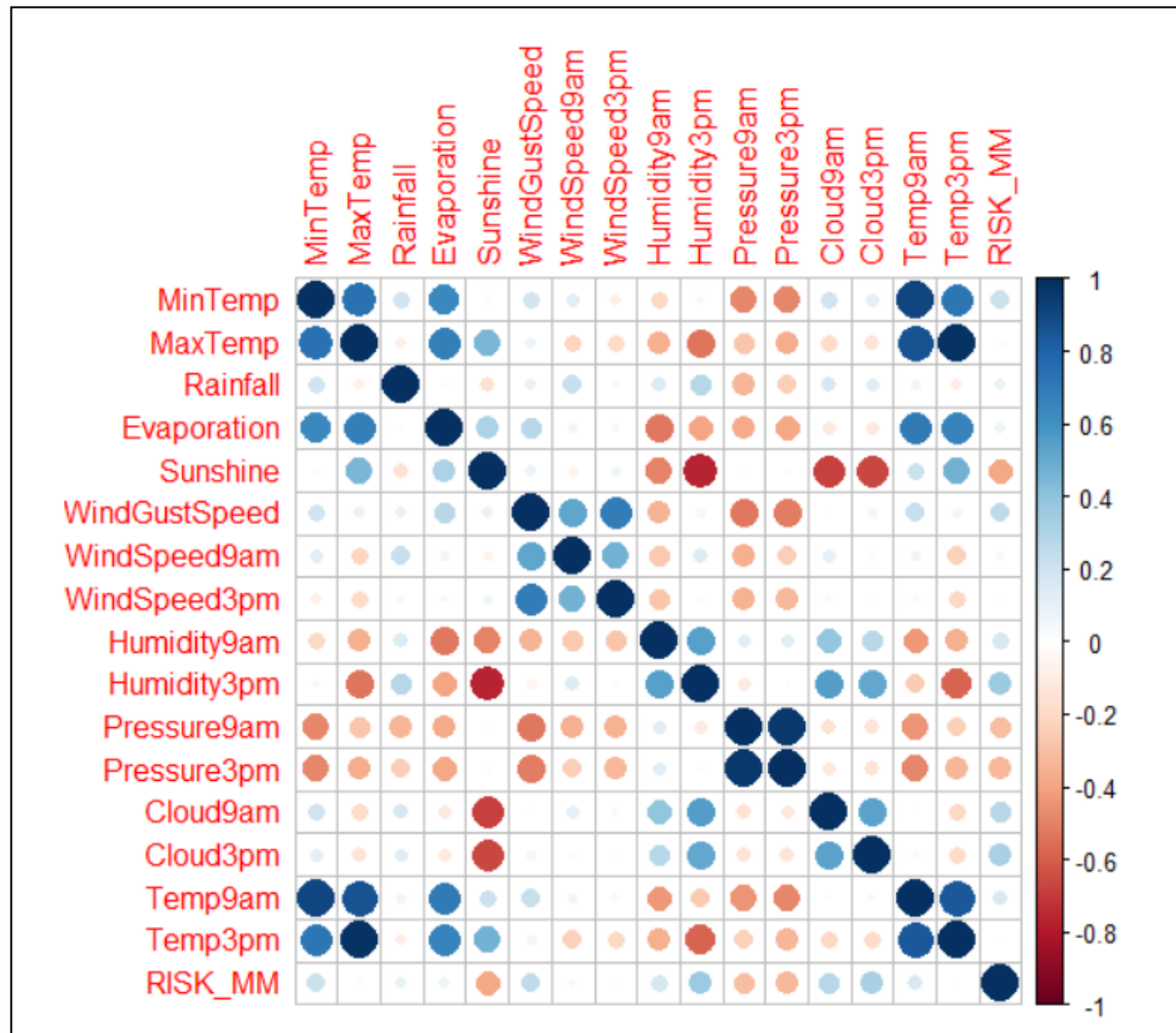


Figure 89: Output of Plotting Correlogram



### 3.8 Geom\_hline and geom\_vline

This technique is used in analysis 7 and 11. Geom\_hline will provide a horizontal line, while geom\_vline will provide a vertical line in a ggplot graph. Comparison of the diagram with and without geom\_hline and geom\_vline will show as below:

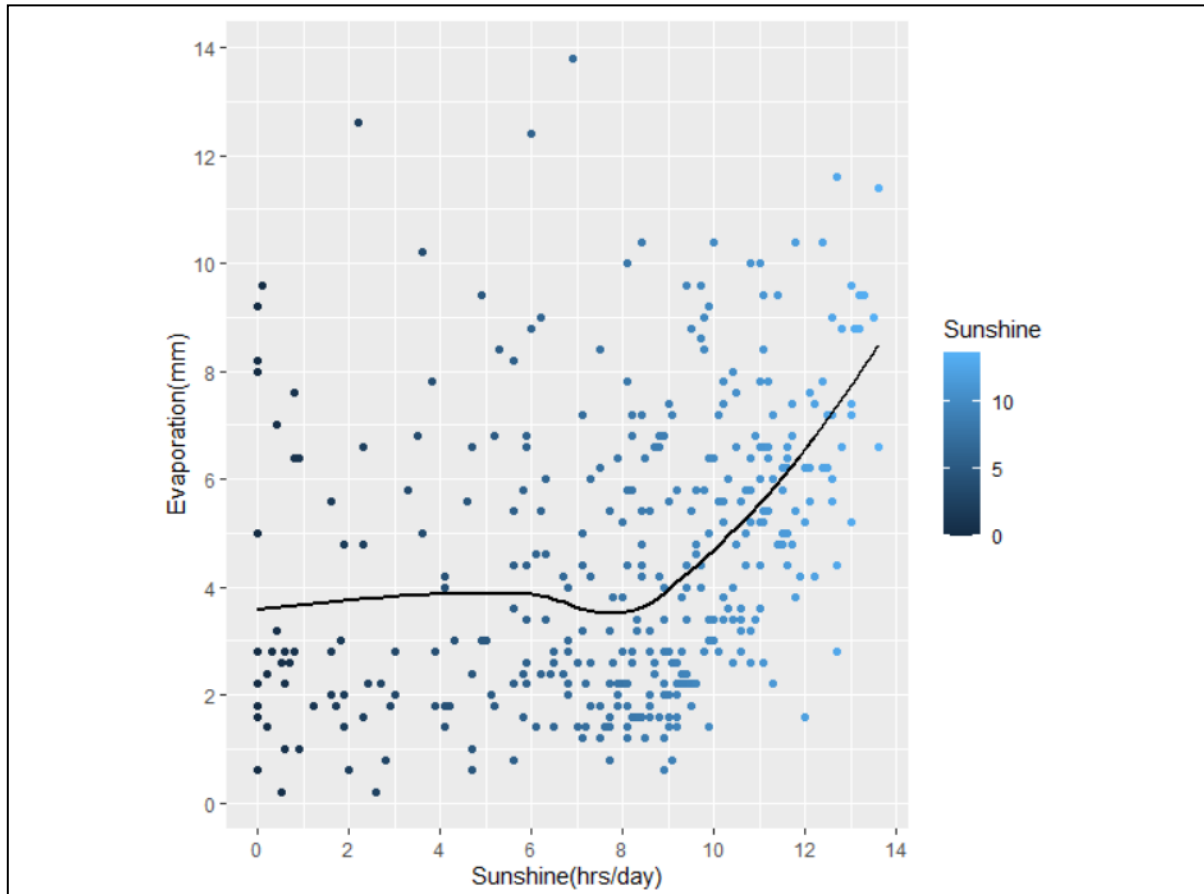


Figure 90: Point graph without geom\_hline and geom\_vline

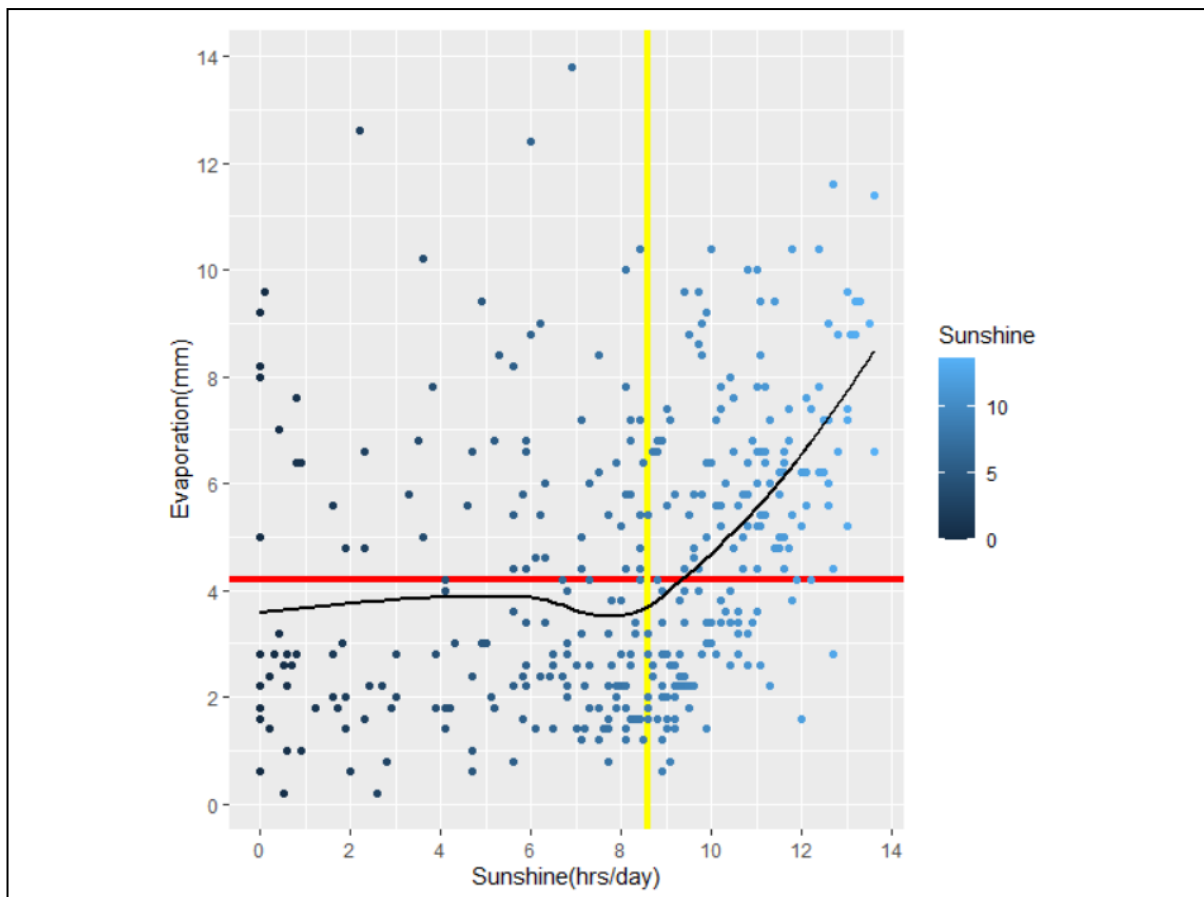


Figure 91: Point graph with geom\_hline and geom\_vline

## 4.0 Conclusion

In this report, the data analysis techniques had been shown to explain the relationship among the complex US daily weather data. Before analysing the dataset, the data import and pre-processing had been completed to facilitate the analysing process. After that, the analysis of US weather data had been done with different topics of questions and several related analyses. At last, a justification had been made after every analysis, and a small conclusion had been made after every question.

Analysis of weather data is essential for weather forecasting in many aspects such as agriculture, fishery, and tourism. Lots of professions are at the mercy of the forces of nature. Weather forecasting helps take precautions against unfavourable weather and helps boost economic development by increasing the productivity of different fields.

Through the research and development of R language, I have learned the fundamentals of data analysis such as mapping, data cleaning, data correlation, and others. It encourages my interest in data analysis and expands my knowledge of weather information.

## 5.0 Reference

Al-Mohannadi, A. and Qatar, M., 2006. *THE EFFECT OF WEATHER CONDITIONS ON THE SEASONAL VARIATION OF PHYSICAL ACTIVITY*. [online] Aspetar.com. Available at: <<https://www.aspetar.com/Journal/upload/PDF/2015610153157.pdf>>

[Accessed 27 May 2021].

Bob Berwyn, 2020. *New Study Shows Global Warming Increasing Frequency of the Most-Destructive Tropical Storms - Inside Climate News*. [online] Inside Climate News. Available at: <<https://insideclimatenews.org/news/19052020/hurricane-tropical-storms-climate-change/>>

[Accessed 27 May 2021].

Center for Climate and Energy Solutions. 2020. *Hurricanes and Climate Change | Center for Climate and Energy Solutions*. [online] Available at: <<http://www.c2es.org/content/hurricanes-and-climate-change/#:~:text=Frequency%20and%20intensity%20vary%20from,year%2C%20including%20about%20seven%20hurricanes.>>>

[Accessed 27 May 2021].

Claude E. Boyd, 2012. *Evaporation affected by sunlight, temperature, wind « Global Aquaculture Advocate*. [online] Global Aquaculture Alliance. Available at: <[https://www.aquaculturealliance.org/advocate/evaporation-affected-by-sunlight-](https://www.aquaculturealliance.org/advocate/evaporation-affected-by-sunlight-temperature-)

[temperature-](https://www.aquaculturealliance.org/advocate/evaporation-affected-by-sunlight-temperature-)

[wind/#:~:text=Solar%20radiation%20and%20temperature%20are,capacity%20to%20hold%20water%20vapor.&text=This%20results%20in%20higher%20water%20temperature%20and%20favors%20greater%20evaporation.>](https://www.aquaculturealliance.org/advocate/evaporation-affected-by-sunlight-temperature-)

[Accessed 26 May 2021].

Climatestotravel.com. 2017. *South Africa climate: average weather, temperature, precipitation, when to go*. [online] Available at: <<https://www.climatestotravel.com/climate/south-africa>>

[Accessed 26 May 2021].

DataFlair. 2019. *Data Manipulation in R - Find all its concepts at a single place! - DataFlair*. [online] Available at: <<https://data-flair.training/blogs/data-manipulation-in-r/>>

[Accessed 23 May 2021].

Kim Love, 2019. *The Advantages of RStudio - The Analysis Factor*. [online] The Analysis Factor. Available at: <<https://www.theanalysisfactor.com/the-advantages-of-rstudio/>> [Accessed 23 May 2021].

Livecaboradio. 2019. *Beaufort Scale on Wind Speeds*. [online] Available at: <<https://livecaboradio.blog/2019/01/04/beaufort-scale-on-wind-speeds/comment-page-1/>> [Accessed 27 May 2021].

Matuszko, D., 2011. Influence of cloudiness on sunshine duration. *International Journal of Climatology*, 32(10), pp.1527-1536.

Scijinks.gov. 2021. *What Makes It Rain? | NOAA SciJinks – All About Weather*. [online] Available at: <<https://scijinks.gov/rain/#:~:text=Clouds%20are%20made%20of%20water,fall%20to%20Earth%20as%20rain.>> [Accessed 26 May 2021].

Smart Fog. 2016. *How Rain and Humidity Connected? | Smart Fog*. [online] Available at: <<https://www.smartfog.com/how-rain-and-humidity-connected.html#:~:text=Connection%20of%20Rain%20and%20Humidity&text=When%20it%20rains%2C%20the%20humidity,completely%20saturated%20with%20water%20vapor.>> [Accessed 26 May 2021].

Stephanie Sigafos, 2012. *What causes wind gusts?*. [online] Wxguys.ssec.wisc.edu. Available at: <<https://wxguys.ssec.wisc.edu/2012/01/09/what-causes-wind-gusts/>> [Accessed 27 May 2021].

Student Materials. n.d. *Evaporation and Climate*. [online] Available at: <[https://serc.carleton.edu/integrate/teaching\\_materials/food\\_supply/student\\_materials/905#:~:text=Humidity%2C%20or%20water%20vapor%20content,and%20less%20evaporation%20can%20occur.](https://serc.carleton.edu/integrate/teaching_materials/food_supply/student_materials/905#:~:text=Humidity%2C%20or%20water%20vapor%20content,and%20less%20evaporation%20can%20occur.)> [Accessed 26 May 2021].

Think Metric!. 2021. *Temperature*. [online] Available at: <<https://thinkmetric.uk/basics/temperature/>> [Accessed 27 May 2021].

TINYIKO R. NKUNA and JOHN O. ODIYO, 2011. *The relationship between temperature and rainfall variability in the Levubu sub-catchment, South Africa*. [online] Iaras.org. Available at: <<https://www.iaras.org/iaras/filedownloads/ijes/2016/008-0011.pdf>> [Accessed 26 May 2021].

Uc-r.github.io. 2020. *Visual Data Exploration · UC Business Analytics R Programming Guide*. [online] Available at: <<http://uc-r.github.io/gda>> [Accessed 23 May 2021].