TIMESERIES

BOUHIA Zakaria

2024-08-15

Introduction:

This project contains the analysis and forecasting models for predicting electricity consumption (kW) for a specific building on 2/21/2010. The project aims to develop two separate forecasts:

1. A forecast without using outdoor temperature data
2. A forecast incorporating outdoor temperature data

The primary dataset, ‘2023-11-Elec-train.xlsx’, includes electricity consumption and outdoor air temperature measurements at 15-minute intervals from 1/1/2010 1:15 to 2/20/2010 23:45. Additionally, we have outdoor air temperature data for the target date (2/21/2010).

Our objective is to test and compare various forecasting models to achieve the most accurate predictions possible. This R Markdown will document the entire process, including:

- Data loading and preprocessing
- Exploratory data analysis
- Model selection and tuning
- Forecast generation
- Model evaluation and comparison

We will use R for our analysis, leveraging various time series forecasting techniques covered in our course. The final output will be two sets of 96 predictions (representing 24 hours at 15-minute intervals) for electricity consumption on 2/21/2010.

Let’s begin by importing the necessary libraries and loading our data.

#“‘{r setup, include=FALSE} #install the packages install.packages(‘readxl’, repos = “http://cran.us.r-project.org”) install.packages(“openxlsx”, repos = “http://cran.us.r-project.org”) install.packages(‘tidyverse’, repos = “http://cran.us.r-project.org”) install.packages(“ggplot2”, repos = “http://cran.us.r-project.org”) install.packages(“forecast”, repos = “http://cran.us.r-project.org”) install.packages(“fpp2”, repos = “http://cran.us.r-project.org”) install.packages(“writexl”, repos = “http://cran.us.r-project.org”)

#“‘

```r
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 4.2.3
```

```r
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.2.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(openxlsx)
```

```
## Warning: package 'openxlsx' was built under R version 4.2.3
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
## Warning: package 'tibble' was built under R version 4.2.3
```

```
## Warning: package 'tidyr' was built under R version 4.2.3
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
## Warning: package 'purrr' was built under R version 4.2.3
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

```
## Warning: package 'forcats' was built under R version 4.2.3
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(ggplot2)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```r
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.2.3

## -- Attaching packages ---------------------------------------------- fpp2 2.5 --
## v fma       2.5     v expsmooth 2.3

## Warning: package 'fma' was built under R version 4.2.3

## Warning: package 'expsmooth' was built under R version 4.2.3

##
```

```r
library(writexl)
```

```
## Warning: package 'writexl' was built under R version 4.2.3
```

Data Importation:

```r
#data Importation
dataset_elect = read_excel('C:\\Users\\33751\\Desktop\\DSTI_course\\TimeSeriesAnalysis\\dataset\\2023-1
head(dataset_elect)
```

```
## # A tibble: 6 x 3
##   Timestamp          `Power (kW)` `Temp (C°)`
##   <chr>                     <dbl>       <dbl>
## 1 40179.052083333336         165.        10.6
## 2 1/1/2010 1:30              152.        10.6
## 3 1/1/2010 1:45              147.        10.6
## 4 1/1/2010 2:00              154.        10.6
## 5 1/1/2010 2:15              154.        10.6
## 6 1/1/2010 2:30              159         10.6
```

This dataset contains time series data with three main columns:

-Timestamp: Represents the date and time of each observation. -Power (kW): Shows the electricity consumption in kW. -Temp (C°): Indicates the outdoor temperature in degrees C°.

The data is recorded at 15-minute intervals, providing a high-resolution view of electricity consumption and temperature changes throughout the day. The dataset spans from January 1, 2010, to February 20, 2010, encompassing a total of 4,987 rows.

It's important to note that the dataset is not perfectly balanced at the start. The first hour of data for January 1, 2010, is missing. To ensure a balanced dataset for our analysis and to have complete days, we will remove the data for the entire first day (January 1, 2010). This will give us a clean starting point from January 2, 2010, with full 24-hour cycles for each day.

By removing the incomplete first day, we ensure that each day in our analysis has an equal number of data points (96 readings per day, given the 15-minute frequency), which will be crucial for accurate time-based analysis and forecasting.

```
#data processing
dataset_elect_balanced=dataset_elect[-(1:91), ] #delete the first incomplete first day
head(dataset_elect_balanced)
```

```
## # A tibble: 6 x 3
##   Timestamp     `Power (kW)` `Temp (C°)`
##   <chr>                <dbl>       <dbl>
## 1 1/2/2010 0:00         163.        13.3
## 2 1/2/2010 0:15         154.        10.6
## 3 1/2/2010 0:30         152.        10.6
## 4 1/2/2010 0:45         159.        10.6
## 5 1/2/2010 1:00         164.        10.6
## 6 1/2/2010 1:15         159.        10
```

Exploratory data analysis (EDA)

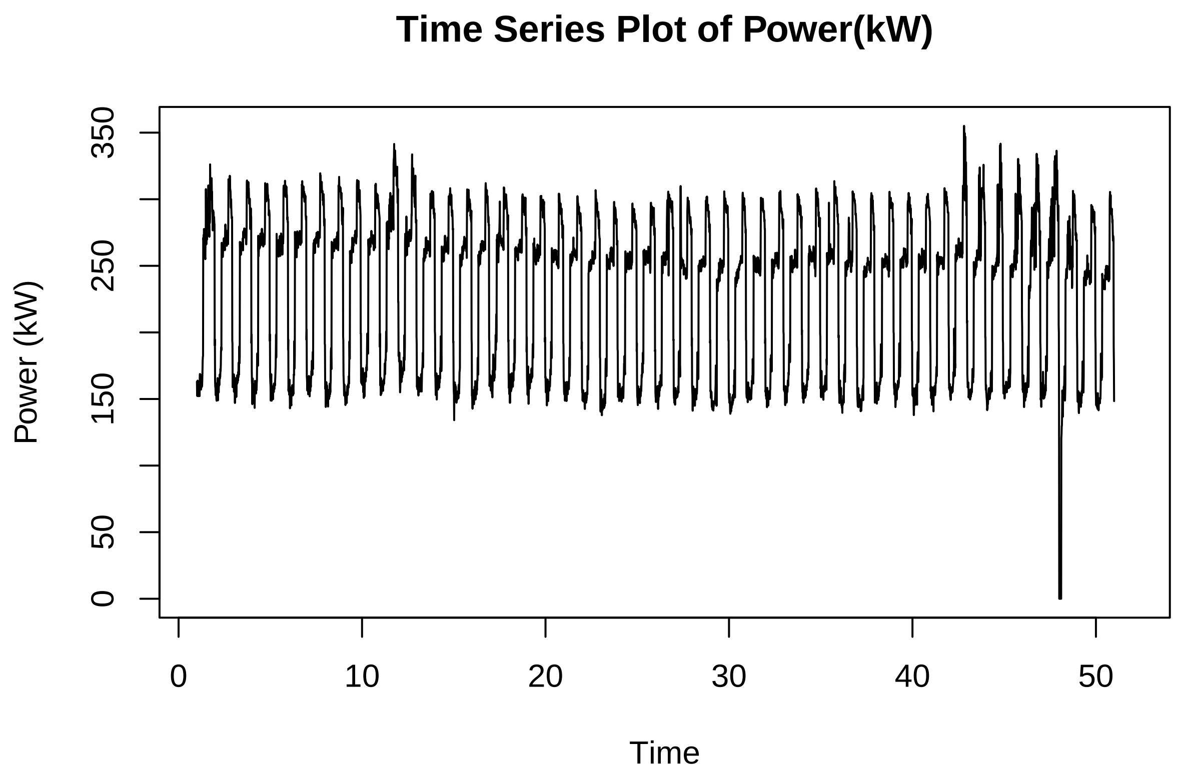
Before diving into the analysis of this time series, let's first transform it into a proper time series format.

```
#timeseries transformation format
ts_elect=ts(dataset_elect_balanced$`Power (kW)`, start=c(1,1), end=c(51,96), freq=96)
head(ts_elect)
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 96
## [1] 163.1 154.4 152.2 158.7 163.8 158.7
```

Let's Visualize this timee serie:

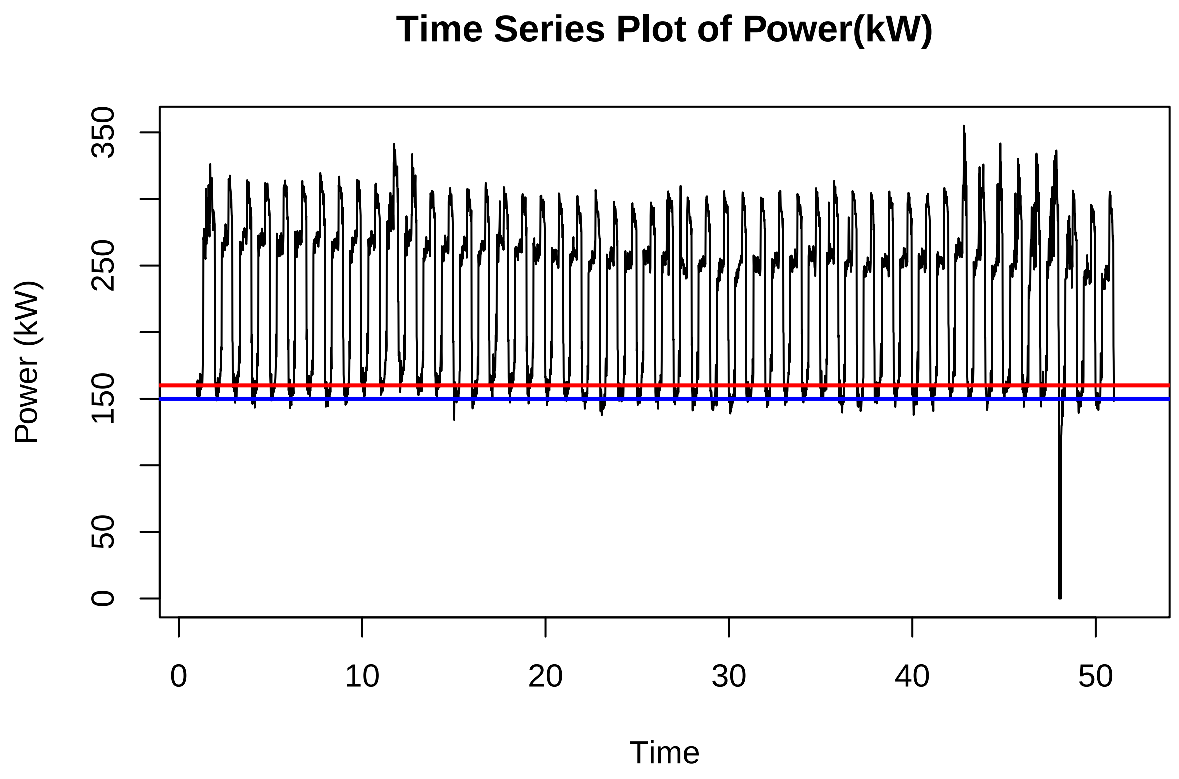
```
# Plot the time series
plot(ts_elect, type="l", main="Time Series Plot of Power(kW)", ylab="Power (kW)", xlab="Time")
```



As we notice, there is an anomalous values of 0. We'll attempt to fix it by assigning a logical value based on data variation.

```r
# Plot the time series
plot(ts_elect, type="l", main="Time Series Plot of Power(kW)", ylab="Power (kW)", xlab="Time")

# Add a horizontal line at y = 170
abline(h=160, col="red", lwd=2)
# Add a horizontal line at y = 150
abline(h=150, col="blue", lwd=2)
```



Graphically, a value between 150 and 170 seems reasonable, but I will opt for 160.

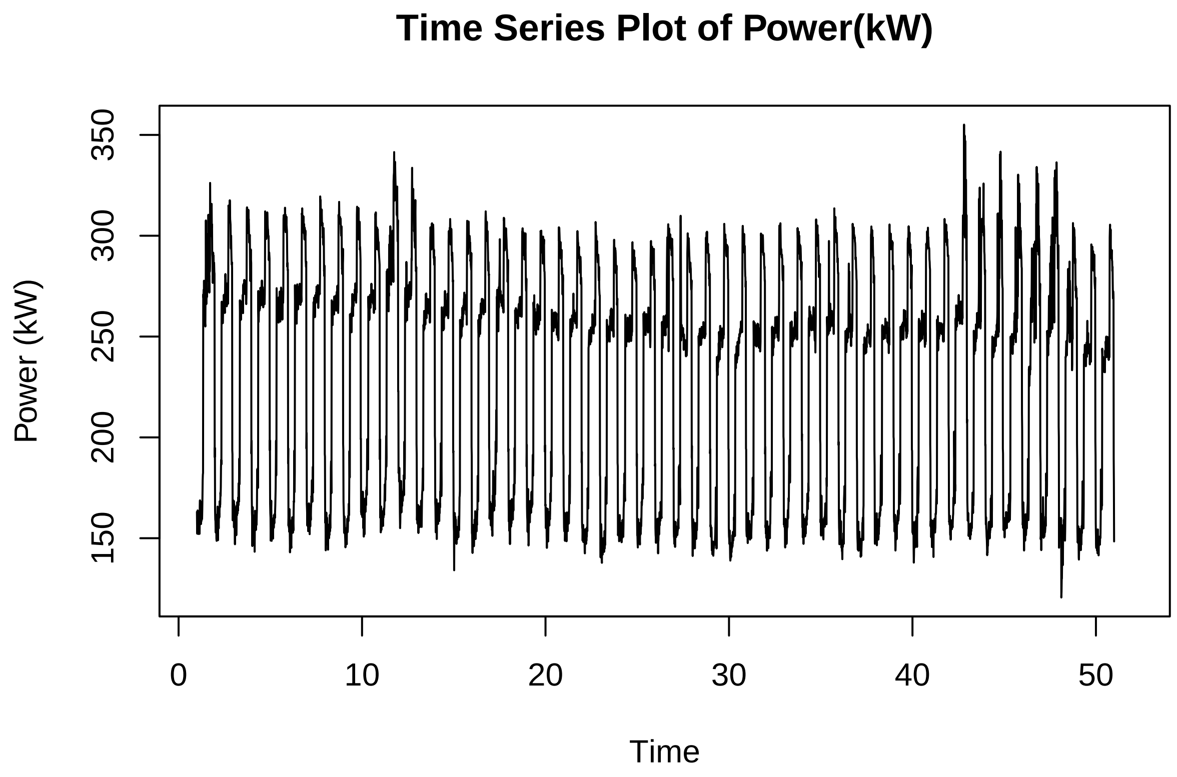
```
dataset_elect_balanced['Power (kW)'][dataset_elect_balanced['Power (kW)'] == 0]= 160
ts_elect=ts(dataset_elect_balanced$`Power (kW)`, start=c(1,1), end=c(51,96), freq=96)
head(ts_elect)
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 96
## [1] 163.1 154.4 152.2 158.7 163.8 158.7
```

```
summary(ts_elect)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   120.6   162.8   252.9   230.8   276.8   355.1      96
```

```r
plot(ts_elect, type="l", main="Time Series Plot of Power(kW)", ylab="Power (kW)", xlab="Time")
```

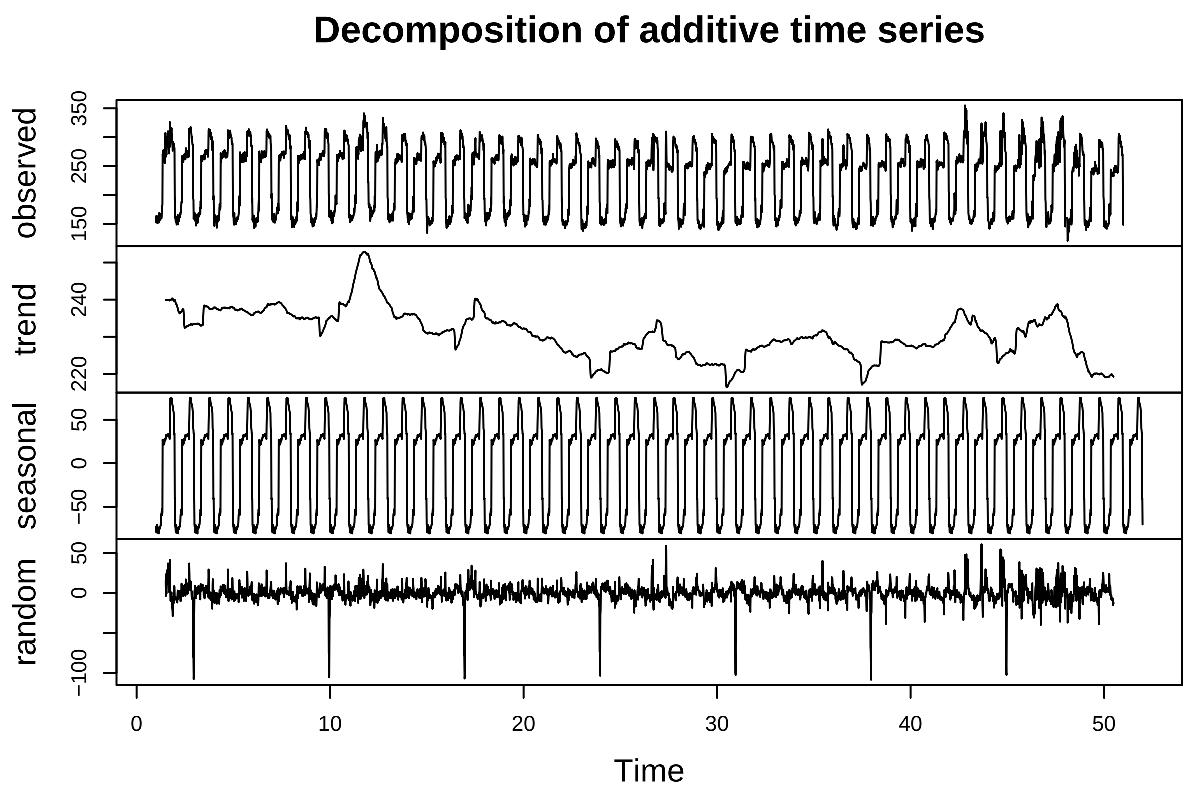


Forcasts without using outdoor temperature:

Decompsoiton Timeserie

Let's explore the decomposition of our time series data to visually analyze its trend and seasonal components.

```r
plot(decompose(ts_elect, type="additive"))
```



Upon examining the decomposition, we observe that there isn't a clear upward or downward trend in the data. However, the seasonal component reveals some consistent patterns, suggesting seasonality in the data. Given these observations, we'll focus on using seasonal models for forecasting, as they are better suited to capture and leverage the seasonal patterns present in the time series.

Train-Test data set Split

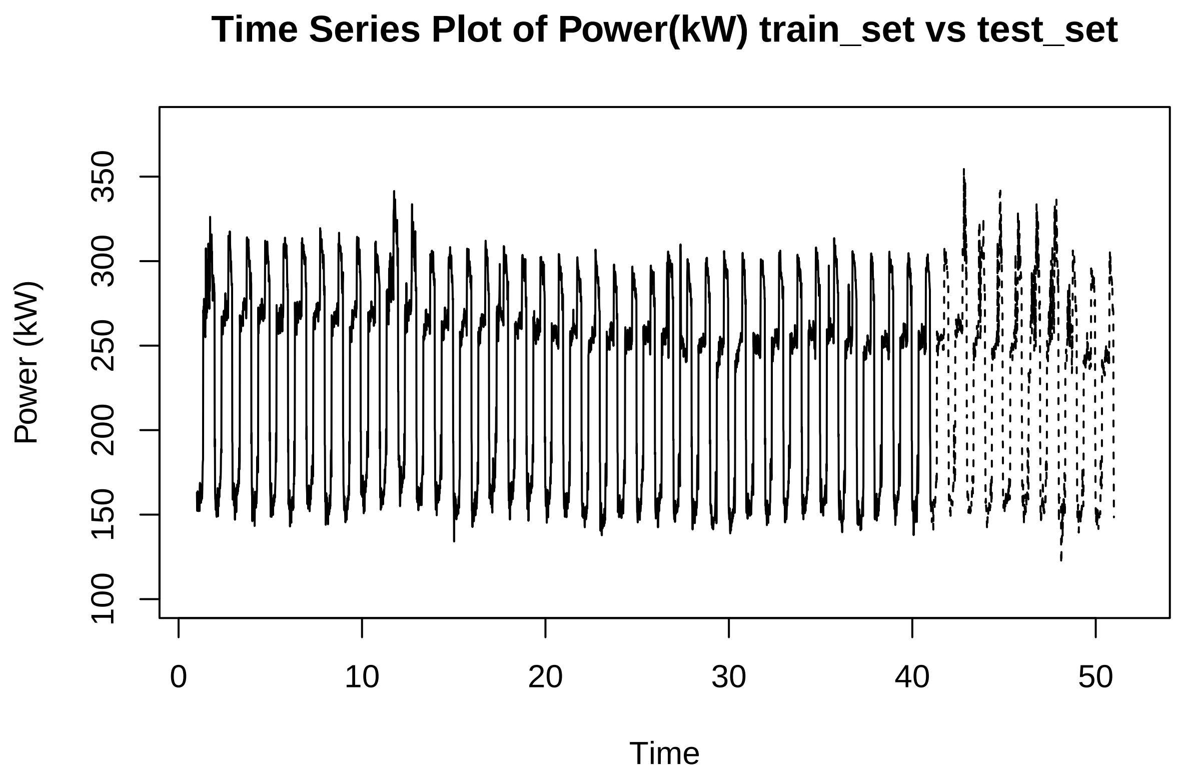
We are considering splitting the data into training and test sets using an 80/20 rule.

The training set will span from January 2, 2010, at 00:00 to February 10, 2010, at 23:45.

```
train_set <- window(ts_elect, start=c(1,1), end=c(40,96))
test_set <- window(ts_elect, start=c(41,1), end=c(50,96))



plot(train_set, xlim=c(1,52), ylim=c(100,380), type="l", main="Time Series Plot of Power(kW) train_set
lines(test_set, lty=2)
```



Simple Exponential Smoothing (SES)

To forecast electricity consumption using a regression model, we'll start with a basic approach by applying Simple Exponential Smoothing (SES). Since we are not considering any trend or seasonal components in this initial model, the forecast will be a constant value based on the level of the time series data.

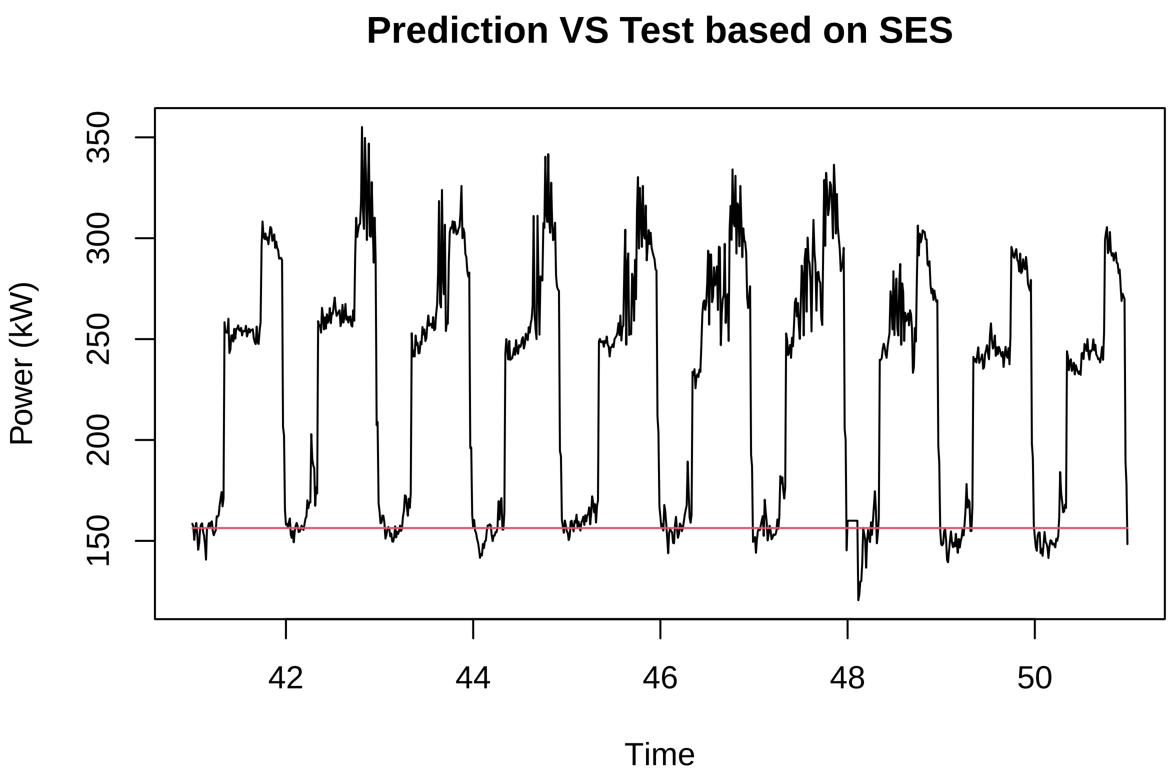
```r
SES=HoltWinters(train_set, alpha=NULL, beta=FALSE, gamma=FALSE)
print(SES)
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = train_set, alpha = NULL, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.9860426
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 156.3719
```

The model predicts future electricity consumption as a constant value of 156.3719. The high alpha value shows that the model is very responsive to recent changes in the data. However, since no trend or seasonal components are included, the model assumes these changes are not part of any ongoing patterns. Consequently, the forecast remains flat at the estimated level of 156.3719. That said, the value we considered to replace the null value (160 kW) is reasonable.

Let's predict for the next 10 days (until February 20, 2010, at 23:45) based on SES:

```
consum_pred1=predict(SES, n.ahead=96*10)
plot(test_set,type="l", main="Prediction VS Test based on SES", ylab="Power (kW)", xlab="Time")
lines(consum_pred1, col=2)
```



Let's Evaluate this model:

```
print(SES$alpha) # 0.9860426
```

```
## [1] 0.9860426
```

```
RMSE_SES=sqrt(mean((consum_pred1 - test_set)^2))
print(RMSE_SES)
```

```
## [1] 93.28456
```

-The high alpha value confirms that the SES model is primarily influenced by recent data, making it very adaptive to short-term changes.

-However, the RMSE of 93.28456 implies that there is still a noticeable level of error in the model's predictions. While the SES model captures the overall level of electricity consumption, the deviations from actual consumption might indicate that the data has underlying patterns (such as trends or seasonality) that are not being fully accounted for by this simple model.

As observed during the decomposition step, there is no trend but a noticeable seasonality. Therefore, we will apply the Holt-Winters seasonal model, considering the gamma coefficient while ignoring the beta component.

Holt-Winters seasonal model

```r
HW_Seasonal=HoltWinters(train_set, alpha=NULL, beta=FALSE, gamma=NULL)
print(HW_Seasonal)
```

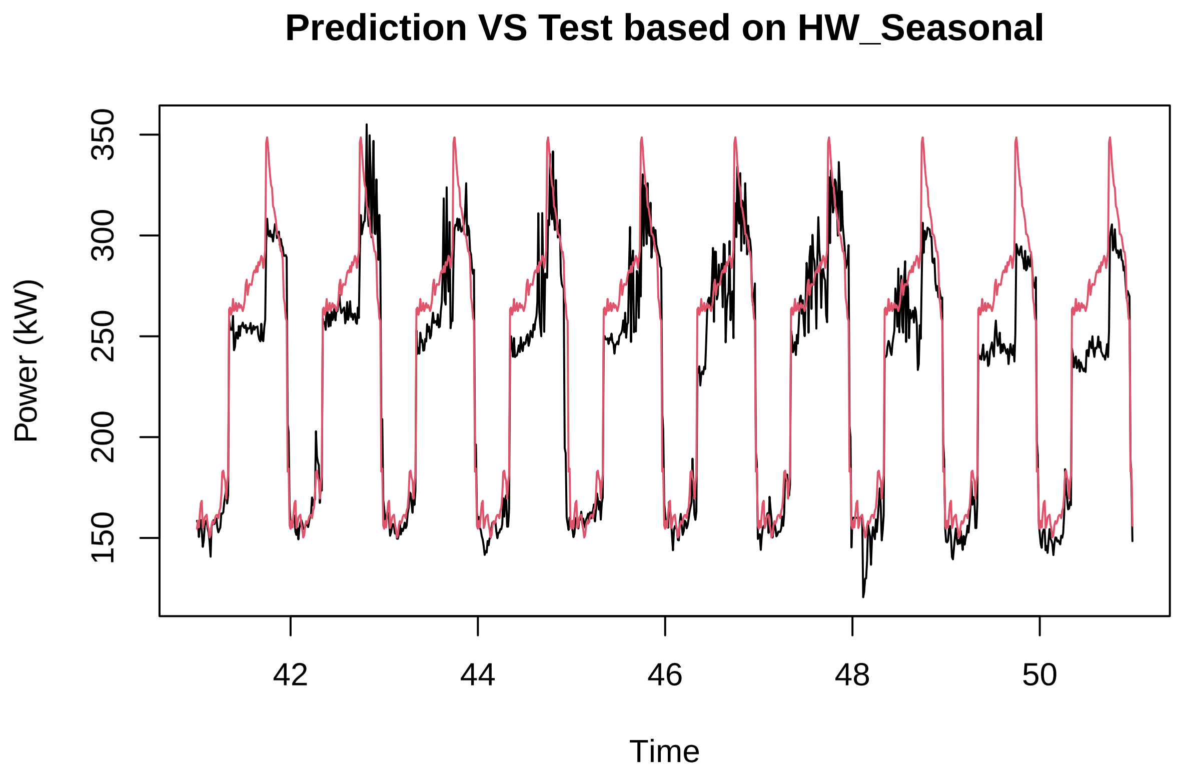
```
## Holt-Winters exponential smoothing without trend and with additive seasonal component.
##
## Call:
## HoltWinters(x = train_set, alpha = NULL, beta = FALSE, gamma = NULL)
##
## Smoothing parameters:
##  alpha: 0.7831196
##  beta : FALSE
##  gamma: 0.8904545
##
## Coefficients:
##          [,1]
## a   240.11894
## s1  -85.53653
## s2  -81.59209
## s3  -84.89480
## s4  -78.20116
## s5  -72.88999
## s6  -71.68576
## s7  -85.21335
## s8  -83.36854
## s9  -79.97476
## s10 -79.19373
## s11 -78.56924
## s12 -83.33520
## s13 -85.83806
## s14 -89.88809
## s15 -88.66898
## s16 -83.77997
## s17 -81.77421
## s18 -82.99303
## s19 -81.97842
## s20 -79.56752
## s21 -78.70675
## s22 -78.71416
## s23 -80.33166
## s24 -76.59747
## s25 -75.18980
## s26 -68.89655
## s27 -57.44692
## s28 -56.74519
## s29 -60.28571
## s30 -61.69323
## s31 -70.64407
## s32 -63.30628
## s33 -58.80193
## s34  23.24709
## s35  24.12167
## s36  20.54998
## s37  22.51192
## s38  28.38224
## s39  22.49509
## s40  23.29417
## s41  26.44203
## s42  22.38805
## s43  24.39604
## s44  26.29052
## s45  24.03414
## s46  25.34776
## s47  24.20117
## s48  22.40583
## s49  24.49166
## s50  27.72696
## s51  35.45720
## s52  37.98888
## s53  30.40060
## s54  34.20281
## s55  35.84587
## s56  35.46848
## s57  35.39524
## s58  38.46045
## s59  41.43965
## s60  42.53769
## s61  41.54371
## s62  44.69122
## s63  41.75707
## s64  46.65363
## s65  45.10486
## s66  47.29578
## s67  49.76163
## s68  48.48894
## s69  43.75145
## s70  47.81166
## s71  52.60843
## s72 105.88779
## s73 108.53680
## s74 103.49910
## s75  95.49748
## s76  89.73647
## s77  84.74433
## s78  83.48915
## s79  74.42472
## s80  73.32390
## s81  70.05516
## s82  67.09232
## s83  60.54797
## s84  60.42753
## s85  59.10355
## s86  55.27885
## s87  52.05517
## s88  51.74057
## s89  47.41802
## s90  29.00408
## s91  25.76486
## s92  18.88309
## s93  17.50113
## s94 -57.21982
## s95 -55.69980
## s96 -84.05392
```

The model captures the underlying level and seasonal patterns in the data, but it does not account for any trend. Alpha=0.7831196: The model gives moderate weight to recent observations when estimating the level of the series.

The seasonal component has significant variations, reflecting the importance of seasonality in the electricity consumption data.

Now we predict for the next 10 days as we did previosuly

```
consum_pred2=predict(HW_Seasonal,n.ahead=96*10)
plot(test_set,type="l", main="Prediction VS Test based on HW_Seasonal", ylab="Power (kW)", xlab="Time")
lines(consum_pred2, col=2)
```



```r
print(HW_Seasonal$alpha) #0.7831196
```

```
##     alpha
## 0.7831196
```

```r
print(HW_Seasonal$gamma) #0.8904545
```

```
##     gamma
## 0.8904545
```

```r
RMSE_HW_Seasonal=sqrt(mean((consum_pred2 - test_set)^2))
print(RMSE_HW_Seasonal)
```

```
## [1] 21.23858
```

The Holt-Winters seasonal model has an alpha of 0.7831196, indicating moderate smoothing of the level, and a gamma of 0.8904545, showing strong emphasis on recent seasonal patterns. The RMSE of 21.23858 suggests that the model provides accurate forecasts than the SES model since we have RMSE_HW_Seasonal<RMSE_SES.

Let's enhance our forecasting analysis by using ARIMA models, beginning with an Auto-ARIMA approach.

Auto-ARIMA

```r
auto_arima=auto.arima(train_set)
summary(auto_arima)
```

```
## Series: train_set
## ARIMA(1,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1
##       0.7815
## s.e.  0.0102
##
## sigma^2 = 95.46:  log likelihood = -13846.39
## AIC=27696.77   AICc=27696.77   BIC=27709.23
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.06254446 9.646271 5.620404 -0.1185649 2.611593 0.7134771
##                      ACF1
## Training set 0.0003086627
```

We can say that the Auto-ARIMA model identifies a SARIMA with an order of 1 and a seasonal period of 96.

```
consum_pred3=forecast(auto_arima,h=96*10)
```

Let's evaluate the model

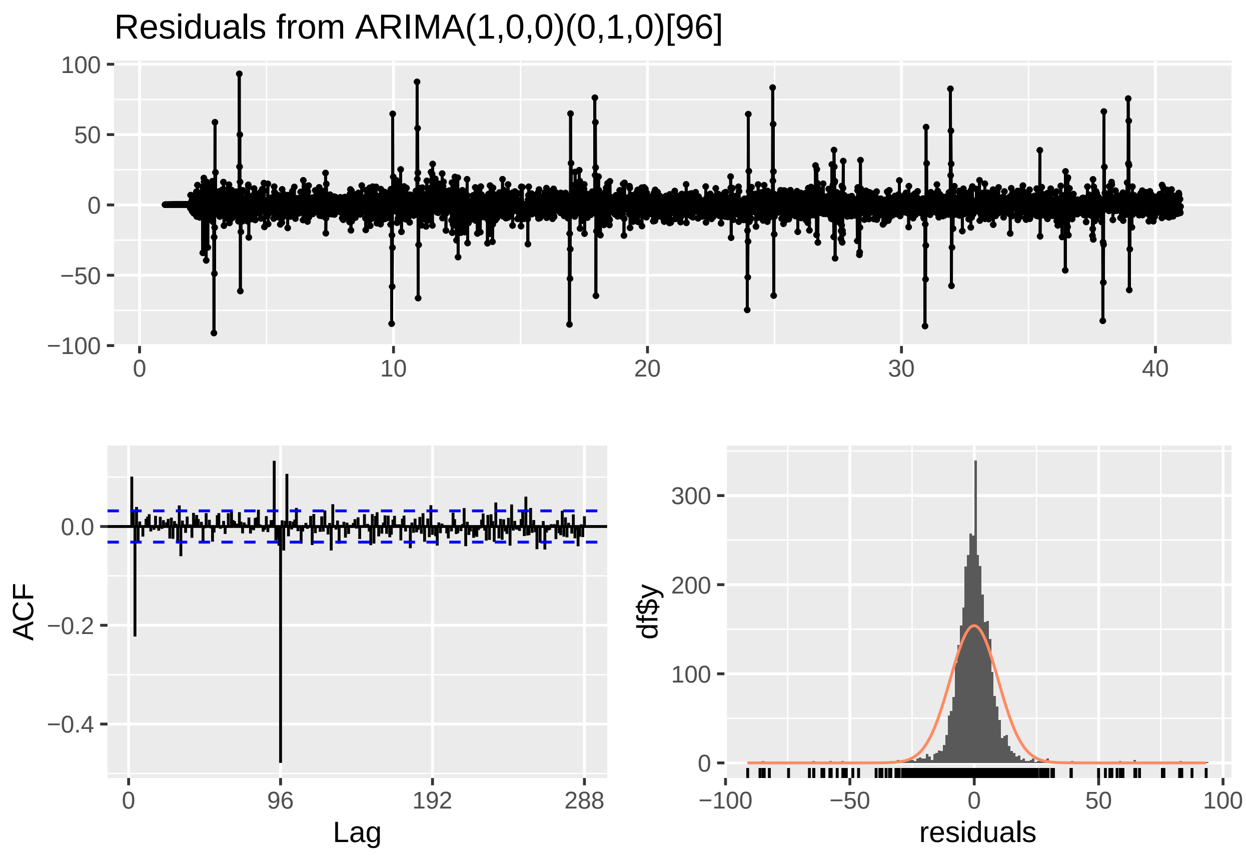
```
RMSE_ARIMA=sqrt(mean((consum_pred3$mean - test_set)^2))
print(RMSE_ARIMA)
```

```
## [1] 15.7564
```

The SARIMA model is currently the best among the three we've developed as it presents the lowest RMSE. However, we need to verify that the residuals are independent of previous values. This step helps validate the model's effectiveness and ensures reliable forecasts.

Residuals analysis

```
checkresiduals(auto_arima)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 1522.8, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

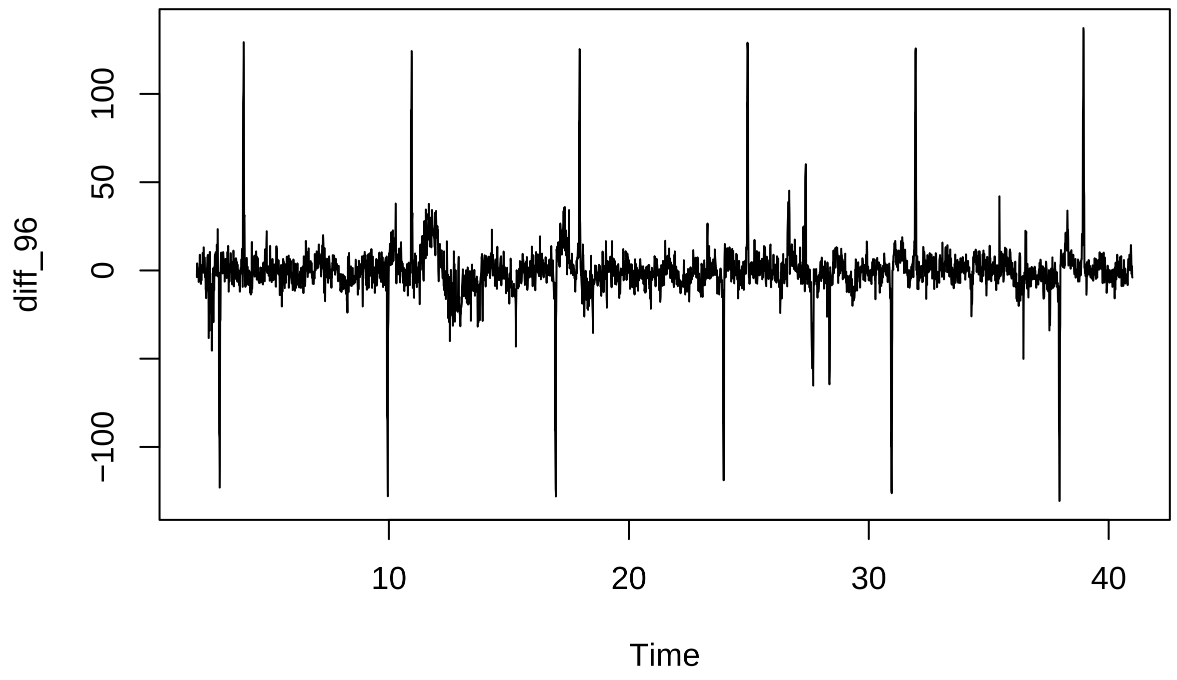
The very small p-value indicates that the residuals are not independent, suggesting significant dependence.This implies that the model might not have captured all underlying patterns. We need to go futher in the analysis by differencing the series to extract the residuals and achieve independence.

In order to launch SARIMA, let's follow thee process:

1- Removing Seasonality:

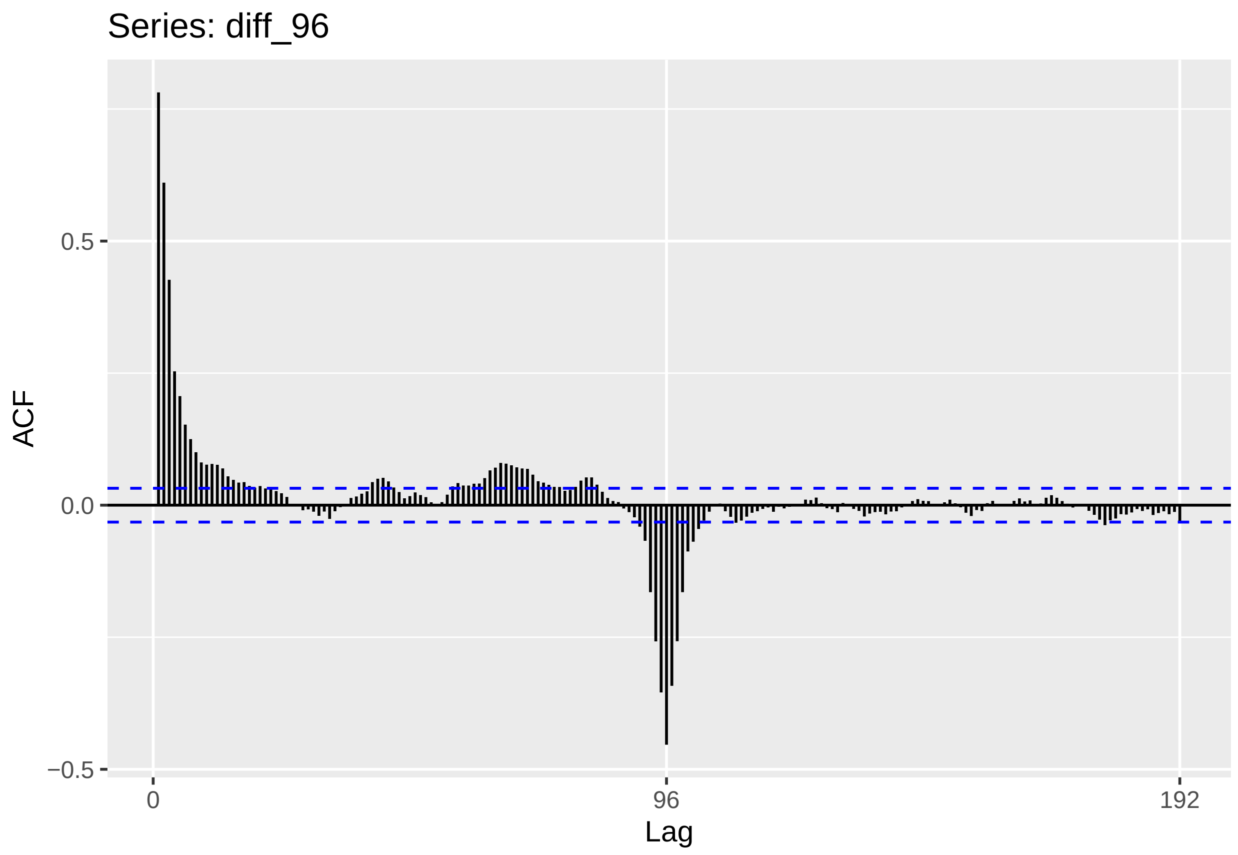
We are differencing the series with the appropriate lag. We choose a lag=96 that corresponding of the seasonal period of the time series.

```
diff_96= diff(train_set, lag=96)  #differcing with lag=96
plot(diff_96)
```



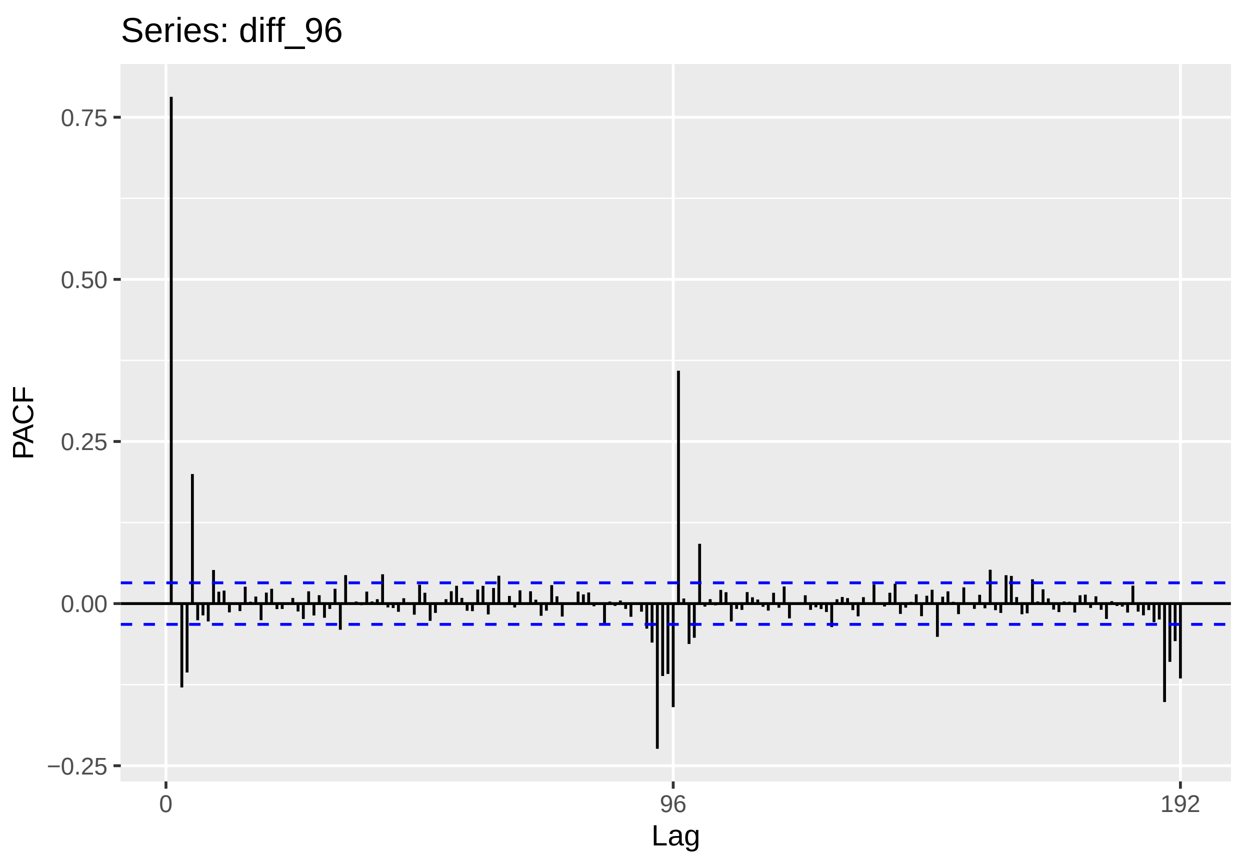
2- Examining the autocorrelation function (ACF):

```
ggAcf(diff_96)
```



3-Examining the Partial autocorrelation function (PACF)

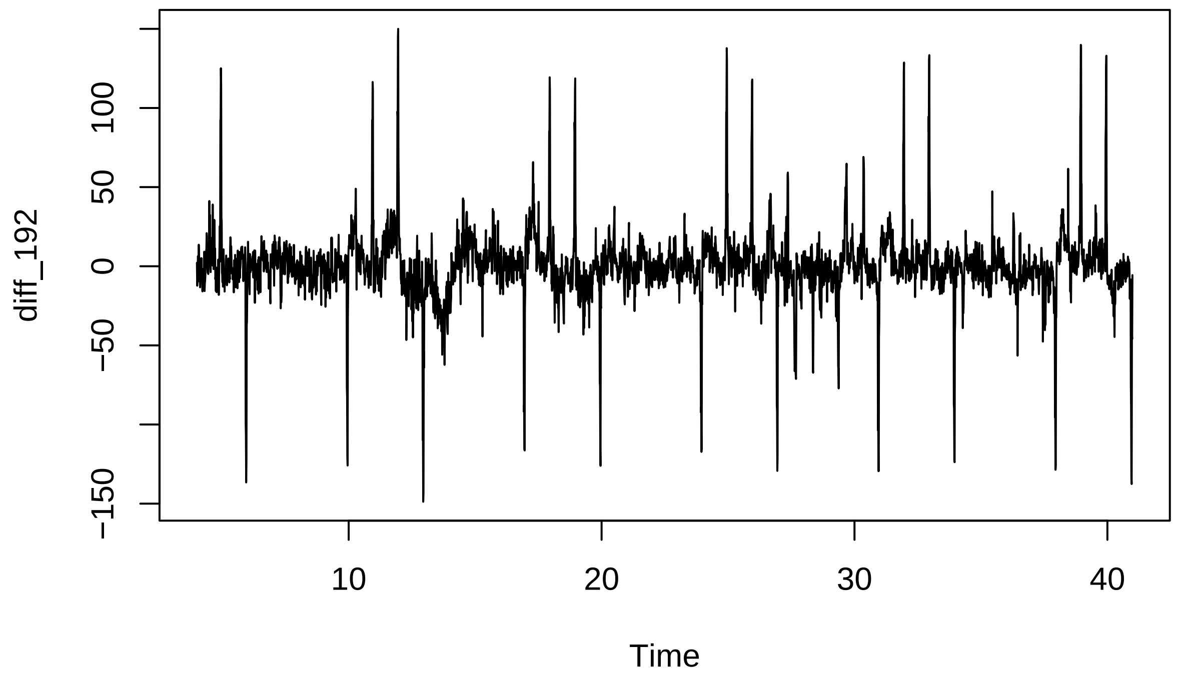
```
ggPacf(diff_96)
```



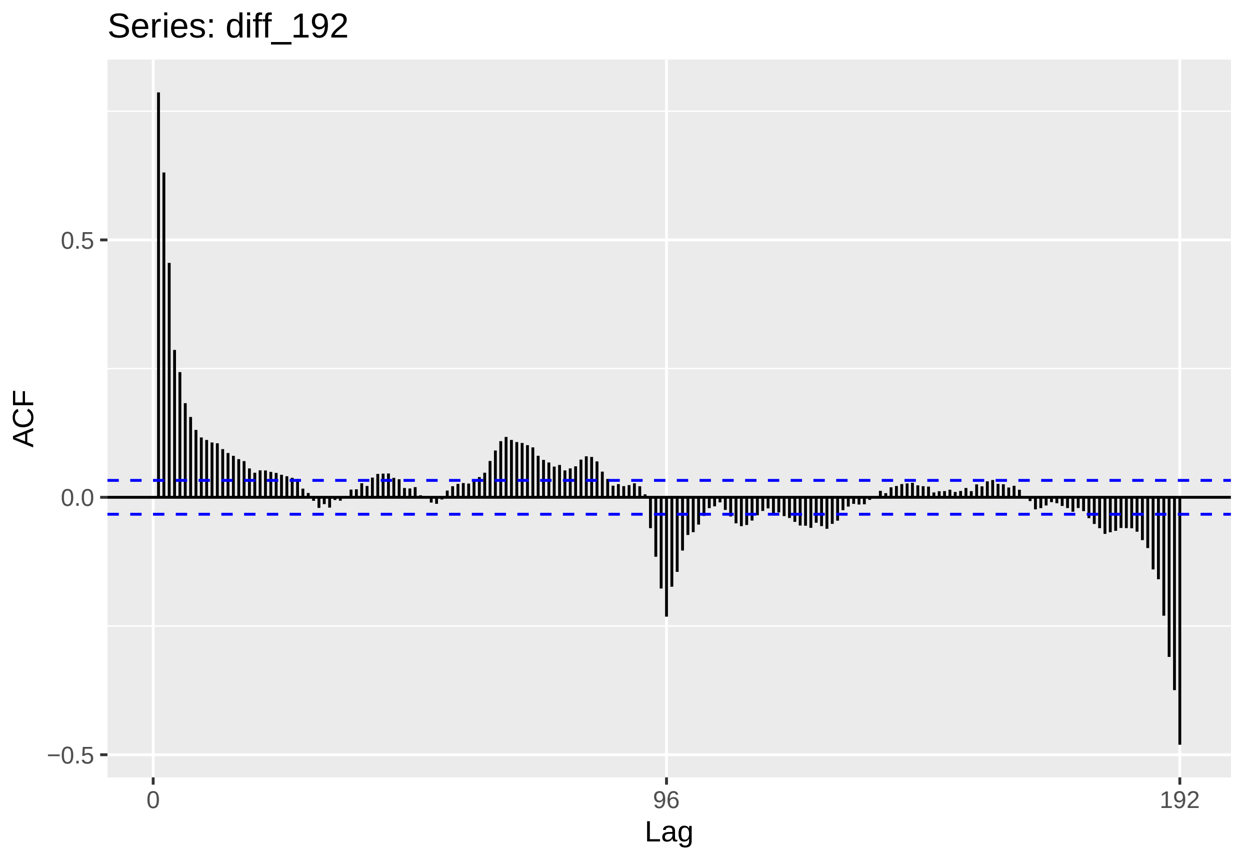
The ACF and PACF plots show multiple spikes beyond the significance bounds at various lags, indicating that significant autocorrelation remains in the residuals.

Let's try further diffrencing the series with lag=96*2

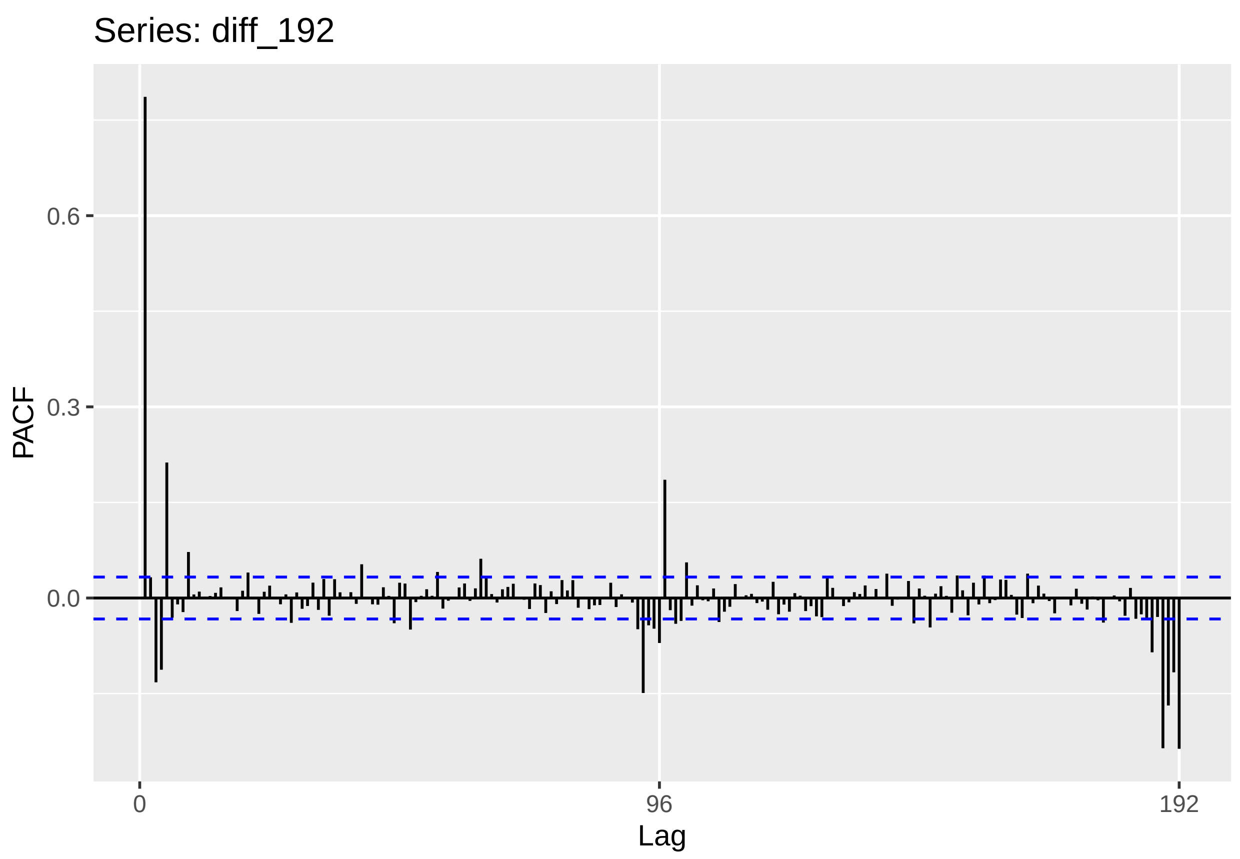
```r
diff_192= diff(diff_96, lag=96*2)   #differcing with lag=96
plot(diff_192)
```



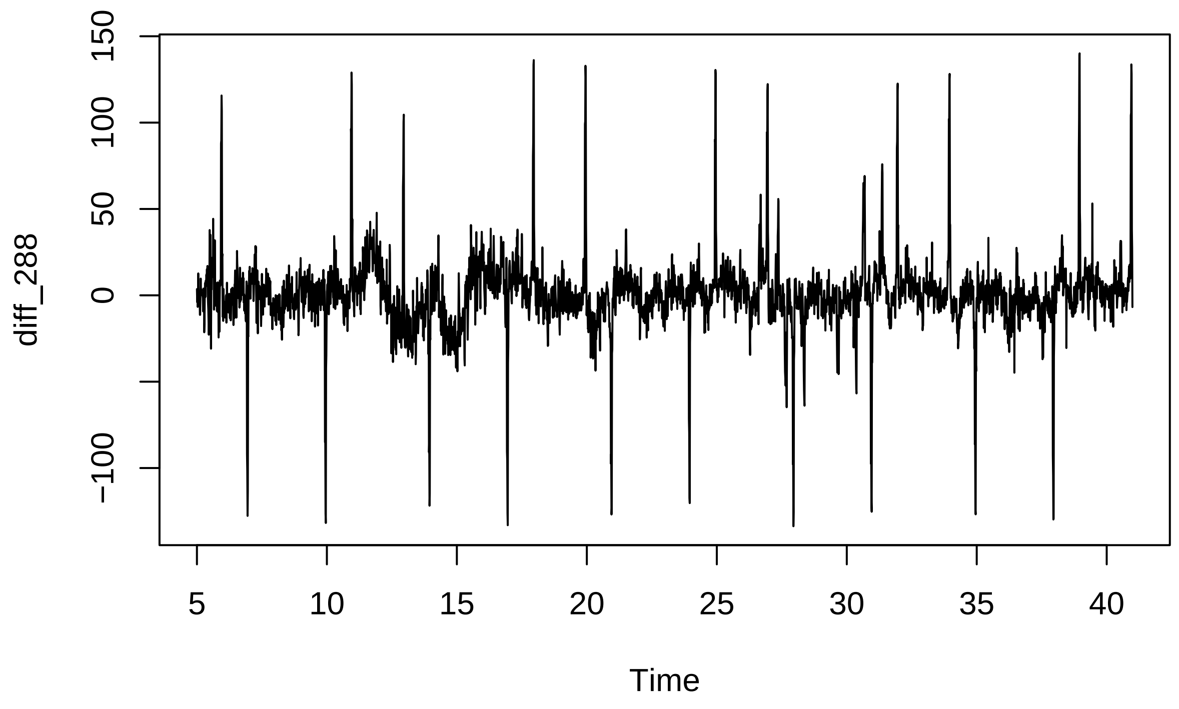
```r
ggAcf(diff_192)
```



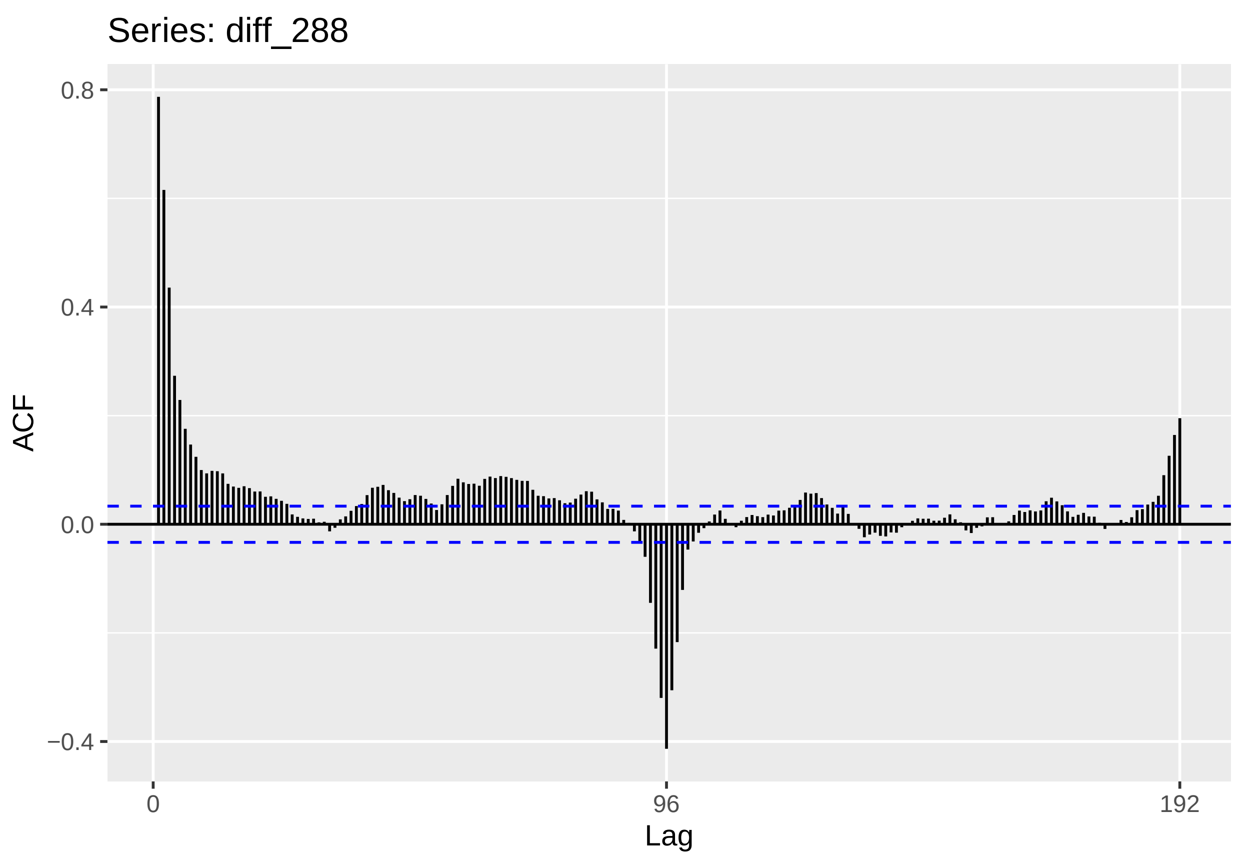
```
ggPacf(diff_192)
```



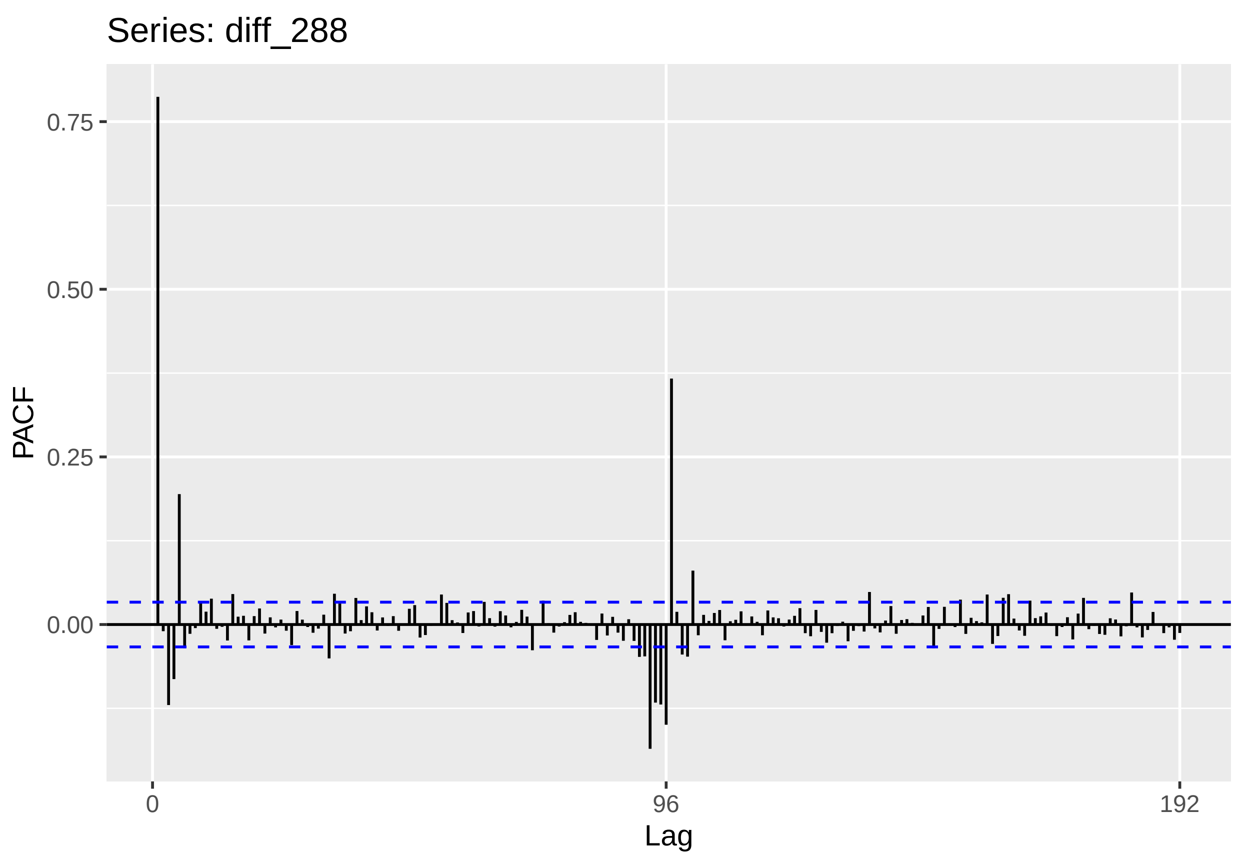
```
diff_288= diff(diff_96, lag=96*3)  #differcing with lag=96
plot(diff_288)
```



```r
ggAcf(diff_288)
```



```r
ggPacf(diff_288)
```



Once the residuals of the series have been made more independent.

Neural Network AutoRegression model

Now let's try to use a neural network model NNAR.

```
nnar_model=nnetar(train_set)
print(nnar_model)
```

```
## Series: train_set
## Model:  NNAR(20,1,11)[96]
## Call:   nnetar(y = train_set)
##
## Average of 20 networks, each of which is
## a 21-11-1 network with 254 weights
## options were - linear output units
##
## sigma^2 estimated as 46.53
```

For the forcasts:

```
consum_pred4 = forecast(nnar_model, h=96*10)
```

Let's evaluate the NNAR model

```r
RMSE_nnAR=sqrt(mean((consum_pred4$mean - test_set)^2))
print(RMSE_nnAR)
```

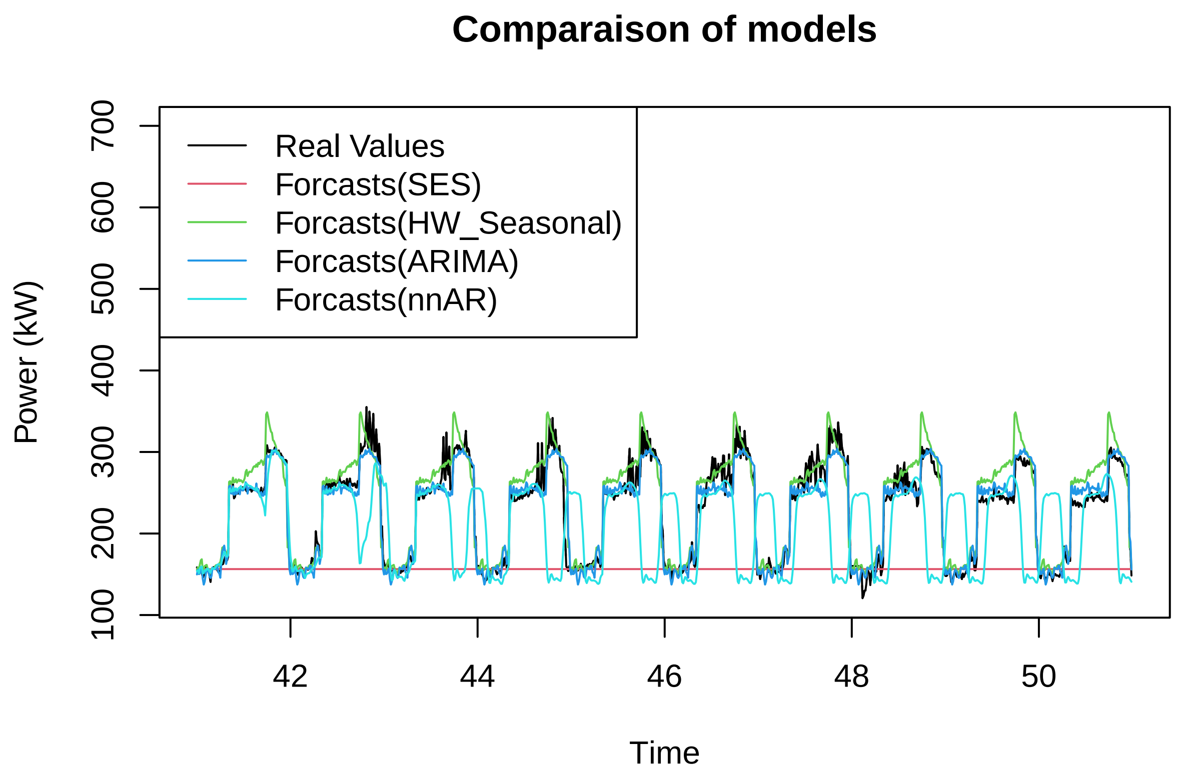
```
## [1] 72.60012
```

The RMSE is quite large, indicating that this neural network model may not be performing well on the data.

Comparaison and Choosing the best model

```r
# Plot the data and predictions
par(mfrow=c(1,1))
plot(test_set, xlim=c(41,51), ylim=c(120,700),main="Comparaison of models", ylab="Power (kW)", xlab="Ti
lines(test_set, lty=2)
lines(consum_pred1, col=2)
lines(consum_pred2, col=3)
lines(consum_pred3$mean, col=4)
lines(consum_pred4$mean, col=5)

# Add a legend
legend('topleft',
       col=1:5,
       lty=1,
       legend=c('Real Values',
                'Forcasts(SES)',
                'Forcasts(HW_Seasonal)',
                'Forcasts(ARIMA)',
                'Forcasts(nnAR)'))
```



Let's compare the RMSE of models,and find the lowest RMSE

```
min_rmse=min(RMSE_HW_Seasonal,RMSE_nnAR,RMSE_SES,RMSE_ARIMA)  #RMSE_SARIMA
cat("Minimum RMSE among the models is ARIMA:", min_rmse,  "\n")
```

```
## Minimum RMSE among the models is ARIMA: 15.7564
```

Since the ARIMA model achieves the lowest RMSE, it is the most suitable choice among the models evaluated, given the parameters we have previously determined.

Forcasts using the best model

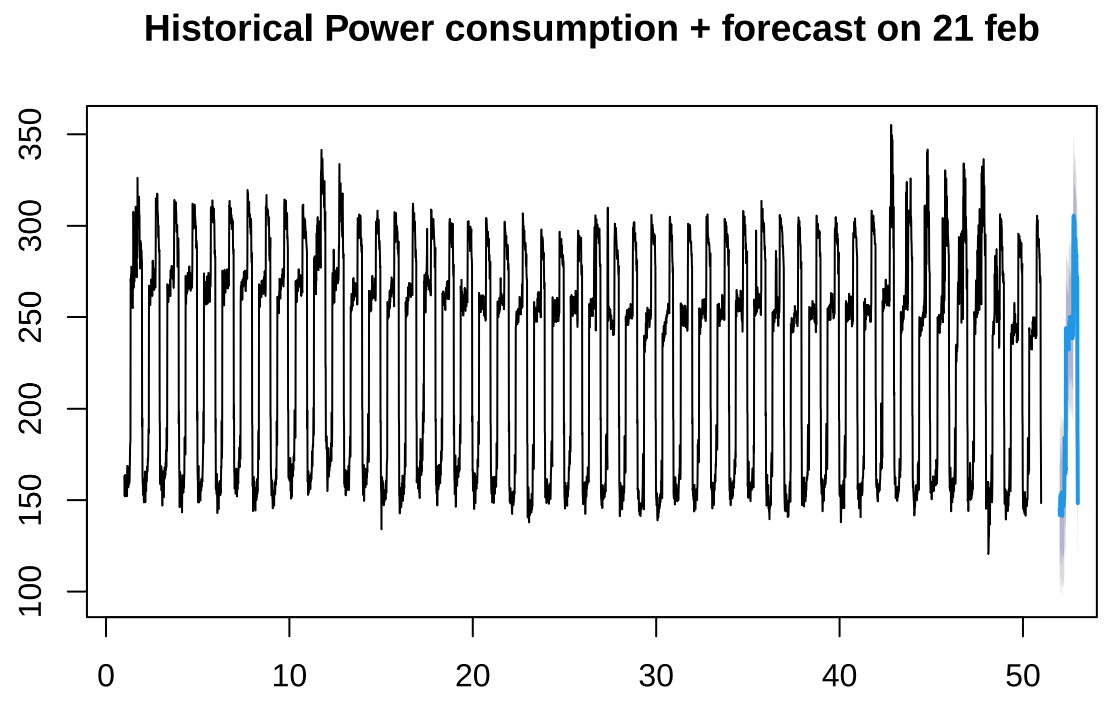
```
ARIMA=Arima(ts_elect, order=c(1,0,0), seasonal=c(0,1,0))
#ts_elect[1:(length(ts_elect) - 96)
summary(ARIMA)
```

```
## Series: ts_elect
## ARIMA(1,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1
##       0.7159
## s.e.  0.0102
```

```
##
## sigma^2 = 128.7:  log likelihood = -18099.21
## AIC=36202.42   AICc=36202.43   BIC=36215.34
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -0.1146769 11.22912 6.405882 -0.1680132 2.891475 0.744182
##                     ACF1
## Training set -0.07073442
```

```
forcasts_elect_feb21=forecast(ARIMA, h=96)

plot(forcasts_elect_feb21, main="Historical Power consumption + forecast on 21 feb")
```



```
#write_xlsx(data.frame(Power_forcast_without_Temp =round(as.numeric(forcasts_elect_feb21$mean),1)), pat
```

Forcasts using outdoor temperature:

We will now incorporate Temperature as a covariate in our Electricity Consumption Forecasting.

Data Preperation

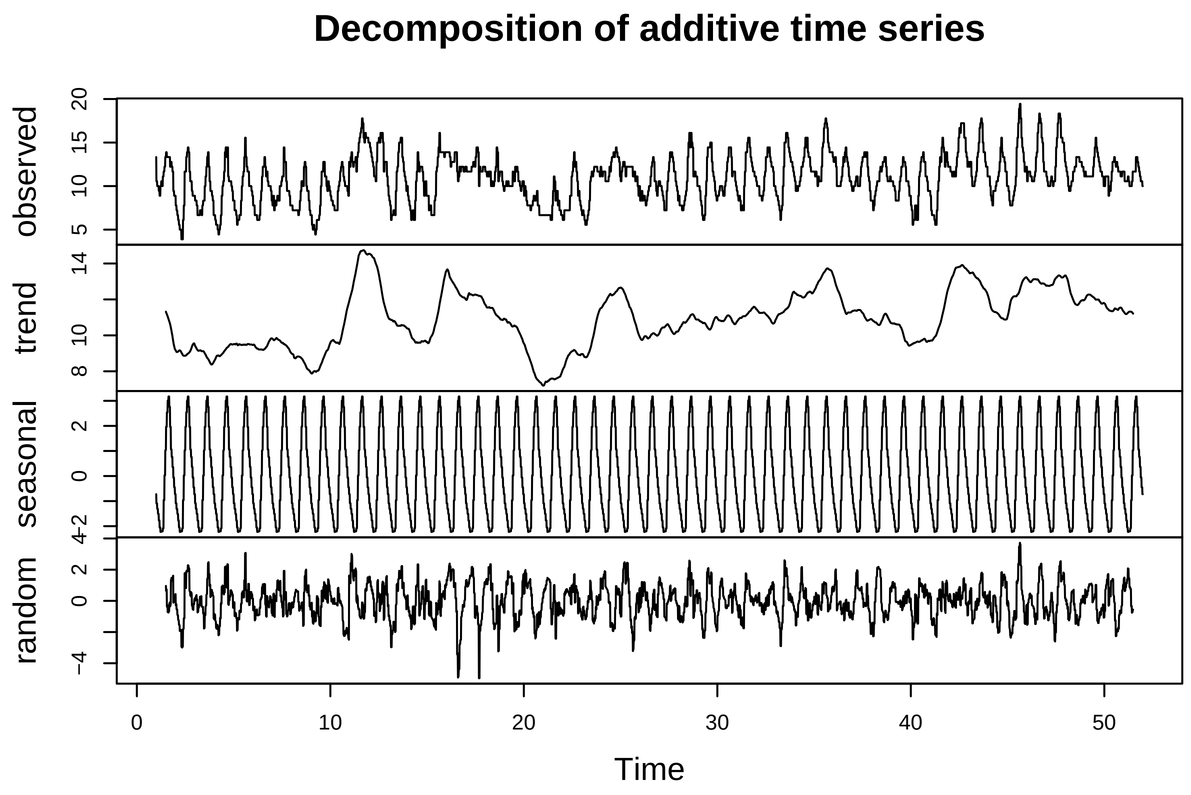
```
ts_power=ts(dataset_elect_balanced$`Power (kW)`, start=c(1,1), end=c(51,96), freq=96)
ts_temp=ts(dataset_elect_balanced$`Temp (C°)`, start=c(1,1), end=c(51,96), freq=96)
head(ts_elect)
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 96
## [1] 163.1 154.4 152.2 158.7 163.8 158.7
```

```
head(ts_temp)
```

```
## Time Series:
## Start = c(1, 1)
## End = c(1, 6)
## Frequency = 96
## [1] 13.33333 10.55556 10.55556 10.55556 10.55556 10.00000
```

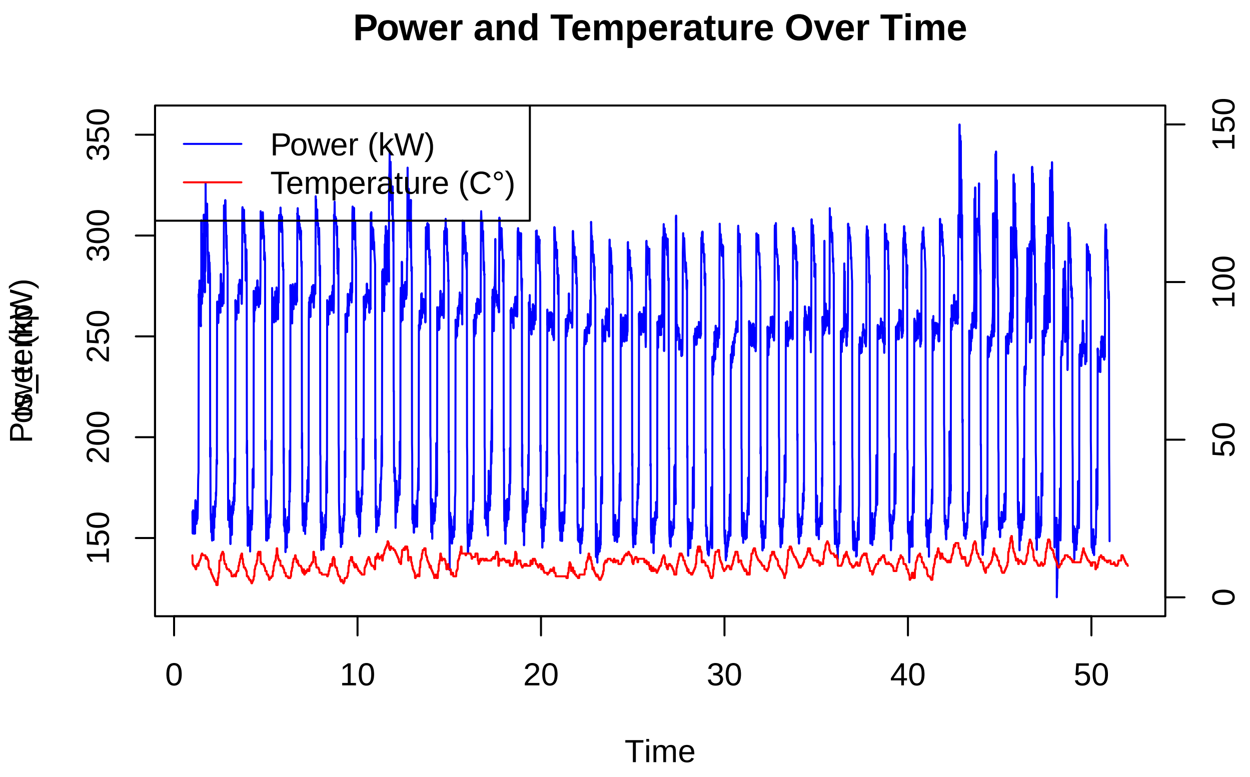
```
plot(decompose(ts_temp, type="additive"))
```



```
# Plot the first time series (Power)
plot(ts_power, type="l", col="blue", ylab="Power (kW)", xlab="Time", main="Power and Temperature Over Ti

# Add the second time series (Temperature) on a new axis
par(new=TRUE)
plot(ts_temp, type="l", col="red", axes=FALSE, xlab="",ylim=c(0, 150))
axis(side=4)
mtext("Temperature (C°)", side=4, line=3)

# Add a legend
legend("topleft", legend=c("Power (kW)", "Temperature (C°)"), col=c("blue", "red"), lty=1)
```



We will now proceed with the Train/Test split, following the previously explained 80/20 rule.

```
#train/test of power time series
power_train_set=window(ts_power, start=c(1,1), end=c(40,96))
power_test_set= window(ts_power, start=c(41,1), end=c(50,96))

#train/test of temperature time series
temp_train_set=window(ts_temp, start=c(1,1), end=c(40,96))
temp_test_set=window(ts_temp, start=c(41,1), end=c(50,96))
```

Now, let's analyze the relationship between these variables by performing two linear regression models and comapare their BIC to select the best model:

Simple Linear Regression:

First, we'll apply a straightforward linear regression between the Power and Temperature time series, ignoring any trends or seasonality. This will give us a basic understanding of the direct relationship between these variables.

```
SLR_model = tslm(power_train_set ~ temp_train_set)
summary(SLR_model)
```

```
##
## Call:
## tslm(formula = power_train_set ~ temp_train_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -121.546  -42.088    2.154   43.064  111.969
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    126.0885     3.4178   36.89   <2e-16 ***
## temp_train_set   9.9474     0.3137   31.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50.83 on 3838 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.2074
## F-statistic:  1006 on 1 and 3838 DF,  p-value: < 2.2e-16
```

The model shows a significant positive relationship between Temperature and Power consumption, with each degree increase in temperature leading to an estimated 9.95 kW rise in power. However, the model explains only about 20.8% of the variance in Power, indicating other factors are also at play.

Advanced Linear Regression with Tendency and Seasonality:

Next, we'll refine our approach by accounting for potential trends and seasonality in the data. We'll detrend and deseasonalize both time series before applying the linear regression. This will allow us to capture the true relationship between Power and Temperature, isolated from underlying patterns in the data.

```
# Create a season factor variable for both train and test sets
season_train = factor(cycle(temp_train_set))
season_test = factor(cycle(temp_test_set))

# Create a trend variable for both train and test sets
trend_train = seq_along(temp_train_set)
trend_test = seq_along(temp_test_set) + length(temp_train_set)

ALR_model =tslm(power_train_set ~ temp_train_set + season_train + trend_train)
summary(ALR_model)
```

```
##
## Call:
## tslm(formula = power_train_set ~ temp_train_set + season_train +
##     trend_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -112.760   -4.613    0.255    4.592   58.295
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.567e+02  2.068e+00  75.763  < 2e-16 ***
## temp_train_set  1.031e+00  9.825e-02  10.489  < 2e-16 ***
## season_train2  -5.474e-01  2.594e+00  -0.211  0.83286
## season_train3  -6.708e+00  2.594e+00  -2.586  0.00974 **
## season_train4  -3.707e-01  2.594e+00  -0.143  0.88636
## season_train5   2.829e+00  2.594e+00   1.091  0.27548
## season_train6   3.267e+00  2.594e+00   1.260  0.20791
## season_train7  -6.320e+00  2.594e+00  -2.436  0.01488 *
## season_train8  -6.446e+00  2.594e+00  -2.485  0.01301 *
## season_train9  -3.086e+00  2.594e+00  -1.190  0.23425
## season_train10 -3.832e+00  2.595e+00  -1.477  0.13981
## season_train11 -1.050e+00  2.595e+00  -0.405  0.68584
## season_train12 -1.848e+00  2.595e+00  -0.712  0.47647
## season_train13 -3.504e-01  2.595e+00  -0.135  0.89258
## season_train14 -4.914e+00  2.595e+00  -1.893  0.05837 .
## season_train15 -6.317e+00  2.595e+00  -2.434  0.01498 *
## season_train16 -7.451e-01  2.595e+00  -0.287  0.77405
## season_train17  7.870e-01  2.595e+00   0.303  0.76174
## season_train18 -1.979e-01  2.597e+00  -0.076  0.93926
## season_train19 -1.146e+00  2.597e+00  -0.441  0.65906
## season_train20  4.788e-01  2.597e+00   0.184  0.85371
## season_train21  2.784e-01  2.597e+00   0.107  0.91462
## season_train22  1.230e+00  2.597e+00   0.474  0.63574
## season_train23  2.233e+00  2.597e+00   0.859  0.39012
## season_train24  4.215e+00  2.597e+00   1.623  0.10476
## season_train25  3.574e+00  2.597e+00   1.376  0.16890
## season_train26  7.148e+00  2.598e+00   2.752  0.00595 **
## season_train27  1.421e+01  2.598e+00   5.469 4.83e-08 ***
## season_train28  1.643e+01  2.598e+00   6.327 2.80e-10 ***
## season_train29  1.690e+01  2.598e+00   6.506 8.75e-11 ***
## season_train30  2.234e+01  2.598e+00   8.600  < 2e-16 ***
## season_train31  2.184e+01  2.598e+00   8.406  < 2e-16 ***
## season_train32  1.641e+01  2.598e+00   6.316 2.99e-10 ***
## season_train33  2.006e+01  2.598e+00   7.721 1.47e-14 ***
## season_train34  1.041e+02  2.597e+00  40.062  < 2e-16 ***
## season_train35  1.021e+02  2.597e+00  39.306  < 2e-16 ***
## season_train36  9.912e+01  2.597e+00  38.163  < 2e-16 ***
## season_train37  9.884e+01  2.597e+00  38.054  < 2e-16 ***
## season_train38  1.015e+02  2.593e+00  39.139  < 2e-16 ***
## season_train39  9.649e+01  2.593e+00  37.203  < 2e-16 ***
## season_train40  9.896e+01  2.593e+00  38.159  < 2e-16 ***
## season_train41  9.988e+01  2.593e+00  38.512  < 2e-16 ***
## season_train42  9.605e+01  2.594e+00  37.020  < 2e-16 ***
## season_train43  9.686e+01  2.594e+00  37.331  < 2e-16 ***
## season_train44  9.845e+01  2.594e+00  37.947  < 2e-16 ***
## season_train45  9.863e+01  2.594e+00  38.014  < 2e-16 ***
## season_train46  1.007e+02  2.599e+00  38.738  < 2e-16 ***
## season_train47  9.885e+01  2.599e+00  38.036  < 2e-16 ***
## season_train48  9.938e+01  2.599e+00  38.239  < 2e-16 ***
## season_train49  1.001e+02  2.599e+00  38.508  < 2e-16 ***
## season_train50  9.980e+01  2.606e+00  38.292  < 2e-16 ***
## season_train51  1.032e+02  2.606e+00  39.603  < 2e-16 ***
## season_train52  1.024e+02  2.606e+00  39.292  < 2e-16 ***
## season_train53  1.005e+02  2.606e+00  38.576  < 2e-16 ***
## season_train54  1.021e+02  2.612e+00  39.104  < 2e-16 ***
## season_train55  1.025e+02  2.612e+00  39.262  < 2e-16 ***
## season_train56  1.018e+02  2.612e+00  38.980  < 2e-16 ***
## season_train57  1.011e+02  2.612e+00  38.712  < 2e-16 ***
## season_train58  1.013e+02  2.617e+00  38.716  < 2e-16 ***
## season_train59  1.015e+02  2.617e+00  38.775  < 2e-16 ***
## season_train60  1.012e+02  2.617e+00  38.689  < 2e-16 ***
## season_train61  1.010e+02  2.617e+00  38.609  < 2e-16 ***
## season_train62  1.010e+02  2.618e+00  38.604  < 2e-16 ***
## season_train63  1.003e+02  2.618e+00  38.336  < 2e-16 ***
## season_train64  1.004e+02  2.618e+00  38.375  < 2e-16 ***
## season_train65  1.003e+02  2.618e+00  38.320  < 2e-16 ***
## season_train66  1.005e+02  2.612e+00  38.459  < 2e-16 ***
## season_train67  1.010e+02  2.612e+00  38.683  < 2e-16 ***
## season_train68  9.851e+01  2.612e+00  37.710  < 2e-16 ***
## season_train69  9.799e+01  2.612e+00  37.512  < 2e-16 ***
## season_train70  1.187e+02  2.604e+00  45.582  < 2e-16 ***
## season_train71  1.327e+02  2.604e+00  50.959  < 2e-16 ***
## season_train72  1.450e+02  2.604e+00  55.680  < 2e-16 ***
## season_train73  1.445e+02  2.604e+00  55.474  < 2e-16 ***
## season_train74  1.422e+02  2.598e+00  54.727  < 2e-16 ***
## season_train75  1.403e+02  2.598e+00  53.999  < 2e-16 ***
## season_train76  1.391e+02  2.598e+00  53.549  < 2e-16 ***
## season_train77  1.404e+02  2.598e+00  54.056  < 2e-16 ***
## season_train78  1.455e+02  2.596e+00  56.040  < 2e-16 ***
## season_train79  1.412e+02  2.596e+00  54.392  < 2e-16 ***
## season_train80  1.409e+02  2.596e+00  54.271  < 2e-16 ***
## season_train81  1.389e+02  2.596e+00  53.504  < 2e-16 ***
## season_train82  1.386e+02  2.595e+00  53.402  < 2e-16 ***
## season_train83  1.370e+02  2.595e+00  52.805  < 2e-16 ***
## season_train84  1.364e+02  2.595e+00  52.565  < 2e-16 ***
## season_train85  1.357e+02  2.595e+00  52.312  < 2e-16 ***
## season_train86  1.343e+02  2.594e+00  51.794  < 2e-16 ***
## season_train87  1.323e+02  2.594e+00  51.001  < 2e-16 ***
## season_train88  1.319e+02  2.594e+00  50.833  < 2e-16 ***
## season_train89  1.299e+02  2.594e+00  50.078  < 2e-16 ***
## season_train90  1.120e+02  2.593e+00  43.204  < 2e-16 ***
## season_train91  1.108e+02  2.593e+00  42.729  < 2e-16 ***
## season_train92  1.052e+02  2.593e+00  40.569  < 2e-16 ***
## season_train93  1.063e+02  2.593e+00  40.976  < 2e-16 ***
## season_train94  3.042e+01  2.593e+00  11.729  < 2e-16 ***
## season_train95  3.244e+01  2.593e+00  12.509  < 2e-16 ***
## season_train96  3.219e+00  2.593e+00   1.241  0.21461
## trend_train    -4.592e-03  1.736e-04 -26.454  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 11.6 on 3742 degrees of freedom
## Multiple R-squared:  0.9598, Adjusted R-squared:  0.9587
## F-statistic: 920.6 on 97 and 3742 DF,  p-value: < 2.2e-16
```

This regression model includes Temperature, seasonality, and trend as predictors of Power consumption. The ALR model explains approximately 96% of the variance in Power ($R^2 = 0.9598$), indicating a strong fit.

Temperature has a positive and significant impact on Power consumption. Trend shows a slight but significant negative impact, indicating a gradual decrease over time. Many seasonal factors are significant, with varying impacts, reflecting the influence of different time periods on Power usage. Overall, the model effectively captures the relationship between Power and the predictors, accounting for both seasonal patterns and trends.

Let's proceed by applying cross-validation to evaluate both linear regression models. This will help us select the better model by determining which one achieves a more effective balance between fit and complexity.

```r
print("results of Cross Validation of Simple Linear Model without Tendency and Seasonality:")
```

```
## [1] "results of Cross Validation of Simple Linear Model without Tendency and Seasonality:"
```

```r
CV(SLR_model)
```

```
##            CV          AIC         AICc          BIC        AdjR2
## 2.584611e+03 3.017475e+04 3.017476e+04 3.019351e+04 2.073967e-01
```

```r
print("results of Cross Validation of Advanced Linear Regression with Tendency and Seasonality:")
```

```
## [1] "results of Cross Validation of Advanced Linear Regression with Tendency and Seasonality:"
```

```r
CV(ALR_model)
```

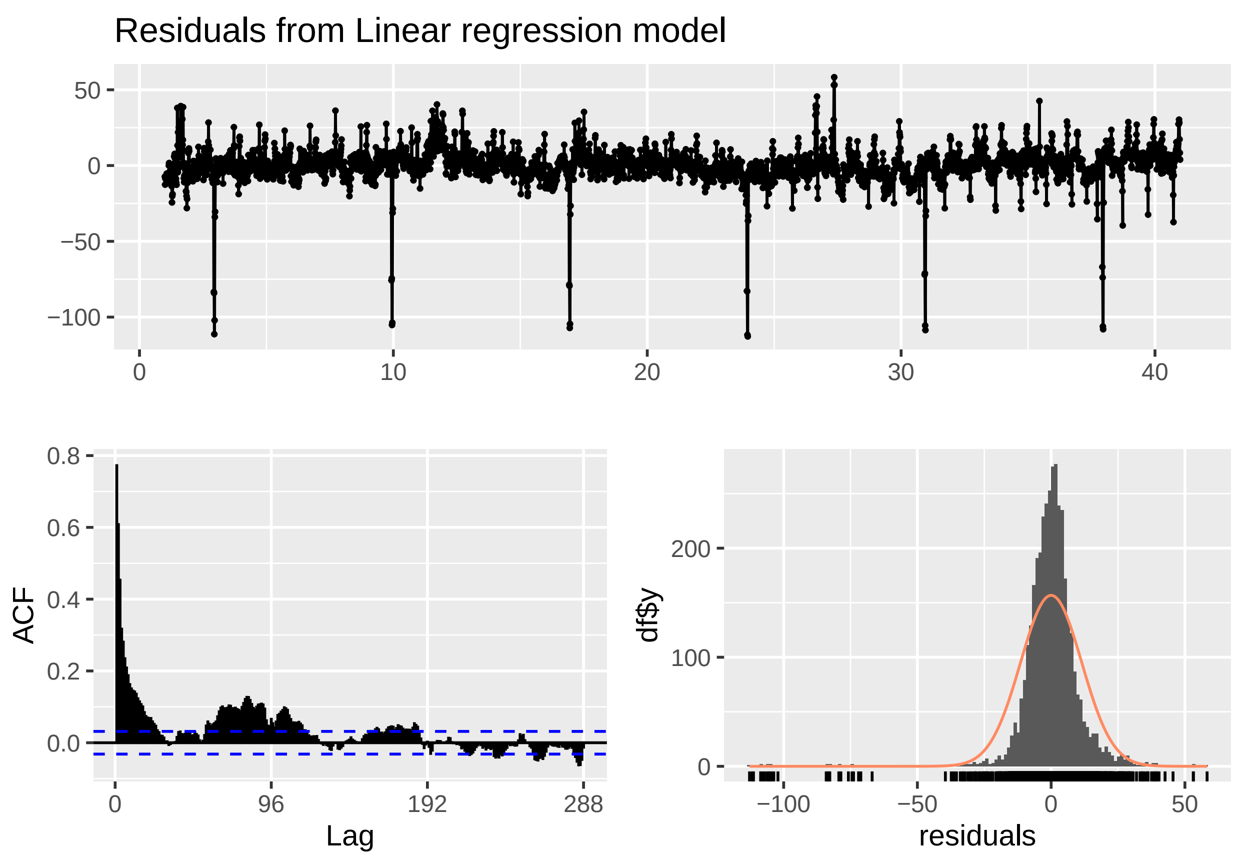
```
##            CV          AIC         AICc          BIC        AdjR2
## 1.380308e+02 1.892094e+04 1.892623e+04 1.954001e+04 9.587367e-01
```

As the ALR_model presents the lowest BIC , it is better.

Residuals Analysis

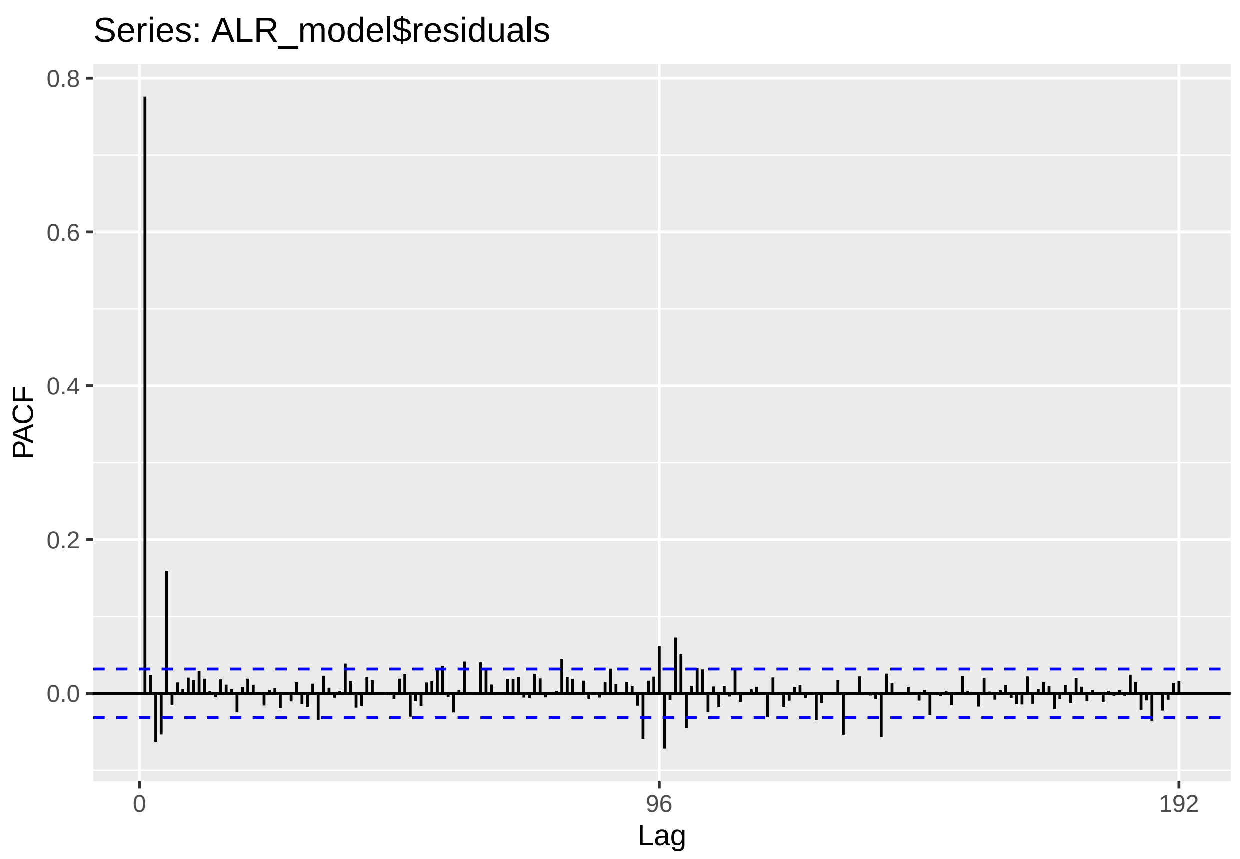
Now, we analyze the residuals,

```r
checkresiduals(ALR_model, test='LB', plot=TRUE)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 8765.4, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
ggPacf(ALR_model$residuals)
```



The Ljung-Box test results indicate that the residuals from the linear regression model exhibit significant autocorrelation (Q* = 8765.4, p-value < 2.2e-16). This suggests that the residuals are not independent, likely due to autocorrelation. To address this, we need to ensure stationarity in the time series data.

Let's design a model for residuals using SARIMA.

We use the Augmented Dickey-Fuller (ADF) test on the residuals of to test for stationarity

```
adf.test(ALR_model$residuals)
```

```
## Warning in adf.test(ALR_model$residuals): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ALR_model$residuals
## Dickey-Fuller = -11.105, Lag order = 15, p-value = 0.01
## alternative hypothesis: stationary
```

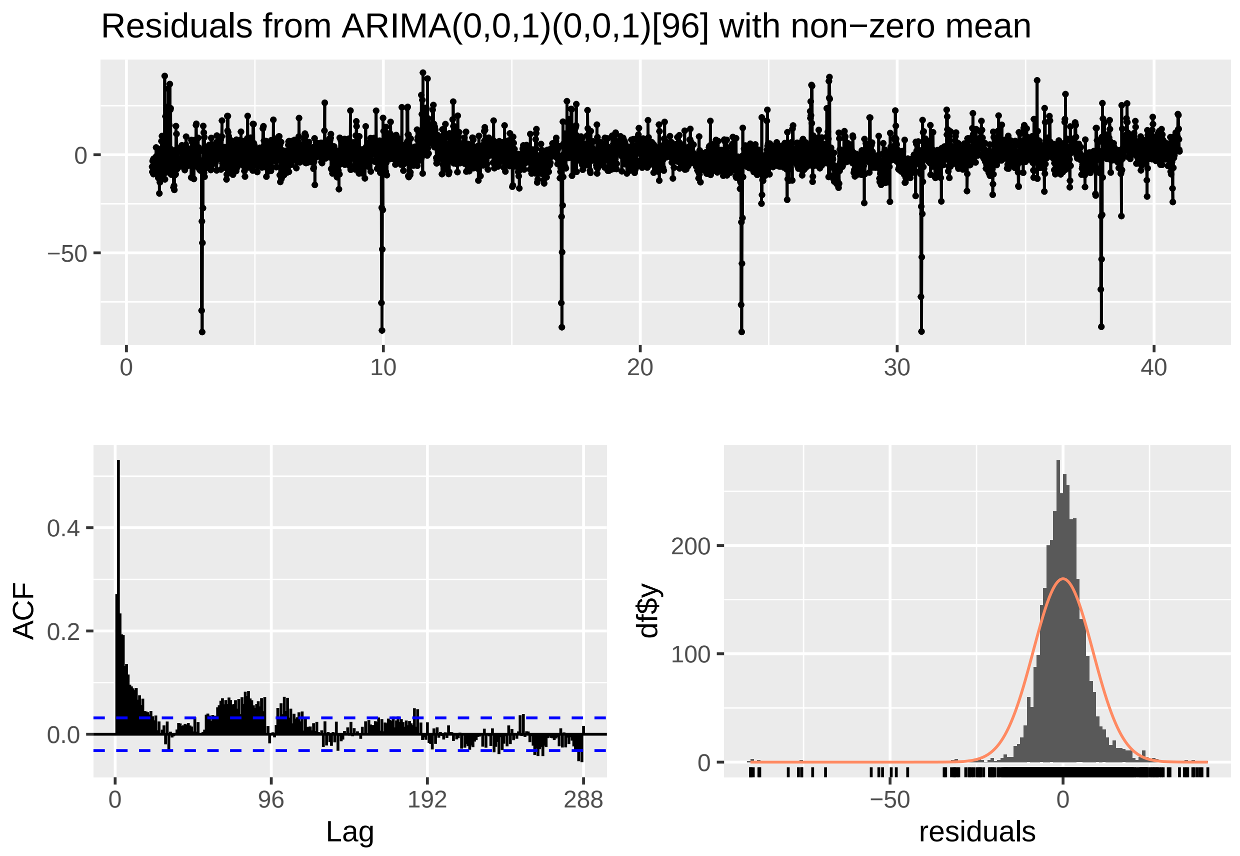
As the test shows that the residuals are stationary.

```
SARIMA_residuals_model <- auto.arima(ALR_model$residuals, seasonal = TRUE)
summary(SARIMA_residuals_model)
```

```
## Series: ALR_model$residuals
## ARIMA(0,0,1)(0,0,1)[96] with non-zero mean
```

```
##
## Coefficients:
##          ma1    sma1    mean
##       0.6412  0.0758  0.0010
## s.e.  0.0105  0.0161  0.2448
##
## sigma^2 = 74.17:  log likelihood = -13716.08
## AIC=27440.15   AICc=27440.16   BIC=27465.16
##
## Training set error measures:
##                       ME    RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 0.005472364 8.60906 5.324612 89.00729 230.8342 0.6775324 0.2716677
```

```r
checkresiduals(SARIMA_residuals_model, test='LB', plot=TRUE)
```

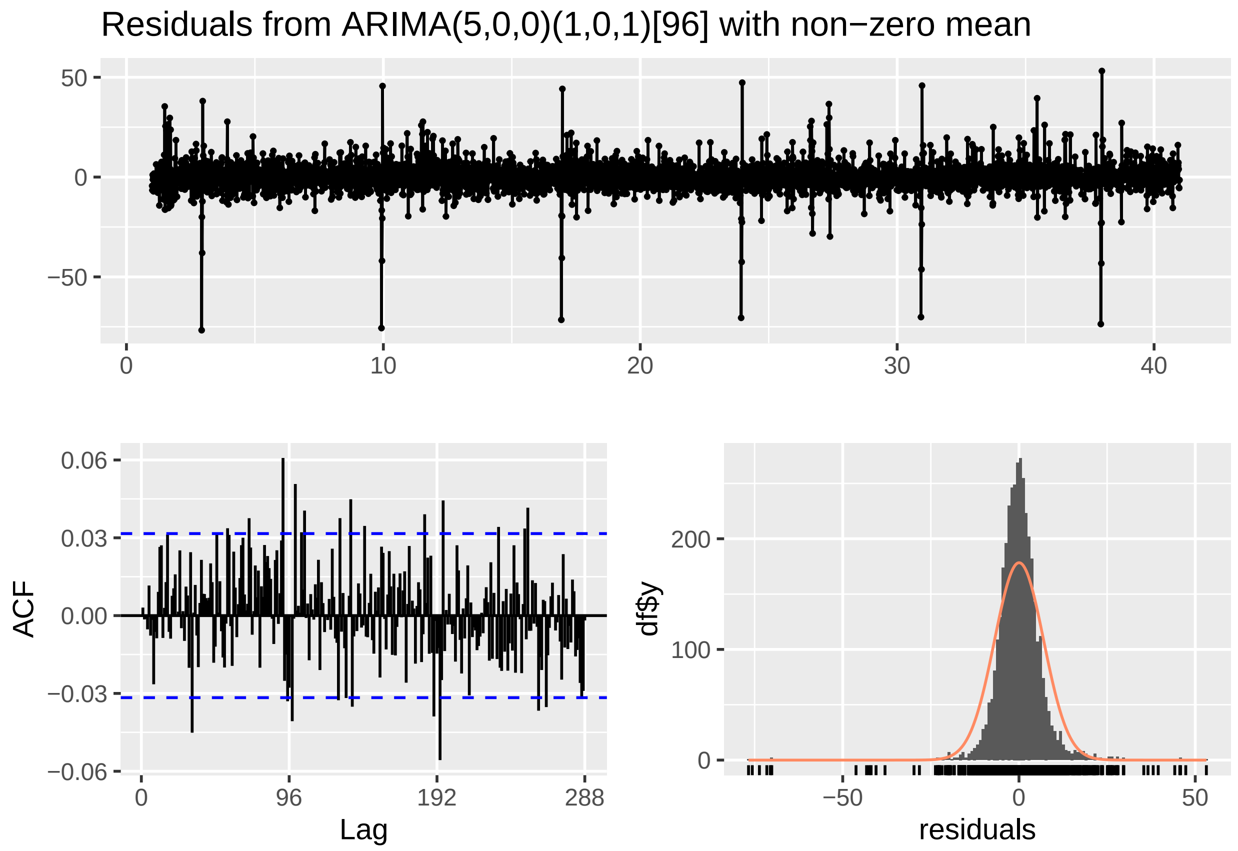


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1)(0,0,1)[96] with non-zero mean
## Q* = 3134.3, df = 190, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 192
```

```r
SARIMA_residuals_model=Arima(ALR_model$residuals,order=c(5,0,0),seasonal=c(1, 0, 1))
summary(SARIMA_residuals_model)
```

```
## Series: ALR_model$residuals
## ARIMA(5,0,0)(1,0,1)[96] with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3      ar4    ar5    sar1     sma1    mean
##       0.7675  0.1096  -0.0532  -0.2015  0.172  0.8767  -0.7761  0.0154
## s.e.  0.0158  0.0208   0.0214   0.0206  0.016  0.0198   0.0228  0.9132
##
## sigma^2 = 48.7:  log likelihood = -12910.88
## AIC=25839.77   AICc=25839.81   BIC=25896.05
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.02753532 6.971408 4.433183 82.53718 263.7721 0.5641022
##                     ACF1
## Training set 0.003101828
```

```r
checkresiduals(SARIMA_residuals_model, test='LB', plot=TRUE)
```



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

```
##
## data:  Residuals from ARIMA(5,0,0)(1,0,1)[96] with non-zero mean
## Q* = 248.15, df = 185, p-value = 0.001334
##
## Model df: 7.   Total lags used: 192
```

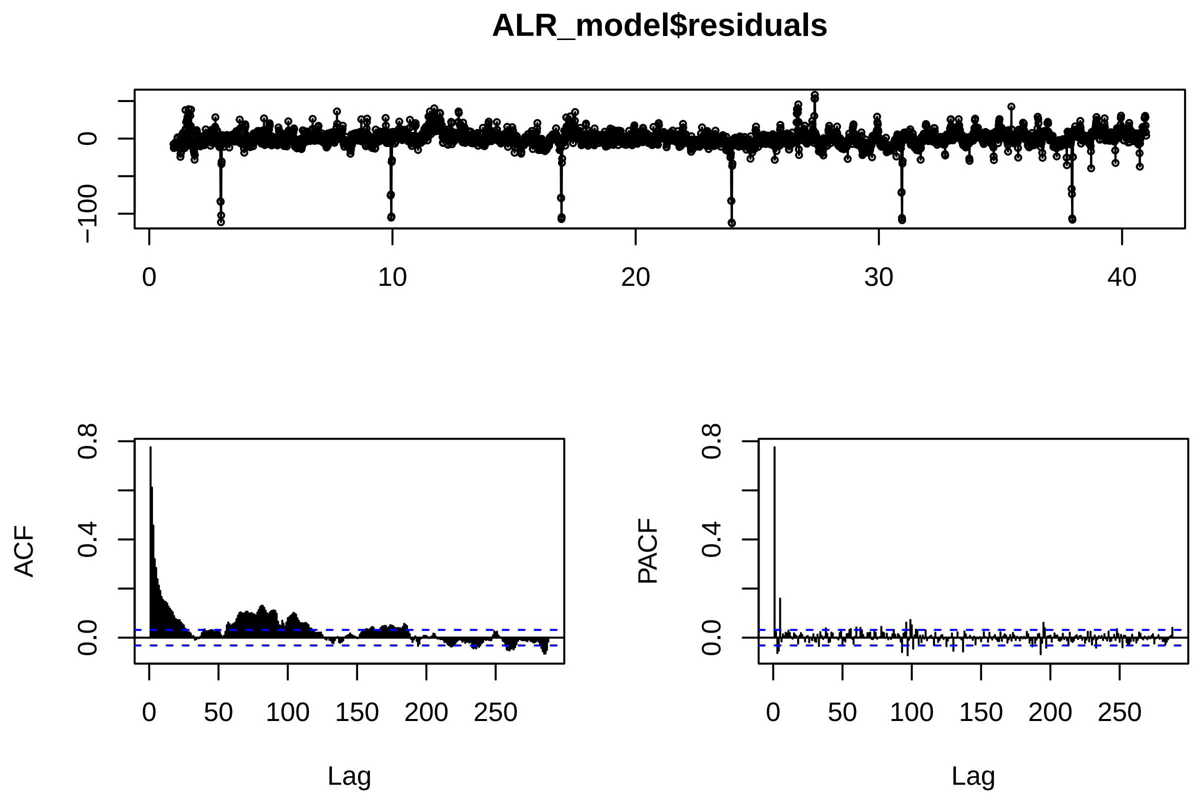
Dynamic Regression Model

Now let's try a Dynamic regression model based on the best SARIMA (5, 0, 0)(0, 1, 1). I've test alot and I choosed this one

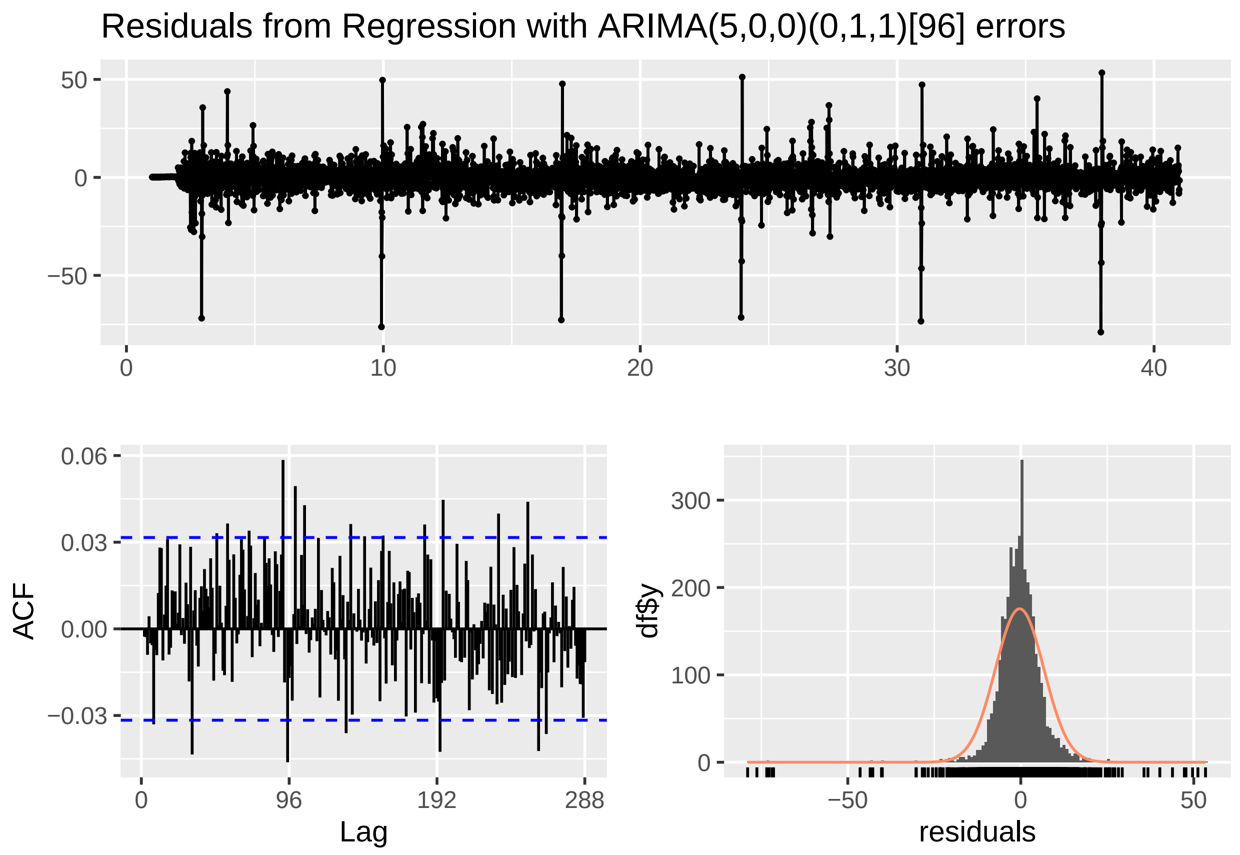
```
DR_model <- Arima(y = power_train_set,
                  xreg = temp_train_set,
                  order = c(5, 0, 0),
                  seasonal = list(order = c(0, 1, 1)))
summary(DR_model)
```

```
## Series: power_train_set
## Regression with ARIMA(5,0,0)(0,1,1)[96] errors
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5     sma1    xreg
##       0.7685  0.1292  -0.0630  -0.2097  0.1808  -0.8395  0.4492
## s.e.  0.0161  0.0201   0.0202   0.0202  0.0161   0.0093  0.2176
##
## sigma^2 = 51:  log likelihood = -12728.46
## AIC=25472.91   AICc=25472.95   BIC=25522.73
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE    MAPE      MASE
## Training set -0.3293857 7.044936 4.467961 -0.2306689 2.07136 0.5671813
##                       ACF1
## Training set -1.921898e-05
```

```
tsdisplay(ALR_model$residuals)
```

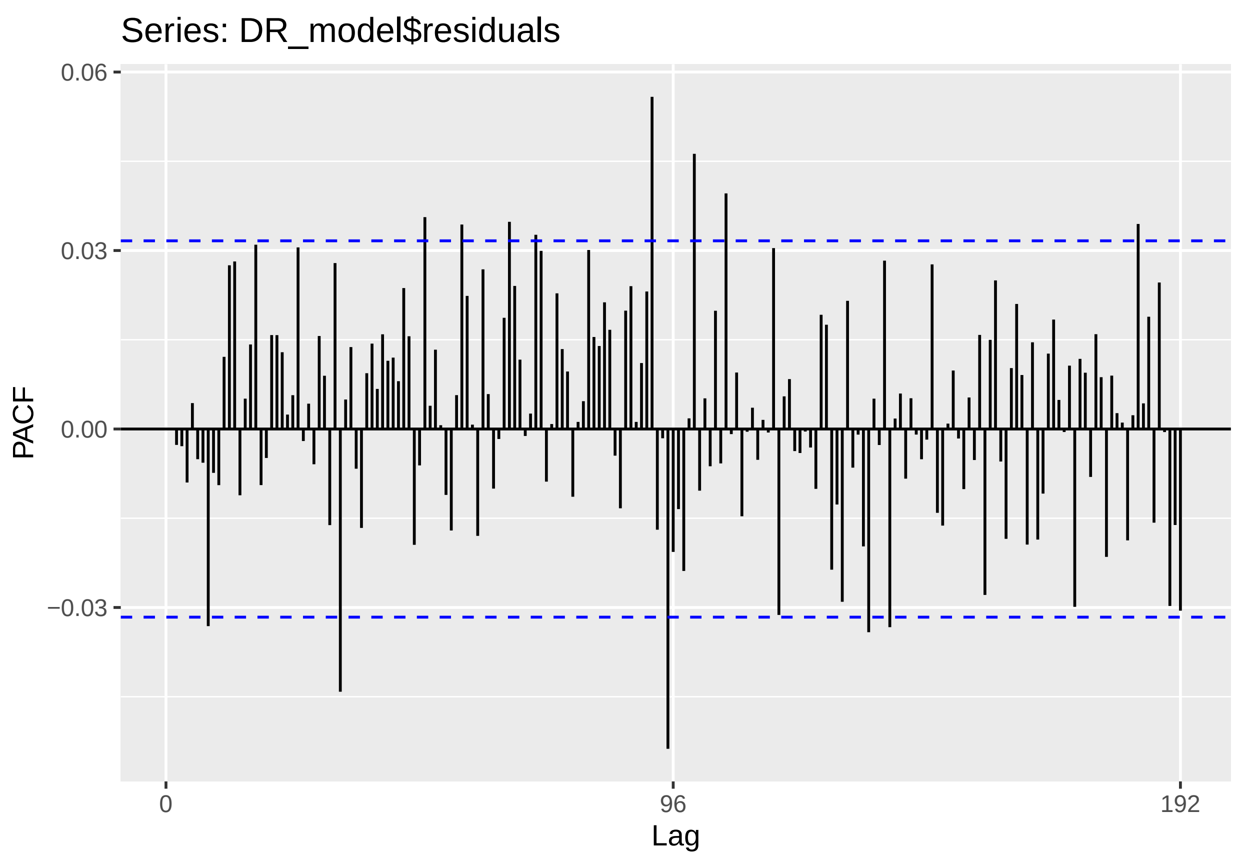


```
checkresiduals(DR_model,test='LB', plot=TRUE)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,1)[96] errors
## Q* = 245.23, df = 186, p-value = 0.002338
##
## Model df: 6.   Total lags used: 192
```

```
ggPacf(DR_model$residuals)
```

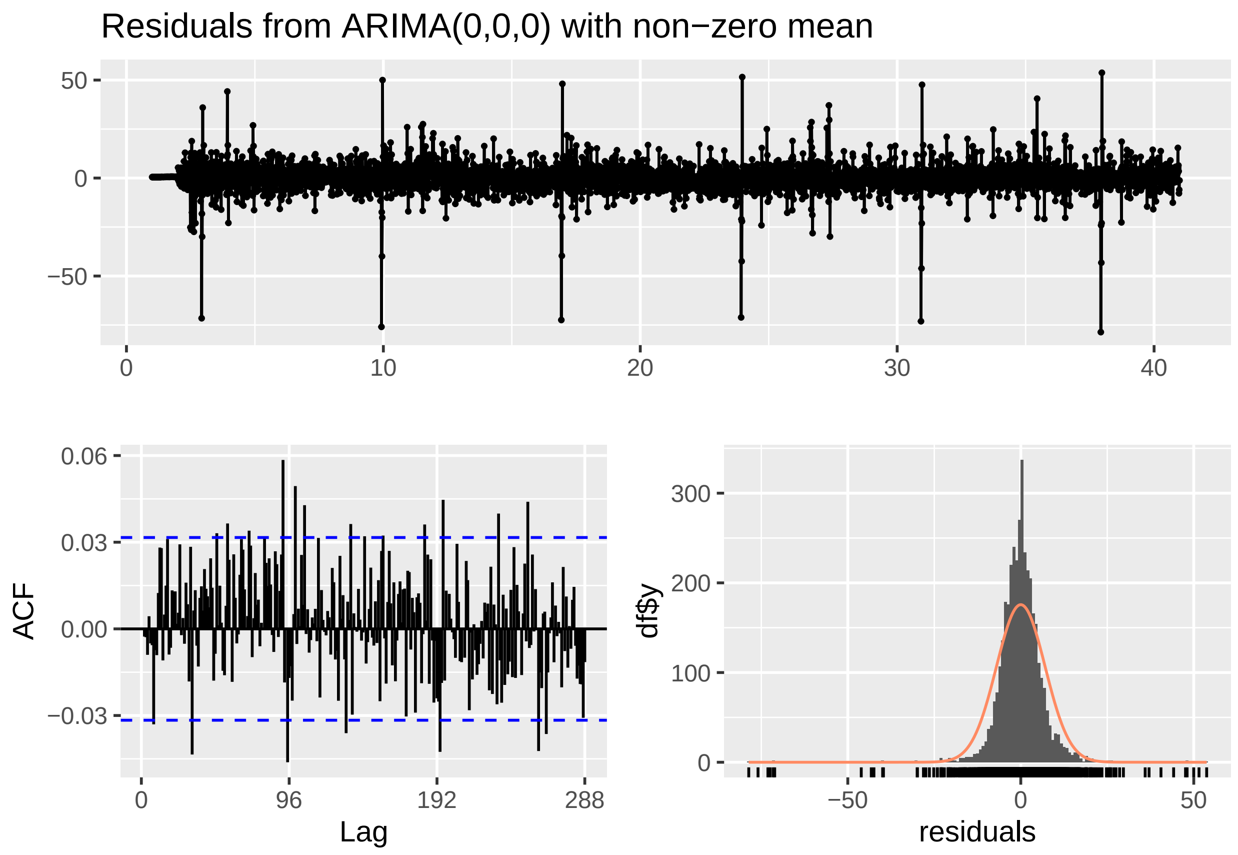


let's see if we can ameloirate the residuals analysis of the dyanmic model by designing an ARIMA mdoel

```
SARIMA_residuals_DR <- auto.arima(DR_model$residuals, seasonal = TRUE)
summary(SARIMA_residuals_DR)
```

```
## Series: DR_model$residuals
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##          mean
##       -0.3294
## s.e.   0.1136
##
## sigma^2 = 49.54:  log likelihood = -12941.39
## AIC=25886.78   AICc=25886.78   BIC=25899.28
##
## Training set error measures:
##                          ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -7.780761e-15 7.037231 4.460568 94.32647 129.6561 0.7204144
##                        ACF1
## Training set -1.921898e-05
```

```
checkresiduals(SARIMA_residuals_DR, test='LB', plot=TRUE)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 245.23, df = 192, p-value = 0.005657
##
## Model df: 0.   Total lags used: 192
```

Neural Network Auto Regession

Let's try to implement a neural network AR for

```
NNAR_model=nnetar(power_train_set, xreg=temp_train_set)
print(NNAR_model)
```

```
## Series: power_train_set
## Model:  NNAR(20,1,12)[96]
## Call:   nnetar(y = power_train_set, xreg = temp_train_set)
##
## Average of 20 networks, each of which is
## a 22-12-1 network with 289 weights
## options were - linear output units
##
## sigma^2 estimated as 46.11
```

Choose th best model model:

Let's compare different RMSE of the models

```r
# Make predictions using the SLR model
SLR_predictions = forecast(SLR_model, newdata = data.frame(temp_train_set = temp_test_set))

# Make predictions using the ALR model
ALR_predictions = forecast(ALR_model, newdata = data.frame(temp_train_set = temp_test_set, season_train
ALR_residual_predictions = forecast(SARIMA_residuals_model, h = length(temp_test_set))
adjusted_ALR_predictions = ALR_predictions$mean + ALR_residual_predictions$mean

# Make predictions using the DR model
DR_predictions = forecast(DR_model,xreg = temp_test_set)

# Make predictions using the nn_AR model
NNAR_predictions = forecast(NNAR_model,xreg = temp_test_set)


# Calculate RMSE for models
RMSE_SLR = sqrt(mean((power_test_set - SLR_predictions$mean)^2))

RMSE_ALR = sqrt(mean((power_test_set - adjusted_ALR_predictions)^2))

RMSE_DR <- sqrt(mean((power_test_set - DR_predictions$mean)^2))

RMSE_NNAR <- sqrt(mean((power_test_set - NNAR_predictions$mean)^2))

# Print RMSE values for all models
cat("RMSE for DR Model:", RMSE_DR, "\n")
```

```
## RMSE for DR Model: 14.66294
```

```r
cat("RMSE for Adjusted ALR Model:", RMSE_ALR, "\n")
```

```
## RMSE for Adjusted ALR Model: 15.74652
```

```r
cat("RMSE for SLR Model:", RMSE_SLR, "\n")
```

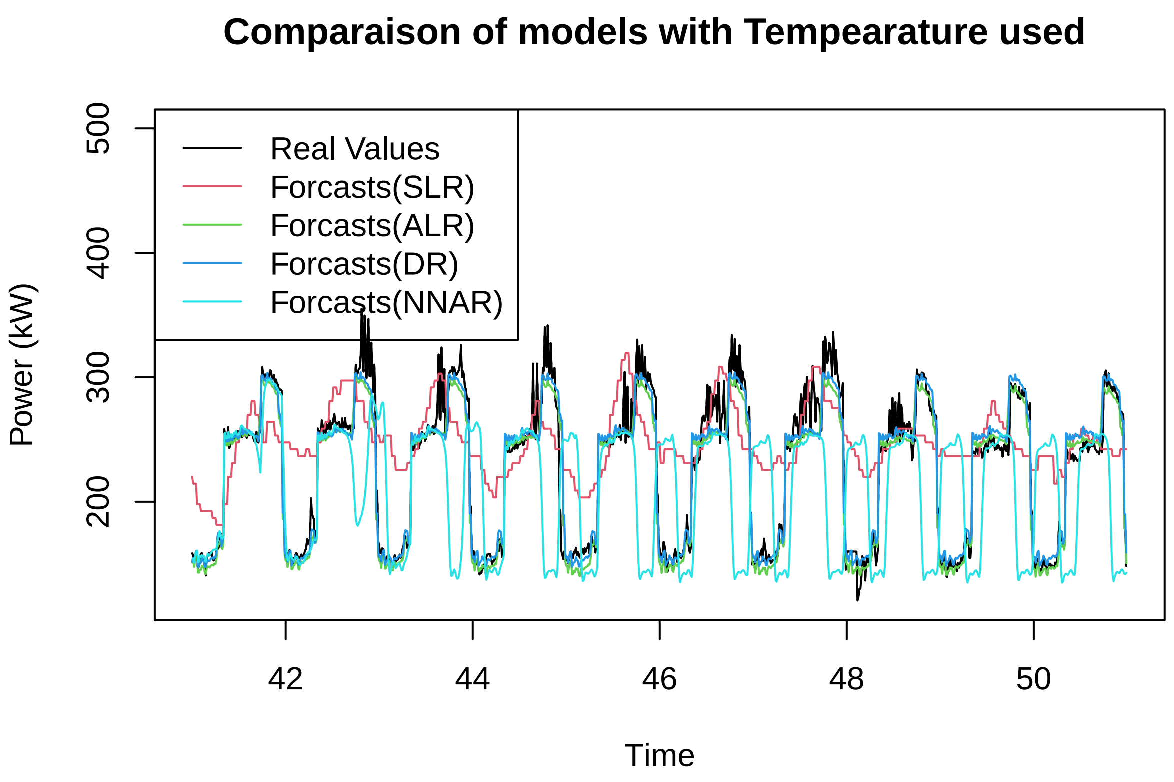
```
## RMSE for SLR Model: 50.8466
```

```r
cat("RMSE for nn_AR Model:", RMSE_NNAR, "\n")
```

```
## RMSE for nn_AR Model: 74.79329
```

```r
# Plot the data and predictions
par(mfrow=c(1,1))
plot(power_test_set, xlim=c(41,51), ylim=c(120,500),main="Comparaison of models with Tempearature used
lines(power_test_set, lty=2)
lines(SLR_predictions$mean, col=2)
lines(adjusted_ALR_predictions, col=3)
lines(DR_predictions$mean, col=4)
lines(NNAR_predictions$mean, col=5)

# Add a legend
legend('topleft',
       col=1:5,
       lty=1,
       legend=c('Real Values',
                'Forcasts(SLR)',
                'Forcasts(ALR)',
                'Forcasts(DR)',
                'Forcasts(NNAR)'))
```



## Forcasts with the best model

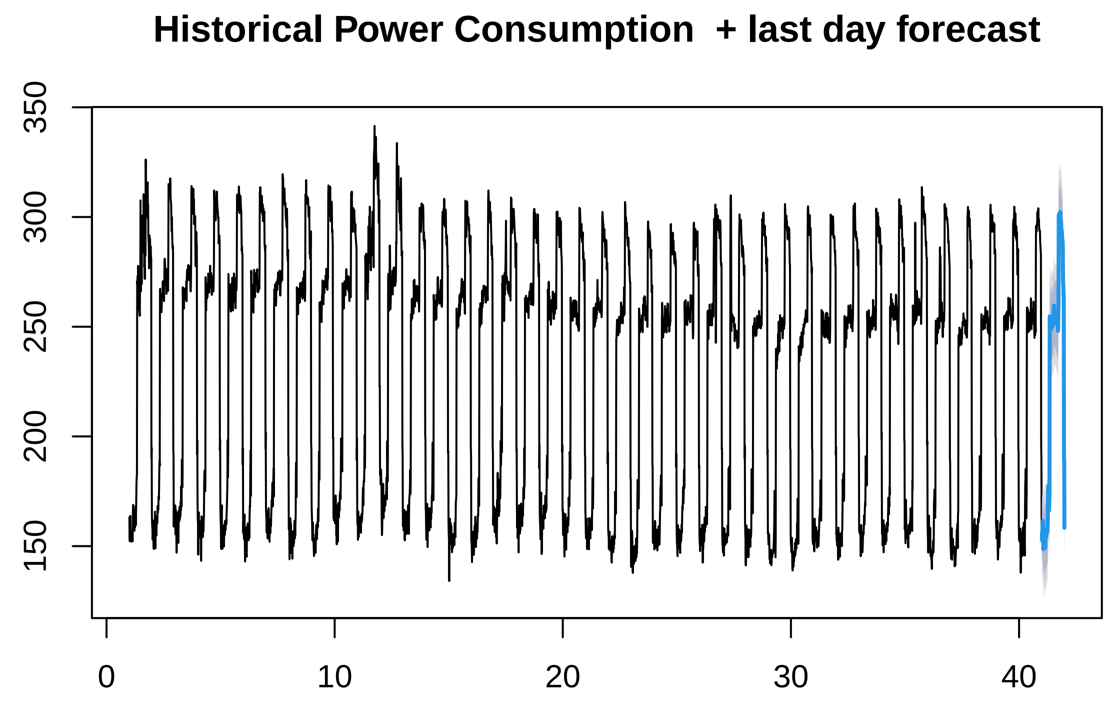
I will select the Dynamic Regression model because it provides better results. While the residuals might require some refinement, the model is still appropriate for use.

Let’s forecast for the day 21 Februrary

```r
ts_temp_21feb=window(ts_temp, start=c(51,1), end=c(51,96))

predict_power_21feb=forecast(DR_model,xreg = ts_temp_21feb)

plot(predict_power_21feb, main="Historical Power Consumption  + last day forecast")
```



```
#write_xlsx(data.frame(Power_forcast_with_Temp = round(as.numeric(predict_power_21feb$mean),1)),path =
```

Cocnlusion

In conclusion, this project successfully demonstrates the application of various forecasting models to predict electricity consumption, both with and without the influence of outdoor air temperature. By rigorously testing and tuning these models, we achieved accurate forecasts that highlight the importance of incorporating environmental factors. The comprehensive report and accompanying R code provide a clear methodology and showcase the analytical process, ensuring reproducibility and adherence to project guidelines.