BAN 673 TIME SERIES ANALYTICS

SUMMER '21

Residential Power Forecasting for a 2-storied residential house

Members:

Devi Nadimpally(yy4246)

Krishna Sahoo(uz6293)

Nishtha Ranpara(gg9226)

Prerna Nautiyal(ru6495)

Yash Shah(ax9972)

Table of Contents **Page No.**

1. SUMMARY 4

2. INTRODUCTION 5

3. MAIN CHAPTER 6

3.1 Define Goal 6

3.2 Get Data 6

3.3 Explore and Visualize Series 7

3.3.1 Check Predictability 9

3.4 Data Processing 10

3.4.1 Applying Regression on Daily Data 10

3.4.2 Modifying Data and Converting data into weekly data 11

3.4.3 Combining all variables into single dataframes 12

3.4.4 Creating Time Series dataframes 13

3.5 Partitioning Time Series 13

3.6 Apply Forecast Methods 14

3.6.1 Regression Models 14

a. Regression Model with Linear Trend 14

b. Regression Models with Quadratic trend 15

c. Regression Models with Seasonality 16

d. Regression Models with Linear + Seasonality 17

e. Regression Models with Quadratic Trend + Seasonality 18

3.6.2 Two-level models 20

3.6.3 Regression Models with external variables 26

3.7 Evaluate and Compare Performance 29

3.7.1 Model 1(Regression with Linear Trend and Seasonality) 29

3.7.2 Model 2(Two-level Model) 34

3.7.3 Model 3(ARIMA Model) 36

3.8 Implement the forecast on Future Data 41

4. CONCLUSION 43

5. BIBLIOGRAPHY 44

1. SUMMARY

For this project, the data set contains daily power usage of a 2storied house located in Houston, Texas, USA (from June 1st, 2016, to July 7th, 2020). We aggregated daily data into a weekly dataset for the analysis purposes (to cancel out excess noise). The objective is to predict weekly power usage for a future month.

From the visualization we identified that aggregated weekly datasets have an additive seasonality which reaches the peak around 32nd week of each year and the trend is not very clear as the stl() plot shows a downward and upward trend at different points of time before becoming stagnant from mid of 2019 onwards. Also, the data is highly auto correlated, as the autocorrelation coefficients in maximum lags, out of 52 lags, are significant.

Regression-based models, Multi regression models and automated autoregressive integrated moving average models (ARIMA) were utilized for this project. In order to find the best model, the different variations of regression models were enhanced, where appropriate, with a trailing moving average for residuals and an autoregressive model for residuals. In the Multi regression model, the selected external variables are used based on the heatmap and highlighted map (which shows correlation between variables). Model accuracy was evaluated based on the RMSE and MAPE. Using the above-mentioned accuracy metrics, the best model to predict future forecast was the regression model with linear trend and seasonality with a trailing moving average for residuals.

2. INTRODUCTION

Texas is a large state with a wealth of energy resources. It leads the nation in energy production, providing more than one-fifth of the country's domestically produced energy. Texas also has abundant renewable energy resources and is first in the nation in wind-generated electricity. With a significant number of sunny days across vast distances, Texas is also among the leading states in solar energy potential.

Residential sector and commercial sector energy consumption are driven by climate, and the Texas climate varies significantly from east to west. Warm, moist air from the Gulf of Mexico sweeps westward across the state, losing moisture as it goes. The result is a climate that ranges from humid to semi-arid and arid in various geographical areas. Frequent freezing temperatures occur in winter in the lightly populated high plains, and summer temperatures average above 90°F in the most densely populated parts of Texas where energy use for cooling is high. Even so, the residential sector accounts for just one-eighth of state end-use energy consumption. In part because of the state's large population, Texas leads the nation in total residential energy use, but it ranks near the lowest one-fifth of states in per capita residential energy consumption.

With a background of it, we have considered the daily power consumption of a standard single household in Houston, Texas. With continuous climate change resulting in increased use of electronic appliances, Texas has seen multiple power outages in the recent past. Estimation of future power usage can give a leverage to power generation companies by helping them to plan the production by addressing the issue of power grid failure due to high demand.

3. MAIN CHAPTER

3.1. Define Goal

The goal of our project is to analyze the Residential power usage for a house from June 2016 to July 2020 and forecast for the future 12 weeks. The main objective is to create a model which will analyze historical data based on seasonality and will forecast weekly power usage values to help the power generating companies to take appropriate measures to overcome the shortage.

In our analysis, we have used several models and the model with the best accuracy will be considered as the model of choice. The result of our project will be a forecast which can be used to estimate the power demand of a residential area. We have used the R programming language to develop, analyze and forecast various models.

3.2. Get data

For this project report, we have taken the time series data from Kaggle.com. Here, we have combined two data sets and created a daily PowerUsage_Combined.csv which contains daily residential power usage and daily weather report (Average for Temperature, Humidity, Pressure, Wind and Dew). Through Predictability Tests, we concluded that the daily dataset is predictable. But, by applying regression, we cannot create a model as we were getting "NA" values in the output. Owing to that, we have aggregated the same data into weekly periods to make more accurate forecasting.

The data set which we combined in this project now contains a weekly dataset for the residential power usage for a house in Houston, Texas, USA. It contains weekly power usage in kwh starting from May 30, 2016, to July 05, 2020. i.e., 214 weeks combined for Power and weather parameters.

3.3. Explore and Visualize Series:

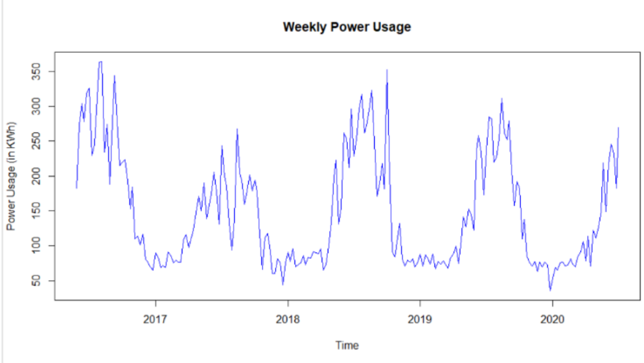

Figure No: 3.3.1

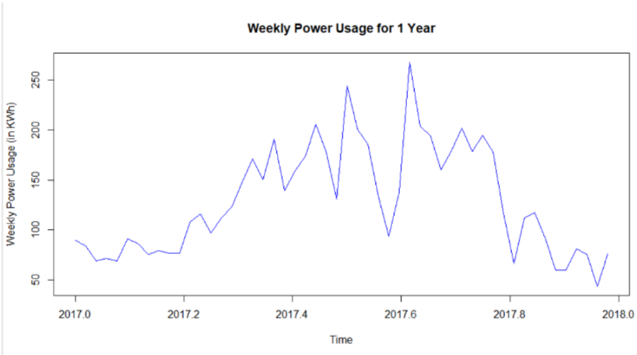

Figure No: 3.3.2

The data plots shown above represent the weekly power usage for the whole data set and for 1 year, respectively. We can see that we have got weekly seasonality for both the time series data set. The data plot for the weekly power usage for 1 year shows that the power usage is maximum during the months of May, June, July and much lower in the months from December through March.

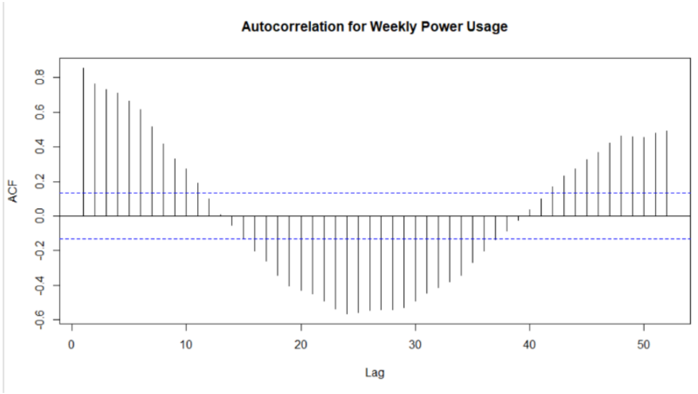

Figure No: 3.3.3

The correlogram for the power usage of the entire data set. We can see that the time series data set has very high positive correlation in initial lags that makes it statistically significant. High positive correlation in lag 1 shows trend and high positive correlation in lag 52 represents weekly seasonality. Based on the time series component chart below, we can conclude that the time series data set has initial weekly seasonality and making trend not clear.

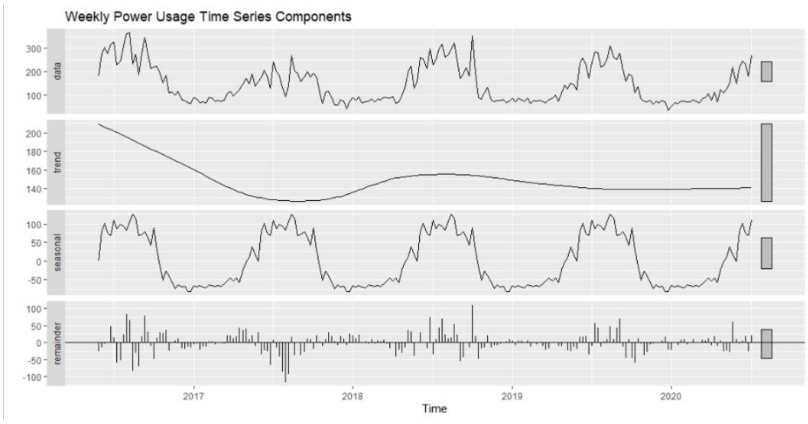

Figure No: 3.3.4

3.3.1. Check Predictability:

The data predictability evaluation is done using the below mentioned approaches:

Approach 1: Hypothesis testing of the autoregressive coefficient in AR(1) model

We applied the AR (1) model to the time series data set value.ts and analyzed the summary.

```
Series: value.ts
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1      mean
      0.8613  153.8810
s.e.  0.0344   19.7106

sigma^2 estimated as 1703:  log likelihood=-1099.41
AIC=2204.82   AICc=2204.93   BIC=2214.92

Training set error measures:
                     ME    RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -0.1798367 41.0718 30.10448 -7.653874 21.37732 0.7843097 -0.0937467
```

Figure No: 3.3.5

On analyzing the summary of the AR(1) model, it can be observed that the beta value is 0.8613, which is close to 1(H_0: β =1). Upon calculation, we get a p-value (2.765483e-05) which is very much less than 0.05, so it can be concluded that the null hypothesis can be rejected and data is predictable.

Approach 2: Autocorrelation coefficient for differenced data

In this second approach to check the predictability, we used diff() function and created a difference data of the time series data set value.ts. Further, we applied the result of the difference in the acf() function with lag.max=52 to check the significance. The correlogram of the time series data set is presented below, which indicates that the data lags 1,2,6,19, and 48 are outside the threshold i.e above upper and below lower significance levels. So there are some statistically significant relationships in the series which shows that the data is predictable.

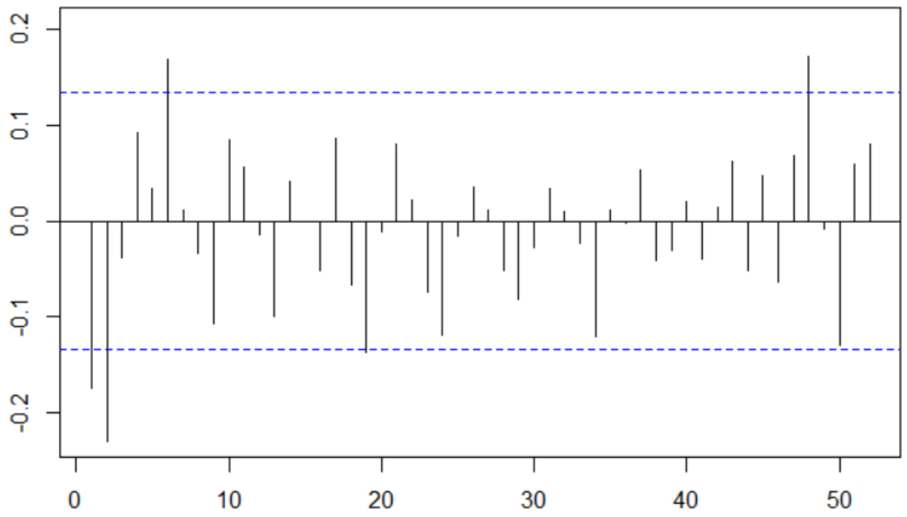

Figure No: 3.3.6

The above two approaches confirmed that the weekly dataset is predictable and hence we continued to prepare the models.

3.4. Data Preprocessing

The initial dataset contains the daily data of residential power usage for a house in Houston, Texas along with daily average value of weather parameters such as temperature, pressure, wind speed, humidity and dew in the "PowerUsage_combined.csv" file. This file is read from the local work directory and stored as "project.data" for further Usage.

3.4.1. Applying Regression on Daily Data

We have created a regression model with quadratic trend and seasonality using the daily power usage data. But, after applying regression we found multiple values as 'NA' in the output as shown in the below figure.

```
Residuals:
ALL 1035 residuals are 0: no residual degrees of freedom!

Coefficients: (2 not defined because of singularities)
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.194e+01         NA      NA       NA
trend       -3.651e-02         NA      NA       NA
I(trend^2)   2.572e-05         NA      NA       NA
season1.5    7.170e+00         NA      NA       NA
season1.75  -3.079e+00         NA      NA       NA
season2.25   1.818e+00         NA      NA       NA
season2.5    4.684e+00         NA      NA       NA
season2.75  -4.227e+00         NA      NA       NA
season3.25   3.027e+00         NA      NA       NA
season3.5    4.012e+00         NA      NA       NA
```

Figure No: 3.4.1

For the ease of analysis, we aggregated the daily data into weekly data in order to be able to create forecasting models.

3.4.2. Adding rows and Converting data into weekly

For aggregating the daily data into weekly we used the following codes.

```
# Aggregate daily data to get weekly power usage
Week <- cut(as.Date(final.data$Date), "week")
project <- aggregate(Value..kWh. ~ Week, final.data, sum)
str(project$Week)

# Aggregate daily data to get weekly average weather parameters
project.temp <- aggregate(Temp_avg ~ Week, final.data, mean)
str(project.temp)
project.hum <- aggregate(Hum_avg ~ Week, final.data, mean)
project.wind <- aggregate(Wind_avg ~ Week, final.data, mean)
project.press <- aggregate(Press_avg ~ Week, final.data, mean)
project.dew <- aggregate(Dew_avg ~ Week, final.data, mean)
```

Figure No: 3.4.2

Note: We have also added 2 rows for the dates "2016-05-30" and "2016-05-31" as well as we have deleted the last two rows "2020-07-06" and "2020-07-07" to avoid a discrepancy of dataset ending in the middle of the week as below:

```r
# Add 2 rows data in the starting to complete the week
previous.records <- data.frame(c("2016-05-30","2016-05-31"), c(30,31), c(79,77.5), c(70.85,72.5),
                               c(73.6,78.7),c(7.4,7),c(29.8,29.8), c(28.027,28.5),c("weekday","weekday"))

names(previous.records) <- c("Date","Day","Temp_avg","Dew_avg","Hum_avg", "Wind_avg", "Press_avg","Value..kWh.","notes")
new.project.data <- rbind(previous.records, project.data)

nrow(new.project.data)
head(new.project.data)
tail(new.project.data)

# Remove 2 rows data at the end to complete the week
final.data  = new.project.data[-c(1499, 1500),]
tail(final.data)
```

Figure No: 3.4.3

```
> head(new.project.data)
        Date Day Temp_avg Dew_avg Hum_avg Wind_avg Press_avg Value..kWh.   notes
1 2016-05-30  30     79.0   70.85    73.6      7.4      29.8      28.027 weekday
2 2016-05-31  31     77.5   72.50    78.7      7.0      29.8      28.500 weekday
3 2016-06-01   1     74.8   71.40    89.4      9.5      29.8      29.691 weekday
4 2016-06-02   2     71.2   70.30    96.8      7.8      29.8      28.789 weekend
5 2016-06-03   3     72.1   70.00    93.6      4.7      29.8      19.247 weekend
6 2016-06-04   4     71.2   70.00    96.1      7.0      29.7      22.883 weekday
>
```

Figure No: 3.4.4

3.4.3. Combining all the variables into a single dataframe

We have also combined all the variables into a dataset for future use.

```r
# Combining all the variables to create a single dataframe
project1 <- data.frame(Temp_avg = c(project.temp$Temp_avg),
                                   Hum_avg = c(project.hum$Hum_avg),
                                   Dew_avg = c(project.dew$Dew_avg),
                                   Press_avg = c(project.press$Press_avg),
                                   Wind_avg = c(project.wind$Wind_avg))
project2 <- cbind(project,project1)
```

Figure No: 3.4.5

3.4.4. Creating 2 time series dataframe

We created multiple time series data for all the individual variables with below dynamics:

Sr. No.	Variable	Time Series Data	Total Observations	Time Period
1	Power	value.ts	214	Week 22 of 2016 to Week 27 of 2020
2	Temperature	temp.ts		
3	Pressure	press.ts		
4	Humidity	hum.ts		
5	Dew	dew.ts		
6	Wind	wind.ts		

Table 1: Time Series Data frame

3.5. Partition Time Series

We created a data partition of 162 records (75% of the dataset) for the training period and 52 records (25% of the dataset) for the validation period. Training data are from 22nd week of 2016 to 27th week of 2019 and Validation data are from 28th week of 2019 to 27th week of 2020. These partitioned training and validation data sets are shown in figure.

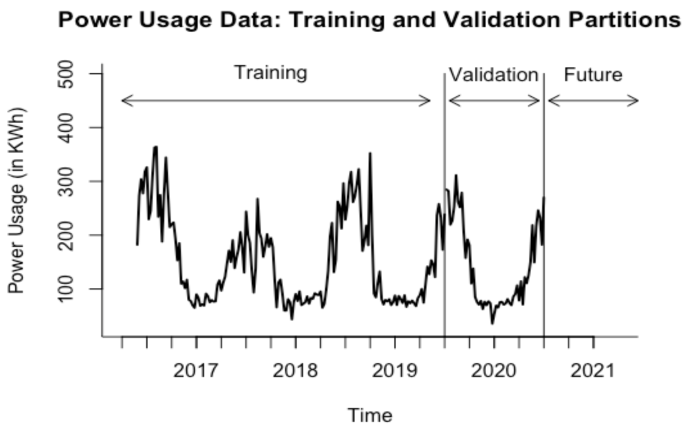

Figure No: 3.5.1

3.6. Apply Forecast Methods

We have applied following models to identify the best fit:

3.6.1. Regression Models

We used Regression-based Models for predicting the weekly power usage. They are easy to use as they can be used with a time series data set that contains both trend and seasonality. Earlier, we applied different types of regression models and among them the best model of choice we have got is the Regression Model with linear trend and seasonality. Here, we have implemented this model on the training/validation data set and on the entire data set.

a. Regression with linear trend:

The following is the summary for the above-mentioned model:

```
> summary(train.lintrend)

Call:
tslm(formula = value.train.ts ~ trend)

Residuals:
    Min      1Q  Median      3Q     Max 
-108.58  -60.47  -23.93   46.05  217.31 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 186.6345    12.5036  14.926  < 2e-16 ***
trend        -0.4198     0.1331  -3.155  0.00192 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 79.2 on 160 degrees of freedom
Multiple R-squared:  0.05856,	Adjusted R-squared:  0.05267 
F-statistic: 9.952 on 1 and 160 DF,  p-value: 0.00192
```

Figure No: 3.6.1

Equation for the model is:

$$\mathbf{y_t = 186.6345 - 0.4198\ t}$$

b. Regression with Quadratic trend:

The following is the summary for the above-mentioned model:

```
> summary(train.quadtrend)

Call:
tslm(formula = value.train.ts ~ trend + I(trend^2))

Residuals:
    Min      1Q  Median      3Q     Max 
-112.72  -57.62  -19.79   55.93  222.61 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 237.381138  18.185539  13.053  < 2e-16 ***
trend        -2.276372   0.515113  -4.419 1.83e-05 ***
I(trend^2)    0.011390   0.003061   3.721 0.000275 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 76.2 on 159 degrees of freedom
Multiple R-squared:  0.134,	Adjusted R-squared:  0.1231 
F-statistic:  12.3 on 2 and 159 DF,  p-value: 1.081e-05
```

Figure No: 3.6.2

Equation for the model is:

$$\mathbf{y_t = 237.381138 - 2.276372\ t + 0.011390\ t^2}$$

c. Regression with seasonality:

The following is the summary for the above-mentioned model:

```
> summary(train.season)

Call:
tslm(formula = value.train.ts ~ season)

Residuals:
     Min       1Q   Median       3Q      Max 
-145.788  -15.508    0.274   11.607  123.545 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  89.4327    26.2731   3.404 0.000928 ***
season2     -12.1473    37.1558  -0.327 0.744342    
season3      -5.6733    37.1558  -0.153 0.878922    
season4     -15.1800    37.1558  -0.409 0.683664    
season5     -17.4700    37.1558  -0.470 0.639156    
season6      -4.4360    37.1558  -0.119 0.905185    
season7      -9.5093    37.1558  -0.256 0.798482    
season8     -14.3463    37.1558  -0.386 0.700159    
season9     -10.6687    37.1558  -0.287 0.774550    
season10    -10.4336    37.1558  -0.281 0.779385    
season11     -8.5533    37.1558  -0.230 0.818363    
season12     -0.7715    37.1558  -0.021 0.983473    
season13      6.3890    37.1558   0.172 0.863792    
season14      4.2093    37.1558   0.113 0.910008    
season15      2.9013    37.1558   0.078 0.937902    
season16      0.6357    37.1558   0.017 0.986381    
season17     29.4003    37.1558   0.791 0.430488    
season18     59.3980    37.1558   1.599 0.112774    
season19     69.2477    37.1558   1.864 0.065029 .  
season20     99.4657    37.1558   2.677 0.008566 ** 
season21     49.2120    37.1558   1.324 0.188090    
season22     64.9173    34.7561   1.868 0.064452 .  
season23    147.2006    34.7561   4.235 4.76e-05 ***
season24    165.9181    34.7561   4.774 5.61e-06 ***
season25    135.8098    34.7561   3.908 0.000161 ***
season26    140.1188    34.7561   4.031 0.000102 ***
season27    169.8771    34.7561   4.888 3.50e-06 ***
season28    138.2497    37.1558   3.721 0.000315 ***
season29    153.1460    37.1558   4.122 7.32e-05 ***
season30    164.7993    37.1558   4.435 2.19e-05 ***
season31    149.8828    37.1558   4.034 0.000102 ***
season32    168.6957    37.1558   4.540 1.44e-05 ***
season33    175.9040    37.1558   4.734 6.59e-06 ***
season34    177.5740    37.1558   4.779 5.48e-06 ***
season35    120.6757    37.1558   3.248 0.001542 ** 
season36    113.1190    37.1558   3.044 0.002917 ** 
season37    148.2298    37.1558   3.989 0.000120 ***
season38    143.5357    37.1558   3.863 0.000190 ***
season39    102.2581    37.1558   2.752 0.006927 ** 
season40    166.2914    37.1558   4.476 1.87e-05 ***
season41    111.4283    37.1558   2.999 0.003351 ** 
season42     44.4893    37.1558   1.197 0.233735    
season43     11.8649    37.1558   0.319 0.750084    
season44     46.9103    37.1558   1.263 0.209428    
season45     30.4657    37.1558   0.820 0.414021    
season46      6.2744    37.1558   0.169 0.866210    
season47    -11.7510    37.1558  -0.316 0.752402    
season48     -3.8413    37.1558  -0.103 0.917846    
season49    -10.0947    37.1558  -0.272 0.786374    
season50    -11.3747    37.1558  -0.306 0.760081    
season51    -28.2157    37.1558  -0.759 0.449245    
season52    -17.2290    37.1558  -0.464 0.643782    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 45.51 on 110 degrees of freedom
Multiple R-squared:  0.7863,	Adjusted R-squared:  0.6873 
F-statistic: 7.938 on 51 and 110 DF,  p-value: < 2.2e-16
```

Figure No: 3.6.3

Equation for the model is:

$$\mathbf{y_t = 89.4327 - 12.1473D_2 - 5.6733\ D_3 - 15.1800\ D_4 \ldots\ldots\ldots - 17.2290\ D_{52}}$$

d. Regression with Linear trend + seasonality:

The following is the summary for the above-mentioned model:

```
> summary(train.lintrend.season)

Call:
tslm(formula = value.train.ts ~ trend + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-145.787  -15.899   -0.384   13.441  113.352 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 105.89697   26.45875   4.002 0.000115 ***
trend        -0.19600    0.07735  -2.534 0.012695 *  
season2     -11.95133   36.27274  -0.329 0.742421    
season3      -5.28133   36.27299  -0.146 0.884507    
season4     -14.59199   36.27340  -0.402 0.688267    
season5     -16.68599   36.27398  -0.460 0.646433    
season6      -3.45598   36.27472  -0.095 0.924273    
season7      -8.33331   36.27563  -0.230 0.818738    
season8     -12.97431   36.27670  -0.358 0.721298    
season9      -9.10064   36.27794  -0.251 0.802396    
season10     -8.66960   36.27934  -0.239 0.811579    
season11     -6.59330   36.28090  -0.182 0.856133    
season12      1.38459   36.28264   0.038 0.969629    
season13      8.74104   36.28453   0.241 0.810083    
season14      6.75738   36.28659   0.186 0.852616    
season15      5.64538   36.28882   0.156 0.876661    
season16      3.57572   36.29121   0.099 0.921694    
season17     32.53639   36.29376   0.896 0.371976    
season18     62.73006   36.29648   1.728 0.086772 .  
season19     72.77573   36.29937   2.005 0.047455 *  
season20    103.18974   36.30242   2.843 0.005345 ** 
season21     53.13207   36.30563   1.463 0.146218    
season22     63.93732   33.93217   1.884 0.062193 .  
season23    146.41657   33.93138   4.315 3.52e-05 ***
season24    165.33007   33.93076   4.873 3.76e-06 ***
season25    135.41783   33.93032   3.991 0.000120 ***
season26    139.92283   33.93005   4.124 7.30e-05 ***
season27    169.87708   33.92997   5.007 2.14e-06 ***
season28    133.34958   36.32416   3.671 0.000376 ***
season29    148.44191   36.32013   4.087 8.38e-05 ***
season30    160.29125   36.31626   4.414 2.40e-05 ***
season31    145.57075   36.31255   4.009 0.000112 ***
season32    164.57959   36.30901   4.533 1.50e-05 ***
season33    171.98393   36.30563   4.737 6.57e-06 ***
season34    173.84993   36.30242   4.789 5.32e-06 ***
season35    117.14760   36.29937   3.227 0.001651 ** 
season36    109.78694   36.29648   3.025 0.003104 ** 
season37    145.09371   36.29376   3.998 0.000117 ***
season38    140.59561   36.29121   3.874 0.000183 ***
season39     99.51408   36.28882   2.742 0.007134 ** 
season40    163.74332   36.28659   4.513 1.63e-05 ***
season41    109.07629   36.28453   3.006 0.003284 ** 
season42     42.33329   36.28264   1.167 0.245852    
season43      9.90481   36.28090   0.273 0.785367    
season44     45.14630   36.27934   1.244 0.216019    
season45     28.89764   36.27794   0.797 0.427437    
season46      4.90241   36.27670   0.135 0.892751    
season47    -12.92702   36.27563  -0.356 0.722263    
season48     -4.82135   36.27472  -0.133 0.894508    
season49    -10.87868   36.27398  -0.300 0.764822    
season50    -11.96268   36.27340  -0.330 0.742190    
season51    -28.60767   36.27299  -0.789 0.432013    
season52    -17.42500   36.27274  -0.480 0.631914    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 44.42 on 109 degrees of freedom
Multiple R-squared:  0.7982,	Adjusted R-squared:  0.702 
F-statistic: 8.293 on 52 and 109 DF,  p-value: < 2.2e-16

> |
```

Figure No: 3.6.4

Equation for the model is:

$$\mathbf{y_t = 105.89697 - 0.19600t - 11.95133\ D_2 - 5.28133\ D_3 - 14.59199\ D_4 \ldots\ldots\ldots - 17.42500\ D_{52}}$$

e. Regression with Quadratic trend + seasonality:

The following is the summary for the above-mentioned model:

```
> summary(train.quadtrend.season)

Call:
tslm(formula = value.train.ts ~ trend + I(trend^2) + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-134.348  -15.493   -0.068   13.914  107.993 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 136.567465  26.503476   5.153 1.17e-06 ***
trend        -1.230340   0.295555  -4.163 6.34e-05 ***
I(trend^2)    0.006346   0.001756   3.613 0.000461 ***
season2     -11.989403  34.419693  -0.348 0.728271    
season3      -5.370165  34.419935  -0.156 0.876309    
season4     -14.744284  34.420343  -0.428 0.669242    
season5     -16.914428  34.420924  -0.491 0.624141    
season6      -3.773263  34.421682  -0.110 0.912915    
season7      -8.752122  34.422626  -0.254 0.799782    
season8     -13.507340  34.423764  -0.392 0.695548    
season9      -9.760582  34.425106  -0.284 0.777313    
season10     -9.469149  34.426663  -0.275 0.783801    
season11     -7.545140  34.428446  -0.219 0.826944    
season12      0.267761  34.430468   0.008 0.993809    
season13      7.446537  34.432744   0.216 0.829191    
season14      5.272506  34.435288   0.153 0.878595    
season15      3.957449  34.438117   0.115 0.908726    
season16      1.672035  34.441247   0.049 0.961370    
season17     30.404263  34.444696   0.883 0.379360    
season18     60.356800  34.448485   1.752 0.082597 .  
season19     70.148646  34.452633   2.036 0.044189 *  
season20    100.296133  34.457161   2.911 0.004380 ** 
season21     49.959263  34.462092   1.450 0.150043    
season22     53.928158  32.317661   1.669 0.098075 .  
season23    136.432795  32.316309   4.222 5.07e-05 ***
season24    155.358989  32.315426   4.808 4.97e-06 ***
season25    125.446743  32.315009   3.882 0.000179 ***
season26    129.939055  32.315059   4.021 0.000108 ***
season27    159.867927  32.315577   4.947 2.79e-06 ***
season28    130.176766  34.479673   3.775 0.000262 ***
season29    145.548310  34.473963   4.222 5.07e-05 ***
season30    157.664163  34.468655   4.574 1.28e-05 ***
season31    143.197492  34.463728   4.155 6.53e-05 ***
season32    162.447462  34.459160   4.714 7.28e-06 ***
season33    170.080242  34.454930   4.936 2.91e-06 ***
season34    172.161996  34.451019   4.997 2.26e-06 ***
season35    115.662726  34.447410   3.358 0.001086 ** 
season36    108.492432  34.444085   3.150 0.002114 ** 
season37    143.976880  34.441028   4.180 5.93e-05 ***
season38    139.643769  34.438224   4.055 9.49e-05 ***
season39     98.714534  34.435659   2.867 0.004988 ** 
season40    163.083375  34.433320   4.736 6.66e-06 ***
season41    108.543258  34.431196   3.152 0.002096 ** 
season42     41.914482  34.429276   1.217 0.226102    
season43      9.587533  34.427550   0.278 0.781173    
season44     44.917858  34.426009   1.305 0.194746    
season45     28.745343  34.424647   0.835 0.405549    
season46      4.813569  34.423456   0.140 0.889052    
season47    -12.965096  34.422432  -0.377 0.707175    
season48     -4.821352  34.421570  -0.140 0.888867    
season49    -10.853299  34.420866  -0.315 0.753134    
season50    -11.924604  34.420319  -0.346 0.729685    
season51    -28.569600  34.419928  -0.830 0.408352    
season52    -17.399621  34.419692  -0.506 0.614228    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42.16 on 108 degrees of freedom
Multiple R-squared:   0.82,	Adjusted R-squared:  0.7316 
F-statistic: 9.282 on 53 and 108 DF,  p-value: < 2.2e-16

>
```

Figure No: 3.6.5

Equation for the model is:

$$y_t = 136.567465 - 1.230340t + 0.003646t^2 - 11.989403\,D_2 - 5.370165\,D_3 - 14.744284\,D_4 \,\ldots\ldots\ldots\, -17.399621\,D_{52}$$

Measuring and comparing Accuracy for all the Regression models:

```
> # Compare the Models based on Accuracy measures
> round(accuracy(train.lintrend.pred$mean, value.valid.ts),3)
              ME   RMSE    MAE   MPE   MAPE  ACF1 Theil's U
Test set 33.683 83.987 65.653 -2.55 46.141 0.835     2.062
> round(accuracy(train.quadtrend.pred$mean, value.valid.ts),3)
               ME   RMSE     MAE      MPE    MAPE  ACF1 Theil's U
Test set -74.379 116.62 106.159 -107.687 119.628 0.874     5.895
> round(accuracy(train.season.pred$mean, value.valid.ts),3)
              ME   RMSE    MAE     MPE   MAPE  ACF1 Theil's U
Test set -8.376 34.359 24.853 -12.951 21.023 0.171     1.096
> round(accuracy(train.lintrend.season.pred$mean, value.valid.ts),3)
              ME   RMSE   MAE   MPE   MAPE  ACF1 Theil's U
Test set 12.596 35.624 26.98 6.861 20.588 0.177     1.008
> round(accuracy(train.quadtrend.season.pred$mean, value.valid.ts),3)
               ME RMSE    MAE     MPE   MAPE  ACF1 Theil's U
Test set -47.899 61.4 56.011 -52.794 55.727 0.354     2.558
> 
```

Figure No: 3.6.6

We have measured accuracies for all the above models to identify the best models to be used for prediction. Below table represents comparison between the RMSE and MAPE values for all the 5 models:

Sr. No.	Models	RMSE	MAPE
1	Regression with linear trend	83.987	46.141
2	Regression with Quadratic trend	116.62	119.628
3	Regression with seasonality	34.359	21.023
4	Regression with Linear trend + seasonality	35.624	20.588
5	Regression with Quadratic trend + seasonality	61.4	55.727

Table 2: Accuracy Measures for Different Regression Models

- From the above accuracy measures, it can be noted that the RMSE value of Regression with Linear Trend and Seasonality is higher than that of Regression with seasonality, but it's MAPE value is the lowest. Thus, considering MAPE as a superior accuracy measure than RMSE, we have selected a Regression model with linear trend and seasonality.

- Further, we have used "Arima Model " and "Two-Level Model (Regression with Linear trend + seasonality model along with Different smoothing models and autoregressive models for residuals)" in search of the best forecasting model.

3.6.2. Two-level Models

From the earlier analysis with different types of regression model, we have found the most accurate result with a regression model with linear trend and seasonality. However, from the correlogram drawn on residuals of the regression model, we can see some statistically significant relationship in some of the lags which are beyond the upper and lower threshold limit of the Acf values.

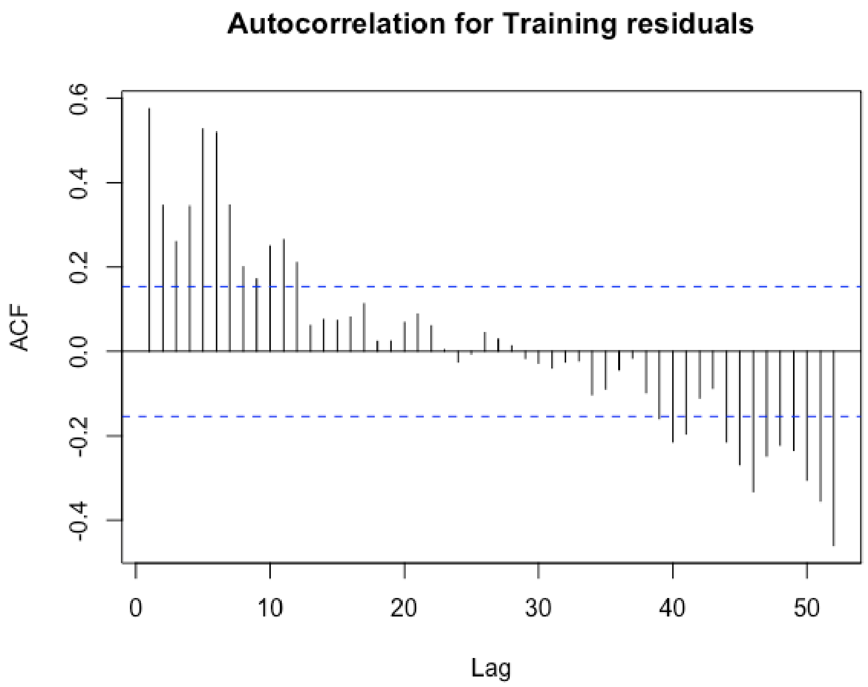

Figure No: 3.6.7

That points to the conclusion that though the regression model has incorporated most of the trend and seasonality components of the historical data, there are still some level components and systematic error are left. Therefore, we are using moving average , simple exponential smoothing and autoregressive models as our second level model, respectively, for the regression residuals, in order to find a better model to forecast.

a. Two level model Regression with linear trend and seasonality + SES model

We have applied a simple exponential smoothing model on the regression residuals as the residuals are free of any trend or seasonality. Created simple exponential smoothing (SES) Using ets() function with model = "ZNN", i.e., with automated additive error(Z), no trend (N), no Seasonality and optimal values for smoothing parameters.

```
> summary(ses)
ETS(A,N,N)

Call:
 ets(y = train.lintrend.season.pred$residuals, model = "ZNN")

  Smoothing parameters:
    alpha = 0.2339

  Initial states:
    l = 29.2195

  sigma:  30.7827

     AIC     AICc      BIC
1938.511 1938.663 1947.774

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
Training set -0.871406 30.59212 21.08011 -317.1216 604.8407 0.4681966 0.2083278
```

Figure No: 3.6.8

However, even after incorporating residuals into the model, we still have the autocorrelations at different lags as shown in the below autocorrelation plot.

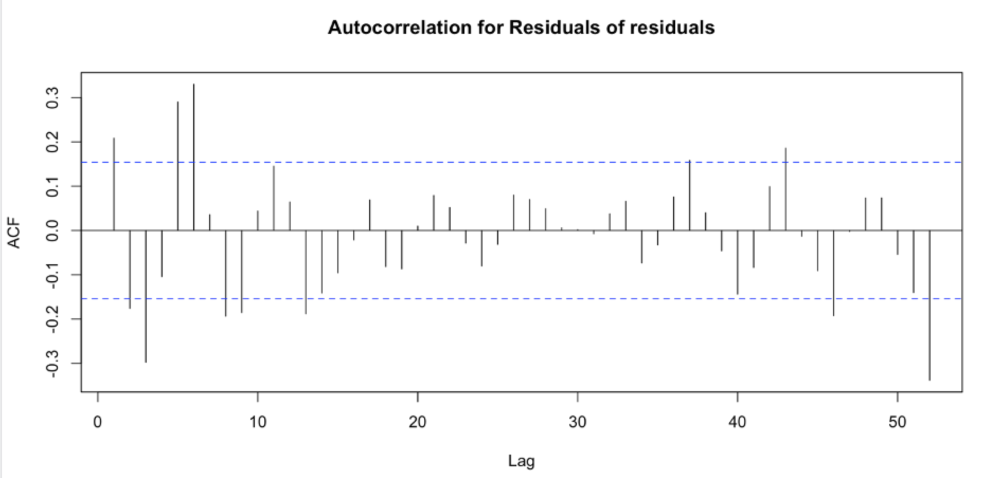

Figure No: 3.6.9

b. Two level model Regression with linear trend and seasonality + trailing MA(1)

Here We have applied a trailing MA model on the regression residuals aiming to handle the remainder components in the dataset.

As there are significant weekly seasonality present in the dataset, the value of smoothing parameter has been taken as, k = 1. The trailing MA forecast (window width of 1) for the regression residuals in the validation period has been calculated using forecast function.

After getting a forecast of the residuals of the regression model in the validation period, the final forecast has been calculated by taking the sum of regression forecast and residual forecast. The accuracy measures for the training dataset are coming as below.

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
Test set	10.855	37.081	27.418	4.005	21.547	0.312	1.181

Figure No: 3.6.10

However, even after incorporating the residuals into this model, we still see some autocorrelation left in residuals of residuals.

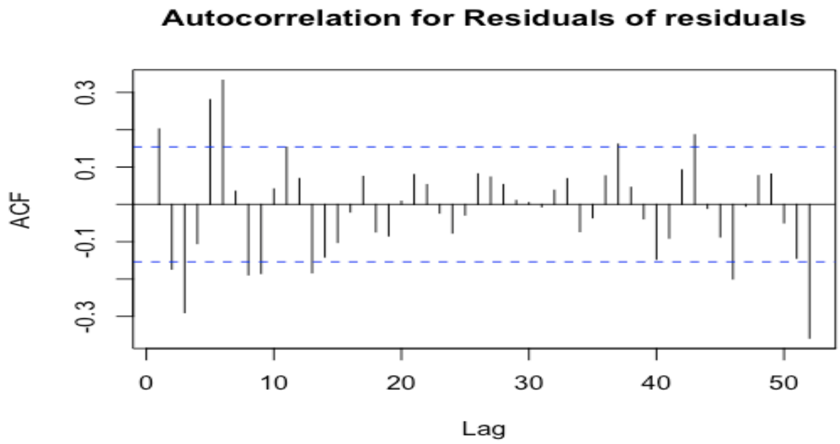

Figure No: 3.6.11

c. Two level model Regression with linear trend and seasonality + AR (1) model

Here we have applied an autoregressive model with order 1 to the residuals of the regression models with linear trend and seasonality. We selected order 1 as in the correlogram for the residuals of the regression model the coefficient of autocorrelation is maximum in lag 1.

Therefore, we created an AR(1) model for the residuals of the regression models by utilizing the function arima() with model parameters (1,0,0). Where p=1, order of autoregression, d=0, differencing and q=0, order of moving average.

Summary of the above model is as below:

```
Series: train.season.pred$residuals
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1    mean
      0.5751  0.0591
s.e.  0.0639  5.6159

sigma^2 estimated as 949.5:  log likelihood=-784.4
AIC=1574.79   AICc=1574.95   BIC=1584.06

Training set error measures:
                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
Training set -0.130517 30.62383 21.07229 25.67094 263.1486 0.4771468 -0.0133655
>
```

Figure No: 3.6.12

The equation for the Ar(1) model for residuals is as follows:

et=0.0591+0.5751et-1

Further on the final forecast for this two level model is produced by joining the forecast of the regression model and the residual forecast of the second level AR(1) model. However, even after incorporating the residuals into this model, we still see some autocorrelation left in residuals of residuals.

Comparison of Performance by Accuracy Measure for different two-level models:

```
> # To forecast on validation data set
> forecast_param1 <- data.frame(trend = c(163:214), temp.train.ts = temp.valid.ts[1:52],
+                               dew.train.ts = dew.valid.ts[1:52])
> lin.season.external2.pred <- forecast(lin.season.external_2, newdata = forecast_param1, level = 0)
> round(accuracy(lintrend.season.twolevel.pred, value.valid.ts),3)
             ME   RMSE    MAE   MPE   MAPE ACF1 Theil's U
Test set 12.672 35.796 27.047 6.857 20.596 0.18     1.008
> round(accuracy(twolvl.reg.ma , value.valid.ts),3)
             ME   RMSE    MAE   MPE   MAPE  ACF1 Theil's U
Test set 12.547 36.054 27.012 5.604 19.999 0.205     0.976
> round(accuracy(twolvl.reg.ses, value.valid.ts),3)
             ME   RMSE    MAE    MPE   MAPE  ACF1 Theil's U
Test set 16.389 37.135 28.877 10.494 22.276 0.177     1.072
> |
```

Figure No: 3.6.13

Below table represents comparison between the RMSE and MAPE values for the above three models to identify the best models to be used for prediction:

Sr. No.	Two level models	RMSE	MAPE
1	Regression with Linear trend and seasonality +AR(1)for regression residuals	35.796	20.596
2	Regression with Linear trend and seasonality +MA(1)for regression residuals	36.054	19.999
3	Regression with Linear trend and seasonality +SES for regression residuals	37.135	22.276

Table 3: Accuracy Measures for Different Two-level Models

From the above accuracy measures, it can be noted that the RMSE value of the two level model - Regression with Linear Trend and Seasonality +MA (1) for regression residuals is higher than that of Two level model - Regression with Linear Trend and seasonality +AR (1) for regression residuals, but it's MAPE value is the lowest. Thus, considering MAPE as a superior accuracy measure than RMSE, we have selected two level models -Regression with Linear Trend and Seasonality +MA (1) for regression residuals.

Further, we have also explored the possibility of External factor interference in our model by utilizing the Multi Variable Regression model (Regression with Linear Trend and Seasonality). As we saw above, there is still some Autocorrelation even after applying various models and fitting. There are high chances of data being affected by external variables/factors (Temperature, Dew, pressure, Humidity and Wind). Thus, we have verified the effect of External Variable as below:

3.6.3. Regression Models with External Variables

We have used corrplot () function in order to identify the correlation between various independent variables affecting Power Usage. We have used below code to generate the highlighted map:

```
project.cor <- subset(project2, select = -c(1))
head(project.cor)
corrplot(cor(project.cor),        # Correlation matrix
         method = "shade", # Correlation plot method
         type = "full",    # Correlation plot style (also "upper" and "lower")
         diag = TRUE,      # If TRUE (default), adds the diagonal
         tl.col = "black", # Labels color
         bg = "white",     # Background color
         title = "",       # Main title
         col = NULL)       # Color palette
```

Figure No: 3.6.14

The below highlighted map is the output for our analysis. By this we can clearly see that factors like Dew and Temperature are affecting the value of Power Usage the most.

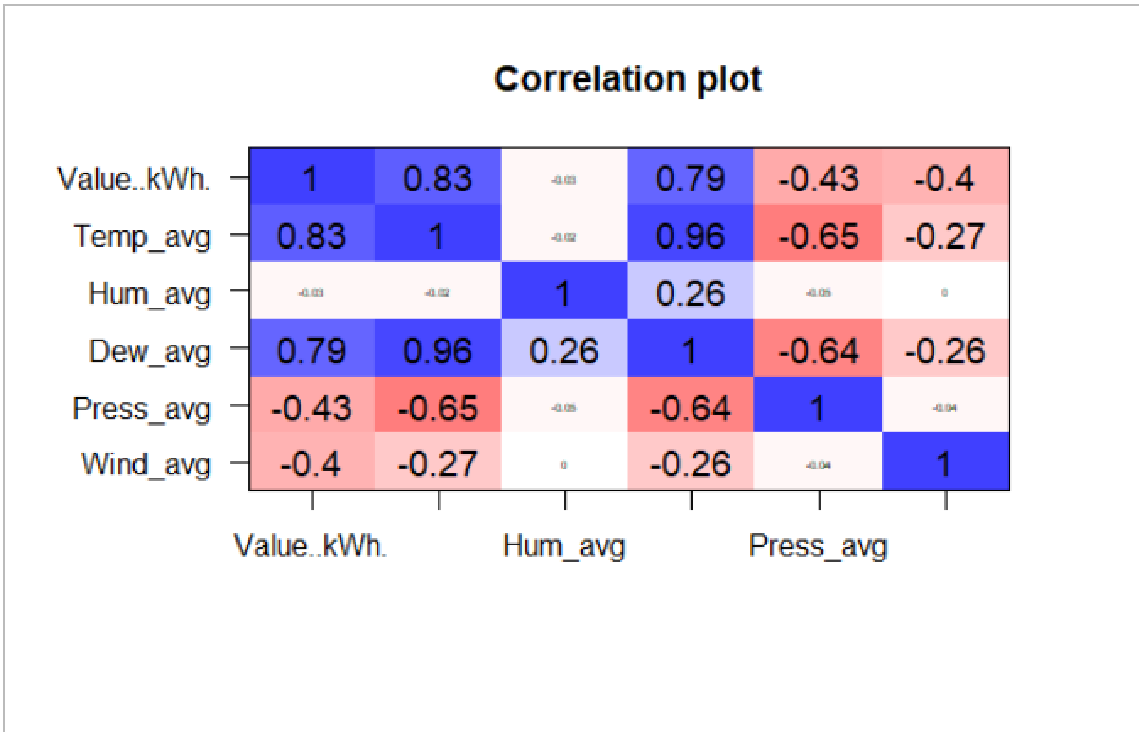

Figure No: 3.6.15

Regression with Linear trend + seasonality with 2 external variables:

There are different types of regression-based models, and its usage depends on the time series components present in the dataset. Regression based models are used for time series data sets that contain both trend and seasonality. Considering the above accuracy measures here from 'Table No 2', we are implementing Regression with Linear Trend and Seasonality along with other external variables. The below is the Code and summary for the above-mentioned model:

```
> summary(lin.season.external_2)

Call:
tslm(formula = value.train.ts ~ trend + season + temp.train.ts + 
    dew.train.ts)

Residuals:
     Min       1Q   Median       3Q      Max 
-139.196  -17.838   -1.488   19.217  107.313 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)  
(Intercept)   -13.64157   59.02374  -0.231   0.8177  
trend          -0.13124    0.08487  -1.546   0.1250  
season2       -38.41087   37.83810  -1.015   0.3123  
season3       -21.02587   36.82060  -0.571   0.5692  
season4       -35.44689   37.13690  -0.954   0.3420  
season5       -43.42457   37.67879  -1.152   0.2517  
season6       -37.71235   39.33406  -0.959   0.3398  
season7       -43.12652   39.52150  -1.091   0.2776  
season8       -50.06075   39.62973  -1.263   0.2093  
season9       -47.71613   40.00651  -1.193   0.2356  
season10      -49.41809   40.11971  -1.232   0.2207  
season11      -49.90649   40.60440  -1.229   0.2217  
season12      -50.73156   42.56715  -1.192   0.2360  
season13      -43.26663   42.87934  -1.009   0.3152  
season14      -41.87309   41.84592  -1.001   0.3193  
season15      -46.44417   42.78392  -1.086   0.2801  
season16      -49.42688   42.89152  -1.152   0.2517  
season17      -28.13500   44.80458  -0.628   0.5314  
season18       -2.63900   46.20428  -0.057   0.9546  
season19        3.88362   47.44289   0.082   0.9349  
season20       26.22196   49.78860   0.527   0.5995  
season21      -23.02227   49.87677  -0.462   0.6453  
season22      -10.69363   48.18894  -0.222   0.8248  
season23       67.03894   49.20392   1.362   0.1759  
season24       78.98469   51.29022   1.540   0.1265  
season25       52.91922   50.70804   1.044   0.2990  
season26       57.65946   50.43747   1.143   0.2555  
season27       82.90962   51.75020   1.602   0.1121  
season28       47.89739   52.81145   0.907   0.3665  
season29       58.07783   54.19961   1.072   0.2863  
season30       69.38195   54.18960   1.280   0.2032  
season31       59.60331   52.76361   1.130   0.2612  
season32       80.01656   52.81207   1.515   0.1327  
season33       86.43321   53.02008   1.630   0.1060  
season34       93.38243   51.71930   1.806   0.0738 .
season35       39.18842   50.72814   0.773   0.4415  

season36       35.53024   49.47407   0.718   0.4742  
season37       71.11317   49.63259   1.433   0.1548  
season38       63.17153   50.96931   1.239   0.2179  
season39       31.05968   47.69365   0.651   0.5163  
season40       92.07765   48.82191   1.886   0.0620 .
season41       40.97623   47.61108   0.861   0.3914  
season42       -7.59493   42.67758  -0.178   0.8591  
season43      -33.92293   40.73408  -0.833   0.4068  
season44      -10.06068   44.29944  -0.227   0.8208  
season45      -13.14885   41.01275  -0.321   0.7491  
season46      -26.79046   38.48810  -0.696   0.4879  
season47      -41.90180   38.23135  -1.096   0.2755  
season48      -37.16912   38.67109  -0.961   0.3386  
season49      -20.43677   36.21597  -0.564   0.5737  
season50      -31.98928   36.87600  -0.867   0.3876  
season51      -58.90935   38.37509  -1.535   0.1277  
season52      -39.28860   37.45351  -1.049   0.2965  
temp.train.ts   2.88455    1.65476   1.743   0.0842 .
dew.train.ts   -0.55804    1.32519  -0.421   0.6745  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 43.8 on 107 degrees of freedom
Multiple R-squared:  0.8074,    Adjusted R-squared:  0.7102 
F-statistic: 8.308 on 54 and 107 DF,  p-value: < 2.2e-16
```

Figure No: 3.6.16

- The regression model with linear trend and seasonality with external variables for training partition contains 1 period index(t) and 51 independent dummy variables for seasons. The dummy variable for season2 is D_2, season3 is D_3, season4 is D_4 and so on.
- The equation of the model is as follows:

$$\mathbf{Y_t = -13.64157 - 0.13124\ t - 38.41087\ D_2 - 21.02587\ D_3 - 35.44689\ D_4 - 43.42457\ D_5 - \ldots - 58.90935\ D_{51} - 39.28860\ D_{52} + 2.88455{*}temperature - 0.55804{*}Dew}$$

- The summary represents that the model has a high Multiple R-squared value of 0.8074 and high adjusted R-squared value of 0.7102, which indicates a good fit for the training data and shows that the model with linear trend and seasonality is statistically significant. It can be noted that the model is statistically significant since the F-statistic p-value is very low than 0.05 or 0.01 (2.2e-16). Also, not all the predictors in the model are significant since their p-values are greater than 0.05.
- After developing the Regression model with linear trend and seasonality along with external variables on training data partition, we predicted the point forecast on validation data set.
- Multivariable regression model is a complex regression model and utilizing it for future forecasting will be a tedious job as well as its MAPE (20.605) is almost equivalent to the MAPE of Regression model with linear trend and seasonality (Table 2.). Thus, we will prefer to go forward with a Regression model with linear trend and seasonality as it will be parsimonious and simple.

Note: We have also applied external variables on Two-level models, but still there was some autocorrelation seen. Hence, we can conclude that there might be some other factors (other than external factors included in the dataset) which are affecting the model performance (Please refer to R-code).

3.7. Evaluate and Compare Performance:

From above created several models, we selected few best fitting models based on accuracy metrics (MAPE and RMSE), which are described below:

<u>Model 1: Regression Model with Linear Trend and Seasonality</u>

Regression Model with Linear Trend and Seasonality on Training/Validation Data Set:

The summary of the regression model with linear trend and seasonality on training partition is shown below:

```
Call:
tslm(formula = value.train.ts ~ trend + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-145.787  -15.899   -0.384   13.441  113.352 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 105.89697   26.45875   4.002 0.000115 ***
trend        -0.19600    0.07735  -2.534 0.012695 *  
season2     -11.95133   36.27274  -0.329 0.742421    
season3      -5.28133   36.27299  -0.146 0.884507    
season4     -14.59199   36.27340  -0.402 0.688267    
season5     -16.68599   36.27398  -0.460 0.646433    
season6      -3.45598   36.27472  -0.095 0.924273    
season7      -8.33331   36.27563  -0.230 0.818738    
season8     -12.97431   36.27670  -0.358 0.721298    
season9      -9.10064   36.27794  -0.251 0.802396    
season10     -8.66960   36.27934  -0.239 0.811579    
season11     -6.59330   36.28090  -0.182 0.856133    
season12      1.38459   36.28264   0.038 0.969629    
season13      8.74104   36.28453   0.241 0.810083    
season14      6.75738   36.28659   0.186 0.852616    
season15      5.64538   36.28882   0.156 0.876661    
season16      3.57572   36.29121   0.099 0.921694    
season17     32.53639   36.29376   0.896 0.371976    
season18     62.73006   36.29648   1.728 0.086772 .  
season19     72.77573   36.29937   2.005 0.047455 *  
season20    103.18974   36.30242   2.843 0.005345 ** 
season21     53.13207   36.30563   1.463 0.146218    
season22     63.93732   33.93217   1.884 0.062193 .  
season23    146.41657   33.93138   4.315 3.52e-05 ***
season24    165.33007   33.93076   4.873 3.76e-06 ***
season25    135.41783   33.93032   3.991 0.000120 ***
season26    139.92283   33.93005   4.124 7.30e-05 ***
season27    169.87708   33.92997   5.007 2.14e-06 ***
season28    133.34958   36.32416   3.671 0.000376 ***
season29    148.44191   36.32013   4.087 8.38e-05 ***
season30    160.29125   36.31626   4.414 2.40e-05 ***
season31    145.57075   36.31255   4.009 0.000112 ***
season32    164.57959   36.30901   4.533 1.50e-05 ***
season33    171.98393   36.30563   4.737 6.57e-06 ***
season34    173.84993   36.30242   4.789 5.32e-06 ***
season35    117.14760   36.29937   3.227 0.001651 ** 
season36    109.78694   36.29648   3.025 0.003104 ** 
season37    145.09371   36.29376   3.998 0.000117 ***
season38    140.59561   36.29121   3.874 0.000183 ***
season39     99.51408   36.28882   2.742 0.007134 ** 
season40    163.74332   36.28659   4.513 1.63e-05 ***
season41    109.07629   36.28453   3.006 0.003284 ** 
season42     42.33329   36.28264   1.167 0.245852    
season43      9.90481   36.28090   0.273 0.785367    
season44     45.14630   36.27934   1.244 0.216019    
season45     28.89764   36.27794   0.797 0.427437    
season46      4.90241   36.27670   0.135 0.892751    
season47    -12.92702   36.27563  -0.356 0.722263    
season48     -4.82135   36.27472  -0.133 0.894508    
season49    -10.87868   36.27398  -0.300 0.764822    
season50    -11.96268   36.27340  -0.330 0.742190    
season51    -28.60767   36.27299  -0.789 0.432013    
season52    -17.42500   36.27274  -0.480 0.631914    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 44.42 on 109 degrees of freedom
Multiple R-squared:  0.7982,	Adjusted R-squared:  0.702 
F-statistic: 8.293 on 52 and 109 DF,  p-value: < 2.2e-16
```

Figure No: 3.7.1

- The regression model with linear trend and seasonality for training partition contains 1 period index(t) and 51 independent dummy variables for seasons. The dummy variable for season2 is D_2, season3 is D_3, season4 is D_4 and so on.
- The equation of the model is as follows:

$$\mathbf{Y_t = 105.897 - 0.196\, t - 11.95\, D_2 - 5.281\, D_3 - 14.592\, D_4 - 16.686\, D_5 - \ldots - 28.608\, D_{51} - 17.425\, D_{52}}$$

- The summary represents that the model has a high R-squared value of 0.7982 (79.82%) and high adjusted R-squared value of 0.702 (7.02%), which indicates a good fit for the training data and shows that the model with linear trend and seasonality is statistically significant. It can be noted that the model is statistically significant since the F-statistic p-value is very much lower than 0.05 or 0.01 (2.2e-16). Also, not all the predictors in the model are significant since their p-values are greater than 0.05.
- After developing the Regression model with linear trend and seasonality on training data partition, we predicted the point forecast on validation data set.

	Week	Start Date	Forecast
1	Week 27	2019-07-08	207.29796
2	Week 28	2019-07-15	222.19429
3	Week 29	2019-07-22	233.84762
4	Week 30	2019-07-29	218.93112
5	Week 31	2019-08-05	237.74396
6	Week 32	2019-08-12	244.95229
7	Week 33	2019-08-19	246.62229
8	Week 34	2019-08-26	189.72396
9	Week 35	2019-09-02	182.16729
10	Week 36	2019-09-09	217.27806
11	Week 37	2019-09-16	212.58396
12	Week 38	2019-09-23	171.30642
13	Week 39	2019-09-30	235.33966
14	Week 40	2019-10-07	180.47662
15	Week 41	2019-10-14	113.53762
16	Week 42	2019-10-21	80.91314
17	Week 43	2019-10-28	115.95862
18	Week 44	2019-11-04	99.51396
19	Week 45	2019-11-11	75.32272
20	Week 46	2019-11-18	57.29729
21	Week 47	2019-11-25	65.20696
22	Week 48	2019-12-02	58.95362
23	Week 49	2019-12-09	57.67362
24	Week 50	2019-12-16	40.83262
25	Week 51	2019-12-23	51.81929
26	Week 52	2019-12-30	69.04829
27	Week 1	2020-01-06	56.90096
28	Week 2	2020-01-13	63.37496
29	Week 3	2020-01-20	53.86829
30	Week 4	2020-01-27	51.57829
31	Week 5	2020-02-03	64.61229
32	Week 6	2020-02-10	59.53896
33	Week 7	2020-02-17	54.70196
34	Week 8	2020-02-24	58.37962
35	Week 9	2020-03-02	58.61466
36	Week 10	2020-03-09	60.49496
37	Week 11	2020-03-16	68.27684
38	Week 12	2020-03-23	75.43729
39	Week 13	2020-03-30	73.25762
40	Week 14	2020-04-06	71.94962
41	Week 15	2020-04-13	69.68396
42	Week 16	2020-04-20	98.44862
43	Week 17	2020-04-27	128.44629
44	Week 18	2020-05-04	138.29596
45	Week 19	2020-05-11	168.51396
46	Week 20	2020-05-18	118.26029
47	Week 21	2020-05-25	128.86953
48	Week 22	2020-06-01	211.15278
49	Week 23	2020-06-08	229.87028
50	Week 24	2020-06-15	199.76203
51	Week 25	2020-06-22	204.07103
52	Week 26	2020-06-29	233.82928

Figure No: 3.7.2

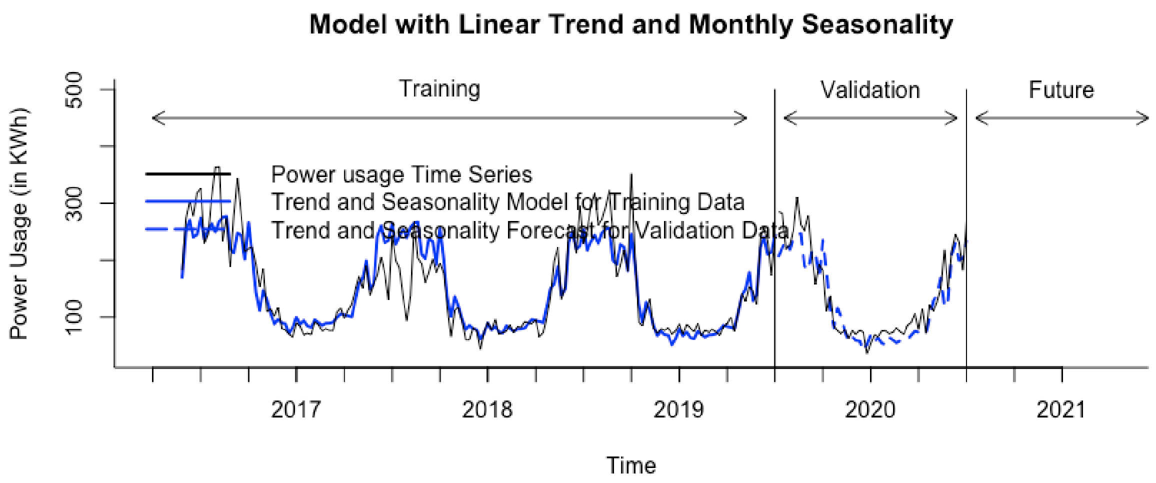

Figure No: 3.7.3

Regression Model with Linear Trend and Seasonality for Entire Data Set

The summary of the regression model with linear trend and seasonality on the entire data set is shown below:

```
Call:
tslm(formula = value.ts ~ trend + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-145.994  -15.538   -0.554   12.971  118.512 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  94.55463   20.74951   4.557 1.02e-05 ***
trend        -0.12789    0.04548  -2.812 0.005535 ** 
season2      -5.11161   28.47855  -0.179 0.857779    
season3      -1.02146   28.47866  -0.036 0.971432    
season4      -5.51532   28.47884  -0.194 0.846683    
season5      -6.97742   28.47910  -0.245 0.806767    
season6       1.64897   28.47943   0.058 0.953900    
season7      -1.49788   28.47983  -0.053 0.958120    
season8      -2.96899   28.48030  -0.104 0.917103    
season9      -1.90210   28.48084  -0.067 0.946836    
season10     -2.60182   28.48146  -0.091 0.927327    
season11      2.84594   28.48215   0.100 0.920532    
season12     10.08000   28.48291   0.354 0.723880    
season13     19.56473   28.48375   0.687 0.493151    
season14     11.08338   28.48466   0.389 0.697716    
season15     19.08452   28.48564   0.670 0.503837    
season16      6.79342   28.48669   0.238 0.811815    
season17     41.18281   28.48781   1.446 0.150225    
season18     60.99120   28.48901   2.141 0.033790 *  
season19     72.31010   28.49028   2.538 0.012096 *  
season20    101.11174   28.49163   3.549 0.000507 ***
season21     80.69514   28.49304   2.832 0.005215 ** 
season22     72.24748   27.01805   2.674 0.008267 ** 
season23    151.65457   27.01771   5.613 8.50e-08 ***
season24    172.61327   27.01744   6.389 1.72e-09 ***
season25    145.91816   27.01725   5.401 2.34e-07 ***
season26    139.49826   27.01713   5.163 7.08e-07 ***
season27    180.92675   27.01709   6.697 3.38e-10 ***
season28    158.27508   28.50121   5.553 1.13e-07 ***
season29    168.89344   28.49943   5.926 1.83e-08 ***
season30    162.14868   28.49772   5.690 5.86e-08 ***
season31    152.89645   28.49609   5.366 2.77e-07 ***
season32    173.66647   28.49453   6.095 7.81e-09 ***
season33    193.78786   28.49304   6.801 1.93e-10 ***
season34    182.93601   28.49163   6.421 1.46e-09 ***
season35    137.80690   28.49028   4.837 3.06e-06 ***
season36    138.96305   28.48901   4.878 2.56e-06 ***
season37    147.51327   28.48781   5.178 6.61e-07 ***
season38    131.75308   28.48669   4.625 7.65e-06 ***
season39    109.44783   28.48564   3.842 0.000175 ***
season40    154.97940   28.48466   5.441 1.94e-07 ***
season41     96.19527   28.48375   3.377 0.000918 ***
season42     53.01016   28.48291   1.861 0.064550 .  
season43     15.45769   28.48215   0.543 0.588076    
season44     39.56195   28.48146   1.389 0.166738    
season45     26.16110   28.48084   0.919 0.359705    
season46      9.87431   28.48030   0.347 0.729264    
season47     -7.29112   28.47983  -0.256 0.798270    
season48      2.15203   28.47943   0.076 0.939860    
season49     -4.01758   28.47910  -0.141 0.887990    
season50     -3.06768   28.47884  -0.108 0.914354    
season51    -16.52779   28.47866  -0.580 0.562485    
season52    -17.54764   28.47855  -0.616 0.538652    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.27 on 161 degrees of freedom
Multiple R-squared:  0.8123,	Adjusted R-squared:  0.7516 
F-statistic:  13.4 on 52 and 161 DF,  p-value: < 2.2e-16
```

Figure No: 3.7.4

- The regression model with linear trend and seasonality on the entire data set contains 1 period index(t) and 51 independent dummy variables for seasons. The dummy variable for season2 is D_2, season3 is D_3, season4 is D_4 and so on.
- The equation of the model is as follows:

$$\mathbf{Y_t = 94.555 - 0.128\ t - 5.112\ D_2 - 1.021\ D_3 - 5.515\ D_4 - 6.977\ D_5 - \ldots - 16.528\ D_{51} - 17.548\ D_{52}}$$

- The summary represents that the model has a high R-squared value of 0.8123 and high adjusted R-squared value of 0.7516, which indicates a very good fit for the training data and shows that the model with linear trend and seasonality is statistically significant. It can be noted that the model is statistically significant since the F-statistic p-value is very much lower than 0.05 or 0.01 (2.2e-16). Also, not all the predictors in the model are significant since their p-values are greater than 0.05.
- After developing the Regression model with linear trend and seasonality on the entire data set, we can forecast for the future period, after comparing for the best accuracy.

Autocorrelation left after the final model:

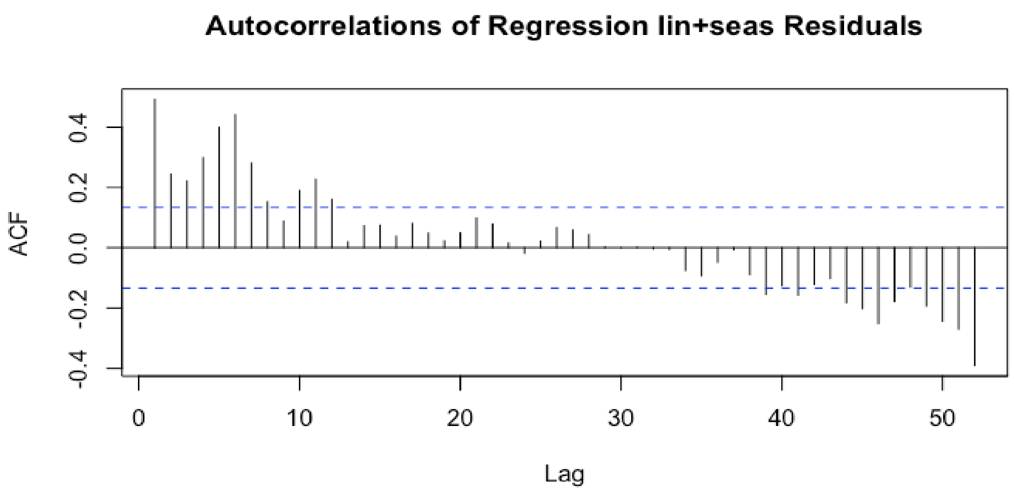

Figure No: 3.7.5

Model 2: Regression with linear trend and seasonality + trailing MA with regression residuals with training and validation dataset

Forecasting with training and validation dataset:

After developing the Regression model with linear trend and seasonality + trailing MA with regression residuals on training data partition, we predicted the point forecast (Regression Forecast+ the residual forecast from MA) on validation data set as shown in below figure.

	Week	Start Date	Twolevel Fst(R.lin.sea+MA(1)		Week	Start Date	
1	Week 27	2019-07-08	202.42667	27	Week 1	2020-01-06	59.43391
2	Week 28	2019-07-15	217.92066	28	Week 2	2020-01-13	66.74522
3	Week 29	2019-07-22	228.34207	29	Week 3	2020-01-20	56.96023
4	Week 30	2019-07-29	211.44093	30	Week 4	2020-01-27	54.73677
5	Week 31	2019-08-05	231.13458	31	Week 5	2020-02-03	67.57955
6	Week 32	2019-08-12	243.20494	32	Week 6	2020-02-10	62.53162
7	Week 33	2019-08-19	244.11125	33	Week 7	2020-02-17	58.08539
8	Week 34	2019-08-26	188.43091	34	Week 8	2020-02-24	61.87789
9	Week 35	2019-09-02	177.77690	35	Week 9	2020-03-02	62.39690
10	Week 36	2019-09-09	212.11997	36	Week 10	2020-03-09	64.36708
11	Week 37	2019-09-16	209.87923	37	Week 11	2020-03-16	71.41157
12	Week 38	2019-09-23	169.81225	38	Week 12	2020-03-23	78.66348
13	Week 39	2019-09-30	236.69605	39	Week 13	2020-03-30	77.13192
14	Week 40	2019-10-07	179.75703	40	Week 14	2020-04-06	75.44473
15	Week 41	2019-10-14	111.68050	41	Week 15	2020-04-13	72.71587
16	Week 42	2019-10-21	79.68227	42	Week 16	2020-04-20	101.72679
17	Week 43	2019-10-28	115.06163	43	Week 17	2020-04-27	132.03555
18	Week 44	2019-11-04	100.95457	44	Week 18	2020-05-04	143.04287
19	Week 45	2019-11-11	75.98049	45	Week 19	2020-05-11	173.19031
20	Week 46	2019-11-18	58.06704	46	Week 20	2020-05-18	123.22922
21	Week 47	2019-11-25	65.99303	47	Week 21	2020-05-25	123.64215
22	Week 48	2019-12-02	61.03344	48	Week 22	2020-06-01	207.33622
23	Week 49	2019-12-09	60.09109	49	Week 23	2020-06-08	224.92808
24	Week 50	2019-12-16	43.08473	50	Week 24	2020-06-15	194.72491
25	Week 51	2019-12-23	54.64945	51	Week 25	2020-06-22	197.19277
26	Week 52	2019-12-30	71.70384	52	Week 26	2020-06-29	227.13714

Figure No: 3.7.6

We have plotted the forecast of regression residuals and MA residuals in training and validation partitions.

The plot can be seen below:

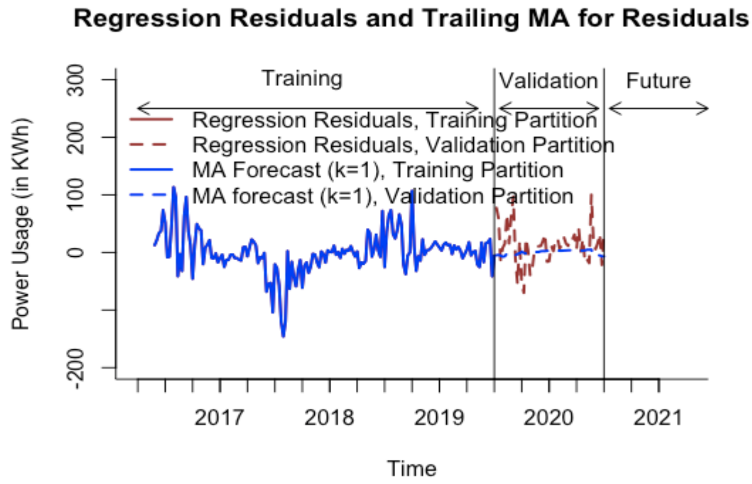

Figure No: 3.7.7

Forecasting on entire dataset:

Further, in this two-level model, the regression model with linear trend and seasonality has been developed based on the entire data set. Then the regression residuals and trailing MA residuals (window width of 1) for the entire data set are identified. Post that, we can develop the regression forecast and trailing MA forecast, window width of 1 for the future 12 weeks in 2020, after comparing for the best accuracy.

Auto correlation left after the final model:

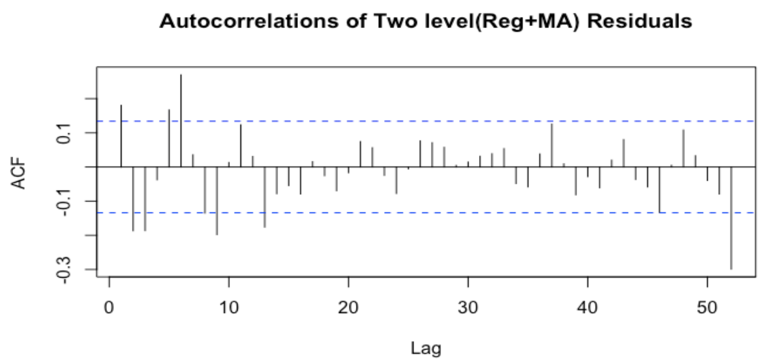

Figure No: 3.7.8

Model 3: ARIMA model

Automated ARIMA model on Training/Validation Data

- The Autoregressive Integrated Moving Average (ARIMA) model can be applied on data with level, trend and seasonality. Here, we have used the auto.arima() function to utilize the optimal values of our model parameters. So, an optimal ARIMA model was generated with automatic selection of (p,d,q) $(P,D,Q)_m$ parameters using the auto.arima() function.
- Below is the output of the Auto ARIMA model on the training data partition.

```
> summary(arima.train)
Series: value.train.ts
ARIMA(2,1,0)(1,1,0)[52]

Coefficients:
          ar1      ar2     sar1
      -0.2876  -0.2352  -0.4999
s.e.   0.0950   0.0937   0.0965

sigma^2 estimated as 2404:  log likelihood=-584.98
AIC=1177.95   AICc=1178.34   BIC=1188.72

Training set error measures:
                   ME  RMSE      MAE       MPE     MAPE      MASE        ACF1
Training set 0.424962 39.66 21.68779 -2.384109 14.97671 0.4910838 -0.08049409
```

Figure No: 3.7.9

- From the summary of the auto ARIMA model, we can see that the ARIMA model developed is (2,1,0) (1,1,0) [52] for order components. The ARIMA (2,1,0)(1,1,0)[52] is a seasonal ARIMA model of the form (p,d,q)$(P,D,Q)_m$ where:
  - p=2, order 2 Autoregressive model AR(2)
  - d=1, order 1 differencing to remove trend
  - q=0, no moving average model for error lags
  - P=1, order 1 Autoregressive Model AR(1) for seasonality
  - D=1, order 1 differencing for seasonality

  - Q=0, no moving average model for seasonal error lags
  - M=52, for weekly seasonality

- The model consists of an AR-1 coefficient, an AR-2 coefficient, and an order 1 seasonal autoregressive model with values of -0.2876, -0.2352, -0.4999 respectively.
- The equation of the model is as follows:

$$\mathbf{y_t - y_{t-1} = -\,0.2876(y_{t-1} - y_{t-2}) - 0.2352(y_{t-2} - y_{t-3}) - 0.4999(y_{t-1} - y_{t-53})}$$

After developing the auto ARIMA model for the time series on training data, we predicted the point forecast on validation data set.

	Week	Start Date	Forecast
1	Week 27	2019-07-08	235.17675
2	Week 28	2019-07-15	237.75334
3	Week 29	2019-07-22	224.08484
4	Week 30	2019-07-29	178.33113
5	Week 31	2019-08-05	204.77597
6	Week 32	2019-08-12	280.30411
7	Week 33	2019-08-19	263.07539
8	Week 34	2019-08-26	220.79086
9	Week 35	2019-09-02	165.06243
10	Week 36	2019-09-09	184.09345
11	Week 37	2019-09-16	209.43636
12	Week 38	2019-09-23	179.72220
13	Week 39	2019-09-30	273.13842
14	Week 40	2019-10-07	189.24162
15	Week 41	2019-10-14	103.84595
16	Week 42	2019-10-21	74.98719
17	Week 43	2019-10-28	111.73298
18	Week 44	2019-11-04	124.62763
19	Week 45	2019-11-11	86.32392
20	Week 46	2019-11-18	65.41055
21	Week 47	2019-11-25	69.51478
22	Week 48	2019-12-02	78.69290
23	Week 49	2019-12-09	77.95425
24	Week 50	2019-12-16	56.63774
25	Week 51	2019-12-23	75.48092
26	Week 52	2019-12-30	88.90558
27	Week 1	2020-01-06	73.77592
28	Week 2	2020-01-13	90.95025
29	Week 3	2020-01-20	75.38904
30	Week 4	2020-01-27	73.00718
31	Week 5	2020-02-03	81.52429
32	Week 6	2020-02-10	76.31372
33	Week 7	2020-02-17	74.67208
34	Week 8	2020-02-24	78.15107
35	Week 9	2020-03-02	79.91382
36	Week 10	2020-03-09	82.49735
37	Week 11	2020-03-16	78.67525
38	Week 12	2020-03-23	85.50666
39	Week 13	2020-03-30	91.58497
40	Week 14	2020-04-06	82.29238
41	Week 15	2020-04-13	73.38475
42	Week 16	2020-04-20	104.13433
43	Week 17	2020-04-27	137.38054
44	Week 18	2020-05-04	162.54844
45	Week 19	2020-05-11	187.70659
46	Week 20	2020-05-18	138.18198
47	Week 21	2020-05-25	137.89442
48	Week 22	2020-06-01	249.51143
49	Week 23	2020-06-08	255.60602
50	Week 24	2020-06-15	222.59643
51	Week 25	2020-06-22	234.35093
52	Week 26	2020-06-29	233.39883

Figure No: 3.7.10

We have plotted the forecast of the Arima model in training and validation partitions.

The plot can be seen below:

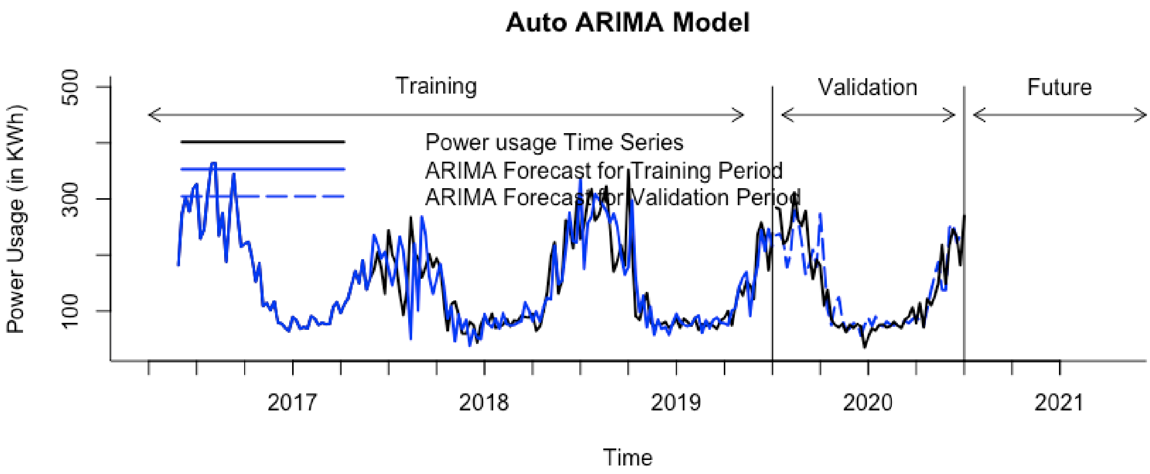

Figure No: 3.7.11

After prediction of point forecast on the validation data partition, we plot the correlogram using lag.max=52 and check the residuals of the time series data set. From the plot, we can see that even after applying the forecasting model, there is still some autocorrelation seen in the correlogram.

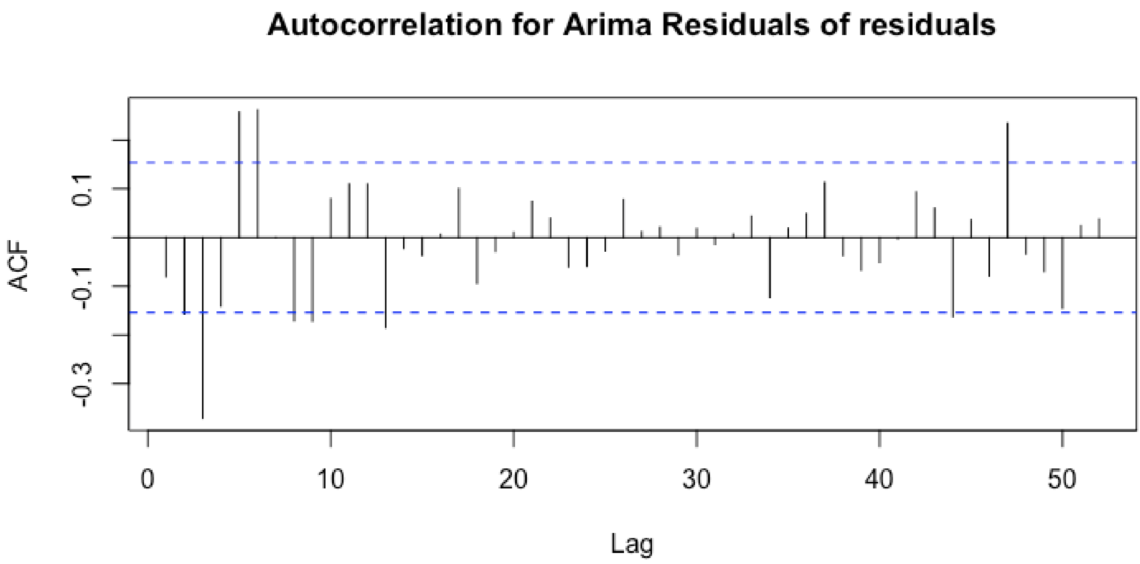

Figure No: 3.7.12

Automated ARIMA model for the Entire Data:

Below is the output of Auto ARIMA model which is executed on the entire data set:

```
Series: value.ts
ARIMA(5,1,0)(1,1,0)[52]

Coefficients:
          ar1      ar2      ar3      ar4      ar5     sar1
      -0.6154  -0.6719  -0.5831  -0.4835  -0.2541  -0.5533
s.e.   0.0771   0.0829   0.0863   0.0821   0.0769   0.0693

sigma^2 estimated as 1577:  log likelihood=-828.33
AIC=1670.67   AICc=1671.4   BIC=1692.24

Training set error measures:
                   ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set 1.418891 33.79621 21.15942 -1.089009 15.76813 0.5512648 -0.003122374
```

Figure No: 3.7.13

- From the summary of the auto ARIMA model on the entire data set, we can see that the ARIMA model developed is (5,1,0)(1,1,0)[52] for order components. The ARIMA (5,1,0)(1,1,0)[52] is a seasonal ARIMA model of the form $(p,d,q)(P,D,Q)_m$ where:
    - p=5, order 5 Autoregressive model AR(3)
    - d=1, order1 differencing to remove trend
    - q=0, no moving average model for error lags
    - P=1, order 1 Autoregressive Model AR(1) for seasonality
    - D=1, order 1 differencing for seasonality
    - Q=0, no moving average model for seasonal error lags
    - M=52, for weekly seasonality
- The model consist of an AR-1 coefficient, an AR-2 coefficient, an AR-3 coefficient, an AR-4 coefficient, an AR-5 coefficient and an order 1 seasonal autoregressive model of -0.6154, -0.6719, -0.5831, -0.4835, -0.2541, and -0.5533, respectively.

- The equation of the model is as follows:

$$\mathbf{y_t - y_{t-1} = -\ 0.6154(y_{t-1} - y_{t-2}) - 0.6719(y_{t-2} - y_{t-3}) - 0.5831(y_{t-3} - y_{t-4}) - 0.4835(y_{t-4} - y_{t-5}) - 0.2541(y_{t-5} - y_{t-6}) - 0.5533(y_{t-1} - y_{t-53})}$$

After developing the auto ARIMA model for the time series on the entire data, we can use the model forecast for the 12-future period prediction of the ARIMA model, after comparing for the best accuracy.

Autocorrelation left after the final model:

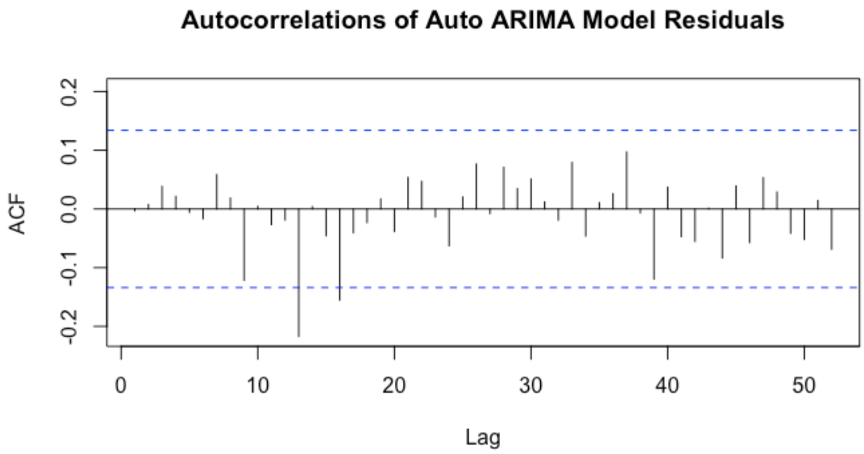

Figure No: 3.7.14

Comparing Performance of the best models selected:

```
> round(accuracy(total.lintrend.season.pred$fitted, value.ts),3)
         ME   RMSE    MAE    MPE   MAPE  ACF1 Theil's U
Test set  0 34.933 23.475 -3.888 15.721 0.491     0.813
> round(accuracy(total.lintrend.season.pred$fitted + tot.ma.trail.res.pred$fitted, value.ts), 3)
             ME   RMSE    MAE    MPE   MAPE  ACF1 Theil's U
Test set -0.584 29.659 20.649 -3.001 14.382 0.179     0.681
> round(accuracy(arima.total.pred$fitted, value.ts),3)
            ME   RMSE    MAE    MPE   MAPE   ACF1 Theil's U
Test set 1.419 33.796 21.159 -1.089 15.768 -0.003     0.895
> round(accuracy((naive(value.ts))$fitted, value.ts), 3)
            ME  RMSE    MAE    MPE   MAPE   ACF1 Theil's U
Test set 0.411 42.57 30.307 -3.222 20.719 -0.175         1
> round(accuracy((snaive(value.ts))$fitted, value.ts), 3)
             ME   RMSE    MAE     MPE   MAPE  ACF1 Theil's U
Test set -9.878 62.404 38.383 -13.385 27.521 0.553     1.386
> 
```

Figure No: 3.7.15

Sr. No.	Model	RMSE	MAPE
1	Regression with Linear Trend and Seasonality	34.933	15.721
2	Two-Level Model: Regression with Linear Trend and Seasonality + Trailing MA for residuals	29.659	14.382
3	ARIMA Model	33.796	15.768
4	Naive Forecasting	42.57	20.719
5	Seasonal Naïve Forecasting	62.404	27.521

Table 4: Accuracy measures for Best identified models, Seasonal Naive and Naive Forecast

From the accuracy Table 4, it can be seen that "Two-Level Model with Linear Trend and Seasonality and Trailing MA" is the best and accurate model for forecasting weekly power usage. It has the RMSE and MAPE values as 29.569 and 14.382, respectively, which is the lowest compared to the final models taken under consideration. Hence, this model can be used to predict the forecasting for Residential Power Usage.

3.8. Implement the Forecast on Future Data

Based on the above evaluation of the models the Best of all the selected models is the two-level Regression model with trend and seasonality + trailing moving average for the residuals In this two-level model, is utilized to forecast the future 12 weeks power usage which can been seen in the table below:

```
> weekly.valid.fst18
      Week   Start Date   Residual Future Forecast (MA)
1  Week 27   2020-07-06                       11.880603
2  Week 28   2020-07-13                       12.336634
3  Week 29   2020-07-20                        7.118369
4  Week 30   2020-07-27                        2.805209
5  Week 31   2020-08-03                        4.141302
6  Week 32   2020-08-10                       14.986960
7  Week 33   2020-08-17                       11.887896
8  Week 34   2020-08-24                       14.968590
9  Week 35   2020-08-31                        8.400736
10 Week 36   2020-09-07                        2.745879
11 Week 37   2020-09-14                        6.134885
12 Week 38   2020-09-21                       10.346108
>
```

Figure No: 3.8.1

The plot for residuals of residuals can be seen in the following figure:

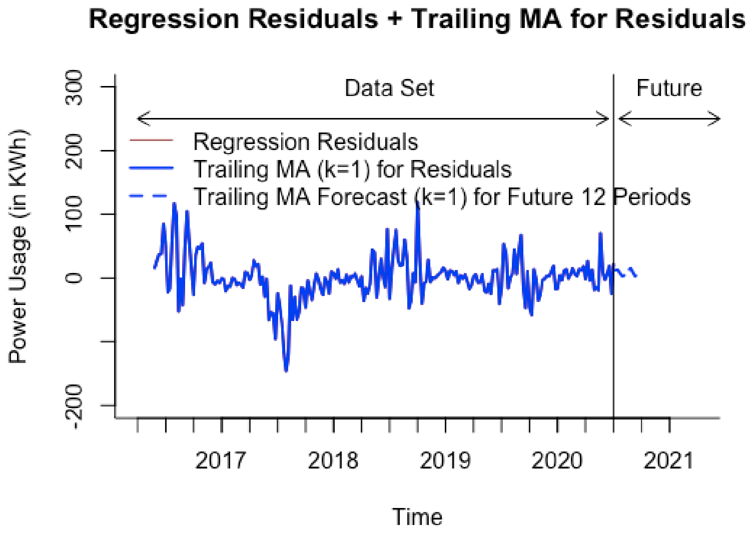

Figure No: 3.8.2

4. CONCLUSION

From the above analysis, we can conclude that the best identified model to forecast the weekly power usage is the Regression model with linear trend and seasonality with a Trailing Moving Average for residual prediction. Furthermore, we can see that even after applying the best model, there is some autocorrelation left in the series which is not accounted for by the models. This can be seen from the correlograms shown earlier. Also, we have applied multivariate regression with external variables to predict power usage. However, the accuracy measures are not as good as the other models without considering independent variables. Hence, we can say that the regression model trailing MA can be utilized to forecast Power Usage for subsequent weeks with a reasonable accuracy which will help the power generation companies to plan their production accordingly. This will assist in avoiding sudden power grid failures due to overload. The model of choice should be reevaluated every 3 to 4 months to ensure that accurate forecasts can be achieved for the subsequent periods.

5. BIBLIOGRAPHY

1. Shmueli, G. and Lichtendahl Jr., K.C. Practical Time Series Forecasting with R, 2nd Edition, Axelrod Schnall Publishers, 2016. ISBN-13: 978-0-9978479-1-8.
2. https://robjhyndman.com/hyndsight/seasonal-periods/
3. https://www.rdocumentation.org/packages/forecast/versions/8.15
4. https://cran.r-project.org/web/packages/forecast/forecast.pdf
5. https://en.wikipedia.org/wiki/2021_Texas_power_crisis
6. https://www.eia.gov/state/analysis.php?sid=TX