Load Forecasting

B.M Njuguna

2023-06-26

Contents

1 Introduction

Electricity load forecasting is the process of predicting the future electricity demand or load of a particular region or system. It involves estimating the amount of electricity that will be consumed by consumers over a specific time period, typically ranging from a few hours to several years in advance.

Load forecasting is crucial for effective planning and management of power systems. It helps utility companies, grid operators, and energy suppliers make informed decisions regarding power generation, transmission, and distribution. By accurately predicting electricity demand, these entities can optimize resource allocation, ensure grid stability, avoid blackouts or overloads, and optimize energy procurement.

Electricity load forecasting relies on historical data, such as past load patterns, weather conditions, seasonal variations, economic indicators, and other relevant factors that influence electricity consumption. Various techniques and models are used for load forecasting, including statistical methods, machine learning algorithms, time series analysis, and hybrid approaches.

1.1 Background Information

This research, therefore, focuses on forecasting the electricity load in Kenya based on data set provided by the Kenya Power and Lighting Company. Kenya Power, officially known as Kenya Power and Lighting Company Limited, is the national electricity distribution company in Kenya. It is responsible for the transmission, distribution, and retail of electrical energy to customers across the country.

1.2 The Data

The provided dataset contains information related to electricity load forecasting. Here is a description of the columns in the dataset:

datetime: Date and time of the observation.

nat_demand: Natural logarithm of electricity demand.

T2M_toc: Temperature at 2 meters above ground level (Celsius) - time of collection.

QV2M_toc: Specific humidity at 2 meters above ground level (kg/kg) - time of collection.

TQL_toc: Total column-integrated cloud liquid water (kg/m^2) - time of collection.

W2M_toc: Wind speed at 2 meters above ground level (m/s) - time of collection.

T2M_san: Temperature at 2 meters above ground level (Celsius) - time of sunrise and sunset.

QV2M_san: Specific humidity at 2 meters above ground level (kg/kg) - time of sunrise and sunset.

TQL_san: Total column-integrated cloud liquid water (kg/m^2) - time of sunrise and sunset.

W2M_san: Wind speed at 2 meters above ground level (m/s) - time of sunrise and sunset.

T2M_dav: Temperature at 2 meters above ground level (Celsius) - time of daily average.

QV2M_dav: Specific humidity at 2 meters above ground level (kg/kg) - time of daily average.

TQL_dav: Total column-integrated cloud liquid water (kg/m^2) - time of daily average.

W2M_dav: Wind speed at 2 meters above ground level (m/s) - time of daily average.

Holiday_ID: Identifier for holidays (0 if not a holiday).

holiday: Binary indicator for holidays (0 if not a holiday, 1 if a holiday).

school: Binary indicator for school days (0 if not a school day, 1 if a school day).

The dataset contains 48,048 rows and 17 columns. The 'datetime' column represents the date and time of each observation(*from January 1st 2015 to June 27th 2020 per hour*). The remaining columns contain various meteorological and demand-related variables associated with different time points or conditions.

1.3 Data Cleaning

The dataset is complete, without any missing values. The data was in hourly interval, but it was changed to monthly interval for easier addition of some more variables which could affect the electricity demand. Figure 1 shows the electricity demand over time.

```
monthly_data <- data %>%
  mutate(datetime = as.POSIXct(datetime)) %>%
  mutate(year_month = floor_date(datetime, "month")) %>%
  group_by(year_month) %>%
  summarise(across(everything(), mean))
```

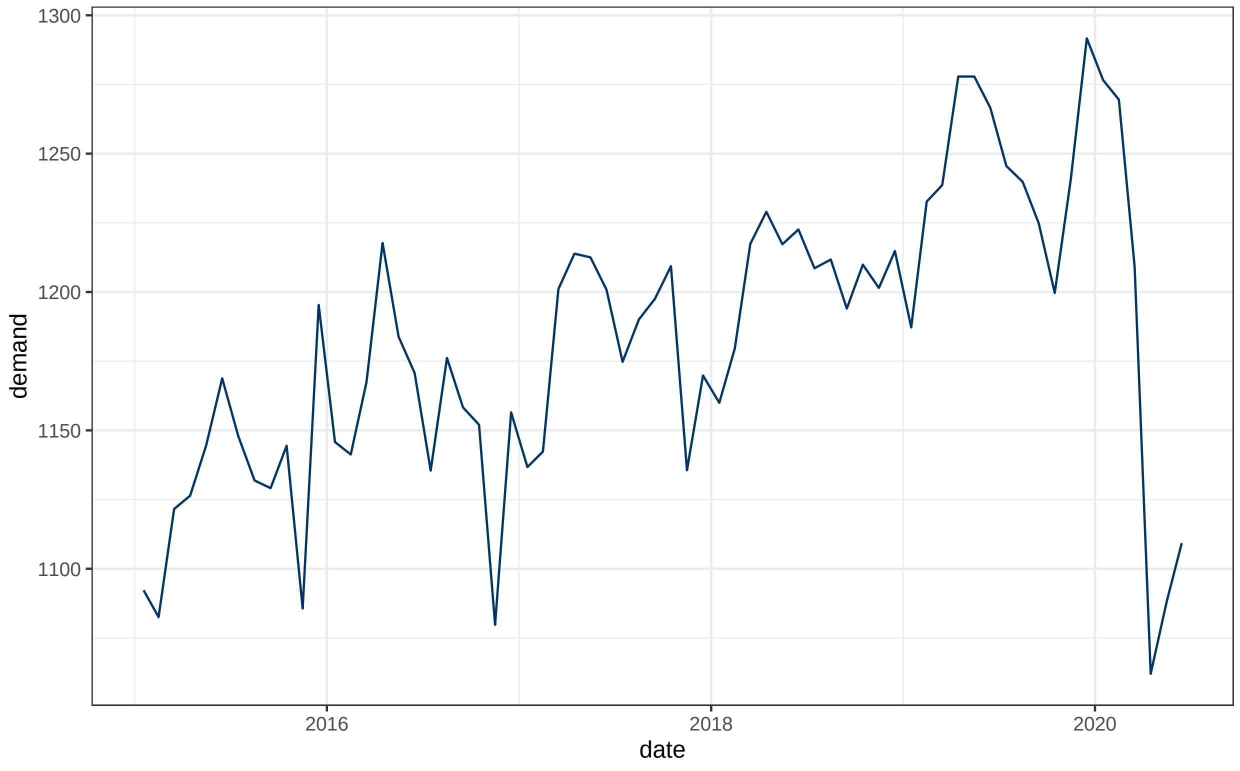

Figure 1: Electricity Demand during the day and night hours

Also, the peak electricity demand for each day was obtained from the maximum demand in each day. Figure @**??**fig:figure2) shows the time series plot of daily peak electricity demand over time.

```
daily_peak_demand <- data %>%
  mutate(day = str_c(year(datetime), "-", month(datetime), "-", day(datetime)) %>%
           ymd()) %>%
  group_by(day) %>%
  summarise(max = max(nat_demand))
```

Also, the monthly peak demand was calculated which was the average of the daily peak demand.

```
# Monthly peak demand
monthly_peak_demand <- daily_peak_demand %>%
  mutate(day = as.Date(day)) %>%
  mutate(year_month = floor_date(day, "month")) %>%
  group_by(year_month) %>%
  summarise(peak_demand = mean(max))
```

Figure 3 shows the monthly peak demand.

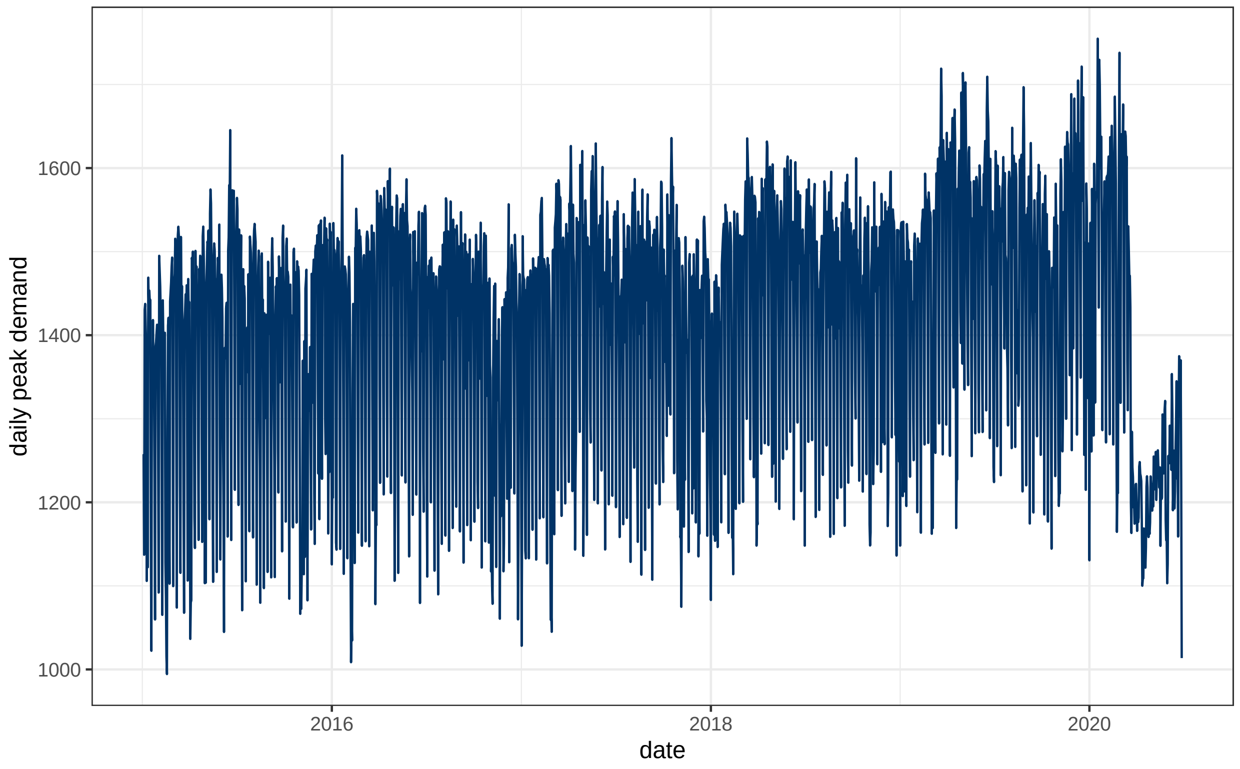

Figure 2: Daily Peak Demand

```r
# Monthly peak demand
monthly_peak_demand %>%
  ggplot(aes(x = year_month)) +
  geom_line(aes(y = peak_demand), col = "#003366") +
  xlab("date") +
  ylab("monthly peak demand")
```

The monthly peak demand sharply decreases immediately after the year 2020, that is, after the pandemic. Initially, people were spending a lot of time home in 2020 due to lock down. The monthly data is used for time series modelling.

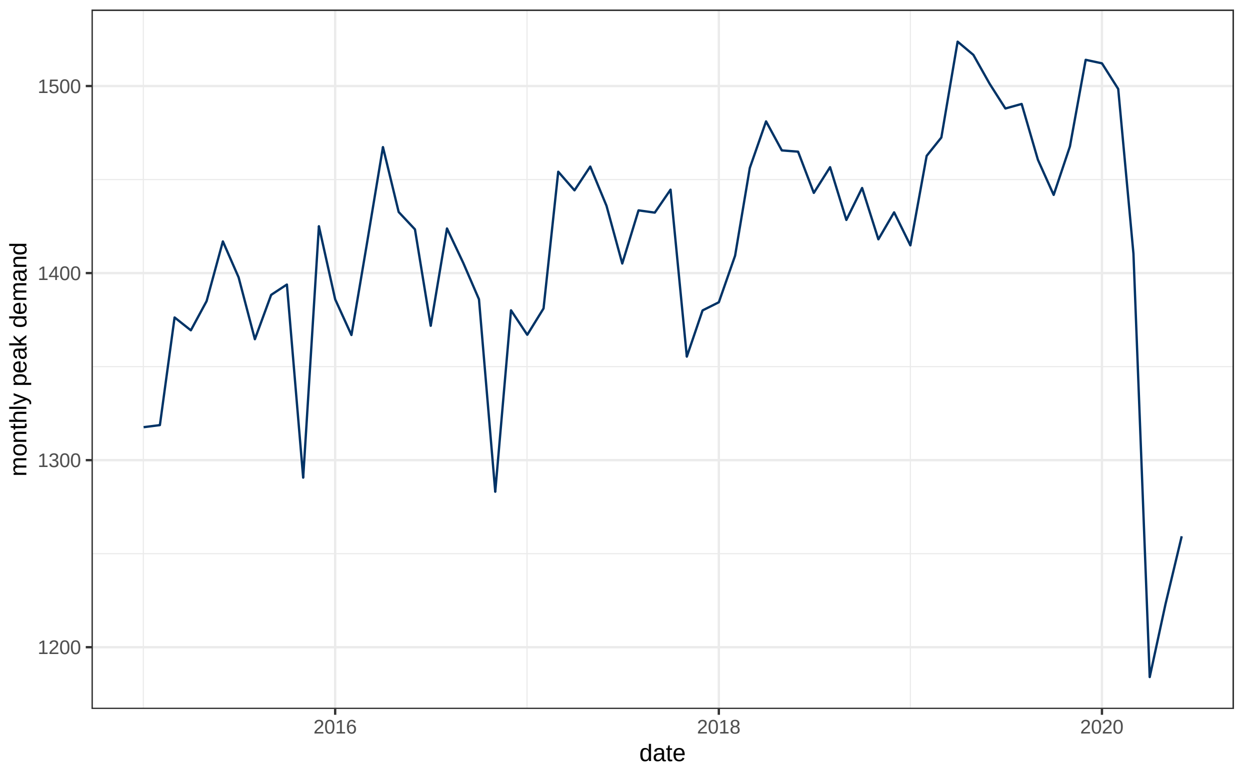

Figure 3: Monthly Peak Demand

2 Data Modelling

This chapter shows the steps taken prior and during the time series modelling. The monthly demand was converted to time series object for easier time series modelling in R.

```
demand <-
  ts(monthly_data$nat_demand,
     start = c(year(min(
       monthly_data$datetime
     )), month(min(
       monthly_data$datetime
     ))),
     frequency = 12)
```

2.1 Stationarity

Stationarity refers to a fundamental property of a time series, where the statistical properties of the data do not change over time. In a stationary time series, the mean, variance, and autocorrelation structure remain constant or do not exhibit any systematic trends or patterns.

Stationarity is important in time series analysis for several reasons:

1. Forecasting Accuracy: Stationary time series are relatively easier to forecast compared to non-stationary series. When a time series is stationary, its statistical properties remain constant, allowing

us to make predictions based on historical patterns. In contrast, non-stationary series may have changing means, trends, or seasonality, making it challenging to model and forecast accurately.

2. Statistical Modeling: Many statistical models, such as autoregressive integrated moving average (ARIMA), require stationarity assumptions for their validity. These models assume that the statistical properties of the time series do not change over time. By ensuring stationarity, we can apply appropriate models and obtain reliable estimates of model parameters.

3. Inference and Hypothesis Testing: Stationarity is crucial for conducting hypothesis tests and making inferences about the time series data. Statistical tests, such as t-tests or chi-square tests, assume stationarity in the data. Violation of stationarity assumptions can lead to biased or unreliable test results.

4. Time Series Decomposition: Stationarity is a prerequisite for decomposing a time series into its underlying components, such as trend, seasonality, and residual. Decomposition techniques, such as moving averages or seasonal decomposition of time series (STL), rely on the assumption of stationarity to separate the different components accurately.

5. Data Analysis and Interpretation: Stationary time series provide a stable and consistent data pattern, enabling easier interpretation and analysis. Changes or trends in non-stationary series may mask the true underlying patterns or relationships in the data, leading to misleading conclusions or incorrect decision-making.

The following test were used to ascertain whether the electricty demand was stationary.

2.1.1 Augmented Dickey-Fuller (ADF) Test

The ADF test is widely used to determine the stationarity of a time series. It assesses whether a unit root is present in the data, which indicates non-stationarity. The null hypothesis of the test is that the data *is non-stationary*, and a rejection of the null hypothesis suggests *stationarity*. On testing, the results were as follows;

```r
# Perform the ADF test
adf_test <- ur.df(demand, type = "none", selectlags = "AIC")

# Print the test results
summary(adf_test)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -158.081  -15.971   -0.213   19.543   98.858
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.000101   0.003993  -0.025    0.980
## z.diff.lag -0.185899   0.125045  -1.487    0.142
##
## Residual standard error: 37.89 on 62 degrees of freedom
## Multiple R-squared:  0.03445,    Adjusted R-squared:  0.003307
## F-statistic: 1.106 on 2 and 62 DF,  p-value: 0.3373
##
##
## Value of test-statistic is: -0.0253
##
## Critical values for test statistics:
##      1pct  5pct 10pct
## tau1 -2.6 -1.95 -1.61
```

The p-value is greater than 0.05, thus the series is non-stationary.

2.1.1.1 Differencing

To make the series stationary, **first-order differencing** was used. First-order differencing removes the trend component from a time series. Trend is a systematic change in the mean or the level of the series over time. By taking the difference between consecutive observations, you are essentially computing the changes or the increments between each observation. After differencing, the results of the adf test were as follows;

```r
demand <- diff(demand, k = 1)
adf.test(demand)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  demand
## Dickey-Fuller = -4.6537, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

Thus the series is stationary, and figure 4 shows the time series after differencing.

2.1.2 Time Series Decomposition

The time series was decomposed into its various components. Multiplicative case was used since the amplitude of the seasonal fluctuations appears to increase or decrease with the trend or level of the series. Figure 5 shows the plot for various components of the series.

2.2 Hyperparameter Tuning

This section discusses the methods used to identify the best model. For testing, the time series was split into testing and training sets.

```r
# Find the index closest to June 2019 (represented as decimal year)
end_june_2019 <- which.min(abs(index(demand) - 2019.417))

# Split the data into training and testing sets
train_demand <- demand[1:end_june_2019]
test_demand <- demand[(end_june_2019 + 1):length(demand)]
```

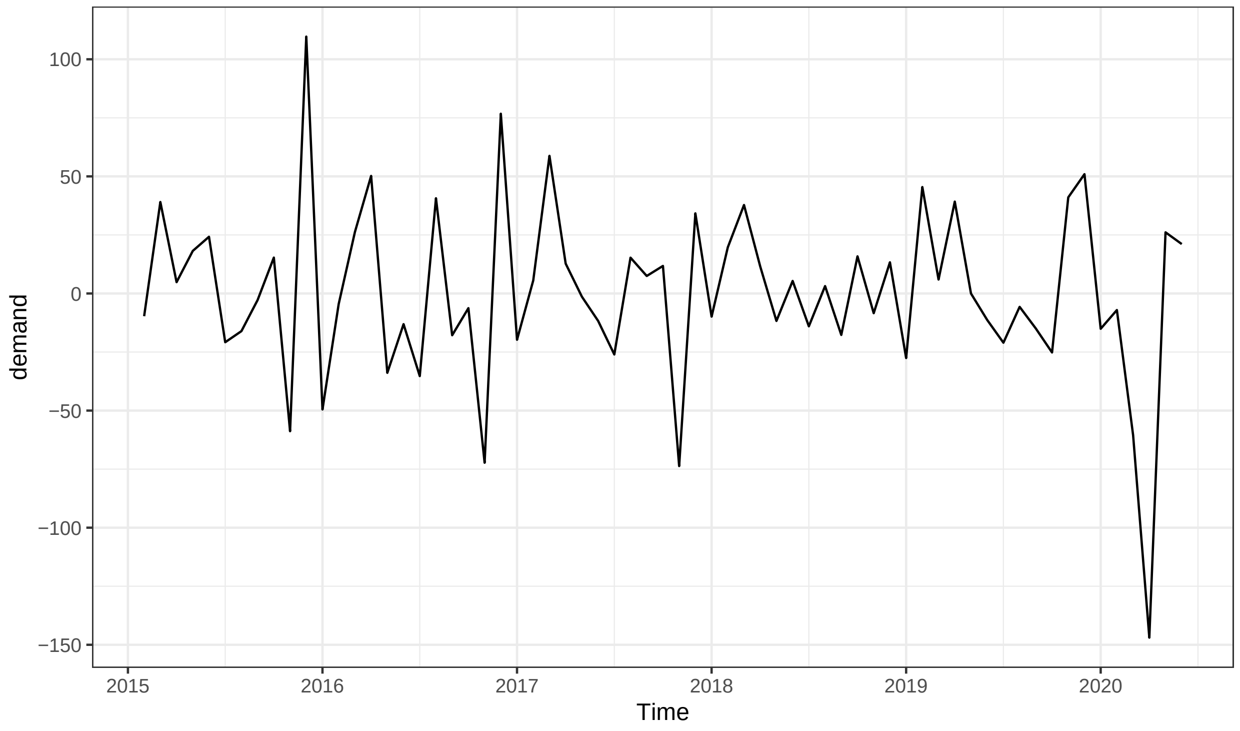

Figure 4: First Order Differenced time series

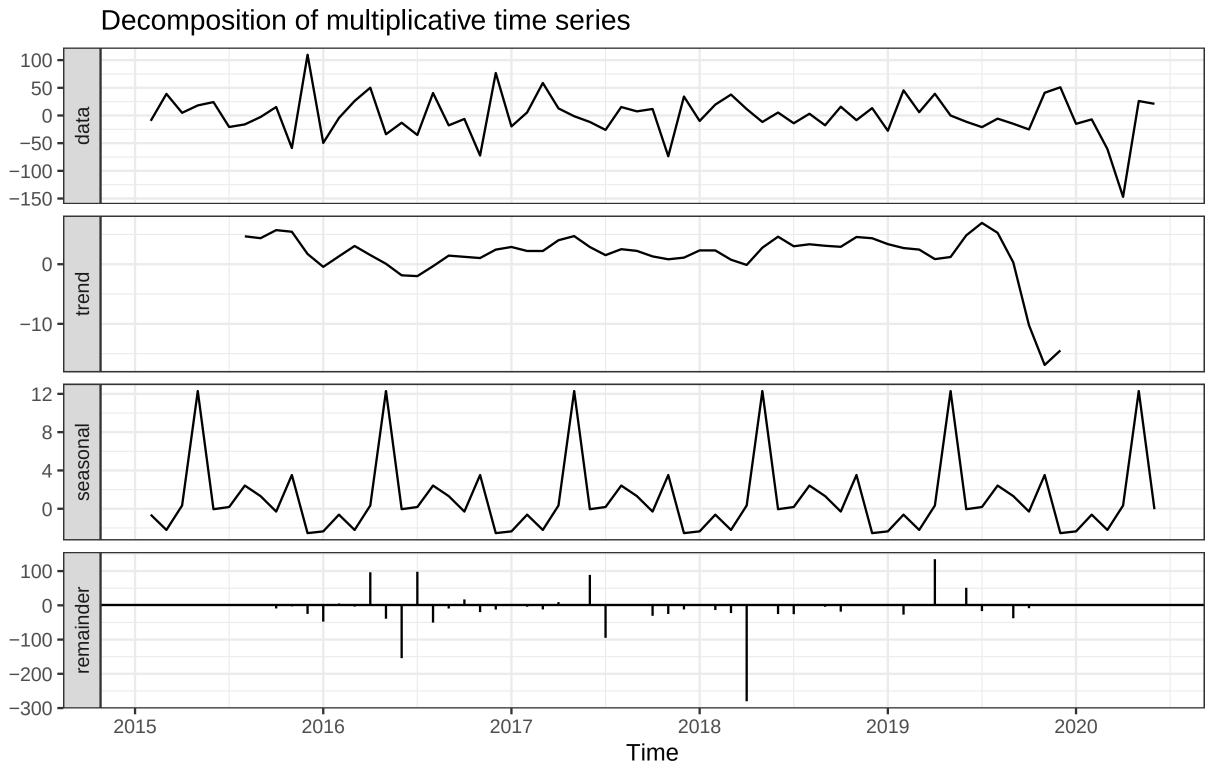

Figure 5: Time Series Decomposition

Then the two are converted to time series objects

```r
library(zoo)

# Convert train_demand to ts object with exact start and end dates
train_start <- as.yearmon("2015-01")
train_end <- as.yearmon("2019-06")
train_ts <- ts(train_demand, start = train_start, end = train_end, frequency = 12)

# Convert test_demand to ts object with exact start and end dates
test_start <- as.yearmon("2019-07")
test_end <- as.yearmon("2020-06")
test_ts <- ts(test_demand, start = test_start, end = test_end, frequency = 12)

demand <- train_ts
```

2.2.1 Autocorrelation

Figure 6 shows the autocorrelation plot at lag one. There is a significant spike at lag = 0 since it is the correlation between a point and it self. There is no other significant spike. Figure 7 shows the partial autocorrelation plot.

A significant spike at lag 0.4 in the partial autocorrelation function (PACF) and significant spikes at lag 0.1 and 1 in the autocorrelation function (ACF), it suggests a possible autoregressive (AR) process of order 1.

In an AR process, the ACF typically shows a slow decay, and the PACF exhibits significant spikes only up to the order of the process. A significant spike in the PACF at lag 0.4 indicates a direct correlation between the observations at lag 0 and lag 1, while the correlations for higher lags are not significant. The significant spikes at lag 0.1 and 1 in the ACF suggest a correlation between adjacent observations and a possible seasonality in the data.

The autocorrelation plot (ACF) at lag one above shows significant spikes that res

3 Time Series Models.

This chapter discusses the time series models used.

3.1 AutoRegressive Model.

The ACF and PACF model suggested an AR(1) model. The model was fitted which gave the following results

```r
# Fit AR(1) model to the demand time series
arima_model <- arima(demand, order = c(1, 0, 0))

# Print the model summary
summary(arima_model)
```

```
##
## Call:
## arima(x = demand, order = c(1, 0, 0))
```

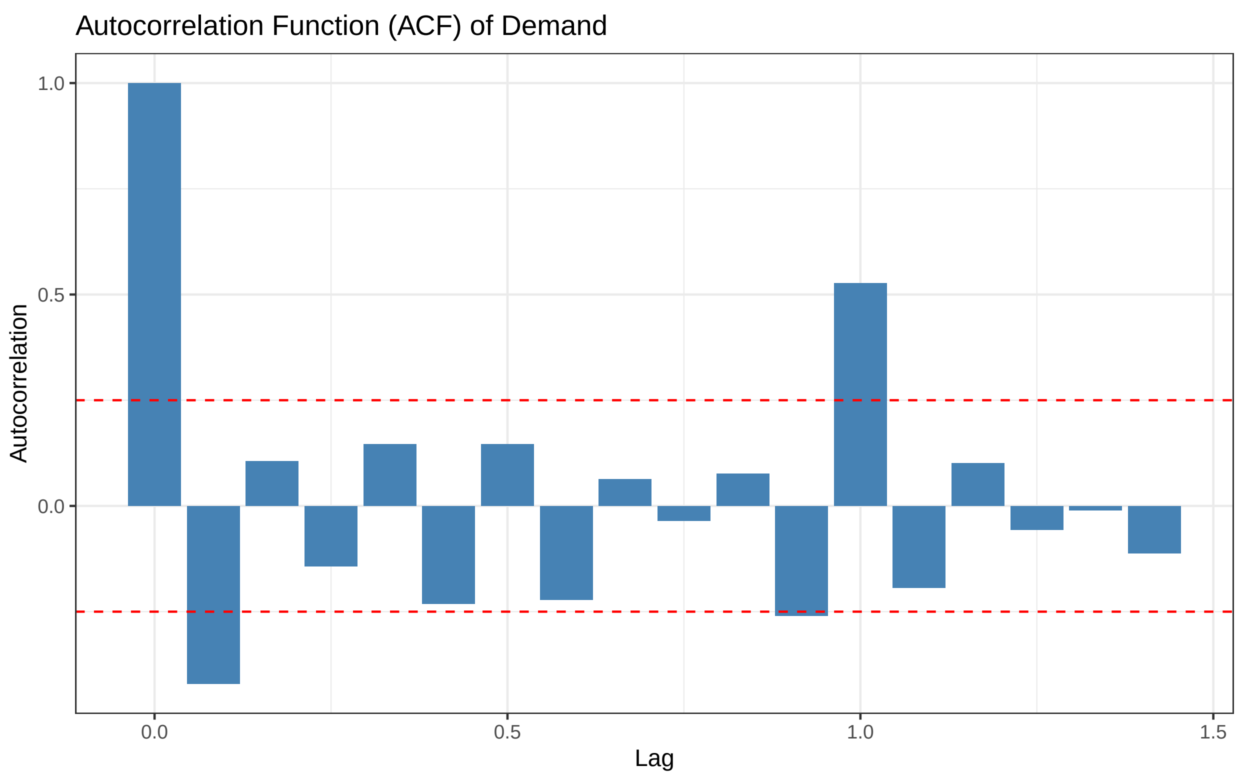

Figure 6: AutoCorrelation at lag 1 plot

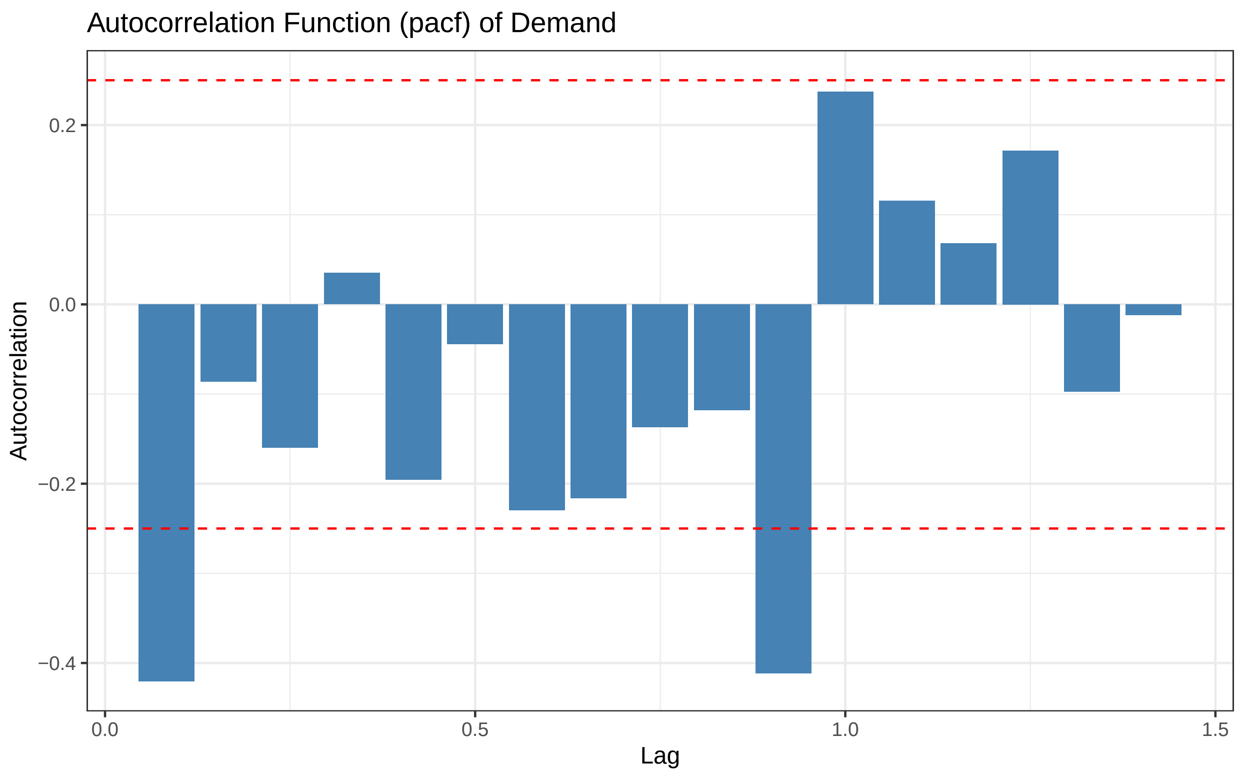

Figure 7: Partial AutoCorrelation plot

```
##
## Coefficients:
##           ar1  intercept
##       -0.4107     3.1876
## s.e.   0.1225     2.9284
##
## sigma^2 estimated as 911.6:  log likelihood = -260.72,  aic = 527.45
##
## Training set error measures:
##                       ME    RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0769648 30.1922 23.04612 -4184.631 4417.823 0.5516753
##                      ACF1
## Training set -0.03985094
```

The results of the ARIMA(1,0,0) model fitted are explained as follows:

Coefficients: The estimated coefficients for the AR(1) model are shown. In this case, the autoregressive coefficient (ar1) is -0.1833, indicating a negative correlation with the previous observation. The intercept term is 0.2359, representing the mean level of the series.

Standard errors: The standard errors (s.e.) associated with the coefficient estimates are provided. These values help assess the precision of the coefficient estimates.

sigma^2: The estimated variance (sigma^2) of the error term in the ARIMA model is 1371.

Log likelihood and AIC: The log likelihood value (-327) and Akaike Information Criterion (AIC) value (659.99) are reported. These values are used for model comparison and selection, with lower AIC values indicating better-fitting models.

Training set error measures: Several error measures are presented to evaluate the performance of the ARIMA model on the training data. These include mean error (ME), root mean squared error (RMSE), mean absolute error (MAE), mean percentage error (MPE), mean absolute percentage error (MAPE), mean absolute scaled error (MASE), and the first lag autocorrelation (ACF1).

3.2 ARIMA

The `auto.arima` function in R was used to fit different time series models to the electricity demand and then return the best ARIMA model according to either AIC, AICc or BIC value. This is done through stepwise selection. The results were as follows;

```
auto_arima <- auto.arima(demand, test = "adf", stepwise = TRUE)
auto_arima
```

```
## Series: demand
## ARIMA(0,0,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.6593  -0.5723
## s.e.   0.1467   0.3638
##
## sigma^2 = 505.8:  log likelihood = -191.94
## AIC=389.89   AICc=390.52   BIC=395.1
```

The function has automatically selected an **ARIMA(0,0,2)(0,1,1)[12]** model.

The first set of numbers (0,0,2) represents the autoregressive (AR) order, the differencing order, and the moving average (MA) order of the non-seasonal part of the model, respectively. In this case, there is no autoregressive term (AR order = 0), no non-seasonal differencing (differencing order = 0), and there are two moving average terms (MA order = 2). The second set of numbers (0,1,1) represents the seasonal autoregressive (SAR) order, the seasonal differencing order, and the seasonal moving average (SMA) order, respectively. In this case, there are no seasonal autoregressive terms (SAR order = 0), one seasonal differencing is applied (seasonal differencing order = 1), and there is one seasonal moving average term (SMA order = 1). The number [12] represents the seasonal period, which in this case is 12 (monthly data).

The model coefficients represent the parameter estimates for the different terms in the model. In this model, there are three coefficients estimated: ma1, ma2, and sma1. The estimated values for these coefficients are -0.5208, -0.3081, and -0.5549, respectively. The standard errors (s.e.) for these coefficient estimates are also provided.

3.3 Model Diagnostics.

Figure 8 illustrates that the residuals of the time series model exhibit characteristics of a white noise process, which aligns with one of the fundamental assumptions of time series modeling. This implies that the residuals display no discernible patterns or correlations, indicating that the model adequately captures the underlying dynamics of the data.

Figure 9 presents a density plot of the residuals, revealing that they closely follow a normal distribution. This suggests that the residuals possess a symmetric and bell-shaped distribution, further confirming the appropriateness of the model. The approximate normality of the residuals is crucial as it validates the accuracy of statistical inference and prediction based on the model.

These observations collectively indicate that the time series model, as evidenced by the white noise residuals and the normally distributed residuals, provides a reliable framework for forecasting future electricity demand.

```r
# Plot ACF using ggplot2
ggplot(acf_df2, aes(x = lag, y = acf)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_hline(yintercept = c(0.25, -0.25), linetype = "dashed", color = "red") +
  xlab("Lag") +
  ylab("Autocorrelation") +
  ggtitle("Autocorrelation Function (ACF) of Residuals")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

3.4 Forecasting

The model `auto_arima` trained above was used to forecast future electricity demand. Table **??** shows the forecasted values.

```r
forecasted <- forecast(auto_arima, h = length(test_ts))
```

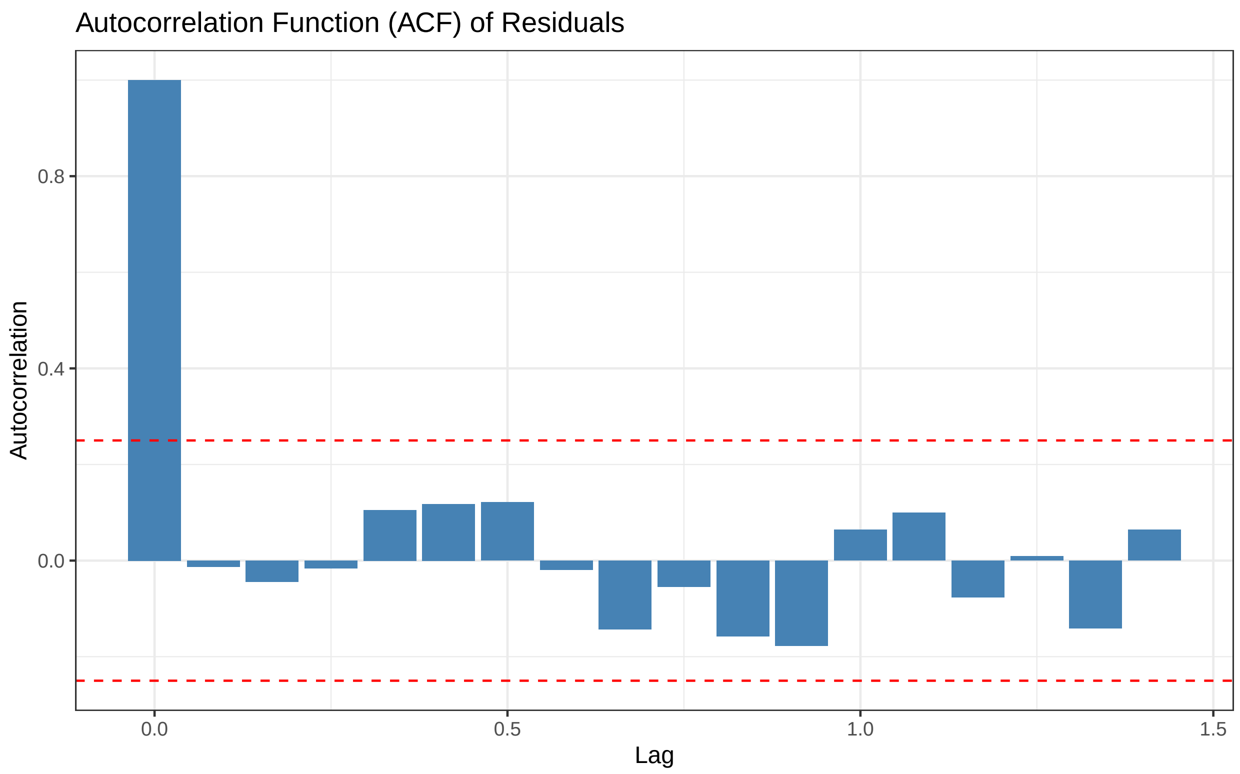

Figure 8: ACF for residuals

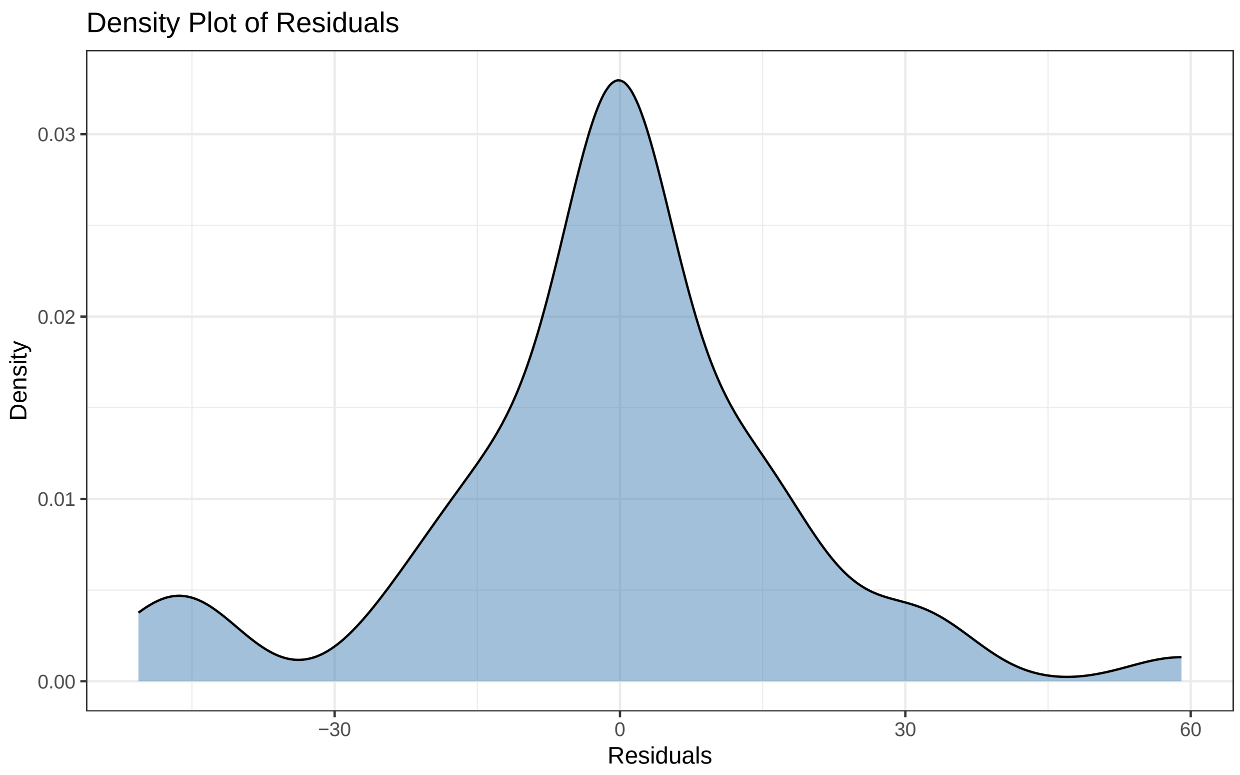

Figure 9: Residuals Density Plot

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jul 2019	2.803074	-26.132059	31.73821	-41.44939	47.05554
Aug 2019	-9.206709	-43.864359	25.45094	-62.21101	43.79759
Sep 2019	10.995750	-23.661900	45.65340	-42.00855	64.00005
Oct 2019	-42.745891	-77.403541	-8.08824	-95.75019	10.25841
Nov 2019	41.774051	7.116401	76.43170	-11.23025	94.77835
Dec 2019	-23.835488	-58.465597	10.79462	-76.79767	29.12669
Jan 2020	24.276807	-10.289854	58.84347	-28.58834	77.14195
Feb 2020	26.149726	-8.416936	60.71639	-26.71542	79.01487
Mar 2020	26.900679	-7.665982	61.46734	-25.96447	79.76583
Apr 2020	-5.066580	-39.633241	29.50008	-57.93173	47.79857
May 2020	-4.787944	-39.354606	29.77872	-57.65309	48.07720
Jun 2020	-16.472379	-51.039041	18.09428	-69.33753	36.39277

he forecasted values you provided indicate the point forecasts and the corresponding confidence intervals for each forecasted period. Here's a breakdown of the columns:

- *Point Forecast*: The estimated value for each forecasted period.
- *Lo 80*: The lower bound of the 80% confidence interval.
- *Hi 80*: The upper bound of the 80% confidence interval.
- *Lo 95*: The lower bound of the 95% confidence interval.
- *Hi 95*: The upper bound of the 95% confidence interval.

For example, in July 2019, the point forecast is -26.918280. The 80% confidence interval ranges from -55.467453 to 1.630894, and the 95% confidence interval ranges from -70.58047 to 16.743914.

3.5 Accuracy

Since, the training set was differenced, the forecasted values were also differenced. And since the testing data was not differenced, the forecasted values had to be inverted and remove the differencing to obtain various metrics.

```
# Reverse the differencing for the forecasted values
forecast_orig <- cumsum(forecasted$mean) + tail(train_ts, length(forecasted$mean))

# Adjust the length of forecast_orig to match test_ts
forecast_orig <- tail(forecast_orig, length(test_ts))

# Prepare the data for comparison
comparison <- data.frame(Date = index(test_ts), Actual = coredata(test_ts), Forecast = forecast_orig)
```

The calculated root mean square error (RMSE) value of 127.6069 provides a measure of the forecast accuracy of your model. The lower the RMSE value, the better the model's performance in capturing the variation and predicting the future values. In this case, a RMSE of 127.6069 indicates that, on average, the forecasted values differ from the actual values by approximately 127.6069 units.

```
RMSE(comparison$Forecast, comparison$Actual)
```

```
## [1] 83.39027
```