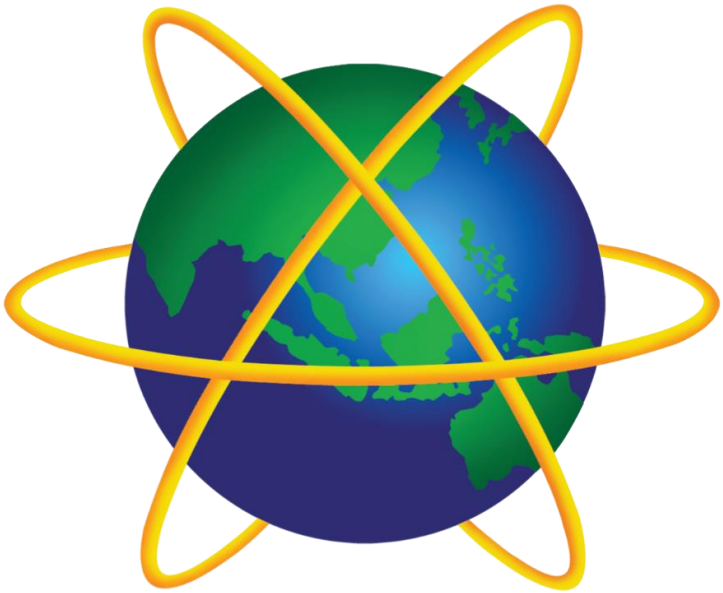


A · P · U

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

INDIVIDUAL ASSIGNMENT

CT127-3-2-PFDA

UC2F2102CS(DA)

NAME: ONG CHENG KEI

TP NO: TP055620

SUBJECT: PROGRAMMING FOR DATA ANALYSIS

LECTURER NAME: MINNU HELEN JOSEPH

Tutor Name : LIEW YEE JING

Table of Contents

Introduction

Weather plays an important role in nature, such as maintaining the water cycle and influencing the surrounding temperature and condition. There is no doubt that weather also can affect people's daily lives too. For example, most people will stay inside a shelter when the weather condition is bad while going outside on a sunny day. Therefore, this assignment is about exploring a weather dataset from the United States to discover any valuable information that may bring benefits. Some of the main data analysis techniques that will be applied are data exploration, data manipulation, data transformation, and data visualization. All the data analysis processes are done with the R programming language and RStudio.

Assumptions

One of the main assumptions is that there is an identical dataset, which is called “weather” in a R package called “rattle.data”. In the below, it will be proven the assumption is true.

```r
# There is an identical dataset in a package called "rattle.data"
install.packages("rattle.data")
library("rattle.data")

# load the dataset
data("weather")
# load the original dataset for assignment
realWeatherData = read.csv("weather.csv")
```

Figure 1: R code snippet

```
weather                  366 obs. of 24 variables
 $ Date          : Date, format: "2007-11-01" "2007-11-02" "2007-11-03" ...
 $ Location      : Factor w/ 49 levels "Adelaide","Albany",..: 10 10 10 10 10 10 10 10 10 10 ...
 $ 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   : Ord.factor w/ 16 levels "N"<"NNE"<"NE"<..: 15 4 15 15 8 7 7 5 9 5 ...
 $ WindGustSpeed: num  30 39 85 54 50 44 43 41 48 31 ...
 $ WindDir9am    : Ord.factor w/ 16 levels "N"<"NNE"<"NE"<..: 11 5 1 14 8 7 7 7 5 9 ...
 $ WindDir3pm    : Ord.factor w/ 16 levels "N"<"NNE"<"NE"<..: 15 13 2 13 6 5 6 5 4 6 ...
 $ WindSpeed9am : num  6 4 6 30 20 20 19 11 19 7 ...
 $ WindSpeed3pm : num  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 2 : Overview of the weather dataset from rattle.data

The above figure shows columns and values for the weather dataset in rattle.data package.

```
realWeatherData          366 obs. of 22 variables
  $ 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  : chr  "NW" "ENE" "NW" "NW" ...
  $ WindGustSpeed: int  30 39 85 54 50 44 43 41 48 31 ...
  $ WindDir9am   : chr  "SW" "E" "N" "WNW" ...
  $ WindDir3pm   : chr  "NW" "W" "NNE" "W" ...
  $ 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    : chr  "No" "Yes" "Yes" "Yes" ...
  $ RISK_MM      : num  3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 0 ...
  $ RainTomorrow : chr  "Yes" "Yes" "Yes" "Yes" ...
```

Figure 3: Overview of the realWeatherData used for this assignment

Aside from having two extra variables in the weather dataset, both datasets in figure 2 and figure 3 are very identical, especially the column names and values. To prove they are identical, firstly the first two columns for the weather dataset will be dropped and the data types will be altered, so both data frames having the same column names and the same data types.

```r
# Before validating it is an identical dataset, first two columns are dropped
weather = weather[,-c(1,2)]

# alter the column types so that both dataset contains same type
weather$WindGustDir = as.character(weather$WindGustDir)
weather$WindDir9am = as.character(weather$WindDir9am)
weather$WindDir3pm = as.character(weather$WindDir3pm)
weather$RainToday = as.character(weather$RainToday)
weather$RainTomorrow = as.character(weather$RainTomorrow)

realWeatherData$WindGustSpeed = as.numeric(realWeatherData$WindGustSpeed)
realWeatherData$WindSpeed9am = as.numeric(realWeatherData$WindSpeed9am)
realWeatherData$WindSpeed3pm = as.numeric(realWeatherData$WindSpeed3pm)
```

Figure 4 : R code snippet

The above figure shows, the first two columns which are date and location are dropped, and the data types for some columns are changed so both data-frames have the same columns with the same type.

```r
# test whether each cell are the same values between two dataset
identical(weather,realWeatherData)
```

Figure 5 : R code snippet

```
> # test whether each cell are the same values between two dataset
> identical(weather,realWeatherData)
[1] TRUE
```

Figure 6 : Result shows that both data frames are identical having the same value for each cell

Based on figure 5 and figure 6, it is clear that both data frames are identical. Therefore, the metadata for the weather dataset from "rattle.data" can be referenced and used for upcoming analysis.

```
> ?weather
```

Figure 7 : R code to get help for weather dataset

`MinTemp`

The minimum temperature in degrees celsius.

`MaxTemp`

The maximum temperature in degrees celsius.

`Rainfall`

The amount of rainfall recorded for the day in mm.

`Evaporation`

The so-called Class A pan evaporation (mm) in the 24 hours to 9am.

`Sunshine`

The number of hours of bright sunshine in the day.

`WindGustDir`

The direction of the strongest wind gust in the 24 hours to midnight.

`WindGustSpeed`

The speed (km/h) of the strongest wind gust in the 24 hours to midnight.

`Temp9am`

Temperature (degrees C) at 9am.

`RelHumid9am`

Relative humidity (percent) at 9am.

`Cloud9am`

Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

`WindSpeed9am`

Wind speed (km/hr) averaged over 10 minutes prior to 9am.

`Pressure9am`

Atmospheric pressure (hpa) reduced to mean sea level at 9am.

`Temp3pm`

Temperature (degrees C) at 3pm.

`RelHumid3pm`

Relative humidity (percent) at 3pm.

Figure 8 : Metadata about the weather dataset from rattle.data

Cloud3pm Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values. WindSpeed3pm Wind speed (km/hr) averaged over 10 minutes prior to 3pm. Pressure3pm Atmospheric pressure (hpa) reduced to mean sea level at 3pm.
RainToday Integer: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0.
RISK_MM The amount of rain. A kind of measure of the "risk". RainTomorrow The target variable. Did it rain tomorrow?

Figure 9 : Metadata about the weather dataset from rattle.data

According to figure 7, figure 8, and figure 9, the metadata can be used as a reference during analysis since it is confirmed that both weather datasets are identical. Through this, it provides quite a lot of information about the measurement scales used when recording the data. Other than that, it is also assumed that the dataset rows are arranged sequentially, meaning the first row represents day 1, the second row represents day 2, and follow on.

Installing Packages

```
# ---------------------- Install necessary packages ----------------------
install.packages("ggplot2")
install.packages("dplyr")
install.packages("ggthemes")
install.packages("RColorBrewer")
install.packages("PerformanceAnalytics")
install.packages("plotly")
install.packages("DataExplorer")
```

Figure 10 : R code snippet

Figure 10 shows the code to install the necessary packages that will be used throughout the data analysis.

Loading Libraries

```
# ------------------------- Load Libraries ------------------------------
library(ggplot2)
library(dplyr)
library(ggthemes)
library(RColorBrewer)
library(PerformanceAnalytics)
library(plotly)
library(DataExplorer)
```

Figure 11 : R code snippet

Figure 11 shows the code that loads the packages needed for data analysis.

- ggplot2 is used for data visualization
- dplyr package is used for data manipulation and data transformation
- ggthemes package is used for customization of themes of ggplot.
- RColorBrewer package is used for customization of colors for ggplot.
- PerformanceAnalytics package is used for creating a correlational matrix plot.
- plotly package is used for creating an interactive ggplot in RStudio.
- DataExplorer is used for performing explanatory data analysis.

Data Import

```
# Change working directory
setwd("D:\\School Materials\\Year 2\\Programming For Data Analysis\\Assignment")

weatherData = read.csv("weather.csv")
```

Figure 12 : R code snippet

Figure 12 shows the code that sets the working directory and imports the dataset which is "weather.csv".

Data Pre-processing

```
#Determine the dimensions of the weatherData
nrow(weatherData)
ncol(weatherData)
```

Figure 13 : R code snippet

```
> nrow(weatherData)
[1] 366
> ncol(weatherData)
[1] 22
>
```

Figure 14 : Results from the R code above

Figure 13 and figure 14 show the code and result that determines the number of rows and number of columns of the weather dataset.

```
# Getting initial view on weatherData
View(head(weatherData))
View(tail(weatherData))
```

Figure 15 : R Code snippet

Figure 15 shows the R code that gets the first 6 rows and the last 6 rows from the weather dataset.

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGust_Direction	WindGust_Speed	WindDirection_9am	WindDirection_3pm	WindSpeed_9am	WindSpeed_3pm
1	8.0	24.3	0.0	3.4	6.3	NW	30	SW	NW	6	20
2	14.0	26.9	3.6	4.4	9.7	ENE	39	E	W	4	17
3	13.7	23.4	3.6	5.8	3.3	NW	85	N	NNE	6	6
4	13.3	15.5	39.8	7.2	9.1	NW	54	WNW	W	30	24
5	7.6	16.1	2.8	5.6	10.6	SSE	50	SSE	ESE	20	28
6	6.2	16.9	0.0	5.8	8.2	SE	44	SE	E	20	24

Figure 16 : First 6 rows of the weather dataset

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGust_Direction	WindGust_Speed	WindDirection_9am	WindDirection_3pm	WindSpeed_9am
361	7.9	26.1	0	6.8	3.5	NNW	43	*NA*	WNW	0
362	9.0	30.7	0	7.6	12.1	NNW	76	SSE	NW	7
363	7.1	28.4	0	11.6	12.7	N	48	NNW	NNW	2
364	12.5	19.9	0	8.4	5.3	ESE	43	ENE	ENE	11
365	12.5	26.9	0	5.0	7.1	NW	46	SSW	WNW	6
366	12.3	30.2	0	6.0	12.6	NW	78	NW	WNW	31

Figure 17 : Last 6 rows of the weather dataset

Figure 16 and figure 17 shows the result.

```
# Getting to know the data types for each column
str(weatherData)
```

Figure 18 : R code snippet

Next, the str() command is used to determine the data types of each column.

```
> str(weatherData)
'data.frame':   366 obs. of  22 variables:
 $ 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  : chr  "NW" "ENE" "NW" "NW" ...
 $ WindGustSpeed: int  30 39 85 54 50 44 43 41 48 31 ...
 $ WindDir9am   : chr  "SW" "E" "N" "WNW" ...
 $ WindDir3pm   : chr  "NW" "W" "NNE" "W" ...
 $ 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    : chr  "No" "Yes" "Yes" "Yes" ...
 $ RISK_MM      : num  3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 0 ...
 $ RainTomorrow : chr  "Yes" "Yes" "Yes" "Yes" ...
```

Figure 19 : Result from the str() command

Based on figure 19, the weather dataset contains some columns with character data types, integer data types, numeric data types.

```
# Get a report on the weather dataset (using DataExplorer)
introduce(weatherData)
```

Figure 20 : R code snippet (extra feature)

In figure 20, introduce() is called under the DataExplorer package. It can help to quickly identify the basic information of the weather dataset.

```
introduce(weatherData)
rows columns discrete_columns continuous_columns all_missing_columns
 366      22                5                  17                   0
total_missing_values complete_rows total_observations memory_usage
                  47           328               8052        62368
```

Figure 21 : Result from introduce() command

Based on figure 21 above, the weather dataset contains 5 discrete columns which are the columns with character values determined by the str() command in figure 19. The weather dataset does not have an all-missing column but does have 47 missing values.

```
# Convert character values into factor objects
weatherData$WindGustDir = factor(weatherData$WindGustDir)
weatherData$WindDir9am = factor(weatherData$WindDir9am)
weatherData$WindDir3pm = factor(weatherData$WindDir3pm)
weatherData$RainToday = factor(weatherData$RainToday,levels = c("Yes","No"))
weatherData$RainTomorrow = factor(weatherData$RainTomorrow,levels = c("Yes","No"))
```

Figure 22 : R code snippet

Figure 13 shows R code that transforms columns with character data types into factor objects. This helps to categorize similar characters into a category. Moving on, missing values for each column is determined.

```
#Check for columns with missing value
summary(weatherData)
```

Figure 23 : R code snippet

By using the summary() command in figure 14, columns with missing or NA values can be found easily.

```
> summary(weatherData)
    MinTemp          MaxTemp         Rainfall       Evaporation        Sunshine       WindGustDir
 Min.   :-5.300   Min.   : 7.60   Min.   : 0.000   Min.   : 0.200   Min.   : 0.000   NW     : 73
 1st Qu.: 2.300   1st Qu.:15.03   1st Qu.: 0.000   1st Qu.: 2.200   1st Qu.: 5.950   NNW    : 44
 Median : 7.450   Median :19.65   Median : 0.000   Median : 4.200   Median : 8.600   E      : 37
 Mean   : 7.266   Mean   :20.55   Mean   : 1.428   Mean   : 4.522   Mean   : 7.909   WNW    : 35
 3rd Qu.:12.500   3rd Qu.:25.50   3rd Qu.: 0.200   3rd Qu.: 6.400   3rd Qu.:10.500   ENE    : 30
 Max.   :20.900   Max.   :35.80   Max.   :39.800   Max.   :13.800   Max.   :13.600   (Other):144
                                                                    NA's   :3        NA's   :  3
 WindGustSpeed     WindDir9am    WindDir3pm   WindSpeed9am     WindSpeed3pm    Humidity9am
 Min.   :13.00   SE     : 47   NW     : 61   Min.   : 0.000   Min.   : 0.00   Min.   :36.00
 1st Qu.:31.00   SSE    : 40   WNW    : 61   1st Qu.: 6.000   1st Qu.:11.00   1st Qu.:64.00
 Median :39.00   NNW    : 36   NNW    : 47   Median : 7.000   Median :17.00   Median :72.00
 Mean   :39.84   N      : 31   N      : 30   Mean   : 9.652   Mean   :17.99   Mean   :72.04
 3rd Qu.:46.00   NW     : 30   ESE    : 27   3rd Qu.:13.000   3rd Qu.:24.00   3rd Qu.:81.00
 Max.   :98.00   (Other):151   (Other):139   Max.   :41.000   Max.   :52.00   Max.   :99.00
 NA's   :2       NA's   : 31   NA's   :  1   NA's   :7
  Humidity3pm     Pressure9am      Pressure3pm        Cloud9am        Cloud3pm        Temp9am
 Min.   :13.00   Min.   : 996.5   Min.   : 996.8   Min.   :0.000   Min.   :0.000   Min.   : 0.100
 1st Qu.:32.25   1st Qu.:1015.4   1st Qu.:1012.8   1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 7.625
 Median :43.00   Median :1020.1   Median :1017.4   Median :3.500   Median :4.000   Median :12.550
 Mean   :44.52   Mean   :1019.7   Mean   :1016.8   Mean   :3.891   Mean   :4.025   Mean   :12.358
 3rd Qu.:55.00   3rd Qu.:1024.5   3rd Qu.:1021.5   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:17.000
 Max.   :96.00   Max.   :1035.7   Max.   :1033.2   Max.   :8.000   Max.   :8.000   Max.   :24.700

    Temp3pm      RainToday     RISK_MM       RainTomorrow
 Min.   : 5.10   Yes: 66   Min.   : 0.000   Yes: 66
 1st Qu.:14.15   No :300   1st Qu.: 0.000   No :300
 Median :18.55             Median : 0.000
 Mean   :19.23             Mean   : 1.428
 3rd Qu.:24.00             3rd Qu.: 0.200
 Max.   :34.50             Max.   :39.800
```

Figure 24 : Result from summary()

It can be determined that :

- Sunshine has 3 missing values
- WindGustSpeed has 2 missing values
- WindDir9am has 31 missing values
- WindDir3pm has 1 missing value
- WindSpeed9am has 7 missing values

Since the number of missing values for columns with missing values is quite small, therefore data treatment is not necessary, and treatment might not even be significant. Before proceeding, some columns are renamed, so that there are more intuitive and easily understandable.

```
# Renaming some columns
names(weatherData)[6:19] = c("WindGust_Direction",
                             "WindGust_Speed",
                             "WindDirection_9am",
                             "WindDirection_3pm",
                             "WindSpeed_9am",
                             "WindSpeed_3pm",
                             "Humidity_9am",
                             "Humidity_3pm",
                             "Pressure_9am",
                             "Pressure_3pm",
                             "Cloud_9am",
                             "Cloud_3pm",
                             "Temp_9am",
                             "Temp_3pm")

names(weatherData)[21] = c("Risk_MM")
```

Figure 25 : R code snippet

Figure 16 shows the R code that renames some columns in the weather dataset. Columns that have been renamed :

Old column names	New column names
WindGustDir	WindGust_Direction
WindGustSpeed	WindGust_Speed
WindDir9am	WindDirection_9am
WindDir3pm	WindDirection_3pm
WindSpeed9am	WindSpeed_9am
WindSpeed3pm	WindSpeed_3pm
Humidity9am	Humidity_9am
Humidity3pm	Humidity_3pm
Pressure9am	Pressure_9am
Pressure3pm	Pressure_3pm
Cloud9am	Cloud_9am
Cloud3pm	Cloud_3pm
Temp9am	Temp_9am
Temp3pm	Temp_3pm
RISK_MM	Risk_MM

Question 1 : How to determine whether it will rain tomorrow?

According to the assumptions, a day is only considered as raining if the precipitation within 24 hours is at least 1mm. In meteorology, precipitation stands for any water that falls from the cloud to the ground, such as snow, and rain (Cambridge Dictionary,2021). In addition, the Risk_MM variable is a "risk" measurement for rain tomorrow in terms of rainfall. Since a day is considered raining if precipitation exceeds 1mm, therefore if the Risk_MM exceeds 1mm it is predicted that it will be raining tomorrow. Therefore, in this analysis, the Risk_MM variable and RainTomorrow variable are used as target variables to try to find the factors that influence these two variables.

Analysis 1 : Determine whether clouds formation affects the rain tomorrow

This analysis is trying to determine whether cloud formation has effects on Risk_MM and RainTomorrow variables. In addition, this analysis will be mainly focusing on the presence of the clouds in the evening which is 3 pm. This is because 3 pm is almost at the end of the day, and near tomorrow. Another thing that is worth mentioning, is the scale used to measure the presence of the clouds. The measurement scale used for clouds is called oktas as shown in the assumption. Okta is a measurement unit used to describe the cloud coverage of a fraction of the sky at any given location (MichaelV,2018). It helps to estimate the sky conditions by determining how many eights of the sky are covered by clouds (MichaelV,2018). The okta value ranges from 0 to 8 as shown in the figure below. 9 oktas is an additional value that describes the sky is obscured from view (MichaelV,2018). 9 oktas are not present in this dataset.

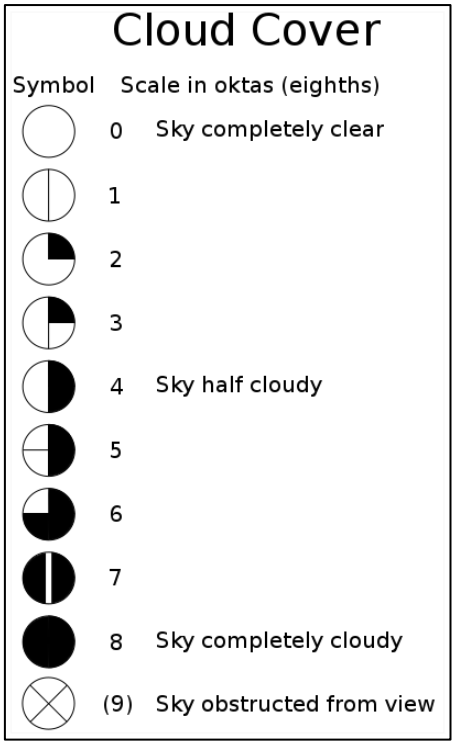


Figure 26 : Cloud cover symbols used in meteorology (MichaelV,2018)

```
# Create a new column that indicates whether sky is more than half cloudy,
# Cloud more than 4 oktas is more than half cloudy
  # As clouds are measured by (oktas) discrete scale of 0-8,
  # convert them into factor

weatherData %>%
  mutate(moreHalfCloudy = factor(weatherData$Cloud_3pm,
                                 levels = c(8,7,6,5,4,3,2,1,0),
                                 labels = c(rep("TRUE",4),rep("FALSE",5)))) %>%
  ggplot(aes(factor(Cloud_3pm),Risk_MM)) +
  geom_jitter(aes(color = moreHalfCloudy)) +
  labs (title = "Relationship between Risk_MM and Clouds at 3pm",
        subtitle = "Does presence of clouds at 3pm influence the risk of raining tomorrow",
        x = "Clouds at 3pm (oktas)",
        y = "Amount of predicted rainfall tomorrow (mm)",
        color = "Is the sky more than half cloudy ?") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text())+
  scale_color_brewer(palette = "Dark2")
```

Figure 27 : R code snippet

The R code snippet above shows the code used to plot a scatter plot that shows the relationship between clouds formation and Risk_MM. As discussed above, clouds measurements are in oktas, therefore it is converted into a factor object before plotting in scatter plot. In addition, a new column which is named “moreHalfCloudy” is created to indicates whether the oktas value is more than 4. It is done by taking the Cloud_3pm variable as factors and change their labels with the factor() function. If the value is more than 4, it will return “TRUE” which means the sky is more than half cloudy, otherwise, it will return “FALSE”. Lastly, a theme called

“fivethirthyeight” is added into the ggplot to beautify the plot. Due to the default configuration of the theme, the x and y-axis are hidden, so it has to be overridden with the theme() function to make the x and y-axis appear.

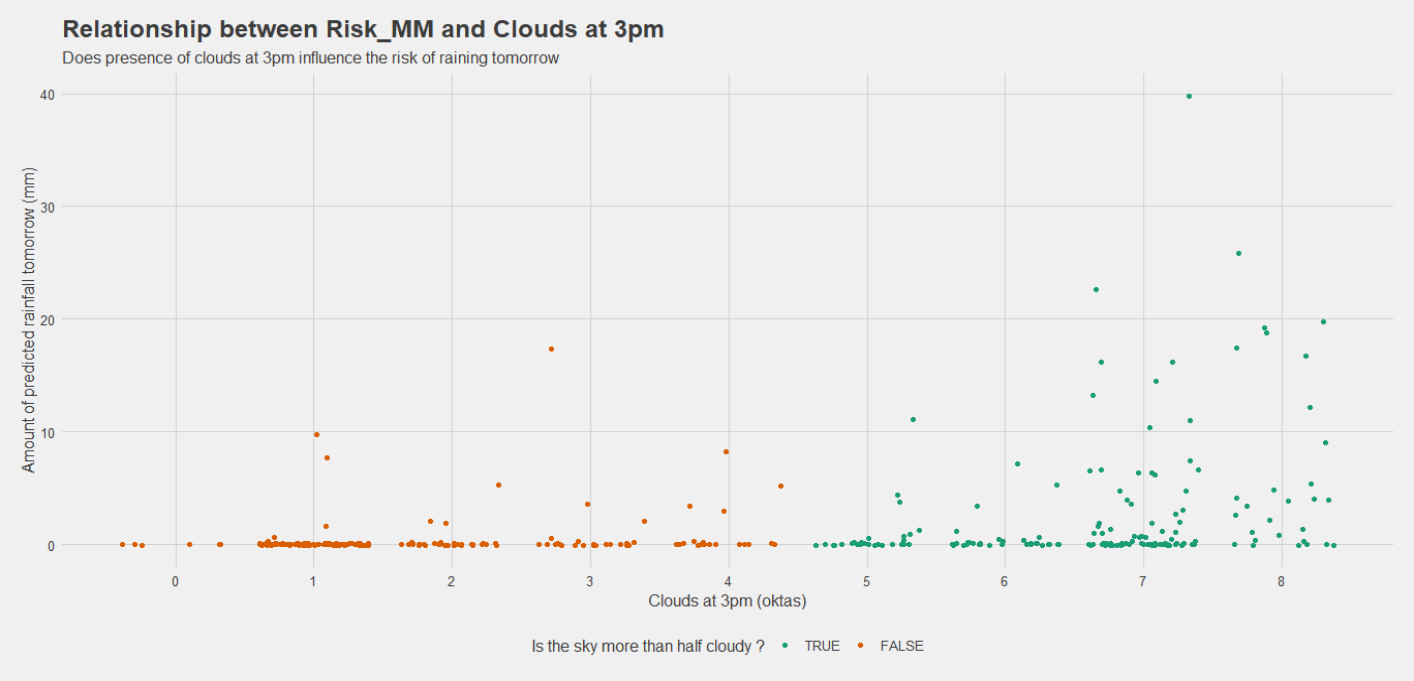


Figure 28 : Scatter Plot generated from R code snippet above

According to figure 28 above, there is an obvious trend, which is the cloudier it gets at 3 pm, the higher the amount of predicted rainfall for tomorrow. This might because clouds are formed from condensed water vapor and dust particles (Toothman,2008). The heat from the sun evaporates the water from the earth's surface into water vapor. As the water vapor rises, it condenses into tiny water droplets and forms clouds. Those tiny droplets are not heavy enough to fall on their own, therefore tiny water droplets must collide with each other to form bigger water droplets (What are clouds and why does it rain?,2013). This also causes the clouds to grow in size and holds more water. Once it is heavy enough, gravity will pull all the water droplets to fall to the ground (Toothman,2008). Therefore, if the sky is very cloudy, this means the clouds are big and often hold more water content. This indicates rain might be near the corner.

Analysis 2 : Determine the relationship between Sunshine and Risk_MM

```
# Create a new column that determines whether sunshine hours is above average

weatherData %>%
  filter(!is.na(Sunshine)) %>%     # filter missing sunshine values
  mutate(moreAvgSunshine = factor(Sunshine > mean(weatherData$Sunshine,na.rm = T),
                                  levels = c("TRUE","FALSE"))) %>%
  ggplot(aes(Sunshine,Risk_MM)) +
  geom_jitter(aes(color = moreAvgSunshine)) +
  labs (title = "Relationship between Risk_MM and Sunshine hours",
        subtitle = "Does sunshine influence the risk of raining tomorrow?",
        x = "Sunshine (hours)",
        y = "Amount of predicted rainfall tomorrow (Millimeter)",
        color = "Is the sunshine hours higher than average?") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text())+
  scale_color_brewer(palette = "Set1") +
  scale_x_continuous(breaks = seq(0,14,2)) # scale x-axis manually
```

Figure 29 : R code snippet

The R code snippet above shows the plotting of a scatter plot with sunshine variable and Risk_MM. Since sunshine contains some missing values, the missing values are filtered out. Then a new column is created to determine whether the hours of sunshine for a particular day are above average. The column contains “TRUE” if the sunshine hours are more than average, otherwise “FALSE”. Then, a scatter plot with a theme is plotted, and by using scale_color_brewer(), a palette color is specified to color the plot.

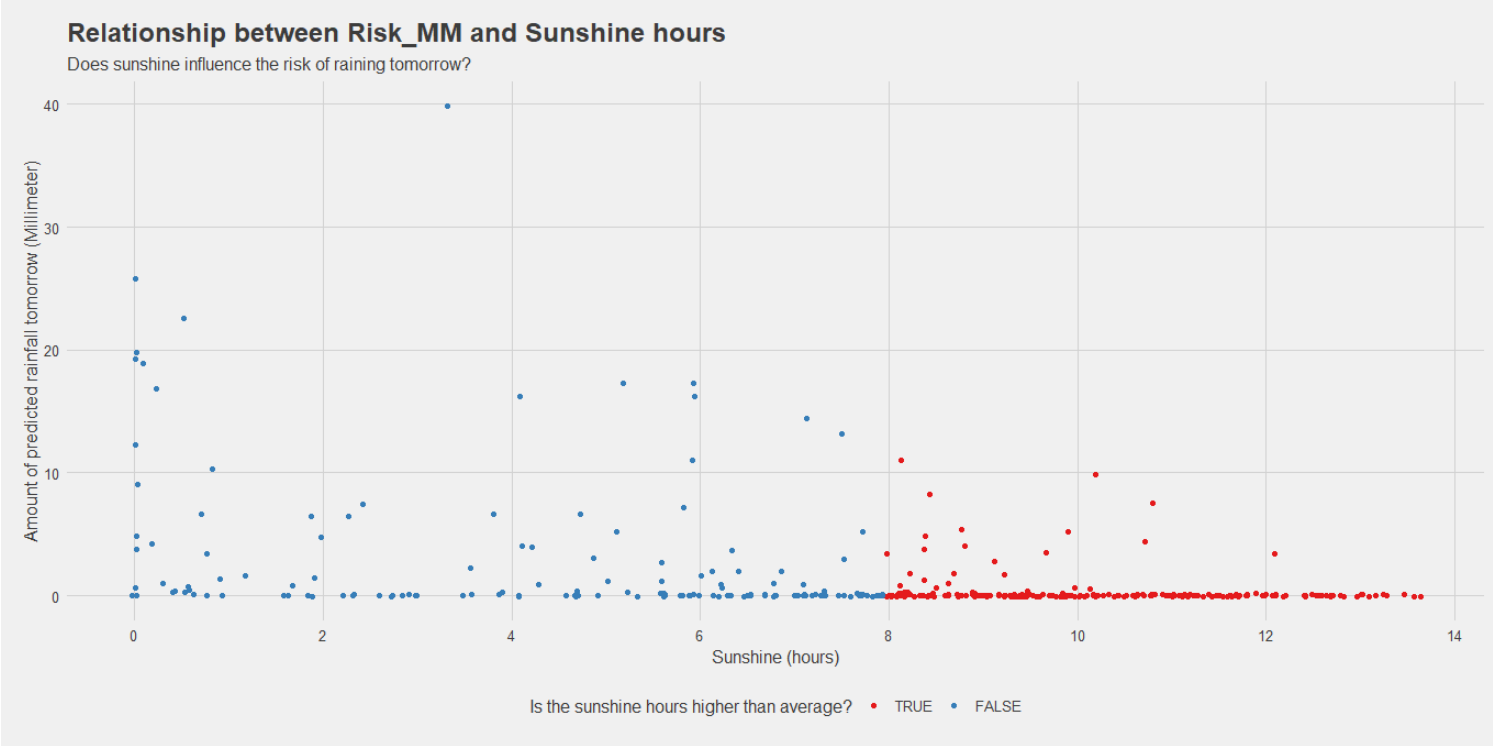


Figure 30 : Scatter Plot generated from the R code snippet above

Based on figure 30, the trend is clear when there are long hours of sunshine the lower the Risk_MM variable. From the previous analysis, it is known that cloud formation and Risk_MM have a proportional relationship, so lower clouds results in lower Risk_MM. Intuitively, lower clouds can also cause a higher number of sunshine hours. This is because there are no clouds that block the sunlight. This assumption can be confirmed by the next plot.

```
# Determine the relationship between sunshine & Clouds

# Create a new column that determines whether sunshine hours is above average
weatherData %>%
  filter(!is.na(Sunshine)) %>%  # filter out missing values
  mutate(moreAvgSunshine = factor(Sunshine > mean(weatherData$Sunshine,na.rm = T),
                                  levels = c("TRUE","FALSE"))) %>%
  ggplot(aes(Sunshine,Risk_MM)) +
  geom_jitter(aes(color = moreAvgSunshine,shape=RainTomorrow)) +
  labs (title = "Relationship between Risk_MM,Sunshine hours,Clouds at 3pm",
        subtitle = "Does clouds influence the number of sunshine hours?",
        x = "Sunshine (hours)",
        y = "Amount of predicted rainfall tomorrow (Millimeter)",
        color = "Is the sunshine hours higher than average?",
        shape = "Will it rain tomorrow ?") +
  facet_wrap(~Cloud_3pm)+
  theme_igray() +
  scale_color_brewer(palette = "Set1") +
  scale_shape_manual(values = c(16,17)) # scale shape of the point manually
```

Figure 31 : R code snippet

Based on the code snippet above, it is very similar to the code snippet in figure 29 above. The main difference is this time the scatter plot is facet into different categories based on the Cloud_3pm variable. This is done to determine the number of hours of sunshine with cloud formation. Besides that, the scale_shape_manual() function is used here to specify the shapes to be used for the scatter plot.

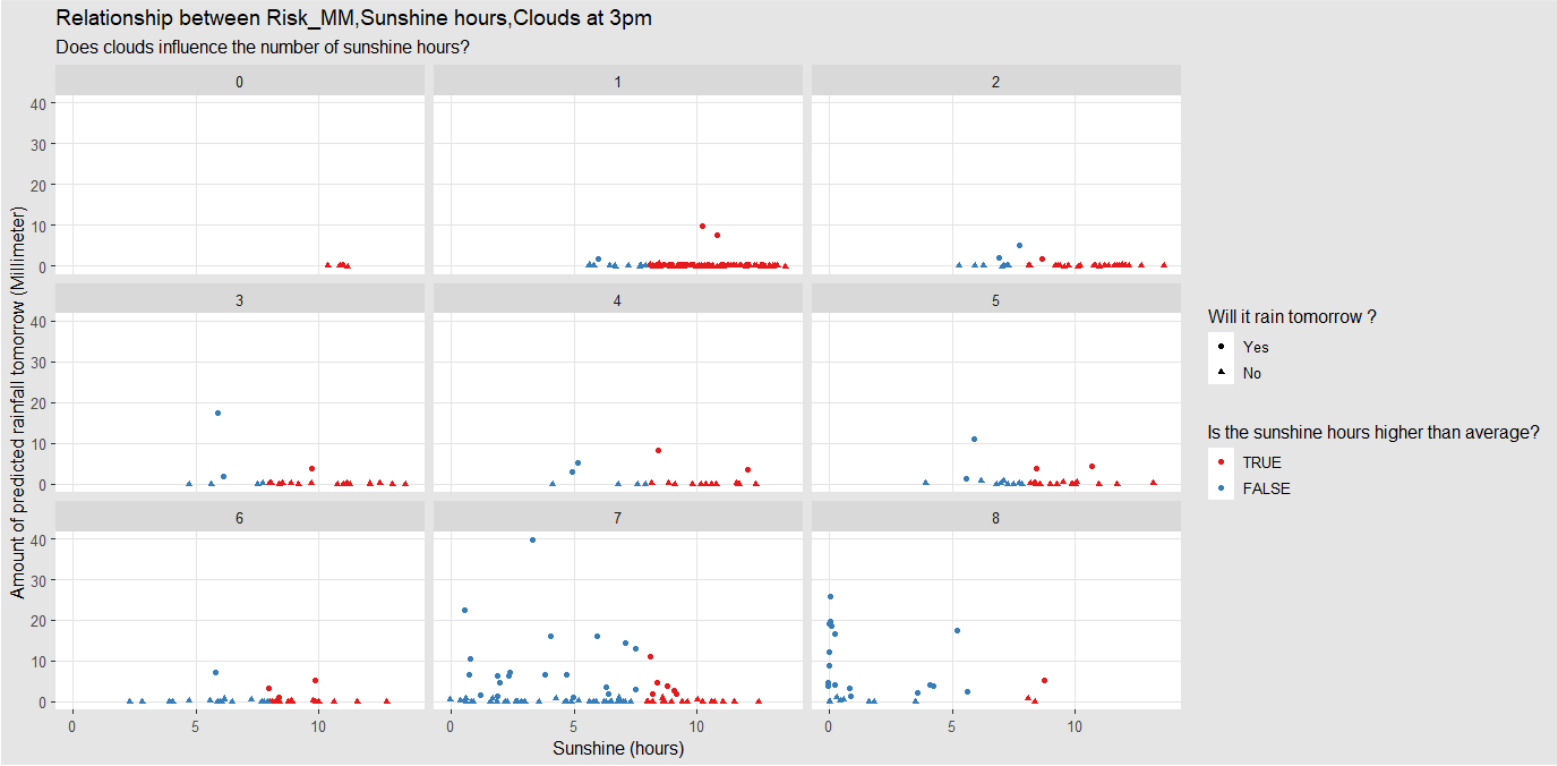


Figure 32 : Scatter Plot with facet (Clouds 3 pm) generated from R code snippet above

According to figure 32, it is clear that as the sky at 3 pm is cloudier, there are a greater number of sunshine hours that are below average. This confirms the assumption from previous analysis, where more clouds in the sky cause shorter hours of sunshine. However, the scatter plot above also shows that there are a few days that are predicted to rain tomorrow, although it has long sunshine hours and fewer clouds.

Analysis 3 : Determine the relationship between atmospheric pressure and rain tomorrow

```
# Density graph for pressure at 3pm
weatherData %>%
  ggplot(aes(Pressure_3pm)) +
  geom_density(aes(fill = RainTomorrow),alpha = 0.6) +
  labs(title = "Relationship between Pressure at 3pm and Rain Tomorrow",
       subtitle = "Does lower pressure at 3pm cause raining tomorrow?",
       x = "Pressure at 3pm (hPA)",
       y = "Density",
       fill = "Will it rain tomorrow? ") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_fill_brewer(palette = "Set2")
```

Figure 33 : R code snippet

Based on the R code snippet above, a density plot is used to visualize the atmospheric pressure at 3 pm for days that are predicted to rain tomorrow and days that are predicted to not rain tomorrow. The scale_fill_brewer() function allows customization of color in density plot by specifying a palette.

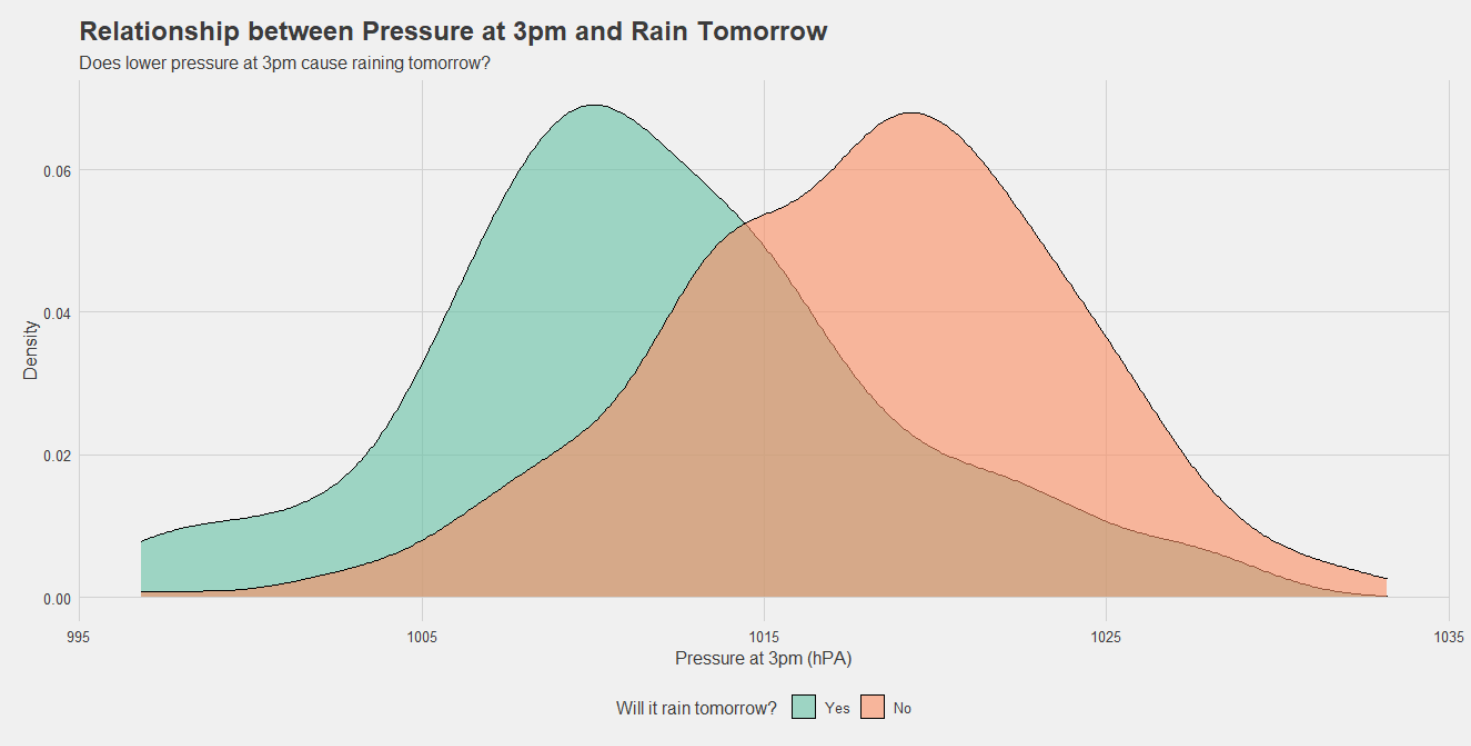
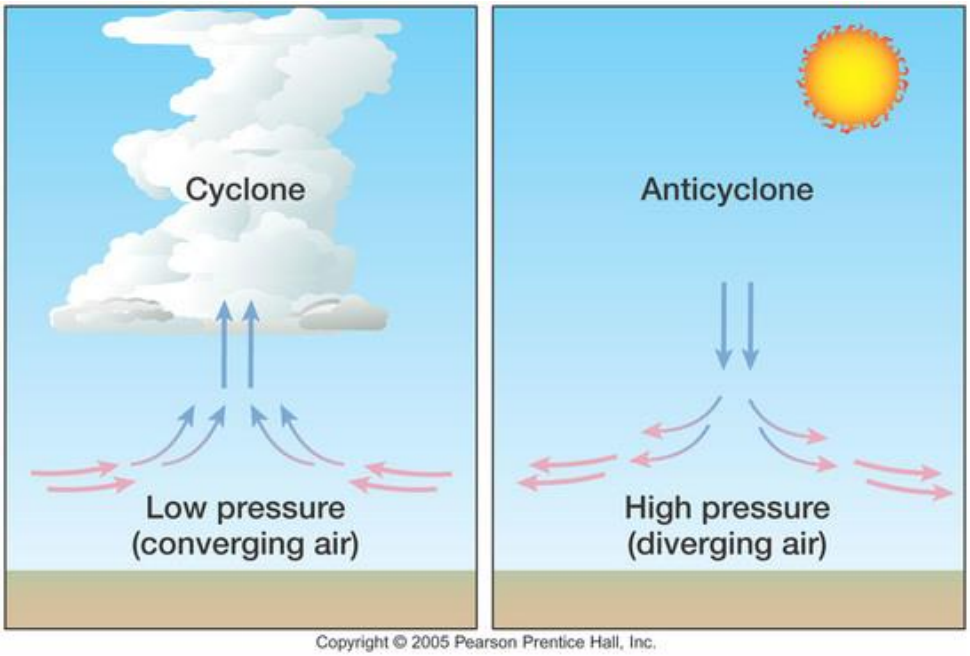


Figure 34 : Density Plot generated by R code snippet above

In figure 34, it is clear that there are two peaks shown by the density plot above. For days that are predicted to rain tomorrow, it will generally have a low atmospheric pressure system. On the other hand, the days that are not predicted to rain tomorrow, generally have a higher

atmospheric pressure system. To understand more about this phenomenon, the concept of air pressure should be understood.

In simple terms, air pressure is the force of the atmospheric air exerting on the ground (weather-station, 2018). Air pressure, also known as atmospheric pressure, can differ according to different conditions such as altitude, and temperature (weather-station, 2018). When an area experiences a low air pressure system, it means that the air is less dense in the atmosphere. This causes the air which is near the ground which has high pressure to rise into the atmosphere to balance out the air pressure (weather-station, 2018). This creates a rising motion of air particles. The air that rises will condense and form clouds, which will return to the ground as rain when the clouds grow bigger (weather-station, 2018). Therefore, a low-pressure system is always associated with unstable weather conditions such as raining, snowing, and storms.



\Figure 35 Illustration of the effects of the low-pressure and high-pressure system (THE GEOGRAPHER ONLINE, n.d.)

```
# group the data by rain tomorrow
pressureRain = weatherData %>%
  group_by(RainTomorrow) %>%
  summarise(minPressure = min(Pressure_3pm),
            avgPressure = mean(Pressure_3pm),
            maxPressure = max(Pressure_3pm)) %>% View
```

Figure 36 : R code snippet

Next, the R code snippet above groups the weather dataset by the RainTomorrow variable, and uses summarise() function to get the minimum, average and maximum pressure for the days that are predicted to rain tomorrow, and days that are not predicted to rain tomorrow.

	RainTomorrow	minPressure	avgPressure	maxPressure
1	Yes	996.8	1011.711	1027.9
2	No	997.5	1017.932	1033.2

Figure 37 : Table shows the pressure information grouped by RainTomorrow variable

From the table above, the maximum pressure recorded for the days that will rain tomorrow is 1027.9 hPa. On the other hand, the minimum pressure recorded for the days that are not predicted to rain tomorrow is 997.5 hPa. This means that if the pressure at 3 pm is higher than 1027.9 hPA, it can be said that it will not rain tomorrow based on the weather dataset. However, if the pressure at 3 pm is lower than 997.5 hPa, then it can be said that the chances of raining tomorrow are high. This is because the weather dataset only has recorded the lowest pressure at 3 pm for days that do not rain tomorrow is 997.5 hPa, therefore if the pressure plunges below that threshold, it has a very high chance of raining. Similarly, the highest pressure at 3 pm ever recorded is 1027.9 hPa for days that will rain tomorrow. Therefore, if the pressure goes higher than 1027.9 hPa, the chances of rain are very slim for tomorrow.

Analysis 4 : Determine how pressure influence rain tomorrow with sunshine and clouds at 3pm.

```
# Create a new column that determines whether sunshine is above average
weatherData %>%
  filter(!is.na(Sunshine)) %>%
  mutate(moreAvgSunshine = factor(Sunshine > mean(weatherData$Sunshine,na.rm = T),
                                  levels = c("TRUE","FALSE"))) %>%
  ggplot(aes(Sunshine,Pressure_3pm)) +
  geom_line(aes(color = RainTomorrow),size = 1) +
  labs(title = "Relationship between sunshine,pressure at 3pm, clouds at 3pm",
       subtitle = paste("Does atmospheric pressure causes rain tomorrow,",
                        "given high sunshine and low clouds formation?"),
       x = "Sunshine (hours)",
       y = "Pressure at 3pm (hPA)",
       color = "Will it rain tomorrow? ") +
  facet_wrap(~Cloud_3pm)+
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_color_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = c(0,2,4,6,8,10,12,14)) # scaling x-axis manually
```

Figure 38 : R code snippet

Based on the R code snippet, a line plot with sunshine at the x-axis, and pressure at 3 pm at the y-axis is plotted. Then, the data is also separated into categories, based on the number of clouds formed at 3 pm. It is done via the facet_wrap(~Cloud_3pm). The purpose of this plot is to show how atmospheric pressure, sunshine, and clouds influence each other and cause rain tomorrow.

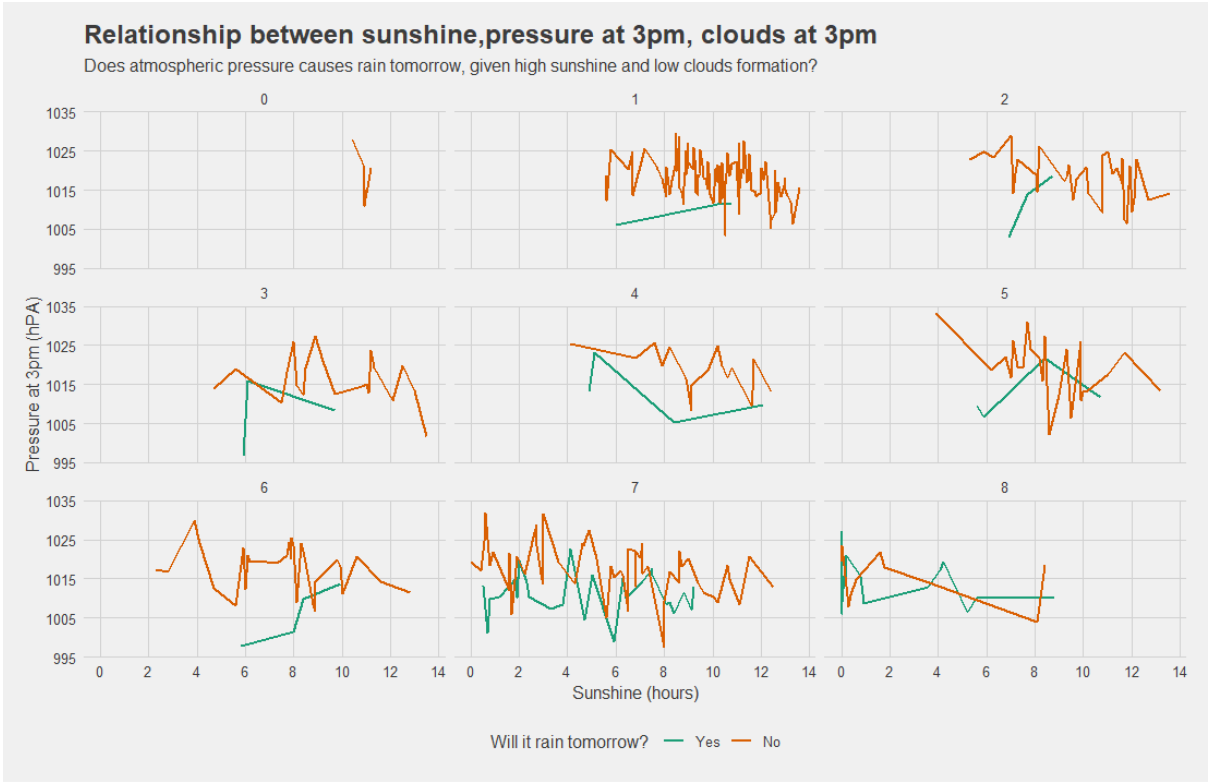


Figure 39 : Line plot generated from the R code snippet above

Based on the line plot above, the green line represents the days that are predicted to rain tomorrow, while the brown line represents the days that are not predicted to rain tomorrow. It can be seen as the green line stays below the brown line most of the time, especially for days with low clouds formation, such as 1okta, and 2 oktas. Therefore, it can be said that pressure at 3 pm does play a somewhat critical role in influencing whether it will rain tomorrow. This is because although some conditions are highly unfavorable to rain based on sunshine and clouds formation, such as there is a day which rains tomorrow with the only 1okta of clouds and more than 10 hours of sunshine. However, it can be seen as that day has a slightly lower atmospheric pressure than the average pressure for days that are not predicted to rain.

```
weatherData %>%
  group_by(Cloud_3pm,RainTomorrow) %>%
  summarise(minPressure = min(Pressure_3pm),
            avgPressure = mean(Pressure_3pm),
            maxPressure = max(Pressure_3pm)) %>% View
```

Figure 40 : R code snippet

Based on the R code snippet, the minimum, average, and maximum pressure is grouped by clouds formation and rain tomorrow variable, to get a better insight on the pressure details according to cloud formation.

	Cloud_3pm	RainTomorrow	minPressure	avgPressure	maxPressure
1	0	No	1010.9	1018.267	1028.0
2	1	Yes	1006.0	1009.733	1011.8
3	1	No	1003.3	1018.073	1029.6
4	2	Yes	1003.0	1011.833	1018.7
5	2	No	1006.3	1018.219	1028.9
6	3	Yes	996.8	1007.000	1015.8
7	3	No	1001.8	1016.343	1027.4
8	4	Yes	1005.3	1012.825	1023.1
9	4	No	1008.4	1018.700	1025.7
10	5	Yes	1006.8	1012.425	1021.8
11	5	No	1002.3	1019.048	1033.2
12	6	Yes	997.7	1005.675	1013.7
13	6	No	1006.8	1017.872	1030.0
14	7	Yes	998.9	1011.785	1027.9
15	7	No	997.5	1017.879	1031.9
16	8	Yes	1006.0	1013.628	1027.2
17	8	No	1004.0	1015.270	1023.5

Figure 41 Table shows the pressure information grouped by cloud formation and rain tomorrow

According to the table above, the minimum pressure for days that are predicted to rain tomorrow regardless of cloud formation is always below 1010 hPA. While there are some extreme values present in the dataset, such as the maximum pressure recorded for days that do not rain tomorrow is lower than the maximum pressure recorded for days that rain tomorrow with 8 oktas of clouds. However, it can be expected that the average pressure for days that rain tomorrow is lower than the average pressure for days that do not rain tomorrow regardless of cloud formation. Hence, the hypothesis from previous analysis, that lower pressure causes rain tomorrow still holds.

Conclusion (Question 1)

From the analysis above, it can be determined that pressure, sunshine, and clouds formation at 3 pm play a critical role in determining whether it will rain tomorrow. When there is a low-pressure system at 3 pm, short sunshine hours, and very cloudy skies, then the chances of raining tomorrow are high. On the other hand, if there is a high-pressure system at 3 pm, long sunshine hours, and clear skies, then the chances of raining tomorrow are low. For a general rule of thumb based on the analysis conducted, if the clouds formation at 3 pm is above 6 oktas, below average sunshine, and lower than 1010 hPa atmospheric pressure, then the chances of raining are high.

Question 2 : How to determine strong wind gusts that can cause damage?

In this question, it is trying to determine the characteristic of strong wind gust that can cause damage to building structure from the weather dataset. A wind gust is a sudden and second-long burst of wind followed by a lull (Oblack,2018).

Analysis 1 : Which direction produce strong wind gust?

Before starting this analysis, a function called "transformBeaufortScale" is created to categorize different wind gust speed into different categories. Beaufort wind force scale is a scale that helps to estimate wind forces without any proper wind instrument (Rutledge et al, 2011). It is created back in 1805, by Sir Francis Beaufort (Rutledge et al, 2011). In this case, the Beaufort scale can be used to describe the wind gust force, such as how strong is the wind gust, given the wind gust speed.

Wind Force	Description	Wind Speed			Specifications	Probable Wave Height		Sea State
		km/h	mph	knots		meters	Max	
0	**Calm**	<1	<1	<1	Smoke rises vertically. Sea like a mirror	--	--	0
1	**Light Air**	1-5	1-3	1-3	Direction shown by smoke drift but not by wind vanes. Sea rippled	0.1	0.1	1
2	**Light Breeze**	6-11	4-7	4-6	Wind felt on face; leaves rustle; wind vane moved by wind. Small wavelets on sea	0.2	0.3	2
3	**Gentle Breeze**	12-19	8-12	7-10	Leaves and small twigs in constant motion; light flags extended. Large wavelets on sea	0.6	1.0	3
4	**Moderate Breeze**	20-28	13-18	11-16	Raises dust and loose paper; small branches moved. Small waves, fairly frequent white horses	1.0	1.5	3-4
5	**Fresh Breeze**	29-38	19-24	17-21	Small trees in leaf begin to sway; crested wavelets form on inland waters. Moderate waves, many white horses	2.0	2.5	4
6	**Strong Breeze**	38-49	25-31	22-27	Large branches in motion; whistling heard in telegraph wires; umbrellas used with difficulty. Large waves, extensive foam crests	3.0	4	5
7	**Near Gale**	50-61	32-38	28-33	Whole trees in motion; inconvenience felt when walking against the wind. Foam blown in streaks across the sea	4.0	5.5	5-6
8	**Gale**	62-74	39-46	34-40	Twigs break off trees; generally impedes progress. Wave crests begin to break into spindrift	5.5	7.5	6-7
9	**Strong Gale**	75-88	47-54	41-47	Slight structural damage (chimney pots and slates removed). Wave crests topple over, spray affects visibility	7.0	10.0	7
10	**Storm**	89-102	55-63	48-55	Seldom experienced inland; trees uprooted; considerable structural damage. Sea surface largely white	9.0	12.5	8
11	**Violent Storm**	103-117	64-72	56-63	Very rarely experienced; accompanied by widespread damage. Medium-sized ships lost to view behind waves. Sea covered in white foam, visibility seriously affected	11.5	16.0	8
12	**Hurricane**	118+	73+	64+	Devastation. Air filled with foam and spray, very poor visibility	14+	---	9

Figure 42 : Table shows the Beaufort wind force scale (RMets Editor,2018)

The table above shows the Beaufort scale that describes the wind condition given the wind speed.

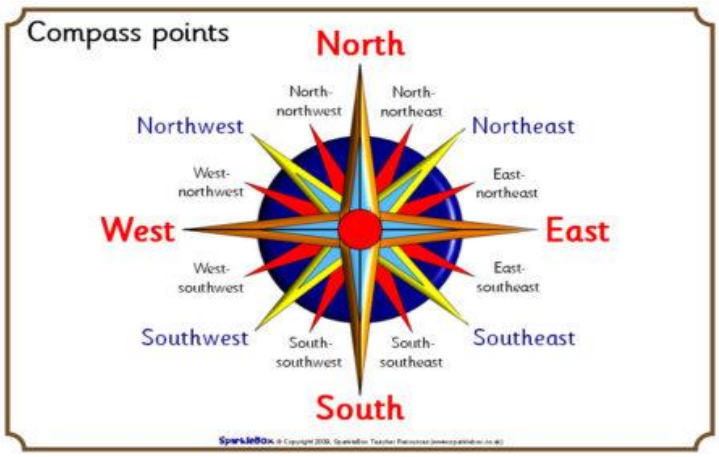


Figure 43 : Illustration of 16 compass point direction (sparklebox,n.d.)

Another thing that is worth mentioning is that the WindGust_Direction variable measures the strongest wind gust direction with a 16 compass point as shown in the figure above. Therefore, in the later plot, there are short abbreviations such as NNE which stands for north-northeast, and many more.

```
# Function name : transformBeaufortScale
# Description   : transform the wind speed into beaufort scale
# Parameter     : Vector containing wind speeds in KM/H
# Return        : Vector containing the beaufort scale for each wind speed passed

transformBeaufortScale = function (windSpeed) {
  dataReturn = c()
  index = 1

  while (index <= length(windSpeed)) {
    if (windSpeed[index] < 1) {
      dataReturn[index] = 0
    } else if (windSpeed[index] <= 5) {
      dataReturn[index]= 1
    } else if (windSpeed[index] <= 11) {
      dataReturn[index] = 2
    } else if (windSpeed[index] <= 19) {
      dataReturn[index] = 3
    } else if (windSpeed[index] <=28) {
      dataReturn[index] = 4
    } else if (windSpeed[index] <= 38) {
      dataReturn[index] = 5
    } else if (windSpeed[index] <= 49) {
      dataReturn[index] = 6
    } else if (windSpeed[index] <= 61) {
      dataReturn[index] = 7
    } else if (windSpeed[index] <= 74) {
      dataReturn[index] = 8
    } else if (windSpeed[index] <= 88) {
      dataReturn[index] = 9
    }else if (windSpeed[index] <= 102) {
      dataReturn[index] = 10
    }else if (windSpeed[index] <= 117) {
      dataReturn[index] = 11
    }else{
      dataReturn[index] = 12
    }
    index = index + 1
  }
  return (dataReturn)
}
```

Figure 44 : Function "transformBeaufortScale"

The figure above shows the function "transformBeaufortScale", which takes in a vector of wind speed and returns a vector of integer ranging from 0 to 12 based on the strength and speed of the wind. The numbers 0 to 12 represent the wind force in the Beaufort scale in figure 42.

```r
# Create new column that contains Beaufort scale that describe the wind gust speed
# Create new column that contains wind speed description based on Beaufort scale

data = weatherData %>%
  filter(!is.na(WindGust_Speed) & !is.na(WindGust_Direction)) %>%
  mutate(BeaufortScale = factor(transformBeaufortScale(WindGust_Speed)),
         WindDescription  = factor(BeaufortScale,
                                   levels = 0:12,
                                   labels = c(rep("Calm - Gentle Breeze",4),
                                              rep("Moderate Breeze",2),
                                              rep("Strong Breeze - Strong Gale",4),
                                              rep("Storm - Hurricane",3))))
```

Figure 45 : R code snippet

In this R code snippet, data transformation and manipulation are done. Firstly, all the missing values of WindGust_Speed variable and WindGust_Direction variable are filtered out. Then, a new column called “BeaufortScale” which contains integers that indicates the strength of the wind gust in the Beaufort scale. Next, another column called “WindDescription” is created to transform the values in the “BeaufortScale” column into factors. This is done by categorizing some values together, such as if the wind gust force in Beaufort scale is 0 to 3, then it will be in the category of “Calm – Gentle Breeze”, else, if the wind gust force in Beaufort scale is 4 to 5, then it will be in the category of “Moderate Breeze”.

```r
data %>%
  ggplot(aes(BeaufortScale)) +
  geom_bar(aes(fill = WindDescription)) +
  labs(title = "Relationship between wind gust speed and direction",
       subtitle = "Which wind gust direction is the most devastating ?",
       x = "How destructive is the wind gust ? (Beaufort Scale)",
       y = "Frequency",
       fill = "Description of wind force") +
  facet_wrap(~WindGust_Direction) +
  theme_stata() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_y_continuous(breaks = c(0,5,10,15,20,25)) +
  scale_fill_brewer(palette = "Reds")
```

Figure 46 : R code snippet

After data manipulation and data transformation, the data is plotted into a bar chart separated by wind gust direction.

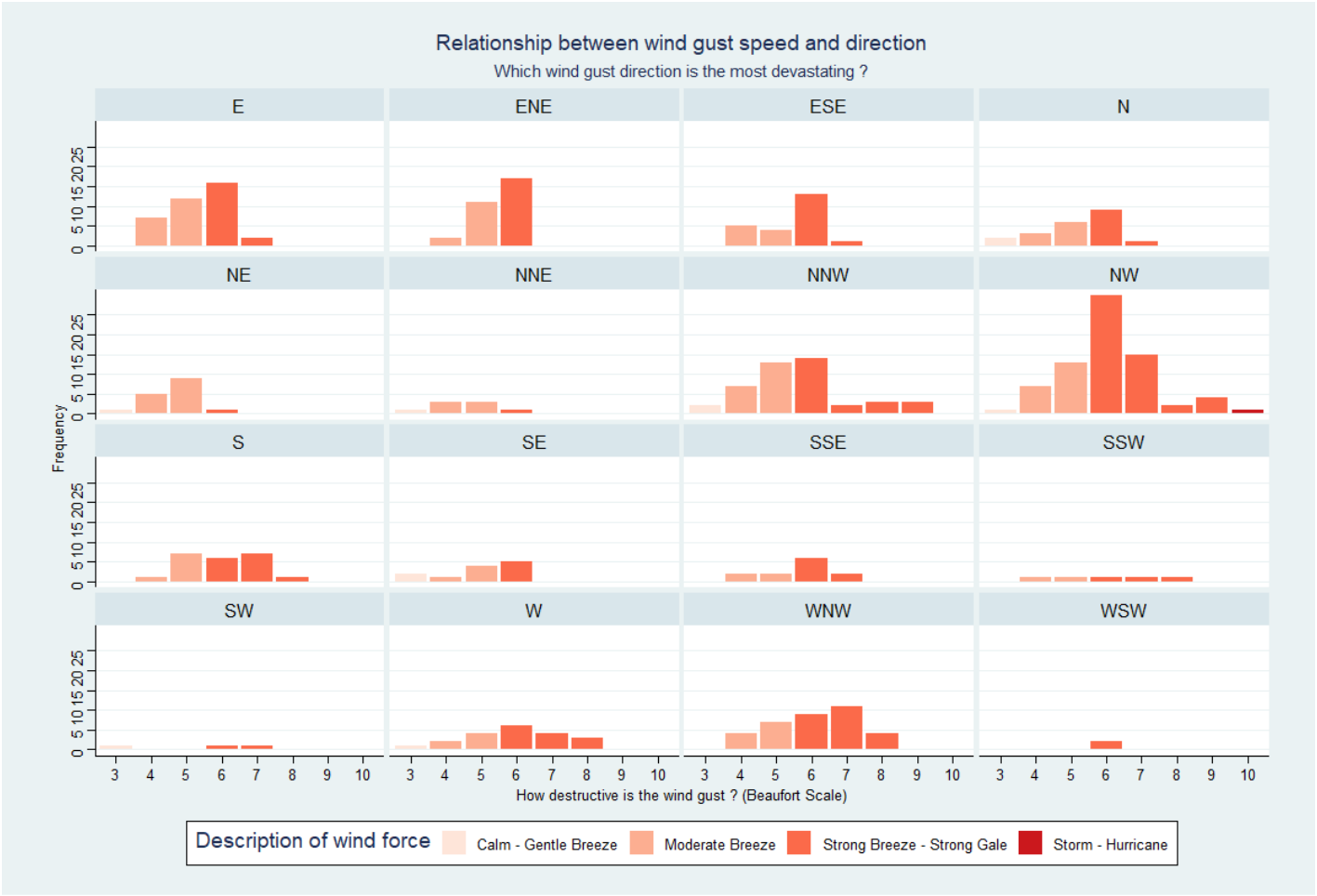


Figure 47 : Bar chart generated from the R code snippet above

From the figure above, there is only one occurrence of a very strong wind gust with a Beaufort scale of 10, equivalent to the wind force of a storm. The strong wind gust is produced from the Northwest direction. Other than that, the majority of the wind gust is categorized in “Strong Breeze-Strong Gale”. From the plot above, it can also be determined that only wind gust direction from south, west, north-northwest, west-northwest, north-west, and south-southwest can produce wind gust force of more than 7 Beaufort scale. Therefore, meteorologists and weather forecasters should be aware when expecting wind gusts from the direction mentioned above. This is because those wind gusts might be very strong based on the plot shown.

Analysis 2 : Determine the relationship between pressure and wind gust

This analysis, it is trying to determine whether wind gust and pressure have a relationship.

```
# Create new column that contains Beaufort scale that describe the wind gust speed
# Create new column that contains wind speed description based on Beaufort scale

# Use pressure at 9am
data = weatherData %>%
  filter (!is.na(WindGust_Speed)) %>% # filter out missing wind gust values
  mutate(BeaufortScale = transformBeaufortScale(WindGust_Speed),
         WindDescription  = factor(BeaufortScale,
                                   levels = 0:12,
                                   labels = c(rep("Calm - Gentle Breeze",4),
                                              rep("Moderate Breeze",2),
                                              rep("Strong Breeze - Strong Gale",4),
                                              rep("Storm - Hurricane",3))))
```

Figure 48 : R code snippet

Based on the R code snippet above, the missing values for the WindGust_Speed variable are filtered out. Then, similar to the previous analysis, a new column called “BeaufortScale” is created which contains the Beaufort Wind Force Scale, and another column called “WindDescription” groups the values into categories.

```
data %>%
  ggplot(aes(Pressure_9am,WindGust_Speed)) +
  geom_point(aes(color = WindDescription)) +
  geom_smooth(method = "lm") +
  labs(title = "Relationship between wind gust speed and pressure at 9am",
       subtitle = "Will atmospheric pressure influence wind gust speed?",
       x = "Pressure at 9am (hPA)",
       y = "Wind Gust Speed (KM/H)",
       color = "Description of wind force") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")
```

Figure 49 : R code snippet

The R code snippet above shows the code to plot a scatter plot with Pressure_9am as the x-axis, and WindGust_Speed as the y-axis. Geom_smooth() function is used here with “lm” as its method, this is to plot a linear model line across the scatter plot.

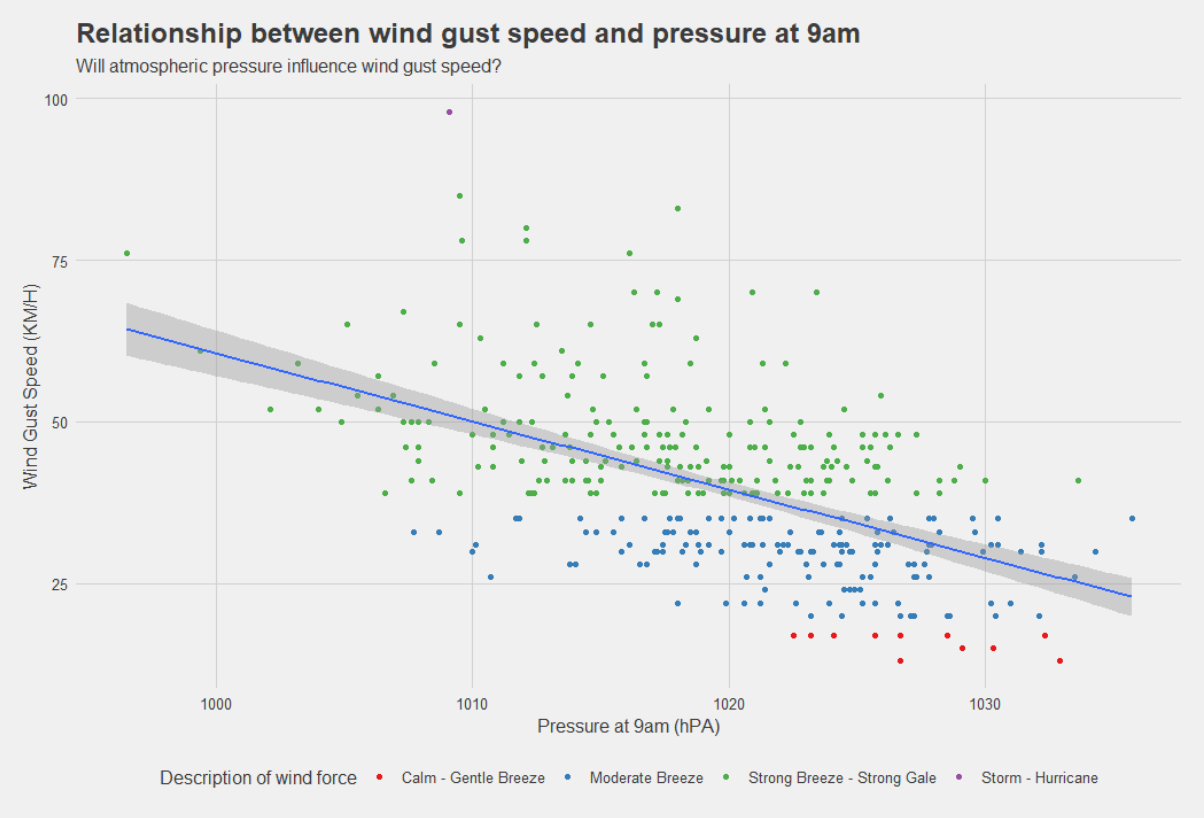


Figure 50 : Scatter plot generated from R code snippet above

A similar scatter plot is also created that uses Pressure_3pm as the variable for the x-axis instead of Pressure_9am. The code and plot are shown below.

```
# Use pressure at 3pm
data = weatherData %>%
  filter (!is.na(WindGust_Speed)) %>%   # filter out missing wind gust values
  mutate(BeaufortScale = transformBeaufortScale(WindGust_Speed),
         WindDescription  = factor(BeaufortScale,
                                   levels = 0:12,
                                   labels = c(rep("Calm - Gentle Breeze",4),
                                              rep("Moderate Breeze",2),
                                              rep("Strong Breeze - Strong Gale",4),
                                              rep("Storm - Hurricane",3))))

data %>%
  ggplot(aes(Pressure_3pm,WindGust_Speed)) +
  geom_jitter(aes(color = WindDescription)) +
  geom_smooth(method = "lm") +  # Using liner model for smooth function
  labs(title = "Relationship between pressure at 3pm and wind gust speed",
       subtitle = "Will atmospheric pressure influence wind gust speed?",
       x = "Pressure at 3pm (hPA)",
       y = "Wind Gust Speed (KM/H)",
       color = "Description of wind force") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")
```

Figure 51 : R code snippet

The code is shown in the figure above is very similar to the R code shown in figure 48 and figure 49. Explanations are provided in both of the figures above.

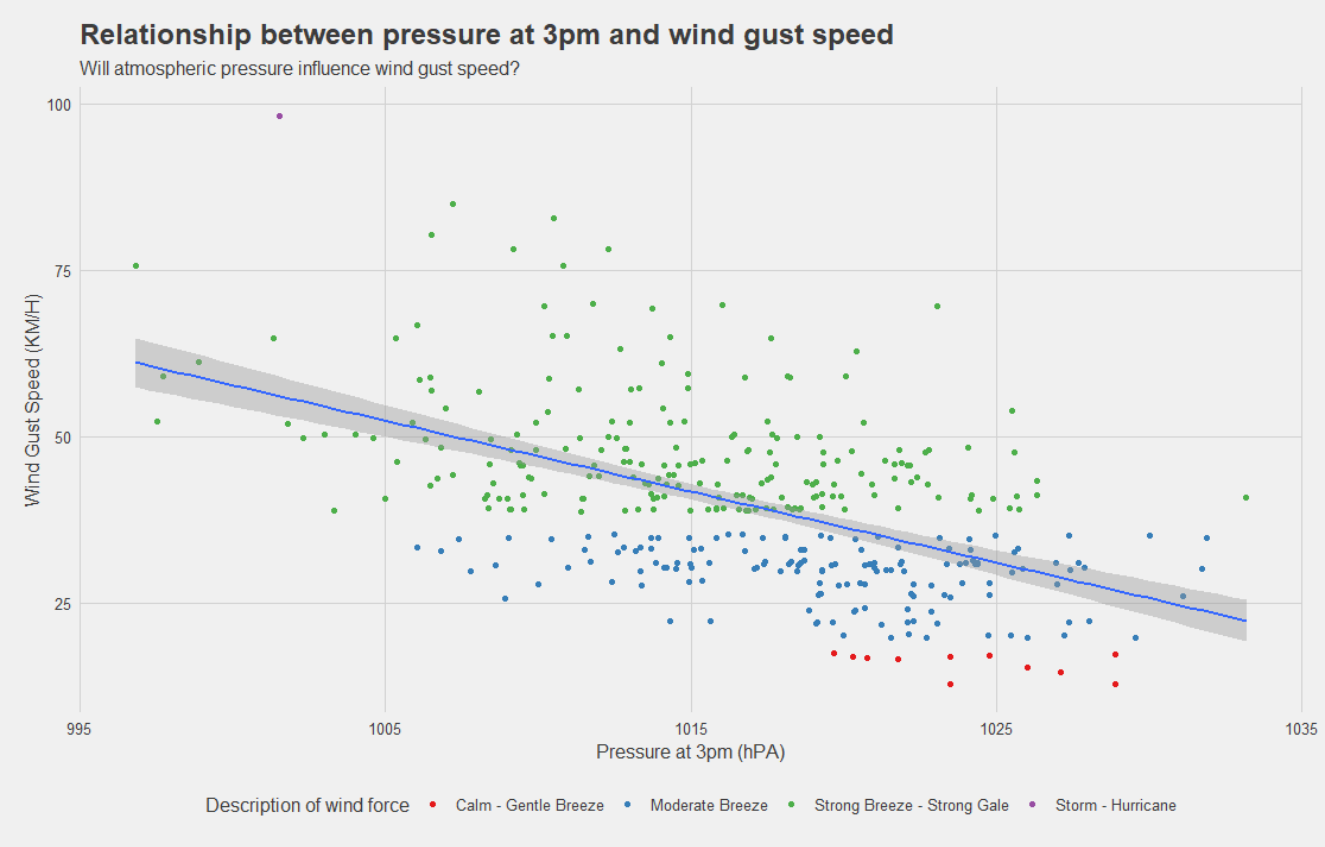


Figure 52 : Scatter plot generated from R code snippet above

Both of the scatter plots shown above exhibits a downward trend where the higher the pressure, the lower the wind gust speed. This is because the wind is generally the air movement from an area of a high-pressure system to low pressure system (Blaettler,2019). As the pressure decreases in a certain area, nearby areas with a high-pressure system will cause air to rush into the low-pressure system to balance out the pressure(Blaettler,2019).

Furthermore, the imbalance of pressure from the two areas is generally caused by the sun. This is because the sun cannot heat the ground evenly due to different geographical locations and causes some areas to be warmer than others (Blaettler,2019). The warmer regions air will heat up faster, and as the air becomes warmer, it becomes less dense and rises to the atmosphere. As more and more air near the ground rises the atmosphere, it creates an empty gap with fewer air particles, so this forms a low-pressure system. While on the other hand, the ground that heats up slower has more air particles near the ground, and this creates a high-pressure system. Naturally, these regions with more air particles will flow into the region with fewer air particles, therefore wind is created (Blaettler,2019).

Analysis 3 : Determine the relationship between wind speed and wind gust speed

This analysis is trying to find is there any positive relationship between wind speed and wind gust speed.

```
# Create new column that contains Beaufort scale that describe the wind gust speed
# Create new column that contains wind speed description based on Beaufort scale
# Create new column that contains average wind speed of the day

data = weatherData %>%
  filter(!is.na(WindGust_Speed),!is.na(WindSpeed_9am)) %>%
  mutate(BeaufortScale = transformBeaufortScale(WindGust_Speed),
         avgWindSpeed = (WindSpeed_9am + WindSpeed_3pm)/2 ,
         WindDescription  = factor(BeaufortScale,
                                   levels = 0:12,
                                   labels = c(rep("Calm - Gentle Breeze",4),
                                              rep("Moderate Breeze",2),
                                              rep("Strong Breeze - Strong Gale",4),
                                              rep("Storm - Hurricane",3))))
```

Figure 53 : R code snippet

Based on the R code snippet above, data manipulation and data transformation techniques are used. Firstly, the rows that contain the missing WindGust_Speed variable and the missing WindSpeed_9am variable is filtered out. Then a column called “BeaufortScale” is created that contains the Beaufort Wind Force scale for the wind gust speed, followed by a column “WindDescription” which groups the values into different categories. In addition to that, since WindGust_Speed variable records the highest wind gust speed occurred during a day and did not specify the time. Therefore, the average wind speed of the day is used for this analysis, the average wind speed is calculated and stored in a new column called “avgWindSpeed” as shown in the figure above.

```
data %>%
  ggplot(aes(avgWindSpeed,WindGust_Speed)) +
  geom_point(aes(color = WindDescription)) +
  geom_smooth(method = "lm") +
  labs(title = "Relationship between wind speed and wind gust speed",
       subtitle = "How wind speed influence wind gust speed?",
       x = "Average wind speed (KM/H)",
       y = "Wind gust speed (KM/H)",
       color = "Description of wind gust force") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(), axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")
```

Figure 54 : R code snippet

Moving on, the code above shows the R code for data visualization. A scatter plot with a linear model line is plotted. The scatter plot graph has average wind speed as its x-axis, and wind gust speed as its y-axis.

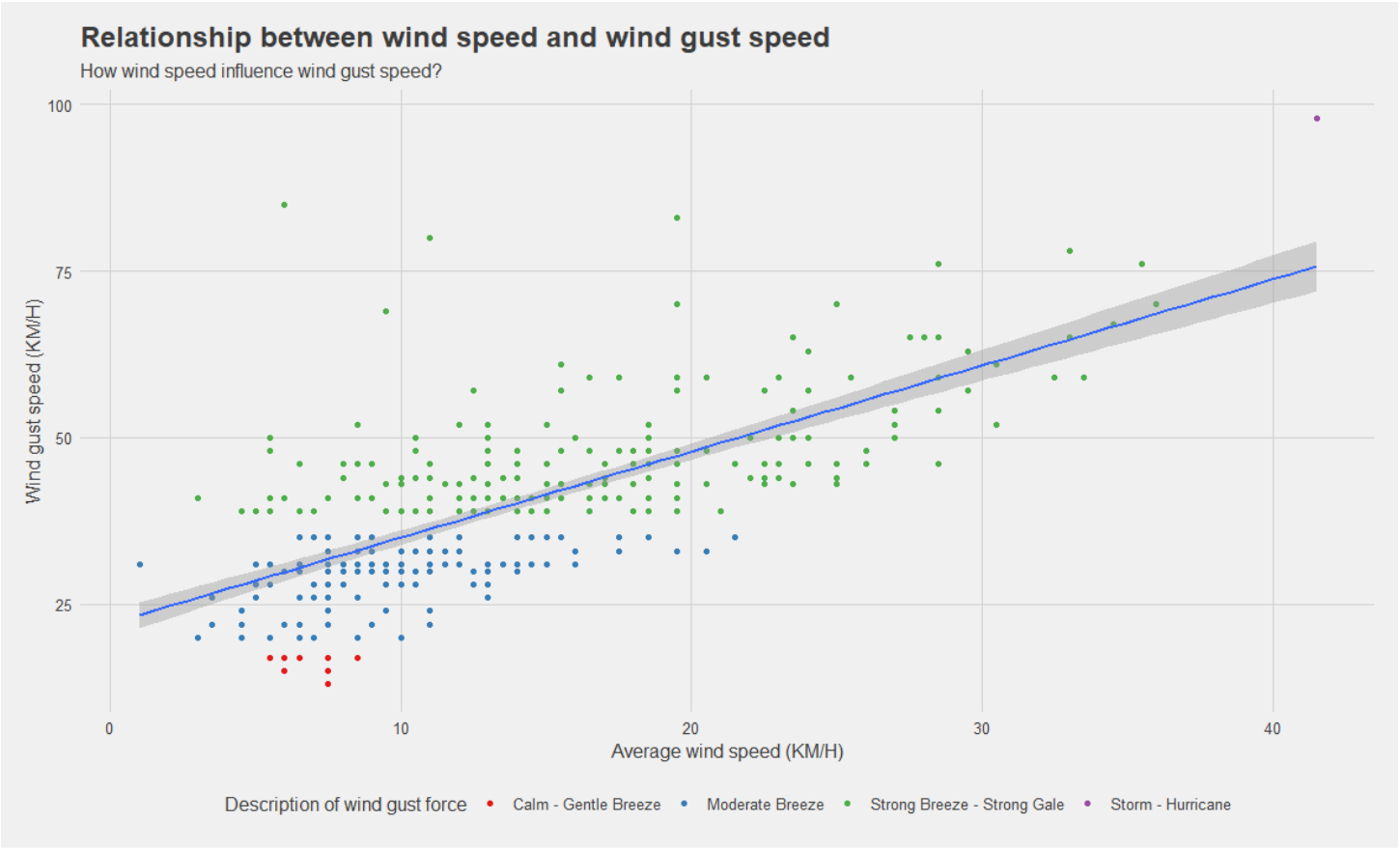


Figure 55 : Scatter plot generated from the R code snippet above

The scatter plot above shows that wind gust speed and wind speed have a positive linear relationship. This can be explained by the fact that wind gust and normal wind are very similar. Commonly, a wind gust is formed when the wind's path is blocked by obstacles such as buildings, hills, and trees. (Oblack,2018) These obstacles cause the wind to slow down and increases its friction. After the wind passes these obstacles, then its speed increases back rapidly, which causes gust. Another common factor is wind traveling through a small alley or gap (Oblack,2018). As all of the wind is forced through a small gap, this drastically increases their speed, which in turn causes gust (Oblack,2018).

Therefore, to relate to this finding, if the wind speed for the day is high, it will produce an even stronger wind gust. This is because wind gust happens when its wind path is blocked by obstacles as mentioned above.

In addition, it is known that most of the days experience an average wind speed of below 40 km/H. However, there is one day with an average wind speed slightly above 40KM/H, which experiences a wind gust with a speed near 100KM/H, which is equivalent to the wind force of a storm.

Conclusion (Question 2)

In conclusion, there are a few things gained from the analysis above. Firstly, wind gust direction from south, west, north-northwest, west-northwest, north-west, and south-southwest can produce wind gust force of more than 7 Beaufort scale. Therefore, meteorologists and weather forecasters should be aware of dangerous wind gusts from the directions specified above.

Next, the analysis also shows that the lower the atmospheric pressure, the higher the wind gust speed. This is because lower pressure causes more amount of wind to blow to that area, as the wind travels from the high-pressure region to the low-pressure region as discussed above.

Finally, from the analysis above, wind speed and wind gust speed have a positive relationship. This means that the faster the wind speed, the faster the wind gust speed produced. This is because wind gust is produced by the wind that is blocked by the obstacle. Another thing is that residents should be careful when the average wind speed reaches above 40km/h. This is because the analysis shows that 40km/h of wind can produce wind gust speed with force equivalent to a storm.

Question 3 : How to determine snowy days and their condition?

This question is trying to determine the days that have snowed in the weather dataset. Snowing is a phenomenon where the temperature of the atmosphere is cold enough to let the water vapor condense into ice crystals in the clouds (Shisia, 2017). On its own, ice crystals formed in the clouds are not heavy enough on their own to fall as snow, therefore many ice crystals in the clouds must collide with each other to form a bigger ice crystal also known as snowflakes (Shisia, 2017). Then, snowflakes are heavy enough to be pulled by gravity and eventually fall as snow(Shisia, 2017). It is also a myth that precipitation can only fall as snow when the temperature reaches below 0 degrees Celsius. Precipitation can fall as snow when the air temperature is below 2 degrees Celsius (Shisia, 2017).

Depending on the air temperature near the ground, there are two major types of snow formed, which are “dry snow” and “wet snow”. In simple terms, dry snow refers to when snowflakes fall through dry air, and the dry air causes the outer parts of snowflakes to dry up. This process weakens the bonds among the ice crystals that form the snowflake and results in less compact and light snow (Shisia, 2017). In contrast, wet snow refers to when snowflakes fall through warmer air and causes the outer part of the snowflake to melt. The snowflake then becomes watery on the outside and allows for more cohesion with water particles. This results in a bigger snowflake and more compact.

Since snowy conditions can be mainly determined from temperature, so in this analysis, the temperature will be mainly used as the target variable. According to the assumptions, the temperature will be in degrees Celsius.

Analysis 1 : Determine the number of days that are cold enough to snow

```
# Snowing can happens as low as 2 Celsius
# Create a new column that determines whether minimum
  # temperature is below 2 Celsius
# Group by the days that below 2 Celsius and above 2 Celsius



weatherData %>%
  mutate (SnowToday = factor(MinTemp<2,levels = c("TRUE","FALSE"))) %>%
  group_by(SnowToday) %>%
  summarise(count = n()) %>%
  ggplot(aes(SnowToday,count)) +
  geom_bar(aes(fill = SnowToday),stat = "identity",show.legend = FALSE) +
  geom_text(aes(label = count),vjust = -0.2) +
  labs(title = "Distribution of days",
       subtitle = "How many days were snowing ?",
       x = "Did it snow today? (Below 2 degree Celsius) ",
       y = "Frequency") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(), axis.title.y = element_text()) +
  scale_fill_brewer(palette = "Set2")
```

Figure 56 : R code snippet

Based on the R code snippet above, a new column called “SnowToday” is created with mutate() function. This column will be determining whether the minimum temperature of a day is less than 2 degrees Celsius, which is the temperature needed to allow precipitation to fall as snow. Next, by using group_by() and summarise(), the number of days that snowed and the number of days that did not snow are counted. Then, the data is plotted in a bar chart using ggplot2.

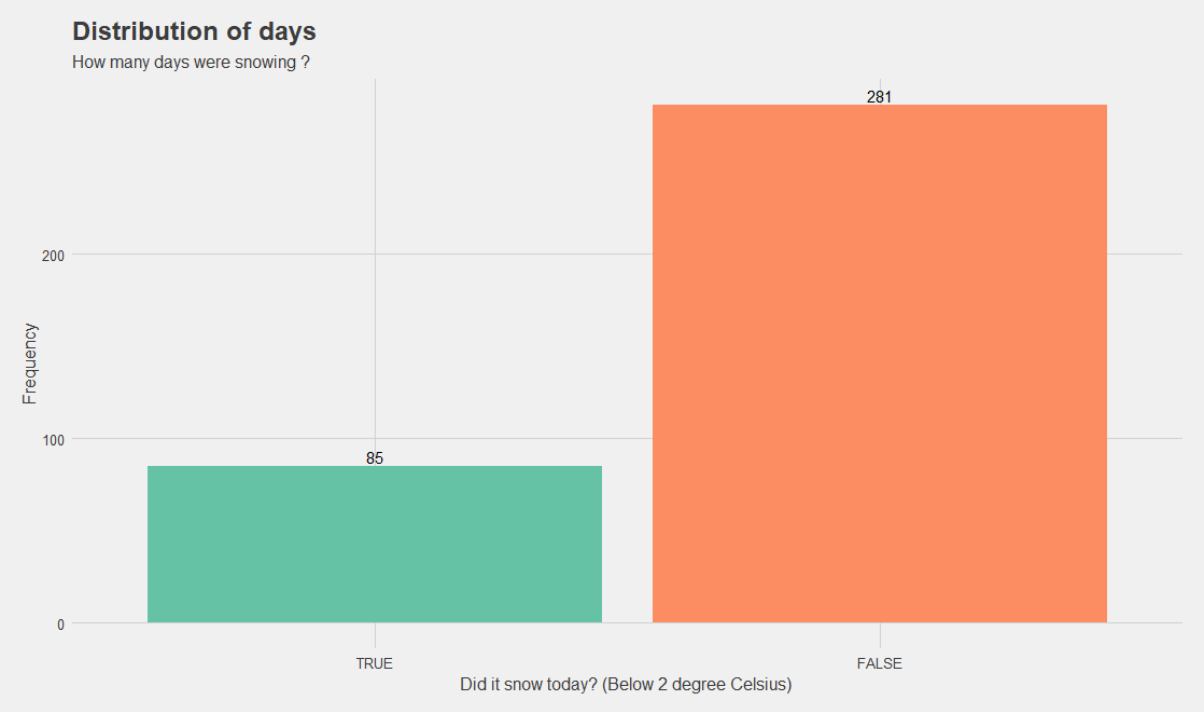


Figure 57 : Bar plot generated from R code snippet above

Based on the bar plot in the figure above, it shows that there are 85 days with minimum temperature below 2 degrees Celsius. This means that overall, 85 days have snowed for the given weather dataset, which is approximately 3 months.

Analysis 2 : Determine the temperature changes for snowy days

In this analysis, it is trying to get an answer to whether the temperature gradually increases throughout the day or decreases throughout the day. Through this, the time of snowing can be estimated.

```
# Determine whether 9am is the coldest temperature


# Create a new column that determines whether temperature has increased

weatherData %>%
  filter(MinTemp < 2) %>%
  mutate(tempIncrease = factor(Temp_3pm > Temp_9am)) %>%
  ggplot(aes(Temp_9am,Temp_3pm,color = tempIncrease)) +
  geom_point() +
  geom_smooth() +
  geom_hline(aes(yintercept = 2,
                 linetype = "2 degree Celsius (Snowing Temperature)"),
             color = "skyblue4", size = 1)+
  geom_vline(aes(xintercept = 2),
             color = "skyblue4", size = 1) +
  labs (title = "Temperature changes for snowy days",
        subtitle = "Temperature changes from 9am to 3pm",
        x = "Temperature at 9am (Celsius)",
        y = "Temperature at 3pm (Celsius)",
        linetype = "",
        color = "Temperature increased from 9am to 3pm") +
  theme_igray()
```

Figure 58 : R code snippet

Based on the R code snippet above, other rows that have a minimum temperature above 2 degrees Celsius are filtered out. This is because, in the previous analysis, 2 degrees Celsius is known as the maximum temperature to allow snowing. Then, a new column is created which determines whether the temperature at 3 pm is bigger than the temperature at 9 am. If yes, the column will contain “TRUE”, otherwise “FALSE”. Next, a scatter plot is plotted with the x-axis being the temperature at 9 am and the y-axis being the temperature at 3 pm. One additional thing is the plotting of horizontal and vertical lines to indicate the snowing temperature which is 2 degrees Celsius.

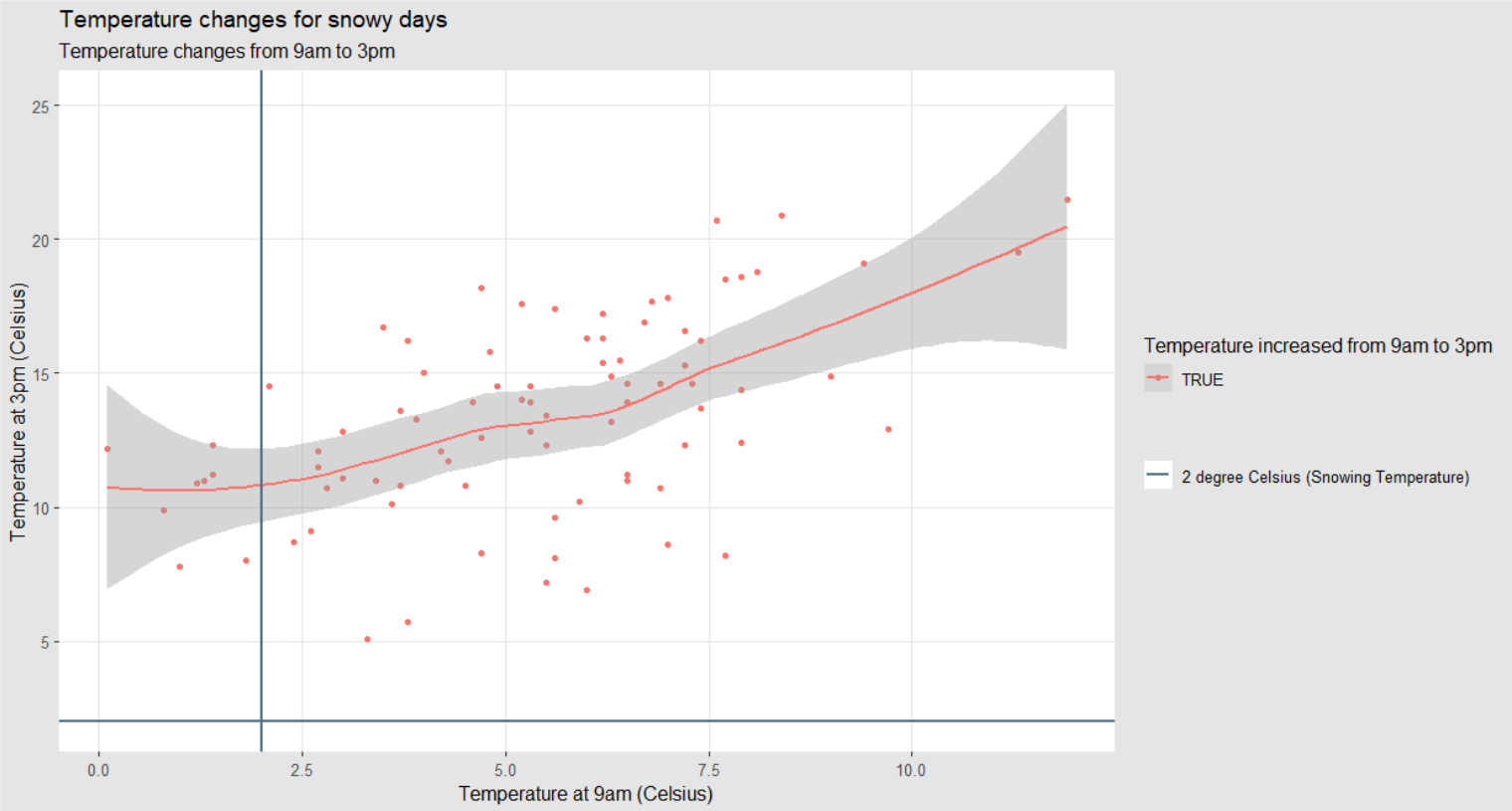


Figure 59 : Scatter plot plotted from R code snippet above

Based on the scatter plot above, one of the obvious trends is that all snowy days have increased temperature from 9 am to 3 pm. This implies that the temperature gradually increases throughout the day. In addition, the smooth line plotted geom_smooth indicates a linear relationship between temperature at 9 am and temperature at 3 pm. Furthermore, there are no days with temperatures below 2 degrees Celsius at 3 pm, as shown above with no point below the blue horizontal line. However, there is some point below 2 degrees Celsius at 9 am. This means that there is no chance of snowing in the evening, and only a few days that are still snowing at 9 am. From this plot, it can be determined that snowing happens early in the morning most of the time, at least before 9 am. The minimum temperature of the day is likely recorded during early morning hours or late at night.

Analysis 3 : Determine the humidity level for snowy days

In this analysis, it is trying to determine the relative humidity level for snowy days. Since the weather dataset only contains relative humidity levels for 9 am and 3 pm, so only the days that are below 2 degrees Celsius at 9 am will be used to determine the humidity level of snowy days. Relative humidity level at 3 pm is not used because it is determined the temperature at 3pm is too high for snow to happen.

```
# Filter out the days that has higher minimum temperature
# Use 9am scale since 9am is cooler than 3pm, more likely to snow
# Snow9am is a new column that determines whether temperature at 9am is
  # below 2 degree Celsius

weatherData %>%
  filter(MinTemp < 2) %>%
  mutate(Snow9am = factor(Temp_9am < 2)) %>%
  ggplot(aes(Temp_9am, Humidity_9am)) +
  geom_point(aes(color = Snow9am)) +
  geom_smooth(method = "lm",alpha = .5)+
  labs(title = "Study the moisture for snowy days",
       subtitle = "Relative Humidity at 9am for snowy days ",
       x = "Temperature at 9am (Celsius)",
       y = "Relative Humidity at 9am (%)",
       color = "Snowing at 9am") +
  theme_igray()
```

Figure 60 : R code snippet

According to the figure above, similar as the previous analysis, days that has more than 2 degree Celsius for minimum temperature is filtered out. This is because 2 degrees Celsius is the maximum temperature to allow for snow. Then, a new column named “Snow9am” is created, this column determines whether the temperature for that day at 9 am is below 2 degrees Celsius. Next, the scatter plot as shown below is plotted, with theme_igray() added for customization.

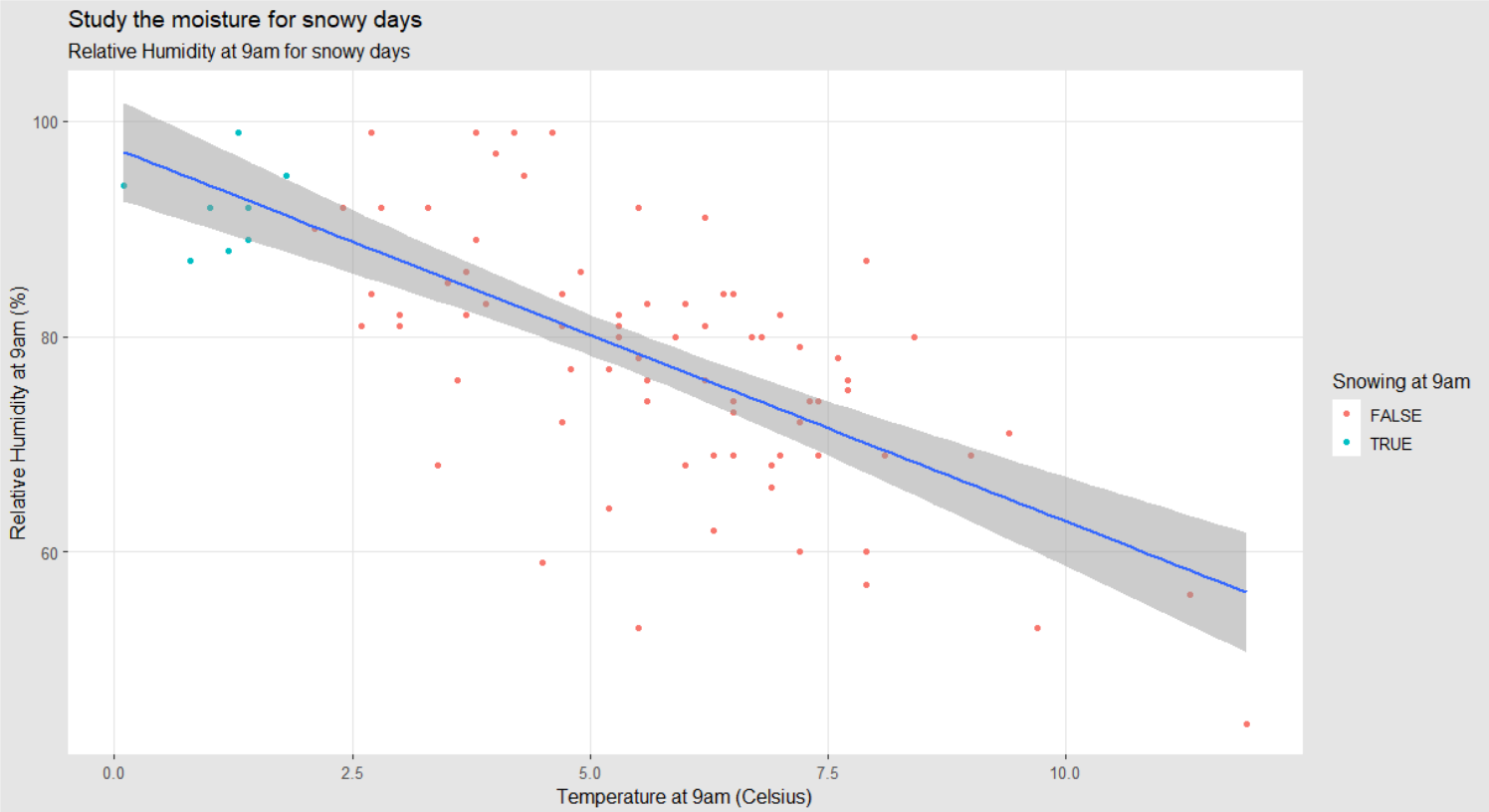


Figure 61 : Scatter plot plotted from R code snippet above

Based on the scatter plot above, there is a downward trend, where the higher the temperature at 9 am, the lower the relative humidity level. In addition, for the days that are still snowing at 9 am (below 2 degrees Celsius), it has a very high relative humidity level as illustrated with green points. Therefore, it can be hypothesized that days that are still snowing at 9 am are experiencing "Wet snow".

As discussed above, wet snow is a type of snow produced when the temperature near the surface is slightly warmer than 0 degrees Celsius (Shisia,2017). Hence, when snowflakes fall through the warmer air, the outer part of the snowflakes melts and becomes watery on the outside. The watery part will then further attract other water particles and form more cohesion between other water particles from the surrounding (Shisia, 2017).

Since the relative humidity level for days that are still snowing at 9 am is high, this implies that surrounding air has more water vapor, this further increases the effect of "Wet snow", which causes snow to be more compact and wet.

Analysis 4 : Determine whether there is a blizzard recorded

In this analysis, it is trying to find whether there is a day that suffers from a blizzard. According to (National Snow & Ice Data Center,2020), a blizzard is a hazardous weather condition, where a large amount of blowing and falling snow. It usually happens with a strong sustained wind speed of more than 56 kilometers per hour (35 miles per hour). Blizzard can create dangerous situations, such as causing trees to fall over due to strong wind, and frostbite to humans (National Snow & Ice Data Center, 2020).

Therefore, this analysis, it is trying to determine whether there is an occurrence of a blizzard by checking the wind speed for snowy days. Due to the weather dataset only contains the wind speed at 9 am, so only the days that are still snowing at 9 am are used to determine whether a blizzard has occurred.

```
# Plot it to determine wind speed for days that are still snowing at 9am
# Blizzard need a sustained wind speed of 56 KM/H

weatherData %>%
  filter(Temp_9am < 2,!is.na(WindSpeed_9am)) %>% # filter out missing values
  ggplot(aes(Temp_9am,WindSpeed_9am)) +
  geom_point() +
  geom_hline(aes(yintercept = 56,
                 linetype = "Minimum sustained wind speed to achieve blizzard"),
             color = "Red",size = 1) +
  labs(title = "Study wind speed at 9am for snowy days",
       subtitle = "Are there a blizzard recorded at 9am ?",
       x = "Temperature at 9am (Celsius)",
       y = "Wind Speed at 9am (KM/H)",
       linetype = "") +
  theme_igray() +
  scale_y_continuous(breaks = seq(0,60,10)) +
  scale_linetype_manual(values = c(4)) # changing the line-type
```

Figure 62 : R code snippet

Based on the R code snippet above, other days with temperatures above 2 degrees Celsius and rows with missing wind speed values are filtered out. Then, the scatter plot is plotted with the x-axis being the temperature at 9 am, and the y-axis being wind speed at 9 am. In addition, a horizontal line is also plotted to indicate the minimum wind speed to achieve a blizzard which is 56 KM/H. The scale_linetype_manual() function is called to give the horizontal line a different line style.

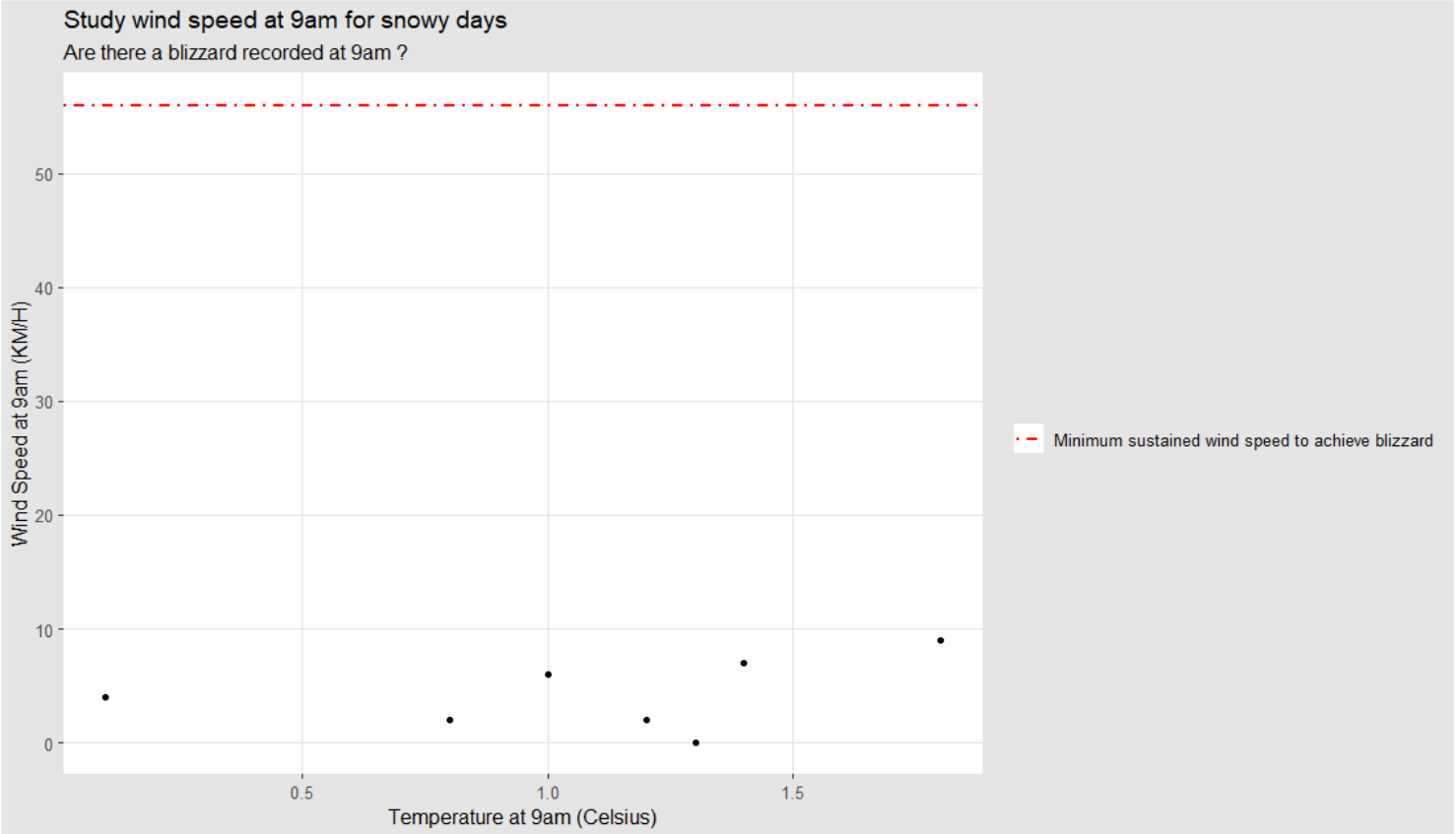


Figure 63 : Scatter plot generated from the R code snippet above

According to the scatter plot above, all the points are below the red dashed line. This means that there are no occurrences of a blizzard for days that are snowing at 9 am from the record. Therefore, it can be said that it is safe to go out and play with snow around 9 am.

Conclusion (Question 3)

In this analysis, it can be determined that snowing generally happens in the early morning before 9 am. In addition, no predicted blizzard occurred during 9 am given the wind speed data. Based on the given weather data, there are approximately three months of snowing

After performing the analysis above, a little information can be gained. One of the pieces of information is that snowing does not happen in the evening. This is because as shown in the analysis above, the temperature at 3 pm is too high for snowing. Therefore, snowing can occur during the early morning, or at night, when the atmosphere is cooler.

Besides, there is no blizzard recorded from the given weather dataset. As shown in the analysis above, a blizzard requires a sustained wind speed of 56 KM/H, but according to the wind speed data at 9 am for the days that are snowing, no wind speed is above 56KM/H.

Lastly, in one of the analyses above, it is also shown that there are only a few numbers of days that are still snowing at 9 am, judging by the temperature data given. However, all the days that are still snowing at 9 am are above 0 degrees Celsius, and below 2 degrees Celsius. This weather condition will cause the formation of “wet snow”, which is snow become more compact and stickier compared to dry snow, which is powdery. This implies that if a day is snowing at 9 am, there is a very high chance of the formation of “wet snow”, which is suitable for building a snowman, and having fun with the family in the snow(DeRoche,n.d.).

Question 4 : How to determine the days that are favorable for wildfires?

In this analysis, it is trying to find out the days that have favorable conditions for wildfires to grow. A wildfire can be said as an uncontrolled fire that burns in wildland and vegetation (National Geographic Society,2019). According to the Union of Concerned Scientists (2018), the wildfire situation is gradually worsening in the United States. There are roughly 100 more wildfires than the year before mainly due to climate changes. In addition, the cost to fight a wildfire is increasing as well (Union of Concerned Scientists,2018).

To combat wildfire, it is important to understand how weather can encourage and suppress the growth of wildfire. Three weather ingredients can influence wildfires, which are temperature, wind speed, and moisture in the air (Means,2018). If the temperature is very high, this gives enough heat to the surrounding materials, such as sticks, and leaves. This causes them to dry up and can become potential fuel for a fire (Means,2018). Next, wind speed can play a critical role in helping to spread the fire. Strong wind encourages the growth of the fire, by further drying up fuel sources, and carrying embers into the air to create additional fire. Finally, moisture in the air can suppress the growth of wildfire. When there is more moisture in the air, the potential fuel sources are less likely to be dry and ignite.

Therefore, this analysis will look in-depth at the temperature, humidity level, and wind speed of all the days recorded. Then it will try to predict which day has favorable conditions to spark a wildfire so that residents living near the area can be prepared.

Analysis 1 : Determine the hottest time of the day

This analysis is trying to determine the hottest time of the day. Since the weather dataset contains temperature that is measured at 9 am and 3 pm, this analysis will determine which time has the highest temperature, to aid further analysis later.

```
# Find whether temperature increase from 9am to 3pm for all the days
# Create a new column that determines whether temperature at 3pm is bigger than 9am

weatherData %>%
  mutate(tempIncrease = factor(Temp_3pm > Temp_9am)) %>%
  ggplot(aes(Temp_9am,Temp_3pm,color = tempIncrease)) +
  geom_point() +
  labs (title = "Temperature changes for all the days",
        subtitle = "Temperature changes from 9am to 3pm",
        x = "Temperature at 9am (Celsius)",
        y = "Temperature at 3pm (Celsius)",
        linetype = "",
        color = "Temperature increased from 9am to 3pm") +
  theme_igray() +
  scale_y_continuous(breaks = seq(0,40,5))
```

Figure 64 : R code snippet

In the figure above, a new column called “tempIncrease” is created to determine whether the temperature at 3 pm is higher than the temperature at 9 am. Then the data is plotted with a scatter plot, with Temp_9am variable is the x-axis and Temp_3pm is the y-axis.

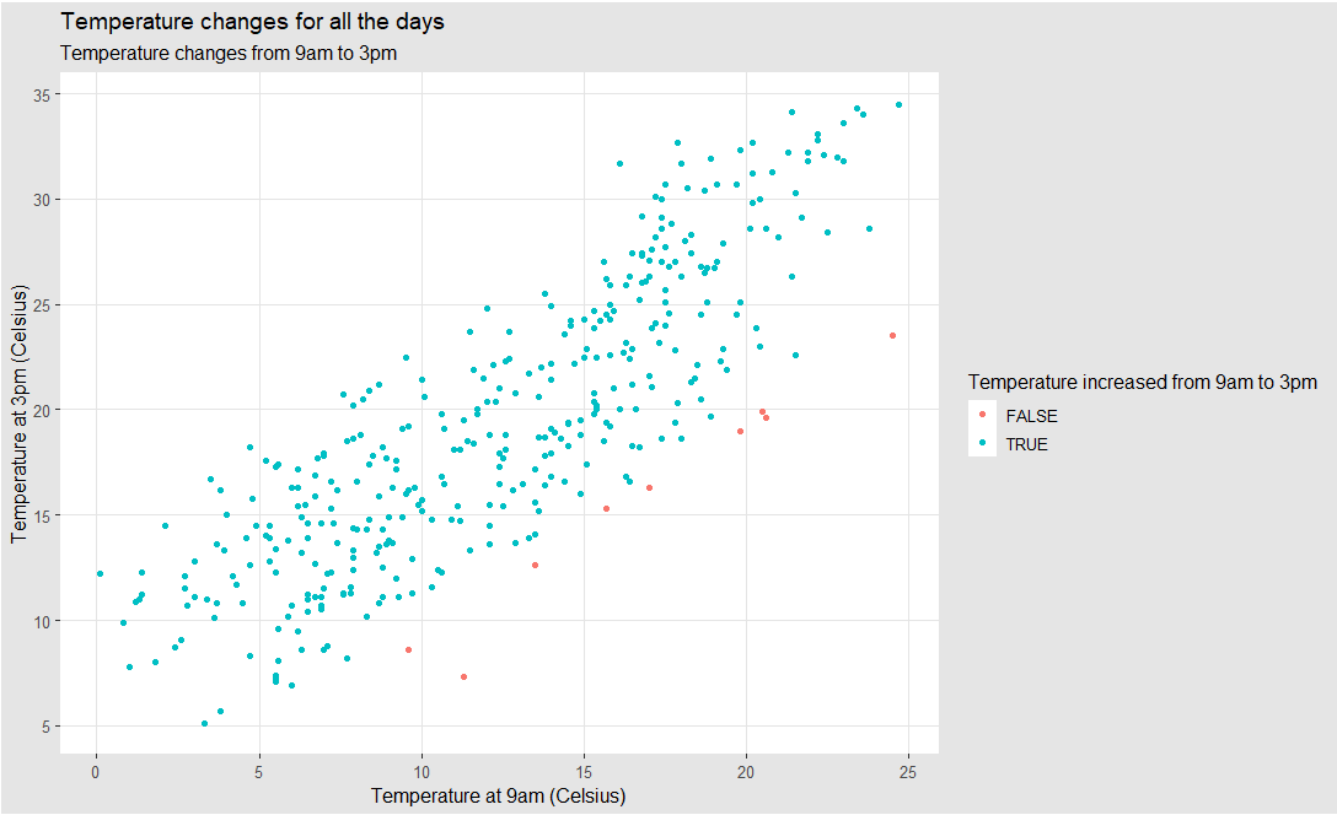


Figure 65 : Scatter plot generated by the R code snippet above

Based on the scatter plot above, it is clear that the majority of the days have a lower temperature at 9 am and a higher temperature at 3 pm. This might because during the morning, the temperature is usually cooler compared to the evening when the sun is on top of the sky supplying heat to the ground. Another trend is that the temperature at 9 am has a positive relationship with the temperature at 3 pm. This means the higher the temperature at 9 is, the hotter it will be during the evening.

From this analysis, it is known that the temperature in the evening is higher than the temperature in the morning. Therefore, further analysis will be focused on using the measurement recorded at 3 pm, since higher temperatures are prone to wildfire compared to the colder temperature, and using 3 pm measurement data can get accurate weather conditions during hot times.

Analysis 2 : Determine the relationship between temperature and humidity

In this analysis, it is trying to determine whether the humidity level decreases when temperature increase. This is because humidity also plays an important role in sparking a wildfire. Before beginning this analysis, it is important to understand about relative humidity level. In simple terms, the relative humidity level is the ratio of how much water vapor is present in the air against the total amount of water vapor the current air can hold (Chandler,2021). If the relative humidity level reaches 100%, this means the air is saturated and cannot hold more water vapor (Chandler,2021).

```r
# Function    : transformHumidity
# Description : transform humidity scale to humidity level
# Parameter   : Vector containing the humidity scale
# Return      : Vector containing the description of humidity level

transformHumidity = function (humidity) {
  transformedScale = c()
  index = 1
  while (index <= length(humidity)) {
    if (humidity[index] < 30) {
      transformedScale[index] = "Too dry"
    } else if (humidity[index] <= 60) {
      transformedScale[index] = "Healthy"
    } else {
      transformedScale[index] = "Too moist"
    }
    index = index + 1
  }
  return (transformedScale)
}
```

Figure 66 : R code snippet

Based on the R code snippet above, it shows a function called “transformHumidity”. The main purpose of this function is to categorize each relative humidity level into three categories relative to human comfort, which is “Too dry”, “Healthy”, “Too moist”. This is done so that the relative humidity level can be easily interpret compared to percentages which might not be intuitive at first glance. Furthermore, this categorization performed is based on a chart found in Aprilaire.com, which is attached as the figure below.

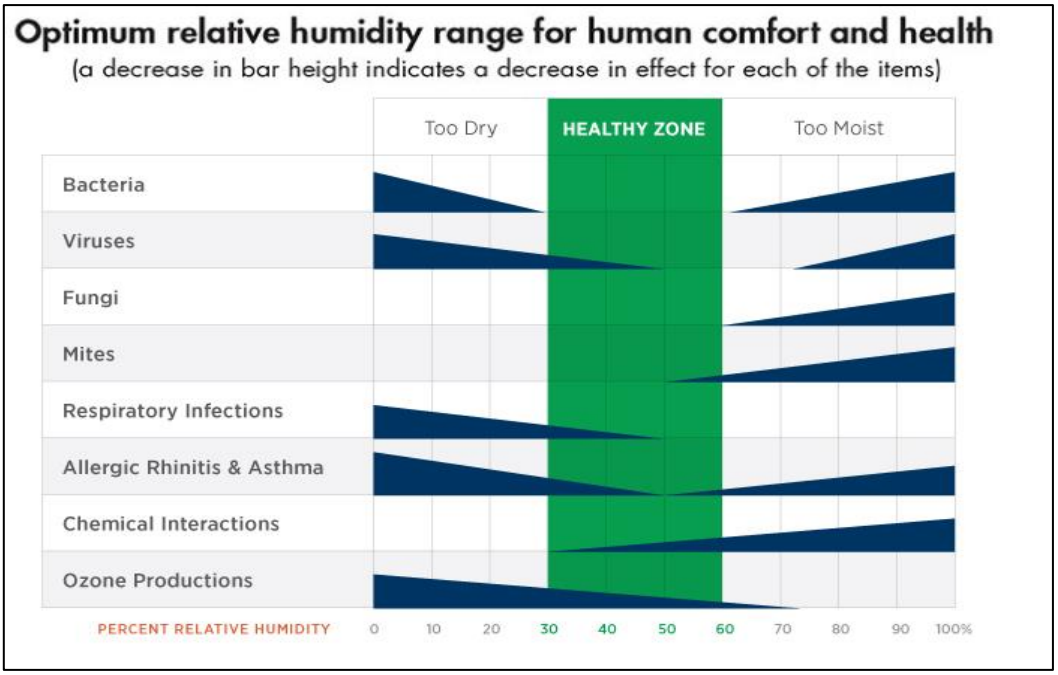


Figure 67 : Optimum relative humidity chart taken from (Aprilaire.com, n.d.)

Based on the chart above, when the relative humidity level is below 30%, it is too dry for human comfort, while more than 60% is too moist for human comfort.

```
# Use transformHumidity function to determine the humidity scale relative to human
  # comfort with three categories : "Too dry" , "Healthy", "Too moist"

# Temperature 3pm vs Humidity 3pm

weatherData %>%
  mutate(HumidityScale = factor(transformHumidity(Humidity_3pm),
                                levels = c("Too dry","Healthy","Too moist"))) %>%
  ggplot(aes(Temp_3pm,Humidity_3pm)) +
  geom_jitter(aes(color = HumidityScale)) +
  geom_smooth(method = "lm") +
  labs(title = "Relationship between temperature and humidity",
       subtitle = "Will temperature influence the relative humidity? ",
       x = "Temperature at 3pm (Celsius)",
       y = "Relative humidity at 3pm (%)",
       color = "Humidity scale") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(), axis.title.y = element_text()) +
  scale_x_continuous(breaks = seq(0,35,5)) +
  scale_y_continuous(breaks = seq(0,100,10)) +
  scale_color_brewer(palette = "Set1")
```

Figure 68 : R code snippet

In the R code snippet above, a new column called “HumidityScale” is created to categorize the relative humidity level at 3 pm into three categories using the transformHumidity() function. Afterward, the data is plotted into a scatter plot, with Temp_3pm as the x-axis, and Humidity_3pm as the y-axis. In addition, a linear model line is plotted to emphasize the relationship between temperature at 3 pm and humidity at 3 pm.

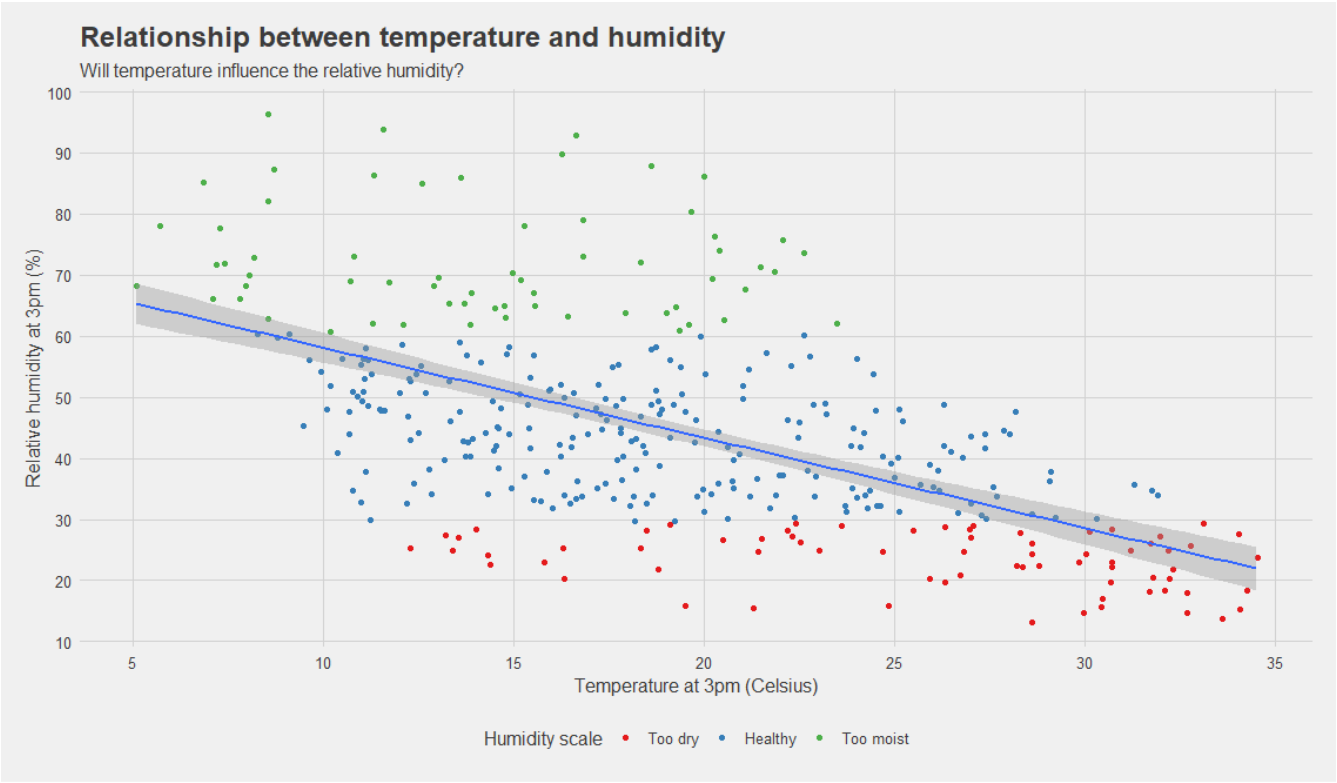


Figure 69 : Scatter plot generated from the R code snippet above

Based on the scatter plot above, the temperature at 3 pm has an inverse proportional relationship with relative humidity level. As shown above, the higher the temperature at 3 pm, the lower the humidity level as highlighted in red points, as “Too dry”.

This might because as the temperature increases, the amount of water vapor that can be held by air also increases(How Does Temperature Affect Humidity? - Optimum Humidity, n.d.). This causes the relative humidity to be lower if there is not much water being evaporated into the air at the same time (How Does Temperature Affect Humidity? - Optimum Humidity, n.d.). This is the case shown by the scatter plot above.

Furthermore, it can be said as it is not a good situation especially for days with high temperatures as the relative humidity is low. This can cause wildfires to spark easily, as lower humidity levels can cause water to evaporate quickly from branches and leaves into the atmosphere(Means,2018). This leaves them dry and more easily to be burned by fire (Means,2018).

Analysis 3 : Determine the relationship between sunshine and temperature

In this analysis, the relationship between sunshine and temperature is determined. Since sunshine is measured throughout the entire day, therefore the maximum temperature is used instead to perform the analysis.

```
# Use transformHumidity function to determine the humidity scale relative to human
  # comfort with three categories : "Too dry" , "Healthy", "Too moist"

weatherData %>%
  filter(!is.na(Sunshine)) %>%  # filter out missing sunshine values
  mutate(HumidityScale = factor(transformHumidity(Humidity_3pm),
                                levels = c("Too dry",
                                           "Healthy",
                                           "Too moist"))) %>%
  ggplot(aes(Sunshine,MaxTemp)) +
  geom_jitter(aes(color = HumidityScale)) +
  geom_smooth() +
  labs(title = "Relationship between sunshine and temperature",
       subtitle = "Will sunshine influence temperature ? ",
       x = "Sunshine (hours)",
       y = "Maximum temperature (Celsius)",
       color = "Humidity scale") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(), axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")
```

Figure 70 : R code snippet

In the R code snippet above, data manipulation is done by filtering out rows that contain missing sunshine values. Then, same as previous analysis, a new column “HumidityScale” is created to categorize the humidity values into different categories. Next, a scatter plot is plotted, with the x-axis represent the number of sunshine hours, and the y-axis represents the maximum temperature recorded.

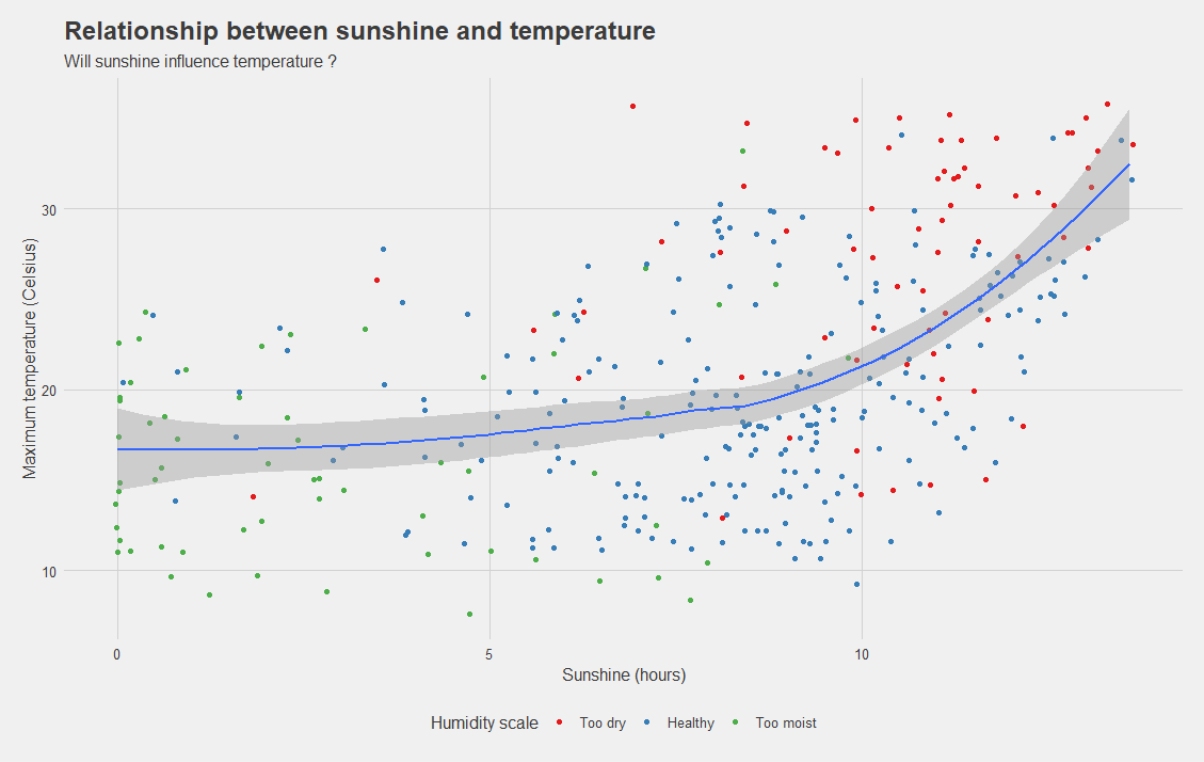


Figure 71 : Scatter plot generated from the R code snippet above

In the scatter plot above, the maximum temperature increases at a higher rate, once the sunshine hours recorded are more than 10 hours. Other than that, there are many days with low relative humidity levels when there are long hours of sunshine. This might because as long hours of sunshine provide more heat to the ground and surrounding. More heat absorbs by the ground increases the temperature, and therefore lowering the relative humidity level. This is due to a higher temperature will lead to lower relative humidity levels because the air can hold more water vapor.

Based on the scatter plot above, it can be determined that the risk of sparking wildfire increases when a day has more than 10 hours of sunshine. This is because more hours of sunshine provide more heat to the ground, trees, and bushes. A high amount of heat will cause wildfires to grow and burn vegetation easily.

Analysis 4 : Find the evaporation for hot days

In this analysis, the amount of evaporation is determined for hot days. Evaporation is important because it tells how dry the surrounding environment is. The drier it is, the more likely it is to ignite.

```
# Get the value for mean and 3rd Quartile for evaporation
summary(weatherData$Evaporation)
# Mean = 4.522
# Third Quartile = 6.400

# Get the value for 3rd Quartile for maximum temperature as reference
summary(weatherData$MaxTemp)
# Third Quartile = 25.5

# Use transformHumidity function to determine the humidity scale relative to human
  # comfort with three categories : "Too dry" , "Healthy", "Too moist"

data = weatherData %>%
  filter(MaxTemp > 25.5) %>%
  mutate(HumidityScale = factor(transformHumidity(Humidity_3pm),
                                levels = c("Too dry",
                                           "Healthy",
                                           "Too moist")))
```

Figure 72 : R code snippet

In the R code snippet above, the summary() function is called to determine the third quartile and mean of evaporation. This is to aid the plotting of lines later on in the scatter plot. Next, the summary() function is also called to get the value of the third quartile of maximum temperature. In addition, the third quartile of maximum temperature will be used as a threshold to determine whether a day is hot. If a day is above the third quartile of maximum temperature, then it is considered a hot day. Since this analysis will be solely focused on the evaporation on hot days, therefore days with a maximum temperature lower than 25.5 degree Celsius is filtered out. Then, the “HumidityScale” is a column that categorizes the relative humidity value.

```r
data %>%
  ggplot(aes(MaxTemp,Evaporation)) +
  geom_point(aes(color = HumidityScale)) +
  geom_hline(aes(linetype = "3rd Quartile", yintercept = 6.4),
             size = 1,color = "tan2") +
  geom_hline(aes(yintercept = 4.5,linetype = "Mean"),
             size = 1,color = "tan2") +
  labs(title = "Relationship between temperature and evaporation",
       subtitle = "Evaporation on hot days (>25.5 degree Celsius)",
       x = "Maximum temperature (Celsius)",
       y = "Evaporation (mm)",
       color = "Humidity scale",
       linetype = "") +
  theme_igray() +
  scale_color_brewer(palette = "Set1")
```

Figure 73 : R code snippet

According to the snippet above, a scatter plot with two lines will be plotted. The scatter plot will feature maximum temperature as the x-axis, and Evaporation as the y-axis. In addition, two horizontal lines are added to the scatter plot. The lines are used to show the mean value of evaporation and the third quartile value of evaporation.

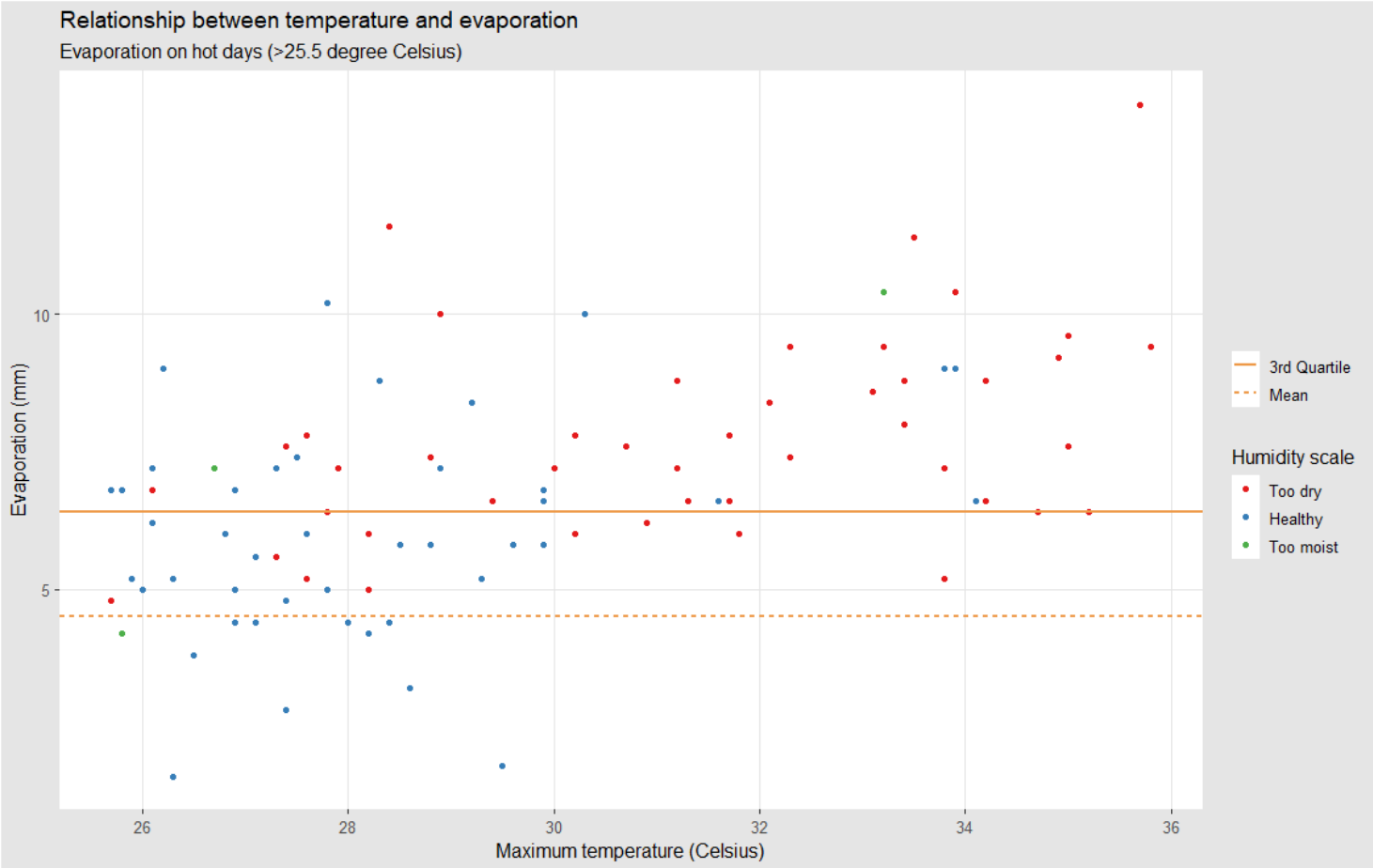


Figure 74 : Scatter plot generated from R code snippet above

Some information is shown by the scatter plot in the figure above. One of the information is that majority of the red point is above the 3rd quartile threshold line. This means that if the environment is drier, the higher the evaporation. Since particles always move through high saturation region to low saturation region, if the atmosphere is having low saturation of water content, this causes the water content from plants and trees to evaporate into the atmosphere (Means,2018). Therefore, drying up the plants more. Other than that, as the maximum temperature increases, evaporation also follows gradually.

Therefore, in this analysis, it can be spotted that there are days with highly favorable conditions for wildfires. For example, most of the days above 32 degree Celsius has evaporation higher than the 3rd quartile value, and the humidity level is dry (below 30%).

Analysis 5 : Determine the wind speed for hot days without rain

In this analysis, it is trying to find the wind speed for hot days that do not rain. This is because wind speed also plays a critical role in helping wildfire to spread. According to National Weather Service (NWS), official warnings about wildfires are made to the people in the affected area when a critical condition or red flag condition is forecasted (Means,2018). The red flag condition may differ but generally involves low humidity level, and wind speed higher than 32 km/h. (Means.2018). Therefore, the purpose of this analysis is to find the days with wind speeds more than 32km/h with low humidity and high temperature. In addition, the wind speed at 3 pm will be used for this analysis.

```
# get third quartile of temperature 3pm
summary(weatherData$Temp_3pm)
# 3rd quartile of Temp 3pm = 24


# only select days that has Temp_3pm above 3rd quartile and did not rain

data = weatherData %>%
  filter(Temp_3pm > 24, RainToday == "No") %>%
  mutate(HumidityScale = factor(transformHumidity(Humidity_3pm),
                                levels = c("Too dry",
                                           "Healthy",
                                           "Too moist")))
```

Figure 75 : R code snippet

Based on the R code snippet above, the summary() function is called to determine the third quartile value for the temperature at 3 pm. Next, days with temperatures less than 25 degrees Celsius, and days that are raining are filtered out. Similar to the previous analysis, the column “HumidityScale” categorizes the humidity value into three different categories.

```
# Danger condition is when dry and wind speed above 32 km/h

data %>%
  ggplot(aes(Temp_3pm,WindSpeed_3pm)) +
  geom_jitter(aes(color = HumidityScale)) +
  geom_hline(aes(yintercept = 32,linetype = "Wind speed that spreads wildfire easily"),
             color = "red3",size = 1) +
  labs(title = "Wind speed on hot days ",
       subtitle = "Determine the wind speed on hot days (more than 25.5 Celsius)",
       x = "Maximum temperature (Celsius)",
       y = "Wind speed at 3pm (KM/H)",
       color = "Humidity scale",
       linetype = "") +
  theme_igray()
```

Figure 76 : R code snippet

Next, a scatter plot that features temperature at 3 pm as the x-axis, and wind speed at 3 pm as the y-axis is plotted. A horizontal line is also plotted to indicate the wind speed which fulfills the red flag condition mentioned above, which is 32km/h.

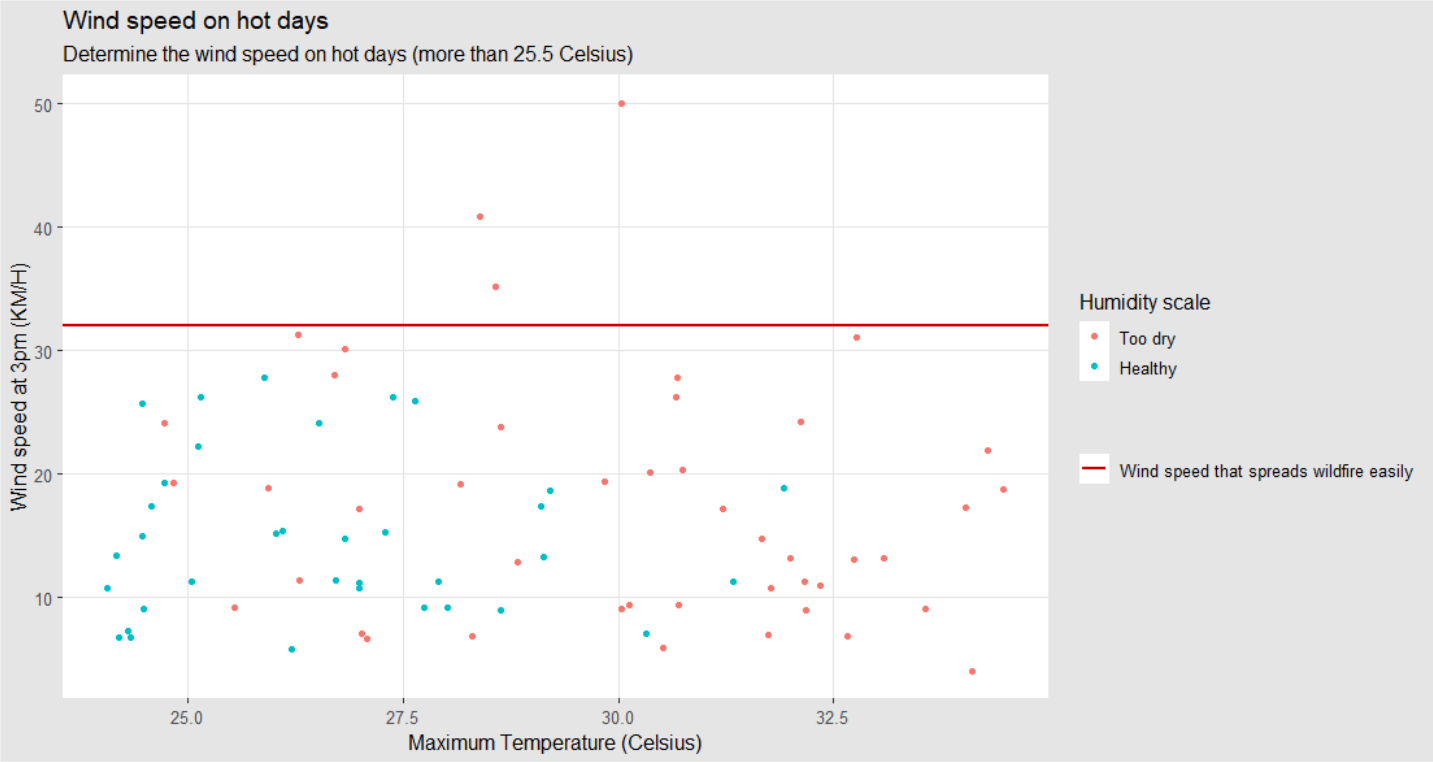


Figure 77 : Scatter plot generated from the R code snippet above

From the scatter plot above, there are only three days with wind speed above 32km/h. However, this does not imply that wildfire will not spark easily, instead of lower wind speed helps to retard the growth of wildfire. This is because a strong wind speed can help to spread wildfire, by blowing hot embers into surrounding areas which can potentially start another fire (Means,2018). Besides, strong wind removes moisture from the vegetation, increasing evaporation, and further dry up the fuel sources, such as grass, bushes, and trees (Means,2018). Wind also supplies rich oxygen to the wildfire that supports combustion.

Analysis 6 : Estimate the days that are likely to spark a wildfire easily

In this final analysis, a model is built to determine the likelihood of hot days to spark a wildfire.

```
# Function name : determineWildfire
# Description   : Determine the likelihood of sparking a wildfire given the weather condition

# Parameter     : Sunshine (Vector containing the sunshine for each day) ,

#                 Humidity (Vector containing the relative humidity at 3pm),

#                 WindSpeed
#                 (Vector containing the wind speed at 3pm),

#                 Rate of evaporation (Vector containing the evaporation for each day)

# Return        : Vector containing the percentage of sparking of wildfire for
#                 each day based on four condition

determineWildfire = function(Sunshine,Humidity,Windspeed,Evaporation) {
  WildfireChances = c()
  index = 1
  while (index <= length(Sunshine)) {
    percentage = 0
    if (Sunshine[index] >= 10) {    # sunshine above/near third quartile
      percentage = percentage + 25
    }
    if (Humidity[index] < 30) {     # Relative humidity less than 30% == too dry
      percentage = percentage + 25
    }
    if (Windspeed[index] >= 32) {  # wind speed >= 32 likely to cause wildfire
      percentage = percentage + 25
    }
    if (Evaporation[index] > 6.4) { # Evaporation more than 3rd quartile
      percentage = percentage + 25
    }
    WildfireChances[index] = percentage
    index = index + 1
  }
  return (WildfireChances)
}
```

Figure 78 : R code snippet

Based on the R code snippet above, the determineWildfire is a function used to calculate an estimate of the likelihood of sparking a wildfire. Four variables are used as a predicting variable to determine whether the weather condition is favorable for a wildfire. The four variables are sunshine, humidity level, wind speed, and evaporation. Generally, if the weather condition is very dry, sunny, and windy, the chance of sparking a wildfire is high. Therefore, to calculate a rough estimate, the determineWildfire function uses a simple “if structure” and each factor, which are the four variables mentioned just now, contributes 25% if the value for each of the variables met a certain threshold. The sunshine hours are set to more than 9 hours, if a hot day has more than 9 hours of sunshine, it is categorized as a hot sunny day which will contribute 25% of the likelihood of sparking a wildfire. The same goes for relative humidity level at 3 pm, which is set to below 30%, wind speed at 3 pm, which is set to above 31km/h, and

evaporation, which is set to more than its third quartile value (6.4). If all four factors met the threshold value specified, that day will be categorized as very likely to spark a wildfire.

```
# Wildfire prediction is done on days that are :
  # above 25.5 Celsius maximum temperature,
  # did not rain

# LikeliHood_Wildfire contains the values returned by determineWildfire model

# Those values are categorize into five categories :
#  "Very likely"   (4 values met the condition)
#  "Likely"        (3 out of 4 values met the conditions)
#  "Neutral"       (2 out of 4 values met the conditions)
#  "Unlikely"      (1 out of 4 values met the conditions)
#  "Very Unlikely" (0 put of 4 values met the conditions)

# 4 values are sunshine, humidity 3pm, wind speed 3pm, and evaporation

data = weatherData %>%
  filter(!is.na(Sunshine),RainToday == "No",MaxTemp > 25.5) %>%
  mutate(Likelihood_Wildfire = factor(determineWildfire(Sunshine,
                                                        Humidity_3pm,
                                                        WindSpeed_3pm,
                                                        Evaporation),
                                      levels = c(100,75,50,25,0),
                                      labels = c("Very Likely",
                                                 "Likely",
                                                 "Neutral",
                                                 "Unlikely",
                                                 "Very Unlikely")) )
```

Figure 79 : R code snippet

In the R code snippet above, data manipulation and transformation are done. For the rows that have missing sunshine values, days that are raining, and days with maximum temperature below 25.5 Celsius, they are all filtered out. This is because the model is only built to check for hot days, which is assumed to have a maximum temperature of more than the third quartile of the MaxTemp variable.

```
data %>%
  ggplot(aes(Temp_3pm,MaxTemp)) +
  geom_jitter(aes(color = Likelihood_Wildfire)) +
  labs (title = "Rough estimate of the likelihood of wildfire",
        subtitle = "Determine the chances of wildfire, for hot days (> 25 degree Celsius)",
        x = "Temperature at 3pm (Celsius)",
        y = "Maximum temperature (Celsius)",
        color = "Chances of sparking a wildfire") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")
```

Figure 80 : R code snippet

Next, the data is plotted into a scatter plot. The scatter plot features temperature at 3 pm as the x-axis variable, and maximum temperature as the y-axis variable. The points are color-coded so that they can be easily interpreted with the probability of sparking a wildfire.

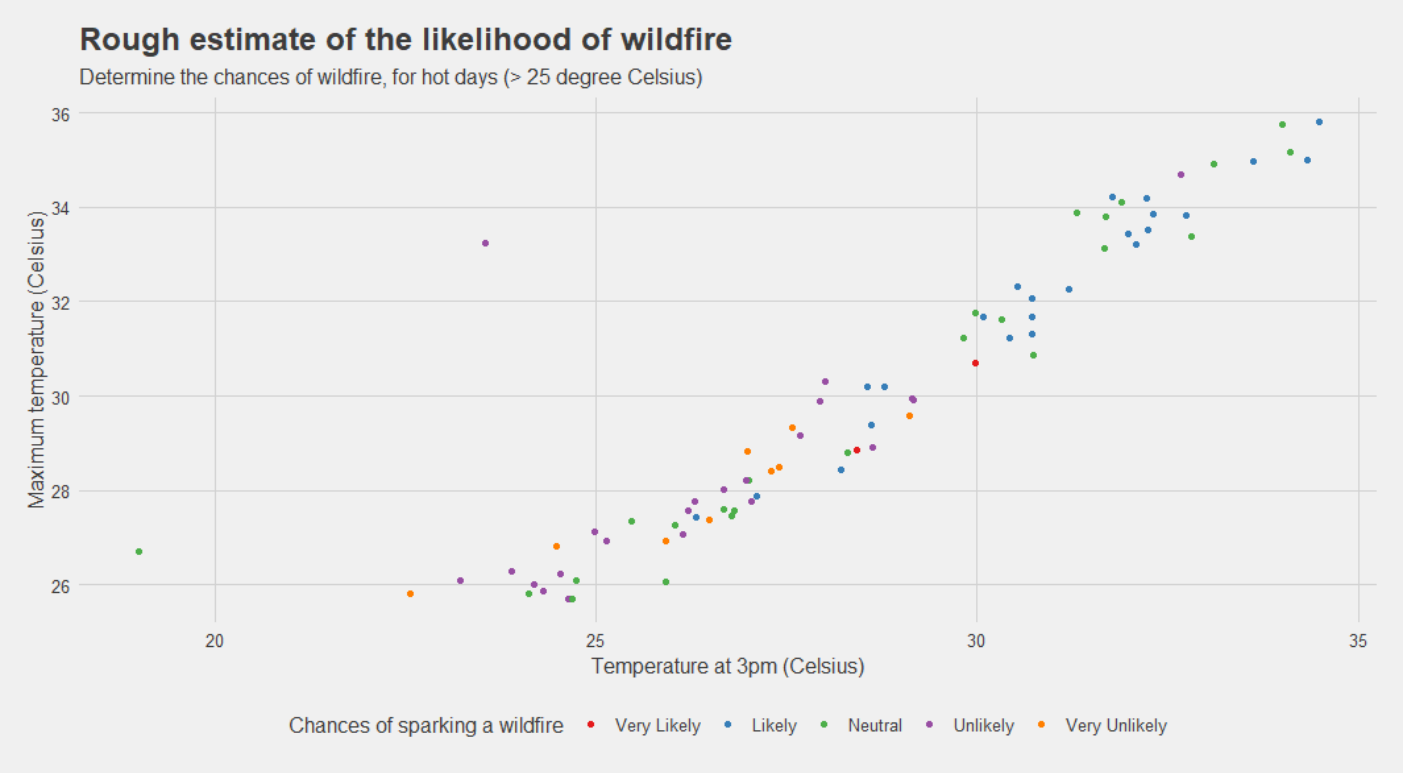


Figure 81 : Scatter plot generated from the R code snippet above

According to the scatter plot above, there are very few days with very likely chances of sparking a wildfire, which is good news. This means there are fewer days that fully fulfill all the four conditions stated in the determineWildfire model. One of the reasons is due to subpar windspeed as discovered in the previous analysis, where most of the hot days do not have higher than 32KM/H sustained wind speed, which retards the growth of wildfire. Furthermore, another information is that when the temperature at 3 pm are over 30 degree Celsius, majority of the days are either likely or neutral to spark a wildfire, compared to the temperature at 3 pm below 30 degree Celsius.

```r
# Plot the distribution
data %>%
  group_by(Likelihood_Wildfire) %>%
  summarise(count = n()) %>%
  ggplot(aes(Likelihood_Wildfire,count)) +
  geom_bar(aes(fill = Likelihood_Wildfire),stat = "identity",show.legend = F) +
  geom_text(aes(label = count),vjust = -0.2) +
  labs(title = "Distribution of likelihood to spark wildfire",
       subtitle = "How many days are likely to spark wildfire?",
       x = "Chances of sparking a wildfire",
       y = "Frequency") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_fill_brewer(palette = "RdYlGn")
```

Figure 82 : R code snippet

In the R code snippet above, data manipulation is done by using the group_by() function and summarise() function. It groups the data with the Likelihood_Wildfire column and uses summarise() to count the number of rows for each category. Then, a bar chart is plotted to shows the distribution of the likelihood to spark a wildfire. Geom_text() function is used to add a label on top of each bar to indicates the total count.

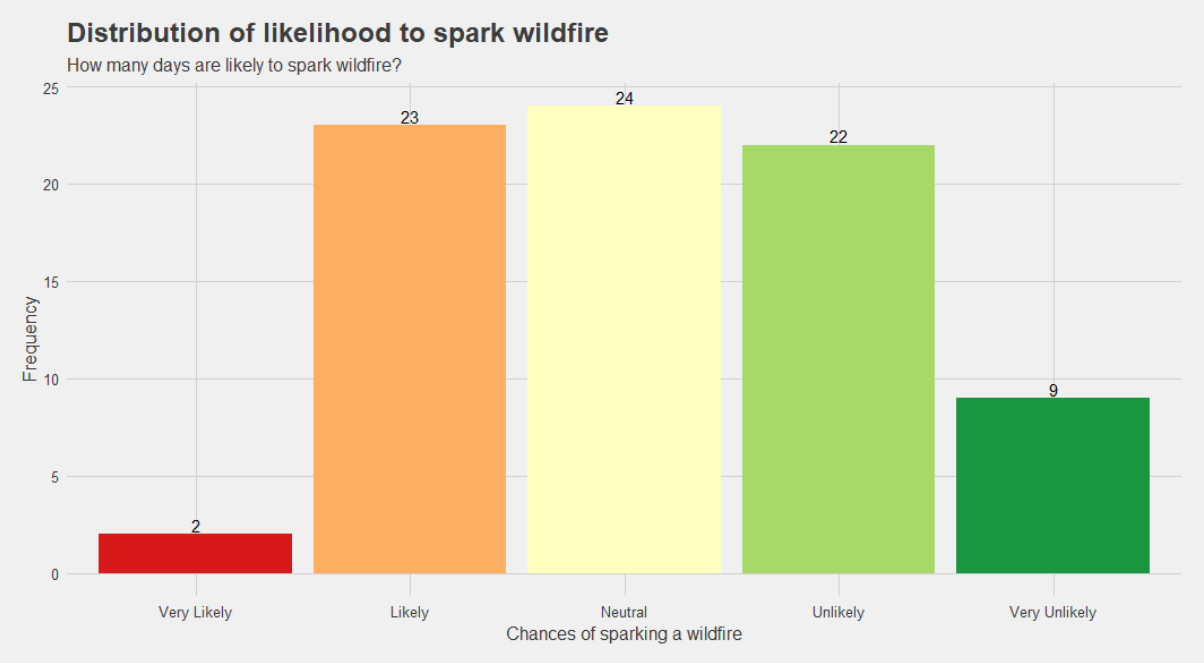


Figure 83 : Bar chart showing the distribution of likelihood to spark a wildfire

Based on the bar chart above, most of the days belong in the category of neutral, likely, and unlikely. The “Very Likely” category shows that there are two days which matches the all the four conditions in the determineWildfire model, this means those days are having dry, sunny, and windy weather condition. For the days that are fallen under the likely category, it means there are only three out of four criteria that are met. While for “Neutral”, only two out of four

criteria are met, the “Unlikely” category only has one out of four criteria are met. Finally, “Very Unlikely” does not have any criteria met to be favorable for sparking a wildfire.

From the bar chart above, it can be said that most of the hot days that did not rain are less prone to catching a wildfire based on the given weather dataset. However, there are still more than 20 days that are likely and very likely to spark a wildfire, based on the given weather dataset. Although the above estimation is not very accurate, it does serve as a direction on how to determine hot days that can spark wildfire easily.

Conclusion (Question 4)

As a conclusion, there are a few things that can be gained from the analysis done above. Firstly, it is confirmed that the majority of the days are the hottest during the evening around 3 pm. This might because in the evening, the sun is directly above the sky and it heats the ground more. Other than that, it can be also determined that temperature and hours of sunshine have a positive relationship. While on the other hand, relative humidity and temperature have a negative relationship. When the temperature gets hotter, the relative humidity level will get lower as their air can hold more water vapor than usual. In addition, the lower humidity level can cause more evaporation, which makes potential fuel sources, such as grass, trees, and branches drier (Means,2018). Finally, a model was built that determines the likeliness of wildfire occurrence in the weather dataset based on four criteria. The results show that majority of the days are not likely to spark a wildfire. However, this does not mean residents and meteorologists can let their guard off, as wildfires are more likely as climate change accelerates, year by year.

Question 5 : What influence heavy rainfall?

This question is trying to determine the characteristics of heavy rainfall and what causes it.

Analysis 1 : Find the pressure for rainy days

Since there is no standard scale that indicates how much rain is heavy, therefore, it is assumed that rainfall below the first quartile of rainfall value will be “light rain”, between the first and third quartile will be “moderate rain”, and any value that is above the third quartile will be “heavy rain”.

```
# determine the 1st and 3rd quartile of rainfall to categorize the rain
weatherData %>%
  filter(RainToday == "Yes") %>%
  select(Rainfall) %>%
  summary()
```

Figure 84 : R code snippet

The R code snippet above will return the summary of rainfall variable given the RainToday equals “Yes”.

```
    Rainfall
 Min.   : 1.200
 1st Qu.: 3.000
 Median : 5.000
 Mean   : 7.664
 3rd Qu.:10.250
 Max.   :39.800
>
```

Figure 85 : Results shows the summary of rainfall variable

Therefore, rainfall under 3mm will be classified as “light rain”, while rainfall between 3mm and 10.25 mm will be classified as “moderate rain”, and above 10.25mm will be classified as “heavy rain”.

```r
# Function Name : transformRainfallScale
# Description   : Categorize the rainfall amount into different categories
# Parameter     : Vector containing rainfall amount for each day
# Return        : Vector that describes the category of rainfall amount for each day

transformRainfallScale = function(rainfall){
  transformed = c()
  index = 1
  while(index <= length(rainfall)){
    if (rainfall[index]<3){ # Below first quartile
      transformed[index] = "Light rain"
    } else if (rainfall[index]<10.25){ # Below third quartile
      transformed[index] = "Moderate rain"
    } else {
      transformed[index] = "Heavy rain"
    }
    index = index + 1
  }
  return (transformed)
}
```

Figure 86 : R code snippet

In the R code snippet above, a function is created to categorize the rainfall value as discussed above. It takes in a vector containing the amount of rainfall and returns a vector that contains the category for each rainfall amount.

```r
weatherData %>%
  filter(RainToday == "Yes") %>%
  mutate(rainCategory = factor(transformRainfallScale(Rainfall),
                               levels = c("Light rain",
                                          "Moderate rain",
                                          "Heavy rain")),
         avgPressure = (Pressure_9am+Pressure_3pm)/2) %>%
  ggplot(aes(rainCategory,avgPressure,fill = rainCategory)) +
  geom_boxplot(outlier.alpha = 0,show.legend = F) +
  geom_point(show.legend = F) +
  labs(title = "Relationship between rainfall amount and pressure",
       subtitle = "Study the atmospheric pressure for each rainfall category",
       x = "Rainfall Category",
       y = "Average pressure (hPA)") +
  theme_minimal()
```

Figure 87 : R code snippet

Based on the R code snippet, the days that did not rain are filtered out. Then a new column “rainCategory” is created to categorize rainfall value into three different categories, which is discussed above. Then, another column which is “avgPressure” is also created that calculates the average pressure of a day by adding the pressure at 9 am and pressure at 3 pm and divide by 2. Afterward, a boxplot is plotted along with a scatter plot on top. In addition, the “outlier.alpha” is set to zero as an argument for geom_boxplot(), so that outliers are not shown by the boxplot. Instead, the scatter plot will do the job of showing outliers.

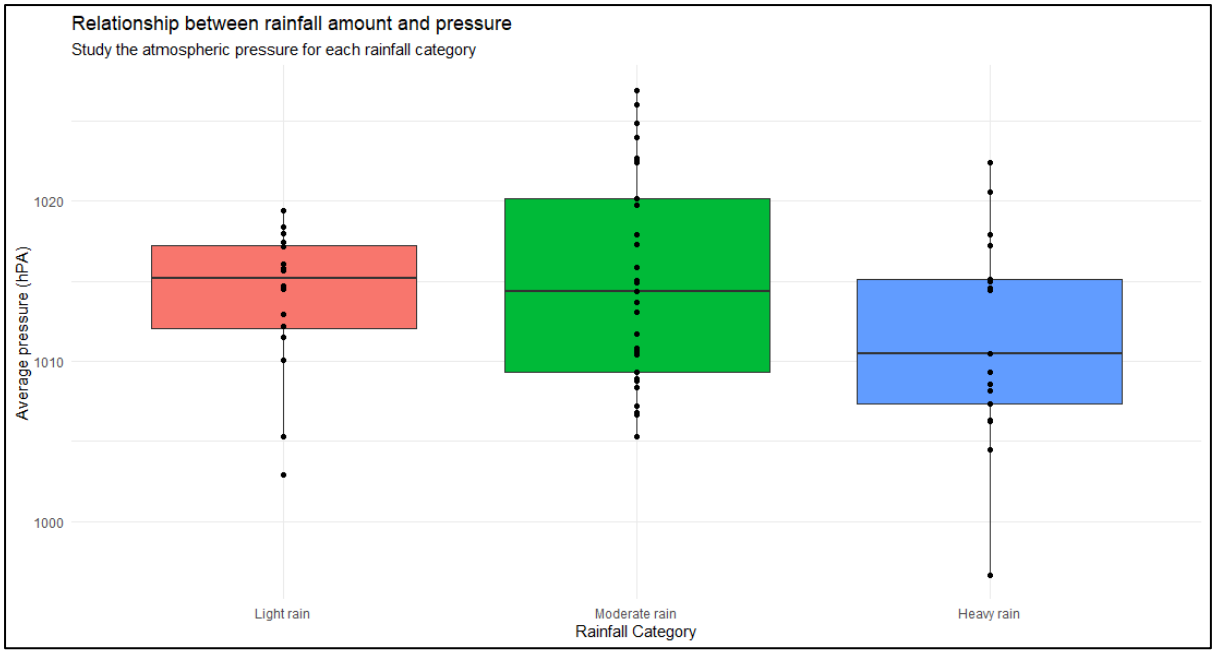


Figure 88 : Boxplot and scatter plot generated from the R code snippet above

According to the figure above, the heavy rain category has the lowest average pressure followed by the moderate rain category and light rain category. This is because heavy rain is commonly associated with a low atmospheric pressure system (Thompson,2020). As discussed above, air naturally moves from an area of high pressure to an area of low pressure, this is the cause of airflow or wind. When an area has low atmospheric pressure, this causes air to rise into the atmosphere, causing condensation and forming precipitation.

Besides that, due to the earth's rotation and friction, airflow is directed inward toward the low-pressure center, while airflow is directed outward away from the high-pressure center (Thompson,2020). Since airflow is directed inward into the low-pressure system the air is forced upward, this causes more amount of air to rise into the atmosphere which condenses and form precipitation (Thompson,2020). More precipitation formed will lead to heavier rain. On the other hand, a high-pressure system is the opposite of a low-pressure system. Air in the high-pressure region is constantly moving away from the high-pressure center to the low-pressure region. As a result, a high-pressure system is associated with stable weathers, because there is not an upward force that lifts the air into the atmosphere which condenses and forms precipitation.

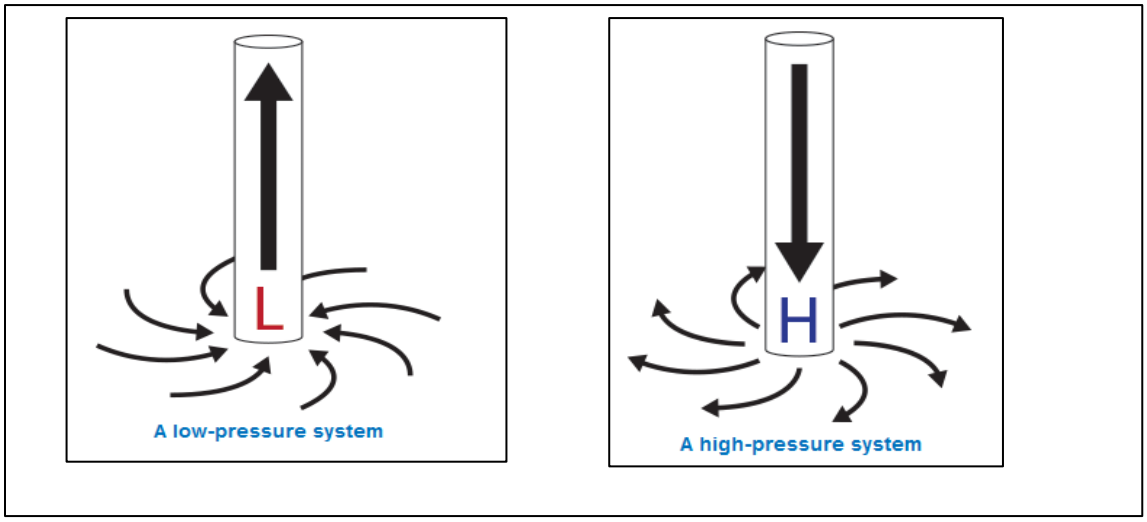


Figure 89 : Illustration of airflow in a low-pressure and high-pressure system (SciJinks,2021)

The figure above shows an illustration of airflow for a low-pressure and high-pressure system, which is discussed above.

Analysis 2 : Determine the cloud formation with rainfall

In this analysis, it is trying to determine whether cloud formation has any effect on rainfall.

```
# Only select the days that rain &
  # Group values by clouds at 9am & clouds at 3pm &
  # Summarize the values by finding average rainfall

# Create a new column that contains the determines the highest clouds formation
  # between 9am and 3pm (higher clouds means rain)

# Categorize rainfall into three different categories

data = weatherData %>%
  filter(RainToday == "Yes") %>%
  group_by(Cloud_9am,Cloud_3pm) %>%
  summarise(count = n(),
            averageRainfall = mean(Rainfall)) %>%
  mutate(maxCloud = max(Cloud_9am,Cloud_3pm),
         rainfallCat = factor(transformRainfallScale(averageRainfall),
                              levels = c("Light rain",
                                         "Moderate rain",
                                         "Heavy rain")))
```

Figure 90 : R code snippet

In the code snippet above, some data manipulation techniques and transformation techniques are done. Firstly, all the days that did not rain are filtered out. Then by using the group_by() function, the data are grouped by Cloud_9am variable and Cloud_3pm variable. The group_by() function is paired with summarise() function to get the number of days with the n() function, and average rainfall for each group. Afterward, two new columns are created with the mutate() function. The column “maxCloud” represents the highest clouds oktas recorded between 9 am and 3 pm. This is because it can be assumed that the time with the most clouds are raining. Next, “rainfallCat” is a column that categorizes the “averageRainfall” value into three different categories with the transformRainfallScale() function.

```
data %>%
  ggplot() +
  geom_point(aes(maxCloud,averageRainfall,size = count,color = rainfallCat)) +
  labs(title = "Relationship between rainfall and cloud formation",
       subtitle = "Average rainfall for rainy days categorized by morning clouds and evening clouds",
       x = "Clouds (oktas)",
       y = "Rainfall (MM)",
       size = "Frequency",
       color = "Rainfall category") +
  theme_minimal()
```

Figure 90 R code snippet

Next, the data manipulated above is plotted into a scatter plot. The scatter plot features maxCloud variable as the x-axis, an averageRainfall variable as the y-axis.

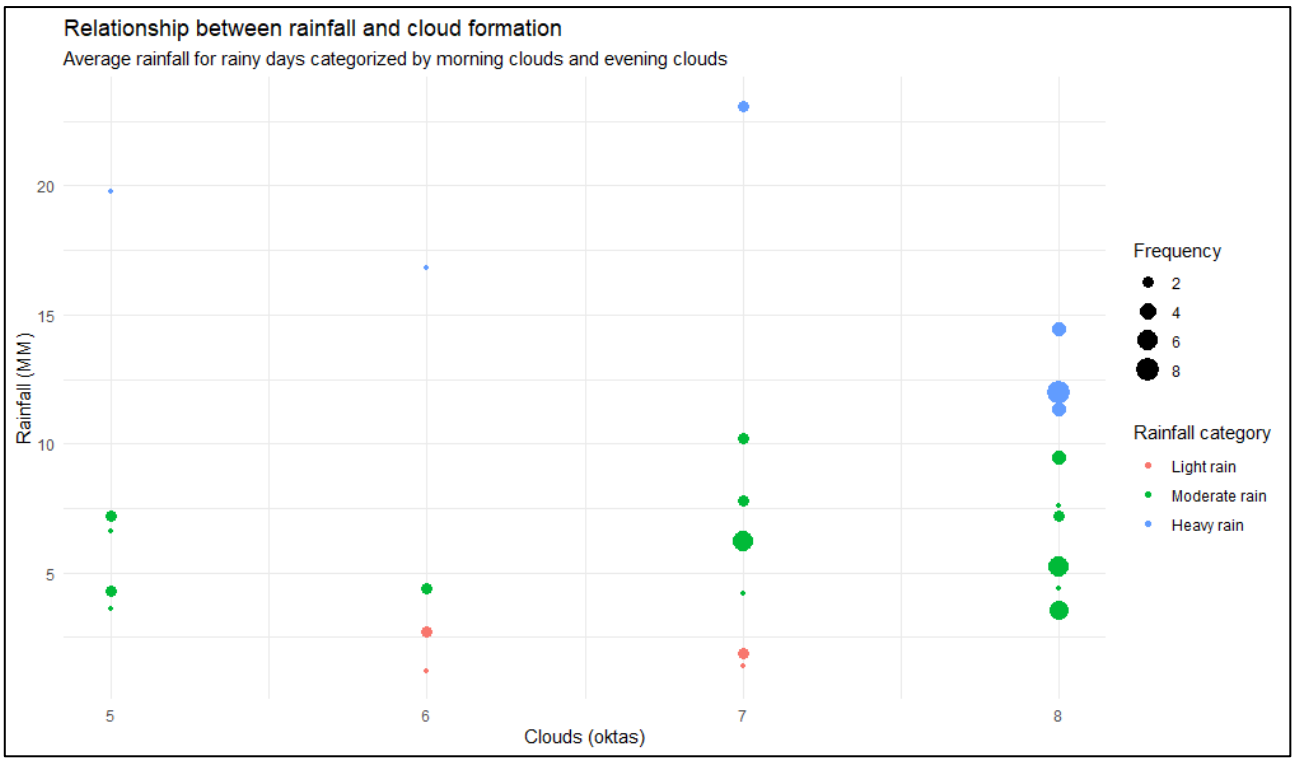


Figure 91 : Scatter plot generated by the code snippet above

From the scatter plot above, the majority of the heavy rain happens with 8 oktas of clouds covering the sky. As the cloud's formation is lesser, the frequencies and amount of rainfall also reduce. This is because as clouds are formed by condensation of water vapor when the clouds get heavier and bigger, it means it is holding a lot of water content, and once the water droplets collide with each other and become heavy enough, they will fall to the ground as rain. Therefore, more cloudy skies result in heavier rain.

To relate to analysis 1, the majority of days with heavy rain have a high number of clouds in the atmosphere is also due to a low-pressure system. As mentioned above, a low-pressure system causes air to rise and form precipitations and clouds, and the heavy rain category contains days with the lowest average pressure compared to other categories. Hence, it can be confirmed clouds and atmospheric pressure has a relationship, and atmospheric pressure influences the formation of clouds.

Conclusion (Question 5)

To sum things up, from the analysis above. It can be determined that pressure can influence the amount of rainfall in a day. In addition, clouds in the sky are a good indicator of whether it will be heavy rain later. Most of the days with heavy rain happened when the skies are fully covered by the clouds (8 oktas). Other than that, it also can be assumed that the lower the atmospheric pressure, the higher the chances of heavy rain. Based on the boxplot in figure 88, there is only one record below 1000 hPA, which results in heavy rain. Hence, it can be assumed from this dataset that any day plunges below 1000hPA might have a higher chance of heavy rain.

Extra features

Extra feature 1 : Data Explorer Package

One of the extra features included is the use of a package called “Data Explorer”. Data Explorer is a package that helps to automate most of the data handling and visualization during Explanatory Data Analysis and helps to facilitate extracting information during the initial analysis process. The usage of the Data Explorer package is shown in code snippets below.

```
# Get a report on the weather dataset (using DataExplorer)
View(introduce(weatherData))
```

Figure 92 : introduce() command in Data Explorer

	rows	columns	discrete_columns	continuous_columns	all_missing_columns	total_missing_values	complete_rows	total_observations	memory_usage
1	366	22	5	17	0	47	328	8052	62368

Figure 93 : Results from the introduce() function

The introduce() function can quickly help to determine the dimension of the weather dataset, the number of discrete columns, continuous columns, and total missing values.

```
#Plotting the initial overview of weatherDataset using DataExplorer
plot_intro(weatherData)
```

Figure 94 : plot_intro() command in Data Explorer

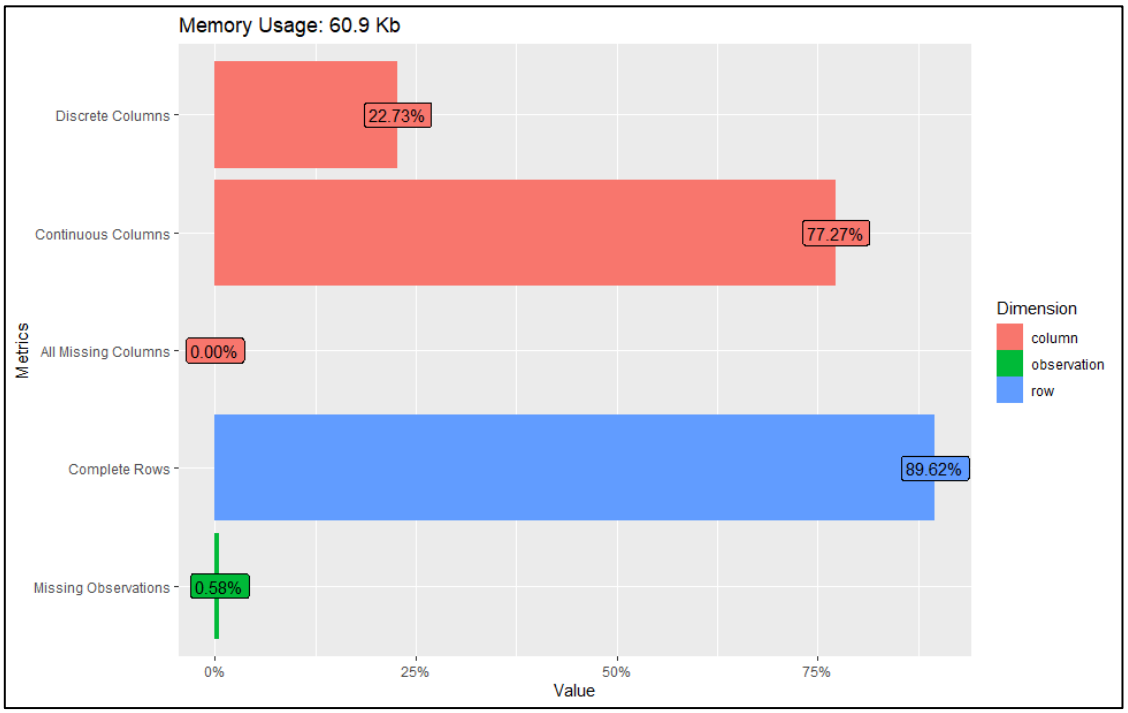


Figure 95 : Results from the plot_intro() command

The plot_intro command plots the results of introduce() function into a bar chart to help better visualize how much proportion of the data is missing, etc.

```
# Plot the missing data value percentage for each column
plot_missing(weatherData)
```

Figure 96 : plot_missing() function

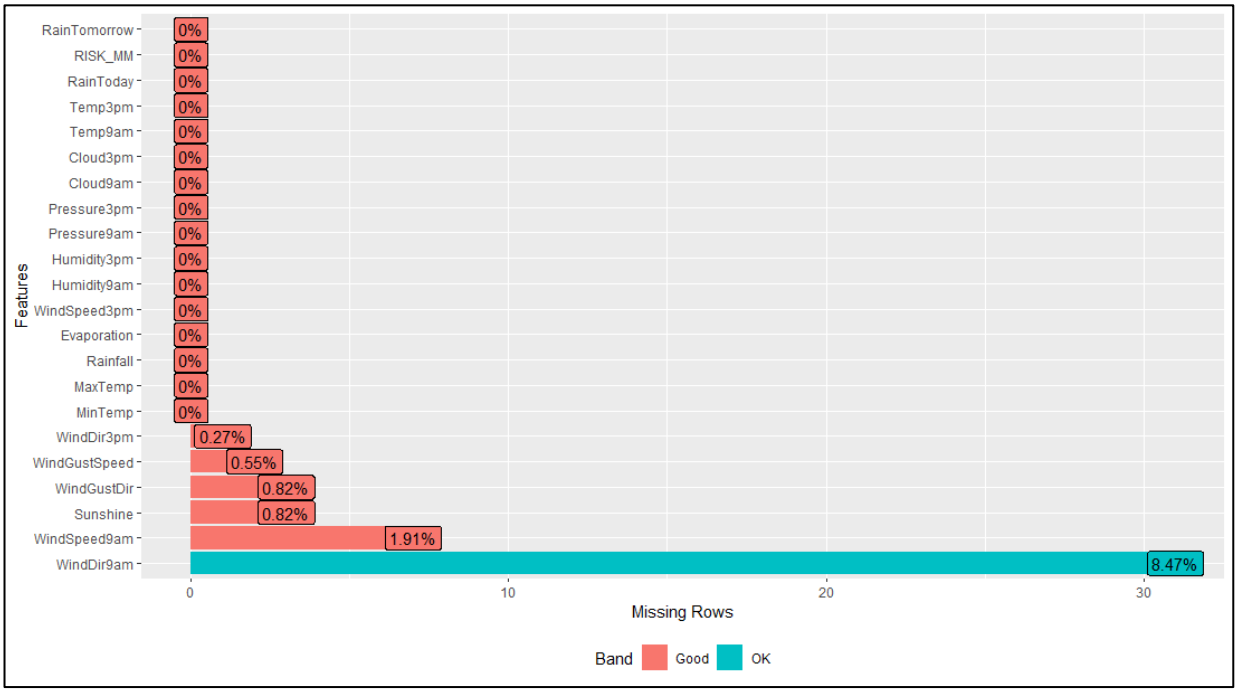


Figure 97 : Results from the plot_missing() function

The plot_missing() function above plots the number of missing values for each column into a bar chart as shown above. This helps to quickly know how many percentages of values are missing for each column, and determine whether the number of missing values is severe.

Extra feature 2: Performance Analytics Package

The Performance Analytics is a R package that is used to generate a correlation plot with Pearson Correlation Coefficient value between all the numeric variables to determine their linear relationship.

```
# Correlation matrix
# Select all the columns with numeric data
numericData = select_if(weatherData,is.numeric)
chart.Correlation(numericData, histogram=TRUE, pch=19)
```

Figure 98 : R code that plots the correlation matrix

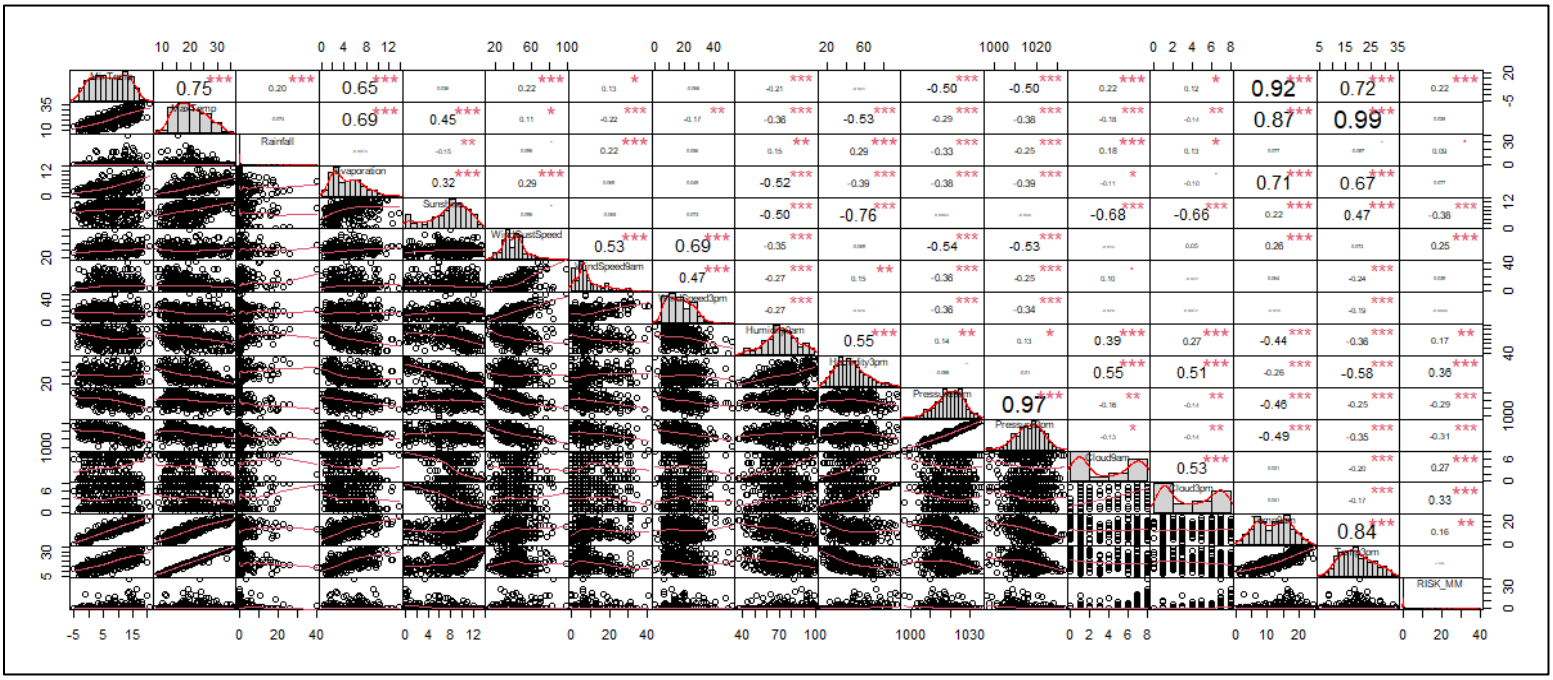


Figure 99 : Correlation Matrix generated from the R code above

Through this, the linear relationship can be determined easily by referring to the Pearson Correlation Coefficient value which is displayed in figure 99 above.

Extra feature 3 : ggthemes Package

The ggthemes package is used to apply a theme to a ggplot, so that it makes the plot more beautiful compared to the default ggplot which is plain and unattractive. Some of the code examples that use ggthemes are shown below.

```
# Find whether temperature increase from 9am to 3pm for all the days
# Create a new column that determines whether temperature at 3pm is bigger than 9am

weatherData %>%
  mutate(tempIncrease = factor(Temp_3pm > Temp_9am)) %>%
  ggplot(aes(Temp_9am,Temp_3pm,color = tempIncrease)) +
  geom_point() +
  labs (title = "Temperature changes for all the days",
        subtitle = "Temperature changes from 9am to 3pm",
        x = "Temperature at 9am (Celsius)",
        y = "Temperature at 3pm (Celsius)",
        linetype = "",
        color = "Temperature increased from 9am to 3pm") +
  theme_igray() +
  scale_y_continuous(breaks = seq(0,40,5))
```

Figure 100 : R code that uses applying a theme to ggplot

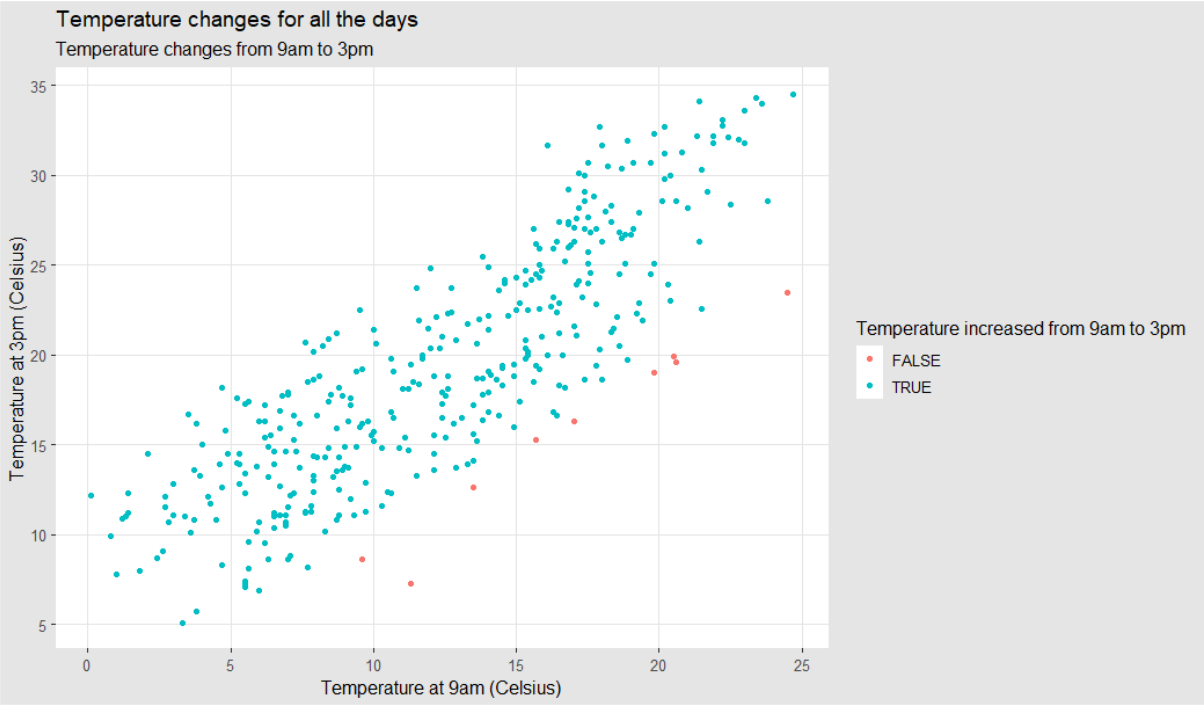


Figure 101 : ggplot with theme_igray() is created

In figure 101, it shows that the ggplot with theme_igray() is created. The plot has different looks compared to the default plot ggplot creates.

Extra feature 4 : RColorBrewer Package

The RColorBrewer Package provides a wide selection of palette colors that can be used to color the plots. This can help to beautify the plots created and also put emphasis on data points with suitable palette colors provided. An example of using RColorBrewer color palettes is shown below.

```
data %>%
  group_by(Likelihood_Wildfire) %>%
  summarise(count = n()) %>%
  ggplot(aes(Likelihood_Wildfire,count)) +
  geom_bar(aes(fill = Likelihood_Wildfire),stat = "identity",show.legend = F) +
  geom_text(aes(label = count),vjust = -0.2) +
  labs(title = "Distribution of likelihood to spark wildfire",
       subtitle = "How many days are likely to spark wildfire?",
       x = "Chances of sparking a wildfire",
       y = "Frequency") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_fill_brewer(palette = "RdYlGn")
```

Figure 102 : R code snippet shows the use of scale_fill_brewer() function

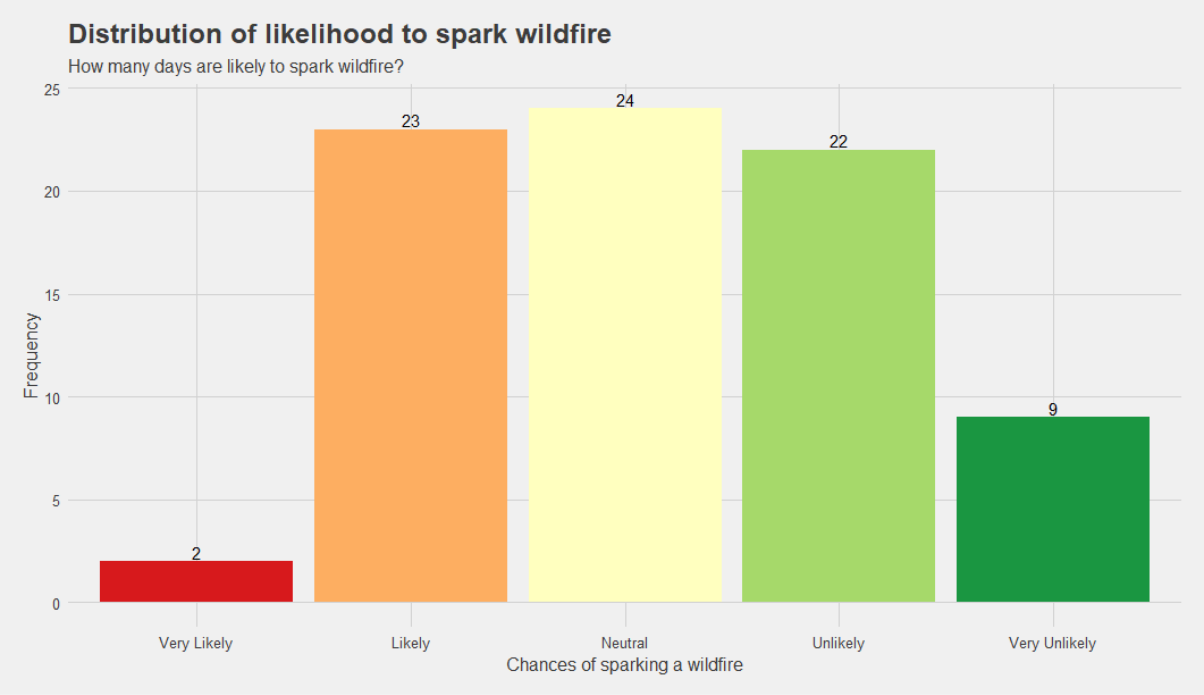


Figure 103 : Bar chart generated with a palette color provided by RColorBrewer package

In the example provided above, the palette color named “RdYlGn” is used to give the bar chart a diverging color palette. Hence, the data presented can be more intuitive and easily understood.

Extra feature 5: Plotly Package

The plotly library is used to produce an interactive version of the plots created. This allows users to interact with the plots created with ggplot and explore the graph. This results in a deeper understanding of the plots generated, and able to quickly retrieve data values from the plot. One of the examples of using Plotly is shown below.

```
data %>%
  ggplot(aes(Temp_3pm,MaxTemp)) +
  geom_jitter(aes(color = Likelihood_Wildfire)) +
  labs (title = "Rough estimate of the likelihood of wildfire",
        subtitle = "Determine the chances of wildfire, for hot days (> 25 degree Celsius)",
        x = "Temperature at 3pm (Celsius)",
        y = "Maximum temperature (Celsius)",
        color = "Chances of sparking a wildfire") +
  theme_fivethirtyeight() +
  theme(axis.title.x = element_text(),axis.title.y = element_text()) +
  scale_color_brewer(palette = "Set1")

# Interactive Plot
ggplotly()
```

Figure 104 : R code that implements plotly to create an interactive plot

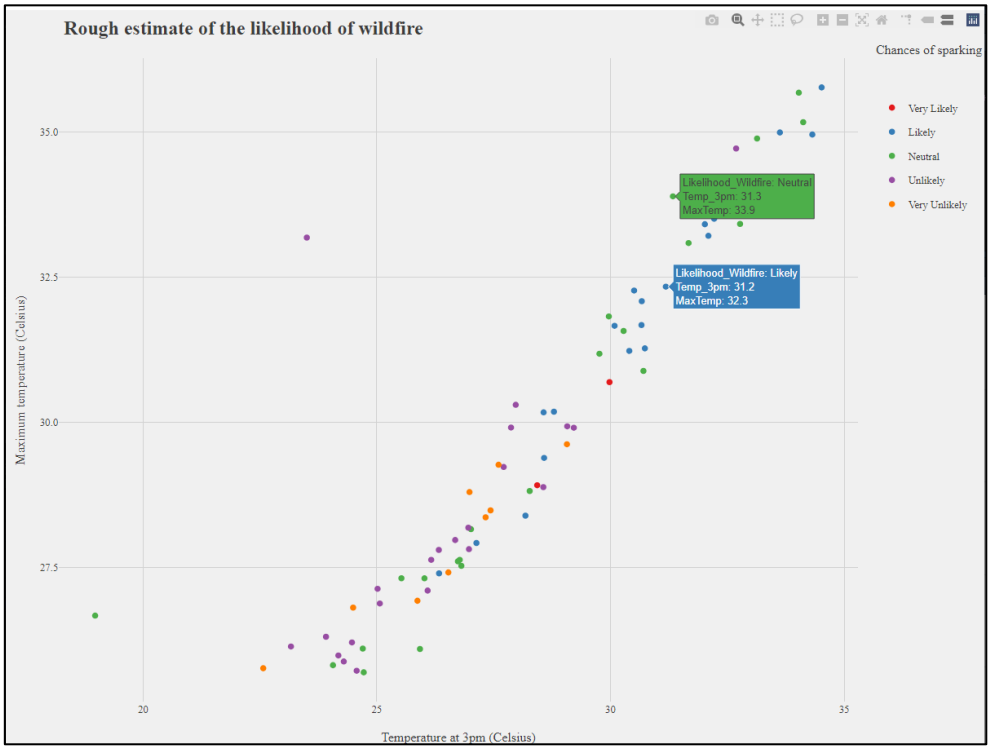


Figure 105 : Interactive scatter plot that can shows the values of the plot

Based on the figure in 104, by using ggplotly() function provided in the Plotly library, it transforms the previous plot generated by ggplot into an interactive plot as shown in figure 105. In the situation where there are many points, Plotly helps to visualize the data points better.

Conclusion

In a nutshell, from the questions and analysis above, a lot of valuable information has been uncovered from the weather dataset such as estimating the likelihood of wildfire, the snowing condition, and whether it will rain tomorrow. The findings are later presented clearly via data visualization techniques. Although the weather dataset was analyzed with basic data analysis skills, it still brings some degree of valuable information that might aid decision-making for authorities or normal people. All in all, there is still room for improvement for analyzing the dataset and further dissecting it to get even more valuable insights.

References

Aprilaire.com. n.d. *Relative Humidity Chart*. [online] Available at: <https://www.aprilaire.com/benefits/preservation/relative-humidity-chart> [Accessed 17 May 2021].

Blaettler, K., 2019. *Do Winds Always Blow From High Pressure to Low Pressure?*. [online] Sciencing. Available at: <https://sciencing.com/winds-always-blow-high-pressure-low-pressure-23398.html> [Accessed 15 May 2021]

Brighthub.com. 2011. *How Does Relative Humidity Affect Evaporation?*. [online] Available at: <https://www.brighthub.com/environment/science-environmental/articles/104601/> [Accessed 15 May 2021].

Chandler, N., 2021. *What Is Relative Humidity and How Does it Affect How I Feel Outside?*. [online] HowStuffWorks. Available at: <https://science.howstuffworks.com/nature/climate-weather/atmospheric/question651.htm> [Accessed 15 May 2021]

CompuWeather. n.d. *The Important Difference Between Wet Snow and Dry Snow*. [online] Available at: <https://www.compuweather.com/the-important-difference-between-wet-snow-and-dry-snow/> [Accessed 17 May 2021].

Dictionary.cambridge.org. n.d. *precipitation*. [online] Available at: <https://dictionary.cambridge.org/dictionary/english/precipitation> [Accessed 14 May 2021].

Geography-site.co.uk. 2013. *What are clouds and why does it rain?*. [online] Available at: <http://www.geography-site.co.uk/pages/physical/climate/why%20does%20it%20rain.html> [Accessed 17 May 2021]

Jessika Toothman "How Clouds Work" 5 May 2008. HowStuffWorks.com. <https://science.howstuffworks.com/nature/climate-weather/atmospheric/cloud.htm> [Accessed 25 May 2021

Means, T., 2018. *What Kind of Weather Is "Fire Weather"?*. [online] ThoughtCo. Available at: <https://www.thoughtco.com/what-is-fire-weather-3443859> [Accessed 16 May 2021]

MichealV, 2018. *Oktas – Measuring Cloud Cover – The Michigan Weather Center*. [online] Michigan-weather-center.org. Available at: <https://michigan-weather-center.org/oktas-measuring-cloud-cover> [Accessed 17 May 2021].

Nsidc.org. n.d. *Snow and Weather | National Snow and Ice Data Center*. [online] Available at: <https://nsidc.org/cryosphere/snow/science/weather.html> [Accessed 16 May 2021].

Oblack, R., 2020. *How to Read a Barometer*. [online] ThoughtCo. Available at: <https://www.thoughtco.com/how-to-read-a-barometer-3444043> [Accessed 20 May 2021].

Optimum Humidity. n.d. *How Does Temperature Affect Humidity? - Optimum Humidity*. [online] Available at: <https://optimumhumidity.com/how-does-temperature-affect-humidity/> [Accessed 18 May 2021].

RMetS Editor, 2018. *Royal Meteorological Society*. [online] RMetS. Available at: https://www.rmets.org/resource/beaufort-scale [Accessed 16 May 2021].

Rutledge, K., Ramroop, T., Boudreau, D., McDaniel, M., Teng, S., Sprout, E., Costa, H., Hall, H. and Hunt, J., 2011. *Beaufort scale*. [online] National Geographic Society. Available at: <https://www.nationalgeographic.org/encyclopedia/beaufort-scale/> [Accessed 15 May 2021].

Scijinks.gov. 2021. *What Are High and Low Pressure Systems? | NOAA SciJinks – All About Weather*. [online] Available at: <https://scijinks.gov/high-and-low-pressure-systems/> [Accessed 15 May 2021].

Shisia, M., 2017. *How is Snow Formed?*. [online] WorldAtlas. Available at: <https://www.worldatlas.com/articles/how-is-snow-formed.html> [Accessed 17 May 2021].

Sparklebox.co.uk. 2021. *16-Point Compass Poster (SB6589) - SparkleBox*. [online] Available at: <https://www.sparklebox.co.uk/6581-6590/sb6589.html> [Accessed 16 May 2021].

THE GEOGRAPHER ONLINE. n.d. *Weather and Climate*. [online] Available at: <https://www.thegeographeronline.net/weather-and-climate.html> [Accessed 18 May 2021].

Thompson, D., 2020. *High and Low Pressure*. [online] Weatherworksinc.com. Available at: <https://weatherworksinc.com/news/high-low-pressure> [Accessed 16 May 2021].

weather-station, 2018. *How Does Air Pressure Affect Weather | The Weather Station*. [online] The Weather Station. Available at: <https://the-weather-station.com/how-does-air-pressure-affect-weather/> [Accessed 16 May 2021].

Union of Concerned Scientists. 2020. *Infographic: Wildfires and Climate Change*. [online] Available at: <https://www.ucsusa.org/resources/infographic-wildfires-and-climate-change> [Accessed 18 May 2021].