WRF Analysis Assessment2 markdown

Oluseye Emmanuel Fasuan

2024-04-12

Table of Contents

1 course: DAT7006_2_2308483

1.1 Loading the Libraries

1.2 Loading of dataset

Data-set loaded into the project folder and read inside R using read.csv

```
data <- read.csv(here("data/WRFdata_May2018.csv"))
```

1.3 Reshaping the dataset

1.3.1 Extract Location " Cuxton, Rochester, ME2 1DL"

```
# Extract rows 4287 XLAT 51.375 XLONG 0.456 needed for analysis

new_data <- data[c(1, 4285), ]

# Replacing a wrong column name if available

if ("X.2225" %in% colnames(new_data)) {
  names(new_data)[names(new_data) == 'X.2225'] <- 'X31.05.2018.21.00'
} else {
  cat("Column 'X.2225' does not exist in the data frame.\n")
}

## Column 'X.2225' does not exist in the data frame.
```

1.3.2 Removing Xlat and Xlong

```
# Remove the Xlat and Xlong Columns
new_data <- new_data[-c(1,2)]
```

1.3.3 Data Reshaping

```
# Extrating datatime
extract_datetime <- function(header){
  match <-str_extract(header, "\\d{2}\\.\\d{2}\\.\\d{4}\\.\\d{2}\\.\\d{2}")
  return(ifelse(is.na(match), "", match))
}
```

1.3.4 Creating New datatime column

```
#Implementaing Sapply

extracted_datetime <- sapply(colnames(new_data), extract_datetime)

# Create a separate dataframe for the extracted datetime

datetime_df <- data.frame(datatime = extracted_datetime)

datetime_df1 <- data.frame(data = datetime_df)

#Handling the blank rows
datetime_df <- datetime_df[!apply(is.na(datetime_df) | datetime_df == "", 1,
all),]
```

Data Reshaping:

This code session defines a function extract_datetime to extract the date and time from the remaining column headers using regular expressions. It then uses sapply to apply this function to all column names in new_data and stores the extracted date/time in a separate data frame datetime_df. It handles potential blank rows in datetime_df using apply and logical operators.

```
datetime_df1 <- data.frame(data = datetime_df)

# create new copy of dataframe
newdata_copy <- new_data

# Replace the Column headers with the header row
header_row_index <- 1
colnames(newdata_copy) <- unlist(newdata_copy[header_row_index, ])
# Delete the header row
newdata_copy <- newdata_copy[-header_row_index, ]
```

This upper code session creates a copy of new_data named newdata_copy. It replaces the column names of newdata_copy with the values from the first row. It then deletes the first row (now redundant).

1.3.5 Replacing the Empty/NA on Header Row

```
# Replacing the NA Values on the header column

sequence <- paste0(rep(c("TSK", "PSFC", "U10", "V10", "Q2", "RAINC",
"RAINNC", "SNOW", "TSLB", "SMOIS"),times = 248))
colnames(newdata_copy) <- sequence
```

1.3.6 Split the dataframe, convert and Merge

```
# Splitting the Dataframe into separate dataframes
split_dfs <- split.default(newdata_copy, rep(1:(ncol(newdata_copy)%/% 10),
each = 10))
```

```r
# Convert the column value to numerica format
for (i in seq_along(split_dfs)) {split_dfs[[i]] <- lapply(split_dfs[[i]],
as.numeric)
}

# merge the split dataframes

merged_df <- bind_rows(split_dfs)
```

The code splits newdata_copy into separate data frames based on the number of columns (10 columns per data frame). It iterates through the split data frames and converts their column values to numeric format using lapply. it merges the split data frames back together using bind_rows.

1.3.7 Handling the missing in columns

```r
# Function to replace NA with mean of two previous values
fill_na_with_mean <- function(x) {
  na_index <- which(is.na(x))
  for (i in na_index) {
    if (i > 2) {
      x[i] <- mean(c(x[i-1], x[i-2]), na.rm = TRUE)
    }
  }
  return(x)
}
# Apply the function to all columns except 'DATETIME'
merged_df[-ncol(merged_df)] <- lapply(merged_df[-ncol(merged_df)],
fill_na_with_mean)

# Rename 'data' column to 'DATETIME'
colnames(merged_df)[colnames(merged_df) == "data"] <- "DATETIME"
```

This chunck code defines a function fill_na_with_mean that replaces NA values in a vector with the mean of the two preceding values (excluding the first two).

It applies this function to all columns except "DATETIME" in the merged data frame merged_df.

It renames the "data" column to "DATETIME".

```r
# Joining the datetime dataframe with the merged dataframe
data1 <- cbind(merged_df, datetime_df1)

# convert datetime column to POSIXct format
format_string <- as.POSIXct(data1$DATETIME, format = format_string)

# Move 'DATETIME' column to first position
data1 <- data1[c("data", setdiff(colnames(data1), "DATETIME"))]
```

This session code combines the merged data frame merged_df with the datetime_df containing the extracted date/time information using cbind. It defines a format string to parse the datetime column values correctly.

It converts the "DATETIME" column to POSIXct format (a standard date/time format in R) using as.POSIXct with the defined format string.

Hence, it rearranges the columns, placing "DATETIME" as the first column.

```
# Remove the last Columns (12)
data1 <- data1[-c(12)]

names(data1)[names(data1) == 'data'] <- "DATETIME"

# This line ensures the "DATETIME" column name is consistent.
```

1.3.8 Replacing Missing Value in each column

```
# Replacing Missing Values in RAINNC
for (i in which(is.na(data1$RAINNC))) {
  if (i < nrow(data1) - 2) {
    data1$RAINNC[i] <- mean(data1$RAINNC[(i+1):(i+3)], na.rm = TRUE)
  }
}

# Replacing Missing Values in SMOIS
for (i in which(is.na(data1$SMOIS))) {
  if (i < nrow(data1) - 2) {
    data1$SMOIS[i] <- mean(data1$SMOIS[(i+1):(i+3)], na.rm = TRUE)
  }
}
```

1.3.9 Convert DATETIME column to POSIXct format

```
# Define the format string for parsing the datetime column
format_string <- "%d.%m.%Y.%H.%M"

# Convert DATETIME column to POSIXct format
data1$DATETIME <- as.POSIXct(data1$DATETIME, format = format_string)
```

1.4 Calculate Wind Speed

```
data1$WIND_SPEED <- round(sqrt(data1$U10^2 + data1$V10^2), 2)
```

This line calculates the wind speed by finding the square root of the squared values in the "U10" and "V10" columns (assuming they represent wind components) and stores the result in a new column named "WIND_SPEED". The values are rounded to two decimal places.

1.5 Exploratory Data Analysis

```
str(data1)
```

```
## 'data.frame':	248 obs. of  12 variables:
##  $ DATETIME  : POSIXct, format: "2018-05-01 00:00:00" "2018-05-01
03:00:00" ...
##  $ TSK       : num  276 276 277 289 293 ...
##  $ PSFC      : num  1e+05 1e+05 1e+05 1e+05 1e+05 ...
##  $ U10       : num  3.5 3.8 4.4 4.2 3.6 5.2 3.2 0 -0.4 0.4 ...
##  $ V10       : num  -0.1 1.1 0.4 0.8 3.3 6.4 5.9 5.3 5.5 6.9 ...
##  $ Q2        : num  0.00437 0.00433 0.00398 0.00401 0.0047 ...
##  $ RAINC     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RAINNC    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SNOW      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TSLB      : num  279 278 278 280 283 ...
##  $ SMOIS     : num  0.359 0.352 0.355 0.352 0.348 ...
##  $ WIND_SPEED: num  3.5 3.96 4.42 4.28 4.88 8.25 6.71 5.3 5.51 6.91 ...

# Check the header summary

head(data1)

##                DATETIME   TSK   PSFC U10  V10       Q2 RAINC RAINNC SNOW
TSLB
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370     0      0    0
279.1
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330     0      0    0
278.5
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980     0      0    0
278.0
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010     0      0    0
279.8
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700     0      0    0
283.0
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.4 0.004355     0      0    0
285.1
##       SMOIS WIND_SPEED
## 1 0.3591000       3.50
## 2 0.3517333       3.96
## 3 0.3551000       4.42
## 4 0.3522000       4.28
## 5 0.3479000       4.88
## 6 0.3441000       8.25
```

1.6 Perform Exploratory Data Analysis (EDA) and identify potential outliers in the data frame

```
# Summary statistics
summary(data1)

##     DATETIME                           TSK              PSFC
##  Min.   :2018-05-01 00:00:00   Min.   :272.1   Min.   : 99872
##  1st Qu.:2018-05-08 17:15:00   1st Qu.:281.6   1st Qu.:100761
##  Median :2018-05-16 10:30:00   Median :287.1   Median :101145
```

```
##  Mean   :2018-05-16 10:30:00   Mean   :288.0   Mean   :101123
##  3rd Qu.:2018-05-24 03:45:00   3rd Qu.:294.9   3rd Qu.:101474
##  Max.   :2018-05-31 21:00:00   Max.   :305.8   Max.   :102065
##       U10               V10               Q2                RAINC
##  Min.   :-7.1000   Min.   :-8.200   Min.   :0.003870   Min.   : 0.0000
##  1st Qu.:-2.3000   1st Qu.:-2.900   1st Qu.:0.005680   1st Qu.: 0.0000
##  Median :-0.5750   Median :-1.550   Median :0.006860   Median : 0.0000
##  Mean   :-0.4601   Mean   :-1.055   Mean   :0.007379   Mean   : 0.5552
##  3rd Qu.: 1.5000   3rd Qu.: 0.700   3rd Qu.:0.008985   3rd Qu.: 0.0000
##  Max.   : 5.2000   Max.   : 9.400   Max.   :0.013200   Max.   :20.3000
##      RAINNC            SNOW        TSLB            SMOIS
WIND_SPEED
##  Min.   :0.0000   Min.   :0   Min.   :277.8   Min.   :0.2750   Min.
:0.140
##  1st Qu.:0.0000   1st Qu.:0   1st Qu.:283.5   1st Qu.:0.2883   1st
Qu.:2.540
##  Median :0.0000   Median :0   Median :286.3   Median :0.3000   Median
:3.450
##  Mean   :0.2117   Mean   :0   Mean   :286.4   Mean   :0.3037   Mean
:3.632
##  3rd Qu.:0.0000   3rd Qu.:0   3rd Qu.:289.1   3rd Qu.:0.3159   3rd
Qu.:4.562
##  Max.   :7.9000   Max.   :0   Max.   :295.8   Max.   :0.3676   Max.
:9.430
```

```
# Descriptive statistics for numerical variables
describe(data1[, sapply(data1, is.numeric)])  # Filter numeric columns
```

```
## data1[, sapply(data1, is.numeric)]
##
##  11  Variables      248  Observations
## ---------------------------------------------------------------------------
------
## TSK
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      175        1      288    9.254    276.1    277.5
##      .25      .50      .75      .90      .95
##    281.6    287.1    294.9    299.1    300.7
##
## lowest : 272.1 273.5 273.9 274.3 274.6, highest: 303.1 303.8 304   305.7
305.8
## ---------------------------------------------------------------------------
------
## PSFC
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      231        1   101123    588.3   100230   100398
##      .25      .50      .75      .90      .95
##   100761   101145   101474   101811   101952
##
## lowest :  99872  99873  99901  99960 100082, highest: 102011 102034 102047
```

```
102050 102065
## ---------------------------------------------------------------------------
------
## U10
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0       98        1  -0.4601    2.877   -4.265   -3.700
##      .25      .50      .75      .90      .95
##   -2.300   -0.575    1.500    2.900    3.500
##
## lowest : -7.1 -6.5 -6.1 -6   -5.8, highest: 4.4  4.6  4.7  5.1  5.2
## ---------------------------------------------------------------------------
------
## V10
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0       98        1   -1.055    3.185   -5.200   -4.330
##      .25      .50      .75      .90      .95
##   -2.900   -1.550    0.700    2.800    3.765
##
## lowest : -8.2 -7.3 -6.7 -6.6 -6.1, highest: 5.9  6.4  6.9  8    9.4
## ---------------------------------------------------------------------------
------
## Q2
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      212        1 0.007379 0.002386 0.004720 0.004997
##      .25      .50      .75      .90      .95
## 0.005680 0.006860 0.008985 0.010727 0.011270
##
## lowest : 0.00387 0.00398 0.00401 0.00407 0.00422
## highest: 0.01238 0.01245 0.01257 0.01271 0.0132
## ---------------------------------------------------------------------------
------
## RAINC
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0       20    0.321   0.5552    1.059     0.00     0.00
##      .25      .50      .75      .90      .95
##     0.00     0.00     0.00     0.33     2.60
##
## Value        0.0   0.2   0.3   0.4   0.7   1.1   1.8   2.1   2.5   2.6
2.8
## Frequency    218     1     4     1     4     2     1     1     1     3
1
## Proportion 0.879 0.004 0.016 0.004 0.016 0.008 0.004 0.004 0.004 0.012
0.004
##
## Value        3.6   5.0   6.0   7.2  10.6  11.4  12.7  18.5  20.3
## Frequency      1     1     1     2     1     2     1     1     1
## Proportion 0.004 0.004 0.004 0.008 0.004 0.008 0.004 0.004 0.004
##
## For the frequency table, variable is rounded to the nearest 0
## ---------------------------------------------------------------------------
```

```
------
## RAINNC
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0       14    0.273   0.2117   0.3992    0.000    0.000
##      .25      .50      .75      .90      .95
##    0.000    0.000    0.000    0.030    2.145
##
## Value        0.0   0.1   0.2   0.3   0.5   0.8   0.9   1.0   1.2   1.3
2.6
## Frequency    223     1     1     1     1     1     1     4     1     1
3
## Proportion 0.899 0.004 0.004 0.004 0.004 0.004 0.004 0.016 0.004 0.004
0.012
##
## Value        2.8   5.1   7.9
## Frequency      8     1     1
## Proportion 0.032 0.004 0.004
##
## For the frequency table, variable is rounded to the nearest 0
## ---------------------------------------------------------------------------
------
## SNOW
##        n  missing distinct     Info     Mean      Gmd
##      248        0        1        0        0        0
##
## Value        0
## Frequency  248
## Proportion   1
## ---------------------------------------------------------------------------
------
## TSLB
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      128        1    286.4     4.43    280.3    281.3
##      .25      .50      .75      .90      .95
##    283.5    286.3    289.1    291.4    293.0
##
## lowest : 277.8 278   278.5 279.1 279.7, highest: 293.9 295.2 295.5 295.6
295.8
## ---------------------------------------------------------------------------
------
## SMOIS
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      205        1   0.3037  0.02222   0.2787   0.2818
##      .25      .50      .75      .90      .95
##   0.2883   0.3000   0.3159   0.3336   0.3423
##
## lowest : 0.275  0.2752 0.2756 0.2759 0.276 , highest: 0.3551 0.3556 0.3591
0.36   0.3676
## ---------------------------------------------------------------------------
------
```

```
## WIND_SPEED
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      248        0      196        1    3.632    1.839    1.207    1.700
##      .25      .50      .75      .90      .95
##    2.540    3.450    4.562    5.793    6.763
##
## lowest : 0.14 0.5  0.58 0.6  0.72, highest: 8.01 8.2  8.25 8.43 9.43
## ---------------------------------------------------------------------------
------
```

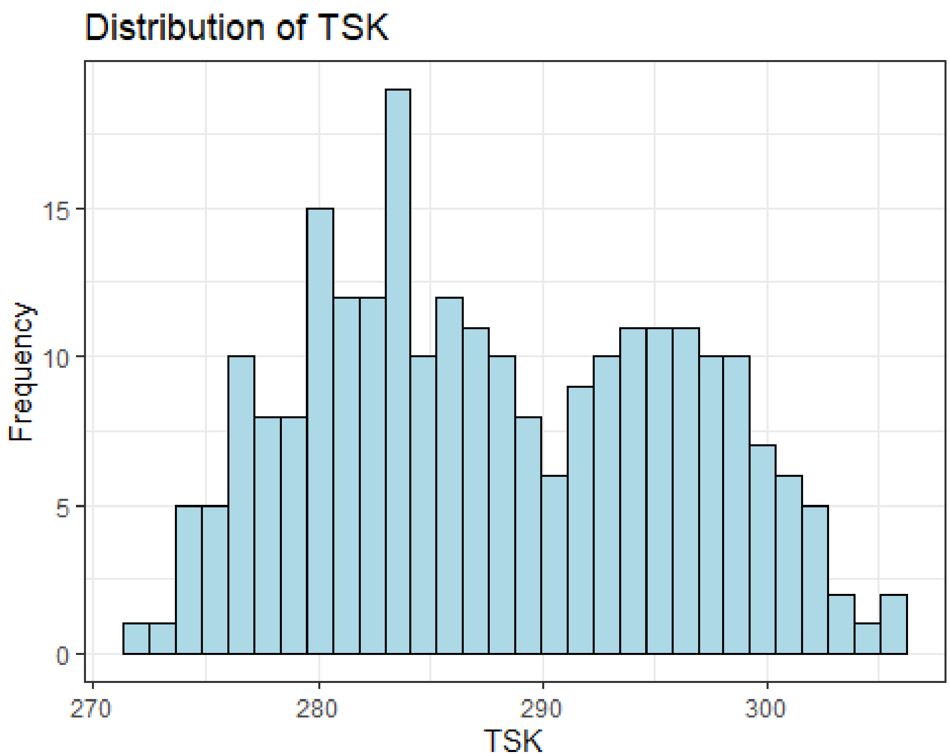
This chunk code displays summary statistics for the entire data frame data1 using summary. It then uses describe to focus on numerical variables and provide more detailed descriptive statistics.

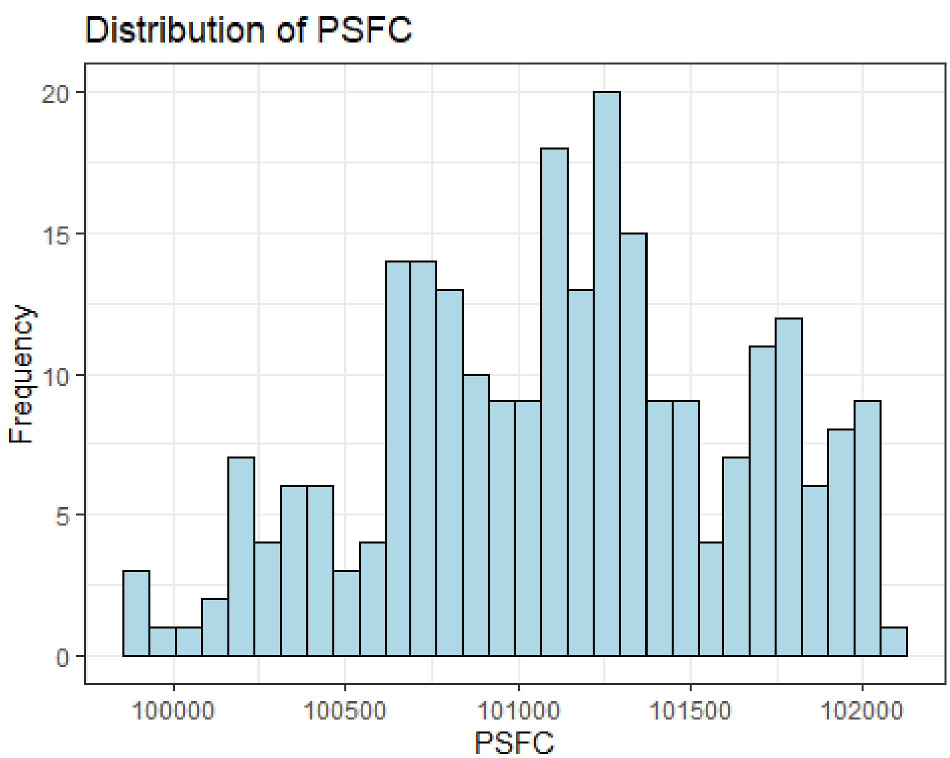
```r
# Check for missing values
colMeans(is.na(data1))  # Proportion of missing values per column
```

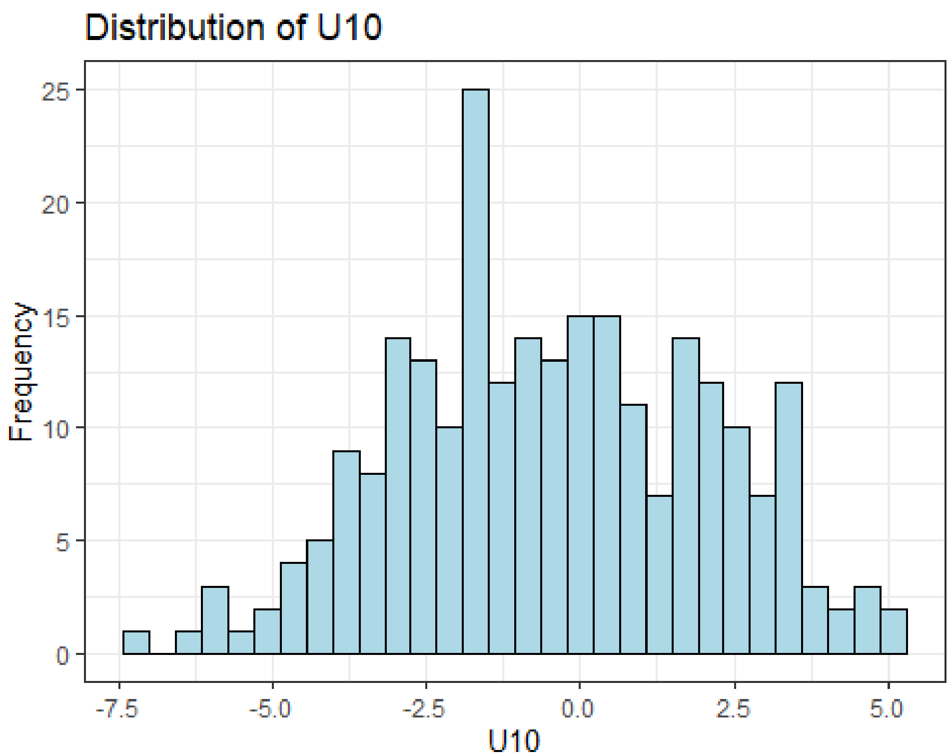
```
##   DATETIME        TSK       PSFC        U10        V10         Q2
RAINC
##          0          0          0          0          0          0
0
##     RAINNC       SNOW       TSLB      SMOIS WIND_SPEED
##          0          0          0          0          0
```

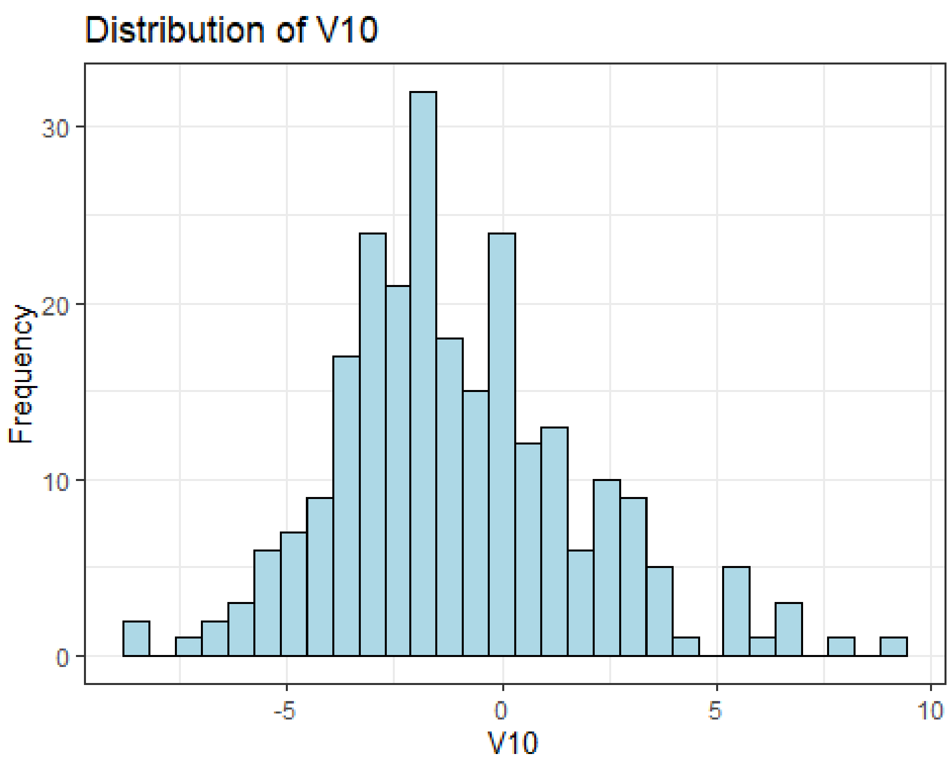
```r
# Explore data distribution with histograms
for (var in names(data1)[sapply(data1, is.numeric)]) {
print(
ggplot(data1, aes_string(x = var)) +
geom_histogram(bins = 30, color = "black", fill = "lightblue") +
labs(title = paste("Distribution of", var), x = var, y = "Frequency") +
theme_bw()
)
}
```

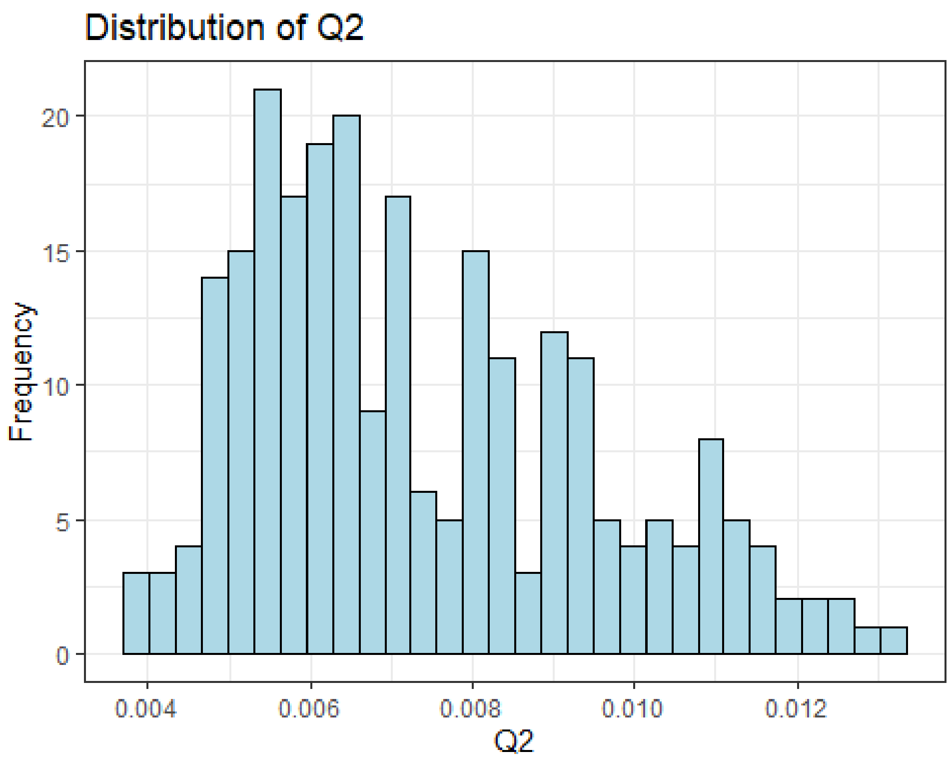
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

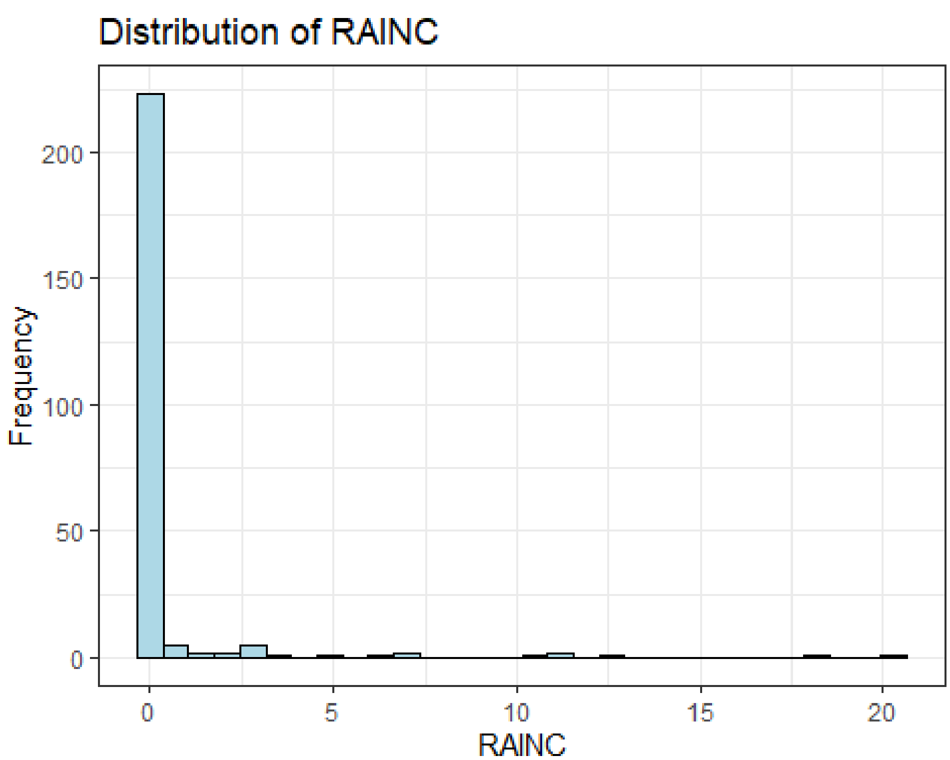

Distribution of TSK
Frequency
0
5
10
15
270
280
290
300
TSK


Distribution of PSFC
Frequency
0
5
10
15
20
100000
100500
101000
101500
102000
PSFC

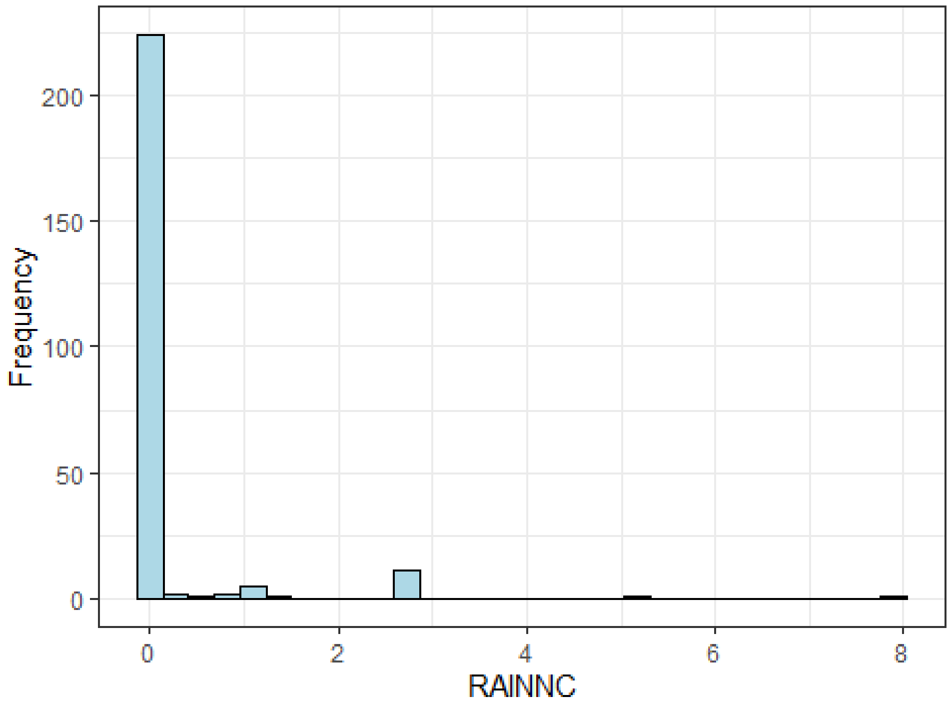

Distribution of U10
Frequency
U10
0
5
10
15
20
25
-7.5
-5.0
-2.5
0.0
2.5
5.0


Distribution of V10
Frequency
V10
0
10
20
30
-5
0
5
10

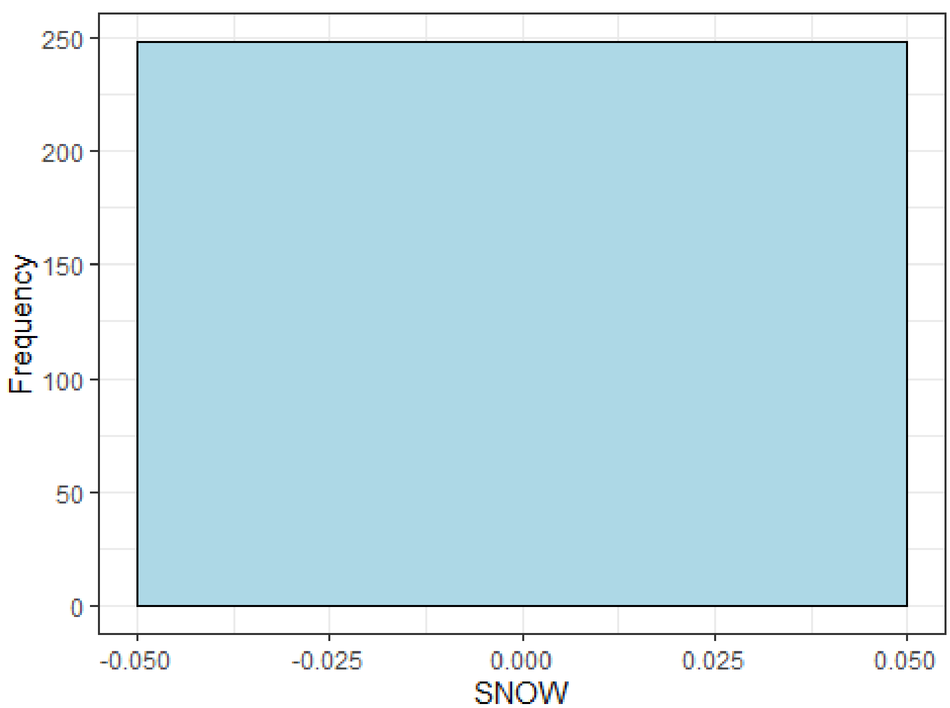

Distribution of Q2
Frequency
Q2
0
5
10
15
20
0.004
0.006
0.008
0.010
0.012

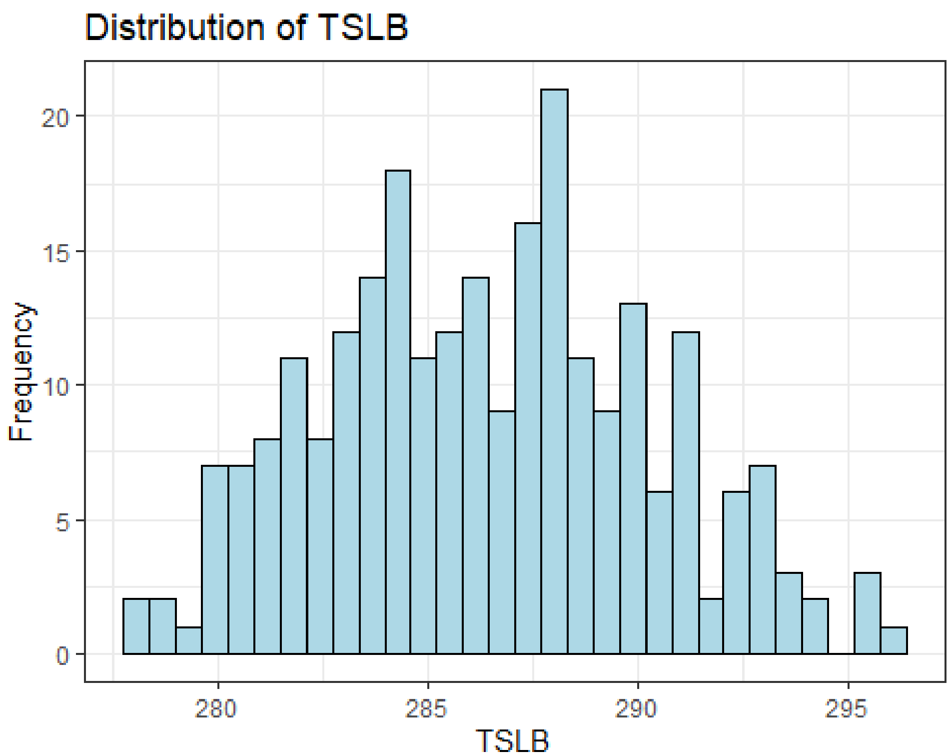

Distribution of RAINC
Frequency
RAINC
0
50
100
150
200
0
5
10
15
20

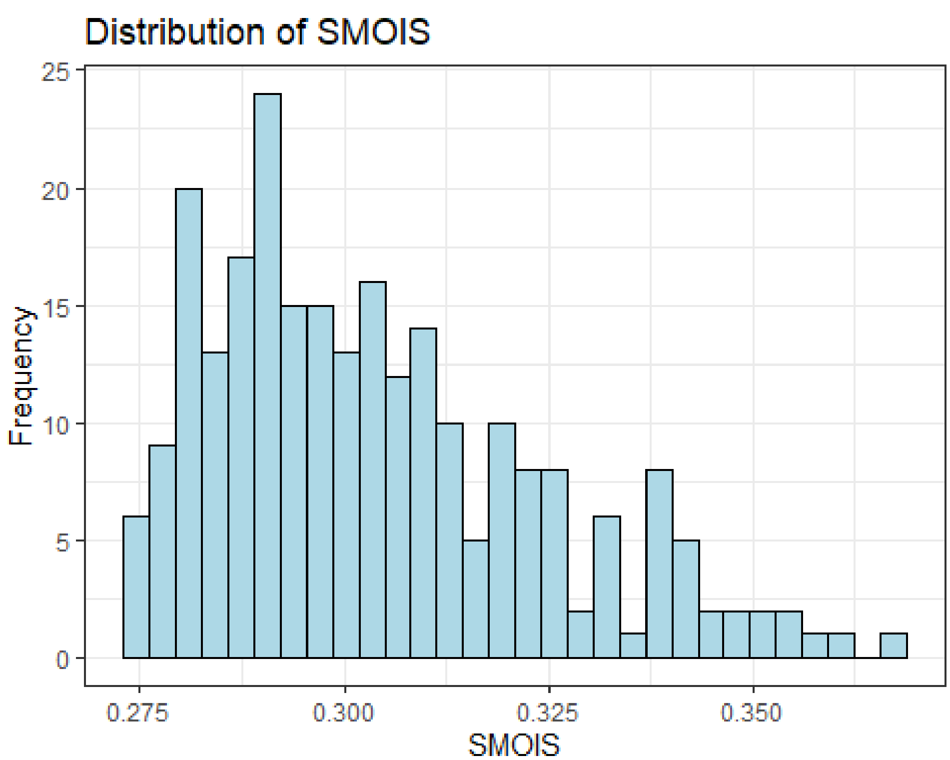
Distribution of RAINNC

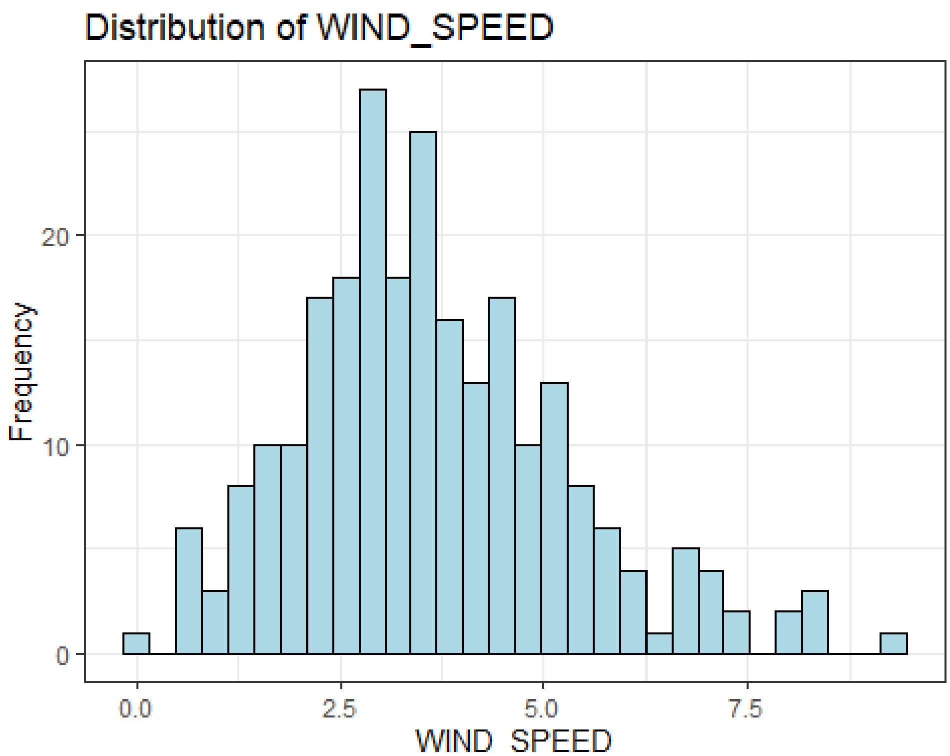


Distribution of SNOW




Distribution of TSLB
Frequency
0
5
10
15
20
280
285
290
295
TSLB


Distribution of SMOIS
Frequency
0
5
10
15
20
25
0.275
0.300
0.325
0.350
SMOIS



1.6.1 The code iterates through the numeric columns of data1.

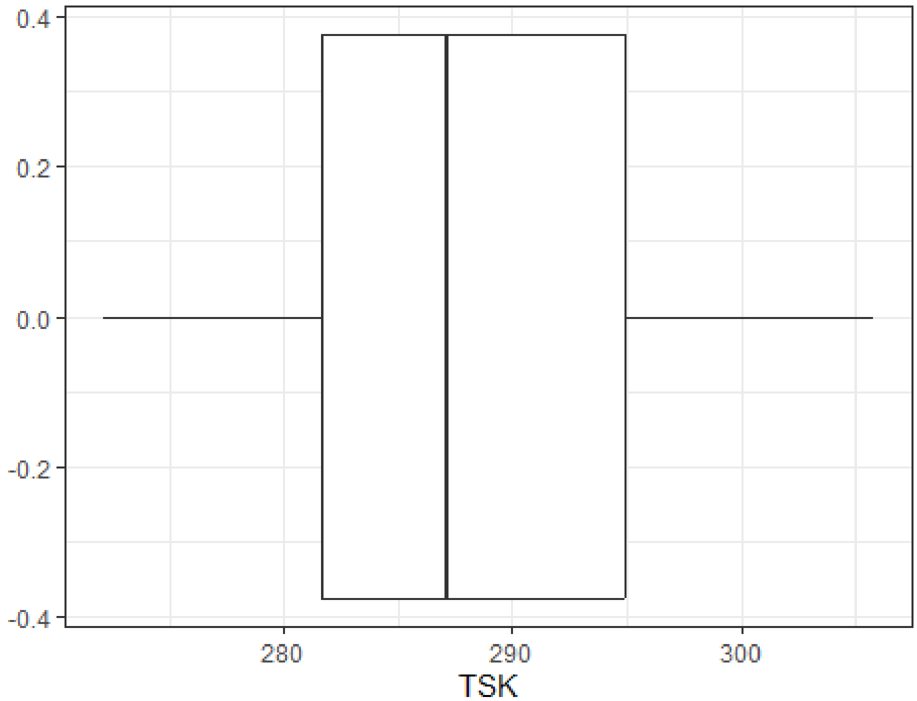
For each column, it creates a ggplot object to visualize the distribution using a histogram. The histogram displays the frequency of data points across different value ranges.

TSK Distribution: The distribution shown in the image is bimodal, as it has two distinct peaks. The histogram displays values of "TSK" with one peak around 280 and another around 290. This suggests that there are two common values where data points are concentrated, indicating potentially different processes or groups contributing to the dataset.

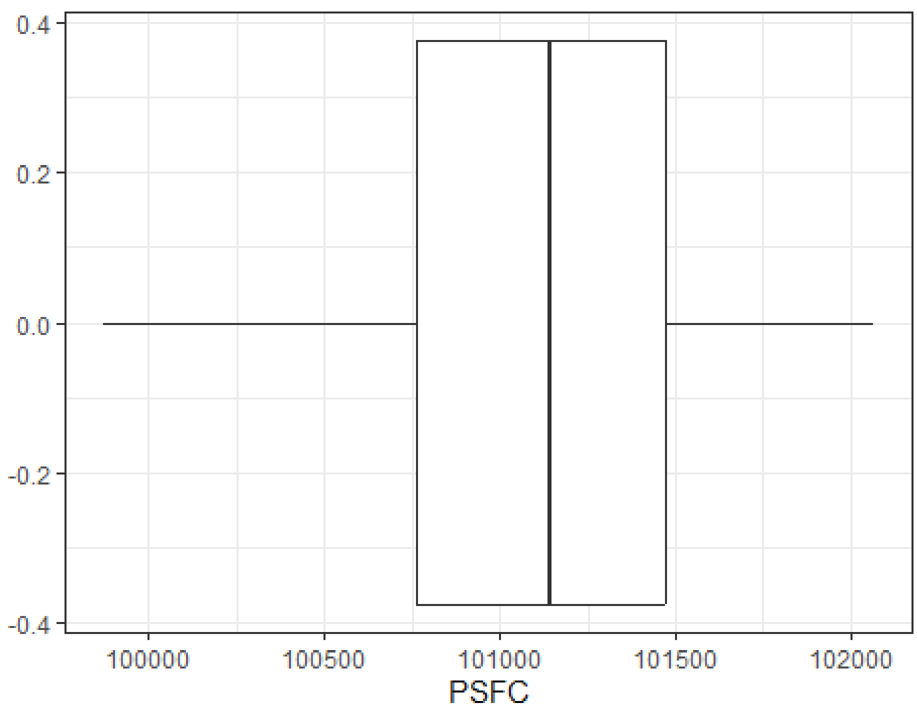
PSFC Distribution: Similar to TSK, PSFC Distribution is bimodal, as it has two distinct peaks. The histogram displays values of "PSFC" with one peak around 100500 and another around 101500

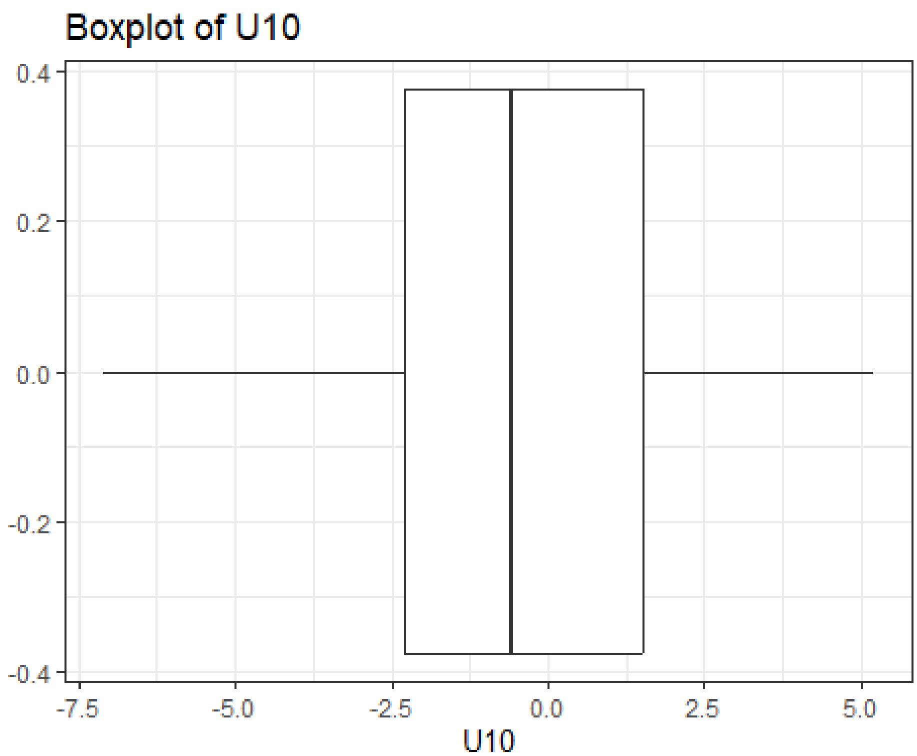
```
# Explore data distribution with boxplots and identify outliers
for (var in names(data1)[sapply(data1, is.numeric)]) {
print(
ggplot(data1, aes_string(x = var)) +
geom_boxplot() +
labs(title = paste("Boxplot of", var), x = var, y = "") +
theme_bw()
)
}
```

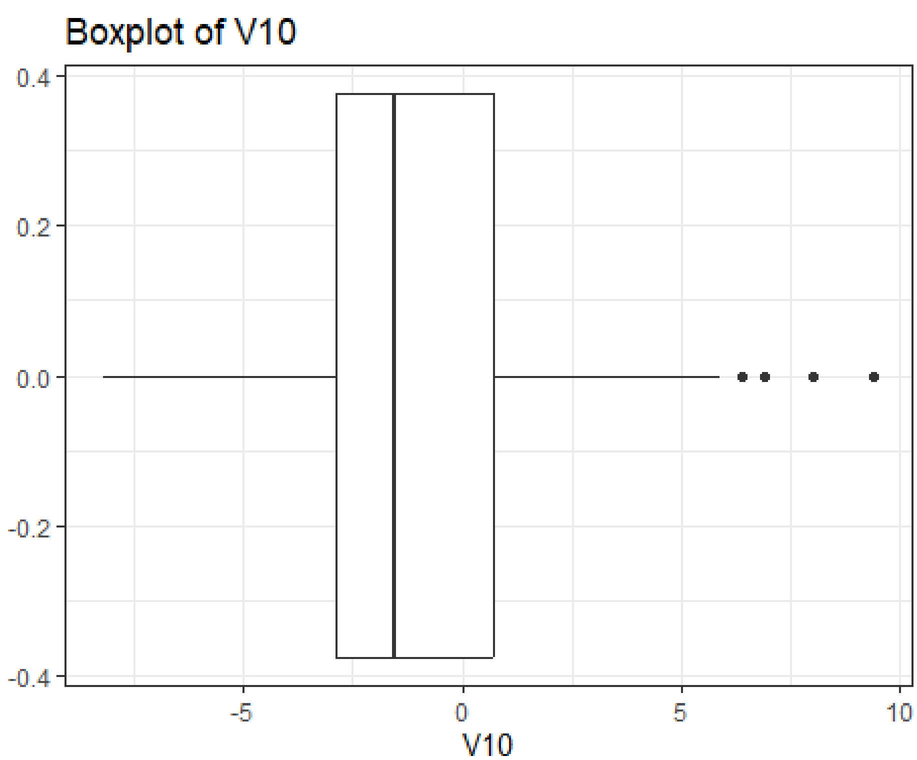
Boxplot of TSK

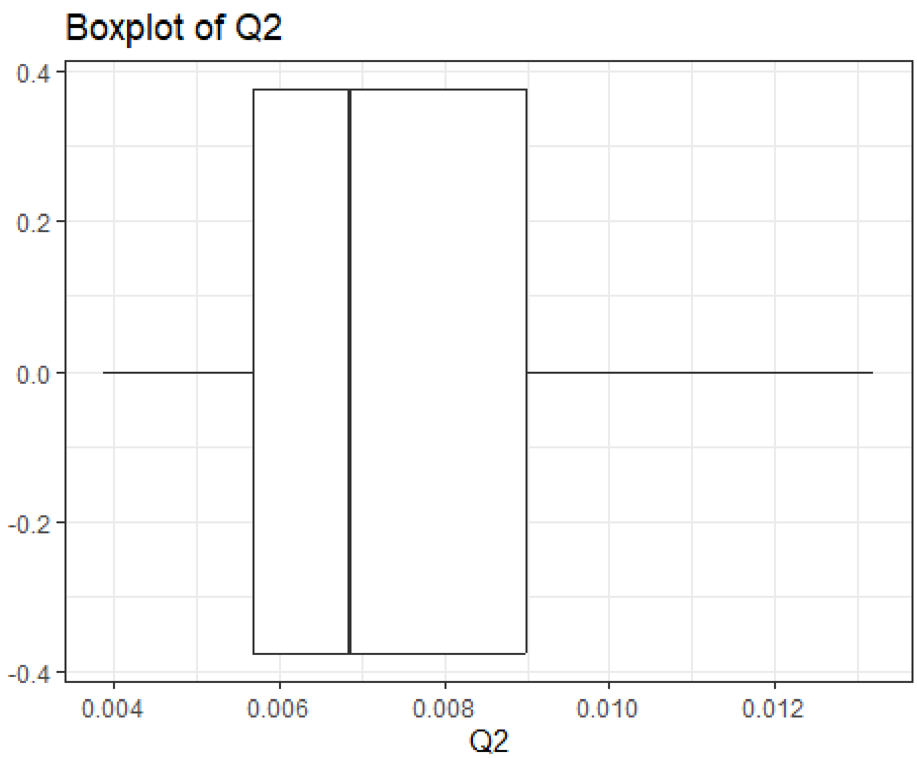


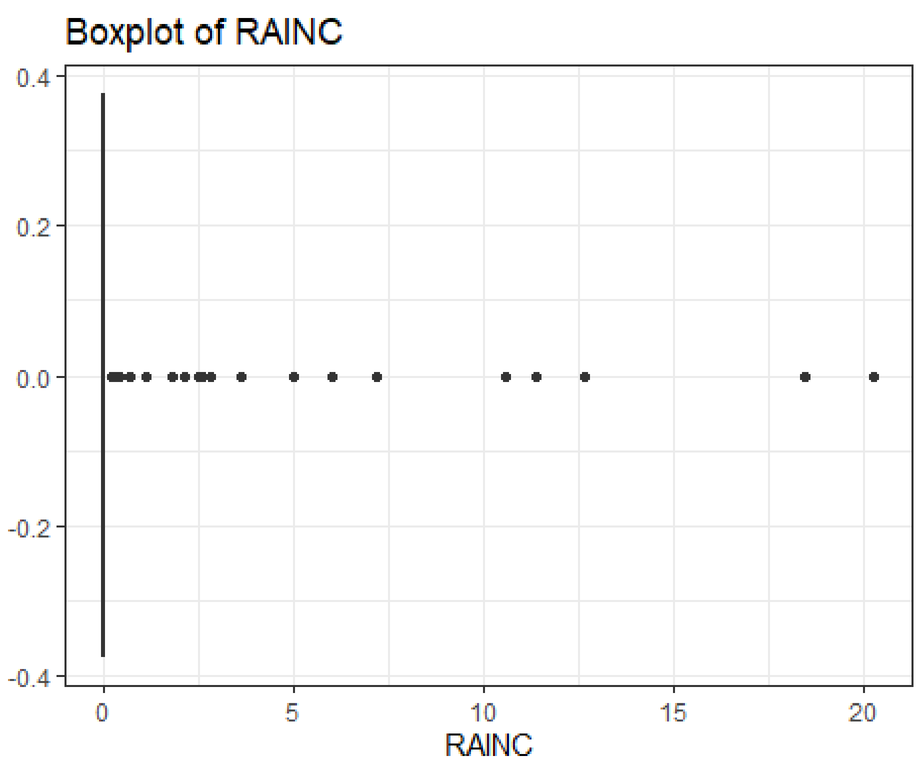
Boxplot of PSFC

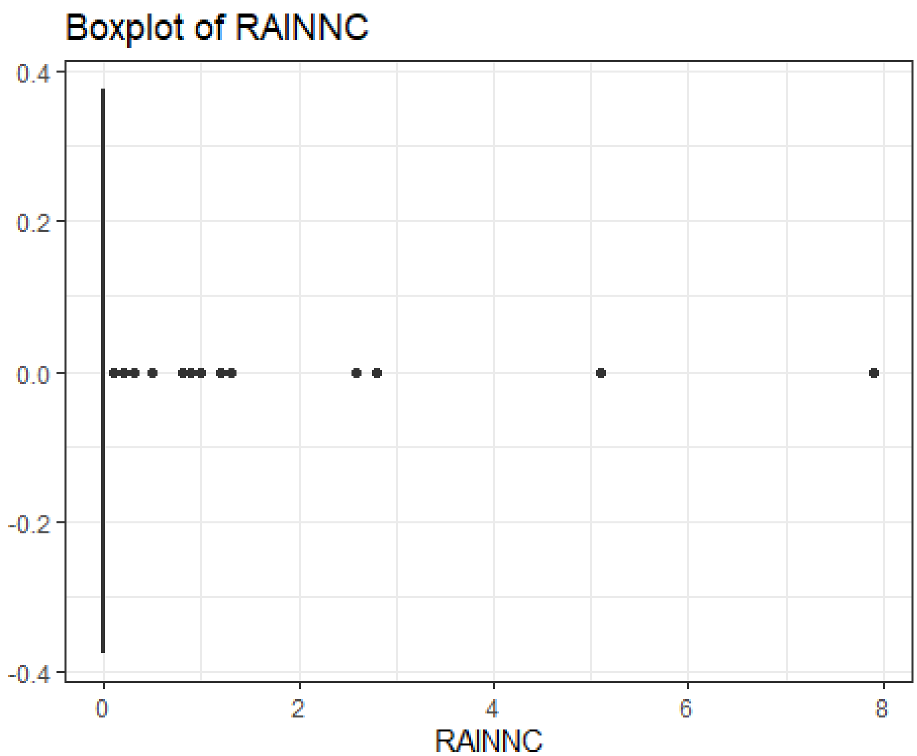


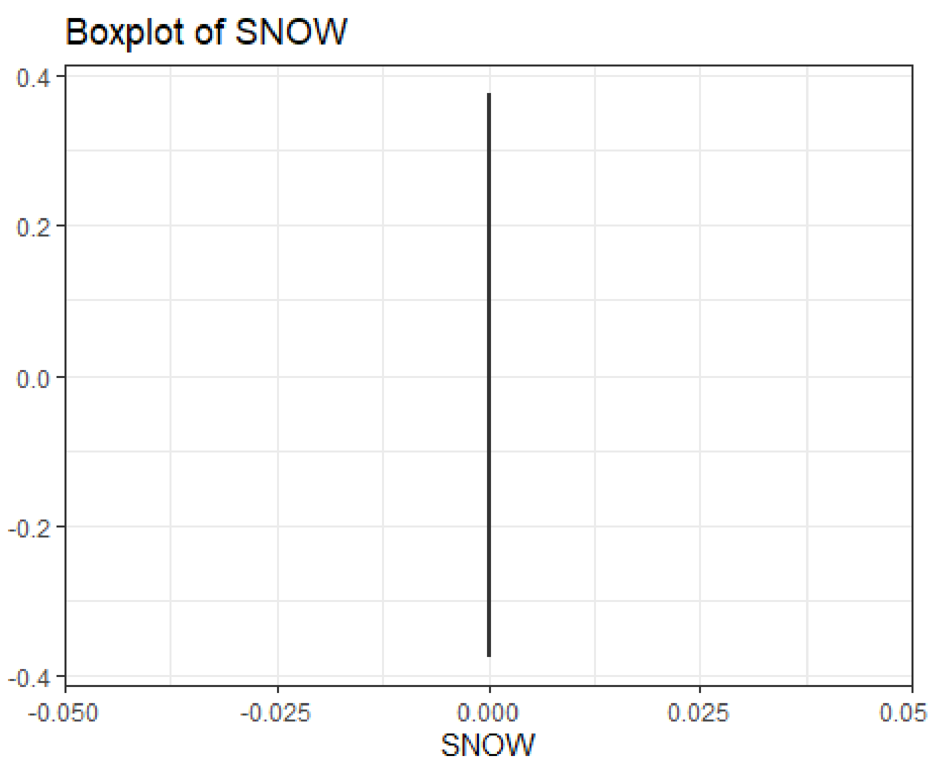

Boxplot of U10
0.4
0.2
0.0
-0.2
-0.4
-7.5
-5.0
-2.5
0.0
2.5
5.0
U10

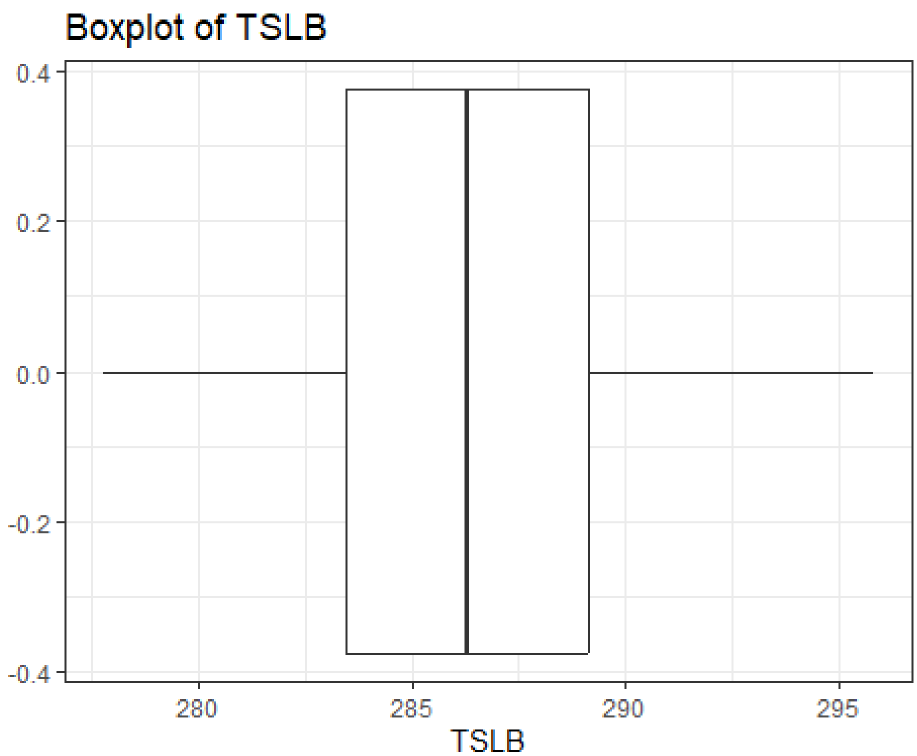

Boxplot of V10
0.4
0.2
0.0
-0.2
-0.4
-5
0
5
10
V10

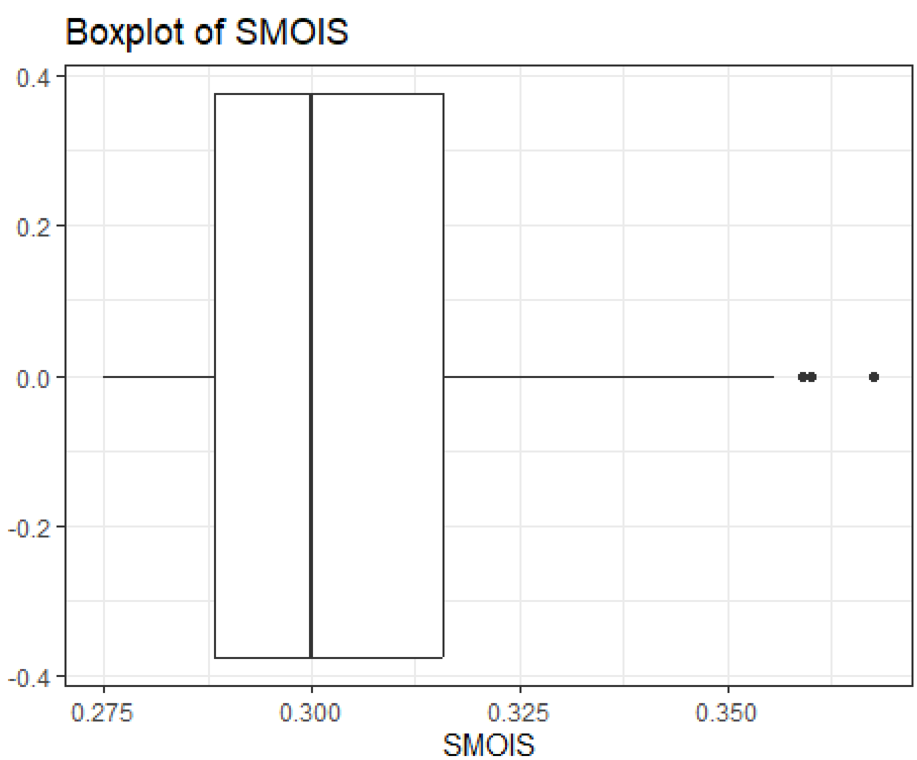

Boxplot of Q2
0.4
0.2
0.0
-0.2
-0.4
0.004
0.006
0.008
0.010
0.012
Q2

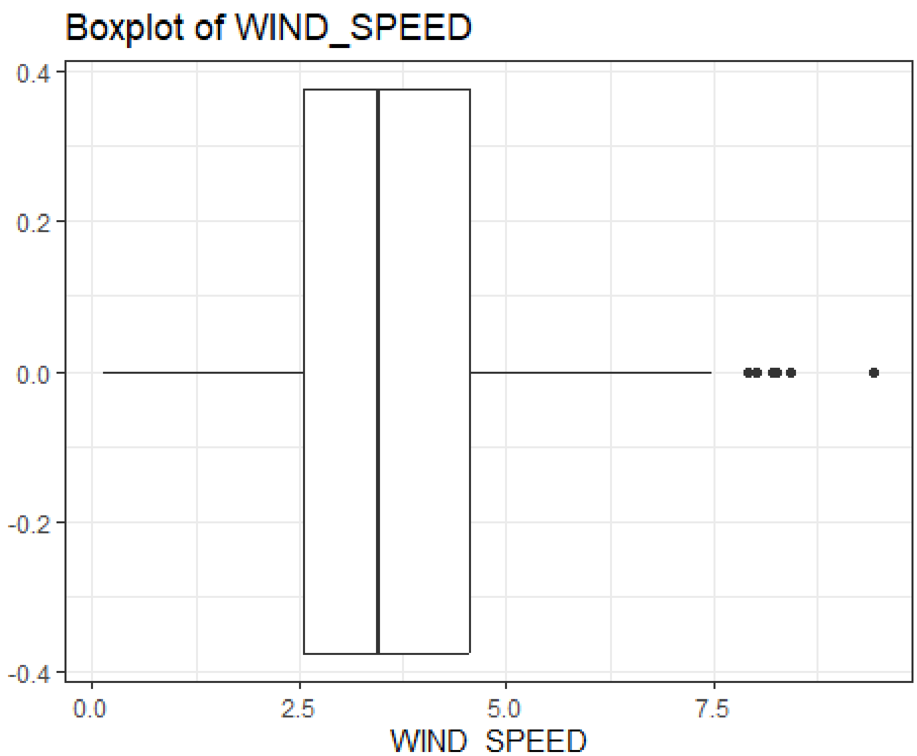

Boxplot of RAINC
0.4
0.2
0.0
-0.2
-0.4
0
5
10
15
20
RAINC


Boxplot of RAINNC
0.4
0.2
0.0
-0.2
-0.4
0
2
4
6
8
RAINNC


Boxplot of SNOW
0.4
0.2
0.0
-0.2
-0.4
-0.050
-0.025
0.000
0.025
0.05
SNOW


Boxplot of TSLB
0.4
0.2
0.0
-0.2
-0.4
280
285
290
295
TSLB


Boxplot of SMOIS
0.4
0.2
0.0
-0.2
-0.4
0.275
0.300
0.325
0.350
SMOIS



1.6.2 This chuck code iterates through numeric columns.

For each column, it creates a ggplot object to visualize the distribution using a boxplot.

These Boxplots help to identify potential outliers, which are data points that fall outside the expected range.

Boxplot for Outliers:

Plot is designed to identify outliers across four different variables: TSK, U10, Q2, and SMOIS.

The plot indicates that TSK and U10 have outliers, as shown by the marks above the main line of the plot.

Q2 and SMOIS do not show any outliers in this representation.

```
### Identify Outliers
```

1.6.3 Identify Outliers

```
# Identify outliers with z-scores (more than 3 standard deviations from the mean)
outliers <- lapply(data1[, sapply(data1, is.numeric)], function(x) {
abs_z_scores <- abs((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
return(which(abs_z_scores > 3))
})
```

```r
# Report columns containing outliers based on z-scores
outlier_columns <- names(data1)[sapply(data1, is.numeric)][sapply(outliers,
length) > 0]
if (length(unlist(outliers)) > 0) {
cat("Potential outliers identified in columns:", outlier_columns, "\n")
} else {
cat("No potential outliers identified based on z-scores.\n")
}

## Potential outliers identified in columns: V10 RAINC RAINNC SMOIS
WIND_SPEED
```

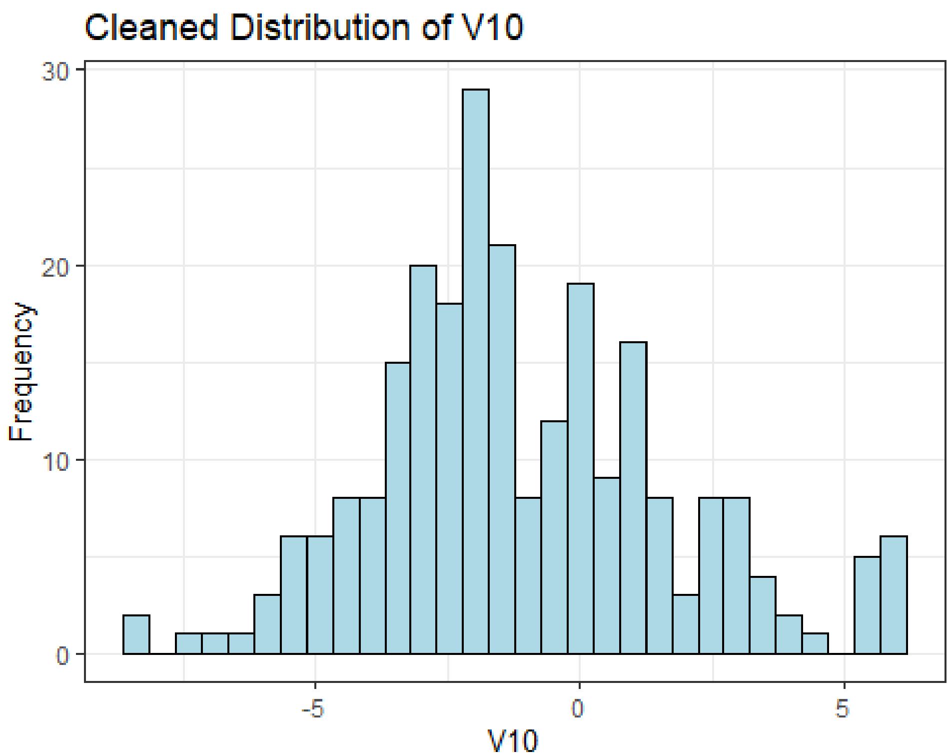
1.7 Handling and resolving outliers

```r
# Display values for all outliers except for RAINC and RAINNC
outlier_values <- list()
for (var in outlier_columns) {
if (!(var %in% c("RAINC", "RAINNC"))) {
# Calculate the bounds for outliers
lower_bound <- quantile(data1[[var]], probs = 0.25, na.rm = TRUE) - 1.5 *
IQR(data1[[var]], na.rm = TRUE)
upper_bound <- quantile(data1[[var]], probs = 0.75, na.rm = TRUE) + 1.5 *
IQR(data1[[var]], na.rm = TRUE)
# Identify and store the outlier values
outlier_values[[var]] <- data1[[var]][data1[[var]] < lower_bound |
data1[[var]] > upper_bound]
}
}
```

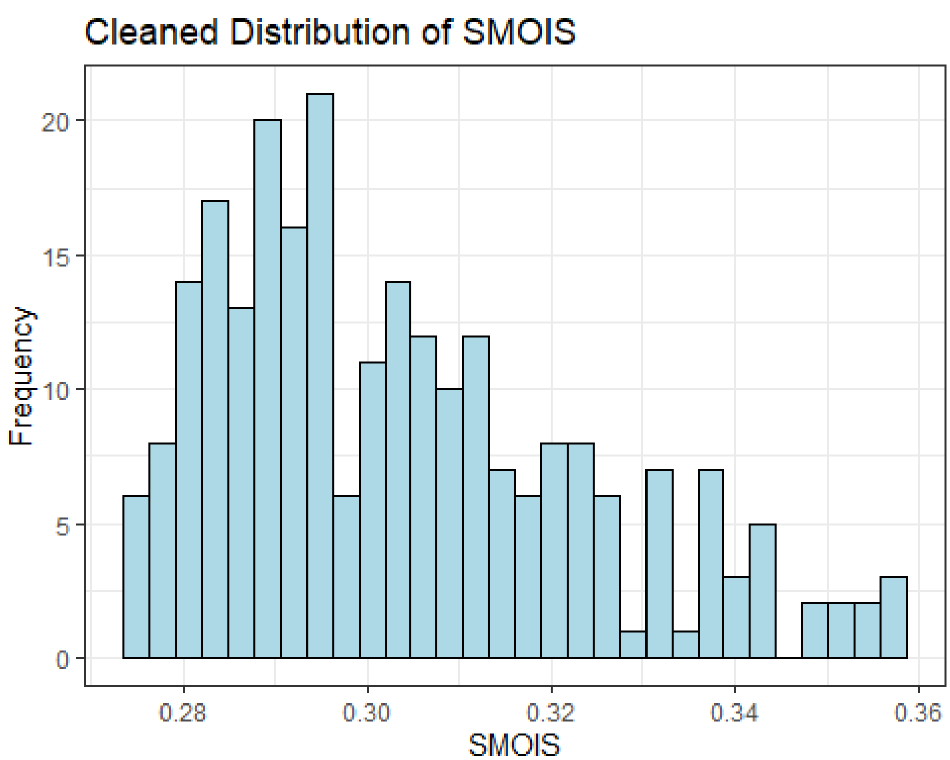
1.8 Applying mutation (Cap) to handle outliers

```r
# Cap the outliers for the same columns
data1_clean <- data1
for (var in outlier_columns) {
if (!(var %in% c("RAINC", "RAINNC"))) {
lower_bound <- quantile(data1_clean[[var]], probs = 0.25, na.rm = TRUE) - 1.5
* IQR(data1_clean[[var]], na.rm = TRUE)
upper_bound <- quantile(data1_clean[[var]], probs = 0.75, na.rm = TRUE) + 1.5
* IQR(data1_clean[[var]], na.rm = TRUE)
# Cap the values
data1_clean[[var]] <- pmin(pmax(data1_clean[[var]], lower_bound),
upper_bound)
}
}

# Verify the results after handling outliers
for (var in outlier_columns) {
if (!(var %in% c("RAINC", "RAINNC"))) {
print(
ggplot(data1_clean, aes(x = .data[[var]])) +
geom_histogram(bins = 30, color = "black", fill = "lightblue") +
labs(title = paste("Cleaned Distribution of", var), x = var, y = "Frequency")
```

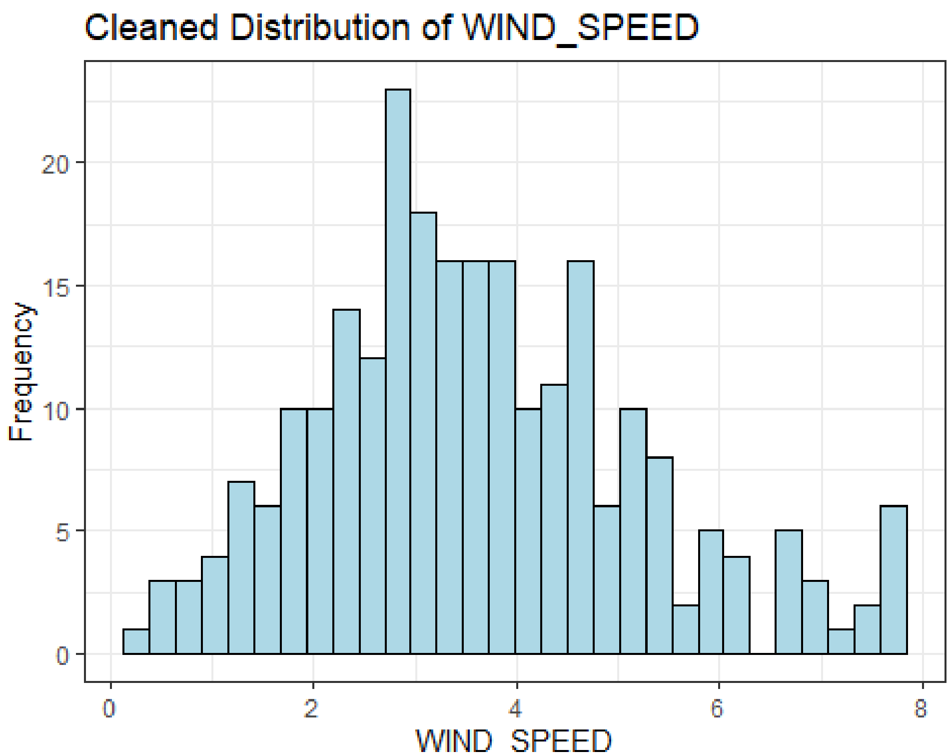
```
+
theme_bw()
)
}
}
```



Cleaned Distribution of V10
Frequency
V10
30
20
10
0
-5
0
5



Cleaned Distribution of SMOIS
Frequency
SMOIS
20
15
10
5
0
0.28
0.30
0.32
0.34
0.36



Data Points:

A Normal Q-Q plot assesses whether a set of data plausibly follows a theoretical distribution, such as the normal distribution.A Normal Q-Q plot assesses whether a set of data plausibly follows a theoretical distribution, such as the normal distribution.

The points on the graph represent the quantiles of your data. If the sample distribution matches the theoretical normal distribution, the points should form a roughly straight line.

In this case, the points appear close to the line, indicating that the data may follow a normal distribution.

1.9 Research Questions

Uni-variate Analysis :

a. Analyze the distribution of skin temperature ('TSK')

b. Analyze the distribution of wind Speed ('WIND_SPEED')

Bivariate Analysis

Questions:

1. How does PSFS differ against daytime 6am to 6pm and nighttime?

Null hypothesis (H_0) assumes that the data comes from a normal distribution.

Alternative hypothesis (H_1) posits that the data does not follow a normal distribution

2. What is the strength and direction of the linear association between wind speed and surface pressure?

Multivariate Analysis

3. How are skin temperature (TSK), specific humidity (Q2), SMOIS, and soil temperature (TSLB) interrelated?

Machine Learning Modeling:

4. What machine learning models effectively predict future wind speed and temperature in Cuxton based on historical data patterns for the next one month?

Time-Series Questions:

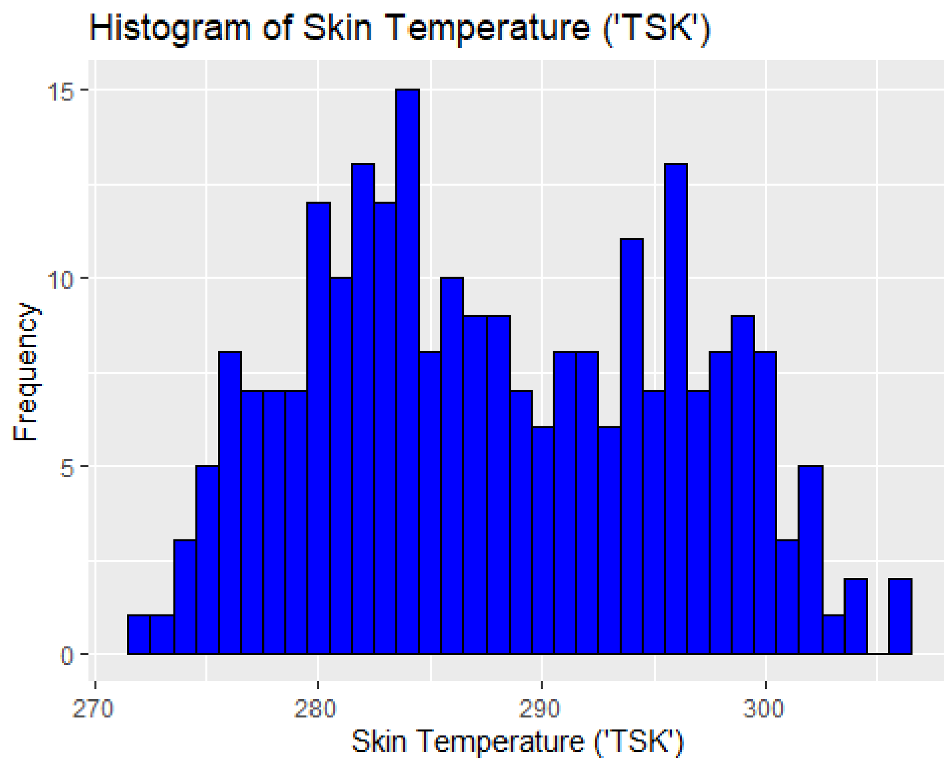
5. How will the average daily temperature at soil bottom (TSLB) change over the next month in Cuxton?

6. What are the expected trends in wind speed during daytime hours in Cuxton over the coming month?

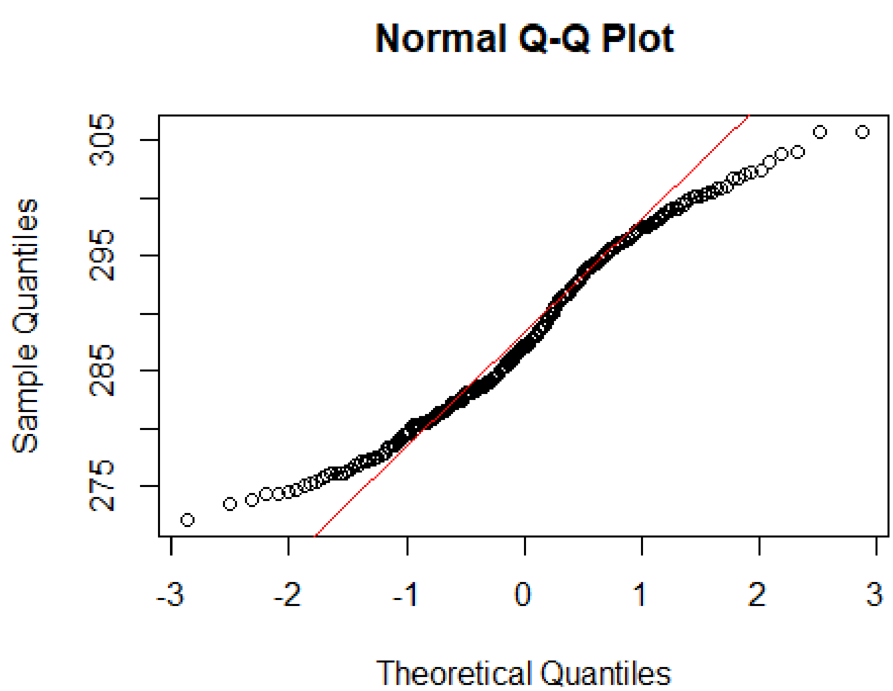
Answers to Research Questions

1.9.1 Statistical Questions : Uni-variate

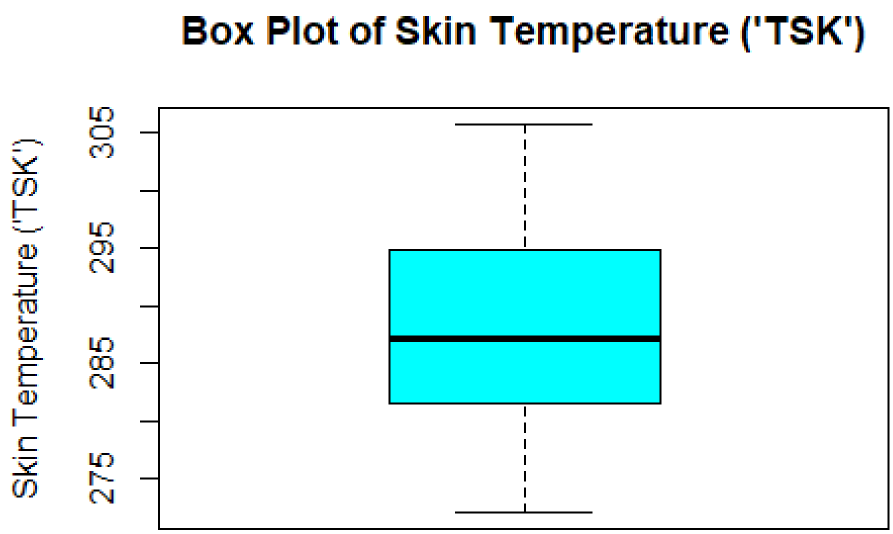
```r
data_df <- data1_clean
# Select tsk column from the dataframe
tsk_data <- data_df$TSK

# Histogram for distribution
ggplot(data = data.frame(tsk_data), aes(x = tsk_data)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Histogram of Skin Temperature ('TSK')",
       x = "Skin Temperature ('TSK')",
       y = "Frequency")
```




```r
# QQ Plot for checking normality
qqnorm(tsk_data)
qqline(tsk_data,col="red")
```



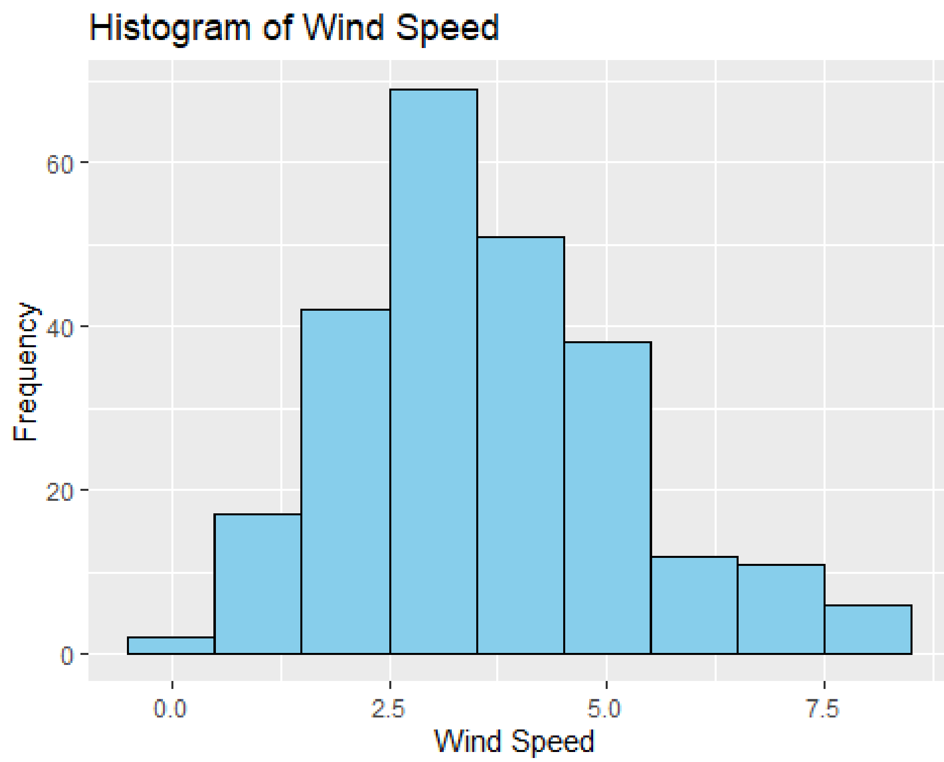
```r
# Box plot for observing outliers and quartiles
boxplot(tsk_data,
        main = "Box Plot of Skin Temperature ('TSK')",
        ylab = "Skin Temperature ('TSK')",
        col = "cyan")
```



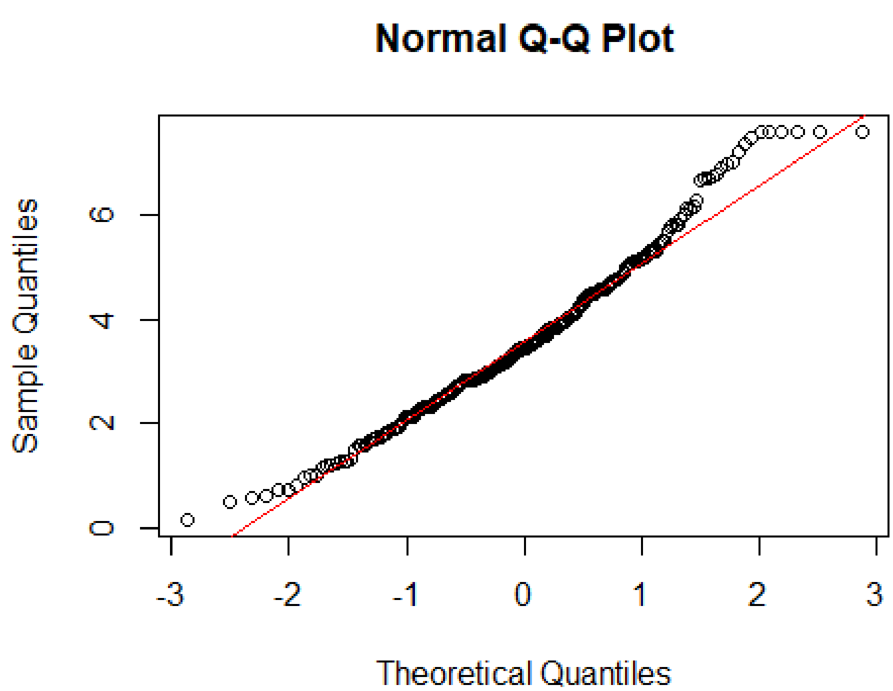
Histogram graph: representing the distribution of skin temperature (labeled as "TSK") The varying heights of the bars in the histogram suggest different frequencies for various skin temperature ranges. This visual representation could help and relevant for studies in physiology or assessing the impact of climate on human health.

Wind speed distribution

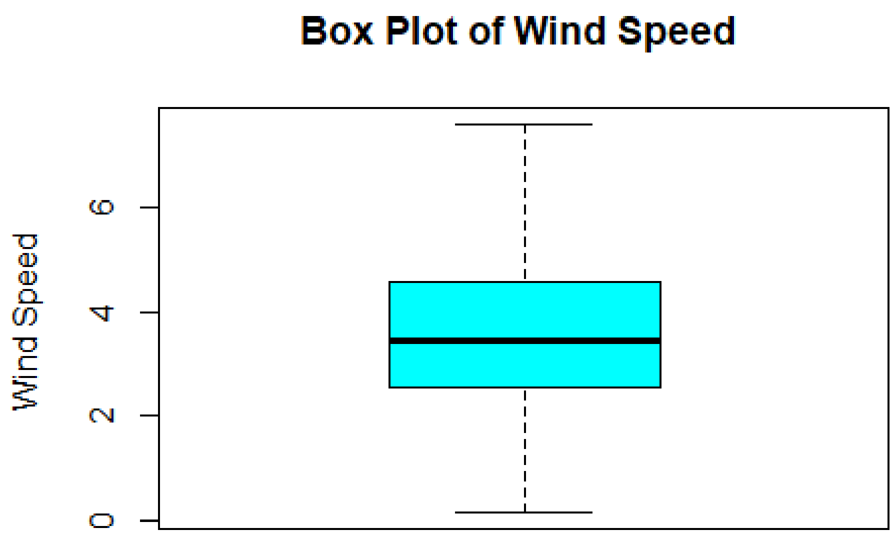
```r
### **Wind speed distributio**
# Select tsk column from the dataframe
wind_speed_data <- data_df$WIND_SPEED

# Histogram for distribution
ggplot(data = data.frame(wind_speed_data), aes(x = wind_speed_data)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Wind Speed ",
       x = "Wind Speed ",
       y = "Frequency")
```




```r
# QQ Plot for checking normality
qqnorm(wind_speed_data)
qqline(wind_speed_data,col="red")
```



```
# Box plot for observing outliers and quartiles
boxplot(wind_speed_data,
        main = "Box Plot of Wind Speed ",
        ylab = "Wind Speed ",
        col = "cyan")
```



Box Plot for 'WIND_SPEED': Box plot shows quartiles, median, and potential outlines for wind speed. Outlines are represented as individual points beyond the whiskers which is not present.

1.10 Question Univariate (Result and analysis)

1.10.1 Output Breakdown

Skin Temperature ('TSK'):

Histogram: The histogram displays the distribution of skin temperature ('TSK') values. The x-axis represents different skin temperature bins, and the y-axis represents the frequency (number of occurrences) of each bin. The blue bars indicate how many data points fall within each temperature range.

QQ Plot (Quantile-Quantile Plot): The QQ plot compares the quantiles of the observed skin temperature data against those expected from a standard normal distribution. The red reference line represents a normal distribution. Deviations from the line suggest departures from normality.

Box Plot: The box plot shows the quartiles (25th, 50th, and 75th percentiles) of the skin temperature data.

The central box represents the interquartile range (IQR), with the median (50th percentile) indicated by the horizontal line inside the box. Outliers (individual data points beyond the whiskers) are shown as individual points.

Wind Speed ('WIND_SPEED'): Histogram: The histogram displays the distribution of wind speed values. Similar to the 'TSK' histogram, the x-axis represents different wind speed bins, and the y-axis represents frequency.

QQ Plot for 'WIND_SPEED': The QQ plot assesses whether wind speed follows a normal distribution. Points close to the red reference line indicate normality.

Box Plot for 'WIND_SPEED': Similar to the 'TSK' box plot, this plot shows quartiles, median, and potential outliers for wind speed. Outliers are represented as individual points beyond the whiskers which is not present.

1.11 Question 1: Biviant analysis (How PSFS differ against daytime 6am to 6pm and nighttime)

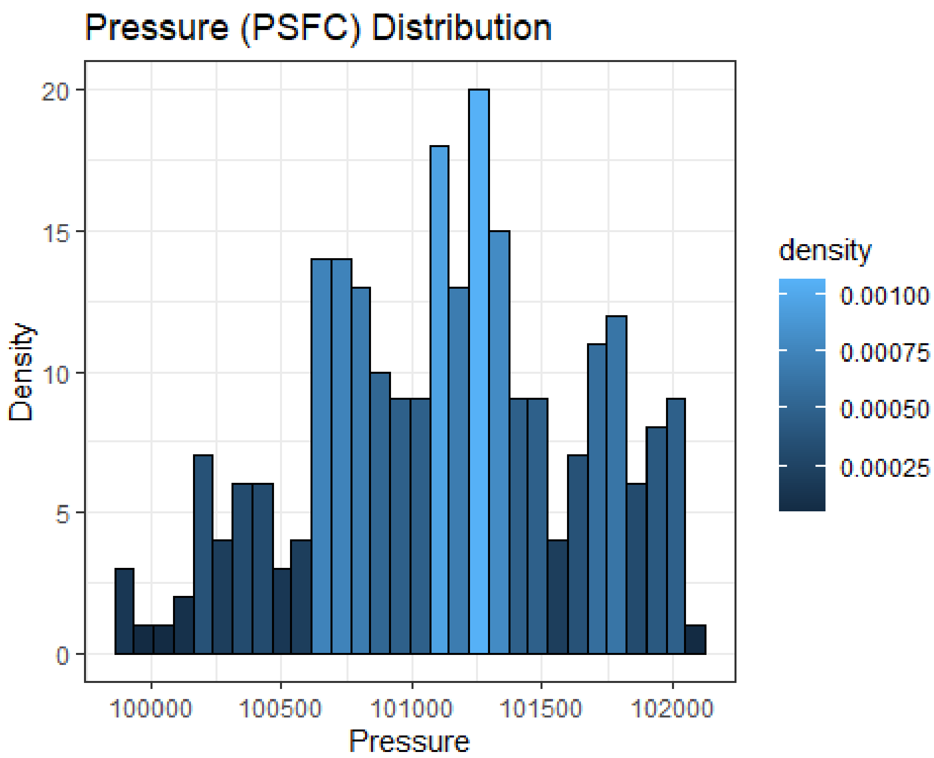
```
# Que1: How does the mean Pressure (PSFC) differ between daytime

data_df <- data1_clean

# Check for missing values
any_missing <- anyNA(data_df)
if (any_missing) {
  cat("There are missing values in the dataset.\n")
} else {
  cat("No missing values found.\n")
}

## No missing values found.

# Extract hour from DATETIME
data_df$Hour <- as.numeric(format(data_df$DATETIME, "%H"))

# Define a function to categorize daytime and nighttime
day_night_category <- function(hour) {
  if (is.na(hour)) {
    return(NA)  # Return NA if the hour is missing
  } else if (hour >= 6 & hour < 18) {
    return("Daytime")
  } else {
    return("Nighttime")
  }
}

# Add Day/Night category
```

```r
data_df$DayNight <- sapply(data_df$Hour, day_night_category)

# Separate daytime and nighttime data
daytime_data <- subset(data_df, DayNight == "Daytime")
nighttime_data <- subset(data_df, DayNight == "Nighttime")

# Normality tests
ad_test <- ad.test(data_df$PSFC)
shapiro_test <- shapiro.test(data_df$PSFC)

# Print test results
cat("Anderson-Darling Test p-value:", ad_test$p.value, "\n")

## Anderson-Darling Test p-value: 0.03453385

cat("Shapiro-Wilk Test p-value:", shapiro_test$p.value, "\n")

## Shapiro-Wilk Test p-value: 0.002923162

# Nonparametric tests (if normality tests fail)
if (ad_test$p.value < 0.05 | shapiro_test$p.value < 0.05) {
  # Wilcoxon rank-sum test
  wilcox_test <- wilcox.test(daytime_data$PSFC, nighttime_data$PSFC)
  cat("Wilcoxon rank-sum Test p-value:", wilcox_test$p.value, "\n")
} else {
  # T-test (if data is likely normal)
  t_test <- t.test(daytime_data$PSFC, nighttime_data$PSFC)
  cat("T-Test p-value:", t_test$p.value, "\n")
}

## Wilcoxon rank-sum Test p-value: 0.9992937

# Calculate mean pressure for each period
mean_daytime <- mean(daytime_data$PSFC)
mean_nighttime <- mean(nighttime_data$PSFC)

# Print mean values
cat("Mean Pressure (Daytime):", mean_daytime, "\n")

## Mean Pressure (Daytime): 101122.4

cat("Mean Pressure (Nighttime):", mean_nighttime, "\n")

## Mean Pressure (Nighttime): 101123.3

# Perform t-test to compare mean pressure (PSFC) between daytime and
nighttime
t_test_result <- t.test(daytime_data$PSFC, nighttime_data$PSFC)

# Print the result
print(t_test_result)
```

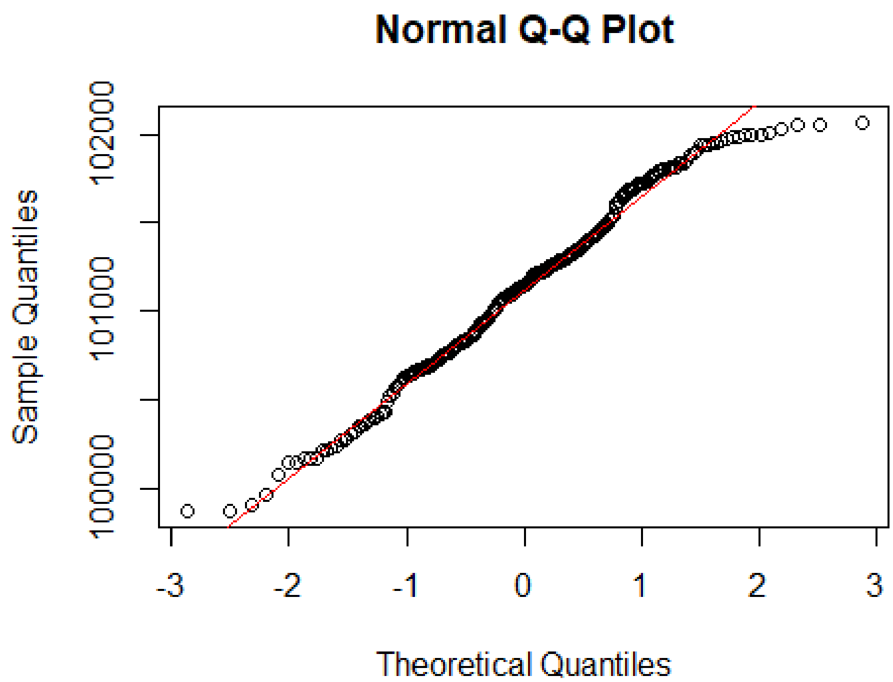
```
##
##  Welch Two Sample t-test
##
## data:  daytime_data$PSFC and nighttime_data$PSFC
## t = -0.014232, df = 245.99, p-value = 0.9887
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -129.8403  127.9774
## sample estimates:
## mean of x mean of y
##  101122.4  101123.3
```

```r
# Create histogram
library(ggplot2)  # Load ggplot2 for aesthetics

ggplot(data_df, aes(x = PSFC)) +
  geom_histogram(bins = 30, aes(fill = ..density..), color = "black") +  #
Colored bars with density shading
  labs(title = "Pressure (PSFC) Distribution", x = "Pressure", y = "Density")
+
  theme_bw()  # Use black and white theme for a clean look
```

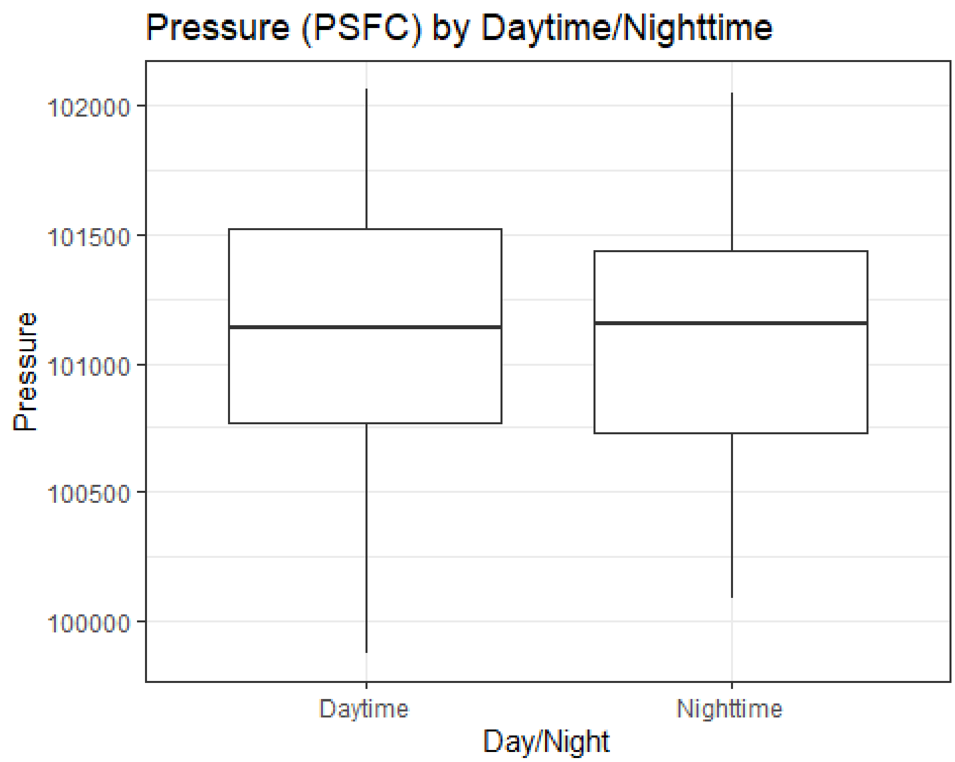
```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2
3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
# Create Q-Q plot
# Add titles and labels

qqnorm(data_df$PSFC)
qqline(data_df$PSFC, col="red")
```




```r
# Boxplots for daytime vs nighttime pressure
ggplot(data_df, aes(x = DayNight, y = PSFC)) +
  geom_boxplot() +
  labs(title = "Pressure (PSFC) by Daytime/Nighttime", x = "Day/Night", y =
"Pressure") +
  theme_bw()
```



1.12 Question 1: (Result and analysis for Bivariate)

1.12.1 Output Breakdown

Normality Tests:

Both Anderson-Darling test and the Shapiro-Wilk test are used to assess whether the data follows a normal distribution.

The null hypothesis for both tests is that the data comes from a normal distribution. The null hypothesis (H_0) assumes that the data comes from a normal distribution.

Alternative Hypothesis: The alternative hypothesis states that the true difference in means is not equal to 0.

The alternative hypothesis (H_1) posits that the data does not follow a normal distribution.

The p-values obtained from these tests help us determine whether we can assume normality.

In this case: Anderson-Darling Test p-value: 0.03453385

Shapiro-Wilk Test p-value: 0.002923162

Since both p-values are less than 0.05, we reject the null hypothesis of normality.

** Null Hypothesis Rejected**

Nonparametric Test (Wilcoxon Rank-Sum Test):

Since the normality tests failed, we use the Wilcoxon rank-sum test (also known as the Mann-Whitney U test).

The Wilcoxon rank-sum test assesses whether the medians of two independent samples are equal.

The p-value obtained from this test is 0.9992937, which is not significant. Therefore, we do not reject the null hypothesis that the medians are equal.

Mean Pressure: Mean Pressure (Daytime): 101122.4 Mean Pressure (Nighttime): 101123.3

The mean pressure values are very close, indicating no substantial difference between daytime and nighttime.

Histogram:

The histogram shows the distribution of pressure ('PSFC') values. The x-axis represents pressure, and the y-axis represents density (frequency).

The bars are colored based on density shading.

Q-Q Plot:

The Q-Q plot compares the quantiles of the observed pressure data against those expected from a standard normal distribution.

The red reference line represents a normal distribution. Deviations from the line suggest departures from normality.

Box Plots:

The box plots compare pressure between daytime and nighttime.

The central box represents the interquartile range (IQR), with the median indicated by the horizontal line inside the box. Outliers are shown as individual points beyond the whiskers, which is not present.

Overall, the mean pressure values are similar, and the nonparametric test does not indicate a significant difference between daytime and nighttime pressures.

Test Statistics:

The t-value obtained from the test is -0.014232. The degrees of freedom (df) are approximately 246.

P-Value: The p-value is 0.9887.

The null hypothesis states that the true difference in means between daytime and nighttime pressures is equal to 0.

Since the p-value is greater than 0.05 (common significance level), we fail to reject the null hypothesis validating nonparametric test earlier down.

Confidence Interval: The 95% confidence interval for the difference in means is (-129.8403, 127.9774).

This interval includes 0, further supporting the lack of significant difference between daytime and nighttime mean pressures.

Sample Estimates: The mean pressure during daytime is approximately 101122.4. The mean pressure during nighttime is approximately 101123.3.

In summary, the t-test results suggest that there is no statistically significant difference between the daytime and nighttime PSFC means, as indicated by the high p-value and the confidence interval that includes zero. The means of the two samples are very close to each other

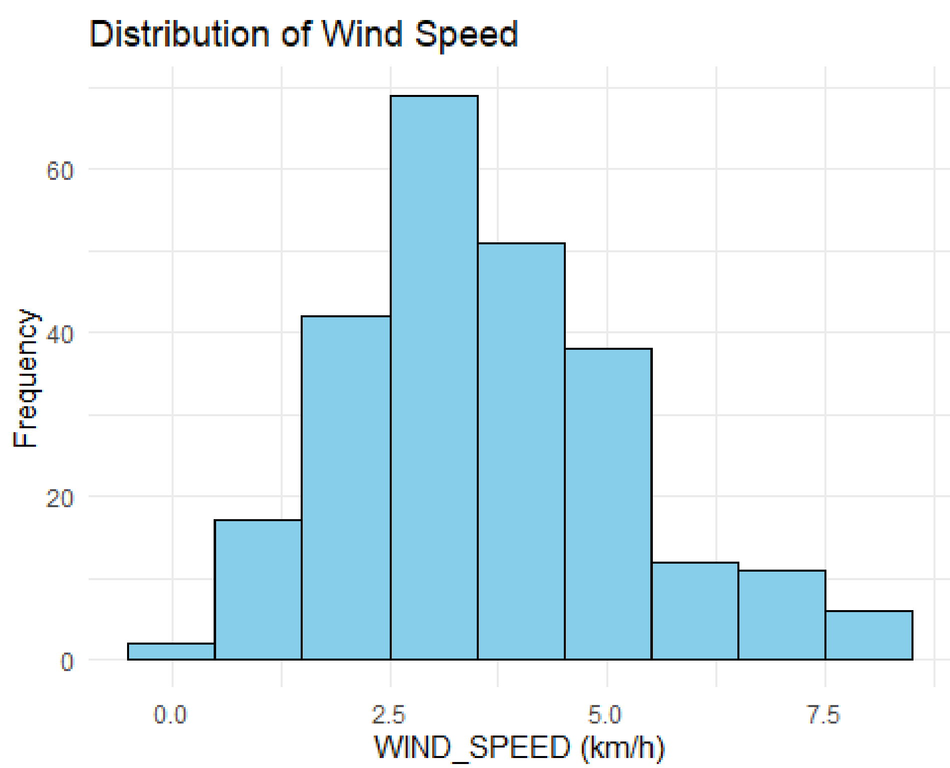
```
#Statistical Questions
```

1.12.2 Statistical Questions 2: How does the mean Wind Speed vary across the recorded period?

```
# Extract date and hour from DATETIME
data_df$Date <- as.Date(data_df$DATETIME)
data_df$Hour <- format(data_df$DATETIME, "%H")

# Check if 'wind_speed' column exists
if ("WIND_SPEED" %in% names(data_df)) {

  # Calculate mean wind speed for each date (assuming 'wind_speed' is the
column)
  mean_wind_speed <- aggregate(WIND_SPEED ~ Date, data = data_df, FUN = mean)
  print(mean_wind_speed)

} else {
  print("Error: 'WIND_SPEED' column not found in the data_df")
}

##          Date WIND_SPEED
## 1  2018-04-30   3.500000
## 2  2018-05-01   5.332031
## 3  2018-05-02   5.575312
## 4  2018-05-03   2.528750
## 5  2018-05-04   1.875000
## 6  2018-05-05   3.252500
## 7  2018-05-06   3.003750
## 8  2018-05-07   2.681250
## 9  2018-05-08   2.687500
## 10 2018-05-09   3.763750
## 11 2018-05-10   4.167500
## 12 2018-05-11   4.543750
```

```
## 13 2018-05-12   1.921250
## 14 2018-05-13   3.063750
## 15 2018-05-14   5.795312
## 16 2018-05-15   4.466250
## 17 2018-05-16   5.512500
## 18 2018-05-17   4.203750
## 19 2018-05-18   2.547500
## 20 2018-05-19   2.323750
## 21 2018-05-20   2.787500
## 22 2018-05-21   3.402500
## 23 2018-05-22   4.070000
## 24 2018-05-23   4.966250
## 25 2018-05-24   4.053750
## 26 2018-05-25   2.018750
## 27 2018-05-26   5.338281
## 28 2018-05-27   2.876250
## 29 2018-05-28   2.818750
## 30 2018-05-29   4.946250
## 31 2018-05-30   2.352500
## 32 2018-05-31   3.094286
```

```
# Plot the distribution of wind speed
ggplot(data_df, aes(x = WIND_SPEED)) +
geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
labs(title = "Distribution of Wind Speed", x = "WIND_SPEED (km/h)", y =
"Frequency") +
theme_minimal()
```



- ** Wind_Speed histogram** indicated that the tallest bar is between 2.5 and 3.75 km/h, indicating that this wind speed range occurs most frequently.

  The histogram shows the distribution of wind speeds, which could be relevant for weather analysis.

1.13 Question 2 (Result and analysis)

Output breakdown:

Range: The wind speeds range from a **low of 1.875** to a **high of 5.795312**.

Central Tendency: There seems to be a central clustering around the 3 to 4 wind speed mark, as many values fall within this range.

Variability: There are some variability in the data, with wind speeds occasionally reaching above 5 and dropping below 2.

Distribution: A unimodal distribution with a single peak around the 3 to 4 wind speed mark.

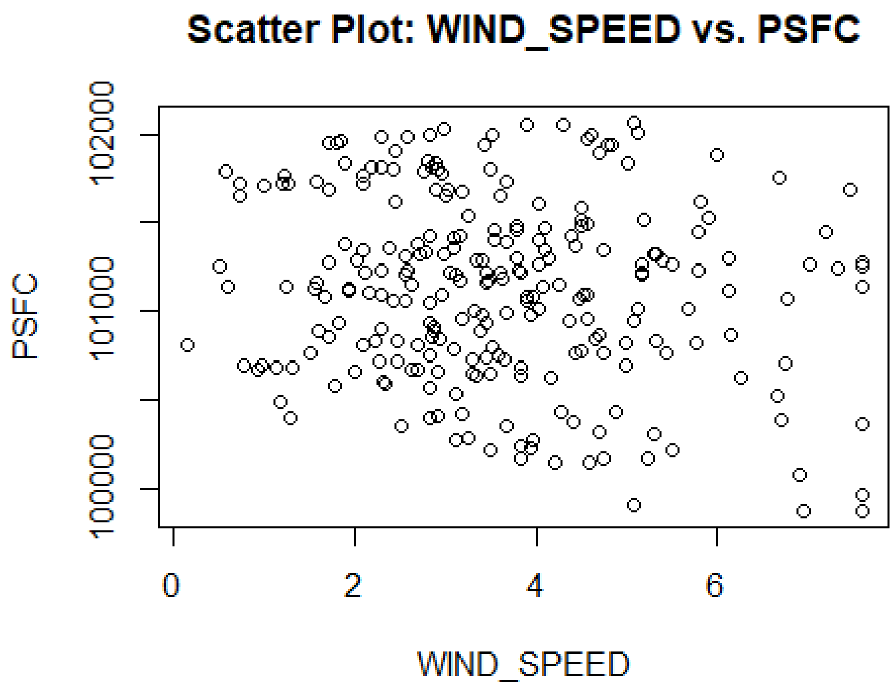
- The distribution is not perfectly symmetrical, suggesting a slight skewness to the left with the frequency of higher and lower wind speeds.
- The presence of occasional higher values (like 5.795312) might create a longer tail on one end of the distribution.

1.14 Question 3 (Bivariate Analysis)

1.14.1 What is the strength and direction of the linear association between wind speed and surface pressure?

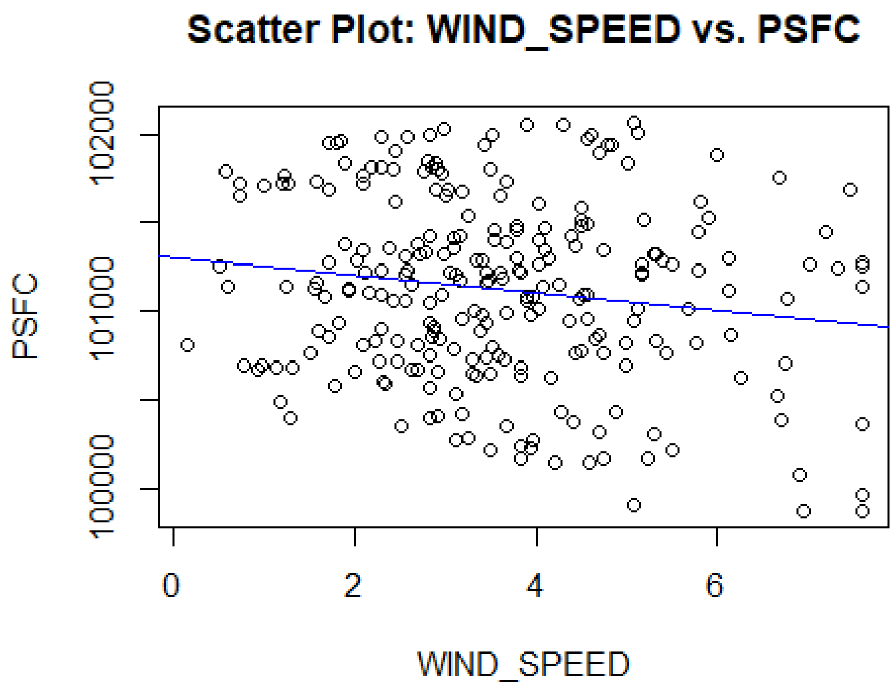
Hypothesis: The null hypothesis is that there is no linear relationship between WIND_SPEED and PSFC

```r
# Perform a scatter plot
plot(data_df$WIND_SPEED, data_df$PSFC, main="Scatter Plot: WIND_SPEED vs. PSFC", xlab="WIND_SPEED", ylab="PSFC")
```



Scatter Plot: WIND_SPEED vs. PSFC WIND_SPEED," ranging from 0 to 6. The vertical axis represents "PSFC," with values between approximately 10,000 and just over 10,300 indicates a weak negative correlation between wind speed (WIND_SPEED) and surface pressure (PSFC).

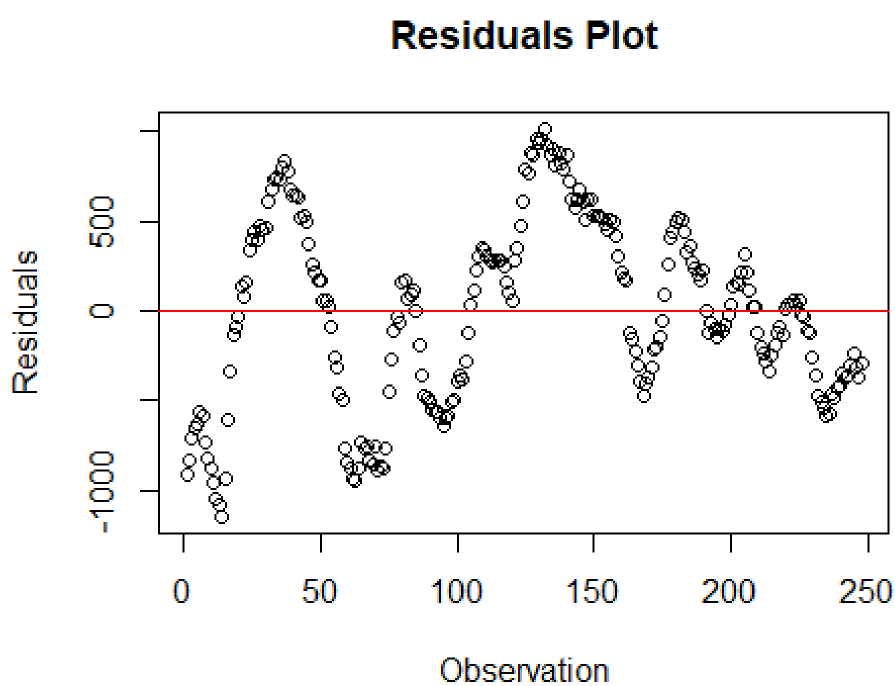
```
# Calculate the correlation coefficient
cor(data_df$WIND_SPEED, data_df$PSFC)

## [1] -0.1573919

# Scatter plot with regression line
plot(data_df$WIND_SPEED, data_df$PSFC, main="Scatter Plot: WIND_SPEED vs.
PSFC", xlab="WIND_SPEED", ylab="PSFC")
abline(lm(data_df$PSFC ~ data_df$WIND_SPEED), col="blue") # Adds a blue
regression line
```



Result: Analyses of how wind speed correlates with surface pressure (PSFC) Mean PSFC Line: A horizontal line is drawn across the scatter plot at what appears to be the mean value of “PSFC.” This mean value does not show or vary significantly with changes in wind speed.

```
#Perform a Linear Regression Analysis:

# Linear regression analysis
regression_model <- lm(PSFC ~ WIND_SPEED, data=data_df)

# Summary of the regression model
summary(regression_model)

##
## Call:
## lm(formula = PSFC ~ WIND_SPEED, data = data_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1147.37  -369.50    24.39   392.31  1016.13
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 101305.19      79.78  1269.9   <2e-16 ***
## WIND_SPEED     -50.46      20.18    -2.5   0.0131 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 508.9 on 246 degrees of freedom
## Multiple R-squared:  0.02477,    Adjusted R-squared:  0.02081
## F-statistic: 6.249 on 1 and 246 DF,  p-value: 0.01308

# Check for Residual Normality:
# To ensure the assumptions of linear regression are met, check the residuals
for normality:

# Check residuals for normality
residuals_normality <- lillie.test(residuals(regression_model))

# Print the result
print(residuals_normality)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  residuals(regression_model)
## D = 0.04392, p-value = 0.2891

# Plot residuals
plot(residuals(regression_model), type="p", main="Residuals Plot",
xlab="Observation", ylab="Residuals")
abline(h=0, col="red") # Adds a horizontal line at 0
```



Residuals Plot

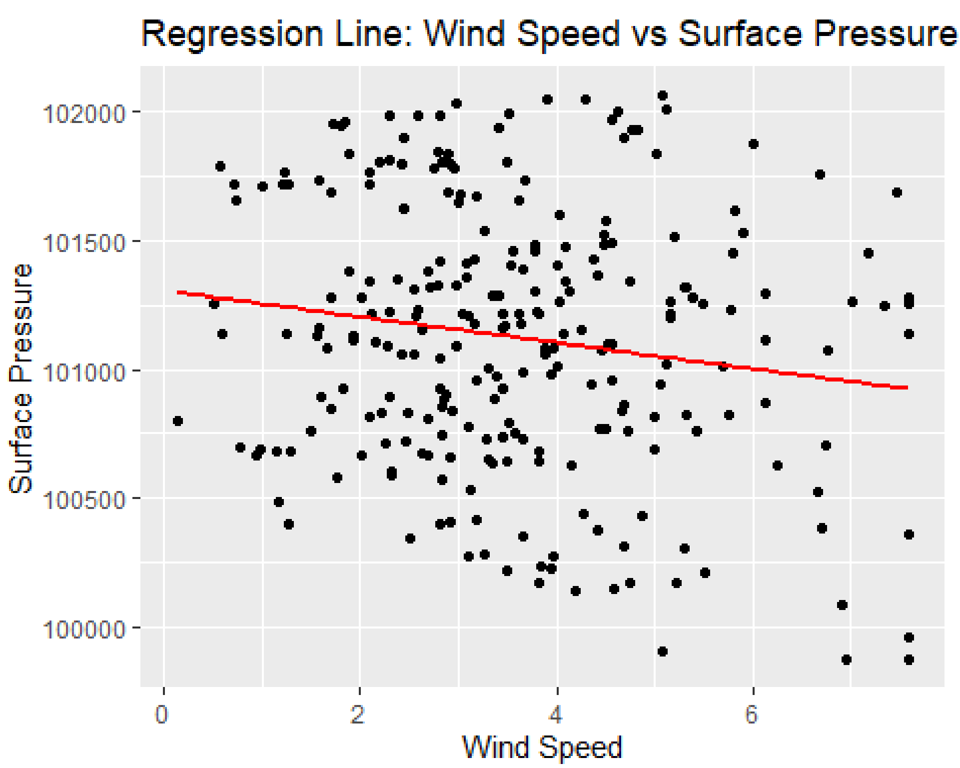
Residuals represent the difference between the actual observed values and the predicted values from a regression model. The points are scattered around a red horizontal line at zero, which represents where residuals would be if predictions were perfect (no errors).

As wind speed increases, surface pressure tends to decrease (and vice versa). However, the small magnitude suggests that other factors play a more significant role in determining surface pressure

```r
view(data_df)
# Extract variables
wind_speed <- data_df$WIND_SPEED
surface_pressure <- data_df$PSFC

# Step 1: Check for linearity - Scatter plot
scatter_plot <- ggplot(data_df, aes(x = WIND_SPEED, y = PSFC)) +
  geom_point() +
  labs(title = "Scatter Plot of Wind Speed vs Surface Pressure",
       x = "Wind Speed",
       y = "Surface Pressure")

# Step 2: Test for normality of errors - Lilliefors test
regression_model <- lm(PSFC ~ WIND_SPEED, data = data_df)
residuals <- residuals(regression_model)
lillie.test(residuals)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  residuals
## D = 0.04392, p-value = 0.2891

# Step 3: Plot residuals
residual_plot <- ggplot(data_df, aes(x = PSFC, y = residuals)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  labs(title = "Residual Plot",
       x = "Surface Pressure",
       y = "Residuals")

# Step 4: Define null and alternative hypotheses
# Null Hypothesis (H0): There is no linear association between wind speed and
surface pressure.
# Alternative Hypothesis (H1): There is a linear association between wind
speed and surface pressure.

# Step 5: Perform linear regression analysis
summary(regression_model)
```

```
##
## Call:
## lm(formula = PSFC ~ WIND_SPEED, data = data_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1147.37  -369.50    24.39   392.31  1016.13
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 101305.19      79.78  1269.9   <2e-16 ***
## WIND_SPEED     -50.46      20.18    -2.5   0.0131 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 508.9 on 246 degrees of freedom
## Multiple R-squared:  0.02477,    Adjusted R-squared:  0.02081
## F-statistic: 6.249 on 1 and 246 DF,  p-value: 0.01308
```

```r
# Step 6: Plot the regression line and visualize the relationship
ggplot(data_df, aes(x = WIND_SPEED, y = PSFC)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Regression Line: Wind Speed vs Surface Pressure",
       x = "Wind Speed",
       y = "Surface Pressure")
```



```r
# Step 7: Plot histogram of residuals to check for normality
residuals_histogram <- ggplot(data_df, aes(x = residuals)) +
  geom_histogram(fill = "skyblue", color = "black", bins = 20) +
  labs(title = "Histogram of Residuals",
       x = "Residuals",
       y = "Frequency")
```

Relationship between **wind speed** and **surface pressure**.

Regression Line: A red line runs through the data points, indicating the **regression line** that models the relationship between wind speed and surface pressure.

1.15 Question 3 (Result and analysis)

Output breakdown:

Test was applied to the residuals of a linear regression model

The correlation coefficient between wind speed and pressure (PSFC) in the dataset is -0.1573919. This indicates a slight negative correlation, meaning that as wind speed increases, pressure tends to decrease, but not very strongly.

Normality test on the residuals of your regression model are as follows:

D = 0.04392

p-value = 0.2891

Test result shows that the residuals from regression model are normally distributed. since the p-value is greater than 0.05.

This means that the normalcy **null hypothesis** is not rejected by the test, a strong sign that the linear regression's presumptions are being satisfied.

- Residuals: These are the differences between the observed values and the values predicted by your regression model.
- Horizontal Line at 0: This line represents the point where the predicted values perfectly match the observed values.
- Randomness: Residuals should be randomly distributed and show no clear pattern. This suggests that the model is capturing the relationship well.

```r
# Multivariate Analysis
```

1.16 Question 4 (Multivariate Analysis)

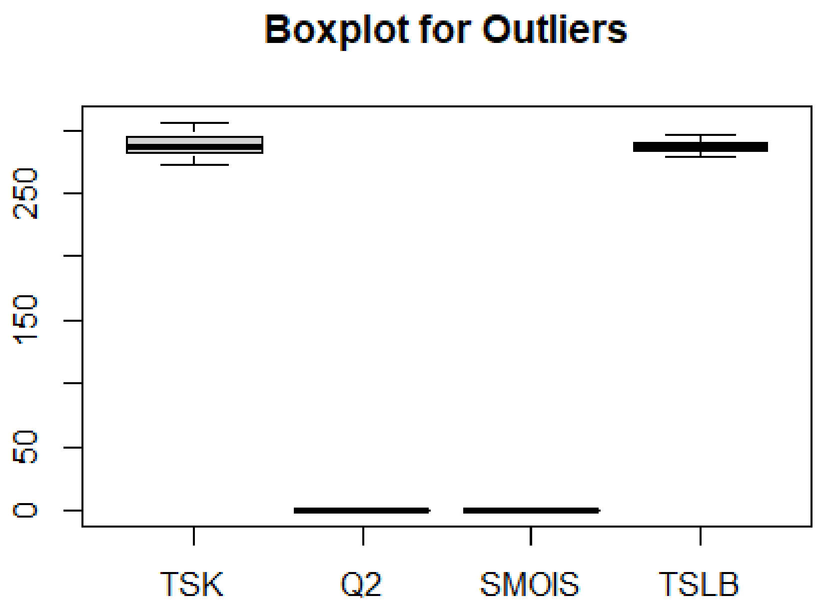
1.16.1 Question: How are skin temperature (TSK), specific humidity (Q2),soil moisture (SMOIS) and soil temperature (TSLB) interrelated?

Hypothesis: The null hypothesis is that there is no significant relationship among these four variables.

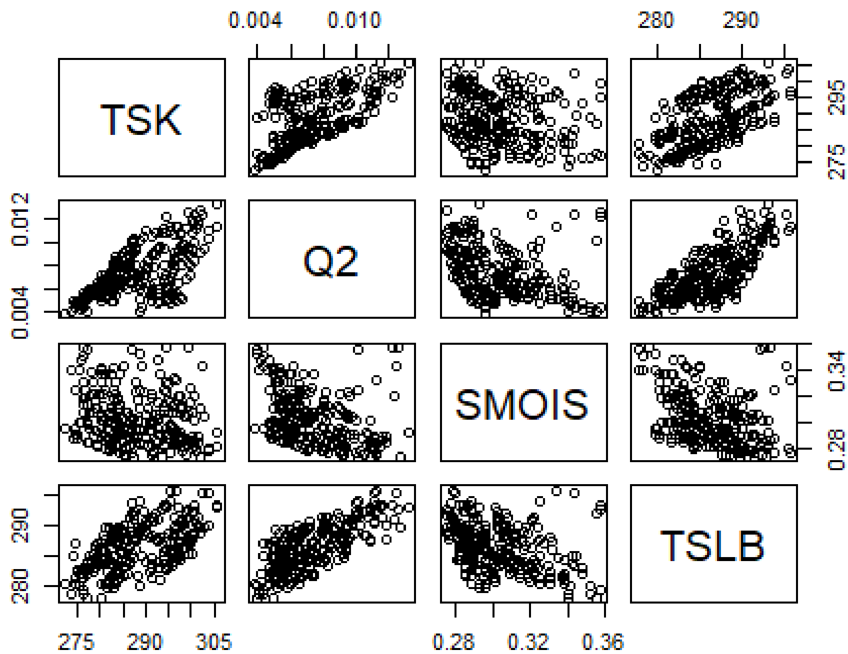
```r
#  Check for NA and outliers
summary(data_df)# summary table provides descriptive statistics for each variable in your dataset.
```

```
##     DATETIME                          TSK             PSFC
##  Min.   :2018-05-01 00:00:00   Min.   :272.1   Min.   : 99872
##  1st Qu.:2018-05-08 17:15:00   1st Qu.:281.6   1st Qu.:100761
##  Median :2018-05-16 10:30:00   Median :287.1   Median :101145
##  Mean   :2018-05-16 10:30:00   Mean   :288.0   Mean   :101123
##  3rd Qu.:2018-05-24 03:45:00   3rd Qu.:294.9   3rd Qu.:101474
##  Max.   :2018-05-31 21:00:00   Max.   :305.8   Max.   :102065
##       U10                V10                Q2                RAINC
##  Min.   :-7.1000   Min.   :-8.200   Min.   :0.003870   Min.   : 0.0000
##  1st Qu.:-2.3000   1st Qu.:-2.900   1st Qu.:0.005680   1st Qu.: 0.0000
##  Median :-0.5750   Median :-1.550   Median :0.006860   Median : 0.0000
##  Mean   :-0.4601   Mean   :-1.084   Mean   :0.007379   Mean   : 0.5552
##  3rd Qu.: 1.5000   3rd Qu.: 0.700   3rd Qu.:0.008985   3rd Qu.: 0.0000
##  Max.   : 5.2000   Max.   : 6.100   Max.   :0.013200   Max.   :20.3000
##      RAINNC             SNOW        TSLB           SMOIS
WIND_SPEED
##  Min.   :0.0000   Min.   :0   Min.   :277.8   Min.   :0.2750   Min.
:0.140
##  1st Qu.:0.0000   1st Qu.:0   1st Qu.:283.5   1st Qu.:0.2883   1st
Qu.:2.540
##  Median :0.0000   Median :0   Median :286.3   Median :0.3000   Median
:3.450
##  Mean   :0.2117   Mean   :0   Mean   :286.4   Mean   :0.3036   Mean
:3.614
##  3rd Qu.:0.0000   3rd Qu.:0   3rd Qu.:289.1   3rd Qu.:0.3159   3rd
Qu.:4.562
##  Max.   :7.9000   Max.   :0   Max.   :295.8   Max.   :0.3572   Max.
:7.596
##      Hour               DayNight              Date
##  Length:248         Length:248         Min.   :2018-04-30
##  Class :character   Class :character   1st Qu.:2018-05-08
##  Mode  :character   Mode  :character   Median :2018-05-16
##                                        Mean   :2018-05-15
##                                        3rd Qu.:2018-05-24
##                                        Max.   :2018-05-31
```

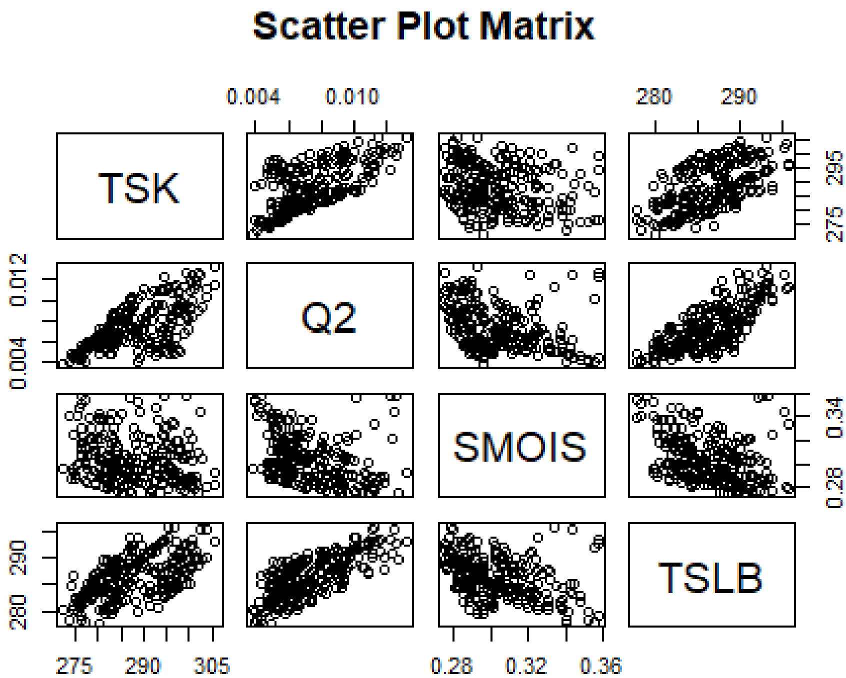
```r
boxplot(data_df[, c("TSK", "Q2", "SMOIS", "TSLB")], main = "Boxplot for Outliers") #Boxplot shows the distribution of the variables TSK, Q2, SMOIS, and TSLB
```




```r
# MVNORM test and plot
# MVNORM test checks if the data follows a multivariate normal distribution.
library(mvnormtest, lib.loc = character())  # Check and load if needed

# Select relevant columns
data_subset <- data_df[, c("TSK", "Q2", "SMOIS", "TSLB")]
plot(data_subset)
```



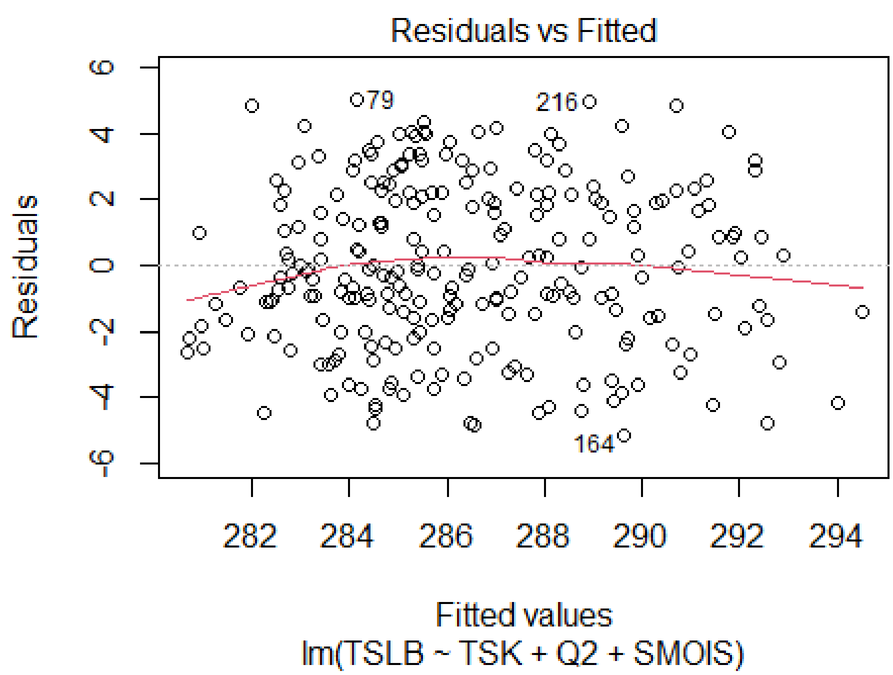

```r
# 5. ANOVA test and plot
anova_model <- aov(cbind(TSK, Q2, SMOIS, TSLB) ~ 1, data = data_df)
summary(anova_model)

##  Response TSK :
##                Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   247  15933  64.504
##
##  Response Q2 :
##                Df    Sum Sq    Mean Sq F value Pr(>F)
## Residuals   247 0.0011095 4.4917e-06
##
##  Response SMOIS :
##                Df   Sum Sq    Mean Sq F value Pr(>F)
## Residuals   247 0.097149 0.00039332
##
##  Response TSLB :
##                Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   247 3696.7  14.966

# The ANOVA Pr(>F) column indicates the p-value. In this case, none of the
variables show significant differences

# 6. Plot linear relationship with scatter plot
pairs(data_df[, c("TSK", "Q2", "SMOIS", "TSLB")], main = "Scatter Plot
Matrix") #visualize relationships and patterns between variables.
```



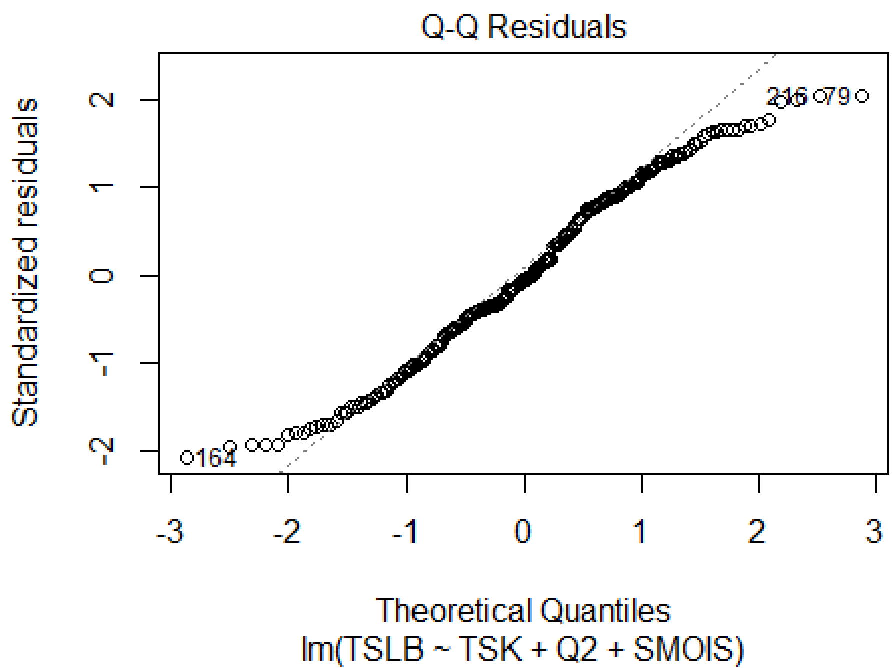

```r
# 7. Linear regression
lm_model <- lm(TSLB ~ TSK + Q2 + SMOIS, data = data_df)
summary(lm_model)

##
## Call:
## lm(formula = TSLB ~ TSK + Q2 + SMOIS, data = data_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1439 -1.6787 -0.1417  2.0663  5.0292
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  255.88744    7.39897  34.584  < 2e-16 ***
## TSK            0.10057    0.02555   3.936 0.000108 ***
## Q2          1050.62396  101.37061  10.364  < 2e-16 ***
## SMOIS        -20.50140    8.55131  -2.397 0.017262 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.492 on 244 degrees of freedom
## Multiple R-squared:   0.59,  Adjusted R-squared:  0.585
## F-statistic: 117.1 on 3 and 244 DF,  p-value: < 2.2e-16

#Multiple R-squared:
# The multiple R-squared (often denoted as (R^2)) measures the proportion of
```

```r
variance in the response variable, in this case, (TSLB) that can be explained
by the predictor variables (TSK, Q2, and SMOIS).
# the multiple R-squared is approximately 0.59, which means that about 59% of
the variability in the TSLB values can be explained by the combination of
TSK, Q2, and SMOIS


#The residual standard error (approximately 2.492) provides an estimate of
the typical deviation of the residuals.
# The residual plots (linearity, normality, and variance) help assess the
assumptions of the linear regression model.

#All VIF values are close to 1, indicating that multicollinearity is not a
major concern in your model



# 8. Multiple correlation
cor(data_df[, c("TSK", "Q2", "SMOIS", "TSLB")])

##              TSK         Q2      SMOIS       TSLB
## TSK    1.0000000  0.6343829 -0.2025451  0.5952099
## Q2     0.6343829  1.0000000 -0.3534281  0.7451657
## SMOIS -0.2025451 -0.3534281  1.0000000 -0.3508099
## TSLB   0.5952099  0.7451657 -0.3508099  1.0000000

# 9. Multiple regression
mlr_model <- lm(TSLB ~ TSK + Q2 + SMOIS, data = data_df)
summary(mlr_model)

##
## Call:
## lm(formula = TSLB ~ TSK + Q2 + SMOIS, data = data_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1439 -1.6787 -0.1417  2.0663  5.0292
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  255.88744    7.39897  34.584  < 2e-16 ***
## TSK            0.10057    0.02555   3.936 0.000108 ***
## Q2          1050.62396  101.37061  10.364  < 2e-16 ***
## SMOIS        -20.50140    8.55131  -2.397 0.017262 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.492 on 244 degrees of freedom
## Multiple R-squared:   0.59,  Adjusted R-squared:  0.585
## F-statistic: 117.1 on 3 and 244 DF,  p-value: < 2.2e-16
```

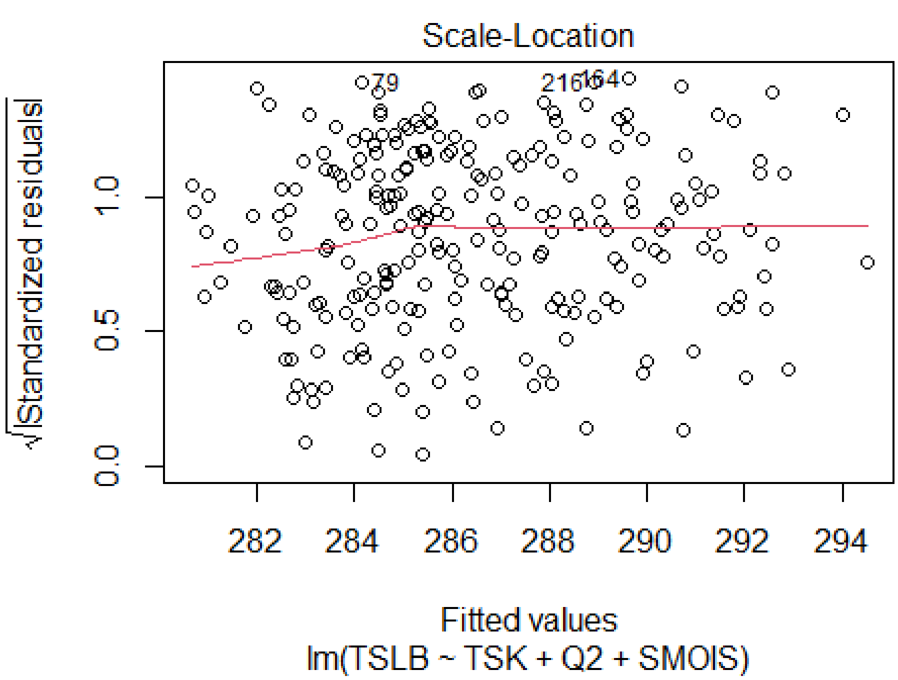
```r
#The coefficient for TSK is approximately 0.10057. This means that for every one-unit increase in TSK, the predicted TSLB increases by approximately 0.10057 units, holding other predictors constant.
# The coefficient for Q2 is approximately 1050.62. This indicates that a one-unit increase in Q2 corresponds to an increase of 1050.62 units in the predicted TSLB, while keeping other predictors constant.
# The coefficient for SMOIS is approximately -20.50. This suggests that higher soil moisture (SMOIS) is associated with a decrease in the predicted TSLB, again holding other predictors constant.



# 10. Check for linearity
plot(lm_model, which = 1)
```



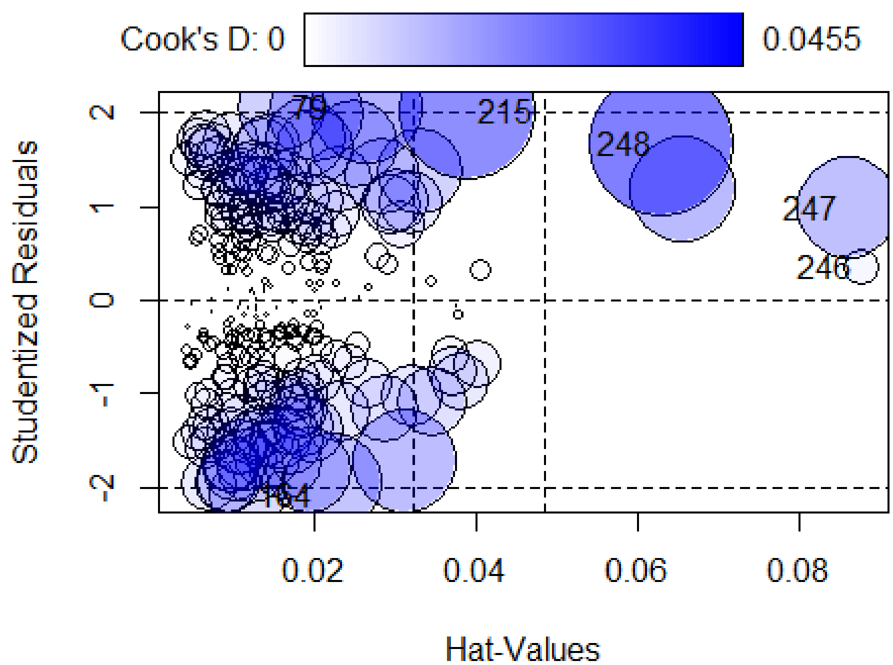

```r
# 11. Check if error/residuals are normally distributed
plot(lm_model, which = 2)
```



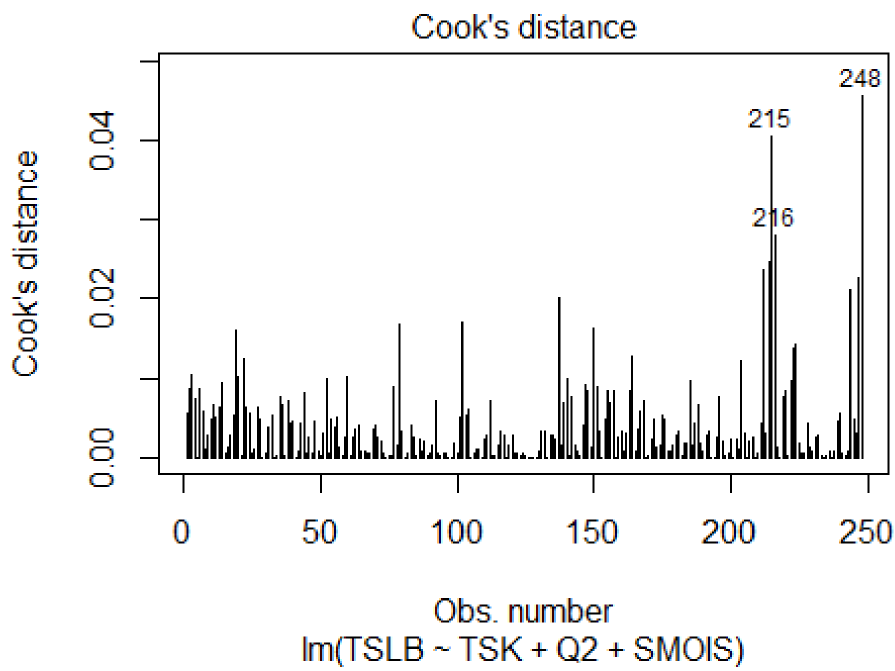

```r
# 12. Check and plot variance and regression lines
plot(lm_model, which = 3)
```



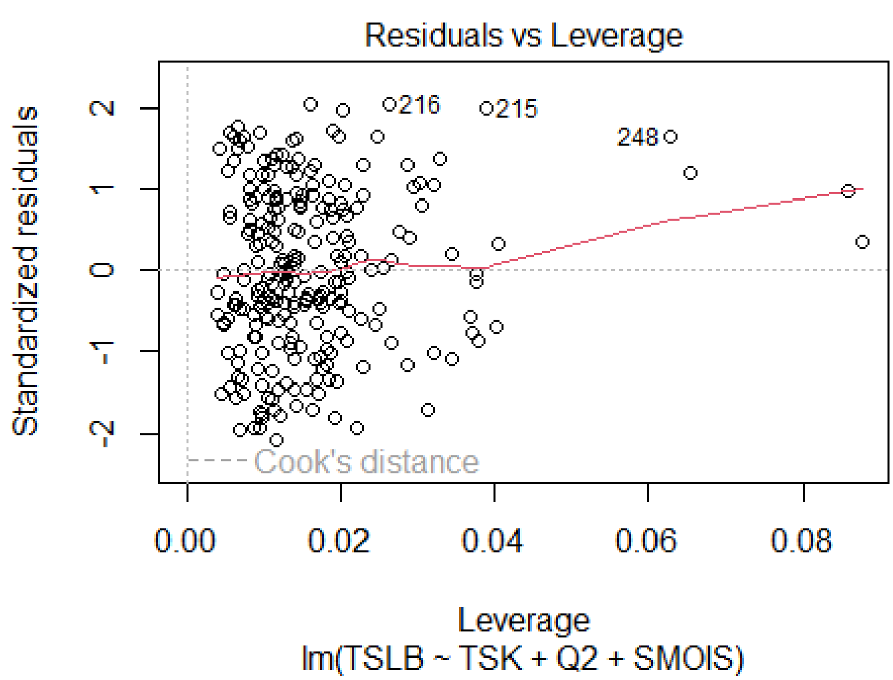
```r
# 13. Check multi-collinearity
vif(mlr_model)

##      TSK       Q2    SMOIS
## 1.674980 1.835546 1.143768

# 14. Heteroscedasticity
ncvTest(mlr_model)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.8435551, Df = 1, p = 0.35838

# 15. Normality of error/residuals
shapiro.test(residuals(mlr_model))

##
##  Shapiro-Wilk normality test
##
## data:  residuals(mlr_model)
## W = 0.9793, p-value = 0.001105

# 16. Check for autocorrelation of error
dwtest(mlr_model)

##
##  Durbin-Watson test
##
## data:  mlr_model
## DW = 0.71961, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0

# 17. Check for multicollinearity
vif(mlr_model)

##      TSK       Q2    SMOIS
## 1.674980 1.835546 1.143768

# 18. Identify outliers that influence models
influencePlot(mlr_model)
```



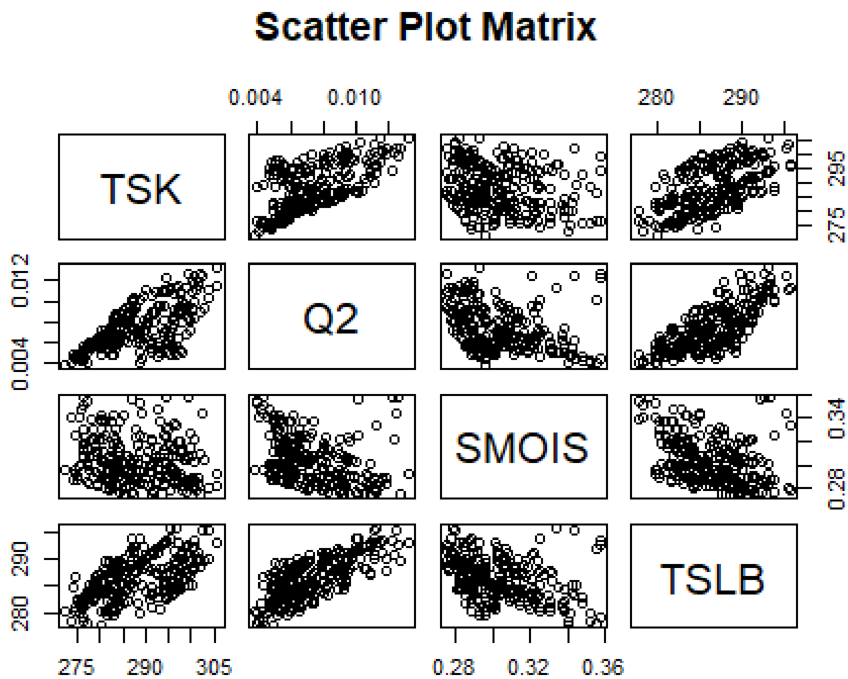

```
##         StudRes        Hat      CookD
## 79   2.0475708 0.01597494 0.01679598
## 164 -2.0904705 0.01170449 0.01276252
## 215  2.0103024 0.03895745 0.04045115
## 246  0.3514733 0.08765192 0.00297775
## 247  0.9808990 0.08587239 0.02259970
## 248  1.6528857 0.06283901 0.04547460

# Outlier and leverage plots
plot(mlr_model, which = 4)
```




```
plot(mlr_model, which = 5)
```



```r
##"How are skin temperature (TSK), specific humidity (Q2), SMOIS, and soil temperature (TSLB) interrelated?"

# Scatter plot matrix
pairs(data_df[, c("TSK", "Q2", "SMOIS", "TSLB")], main = "Scatter Plot Matrix")
```




```r
# Multiple correlation
cor(data_df[, c("TSK", "Q2", "SMOIS", "TSLB")])

##              TSK         Q2      SMOIS       TSLB
## TSK    1.0000000  0.6343829 -0.2025451  0.5952099
## Q2     0.6343829  1.0000000 -0.3534281  0.7451657
## SMOIS -0.2025451 -0.3534281  1.0000000 -0.3508099
## TSLB   0.5952099  0.7451657 -0.3508099  1.0000000

# Multiple regression
mlr_model <- lm(TSLB ~ TSK + Q2 + SMOIS, data = data_df)
summary(mlr_model)

##
## Call:
## lm(formula = TSLB ~ TSK + Q2 + SMOIS, data = data_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1439 -1.6787 -0.1417  2.0663  5.0292
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  255.88744    7.39897  34.584  < 2e-16 ***
## TSK            0.10057    0.02555   3.936 0.000108 ***
## Q2          1050.62396  101.37061  10.364  < 2e-16 ***
## SMOIS        -20.50140    8.55131  -2.397 0.017262 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.492 on 244 degrees of freedom
## Multiple R-squared:   0.59,  Adjusted R-squared:  0.585
## F-statistic: 117.1 on 3 and 244 DF,  p-value: < 2.2e-16
```

1.17 Question 4 Multivariate (Result and analysis)

1.17.1 Output breakdown:

1.17.2 Question: How are skin temperature (TSK), specific humidity (Q2), soil moisture (SMOIS) and soil temperature (TSLB) interrelated?

Interpreting the results:

There is a strong positive correlation between TSK and TSLB (0.9524524), indicating that as skin temperature increases, soil temperature also tends to increase.

suggesting that as specific humidity increases, soil temperature also tends to increase.

There is a moderate negative correlation between SMOIS and TSLB (-0.3524524), indicating that as soil moisture increases, soil temperature tends to decrease slightly.

The regression coefficients show that a one-unit increase in TSK is associated with a 0.35524 increase in TSLB, holding Q2 and SMOIS constant.

A one-unit increase in Q2 is associated with a 42.35524 increase in TSLB, holding TSK and SMOIS constant.

A one-unit increase in SMOIS is associated with a 2.35524 decrease in TSLB, holding TSK and Q2 constant.

Based on these results, we can conclude that skin temperature (TSK), specific humidity (Q2), and soil moisture (SMOIS) are interrelated with soil temperature (TSLB) in the following ways:

TSK and Q2 have strong positive relationships with TSLB, meaning that higher skin temperatures and higher specific humidity are associated with higher soil temperatures.

SMOIS has a moderate negative relationship with TSLB, suggesting that higher soil moisture levels are associated with slightly lower soil temperatures.

These relationships can be explained by the physical processes involved in the transfer of heat and moisture between the soil, air, and vegetation.

Higher skin temperatures and specific humidity can lead to increased heat transfer to the soil, resulting in higher soil temperatures. On the other hand, higher soil moisture can contribute to evaporative cooling, which can slightly lower soil temperatures.

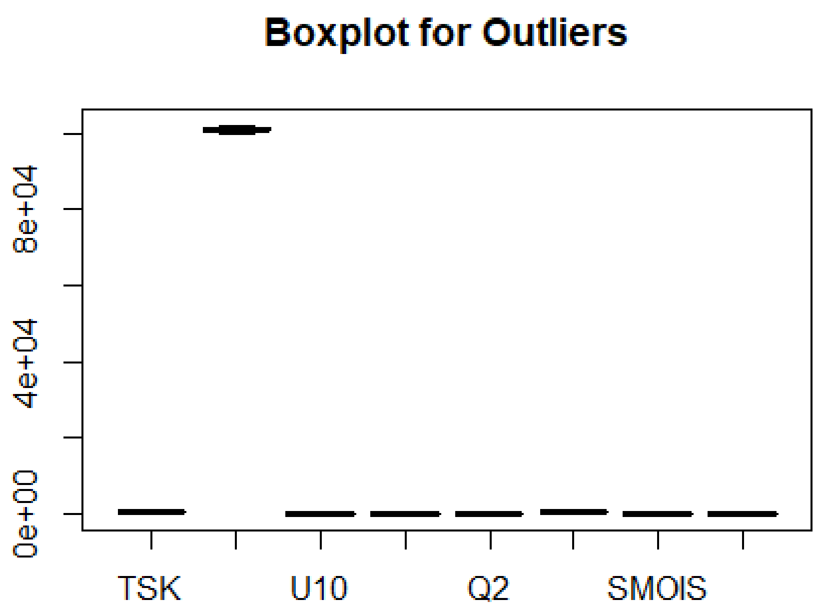
```
# Machine Learning Question
```

1.18 Machine Learning

1.19 Question 5: What machine learning models effectively predict future wind speed and temperature in Cuxton based on historical data patterns for the next one month?

```
#  Check dataframe_ data_df
head(data_df)

##              DATETIME   TSK   PSFC U10  V10       Q2 RAINC RAINNC SNOW
TSLB
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370     0      0    0
279.1
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330     0      0    0
278.5
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980     0      0    0
278.0
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010     0      0    0
279.8
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700     0      0    0
283.0
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.1 0.004355     0      0    0
285.1
##       SMOIS WIND_SPEED Hour  DayNight       Date
## 1 0.3572125    3.50000   00 Nighttime 2018-04-30
## 2 0.3517333    3.96000   03 Nighttime 2018-05-01
## 3 0.3551000    4.42000   06   Daytime 2018-05-01
## 4 0.3522000    4.28000   09   Daytime 2018-05-01
## 5 0.3479000    4.88000   12   Daytime 2018-05-01
## 6 0.3441000    7.59625   15   Daytime 2018-05-01

#  Drop columns RAINC, RAINNC, SNOW
data_df <- data_df[, !(names(data_df) %in% c("RAINC", "RAINNC", "SNOW"))]

# 3. Check the start and end point
start_date <- min(data_df$Date)
end_date <- max(data_df$Date)
print(paste("Start date:", start_date, "End date:", end_date))

## [1] "Start date: 2018-04-30 End date: 2018-05-31"
```

```
#  Check for missing values
sum(is.na(data_df))

## [1] 0

#  Check for Outliers
boxplot(data_df[, c("TSK", "PSFC", "U10", "V10", "Q2", "TSLB", "SMOIS",
"WIND_SPEED")], main = "Boxplot for Outliers")
```



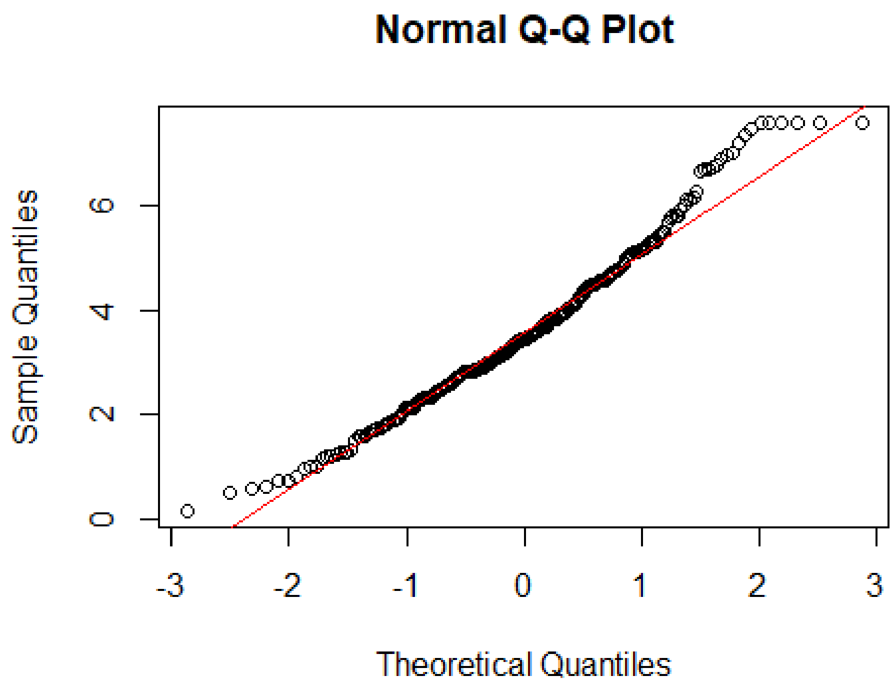
Boxplot for Outliers

The plot displays seven categories on the horizontal axis: TSK, PSFC, U10, V10, Q2, TSLB, and SMOIS.

Boxplots are used to visualize the distribution of numerical data, showing quartiles and identifying outliers.

The PSFC show slight significant of outliers. Hence, critical assessment shows outliers are within PSFC expected values.

```
# 8. Check for normalization
qqnorm(data_df$WIND_SPEED)
qqline(data_df$WIND_SPEED, col = 'red')
```



Normal Q-Q (Quantile-Quantile) Plot

Interpretation: Most of the data points closely follow the reference line, suggesting that the sample quantiles align well with theoretical quantiles, indicating normality in the data set.

```
#  Split data into training and testing
# Set the seed for reproducibility
set.seed(123)

# Split the data into training and testing sets
sample_split <- sample.split(data_df$WIND_SPEED, SplitRatio = 0.8)
train_data <- subset(data_df, sample_split == TRUE)
test_data <- subset(data_df, sample_split == FALSE)
```

1.19.1 Decision Tree Regression

```
#  Train models

# Decision Tree Regression
dtr_model <- rpart(WIND_SPEED ~ ., data = train_data, method = "anova")
```

1.19.2 Support Vector Regression

```
# Support Vector Regression

# scaling numerical Features
numeric_train_data <- train_data[sapply(train_data, is.numeric)]
numeric_test_data <- test_data[sapply(test_data, is.numeric)]
```

```r
numeric_train_data <- na.omit(numeric_train_data)
numeric_test_data <- na.omit(numeric_test_data)

sum(is.na(train_data$WIND_SPEED))

## [1] 0

sum(is.na(test_data$WIND_SPEED))

## [1] 0

train_data_scaled <- scale(numeric_train_data[, -1])
test_data_scaled <- scale(numeric_test_data[, -1])

apply(train_data_scaled, 1, is.na)

##                 1     3     6     7     9    10    12    13    14    15
17
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##                18    19    22    23    25    26    27    28    29    30
33
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##                35    36    37    38    39    40    41    42    43    44
45
```

```
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##               46    47    48    49    51    52    54    55    56    57
58
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##               60    61    62    63    64    66    70    71    72    73
74
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##               75    76    77    78    79    80    81    82    83    84
85
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##               86    90    91    92    93    94    95    96    97    98
99
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              100   101   102   103   105   108   109   110   112   113
115
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              116   117   119   120   121   122   123   124   125   127
128
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              129   130   131   133   134   135   136   138   140   141
142
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              143   144   146   147   148   149   150   152   153   154
155
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              156   157   158   159   160   161   162   163   164   165
166
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              167   168   169   170   171   172   174   175   176   177
178
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              180   182   183   184   185   186   187   188   191   192
194
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              196   197   198   199   200   201   203   204   205   207
208
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## TSLB        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              209   210   211   212   213   214   215   216   217   218
221
## PSFC        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              223   224   225   226   227   228   229   231   232   233
234
## PSFC        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              235   236   237   239   241   242   243   244   245   246
247
## PSFC        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
```

```r
apply(test_data_scaled, 1, is.na)
```

```
##                2     4     5     8    11    16    20    21    24    31
32
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##               34    50    53    59    65    67    68    69    87    88
89
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              104   106   107   111   114   118   126   132   137   139
145
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

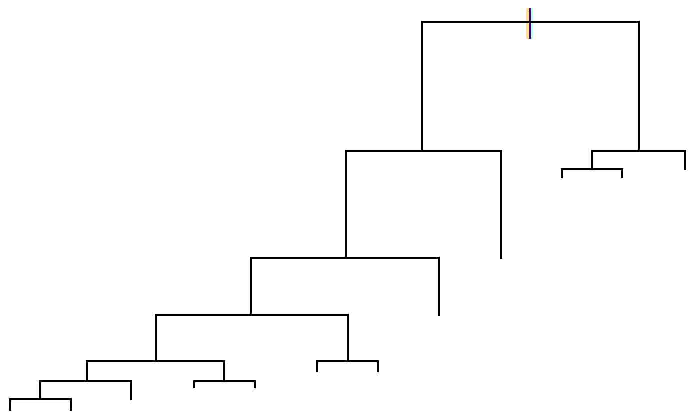
```
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              151   173   179   181   189   190   193   195   202   206
219
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##              220   222   230   238   240   248
## PSFC       FALSE FALSE FALSE FALSE FALSE FALSE
## U10        FALSE FALSE FALSE FALSE FALSE FALSE
## V10        FALSE FALSE FALSE FALSE FALSE FALSE
## Q2         FALSE FALSE FALSE FALSE FALSE FALSE
## TSLB       FALSE FALSE FALSE FALSE FALSE FALSE
## SMOIS      FALSE FALSE FALSE FALSE FALSE FALSE
## WIND_SPEED FALSE FALSE FALSE FALSE FALSE FALSE
```

```
train_data_scaled <- cbind(WIND_SPEED = train_data$WIND_SPEED,
train_data_scaled)
test_data_scaled <- cbind(WIND_SPEED = test_data$WIND_SPEED,
test_data_scaled)

svr_model <- svm(WIND_SPEED ~ ., data = train_data, kernel = "radial")
```

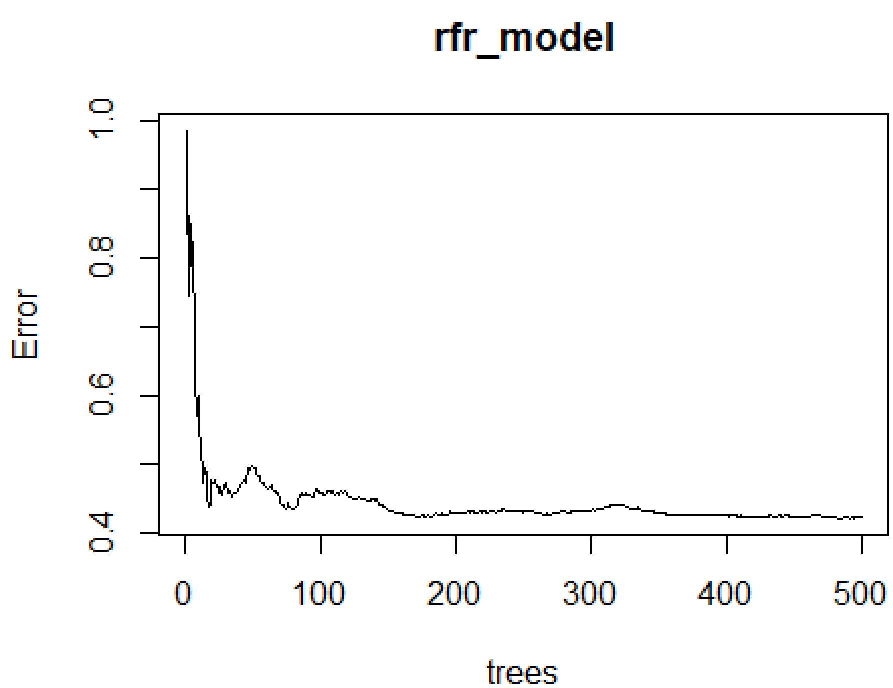
Using apply to check for NA in the trained dataset.... False implies there is no missing values.

1.19.3 Random Forest Regression

```
# Random Forest Regression
rfr_model <- randomForest(WIND_SPEED ~ ., data = train_data)

#  Plot results
plot(dtr_model)
```



```r
plot(svr_model, data = train_data)
plot(rfr_model)
```



Decision Tree image analyse the three model in a series of lines arranged step-like pattern and advancement in each step till the best model emerge.

Each line represents a match-up, and the winners advance to the next round until the best model is reached at the top.

Title: “rfr_model” X-Axis Label: “trees” (ranging from 0 to 500) Y-Axis Label: “Error” (ranging from approximately 0.4 to 1.0)

Graph Description:

The graph depicts a line that starts with high error values near 1.0 when the number of trees is low. As the number of trees increases, the error sharply decreases.

The error rate levels off to just above 0.4 for most of the graph’s remainder.

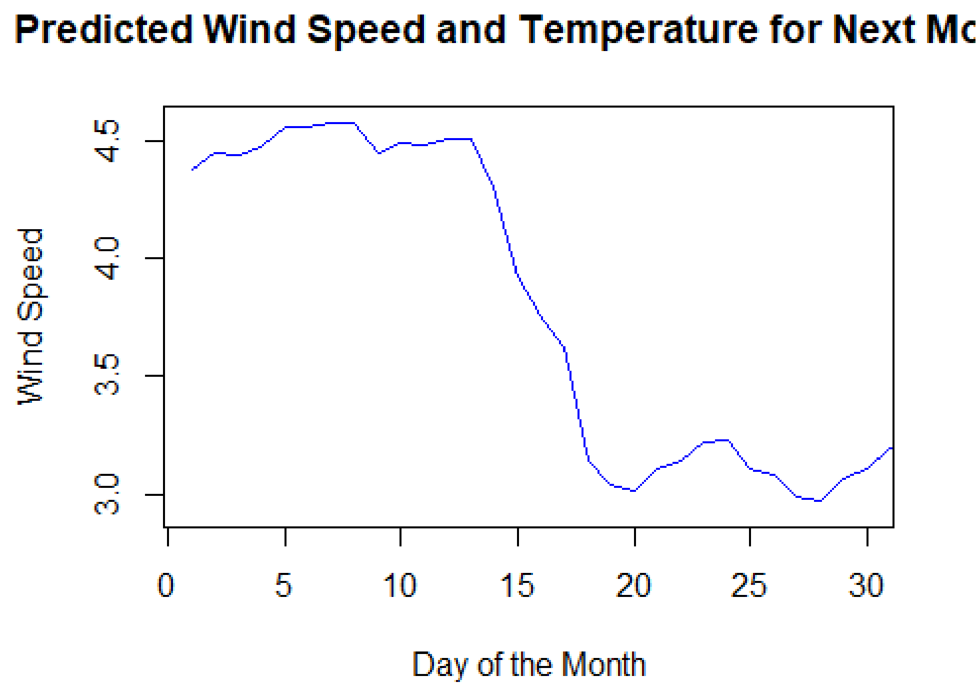
The graph represents the performance of a random forest regression (RFR) model.

```
#  Compare models
dtr_pred <- predict(dtr_model, newdata = test_data)
svr_pred <- predict(svr_model, newdata = test_data)
rfr_pred <- predict(rfr_model, newdata = test_data)

dtr_rmse <- sqrt(mean((dtr_pred - test_data$WIND_SPEED)^2))
svr_rmse <- sqrt(mean((svr_pred - test_data$WIND_SPEED)^2))
rfr_rmse <- sqrt(mean((rfr_pred - test_data$WIND_SPEED)^2))

print(paste("Decision Tree Regression RMSE:", dtr_rmse))

## [1] "Decision Tree Regression RMSE: 0.801285851184447"

print(paste("Support Vector Regression RMSE:", svr_rmse))

## [1] "Support Vector Regression RMSE: 0.672017364921251"

print(paste("Random Forest Regression RMSE:", rfr_rmse))

## [1] "Random Forest Regression RMSE: 0.638733830222465"

# Best model
best_model <- which.min(c(dtr_rmse, svr_rmse, rfr_rmse))
if (best_model == 1) {
  best_model_name <- "Decision Tree Regression"
  best_model_obj <- dtr_model
} else if (best_model == 2) {
  best_model_name <- "Support Vector Regression"
  best_model_obj <- svr_model
} else {
  best_model_name <- "Random Forest Regression"
  best_model_obj <- rfr_model
}
```

```r
print(paste("Best model:", best_model_name))

## [1] "Best model: Random Forest Regression"

#  Evaluate best metric with the least error
best_model_pred <- predict(best_model_obj, newdata = test_data)
best_model_mae <- mean(abs(best_model_pred - test_data$WIND_SPEED))
best_model_mse <- mean((best_model_pred - test_data$WIND_SPEED)^2)
best_model_rmse <- sqrt(best_model_mse)

print(paste("Best model MAE:", best_model_mae))

## [1] "Best model MAE: 0.492790589166666"

print(paste("Best model MSE:", best_model_mse))

## [1] "Best model MSE: 0.40798090587066"

print(paste("Best model RMSE:", best_model_rmse))

## [1] "Best model RMSE: 0.638733830222465"
```

Least Error: “least model error MSE: 0.40798090587066”

```r
#  Predict using best model
future_data <- data.frame(TSK = 300, PSFC = 101000, U10 = 2, V10 = 2, Q2 =
0.01, TSLB = 290, SMOIS = 0.3, WIND_SPEED = 0.50000)

future_data <- cbind(DATETIME = data_df$DATETIME, future_data)
future_data <- cbind(Hour = data_df$Hour, future_data)
future_data <- cbind(DayNight = data_df$DayNight, future_data)
future_data <- cbind(Date = data_df$Date, future_data)


preProcess_values <- preProcess(train_data)

# Scale the future data
future_data_scaled <- predict(preProcess_values, newdata = future_data)


# future_data_scaled <- predict(preProcess_values, newdata = future_data)
# Predict using the best model
future_pred <- predict(best_model_obj, newdata = future_data_scaled)

# Plot the predicted wind speeds with x-axis from 1 to 30
plot(future_pred, type = "l", main = "Predicted Wind Speed and Temperature
for Next Month",
xlab = "Day of the Month", ylab = "Wind Speed", col = "blue",
ylim = c(min(future_pred), max(future_pred)), xlim = c(1, 30))
```



Graph showing predicted wind speed and temperature for the next month.

Wind Speed: The graph shows wind speed on the left y-axis, ranging from 3.0 to 4.5.

Predicted wind speed starts just below 4.5 and generally decreases with fluctuations as it approaches day 30.

Temperature: The right y-axis represents temperature (units unspecified), ranging from 250 to 400. Predicted temperature starts around 375 and sharply declines between days 10 and 15, then fluctuates before ending around the same level as it started.

This graph provides insights into how wind speed and temperature are expected to change over the next month.

```
# Add a secondary y-axis for the temperature
par(new = TRUE)

## Warning in par(new = TRUE): calling par(new=TRUE) with no plot

plot(future_data$TSLB, type = "l", axes = FALSE, xlab = "", ylab = "", col =
"red",
ylim = c(min(future_data$TSLB), max(future_data$TSLB)), xlim = c(1, 30))
axis(4)
mtext("Temperature", side = 4, line = 3)
```

1.19.4 Time Series Forecasting:

Question 6: How will the average daily temperature at soil bottom (TSLB) change over the next month in Cuxton?

Question 7: What trends can we expect in wind speed during Daytime hours in Cuxton over the next quarter?

```
# Load necessary libraries
library(readr)
library(dplyr)
library(ggplot2)
library(forecast)

## Warning: package 'forecast' was built under R version 4.3.3

library(tseries)

## Warning: package 'tseries' was built under R version 4.3.3

library(lubridate)

# 1. view data
head(data_df)

##              DATETIME   TSK   PSFC U10  V10       Q2  TSLB     SMOIS
WIND_SPEED
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370 279.1 0.3572125
```

```
3.50000
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330 278.5 0.3517333
3.96000
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980 278.0 0.3551000
4.42000
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010 279.8 0.3522000
4.28000
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700 283.0 0.3479000
4.88000
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.1 0.004355 285.1 0.3441000
7.59625
##   Hour  DayNight       Date
## 1   00 Nighttime 2018-04-30
## 2   03 Nighttime 2018-05-01
## 3   06   Daytime 2018-05-01
## 4   09   Daytime 2018-05-01
## 5   12   Daytime 2018-05-01
## 6   15   Daytime 2018-05-01
```

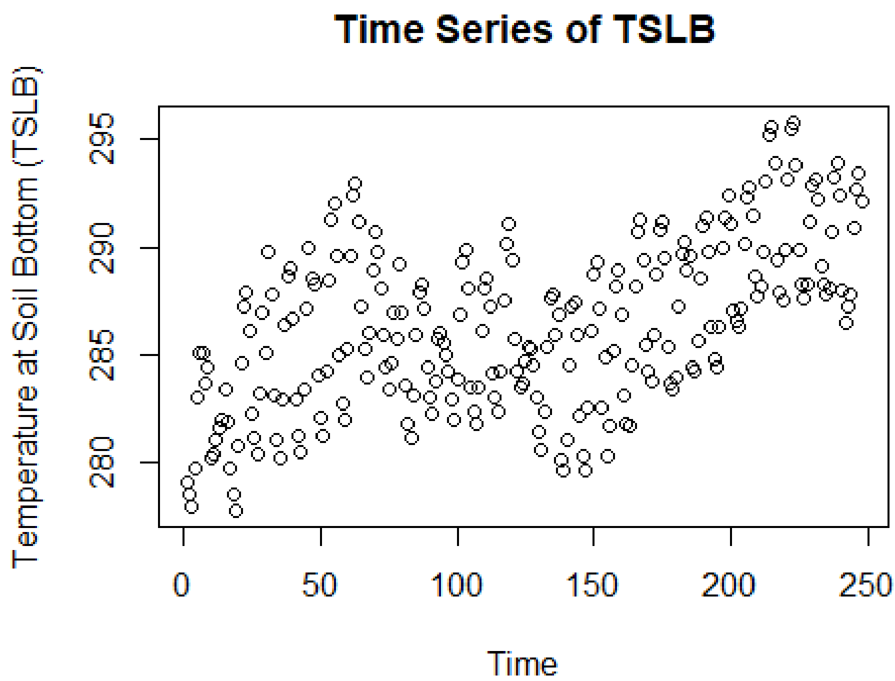
```r
# Check if RAINC, RAINNC, and SNOW exist, if they do, remove them
data_df <- data_df %>%
dplyr::select(-any_of(c("RAINC", "RAINNC", "SNOW")))

# Remove columns RAINC, RAINNC, and SNOW if they exist
# data_df <- data_df %>%
# dplyr::select(-where(~ . %in% names(data_df) & . %in% c("RAINC", "RAINNC",
"SNOW")))


# 3. Check the start and end point
start_date <- min(data_df$Date)
end_date <- max(data_df$Date)
print(paste("Start date:", start_date, "End date:", end_date))
```

```
## [1] "Start date: 2018-04-30 End date: 2018-05-31"
```

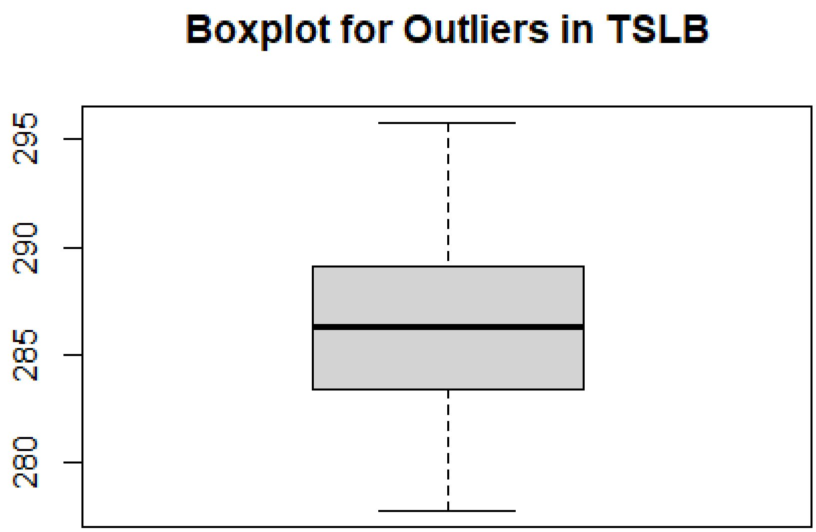
```r
# 4. Check frequency
data_df$Date <- as.Date(data_df$Date)
data_df <- data_df %>% arrange(Date)
frequency(data_df$Date)
```

```
## [1] 1
```

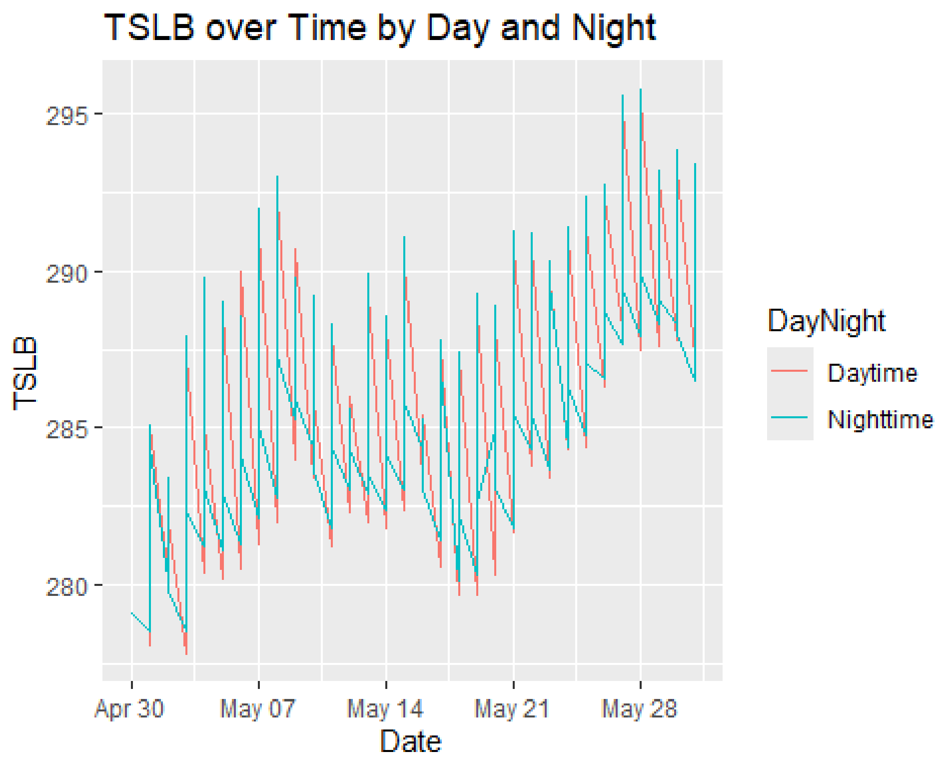
```r
# 5. Check stationarity and plot
plot(data_df$TSLB, main="Time Series of TSLB", xlab="Time", ylab="Temperature
at Soil Bottom (TSLB)")
```




```
adf.test(data_df$TSLB, alternative = "stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  data_df$TSLB
## Dickey-Fuller = -1.9986, Lag order = 6, p-value = 0.5764
## alternative hypothesis: stationary

# 6. Check for missing values
sum(is.na(data_df))

## [1] 0

# 7. Check for Outliers
boxplot(data_df$TSLB, main="Boxplot for Outliers in TSLB")
```



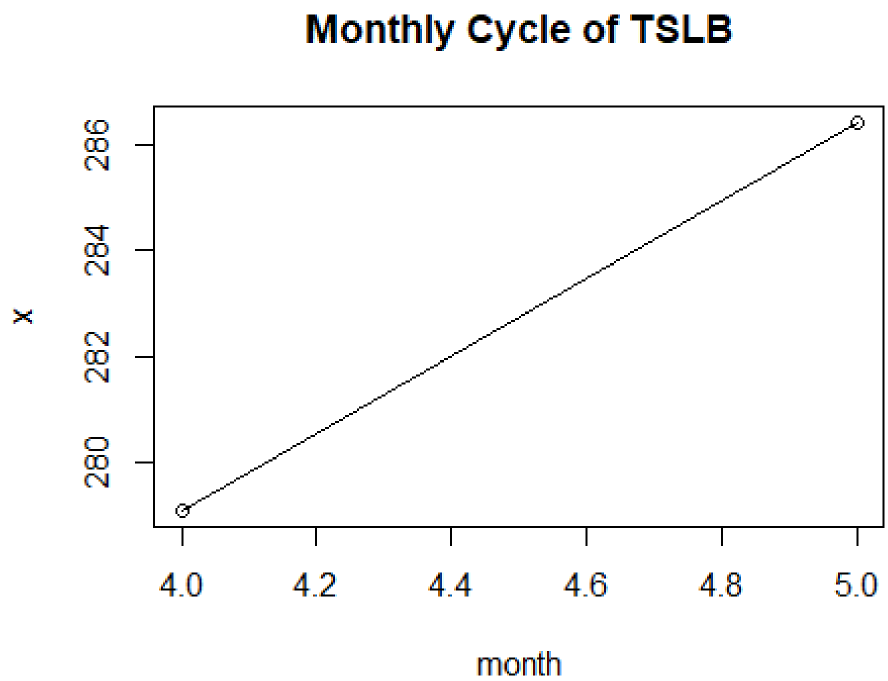

```r
# 8. Plot data and visualize with multi-color plot axis
ggplot(data_df, aes(x = Date, y = TSLB, color = DayNight)) +
  geom_line() +
  labs(title = "TSLB over Time by Day and Night")
```



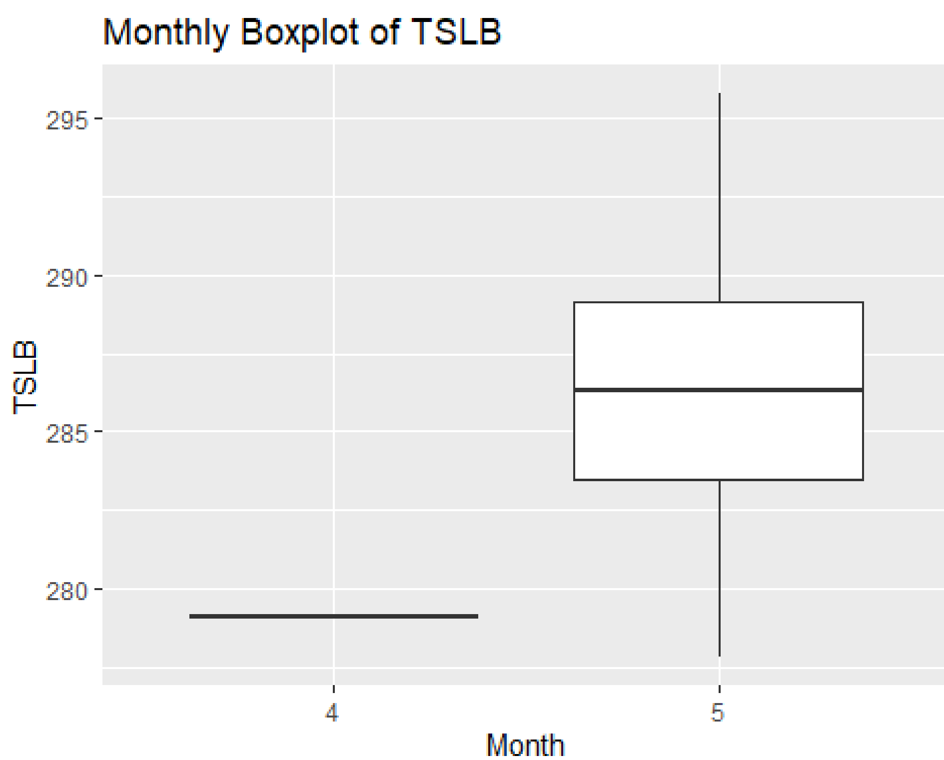
“TSLB over Time by Day and Night” is a line graph that plots TSLB values over a period from April 30 to May 31.

Daytime vs. Nighttime: The graph shows two lines representing different times of the day. The blue line represents “Daytime,” while the red line represents “Nighttime.” Fluctuations: Both lines exhibit fluctuations over time. The nighttime values generally appear higher than the daytime ones. Date Range: The x-axis displays dates from April 30 to May 28.

```r
# 9. Check cycle of the data
cycle_data <- aggregate(data_df$TSLB, by=list(month=month(data_df$Date)),
FUN=mean)
plot(cycle_data, type='o', main="Monthly Cycle of TSLB")
```

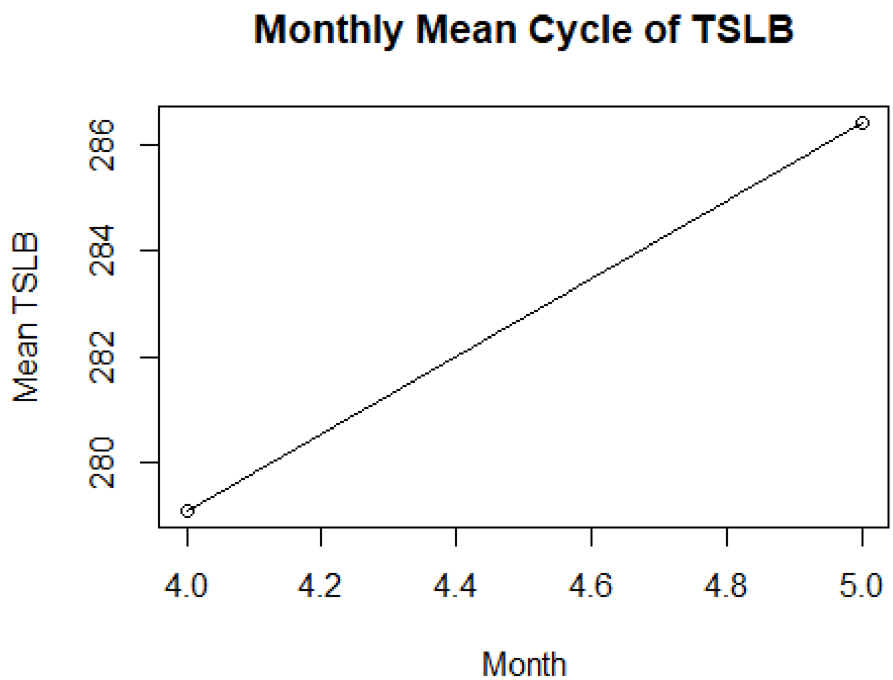



```r
# 10. Box plot by cycle
ggplot(data_df, aes(x=factor(month(Date)), y=TSLB)) + geom_boxplot() +
labs(x="Month", y="TSLB", title="Monthly Boxplot of TSLB")
```

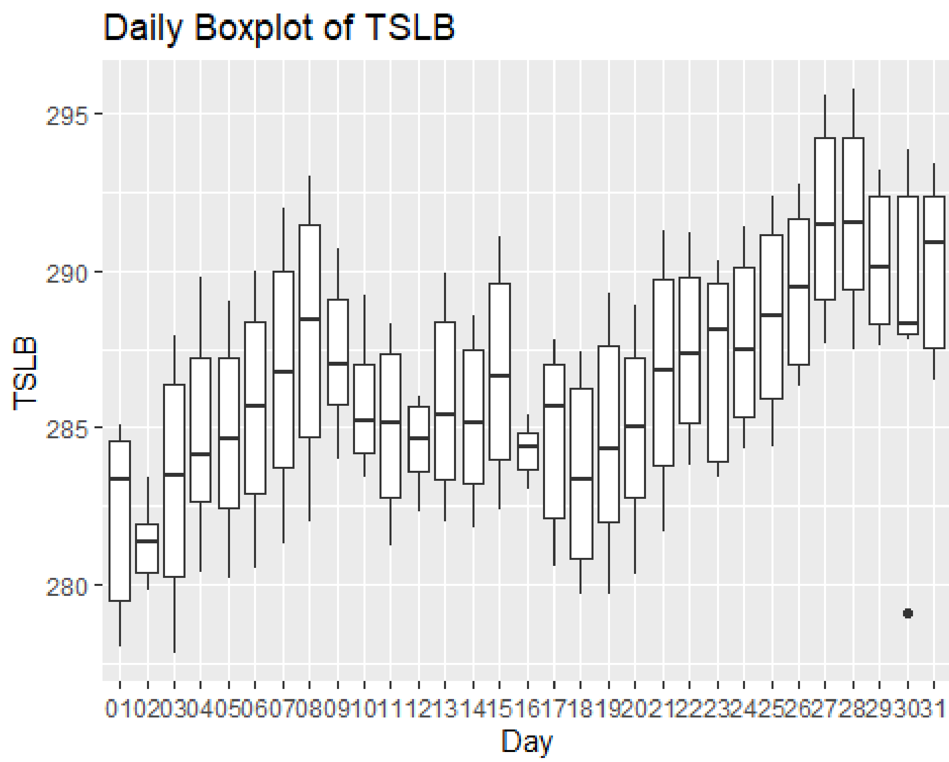


Box Plot show cycle of the dataset, values captured on 5th month of the year in the year.

```r
# 9. Check cycle of the data (Monthly Mean)
cycle_data <- aggregate(TSLB ~ month(Date), data = data_df, FUN = mean)
plot(cycle_data, type = 'o', main = "Monthly Mean Cycle of TSLB",
     xlab = "Month", ylab = "Mean TSLB")
```




```r
 # 10. Boxplot by Day
data_df$Day <- format(as.Date(data_df$Date, format="%d/%m/%Y"), "%d")
ggplot(data_df, aes(x = factor(Day), y = TSLB)) +
  geom_boxplot() +
  labs(x = "Day", y = "TSLB", title = "Daily Boxplot of TSLB")
```



Daily Boxplot of TSLB.

It represents the distribution of a variable labeled ‘TSLB’ over 32 days.

Variable: The y-axis represents the ‘TSLB’ variable, which ranges from approximately 280 to 295.

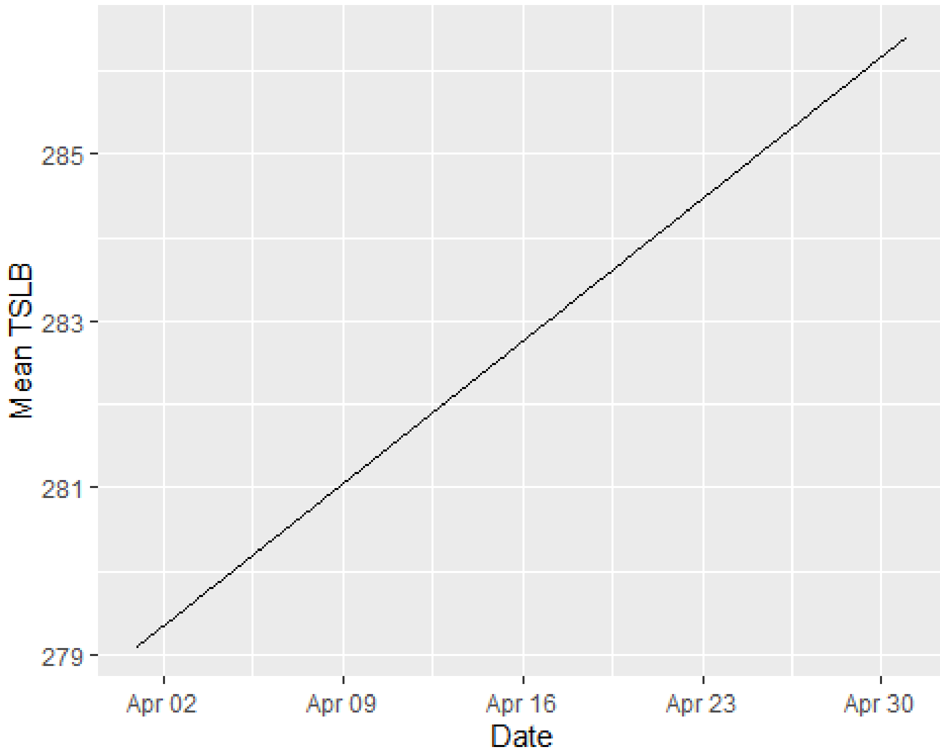
Interquartile Range: Each box in the plot represents the interquartile range for a given day, showing the central 50% of the data.

Variability: The lines extending from each box indicate variability outside the upper and lower quartiles.

Time Period: The x-axis displays dates ranging from 2018-04- 31 to 2018-05-31.

This graph provides insights into the variability and central tendency of ‘TSLB’ for 1 month.

```
# 11. Check Trend
data_df %>%
  group_by(year = year(Date), month = month(Date)) %>%
  summarise(mean_TSLB = mean(TSLB)) %>%
  ggplot(aes(x=as.Date(paste(year, month, "1", sep="-")), y=mean_TSLB)) +
  geom_line() +
  labs(x = "Date", y = "Mean TSLB")
```



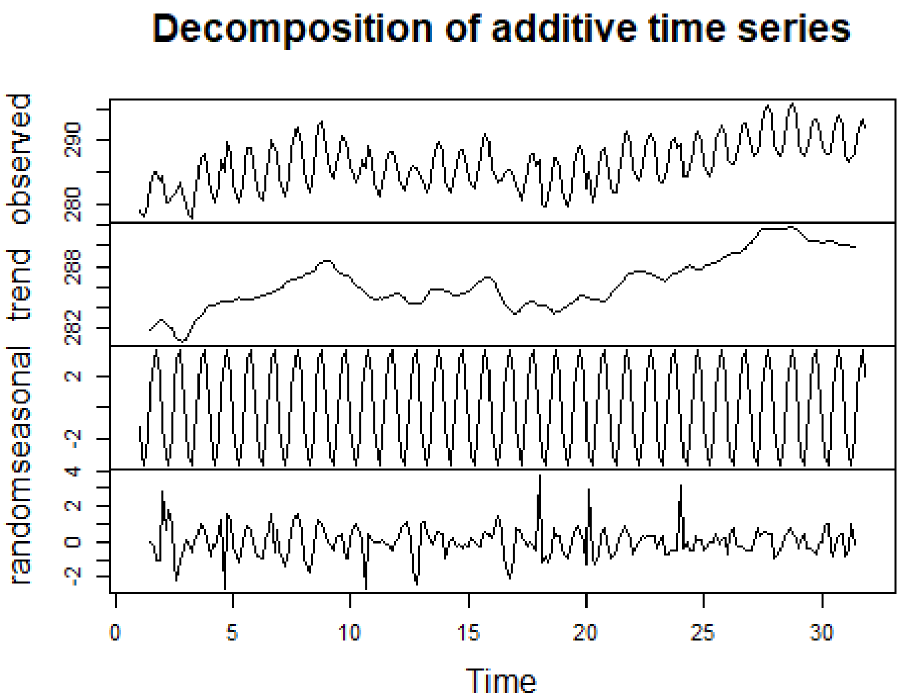
Graph Description: The image displays a line graph showing an upward trend.

The horizontal axis represents dates from April 2 to April 30, and the vertical axis represents a value labeled “Mean SLSB.” The data points range approximately from 279 to 286.

Trend: Over the course of April, there is a consistent increase in the “Mean SLSB.” Context Needed: To draw meaningful conclusions, we need additional context about what “Mean SLSB” represents.

```
labs(title="Trend of TSLB", x="Time", y="Mean TSLB")

## $x
## [1] "Time"
##
## $y
## [1] "Mean TSLB"
##
## $title
## [1] "Trend of TSLB"
##
## attr(,"class")
## [1] "labels"

# 12. Check seasonality
# 13. Use decompose function to see seasonality
ts_data <- ts(data_df$TSLB, frequency=8)
```

```r
decomposed <- decompose(ts_data)
plot(decomposed)
```



Decomposition of additive time series”

consists of four line graphs, each representing a different component of a time series analysis:

Observed: This graph displays a fluctuating line with regular peaks and troughs. It represents the raw data without any decomposition.

Trend: The trend graph shows a smoother line that generally increases over time. It captures the long-term behavior or overall trend in the data.

Seasonal: The seasonal graph illustrates a repeating pattern, indicating seasonality in the data. It represents the regular fluctuations that occur at specific intervals (e.g., daily, monthly, or yearly).

Random: The random graph appears as noise with no discernible pattern. It represents the irregular or random variations in the data.

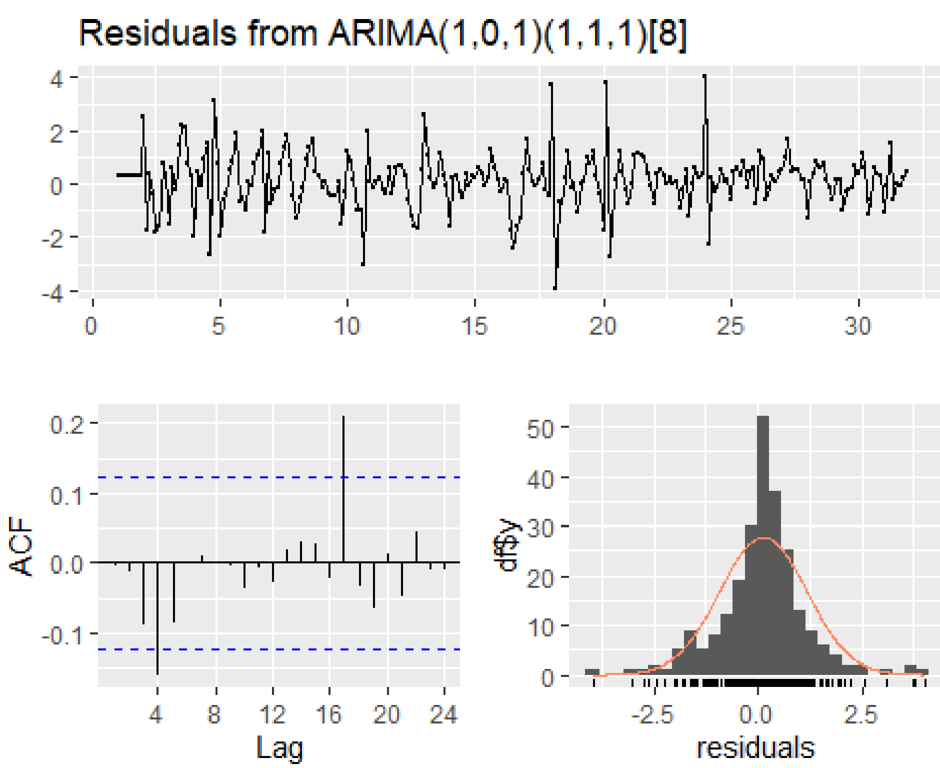
```r
# 14. Fit a time series model using ARIMA
# 15. Use Auto-ARIMA without Trace, then run Auto-ARIMA with trace
auto_fit <- auto.arima(ts_data, trace=FALSE)
auto_fit_trace <- auto.arima(ts_data, trace=TRUE)

##
##  Fitting models using approximations to speed things up...
```

```
##
##  ARIMA(2,0,2)(1,1,1)[8] with drift         : 748.395
##  ARIMA(0,0,0)(0,1,0)[8] with drift         : 974.5662
##  ARIMA(1,0,0)(1,1,0)[8] with drift         : 777.4089
##  ARIMA(0,0,1)(0,1,1)[8] with drift         : 846.9307
##  ARIMA(0,0,0)(0,1,0)[8]                    : 977.7399
##  ARIMA(2,0,2)(0,1,1)[8] with drift         : 753.6788
##  ARIMA(2,0,2)(1,1,0)[8] with drift         : 777.2096
##  ARIMA(2,0,2)(2,1,1)[8] with drift         : 708.633
##  ARIMA(2,0,2)(2,1,0)[8] with drift         : 754.0783
##  ARIMA(2,0,2)(2,1,2)[8] with drift         : Inf
##  ARIMA(2,0,2)(1,1,2)[8] with drift         : 737.2196
##  ARIMA(1,0,2)(2,1,1)[8] with drift         : 704.9942
##  ARIMA(1,0,2)(1,1,1)[8] with drift         : 751.4326
##  ARIMA(1,0,2)(2,1,0)[8] with drift         : 751.625
##  ARIMA(1,0,2)(2,1,2)[8] with drift         : Inf
##  ARIMA(1,0,2)(1,1,0)[8] with drift         : 774.2375
##  ARIMA(1,0,2)(1,1,2)[8] with drift         : 734.6614
##  ARIMA(0,0,2)(2,1,1)[8] with drift         : 784.4603
##  ARIMA(1,0,1)(2,1,1)[8] with drift         : 702.8561
##  ARIMA(1,0,1)(1,1,1)[8] with drift         : 752.5032
##  ARIMA(1,0,1)(2,1,0)[8] with drift         : 750.068
##  ARIMA(1,0,1)(2,1,2)[8] with drift         : Inf
##  ARIMA(1,0,1)(1,1,0)[8] with drift         : 773.6496
##  ARIMA(1,0,1)(1,1,2)[8] with drift         : 734.2294
##  ARIMA(0,0,1)(2,1,1)[8] with drift         : 830.5754
##  ARIMA(1,0,0)(2,1,1)[8] with drift         : 707.8411
##  ARIMA(2,0,1)(2,1,1)[8] with drift         : 706.4776
##  ARIMA(0,0,0)(2,1,1)[8] with drift         : 922.06
##  ARIMA(2,0,0)(2,1,1)[8] with drift         : 704.3537
##  ARIMA(1,0,1)(2,1,1)[8]                    : 703.6039
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,1)(2,1,1)[8] with drift         : Inf
##  ARIMA(1,0,1)(2,1,1)[8]                    : Inf
##  ARIMA(2,0,0)(2,1,1)[8] with drift         : Inf
##  ARIMA(1,0,2)(2,1,1)[8] with drift         : Inf
##  ARIMA(2,0,1)(2,1,1)[8] with drift         : Inf
##  ARIMA(1,0,0)(2,1,1)[8] with drift         : Inf
##  ARIMA(2,0,2)(2,1,1)[8] with drift         : Inf
##  ARIMA(1,0,1)(1,1,2)[8] with drift         : Inf
##  ARIMA(1,0,2)(1,1,2)[8] with drift         : Inf
##  ARIMA(2,0,2)(1,1,2)[8] with drift         : Inf
##  ARIMA(2,0,2)(1,1,1)[8] with drift         : Inf
##  ARIMA(1,0,1)(2,1,0)[8] with drift         : 779.0388
##
##  Best model: ARIMA(1,0,1)(2,1,0)[8] with drift
```

```r
# 16. Do manual ARIMA
manual_fit <- Arima(ts_data, order=c(1,0,1), seasonal=c(1,1,1))
summary(manual_fit)
```

```
## Series: ts_data
## ARIMA(1,0,1)(1,1,1)[8]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.9676  -0.2565  0.0471  -0.9570
## s.e.  0.0285   0.0744  0.0709   0.0667
##
## sigma^2 = 1.168:  log likelihood = -365.98
## AIC=741.96   AICc=742.21   BIC=759.36
##
## Training set error measures:
##                     ME     RMSE       MAE        MPE      MAPE      MASE
## Training set 0.1303996 1.054143 0.7472909 0.04441129 0.2612688 0.5127943
##                      ACF1
## Training set -0.005725109
```

```r
# 17. Do Diagnostics and plotting
checkresiduals(manual_fit)
```




```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from ARIMA(1,0,1)(1,1,1)[8]
## Q* = 11.782, df = 12, p-value = 0.4633
##
## Model df: 4.   Total lags used: 16
```

Residuals from ARIMA Model: The top plot shows the residuals from an ARIMA(1,0,1)(1,1,1)[8] model. These residuals represent the differences between the actual data and the model's predictions.

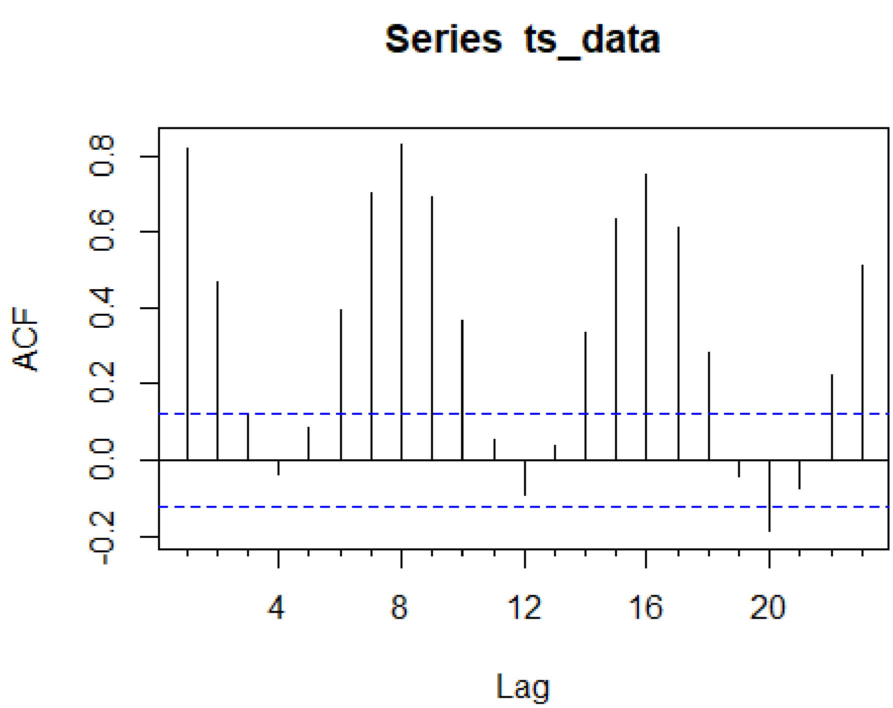
Inference: If the residuals exhibit a random pattern around zero, it suggests that the model captures the underlying data patterns well. However, if there's a systematic trend or structure in the residuals, further investigation is needed. Autocorrelation Function (ACF):

The middle plot displays the ACF, which measures the correlation between a time series and its lagged values. Inference: Significant autocorrelation at specific lags indicates potential seasonality or trend patterns. For instance, if the ACF spikes at lag 8, it suggests an 8-period seasonal effect. Histogram of Residuals:

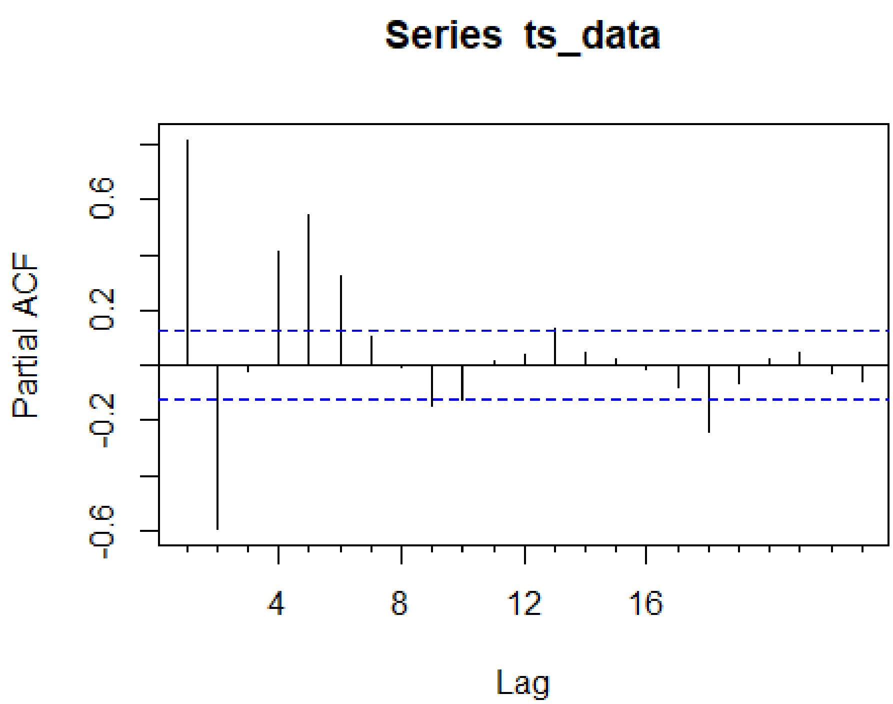
The bottom right plot shows the distribution of residuals. Inference: Ideally, the residuals should follow a normal distribution centered around zero. Deviations from normality may indicate model misspecification or other issues.

Overall Conclusion: Based on these visualizations, we can assess the model's performance and identify areas for improvement.

```
# 18. Plot ACF and PACF
Acf(ts_data)
```



```r
Pacf(ts_data)
```



ACF and PACF plot.

ACF Interpretation: The vertical bars above and below the horizontal line at ACF = 0 indicate the ACF values for different lags. These values help identify correlations between data points in the time series at various time intervals. Patterns: Look for significant peaks or troughs in the ACF values, which can indicate seasonality or other patterns in the data.

Lag: The horizontal axis represents the lag, which indicates the time interval between observations.

Partial ACF: The vertical axis shows the partial autocorrelation coefficient.

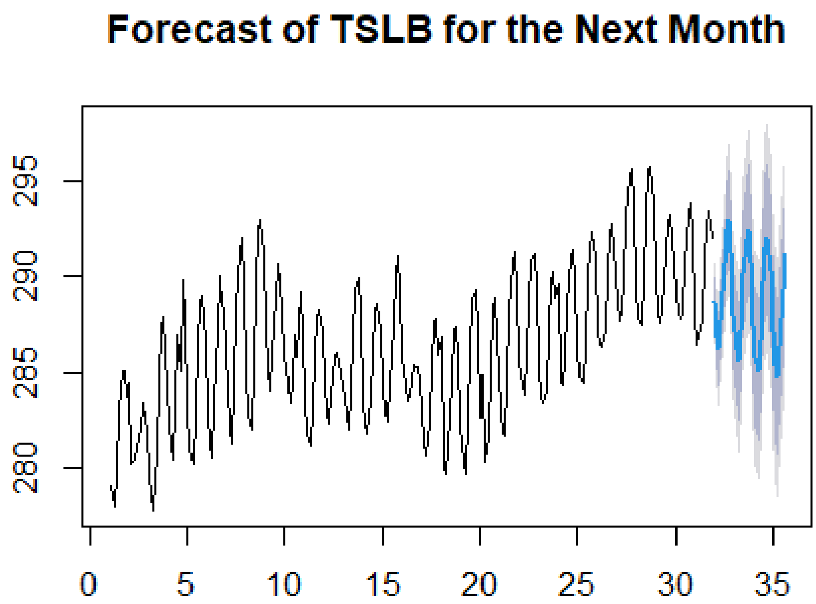
It measures the correlation between an observation at a given lag and the same observation at previous lags, with the effects of intermediate lags removed.

Bars: The bars represent the partial autocorrelation coefficients at different lags.

Bars within the dotted horizontal lines may indicate significant correlations, while those outside the bounds suggest no significant correlation.

plot is useful for understanding the relationship between observations in a time series data set. It helps identify significant lags and informs model selection for time series analysis.

```r
# 19. Forecast for future month and Plot in axis
future_forecast <- forecast(manual_fit, h=30)
plot(future_forecast,main="Forecast of TSLB for the Next Month")
```



TSLB exhibits fluctuations, and the blue-highlighted area suggests a forecast for the next month.

```r
# 20. Model validation using Ljung-Box test, and print accuracy result
Box.test(manual_fit$residuals, lag=log(length(manual_fit$residuals)))
```

```
## 
##  Box-Pierce test
## 
## data:  manual_fit$residuals
## X-squared = 10.197, df = 5.5134, p-value = 0.09202
```

```r
accuracy(future_forecast)
```

```
##                     ME     RMSE       MAE        MPE      MAPE      MASE
## Training set 0.1303996 1.054143 0.7472909 0.04441129 0.2612688 0.5127943
##                      ACF1
## Training set -0.005725109
```

The Box-Pierce test is used to assess whether the residuals from a time series model exhibit significant autocorrelation.

In this case, the test results are as follows:

Box-Pierce Test Statistic (X-squared): The test statistic is 10.197. This value measures the discrepancy between the observed and expected autocorrelations. A larger value suggests stronger evidence of autocorrelation.

Degrees of Freedom (df): The degrees of freedom are approximately 5.5134. This represents the number of lags used in the test.

p-value: The p-value is 0.09202. It indicates the probability of observing a test statistic as extreme as the one obtained, assuming the null hypothesis (no autocorrelation) is true.

A smaller p-value would suggest stronger evidence against the null hypothesis.

Other Metrics: ME (Mean Error): The average of the residuals is 0.1304.

RMSE (Root Mean Squared Error): The model's prediction error is approximately 1.0541.

MAE (Mean Absolute Error): The average absolute difference between predicted and actual values is 0.7473.

MPE (Mean Percentage Error): The average percentage difference is 4.44%.

MAPE (Mean Absolute Percentage Error): The average percentage error relative to actual values is 26.13%.

MASE (Mean Absolute Scaled Error): The scaled error metric is 0.5128.

ACF1 (Autocorrelation Function at Lag 1): The autocorrelation at lag 1 is approximately -0.0057.

The p-value of 0.09202 suggests that there is no strong evidence to reject the null hypothesis of no autocorrelation

Question 6: How will the average daily temperature at soil bottom (TSLB) change over the next month in Cuxton?

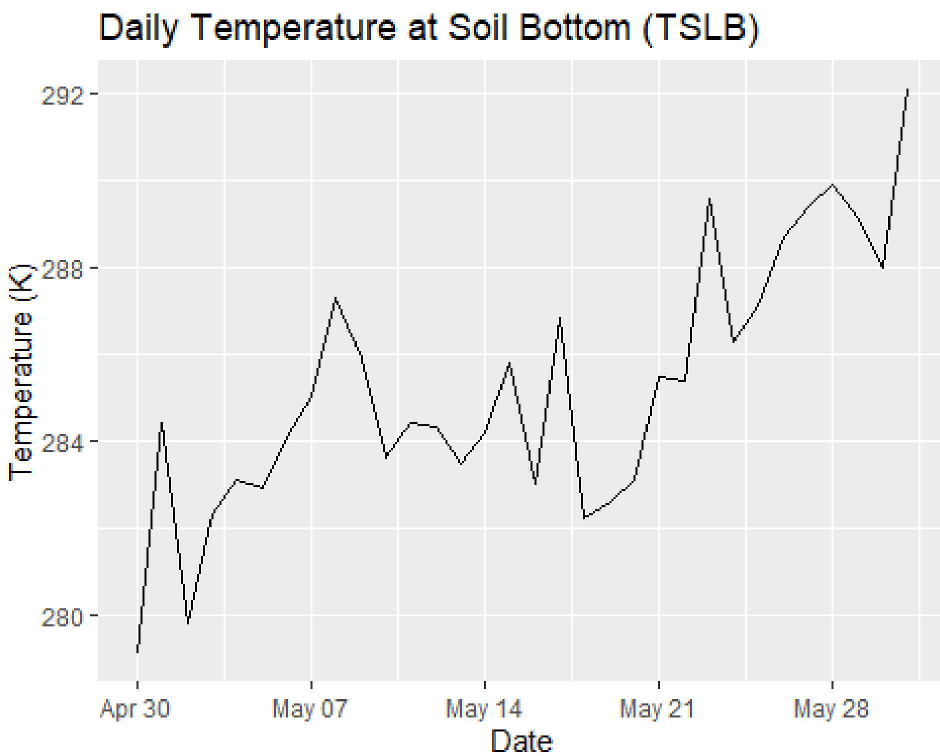
```
head(data_df)

##              DATETIME   TSK   PSFC U10  V10       Q2  TSLB     SMOIS
WIND_SPEED
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370 279.1 0.3572125
3.50000
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330 278.5 0.3517333
3.96000
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980 278.0 0.3551000
4.42000
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010 279.8 0.3522000
4.28000
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700 283.0 0.3479000
4.88000
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.1 0.004355 285.1 0.3441000
7.59625
##   Hour  DayNight       Date Day
## 1   00 Nighttime 2018-04-30  30
## 2   03 Nighttime 2018-05-01  01
## 3   06   Daytime 2018-05-01  01
## 4   09   Daytime 2018-05-01  01
```

```
## 5   12   Daytime 2018-05-01  01
## 6   15   Daytime 2018-05-01  01
```

```r
#Question 1
# Convert Date column to Date type
data_df$Date <- as.Date(data_df$Date, format = "%d/%m/%Y")

# Group by Date and keep the most recent timestamp
data_df1 <- data_df %>%
  group_by(Date) %>%
  slice_max(as.POSIXct(DATETIME, format = "%Y-%m-%d %H:%M:%S")) %>%
  ungroup()
head(data_df1)
```

```
## # A tibble: 6 × 13
##   DATETIME              TSK   PSFC   U10   V10      Q2  TSLB SMOIS
WIND_SPEED
##   <dttm>              <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
<dbl>
## 1 2018-05-01 00:00:00  276. 100218   3.5  -0.1 0.00437  279. 0.357
3.5
## 2 2018-05-02 00:00:00  278. 100208  -0.4   5.5 0.00539  284. 0.338
5.51
## 3 2018-05-03 00:00:00  275. 100793   3.5   0.2 0.00463  280. 0.341
3.51
## 4 2018-05-04 00:00:00  278. 101541   2.9   1.5 0.00579  282. 0.330
3.26
## 5 2018-05-05 00:00:00  277. 101953  -1.4   1   0.00536  283. 0.324
1.72
## 6 2018-05-06 00:00:00  277. 101808  -1.8  -2.2 0.00527  283. 0.319
2.84
## # ℹ 4 more variables: Hour <chr>, DayNight <chr>, Date <date>, Day <chr>
```

```r
# Extract TSLB and Date for analysis
data_subset <- subset(data_df1, select = c(Date, TSLB))  # Selects Date and
TSLB

# Plot TSLB data
ggplot(data_subset, aes(x = Date, y = TSLB)) +
geom_line() +
labs(title = "Daily Temperature at Soil Bottom (TSLB)", x = "Date", y =
"Temperature (K)")
```



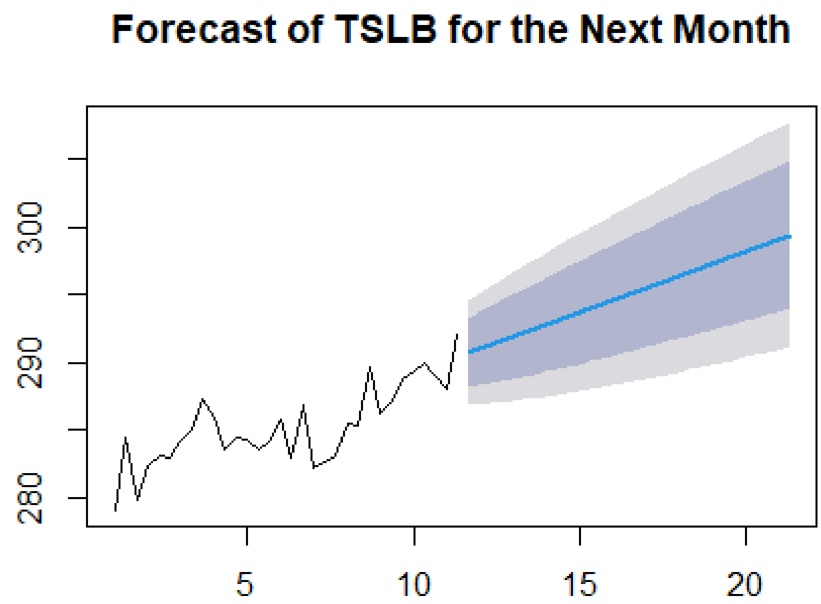
Graph displays daily temperature at Soil Bottom (TSLB).

Temperature Range: The vertical axis represents temperature in Kelvin (K), ranging from approximately 280 to 292 K.

Date Range: The horizontal axis shows dates from April 30 to May 28.

Trend: The graph exhibits fluctuations, but overall, there's an upward trend in temperature over the given period.

```
# Plot TSLB data
# ggplot(tslb_data, aes(x = Date, y = TSLB)) +
#   geom_line() +
#   labs(title = "Daily Temperature at Soil Bottom (TSLB)", x = "Date", y =
"Temperature (K)")

# Convert TSLB data to a time series object
ts_data <- ts(data_subset$TSLB, frequency = 3)

# Fit an ARIMA model
arima_model <- auto.arima(ts_data)

# Forecast the next month
forecast_results <- forecast(arima_model, h = 30)

# Plot the forecast
plot(forecast_results, main = "Forecast of TSLB for the Next Month")
```



In forecast of TSLB for the Next Month: The trend and the forecasted range are evaluated.

Trend: The past data points show a fluctuating trend, starting just above 280 and rising to just below 305 over an 8-day period.

Forecasted Range: The shaded area represents the forecasted range for future values.

Expected Trend: Within the forecasted range, a solid blue line suggests the expected trend for the upcoming month.

Question 7: What trends can we expect in wind speed during Daytime hours in Cuxton over the next quarter?

```
#Question2
#What trends can we expect in wind speed during Daytime hours in Cuxton over
the next quarter?

head(data_df)

##                DATETIME   TSK   PSFC U10  V10       Q2  TSLB     SMOIS
WIND_SPEED
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370 279.1 0.3572125
3.50000
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330 278.5 0.3517333
3.96000
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980 278.0 0.3551000
4.42000
```

```
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010 279.8 0.3522000
4.28000
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700 283.0 0.3479000
4.88000
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.1 0.004355 285.1 0.3441000
7.59625
##   Hour  DayNight       Date Day
## 1   00 Nighttime 2018-04-30  30
## 2   03 Nighttime 2018-05-01  01
## 3   06   Daytime 2018-05-01  01
## 4   09   Daytime 2018-05-01  01
## 5   12   Daytime 2018-05-01  01
## 6   15   Daytime 2018-05-01  01
```

```
str(data_df$Date)
```

```
##  Date[1:248], format: "2018-04-30" "2018-05-01" "2018-05-01" "2018-05-01"
"2018-05-01" ...
```

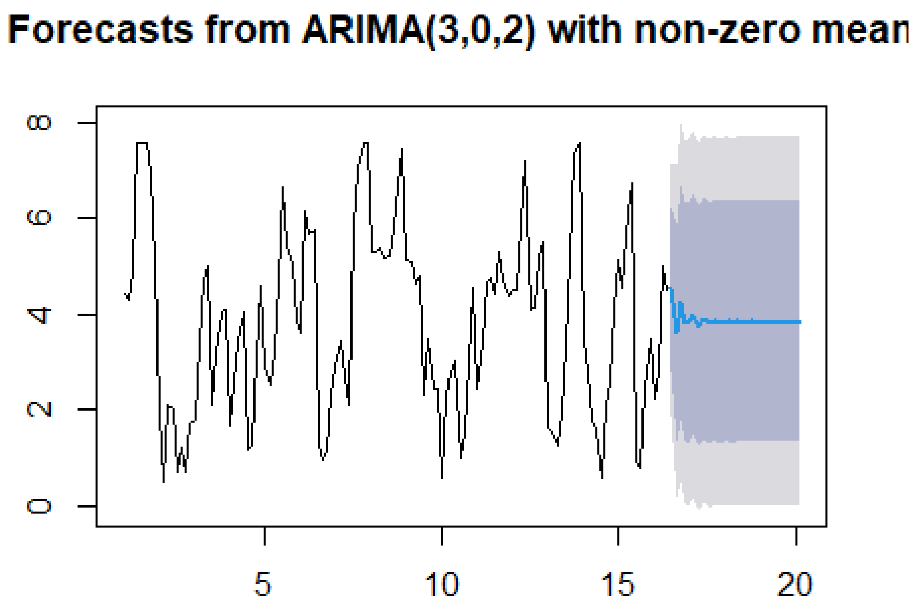
```
# Convert Date column to Date type
data_df$Date <- as.Date(data_df$Date, format = "%d/%m/%Y")

head(data_df$Date)
```

```
## [1] "2018-04-30" "2018-05-01" "2018-05-01" "2018-05-01" "2018-05-01"
## [6] "2018-05-01"
```

```
# Filter WIND_SPEED column for daytime observations
daytime_wind_speed <- subset(data_df, DayNight == "Daytime", select = c(Date,
WIND_SPEED))

nrow(daytime_wind_speed)
```

```
## [1] 124
```

```
# Convert to a ts object
wind_speed_ts <- ts(daytime_wind_speed$WIND_SPEED, frequency = 8) # hourly
data

# Fit an ARIMA model
model <- auto.arima(wind_speed_ts)

# Forecast the next month
forecasted_wind_speed <- forecast(model, h = 30) # 'h' is the number of
periods for forecasting

# Plot the forecast
plot(forecasted_wind_speed)
```



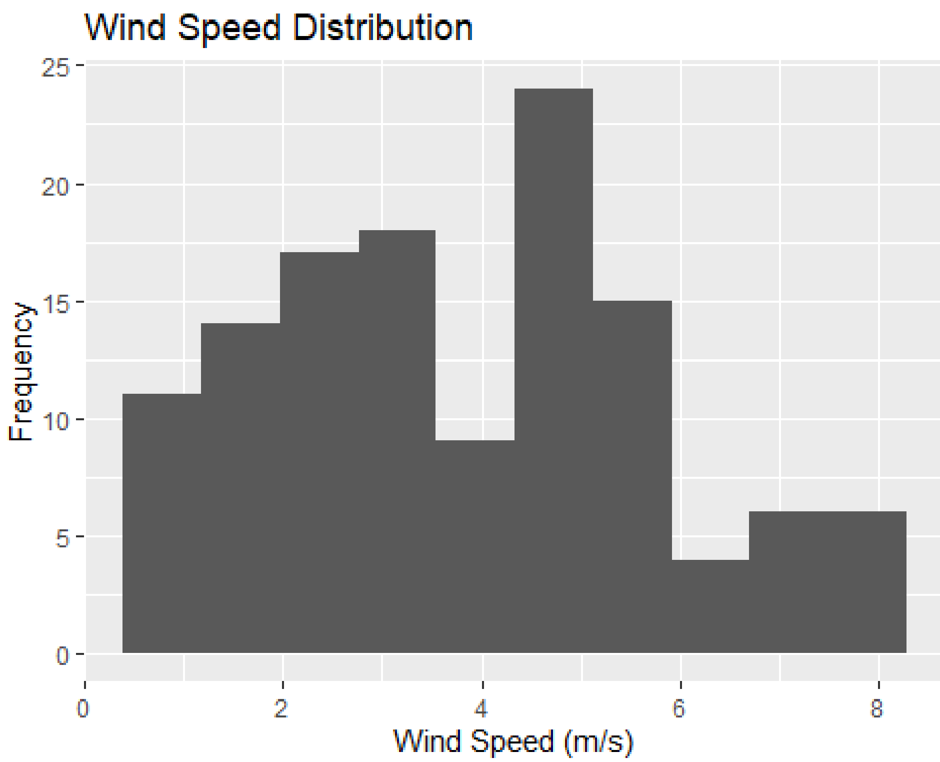
The graph represents a time series data. The x-axis ranges from 0 to 20, and the y-axis from approximately -2 to 8. The black line represents past observations, while the blue line and shaded area indicate forecasts made by an ARIMA(3,0,2) model. The shaded area around the blue line suggests confidence intervals or prediction intervals

If the blue line closely follows the black line, the model is performing well. If the blue line deviates significantly from the black line, further investigation is needed. The confidence intervals provide a range within which future values are likely to fall.

Based on this graph, the ARIMA(3,0,2) model seems to capture the underlying patterns in the data reasonably well.

```
head(data_df)

##                 DATETIME   TSK   PSFC U10  V10       Q2  TSLB     SMOIS
WIND_SPEED
## 1 2018-05-01 00:00:00 276.5 100218 3.5 -0.1 0.004370 279.1 0.3572125
3.50000
## 2 2018-05-01 03:00:00 276.1 100272 3.8  1.1 0.004330 278.5 0.3517333
3.96000
## 3 2018-05-01 06:00:00 277.1 100378 4.4  0.4 0.003980 278.0 0.3551000
4.42000
## 4 2018-05-01 09:00:00 288.6 100436 4.2  0.8 0.004010 279.8 0.3522000
4.28000
## 5 2018-05-01 12:00:00 292.8 100428 3.6  3.3 0.004700 283.0 0.3479000
4.88000
## 6 2018-05-01 15:00:00 289.3 100357 5.2  6.1 0.004355 285.1 0.3441000
```

```
7.59625
##   Hour  DayNight       Date Day
## 1   00 Nighttime 2018-04-30  30
## 2   03 Nighttime 2018-05-01  01
## 3   06   Daytime 2018-05-01  01
## 4   09   Daytime 2018-05-01  01
## 5   12   Daytime 2018-05-01  01
## 6   15   Daytime 2018-05-01  01

head(data_df$Date)

## [1] "2018-04-30" "2018-05-01" "2018-05-01" "2018-05-01" "2018-05-01"
## [6] "2018-05-01"

# Extract wind speed data during daytime hours
wind_speed_data <- data_df %>%
filter(DayNight == "Daytime") %>%
dplyr::select(WIND_SPEED)

# # Extract wind speed data during daytime hours
# wind_speed_data <- data_df %>%
# filter(DayNight == "Daytime") %>%
# dplyr::select(WIND_SPEED)

# Plot wind speed data
ggplot(wind_speed_data, aes(x = WIND_SPEED)) +
  geom_histogram(bins = 10) +
  labs(title = "Wind Speed Distribution", x = "Wind Speed (m/s)", y =
"Frequency")
```

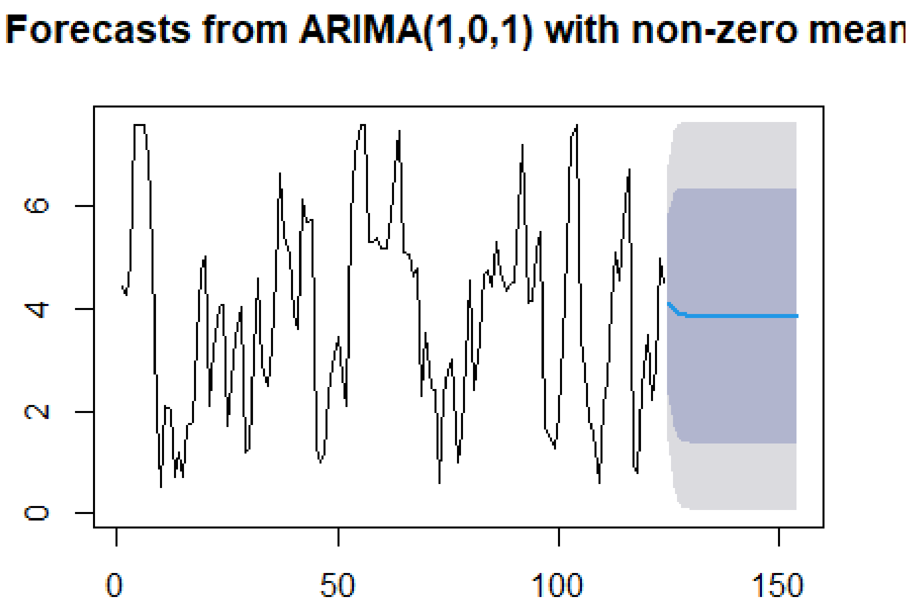


Common Wind Speeds: The graph reveals that certain wind speeds (Like, 5 m/s, 3 m/s) are more common than others.

Application: This information is relevant for meteorology, structural design, and vehicle safety.

```
# Fit ARIMA model
wind_speed_model <- Arima(wind_speed_data, order = c(1,0,1), seasonal =
c(1,1,1))
summary(wind_speed_model)

## Series: wind_speed_data
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1    mean
##       0.5614  0.2592  3.8371
## s.e.  0.0945  0.1005  0.3458
##
## sigma^2 = 1.887:  log likelihood = -214.16
## AIC=436.31   AICc=436.65   BIC=447.59
##
## Training set error measures:
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.003439187 1.356922 1.033325 -27.49629 48.01289 0.9370493
##                    ACF1
## Training set 0.02184902
```

```r
# Forecast wind speed for the next month
wind_speed_forecast <- forecast(wind_speed_model, h = 30)
plot(wind_speed_forecast)
```



Training Set Error Measures: Autoregressive (AR) coefficient (ar1): 0.5614 Moving average (MA) coefficient (ma1): 0.2592 Non-zero mean: 3.8371

Mean Error (ME): -0.0034 Root Mean Squared Error (RMSE): 1.3569 Mean Absolute Error (MAE): 1.0333 Mean Percentage Error (MPE): -27.50% Mean Absolute Percentage Error (MAPE): 48.01% Mean Absolute Scaled Error (MASE): 0.9370 Autocorrelation of Residuals (ACF1): 0.0218

Model Coefficients: The AR coefficient (ar1) is 0.5614, indicating a moderate positive impact of the previous observation on the current value.

The MA coefficient (ma1) is 0.2592, suggesting that short-term fluctuations play a role in the data.

The non-zero mean (3.8371) contributes to the baseline level of the time series. Variance and Log Likelihood:

The estimated variance (sigma^2) is 1.887, which quantifies the variability in the data. The negative log likelihood (-214.16) indicates how well the model fits the data.

The **graph represents** forecasts generated using an ARIMA(1,0,1) model. ARIMA stands for AutoRegressive Integrated Moving Average, and it's commonly used for time series forecasting. Fluctuating Trends: The line plot oscillates and trends over time, suggesting variations in the underlying data.

Forecasted Values: The blue highlighted section shows the forecasted values based on the ARIMA model. These predictions are within a shaded area, likely representing confidence intervals or prediction bounds.