Abstract

Data analysis is the process where a piece of data is inspected, cleansed, transformed and modeled with the intention of discovering meaningful information. Analysis of data makes study facile and more accurate by informing conclusions and supporting decision making. In this project, data analysis is done, using a dataset named ‘Hourly Weather Data’. The dataset provides weather information about two different airports in USA : John F. Kennedy (JFK) and LaGuardia Airport (LGA). It consists of 15 columns and about 17,412 rows in total. On the basis of ‘R Programming’, provided set of data is explored, manipulated and analyzed in detail. Necessary data packages are imported along with significant functions, which helped in studying the dataset enormously. Provided set of data is pre-processed in such a way that it provides quality result during analysis. A total of 16 weather analysis is done in the project, including two additional features.

Acknowledgement

First of all, I would like to manifest my heartfelt appreciation towards our subject teacher, **Mr. Sandip Adhikari** sir. His continual encouragement and positive motivation throughout this project has been huge. Without his productive guidance and assistance, I might not have been able to accomplish the project goal, with this level of comprehensive understanding. His methodology of carrying out the project guidance is extra efficacious and it has been a privilege to work under his enlightenment.

Furthermore, I would also like to convey my sincere thankfulness to APU (Asia Pacific University) for providing me with this wonderful opportunity to learn Information Technology and be a part of this project. Winding up, I would like to make use of this moment to demonstrate my deep regards to all my colleagues who helped me during this project.

Contents

1. Introduction and Assumptions

The course study assigns us to explore various data analysis methods and to perform operations such as Data Manipulation, Data Exploration and Data Visualization. A set of meteorological dataset from two different airports (i.e, LaGuardia and John F. Kennedy) is provided, which is analyzed in detail to redeem requisite data for decision making. The analysis is done using R Programming : a language for statistical computing, designed by statisticians Robert Gentleman and Ross Ihaka in 1993. With the usage of packages and libraries available in RStudio, the data is pre-processed, analyzed, visualized & explored accordingly, so that the task goal is accomplished.

Figure 01 : RStudio IDE for R programming (RStudio Logo Usage Guidelines, n.d.)

The provided dataset comprise of different attributes related to time, pressure, origin, wind, etc. There are a total of 15 columns and 17,412 rows consisting hourly data from LGA and JFK airport stations. Since airports are supposed to have exemplary conditions such as high visibility and weak windings, the goal is to analyze data, make required decisions and find about the actual condition through data results. Several manipulations related to wind, frequencies, directions, temperature and relations are calculated. The outcome produced is supposed to assist employees in airport take appropriate decisions. In time, Turnitin : a plagiarism detection tool, is also used to detect plagiarism offense during the submission of 'Weather Analysis Project'.

2. Key Terms

Analysis, Function geom(), Function ggplot(), Manipulation, RStudio, Visualization

3. Project Aims

This analysis aims to come up with enough evidences about the hourly weather data that will assist in recouping necessary information from JFK and LGA airports to make sensible decisions.

4. Project Objectives

- Identify data manipulation techniques to make extensive data well organized
- Implement visualization methodologies to communicate results more effectively
- Compare and contrast the dataset variables with each other and find meaningful outcomes
- Design and display the information retrieved from JFK and LGA airports to come up with consequential resolutions

5. Literature Review

R programming is often used for purposes like data mining and statistical analysis. There are hundreds and thousands of analysis conducted using R programming language. In an article from Team (2021), big tech giants such as Microsoft and Google uses R to improve their search engines.

A research from Petzoldt (2018) claims that they used R programming to explore, manipulate and visualize the dataset of seven lakes in northern part of Germany. They did used several basic R programming principles to find out what the characteristics of lakes

were, depending on climate change. In summary, they found out that around 88% of their variance fell into principle constituents.

Likewise, a dataset from Athletics Weekly (2006) was also studied to track most of the field events. This analysis was done back in 2006 where data were explored to accumulate information in data frame variables. Then, with necessary manipulation and visualization techniques, meaningful outcomes related to track events were recorded.

gbrathletics
www.gbrathletics.com
The best historical British athletics stats site
Home Page

WORLD RECORDS AND BEST PERFORMANCES
as at 9th August 2006
Compiled by Athletics Weekly

MEN'S TRACK & FIELD

100 m	9.77	Asafa Powell (JAM)	14 Jun 2005	Athens
	9.77	Justin Gatlin (USA)	12 May 2006	Doha
	9.77	Asafa Powell (JAM)	11 Jun 2006	Gateshead
150 m	14.93+	# John Regis (GBR)	20 Aug 1993	Stuttgart
	14.97	# Linford Christie ¶ (GBR)	4 Sep 1994	Sheffield
	14.8	# Pietro Mennea (ITA)	22 May 1983	Cassino
200 m	19.32	Michael Johnson (USA)	1 Aug 1996	Atlanta
300 m	30.85A	# Michael Johnson (USA)	24 Mar 2000	Pretoria
400 m	43.18	Michael Johnson (USA)	26 Aug 1999	Seville
500 m	1:00.08	# Donato Sabia (ITA)	26 May 1984	Busto Arsizio
600 m	1:12.81	# Johnny Gray (USA)	24 May 1986	Santa Monica
800 m	1:41.11	Wilson Kipketer (DEN)	24 Aug 1997	Cologne
1000 m	2:11.96	Noah Ngeny (KEN)	5 Sep 1999	Rieti
1500 m	3:26.00	Hicham El Guerrouj (MAR)	14 Jul 1998	Rome
1 mile	3:43.13	Hicham El Guerrouj (MAR)	7 Jul 1999	Rome
2000 m	4:44.79	Hicham El Guerrouj (MAR)	7 Sep 1999	Berlin
3000 m	7:20.67	Daniel Komen (KEN)	1 Sep 1996	Rieti
2 miles	7:58.61	# Daniel Komen (KEN)	19 Jul 1997	Hechtel
5000 m	12:37.35	Kenenisa Bekele (ETH)	31 May 2004	Hengelo
10000 m	26:17.53	Kenenisa Bekele (ETH)	26 Aug 2005	Brussels
15000 m	42:34.0+	# Arturo Barrios (MEX)	30 Mar 1991	La Flèche
20000 m	56:55.6+	Arturo Barrios (MEX)	30 Mar 1991	La Flèche
25000 m	1:13:55.8+	Toshihiko Seko (JPN)	22 Mar 1981	Christchurch
30000 m	1:29:18.8	Toshihiko Seko (JPN)	22 Mar 1981	Christchurch
50000 m	2:48:06	# Jeff Norman (GBR)	7 Jun 1980	Timperley
40 miles	3:48:35+	# Don Ritchie (GBR)	16 Oct 1982	London (He)
100 km	6:10:20	# Don Ritchie (GBR)	28 Oct 1978	London (CP)
100 miles	11:28:03	# Oleg Kharitonov (RUS)	20 Oct 2002	London (CP)
1 hour	21,101m	Arturo Barrios (MEX)	30 Mar 1991	La Flèche
2 hours	37,994m	# Jim Alder (GBR)	17 Oct 1964	Walton-on-Thames
24 hours	303.506km	# Yiannis Kouros (AUS)	5 Oct 1997	Adelaide
48 hours	473.797km	# Yiannis Kouros (AUS)	5 May 1996	Surgères
2000 m SC	5:14.43	# Julius Kariuki (KEN)	21 Aug 1990	Rovereto
3000 m SC	7:53.63	Saif Saeed Shaheen (QAT)	3 Sep 2004	Brussels
110 m Hurdles	12.88	Liu Xiang (CHN)	11 Jul 2006	Lausanne
200 m Hurdles	22.55	# Laurent Ottoz (ITA)	31 May 1995	Milan
	21.9y st	# Don Styron (USA)	2 Apr 1960	Baton Rouge
	22.5	# Martin Lauer (FRG/GER)	7 Jul 1959	Zürich

Figure 02 : Dataset of world records and best performances (Athletics Weekly, 2006)

Distinctive relations between variables such as roads, tracks, time and distance were analyzed using R programming techniques. Packages like ggplot and dplyr were brought in use whereas visualization were depicted in form of scatterplot matrices, model objects and generic functions.

6. Importing Data

Before we get into exploring and analyzing data, we need to import the data into the R environment. Firstly, we need to set the working directory so that a clear file path is constructed. Afterwards, we can import the data (CSV file) which is to be explored.

6.1 Setting up the working directory

To set the working directory, we need to use setwd( ) function. It is a built-in R function that creates a file path towards the folder where R project file and CSV file are located. In my case, the name of my project is given 'Hourly Weather Analysis' and is saved with .Rproj extension. The figure below displays the use of functions setwd( ) and getwd( ) so that the current directory is transformed into working directory.

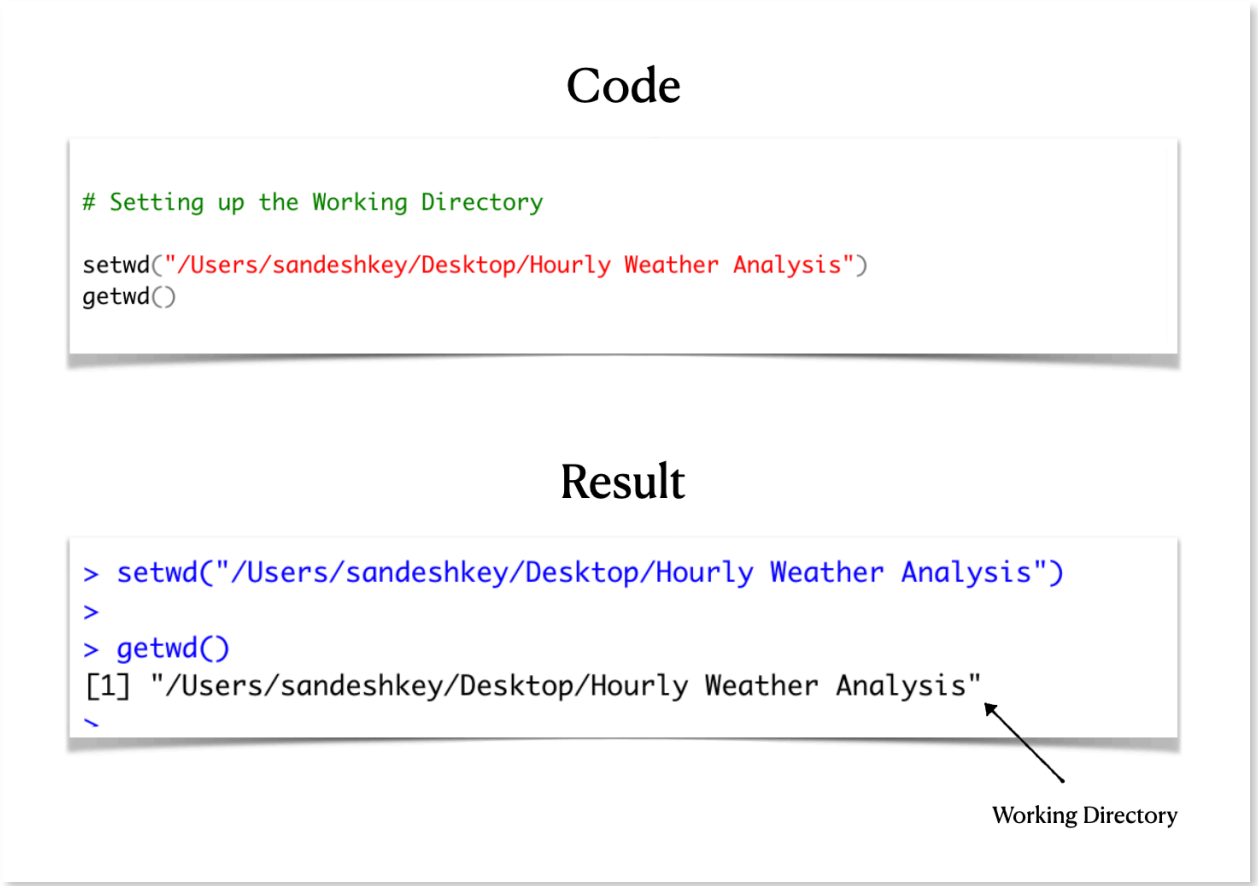

Figure 03 : Screenshot displaying the use of directory functions

6.2 Importing the CSV File (Hourly Weather Data.csv)

Now, to import the CSV file, we need to use read.csv( ) function. The name of CSV file is entered inside the function, incorporating with inverted commas. In my case, I have named the CSV file as 'Hourly Weather Data.csv'. Once the file is imported, we can then explore, manipulate and visualize the data according to requirements.

```
# Importing Data (CSV File : Hourly Weather Data)

weather_data <- read.csv("Hourly Weather Data.csv")

head(weather_data) #Displays top 6 rows from the data
tail(weather_data) #Displays bottom 6 rows from the data
View(weather_data) #Displays entire data in a spreadsheet
```

Figure 04 : Screenshot displaying the source code for data importing

In the figure 03 above, an object is created with name 'weather_data'. This is stored in R environment and can simply be used whenever we have to manipulate the CSV file. When the data is imported, there are some lines of codes used to view the imported data as shown in the figure. Moreover, there is also use of comments after codes, in order to improve code readability and to make future maintenance facile.

To explore the imported dataset further, we can view and summarize the data using respective functions. Some of the basic functions are:

a) Summary(dataset_name)
b) View(dataset_name)

Summary() is a built-in function, that sums up data functions and provide a brief output. Similarly, the View() function shows data in a separate spreadsheet which make it easy to analyze it.

```
> summary(weather_data)
   origin               year          month             day              hour            temp
Length:17412       Min.   :2013   Min.   : 1.000   Min.   : 1.00   Min.   : 0.00   Min.   :12.02
Class :character   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.: 6.00   1st Qu.:39.92
Mode  :character   Median :2013   Median : 7.000   Median :16.00   Median :11.00   Median :55.04
                   Mean   :2013   Mean   : 6.504   Mean   :15.68   Mean   :11.49   Mean   :55.12
                   3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:17.00   3rd Qu.:69.98
                   Max.   :2013   Max.   :12.000   Max.   :31.00   Max.   :23.00   Max.   :98.96

     dewp            humid           wind_dir       wind_speed       wind_gust         precip
Min.   :-9.94   Min.   : 12.74   Min.   :  0.0   Min.   : 0.000   Min.   :16.11   Min.   :0.000000
1st Qu.:26.06   1st Qu.: 46.85   1st Qu.:120.0   1st Qu.: 6.905   1st Qu.:21.86   1st Qu.:0.000000
Median :42.08   Median : 61.15   Median :220.0   Median :10.357   Median :25.32   Median :0.000000
Mean   :41.23   Mean   : 62.26   Mean   :201.9   Mean   :11.046   Mean   :26.18   Mean   :0.004183
3rd Qu.:57.02   3rd Qu.: 78.66   3rd Qu.:300.0   3rd Qu.:14.960   3rd Qu.:29.92   3rd Qu.:0.000000
Max.   :78.08   Max.   :100.00   Max.   :360.0   Max.   :42.579   Max.   :66.75   Max.   :0.820000
                                 NA's   :204     NA's   :3        NA's   :13877
   pressure          visib        time_hour            snow
Min.   : 983.8   Min.   : 0.000   Length:17412       Length:17412
1st Qu.:1013.5   1st Qu.:10.000   Class :character   Class :character
Median :1017.9   Median :10.000   Mode  :character   Mode  :character
Mean   :1017.9   Mean   : 9.245
3rd Qu.:1022.3   3rd Qu.:10.000
Max.   :1042.1   Max.   :10.000
```

Figure 05 : Screenshot displaying the summary of imported data

Weather_Analysis.R | weather_data | Weather_Analysis_R.R*

Filter

	origin	year	month	day	hour	temp	dewp	humid	wind_dir	wind_speed	wind_gust	precip	pressure	visib	time_hour
1	JFK	2013	1	1	1	-32.47	-33.71	59.37	260	12.65858	0.00000	0	1012.600	10	01/01/2013 01:00
2	JFK	2013	1	1	2	-32.47	-33.71	59.37	270	11.50780	0.00000	0	1012.400	10	01/01/2013 02:00
3	JFK	2013	1	1	3	-32.38	-33.62	59.50	260	14.96014	0.00000	0	1012.700	10	01/01/2013 03:00
4	JFK	2013	1	1	4	-32.38	-33.52	62.21	250	17.26170	0.00000	0	1012.600	10	01/01/2013 04:00
5	JFK	2013	1	1	5	-32.47	-33.62	61.63	260	14.96014	0.00000	0	1012.100	10	01/01/2013 05:00
6	JFK	2013	1	1	6	-32.57	-33.62	64.29	260	13.80936	0.00000	0	1012.600	10	01/01/2013 06:00
7	JFK	2013	1	1	7	-32.47	-33.52	64.43	260	13.80936	0.00000	0	1012.500	10	01/01/2013 07:00
8	JFK	2013	1	1	8	-32.38	-33.62	59.50	260	17.26170	0.00000	0	1012.600	10	01/01/2013 08:00
9	JFK	2013	1	1	9	-32.38	-33.62	59.50	260	16.11092	0.00000	0	1013.000	10	01/01/2013 09:00
10	JFK	2013	1	1	10	-32.28	-33.52	59.65	260	16.11092	0.00000	0	1012.800	10	01/01/2013 10:00
11	JFK	2013	1	1	11	-32.28	-33.62	57.06	270	14.96014	0.00000	0	1011.700	10	01/01/2013 11:00
12	JFK	2013	1	1	13	-32.57	-33.66	64.70	340	14.96014	0.00000	0	1017.932	10	01/01/2013 13:00
13	JFK	2013	1	1	14	-32.47	-33.89	54.68	310	11.50780	0.00000	0	1011.200	10	01/01/2013 14:00
14	JFK	2013	1	1	15	-32.47	-34.00	52.26	290	12.65858	0.00000	0	1011.700	10	01/01/2013 15:00
15	JFK	2013	1	1	16	-32.57	-34.48	44.00	320	17.26170	24.16638	0	1012.100	10	01/01/2013 16:00
16	JFK	2013	1	1	17	-32.66	-34.57	43.85	330	16.11092	25.31716	0	1013.200	10	01/01/2013 17:00
17	JFK	2013	1	1	18	-32.85	-34.86	41.51	310	14.96014	0.00000	0	1014.200	10	01/01/2013 18:00
18	JFK	2013	1	1	19	-33.04	-34.86	44.92	350	13.80936	24.16638	0	1014.300	10	01/01/2013 19:00
19	JFK	2013	1	1	20	-33.14	-34.86	46.92	330	16.11092	0.00000	0	1015.100	10	01/01/2013 20:00
20	JFK	2013	1	1	21	-33.33	-35.15	44.41	330	21.86482	29.92028	0	1015.300	10	01/01/2013 21:00
21	JFK	2013	1	1	22	-33.43	-35.15	46.41	320	16.11092	0.00000	0	1016.400	10	01/01/2013 22:00

Figure 06 : Screenshot displaying the spread-sheet view of imported data

7. Installing and Loading Packages

7.1 Installing Packages

Packages are basically the stack of sample data, built-in codes and functions. In R, there are plenty of packages with each providing distinctive functionalities. There are few packages which gets installed while installing the language. However, to access more packages according to the project requirement, we need to install them using the function 'install.packages()' in RStudio.

```
# Installing Packages

install.packages("ggplot2")
install.packages("dplyr")
install.packages("crayon") # Colored Terminal Output
install.packages("magrittr") # Pipe Operator
install.packages("hexbin") # Extra Feature
install.packages("weathermetrics") # Used in pre processing temp and dewp
install.packages("hexbin") # Extra Feature
```

Figure 07 : Screenshot showing package installation in RStudio

7.2 Loading Packages

To use a packages in our script, we need to load them beforehand. In R, there are two easy functions that are used to load packages. They are :

c) library(package_name)
d) require(package_name)

```
# Loading Packages for their use ( Using "library()" or "require()" function)
library(ggplot2)
library(dplyr)
library(crayon)
library(magrittr)
library(corrplot)
library(hexbin)
library(weathermetrics)
library(hexbin)
```

Figure 08 : Screenshot showing package loading in RStudio

8. Data Preprocessing

Data preprocessing is a crucial phase where we prepare our data systematically, ensuring they are accurate and required. There are several steps in data preprocessing to be followed in order to make our data useful and precise during analysis.

Finding the missing data

```
# Data Pre-processing

# 1. Finding all the missing values from the dataset

sum(is.na(weather_data$origin))
sum(is.na(weather_data$year))
sum(is.na(weather_data$month))
sum(is.na(weather_data$day))
sum(is.na(weather_data$hour))
sum(is.na(weather_data$temp))
sum(is.na(weather_data$dewp))
sum(is.na(weather_data$humid))
sum(is.na(weather_data$wind_dir))
sum(is.na(weather_data$wind_speed))
sum(is.na(weather_data$wind_gust))
sum(is.na(weather_data$precip))
sum(is.na(weather_data$pressure))
sum(is.na(weather_data$visib))
sum(is.na(weather_data$time_hour))
```

Figure 09 : Screenshot displaying the source code for finding missing(NA) values

In order to identify the missing or unavailable values (NA), we use is.na function. As shown in the figure above, the is.na( ) function recognizes all the missing values from the dataset. Likewise, the sum( ) function used ahead of is.na( ) is used to add the missing value data of a specific attribute. Using this function, we found out that there were few variables missing values. These variables will be preprocessed, either by replacing with mean or simply adding a zero value depending on the data.

8.1 Preprocessing Wind Direction (wind_dir)

```
# Data Preprocessing : Wind Direction (wind_dir)

avg_Wind_Dir <- mean(weather_data$wind_dir,
                     na.rm = TRUE)
weather_data$wind_dir <- ifelse(
  is.na(weather_data$wind_dir),
  avg_Wind_Dir,
  weather_data$wind_dir)
sum(is.na(weather_data$wind_dir))
```

Figure 10 : Screenshot displaying data preprocessing of wind direction

After calculation, the number of unavailable rows in wind direction crossed 200. To avoid this issue of unavailable values, I simply replaced those vacant (N/A) with mean of wind direction attribute which contains legit values. This way, we can avoid losing any sort of valuable data and make sure of the data quality as well. The first step is to identify the mean value of available values, which is 201.9. Then, a new object ‘avg_Wind_Dir' is declared for the mean value and with if_else function, all the unavailable values are replaced with mean values. After all, a final check is done to ensure there is no missing value in that specific column.

8.2 Preprocessing Wind Speed (wind_speed)

```
# Data Pre processing : Wind Speed (wind_speed)

avg_Wind_Speed <- mean(weather_data$wind_speed,
                       na.rm = TRUE)
weather_data$wind_speed <- ifelse(
  is.na(weather_data$wind_speed),
  avg_Wind_Speed,
  weather_data$wind_speed)
sum(is.na(weather_data$wind_speed))
```

Figure 11 : Screenshot displaying the data preprocessing wind speed

In case of wind speed, a total of just 3 rows were unavailable. However, to make sure the quality of data remains stable, a similar process as of wind direction was followed in this case as well. Firstly, the mean of available values was calculated, which is 11.046. Then, a variable named 'avg_Wind_Speed' was created and with the implementation of if_else function, all unavailable values were replaced with the mean value.

8.3 Preprocessing Wind Gust (wind_gust)

```
# Data Pre processing : Wind Gust (wind_gust)

weather_data$wind_gust <- ifelse(
  is.na(weather_data$wind_gust),
  0,
  weather_data$wind_gust)
sum(is.na(weather_data$wind_gust))

avg_Wind_Gust <- weather_data %>%
  group_by(origin, month) %>%
  summarise(
    avg_Temp = mean(temp),
    wind_Gust_1 = mean(wind_gust, na.rm = TRUE)
  )
avg_Wind_Gust
```

Figure 12 : Screenshot displaying the data preprocessing wind gust

Unlike other attributes, the wind gust attribute comprise of a lot of missing values i.e, 13877. This is 79.69% of missing values from a single column. Therefore, replacing all the missing values with mean might not be ideal as we might get data with dramatic changes. In this case, missing values will simply be replaced with zero (0), so that we get to eliminate all observations of unavailable data. Here, the if-else function is used in such a way that it replaces unavailable values with 0 and leave other observations the way they are.

8.4 Preprocessing Temperature : Fahrenheit to Celsius conversion

```
# Data Preprocessing : Temperature (temp) and Dewpoint (dewp)

# Temperature conversion from Fahrenheit to Celsius
weather_data$temp <- fahrenheit.to.celsius(weather_data$temp)
weather_data$dewp <- fahrenheit.to.celsius(weather_data$dewp)
```

Figure 13 : Screenshot displaying the source code of temperature conversion

While installing packages into R environment earlier, we also imported a package named 'weathermetrics'. This is where it will be used. Since the temperature is given in Fahrenheit (F), the purpose of this preprocessing is to convert temperature into Celsius. From the installed package, we can simply use fahrenheit.to.celsius( ) function which will transform all the values from Fahrenheit to Celsius.

8.5 Preprocessing Mutation : Addition of a new column

```
# Data Mutation : Addition of a new column in imported dataset

weather_data = mutate(weather_data, snow =
                        if_else((month==3&day>=20)|month==4|month==5|(month==6&day<=19),"snow",
                                if_else((month==6&day>=20)|month==7|month==8|(month==9&day<=19),"partial snow",
                                        if_else((month==9&day>=20)|month==10|month==11|(month==12&day<=19),"snow","no snow"))))

View(weather_data)
```

Figure 14 : Screenshot displaying the source code of column (variable) mutation

This is a simple preprocessing technique which uses mutate function to add a new column in the dataset. Here, a data related to snowfall is added to the imported file, using mutate( ) function. Then, if-else functions new used multiple time, to specify data as instructed (i.e, by seasonal months).

9. R Functions

Throughout the analysis, a considerable number of R programming functions have been used. Some of the functions were available in-built within the language, whereas a some required package installation for the access. The frequently used functions during the analysis and their use are listed below :

a) summary( ) :

Summary() is a built-in generic function that computes result summaries of data.

b) mean( ) :

The mean( ) function computes the average of a particular data elements.

c) select( ) :

The select function comes under dplyr package and is used for selecting variables or columns based on analysis circumstances.

d) filter( ) :

The filter( ) function also comes under dplyr package and is used for selection of rows according to required values.

e) mutate( ) :

Mutate is another function of dplyr package which is used to create new variables.

f) group_by( ) :

The group_by( ) function also falls under dplyr package. It is used in grouping the data frame on the basis of input variables.

g) ggplot( ) :

The use of ggplot( ) function is to proclaim input data frame to create graphics. It is also used in initializing ggplot object.

h) geom_line( ) :

The geom_line( ) function in R is used to link specific rows, as structured by x value. It is used while creating line graph visualization as well.

i) geom_boxplot( ) :

The geom_boxplot() function demonstrates the dispensation of incessant variables. It is uses while creating box plot during analysis.

j) geom_smooth( ) :

The geom_smooth() function appends a smooth line of regression which aids eye in viewing patterns.

k) cor( ) :

The use of cor( ) function is to find the value of correlation coefficient between two vectors.

l) labs( ) :

The labs( ) function in R is used to revise axis and plot customized labels to graphs

m) scale_x_continous( ) :

This is a default scale for continuous x axis. It is used to provide aesthetic values to x-axis default elements

n) theme( ) :

The use of theme( ) function is to customize graph plots during visualization.

o) facet_wrap( ) :

The function of facet_wrap is to wrap 1d ribbon panels into 2d. It helps in creating multi-panel graph plots during analysis.

Extra Functions

a) geom_hex( ) :

The geom_hex( ) function produce a hexagonal heat map by dividing ribbon planes into hexagons.

b) geom_density( ) :

The geom_density( ) function is used in extra feature of my analysis. It is used to draw a density plot according to provided values.

10. Data Analysis

Once all the necessary packages are installed and data are pre processed, it is time to jump into data analysis. During analysis, there comes tasks like exploration, manipulation and visualization of data. The provided hourly dataset (weather_data) is studied, analyzed and compared in detail to get meaningful outcomes. Moreover, suitable graphs are plotted with the intention of making analysis easier and convenient while making decisions. The components provided as variables will be studied to find relationships in between them.

Analysis 01 : Analyzing Dew Point against Humidity using Scatterplot

In this analysis, relationship between dew point and humidity is studied with the help of scatterplot diagram. We are given the dew point temperature and humidity of both JFK and LGA airports. Dew point is a temperature required by air to cool whereas humidity is concentration of vapor in air. The code below is used to create a scatterplot in displaying relation between dew point and humidity.

```
# Analysis Example 01 : Analyzing Dew Point against Humid using Scatter plot

# Manipulation
weather_data %>%
  summarise(dewp)

# Visualization
ggplot(data = weather_data, mapping = aes(x = dewp, y = humid, color = origin)) +
  geom_point(alpha = 0.2) +
  stat_smooth(method = "lm") +
  labs(title = "Scatter plot between Dew Point against Humid", x = "Dew Point", y = "Humid") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 15 : Source code to create scatterplot between dew point and humidity

Firstly, the object or provided CSV file is manipulated using pipe operator. Then, it is summarized to get the quality value from data observation. In visualization, the ggplot( ) function is used with aes( ), where variable names are inserted.

The graph color is differentiated in terms of origin (i.e, JFK and LGA) whereas method = “lm” is also used for plotting a linear line. Before that, a function named stat_smooth( ) is used which detects the relation between inserted variables. Title and labels to both x and y axis are provided using labs( ) function, which is then customized with theme( ) function. Inside theme function, ‘hjust’ is used to align the title above the graph whereas size and color refers to font size and title color respectively.

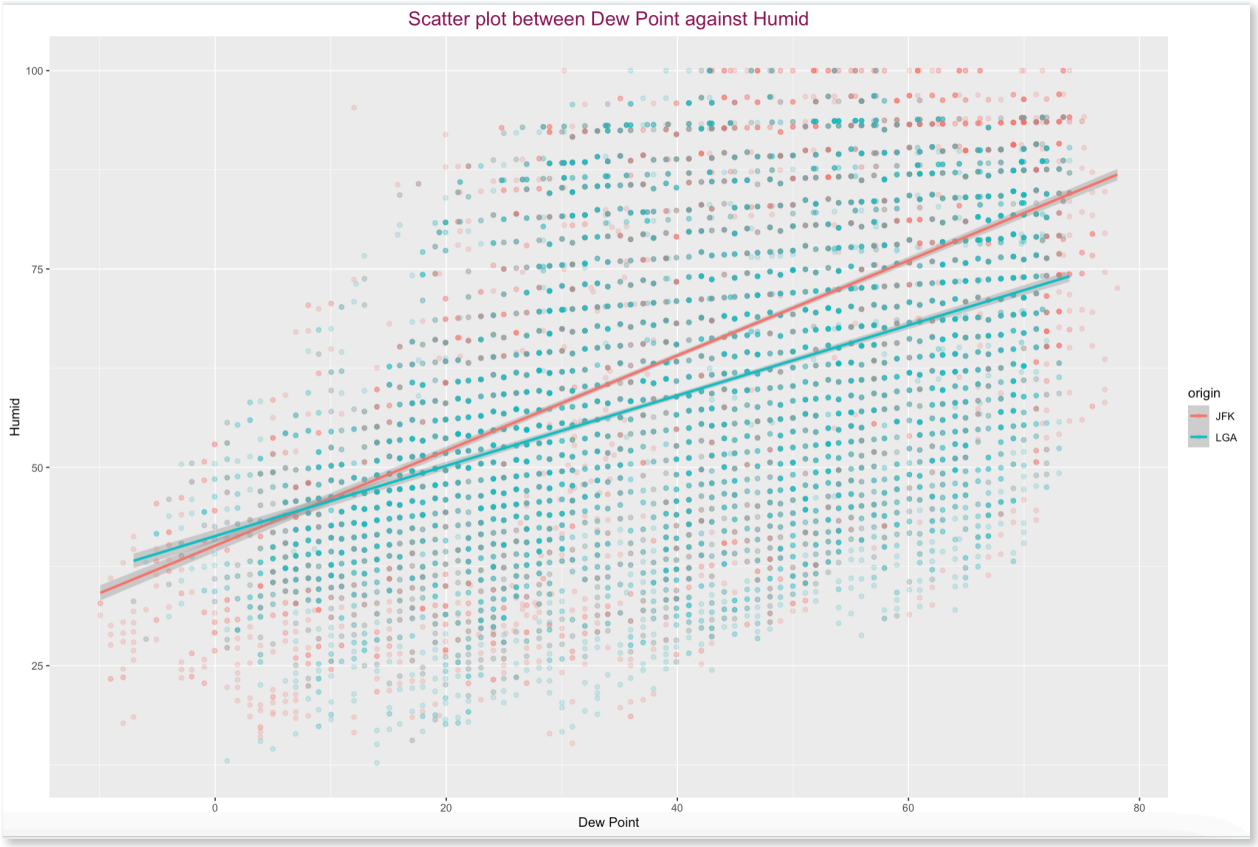

Figure 16 : Scatterplot produced between dew point and humidity

The figure above is a output produced in form of scatterplot. It demonstrates how dew point and humidity were seen in JFK and LGA airports in 2013. From the figure, the reddish scattered plots exhibit data from JFK airport whereas the bluish plots are of LGA airport. Observing the scatterplot, we can analyze the direct relationship between dew point and humidity. When the dew point temperature goes higher, the humidity rises as well. The regression line shows that JFK had the average of -10 degrees of dew point and around 40 gm/cm-3 humidity. Similarly, LGA seems to be in slightly high position of around -7 degrees and 43 gm/cm-3 humidity. With time, both JFK and LGA airports experienced increase in both humidity and temperature. The increase in dew points refers to high altitude of density. This affects the performance of airplanes in airports. So, this data helps in making a sensible decision.

Analysis 02 : Analyzing snowfall based on temperature, using box plot

In this analysis, data regarding snowfall is analyzed based on airport's temperature. For this, we need to consider the new column that was added into the dataset during data pre processing.

```
# Analysis Example 2 : Analyzing snowfall based on the temperature with Boxplot

# Manipulation
weather_data %>%
  group_by(snow) %>%
  summarise(avg_temper = mean(temp),
            max_temper = max(temp),
            min_temper = min(temp),
            .groups = 'drop')

# Visualization
ggplot(weather_data, aes(snow, temp, fill = snow)) +
  geom_boxplot(aes(group = snow)) +
  labs(title = "Snowfall Based on Temperature", y = "Temperature (In Celsius)") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 17 : Source code to create a box plot of snowfall based on temperature

Firstly, we need to manipulate the data from the weather_data file. Once the data is grouped based on 'snow' variable, we can then summarize mean, minimum and maximum value of that specified data. There are three new objects created inside the summaries( ) function : avg_temper for mean of temperature, max_temper for maximum value of temperature and min_temper for minimum value of temperature within that data. Once this is done, R provides a numeric results displaying the output values.

	snow	avg_temper	max_temper	min_temper
	<chr>	<dbl>	<dbl>	<dbl>
1	no snow	-16.3	-6.33	-23.9
2	partial snow	-4.37	2.89	-12.6
3	snow	-11.0	1.06	-23.6

Figure 18 :Numeric outcome of snowfall after data manipulation

Then comes data visualization where ggplot function is used with aesthetics to provide necessary parameters. The geom_boxplot function is used by grouping it with snow variable followed by labels and themes for customized labelling.

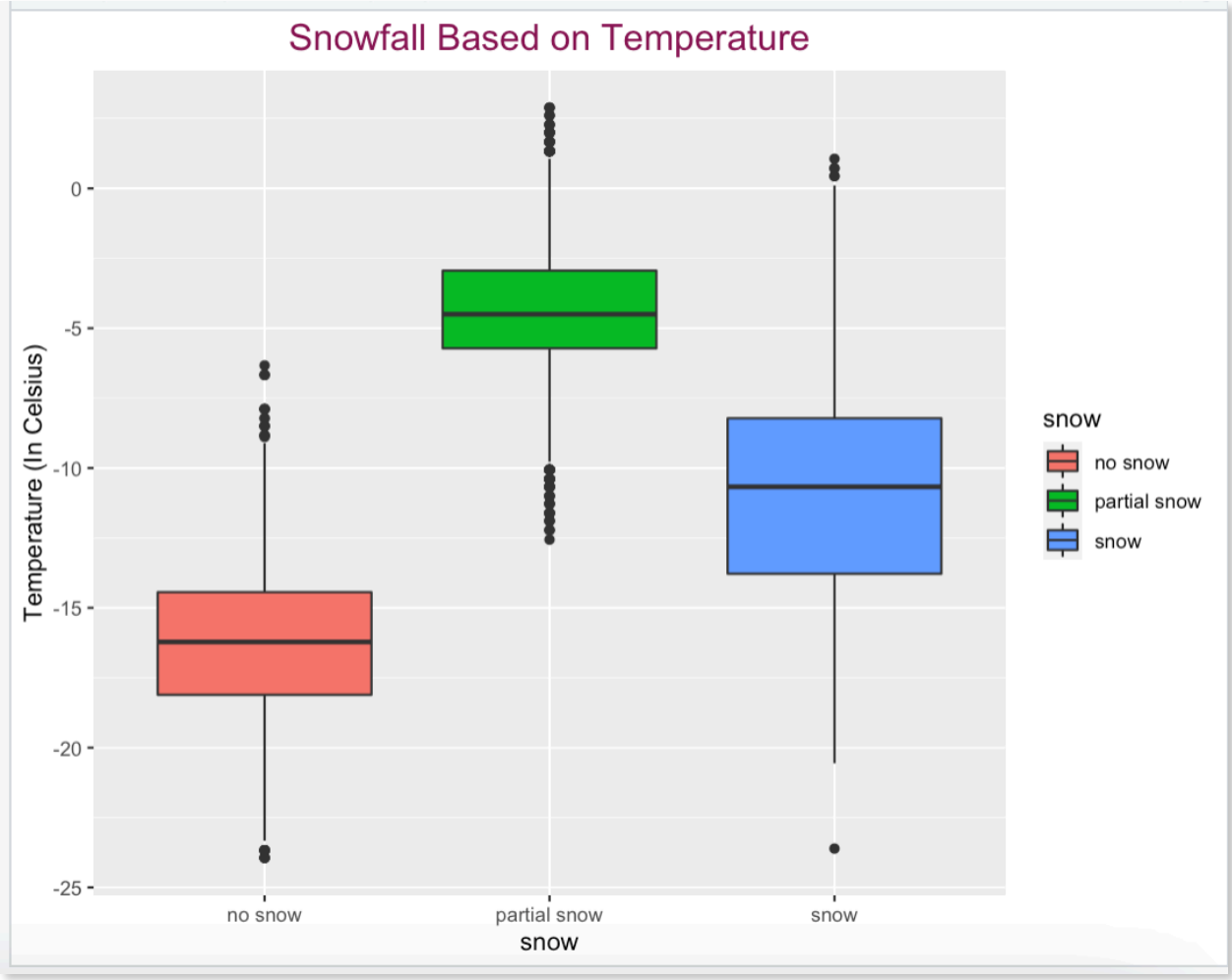

Figure 19 : Box plots of snowfall based on airport temperature

The outcome above is nothing but the graphical representation of numeric result. Based on the dataset observations, three different boxes are created. The observation with 'no snow' is depicted by reddish box whereas observations with 'partial snow' and 'snow' are depicted with green and blue boxes respectively. From the diagram, it is clear that the average temperature is highest during non-snowy days. Similarly, the minimum

temperature during 'no-snow' period goes to -23.9°C and maximum temperature reaches -6.33°C. Likewise, the minimum temperature of snow goes down to -23.6°C and reaches up to 1.06°C in maximum.

*(Note : The **'snow'** variable is mutated just to demonstrate the knowledge of data analysis. Since it is not the real data, the outcome might not match real-life scenario)*

Analysis 03 : Analyzing Average Pressure at both airports for each month using Line Graph

In this analysis, average pressure in both JFK and LGA airports is analyzed using a line graph. The analysis displays the pressure in both airports so that a fair comparison can be made. The outcome is expected to deliver month by month changes in year 2013.

```
# Analysis Example 3 : Analyzing Average Pressure at both airports for each month using Line Graph

# Data Manipulation
pressure_LGA <- weather_data %>%
  filter(origin == "LGA") %>%
  group_by(month) %>%
  summarise(pressure = mean(pressure), origin, .groups = 'drop')
summary(pressure_LGA)

# Data Manipulation
pressure_JFK <- weather_data %>%
  filter(origin == "JFK") %>%
  group_by(month) %>%
  summarise(pressure = mean(pressure), origin, .groups = 'drop')
summary(pressure_JFK)

# Visualization
ggplot() +
  geom_line(data = pressure_LGA, aes(x = month, y = pressure, color = origin)) +
  geom_line(data = pressure_JFK, aes(x = month, y = pressure, color = origin)) +
  scale_x_continuous(breaks = 1:12, labels = c("Jan", "Feb", "Mar", "Apr",
                                               "May", "Jun", "Jul", "Aug",
                                               "Sep", "Oct", "Nov", "Dec")) +
  labs(title = "Pressure in LGA and JFK (by months)",
       x = "Month", y = "Pressure (Pa)") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 20 : Source code to display average pressure of both airports with line graph

In the source code above, data manipulation of pressure in both airports is done. Firstly, objects named 'pressure_LGA' and 'pressure_JFK' is declared for two different airports. Then, the weather_data is filtered on the basis of 'origin' & grouped by 'month'. This will provide a numerical outcome which is shown below :

```
> # Data Manipulation
>
> pressure_JFK <- weather_data %>%
+   filter(origin == "JFK") %>%
+   group_by(month) %>%
+   summarise(pressure = mean(pressure), origin, .groups = 'drop')
> summary(pressure_JFK)
     month           pressure       origin
 Min.   : 1.000   Min.   :1014   Length:8706
 1st Qu.: 4.000   1st Qu.:1017   Class :character
 Median : 7.000   Median :1018   Mode  :character
 Mean   : 6.504   Mean   :1018
 3rd Qu.: 9.000   3rd Qu.:1020
 Max.   :12.000   Max.   :1022
>
```

Figure 21 :Numeric outcome of average pressure in JFK

```
> # Data Manipulation
>
> pressure_LGA <- weather_data %>%
+   filter(origin == "LGA") %>%
+   group_by(month) %>%
+   summarise(pressure = mean(pressure), origin, .groups = 'drop')
> summary(pressure_LGA)
     month           pressure       origin
 Min.   : 1.000   Min.   :1014   Length:8706
 1st Qu.: 4.000   1st Qu.:1017   Class :character
 Median : 7.000   Median :1018   Mode  :character
 Mean   : 6.504   Mean   :1018
 3rd Qu.: 9.000   3rd Qu.:1019
 Max.   :12.000   Max.   :1022
>
```

Figure 22 :Numeric outcome of average pressure in LGA

Once the manipulation is done, the next step is to perform data visualization. The geom_line( ) function is used in creation of line graph. Aesthetic mappings are provided to deploy month and pressure as axes. Moreover, in order to make result understandable and more informative, function scale_x_continuous( ) is used. With this function, the name of months are inserted in graph which is followed by customized labels and title.

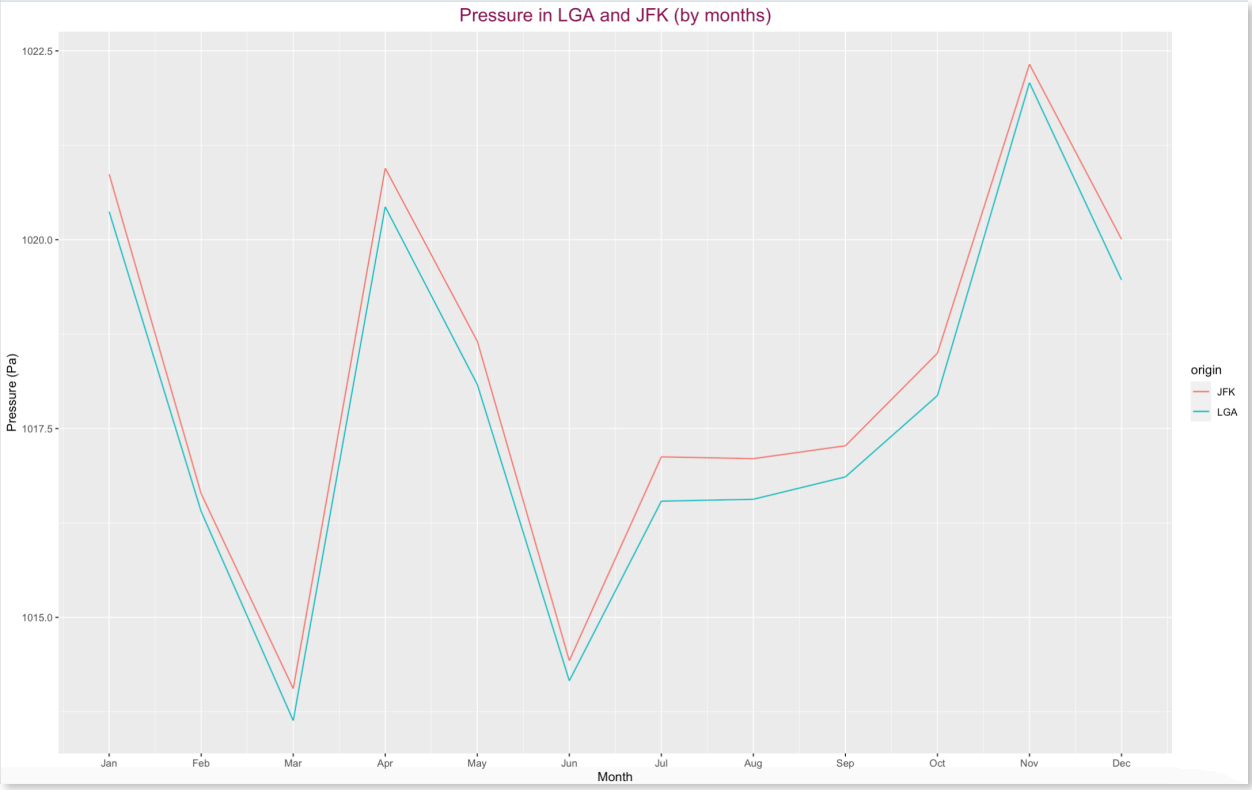

Figure 23 : Line graph of average pressure in JFK and LGA airports

From the graph above, the red line denotes the data of JFK whereas the bluish line denotes the data of LGA airport. Analyzing the graph, we can see that the minimum pressure of JFK and LGA was around 1014 Pa. With LGA being slightly lower. Similarly, the maximum pressure was around 1022 with JFK being slightly over by few decimals. The average pressure was calculated to be 6.504. This shows the pressure in both airports are parallel during inclination as well as declination. At minimum pressure, the density or air decreases gradually. This also means density altitude which affects flight performance.

Analysis 04 : Analyzing relation between Temperature and Wind Gust using Regression Equation Line

In this analysis, relation between temperature and wind gust is analyzed using the regression equation line. The outcome of temperature is displayed in °C whereas the gust is displayed in Mph.

```
# Analysis Example 04 : Analyzing relation between Temperature and Wind Gust using Regression Equation Line

# Manipulation
avg_Wind_Gust <- weather_data %>%
  group_by(origin, month) %>%
  summarise(
    avg_Temp = mean(temp),
    wind_Gust_1 = mean(wind_gust, na.rm = TRUE)
  )
avg_Wind_Gust

# Visualization
ggplot(avg_Wind_Gust, aes(x = avg_Temp, y = wind_Gust_1)) +
  geom_point(color = 'red') +
  geom_smooth(method = lm, formula = y ~ x, color = 'orchid', size = 1) +

  labs(title = "Wind Gust and Temperature Regresssion Line",
       x = "Temperature (In C)", y = "Wind Gust (In Mph)") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 24 : Source code to display relation between temperature and wind gust

At first, an object named ‘avg_Wind_Gust’ is declared to manipulate weather_data. Then, the data is grouped on the basis of origin and month. Once summarized, new objects are created to store the mean value of both temperature and wind gust. Then, for visualization, ggplot function is used with geom_point in order to plot the graph with regression points. Aesthetic mappings are present where avg_Temp and wind_Gust are inserted as parameters. Function geom_smooth( ) is also used to make graphic aid eye during analysis. Finally, labs( ) and theme( ) functions are used to label axes and customize titles.

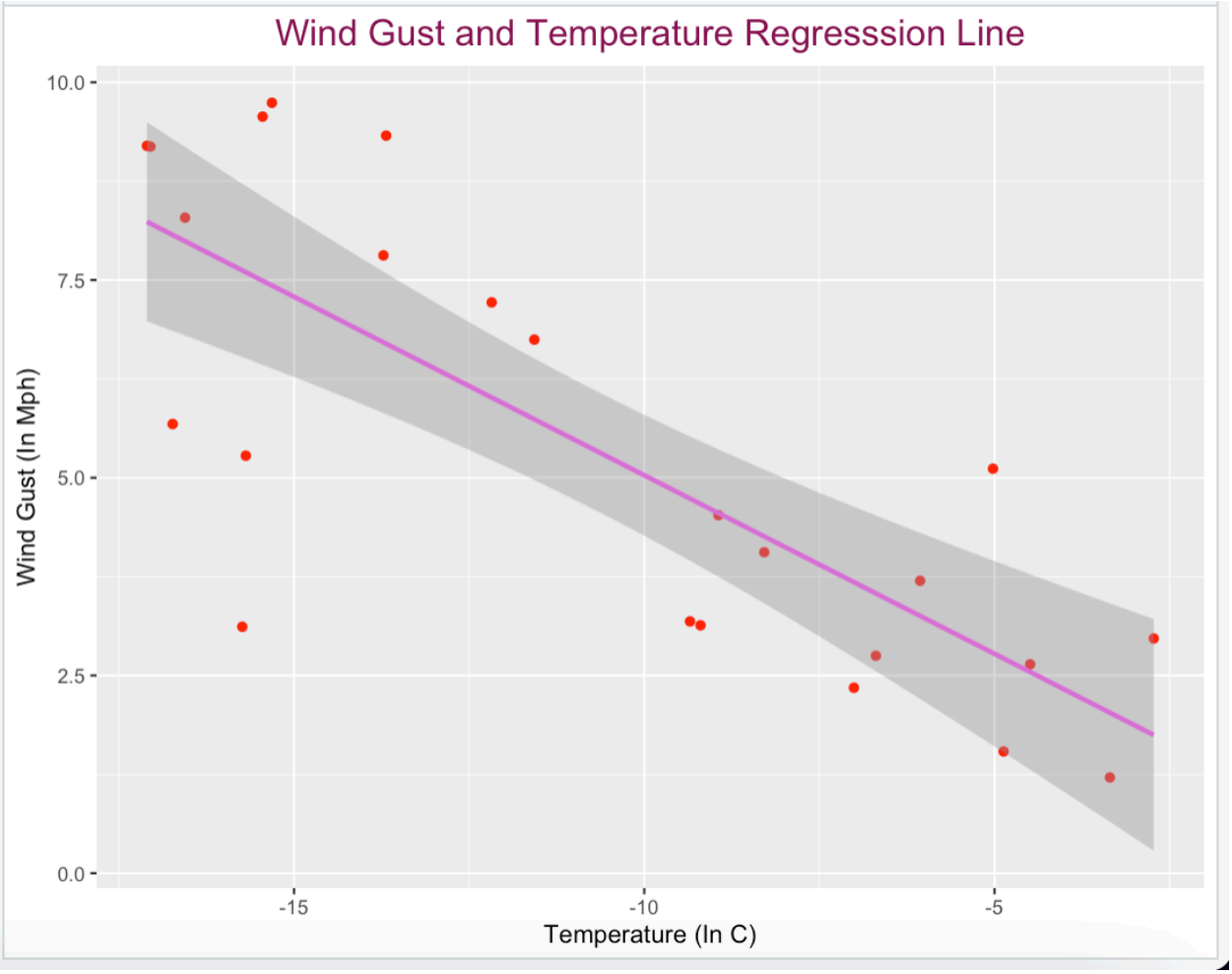

Figure 25 : Regression Equation Line of wind gust and temperature

The graph above shows how temperature and wind gust are related to each other. According to the data, it shows a correlation of -0.50, indicating modest downgrade in linear relation. Analyzing the connection, increase in temperature causes liner decrease in wind gust. So, considering the situation of high wind gust, it is expected to delay the time taken for a flight. Turbulence in planes and vandalization to airport buildings and stations might also be some issues of wind gust (Schrader, 2021). Therefore, when the temperature lowers, there is a possibility of wind gust occurring. Considering this analysis airport staffs and flight management can arrange less flights and prevent any sort of future accidents.

Analysis 05 : Comparing dew point of each months for JFK and LGA using Bar plot

In this analysis, dew points of two airports throughout the year is analyzed. Using bar plot, data of drew point by each month is presented in graphical way.

```
# Analysis Example 05 : Comparing Dewpoint per months for JFK and LGA using Bar

# Manipulation
avg_Temp <- weather_data %>%
  group_by(origin, month) %>%
  summarise(
    mean_Temp = mean(temp, na.rm = TRUE),
    mean_Dew_Point = mean(dewp, na.rm = TRUE)
  )
avg_Temp

# Visualization
ggplot(avg_Temp, aes(x = month, y = mean_Dew_Point, fill = origin)) +
  geom_bar(stat = "identity", position = "dodge", color = "black") +
  scale_fill_manual("origin", values = c("JFK" = "steel blue", "LGA" = "pink")) +
  scale_x_continuous(breaks = 1:12,
                     labels = c("Jan", "Feb", "Mar", "Apr",
                                "May", "Jun", "Jul", "Aug",
                                "Sep", "Oct", "Nov", "Dec")) +
  labs(title = "Dewpoint of each month",
       x = "Month", y = "Dewpoint (In F)") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4")) +
  annotate("text", label = "Average Dew Point = -14.93°C", size = 4, x = 3, y = 3, color = "red")
```

Figure 26 : Source code to display dew points of both airports in 2013

In the beginning, a new object called avg_Temp is declared so that mean temperature and dew points are stored. For this, the weather_data is grouped in terms of origin and month. Then, the data is summarized and two more objects with name ‘mean_Temp’ and ‘mean_Dew_Point’ are created to store corresponding mean values. The ‘na.rm = TRUE’ removes unavailable values (N/A) from the dataset. Although I have already replaced and eliminated all those unavailable values during pre processing, this is just to ensure data. The numeric outcome from this manipulation provides values of mean_Temp and mean_Dew_Point based on origin and months as shown below :

```
> avg_Temp
# A tibble: 24 x 4
# Groups:   origin [2]
   origin month mean_Temp mean_Dew_Point
   <chr>  <int>     <dbl>          <dbl>
 1 JFK        1    -16.7          -20.8
 2 JFK        2    -17.1          -21.0
 3 JFK        3    -15.4          -20.1
 4 JFK        4    -12.2          -16.9
 5 JFK        5     -9.35         -12.6
 6 JFK        6     -6.06          -9.05
 7 JFK        7     -3.35          -6.70
 8 JFK        8     -4.87          -8.66
 9 JFK        9     -7.01         -11.1
10 JFK       10     -9.20         -13.1
# … with 14 more rows
>
```

Figure 27 :Numeric outcome of avg_Temp based on origin and month

Up next, visualization is done where ggplot function along with geom_bar is used to create a bar plot. Furthermore, the scale_fill_manual( ) function is used which allows to customize the bar plot with color fillings. Following that is scale_x_continuous function which, similar to 'Analysis 3' is used to replace the default numbers of x-axis with name of months. Then, labs( ) and theme( ) functions are used as customization of axes and titles. Additionally, annotate( ) function is also used in this analysis which displays a clear annotation in the graph. Attributes such as position, label, size and colors of annotation are provided as the parameter.

The resulted diagram displayed below shows how dew points have varied months by months in both JFK and LGA airports. From the graph, it is evident that dew points fall down miserable in months of November to March. Then, it gradually gets high up to the month of July. Comparing the computed data, JFK had the higher dew point rate to that of LGA with -6.6960 °C and -7.4115 °C respectively. Similarly, the lowest dew point for JFK was in February with -20.996 °C and it was -21.125 in the same month (February).

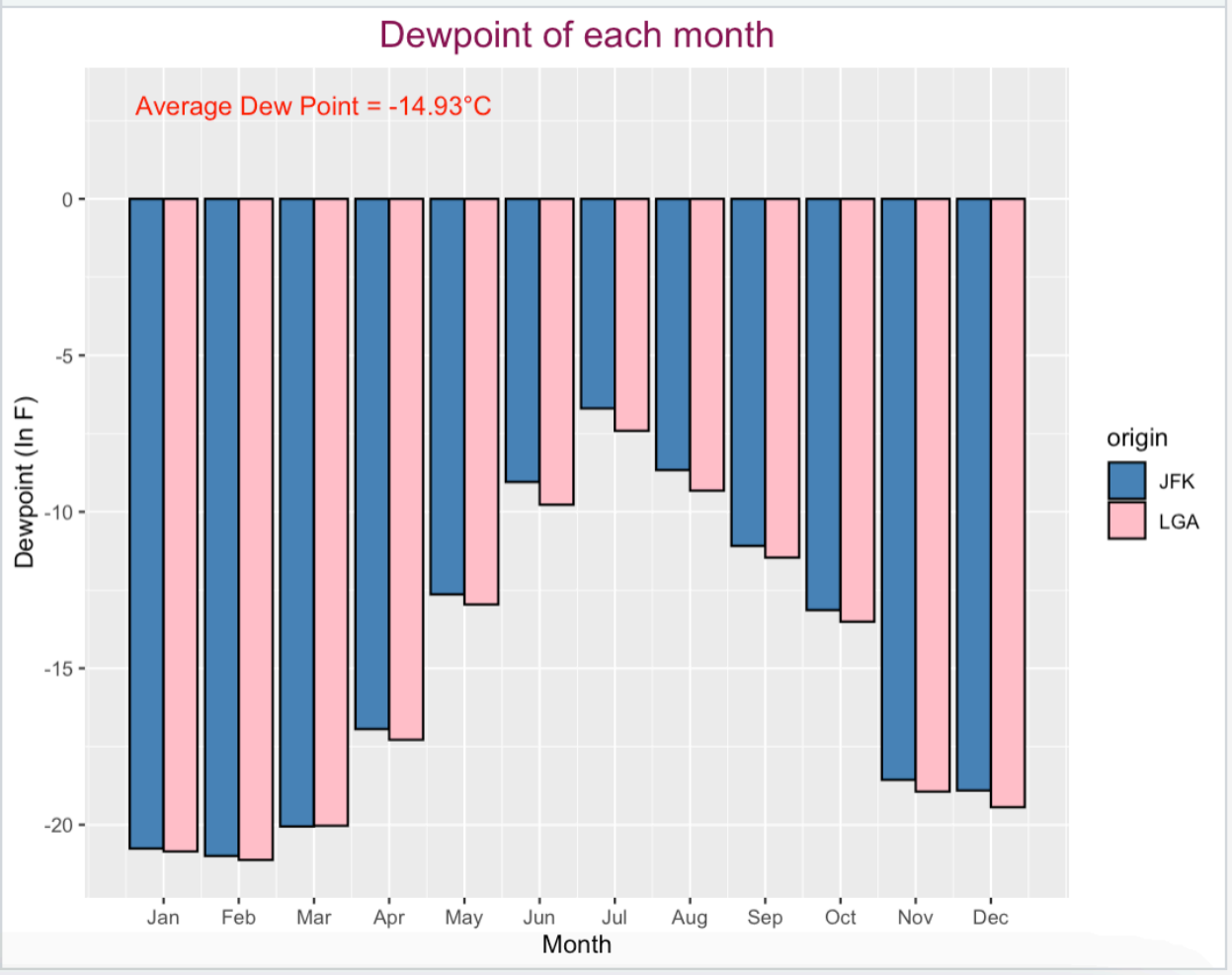

Figure 28 : Bar plot graph of dew points in both airports (JFK and LGA - 2013)

Dew point can be understood as a point where water vapor present in sky condense and turns into clouds (Sky brary, 2021). High dew point means there is presence of more clouds in sky. This will affect flight as airplane or Boeing pilots need to maneuver around the airport so that they can land safely. This analysis concludes that the chances of having clear sky and easy landing are higher in May-August. However, in between November to March, the sky is likely to be foggy which means difficulty in landing comparatively.

Analysis 06 : Analyzing Humidity distribution in Weather Data using Histogram

In this analysis, study of humidity distribution is done, throughout the year 2013. The analysis tends to visualize how humidity changes month by month in both JFK and LGA airports.

```
# Analysis Example xx : Analyzing Humidity distribution in Weather Data using histogram

# Manipulation
weather_data %>%
  select(humid) %>%
  summarise(humid)

# Visualization
ggplot(data = weather_data, mapping = aes(x = humid, fill = origin)) +
  geom_histogram() +
  labs(title = "Distribution of Humidity by Months", x = "Temperature") +
  theme(plot.title = element_text(hjust = 0.5, size = 26, color = "deeppink4")) +
  theme(panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5, size = 16)) +
  facet_wrap(~month)
```

Figure 29 : Source code to display humidity distribution throughout the year 2013

Firstly, in order to get the concise data, the weather data is manipulated and humid variable is summarized.

Then, moving into visualization, aesthetic mapping is used for axes parameters. The geom_histogram( ) function used in creating the histogram graph. Similar to other analysis, labs( ) function is used to provide title and label axes whereas the theme( ) function is used for maintaining alignment and size. In the final line of code, facet_wrap( ) is used which helps to create multi panel plots in 2d. The parameter month means the visualization outcome is to be wrapped on the basis of months.

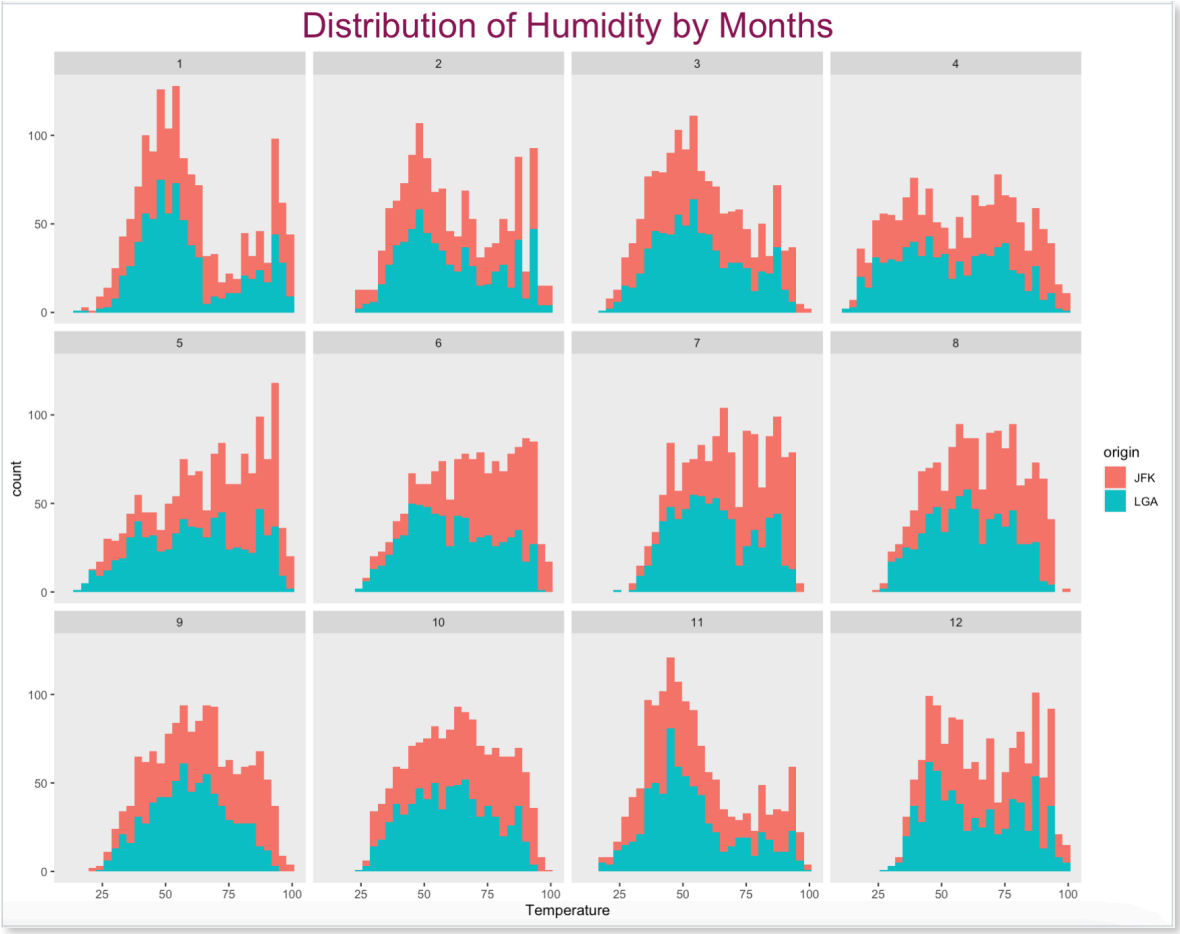

Figure 30 : Histogram of humidity throughout the year (JFK and LGA - 2013)

Humidity is inversely proportional to air pressure. Meaning, when humidity rises up, the air pressure goes down accordingly. This results in decrease of molecules in the air which eventually affects the lifting of airplanes (B, 2011). In this way, high humidity affects flights. The resulted diagram above shows how humidity temperature has fluctuated in two airports throughout 2013. From figure, humidity data of JFK is denoted by reddish color whereas the bluish section displays data of LGA airport. The histogram clearly shows that humidity temperature in JFK have had higher average value in each month. From this analysis, one can make a clear decision of when to fly airplane with full safety. During the season of high humidity, it is likely there is high density altitude effect in the sky. This is considered unsafe compared to the time when there is less humidity.

Analysis 07 : Analyzing rainy days throughout the year in both airports using scatterplot

In this analysis, number of rainy days throughout the year is analyzed. The value of humidity is considered as a specific point in denoting rainfall.

```
# Analysis Example 07 : Analyzing number of rainy days in both airports throughout the year

# Analyzing Rainy Days
jfk_rain=weather_data %>%
  filter(humid==100, origin=="JFK") %>%
  group_by(month) %>% #JFK

  summarise(rain=n_distinct(day),origin,.groups='drop') #multiple rain in a day will be counted as once

lga_rain=weather_data %>%
  filter(humid==100, origin=="LGA") %>%
  group_by(month) %>% #LGA

  summarise(rain=n_distinct(day),origin,.groups='drop') #multiple rain in a day will be counted as once

#-visualization
ggplot() +
  geom_point(data=jfk_rain, aes(x = month, y = rain, color=origin),size=3) + #JFK plot
  geom_point(data=lga_rain, aes(x = month, y = rain, color=origin),size=3) + #LGA plot
  annotate("rect", xmin=3.6, xmax=6.6, ymin=1 , ymax=6, alpha=0.1, fill="green")+ #Summer time
  scale_x_continuous(breaks=1:12,labels=c("jan","feb","mar","apr","may","jun",
                                          "jul","aug","sep","oct","nov","dec"))+
  scale_y_continuous(breaks=1:7) +
  labs(title = "Number of Rainy Days in a Year", x="Month", y="Rainy Days") +
  theme(plot.title = element_text(hjust = 0.5, size = 26, color = "deeppink4"))
```

Figure 31 : Source code to display rainy days throughout the year 2013

For data manipulation, the dataset (weather_data) is analyzed using different sort of functions. Firstly, objects with name ‘jfk_rain’ and ‘lga_rain’ are created to store the manipulated values. They are then filtered, considering humidity value of 100 to be the point when it starts raining. Then, it is grouped using group_by( ) function on the basis of month. In the end, both of them are summarized so that multiple rains in a specific day can be considered as a single rain (i.e, one).

Then, ggplot( ) function is used followed by geom_point( ) function which is used in creating a scatter plot diagram. The manipulated data is provided as a parameter and the

aesthetic mapping is provided for axes, color and size. Then, annotate( ) function is used followed by scale_x_continuous( ) as well as scale_y_continuous( ) functions. The x-axis is scaled with months name while the y-axis denotes days. In the end, labs( ) and theme( ) functions are used for labeling of axes and customized title.

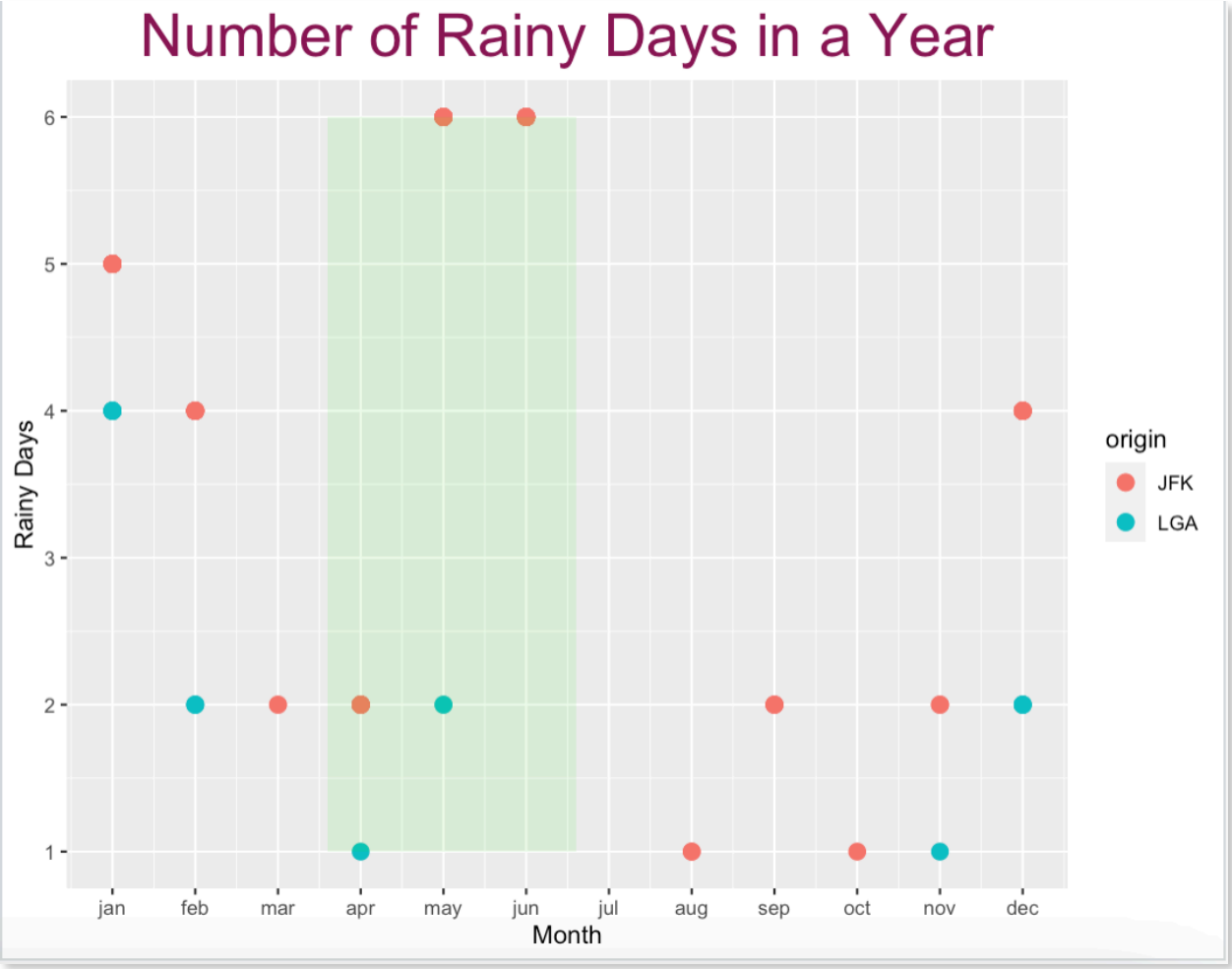

Figure 32 : Scatterplot displaying number of rainy days (JFK and LGA - 2013)

From the figure above, orange dots denotes data of JFK airport whereas blue dots notifies data of LGA airport. The graph demonstrates the number of rainy days for both airports in each months. Analyzing the graph, it is evident that JFK airport experience maximum rainfall in May and June (6 days each). On the other hand, LGA airport experience most rain days in the month of January (4 days). Likewise, minimum rainfall

at JFK is experienced in April and November whereas for LGA airport, minimum rainfall is seen in August and October. Although rain does not affect flight performance, it does affect wind and clouds which plays role in airplane performance. With this data, airports can schedule flights accordingly and suggest best possible flight timing.

Analysis 08 : Line Chart of Average Temperature by month in JFK and LGA

In this analysis, average temperature of both airports in year 2013 is analyzed. With this analysis one can observe how the temperature fluctuated throughout the year 2013 in JFK and LGA airports.

```
#Analysis Example 8 : Line Chart of Average Temperature by month in JFK and LGA

# Manipulation and Visualization
weather_data %>%
  select(origin, month, temp) %>%
  group_by(origin, month) %>%
  summarise(avg_Temp = mean(temp)) %>%

  ggplot(aes(x = month, y = avg_Temp, color = origin)) +
  geom_point() +
  geom_line() +
  labs(title = 'Average Temperature by Month in JFK and LGA',
       x = 'Month', y = 'Temperature (c)', color = 'Origin') +
  theme(plot.title = element_text(hjust = 0.5, color = "deeppink")) +
  scale_x_discrete(limits = month.abb) +
  facet_wrap(~ origin, ncol = 1)
```

Figure 33 : Source code to display average temperature throughout the year 2013

For data manipulation, three variables (i.e, origin, month and temperature) is selected using select( ) function. Then, origin and month are further categorized by months. And, while summarizing the data, an object named ‘avg_Temp’ is created to store mean value of temperature.

Then, the manipulation is continued to visualization using pipe operator (%>%). In there, the ggplot( ) function is used, followed by geom_point( ) and geom_line( ) to create scatter points and line graph. Then, labs( ) and theme( ) function are used to label axes and customize titles. Moreover, scale_x_discrete( ) function is used along with facet_wrap( ), which creates multi panel plots and present them in form of 2d graph.

Besides, a numeric data is also created for this analysis where two new objects with name 'avg_Temp_JFK' and 'avg_Temp_LGA'. The, both of them are summarized with average temperature and the output is printed using print( ) function.

```
# Analysis Example 8 : Numerical and Graph Information

avg_Temp_JFK = summarise(monthly_JFK, avg_Temp = mean(temp))
avg_Temp_LGA = summarise(monthly_LGA, avg_Temp = mean(temp))
print(summarise(avg_Temp_JFK, highest_Temp = max(avg_Temp),
                lowest_Temp = min(avg_Temp), avg_Temp = mean(avg_Temp)))
print(summarise(avg_Temp_LGA, highest_Temp = max(avg_Temp),
                lowest_Temp = min(avg_Temp), avg_Temp = mean(avg_Temp)))
```

Figure 34 : Source code for numerical values of Analysis 8

The outcome of numerical code displays highest, lowest and average temperature values, categorized in months. The outcome is shown below :

```
# A tibble: 12 × 4
   month highest_Temp lowest_Temp avg_Temp
   <int>        <dbl>       <dbl>    <dbl>
 1     1        -31.5       -34.7    -32.8
 2     2        -31.9       -33.8    -32.9
 3     3        -31.4       -33.1    -32.4
 4     4        -29.9       -32.5    -31.2
 5     5        -28.2       -31.4    -30.2
 6     6        -28.2       -30.3    -29.2
 7     7        -27.4       -29.9    -28.5
 8     8        -28.5       -29.6    -29.0
 9     9        -28.1       -30.7    -29.7
10    10        -29.1       -31.7    -30.4
11    11        -29.9       -33.6    -31.9
12    12        -30.4       -33.8    -32.5
>
```

Figure 35 :Numeric outcome of summary temperature based on month

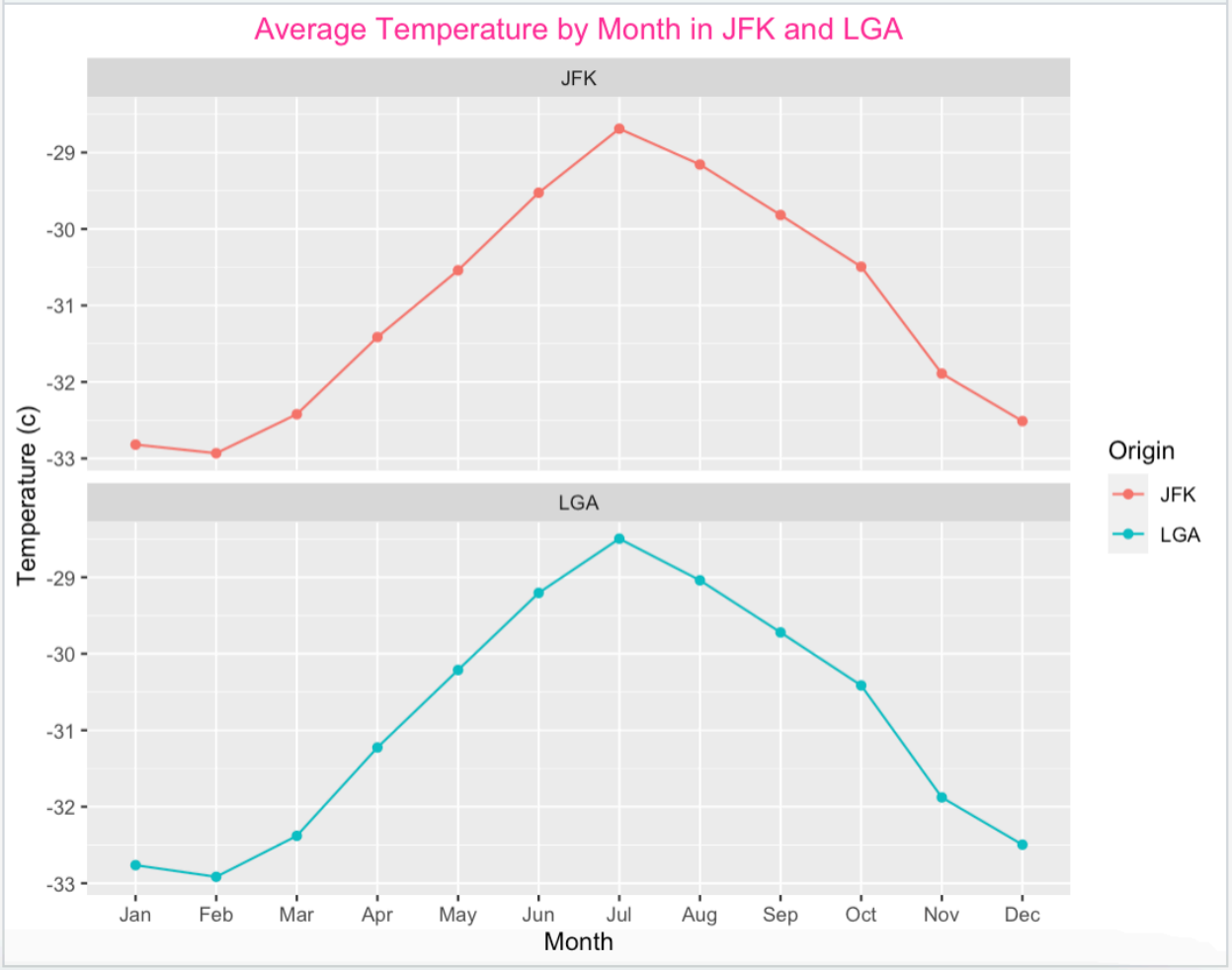

Figure 36 : Line Graph displaying average values by month (JFK and LGA - 2013)

Coming into the visualization output, two line graphs are created with reddish color denoting JFK airport and bluish color notifying LGA airport. As seen in the graph, both airports are following the upward trend in the beginning, which eventually goes down by the end of the year. The highest temperature experienced by JFK and LGA were found in month of July. Similarly, the lowest temperature was recorded in the month of February for both airports. Temperature plays an important role in change of air density. Higher the temperature, lower will be the air density which will reduce flight performances.

Analysis 09: Analyzing weather humidity according to snowfall using bar plot

In this analysis, average humidity of the data on the basis of snowfall is studied. The analysis is expected to demonstrate how humidity varies in snowy, partially snowy and non-snowy days.

```
# Analysis Example 9 : Analyzing weather HUMIDITY according to Snowfall using bar plot

weather_data %>%
  select(snow, humid) %>%
  group_by(snow) %>%
  summarise(avg_Humidity = mean(humid), .groups = 'drop')

ggplot(weather_data, aes(x=humid, y=frequency(humid), fill=factor(snow))) +
  geom_bar(width = 0.20, stat = "identity") +
  theme_bw() +
  facet_wrap(~snow) +
  labs(fill = "Snow", x="Humidity", y="Count",
       title = "Humidity according to Snowfall", subtitle = "2013") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 37 : Source code to display humidity based on snowfall

During data manipulation, the dataset (i.e, weather data) is categorized by selecting snow and humid variables. The, the imported data is grouped by snow using group_by( ) function. Once done, the data is summarized by storing the average value of humidity. The data manipulation produce a numeric outcome which is shown below :

```
# A tibble: 3 × 2
  snow         avg_Humidity
  <chr>               <dbl>
1 no snow              61.6
2 partial snow         64.8
3 snow                 61.3
>
```

Figure 38 :Numeric outcome of average humidity based on snowfall

Then, ggplot( ) function is used for visualization followed by geom_bar( ), for the creation of bar graphs. Necessary parameters are provided to aesthetic mapping and facet_wrap( ) function is used, to categorize bar and display it in 2d plots. Im the end, labs( ) and theme( ) functions are used to label axes, titles and customize their size, font and positioning.

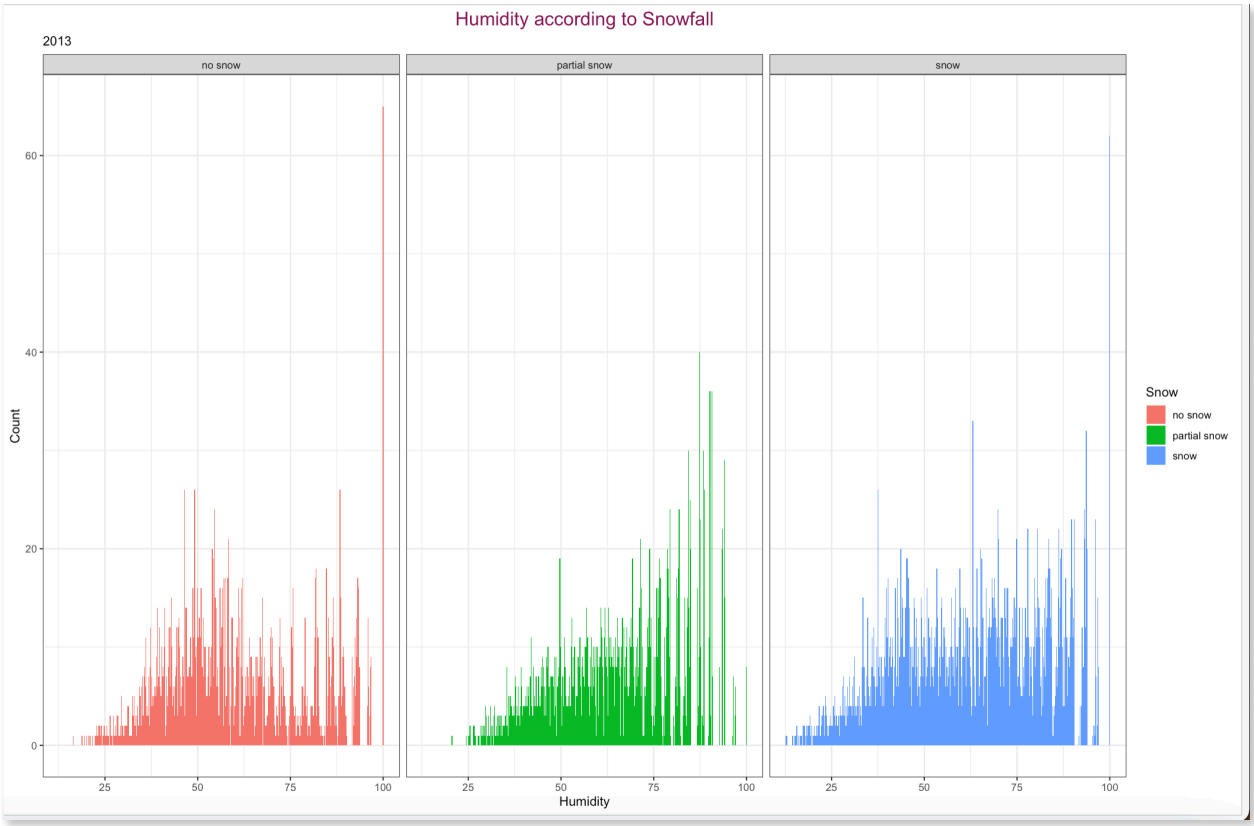

Figure 39 : Bar Graph displaying humidity based on snowfall

The bar chart above depicts how humidity fluctuates depending on snowfall. In the graph above, the red bar shows values of ‘no snow’ conditions whereas green and blue shows value of ‘partial_snow’ and ‘snow’ condition respectively. Considering the number of values, the bar chart looks a bit congested. However, basic information regarding to humidity and snowfall can reality be revived. Analyzing the graph, the average maximum

humidity is experienced in 'partial snow' area I.e, 64.8. Followed by that, 'snow' section had humidity value of 61.6 and 'no snow' had 61.3 humid value.

The influence of humidity in airplane flights is intense. Since humidity and pressure are inversely proportional, increase in humidity reduces the pressure. This eventually causes declination in air molecules which affects flight performance of aircrafts.

*(Note : The **'snow'** variable is mutated just to demonstrate the knowledge of data analysis. Since it is not the real data, the outcome might not match real-life scenario)*

Analysis 10 : Predicting snowfall in both airports using jitter scatterplot

In this analysis, snowfall prediction in both JFK airport and LGA airport id done. To depict the outcome, scatterplot is used along with jitter feature. In this analysis, we have considered negative dew point value as a snowing possibility.

```
# For Snow (Considering below 0 degree of dewp as a snowing possibility)
# Manipulation and Visualization

weather_data %>%
  filter(dewp<0) %>%
  ggplot(aes(x=month, y=dewp, group=month, color=factor(month), na.omit())) +
  geom_jitter() +
  facet_wrap(~origin) +
  scale_x_continuous(breaks = 1:12, labels = c("Jan", "Feb", "Mar", "Apr",
                                               "May", "Jun", "Jul", "Aug",
                                               "Sep", "Oct", "Nov", "Dec")) +
  labs(title = "Scatter plot (Jitter) to predict Snowfall",
       subtitle = "Both JFK and LGA",
       x = "Month", y = "Dew Point (C)") +
  theme(plot.title = element_text(hjust = 0.5, size = 26, color = "deeppink4"))
```

Figure 40 : Source code to predict snowfall in 2013 (JFK and LGA)

The code above shows data manipulation and visualization to find the possibility of snowfall in JFK and LGA airports. Firstly, the weather_data is manipulated by selecting dew point (dewp) and filtering it with negative boundary (i.e, > 0). Then, a pipe operator is used which has given continuity to analysis towards visualization. In there, ggplot( ) function is used with aesthetic mappings. What's more is, a function named geom_jitter() is also used for creation of jittering plots in the graph. Then, the outcome is wrapped in 2d form divided by origin. Similarly, scale_x_continuous( ) function is used to replace all the default numbers in x axis with the name of months. This is followed by labs( ) and theme( ) function which is used for customizing the size, font, color and other attributes of axes and title.

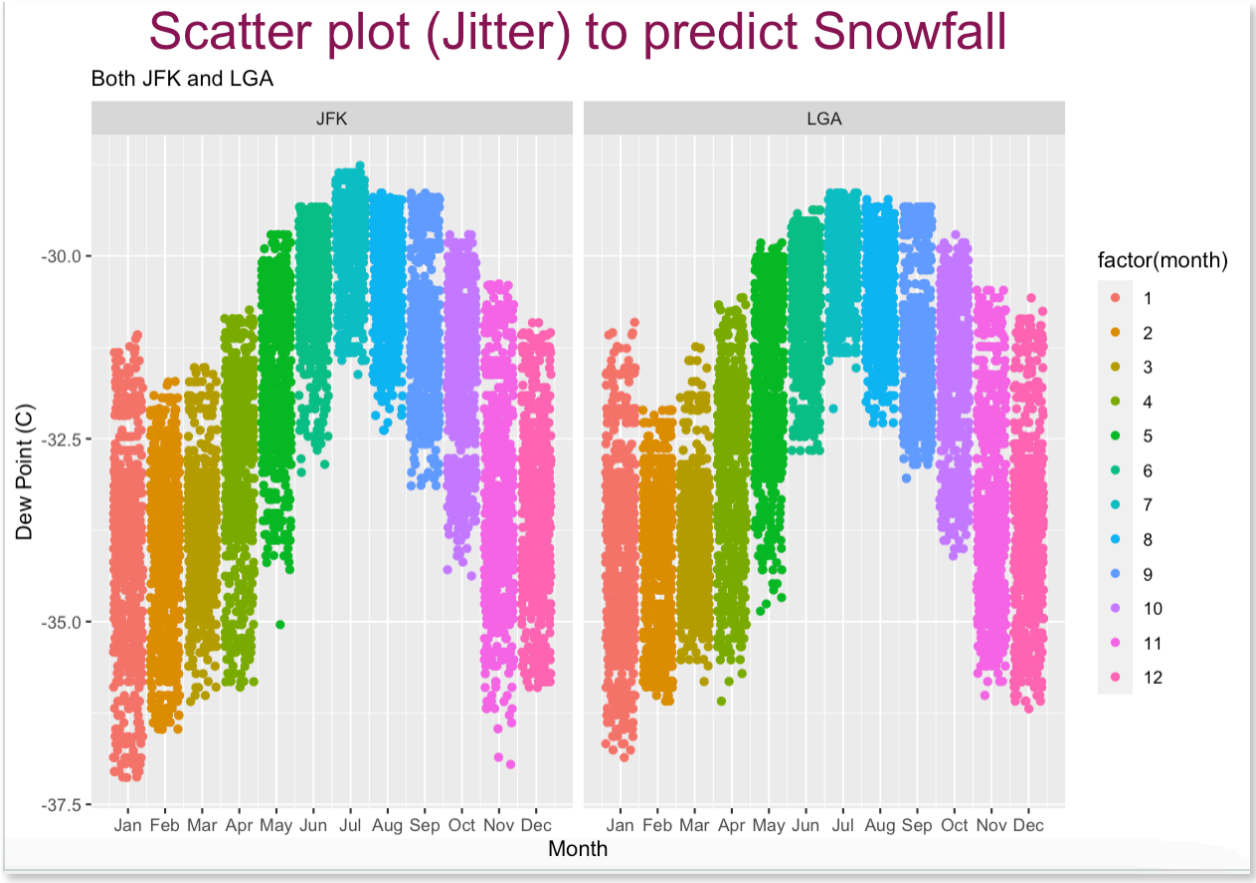

Figure 41 : Scatterplot predicting snowfall in JFK and LGA airports

From the resulted scatterplot shown above, it is clear that graph is following both upward and downward trend continually. In the beginning of year 2013, the chart shows a slight declination trend in both airports. However, the dew point increases after the first month and keeps increasing till the month of July. Then, it again follows downhill trend until the end of the year. Observing the dew point value and scatters in each months, we can simply predict snowfall throughout the year. The value of dew point is inversely proportional to the possibility of snowfall. Therefore, months from November to January or nearby months have higher possibility of snowfall, compared to June and August.

In context of airports, snowfall might not stop airplane flights entirely but it does affect certain areas of flights compared to normal weather. During snowy days, the flight is risky considering the visibility and wetness caused by snow. In such cases, airfare controllers need to work on maintaining air and landing gaps (Met Office, 2018). So, with this analysis, it can help air traffics and management to make sensible decisions based on possibility the snowfall possibility.

Analysis 11 : Line chart to analyze airport visibility in JFK and LGA in 2013

```
# Analysis Example 11 : Line chart to analyze airport visibility in JFK and LGA

# Manipulation and Visualization
weather_data %>%
  select(origin, month, visib) %>%
  group_by(origin, month) %>%
  summarise(avg_Visib = mean(visib)) %>%

  ggplot(aes(x = month, y = avg_Visib, color = origin)) +
  geom_point() +
  geom_line() +
  labs(title = 'Airport Visibility in JFK and LGA',
       x = 'Month', y = 'Visibility (in Mile)', color = 'Origin') +
  theme(plot.title = element_text(size = 14, face = 'bold'),
        legend.position = 'top') +
  scale_x_discrete(limits = month.abb)
```

Figure 42 : Source code to analyze airport visibility in 2013 (JFK and LGA)

In this analysis, visibility of two different airports (JFK and LGA) is studied. The outcome is produced in form of line graph for visual clarifications.

At first, the weather dataset (weather_data) is manipulated using functions such as select( ), group_by( ) and summarise( ). Three different variables with name ‘origin’, ‘month’ and ‘visib’ are selected which is grouped on the basis of origin and month. Then the data is summarized by creating a new object ‘avg_Visib’ to store average visibility. The analysis is continued using the pipe operator (%>%) to visualization. The ggplot( ) function is used with aesthetic mappings. Then, geom_point( ) function is used for both airports to create scatter point for precise values. Besides, labs( ) and theme( ) functions are used for customizing the size, font, color and other attributes of axes and title. In the end, scale_x_continuous( ) is used to replace default numbers with month name in x-axis.

Moreover, a numerical analysis has also been done for this analysis. Two objects named ‘avg_Visib_JFK’ and ‘avg_Visib_LGA’ are created to store summarized values of average visibility. The source code and result of numerical analysis are shown below :

```
# Analysis Example 5 : Numerical and Graph Information

avg_Visib_JFK = summarise(monthly_JFK, avg_Visib = mean(visib))
avg_Visib_LGA = summarise(monthly_LGA, avg_Visib = mean(visib))
print(summarise(avg_Visib_JFK, highest_Visib = max(avg_Visib),
                lowest_Visib = min(avg_Visib)))
print(summarise(avg_Visib_LGA, highest_Visib = max(avg_Visib),
                lowest_Visib = min(avg_Visib),
                avg_Visib = mean(avg_Visib)))
```

Figure 43 : Source code for numeric analysis of airport visibility

```
# A tibble: 12 × 4
   month highest_Visib lowest_Visib avg_Visib
   <int>         <dbl>        <dbl>     <dbl>
 1     1            10         2.76      8.71
 2     2            10         4.41      8.83
 3     3            10         5.02      9.33
 4     4            10         6.14      9.67
 5     5            10         4.05      8.98
 6     6            10         3.67      9.43
 7     7            10         6.79      9.68
 8     8            10         8.09      9.79
 9     9            10         6.67      9.68
10    10            10         4.67      9.58
11    11            10         3.46      9.47
12    12            10         4.17      8.58
> 
```

Figure 44 : Result of numeric analysis to compute airport visibility

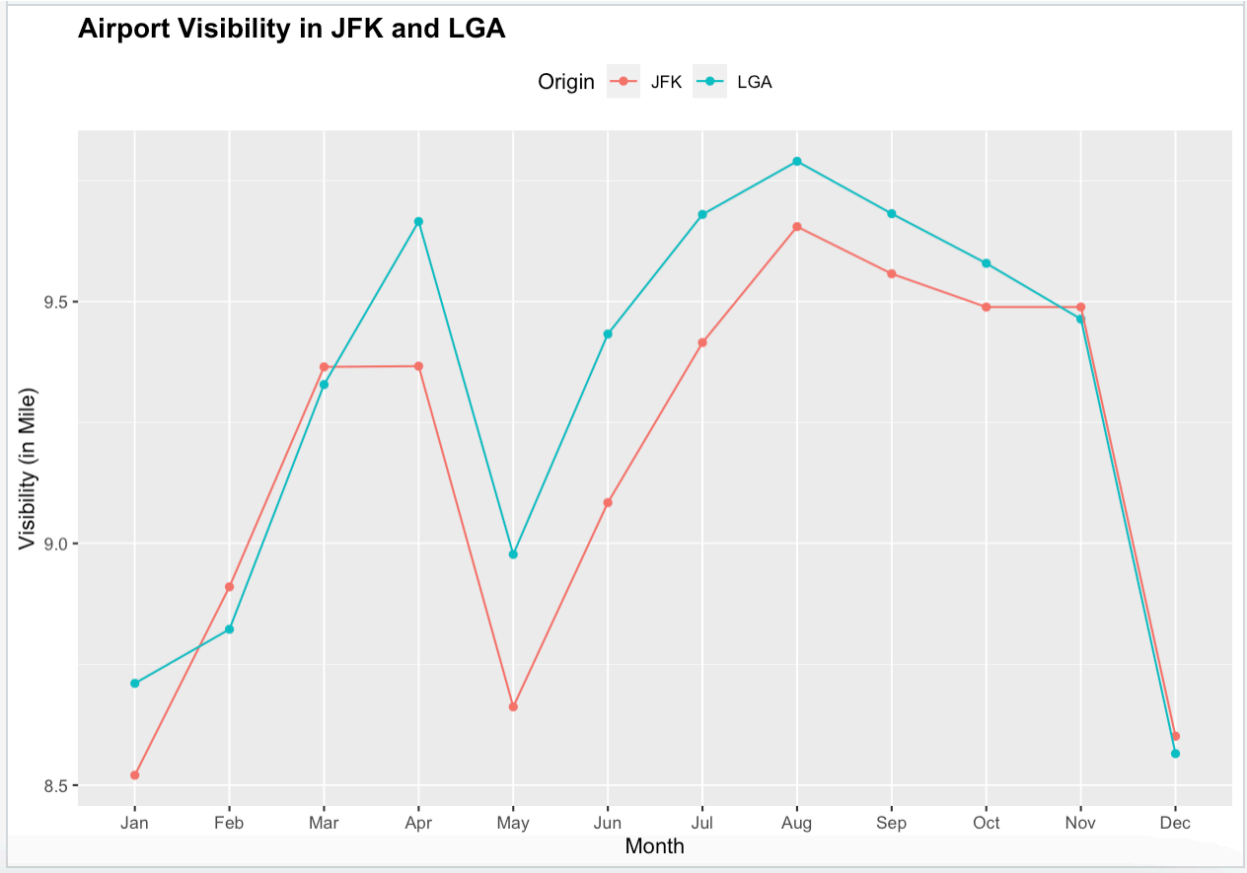

Figure 45 : Line Graph displaying airport visibility throughout the air (2013)

The visual graph of resulted outcome is shown below. The graph shows how changes in visibility occurred throughout 2013. The red line and point shows data of JFK whereas the blue line and point shows data from LGA airport. Observing the line graph, January had the lowest visibility in JFK airport whereas LGA airport the least visibility in December. The visibility then increased and reached around 9.3 miles in April and got back to 8.7 miles in May for JFK. The highest visibility in JFK was around 9.65 miles and 9.79 miles in LGA airport. At the end of the year, both airports had comparatively lower visibility with around 8.5 miles.

According to Met Office (2018), visibility of 10 miles is considered to be ideal. Usually, visibility of 5 miles is considered to be fair whereas the visibility of less than 3 miles is considered poor. Lower visibility is usually caused by fogs, heavy rains and high humidity as well. Using this analysis airport traffic and flight management teams can come up with appropriate planning and decision making.

Analysis 12: Polar Bar Chart of Wind Direction Distribution in JFK and LGA

In this analysis, the wind direction distribution in two different airports (JFK and LGA) is analyzed. The concept of 'polar bar chart' is used to demonstrate the analysis in graphical form. The manipulation technique is similar to most of other analysis. The imported dataset (weather_data) is used for the data. The select( ) function is used where origin and wind_dir variables are selected. Then, mutation is done where a new object named dir_EWNS is added. Using the if_else( ) function, directions are categorized into four sides ('NE', 'SE', 'SW' and 'NW') according to the degree of direction. Moreover, the object is further grouped using group_by( ) function and the sum is summarized. The

manipulation of data provides a numeric chart as outcome. The result is shown in the figure below :

```
# Analysis Example 12 : Polar Bar Chart of Wind Direction Distribution in JFK and LGA

# Manipulation
weather_data %>%
  select(origin, wind_dir) %>%
  mutate(dir_EWNS =
           ifelse(wind_dir > 0 & wind_dir <= 90, 'NE',
                  ifelse(wind_dir >90 & wind_dir <= 180, 'SE',
                         ifelse(wind_dir >180 & wind_dir <= 270, 'SW',
                                ifelse(wind_dir >270 & wind_dir <= 360, 'NW', 0 ))))) %>%
  group_by(origin, dir_EWNS) %>%
  summarise(count = sum(dir_EWNS == 'NE', dir_EWNS == 'SE',
                        dir_EWNS == 'SW', dir_EWNS == 'NW'))

# Visualization
ggplot(weather_data) +
  geom_bar(aes(x = wind_dir)) +
  coord_polar() +
  scale_x_continuous(limits = c(0, 360), breaks = seq(0, 360, 45)) +
  labs(title = 'Distribution of Wind Direction (JFK and LGA)',
       x = 'Degree of Wind Direction') +
  theme(plot.tag = element_text(size = 14, face = bold)) +
  facet_wrap(~ origin, nrow = 1) +
  theme_classic()
```

Figure 46 : Source code to analyze wind direction in 2013 (JFK and LGA)

```
# Groups:   origin [2]
   origin dir_EWNS count
   <chr>  <chr>    <int>
 1 JFK    0            0
 2 JFK    NE        1314
 3 JFK    NW        2608
 4 JFK    SE        1590
 5 JFK    SW        2881
 6 LGA    0            0
 7 LGA    NE        1775
 8 LGA    NW        2750
 9 LGA    SE        1390
10 LGA    SW        2434
>
```

Figure 47 : Result of numeric analysis to analyze wind direction in 2013

Similarly, the visualization process is also similar to analysis example 11. The key difference in this analysis is the use of coord_polar( ) function which creates bar chart with polar coordinates. Moving on, there is also use of seale_x_continious( ) which, in this case includes limits to 0 degree to 360 degrees. As of other analysis, labs( ) and theme( ) functions are used for customizing the size, font, color and other attributes of axes and title.

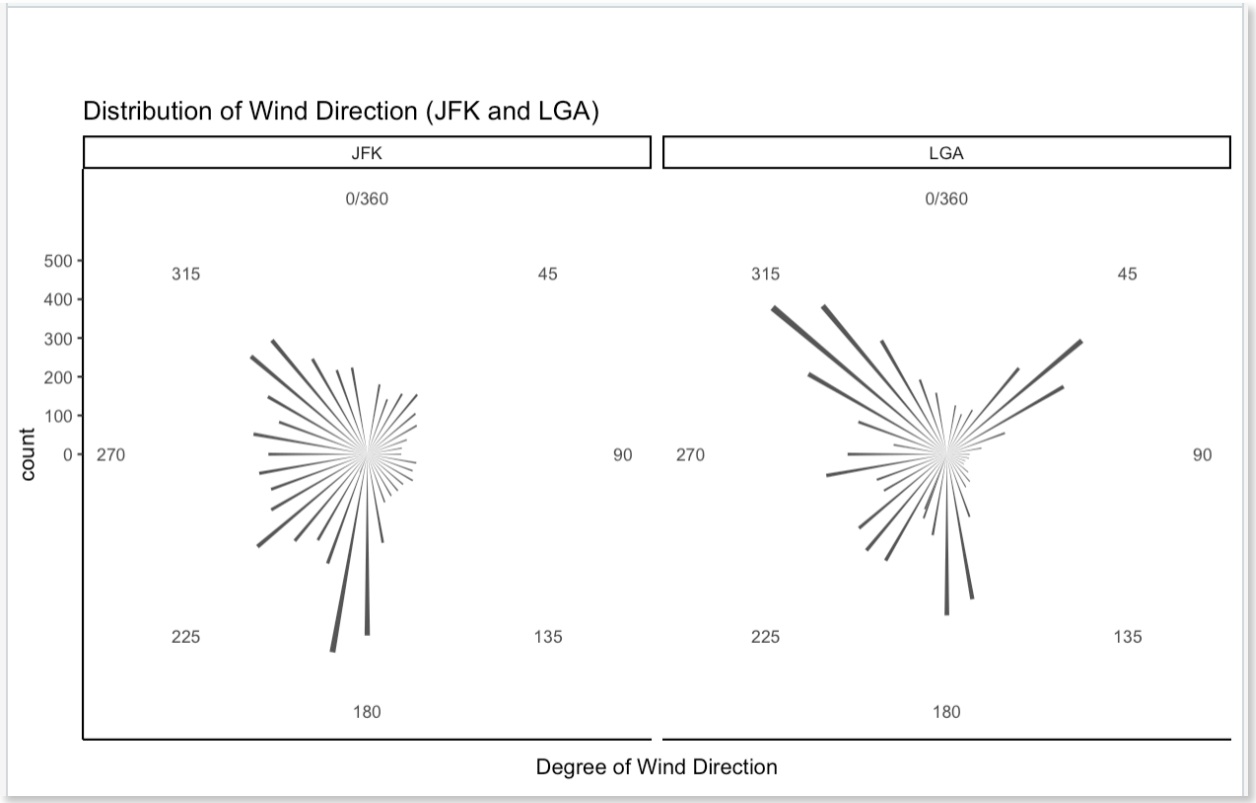

Figure 48 : Polar Bar Graph displaying wind direction throughout the air (2013)

The figure above demonstrates how the wind direction is distributed in two airports using polar bar chart. Analyzing those bars and based on mutation, we can conclude that wind direction from 0° to 90° is NE, 91° to 180° is SE, 181° to 270° is SW and 271° to 360° is NW. And from the outcome, it is seen that high average wind direction in JFK is SW whereas in LGA is NW. With this analysis, air traffic controllers can predict the temperature and make flight schedules accordingly.

Analysis 13 : Analyzing the effect of wind speed and wind direction towards flights (JFK airport)

In this analysis, wind speed, wind direction and their effect on aircraft flights at JFK airport are analyzed. To display the outcome, box plot is being used as a graphical figure.

```
# Analysis Example 13 : Analyzing the effect of wind speed and wind direction
#                         towards flights at JFK airports

# Manipulation

wind_JFK <- weather_data %>%
  filter(origin == "JFK") %>%
  mutate(knot_speed = wind_speed/1.151,
         headwind = knot_speed*cos(wind_dir-45)) %>%
  select(month, knot_speed, wind_dir, headwind) %>%
  filter(headwind >= 0)

wind_JFK %>%
  group_by(month) %>%
  summarise(avg_wind = mean(avg_wind),
            max_wind = max(max_wind),
            .groups = 'drop')

# Visualization
ggplot(wind_JFK, aes(month, headwind)) +
  geom_boxplot(aes(group = month)) +
  scale_x_continuous(breaks = 1:12, labels = c("Jan", "Feb", "Mar", "Apr",
                                               "May", "Jun", "Jul", "Aug",
                                               "Sep", "Oct", "Nov", "Dec")) +
  labs(x="Month", y="Headwind",
       title = "Effect of wind speed and direction towards flight", subtitle = "2013") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 49 : Source code to analyze effect of wind speed and direction (JFK)

Firstly, a new object with name ‘wind_JFK’ is created during data manipulation. Then, the ‘weather_data’ is filtered with JFK origin using pipe operator. Furthermore, mutate( ) function is used to add a new object that stores ‘knot speed’ and ‘headwind’. Once done, those created objects are supplied as a parameter along with ‘month’ and ‘wind_dir’. A condition of headwind being equal or greater than 0 is provided using filter( ) function. Then, the whole object (wind_JFK) is again grouped on the basis of month and summarized, storing values like average and maximum wind values.

Similarly, ggplot( ) function is used in visualization followed by geom_boxplot( ) function which is used to create box plots. Both of these functions are provided with aesthetic mappings and required parameters. Then, scale_x_continuous( ) is used which limits months in x-axis of the graph. And in the end, there is use of labs( ) and theme( ) function for labeling and customizing, like in every previous analysis.

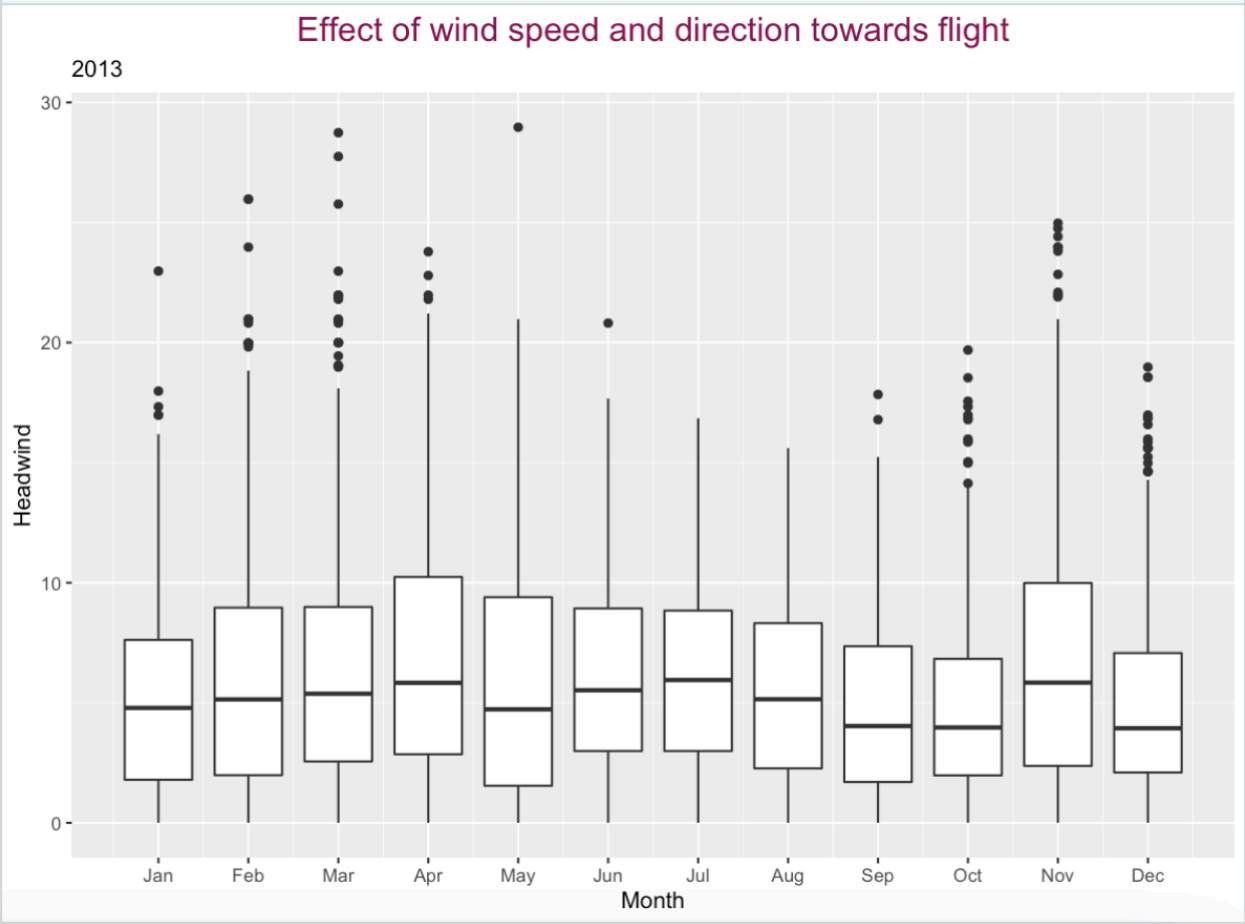

Figure 50 : Box plot graph displaying the effect of wind speed and direction towards flight

The resulted figure shows how wind speed and wind direction affects air flights. Data regarding to wind speed and directions are manipulated with an object which is represented as ‘headwind’ in y-axis. Analyzing the graph, we can see that the maximum headwind is experienced in March and June. The median of wind speed during those

months are 5 m/s and 4 m/s respectively. In graphical representation, a maximum of around 30m/s headwind is experienced. The graph also shows 1st quartile, median, 3rd quartile and maximum values of each months in 2013. During headwinds, the wind blows against the direction of flight. The airplane needs to fly against the direction of headwind, which increases the airflow and slows down the flight.

Analysis 14 : Analyzing humidity details of both airports in year 2013 using box plots

In this analysis, humidity details of both JFK and LGA airports in year 2013 is being resolved. To demonstrate the outcome, two different box plots (whisker diagram) is produced with each of them showing data of one specific airport (JFK and LGA).

```
# Analysis 14 : Analyzing humidity details in JFK and LGA using box plot

# Manipulation and Visualization

JFK <- filter(weather_data, origin == 'JFK')

LGA <- filter(weather_data, origin == 'LGA')

ggplot(rbind(JFK, LGA),
       aes(x = origin, y = humid, fill = origin)) +
  geom_boxplot(color = "steelblue") +
  scale_fill_manual("origin",
                    values = c("JFK" = "coral", "LGA" = "pink")) +
  labs(title = "Humidity Information in JFK and LGA in 2013",
       x = 'Origin', y = 'Humidity Values') +
  theme(plot.title = element_text(hjust = 0.5, size = 16, color = "deeppink4"))
```

Figure 51 : Source code to study humidity details of both airports (JFK and LGA - 2013)

Firstly, two new objects for two different airports are created : JFK and LGA. The data is then filtered during manipulation, based on origin. Then, ggplot( ) function is used for visualization, which is parameterized with rbind( ) function. The function refers to row binding which is used to merge or bind multiple rows of vectors, matrices or data frame. Aesthetic mappings are provided with 'origin' being x-axis and 'humid' being the y-axis. The geom_boxplot( ) function is used to create whicker boxes and steel blue color is provided to the box plot linings and borders. To fill the box with customized colors, I have used scale_fill_manual( ) function. Two different colors (i.e, coral and pink) are in use for two different whisker diagrams. And finally, labs( ) and theme( ) functions are used for labeling the graph and customizing the size, font, color and other attributes of axes and title.

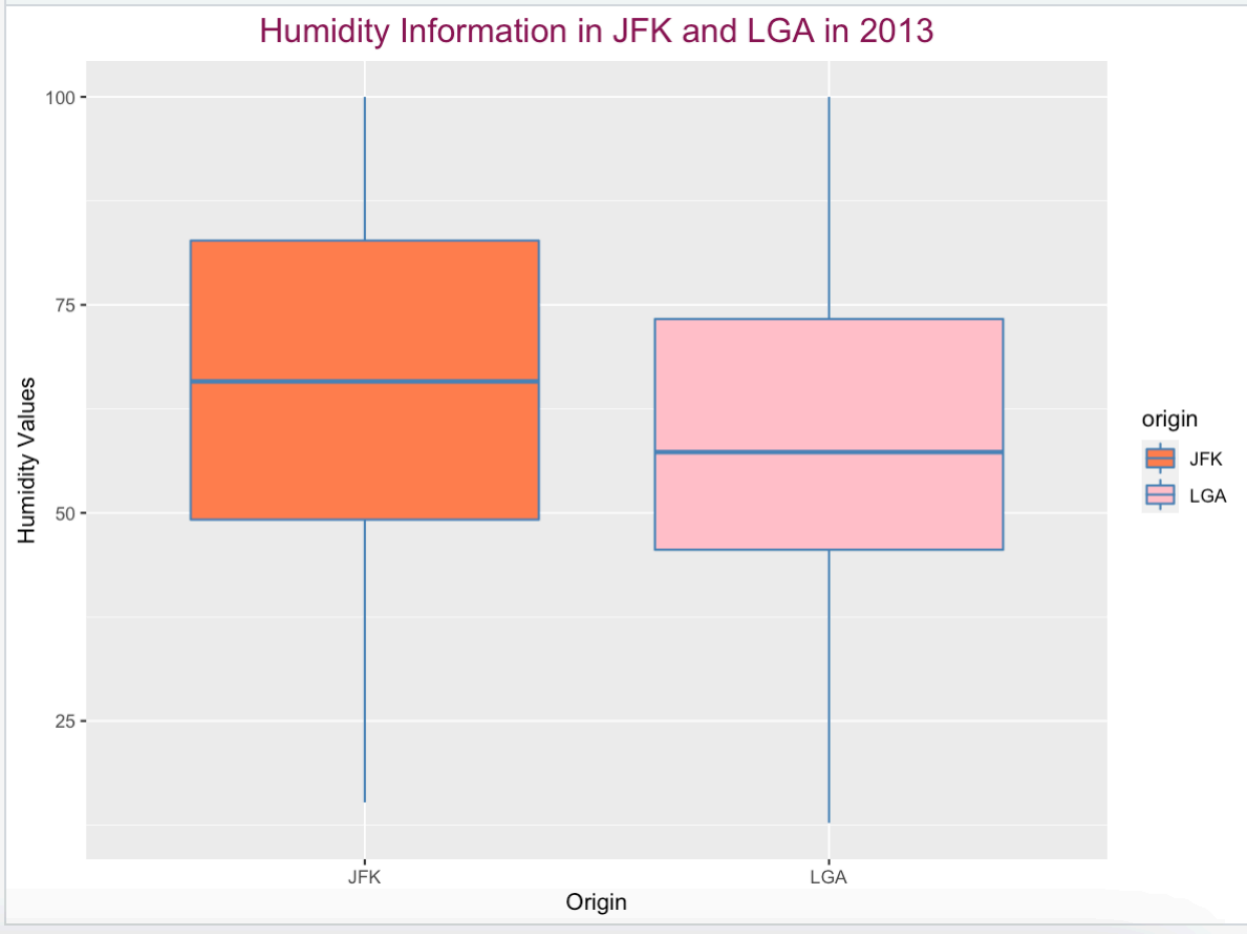

Figure 52 : Whisker diagram displaying humidity information (JFK and LGA - 2013)

The resulted whisker graph demonstrates the humidity information of both JFK and LGA airports in year 2013. From the graph, we can analyze multiple humidity data such as minimum and maximum percentage, quartiles and median of both airports. The figure above shows 100% as the maximum percentage of relative humidity. Out of two airports, the minimum relative humidity was of LGA with 12.74% whereas JFK had minimum humidity of 15.21%. Observing the diagram upward, the point is 1st quartile which is 49.19% and 45.57% for JFK and LGA. Then, the median humidity value of JFK airport is found to be 68.80% whereas for LGA, it is 53.70%. Moving further upward, the 3rd quartile of JFK and LGA are 82.73% and 73.30% respectively.

As discussed in analysis before, humidity does influence flight performance in several ways. Since humidity is inversely proportions to air pressure, increase in humidity decreases the air pressure. The decrease in air pressure means declination of air molecule on wings, which affects flight performance (PhysLink.com, Anton Skorucak, n.d.).

11. Extra Features

Extra Feature 01 : Using hexagonal bins to analyze relationship between humidity and wind speed

A hexbin graph is a 2d density graph that permits visualization between two numeric columns. The prime purpose of using hexbin is to avoid congestion : I.e, to plot density instead of points so that there is no issue of overlapping (ferdio, 2017).

```
# Extra Feature 1 : Using hexagonal bins to demonstrate the relationship between pressure and wind speed

ggplot(avg_Humid, aes(x = pressure, y=wind_speed)) +
  geom_hex(bins = 20) +
  labs(x="Pressure", y="Wind Speed",
       title = "Relationship between pressure and wind speed",
       subtitle = "Using Hexbin") +
  theme(plot.title = element_text(hjust = 0.5, size = 26, color = "deeppink4"))
```

Figure 53 : Source code to analyze relationship between humidity and wind speed

The source code above shows how pressure and wind speed are interrelated to each other. Firstly, ggplot( ) function is used where 'avg_Pressure' is passed as a parameter to import the data. Then aesthetic mapping is provide for pressure and wind_speed for further data filtering. Then the geom_hex( ) function is used for creating hexagonal bins in graphical form. And finally, labs( ) and theme( ) function is used for titles and axes labels.

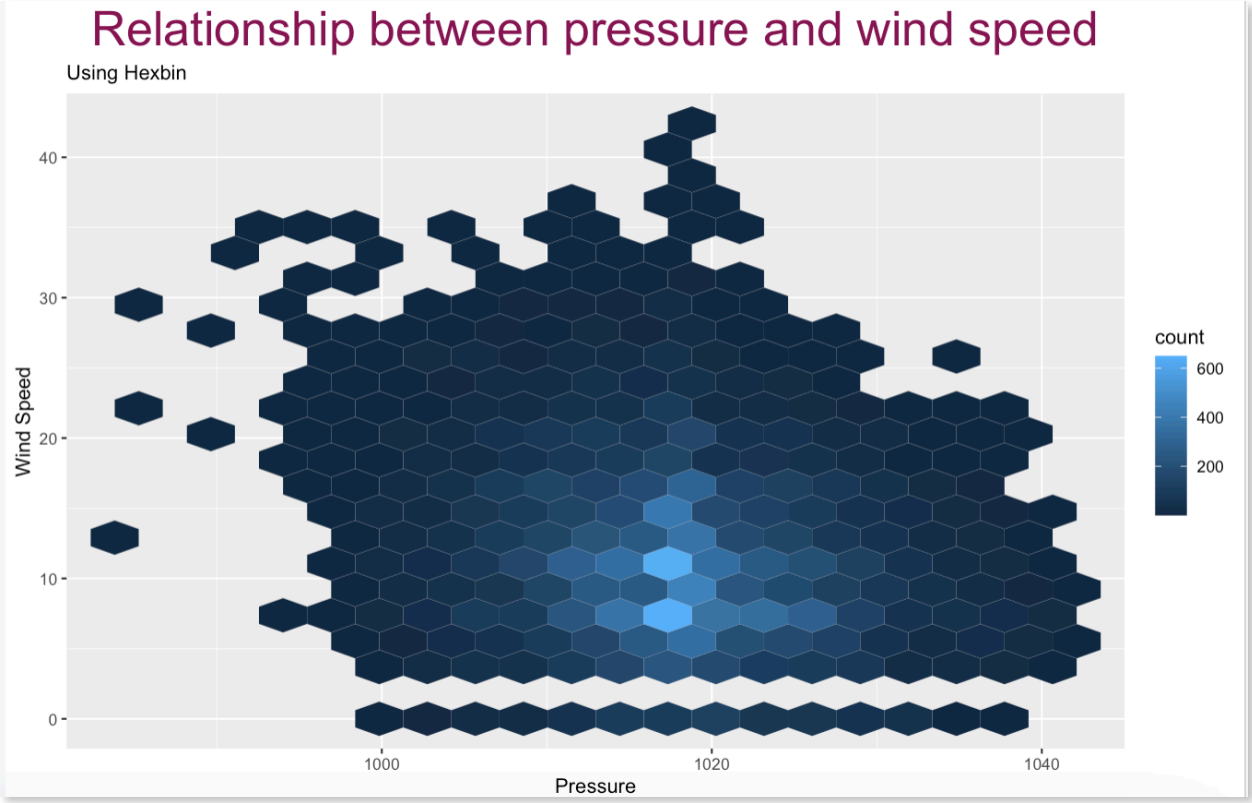

Figure 54 : Hexagonal bins displaying relationship between pressure and wind speed

The figure above displays the result of hexagonal scatterplot. From the graph above, one can study the relationship between wind speed and pressure, with the help of colors. In the figure above, frequency is inversely proportional to darkness of color. Meaning, the dark blue color in hexagon refers that it has low frequency whereas the light blue section has high frequency. Considering this, we can clearly see that the wind speed of highest frequency is at 1018.2 millibars pressure.

Extra Feature 02 : Analyzing average humid density per hour using density graph

The second extra feature is the use of density graph. It is a deviation of histogram that is based on kernel smoothing. A density graph helps to depict numeric variables and their dispensation. Basically, density graph distributes specific set of data into continuous recesses and time period.

```
# Extra Feature 2 : Calculating average humid density per hour using density graph
ggplot(avg_Humid, aes(x=humid)) +
  geom_density(color="coral", fill="beige") +
  labs(title = "Density Graph",
       x="Humidity", y="Density") +
  theme(plot.title = element_text(hjust = 0.5, size = 26, color = "deeppink4"))
```

Figure 55 : Source code to analyze average humid density

In the code above, I have used density graph to compute average humid density per hour. To draw a density chart, geom_density( ) is used. Like other visualizations, ggplot() function is used and 'avg_Humid' is passed in order to import data. Inside the function geom_density( ), two different colors are passes : one for border and other to fill the graph. In the end, labs( ) and theme( ) are used in labeling and positioning title and axes.

The figure below shows how humidity in air relates to density. As we can see, the air density increases with increase in humid. If we look at minimum and maximum point, the minimum density is at the beginning (i.e, at 0°C). Similarly, the maximum density point is seen at around 47 °C. Once the halfway point crossed, the density begins to decline moderately.

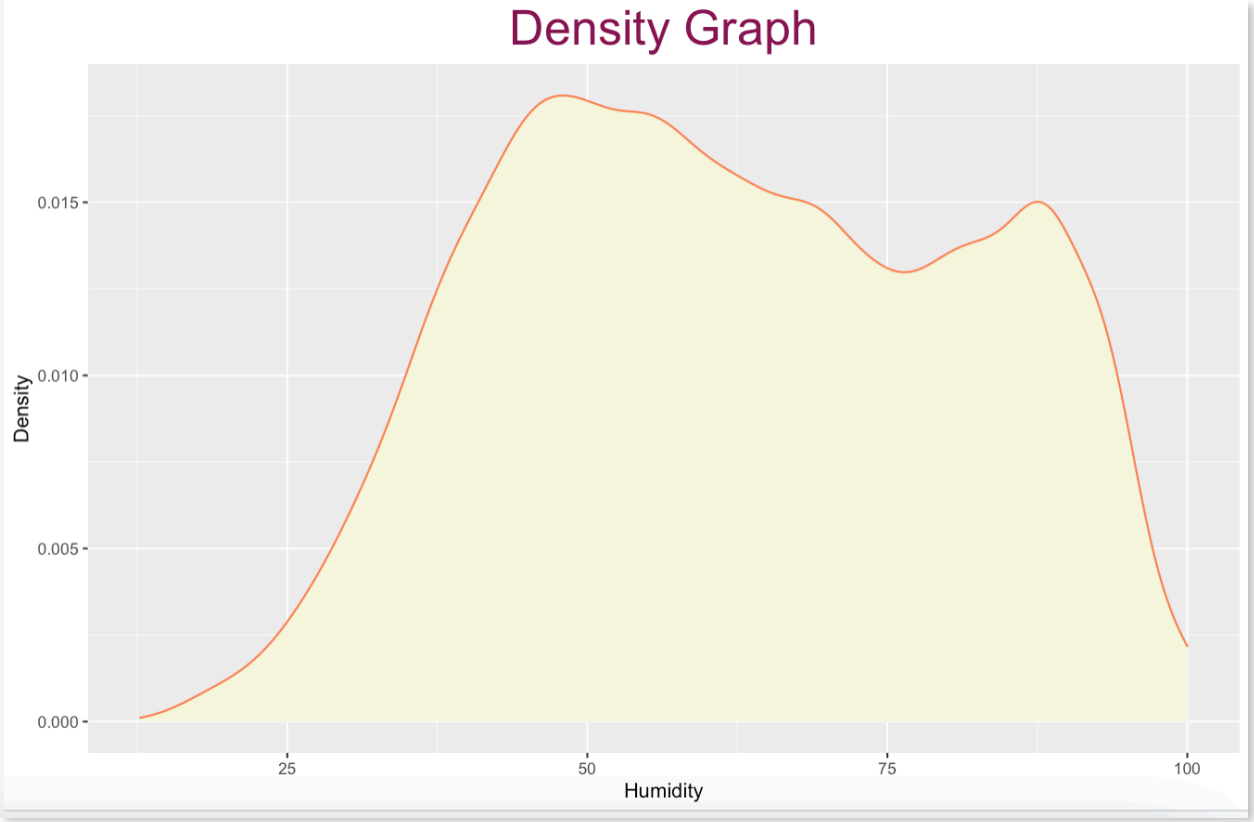

Figure 56 : Density Graph displaying the average humid density

12. Recommendation and Future Enhancements

The 'hourly weather' dataset has most of the climatic information which helped in analyzing weather in multiple aspects. However, there are some loopholes which can be improved in order to upgrade analysis and quality of data. Some of the recommendations regarding weather dataset and analysis are :

- There were variables in the dataset which comprised of unavailable values (N/A). This severely degrades the quality of data as result might be imprecise. So, it is highly recommended that datasets are created with enough data to analyze it.
- Also, it would be meaningful if related data from previous years were also provided. This would help in comparing changes through a specific period of years.

- Having other relevant data such as rainfall information and snowfall rate information would help analysts make deep analysis on airport weather.

- In future, I plan to study how airfare traffic controllers can make appropriate decisions based on the analysis
- Also, I plan to study how humidity and air pressure affects flights based on the size of airplane as well.

13. Conclusion

The hourly weather data analysis project is done to identify, inspect and analyze the weather condition in JFK airport and LGA airport. The analysis project demands serious knowledge of R programming concepts and its implementation to retrieve a meaningful information. The project is done in a free and open source software called RStudio : a R programming IDE. Throughout the analysis project, distinctive concepts related to data manipulation, exploration and visualization were used. I studied a total of 16 analysis with two extra features included in it. The concepts of variables, comments and functions are widely and appropriately used to make project more readable and maintainable. The project of data analysis helped me personally to learn data science techniques which I expect to use in future projects as well.

References

B. (2011, August 23). *Three Hs of Aircraft Performance - Hot, High and Humid*. Desert Jet. https://desertjet.com/2011/08/23/aircraft-performance/

ferdio. (2017, September 22). *Hexagonal Binning*. Data Viz Project. https://datavizproject.com/data-type/hexagonal-binning/

Met Office. (2018, November 19). *Why does the weather affect flights?* https://www.metoffice.gov.uk/weather/warnings-and-advice/seasonal-advice/travel/why-does-the-weather-affect-flights

Petzoldt, T. (2018, October). *Data Analysis with R Selected Topics and Examples*. TECHNISCHE UNIVERSITAT DRESDEN. https://wwwpub.zih.tu-dresden.de/~petzoldt/elements_en.pdf

PhysLink.com, Anton Skorucak. (n.d.). How does humidity effect the way that an airplane flies? Retrieved from https://www.physlink.com/education/askexperts/ae652.cfm

RStudio Logo Usage Guidelines. (n.d.). RStudio. https://www.rstudio.com/about/logos/

Schrader, R. (2021, September 24). *What Wind Speed Delays Flights?* Skyscanner US. https://www.skyscanner.com/tips-and-inspiration/what-windspeed-delays-flights#:

%7E:text=In%20fact%2C%20winds%20high-up,during%20certain%20times%20of%20year.

Sky brary. (2021, June 22). *Dew Point*. SKYbrary Aviation Safety. https://skybrary.aero/articles/dew-point

Team, T. (2021, July 6). *R Applications – 9 Real-world Use Cases of R programming*. TechVidvan. https://techvidvan.com/tutorials/r-applications/

Name : Sandesh Subedi ‘A’

TP Number : NPI000040

Subject : Programming for Data Analysis (Individual Assignment)
