A COMPREHENDISVE ANALYSIS OF METEOROLOGICAL IN EHANCING WEATHER FORECAST AND ACCURACY FOR SHEFFIELD

An Assignment Submitted in the partial fulfilment for

the award of a Master's degree in Data Analytics and Technologies

DAT7006 DATA SCIENCE

WORKSHOP SESSION: Monday 14:00 -17:00

STUDENT NAME: Oluwatosin Ojo

STUDENT ID: 2312602

MODUL TUTOR: Dr. Anchal Garg

Abstract

This report delves into the importance of weather forecasting in Sheffield focusing on the analysis of meteorological variables such as surface temperature, humidity, windspeed, etc. It further identifies the various stakeholders that might benefit from accurate forecasting, how this would impact their business and decision-making process, and ultimately how it can improve the lives of those who work and live in Sheffield.

This report adopts the CRISP-DM methodology to understand the dataset, explore the relationship between the variables in the dataset, use historical data to make time series and machine learning predictions and evaluate the results. The results of the statistical analysis are then used to answer the suggested hypothesis and research questions.

The approach used in this research study is regarded as a novel approach as the focus is primarily on Sheffield location.

Keywords:

- Weather forecast
- Machine Learning
- Time series forecasting
- Deep Learning
- Artificial Intelligence
- Analysis
- Meteorological

Table of Contents

List of Figures

Abbreviations

ML – Machine Learning

AI – Artificial Intelligence

DL – Deep Learning

DSS – Decision Support System

EDA – Exploratory Data Analysis

GDPR – General Data Protection Regulation

1. INTRODUCTION

Pacman library package loads pacman which contains several other packages in it.

p_load(pacman, dplyr, GGally, ggplot2, ggthemes, ggvis, httr, lubridate, plotly, rio, rmarkdown, shiny, stringr, tidyr)

```
library(pacman)                          # No message
library(dplyr)

##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1.1 Background

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	X	X.1	X01.05.20	X.2	X.3	X.4	X.5	X.6	X.7	X.8	X.9	X.10	X01.05.20	X.11	X.12
2	XLAT	XLONG	TSK	PSFC	"U10"	"V10"	"Q2"	RAINC	RAINNC	SNOW	TSLB	SMOIS	TSK	PSFC	"U10"
3	48.871	-11.221	NA	101418	6.9	5.5	0.00602	0	0	0	273.2	1	285.2	101070	7.4
4	49.01	-11.24	285.2	101388	7	5.8	0.00603	0	0	0	273.2	1	285.2	101033	7.4
5	49.149	-11.259	285.2	101357	7	6	0.00604	0	0	0	273.2	1	285.2	100997	7.4
6	49.288	-11.278	285.2	101327	7	6.3	0.00605	0	0	0	273.2	1	285.2	100960	7.4
7	49.427	-11.298	285.2	101296	7	6.6	0.00605	0	0	0	273.2	1	285.2	100923	7.3
8	49.566	-11.317	285.2	101265	7	6.9	0.00606	0	0	0	273.2	1	285.2	100887	7.3
9	49.705	NA	285.1	NA	7	7.1	0.00607	0	0	0	273.2	1	285.1	100852	7.2
10	49.843	NA	285.1	101203	6.9	7.4	0.00608	0	0	0	273.2	1	285.1	100816	7
11	49.982	-11.377	285	101172	6.8	7.7	0.00609	0	0	0	273.2	1	285	100781	NA
12	50.121	-11.397	285	101141	6.8	8	0.0061	0	0	0	273.2	1	NA	100746	6.9
13	50.259	-11.417	284.9	101110	6.6	8.3	0.0061	0	0	0	273.2	1	284.9	100712	6.8
14	50.398	-11.437	284.9	101078	6.5	8.5	0.00611	0	0	0	273.2	1	284.9	100678	6.7
15	50.537	-11.457	284.9	101047	6.4	8.8	0.00613	NA	0	0	273.2	1	284.9	100644	6.6
16	50.675	-11.478	284.9	101016	6.3	9	0.00615	0	0	0	273.2	1	284.9	NA	6.5
17	50.814	-11.498	284.9	100985	6.2	9.2	0.00617	0	0	0	273.2	1	284.9	100577	6.4
18	50.952	-11.519	284.8	100954	6.1	9.4	NA	0	0	0	273.2	1	284.8	100545	6.3
19	51.091	-11.54	284.8	100922	6.1	9.7	0.00621	0	0	0	273.2	1	284.8	100512	6.2
20	51.229	-11.561	284.7	100891	6	9.9	0.00622	0	0	0	273.2	1	284.7	100480	6.1

Figure 1: Snapshot of Initial Dataset

The introduction in big data, Machine Learning (ML), Artificial Intelligence (AI), Deep Learning has brought about an increase in the number of technological solutions being developed in various industries. Some of these developments have led to significant leaps in understanding and forecasting weather conditions with an impact on business operations and the general wellbeing of communities (Serradilla et al., 2020). As described by (Abdallah et al., 2020) weather forecasting refers to using scientific techniques to predict the atmospheric condition over a specific period based on a specific location. Forecasting involves understanding current weather patterns but examining the relationship between the variables, making calculated estimates of the changes to expect in the future and constantly making changes to details through meteorological practices. The given dataset consists of 5452 rows and 2482 columns. The dataset contains Latitude (XLAT) and Longitude (XLONG) for different locations which will help in ensuring a targeted weather forecast. It also contains other meteorological variables such as Skin Temperature (TSK), Surface Pressure (PSFC), 2-meter specific Humidity (Q2), wind dynamics with the X and Y components of wind at 10 meters (U10 and V10), Convective Rain (Rainc), Non-convective Rain (RAINNC), Snow Water Equivalent (SNOW), Soil Temperature (TSLB) and Soil Moisture (SMOIS). Using data science techniques will help in determining the relationships between the variables identified in the dataset and this will further provide help in developing the models to be used for forecasting (Hewage et al., 2019) (Kelleher and Tierney, 2018). Due to the machine learning and statistical needs of the project, R programming language was used. (Wickham and Grolemund, 2016).

Load the assessment dataset.

```r
Assessment  <- read.csv("C:/WRFdata_May2018.csv", header = FALSE, skip = 1)

View(Assessment)
```

Rows and Column Count

checking for number of rows and columns in the dataset

```
ncol(Assessment)

## [1] 2482

nrow(Assessment)

## [1] 5452
```

Specify New Header Use row 2 as header while deleting row 1 Remove the first row (used as header)

```
names(Assessment) <- as.matrix(Assessment[1,])

Assessment <- Assessment[-1,]
```

Check number of rows in the intial dataset

```
nrow(Assessment)

## [1] 5451
```

check number of columns

```
ncol(Assessment)

## [1] 2482
```

1.2 About Location - Sheffield

After carefully reviewing the weather forecast dataset provided, the location Sheffield was identified from the Latitude (XLAT) and Longitude (XLONG) columns. The latitude 53.37 and longitude -1.448 were checked against Google maps and gave a result of Sheffield. Sheffield is a historically known city of industrialization popularly known for its steel industry which earned it its moniker “Steel City”. It is in South Yorkshire; England and its location suggests exposure to varying weather conditions due to its northern latitude. The weather conditions in Sheffield changes over time ranging from mild to extreme conditions and this plays a critical role in industrial operations and optimization of resources. The longitude and latitude row were identified as row 3587 and was extracted into a new data frame named Sheffield. The new data frame has 2,482 columns and 1 row. Upon extracting the new dataframe, the XLONG and XLAT columns were dropped giving us a total of 2,480 columns left.

```
Sheffield <- Assessment[3587,]

View(Sheffield)
```

Check for the number of columns.

```
ncol(Sheffield)

## [1] 2482

nrow(Sheffield)

## [1] 1
```

Remove the longitude and latitude column as they are no longer required.

```
Sheffield1 <- Sheffield[, -(1:2)]
```

check for number of columns left

```
ncol(Sheffield1)

## [1] 2480
```

1.3 Significance of Sheffield

Sheffield offers a growing level of industrialization coupled with its other thriving sectors such as agriculture, transportation, and urban relations. An understanding of the weather dynamics is crucial as this would aid in planning business operations, help with optimizing allocation of resources, ease of decision making while also ensuring the security of lives.

1.4 Problem Statement

The aim of this report is to review and analyze the weather forecast data for Sheffield using various meteorological parameters given in the dataset to understand the trends and patterns over time. Due to the uncertainty of the weather which can have negative impacts on productivity and operations in general. With the growing community in the city of Sheffield and increase in industrialization, it has become more significant to identify the weather patterns in the city. Conventional approaches to predicting or forecasting the weather patterns have proved unreliable, often leading to reactive approaches and reduced productivity.

1.5 Research Hypothesis

The goal of this project is to develop an adequate time-series and machine learning model to be used for weather forecasting. After establishing the goal and reviewing the Sheffield dataset extracted, the following hypothesis was deduced.

1. Null Hypothesis: There is a significant correlation between Convective rain (Accumulated precipitation) and 2- meter specific humidity.

Alternative Hypothesis: There is no significant correlation between Convective rain (Accumulated precipitation) and 2- meter specific humidity.

2. Null Hypothesis: There is a significant correlation between Surface Pressure (PSFC) and Wind Speed (U10).

Alternative Hypothesis: There is no significant correlation between Surface Pressure (PSFC) and Wind Speed (U10).

3. Null Hypothesis: There is a significant correlation between Surface Pressure (TSK) and Windspeed (WINDS).

Alternative Hypothesis: There is no significant correlation between Surface Pressure (TSK) and Windspeed (WINDS).

1.6 Research Questions

The following research questions have been deduced after carefully reviewing the initial dataset, extracting the Sheffield location data into a new dataset.

Some of the suggested research questions for this report include:

1. What is the standard deviation of Surface temperature (TSK)?
2. Is there a relationship between Surface Pressure (PSFC) and Wind Speed (U10)?
3. Is there a relationship between Convective rain (Accumulated precipitation) and 2- meter specific humidity(Q2)?
4. Is there a relationship between Surface Pressure or Skin Temperature (TSK) and Windspeed (WINDS)?
5. Is there a relationship between Soil Temperature (TSLB), Soil Moisture (SMOIS) and Surface Pressure or Skin Temperature (TSK)?
6. What is the best model to use for time series forecasting?
7. What is the best model to use for Machine learning?

1.7 Significance of Data Analysis

1.7.1 Rationale for data analysis

Analyzing the meteorological variables in Sheffield dataset will provide insight into the relationships between the variables, an understanding of the historical trends therefore identifying trends or patterns and help develop machine learning and time series models to be used for forecasting purposes.

1.7.2. Stakeholders and why is analysis important for them?

Some of the stakeholders in Sheffield include farmers, steel industries, hospitals, local authorities, government agencies, emergency response teams.

Some of the stakeholders in Sheffield considered and why analysis is important to them are:

1. Public Safety and Security: The accuracy of weather forecasts helps businesses, communities, and their governments to prepare for extreme weather conditions, ensuring adequate response to minimize the effects. Occurrences like snowfalls, snow pellets, heatwaves, flood etc. People can take proactive measures to prevent such occurrences, thereby ensuring the safety and security of lives and properties.

2. Agriculture: Another major significance of choosing Sheffield is for its agricultural exploits. The surrounding countryside in Sheffield has good agricultural land which is used for growing crops, rearing livestock, and other agricultural produce. Accurate weather forecasts help farm businesses and owners make decisions on where to plant, fertilize their crops and harvest their yields which in turn leads to operational effectiveness and efficiency.

3. Steel Industry: Sheffield is popularly known as the steel industry a lot of industrialization occurs in the city. From past experiences, weather conditions have played a major role in the steel industrialization and the accurate weather forecasts will lead to an optimization of resource utilization such as coal and water therefore leading to less downtimes in operations. Another thing to consider in the steel industry business is the supply chain process which involves the acquisition of materials, importing and exporting of goods and services. Accurate weather predictions allow businesses to make plans for changing

weather conditions, therefore meeting customer needs, and ensuring no operational disruptions.

4. Health: By analyzing weather conditions, patterns can be made regarding the nature of public health at different time intervals. Analyzing the trends can help make preventive decisions against illnesses such as infections, heat cramps, etc., that might occur at a particular period.

1.7.3. Impact on Decision Making.

Considering some of the stakeholders mentioned above, some of the impact of data analysis on decision making include:

1. Scheduling irrigation and planting seasons for farmers.
2. Anticipate pest breakouts on farmlands and set proactive measures against them.
3. Mitigating against road and rail network issues and dangers based on historical data.
4. Ensuring operational effectiveness and efficiency in industries and businesses.
5. Mitigating against risks associated with certain weather conditions for staff by setting up preventive measures.
6. Resource optimization for organizations and government alike. Government can adequately make plans on where and when to allocate resources, when the resources need to be allocated and how many resources might be needed.

2. LITERATURE REVIEW

2.1 Systematic Review Method

In this section of the report, a systematic review method involving a thorough review process of highlighting applicable research books, articles, journals, and web pages related on weather forecasting, time series forecasting, machine learning forecasting (Schröer et al., 2021) , (Cheng et al., 2022), (Carvalho et al., 2019), .

The review involves a exhaustive process of researching the research topic materials, reviewing the relevant literature extracted in the research field and understanding if the information contained is useful towards the research topic or not. Materials are collected from various sources such as Bolton library, IEEE Explore library, Google Scholar, AI Search, Google Internet search etc.

The review process also includes researching on the various machine learning and time series technologies adopted and how to check for their levels of accuracy in forecasting (Amram et al., 2021).

It involves a mixed research approach as it combines quantitative and qualitative exploration, and making relevant deductions from the observations. (Daniel, 2016), (Beinschroth, 2022), (Lee et al., 2020), (Dayo-Olupona et al., 2023)

Based on the search results, previous studies have utilized similar machine learning, time series and statistical techniques to analyze meteorological parameters which has given valuable insight into weather patterns, changes in climate, etc.

2.2 Literatures Reviewed

In this section of the report, studies that have been deemed relevant to the research topic are reviewed, their implications, strengths and weaknesses are analyzed.

Some of the variables considered for forecasting are Surface pressure, Windspeed, Soil Moisture etc. Especially the impact on windspeed on business operations and urban development (Tawn and Browell, 2022), (Xinxin et al., 2023). The report's limitation, however, is its focus on windspeed as this might limit the generalization of findings based on the independent variables used.

Progress and prospects in weather and climate modeling (Raghavan et al., 2020), gives a detailed rundown of recent advancements made in short to medium range weather prediction models. It further highlights the importance of accurate weather forecasts in minimizing the impact of extreme weather conditions on urban communities. While the report identifies areas of significant achievements and highlights potential areas for improvement, it lacks specificity in certain areas limiting it to specific issues.

Deep learning has also been used and shown promising results in weather forecasting due to its ability to capture complex nonlinear relationships withing meteorological data. Its focus on machine learning fault diagnosis is significant however it might also be considered a weakness as it limits the depth of the analysis conducted (Zhu et al., 2022).

Another suggested method to use is the Hilbert-Huang Transform or HHT, which is a powerful tool for analyzing nonlinear relationships and non- stationary time series data. This was considered to be used for the time-series forecasting due to its ability to draw out valuable information from weather data and its accuracy in prediction, however it doesn't address limitations that the HHT method might have in weather forecast application (Huang and Wu, 2008).

Further exploration into related methods suggested the Convolutional Neural Networks which excels in its analysis of complex meteorological dataset as detailed in by (Liu et al., 2019) when it deployed CNN in interpreting atmospheric data and oceanic remote sensing data. The limitation is its areas of focus which can affect the depth of analysis and insights generated.

3. METHODOLOGY AND TECHNIQUES

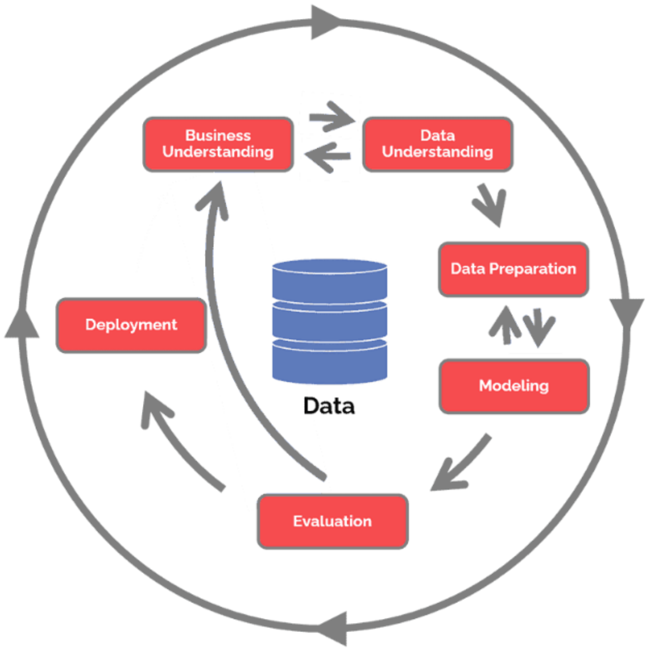

Figure 2: CRISP-DM

Source:(Hotz, 2018a)

3.1 CRISP-DM Methodology

"Cross Industry Standard Process for Data Mining" is regarded as a standard data science approach adopted to provide structure to the planning process involved in the projects, organizing of the project, and its implementation of the project (Talaviya, 2023), (Hotz, 2018b), (Hayat Suhendar and Widyani, 2023).

1. Business Understanding: This step involves understanding the problem, defining the aim and objectives, and proposed goal for weather forecasting. Sheffield is our focus and therefore the needs of Sheffield are considered in this phase of the project. Its desired goals and objectives are set, the expected outcomes are suggested, and the resources available for the project are listed.

2. Data Understanding: This involves collecting and analyzing the Sheffield datasets. In this step, dataset exploration occurs, checking the relationship between the variables, and data quality is verified. In this phase, the Sheffield dataset is explored to understand the relationships that exist between the meteorological variables.

3. Data preparation: Just as the step says, it involves preparing the dataset for modelling. Data preparation is the most critical part of each project, and it can be referred to as the data processing steps such as data restructuring, handling missing values, handling outliers, and creating new variables such as the Windspeed column. Missing values are checked and handled, outliers are checked and confirmed if they are true outliers or not, and handled if they are. The windspeed is also calculated using the square root of the sum of the square of the x component of wind(U10) and the y component of wind (v10).

4. Modelling: This phase involves using statistical analysis, machine learning techniques and time series techniques to analyze the trends and patterns observed in the dataset. The observations are then used to make predictions or forecasts. Data modelling involves selecting the preferred modelling techniques, splitting the dataset, training the model using the split data and eventually deploying the model trained. In machine learning, the restructured and pre-processed Sheffield dataset is split using the train test split, the modelling techniques are trained, and deployed.

5. Evaluation: This phase involves checking the accuracy and reliability of the models deployed. This is used to confirm if the models deployed got a better result or not. A low accuracy would suggest that the model deployed isn't the best model to use. The root mean square error (RMSE) is used to evaluate the deployed models for machine learning, and the model with the least error is selected as the preferred model. The AIC score is used to determine the best model for time series forecasting.

6. Deployment: Deployment phase involves taking the results of the models generated and integrating them into the stakeholders identified in Sheffield. The models are deployed into a production environment to help in weather forecasting.

4. DATA PRE-PROCESSING STEPS

Data pre-processing are the steps taken to ensure data quality and data reliability. These are steps carried out to ensure a dataset is ready before analysis is carried out. This section of the report explores the pre-processing steps carried out before performing EDA and statistical analysis. Steps such as data cleaning, handling missing values using linear interpolation, checking for outliers using boxplot, printing out the list of outliers and crosschecking the printed list with the dataset if they are true outliers or not before handling them.

4.1 Data Restructuring

The dataset currently has 2480 columns with 10 identical column headers being constantly repeated. The dataset is to be re-arranged where each column head value is stacked one after the other. This is done by turning the elongated columns into rows of 10s with the values following one another. The code below attempts to do this by splitting the dataframe into groups of 10s.

```
Splitsheffield <- split.default (Sheffield1, rep(1:248, each = 10))

View(Splitsheffield)
```

Combine column names + Sheffiled Using lapply

```
#using lapply
Newshef <- lapply(Splitsheffield, function(x){
  Newshef <- cbind(x)
  colnames(Newshef) <- c("TSK", "PSFC", "U10", "V10", "Q2", "RAINC",
"RAINNC", "SNOW", "TSLB", "SMOIS")
  return(Newshef)})
View(Newshef)
```

rbind is used to bind the respective column headers

```
Sheffield2 <- do.call(rbind, Newshef)

View(Sheffield2)
```

check for number of rows in Sheffield 2

```
nrow(Sheffield2)

## [1] 248
```

Check for number of columns

```
ncol(Sheffield2)

## [1] 10
```

Check the characters

```
str(Sheffield2)

## 'data.frame': 248 obs. of 10 variables:
## $ TSK : chr "276.3" "276.1" "278.3" "287.7" ...
## $ PSFC : chr "98848" "98817" "98823" "98787" ...
## $ U10 : chr "3.5" "4.8" "4.6" "3.6" ...
## $ V10 : chr "-2.5" "-0.7" "-0.1" "2.9" ...
## $ Q2 : chr "0.00314" "0.00430" "0.00342" "0.00337" ...
## $ RAINC : chr "0.0" "0.0" "0.0" "0.0" ...
## $ RAINNC: chr "0.0" "0.0" "0" "0.0" ...
## $ SNOW : chr "0.0" "0.0" "0.0" "0.0" ...
## $ TSLB : chr "277.5" "276.8" "277.4" "283.4" ...
## $ SMOIS : chr "0.3212" "0.3207" "0.3201" "0.3196" ...
```

To change the dataset to its actual character representation, type.convert is used as shown below.

```
Sheffield2[] <- lapply(Sheffield2, function(x) type.convert(as.character(x),
as.is = FALSE))
```

check the new character.

```
str(Sheffield2)

## 'data.frame': 248 obs. of 10 variables:
## $ TSK : num 276 276 278 288 293 ...
## $ PSFC : int 98848 98817 98823 98787 98694 98607 98571 98421 98250
98044 ...
## $ U10 : num 3.5 4.8 4.6 3.6 4.3 5.3 2.5 1.4 1.5 0.7 ...
## $ V10 : num -2.5 -0.7 -0.1 2.9 4.7 5.9 5.7 7.2 8.3 9.2 ...
## $ Q2 : num 0.00314 0.0043 0.00342 0.00337 0.00406 0.00421 0.00467
0.00493 0.0052 NA ...
## $ RAINC : num 0 0 0 0 0 0 0 NA 0 0 ...
## $ RAINNC: num 0 0 0 0 0 0 NA 0 0.1 0.5 ...
## $ SNOW : num 0 0 0 0 0 0 0 0 0 0 ...
## $ TSLB : num 278 277 277 283 289 ...
## $ SMOIS : num 0.321 0.321 0.32 0.32 0.319 ...
```

check data summary.

```
summary(Sheffield2)

##       TSK             PSFC              U10                V10
##  Min.   :276.1   Min.   : 97884   Min.   :-6.8000   Min.   :-5.3000
##  1st Qu.:282.8   1st Qu.: 99572   1st Qu.:-1.4000   1st Qu.:-3.1250
##  Median :287.1   Median :100097   Median :-0.2000   Median :-1.0000
##  Mean   :288.5   Mean   : 99947   Mean   : 0.1133   Mean   :-0.5775
##  3rd Qu.:294.1   3rd Qu.:100361   3rd Qu.: 1.7000   3rd Qu.: 1.7250
##  Max.   :304.2   Max.   :101074   Max.   : 6.5000   Max.   : 9.2000
##  NA's   :8       NA's   :8        NA's   :7         NA's   :8
```

```
##          Q2                    RAINC                 RAINNC                  SNOW
##   Min.     :0.003140    Min.     :0.00000    Min.     : 0.0000    Min.     :0
##    1st  Qu.:0.005155     1st  Qu.:0.00000     1st  Qu.: 0.0000     1st  Qu.:0
##    Median  :0.006550     Median  :0.00000     Median  : 0.0000     Median  :0
##   Mean     :0.006506    Mean     :0.05546    Mean     : 0.5481    Mean     :0
##    3rd  Qu.:0.007760     3rd  Qu.:0.00000     3rd  Qu.: 0.1000     3rd  Qu.:0
##   Max.     :0.010470    Max.     :2.60000    Max.     :12.2000    Max.     :0
##   NA's    :5            NA's    :10          NA's    :7            NA's    :6
##                            TSLB                                          SMOIS
##         Min.                    :276.8                 Min.                :0.2704
##              1st         Qu.:283.0                          1st        Qu.:0.2834
##              Median         :285.9                          Median        :0.2993
##         Mean                    :287.4                 Mean                :0.2965
##              3rd         Qu.:291.4                          3rd        Qu.:0.3079
##         Max.                    :300.4                 Max.                :0.3265
##  NA's   :5        NA's   :12
```

4.2 Missing Values

4.2.1 Identify Missing Values.

To check for missing values in a dataset, is.na(“name of dataframe”) is used as shown below.

```
is.na(Sheffield2)
```

```
##           TSK   PSFC    U10    V10     Q2  RAINC  RAINNC   SNOW   TSLB  SMOIS
## 1       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 2       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 3       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 4       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 5       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 6       FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
```

```
## 7     FALSE FALSE FALSE FALSE FALSE FALSE   TRUE FALSE FALSE FALSE
## 8     FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 9     FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 10    FALSE FALSE FALSE FALSE  TRUE FALSE  FALSE FALSE FALSE FALSE
## 11    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 12    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 13    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 14    FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 15    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 16    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 17    FALSE FALSE FALSE FALSE  TRUE FALSE  FALSE FALSE FALSE  TRUE
## 18    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 19    FALSE FALSE  TRUE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 20    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 21    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 22    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  TRUE FALSE
## 23    FALSE FALSE  TRUE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 24    FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 25    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 26    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 27    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 28    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 29    FALSE FALSE FALSE  TRUE FALSE FALSE  FALSE FALSE FALSE FALSE
## 30    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 31    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 32    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 33    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 34    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 35    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 36    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 37    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 38    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 39    FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  TRUE FALSE FALSE
```

```
## 40   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 41   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE  TRUE FALSE
## 42   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 43   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 44   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 45   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE  TRUE
## 46   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 47   FALSE FALSE  TRUE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 48   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 49   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 50   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 51   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 52   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 53   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 54   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 55   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 56   FALSE  TRUE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 57   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 58   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 59   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 60   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 61   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 62   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 63   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 64   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 65   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 66   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 67   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 68   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 69   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 70   FALSE FALSE  TRUE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 71   FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 72   FALSE FALSE  TRUE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
```

```
## 73    FALSE   TRUE   TRUE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE   TRUE
## 74    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 75    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 76    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 77    FALSE  FALSE  FALSE  FALSE  FALSE   TRUE   FALSE  FALSE  FALSE  FALSE
## 78    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 79    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 80     TRUE  FALSE  FALSE  FALSE  FALSE   TRUE   FALSE  FALSE  FALSE  FALSE
## 81    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 82    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 83    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 84    FALSE   TRUE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 85    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE   TRUE  FALSE  FALSE
## 86    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 87    FALSE  FALSE  FALSE  FALSE  FALSE   TRUE   FALSE  FALSE  FALSE  FALSE
## 88    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 89    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE    TRUE  FALSE  FALSE   TRUE
## 90    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 91    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 92    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 93    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 94    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 95    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE   TRUE
## 96    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 97    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 98    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 99    FALSE   TRUE  FALSE  FALSE   TRUE  FALSE   FALSE  FALSE  FALSE  FALSE
## 100   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 101   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 102   FALSE  FALSE  FALSE   TRUE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 103   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
## 104   FALSE   TRUE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE   TRUE
## 105   FALSE  FALSE  FALSE  FALSE  FALSE  FALSE   FALSE  FALSE  FALSE  FALSE
```

```
## 106 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE  TRUE
## 107 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 108 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 109 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 110 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 111 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 112 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE  TRUE FALSE
## 113 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 114 FALSE  TRUE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 115 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 116 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 117 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 118 FALSE FALSE FALSE  TRUE FALSE FALSE   FALSE FALSE FALSE FALSE
## 119 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 120 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 121 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 122 FALSE FALSE FALSE FALSE  TRUE FALSE   FALSE FALSE FALSE FALSE
## 123 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 124 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE  TRUE FALSE FALSE
## 125 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 126 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 127 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 128 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 129 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 130 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 131 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 132  TRUE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 133 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 134 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 135 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 136 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 137 FALSE FALSE FALSE  TRUE FALSE FALSE   FALSE FALSE FALSE  TRUE
## 138 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
```

```
## 139 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 140 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 141 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 142 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE  TRUE FALSE FALSE
## 143 FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 144  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 145  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 146 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 147 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 148 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 149 FALSE FALSE FALSE  TRUE FALSE FALSE  FALSE FALSE FALSE FALSE
## 150 FALSE FALSE FALSE FALSE  TRUE FALSE  FALSE  TRUE FALSE FALSE
## 151 FALSE FALSE  TRUE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 152 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 153 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 154 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 155 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 156 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 157 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 158 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 159 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 160 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 161 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 162 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 163  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 164 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 165 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 166 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 167 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 168 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 169 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 170 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 171 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
```

```
## 172 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 173 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE  TRUE
## 174 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 175 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 176 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 177 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 178 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 179 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 180  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 181 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 182 FALSE FALSE FALSE FALSE FALSE FALSE   TRUE FALSE FALSE FALSE
## 183 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 184 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 185 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 186 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 187 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 188  TRUE FALSE FALSE  TRUE FALSE FALSE  FALSE FALSE FALSE FALSE
## 189 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 190 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 191 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 192 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 193 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 194 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 195 FALSE  TRUE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 196 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE  TRUE
## 197 FALSE  TRUE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 198 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 199 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 200 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 201 FALSE FALSE FALSE FALSE FALSE FALSE   TRUE FALSE FALSE FALSE
## 202 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  TRUE FALSE
## 203 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 204 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
```

```
## 205 FALSE FALSE FALSE FALSE FALSE  TRUE   FALSE FALSE  TRUE FALSE
## 206 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 207 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 208 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 209 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 210 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 211 FALSE FALSE FALSE FALSE FALSE FALSE    TRUE FALSE FALSE FALSE
## 212 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 213 FALSE FALSE FALSE  TRUE FALSE FALSE    TRUE  TRUE FALSE FALSE
## 214 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 215 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 216 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 217 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 218 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 219 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 220 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 221 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 222 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 223 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 224 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 225 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 226 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 227 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 228 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 229 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 230 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 231 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 232 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE  TRUE
## 233 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 234  TRUE FALSE FALSE FALSE FALSE FALSE    TRUE FALSE FALSE FALSE
## 235 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 236 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
## 237 FALSE FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE FALSE
```

```
## 238 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 239 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 240 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 241 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 242 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 243 FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 244 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE  TRUE
## 245 FALSE FALSE FALSE FALSE FALSE  TRUE  FALSE FALSE FALSE FALSE
## 246 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 247 FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE FALSE FALSE
## 248 FALSE FALSE FALSE  TRUE FALSE FALSE  FALSE FALSE FALSE FALSE
```

The output gives a general view of all the dataset and possible missing values.in the output above, missing values are represented with TRUE. This is because the code checks each row in the column to confirm if there's a missing value, it returns FALSE if there is no missing value and TRUE if any is identified. however, due to the volume of data being displayed, it can be confusing to identify all the missing values (TRUE). This method of identifying missing values is not the best approach as it can be misdirecting and time confusing.

Therefore, we first try to find out the total number of missing values in the dataset as shown below.

```
sum(is.na(Sheffield2))

## [1] 76
```

Upon identifying the total number of missing values in the dataset, the code is written to display the total number of missing values in each column as shown below.

```
colSums(is.na(Sheffield2))

##      TSK     PSFC      U10      V10       Q2   RAINC  RAINNC     SNOW     TSLB   SMOIS
##        8        8        7        8        5      10       7        6        5      12
```

4.2.2 Visualize the Missing values.

To visualize the missing values, the sum of missing values in each column is saved into a dataframe named Nullvalue, converts the null values vector to a regular vector that stores the count of missing values and stores that in the Nulldf dataframe.

```
Nullvalue <- colSums(is.na(Sheffield2)) #save the missing values in each column into a dataframe
Nulldf <- tibble (variable = names(Nullvalue), missing = as.vector(Nullvalue))
View(Nulldf) #view the dataframe
```

Upon saving the missing values identified into a dataframe, a bar chart is plotted to visualize the dataframe.

```
#intiate a barplot using ggplot

library(ggplot2)
ggplot(data = Nulldf, aes(x = variable, y = missing, fill = variable)) + geom_bar(stat = "identity") +
  labs(x = "Null Values in each column", y = NULL, title = "Missing Values Visualization") +
  scale_fill_manual(values = rainbow(nrow(Nulldf)))
```

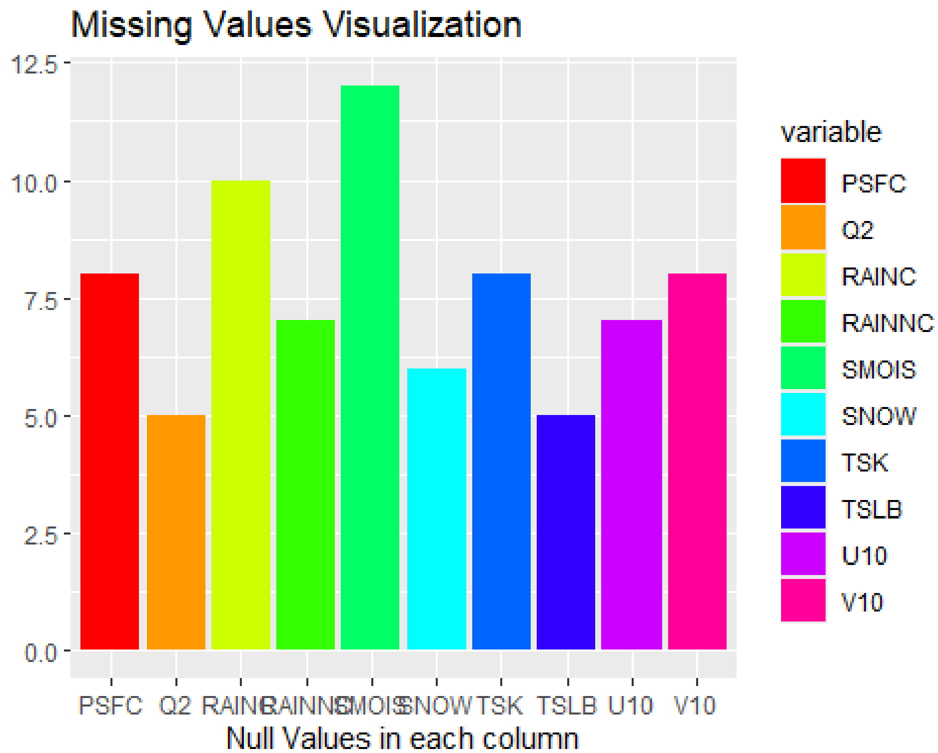

Figure 3: Visualize Missing Values

4.2.3 Handle Missing values

To handle missing values, the linear interpolate function is called.

```
library(imputeTS) #using linear interpolate.

##      Registered      S3      method      overwritten      by      'quantmod':
##               method                                                      from
##   as.zoo.data.frame zoo
```

The na_interpolation() function is called to handle missing values. Interpolation is the process of estimating the values of the missing values based on the values close to or around the missing value.

```
Sheffield2 <- na_interpolation(Sheffield2)
```

Check for the sum of missing values in each colum respectively after handling missing values.

```
colSums(is.na(Sheffield2))#check for the total sum of missing values in each columns
##    TSK   PSFC    U10    V10     Q2  RAINC RAINNC   SNOW   TSLB  SMOIS
##      0      0      0      0      0      0      0      0      0      0
```

4.3 Outliers

4.3.1 Detect Outliers

Outliers are considered as datapoints that differ from the actual observations. There are different methods of detecting outliers, some of which are boxplot, scatterplot, etc. in this section of the report, boxplot is used to visualize the outliers.

```
#TSK
boxplot(Sheffield2$TSK, main = "TSK Boxplot")
```

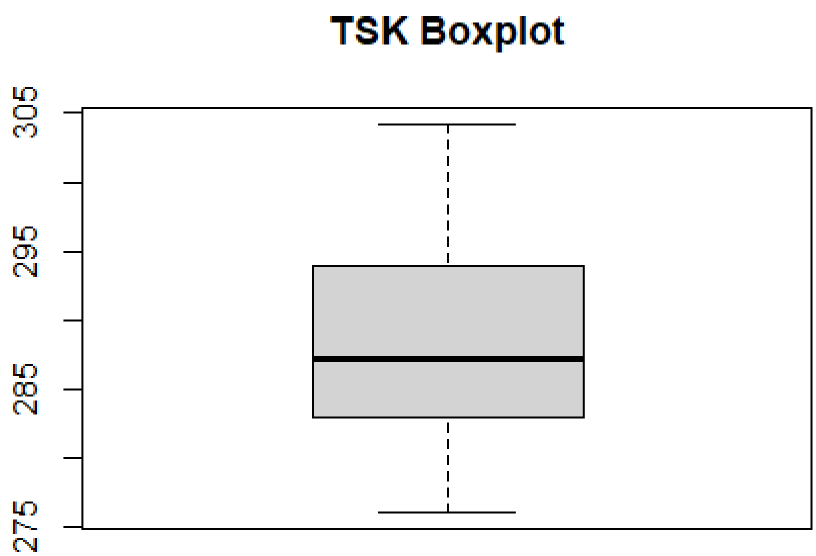

Figure 4: TSK Boxplot

```r
hist(Sheffield2$TSK, main = "TSK Histogram")
```

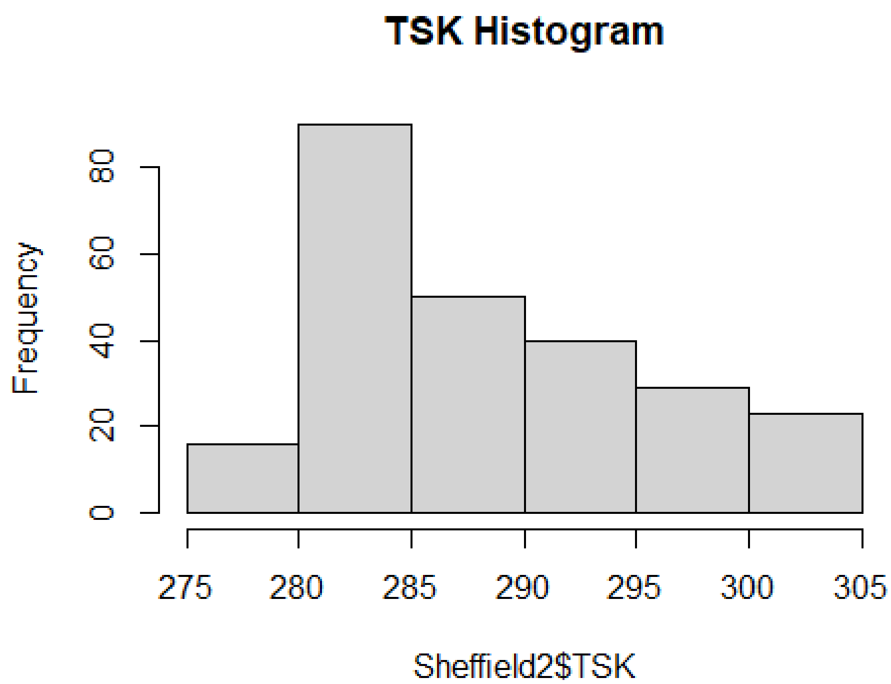

Figure 5: TSK Histogram

```r
#PSFC
boxplot(Sheffield2$PSFC, main = "PSFC Boxplot")
```

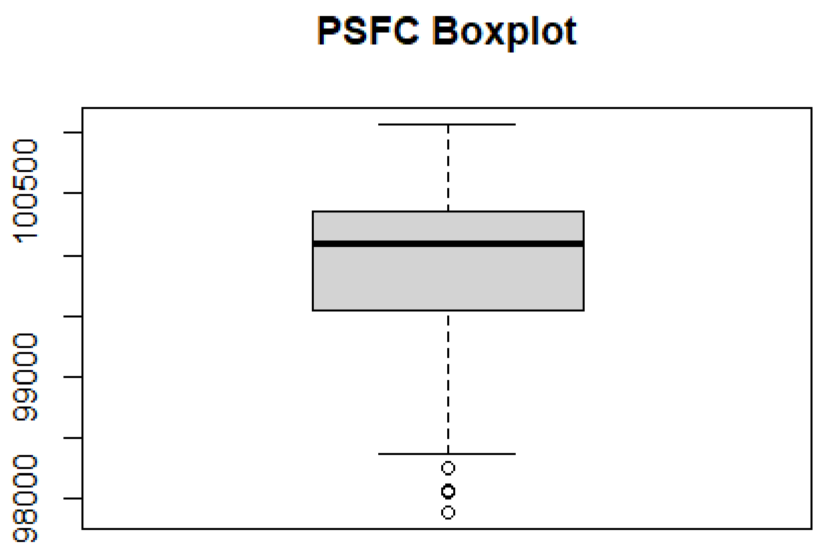

Figure 6: PSFC Boxplot

Figure 6 shows outliers are detected.

```
#U10
boxplot(Sheffield2$U10, main = "U10 Boxplot")
```

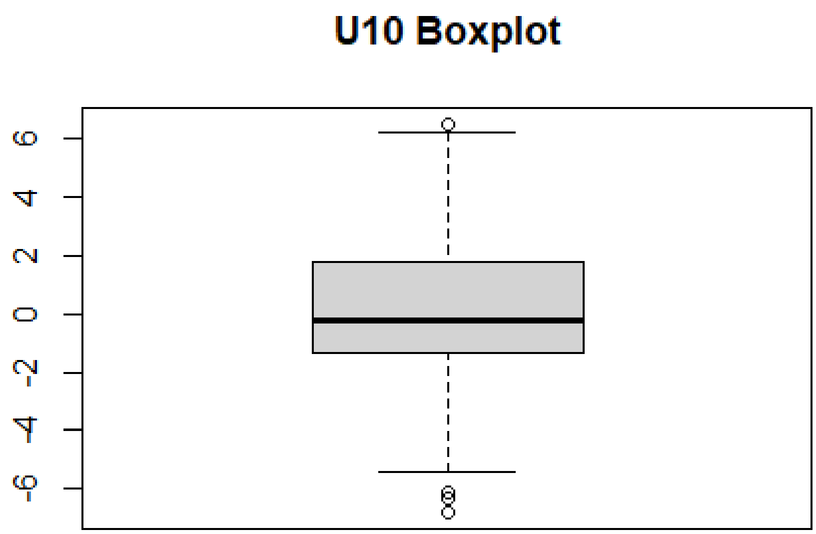

Figure 7: U10 Boxplot

Figure 7 shows outliers are detected.

```
#V10
boxplot(Sheffield2$V10, main = "V10 Boxplot")
```

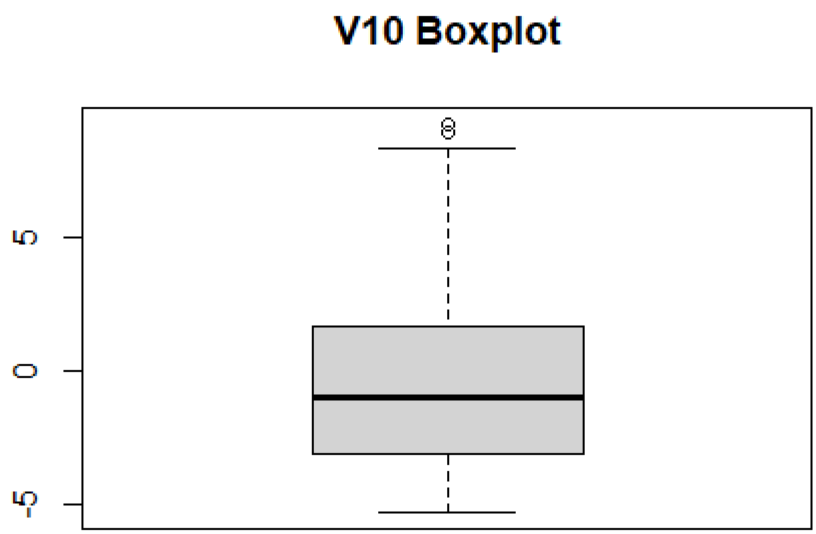

Figure 8: V10 Boxplot

Figure 8 shows outliers are detected.

```r
#Q2
boxplot(Sheffield2$Q2, main = "Q2 Boxplot")
```

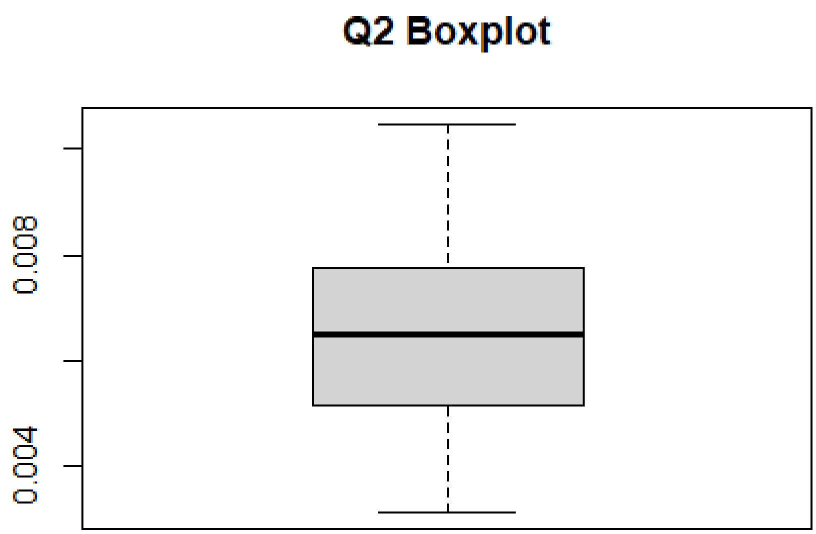

Figure 9: Q2 Boxplot

```r
#RAINC
boxplot(Sheffield2$RAINC, main = "RAINC Boxplot")
```

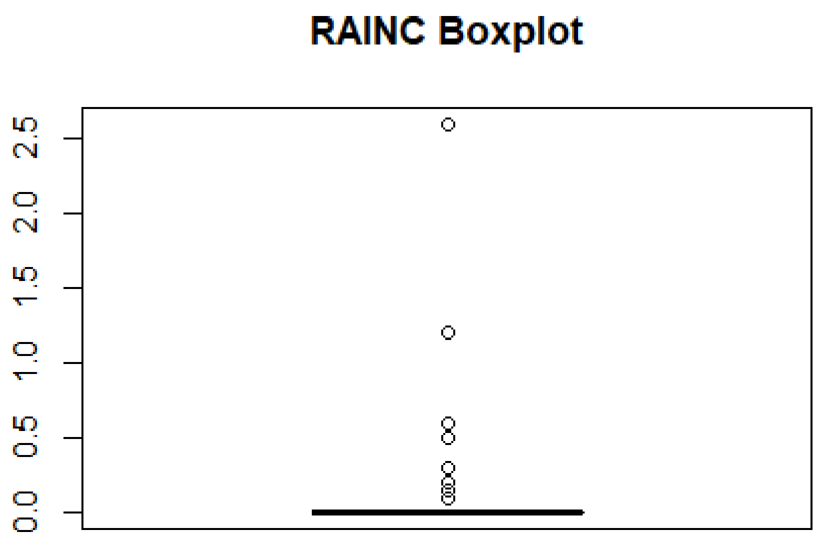

Figure 10: RAINC Boxplot

Figure 10 shows Outliers detected.

```
#RAINNC
boxplot(Sheffield2$RAINNC, main = "RAINNC Boxplot")
```

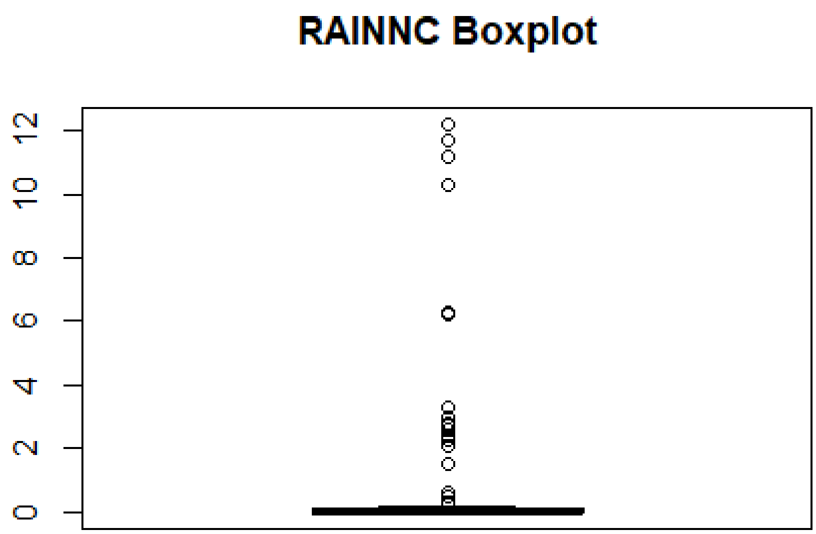

Figure 11: RAINNC Boxplot

Figure 11 shows outliers detected.

```
#SNOW
boxplot(Sheffield2$SNOW, main = "SNOW Boxplot")
```

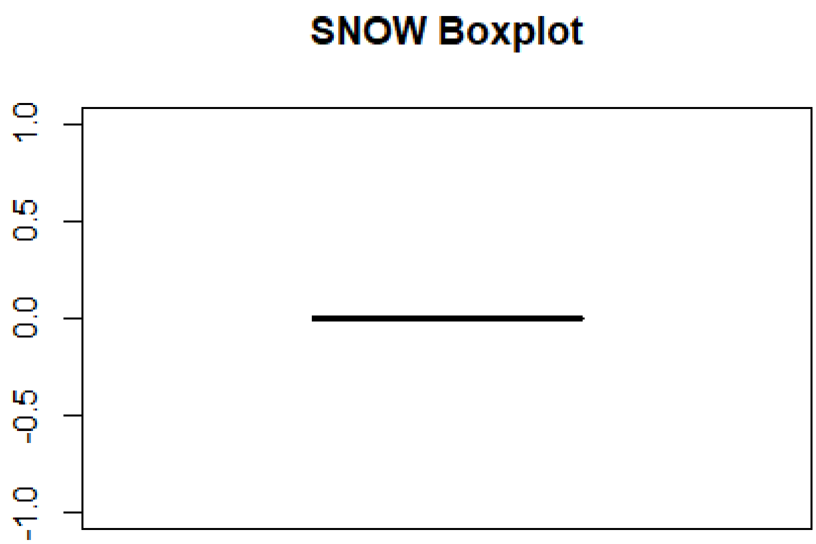

Figure 12: SNOW Boxplot

```r
#TSLB
boxplot(Sheffield2$TSLB, main = "TSLB Boxplot")
```

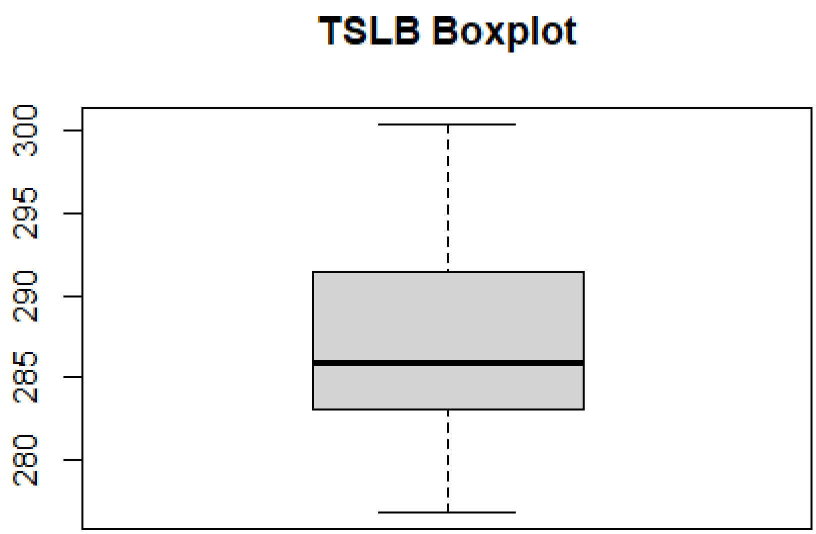

Figure 13: TSLB Boxplot

```r
#SMOIS
boxplot(Sheffield2$SMOIS, main = "SMOIS Boxplot")
```

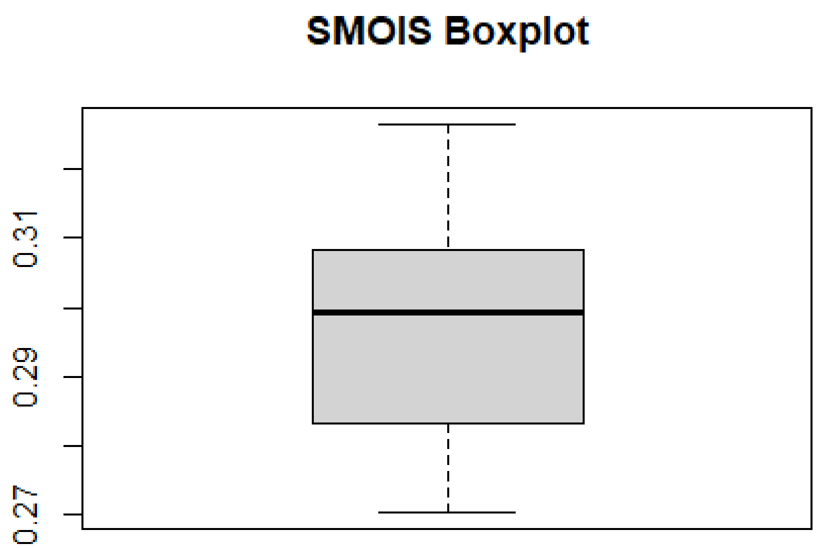

Figure 14: SMOIS Boxplot

To confirm if the outliers detected in the visualization are true outliers, the code below attempts to print a list of all outliers identified in each column. The outlier list of each column is checked against the actual dataset to confirm if they are true outliers and is needed to be handled or not.

```
detect_outliers <- function(x) {
  q1 <- quantile(x, 0.25, na.rm = TRUE)
  q3 <- quantile(x, 0.75, na.rm = TRUE)
  iqr <- q3 - q1
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  outliers <- x[x < lower_bound | x > upper_bound]
  return(outliers)
}
```

```
# apply the defined function to each variable in the Sheffield2 dataset
outliers <- lapply(Sheffield2, detect_outliers)

# Print outliers for each column
for (i in 1:length(outliers)) {
  cat("Outliers in column", names(outliers)[i], ":", toString(outliers[[i]]),
"\n")
}

## Outliers in column TSK :
## Outliers in column PSFC : 98250, 98044, 97884, 98061, 98356
## Outliers in column U10 : 6.5, -6.1, -6.3, -6.8
## Outliers in column V10 : 9.2, 8.9
## Outliers in column Q2 :
## Outliers in column RAINC : 0.6, 1.2, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.2, 0.2, 0.1, 0.5, 0.5, 0.5, 0.5, 0.15, 0.3, 2.6, 2.6
## Outliers in column RAINNC : 0.5, 2.6, 3, 3, 3, 3, 3, 2.1, 3.3, 3.3, 3.3,
3.3, 3.3, 3.3, 3.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.5, 2.3, 2.4, 2.4,
2.4, 2.4, 2.4, 2.4, 0.3, 0.6, 1.5, 2.7, 2.8, 2.8, 2.8, 6.3, 10.3, 11.2, 11.7,
12.2, 6.25, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3
## Outliers in column SNOW :
## Outliers in column TSLB :
## Outliers in column SMOIS :
```

Outliers listed above are crosschecked against the dataset. Only RAINNC has true outliers as the results are far from the observations. To handle the outliers, the mean imputation method is used.

4.3.2 Handle Outliers

```
library(Hmisc)

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units

#handle outliers in RAINNC using Winsorization
library(DescTools)

##
## Attaching package: 'DescTools'

## The following objects are masked from 'package:Hmisc':
##
##     %nin%, Label, Mean, Quantile

# Calculate winsorization limits
lower_limit <- quantile(Sheffield2$RAINNC, 0.05)
upper_limit <- quantile(Sheffield2$RAINNC, 0.95)

# Apply winsorization
Sheffield2$RAINNC[Sheffield2$RAINNC < lower_limit] <- lower_limit
Sheffield2$RAINNC[Sheffield2$RAINNC > upper_limit] <- upper_limit
```

plot a boxplot to check if the outliers have been handled.

```r
boxplot(Sheffield2$RAINNC, main = "New RAINNC Boxplot")
```

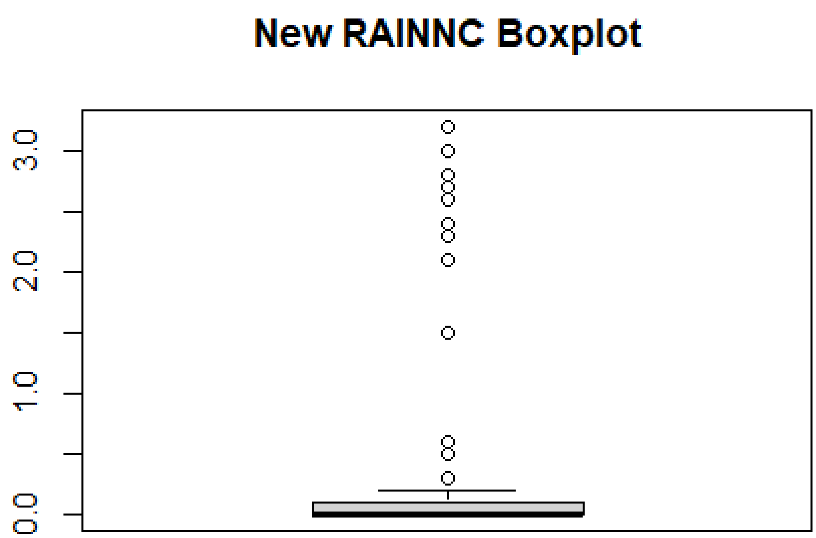

Figure 15: New RAINNC Boxplot

To easily see the values of the handled outliers and compare them against one another, print the detected outliers.

```r
detect_outliers2 <- function(x) {
  q1 <- quantile(x, 0.25, na.rm = TRUE)
  q3 <- quantile(x, 0.75, na.rm = TRUE)
  iqr <- q3 - q1
  lower_bound <- q1 - 1.5 * iqr
  upper_bound <- q3 + 1.5 * iqr
  outliers2 <- x[x < lower_bound | x > upper_bound]
```

```
  return(outliers2)
}

## Apply the defined function to the 'RAINNC' column in the Sheffield2 dataset
outliers2 <- detect_outliers2(Sheffield2$RAINNC)

# Print outliers for the 'RAINNC' column
cat("Outliers in column RAINNC:", toString(outliers2), "\n")

## Outliers in column RAINNC: 0.5, 2.6, 3, 3, 3, 3, 3, 2.1, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.5, 2.3, 2.4, 2.4, 2.4, 2.4, 2.4, 2.4, 0.3, 0.6, 1.5, 2.7, 2.8, 2.8, 2.8, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 3.19499999999999, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3
```

4.4 Windspeed

To Calculate the value of windspeed, we make use of the U10 and V10 variable respectively. Windspeed is the square root of the sum U10 Squared and V10 squared.

sum = (U10 2) + (V10 2)

Windspeed (WINDS) = Sqrt(sum)

```
Sheffield3 <- Sheffield2 #create a copy of the dataset

Sheffield3$WINDS <- sqrt((Sheffield3$U10)**2 + (Sheffield3$V10)**2)
View(Sheffield3)
```

5. EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis, also known as EDA, involves exploring relationships, understanding patterns, and peculiarities between variables. EDA is used to have an in-depth insight into the levels of distribution and structure of variables in a dataset. In this section of the report, the relationships between various variables are explored, outliers are detected, trends or patterns are established, correlation levels are checked, and visualizations are plotted.

Descriptive Statistics

The dplyr package for data manipulation. The summary() function to calculate summary statistics (including mean, median, quartiles, min, and max) for numerical variables in the dataset. Additionally, calculate mean, median, standard deviation, min, and max values for numerical variables using sapply() function. We combine the calculated summary statistics into a dataframe named summary_df. Finally, we print the summary_df dataframe to display the descriptive statistics for each numerical variable in the Sheffield dataset.

```
# Display summary statistics for numerical variables
Summarystats <- summary(Sheffield3)

#USING SAPPLY
# Display mean, median, standard deviation, min, max for numerical variables
mean_values <- sapply(Sheffield3, mean, na.rm = TRUE)
median_values <- sapply(Sheffield3, median, na.rm = TRUE)
std_dev_values <- sapply(Sheffield3, sd, na.rm = TRUE)
min_values <- sapply(Sheffield3, min, na.rm = TRUE)
max_values <- sapply(Sheffield3, max, na.rm = TRUE)

# Combine summary statistics into a dataframe
```

```
summarydf <- data.frame(
  Mean = mean_values,
  Median = median_values,
  StdDev = std_dev_values,
  Min = min_values,
  Max = max_values
)

# Print summary statistics
print(summarydf)

##                 Mean        Median       StdDev          Min          Max
## TSK     2.884972e+02    287.200000   6.95667484  276.1000000 3.042000e+02
## PSFC    9.993933e+04 100092.500000 633.43758194 97884.0000000 1.010740e+05
## U10     1.651210e-01     -0.200000   2.47512702   -6.8000000 6.500000e+00
## V10    -5.854839e-01     -1.000000   2.99113924   -5.3000000 9.200000e+00
## Q2      6.490464e-03      0.006500   0.00162157    0.0031400 1.047000e-02
## RAINC   5.625000e-02      0.000000   0.26797949    0.0000000 2.600000e+00
## RAINNC  3.997379e-01      0.000000   0.95394198    0.0000000 3.195000e+00
## SNOW    0.000000e+00      0.000000   0.00000000    0.0000000 0.000000e+00
## TSLB    2.874038e+02    285.900000   5.64450536  276.8000000 3.004000e+02
## SMOIS   2.965800e-01      0.299350   0.01521610    0.2704000 3.265000e-01
## WINDS   3.610047e+00      3.580503   1.53607374    0.5830952 9.226592e+00

View(summarydf)
```

Correlation Matrix of Sheffield

```
# Correlation analysis
correlation_matrix <- cor(Sheffield3[, -c(1, 12)], use = "complete.obs")

## Warning in cor(Sheffield3[, -c(1, 12)], use = "complete.obs"): the
standard
## deviation is zero
```

```
# Visualization of correlation matrix
require(corrplot)

## Loading required package: corrplot

## corrplot 0.92 loaded

corrplot(correlation_matrix, method = "color", type = "lower", addCoef.col =
"black", tl.col = "black", tl.srt = 45)
```

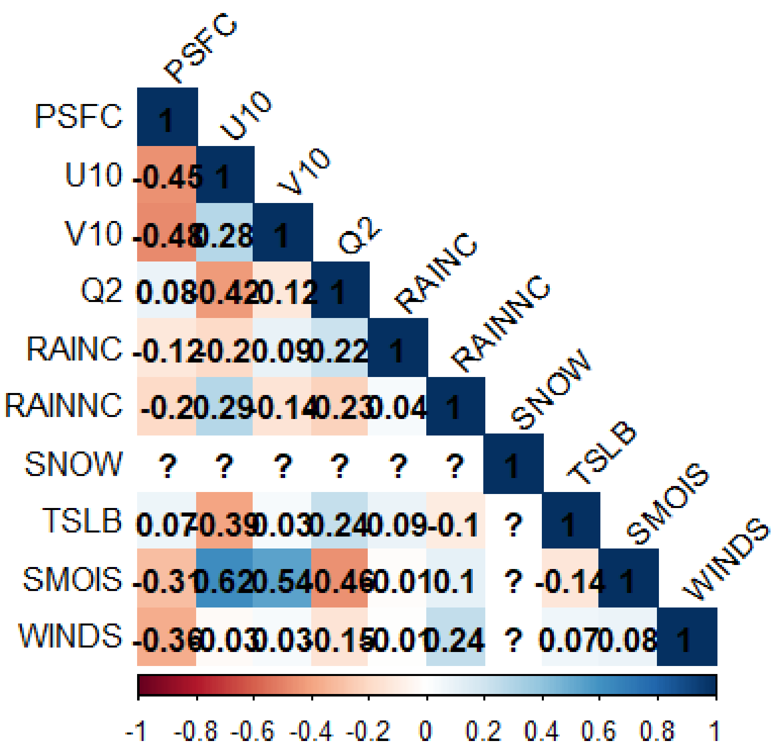

Figure 16: Sheffield Correlation Matrix

6. STATISTICAL ANALYSIS

Statistical analysis involves performing experiments and tests to confirm the hypothesis and provide answers to the research questions (Wardah et al., 2011), (Liu, 2014).

6.1 Univariate Analysis

Univariate analysis just like the name says deals with individual variables. It focuses on understanding the central tendencies, distributions, and variable variability.

Q1. Describe TSK.

```
summary(Sheffield3$TSK)

##          Min.   1st   Qu.     Median         Mean   3rd   Qu.          Max.
##   276.1   282.9   287.2   288.5   293.9   304.2
```

This shows the distribution and level of central tendency of the Surface Pressure (TSK). The min is the lowest record of TSK in the dataset. The first quartile means 25% of observations or records have a surface temperature lower than or equal to 282.9. The measure of central tendency is 288.5.

6.2 Bivariate Analysis

Bivariate analysis works with two variables. It analyzes the relationships and dependencies between the pairs of variables. The methods of analyzing bivariate relationships are dependent on the type of dataset and variables we are working on. In this dataset, the variables are numerical. Scatterplot is used to plot the graph of the variables against one another while we use correlation to check the relationship between the said variables to find if there's any significant relationship.

Q2. Is there a relationship between Surface Pressure (PSFC) and Wind Speed (U10) ?

```
# Bivariate Analysis - Correlation between Surface Pressure (PSFC) and Wind Speed (U10)
library(ggplot2)
ggplot(Sheffield3, aes(x = PSFC, y = U10)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") + # Add line of best fit
  labs(x = "Surface Pressure (PSFC)", y = "Wind Speed (U10)", title = "Scatter Plot: PSFC vs. U10") +
  theme_minimal()

## `geom_smooth()` using formula = 'y ~ x'
```

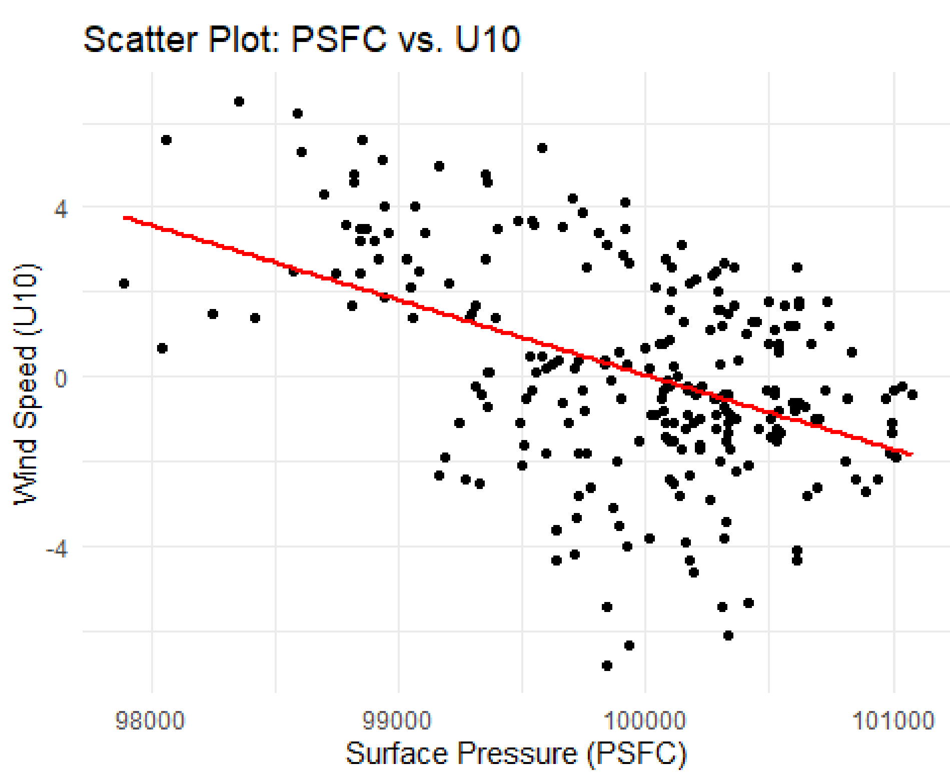

Figure 17: Scatterplot PSFC vs U10.

To better understand the result of the graph, the correlation coefficient is calculated.

```
# Correlation coefficient
cor(Sheffield3$PSFC, Sheffield3$U10, method = 'spearman')

## [1] -0.3782895
```

It is negatively correlated given the correlation coefficient between PSFC and U10 is less than 1. (< 1) Hypothesis: There is no significant correlation between Surface Pressure (PSFC) and Wind Speed (U10).

Q3. Is there a relationship between Convective rain (Accumulated precipitation) and 2- meter specific humidity?

```
ggplot(Sheffield3, aes(x = Q2, y = RAINC)) + geom_point() +
  labs(x = "Specific Humidity (Q2)", y = "Convective Rain (RAINC)", title =
"Scatter Plot: Q2 vs. RAINC") +
  theme_minimal()
```

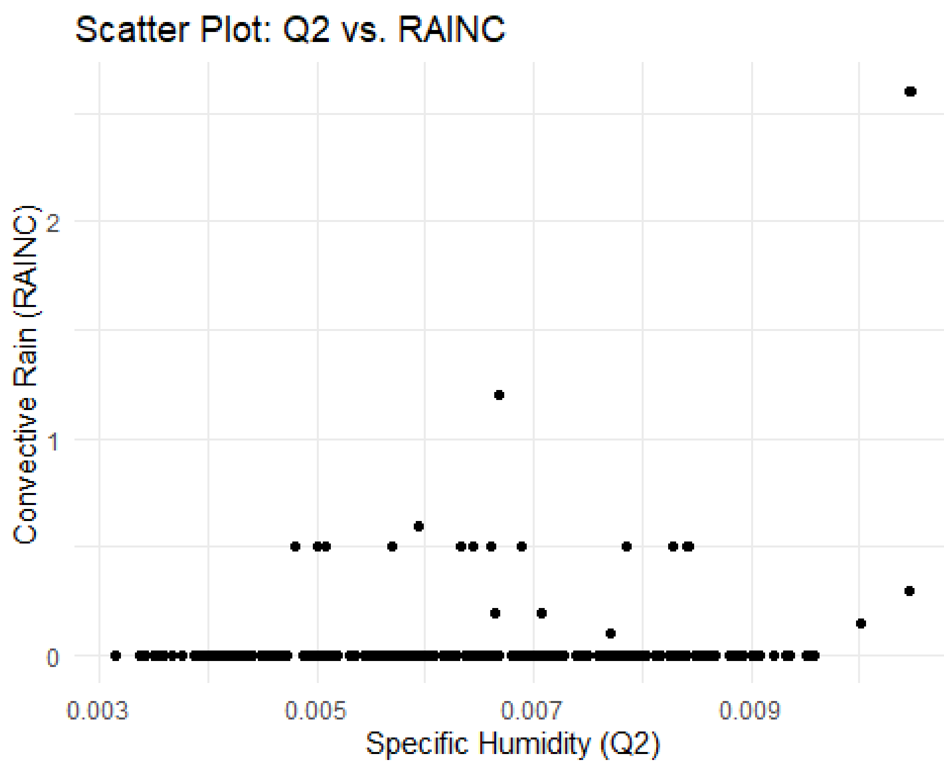

Figure 18: Scatterplot Q2 vs RAINC.

```
# Correlation coefficient
cor.test(Sheffield3$Q2, Sheffield3$RAINC, method = 'spearman')

## Warning in cor.test.default(Sheffield3$Q2, Sheffield3$RAINC, method =
## "spearman"): Cannot compute exact p-value with ties

##
## Spearman's rank correlation rho
##
## data: Sheffield3$Q2 and Sheffield3$RAINC
## S = 2199476, p-value = 0.03387
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
## 0.1347879
```

It is negatively correlated given result is less than 1(< 1) showing there is no relationship between Q2 and RAINC. Hypothesis: Accept Alternative Hypothesis as there is no correlation between Q2 and RAINC.

Q4. Is there a relationship between TSK and WINDS?

```
# Scatter plots for selected pairs of variables : TSK - WINDS
ggplot(Sheffield3, aes(x = TSK, y = WINDS)) +
  geom_point() +
  labs(x = "TSK", y = "WINDS", title = "Scatter Plot: TSK vs. WINDS")
```

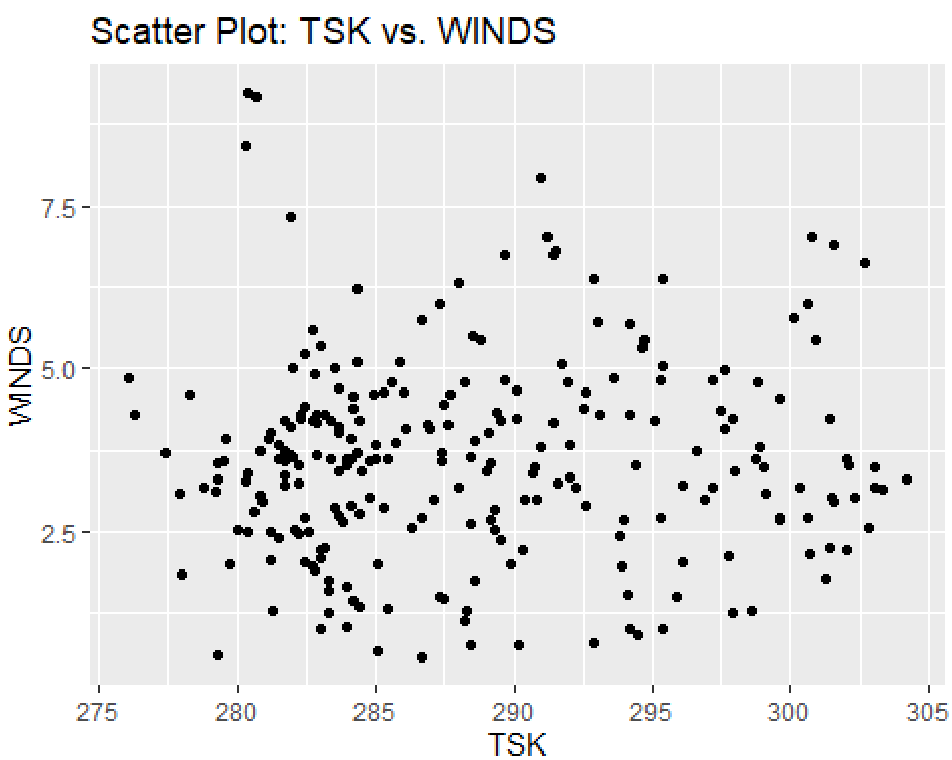

Figure 19: Scatterplot TSK vs WINDS.

```
# Correlation coefficient
cor.test(Sheffield3$WINDS, Sheffield3$TSK, method = 'spearman')
```

```
## Warning in cor.test.default(Sheffield3$WINDS, Sheffield3$TSK, method =
## "spearman"): Cannot compute exact p-value with ties

##
## Spearman's rank correlation rho
##
## data: Sheffield3$WINDS and Sheffield3$TSK
## S = 2464090, p-value = 0.6305
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
## rho
## 0.03069644
```

There is no relationship between TSK and WINDS as it is negatively correlated given the result is less than 1.

6.3. Multivariate Analysis

Multivariate analysis involves analysis variables that are more than two simultaneously to understand the complexity, patterns, and interactions between the variables.

Q5. Is there a relationship between TSLB, SMOIS and TSK?

Multiple correlation

```
#selected_data <- Sheffield[, c("TSK","RAINC", "SMOIS")]
correlation_matrix <- cor(Sheffield3[, c("TSK","RAINC", "SMOIS")])
multiple_correlation <- sqrt(det(correlation_matrix))
print(multiple_correlation)

## [1] 0.9930693
```

There is no relationship between TSLB, SMOIS and TSK as it is negatively correlated given the result is less than 1.

Alternatively

To give a more accurate result of the relationship between the variables.

```
Multivariatedata <- Sheffield3[, c("TSK", "RAINC", "SMOIS")]

library(mvnormtest)

library(dplyr)
glimpse(Multivariatedata)

##                                Rows:                                248
##                               Columns:                                3
## $ TSK    <dbl> 276.3, 276.1, 278.3, 287.7, 292.9, 291.0, 284.3, 281.9,
280.3,                                                                  2…
## $ RAINC <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,                                                                      0…
## $ SMOIS <dbl> 0.3212, 0.3207, 0.3201, 0.3196, 0.3191, 0.3186, 0.3181,
0.3177, …

mshapiro.test(t(Multivariatedata))

##
##                  Shapiro-Wilk            normality            test
##
##                   data:                                          Z
## W = 0.27879, p-value < 2.2e-16
```

The Shapiro test is used for Null Hypothesis.

Since the p value of the test is less than 0.05, we reject the Null hypothesis: there is a relationship between TSLB, SMOIS and TSK. The data is not normally distributed.

Using ANOVA

```
Anovaresult <- aov(TSK ~ RAINC + SMOIS , Multivariatedata)
summary(Anovaresult)

##              Df Sum Sq Mean Sq F value Pr(>F)
## RAINC         1     23   22.82   0.474 0.4917
## SMOIS         1    141  140.84   2.927 0.0884 .
## Residuals   245  11790   48.12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

RAINC and SMOIS have 1 degree of freedom each suggesting they are single predicator variables. RAINC has 0.4917 suggesting no statistical significance. SMOIS has 0.0884 suggesting statistical significance, although it is marginal as it falls between 0.05 and 0.1.

7. TIME-SERIES FORECASTING

Time-series forecasting involves predicting future values based on past or historical data. For a dataset to be considered a time series dataset, it must contain data collected at different points in time. In this section of the report, ARIMA model and Auto ARIMA are employed to analyze the time series data (Waheeb et al., 2019), (Sujjaviriyasup and Pitiruek, 2017), (Hasri et al., 2023).

Q6. What is the best model to use for time series forecasting?

```
Sheffield4 <- Sheffield3 #create a copy for time series
```

7.1 Generate Timestamp

Generate timestamps for time series starting 01,05,2018 at a 3-hour interval.

```
#Generate the sequence of timestamps with 3 hour intervals
starttime <- as.POSIXct("2018-05-01 00:00:00", format = "%Y-%m-%d %H:%M:%S")

#calculate end of timestamp
endtime <- starttime + (nrow(Sheffield4)-1) * 3 * 3600

#join timestamp
Sheflocsequence <- seq(starttime, endtime, by = "3 hours")

#Add the datetime column to the final dataframe
Sheffield4$DATETIME <- Sheflocsequence
```

View the new dataset with DATETIME column.

```
head(Sheffield4,5)
```

```
##       TSK   PSFC  U10   V10        Q2  RAINC  RAINNC  SNOW   TSLB   SMOIS     WINDS
## 1  276.3  98848  3.5  -2.5  0.00314      0        0      0  277.5  0.3212  4.301163
## 2  276.1  98817  4.8  -0.7  0.00430      0        0      0  276.8  0.3207  4.850773
## 3  278.3  98823  4.6  -0.1  0.00342      0        0      0  277.4  0.3201  4.601087
## 4  287.7  98787  3.6   2.9  0.00337      0        0      0  283.4  0.3196  4.622770
## 5  292.9  98694  4.3   4.7  0.00406      0        0      0  288.6  0.3191  6.370243
##                                                                          DATETIME
##                   1                 2018-05-01                           00:00:00
##                   2                 2018-05-01                           03:00:00
##                   3                 2018-05-01                           06:00:00
##                   4                 2018-05-01                           09:00:00
## 5 2018-05-01 12:00:00
```

check number of rows.

```
nrow(Sheffield4)

## [1] 248
```

check the number of columns.

```
ncol(Sheffield4)

## [1] 12
```

Create a time series dataframe. Skin temperature has been selected as a major factor of choosing Sheffield.

```
Sheffield5 <- Sheffield4[, c("TSK", "DATETIME")]
```

7.2 Plotting seasonal pattern.

```
#Seasonality plot against TSK - for Tosin
ggplot(Sheffield5, aes(x = DATETIME, y = TSK)) + geom_line(color = "blue") +
  labs(title = "TSK Seasonality", x = "Datetime", y ="TSK") + theme_minimal()
```

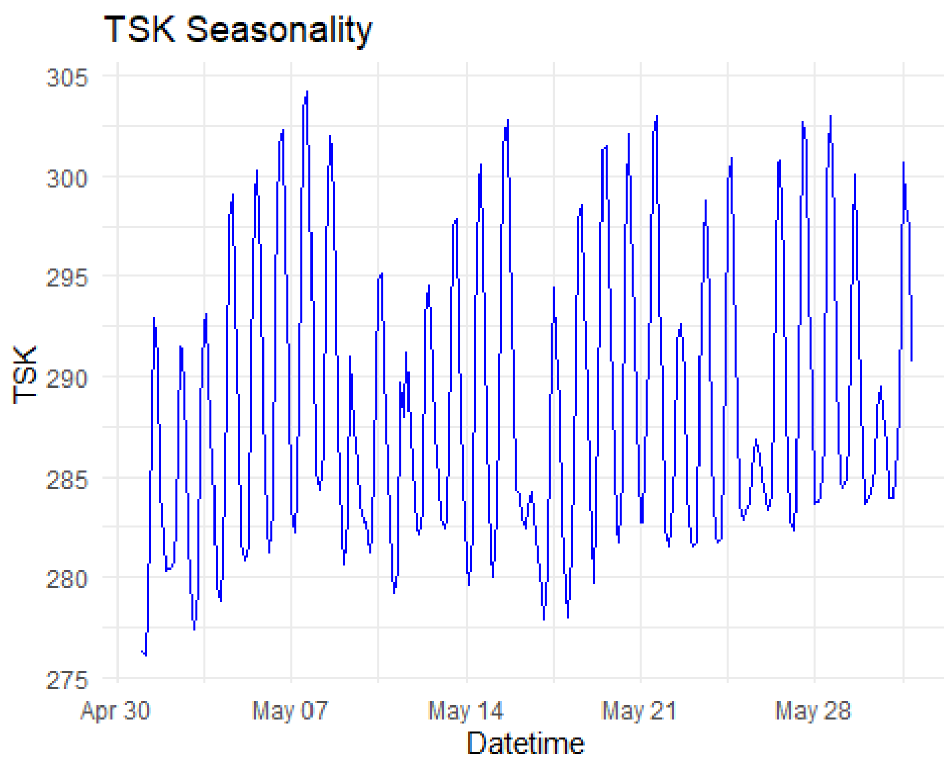

Figure 20: Plot Seasonal Pattern.

Check for dataframe structure.

```
str(Sheffield5)

## 'data.frame': 248 obs. of 2 variables:
## $ TSK : num 276 276 278 288 293 ...
## $ DATETIME: POSIXct, format: "2018-05-01 00:00:00" "2018-05-01 03:00:00"
...
```

Check for missing values.

```
sum(is.na(Sheffield5))

## [1] 0
```

Split the datetime column into data and time.

```
# Extract date and time components separately
Sheffield5$DATE <- format(Sheffield5$Datetime, "%Y-%m-%d")
Sheffield5$TIME <- format(Sheffield5$Datetime, "%H:%M:%S")
```

Remove the initial datetime column.

```
#remove initial datetime column
Sheffield5 <- subset(Sheffield5, select = c(-DATETIME))
```

check dataframe structure.

```
str(Sheffield5)

## 'data.frame': 248 obs. of 3 variables:
## $ TSK : num 276 276 278 288 293 ...
## $ DATE: chr "NULL" "NULL" "NULL" "NULL" ...
## $ TIME: chr "NULL" "NULL" "NULL" "NULL" ...
```

Create Copy of dataset

```
Timeseriesdata <- Sheffield5[,c("TSK")] #Create a subset of TSK for timeseries analysis
```

7.3 Convert to a time series data.

Given the data points were collected on a 3 -hour interval in 24 hours, we have 8 observations collected per day.

F = 8

```
n_hours <- nrow(Timeseriesdata)

Timeseriesdata <- ts(Timeseriesdata, start = 1, frequency = 8)
```

7.4 Seasonal Decompose

```
plot(decompose(Timeseriesdata))
```

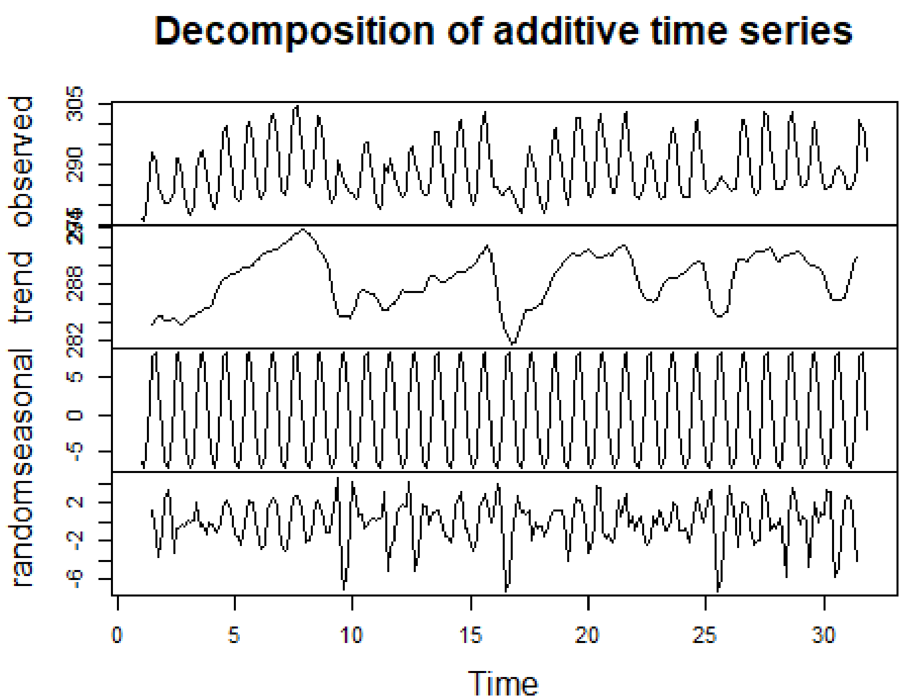

Figure 21: Seasonal Decomposition

```
time(Timeseriesdata)

##                    Time                    Series:
##         Start           =         c(1,          1)
##         End             =         c(31,         8)
##            Frequency             =               8
##   [1]  1.000  1.125  1.250  1.375  1.500  1.625  1.750  1.875  2.000
2.125
##  [11]  2.250  2.375  2.500  2.625  2.750  2.875  3.000  3.125  3.250
3.375
##  [21]  3.500  3.625  3.750  3.875  4.000  4.125  4.250  4.375  4.500
4.625
##  [31]  4.750  4.875  5.000  5.125  5.250  5.375  5.500  5.625  5.750
```

```
5.875
##  [41]  6.000  6.125  6.250  6.375  6.500  6.625  6.750  6.875  7.000
7.125
##  [51]  7.250  7.375  7.500  7.625  7.750  7.875  8.000  8.125  8.250
8.375
##  [61]  8.500  8.625  8.750  8.875  9.000  9.125  9.250  9.375  9.500
9.625
##   [71]   9.750   9.875  10.000  10.125  10.250  10.375  10.500  10.625  10.750
10.875
##   [81]  11.000  11.125  11.250  11.375  11.500  11.625  11.750  11.875  12.000
12.125
##   [91]  12.250  12.375  12.500  12.625  12.750  12.875  13.000  13.125  13.250
13.375
##  [101]  13.500  13.625  13.750  13.875  14.000  14.125  14.250  14.375  14.500
14.625
##  [111]  14.750  14.875  15.000  15.125  15.250  15.375  15.500  15.625  15.750
15.875
##  [121]  16.000  16.125  16.250  16.375  16.500  16.625  16.750  16.875  17.000
17.125
##  [131]  17.250  17.375  17.500  17.625  17.750  17.875  18.000  18.125  18.250
18.375
##  [141]  18.500  18.625  18.750  18.875  19.000  19.125  19.250  19.375  19.500
19.625
##  [151]  19.750  19.875  20.000  20.125  20.250  20.375  20.500  20.625  20.750
20.875
##  [161]  21.000  21.125  21.250  21.375  21.500  21.625  21.750  21.875  22.000
22.125
##  [171]  22.250  22.375  22.500  22.625  22.750  22.875  23.000  23.125  23.250
23.375
##  [181]  23.500  23.625  23.750  23.875  24.000  24.125  24.250  24.375  24.500
24.625
##  [191]  24.750  24.875  25.000  25.125  25.250  25.375  25.500  25.625  25.750
25.875
```

```
## [201] 26.000 26.125 26.250 26.375 26.500 26.625 26.750 26.875 27.000
27.125
## [211] 27.250 27.375 27.500 27.625 27.750 27.875 28.000 28.125 28.250
28.375
## [221] 28.500 28.625 28.750 28.875 29.000 29.125 29.250 29.375 29.500
29.625
## [231] 29.750 29.875 30.000 30.125 30.250 30.375 30.500 30.625 30.750
30.875
## [241] 31.000 31.125 31.250 31.375 31.500 31.625 31.750 31.875
```

7.4.1 Visualize

```
#Draw                                                                    plot
plot(Timeseriesdata)
```

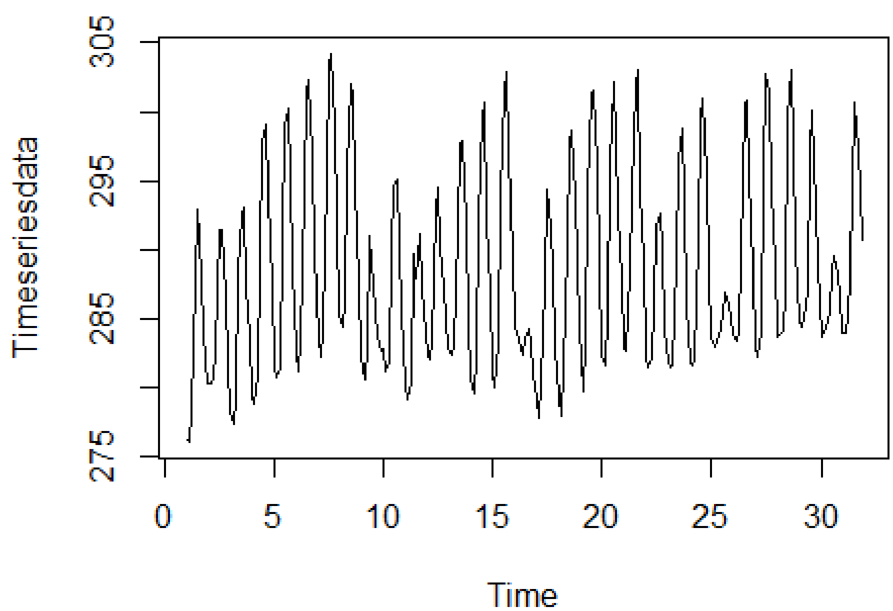

Figure 22: Visualize Timeseriesdata.

7.5 Stationarity

Check for stationarity of the time-series data.

```
library(tseries)

##
## Attaching package: 'tseries'

## The following object is masked from 'package:imputeTS':
##
##     na.remove

# Perform the ADF test directly on the 'TSK' column
adf_result2 <- adf.test(Timeseriesdata)

# Print the ADF statistics and p-value
print(paste("ADF Statistics:", adf_result2$statistic))

## [1] "ADF Statistics: -3.14991556052555"

print(paste("p-value:", adf_result2$p.value))

## [1] "p-value: 0.0967474065790744"



# Interpret the results
if (adf_result2$p.value < 0.05) {
  print("Reject NULL Hypothesis, data is likely stationary.")
} else {
  print("Accept Null Hypothesis, data is likely not stationary.")
}

## [1] "Accept Null Hypothesis, data is likely not stationary."
```

As shown above, the Null hypothesis is accepted, data is not stationary which implies there is seasonality i.e. differencing is needed to address stationarity. The p-value displayed above is greater than 0.05 meaning the data is not stationary.

7.5.1 Differencing

Differencing is technique in data science used to remove trends and seasonality observed in a time series dataset. Using Ndifs to find out how many times difference needs to be done before the data is stationary.

```
library(forecast)

##
## Attaching package: 'forecast'

## The following object is masked from 'package:DescTools':
##
##     BoxCox

ndiffs(Timeseriesdata, test = "adf")

## [1] 0
```

The output above shows Zero (0) which suggests no need for differencing, however due to the result of the p-value which suggests the data is nonstationary, differencing will be done once as it might be necessary for ARIMA Modelling. The result would be analyzed after differencing.

Check length of timeseriesdata before differencing.

```
length(Timeseriesdata)

## [1] 248
```

Autocorrelation is used to measure the linear relationship between identified lag values in a time series dataset (Hyndman and Athanasopoulos, 2018).

Auto correlation function is used for differencing.

```
tsdisplay(Timeseriesdata)
```

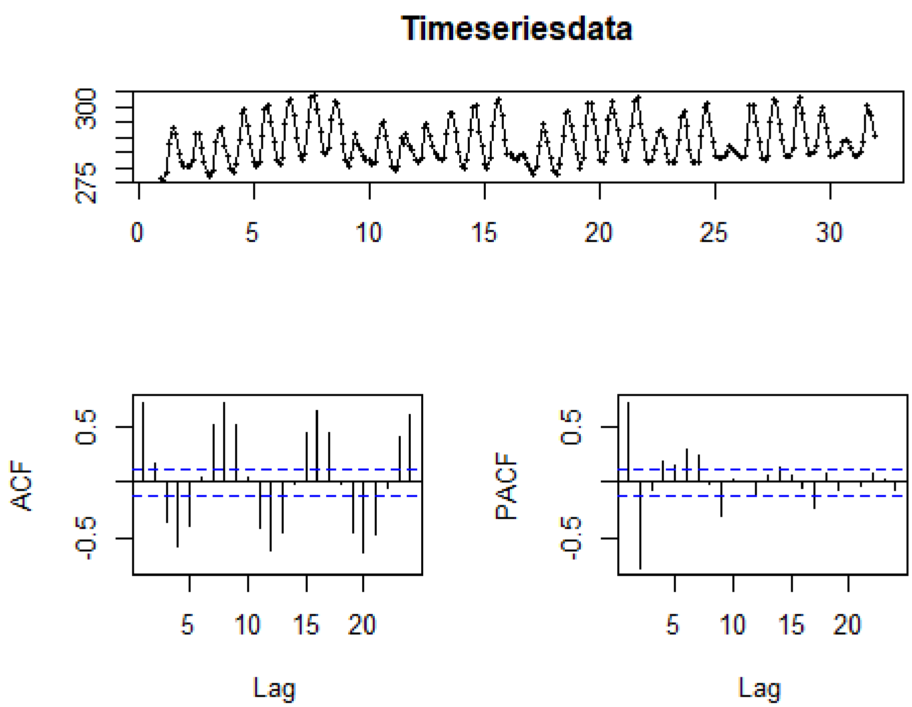

Figure 23: ACF and PACF Plot

Autocorrelation Function (ACF) Auto correlation function (ACF) is used for differencing.

ACF shows the indirect effect while PACF shows the direct effect.

```
# Calculate 1st order difference
Timeseriesdata1 <- diff(Timeseriesdata, lags = 1)
```

Order of differencing(d) = 1

7.5.3 Visualize result.

```
plot(Timeseriesdata1, main = "1st Order Difference")
```

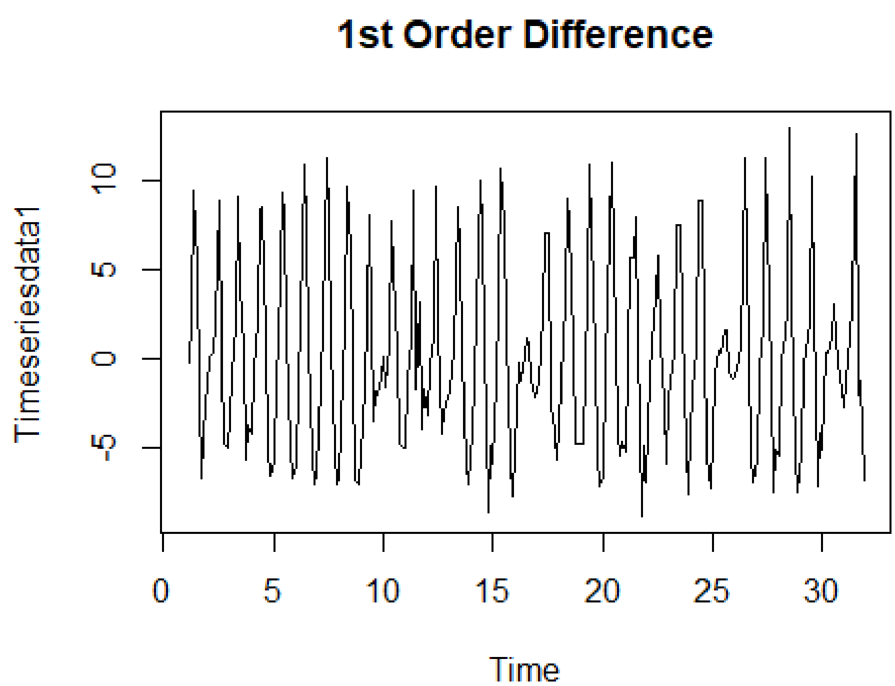

Figure 24: Order of differencing.

```
tsdisplay(Timeseriesdata1)
```

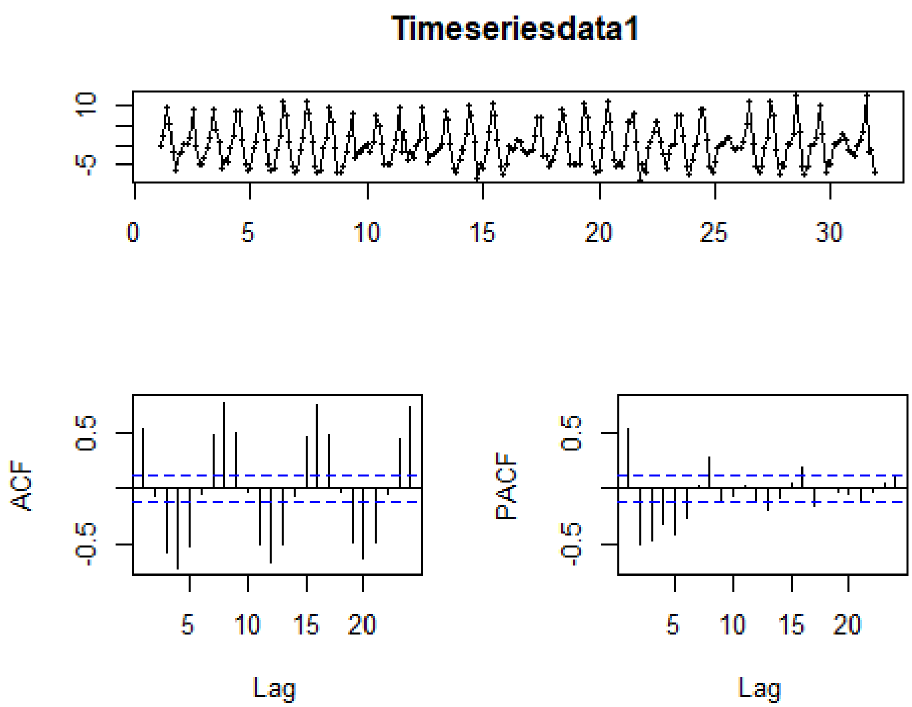

Figure 25: New ACF and PACF plot.

7.5.3 Recheck Stationarity

```
# Perform the ADF test directly on the dataset
adf.test(Timeseriesdata1)

## Warning in adf.test(Timeseriesdata1): p-value smaller than printed p-value

##
## Augmented Dickey-Fuller Test
##
## data: Timeseriesdata1
## Dickey-Fuller = -10.767, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

The p value is 0.01 which is less than 0.05 meaning differencing the time series has made the time series data become stationary.

7.7 Modelling

Now that the data is stationary, we move to splitting the dataset into train and test dataset and deploy the ARIMA and Auto ARIMA Model.

7.7.1 Develop Model

To perform ARIMA Modelling, the order of differencing (d), Autoregressive order(p) and Moving Average (q) which are gotten from the ACF and PACF plots.

Autocorellation Function (ACF)

```
acf(Timeseriesdata1)
```

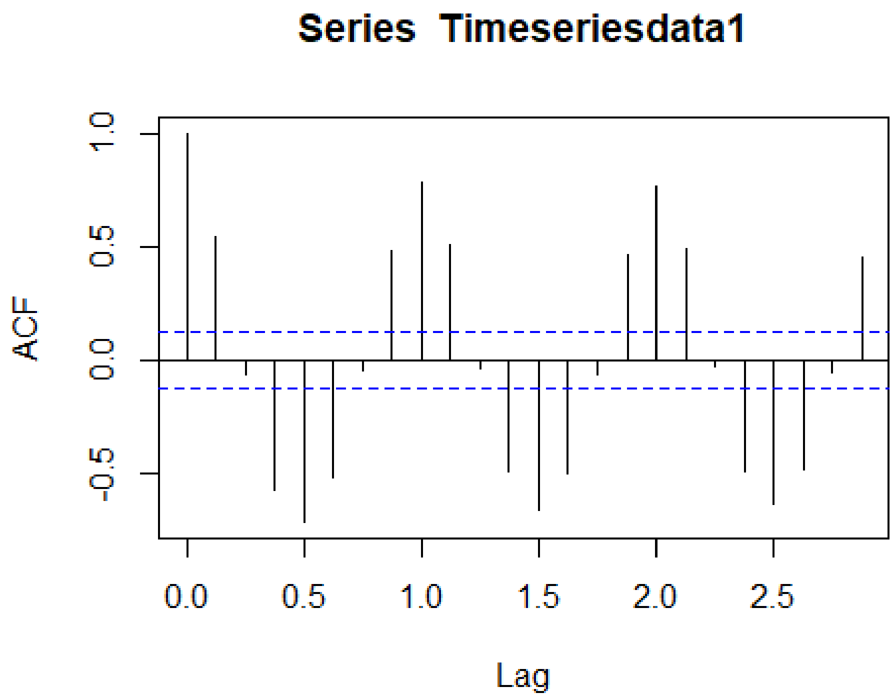

Figure 26: ACF Plot

The Autocorrelation function (ACF) is used to identify the presence of a moving average (MA) by displaying the correlation between a time series and its lag values. The ACF shows a significant autocorrelation at all the bars which shows that at this lags there's statistical and significant correlation with the values before the lag By observing the ACF plot above, significant spikes are observed with no rapid decrease. There is a consistent interval between the spikes. The lags with spikes are recorded: 0.3, 0.7, 1, 1.5, 1.7, 2.5, 2.7, 3.7, 4, 4.3, 5, 5.3, 5.7, 6. 3, 6.7, 7, 7.7.

The two consistent intervals between the observed values are 0.3 and 0.4, and the result of finding the mode is 0.3 as shown below.

```
# Given sequence of lags
lago <- c(0.3, 0.7, 1, 1.5, 1.7, 2.5, 2.7, 3.7, 4, 4.3, 5, 5.3, 5.7, 6.3,
6.7, 7, 7.7)
```

```r
# Compute the differences between consecutive lags
differencelag <- diff(lago)

# Compute the frequency of each difference
difference_freq <- table(differencelag)

# Find the mode (most frequent difference)
mode_difference <- names(difference_freq)[which.max(difference_freq)]

# Print the mode difference
print(mode_difference)

## [1] "0.3"
```

The Interval (i) is 0.3.

Partial Autocorrelation Function (PACF)

```r
pacf(Timeseriesdata1)
```

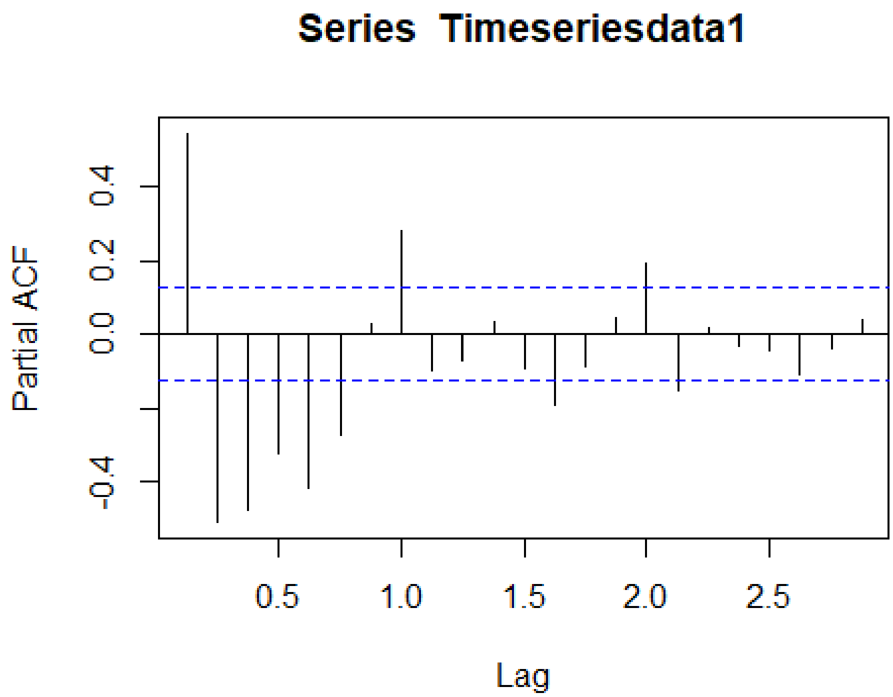

Figure 27: PACF Plot

PACF- bars that cut across are statistically significant while the once that didn't cut across the dotted line aren't. So that means those that are statistically significant have significant correlation with the values before lagging even after removing effects of intervening correlations. Tt is observed there are significant values for p as they are above the confidence interval line. From the plot above, 10 significant spikes are observed at different lags but lag 11 and beyond fall within the confidence interval. The last lag before they fall beyond the confidence interval is lag 2.2.

Our Autoregressive order(p) lag 2.2.

Check Timeseries Data.

```
head(Timeseriesdata1)
```

```
## Time Series:
## Start = c(1, 2)
## End = c(1, 7)
## Frequency = 8
## [1] -0.2  2.2  9.4  5.2 -1.9 -6.7
```

Frequency is 8 meaning there are 8 observations per unit time (day).

```
length(Timeseriesdata1)

## [1] 247
```

Length of Timeseries data after differencing is 247.

7.7.2 Split the dataset.

Unlike Machine learning where the split ratio is mentioned and data is split randomly, time series split is not random as it has to be specified. Tt takes the first half of the data based on the margin specified as train and the second half based on the margin left.

Training and Testing Dataset Specify the train and test data based on the number of rows to be saved into each. To split the dataset, the first 200 rows are saved into a test dataframe, and the remaining 47 rows are saved into a test dataframe.

```
# Convert the time series data into a time series object
Timeseriesdata_ts <- ts(Timeseriesdata1, start = 1, frequency = 1)
# Define the number of rows for training and testing
train_rows <- 200
test_rows <- 47
# Split the time series object into training and test sets
```

```
Tstraindata <- head(Timeseriesdata_ts, train_rows)
Tstestdata <- tail(Timeseriesdata_ts, test_rows)
```

check training data length.

```
# Check the dimensions of the training and test sets
print(length(Tstraindata))

## [1] 200
```

check testing data length.

```
print(length(Tstestdata))

## [1] 47
```

The code above is initiated to save the first 200 rows into train and the remaining 47 rows as test dataset.

7.7.3 AUTO-ARIMA

```
library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ lubridate 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ tidyr     1.3.1
## ✔ readr     2.1.5
## ── Conflicts ──────────────────────────────────────── tidyverse_conflicts() ──
```

```
## ✖ dplyr::filter()     masks stats::filter()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ Hmisc::src()        masks dplyr::src()
## ✖ Hmisc::summarize()  masks dplyr::summarize()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(forecast)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

Fit the model.

```
AAmodel <- auto.arima(Tstraindata)
AAmodel
```

```
## Series: Tstraindata
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4019  -0.9762  -1.2474  0.6450
## s.e.  0.0165   0.0194   0.0625  0.1236
##
```

```
## sigma^2 = 6.316: log likelihood = -467.85
## AIC=945.69 AICc=946 BIC=962.18
```

AIC = 945.69

7.7.4 ARIMA

To use the manual Arima model, we need the values of p, d and q. p = Autoregressive order d = Order of differencing needed to achieve stationarity q = Moving Average (MA) i = interval (consistent interval observed)

The values of p and d were observed and registered from the order of differencing and PACF plot. However, the ACF plot shows significant lags with no rapid decrease in autocorrelation value suggesting persistent seasonality in the time series data and does not decay easily. To solve for the value of q, the consistent interval(I) value was gotten.

The code below is used to calculate the value of q considering there is a consistent interval observed.

```
# Determine the lag order (q) based on the consistent interval observed in the ACF plot
interval <- 0.3 #Consister interval is observed every 0.3 lags

#To find the moving average based on the reciprocal of the interval
q <- round(1 / interval) - 1
q

## [1] 2
```

Moving Average(q) = 2

Fit the model.

```
library(stats)

p <- 2.2 #observed from the PACF graph
d <- 1 #order of differencing
q <- 2 #calculated from the consistent intervals observed

ARIMAmodel <- arima(Tstraindata, order = c(p, d, q))
summary(ARIMAmodel)

##
## Call:
## arima(x = Tstraindata, order = c(p, d, q))
##
## Coefficients:
## ar1 ar2 ma1 ma2
## -0.1548 -0.7725 0.3960 0.9328
## s.e. 0.0732 0.0646 0.0357 0.0608
##
## sigma^2 estimated as 18.07: log likelihood = -571.18, aic = 1152.37
##
## Training set error measures:
## ME RMSE MAE MPE MAPE MASE
## Training set -0.00265982 4.240576 3.452907 -23.64096 291.4269 0.925467
## ACF1
## Training set 0.02271843
```

AIC = 1152.37

7.7.5 BEST MODEL - AUTO ARIMA

```
AICArima = 1152.37
AICAuto = 945.69
```

```
if (AICArima < AICAuto){
  print("Manua ARIMA Model is the best model")
} else
  print("Auto ARIMA Model is the best model")

## [1] "Auto ARIMA Model is the best model"
```

Based on the AIC values observed, the manual Auto ARIMA model is the best model to use as it has an AIC value of 945.69 compared to the ARIMA model that has an AIC value of AIC 1152.37.

7.7.6 Forecast

```
AAforecast <- forecast(AAmodel, h = length(Tstestdata))
AAforecast

##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 201     -1.7616318 -4.982422 1.459159 -6.687406 3.164142
## 202     -0.1869872 -3.445997 3.072023 -5.171213 4.797238
## 203      1.4574975 -1.822329 4.737324 -3.558564 6.473559
## 204      2.2258495 -1.203918 5.655617 -3.019527 7.471226
## 205      1.6977434 -1.887987 5.283474 -3.786158 7.181645
## 206      0.2073345 -3.411362 3.826031 -5.326985 5.741654
## 207     -1.3666063 -5.001041 2.267829 -6.924995 4.191782
## 208     -2.1182881 -5.874677 1.638100 -7.863189 3.626612
## 209     -1.6356739 -5.523193 2.251845 -7.581121 4.309773
## 210     -0.2253146 -4.141967 3.691338 -6.215317 5.764688
## 211      1.2808093 -2.648000 5.209619 -4.727786 7.289405
## 212      2.0155577 -2.014666 6.045782 -4.148138 8.179254
## 213      1.5754043 -2.567129 5.717938 -4.760054 7.910863
## 214      0.2411021 -3.927574 4.409779 -6.134339 6.616543
## 215     -1.1998432 -5.378029 2.978343 -7.589828 5.190141
```

```
##   216              -1.9174595  -6.181230  2.346311  -8.438333  4.603414
##   217              -1.5169131  -5.878098  2.844272  -8.186770  5.152944
##   218              -0.2548610  -4.639749  4.130027  -6.960968  6.451246
##   219               1.1234570  -3.268924  5.515838  -5.594110  7.841024
##   220               1.8238019  -2.641556  6.289160  -5.005374  8.652977
##   221               1.4601765  -3.090451  6.010804  -5.499407  8.419760
##   222               0.2667451  -4.305529  4.839020  -6.725945  7.259436
##   223              -1.0514121  -5.629614  3.526790  -8.053168  5.950344
##   224              -1.7344002  -6.375305  2.906504  -8.832051  5.363251
##   225              -1.4051687  -6.121223  3.310886  -8.617752  5.807414
##   226              -0.2768984  -5.012827  4.459031  -7.519877  6.966080
##   227               0.9834812  -3.757143  5.724106  -6.266678  8.233640
##   228               1.6490767  -3.145734  6.443888  -5.683954  8.982107
##   229               1.3518622  -3.509529  6.213254  -6.082994  8.786719
##   230               0.2854558  -4.594254  5.165165  -7.177416  7.748327
##   231              -0.9194483  -5.802875  3.963978  -8.388005  6.549108
##   232              -1.5676601  -6.498124  3.362803  -9.108153  5.972833
##   233              -1.3002279  -6.289923  3.689468  -8.931309  6.330853
##   234              -0.2925437  -5.299172  4.714085  -7.949521  7.364434
##   235               0.8591080  -4.150457  5.868673  -6.802360  8.520576
##   236               1.4899857  -3.560553  6.540524  -6.234147  9.214118
##   237               1.2502354  -3.853172  6.353643  -6.554753  9.055224
##   238               0.2982803  -4.820814  5.417375  -7.530699  8.127260
##   239              -0.8022648  -5.923670  4.319140  -8.634778  7.030248
##   240              -1.4158950  -6.573092  3.741302  -9.303147  6.471357
##   241              -1.2018531  -6.406365  4.002659  -9.161467  6.757761
##   242              -0.3027762  -5.521845  4.916293  -8.284653  7.679101
##   243               0.7487328  -4.472144  5.969610  -7.235910  8.733375
##   244               1.3452352  -3.906977  6.597448  -6.687331  9.377801
##   245               1.1550487  -4.139600  6.449698  -6.942418  9.252515
##   246               0.3061348  -5.002039  5.614309  -7.812017  8.424287
## 247     -0.6983354 -6.007915 4.611244 -8.818637 7.421966
```

7.7.7 Evaluate

```
# Evaluate the model
AAaccuracy <- accuracy(AAforecast, Tstestdata)
AAaccuracy

##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.03118381 2.487937 1.857283 -18.24363 132.6072 0.4977991
## Test set     0.10115631 4.533665 3.428470       Inf      Inf 0.9189174
##                    ACF1 Theil's U
## Training set 0.03879467        NA
## Test set     0.34955764       NaN

cat("The accuracy of the Auto ARIMA Model is:", AAaccuracy[,'RMSE'])

The accuracy of the Auto ARIMA Model is: 2.487937 4.533665.
```

7.7.8 check residual

```
checkresiduals(AAforecast)
```

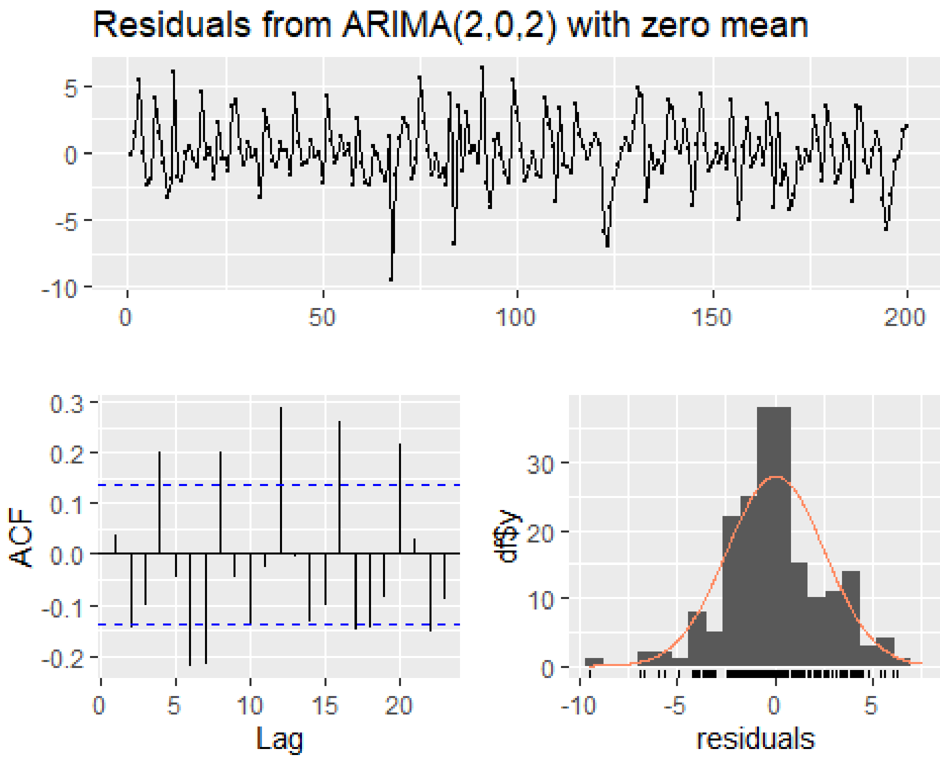

Figure 28: Residual Plot

```
##
##                                         Ljung-Box                    test
##
##     data:       Residuals    from    ARIMA(2,0,2)    with    zero    mean
##     Q*     =     48.445,     df     =     6,     p-value     =     9.628e-09
##
## Model df: 4.   Total lags used: 10
```

There is significant autocorrelation present in the residuals of the model.

8. MACHINE LEARNING

Machine learning regression models are utilized in this section of the report to predict continuous variables such as surface temperature. Models such as linear regression model, random forest model, and Support Vector regression model are trained using the training dataset and deployed using the testing dataset for their predictive performance. After the models are deployed, the accuracy level for each model is checked and compares against one another to select the best model with the least error (Sulaimon et al., n.d.), (Scher and Messori, 2018).

```
library(caTools)
set.seed(123) #set seeed for reproducability
```

Create a copy of sheffield dataset for ML

```
Sheffieldml <- Sheffield4[,c("TSK", "PSFC", "Q2", "TSLB","SMOIS", "WINDS")]
#create a copy of the dataset for ML

str(Sheffieldml)

## 'data.frame': 248 obs. of 6 variables:
## $ TSK : num 276 276 278 288 293 ...
## $ PSFC : num 98848 98817 98823 98787 98694 ...
## $ Q2 : num 0.00314 0.0043 0.00342 0.00337 0.00406 ...
## $ TSLB : num 278 277 277 283 289 ...
## $ SMOIS: num 0.321 0.321 0.32 0.32 0.319 ...
## $ WINDS: num 4.3 4.85 4.6 4.62 6.37 ...
```

8.1 Split the dataframe.

Split the dataframe using 80/30 split

```
splitdata <- sample.split(Sheffieldml, SplitRatio = 0.8)
```

Initiate training set and testing set

```
mltrain                       <-                       Sheffieldml[splitdata,]
mltest <- Sheffieldml[!splitdata,]
```

check for number of rows in each.

```
nrow(mltrain)

## [1] 166

nrow(mltest)

## [1] 82
```

The data has been split using 80/20 split. 80% of the data goes to the training model and 20% goes to test model.

Initiate X train, x test, y train, y test

```
xtrain   <-   mltrain[,c("TSK",   "PSFC",   "Q2",   "TSLB","SMOIS",   "WINDS")]
ytrain                             <-                              mltrain$TSK

xtest   <-   mltest[,c(   "TSK",   "PSFC",   "Q2",   "TSLB","SMOIS",   "WINDS")]
ytest <- mltest$TSK
```

8.2 Linear Regression

Initiate Regression Model

```
Regressionmodel              <-              lm(TSK~.,              mltrain)
summary(Regressionmodel)
```

```
##
## Call:
## lm(formula = TSK ~ ., data = mltrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6986 -1.3342 -0.3497  0.9749  6.7507
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.305e+01  2.968e+01  -2.461   0.0149 *
## PSFC         2.178e-04  2.813e-04   0.774   0.4398
## Q2          -2.638e+02  1.119e+02  -2.358   0.0196 *
## TSLB         1.192e+00  2.957e-02  40.322   <2e-16 ***
## SMOIS       -1.171e+00  1.223e+01  -0.096   0.9239
## WINDS       -2.100e-01  1.130e-01  -1.858   0.0650 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.031 on 160 degrees of freedom
## Multiple R-squared:  0.9172, Adjusted R-squared:  0.9146
## F-statistic: 354.5 on 5 and 160 DF,  p-value: < 2.2e-16
```

8.2.1 check for multiple regression

```
summary(Regressionmodel)$r.squared

## [1] 0.9172014
```

This indicates a strong relationship between the dependent and independent variables.

8.2.2 Correlation test

```
cor(Sheffieldml)

##                   TSK          PSFC            Q2          TSLB         SMOIS
WINDS
##  TSK        1.00000000    0.07980470    0.18236615    0.95524014   -0.10902606
0.02571423
##  PSFC      0.07980470    1.00000000    0.08188328    0.07036240   -0.30719298  -
0.36197017
##  Q2         0.18236615    0.08188328    1.00000000    0.23503894   -0.45938824  -
0.14899028
##  TSLB       0.95524014    0.07036240    0.23503894    1.00000000   -0.13793635
0.06512604
##   SMOIS   -0.10902606   -0.30719298   -0.45938824   -0.13793635     1.00000000
0.08044007
##   WINDS     0.02571423   -0.36197017   -0.14899028    0.06512604    0.08044007
1.00000000
```

8.2.3 Multiple correlation

```
par(mfrow                                    =                                    c(2,2))
plot(Regressionmodel)
```

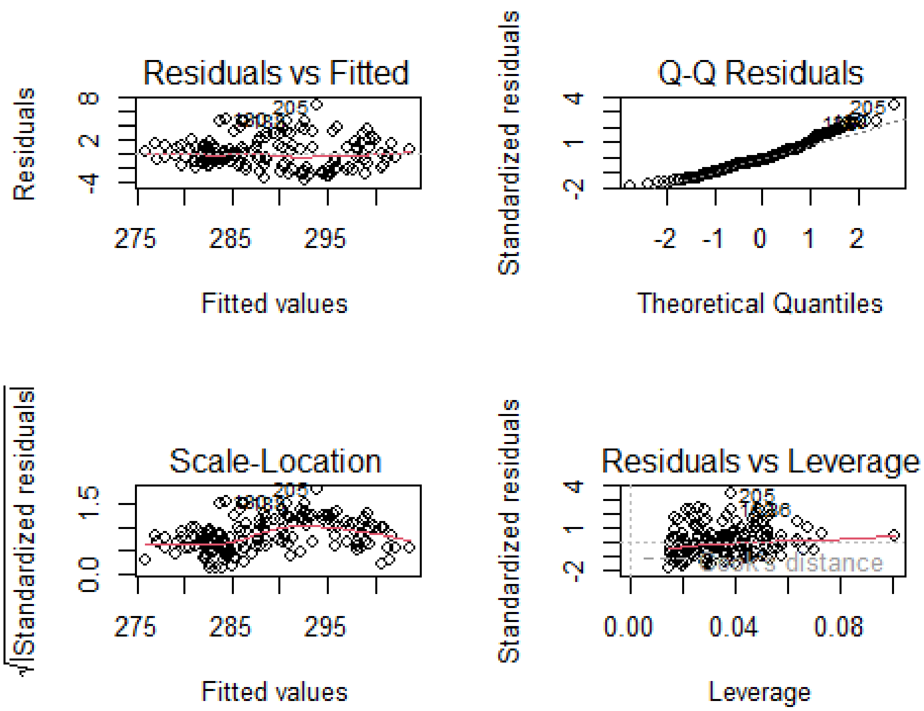

Figure 29: Visualize Linear Regression Multiple Correlation

8.2.4 Linearity

```r
pairs(Sheffieldml)
```

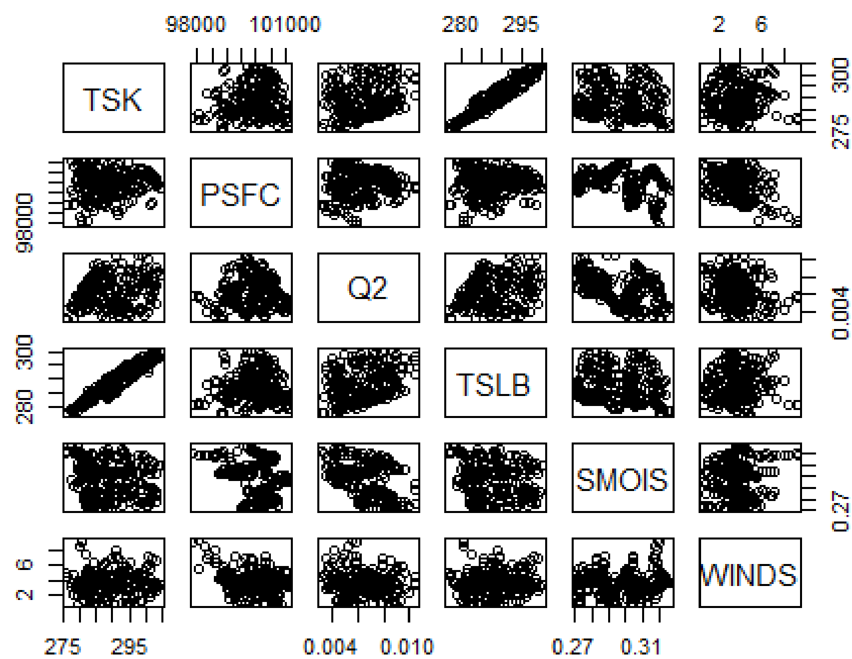

Figure 30: Plot Linearity

```r
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:imputeTS':
##
##     na.locf

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
raintest(Regressionmodel)

##
##                                              Rainbow                       test
##
##                  data:                                            Regressionmodel
## Rain = 1.2018, df1 = 83, df2 = 77, p-value = 0.2076
```

P-value of 0.2076 is greater than 0.05 meaning it is linear.

8.2.5 Heterocidacity

```
library(car)

## Loading required package: carData

##
## Attaching package: 'car'

##    The    following    object    is    masked    from    'package:purrr':
##
##     some

##    The    following    object    is    masked    from    'package:DescTools':
##
##     Recode

##    The    following    object    is    masked    from    'package:dplyr':
##
##     recode

ncvTest(Regressionmodel)

##           Non-constant           Variance           Score            Test
##           Variance          formula:          ~          fitted.values
## Chisquare = 9.54898, Df = 1, p = 0.0020006
```

P value is 0.0020006 which is less than 0.05 meaning there is heterocidacity.

8.2.6 Normality

```
shapiro.test(Regressionmodel$residuals) #p>0.05 so distribution is normal

##
##                           Shapiro-Wilk            normality            test
##
##                data:                              Regressionmodel$residuals
## W = 0.96311, p-value = 0.0002169
```

P-value is 0.0002169 which is less than 0.05, the distribution is not normal.

8.2.7 Autocorrelation of errors

```
#Durbin-Watson                                                             test
library(lmtest)
#null   hypo:   there   is   no   autocorrelation   (errors   are   independent)
#alternate  hypothesis:  there  is  autocorrelation  (errors  are  dependent)
dwtest(Regressionmodel) #since p>0.05 - there is no autocorrelation

##
##                                          Durbin-Watson                   test
##
##                   data:                                       Regressionmodel
##         DW         =         1.4015,         p-value         =       1.399e-05
## alternative hypothesis: true autocorrelation is greater than 0
```

Since p value is 1.399e-05 which is less than 0.05, there is autocorrelation.

8.2.8 Multicolinearity

```
vif(Regressionmodel) #vif>10 strong multicollinearity

##          PSFC              Q2           TSLB          SMOIS          WINDS
## 1.253120 1.377591 1.130288 1.389187 1.183866
```

No multicolinearitty found since all the VIF values are below 10.

8.2.9 LPredict

```
lpredict <- predict(Regressionmodel, newdata = xtest)
lpredict

##        4        5       10       11       16       17       22       23
## 284.1118 289.7428 278.7126 278.8335 283.7699 278.8446 290.2384 289.4151
##       28       29       34       35       40       41       46       47
## 285.4094 294.4590 281.3158 281.4673 291.4286 286.7051 302.3704 299.5685
##       52       53       58       59       64       65       70       71
## 291.3484 300.1064 284.4920 285.1406 290.3267 283.5078 286.9395 284.6556
##       76       77       82       83       88       89       94       95
## 287.2290 293.3274 280.2763 280.3195 286.1517 283.3335 292.7653 289.5917
##      100      101      106      107      112      113      118      119
## 288.7085 295.8274 280.9474 281.5019 289.3737 282.6750 303.2914 300.1031
##      124      125      130      131      136      137      142      143
## 283.0417 283.4276 279.5935 279.8698 286.3474 281.5780 298.5504 297.0031
##      148      149      154      155      160      161      166      167
## 291.1884 299.5635 282.3976 283.6037 290.5162 284.3206 303.1224 297.3860
##      172      173      178      179      184      185      190      191
## 284.6881 289.5990 282.3415 282.1274 289.1739 282.9099 300.5782 297.7331
##      196      197      202      203      208      209      214      215
## 283.6600 284.5568 284.1607 283.7614 291.2929 283.9759 301.7990 296.6648
##      220      221      226      227      232      233      238      239
## 285.7476 295.5689 285.0379 284.7841 290.2486 284.7613 289.4042 289.0376
##                                  244                                 245
## 287.1145 295.4160
```

8.3 Support Vector Regression

```
library(caTools)
library(e1071)

##
## Attaching package: 'e1071'

## The following object is masked from 'package:Hmisc':
##
## impute

set.seed(123)
```

8.3.1 Feature Scaling

```
svrtrainset = scale(xtrain)
svrtestset = scale(xtest)
```

8.3.2 Train Model

```
SVRmodel <- svm( TSK ~ ., data = svrtrainset, type <- 'eps-regression',
kernel <- 'linear')
summary(SVRmodel)

##
## Call:
## svm(formula = TSK ~ ., data = svrtrainset, type <- "eps-regression",
## kernel <- "linear")
##
##
## Parameters:
## SVM-Type: eps-regression
## SVM-Kernel: linear
## cost: 1
```

```
##                                          gamma:              0.2
##                                       epsilon:              0.1
##
##
## Number of Support Vectors:  119
```

8.3.3 SVRPredict

```
svrpredict <- predict(SVRmodel, newdata = svrtestset)
svrpredict

##           4           5          10          11          16          17
## -0.58975411  0.23066261 -1.38596042 -1.36896780 -0.65220806 -1.37190108
##          22          23          28          29          34          35
##  0.28566875  0.16069329 -0.44266780  0.87748496 -1.04381702 -1.01886634
##          40          41          46          47          52          53
##  0.42094294 -0.26463543  2.03996320  1.62602943  0.40623033  1.69160136
##          58          59          64          65          70          71
## -0.59222268 -0.49559867  0.27715342 -0.71239700 -0.20090738 -0.54049515
##          76          77          82          83          88          89
## -0.15903232  0.73771881 -1.18470593 -1.17214504 -0.32368483 -0.73727998
##          94          95         100         101         106         107
##  0.64591538  0.18172517  0.05501578  1.09965023 -1.07502091 -0.99787195
##         112         113         118         119         124         125
##  0.14796442 -0.85242637  2.17903835  1.70073251 -0.79502474 -0.73763854
##         130         131         136         137         142         143
## -1.29530361 -1.25459591 -0.30577013 -1.01183100  1.47997935  1.25223260
##         148         149         154         155         160         161
##  0.39391377  1.61924629 -0.90064661 -0.72980724  0.28245168 -0.62791074
##         166         167         172         173         178         179
##  2.13282022  1.29932429 -0.56124320  0.15657535 -0.91003068 -0.94056814
##         184         185         190         191         196         197
##  0.08619857 -0.83293740  1.75978084  1.34213146 -0.72443473 -0.59007209
```

```
##         202         203         208         209         214         215
## -0.64362719 -0.70216481  0.39963705 -0.67196244  1.93389292  1.18113406
##         220         221         226         227         232         233
## -0.42600874  1.00768420 -0.53200062 -0.56905369  0.22952266 -0.56598410
##                238                  239                  244          245
##  0.10525202  0.05084291 -0.23518170  0.97490371
```

8.4 Random Forest Model

```
# Load the required packages
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

8.4.1 Train Model

```
RFModel <- randomForest(y = ytrain, x =xtrain)
RFModel
```

```
##
##                                                          Call:
##          randomForest(x     =     xtrain,     y     =     ytrain)
##                                  Type  of  random  forest:  regression
##                                            Number  of  trees:  500
##     No.    of    variables    tried    at    each    split:    2
##
##                               Mean  of  squared  residuals:  1.016488
##                    % Var explained: 97.88
```

8.4.2 RFPredict

```
RFpredict      <-      predict(RFModel,      newdata      =      xtest)
RFpredict

##        4        5       10       11       16       17       22       23
## 284.1965 291.7665 280.6324 280.6104 282.2366 278.4928 291.3817 287.3772
##       28       29       34       35       40       41       46       47
## 287.5559 295.8536 281.1457 281.1736 289.2128 284.4394 300.9580 298.1218
##       52       53       58       59       64       65       70       71
## 294.4391 301.4054 284.8595 286.2991 288.5297 282.4268 285.1479 283.6886
##       76       77       82       83       88       89       94       95
## 288.7087 293.9461 280.1203 280.3520 285.1924 282.6157 291.0054 288.3267
##      100      101      106      107      112      113      118      119
## 290.7155 296.6311 280.4186 281.5461 288.0771 281.6818 301.0488 298.7264
##      124      125      130      131      136      137      142      143
## 282.5690 283.3149 279.2831 279.9545 283.8854 280.4256 298.1328 294.6946
##      148      149      154      155      160      161      166      167
## 293.6919 300.0433 282.0879 284.1910 288.8657 283.4294 301.7390 295.0287
##      172      173      178      179      184      185      190      191
## 285.4901 291.6601 281.8069 281.7857 287.6761 282.0595 300.1362 295.5139
##      196      197      202      203      208      209      214      215
## 283.8904 285.0964 283.7495 283.7875 289.2937 283.3623 300.9795 294.9476
```

```
##      220      221      226      227      232      233      238      239
## 286.5721 298.1852 284.3976 284.7490 289.5510 284.1118 289.4344 288.4750
##                                  244                                  245
## 287.5575 297.8277
```

8.5 Model Evaluation

8.5.1 Line.Reg RMSE

```
RMSEReg <- sqrt(mean((ytest - lpredict)^2))
cat("RMSE for Linear Regression:", RMSEReg)

## RMSE for Linear Regression: 2.057116
```

8.5.2 SVR RMSE

```
RMSEsvr <- sqrt(mean((ytest - svrpredict)^2))
cat("RMSE for SVR:", RMSEsvr)

## RMSE for SVR: 288.2767
```

8.5.3 RF RMSE

```
RMSErf <- sqrt(mean((ytest - RFpredict)^2))
cat("RMSE for RF:", RMSErf)

## RMSE for RF: 0.9797684
```

8.5.4 Visualize

```
# Adjust the width and height as needed
options(repr.plot.width = 6, repr.plot.height = 8)
```

```
# Define the RMSE values
models <- c("Linear Regression", "SVR", "Random Forest")
RMSE <- c(RMSEReg, RMSEsvr, RMSErf)

# Create a bar plot
barplot(RMSE, names.arg = models, col = "skyblue", main = "RMSE Comparison",
xlab = "Regression Models", ylab = "RMSE")
```

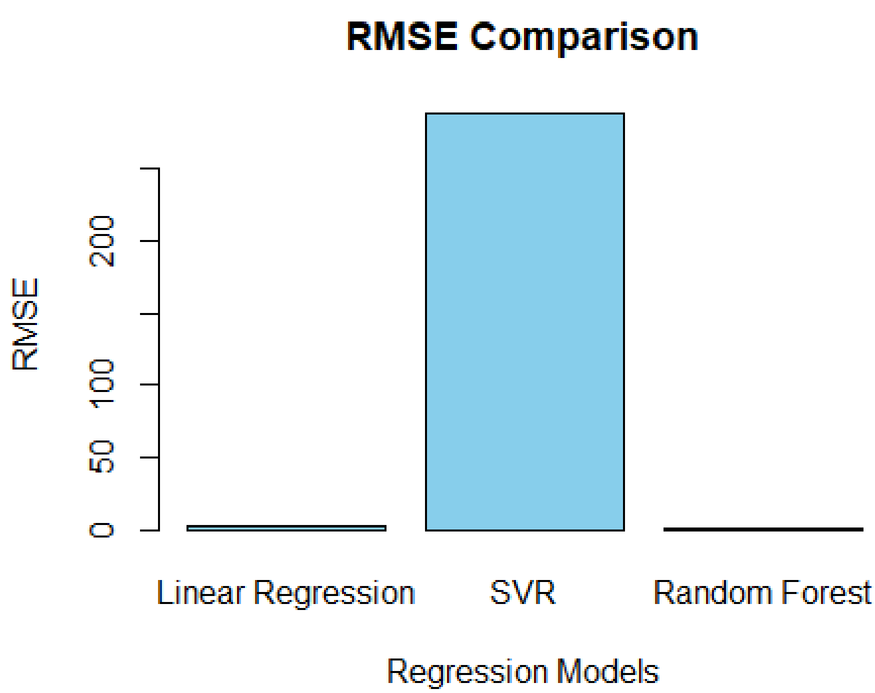

Figure 31: RMSE Comparison

```
# Determine the model with the least error
min_error_model <- min(RMSEReg, RMSEsvr, RMSErf)

if (min_error_model == RMSEReg) {
  print("Linear Regression has the least error.")
} else if (min_error_model == RMSEsvr) {
  print("SVR has the least error.")
```

```
} else {
    print("Random Forest has the least error.")
}

## [1] "Random Forest has the least error."
```

The preferred model to be used for prediction is the Random Forest model as it has the least number of errors.

9.0 DISCUSSION

In this section, the findings are interpreted from the analysis. It points out the patterns, trends and relationships between the variables therefore answering our research questions and insights into the hypothesis.

Discussion of Findings:

1. Reject Null Hypothesis 1: There is no significant correlation between Surface Pressure (PSFC) and Wind Speed (U10). The analysis rejects the null hypothesis, indicating a significant correlation between PSFC and U10. This suggests that changes in surface pressure are associated with variations in wind speed.

2. Reject Null Hypothesis 2: There is no correlation between Q2 and RAINC. Similarly, the null hypothesis is rejected, indicating a correlation between Q2 and RAINC. This implies that changes in specific humidity are related to precipitation.

3. Reject Null Hypothesis 3: There is no relationship between TSK and WINDS as it is negatively correlated given the result is less than 1. This suggests that higher surface temperatures are associated with lower wind speeds.

4. Reject Null Hypothesis 4: There is no correlation between TSLB, SMOIS, and TSK, suggesting that higher soil temperatures and moisture levels are associated with lower surface temperatures. RAINC and SMOIS each have 1 degree of freedom, indicating they are single predictor variables. RAINC's p-value of 0.4917 suggests no statistical significance, while SMOIS's p-value of 0.0884 indicates statistical significance, albeit marginally, falling between 0.05 and 0.1, suggesting a moderate level of significance.

5. The manual Auto ARIMA model is the best model to use as it has an AIC value of 945.69 compared to the ARIMA model that has an AIC value of 1152.37.

6. Among the three models used, linear regression, random forest, and support vector regression (SVR), the preferred model for prediction is the Random Forest model as it has the least number of errors

1.0 CONCLUSION

The report examines the merits of having an accurate weather forecast, with a focus on the meteorological parameters collected in the city of Sheffield. By adopting the CRISP-DM methodology, the study critically studies the dataset, elucidates the relationship existing between the variables, and makes use of the historical data provided to make time series forecasts and machine learning predictions.

The analysis starts off with a dataset comprised of different longitudinal and latitudinal location data and several variables for each location. Sheffield location data was extracted and saved into a new dataframe for further analysis. The significance of an accurate weather forecast would have an impact on various stakeholders including but not limited to businesses such as the steel industry and agriculture industry, government, emergency response units, public health, and safety.

Data pre-processing of the dataset, including data restructuring, handling missing values and checking for outliers and if need be, handling them to ensure data reliability and integrity.

Furthermore, statistical analysis was done to provided answers to the research questions and insights into accepting or rejecting the null hypothesis. The statistical analysis done covered exploring the univariate, bivariate and multivariate relationship between the parameters.

Auto ARIMA model was selected as the optimal model for time series prediction of surface pressure (TSK) as highlighted by its low AIC value. Moreover, the random forest model was selected as the best regression machine learning model for surface pressure meteorological phenomena.

In conclusion, the best model in providing the necessary insights into atmospheric data, climate conditions and atmospheric dynamics which affects all the stakeholders involved are the AUTO ARIMA Model Time series model and the Random Forest Regression Model.

The insights derived from the forecasts can aid the stakeholders in improving decision making, improving operational effectiveness and efficiency by adequately allocating resources and mitigate against risks associated with the changing weather conditions ultimately leading to a more progressive and safer environment.

REFERENCE

ADDIN ZOTERO_BIBL {"uncited":[],"omitted":[],"custom":[]} CSL_BIBLIOGRAPHY Abdallah, W., Abdallah, N., Marion, J.-M., Oueidat, M. and Chauvet, P. (2020) 'A Hybrid Methodology for Short Term Temperature Forecasting.' *International Journal of Intelligence Science*, 10(03) pp. 65–81.

Amram, M., Dunn, J., Toledano, J. J. and Zhuo, Y. D. (2021) 'Interpretable predictive maintenance for hard drives.' *Machine Learning with Applications*, 5, September, p. 100042.

Beinschroth, J. (2022) 'Implementing an effective qualitative risk analysis.' *In 2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC)*. Reykjavík, Iceland: IEEE, pp. 000143–000148.

Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. da P., Basto, J. P. and Alcalá, S. G. S. (2019) 'A systematic literature review of machine learning methods applied to predictive maintenance.' *Computers & Industrial Engineering*, 137, November, p. 106024.

Cheng, X., Chaw, J. K., Goh, K. M., Ting, T. T., Sahrani, S., Ahmad, M. N., Abdul Kadir, R. and Ang, M. C. (2022) 'Systematic Literature Review on Visual Analytics of Predictive Maintenance in the Manufacturing Industry.' *Sensors*. Multidisciplinary Digital Publishing Institute, 22(17) p. 6321.

Daniel, E. (2016) 'The Usefulness of Qualitative and Quantitative Approaches and Methods in Researching Problem-Solving Ability in Science Education Curriculum.' *Journal of Education and Practice*.

Dayo-Olupona, O., Genc, B., Celik, T. and Bada, S. (2023) 'Adoptable approaches to predictive maintenance in mining industry: An overview.' *Resources Policy*, 86, October, p. 104291.

Hasri, H., Mohd Aris, S. A. and Ahmad, R. (2023) 'Comparison of Auto ARIMA and Auto SARIMA Performance in COVID-19 Prediction.' *In 2023 IEEE 2nd National Biomedical Engineering Conference (NBEC)*. Melaka, Malaysia: IEEE, pp. 106–110.

Hayat Suhendar, M. T. and Widyani, Y. (2023) 'Machine Learning Application Development Guidelines Using CRISP-DM and Scrum Concept.' *In 2023 IEEE International Conference on Data and Software Engineering (ICoDSE)*. Toba, Indonesia: IEEE, pp. 168–173.

Hewage, P., Behera, A., Trovati, M. and Pereira, E. (2019) 'Long-Short Term Memory for an Effective Short-Term Weather Forecasting Model Using Surface Weather Data.' *In* MacIntyre, J., Maglogiannis, I., Iliadis, L., and Pimenidis, E. (eds) *Artificial Intelligence Applications and Innovations*. Cham: Springer International Publishing (IFIP Advances in Information and Communication Technology), pp. 382–390.

Hotz, N. (2018a) 'What is CRISP DM?' Data Science Process Alliance. 10th September. [Online] Available from: https://www.datascience-pm.com/crisp-dm-2/ [Accessed on 11th March 2024].

Hotz, N. (2018b) 'What is CRISP DM?' Data Science Process Alliance. 10th September. [Online] Available from: https://www.datascience-pm.com/crisp-dm-2/ [Accessed on 10th March 2024].

Huang, N. E. and Wu, Z. (2008) 'A review on Hilbert-Huang transform: Method and its applications to geophysical studies.' *Reviews of Geophysics*, 46(2) p. 2007RG000228.

Hyndman, R. J. and Athanasopoulos, G. (2018) *Forecasting: principles and practice*. 2nd edition, Lexington, Ky.: Otexts, online, open-access textbook.

Kelleher, J. D. and Tierney, B. (2018) *Data science*. Cambridge, Massachusetts London, England: The MIT Press (The MIT Press essential knowledge series).

Lee, J., Ni, J., Singh, J., Jiang, B., Azamfar, M. and Feng, J. (2020) 'Intelligent Maintenance Systems and Predictive Manufacturing.' *Journal of Manufacturing Science and Engineering*, 142(11) p. 110805.

Liu, H. (2014) 'Discussion on the Statistical Analysis Method.' *In 2014 Seventh International Joint Conference on Computational Sciences and Optimization*. Beijing, China: IEEE, pp. 383–385.

Liu, X., Han, F., Ghazali, K., Mohamed, I. and Zhao, Y. (2019) ‘A review of Convolutional Neural Networks in Remote Sensing Image.’ *In*, pp. 263–267.

Raghavan, K., Singh, M., Vellore, R. and Mujumdar, M. (2020) *Progress and Prospects in Weather and Climate Modelling*.

Scher, S. and Messori, G. (2018) ‘Predicting weather forecast uncertainty with machine learning.’ *Quarterly Journal of the Royal Meteorological Society*, 144(717) pp. 2830–2841.

Schröer, C., Kruse, F. and Gómez, J. M. (2021) ‘A Systematic Literature Review on Applying CRISP-DM Process Model.’ *Procedia Computer Science*, 181 pp. 526–534.

Serradilla, O., Zugasti, E. and Zurutuza, U. (2020) ‘Deep learning models for predictive maintenance: a survey, comparison, challenges and prospect.’ arXiv.

Sujjaviriyasup, T. and Pitiruek, K. (2017) ‘A comparison between MODWT-SVM-DE hybrid model and ARIMA model in forecasting primary energy consumptions.’ *In 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. Singapore: IEEE, pp. 799–802.

Sulaimon, I. A., Alaka, H. A., Olu-Ajayi, R., Ahmad, M., Ajayi, S. and Hye, A. (n.d.) ‘Air Pollution Prediction using Machine Learning – A Review.’

Talaviya, A. (2023) ‘CRISP-DM framework: A foundational data mining process model.’ Medium. 30th October. [Online] Available from: https://medium.com/@avikumart_/crisp-dm-framework-a-foundational-data-mining-process-model-86fe642da18c [Accessed on 15th May 2024].

Tawn, R. and Browell, J. (2022) ‘A review of very short-term wind and solar power forecasting.’ *Renewable and Sustainable Energy Reviews*, 153, January, p. 111758.

Waheeb, W., Ghazali, R. and Shah, H. (2019) ‘Nonlinear Autoregressive Moving-average (NARMA) Time Series Forecasting Using Neural Networks.’ *In 2019 International Conference on Computer and Information Sciences (ICCIS)*. Sakaka, Saudi Arabia: IEEE, pp. 1–5.

Wardah, T., Kamil, A. A., Sahol Hamid, A. B. and Maisarah, W. W. I. (2011) 'Statistical verification of numerical weather prediction models for quantitative precipitation forecast.' *In 2011 IEEE Colloquium on Humanities, Science and Engineering*. Penang, Malaysia: IEEE, pp. 88–92.

Wickham, H. and Grolemund, G. (2016) *R for data science: import, tidy, transform, visualize, and model data*. First edition, Sebastopol, CA: O'Reilly.

Xinxin, W., Xiaopan, S., Xueyi, A. and Shijia, L. (2023) 'Short-term wind speed forecasting based on a hybrid model of ICEEMDAN, MFE, LSTM and informer.' Wang, L. (ed.) *PLOS ONE*, 18(9) p. e0289161.

Zhu, Z.-Q., Lei, Y., Qi, G., Chai, Y., Mazur, N., An, Y. and Huang, X. (2022) 'A review of the application of deep learning in intelligent fault diagnosis of rotating machinery.' *Measurement*, 206, December, p. 112346.