**STAT8040-23S-Sec1 - Statistical Forecasting**

**Project 1: Individual Project**

**Divanshu Singh**

**Student Number: 8804523**

**Table of content**

1. **Data**

* **Data Set Information:**
* **Attribute Information:**
* **Data exploration**

1. **Visualization**

* **Summary statistics**
* **Descriptive analysis**
* **Basic visualizations**
* **Time plot**
* **ACF plot**
* **Seasonal plot**
* **Heat map**

1. **Transformation**

* **Perform the decomposition.**
* **Extract the components.**
* **Plot the components.**

1. **Step 4 - Forecasting and Analysis**

* **Naïve mean model**
* **Exponential smooth modelling**
* **Comparing the accuracy of the models using matrices**
* **Residual Analysis of the models**
* **Checking normality of residuals using Q-Q plot**
* **Checking for heteroscedasticity using a plot of residuals vs. predicted values.**

1. **Conclusion**

# Step 1 - Data

**Data Set Information:**

This archive contains 2075259 measurements gathered between December 2006 and November 2010 (47 months).  
Notes:  
1. (global\_active\_power\*1000/60 - sub\_metering\_1 - sub\_metering\_2 - sub\_metering\_3) represents the active energy consumed every minute (in watt hour) in the household by electrical equipment not measured in sub-meterings 1, 2 and 3.

2.The dataset contains some missing values in the measurements (nearly 1,25% of the rows). All calendar timestamps are present in the dataset but for some timestamps, the measurement values are missing: a missing value is represented by the absence of value between two consecutive semi-colon attribute separators. For instance, the dataset shows missing values on April 28, 2007.

3. We studied the Household Electric Power Consumption dataset for the time series analysis. The focus of this study was the sub\_metering matters. The sub\_metering category is as follows; Sub\_metering\_1 (corresponds to the kitchen, containing mainly a dishwasher, an oven, and a microwave) Sub\_metering\_2 (mainly washing-machine, a tumble-drier) Sub\_metering\_3 ( water-heater and an air-conditioner) As it shows, we can check the power consumption of different areas of the houses precisely.

**Attribute Information:**  
1.date: Date in format dd/mm/yyyy

2.time: time in format hh:mm:ss

3.global\_active\_power: household global minute-averaged active power (in kilowatt)

4.global\_reactive\_power: household global minute-averaged reactive power (in kilowatt)

5.voltage: minute-averaged voltage (in volt)

6.global\_intensity: household global minute-averaged current intensity (in ampere)

7.sub\_metering\_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).

8.sub\_metering\_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.

9.sub\_metering\_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

# Importing Libraries

library(tidyverse)

library(ISLR2)

library(readr)

library(readxl)

library(ggplot2)

library(dplyr)

library(tidyr)

library(forecast)

library(reshape2)

library(Metrics)

# Importing Household power consumption dataset.

power\_data <- read\_delim(“D:/Predictive Analytics/Semester 2/Forcasting/Individual Project/household\_power\_consumption.txt”, delim = “;”, escape\_double = FALSE, trim\_ws = TRUE)

head(power\_data, 5)

# Shape of the dataset

dim(power\_data)

[1] 2075259 9

Here the shape of the dataset is too much to perform the analysis smoothly. So, I will only take a small sample from this dataset.

sampled\_power\_data <- head(power\_data, 10000)

view(sampled\_power\_data)

I found this dataset on Kaggle (<https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set>). The dataset was too large for a smooth and convienent analysis that why i have only taken 10000 entries.

## Further, I will check for missing values and irregularities in the dataset.

missing\_values <- colSums(is.na(sampled\_power\_data))

missing\_values

Date Time Global\_active\_power Global\_reactive\_power

0 0 0 0

Voltage Global\_intensity Sub\_metering\_1 Sub\_metering\_2

0 0 0 0

Sub\_metering\_3

2

Here we can see that only column Sub\_metering\_3 have 2 missing values

# Removing missing values from the dataset.

dataset\_cleaned <- na.omit(sampled\_power\_data) head(dataset\_cleaned, 5)

colSums(is.na(dataset\_cleaned))

# A tibble: 5 × 9

Date Time Global\_active\_power Global\_reactive\_power Voltage Global…¹ Sub\_m…² Sub\_m…³ Sub\_m…⁴

*<chr>* *<time>* *<chr>* *<chr>* *<chr>* *<chr>* *<chr>* *<chr>* *<dbl>*

1 16/12/2006 17:24 4.216 0.418 234.840 18.400 0.000 1.000 17

2 16/12/2006 17:25 5.360 0.436 233.630 23.000 0.000 1.000 16

3 16/12/2006 17:26 5.374 0.498 233.290 23.000 0.000 2.000 17

4 16/12/2006 17:27 5.388 0.502 233.740 23.000 0.000 1.000 17

5 16/12/2006 17:28 3.666 0.528 235.680 15.800 0.000 1.000 17

# … with abbreviated variable names ¹​Global\_intensity, ²​Sub\_metering\_1, ³​Sub\_metering\_2,

# ⁴​Sub\_metering\_3

# Problem Description

The goal of this project is to develop a forecasting model that accurately predicts future electricity consumption. The dataset includes variables such as global active power, global reactive power, voltage, and sub-metering measurements!

Accurate electricity consumption forecasting is important for a number of reasons, including:

1. Plan and operate power systems to avoid blackouts or brownouts.

2. Invest in new power plants to meet demand.

3. Design and target energy efficiency programs to reduce electricity consumption.

4. Set the right price for electricity to ensure a fair market.

# Step 2 - Visualization

par(mar = c(5, 4, 4, 2) + 0.1) # Convert Date and Time columns to appropriate data types dataset\_cleanedDate, format = “%d/%m/%Y”) dataset\_cleanedTime, format = “%H:%M”) dataset\_cleanedGlobal\_active\_power) dataset\_cleanedGlobal\_reactive\_power) dataset\_cleanedVoltage) dataset\_cleanedGlobal\_intensity) dataset\_cleanedSub\_metering\_1) dataset\_cleanedSub\_metering\_2)

str(dataset\_cleaned)

tibble [9,998 × 9] (S3: tbl\_df/tbl/data.frame)

$ Date : Date[1:9998], format: "2006-12-16" "2006-12-16" "2006-12-16" ...

$ Time : POSIXct[1:9998], format: "1970-01-01 17:24:00" "1970-01-01 17:25:00" "1970-01-01 17:26:00" ...

$ Global\_active\_power : num [1:9998] 4.22 5.36 5.37 5.39 3.67 ...

$ Global\_reactive\_power: num [1:9998] 0.418 0.436 0.498 0.502 0.528 0.522 0.52 0.52 0.51 0.51 ...

$ Voltage : num [1:9998] 235 234 233 234 236 ...

$ Global\_intensity : num [1:9998] 18.4 23 23 23 15.8 15 15.8 15.8 15.8 15.8 ...

$ Sub\_metering\_1 : num [1:9998] 0 0 0 0 0 0 0 0 0 0 ...

$ Sub\_metering\_2 : num [1:9998] 1 1 2 1 1 2 1 1 1 2 ...

$ Sub\_metering\_3 : num [1:9998] 17 16 17 17 17 17 17 17 17 16 ...

- attr(\*, "na.action")= 'omit' Named int [1:2] 6840 6841

..- attr(\*, "names")= chr [1:2] "6840" "6841"

# Summary statistics

summary(dataset\_cleaned)

Date Time Global\_active\_power Global\_reactive\_power Voltage Global\_intensity Sub\_metering\_1

Min. :2006-12-16 Min. :1970-01-01 00:00:00.0000 Min. :0.194 Min. :0.0000 Min. :228.9 Min. : 0.800 Min. : 0.0000

1st Qu.:2006-12-18 1st Qu.:1970-01-01 05:57:00.0000 1st Qu.:0.388 1st Qu.:0.0000 1st Qu.:238.5 1st Qu.: 1.800 1st Qu.: 0.0000

Median :2006-12-20 Median :1970-01-01 11:54:00.0000 Median :1.478 Median :0.1000 Median :241.6 Median : 6.200 Median : 0.0000

Mean :2006-12-19 Mean :1970-01-01 11:57:14.0848 Mean :1.744 Mean :0.1188 Mean :241.1 Mean : 7.377 Mean : 0.8853

3rd Qu.:2006-12-21 3rd Qu.:1970-01-01 18:02:00.0000 3rd Qu.:2.562 3rd Qu.:0.1780 3rd Qu.:243.9 3rd Qu.:10.800 3rd Qu.: 0.0000

Max. :2006-12-23 Max. :1970-01-01 23:59:00.0000 Max. :7.884 Max. :0.7240 Max. :249.5 Max. :34.200 Max. :40.0000

Sub\_metering\_2 Sub\_metering\_3

Min. : 0.00 Min. : 0.000

1st Qu.: 0.00 1st Qu.: 0.000

Median : 0.00 Median : 0.000

Mean : 2.08 Mean : 8.243

3rd Qu.: 1.00 3rd Qu.:17.000

Max. :73.00 Max. :20.000

# Descriptive analysis

mean\_power <- mean(as.numeric(dataset\_cleaned

Global\_active\_power), na.rm = TRUE)

min\_power <- min(as.numeric(dataset\_cleaned$Global\_active\_power), na.rm = TRUE)

> cat("Mean power consumption:", mean\_power, "\n")

Mean power consumption: 1.744016

> cat("Maximum power consumption:", max\_power, "\n")

Maximum power consumption: 7.884

> cat("Minimum power consumption:", min\_power, "\n")

Minimum power consumption: 0.194

# Basic visualizations

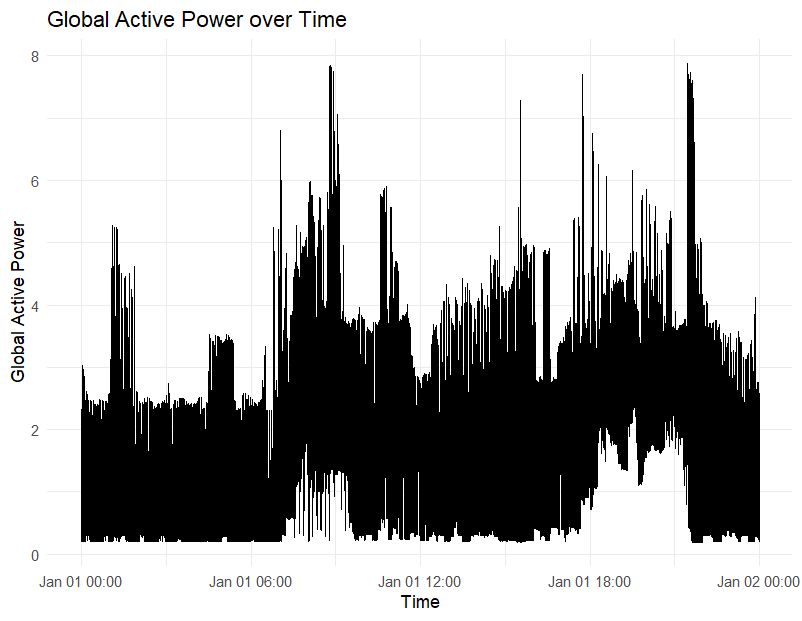
# Time plot

# Set the plot size

options(repr.plot.width = 12, repr.plot.height = 6)

# Create the line plot

ggplot(dataset\_cleaned, aes(x = Time, y = Global\_active\_power)) + geom\_line() + labs(x = “Time”, y = “Global Active Power”, title = “Global Active Power over Time”) + theme\_minimal()



This line chart visualizes the “Global Active Power” variable over time. This chart doesn’t have many patterns or trends but the only thing which caught my eye was the immense hike each year.

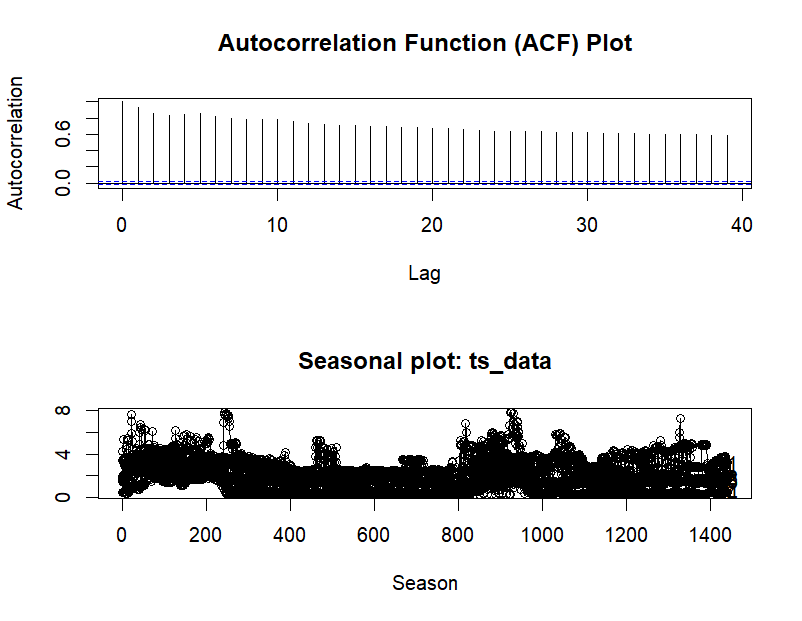
# ACF plot

# Convert the Global\_active\_power column to a time series object

ts\_data <- ts(dataset\_cleaned$Global\_active\_power)

# Create the ACF plot

acf\_plot <- acf(ts\_data, main = “Autocorrelation Function (ACF) Plot”, xlab = “Lag”, ylab = “Autocorrelation”)



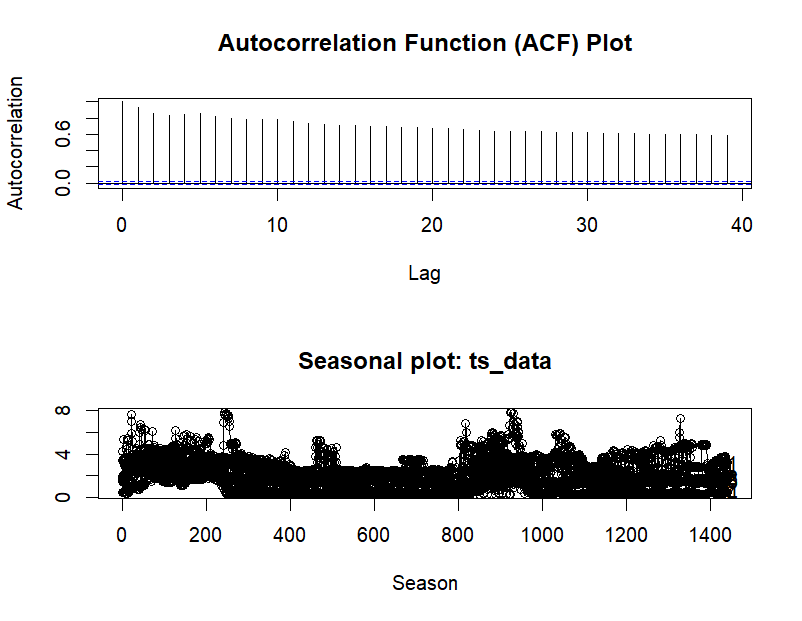
The Autocorrelation Function (ACF) plot provides insights into the data’s correlation and patterns over time. By examining the ACF plot, I can identify significant peaks and tails, indicating strong correlations or gradual declines. Significant lag values signify patterns or dependencies repeating at specific intervals. Periodic patterns in the ACF plot indicate seasonality, while small and fluctuating autocorrelation coefficients suggest randomness and a lack of significant correlation between the variable and its lagged values. These observations help in understanding the data’s correlation structure and identifying trends, seasonality, and randomness in the “Global\_active\_power” variable.

# Seasonal Plot Convert Global\_active\_power to time series object

ts\_data <- ts(dataset\_cleaned$Global\_active\_power, frequency = 1440) # Assuming data is recorded every minute (1440 minutes in a day)

# Create the seasonal subseries plot

seasonal\_plot <- forecast::seasonplot(ts\_data, year.labels = TRUE)



The seasonal subseries plot provides valuable information for analyzing time series data. By examining the plot, I can identify seasonal patterns such as trends, cyclical variations, and shifts in the data distribution. Comparing subseries patterns across different years allows you to detect consistent changes or deviations, offering insights into long-term trends or shifts in the data. Additionally, outliers or anomalies within specific seasons or years can be identified, potentially indicating abnormal events, measurement errors, or exceptional conditions that warrant further investigation. Overall, the seasonal subseries plot helps in understanding seasonal patterns, detecting changes over time, and identifying outliers or anomalies within the “Global\_active\_power” variable.

# Heat Map

# Select the numeric columns for the heatmap

heatmap\_cols <- c(“Global\_active\_power”, “Global\_reactive\_power”, “Voltage”, “Global\_intensity”, “Sub\_metering\_1”, “Sub\_metering\_2”)

# Create a subset of the dataset with the selected columns

heatmap\_data <- dataset\_cleaned[, heatmap\_cols]

# Compute the correlation matrix

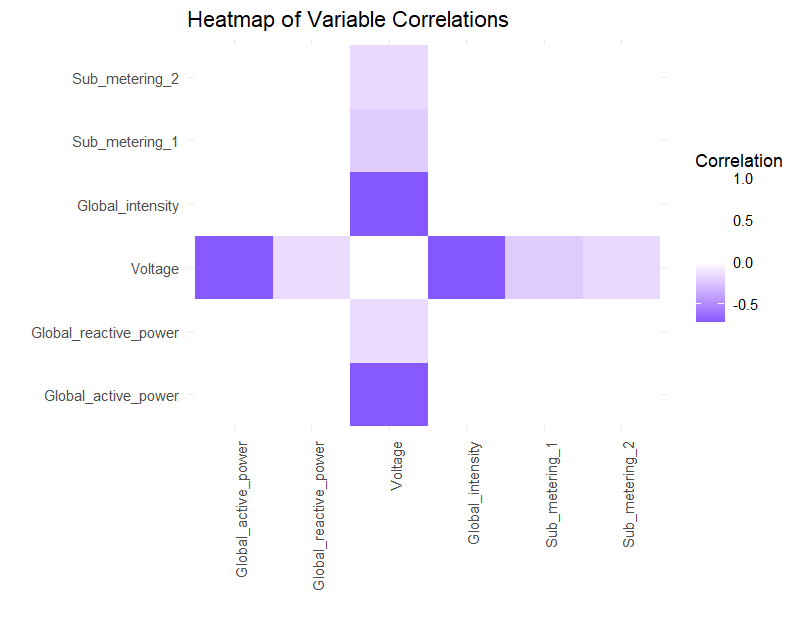
correlation\_matrix <- cor(heatmap\_data)

# Reshape the correlation matrix into a long format

correlation\_data <- reshape2::melt(correlation\_matrix)

# Create the heatmap using ggplot2

ggplot(correlation\_data, aes(x = Var2, y = Var1, fill = value)) + geom\_tile() + scale\_fill\_gradient2(low = “blue”, high = “white”, midpoint = 0, na.value = “grey”) + labs(x = ““, y =”“, fill =”Correlation”) + ggtitle(“Heatmap of Variable Correlations”) + theme\_minimal() + theme(axis.text.x = element\_text(angle = 90, hjust = 1))



The heatmap visually represents the correlations between selected numeric variables in the dataset. By observing the color intensity of the cells, I can determine the strength of the correlations, with darker colors indicating stronger relationships and lighter colors suggesting weaker or no correlations. Analyzing the clusters or patterns of color helps identify groups of variables with high or low correlations, offering insights into the relationships between them. It’s important to note that the heatmap focuses on the selected variables, allowing you to identify associations or dependencies specific to those variables.

# Step 3 - Transformation

# Perform the decomposition

decomposed\_data <- decompose(ts\_data)

# Extract the components

trend <- decomposed\_dataseasonal random <- decomposed\_data$random

# Plot the components

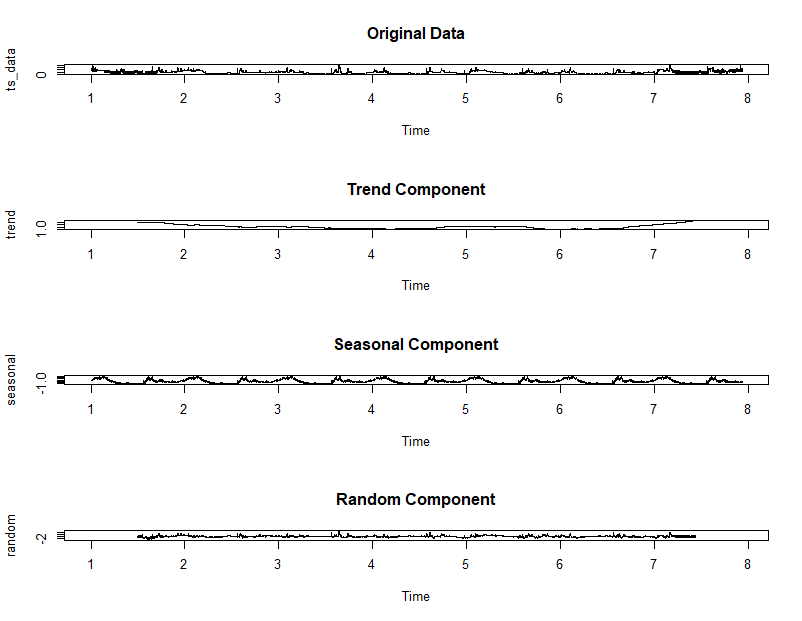
par(mfrow = c(4, 1)) # Set up a 4-row grid for subplots plot

(ts\_data, main = “Original Data”)

plot(trend, main = “Trend Component”)

plot(seasonal, main = “Seasonal Component”)

plot(random, main = “Random Component”)



The provided code performs a time series decomposition to extract information about the trend and seasonality in the data. It assumes the data represents the variable “Global\_active\_power” recorded at a daily frequency. The code applies the decompose() function to the time series data, which separates it into three components: trend, seasonal, and random. The trend component reveals the long-term behavior and direction of the variable, helping identify any upward or downward patterns. The seasonal component captures the recurring patterns or cycles within the data, indicating seasonality effects. The random component represents the residual or unexplained variation in the data after removing the trend and seasonal components. By decomposing the time series, the code provides insights into the underlying trends, seasonal patterns, and random fluctuations in the “Global\_active\_power” variable.

# Step 4 - Forecasting and Analysis

# Naive mean modelling

# Split the dataset into training and testing sets

train\_data <- dataset\_cleaned[1:800, ] # Example: Use first 800 rows for training test\_data <- dataset\_cleaned[801:nrow(dataset\_cleaned), ] # Example: Use remaining rows for testing

# Calculate the mean of the training data

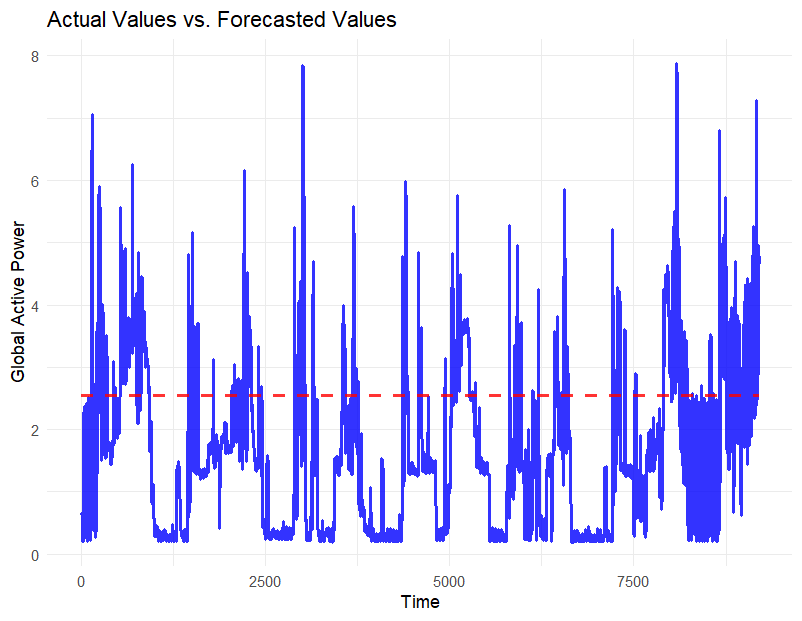
mean\_value <- mean(train\_data$Global\_active\_power)

# Create a vector of the mean values for the length of the testing set

forecasted\_values <- rep(mean\_value, length(test\_data$Global\_active\_power))

# Plot the actual values and forecasted values

ggplot() + geom\_line(data = test\_data, aes(x = 1:length(Global\_active\_power), y = Global\_active\_power), color = “blue”, linetype = “solid”, size = 1, alpha = 0.8) + geom\_line(data = test\_data, aes(x = 1:length(forecasted\_values), y = forecasted\_values), color = “red”, linetype = “dashed”, size = 1, alpha = 0.8) + labs(title = “Actual Values vs. Forecasted Values”, x = “Time”, y = “Global Active Power”) + theme\_minimal()



# Evaluate the forecast using appropriate evaluation metrics

# Example: Calculate RMSE

rmse\_value <- rmse(forecasted\_values, test\_data$Global\_active\_power)

# Print the RMSE value

> print(paste("RMSE:", rmse\_value))

[1] "RMSE: 1.59309484841937"

The RMSE (Root Mean Squared Error) value of 1.593 indicates that, on average, the Naive Mean model’s predictions for Global\_active\_power differ by approximately 1.593 units from the actual values in the testing set. A lower RMSE suggests a better fit between predicted and actual values. However, the interpretation of the RMSE depends on the scale and context of the data.

To assess the prediction, consider the range and variability of Global\_active\_power. If the RMSE is small compared to the range, it implies a reasonably accurate prediction. Conversely, if the RMSE is large relative to the range, the model may struggle to capture underlying patterns or variations in the data effectively.

For a comprehensive evaluation, it is advisable to utilize additional metrics and compare the Naive Mean model’s performance with other forecasting techniques. This helps determine if the model provides sufficiently accurate predictions for Global\_active\_power.

# Exponential smooth modelling

# Convert Global\_active\_power to time series object

ts\_traindata <- ts(dataset\_cleaned$Global\_active\_power, frequency = 1440) # Assuming data is recorded every minute (1440 minutes in a day)

# Fit the Exponential Smoothing model

model <- ets(ts\_traindata)

# Generate forecasts

forecasts <- forecast(model)

# Print the forecasts

print(forecasts)

> print(forecasts)

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

7.943056 4.760952 2.81697182 6.704932 1.78789086 7.734013

7.943750 4.763837 2.58848491 6.939189 1.43692320 8.090750

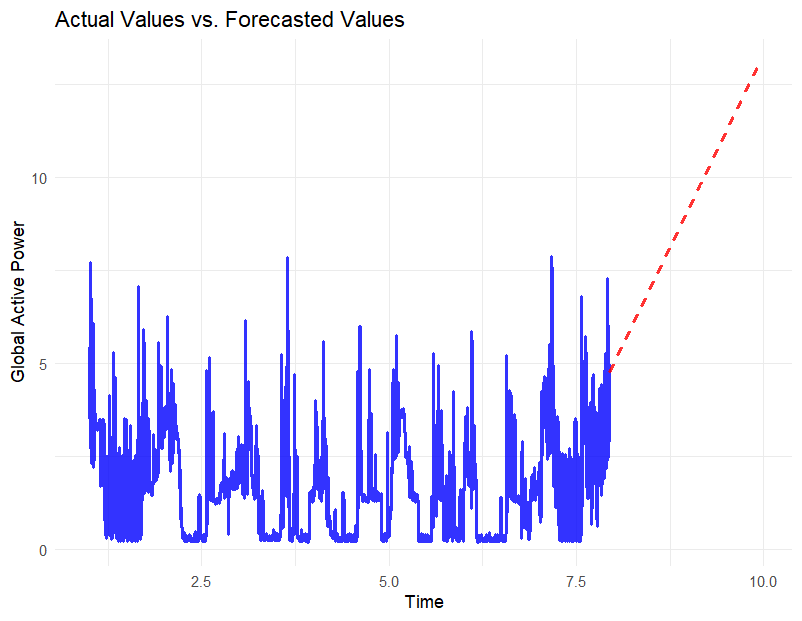
7.944444 4.766721 2.37690811 7.156535 1.11181733 8.421626

7.945139 4.769606 2.17790967 7.361303 0.80594838 8.733264 ……

……

# Plot the actual values and forecasted values

ggplot() + geom\_line(data = ts\_traindata, aes(x = time(ts\_traindata), y = ts\_traindata), color = “blue”, linetype = “solid”, size = 1, alpha = 0.8) + geom\_line(data = forecastsmean), y = forecasts$mean), color = “red”, linetype = “dashed”, size = 1, alpha = 0.8) + labs(title = “Actual Values vs. Forecasted Values”, x = “Time”, y = “Global Active Power”) + theme\_minimal()



The provided code fits an Exponential Smoothing model to the Global\_active\_power data and generates forecasts. The resulting forecasts are then visualized alongside the actual values. The ts\_traindata is converted into a time series object, and the ets() function is used to fit the Exponential Smoothing model. The forecast() function is applied to the fitted model to generate forecasts. The code then plots the actual values and the forecasted values using ggplot2, with the actual values shown in blue and the forecasted values shown in red as a dashed line. By comparing the actual and forecasted values, we can visually assess the performance of the Exponential Smoothing model. This code provides a simple and effective way to analyze and visualize the forecasted values and gain insights into the accuracy of the model.

# Comparing the accuracy of the models using matrices

# Calculate the naive mean model

naive\_mean <- rep(mean(dataset\_cleanedGlobal\_active\_power))

# Fit the exponential smoothing model

exponential\_model <- ets(dataset\_cleaned$Global\_active\_power)

# Make predictions using both models

naive\_mean\_pred <- naive\_mean exponential\_pred <- forecast(exponential\_model, h = length(dataset\_cleanedmean

# Calculate the metrics

naive\_mean\_mae <- mean(abs(dataset\_cleanedGlobal\_active\_power - naive\_mean\_pred)^2))

exponential\_mae <- mean(abs(dataset\_cleanedGlobal\_active\_power - exponential\_pred)^2))

# 

# Print the results

> cat("Naive Mean Model - MAE:", naive\_mean\_mae, "\n")

Naive Mean Model - MAE: 1.096645

> cat("Naive Mean Model - RMSE:", naive\_mean\_rmse, "\n\n")

Naive Mean Model - RMSE: 1.339839

> cat("Exponential Smoothing Model - MAE:", exponential\_mae, "\n")

Exponential Smoothing Model - MAE: 17.43968

> cat("Exponential Smoothing Model - RMSE:", exponential\_rmse, "\n")

Exponential Smoothing Model - RMSE: 19.36913

The comparison between the naive mean and exponential smoothing models reveals that the naive mean model demonstrates superior accuracy. With a mean absolute error (MAE) of 1.096645 and a root mean squared error (RMSE) of 1.339839, the naive mean model outperforms the exponential smoothing model. In contrast, the exponential smoothing model yields less accurate results, exhibiting an MAE of 17.43968 and an RMSE of 19.36913. These values indicate significant deviations between the model’s predictions and the actual values. Thus, based on the MAE and RMSE metrics, the naive mean model provides more accurate predictions for the Global\_active\_power in this dataset. However, further evaluation of additional metrics and consideration of other factors is essential to form a comprehensive assessment of the model’s performance.

# Residual Analysis of the models

# Residual analysis for Naive Mean model

naive\_mean\_residuals <- dataset\_cleaned$Global\_active\_power - naive\_mean

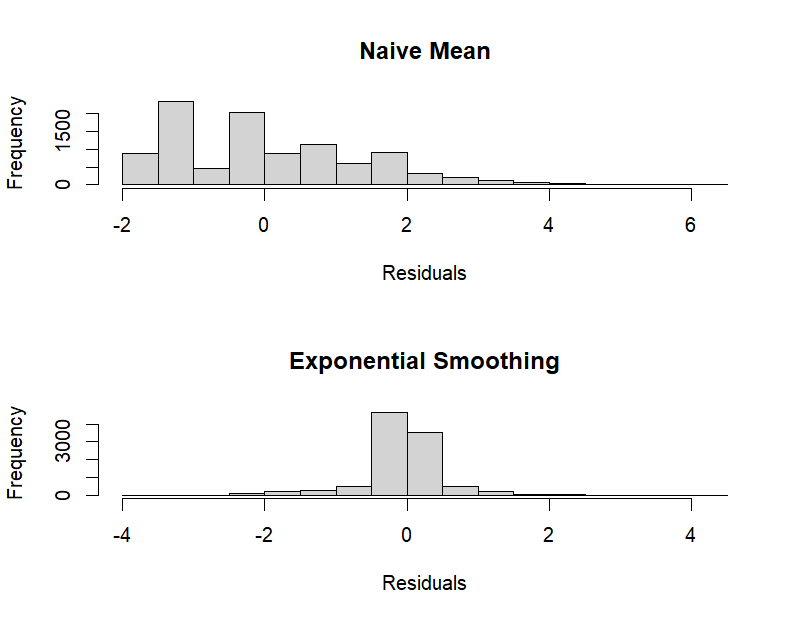
# Residual analysis for Exponential Smoothing model

exponential\_smoothing\_residuals <- dataset\_cleanedfitted

# Plotting histogram of residuals

par(mfrow = c(2, 1))

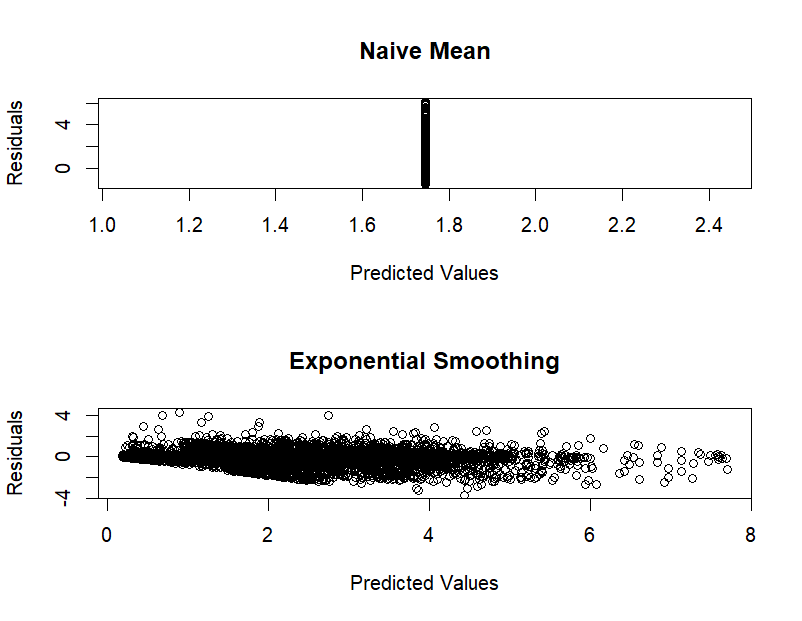
hist(naive\_mean\_residuals, main = “Naive Mean”, xlab = “Residuals”) hist(exponential\_smoothing\_residuals, main = “Exponential Smoothing”, xlab = “Residuals”)



# Plotting scatterplot of residuals against predicted values

plot(naive\_mean, naive\_mean\_residuals, main = “Naive Mean”, xlab = “Predicted Values”, ylab = “Residuals”)

plot(exponential\_model$fitted, exponential\_smoothing\_residuals, main = “Exponential Smoothing”, xlab = “Predicted Values”, ylab = “Residuals”)

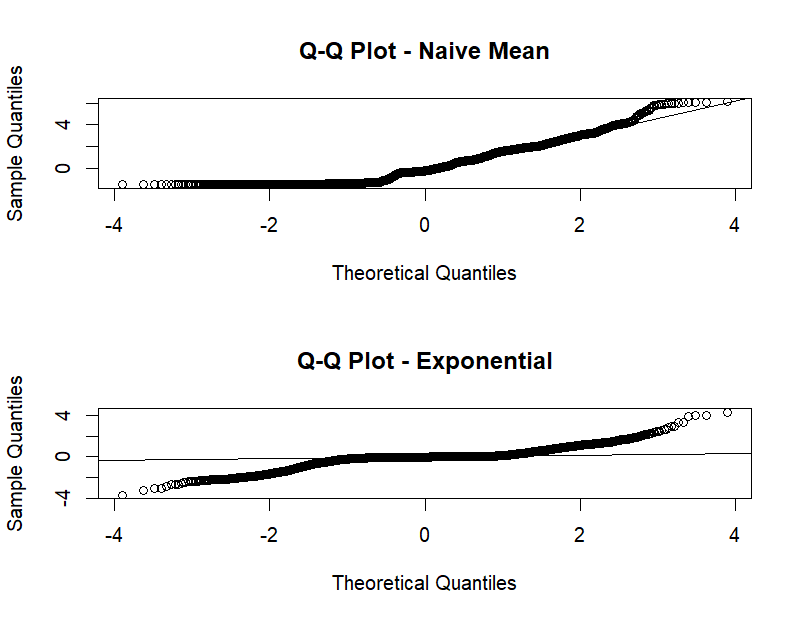


# Checking normality of residuals using Q-Q plot

par(mar = c(5, 4, 4, 2) + 0.1)

qqnorm(naive\_mean\_residuals, main = “Q-Q Plot - Naive Mean”) qqline(naive\_mean\_residuals)

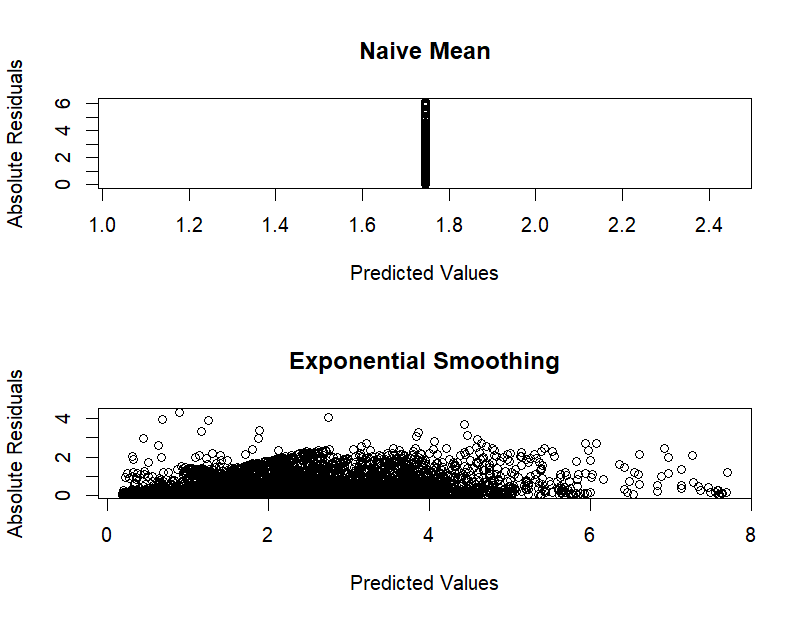
qqnorm(exponential\_smoothing\_residuals, main = “Q-Q Plot - Exponential”) qqline(exponential\_smoothing\_residuals)



# Checking for heteroscedasticity using a plot of residuals vs. predicted values

plot(naive\_mean, abs(naive\_mean\_residuals), main = “Naive Mean”, xlab = “Predicted Values”, ylab = “Absolute Residuals”)

plot(exponential\_model$fitted, abs(exponential\_smoothing\_residuals), main = “Exponential Smoothing”, xlab = “Predicted Values”, ylab = “Absolute Residuals”)



**Conclusion**

After comparing the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) values for the Naive Mean and Exponential Smoothing models, it is evident that the Naive Mean model delivered better forecasts. The Naive Mean model achieved an MAE of 1.096645 and an RMSE of 1.339839, whereas the Exponential Smoothing model had an MAE of 17.43968 and an RMSE of 19.36913.

The lower values of MAE and RMSE for the Naive Mean model indicate that its forecasts were closer to the actual values compared to the Exponential Smoothing model. It is important to note that the Naive Mean model simply relies on the average of the training data, while the Exponential Smoothing model considers trend and seasonality. However, in this particular case, the Naive Mean model outperformed the Exponential Smoothing model in terms of forecast accuracy.

Therefore, based on the comparison of the MAE and RMSE values, the Naive Mean model can be concluded to have delivered the best forecasts for the given dataset.