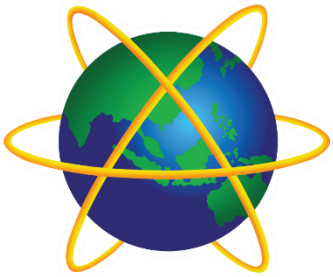

A P U

ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Introduction to R Programming (112022-MHJ)

Assignment Title	**:** U.S Weather Report Data Management
Name	**:** Afif Rifa'ie bin Mohd Gazali
TP Number	**:** TP072295
Intake	: APDMP2301DSBA(BI)(PR)
Lecturer	**:** Ms. Minnu Helen Joseph

Table of content

Introduction

The purpose of this report is to investigate and analyse weather data from the United States by utilising a variety of methods that are capable of retrieving the necessary information. The data contains 24 columns and 366 rows of information regarding rain factors and tabulation in the United States for a period of one year. The variables included in the data are temperature, humidity, and a number of others. The method begins with performing data importing, data exploration, data cleaning and data pre-processing, followed by investigation and visualisation into a visual graph using R Studio.

Data Import

The first step of data analysis is to import the given data set. The data was given in format called csv. Which the data in the file is separated by commas.

```
#Get working directory
getwd()

#Set working directory
setwd("C:\\Users\\Afif Rifa'ie\\Desktop\\A.P.U Master\\R Programming")

#install package
install.packages("ggplot2")
install.packages("dplyr")
install.packages("tidyverse")
install.packages("plotly")

#upload package
library(ggplot2)
library(dplyr)
library(tidyverse)
library(plotly)

#Data import and set the data into a variable
weather_data = read.csv("C:\\Users\\Afif Rifa'ie\\Desktop\\weather.csv",header=TRUE)
weather_data
```

Figure 1: setting working directory, installing packages and importing data set

Based on the above snippet, the first thing I did was set the working directory and save it in my personal file. Then I download the necessary packages and install it in my R Studios. To import the data, I used the read.csv() function, which round bracket is the location of the data set on my computer, and I set the given data set as a weather_data variable. I install packages with install.packages() function and the library() function to store the code in the R environment under a directory called library.

Data Exploration

After importing the Data Set into the R Studio. The next step is data exploration which to understand the data set. This is to do research to understand better of the rain attributes. I did research on each of the columns name in the data. After the research, I get basic information on what are the meaning of each column provided in the data set. After doing research, the next step is to know the basic information of the data such its structure, dimensions, columns names and summary.

```
#Data exploration
#structure
str(weather_data)

#Number of Row and Columns
dim(weather_data)

#column name
colnames(weather_data)

#to get all the header
names(weather_data)

#summary of data
summary(weather_data)

#to view data in table format
View(weather_data)
```

Figure 2: Data exploration

To explore the data, I use str() function to find out the class of each column in data set. Then, I use colnames() function to find out the names of the

data. I use the summary() function to get the data's mean, lowest, and highest values. View() function is used to view the data set in table format. The output of the function as below.

```
> str(weather_data)
'data.frame':   366 obs. of  23 variables:
 $ Min_Temp       : num  8 14 13.7 13.3 7.6 6.2 6.1 8.3 8.8 8.4 ...
 $ Max_Temp       : num  24.3 26.9 23.4 15.5 16.1 16.9 18.2 17 19.5 22.8 ...
 $ Rain_Fall      : 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 ...
 $ Wind_Gust_Dir  : chr  "NW" "ENE" "NW" "NW" ...
 $ Wind_Gust_Speed: int  30 39 85 54 50 44 43 41 48 31 ...
 $ Wind_Dir_9am   : chr  "SW" "E" "N" "WNW" ...
 $ Wind_Dir_3pm   : chr  "NW" "W" "NNE" "W" ...
 $ Wind_Speed_9am : int  6 4 6 30 20 20 19 11 19 7 ...
 $ Wind_Speed_3pm : int  20 17 6 24 28 24 26 24 17 6 ...
 $ Humidity_9am   : int  68 80 82 62 68 70 63 65 70 82 ...
 $ Humidity_3pm   : int  29 36 69 56 49 57 47 57 48 32 ...
 $ Pressure_9am   : num  1020 1012 1010 1006 1018 ...
 $ Pressure_3pm   : num  1015 1008 1007 1007 1018 ...
 $ Cloud_9am      : int  7 5 8 2 7 7 4 6 7 7 ...
 $ Cloud_3pm      : int  7 3 7 7 7 5 6 7 7 1 ...
 $ Temp_9am       : num  14.4 17.5 15.4 13.5 11.1 10.9 12.4 12.1 14.1 13.3 ...
 $ Temp_3pm       : num  23.6 25.7 20.2 14.1 15.4 14.8 17.3 15.5 18.9 21.7 ...
 $ Rain_Today     : chr  "No" "Yes" "Yes" "Yes" ...
 $ Risk_MM        : num  3.6 3.6 39.8 2.8 0 0.2 0 0 16.2 0 ...
 $ Rain_Tomorrow  : chr  "Yes" "Yes" "Yes" "Yes" ...
 $ Days           : int  1 2 3 4 5 6 7 8 9 10 ...
```

Figure 3: str() function output

str() function gives an overview on how many observations and variables are in the data set. This function also provides the information about the type of data, which useful when determining the graph for visualization.

```
> dim(weather_data)
[1] 366  23
```

Figure 4: dim() function output

dim() function provides information about how many rows and columns are in the data set. There are 366 rows and 23 columns in the weather_data.

```
> colnames(weather_data)
 [1] "Min_Temp"       "Max_Temp"       "Rain_Fall"      "Evaporation"    "Sunshine"       "Wind_Gust_Dir"
 [7] "Wind_Gust_Speed" "Wind_Dir_9am"   "Wind_Dir_3pm"   "Wind_Speed_9am" "Wind_Speed_3pm" "Humidity_9am"
[13] "Humidity_3pm"   "Pressure_9am"   "Pressure_3pm"   "Cloud_9am"      "Cloud_3pm"      "Temp_9am"
[19] "Temp_3pm"       "Rain_Today"     "Risk_MM"        "Rain_Tomorrow"
```

Figure 5: colnames() function output

colnames() function provides the names of each column in the data frame. It is also function to change the names of the columns.

```
> names(weather_data)
 [1] "Min_Temp"       "Max_Temp"       "Rain_Fall"      "Evaporation"    "Sunshine"       "Wind_Gust_Dir"
 [7] "Wind_Gust_Speed" "Wind_Dir_9am"   "Wind_Dir_3pm"   "Wind_Speed_9am" "Wind_Speed_3pm" "Humidity_9am"
[13] "Humidity_3pm"   "Pressure_9am"   "Pressure_3pm"   "Cloud_9am"      "Cloud_3pm"      "Temp_9am"
[19] "Temp_3pm"       "Rain_Today"     "Risk_MM"        "Rain_Tomorrow"
```

Figure 6 : names() function output

Similarly, names() function gives the name of the object's header in a data frame.

```
> #summary of data
> summary(weather_data)
    Min_Temp         Max_Temp       Rain_Fall       Evaporation        Sunshine      Wind_Gust_Dir
 Min.   :-5.300   Min.   : 7.60   Min.   : 0.000   Min.   : 0.200   Min.   : 0.000   Length:366
 1st Qu.: 2.300   1st Qu.:15.03   1st Qu.: 0.000   1st Qu.: 2.200   1st Qu.: 5.950   Class :character
 Median : 7.450   Median :19.65   Median : 0.000   Median : 4.200   Median : 8.600   Mode  :character
 Mean   : 7.266   Mean   :20.55   Mean   : 1.428   Mean   : 4.522   Mean   : 7.909
 3rd Qu.:12.500   3rd Qu.:25.50   3rd Qu.: 0.200   3rd Qu.: 6.400   3rd Qu.:10.500
 Max.   :20.900   Max.   :35.80   Max.   :39.800   Max.   :13.800   Max.   :13.600
                                                                    NA's   :3
 Wind_Gust_Speed Wind_Dir_9am       Wind_Dir_3pm       Wind_Speed_9am   Wind_Speed_3pm   Humidity_9am
 Min.   :13.00   Length:366         Length:366         Min.   : 0.000   Min.   : 0.00   Min.   :36.00
 1st Qu.:31.00   Class :character   Class :character   1st Qu.: 6.000   1st Qu.:11.00   1st Qu.:64.00
 Median :39.00   Mode  :character   Mode  :character   Median : 7.000   Median :17.00   Median :72.00
 Mean   :39.84                                         Mean   : 9.652   Mean   :17.99   Mean   :72.04
 3rd Qu.:46.00                                         3rd Qu.:13.000   3rd Qu.:24.00   3rd Qu.:81.00
 Max.   :98.00                                         Max.   :41.000   Max.   :52.00   Max.   :99.00
 NA's   :2                                             NA's   :7
  Humidity_3pm    Pressure_9am     Pressure_3pm      Cloud_9am       Cloud_3pm        Temp_9am         Temp_3pm
 Min.   :13.00   Min.   : 996.5   Min.   : 996.8   Min.   :0.000   Min.   :0.000   Min.   : 0.100   Min.   : 5.10
 1st Qu.:32.25   1st Qu.:1015.4   1st Qu.:1012.8   1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 7.625   1st Qu.:14.15
 Median :43.00   Median :1020.1   Median :1017.4   Median :3.500   Median :4.000   Median :12.550   Median :18.55
 Mean   :44.52   Mean   :1019.7   Mean   :1016.8   Mean   :3.891   Mean   :4.025   Mean   :12.358   Mean   :19.23
 3rd Qu.:55.00   3rd Qu.:1024.5   3rd Qu.:1021.5   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:17.000   3rd Qu.:24.00
 Max.   :96.00   Max.   :1035.7   Max.   :1033.2   Max.   :8.000   Max.   :8.000   Max.   :24.700   Max.   :34.50

  Rain_Today          Risk_MM       Rain_Tomorrow
 Length:366         Min.   : 0.000   Length:366
 Class :character   1st Qu.: 0.000   Class :character
 Mode  :character   Median : 0.000   Mode  :character
                    Mean   : 1.428
                    3rd Qu.: 0.200
                    Max.   :39.800
```

Figure 7 : summary() function output

summary() function provides an overview of the variables, including their minimum, first quartile, median, mean, third quantile, and maximum values.

```
> View(weather_data)
```

Figure 8 : Views() function

View() function is used to produce table style data viewer.

Data Cleaning

Once I explored the data, I proceed with data cleaning. Basically means to make sure that all the columns are set up in a way that is easy for the user to understand and to make sure the columns are set without spacing and inconsistent use of capital and small letter. As the snipped below, I reset the column names with proper spacing using underscore and start the name with capital letter. I state the new columns name into the vector c(" ") and set it back to the weather_data.

```
#Data cleaning
#rename all the header
colnames(weather_data)<-c("Min_Temp", "Max_Temp", "Rain_Fall", "Evaporation", "Sunshine", "Wind_Gust_Dir",
                          "Wind_Gust_Speed", "Wind_Dir_9am", "Wind_Dir_3pm", "Wind_Speed_9am",
                          "Wind_Speed_3pm", "Humidity_9am", "Humidity_3pm", "Pressure_9am",
                          "Pressure_3pm", "Cloud_9am", "Cloud_3pm", "Temp_9am", "Temp_3pm",
                          "Rain_Today", "Risk_MM", "Rain_Tomorrow")
```

Figure 9: Data cleaning and rename the header

Data Pre-processing

After cleaning the data set, the next step is to do the data pre-processing. From the data set, there are no dates given for when each of data attributes happened. As below snipped, I add new Days columns to the data set. With Days column, it would be useful to measure when each of the attribute happened. First, I set new vector of 1:366 because there are 366 rows of data. The assumption I make is each of the row represent a day. Then I used cbind() function to add new Days column in the data set.

```
#add Days column
Days = c(1:366)
weather_data = cbind(weather_data, Days)
```

Figure 10: Add new column

Question 1 The United States Rain Analysis

Analysis 1.1: Days with and without rain over the course of a year.

```
#Raining and non-raining days
weather_data %>%
  ggplot(aes(Rain_Today))+
  geom_bar(fill = "#47B23F")+
  theme_bw()+
  labs(x = "Rain",
       y = "Occurance (Days)",
       title = "Number of raining and not raining day in a year")
```

Figure 11: Analysis 1.1

In this assignment, pipe function %>% is used to start the code as it takes output of one function, and passes it into the another function as an argument. The stored data variable is weather_data. geom_bar() function under ggplot() package is used to build a bar chart graph. The x-axis shows days it rained or didn't rain, and the y-axis shows how many days it happened. The bar's colour is set by the fill function in geom_bar(). The theme_bw function is to set theme as grey theme with grey lines on white background. While labs() function is used to change the names of the x axis, y axis and to give title of the graph.

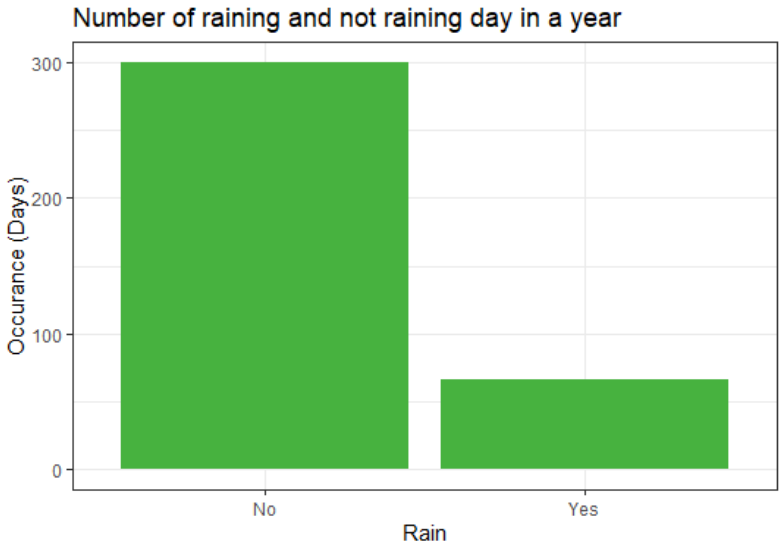

Figure 12 : output of analysis 1.1

The figure above shows that the number of days when it does not rain is more than the number of days when it does rain. Meaning that it does not rain very often in the U.S.

Analysis 1.2 Number of rainy and not rainy days for tomorrow

```
#Raining and non-raining days for tomorrow
weather_data %>%
  ggplot(aes(Rain_Tomorrow))+
  geom_bar(fill = "#447ED2")+
  theme_bw()+
  labs(x = "Rain Tomorrow",
       y = "Occurance (Days)",
       title = "Number of raining and not raining the next day in a year")
```

Figure 13: Analysis 1.2

The code is used to make a bar chart graph, where x representing will it rain tomorrow and y axis representing how often it will rain. The bar's colour is set by the fill function in geom_bar().

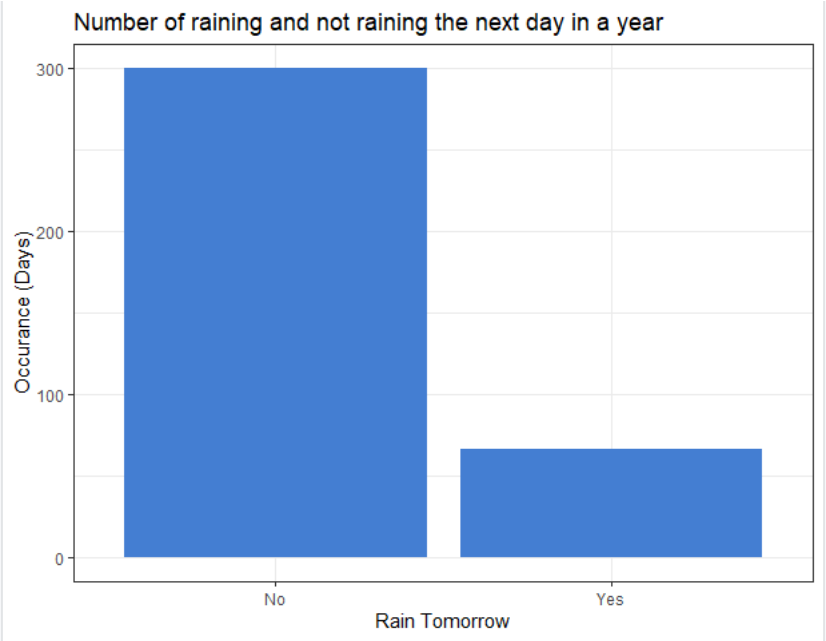

Figure 14: output of analysis 1.2

Similar to rain today bar chart, rain tomorrow has similar amount of rainy and not rainy days where no rain takes the majority of the days in the United States throughout a year.

Analysis 1.3 Total rainfall for raining days

```
#Total rainfall on raining days
weather_data %>%
  ggplot(aes(Days, Rain_Fall, colour = Rain_Today))+
  geom_point()+
  geom_line()+
  geom_smooth(method = lm)+
  theme_bw()+
  labs(x = "Days",
       y = "Rainfall",
       title = "Rainfall amount during rain")
```

Figure 15: Analysis 1.3

This code is used to build a scatter plot with a tabulation line that shows how much rain falls throughout a year, with X axis representing the number of days and Y axis representing the amount of rain. In addition to that, colour = Rain_Today in ggplot() function to differentiate categorical variable for Rain_Today in the plot. Green represent yes for raining days while orange represent not raining days. geom_smooth(method = lm) function to add a straight regression line into the scatter plot.

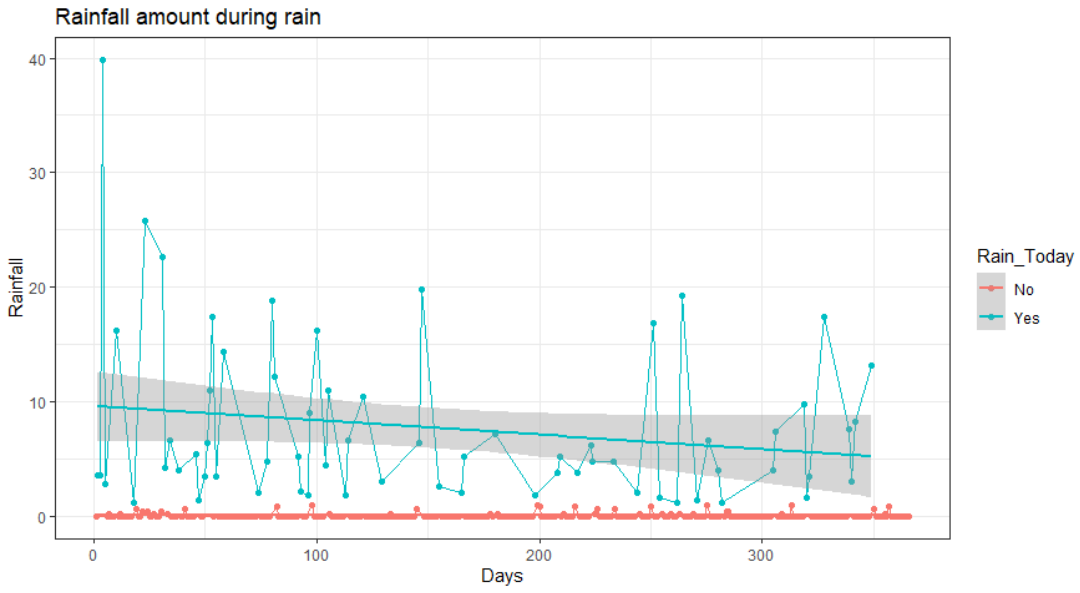

Figure 16: Output of analysis 1.3

The figure above shows that most of the rain falls between 0 and 20 throughout the year, and only one day has high amount of rain fall.

Analysis 1.4 Relation between cloud at 3pm and rain today

```
#Relation between cloud at 3pm and Rain Today
weather_data %>%
  ggplot(aes(Cloud_3pm,Rain_Today))+
  geom_boxplot(fill = "#D997E2")+
  theme_bw()+
  labs(x = "Cloud level",
       y = "Rain today",
       title = "Relation between cloud at 3pm and Rain Today")
```

Figure 17 : Analysis 1.4

The code is used to build a boxplot graph. The x axis representing cloud level, the y axis representing rain today. The colour of the boxplot is set by the fill function in geom_boxplot.

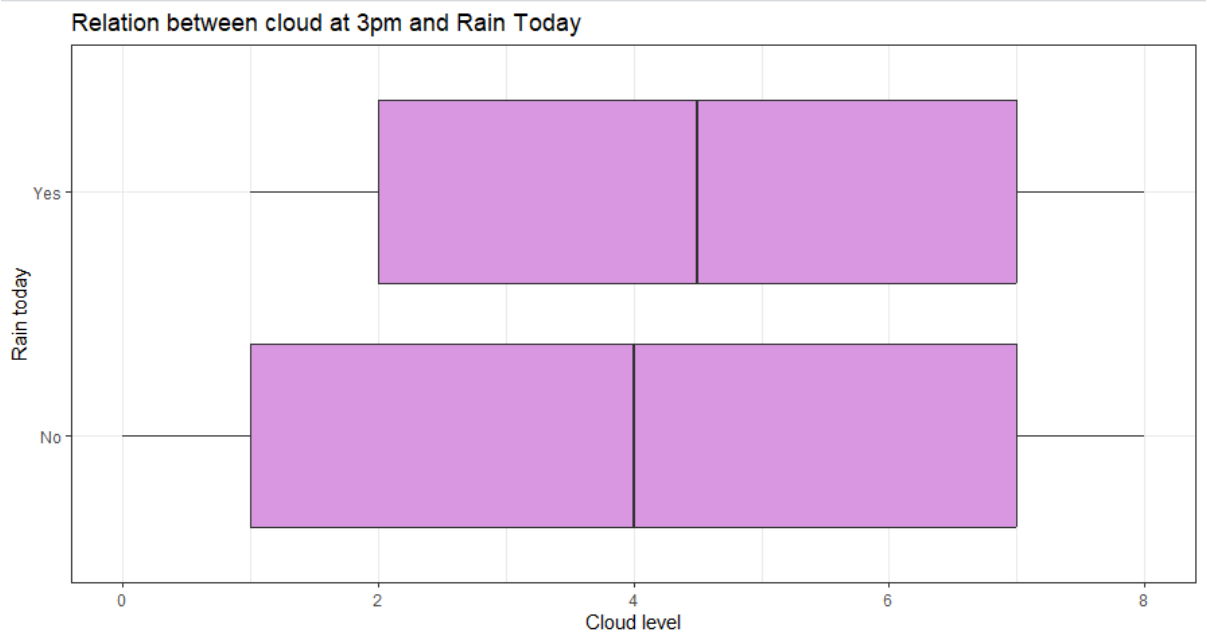

Figure 18: Output of analysis 1.4

The line in the middle of the box shows the average value, while the left and right walls show the lowest and highest values. The boxplot above shows that on days when it rains, the clouds are more likely to be in the 2–7 cloud level range than on days when it doesn't rain, when the cloud level can go as low as 1.

Question 2 Level of dryness in the United States

Analysis 2.1 –Average Temperature at 3pm

```
weather_data %>%
  ggplot(aes(Rain_Today,Temp_3pm))+
  geom_boxplot(fill = "#D55F84")+
  coord_flip()+
  theme_bw()+
  labs(x = "Rain",
       y = "Temperature",
       title = "Average temperature at 3pm when raining and not raining days")
```

Figure 19: Analysis 2.1

The code is used to build a boxplot graph. The x axis representing rain today, the y axis representing temperature at 3pm. The colour of the boxplot is set by the fill function in geom_boxplot. The coord_flip() function is used to flip the graph so that it can have better view.

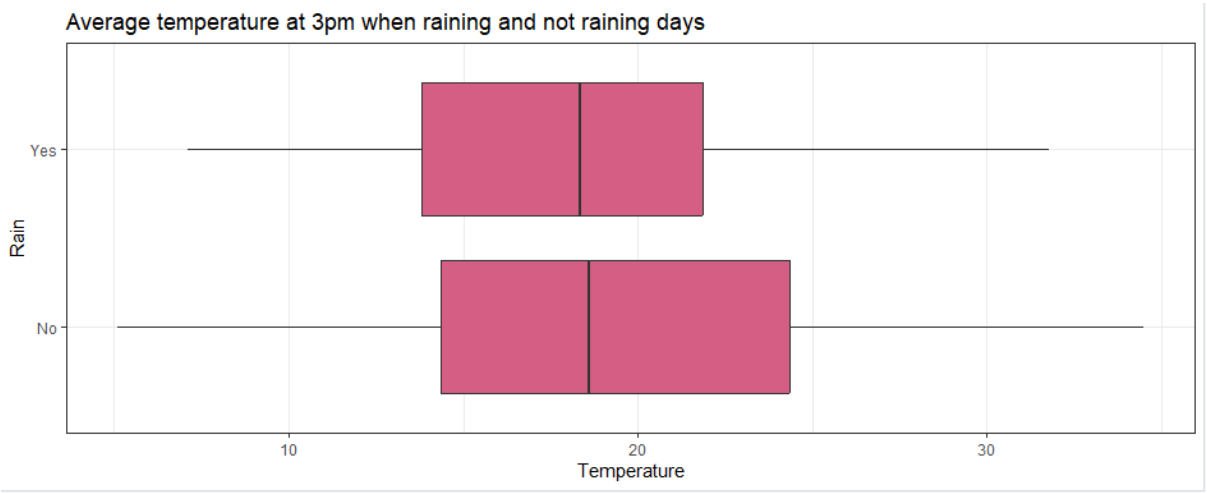

Figure 20: Output of analysis 2.1

The line in the middle of the box shows the average value, while the left and right walls show the lowest and highest values. The boxplot above shows that when it rains, the average temperature is a little lower than when it doesn't. The lowest temperature is 5.10 when raining. On days when it doesn't rain, the temperature can get as high as 34.50.

Analysis 2.2 – Compare temperature and sunshine

```
#Compare temperature at 3pm and sunshine
weather_data %>%
  ggplot(aes(Sunshine,Temp_3pm))+
  geom_point()+
  geom_smooth(method=lm)+
  theme_bw()+
  labs(x = "Sunshine",
       y = "Temperature",
       title = "Relation between Sunshine and Temperature at 3pm")
```

Figure 21: Analysis 2.1

The code is used to build a scatter plot. The X-axis shows the amount of sunshine, and the y-axis shows the temperature at 3 p.m. For the straight linear line on the graph, the geom_smooth() function with method = lm is used.

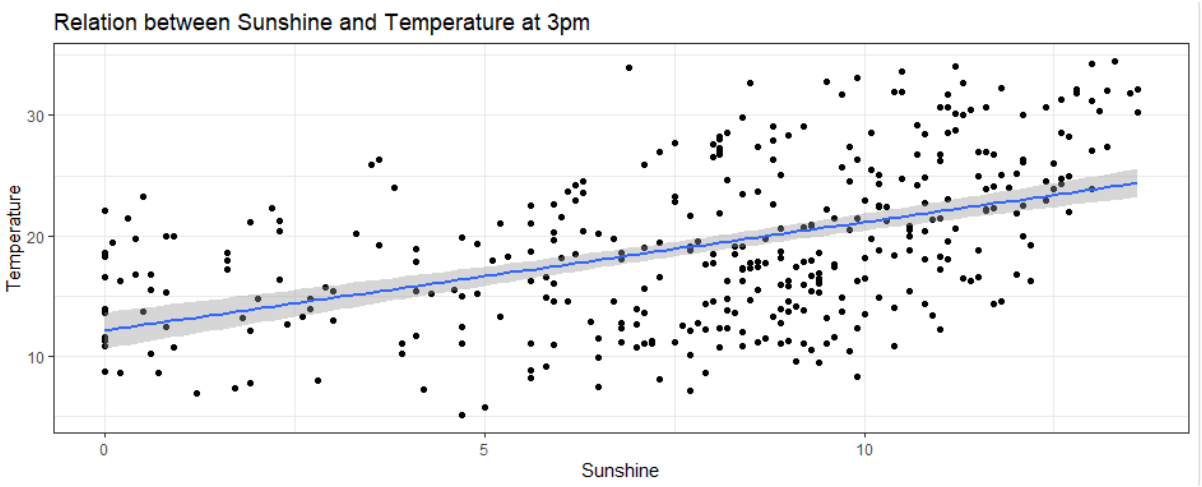

Figure 22: Output of analysis 2.1

The figure above shows that there is a positive relationship between the amount of sunshine and the temperature. When there is more sunshine, the temperature goes up.

Analysis 2.3 – Compare temperature and humidity

```
#compare temperature and humidity at 9am
weather_data %>%
  ggplot(aes(Humidity_9am,Temp_9am))+
  geom_point()+
  geom_smooth()+
  theme_bw()+
  labs(x = "Humidity",
       y = "Temperature",
       title = "Relation between Humidity and Temperature at 9am")
```

Figure 23: Analysis 2.3

The code above is used to build a scatter plot. The x axis representing humidity, the y axis representing temperature at 9am. The geom_smooth() function is used to get the linear line in a graph.

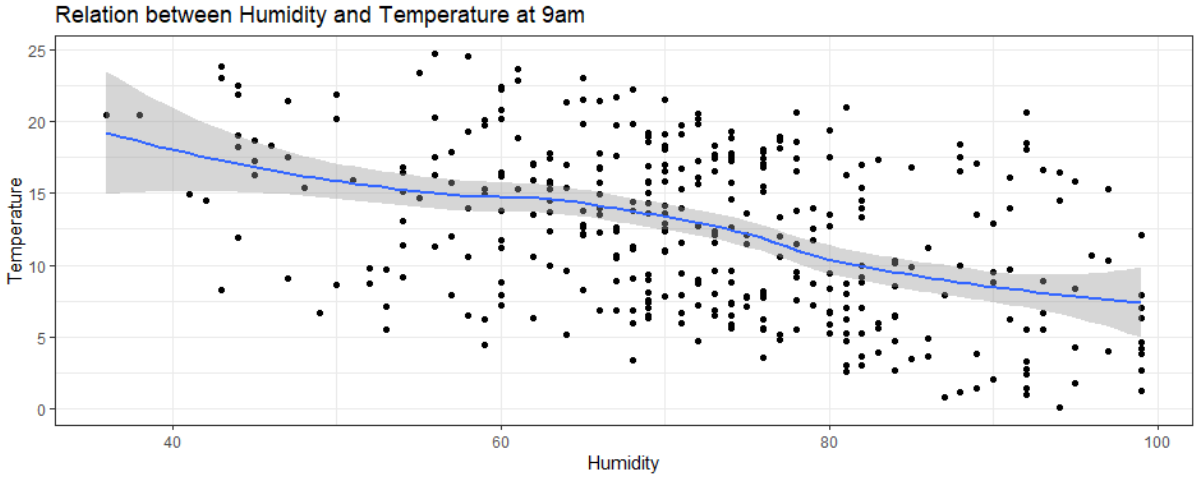

Figure 24: Output of analysis 2.4

The above figure shows that the humidity has negative relationship against temperature. When humidity is high, the temperature goes down.

Analysis 2.4 – Compare temperature and evaporation

```
#compare temperature and evaporation
weather_data %>%
  ggplot(aes(Temp_9am,Evaporation))+
  geom_point(colour = "#EDAEAE")+
  geom_smooth(method = lm)+
  theme_bw()+
  labs(x = "Temperature",
       y = " Evaporation",
       title = "Relation between temperature and evaporation")
```

Figure 25: Analysis 2.4

The above code is used to build a scatter plot. The x axis representing temperature at 9am, the y axis representing evaporation. The fill function in geom_boxplot is to set the colour of the point. The geom_smooth(lm) function is to get the linear line in the graph.

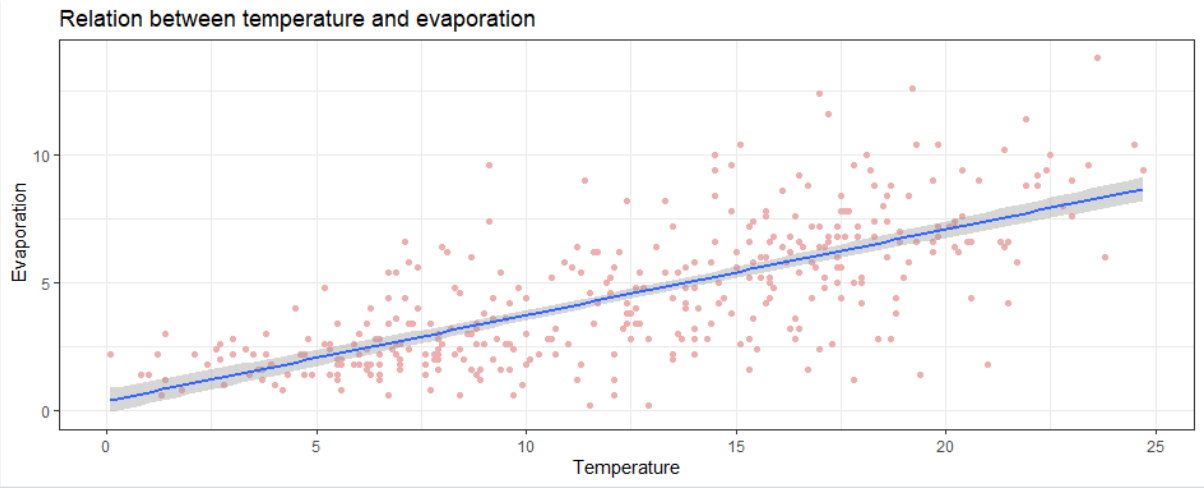

Figure 26: Output of analysis 2.4

The above figure shows that the evaporation has positive relations with the temperature. The amount of water that evaporates depends on how high the temperature is.

Question 3 Common Breeze in the United States

Analysis 3.1 –Average Wind Gust Speed

```
weather_data %>%
  ggplot(aes(Days, Wind_Gust_Speed))+
  geom_point(color = "#049599")+
  geom_smooth(method = lm)+
  theme_minimal()+
  labs(x = "Days",
       y = "Wind Gust Speed",
       title = "Wind Gust Speed throughout a year")
```

Figure 27: Analysis 3.1

The code is used to build a scatter point graph, where the x axis shows the number of days and the y axis shows the speed of the wind gusts. The geom_smooth(lm) function is get the linear line in the graph.

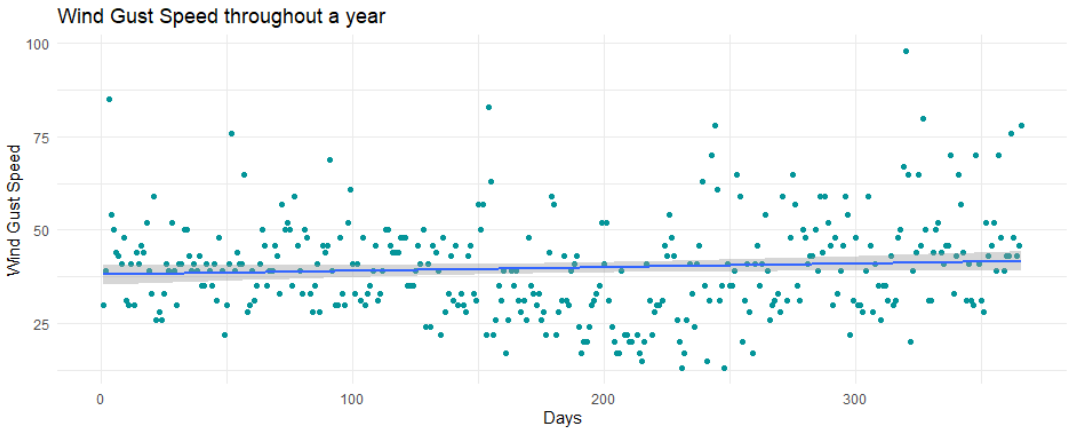

Figure 28: Output of analysis 3.1

The figure above shows that the average gust speed in the United States is 39.00 miles per hour. The highest possible value is 98, and the lowest possible value is 13.

Analysis 3.2 –Major Wind Gust Speed occurrence

```
#Wind Gust Direction Occurance
weather_data %>%
  drop_na(Wind_Gust_Dir) %>%
  ggplot(aes(fct_infreq(Wind_Gust_Dir)))+
  geom_bar(fill = "#097679")+
  coord_flip()+
  theme_bw()+
  labs(x = "Wind Gust Direction",
       y = "Occurance",
       title = "Number of Wind Gust Direction Occurance")
```

Figure 29: Analysis 3.2

The code is used to build a bar chart graph, with the x axis showing the direction of wind gusts and the y axis showing how often they happen. The fct_infreq function is used to make the number of occurrence of wind gust direction in ascending order. The coord_flip() function is used to turn the graph around so it can be seen better.

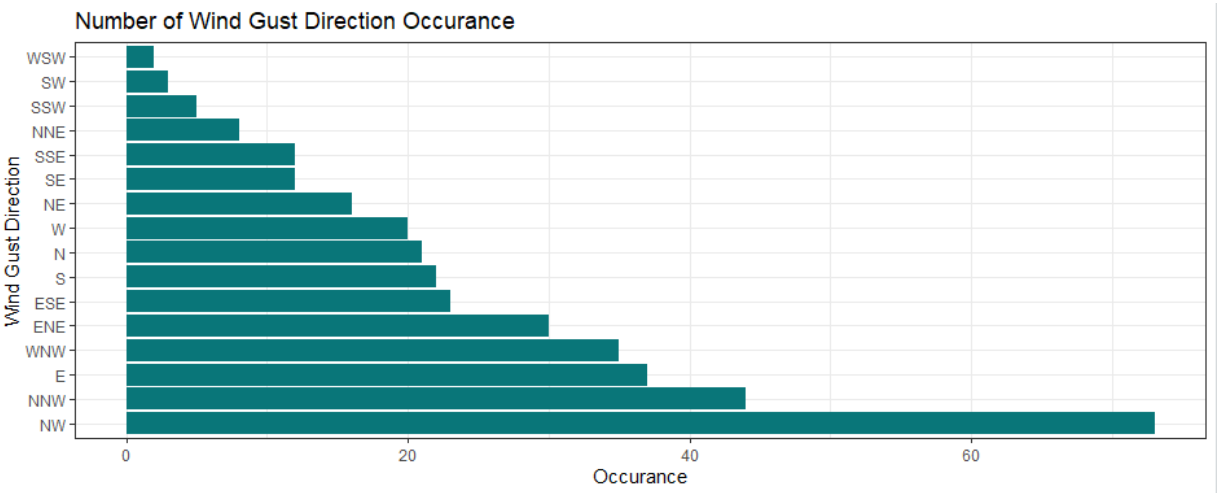

Figure 30: Output of analysis 3.2

The figure above shows that wind gusts are most likely to come from the NW and least likely to come from the WSW.

Analysis 3.3 –Highest wind gust speed for each wind gust direction

```
#High wind gust speed for each wind gust direction_
weather_data %>%
  drop_na(Wind_Gust_Speed, Wind_Gust_Dir) %>%
  ggplot(aes(Wind_Gust_Speed,Wind_Gust_Dir))+
  geom_point(colour = "#D9CE2A")+
  geom_line(size = 0.8)+
  theme_minimal()+
  labs(x = "Wind Gust Speed",
       y = "Wind Gust Direction",
       title = "Number of Wind Gust Speed for each Wind Gust Direction")
```

Figure 31: Analysis 3.3

The code is used to build a scatter point graph with line. The x-axis shows the speed of wind gusts, and the y-axis shows the direction of wind gusts. drop_na() fuction is to drop rows that contain missing values. The geom_line() function is used to put a line in between of each point.

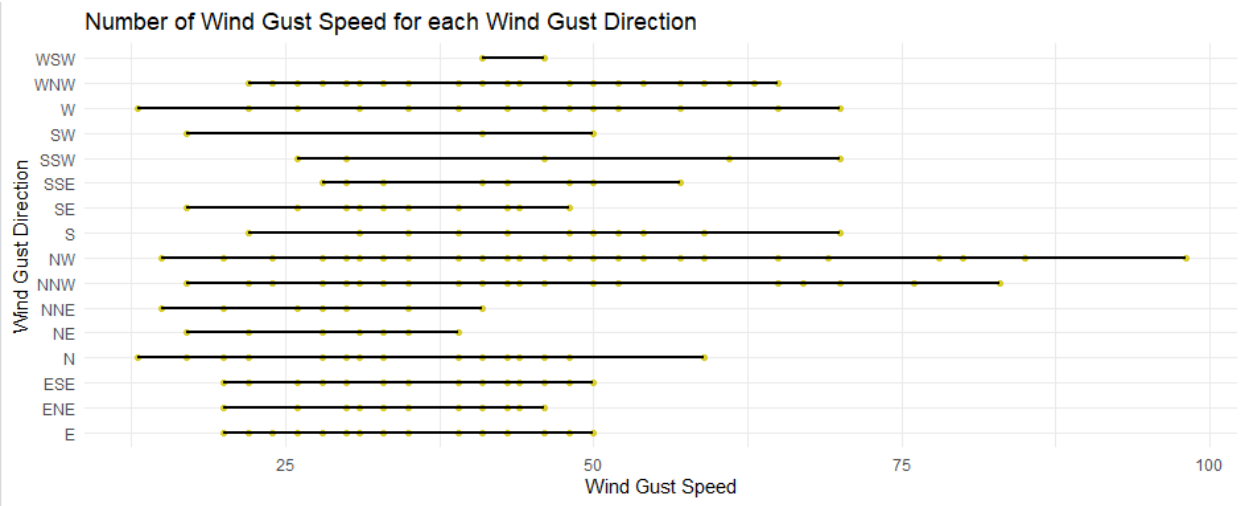

Figure 32: Output of analysis 3.3

The figure above shows that the wind gust speed is highest at the NW, at 98.00, and lowest at the W and N, at 13.00.

Analysis 3.4 –Average wind speed at 3pm when raining

```
#find average wind speed at 3pm when raining
weather_data %>%
  ggplot(aes(Rain_Today, Wind_Speed_3pm))+
  geom_boxplot(fill = "#42D2E3")+
  coord_flip()+
  theme_bw()+
  labs(x = "Rain Today",
       y = "Wind Speed",
       title = "Average wind speed at 3pm when raining")
```

Figure 31: Analysis 3.4

The code is used to build a boxplot graph. The x axis representing rainy and not rainy days and y representing how fast the wind speed is at 3pm. The coord_flip() function is used to rotate the graph for better view.

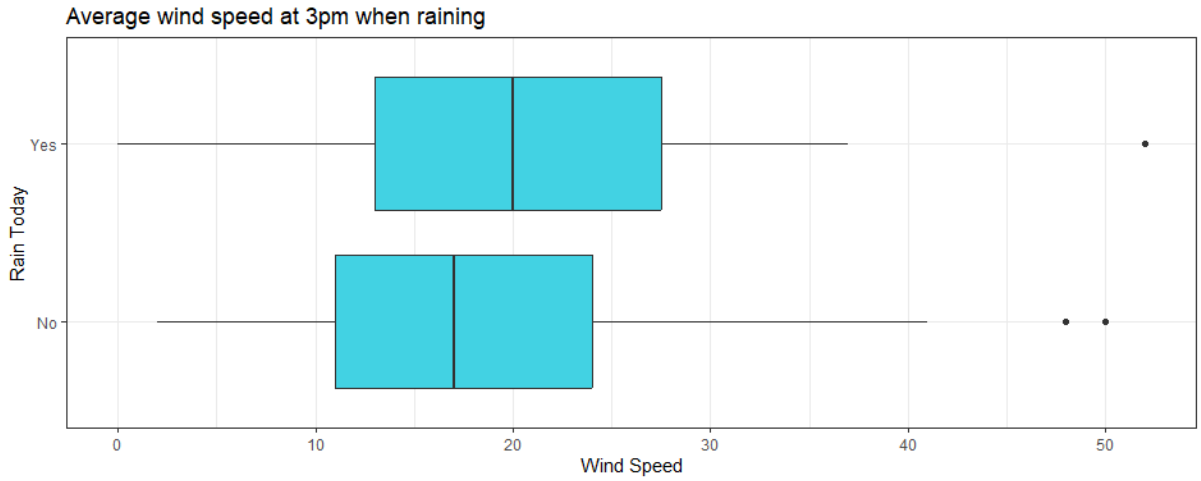

Figure 32: Output of analysis 3.4

The line in the box shows the average value, and the walls on the left and right show the lowest and highest values. The figure above shows that when it rains, the average wind speed is 20, which is faster than when it doesn't rain. In the same way, both the slowest and the fastest wind speeds are higher when it rains.

Analysis 3.5 –Wind gust direction when wind gust speed is over 50km/h while raining and identify the rain volume

```
#find wind gust dir when wind gust speed is over 40kmh while raining with rain volume
weather_data %>%
  filter(Wind_Gust_Speed > 50,
         Rain_Today != "No") %>%
  drop_na(Wind_Gust_Dir) %>%
  ggplot(aes(Wind_Gust_Dir, Wind_Gust_Speed))+
  geom_point(aes(color = Rain_Today,
                 size = Rain_Fall))+
  theme_bw()+
  coord_flip()+
  labs(x = "Wind Gust Direction",
       y = "Wind Gust Speed",
       title = "Wind Gust Direction when wind speed is above 50km/h")
```

Figure 33: Analysis 3.5

This code is used to construct a scatter plot graph. The x-axis shows where wind gusts are coming from, and the y-axis shows how fast they are moving. The filter() function is used to make sure that only days when it rains are shown when the wind gust speed is over 50 km/h. The geom_point() function is used with "aes" of "size" to show how much rain there is. The coord_flip() function rotates the x and y axes to make the graph easier to read.

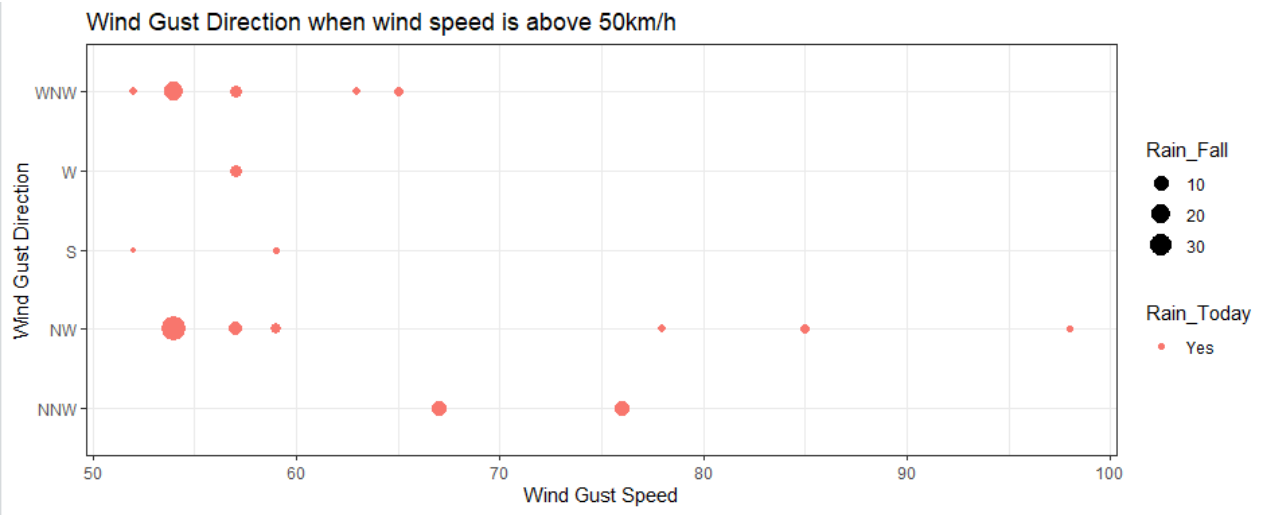

Figure 34: Output of analysis 3.5

The above scatter plot shows that when wind gusts are over 50 km/h, the strongest gusts are in the NW and the most rain falls are in the WMW and NW.

Question 4 Why no rain take majority in the United States?

Analysis 4.1 - Relationship between sunshine and evaporation

```
#Analysis 4.1  to find relationship between sunshine and evaporation
weather_data %>%
  ggplot(aes(Sunshine,Evaporation))+
  geom_point(color = "#18AEDA")+
  geom_smooth()+
  theme_bw()+
  labs(title="Relation between Sunshine and Evaporation")
```

Figure 35: Analysis 4.1

This code is used to construct a scatter graph to visualise whether or not the amount of sunshine has an impact on the amount of evaporation. The x axis represents the amount of sunshine, and the y axis represents the amount of evaporation.

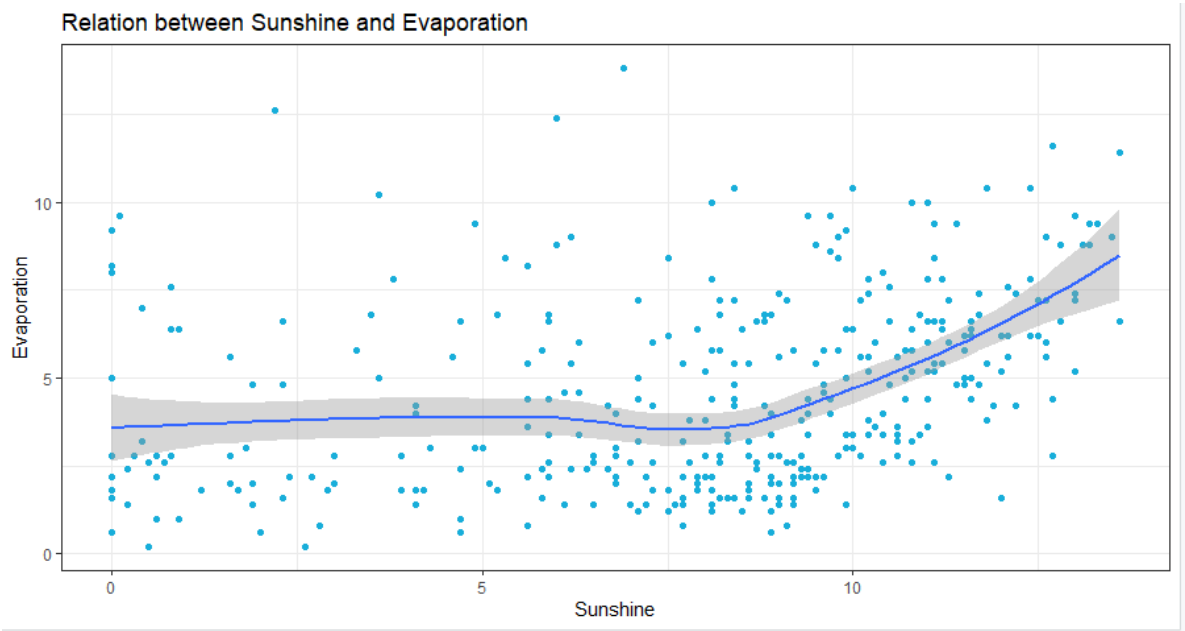

Figure 36: Output of analysis 4.1

The figure above shows that the rate of evaporation tends to rise when the sunshine level is close to 10 or higher. After the sun's intensity reaches a higher level, which causes a rise in the rate of evaporation, it is likely that it will begin to rain in the United States.

Analysis 4.2 Relationship between sunshine and humidity

```
#Analysis 4.2
#Relationship between sunshine and humidity
weather_data %>%
  ggplot(aes(Sunshine,Humidity_9am))+
  geom_point()+
  geom_smooth()+
  theme_bw()+
  labs(title = "Relation between Sunshine and Humidity")
```

Figure 37: Analysis 4.2

This code is used to construct a point graph that compares whether or not the amount of sunshine throughout the year has an influence on the humidity level. The x-axis of the graph indicate the amount of sunshine, and the y-axis will represent the humidity level at 9 a.m.

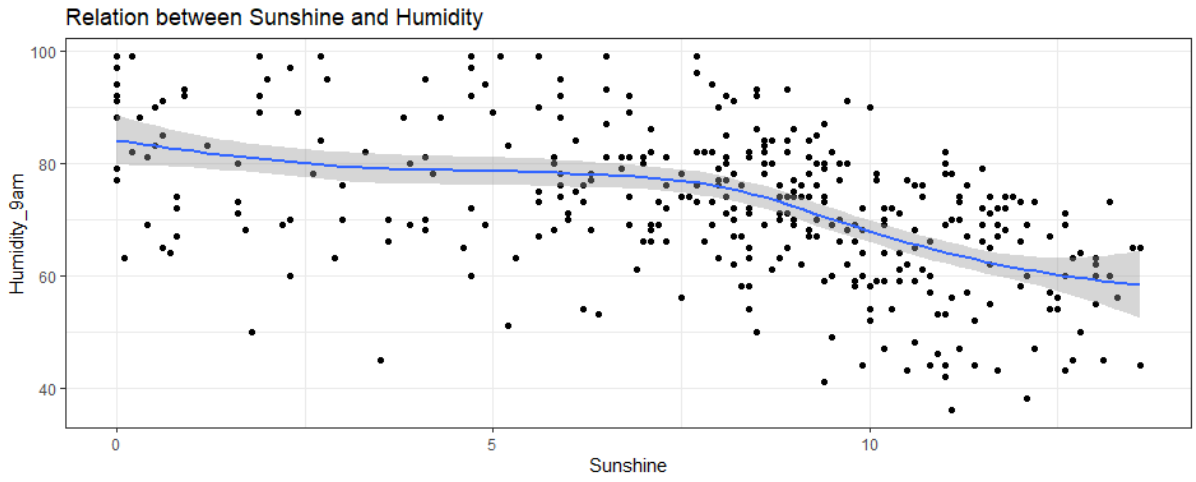

Figure 38: Output of analysis 4.2

The figure above shows that when the sunshine level is close to 10 or higher, the rate of humidity tends to go down.

Conclusion

In conclusion, I have investigated and analysed the data on the weather in the United States using a variety of methods. I did this by employing the ggplot2 tools in order to generate a visual graph and draw a conclusion. Bar charts, box plots, and scatter graphs with linear lines are representations of graphs that are used in this report. According to the data given, there are a greater number of days in the United States that do not contain rain than there are days that do contain rain. This could be many reasons and one of it is the temperatures, which impact the degree of evaporation and, as a result, cause fewer days with rain.

Reference

Wickham, H., Chang, W., & Wickham, M. H. (2016). Package 'ggplot2'. Create elegant data visualisations using the grammar of graphics. Version, 2(1), 1-189.

Dinea, E. (2021, November 24). PREDICTION OF RAIN TOMORROW IN AUSTRALIA. Jovian. Retrieved December 17, 2022, from https://jovian.ai/edaaydinea/prediction-of-raintomorrow

R Programming 101. (2021, February 2). Ggplot for plots and graphs. An introduction to data visualization using R programming [Video]. YouTube. https://www.youtube.com/watch?v=HPJn1CMvtmI&list=LL&index=6

In order for this submission to correctly submit and render inside the Turnitin Feedback Studio, it needs 20 selectable words as part of the document