Forecasting Wind Speed at Dogger Bank

Mehrnaz Kashfi

2024-05-21

Data Preprocessing

Data preprocessing involves preparing the dataset for analysis which includes loading the data, filtering based on specific conditions, and ensuring data types are appropriate for analysis. This section is detailed to ensuring the dataset is ready for in-depth analysis without repeating the specific steps here.

Loading Data & Geographic Filtering & Removing LAT, LONG

```r
# Read the dataset and skip first row
data <- read.csv("C:/Users/User/Documents/WRFdata_May2018.csv", header = T,
skip = 1)

dim(data)

## [1] 5451 2482

# Filter location
data <- data[!is.na(data$XLAT) & !is.na(data$XLONG), ]

# Convert the first two columns to numeric type if they are not already
data$XLAT <- as.numeric(as.character(data$XLAT))
data$XLONG <- as.numeric(as.character(data$XLONG))

# Now, ensure that the latitude and longitude values are within the
appropriate range
# Latitude ranges from -90 to 90 and Longitude ranges from -180 to 180
data <- data[data$XLAT >= -90 & data$XLAT <= 90 & data$XLONG >= -180 &
data$XLONG <= 180, ]

# Filter the data based on geographic coordinates
data <- data[data$XLAT == 54.51 & data$XLONG == 1.947, ]
dim(data)

## [1]    1 2482

#Assuming that latitude and longitude are not required for ongoing analysis..
data <- data[, -c(1, 2)]

dim(data)

## [1]    1 2480
```

Reshaping data

```
# Define column names
col_names <- c("TSK", "PSFC", "U10", "V10", "Q2", "RAINC", "RAINNC", "SNOW",
"TSLB", "SMOIS")

# Define column indexes for each row
cols <- seq(1, ncol(data), by=10)
length(cols) #2480/10 = 248

## [1] 248

#Generates an empty vector
emp_df <- c()

#Reread the dataset, without 2 first columns
data2 <- read.csv("C:/Users/User/Documents/WRFdata_May2018.csv", header = T)
%>% dplyr::select(-c(1,2))


#Reshap the dataset
for (i in seq_along(cols)) {
  df0 <- data %>% dplyr::select(cols[i]:(cols[i]+9)) #Selects a subset of
data from 'data' that start from cols[i]

  dt <- colnames(data2)[cols[i]] #Extract the column name of dates from data2

  df0_ll <- cbind.data.frame(dt, df0) #Combining the selected data with the
date name

  colnames(df0_ll) <- c("date", col_names) #Setting column name of date for
df0_ll

  emp_df <- rbind.data.frame(emp_df, df0_ll) #Appending the new row to the
existing dataframe

}


#Correct the value of date_time columns
date_time <- str_remove(emp_df$date, "X")

date_time <- gsub(".", "-", date_time, fixed = TRUE)

emp_df$date <- sub("X\\.2225", "X31.05.2018.21.00", emp_df$date)


#Remove the column of date to date_time
new_df <- emp_df %>%
  dplyr::select(-date) %>%
```

```r
  mutate(date_time=date_time) %>%
  relocate(date_time, .before = TSK)


dim(new_df)
```

```
## [1] 248  11
```

```r
head(new_df)
```

```
##              date_time   TSK   PSFC U10   V10      Q2 RAINC RAINNC SNOW
TSLB
## 4702  01-05-2018-00-00 280.5 100053 5.5 -11.7 0.00539     0    0.9    0
273.2
## 47021 01-05-2018-03-00 280.5 100124 7.9 -10.5      NA     0    1.0    0
273.2
## 47022 01-05-2018-06-00 280.5 100245 9.6  -8.9 0.00513     0    1.0    0
273.2
## 47023 01-05-2018-09-00 280.5 100453 9.4  -4.7 0.00493     0    1.0    0
273.2
## 47024 01-05-2018-12-00 280.5 100613 6.9   1.5 0.00525     0    1.0    0
273.2
## 47025 01-05-2018-15-00 280.5 100621 2.4   7.1 0.00535     0    1.0    0
273.2
##       SMOIS
## 4702      1
## 47021     1
## 47022     1
## 47023     1
## 47024     1
## 47025     1
```

Calculating wind speed

```r
#Step1: Conversion to Numeric Data Type
new_df$U10 <- as.numeric(as.character(new_df$U10))
new_df$V10 <- as.numeric(as.character(new_df$V10))
na_count_u10 <- sum(is.na(new_df$U10))
na_count_v10 <- sum(is.na(new_df$V10))
print(paste("Number of NA values in U10: ", na_count_u10))
```

```
## [1] "Number of NA values in U10:  6"
```

```r
print(paste("Number of NA values in V10: ", na_count_v10))
```

```
## [1] "Number of NA values in V10:  1"
```

```r
#Step2: Handling Missing Values
replace_na_with_neighbors <- function(x) {
  # Loop through each element in the vector
  for (i in 1:length(x)) {
    # Check if the current element is NA
```

```r
    if (is.na(x[i])) {
      # Find non-NA values before and after the current NA
      before <- x[1:i][!is.na(x[1:i])]
      after <- x[i:length(x)][!is.na(x[i:length(x)])]

      # Use the last value from 'before' and the first from 'after'
      if (length(before) > 0 && length(after) > 0) {
        x[i] <- mean(c(tail(before, 1), head(after, 1)), na.rm = TRUE)
      } else if (length(before) > 0) {
        x[i] <- tail(before, 1)
      } else if (length(after) > 0) {
        x[i] <- head(after, 1)
      }
    }
  }
  return(x)
}
new_df$U10 <- replace_na_with_neighbors(new_df$U10)
new_df$V10 <- replace_na_with_neighbors(new_df$V10)
sum(is.na(new_df$U10))  # Should be 0 if all NAs were replaced

## [1] 0

sum(is.na(new_df$V10))  # Should be 0 if all NAs were replaced

## [1] 0

#Step3: Wind Speed Calculation
new_df$WindSpeed <- sqrt(new_df$U10^2 + new_df$V10^2)

sum(is.infinite(new_df$WindSpeed))  # Inf or -Inf

## [1] 0

sum(is.na(new_df$WindSpeed))        # NA

## [1] 0

#After calculate wind speed, delete U10 and V10
glimpse(new_df)

## Rows: 248
## Columns: 12
## $ date_time <chr> "01-05-2018-00-00", "01-05-2018-03-00", "01-05-2018-06-
00", …
## $ TSK       <dbl> 280.5, 280.5, 280.5, 280.5, 280.5, 280.5, 280.5, 280.5,
280.…
## $ PSFC      <int> 100053, 100124, 100245, 100453, 100613, 100621, 100587,
1005…
## $ U10       <dbl> 5.5, 7.9, 9.6, 9.4, 6.9, 2.4, -0.1, 1.4, 3.2, 1.2, -0.8,
-1.…
## $ V10       <dbl> -11.7, -10.5, -8.9, -4.7, 1.5, 7.1, 9.9, 9.4, 10.8,
```

```
12.7, 14…
## $ Q2        <dbl> 0.00539, NA, 0.00513, 0.00493, 0.00525, 0.00535, 
0.00521, 0.…
## $ RAINC     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, …
## $ RAINNC    <dbl> 0.9, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 
0.2, …
## $ SNOW      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 
0, …
## $ TSLB      <dbl> 273.2, 273.2, 273.2, 273.2, 273.2, 273.2, 273.2, 273.2, 
273.…
## $ SMOIS     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 
1, …
## $ WindSpeed <dbl> 12.928264, 13.140015, 13.090836, 10.509519, 7.061161, 
7.4946…

new_df <- subset(new_df, select = -c(V10, U10))
```

Preprocessing for another columns

In this part, first check NA values and handle them, then detect and delete some variable that don’t have enough variation to check correlation test and another steps.

For handling missing data in secondary variables, linear interpolation was used. This method assumes that changes between available data points are gradual and linear, helping to estimate missing values effectively. This choice was made due to its simplicity and efficiency in maintaining the overall trend of the data without adding unnecessary complexity to the analysis.

```
# Check the structure of the data to confirm changes
str(new_df) # because in first steps, convert all columns to numeric just 
check them

## 'data.frame':    248 obs. of  10 variables:
##  $ date_time: chr  "01-05-2018-00-00" "01-05-2018-03-00" "01-05-2018-06-
00" "01-05-2018-09-00" ...
##  $ TSK      : num  280 280 280 280 280 ...
##  $ PSFC     : int  100053 100124 100245 100453 100613 100621 100587 100530 
100516 100447 ...
##  $ Q2       : num  0.00539 NA 0.00513 0.00493 0.00525 0.00535 0.00521 
0.00576 0.00578 0.00557 ...
##  $ RAINC    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ RAINNC   : num  0.9 1 1 1 1 1 1 1 0 0 ...
##  $ SNOW     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TSLB     : num  273 273 273 273 273 ...
##  $ SMOIS    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ WindSpeed: num  12.93 13.14 13.09 10.51 7.06 ...

#Check NA values
# Sum of NA values in each column
na_count <- sapply(new_df, function(x) sum(is.na(x)))
```

```
# Print the results
print(na_count)

## date_time       TSK      PSFC        Q2     RAINC    RAINNC      SNOW
TSLB
##         0         7        10         7         9         7         9
9
##     SMOIS WindSpeed
##         4         0

# Apply linear interpolation to each column of the dataset
# We use lapply to apply the function to each column separately
data_interpolated <- data.frame(lapply(new_df, function(x) {
  # Check if there are any NA values in the column
  if (any(is.na(x))) {
    # Perform linear interpolation
    return(na.approx(x, na.rm = FALSE))
  } else {
    # Return the column unchanged if no NA values are present
    return(x)
  }
}))
na_count <- sapply(data_interpolated, function(x) sum(is.na(x)))
# Print the results to recheck NA
print(na_count)

## date_time       TSK      PSFC        Q2     RAINC    RAINNC      SNOW
TSLB
##         0         0         0         0         0         0         0
0
##     SMOIS WindSpeed
##         0         0

#Delete variables with limited variation
#Checking variables with limited variation
summary(data_interpolated)

##   date_time              TSK             PSFC              Q2
##  Length:248         Min.   :280.2   Min.   :100053   Min.   :0.004930
##  Class :character   1st Qu.:282.5   1st Qu.:101517   1st Qu.:0.006490
##  Mode  :character   Median :283.1   Median :102047   Median :0.007270
##                     Mean   :283.0   Mean   :101880   Mean   :0.007138
##                     3rd Qu.:283.7   3rd Qu.:102399   3rd Qu.:0.007570
##                     Max.   :285.7   Max.   :102744   Max.   :0.008990
##      RAINC       RAINNC            SNOW        TSLB             SMOIS
##  Min.   :0   Min.   :0.0000   Min.   :0   Min.   :273.2   Min.   :1
##  1st Qu.:0   1st Qu.:0.0000   1st Qu.:0   1st Qu.:273.2   1st Qu.:1
##  Median :0   Median :0.0000   Median :0   Median :273.2   Median :1
##  Mean   :0   Mean   :0.1044   Mean   :0   Mean   :273.2   Mean   :1
##  3rd Qu.:0   3rd Qu.:0.0000   3rd Qu.:0   3rd Qu.:273.2   3rd Qu.:1
##  Max.   :0   Max.   :4.7000   Max.   :0   Max.   :273.2   Max.   :1
```

```
##    WindSpeed
##  Min.   : 0.8246
##  1st Qu.: 4.5757
##  Median : 6.1503
##  Mean   : 6.2698
##  3rd Qu.: 7.3466
##  Max.   :15.2948
```

```r
#Removing variables with limited variation
data_interpolated <- data_interpolated[, !(names(data_interpolated) %in%
c("RAINC", "SNOW", "SMOIS", "TSLB"))]
#Check the structure of the data after removal
summary(data_interpolated)
```

```
##   date_time              TSK              PSFC              Q2
##  Length:248         Min.   :280.2   Min.   :100053   Min.   :0.004930
##  Class :character   1st Qu.:282.5   1st Qu.:101517   1st Qu.:0.006490
##  Mode  :character   Median :283.1   Median :102047   Median :0.007270
##                     Mean   :283.0   Mean   :101880   Mean   :0.007138
##                     3rd Qu.:283.7   3rd Qu.:102399   3rd Qu.:0.007570
##                     Max.   :285.7   Max.   :102744   Max.   :0.008990
##      RAINNC          WindSpeed
##  Min.   :0.0000   Min.   : 0.8246
##  1st Qu.:0.0000   1st Qu.: 4.5757
##  Median :0.0000   Median : 6.1503
##  Mean   :0.1044   Mean   : 6.2698
##  3rd Qu.:0.0000   3rd Qu.: 7.3466
##  Max.   :4.7000   Max.   :15.2948
```

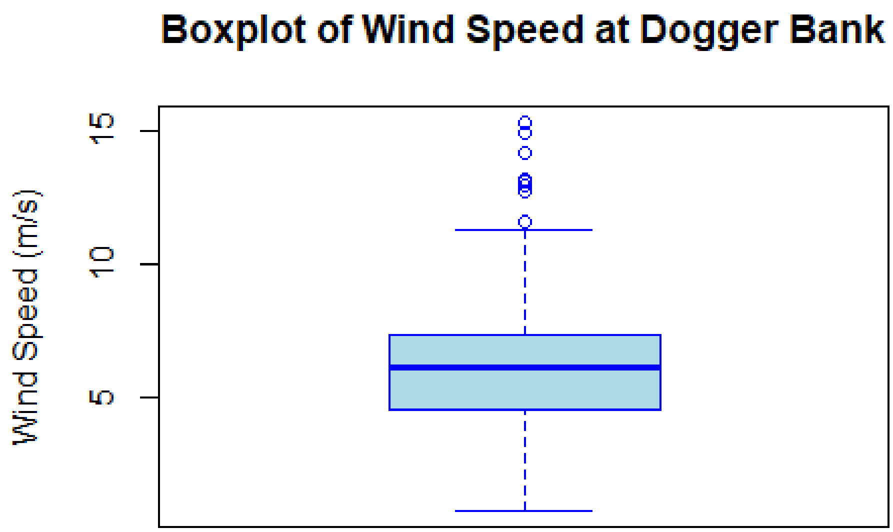
Exploratory Data Analysis (EDA)

In this part, exploratory data analysis seeks out structures, ouliers, or distribution in wind speed data that may have an impact on prediction models. This stage is necessary to lay the framework for robust, data-driven insights.

```r
# Basic statistics
summary(data_interpolated$WindSpeed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8246  4.5757  6.1503  6.2698  7.3466 15.2948
```

```r
#Outlier Detection
#Find outliers with Boxplot
boxplot(data_interpolated$WindSpeed,
        main = "Boxplot of Wind Speed at Dogger Bank",
        ylab = "Wind Speed (m/s)",
        col = "lightblue",
        border = "blue")
```



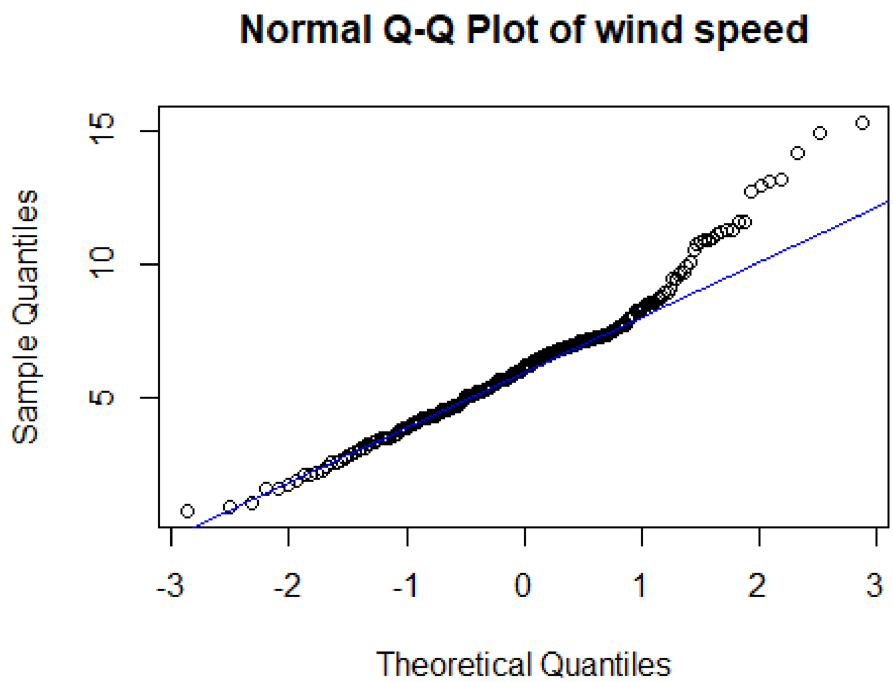

```
#Check outliers
#Before check outliers is needed to have just numeric columns
numeric_columns <- sapply(data_interpolated, is.numeric)
data_numeric <- data_interpolated[, numeric_columns]
all(sapply(data_numeric, is.numeric))

## [1] TRUE

# Function to find outliers using IQR
iqr_outliers <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR
  return(which(x < lower | x > upper))
}
# Apply the function to each numeric column
iqr_outliers_list <- sapply(data_numeric, iqr_outliers)
iqr_outliers_list  # Display indices of outliers

## $TSK
##  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
18  19
## [20]  20  21  22  23  24 225 226 227 228 229 230 231 232 241 242 243 244
245 246
## [39] 247 248
```

```
##
## $PSFC
## [1] 1 2
##
## $Q2
## integer(0)
##
## $RAINNC
##  [1]   1   2   3   4   5   6   7   8  12  13  14  15  16 115 116 117 118
119 120
## [20] 125 126 127 128 237 238 239 240 248
##
## $WindSpeed
## [1]   1   2   3  10  11  12  13 127 128
```

```
summary(data_numeric)
```

```
##       TSK              PSFC              Q2                RAINNC
##  Min.   :280.2   Min.   :100053   Min.   :0.004930   Min.   :0.0000
##  1st Qu.:282.5   1st Qu.:101517   1st Qu.:0.006490   1st Qu.:0.0000
##  Median :283.1   Median :102047   Median :0.007270   Median :0.0000
##  Mean   :283.0   Mean   :101880   Mean   :0.007138   Mean   :0.1044
##  3rd Qu.:283.7   3rd Qu.:102399   3rd Qu.:0.007570   3rd Qu.:0.0000
##  Max.   :285.7   Max.   :102744   Max.   :0.008990   Max.   :4.7000
##    WindSpeed
##  Min.   : 0.8246
##  1st Qu.: 4.5757
##  Median : 6.1503
##  Mean   : 6.2698
##  3rd Qu.: 7.3466
##  Max.   :15.2948
```

```
#The outliers that are detected, all they are in the range of variables. So
it doesn't need to handle.


# Normality Check
# QQplot of normally distributed values
qqnorm(data_interpolated$WindSpeed, main = "Normal Q-Q Plot of wind speed")
# Add qqline to plot
qqline(data_interpolated$WindSpeed, col = "blue")
```



These figures illustrate that, while wind speeds on Dogger Bank are usually modest and consistent, they can be unusually high or low at times. Identifying high wind speed outliers can help with planning, risk management, and wind energy optimisation. This combined understanding of wind speed extremes and averages enables Dogger Bank to build more exact models for wind resource forecasts and management.

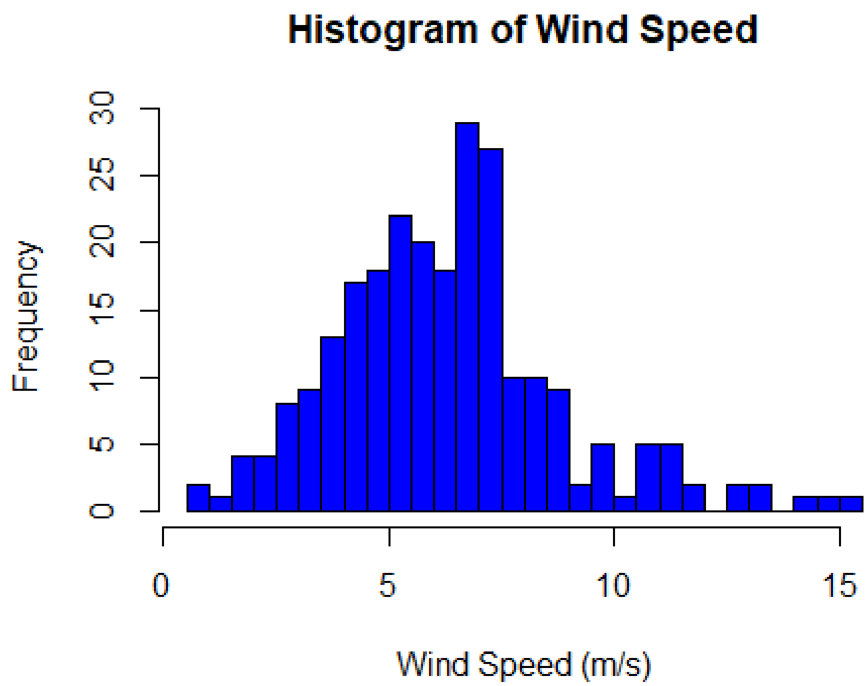
Statistical Analyis

The study's following section includes a statistical analysis to validate the hypotheses based on the original research questions. This section examines the correlations and impacts of several environmental factors on wind speed at Dogger Bank using univariate, bivariate, and multivariate analysis.For correlation test, according Shapiro-Wilk Test result that shows non-parametric and the type of data that is numeric except date_time column, spearman is selected.

Univariate Analysis

Examining Wind Speed Distribution

```
# Histogram of Wind Speed
hist(new_df$WindSpeed, main = "Histogram of Wind Speed", xlab = "Wind Speed
(m/s)", col = "blue", breaks = 31)
```




```r
# Perform Shapiro-Wilk Test for wind speed
shapiro_test <- shapiro.test(new_df$WindSpeed)
# show result
shapiro_test

##
##  Shapiro-Wilk normality test
##
## data:  new_df$WindSpeed
## W = 0.96495, p-value = 9.143e-06

#p<0.05 reject null hypothesis,data is not normally distributed.
```

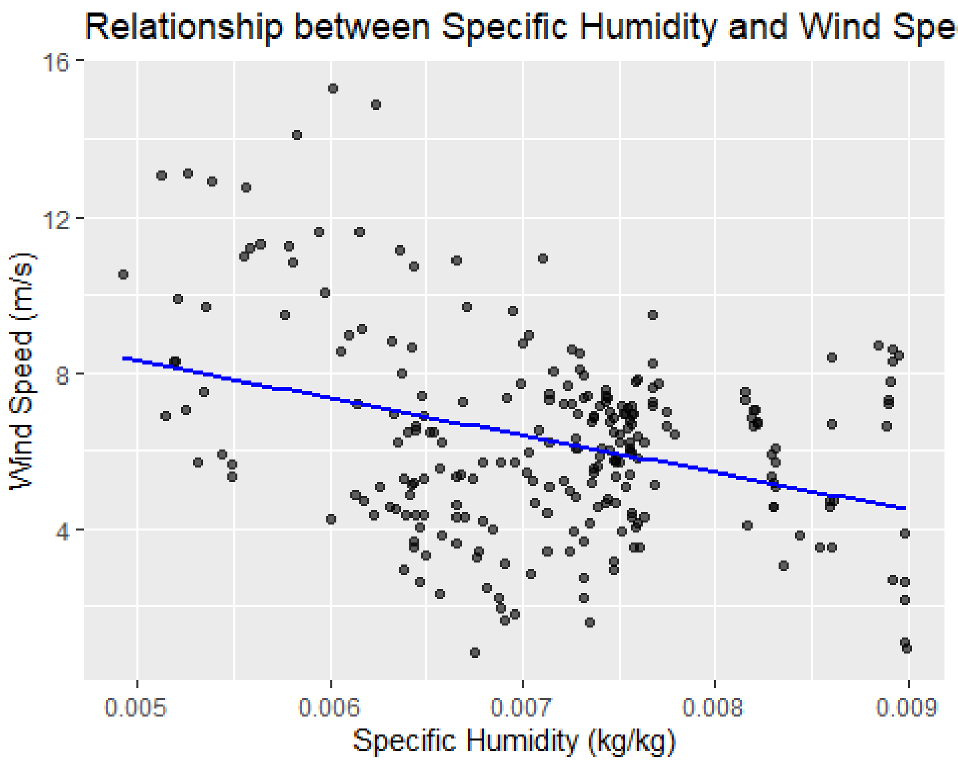
The wind speed data's Shapiro-Wilk test result clearly indicates that it does not follow a normal distribution. This supports the use of non-parametric techniques for further statistical analysis by validating the presence of skewness or other non-normal features found in the histogram.

Bivariate Analysis

Analyse the relationship between specific humidity (Q2) and wind speed at Dogger Bank

```r
# Plotting the relationship between Specific Humidity and Wind Speed
ggplot(data_interpolated, aes(x = Q2, y = WindSpeed)) +
  geom_point(alpha = 0.6) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  labs(title = "Relationship between Specific Humidity and Wind Speed",
       x = "Specific Humidity (kg/kg)", y = "Wind Speed (m/s)")
```

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



The scatter plot indicates a small inverse relationship between specific humidity (Q2) and wind speed at Dogger Bank, as evidenced by the trend line's negative slope.

```
#Correlation Test
cor.test(data_interpolated$WindSpeed, data_interpolated$Q2, method =
"spearman")

## Warning in cor.test.default(data_interpolated$WindSpeed,
data_interpolated$Q2,
## : Cannot compute exact p-value with ties

##
##  Spearman's rank correlation rho
##
## data:  data_interpolated$WindSpeed and data_interpolated$Q2
## S = 3013581, p-value = 0.003375
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## -0.1854578

#p<0.05 reject null hypothesis, that means correlation between Q2 and Wind
Speed
```

The value rho indicates a slightly negative association (-0.1854578) between wind speed and Q2.The p-value of 0.003375 suggests that this association is statistically significant, implying that it was unlikely to happen by chance if there was no true relationship.

```r
linear_model <- lm(WindSpeed ~ Q2, data = data_interpolated)

summary(linear_model)  # Displays the regression output including
coefficients

##
## Call:
## lm(formula = WindSpeed ~ Q2, data = data_interpolated)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8157 -1.7016 -0.0724  1.4211  7.9571
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13.088      1.203  10.883  < 2e-16 ***
## Q2          -955.185    167.161  -5.714 3.17e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.364 on 246 degrees of freedom
## Multiple R-squared:  0.1172, Adjusted R-squared:  0.1136
## F-statistic: 32.65 on 1 and 246 DF,  p-value: 3.175e-08
```

The analysis shows a significant intercept for the base wind speed at 13.088 m/s when specific humidity is zero, and a substantial negative slope of -955.185 for specific humidity, indicating that an increase in humidity significantly decreases wind speed by about 955 m/s. Although the model explains about 11.72% of the variability in wind speed, demonstrating that other factors also significantly influence wind speed, the highly significant F-statistic confirms the relationship between specific humidity and wind speed is statistically reliable and not random.

According to the findings, the correlation between specific humidity (Q2) and wind speed is statistically significant ($p < 0.05$), indicating that it is not due to random chance. However, the model explains only 11.72% of the variability in wind speed, suggesting that the impact of specific humidity, while significant, is limited.

Multivariate Analysis

The purpose of multivariate analysis in this study is to understand how different climatic variables affect wind speed at Dogger Bank at the same time.

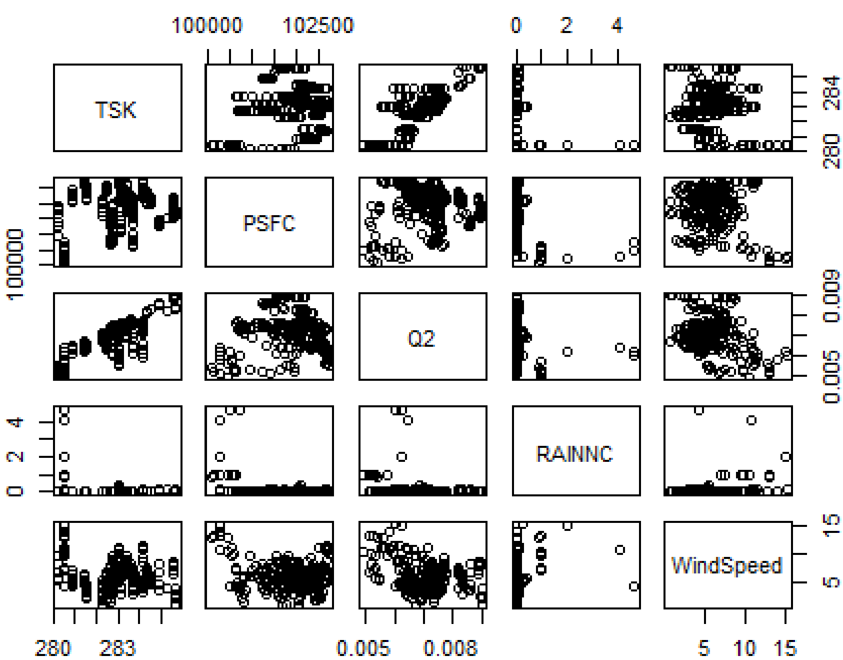
Checking Multiple Regression Assumptions

```r
#Assumptions
#1. Have linear relationship (scatter plot)
#2. Errors/Residuals are normally distributed
```

```r
#3. Errors are independent and there is no autocorrelation between errors
#4. Constant error variance- homoscedasticity of residuals or equal variance
#i.e. variance around regression line is same for all values of other
predictor variables
#5. No multi-collinearity between predictors
head(data_interpolated)
```

```
##            date_time   TSK   PSFC      Q2 RAINNC WindSpeed
## 1 01-05-2018-00-00 280.5 100053 0.00539    0.9 12.928264
## 2 01-05-2018-03-00 280.5 100124 0.00526    1.0 13.140015
## 3 01-05-2018-06-00 280.5 100245 0.00513    1.0 13.090836
## 4 01-05-2018-09-00 280.5 100453 0.00493    1.0 10.509519
## 5 01-05-2018-12-00 280.5 100613 0.00525    1.0  7.061161
## 6 01-05-2018-15-00 280.5 100621 0.00535    1.0  7.494665
```

```r
dataset <- data_interpolated[2:6]

regressor<- lm(WindSpeed ~., dataset)
#summary(regressor)


#1. Have linear relationship (scatter plot)
pairs(dataset)
```



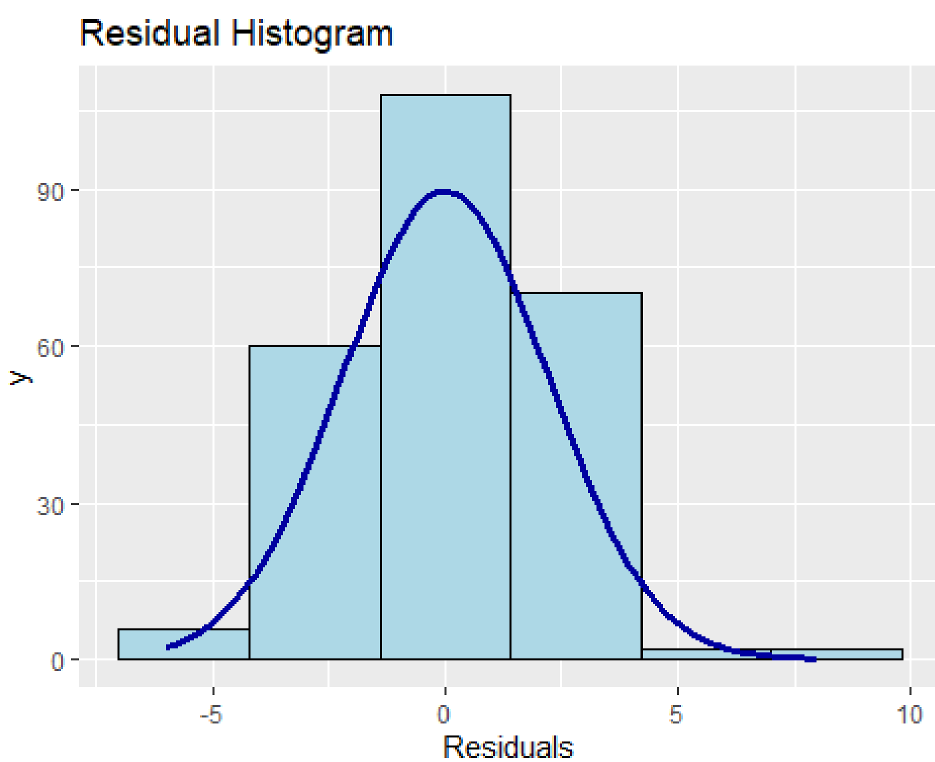
In the scatterplot matrix, WindSpeed does not show any clear relationships with TSK, PSFC, or Q2, indicating no strong or linear correlations between these variables and wind speed.

Similarly, with RAINNC, there is no obvious pattern as the points are broadly scattered, suggesting no direct relationship exists.

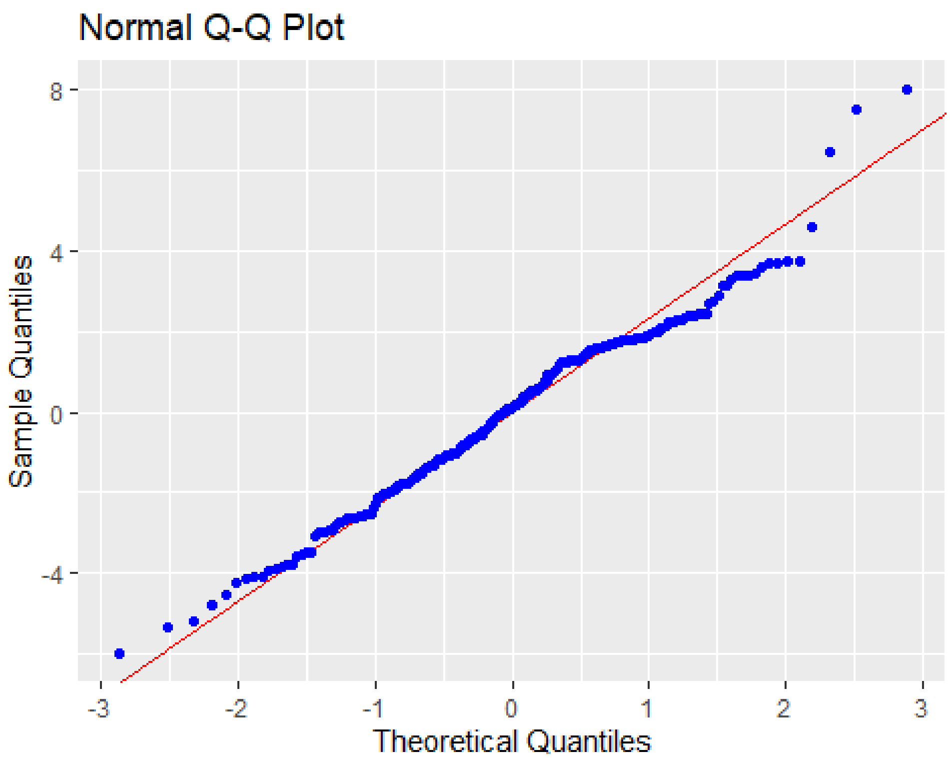
```
#Check for linearity
#rainbow test for checking linearity (lmtest package)
#p<0.05 means non-linearity
raintest(regressor)
```

```
##
##  Rainbow test
##
## data:  regressor
## Rain = 1.7235, df1 = 124, df2 = 119, p-value = 0.001497
```

```
#The result show that data is non-linearity.

#2. Errors/Residuals are normally distributed
ols_plot_resid_hist(regressor)
```




```
ols_plot_resid_qq(regressor)
```




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

##
##  Shapiro-Wilk normality test
##
## data:  regressor$residuals
## W = 0.98314, p-value = 0.004895

#Result shows that is non-distribution.

#3. Errors are independent and there is no autocorrelation between errors
#null hypo: there is no autocorrelation (errors are independent)
dwtest(regressor) #since p>0.05 - there is no autocorrelation

##
##  Durbin-Watson test
##
## data:  regressor
## DW = 0.24215, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0

# The result shows autocorrelation and reject null hypothesis.

#4. Constant error variance- homoscedasticity of residuals or equal variance
#No hetroscedasticity
#homoscedasticity:variance around regression line is same for all values of
```

```r
the predictor variables
ncvTest(regressor) #p>0.05, no hetroscedasticity

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 16.68785, Df = 1, p = 4.4062e-05

#The result show that is hetroscedasticity because it's less than 0.05.

# 5. No multi-collinearity between predictors
#No multicollinearity
# two or more predictor variables are highly correlated
vif(regressor)  #vif>10 strong multicollinearity

##      TSK     PSFC       Q2   RAINNC
## 4.311142 1.260925 4.064979 1.266603

#Results shows that data is multicollinearity because vif is less than 10

#summary(regressor)$r.squared

#lm_model <- lm(WindSpeed ~ TSK + PSFC + Q2 + RAINNC, data =
data_interpolated)
summary(regressor)

##
## Call:
## lm(formula = WindSpeed ~ ., data = dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0045 -1.5478  0.1112  1.6176  8.0156
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.270e+02  5.797e+01  -3.915 0.000117 ***
## TSK          1.158e+00  2.151e-01   5.384 1.71e-07 ***
## PSFC        -7.617e-04  2.459e-04  -3.097 0.002182 **
## Q2          -2.380e+03  3.177e+02  -7.492 1.25e-12 ***
## RAINNC       3.375e-01  2.985e-01   1.131 0.259310
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.229 on 243 degrees of freedom
## Multiple R-squared:  0.2253, Adjusted R-squared:  0.2125
## F-statistic: 17.67 on 4 and 243 DF,  p-value: 9.615e-13
```

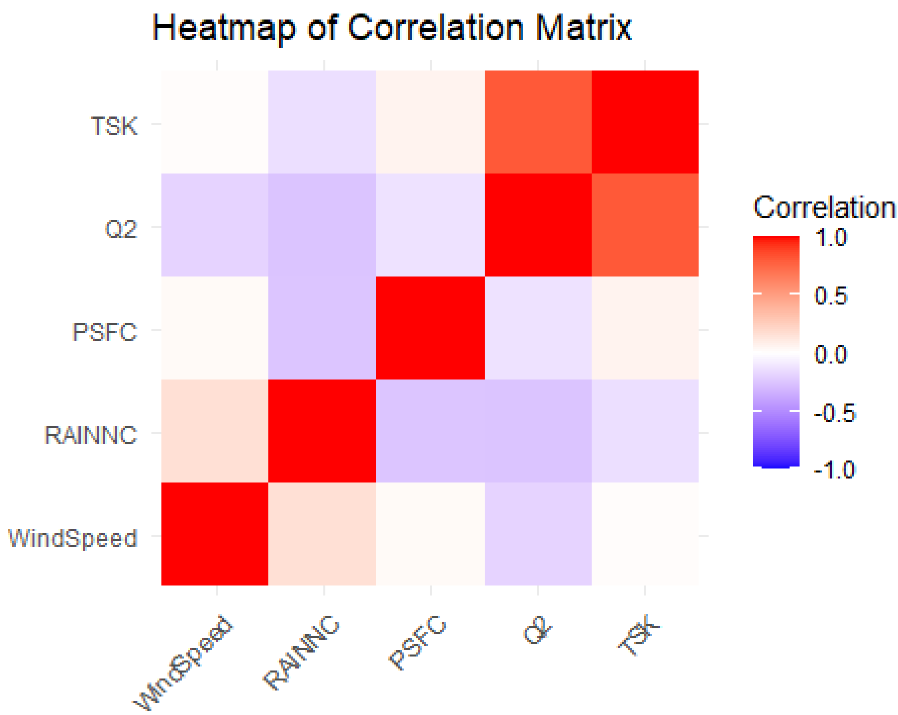
The regression analysis of wind speed data at Dogger Bank reveals substantial limitations due to key assumptions being violated: the presence of non-linearity confirmed by the Rainbow test (p-value = 0.002937), heteroscedasticity indicated by a Non-constant Variance Score Test (p-value = 4.4062e-05), and non-normally distributed residuals as shown by the Shapiro-Wilk test (p-value = 0.004895). Additionally, significant autocorrelation in the residuals (p-value < 2.2e-16), despite acceptable VIF values, complicates the model's reliability. With an adjusted R-squared value of 0.2253, indicating limited explanatory power, the results suggest that multiple linear regression is unsuitable for accurately predicting wind speeds at Dogger Bank.

Correlation Analysis & Heatmap

```r
cor_matrix <- cor(data_interpolated[, c("WindSpeed", "RAINNC", "PSFC", "Q2",
"TSK")], method = "spearman")
#Correlation Analysis
#cor_matrix <- cor(data_interpolated[, sapply(data_interpolated,
is.numeric)], method = "spearman")
print(cor_matrix)
```

```
##             WindSpeed     RAINNC        PSFC         Q2         TSK
## WindSpeed  1.00000000  0.1624699  0.02872617 -0.1854578  0.01679236
## RAINNC     0.16246992  1.0000000 -0.24686909 -0.2541748 -0.13795838
## PSFC       0.02872617 -0.2468691  1.00000000 -0.1247916  0.06267968
## Q2        -0.18545785 -0.2541748 -0.12479155  1.0000000  0.80890606
## TSK        0.01679236 -0.1379584  0.06267968  0.8089061  1.00000000
```

```r
#Heatmap
cor_data <- melt(cor_matrix)
ggplot(cor_data, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint =
0, limit = c(-1, 1)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "", y = "", title = "Heatmap of Correlation Matrix", fill =
"Correlation")
```



Q2 and WindSpeed (-0.1855): This negative correlation suggests that higher levels of specific humidity are slightly associated with lower wind speeds, which might indicate moisture-laden air being heavier and potentially less mobile.

RAINNC and WindSpeed (0.1625): A moderate positive correlation, which might imply that increased cumulative rainfall (RAINNC) correlates with slight increases in wind speed, perhaps due to the atmospheric disturbances rain can cause.

PSFC and WindSpeed (0.0287): Shows a very weak positive correlation between surface pressure and wind speed, suggesting that changes in surface pressure have a minimal direct impact on wind speed.

Time Series

In this part, we analyse wind speed data to improve forecasting accuracy by using machine learning methods integrated with time-series analysis. We explore linear modeling and SARIMA models.

STEP1: Read dataset

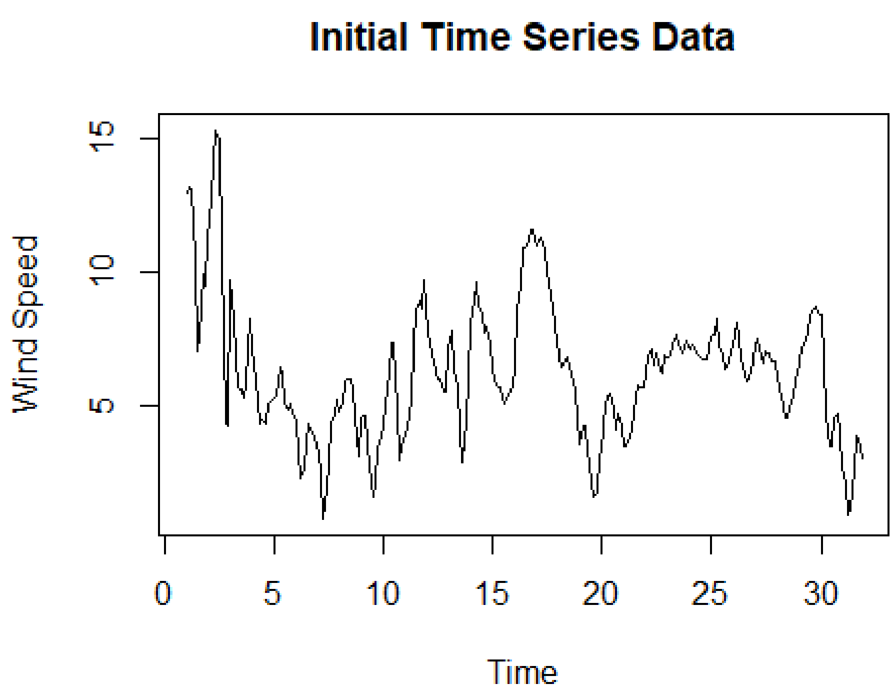
```
dataset <- data_interpolated
dataset$date_time <- as.POSIXct(dataset$date_time, format = "%d-%m-%Y-%H-%M")
timeseries_data <- dataset[, c(1, ncol(dataset))]
dim(timeseries_data)

## [1] 248   2
```

STEP2: Convert to TS

```
ts_data <- ts(timeseries_data$WindSpeed, frequency = 8)
head(ts_data)

## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 8
## [1] 12.928264 13.140015 13.090836 10.509519  7.061161  7.494665
```

STEP3: Plotting the time series data

```
#Plotting the initial time series data
plot(ts_data, main="Initial Time Series Data", xlab = "Time", ylab = "Wind
Speed")
```

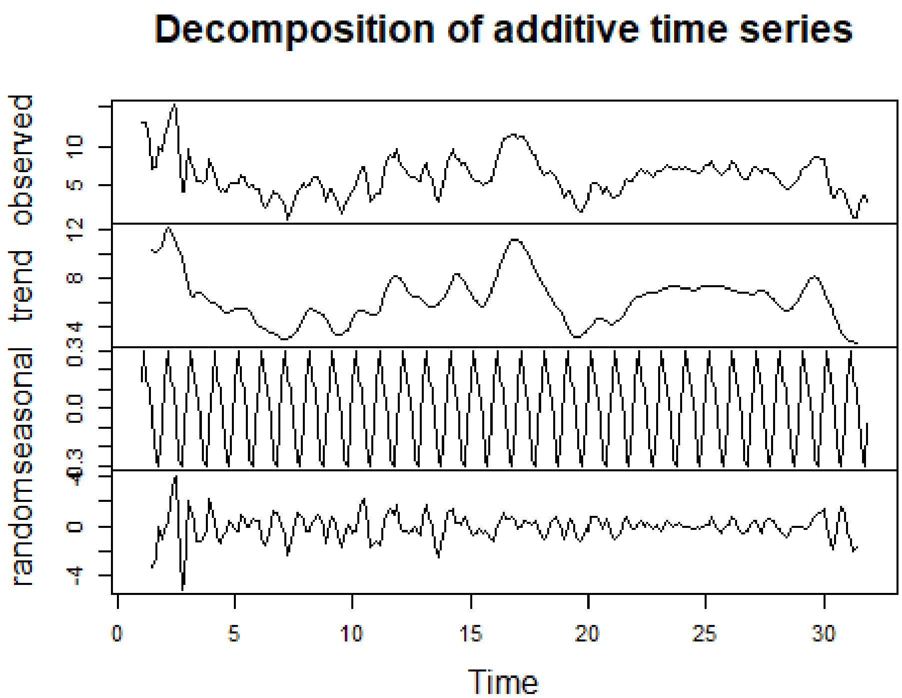


This plot displays the initial time series of wind speed at Dogger Bank, illustrating the variable nature of wind speed over time.

Decompose the time series data

The data is decomposed into observed, seasonality, trend, and randomness to understand its patterns.

```
decomposed_data <- decompose(ts_data)
plot(decomposed_data)
```



The decomposition of the time series clearly shows seasonality, indicated by the regular, continuous pattern in the seasonal component. The trend component reveals gradual changes over time, that it's not completely increasing or decreasing.

STEP4: Handling missing values and outliers

```
#check missing value
missing_values_count <- sum(is.na(ts_data))
cat("Number of missing values:", missing_values_count, "\n")

## Number of missing values: 0

#check outliers
outliers <- tsoutliers(ts_data)
outliers

## $index
## integer(0)
##
## $replacements
## numeric(0)

#ts_data_clean <- tsclean(ts_data)
#plot(ts_data_clean, main = "Cleaned Wind Speed Time Series Data", xlab =
"Time", ylab = "Wind Speed")
```

The results show 0 missing value and outliers, so it doesn't need any tsclean function.

STEP5: Check for stationarity

To ensure reliable forecasting, data should be check for stationary.

```
#Perform the Augmented Dickey-Fuller Test.
adf_test_result <- adf.test(ts_data, alternative="stationary")
print(adf_test_result)

##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data
## Dickey-Fuller = -3.3501, Lag order = 6, p-value = 0.06339
## alternative hypothesis: stationary

#Handle non-stationary, by perform diff function
ts_data_diff <- diff(ts_data)
adf_test_diff <- adf.test(ts_data_diff, alternative="stationary")

## Warning in adf.test(ts_data_diff, alternative = "stationary"): p-value
smaller
## than printed p-value

print(adf_test_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  ts_data_diff
## Dickey-Fuller = -7.0871, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

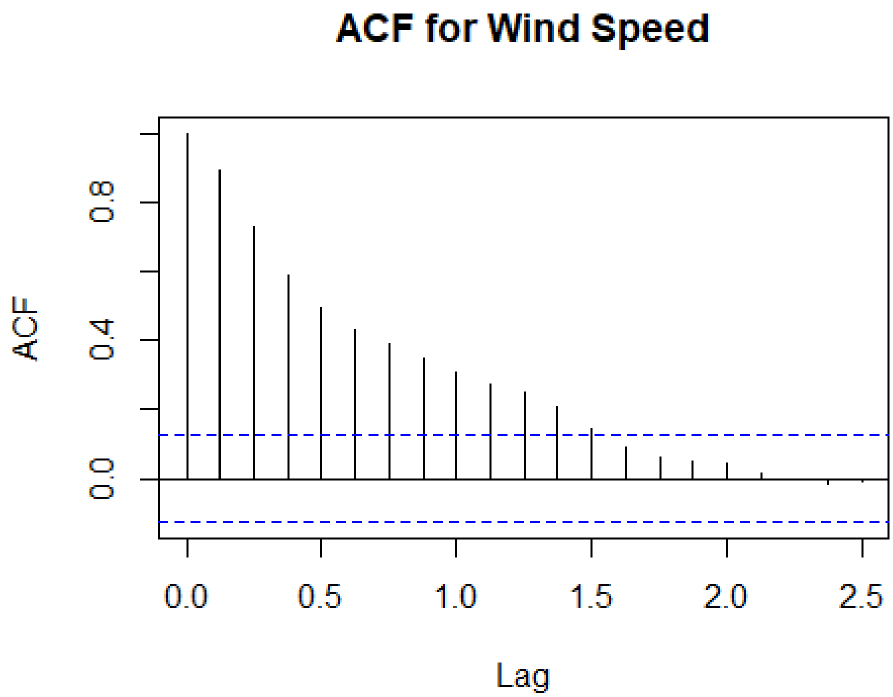
The initial result of the ADF test resulted a p-value of 0.06339, which, being higher than 0.05, hence that the data is not stationary, and the null hypothesis of non-stationarity is accepted. After applying the differencing function and retesting, the p-value dropped to 0.01, confirming that our data has become stationary.

STEP6: Splitting data into training and test sets

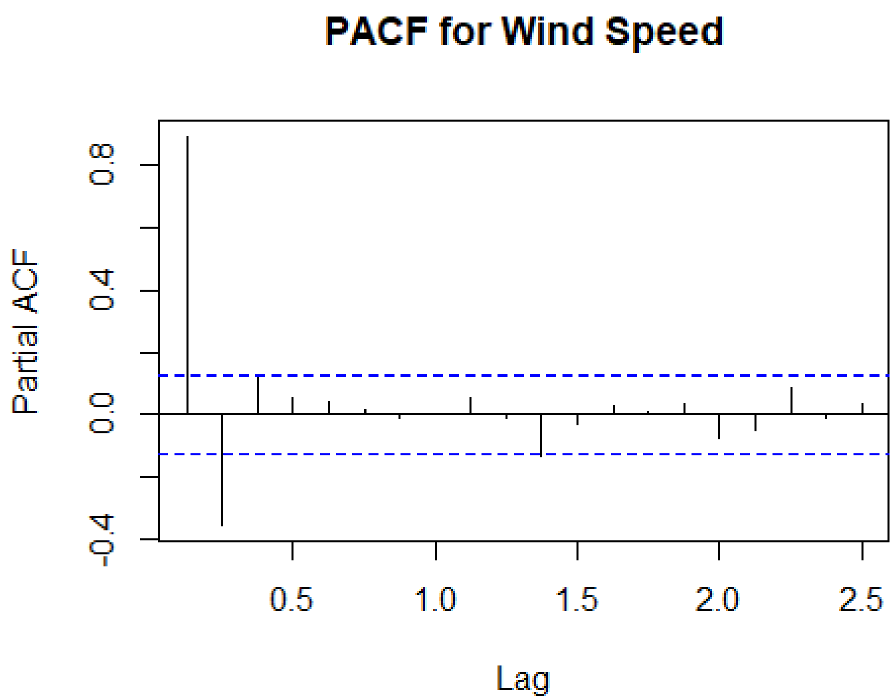
```
# Split into training and testing sets
train_data <- window(ts_data_diff, end=c(21,7))
test_data <- window(ts_data_diff, start=c(21,8))

# Summary of Splits
cat("Training Data Size:", length(train_data), "\n")

## Training Data Size: 166

cat("Testing Data Size:", length(test_data), "\n")

## Testing Data Size: 81
```

STEP7: Model creation with tslm (Time Series Linear Model)

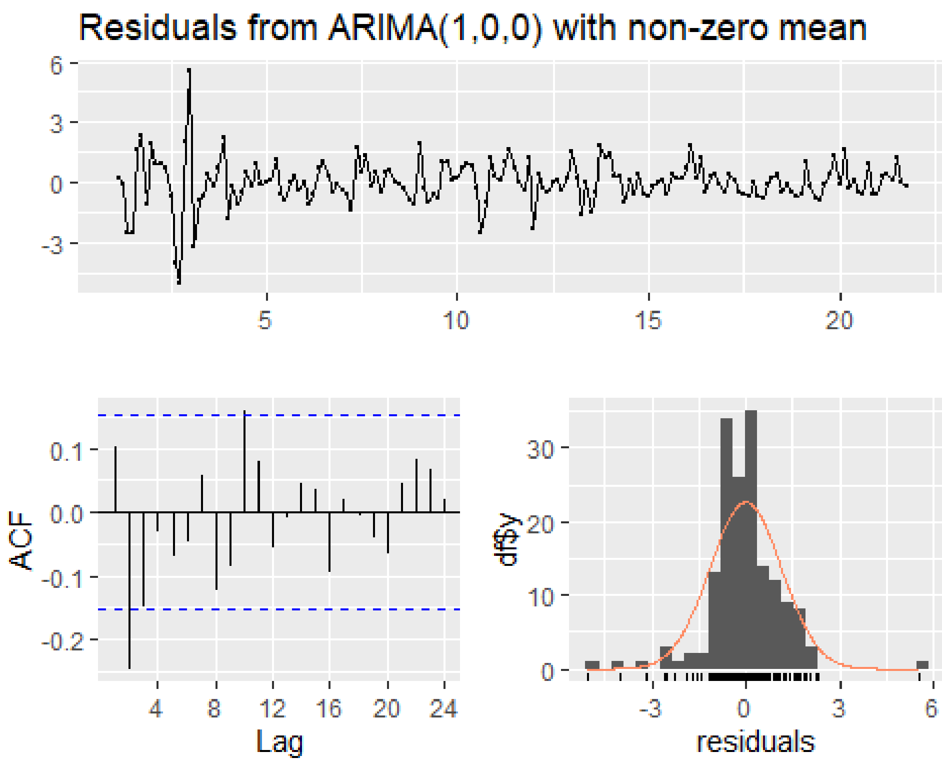
```
#Manual ARIMA
#ACF and PACF plots
#Auto-correlation
acf(ts_data, main="ACF for Wind Speed",lag.max = 20)
```




```
#Partial correlation
pacf(ts_data, main="PACF for Wind Speed", lag.max = 20)
```




```r
#Test AR model
model_arima1<- Arima(train_data, order = c(1,0,0))
model_arima1

## Series: train_data
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1     mean
##       0.3287  -0.0431
## s.e.  0.0730   0.1313
##
## sigma^2 = 1.312:  log likelihood = -257.16
## AIC=520.32   AICc=520.47   BIC=529.65

checkresiduals(model_arima1)
```

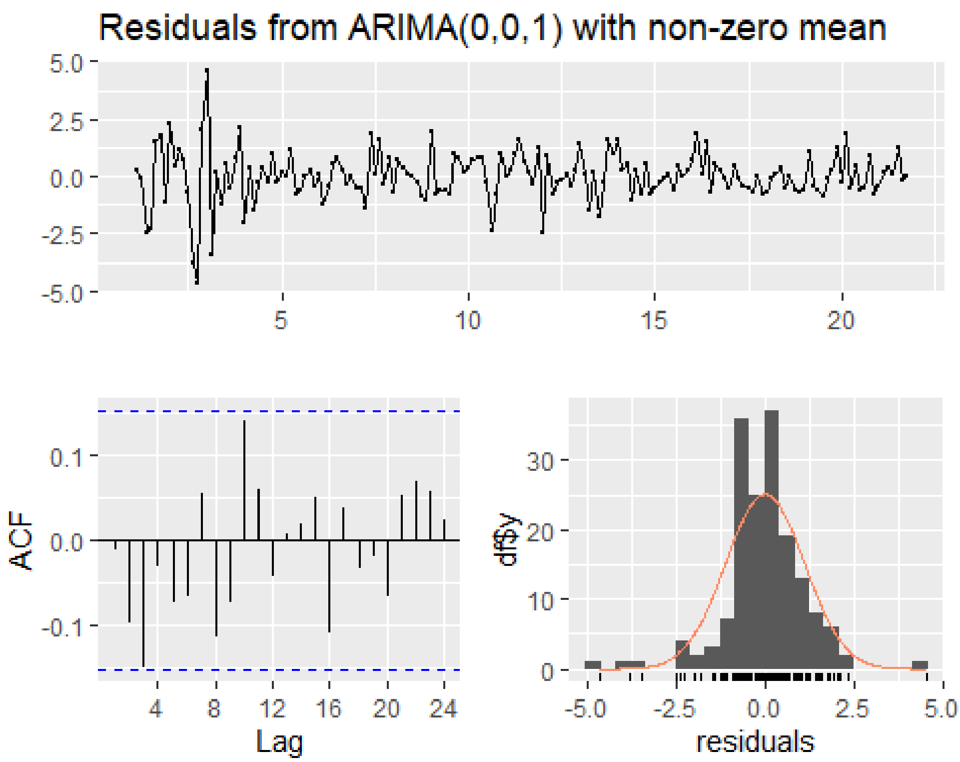



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with non-zero mean
## Q* = 30.202, df = 15, p-value = 0.01121
##
## Model df: 1.   Total lags used: 16
```

```
#Test MA model
model_arima2<- Arima(train_data, order = c(0,0,1))
model_arima2
```

```
## Series: train_data
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##          ma1     mean
##       0.4524  -0.0436
## s.e.  0.0677   0.1242
##
## sigma^2 = 1.234:  log likelihood = -252.07
## AIC=510.15   AICc=510.3   BIC=519.49
```

```
checkresiduals(model_arima2)
```

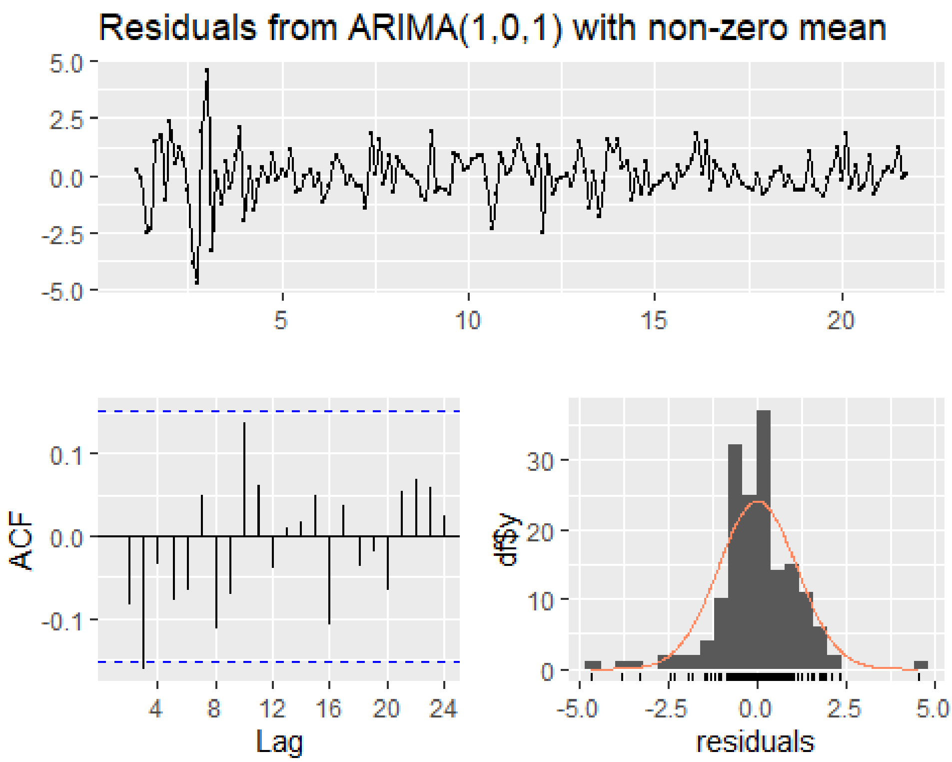



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 18.526, df = 15, p-value = 0.236
##
## Model df: 1.   Total lags used: 16
```

```
#Test combination models of AR and MA.
model_arima3<- Arima(train_data, order = c(1,0,1))
model_arima3
```

```
## Series: train_data
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##           ar1     ma1     mean
##       -0.0508  0.4895  -0.0436
## s.e.   0.1498  0.1257   0.1212
##
## sigma^2 = 1.24:  log likelihood = -252.02
## AIC=512.03   AICc=512.28   BIC=524.48
```

```
checkresiduals(model_arima3)
```

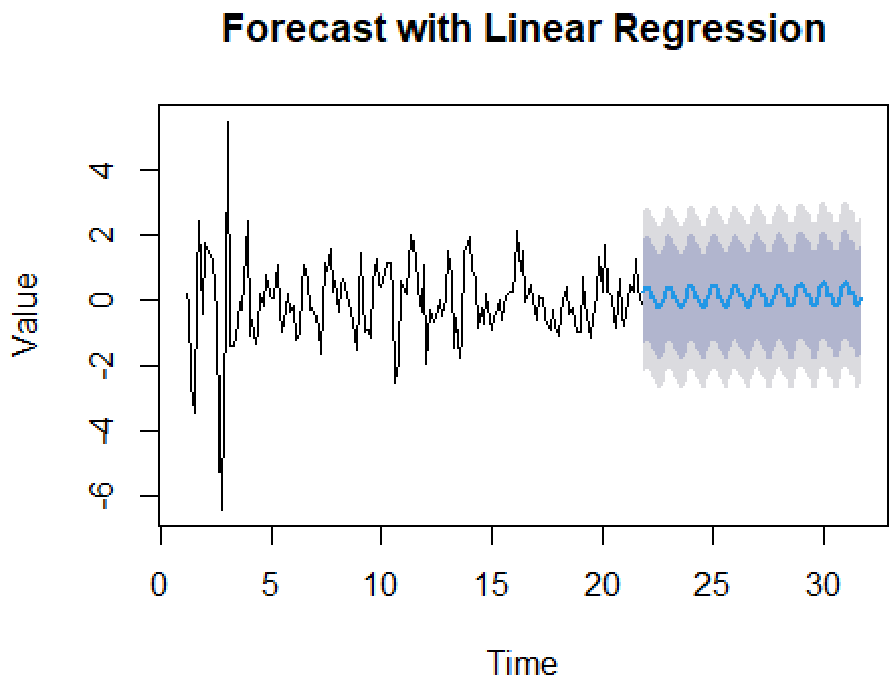



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 18.355, df = 14, p-value = 0.1911
##
## Model df: 2.   Total lags used: 16
```

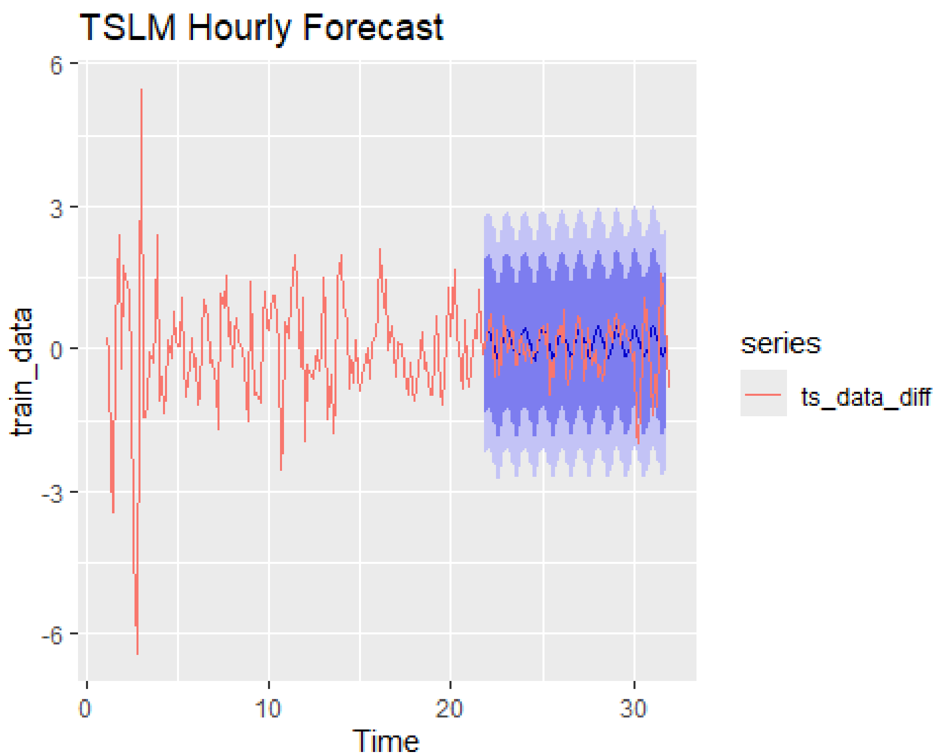
According the autocorrelation results in checkresiduals and AIC value(510.15) the best model of manual ARIMA is c(0,0,1).

```
#check linearity
model<- lm(WindSpeed ~., timeseries_data)
raintest(model)

##
##  Rainbow test
##
## data:  model
## Rain = 1.3382, df1 = 124, df2 = 121, p-value = 0.05434

#Because p value is upper that 0.05 we can use linear model (p-value =
0.05434).

# Build a linear model considering seasonality and trend
model_tslm <- tslm(train_data ~ season + trend)
model_forecast_tslm <- forecast(model_tslm, h = 80)
```

```r
# Plot Forecast
plot(model_forecast_tslm, main="Forecast with Linear Regression",
xlab="Time", ylab="Value")
```




```r
autoplot(model_forecast_tslm) + autolayer(ts_data_diff) + ggtitle("TSLM
Hourly Forecast")
```



SARIMA modeling

```
# Build and SARIMA models
# According decompose seasonal diagram showed that time-series data is
seasonal, we can use sarima model.
#fit_sarima <- auto.arima(train_data, seasonal=TRUE)
fit_sarima <- auto.arima(train_data,
                              stepwise=FALSE,
                              approximation=FALSE,
                              seasonal=TRUE, # This will extent to SARIMA
                              allowdrift=FALSE,
                              # parallel = TRUE,  # speeds up computation,
but tracing not available
                              trace=TRUE)

##
##  ARIMA(0,0,0)           with zero mean     : 535.6582
##  ARIMA(0,0,0)           with non-zero mean : 537.4901
##  ARIMA(0,0,0)(0,0,1)[8] with zero mean     : 534.4668
##  ARIMA(0,0,0)(0,0,1)[8] with non-zero mean : 536.2303
##  ARIMA(0,0,0)(0,0,2)[8] with zero mean     : 535.0591
##  ARIMA(0,0,0)(0,0,2)[8] with non-zero mean : 536.7736
##  ARIMA(0,0,0)(1,0,0)[8] with zero mean     : 535.0317
##  ARIMA(0,0,0)(1,0,0)[8] with non-zero mean : 536.8263
##  ARIMA(0,0,0)(1,0,1)[8] with zero mean     : 535.6098
##  ARIMA(0,0,0)(1,0,1)[8] with non-zero mean : 537.3568
##  ARIMA(0,0,0)(1,0,2)[8] with zero mean     : 537.1424
##  ARIMA(0,0,0)(1,0,2)[8] with non-zero mean : 538.8827
```

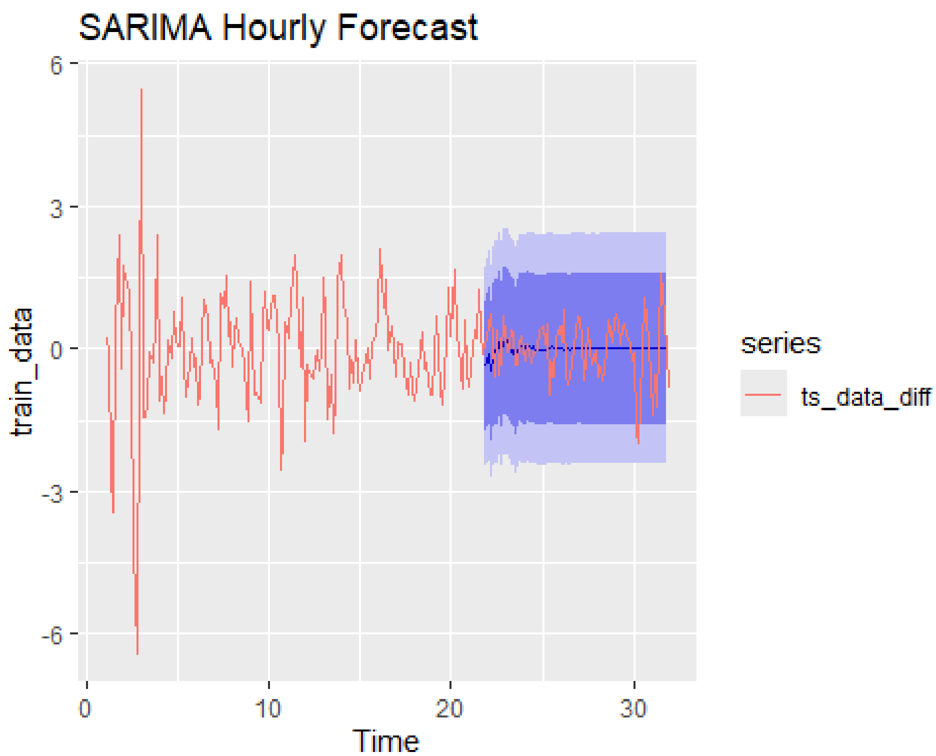
```
##  ARIMA(0,0,0)(2,0,0)[8] with zero mean     : 534.7983
##  ARIMA(0,0,0)(2,0,0)[8] with non-zero mean : 536.5014
##  ARIMA(0,0,0)(2,0,1)[8] with zero mean     : 536.8936
##  ARIMA(0,0,0)(2,0,1)[8] with non-zero mean : 538.6248
##  ARIMA(0,0,0)(2,0,2)[8] with zero mean     : 538.7479
##  ARIMA(0,0,0)(2,0,2)[8] with non-zero mean : 540.5334
##  ARIMA(0,0,1)           with zero mean     : 508.3465
##  ARIMA(0,0,1)           with non-zero mean : 510.2981
##  ARIMA(0,0,1)(0,0,1)[8] with zero mean     : 507.3477
##  ARIMA(0,0,1)(0,0,1)[8] with non-zero mean : 509.276
##  ARIMA(0,0,1)(0,0,2)[8] with zero mean     : 506.8643
##  ARIMA(0,0,1)(0,0,2)[8] with non-zero mean : 508.769
##  ARIMA(0,0,1)(1,0,0)[8] with zero mean     : 508.0947
##  ARIMA(0,0,1)(1,0,0)[8] with non-zero mean : 510.0425
##  ARIMA(0,0,1)(1,0,1)[8] with zero mean     : 508.2445
##  ARIMA(0,0,1)(1,0,1)[8] with non-zero mean : 510.1747
##  ARIMA(0,0,1)(1,0,2)[8] with zero mean     : 508.9362
##  ARIMA(0,0,1)(1,0,2)[8] with non-zero mean : 510.8667
##  ARIMA(0,0,1)(2,0,0)[8] with zero mean     : 505.8352
##  ARIMA(0,0,1)(2,0,0)[8] with non-zero mean : 507.7185
##  ARIMA(0,0,1)(2,0,1)[8] with zero mean     : 507.9611
##  ARIMA(0,0,1)(2,0,1)[8] with non-zero mean : 509.8715
##  ARIMA(0,0,1)(2,0,2)[8] with zero mean     : 508.5413
##  ARIMA(0,0,1)(2,0,2)[8] with non-zero mean : 510.5395
##  ARIMA(0,0,2)           with zero mean     : 510.1956
##  ARIMA(0,0,2)           with non-zero mean : 512.1591
##  ARIMA(0,0,2)(0,0,1)[8] with zero mean     : 509.3112
##  ARIMA(0,0,2)(0,0,1)[8] with non-zero mean : 511.2507
##  ARIMA(0,0,2)(0,0,2)[8] with zero mean     : 508.8674
##  ARIMA(0,0,2)(0,0,2)[8] with non-zero mean : 510.7811
##  ARIMA(0,0,2)(1,0,0)[8] with zero mean     : 510.0349
##  ARIMA(0,0,2)(1,0,0)[8] with non-zero mean : 511.9943
##  ARIMA(0,0,2)(1,0,1)[8] with zero mean     : 510.2359
##  ARIMA(0,0,2)(1,0,1)[8] with non-zero mean : 512.1758
##  ARIMA(0,0,2)(1,0,2)[8] with zero mean     : 510.9665
##  ARIMA(0,0,2)(1,0,2)[8] with non-zero mean : 512.9064
##  ARIMA(0,0,2)(2,0,0)[8] with zero mean     : 507.8772
##  ARIMA(0,0,2)(2,0,0)[8] with non-zero mean : 509.7717
##  ARIMA(0,0,2)(2,0,1)[8] with zero mean     : 510.0291
##  ARIMA(0,0,2)(2,0,1)[8] with non-zero mean : 511.9514
##  ARIMA(0,0,3)           with zero mean     : 505.8131
##  ARIMA(0,0,3)           with non-zero mean : 507.6361
##  ARIMA(0,0,3)(0,0,1)[8] with zero mean     : 505.0066
##  ARIMA(0,0,3)(0,0,1)[8] with non-zero mean : 506.7339
##  ARIMA(0,0,3)(0,0,2)[8] with zero mean     : 504.5941
##  ARIMA(0,0,3)(0,0,2)[8] with non-zero mean : 506.2644
##  ARIMA(0,0,3)(1,0,0)[8] with zero mean     : 505.7355
##  ARIMA(0,0,3)(1,0,0)[8] with non-zero mean : 507.5093
##  ARIMA(0,0,3)(1,0,1)[8] with zero mean     : 505.9346
##  ARIMA(0,0,3)(1,0,1)[8] with non-zero mean : 507.6464
```

```
##  ARIMA(0,0,3)(2,0,0)[8] with zero mean     : 503.7239
##  ARIMA(0,0,3)(2,0,0)[8] with non-zero mean : 505.3547
##  ARIMA(0,0,4)           with zero mean     : 504.7258
##  ARIMA(0,0,4)           with non-zero mean : 506.3746
##  ARIMA(0,0,4)(0,0,1)[8] with zero mean     : 503.8096
##  ARIMA(0,0,4)(0,0,1)[8] with non-zero mean : 505.3423
##  ARIMA(0,0,4)(1,0,0)[8] with zero mean     : 504.5969
##  ARIMA(0,0,4)(1,0,0)[8] with non-zero mean : 506.1883
##  ARIMA(0,0,5)           with zero mean     : 506.4186
##  ARIMA(0,0,5)           with non-zero mean : 508.0339
##  ARIMA(1,0,0)           with zero mean     : 518.4993
##  ARIMA(1,0,0)           with non-zero mean : 520.4664
##  ARIMA(1,0,0)(0,0,1)[8] with zero mean     : 517.1235
##  ARIMA(1,0,0)(0,0,1)[8] with non-zero mean : 519.0733
##  ARIMA(1,0,0)(0,0,2)[8] with zero mean     : 517.2362
##  ARIMA(1,0,0)(0,0,2)[8] with non-zero mean : 519.1707
##  ARIMA(1,0,0)(1,0,0)[8] with zero mean     : 517.8455
##  ARIMA(1,0,0)(1,0,0)[8] with non-zero mean : 519.8117
##  ARIMA(1,0,0)(1,0,1)[8] with zero mean     : 518.2048
##  ARIMA(1,0,0)(1,0,1)[8] with non-zero mean : 520.1593
##  ARIMA(1,0,0)(1,0,2)[8] with zero mean     : 519.3278
##  ARIMA(1,0,0)(1,0,2)[8] with non-zero mean : 521.2887
##  ARIMA(1,0,0)(2,0,0)[8] with zero mean     : 516.5024
##  ARIMA(1,0,0)(2,0,0)[8] with non-zero mean : 518.4239
##  ARIMA(1,0,0)(2,0,1)[8] with zero mean     : 518.6288
##  ARIMA(1,0,0)(2,0,1)[8] with non-zero mean : 520.5767
##  ARIMA(1,0,0)(2,0,2)[8] with zero mean     : 519.5971
##  ARIMA(1,0,0)(2,0,2)[8] with non-zero mean : 521.6101
##  ARIMA(1,0,1)           with zero mean     : 510.3098
##  ARIMA(1,0,1)           with non-zero mean : 512.281
##  ARIMA(1,0,1)(0,0,1)[8] with zero mean     : 509.3809
##  ARIMA(1,0,1)(0,0,1)[8] with non-zero mean : 511.3287
##  ARIMA(1,0,1)(0,0,2)[8] with zero mean     : 508.9301
##  ARIMA(1,0,1)(0,0,2)[8] with non-zero mean : 510.8535
##  ARIMA(1,0,1)(1,0,0)[8] with zero mean     : 510.1163
##  ARIMA(1,0,1)(1,0,0)[8] with non-zero mean : 512.0838
##  ARIMA(1,0,1)(1,0,1)[8] with zero mean     : 510.3048
##  ARIMA(1,0,1)(1,0,1)[8] with non-zero mean : 512.2541
##  ARIMA(1,0,1)(1,0,2)[8] with zero mean     : 511.029
##  ARIMA(1,0,1)(1,0,2)[8] with non-zero mean : 512.9786
##  ARIMA(1,0,1)(2,0,0)[8] with zero mean     : 507.9198
##  ARIMA(1,0,1)(2,0,0)[8] with non-zero mean : 509.8228
##  ARIMA(1,0,1)(2,0,1)[8] with zero mean     : Inf
##  ARIMA(1,0,1)(2,0,1)[8] with non-zero mean : Inf
##  ARIMA(1,0,2)           with zero mean     : 503.3726
##  ARIMA(1,0,2)           with non-zero mean : 504.7853
##  ARIMA(1,0,2)(0,0,1)[8] with zero mean     : 503.7593
##  ARIMA(1,0,2)(0,0,1)[8] with non-zero mean : 505.1734
##  ARIMA(1,0,2)(0,0,2)[8] with zero mean     : 503.55
##  ARIMA(1,0,2)(0,0,2)[8] with non-zero mean : 504.9295
```

```
##  ARIMA(1,0,2)(1,0,0)[8] with zero mean     : 504.1883
##  ARIMA(1,0,2)(1,0,0)[8] with non-zero mean : 505.6156
##  ARIMA(1,0,2)(1,0,1)[8] with zero mean     : 504.8755
##  ARIMA(1,0,2)(1,0,1)[8] with non-zero mean : 506.3005
##  ARIMA(1,0,2)(2,0,0)[8] with zero mean     : 502.7144
##  ARIMA(1,0,2)(2,0,0)[8] with non-zero mean : 504.0416
##  ARIMA(1,0,3)           with zero mean     : 503.0178
##  ARIMA(1,0,3)           with non-zero mean : 504.4644
##  ARIMA(1,0,3)(0,0,1)[8] with zero mean     : 503.003
##  ARIMA(1,0,3)(0,0,1)[8] with non-zero mean : 504.4439
##  ARIMA(1,0,3)(1,0,0)[8] with zero mean     : 503.5729
##  ARIMA(1,0,3)(1,0,0)[8] with non-zero mean : 505.0371
##  ARIMA(1,0,4)           with zero mean     : 505.1578
##  ARIMA(1,0,4)           with non-zero mean : 506.6273
##  ARIMA(2,0,0)           with zero mean     : 504.957
##  ARIMA(2,0,0)           with non-zero mean : 506.8485
##  ARIMA(2,0,0)(0,0,1)[8] with zero mean     : 503.9391
##  ARIMA(2,0,0)(0,0,1)[8] with non-zero mean : 505.7699
##  ARIMA(2,0,0)(0,0,2)[8] with zero mean     : 503.377
##  ARIMA(2,0,0)(0,0,2)[8] with non-zero mean : 505.1609
##  ARIMA(2,0,0)(1,0,0)[8] with zero mean     : 504.7366
##  ARIMA(2,0,0)(1,0,0)[8] with non-zero mean : 506.6021
##  ARIMA(2,0,0)(1,0,1)[8] with zero mean     : 504.8295
##  ARIMA(2,0,0)(1,0,1)[8] with non-zero mean : 506.6512
##  ARIMA(2,0,0)(1,0,2)[8] with zero mean     : 505.4574
##  ARIMA(2,0,0)(1,0,2)[8] with non-zero mean : 507.265
##  ARIMA(2,0,0)(2,0,0)[8] with zero mean     : 502.2631
##  ARIMA(2,0,0)(2,0,0)[8] with non-zero mean : 504.0096
##  ARIMA(2,0,0)(2,0,1)[8] with zero mean     : 504.415
##  ARIMA(2,0,0)(2,0,1)[8] with non-zero mean : 506.1895
##  ARIMA(2,0,1)           with zero mean     : 504.2443
##  ARIMA(2,0,1)           with non-zero mean : 505.828
##  ARIMA(2,0,1)(0,0,1)[8] with zero mean     : 502.4688
##  ARIMA(2,0,1)(0,0,1)[8] with non-zero mean : 503.9675
##  ARIMA(2,0,1)(0,0,2)[8] with zero mean     : 502.5318
##  ARIMA(2,0,1)(0,0,2)[8] with non-zero mean : 503.9736
##  ARIMA(2,0,1)(1,0,0)[8] with zero mean     : 503.4324
##  ARIMA(2,0,1)(1,0,0)[8] with non-zero mean : 504.9949
##  ARIMA(2,0,1)(1,0,1)[8] with zero mean     : 503.6443
##  ARIMA(2,0,1)(1,0,1)[8] with non-zero mean : 505.1328
##  ARIMA(2,0,1)(2,0,0)[8] with zero mean     : 501.3403
##  ARIMA(2,0,1)(2,0,0)[8] with non-zero mean : 502.706
##  ARIMA(2,0,2)           with zero mean     : 503.2584
##  ARIMA(2,0,2)           with non-zero mean : 504.7243
##  ARIMA(2,0,2)(0,0,1)[8] with zero mean     : 502.941
##  ARIMA(2,0,2)(0,0,1)[8] with non-zero mean : 504.4074
##  ARIMA(2,0,2)(1,0,0)[8] with zero mean     : 503.6196
##  ARIMA(2,0,2)(1,0,0)[8] with non-zero mean : 505.1148
##  ARIMA(2,0,3)           with zero mean     : 505.1578
##  ARIMA(2,0,3)           with non-zero mean : 506.6278
```

```
## ARIMA(3,0,0)           with zero mean     : 506.7519
## ARIMA(3,0,0)           with non-zero mean : 508.6535
## ARIMA(3,0,0)(0,0,1)[8] with zero mean     : 505.5488
## ARIMA(3,0,0)(0,0,1)[8] with non-zero mean : 507.369
## ARIMA(3,0,0)(0,0,2)[8] with zero mean     : 505.1449
## ARIMA(3,0,0)(0,0,2)[8] with non-zero mean : 506.9233
## ARIMA(3,0,0)(1,0,0)[8] with zero mean     : 506.3979
## ARIMA(3,0,0)(1,0,0)[8] with non-zero mean : 508.2606
## ARIMA(3,0,0)(1,0,1)[8] with zero mean     : 506.5295
## ARIMA(3,0,0)(1,0,1)[8] with non-zero mean : 508.3434
## ARIMA(3,0,0)(2,0,0)[8] with zero mean     : 503.985
## ARIMA(3,0,0)(2,0,0)[8] with non-zero mean : 505.7184
## ARIMA(3,0,1)           with zero mean     : 503.6641
## ARIMA(3,0,1)           with non-zero mean : 505.1105
## ARIMA(3,0,1)(0,0,1)[8] with zero mean     : 503.3959
## ARIMA(3,0,1)(0,0,1)[8] with non-zero mean : 504.8575
## ARIMA(3,0,1)(1,0,0)[8] with zero mean     : 504.0654
## ARIMA(3,0,1)(1,0,0)[8] with non-zero mean : 505.5477
## ARIMA(3,0,2)           with zero mean     : 505.3813
## ARIMA(3,0,2)           with non-zero mean : 506.8646
## ARIMA(4,0,0)           with zero mean     : 507.7153
## ARIMA(4,0,0)           with non-zero mean : 509.6224
## ARIMA(4,0,0)(0,0,1)[8] with zero mean     : 505.9379
## ARIMA(4,0,0)(0,0,1)[8] with non-zero mean : 507.7212
## ARIMA(4,0,0)(1,0,0)[8] with zero mean     : 506.9498
## ARIMA(4,0,0)(1,0,0)[8] with non-zero mean : 508.7929
## ARIMA(4,0,1)           with zero mean     : 505.0661
## ARIMA(4,0,1)           with non-zero mean : 506.5755
## ARIMA(5,0,0)           with zero mean     : 507.2182
## ARIMA(5,0,0)           with non-zero mean : 509.0943
##
##
##
## Best model: ARIMA(2,0,1)(2,0,0)[8] with zero mean
```

```r
# Forecast future points
forecast_sarima <- forecast(fit_sarima, h = 80)

# Plot Forecasts
autoplot(forecast_sarima) + autolayer(ts_data_diff) + ggtitle("SARIMA Hourly
Forecast")
```



The optimal SARIMA model, ARIMA(2,0,1)(2,0,0)[8], with zero mean, has the lowest AIC value of 503.985.

STEP8: Check accuracy

```
accuracy_sarima <- accuracy(forecast_sarima, test_data)
accuracy_tslm <-accuracy(model_forecast_tslm,test_data)

cat("SARIMA Accuracy:\n")

## SARIMA Accuracy:

print(accuracy_sarima)

##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.08216456 1.0507289 0.7705009     -Inf      Inf 0.6023207
## Test set     -0.01027584 0.6255506 0.4698686 112.2264 112.4531 0.3673085
##                    ACF1 Theil's U
## Training set 0.02739543        NA
## Test set     0.46850129  1.041009

cat("TSLM Accuracy:\n")

## TSLM Accuracy:

print(accuracy_tslm)

##                         ME      RMSE       MAE      MPE    MAPE      MASE
## Training set -2.538731e-17 1.1832153 0.8342955      Inf     Inf 0.6521905
```

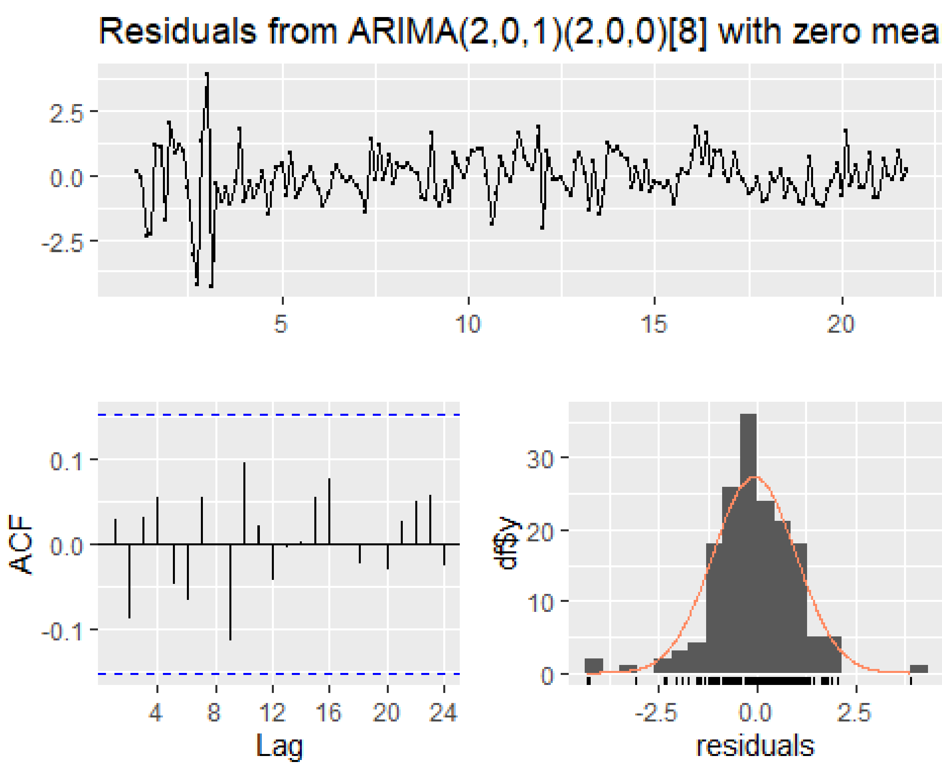
```
## Test set     -1.554078e-01 0.6817632 0.4810236 81.19361 163.87 0.3760287
##                   ACF1 Theil's U
## Training set 0.3179268        NA
## Test set     0.4886656  1.054275
```

Evaluation of Accuracy Test for SARIMA and TSLM Models

Both models perform well in managing new data, as indicated by lower RMSE and MAE on test sets. However, both models are affected by percentage errors, which could be due to anomalies or specific characteristics of the wind speed data. The ACF1 values for both models show small residual autocorrelations, which might be addressed with model adjustments. The SARIMA model seems to handle the data slightly better, given the context of lower autocorrelation and slightly better percentage errors on the test data.

STEP9: Check residuals

```
checkresiduals(fit_sarima)
```



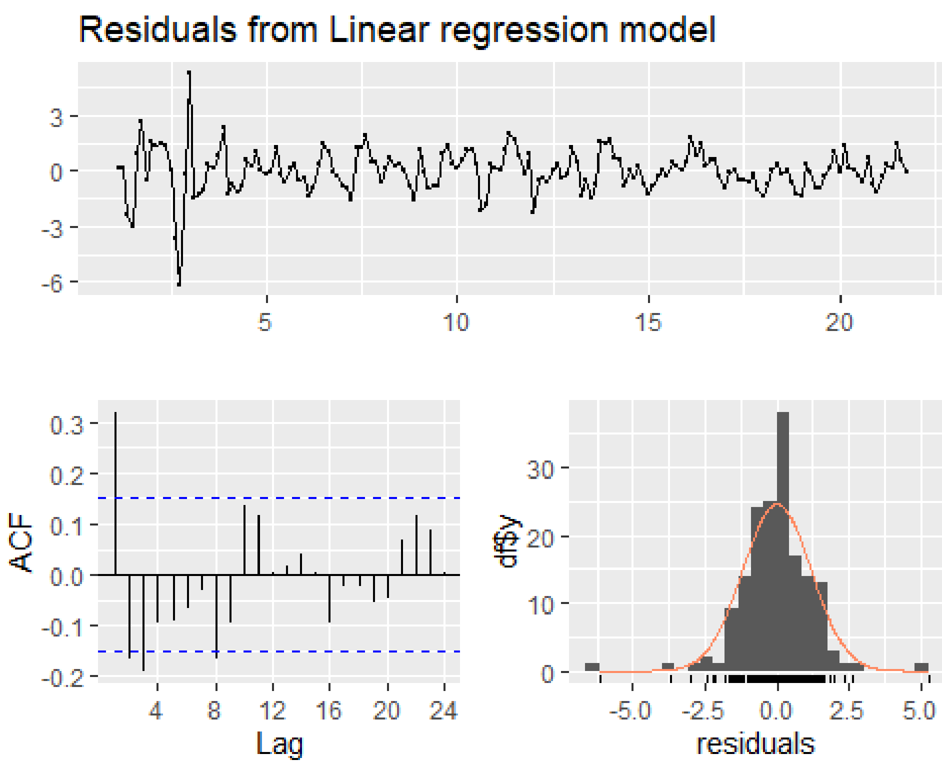

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1)(2,0,0)[8] with zero mean
## Q* = 9.8178, df = 11, p-value = 0.5469
##
## Model df: 5.   Total lags used: 16

shapiro.test(fit_sarima$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit_sarima$residuals
## W = 0.95167, p-value = 1.778e-05
```

Evaluation of Checking Residuals for SARIMA Model:

The ARIMA(2,0,1)(2,0,0)[8] model used in this study indicates that the residuals fluctuate randomly about zero with no trends and closely resemble a normal distribution but the Shapiro test shows not normally distribution for residual because the p value is less than 0.05. The Ljung-Box test, with a p-value of 0.5469, indicates that these residuals do not show autocorrelation.Finally, the result shows that this model is well-suited for the dataset.

```r
checkresiduals(model_tslm)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 16
##
## data:  Residuals from Linear regression model
## LM test = 44.885, df = 16, p-value = 0.0001445
```

```r
shapiro.test(model_tslm$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model_tslm$residuals
## W = 0.92862, p-value = 2.544e-07
```

Evaluation of Checking Residuals for TSLM Model:

Analysing the linear regression model designed to anticipate wind speed, the presence of significant spikes in the residual plot shows outliers, which are most likely caused by storm occurrences, and the Shapiro test shows not normally distribution for residual because the p value is less than 0.05. Furthermore, the ACF plot demonstrates autocorrelation, which is confirmed by the Breusch-Godfrey test, which yields a p-value of 0.0001445, demonstrating significant serial correlation in residuals.This shows that the model is unsuitable for our dataset.However, the best result for this dataset was found using the last model, SARIMA.

Machine Learning Prediction Algorithms

In our investigation into optimising wind speed predictions at Dogger Bank, we performed a comparative analysis of various machine learning regression techniques, including Decision Trees, Support Vector Regression (SVR), and multiple configurations of Random Forests, specifically models with 100, 150, and 200 trees. We evaluated each model's performance using MAE and RMSE metrics.

```
# Load and prepare the data
head(data_interpolated)

##          date_time   TSK   PSFC      Q2 RAINNC WindSpeed
## 1 01-05-2018-00-00 280.5 100053 0.00539    0.9 12.928264
## 2 01-05-2018-03-00 280.5 100124 0.00526    1.0 13.140015
## 3 01-05-2018-06-00 280.5 100245 0.00513    1.0 13.090836
## 4 01-05-2018-09-00 280.5 100453 0.00493    1.0 10.509519
## 5 01-05-2018-12-00 280.5 100613 0.00525    1.0  7.061161
## 6 01-05-2018-15-00 280.5 100621 0.00535    1.0  7.494665

dataset <- data_interpolated[2:6]

#Split data into test and train
set.seed(123)
split=sample.split(dataset$WindSpeed, SplitRatio = 0.8)
training_set <- subset(dataset, split == TRUE)
testing_set <- subset(dataset, split== FALSE)
glimpse(training_set)

## Rows: 198
## Columns: 5
## $ TSK       <dbl> 280.5, 280.5, 280.5, 280.5, 280.5, 280.5, 280.5, 280.5,
280.…
## $ PSFC      <dbl> 100053.0, 100245.0, 100621.0, 100587.0, 100516.0,
100447.0, …
## $ Q2        <dbl> 0.00539, 0.00513, 0.00535, 0.00521, 0.00578, 0.00557,
```

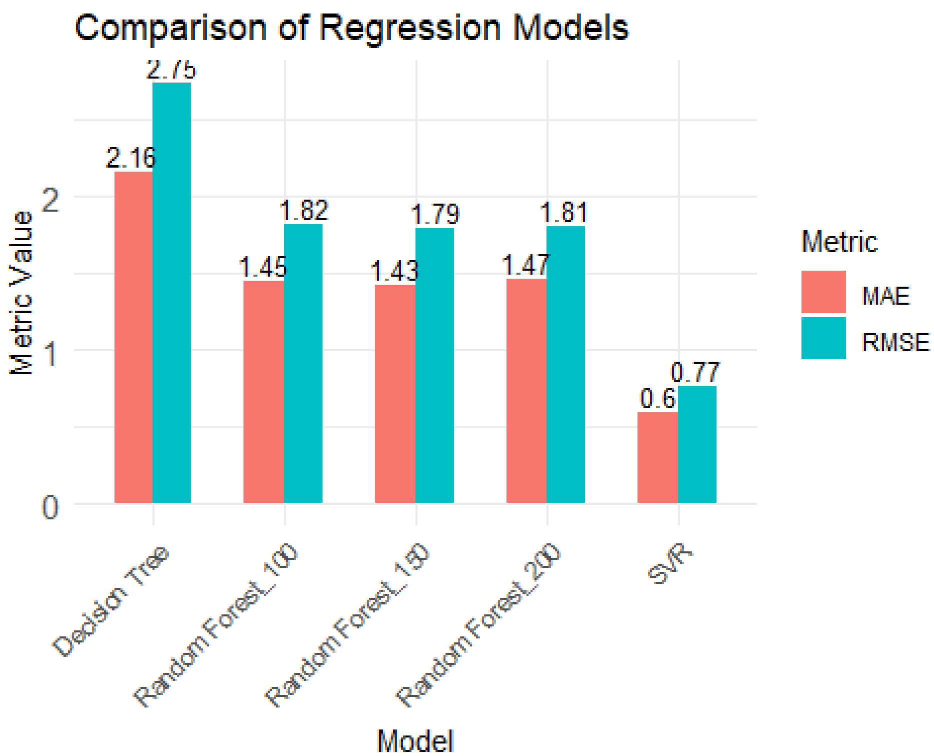
```
0.0060…
## $ RAINNC    <dbl> 0.9, 1.0, 1.0, 1.0, 0.0, 0.0, 0.2, 2.0, 4.1, 4.7, 0.0,
0.0, …
## $ WindSpeed <dbl> 12.928264, 13.090836, 7.494665, 9.900505, 11.264102,
12.7565…
```

```
train_set_scaled <- as.data.frame(scale(training_set)) #return as dataframe
colnames(train_set_scaled) <- colnames(training_set)   #used the previous
name of columns to new dataframe
test_set_scaled <- as.data.frame(scale(testing_set))
colnames(test_set_scaled) <- colnames(testing_set)

#Random Forest with varying number of trees
model_rf100 <- randomForest(WindSpeed ~ ., data = training_set, ntree = 100)
predictions_rf100 <- predict(model_rf100, testing_set)
mae_rf100 <- MAE(predictions_rf100, testing_set$WindSpeed)
rmse_rf100 <- RMSE(predictions_rf100, testing_set$WindSpeed)

model_rf150 <- randomForest(WindSpeed ~ ., data = training_set, ntree = 150)
predictions_rf150 <- predict(model_rf150, testing_set)
mae_rf150 <- MAE(predictions_rf150, testing_set$WindSpeed)
rmse_rf150 <- RMSE(predictions_rf150, testing_set$WindSpeed)

model_rf200 <- randomForest(WindSpeed ~ ., data = training_set, ntree = 200)
predictions_rf200 <- predict(model_rf200, testing_set)
mae_rf200 <- MAE(predictions_rf200, testing_set$WindSpeed)
rmse_rf200 <- RMSE(predictions_rf200, testing_set$WindSpeed)

# Fit a Decision Tree model
model_dt <- rpart(WindSpeed ~ ., data = training_set, control =
rpart.control(minsplit = 1))
predictions_dt <- predict(model_dt, testing_set)
mae_dt <- MAE(predictions_dt, testing_set$WindSpeed)
rmse_dt <- RMSE(predictions_dt, testing_set$WindSpeed)

# Fit a Support Vector Regression model
model_svr <- svm(WindSpeed ~ ., data = train_set_scaled, type <- 'eps-
regression', kernel <- 'radial') #non-linear regression
predictions_svr <- predict(model_svr, test_set_scaled)
mae_svr <- MAE(predictions_svr, test_set_scaled$WindSpeed)
rmse_svr <- RMSE(predictions_svr, test_set_scaled$WindSpeed)

predict(model_rf100, data.frame(TSK=280, PSFC=100053 ,Q2=0.005, RAINNC=0.1))
```

```
##        1
## 11.64957
```

```
predict(model_rf150, data.frame(TSK=280, PSFC=100053 ,Q2=0.005, RAINNC=0.1))
```

```
##         1
## 10.91874

predict(model_rf200, data.frame(TSK=280, PSFC=100053 ,Q2=0.005, RAINNC=0.1))

##         1
## 10.99207

predict(model_dt, data.frame(TSK=280, PSFC=100053 ,Q2=0.005, RAINNC=0.1))

## [1] 6.192057

predict(model_svr, data.frame(TSK=280, PSFC=100053 ,Q2=0.005, RAINNC=0.1))

##         1
## 0.8944033

# Store results in a data frame
results <- data.frame(Model = c("Random Forest_100","Random
Forest_150","Random Forest_200","SVR", "Decision Tree"),
                      MAE = c(mae_rf100,mae_rf150,mae_rf200,mae_svr, mae_dt),
                      RMSE = c(rmse_rf100,rmse_rf150,rmse_rf200,rmse_svr,
rmse_dt))

# Melt data for visualization
results_long <- melt(results, id.vars = "Model", variable.name = "Metric",
value.name = "Value")

# Plot the results
ggplot(results_long, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(), width = 0.6) +
  geom_text(aes(label = round(Value, 2)), position = position_dodge(width =
0.6), vjust = -0.25, size = 3.5) +
  scale_y_continuous(labels = scales::comma) +  # Formats the y-axis values
to include commas for thousands
  labs(title = "Comparison of Regression Models", x = "Model", y = "Metric
Value") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Adjusts the
angle and horizontal adjustment of x-axis labels
        axis.text.y = element_text(size = 12))  # Increases the size of y-
axis text for better readability
```



This bar chart compares the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) of various regression models. Notably, the Decision Tree model has the greatest error rates, while the Support Vector Regression (SVR) has the lowest, showing that it is the most accurate at forecasting wind speed. For Random Forest models, the error decreases as the number of trees increases up to 150; however, the model with 200 trees does not follow this trend and shows a slight increase in error compared to the 150-tree model, meaning that adding more trees does not consistently reduce prediction error and that an optimal number of trees may exist.