# TIMES-SERIES-PROJECT

BOA THIEMELE JEAN-LUC

2024-08-12

## I-SEASONAL MODEL

### Installing libraries and packages

```
#Installing Library
#install.packages('imputeTS')
#install.packages('forecast')
#install.packages("openxlsx")

#Loading Library
library('imputeTS')
```

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

```
library('readxl')
library('forecast')
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
library('ggplot2')
library('randomForest')
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library('xgboost')
library('e1071')
library('openxlsx')
```

```
## Warning: package 'openxlsx' was built under R version 4.3.3
```

### Loading the data

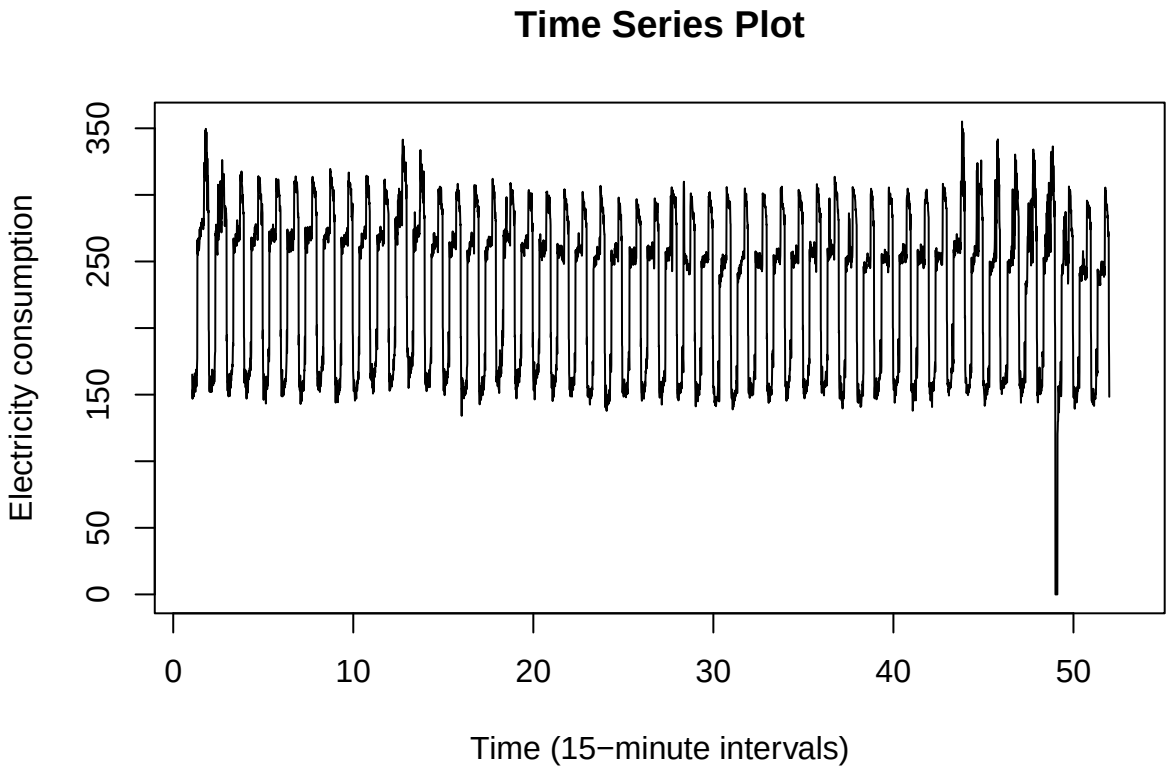
```
#Loading datasets
data <- read_excel("/Users/jlbt/Downloads/2023-11-Elec-train.xlsx")
colnames(data) <- c("times","power","temperature")
```

#Creation of times series object

```
#Transform data into a ts object
ts_data = ts(data,start=c(1,6), freq=(60*24)/15)
```

#Plotting the time series

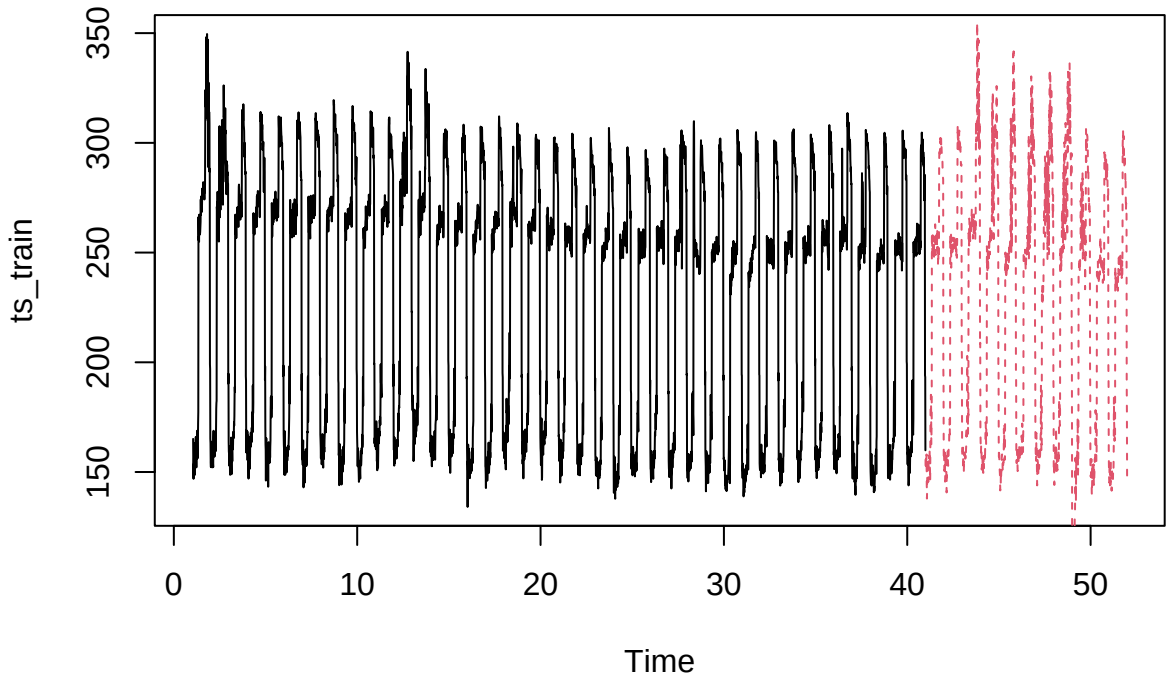
```
#Plot of the times series
plot(ts_data[,2], xlab="Time (15-minute intervals)", ylab="Electricity consumption",
     main="Time Series Plot")
```



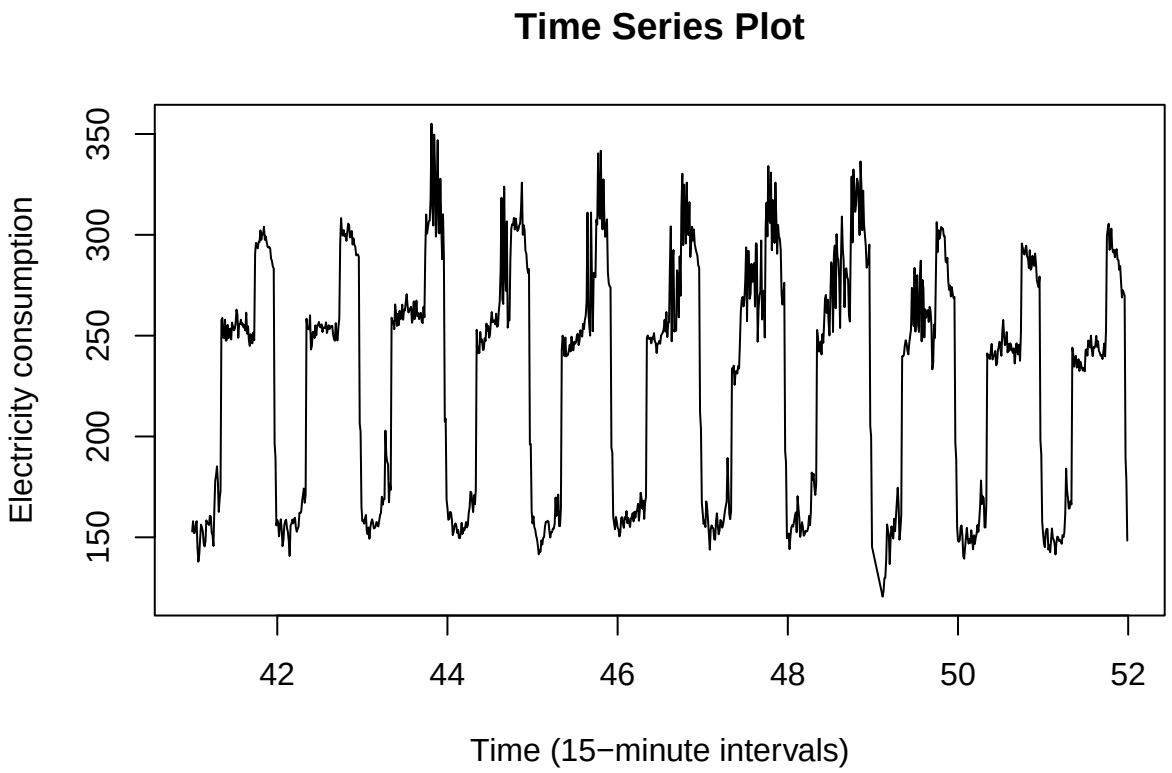

We can see that we have a span of observation that are 0, which goes from 2/18/2010 0:00 to 2/18/2010 2:30. We are going to apply an interpolation in order to have an estimated value for that span missing. But before we are going to split the data into train and validation.

#Let's create a train and test set

```
#let's create train and validation set and visualize t.hem
ts_train=window(ts_data[,2],start=c(1,6), end=c(40,96))
ts_test=window(ts_data[,2],start=c(41,1), end=c(51,96))

#Plotting of the test and train
plot(ts_train, xlim=c(1,52))
lines(ts_test,lty=2,col=2)
```



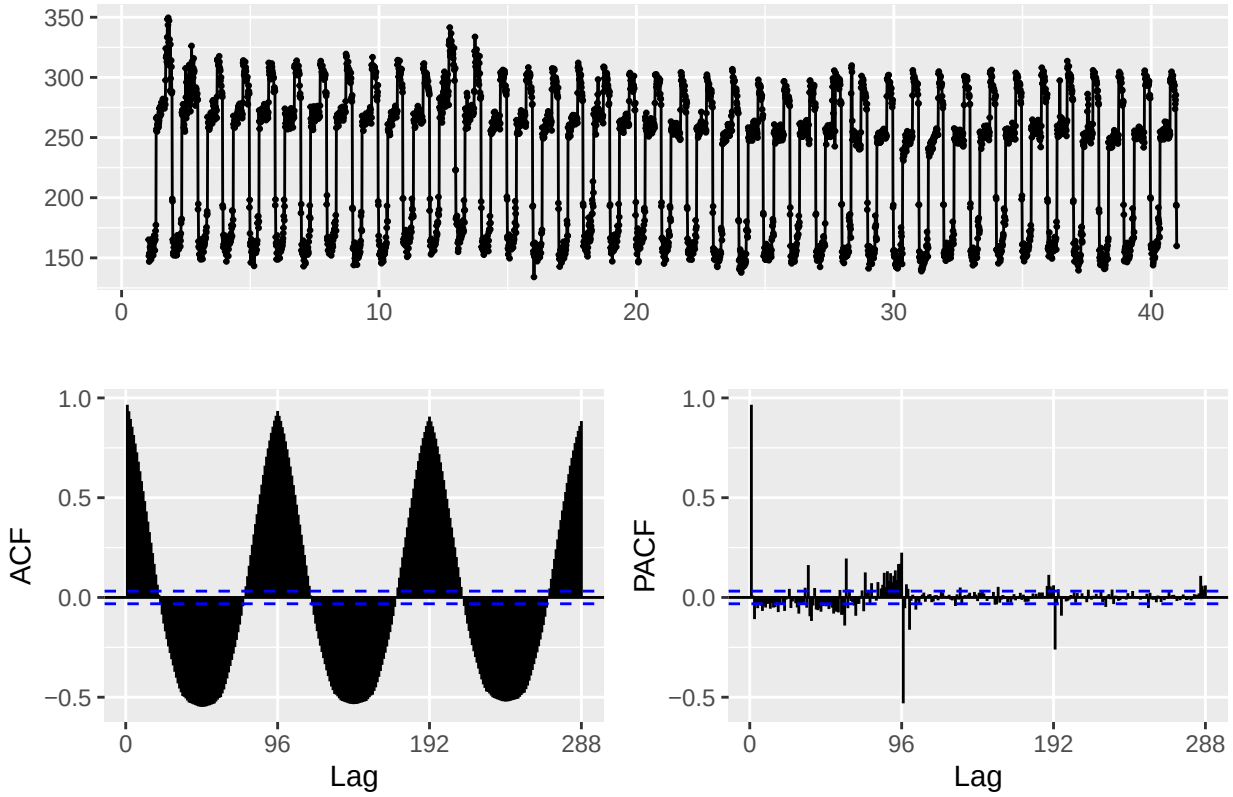

```
#Interpolation
ts_test[ts_test == 0] <- NA
ts_test=na_interpolation(ts_test)

#Plot of the result of the interpolation
plot(ts_test, xlab="Time (15-minute intervals)", ylab="Electricity consumption", main="Time Series Plot
```



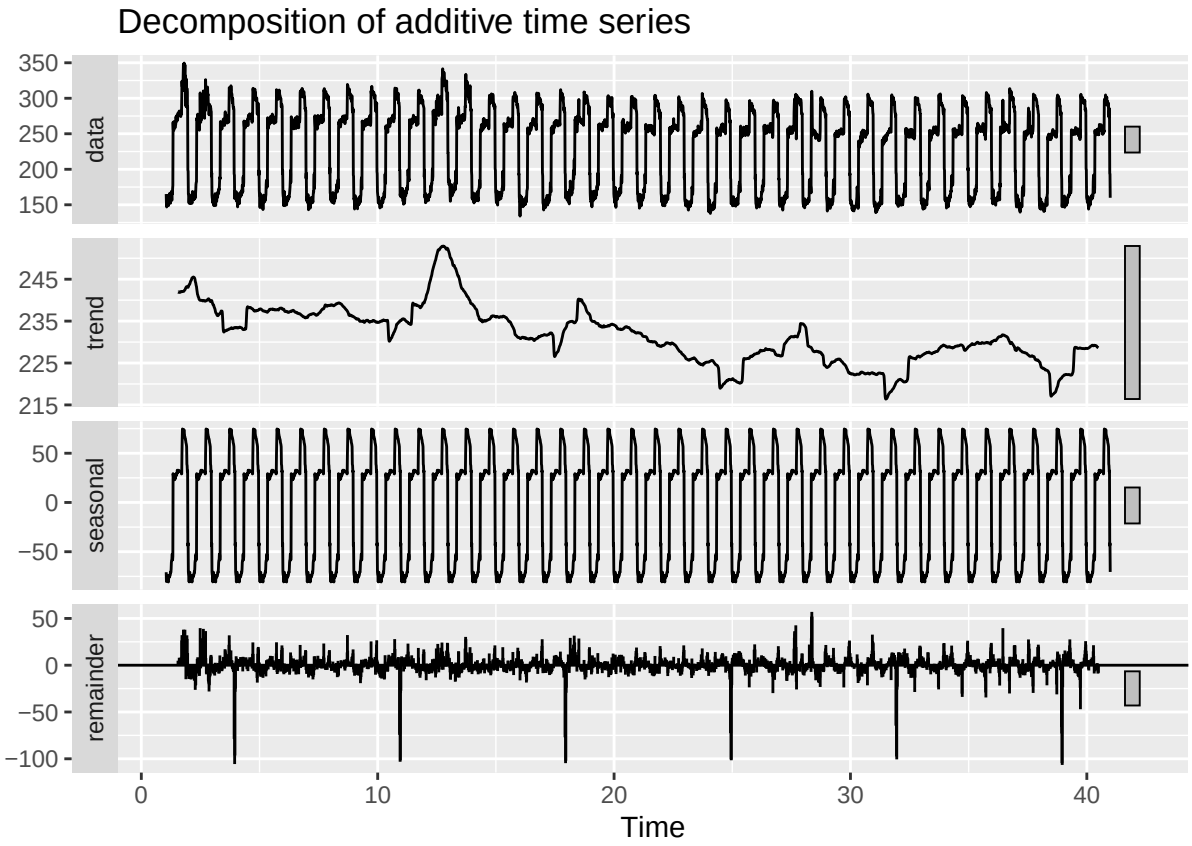

##

Let's Analyse the train data

```
ggtsdisplay(ts_train)
```




```
power = decompose(ts_train)
autoplot(power)
```

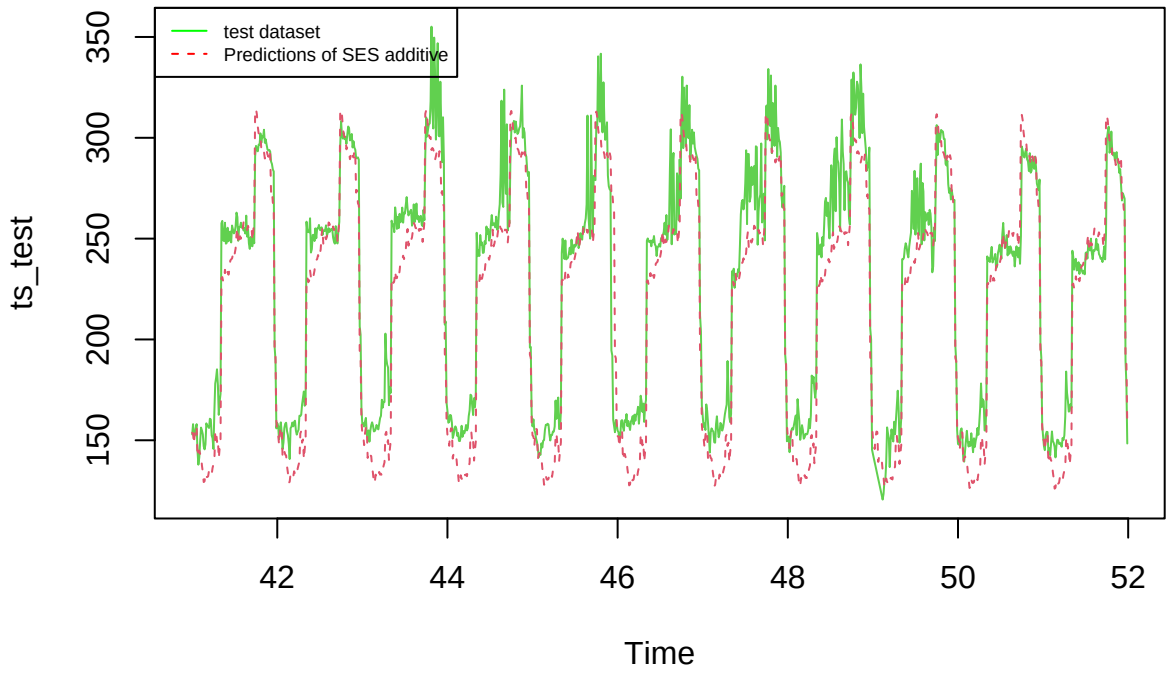



Graph 1: The ACF graph indicates a strong seasonal pattern in the data.

Graph 2: This graph shows a decreasing trend in the data. Given these observations, our initial models for the dataset will include Simple Exponential Smoothing with additive seasonality and Exponential Smoothing with multiplicative seasonality. However, due to the frequency being 96, we cannot utilize the hw function to apply the damping effect. We will proceed with the HoltWinters() functions.

## I-1 - Holt winters with additive seasonal.

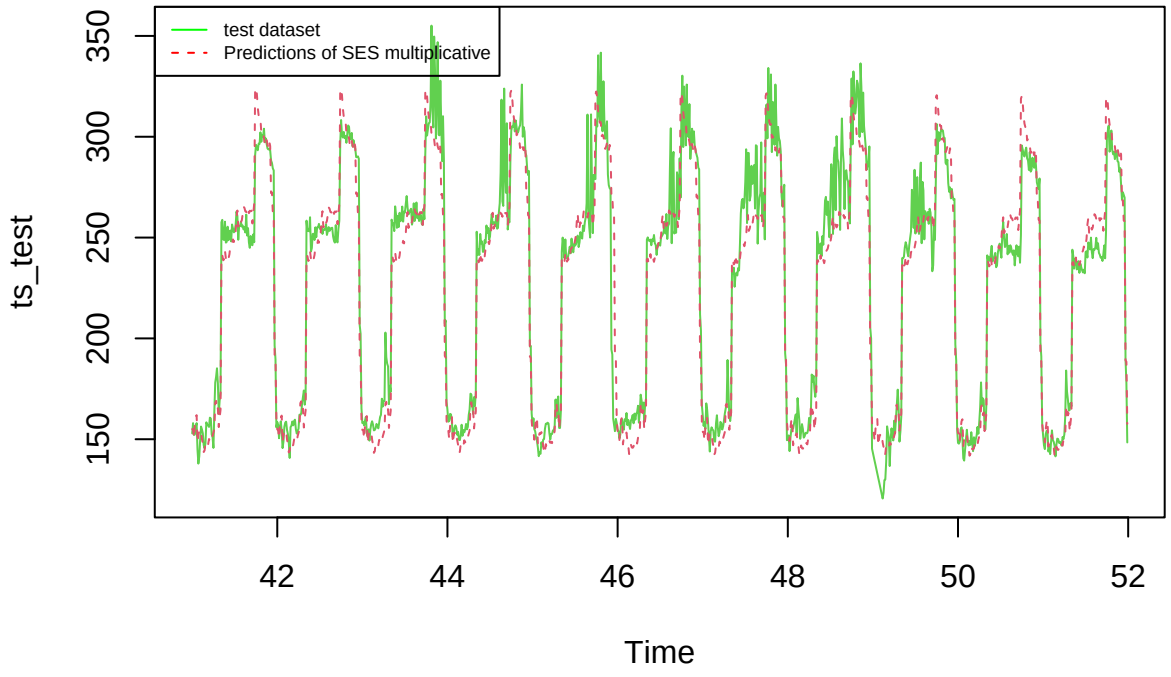
```
#Let's define the model
LES_additive = HoltWinters(ts_train,alpha=NULL,beta=NULL,gamma=NULL, seasonal = "additive")

#Prediction with the model
p_additive<-predict(LES_additive,n.ahead=96*11)

#Let's plot the test set and the prediction
plot(ts_test,col=3)
lines(p_additive,col=2,lty=2)
legend("topleft", legend = c("test dataset", "Predictions of SES additive"), col = c("Green", "red"), l
```




# I-2 -

Holt winters with multiplicative seasonal.

```
#Let's define the model
LES_multi=HoltWinters(ts_train, alpha=NULL, beta=NULL, gamma=NULL, seasonal = "multi")

#Prediction with the model
p_multi<-predict(LES_multi,n.ahead=96*11)

#Let's plot the test set and the prediction
plot(ts_test,col=3)
legend("topleft", legend = c("test dataset", "Predictions of SES multiplicative"), col = c("Green", "re
lines(p_multi,col=2,lty=2,)
```




##

Let's compute RMSE for both models

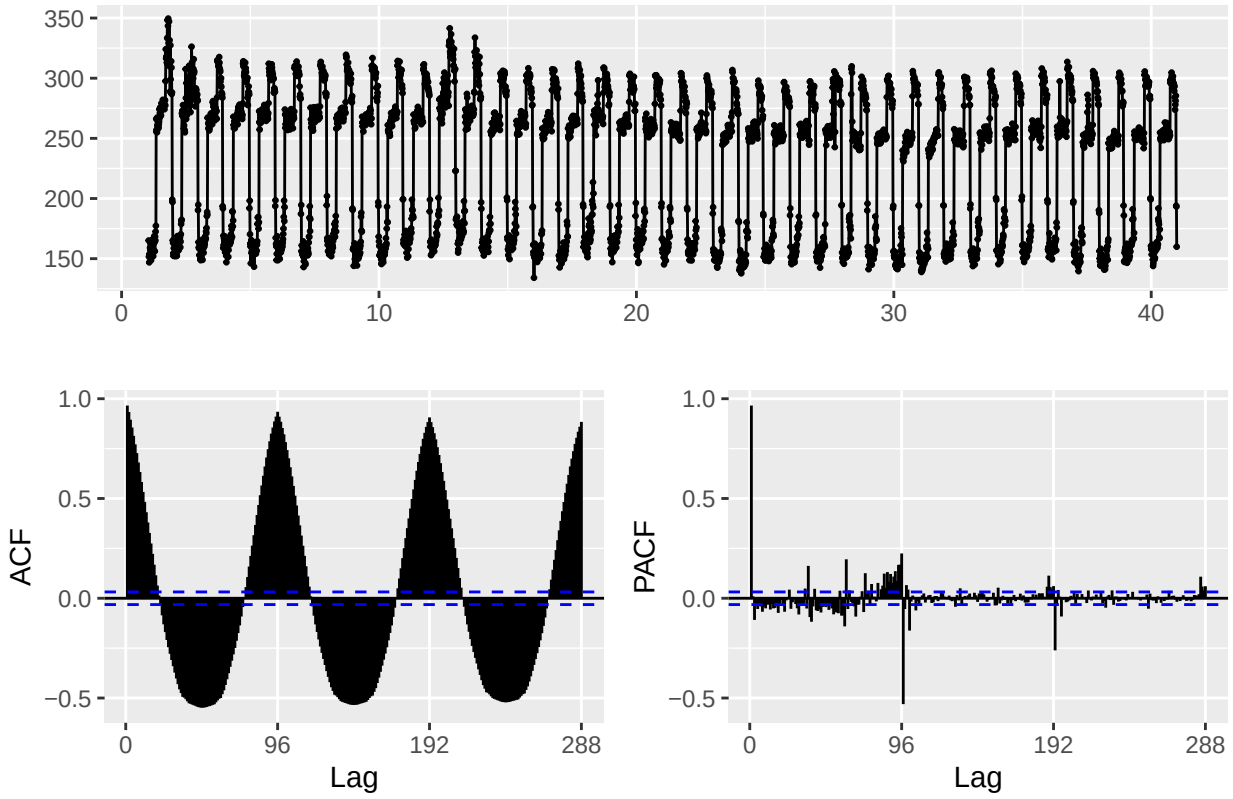
```
# Calculate RMSE for additive model
RMSE_additive <- sqrt(mean((p_additive - ts_test)^2))

# Calculate RMSE for multiplicative model
RMSE_multi <- sqrt(mean((p_multi - ts_test)^2))

# Save information into a DataFrame for analysis
results_df <- data.frame(
  Model = c(' SES Additive', 'SES Multiplicative'),
  RMSE = c(RMSE_additive, RMSE_multi)
)

print(results_df)
```

```
##                Model     RMSE
## 1       SES Additive 19.37620
## 2 SES Multiplicative 15.09799
```

The results indicate that SES with multiplicative seasonality performs better than SES with additive seasonality. Given the strong seasonality present in the model, we will proceed to build a SARIMA model.

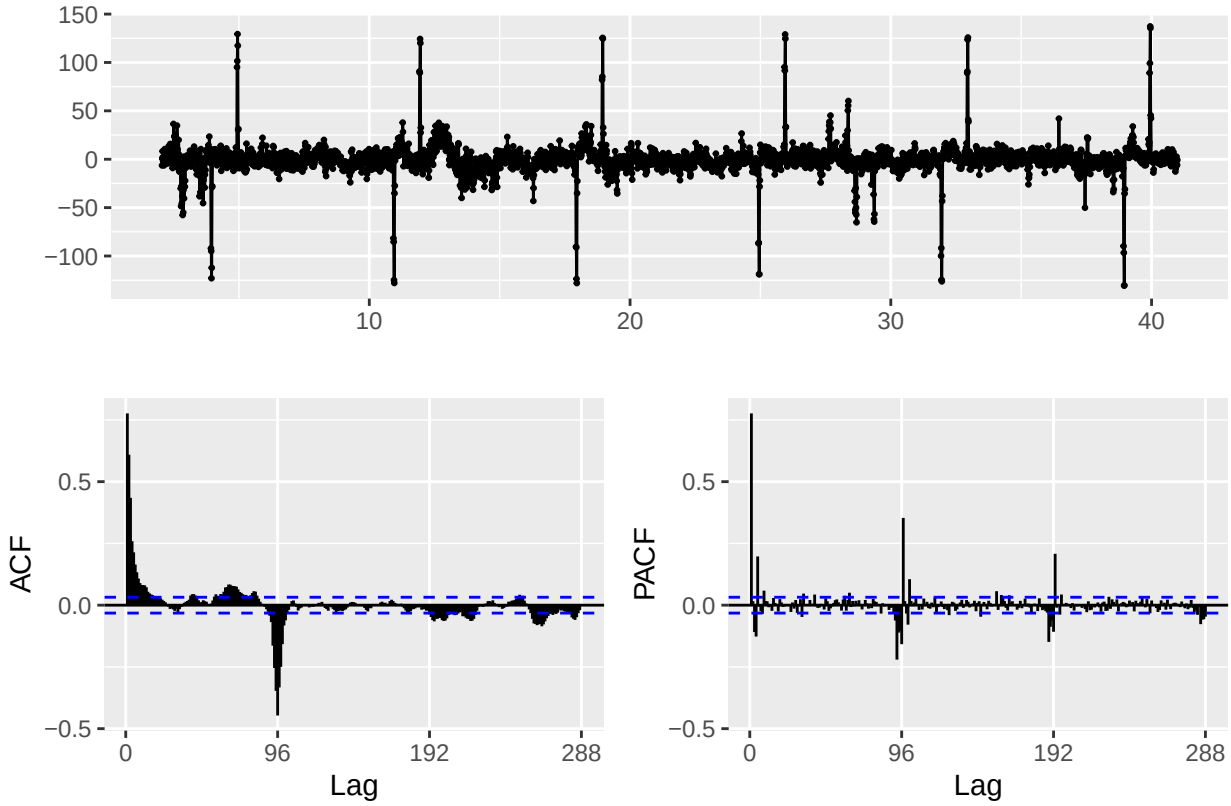
## II - STATIONNARY MODEL

```
ggtsdisplay(ts_train)
```



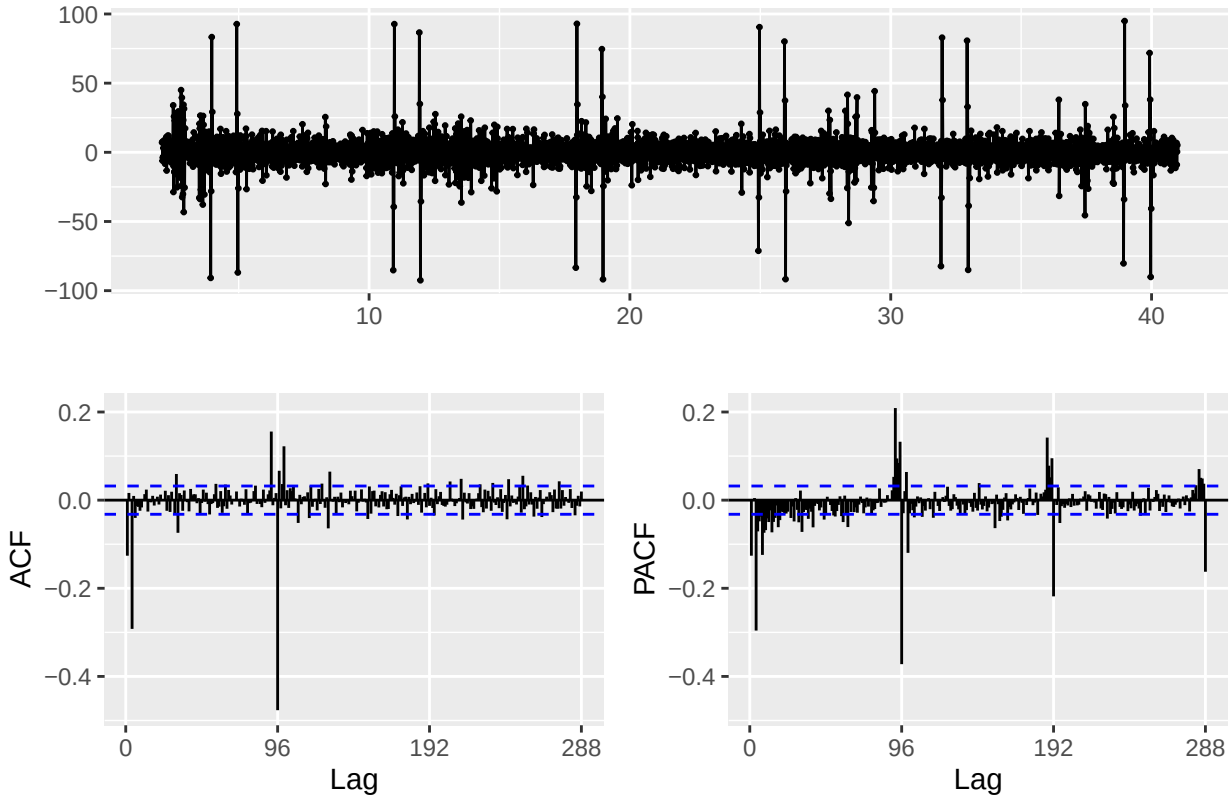

There is a strong seasonal trend in the time series. We are going to differentiate with a lag=96

```
ggtsdisplay(diff(ts_train,lag = 96,differences = 1))
```

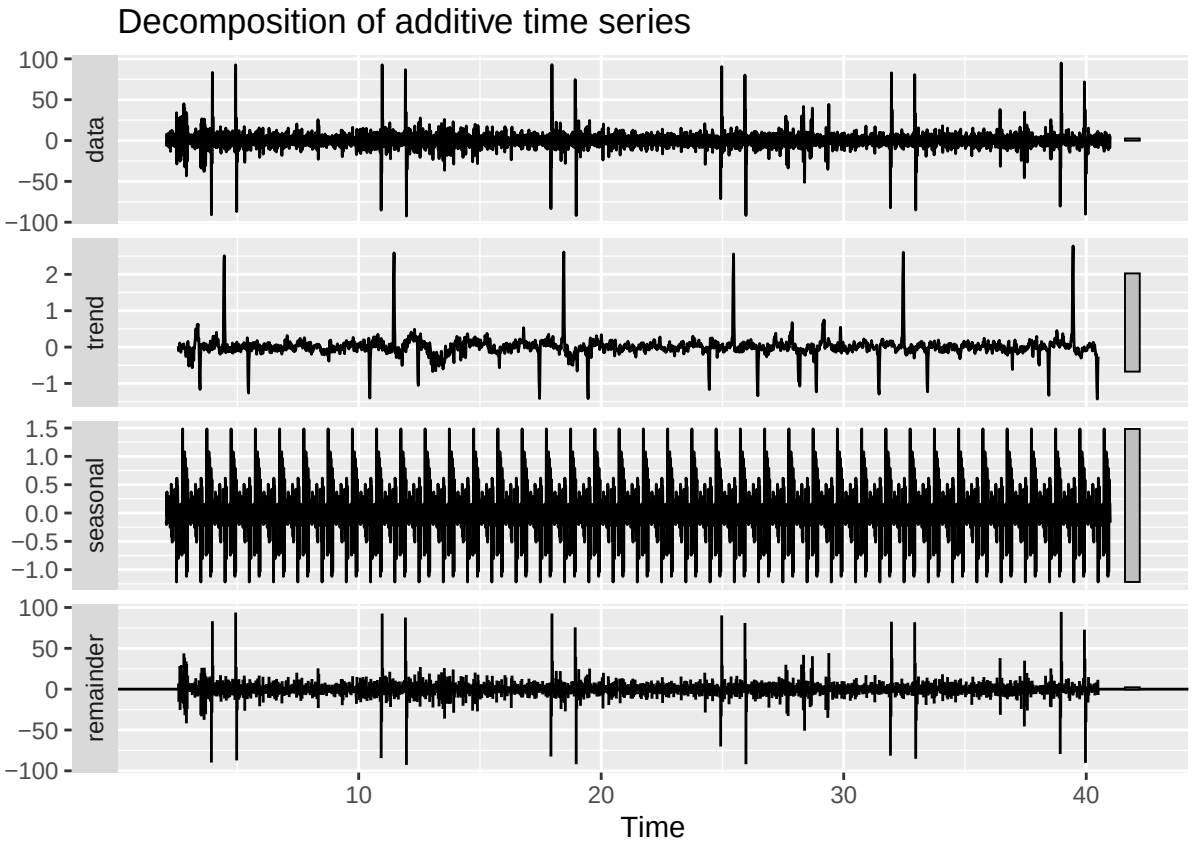



There is a slight decreasing trend at the end of the time series. To eliminate this, we will apply an additional differencing step.

```
ggtsdisplay(diff(diff(ts_train,lag = 96,differences = 1)))
```




```
power = decompose(diff(diff(ts_train,lag = 96,differences = 1)))
autoplot(power)
```




We succeded in suppressing all trend and seasonnal patterns in the times series, let's see if the autocorelation are white noise or not.

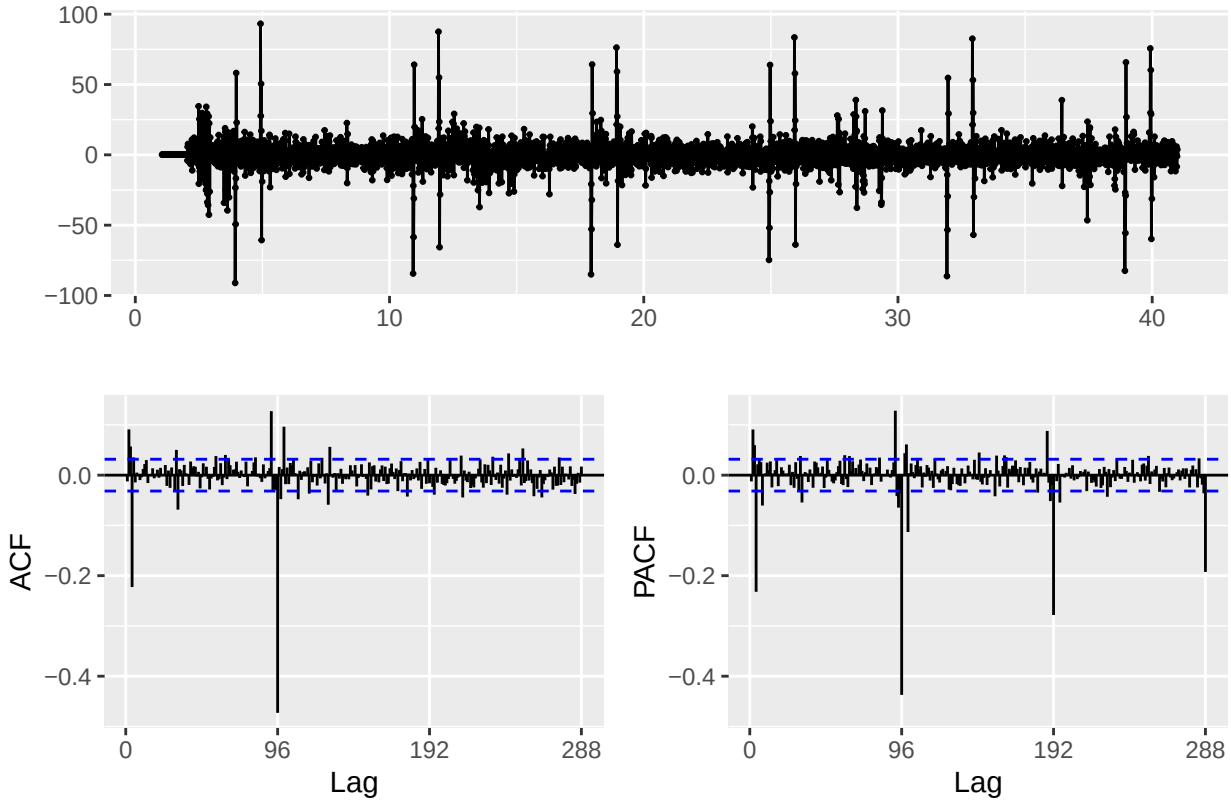
```
#let's perform Box-pierce test to see if we have white noise
Box.test(diff(diff(ts_train,lag = 96,differences = 1)),lag=10, type= 'Box-Pierce')
```

```
##
##  Box-Pierce test
##
## data:  diff(diff(ts_train, lag = 96, differences = 1))
## X-squared = 390.44, df = 10, p-value < 2.2e-16
```

We conlude that the autocorelation are significant and we are not in presence of white noise. There is some autocorrelations to modelize. Let's find a good model.

# II-1 - AUTO.ARIMA MODELING

```
# Let's apply auto arima and visualize the results
auto_arima=auto.arima(ts_train)
auto_arima %>% residuals() %>% ggtsdisplay()
```

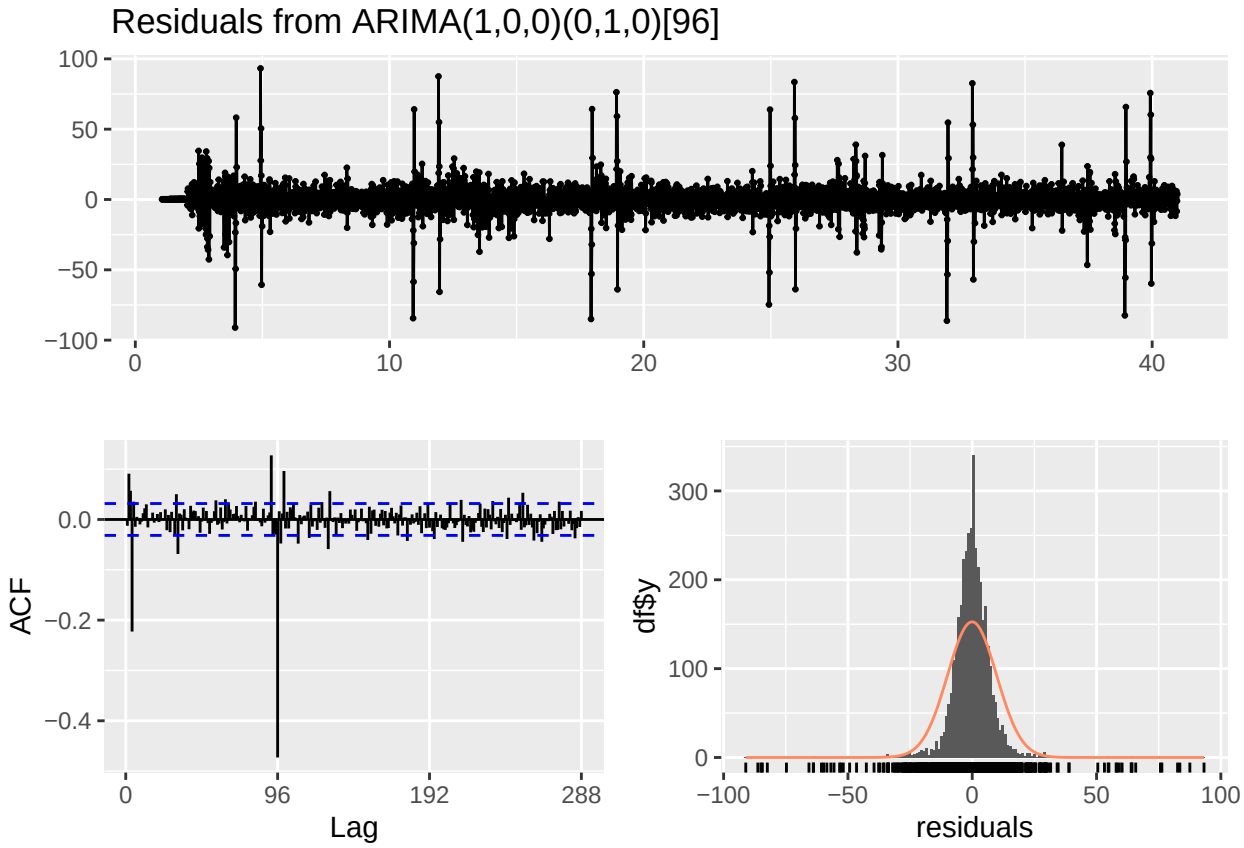



```
#Let's show the components of the model
auto_arima
```

```
## Series: ts_train
## ARIMA(1,0,0)(0,1,0)[96]
##
## Coefficients:
##          ar1
##       0.7764
## s.e.  0.0103
##
## sigma^2 = 100.1:  log likelihood = -13917.31
## AIC=27838.61   AICc=27838.62   BIC=27851.07
```

The best model selected by auto.arima is ARIMA(1,0,0)(0,1,0)[96], with an AIC of 27838.62 and a BIC of 27851.07. Furthermore, the ar1 coefficients is significant, as it's estimated values are more than twice its standard errors. Let's proceed to check the residuals.

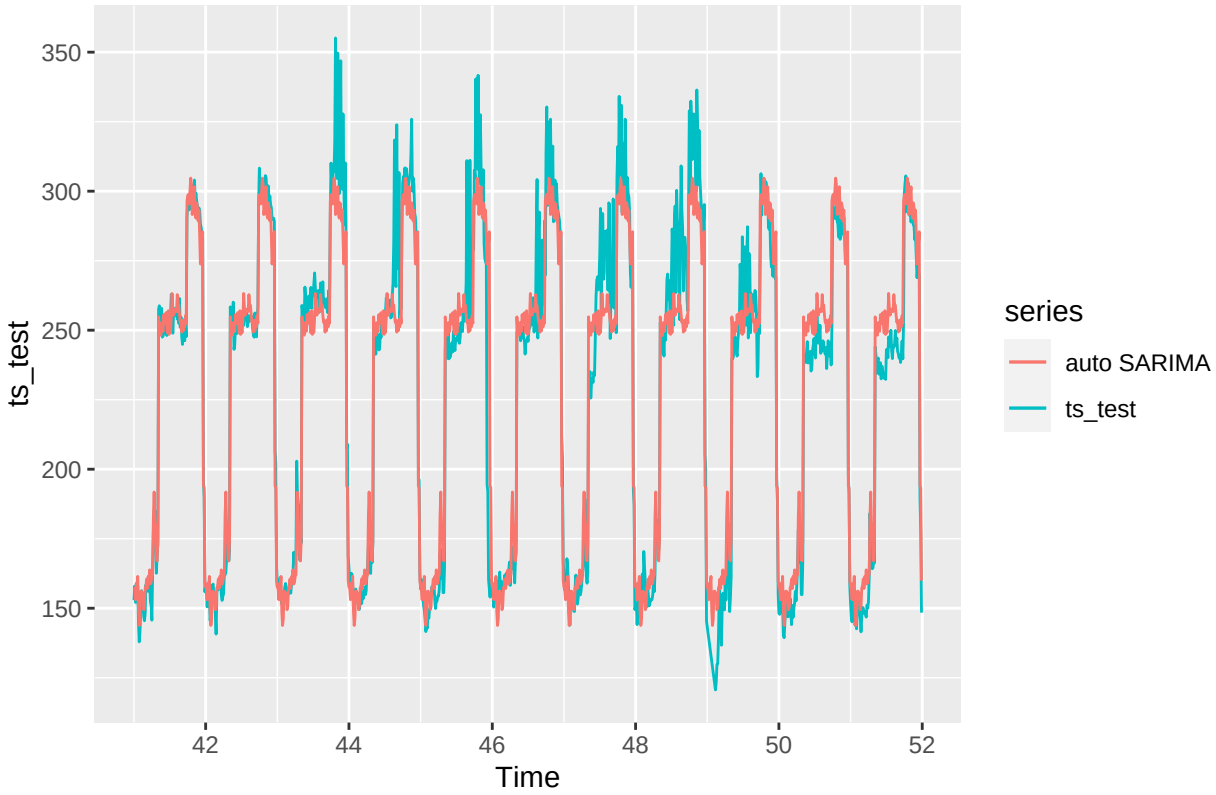
```
checkresiduals(auto_arima,lag=12)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 243.83, df = 11, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 12
```

We can see that the pvalue $< 2.2e\text{-}16$ for lag=12. So the modelization does not capture all correlation in the data. Let's see results of prediction with that model.

```
prev_auto_arima = forecast(auto_arima,h=96*11)
autoplot(ts_test, series='ts_test')+autolayer(prev_auto_arima$mean,series="auto SARIMA")
```



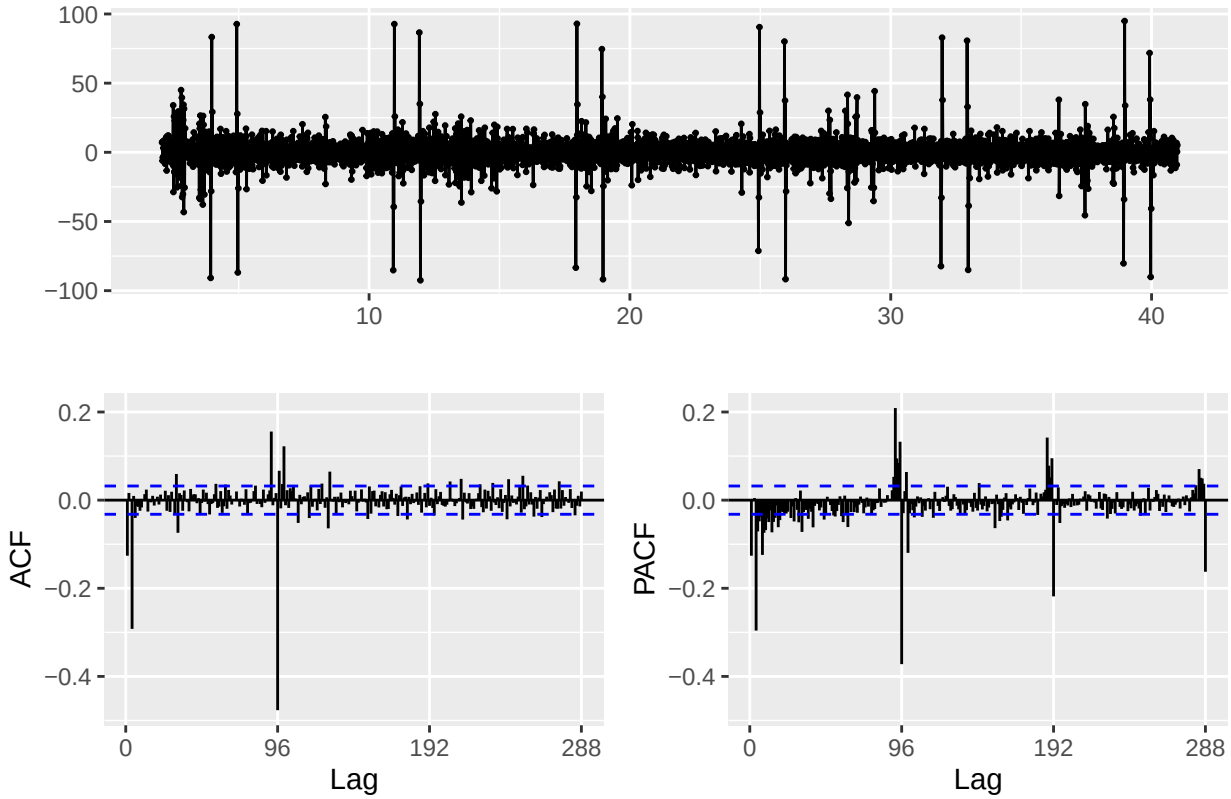

```
# Calculate RMSE for auto arima
RMSE_auto_arima = sqrt(mean((prev_auto_arima$mean - ts_test)^2))
RMSE_auto_arima
```

```
## [1] 15.03965
```

The RMSE is 15.03 and so far the auto.arima is the best model to predict power. Let's try to found a better model manually.

# II-2 - MANNUAL ARIMA MODELING

```
ggtsdisplay(diff(diff(ts_train,lag = 96,differences = 1)))
```

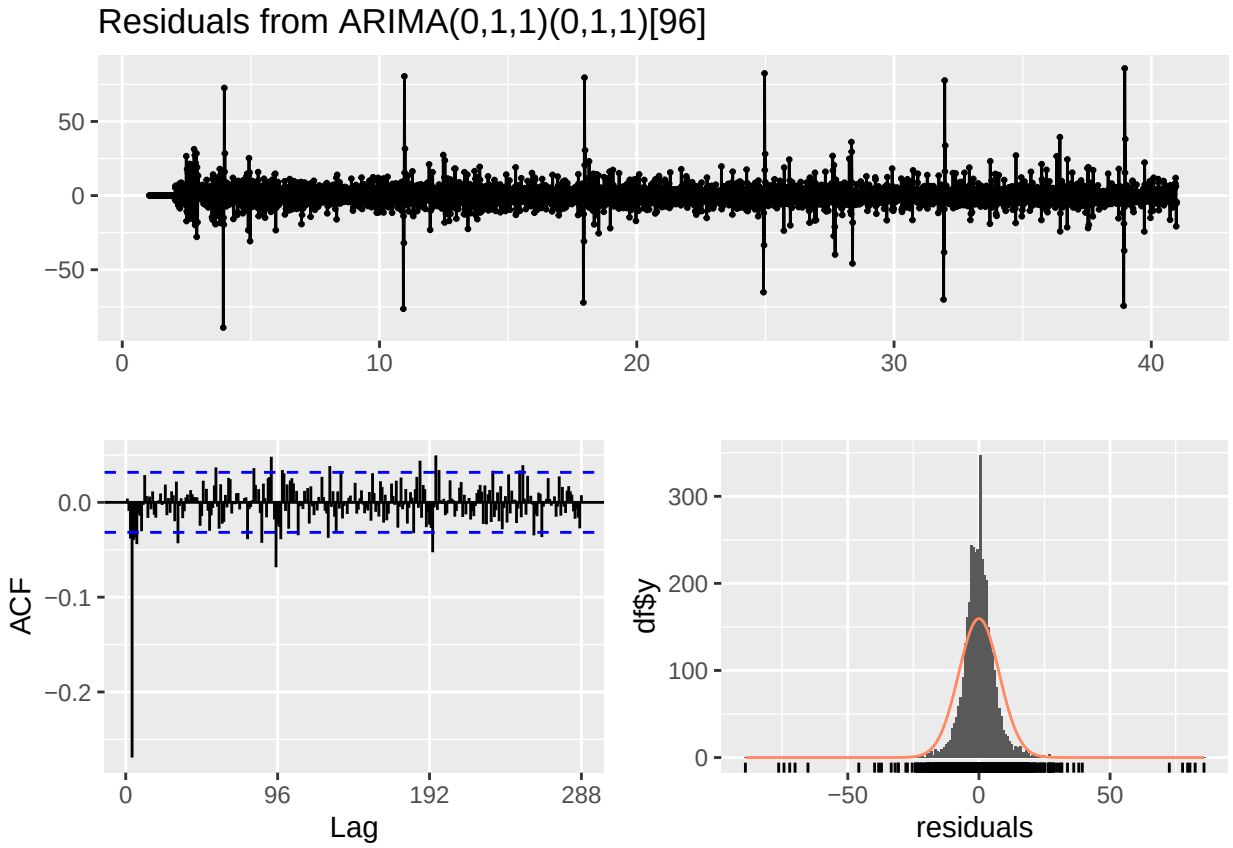



Upon examining the PACF, we observe an exponential decrease at lags 96, 192, and 288, which correspond to the different periods of the dataset. This suggests that a seasonal ARIMA model may be appropriate.

In the ACF graph, there is a significant correlation at lag 96, which aligns with the period of the time series. Based on these observations, we can propose a SARIMA model.

For the seasonal part of the SARIMA model, we suggest using (0,1,1). For the non-seasonal ARMA part, the significant correlation at lag 96 indicates the characteristics of an MA(1) process, leading us to propose (0,1,1).

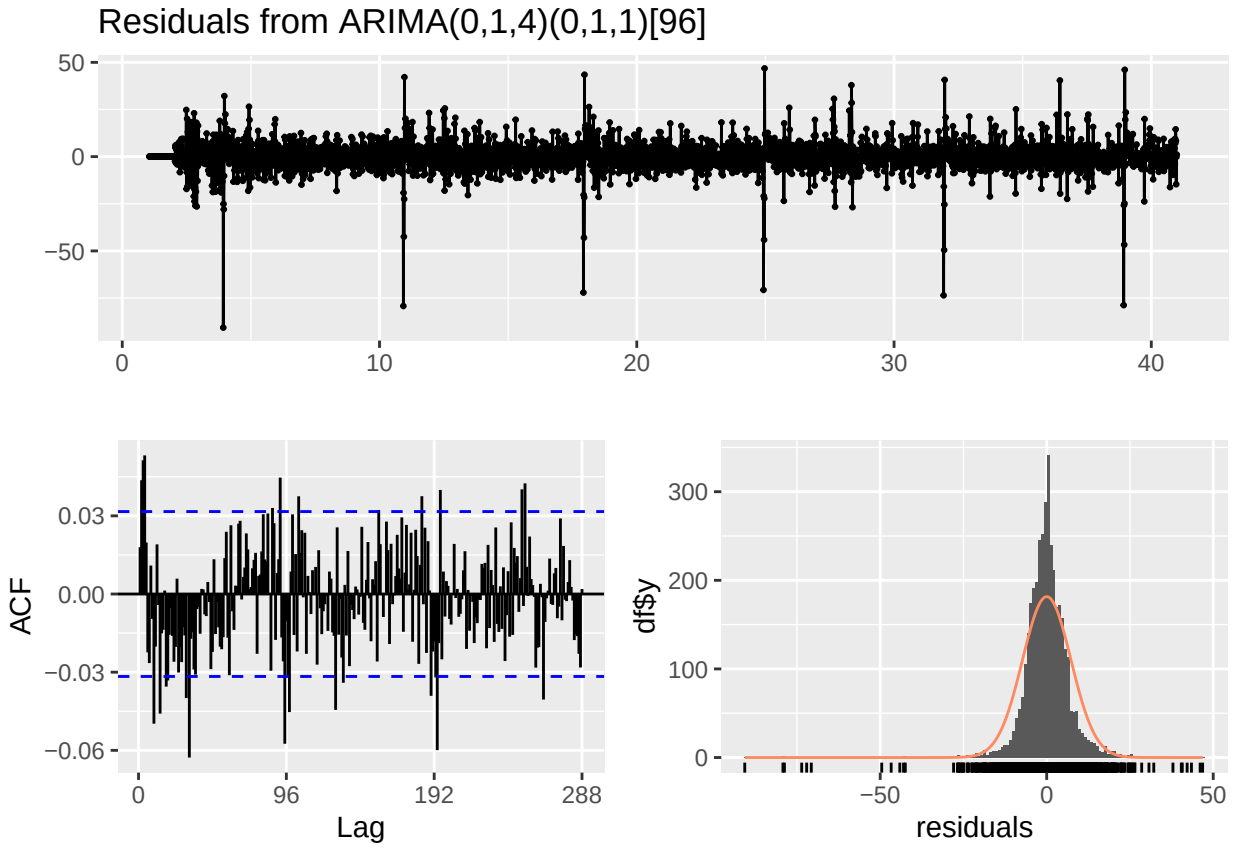
```
#Let's define the model
fit_manual_arima=Arima(ts_train, order=c(0,1,1), seasonal=c(0,1,1))

#Let's analyze residuals
checkresiduals(fit_manual_arima,lag=20)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[96]
## Q* = 314.79, df = 18, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 20
```

The Ljung-box test gives us a p-value $< 2.2e\text{-}16$. The residuals are significantly correlated, so the (0,1,1)(0,1,1) does not capture all the correlation. We can see on the ACF that there is a significant correlation at lag=4. We will try a Sarima(0,1,4)(0,1,1)

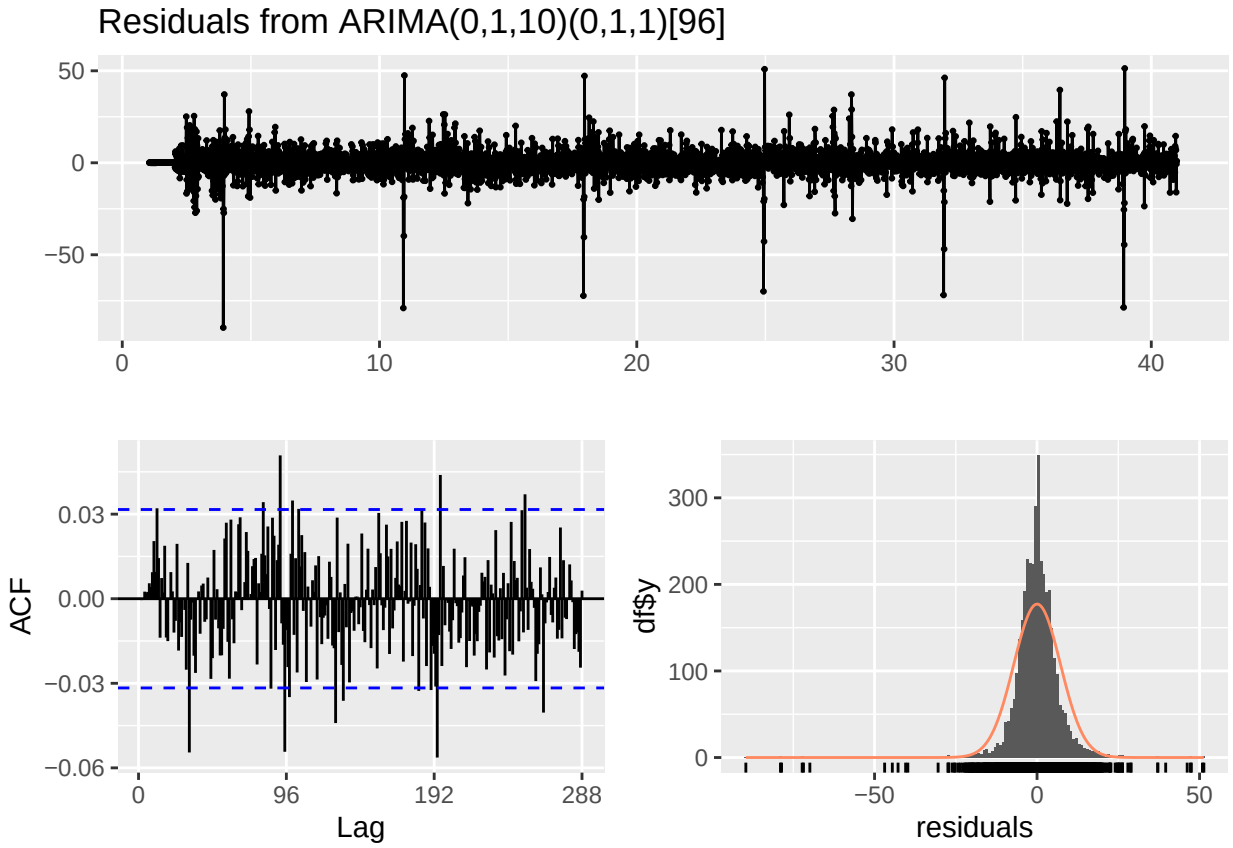
```
#Let's define the model
fit_manual_arima=Arima(ts_train, order=c(0,1,4), seasonal=c(0,1,1))

#Let's analyze residuals
checkresiduals(fit_manual_arima,lag=20)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,4)(0,1,1)[96]
## Q* = 68.819, df = 15, p-value = 7.245e-09
##
## Model df: 5.   Total lags used: 20
```

p-value < 7.117e-09. The residuals are still significantly correlated, so the (0,1,4)(0,1,1) does not capture all the correlation. we can see on the ACF that there is significant correlation at lag=10; we will try a sarima((0,1,10)(0,1,1))

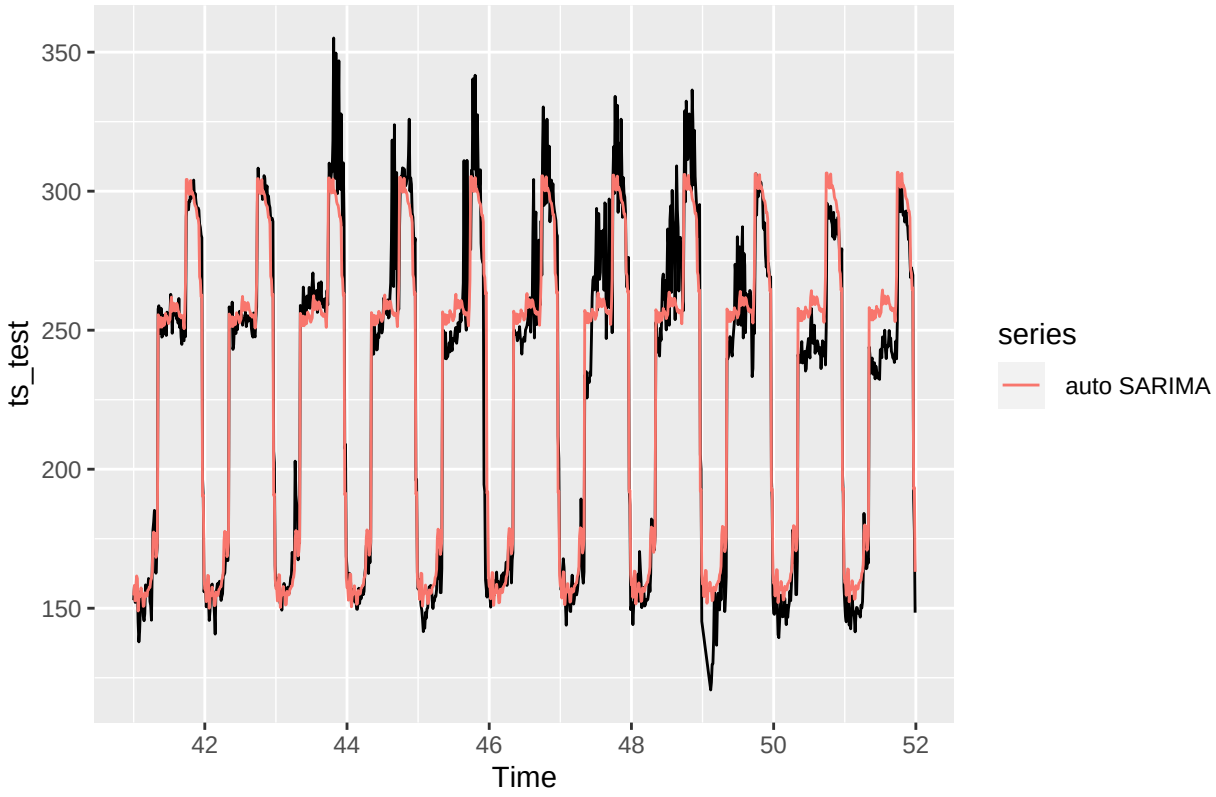
```
#Let's define the model
fit_manual_arima=Arima(ts_train, order=c(0,1,10), seasonal=c(0,1,1))

#Let's analyze residuals
checkresiduals(fit_manual_arima,lag=20)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,10)(0,1,1)[96]
## Q* = 11.234, df = 9, p-value = 0.26
##
## Model df: 11.   Total lags used: 20
```

We can see that the p-value = 0.26 , so we capture all correlation. We can use this model for forecasting.

```
#Prediction with the model
prev_manual_arima=forecast(fit_manual_arima,h=96*11)

#Ploting results
autoplot(ts_test)+autolayer(prev_manual_arima$mean,series=" auto SARIMA")
```




```
#Let's analyse de model
fit_manual_arima
```

```
## Series: ts_train
## ARIMA(0,1,10)(0,1,1)[96]
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5      ma6      ma7      ma8
##       -0.2489  -0.0643  -0.0941  -0.3254  -0.0110  -0.0564  -0.0533  -0.0064
## s.e.   0.0163   0.0169   0.0169   0.0171   0.0175   0.0174   0.0176   0.0171
##           ma9     ma10     sma1
##       -0.0258  -0.0561  -0.8504
## s.e.   0.0169   0.0160   0.0094
##
## sigma^2 = 53.32:  log likelihood = -12792.89
## AIC=25609.78   AICc=25609.87   BIC=25684.5
```

All coefficients are significant except ma5,ma8 and ma9, the BIC is of 25684.5 which is lower than the one of the auto.arima. it is the same for the AIC. Let's compute RMSE for the manual ARIMA.

```
# Evaluation of RMSE. for manual ARIMA
RMSE_manual_arima = sqrt(mean((prev_manual_arima$mean - ts_test)^2))
RMSE_manual_arima
```

```
## [1] 14.79096
```

The RMSE is lower than everything we had before, showing the good prediction given by the model. to explore further, we are going to test machine learning model to see if we can get better results.

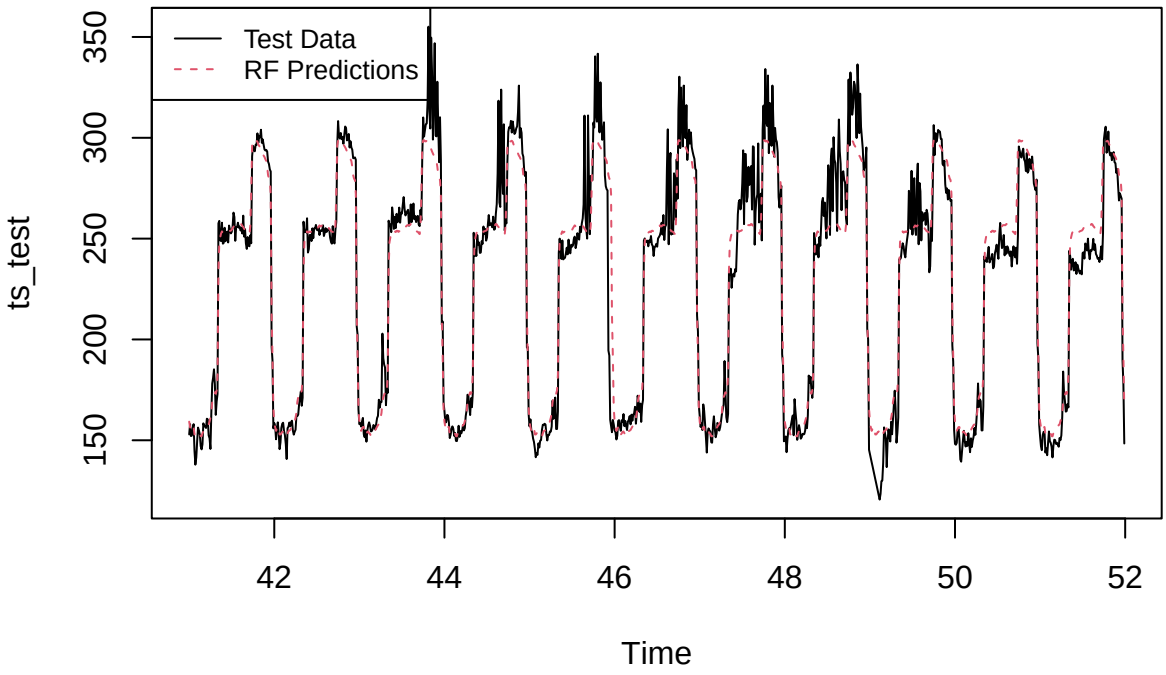
# III - MACHINE LEARNING MODEL : RF,SVM,XGBOOST

```r
# Convert the time series to a vector
ts_vector <- as.vector(ts_train)

# Create the lagged data matrix
n <- length(ts_vector)
window_size <- 97 #We are going to use the 96 last value to predict the 97 value.
data_ml <- matrix(nrow = n - window_size, ncol = window_size)

for (i in 1:(n - window_size)) {
  data_ml[i, ] <- ts_vector[i:(i + window_size - 1)]
}
```

```r
# Prepare the data for the model
x_data <- data_ml[, -window_size]
y_data <- data_ml[, window_size]
```

## III-1 Random Forest

```r
# Model RF
set.seed(123)
fitRF <- randomForest(x = x_data, y = y_data, mtry= 3, ntree=1000)

#RF
predRF <- numeric(96)
newdataRF <- tail(ts_vector, window_size-1)

for (t in 1:96) {
  predRF[t] <- predict(fitRF, newdata = matrix(newdataRF, nrow = 1, byrow = TRUE))
  newdataRF <- c(newdataRF[-1], predRF[t])
}

prevRF <- ts(predRF, start = c(41, 1), end = c(51, 96), frequency = 96)

plot(ts_test)
lines(prevRF,col=2,lty=2)
legend("topleft", legend = c("Test Data", "RF Predictions"),
       col = c(1, 2), lty = c(1, 2), cex = 0.8)
```



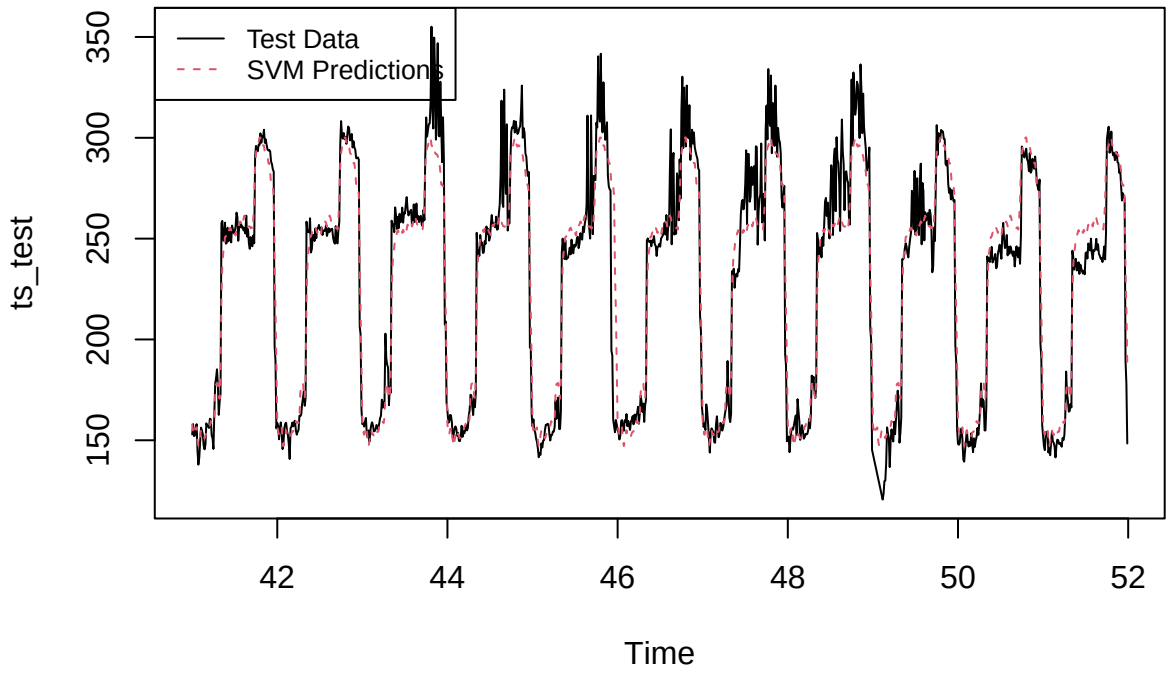

```
cat('RMSE with RF :',sqrt(mean((ts_test-prevRF)^2)),'\n')
```

```
## RMSE with RF : 14.98619
```

The Random forest gives an RMSE of 14.986 which is quite close of the one of manual arima we found in the previous section. We are going to test Support Vecteur Model (SVM) to see if it yield better result.

## III-2 SVM

```
# Model SVM
set.seed(123)
fitSVM=svm(x = x_data, y = y_data,kernel='radial',gamma=0.002)

#SVM
predSVM <- numeric(96)
newdataSVM <- tail(ts_vector, window_size-1)


for (t in 1:96) {
  predSVM[t]=predict(fitSVM,newdata=matrix(newdataSVM,1,96))
  newdataSVM=c(newdataSVM[-1],predSVM[t])
}

prevSVM <- ts(predSVM, start = c(41, 1), end = c(51, 96), frequency = 96)

plot(ts_test)
lines(prevSVM,col=2,lty=2)
legend("topleft", legend = c("Test Data", "SVM Predictions"),
       col = c(1, 2), lty = c(1, 2), cex = 0.8)
```




```
cat('RMSE with SVM :',sqrt(mean((ts_test-prevSVM)^2)),'\n')
```

```
## RMSE with SVM : 15.74482
```

The RMSE is 15.74 with a gamma = 0.002 which is the best. Random Forest perform better than SVM. let's test XGBOOST

## III-3 - XGBOOST

```
# Model XGBOOST
set.seed(123)
model<- xgboost(data= as.matrix( x_data), label = y_data,
max_depth = 3, eta = 0.5, nrounds = 200,
nthread = 2, objective = "reg:squarederror", alpha=1)
```

```
## [1]  train-rmse:119.601066
## [2]  train-rmse:60.496995
## [3]  train-rmse:31.224931
## [4]  train-rmse:17.217657
## [5]  train-rmse:10.997182
## [6]  train-rmse:8.597688
## [7]  train-rmse:7.683773
## [8]  train-rmse:7.228186
## [9]  train-rmse:7.068987
## [10] train-rmse:6.911198
## [11] train-rmse:6.838310
## [12] train-rmse:6.787270
## [13] train-rmse:6.733812
## [14] train-rmse:6.705037
## [15] train-rmse:6.556747
## [16] train-rmse:6.398795
## [17] train-rmse:6.341535
## [18] train-rmse:6.285283
```

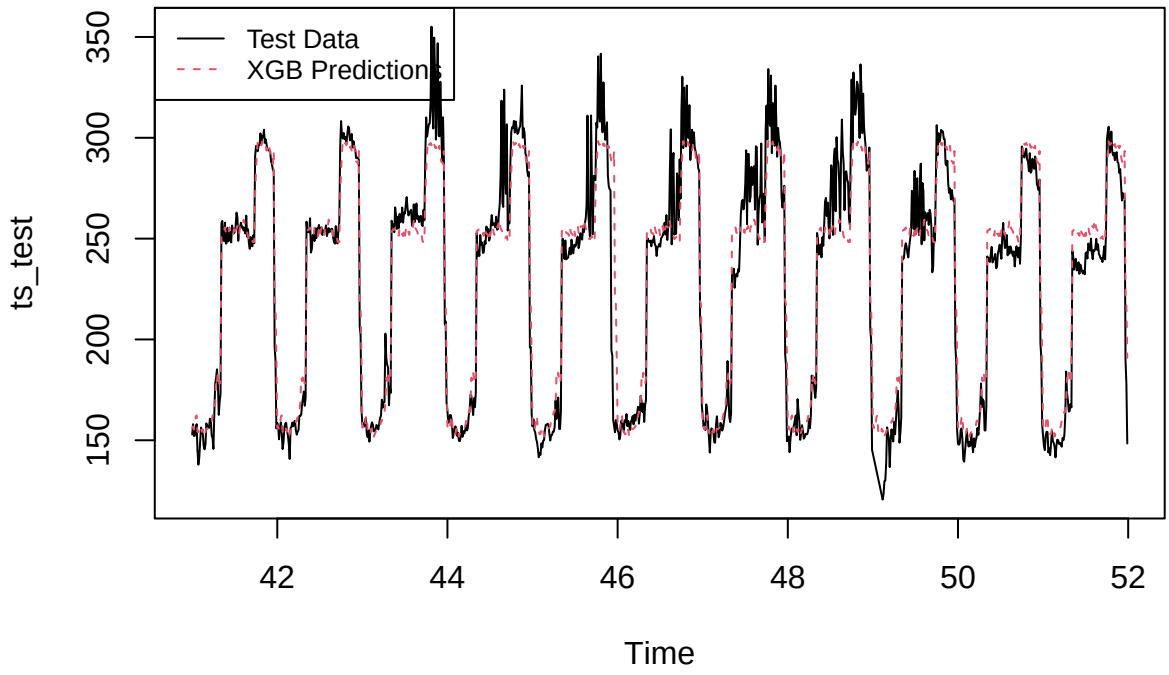
```
## [19] train-rmse:6.209737
## [20] train-rmse:6.123186
## [21] train-rmse:6.096888
## [22] train-rmse:5.976747
## [23] train-rmse:5.878138
## [24] train-rmse:5.837699
## [25] train-rmse:5.789197
## [26] train-rmse:5.749376
## [27] train-rmse:5.710950
## [28] train-rmse:5.692674
## [29] train-rmse:5.647520
## [30] train-rmse:5.599711
## [31] train-rmse:5.573887
## [32] train-rmse:5.560827
## [33] train-rmse:5.520302
## [34] train-rmse:5.480944
## [35] train-rmse:5.450795
## [36] train-rmse:5.393392
## [37] train-rmse:5.357999
## [38] train-rmse:5.308569
## [39] train-rmse:5.276638
## [40] train-rmse:5.220854
## [41] train-rmse:5.151552
## [42] train-rmse:5.076409
## [43] train-rmse:5.016870
## [44] train-rmse:4.952345
## [45] train-rmse:4.903639
## [46] train-rmse:4.865904
## [47] train-rmse:4.832160
## [48] train-rmse:4.814326
## [49] train-rmse:4.763306
## [50] train-rmse:4.721376
## [51] train-rmse:4.678797
## [52] train-rmse:4.665961
## [53] train-rmse:4.640433
## [54] train-rmse:4.612867
## [55] train-rmse:4.574579
## [56] train-rmse:4.545681
## [57] train-rmse:4.508416
## [58] train-rmse:4.491707
## [59] train-rmse:4.469428
## [60] train-rmse:4.430643
## [61] train-rmse:4.402639
## [62] train-rmse:4.381817
## [63] train-rmse:4.365930
## [64] train-rmse:4.354230
## [65] train-rmse:4.346614
## [66] train-rmse:4.328248
## [67] train-rmse:4.307448
## [68] train-rmse:4.296818
## [69] train-rmse:4.271889
## [70] train-rmse:4.253312
## [71] train-rmse:4.215454
## [72] train-rmse:4.192478
```

```
## [73] train-rmse:4.166516
## [74] train-rmse:4.139715
## [75] train-rmse:4.112373
## [76] train-rmse:4.086567
## [77] train-rmse:4.068103
## [78] train-rmse:4.038372
## [79] train-rmse:4.021618
## [80] train-rmse:3.999528
## [81] train-rmse:3.975247
## [82] train-rmse:3.961685
## [83] train-rmse:3.938488
## [84] train-rmse:3.928628
## [85] train-rmse:3.915540
## [86] train-rmse:3.893665
## [87] train-rmse:3.872754
## [88] train-rmse:3.855142
## [89] train-rmse:3.839883
## [90] train-rmse:3.819084
## [91] train-rmse:3.795544
## [92] train-rmse:3.783885
## [93] train-rmse:3.764703
## [94] train-rmse:3.743699
## [95] train-rmse:3.733085
## [96] train-rmse:3.716224
## [97] train-rmse:3.706104
## [98] train-rmse:3.686605
## [99] train-rmse:3.667401
## [100]    train-rmse:3.646311
## [101]    train-rmse:3.633500
## [102]    train-rmse:3.591432
## [103]    train-rmse:3.576294
## [104]    train-rmse:3.559196
## [105]    train-rmse:3.546310
## [106]    train-rmse:3.537580
## [107]    train-rmse:3.522669
## [108]    train-rmse:3.502287
## [109]    train-rmse:3.473864
## [110]    train-rmse:3.447143
## [111]    train-rmse:3.432408
## [112]    train-rmse:3.426781
## [113]    train-rmse:3.402200
## [114]    train-rmse:3.383583
## [115]    train-rmse:3.360363
## [116]    train-rmse:3.342790
## [117]    train-rmse:3.317182
## [118]    train-rmse:3.304909
## [119]    train-rmse:3.291001
## [120]    train-rmse:3.281965
## [121]    train-rmse:3.270237
## [122]    train-rmse:3.263338
## [123]    train-rmse:3.242999
## [124]    train-rmse:3.224384
## [125]    train-rmse:3.213980
## [126]    train-rmse:3.202759
```

```
## [127]    train-rmse:3.185297
## [128]    train-rmse:3.181507
## [129]    train-rmse:3.160417
## [130]    train-rmse:3.147209
## [131]    train-rmse:3.138293
## [132]    train-rmse:3.122848
## [133]    train-rmse:3.107750
## [134]    train-rmse:3.093900
## [135]    train-rmse:3.079256
## [136]    train-rmse:3.072098
## [137]    train-rmse:3.058541
## [138]    train-rmse:3.044948
## [139]    train-rmse:3.027788
## [140]    train-rmse:3.019343
## [141]    train-rmse:3.013705
## [142]    train-rmse:2.994158
## [143]    train-rmse:2.967202
## [144]    train-rmse:2.951966
## [145]    train-rmse:2.932977
## [146]    train-rmse:2.916088
## [147]    train-rmse:2.905700
## [148]    train-rmse:2.889693
## [149]    train-rmse:2.875117
## [150]    train-rmse:2.855959
## [151]    train-rmse:2.841489
## [152]    train-rmse:2.826608
## [153]    train-rmse:2.815519
## [154]    train-rmse:2.806693
## [155]    train-rmse:2.789007
## [156]    train-rmse:2.779877
## [157]    train-rmse:2.763729
## [158]    train-rmse:2.743892
## [159]    train-rmse:2.731034
## [160]    train-rmse:2.710704
## [161]    train-rmse:2.697523
## [162]    train-rmse:2.687694
## [163]    train-rmse:2.672135
## [164]    train-rmse:2.667627
## [165]    train-rmse:2.646410
## [166]    train-rmse:2.635682
## [167]    train-rmse:2.622781
## [168]    train-rmse:2.609753
## [169]    train-rmse:2.592460
## [170]    train-rmse:2.586617
## [171]    train-rmse:2.578797
## [172]    train-rmse:2.562636
## [173]    train-rmse:2.549559
## [174]    train-rmse:2.534173
## [175]    train-rmse:2.521504
## [176]    train-rmse:2.501898
## [177]    train-rmse:2.492669
## [178]    train-rmse:2.481291
## [179]    train-rmse:2.471628
## [180]    train-rmse:2.469288
```

```
## [181]    train-rmse:2.462687
## [182]    train-rmse:2.455947
## [183]    train-rmse:2.445480
## [184]    train-rmse:2.434610
## [185]    train-rmse:2.426259
## [186]    train-rmse:2.411754
## [187]    train-rmse:2.402666
## [188]    train-rmse:2.393357
## [189]    train-rmse:2.382587
## [190]    train-rmse:2.372935
## [191]    train-rmse:2.361694
## [192]    train-rmse:2.347368
## [193]    train-rmse:2.334255
## [194]    train-rmse:2.321640
## [195]    train-rmse:2.310782
## [196]    train-rmse:2.302175
## [197]    train-rmse:2.292239
## [198]    train-rmse:2.280597
## [199]    train-rmse:2.269561
## [200]    train-rmse:2.264500
```

```r
#XGB
predXGB <- numeric(96)
newdataXGB <- as.matrix(tail(ts_vector, window_size-1))


for (t in 1:96) {
  predXGB[t] <- predict(model, newdata =  xgb.DMatrix(matrix(newdataXGB, nrow = 1, byrow = TRUE)))
  newdataXGB <- c(newdataXGB[-1], predXGB[t])
}

prevXGB <- ts(predXGB, start = c(41, 1), end = c(51, 96), frequency = 96)

plot(ts_test)
lines(prevXGB,col=2,lty=2)
legend("topleft", legend = c("Test Data", "XGB Predictions"), col = c(1, 2), lty = c(1, 2), cex = 0.8)
```

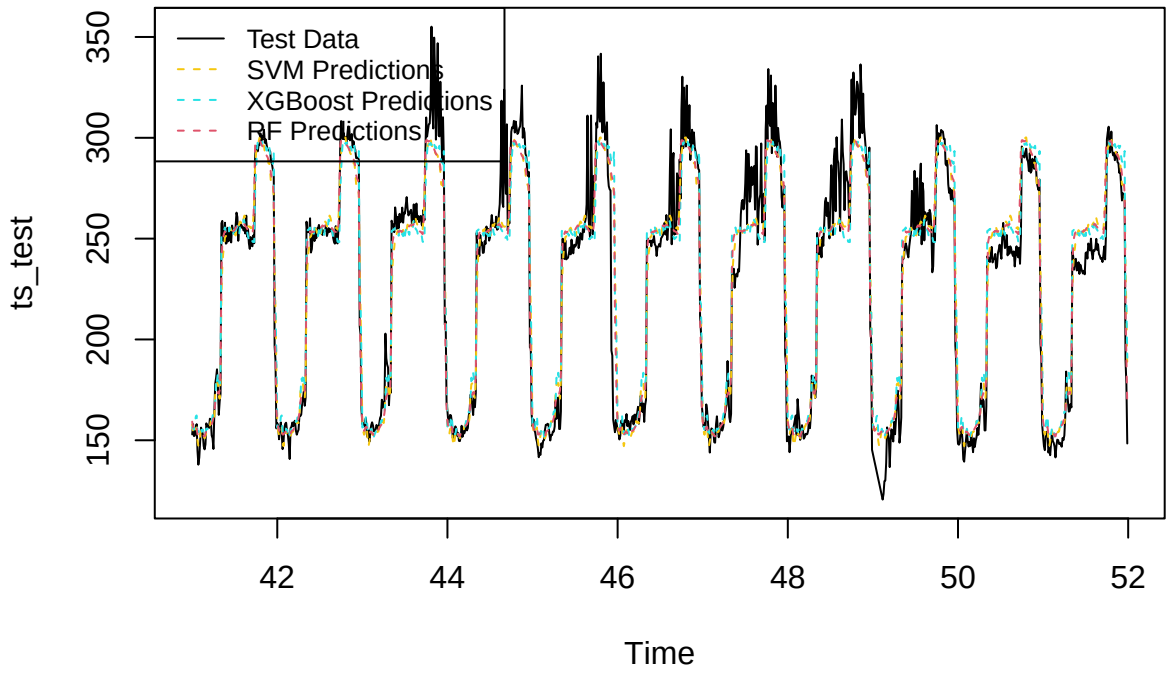



```
cat('RMSE with XGB :',sqrt(mean((ts_test-prevXGB)^2)),'\n')
```

```
## RMSE with XGB : 16.42254
```

The XGBOOST gives the worst result when comparing RMSE to the one of SVM and RF, we got a RMSE of 16.42. None of the model are better than the manual arima that we found which has an RMSE of 14.79. we will then perform our final prediction with manual auto arima. . Let's plot the result of the machine learning part

```
plot(ts_test)
lines(prevSVM,col=7, lty=2)
lines(prevXGB,col=5,lty=2)
lines(prevRF,col=2,lty=2)
legend("topleft", legend = c("Test Data", "SVM Predictions", "XGBoost Predictions", "RF Predictions"),
       col = c(1, 7, 5, 2), lty = c(1, 2, 2, 2), cex = 0.8)
```



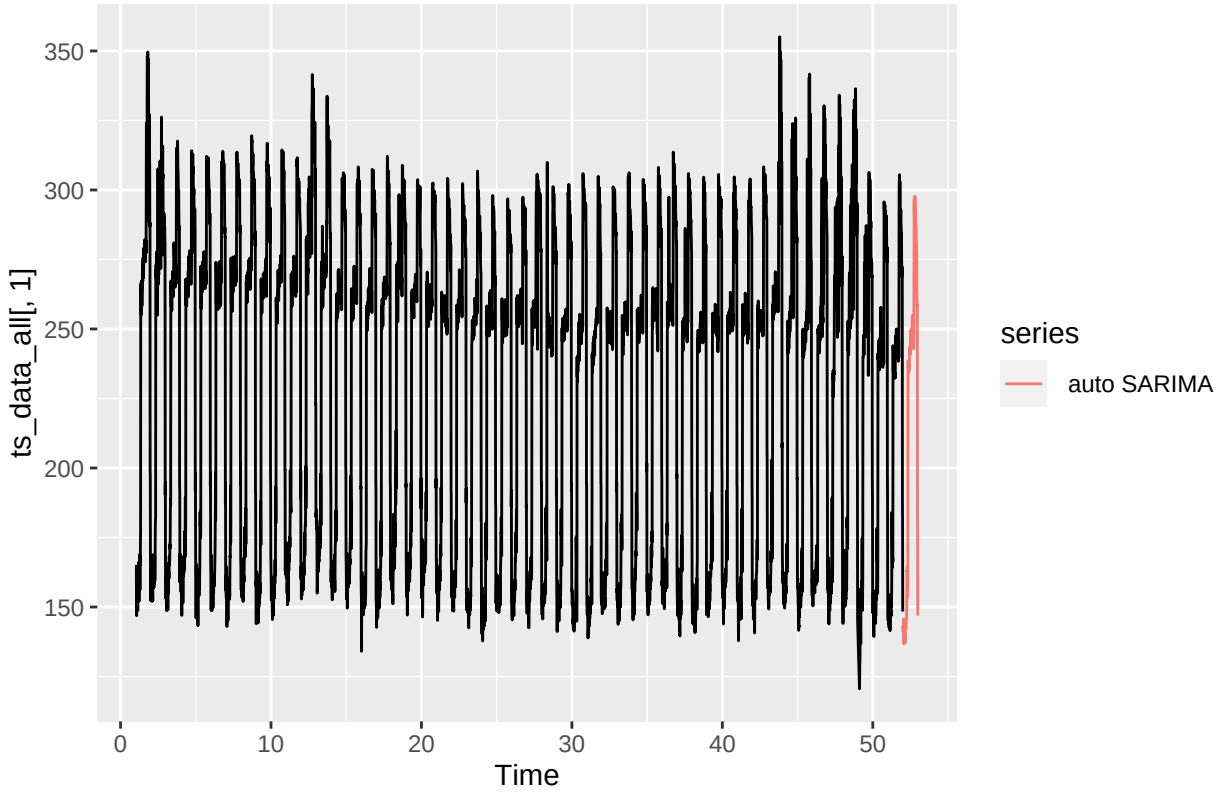

```
RMSE_RF = sqrt(mean((ts_test-prevRF)^2))
RMSE_SVM = sqrt(mean((ts_test-prevSVM)^2))
RMSE_XGB = sqrt(mean((ts_test-prevXGB)^2))

# Save information into a DataFrame for analysis
results_df <- data.frame(
  Model = c(' SES Additive', 'SES Multiplicative','auto.arima model','manual arima model','RF','SVM','X
  RMSE = c(RMSE_additive, RMSE_multi,RMSE_auto_arima, RMSE_manual_arima,RMSE_RF,RMSE_SVM,RMSE_XGB)
)
results_df <- results_df[order(results_df$RMSE),]

# Afficher le data.frame rangé

print(results_df)
```

```
##                Model     RMSE
## 4 manual arima model 14.79096
## 5                 RF 14.98619
## 3   auto.arima model 15.03965
## 2 SES Multiplicative 15.09799
## 6                SVM 15.74482
## 7            XGBOOST 16.42254
## 1       SES Additive 19.37620
```

#lET'S MAKE A PREDICTION WITH ALL THE DATA

```
#Selecting all the power data without the day to predict ( which is empty)
all_data = data[1:4891,2:3]
colnames(all_data) <- c("power","temperature")

#Interpolation on all the data.
all_data[all_data == 0] <- NA
all_data=na_interpolation(all_data)
```

```r
#Make all_data a ts object
ts_data_all = ts(all_data,start=c(1,6), freq=(60*24)/15)

#Training of the model with the manual arima
fit_manual_arima=Arima(ts_data_all[,1], order=c(0,1,10), seasonal=c(0,1,1))

#Making the prediction
prev_manual_arima=forecast(fit_manual_arima,h=96)

#Let's see the result
autoplot(ts_data_all[,1])+autolayer(prev_manual_arima$mean,series=" auto SARIMA")
```



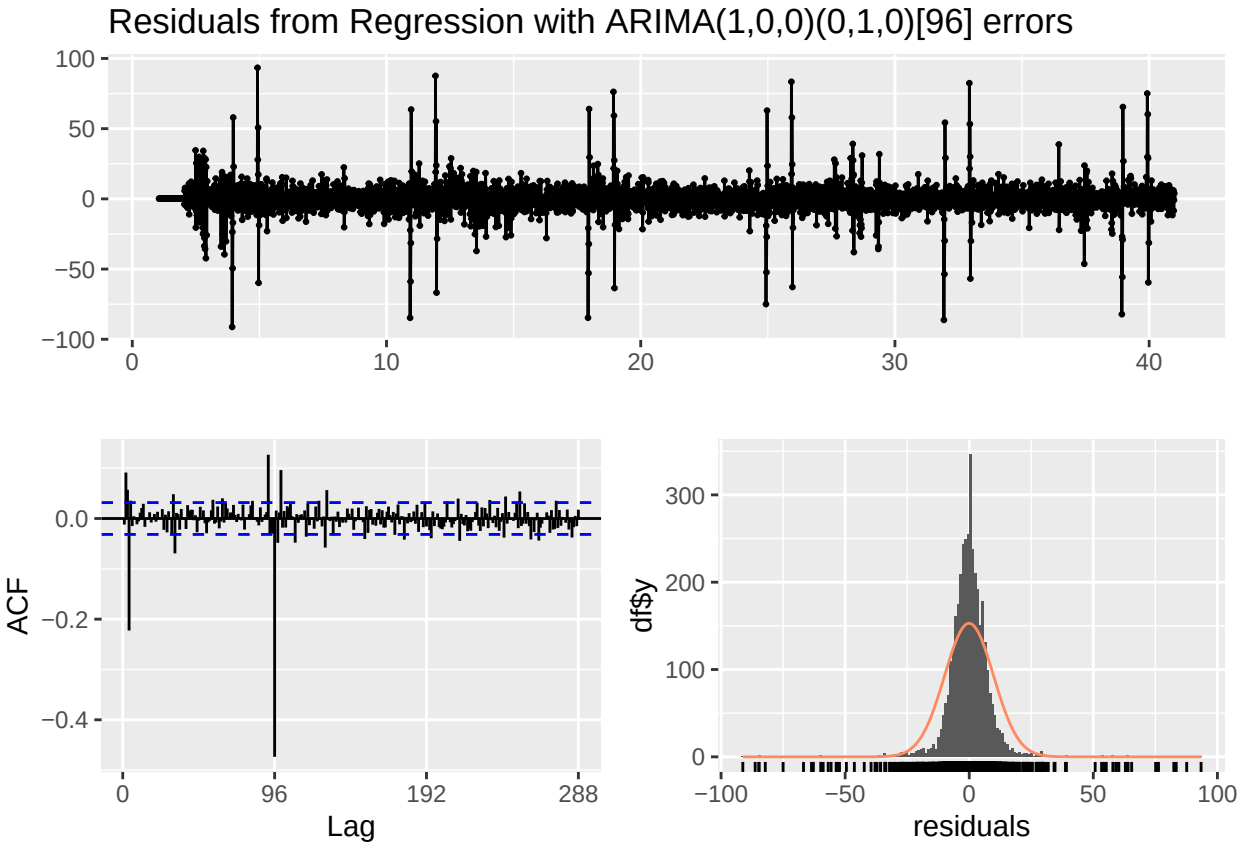

```r
#Save it in xlsx file
forecast_results <- data.frame(forecast_without_using_outdoor_Temperature =
                                 as.numeric(prev_manual_arima$mean))
write.xlsx(forecast_results, "BOA-THIEMELE.xlsx")
```

# VI- PREDICTION WITH COVARIATE : OUTDOOR TEMPERATURE

```r
#Let's redefine train and test set
ts_train_2=window(ts_data,start=c(1,6), end=c(40,96))

ts_test_2=window(ts_data,start=c(41,1), end=c(51,96))
ts_test_2[ts_test_2 == 0] <- NA
ts_test_2=na_interpolation(ts_test_2)
```

## VI-1- AUTO-ARIMA with covariate

```
#Applying auto Arima with covariate (xreg = temperature)
auto_arima_fit_covariate=auto.arima(ts_train_2[,"power"],xreg=ts_train_2[,"temperature"])
#Checking residuals
checkresiduals(auto_arima_fit_covariate,lag=20)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0)(0,1,0)[96] errors
## Q* = 249.48, df = 19, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 20
```

There is still some autocorrelations in the final modelization. we are going to see if we can do better with manual arima. but before let's analyse the model and make forecast with it

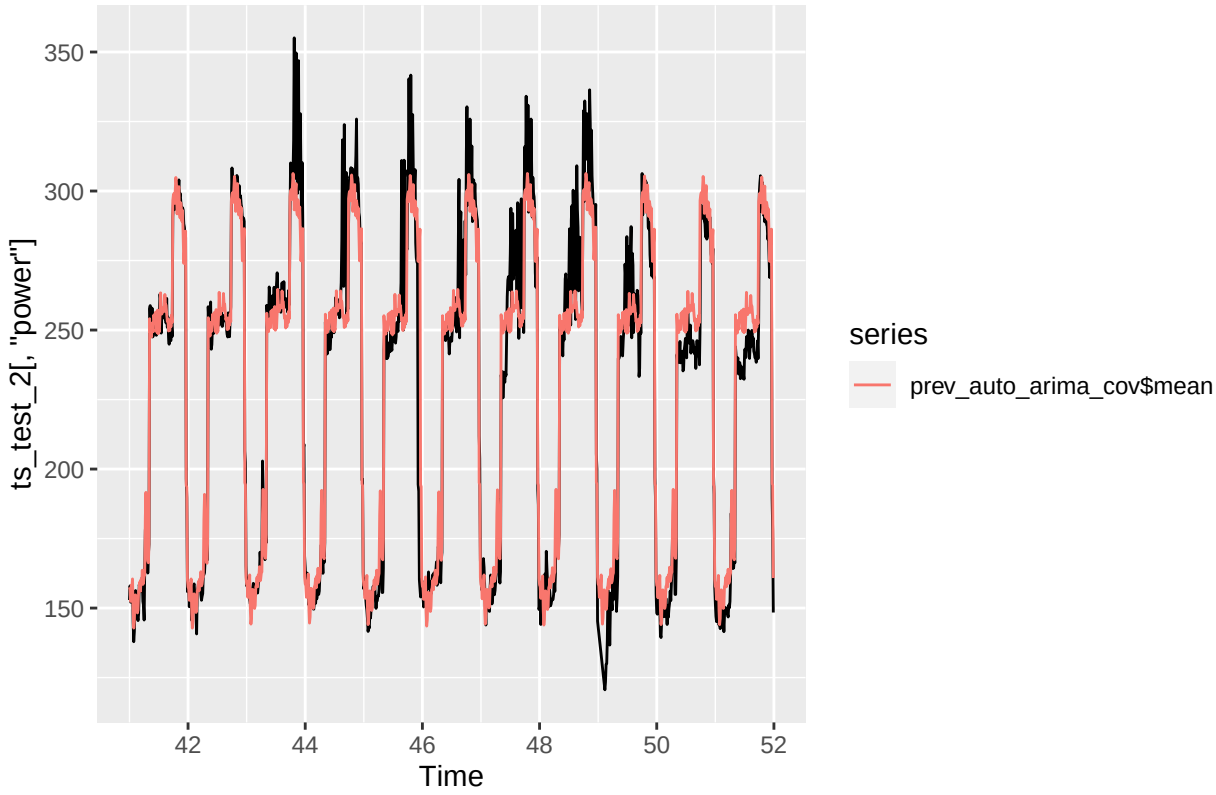
```
# Model analysis
auto_arima_fit_covariate
```

```
## Series: ts_train_2[, "power"]
## Regression with ARIMA(1,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1    xreg
##       0.7744  0.3238
## s.e.  0.0104  0.2441
```

```
##
## sigma^2 = 100.1:  log likelihood = -13916.47
## AIC=27838.94   AICc=27838.94   BIC=27857.62
```

AIC is 27838.94 and all coefficient are significant, the arima chosen is the same as previous ARIMA(1,0,0)(0,1,0)[96] .Let's make prediction with the model

```
#Forecasting with the build model
prev_auto_arima_cov=forecast(auto_arima_fit_covariate,h=96*11,xreg=ts_test_2[,"temperature"])

#Ploting the results
autoplot(ts_test_2[,"power"])+autolayer(prev_auto_arima_cov$mean)
```




```
#Computing RMSE of predictions with covariate
RMSE_covariate = sqrt(mean((prev_auto_arima_cov$mean-ts_test_2[,"power"])^2))
RMSE_covariate
```

```
## [1] 14.86413
```

# VI-2- MANUAL ARIMA WITH COVARIATE

```
#Removing covariate effect
lstm_fit=tslm(power~temperature+trend+season ,data=ts_train_2)
summary(lstm_fit)
```

```
##
## Call:
## tslm(formula = power ~ temperature + trend + season, data = ts_train_2)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -112.381   -4.565    0.205    4.631   59.010
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.562e+02  2.100e+00  74.398  < 2e-16 ***
## temperature  1.158e+00  9.775e-02  11.849  < 2e-16 ***
## trend       -4.875e-03  1.741e-04 -28.002  < 2e-16 ***
## season2     -6.668e-01  2.641e+00  -0.252  0.80070
## season3     -6.834e+00  2.641e+00  -2.587  0.00971 **
## season4     -3.391e-01  2.641e+00  -0.128  0.89785
## season5      2.848e+00  2.641e+00   1.078  0.28101
## season6      3.093e+00  2.625e+00   1.178  0.23869
## season7     -6.504e+00  2.625e+00  -2.478  0.01327 *
## season8     -6.572e+00  2.625e+00  -2.503  0.01234 *
## season9     -3.132e+00  2.625e+00  -1.193  0.23291
## season10    -4.103e+00  2.625e+00  -1.563  0.11814
## season11    -1.361e+00  2.625e+00  -0.518  0.60421
## season12    -2.156e+00  2.625e+00  -0.821  0.41155
## season13    -4.712e-01  2.625e+00  -0.179  0.85757
## season14    -5.117e+00  2.626e+00  -1.948  0.05144 .
## season15    -6.554e+00  2.626e+00  -2.496  0.01261 *
## season16    -8.845e-01  2.626e+00  -0.337  0.73627
## season17     5.479e-01  2.626e+00   0.209  0.83475
## season18    -3.644e-01  2.627e+00  -0.139  0.88969
## season19    -1.300e+00  2.627e+00  -0.495  0.62088
## season20     4.478e-01  2.627e+00   0.170  0.86466
## season21    -4.230e-02  2.627e+00  -0.016  0.98715
## season22     9.778e-01  2.628e+00   0.372  0.70988
## season23     2.015e+00  2.628e+00   0.767  0.44328
## season24     4.280e+00  2.628e+00   1.629  0.10350
## season25     3.832e+00  2.628e+00   1.458  0.14487
## season26     6.949e+00  2.628e+00   2.644  0.00823 **
## season27     1.378e+01  2.628e+00   5.244 1.66e-07 ***
## season28     1.581e+01  2.628e+00   6.016 1.96e-09 ***
## season29     1.598e+01  2.628e+00   6.081 1.31e-09 ***
## season30     2.186e+01  2.628e+00   8.318  < 2e-16 ***
## season31     2.165e+01  2.628e+00   8.236 2.43e-16 ***
## season32     1.624e+01  2.628e+00   6.179 7.16e-10 ***
## season33     1.991e+01  2.628e+00   7.577 4.44e-14 ***
## season34     1.040e+02  2.628e+00  39.580  < 2e-16 ***
## season35     1.017e+02  2.628e+00  38.721  < 2e-16 ***
## season36     9.907e+01  2.628e+00  37.700  < 2e-16 ***
## season37     9.873e+01  2.628e+00  37.572  < 2e-16 ***
## season38     1.014e+02  2.624e+00  38.633  < 2e-16 ***
## season39     9.647e+01  2.624e+00  36.758  < 2e-16 ***
## season40     9.893e+01  2.624e+00  37.695  < 2e-16 ***
## season41     9.960e+01  2.624e+00  37.951  < 2e-16 ***
## season42     9.584e+01  2.626e+00  36.504  < 2e-16 ***
## season43     9.657e+01  2.626e+00  36.781  < 2e-16 ***
## season44     9.821e+01  2.626e+00  37.407  < 2e-16 ***
## season45     9.858e+01  2.626e+00  37.546  < 2e-16 ***
## season46     1.005e+02  2.630e+00  38.219  < 2e-16 ***
## season47     9.857e+01  2.630e+00  37.478  < 2e-16 ***
```

```
## season48     9.924e+01  2.630e+00  37.734  < 2e-16 ***
## season49     1.000e+02  2.630e+00  38.037  < 2e-16 ***
## season50     9.941e+01  2.637e+00  37.695  < 2e-16 ***
## season51     1.028e+02  2.637e+00  38.962  < 2e-16 ***
## season52     1.019e+02  2.637e+00  38.647  < 2e-16 ***
## season53     1.005e+02  2.637e+00  38.098  < 2e-16 ***
## season54     1.020e+02  2.642e+00  38.597  < 2e-16 ***
## season55     1.023e+02  2.642e+00  38.724  < 2e-16 ***
## season56     1.015e+02  2.642e+00  38.419  < 2e-16 ***
## season57     1.009e+02  2.642e+00  38.200  < 2e-16 ***
## season58     1.010e+02  2.648e+00  38.149  < 2e-16 ***
## season59     1.013e+02  2.648e+00  38.274  < 2e-16 ***
## season60     1.010e+02  2.648e+00  38.132  < 2e-16 ***
## season61     1.008e+02  2.648e+00  38.062  < 2e-16 ***
## season62     1.005e+02  2.649e+00  37.955  < 2e-16 ***
## season63     1.002e+02  2.649e+00  37.814  < 2e-16 ***
## season64     1.002e+02  2.649e+00  37.821  < 2e-16 ***
## season65     1.001e+02  2.649e+00  37.790  < 2e-16 ***
## season66     1.005e+02  2.644e+00  38.031  < 2e-16 ***
## season67     1.010e+02  2.644e+00  38.201  < 2e-16 ***
## season68     9.845e+01  2.644e+00  37.239  < 2e-16 ***
## season69     9.845e+01  2.644e+00  37.237  < 2e-16 ***
## season70     1.198e+02  2.636e+00  45.456  < 2e-16 ***
## season71     1.337e+02  2.636e+00  50.734  < 2e-16 ***
## season72     1.450e+02  2.636e+00  54.998  < 2e-16 ***
## season73     1.443e+02  2.636e+00  54.755  < 2e-16 ***
## season74     1.420e+02  2.629e+00  54.012  < 2e-16 ***
## season75     1.407e+02  2.629e+00  53.500  < 2e-16 ***
## season76     1.398e+02  2.629e+00  53.166  < 2e-16 ***
## season77     1.402e+02  2.629e+00  53.316  < 2e-16 ***
## season78     1.459e+02  2.627e+00  55.547  < 2e-16 ***
## season79     1.419e+02  2.627e+00  54.022  < 2e-16 ***
## season80     1.406e+02  2.627e+00  53.511  < 2e-16 ***
## season81     1.391e+02  2.627e+00  52.926  < 2e-16 ***
## season82     1.391e+02  2.626e+00  52.981  < 2e-16 ***
## season83     1.368e+02  2.626e+00  52.100  < 2e-16 ***
## season84     1.367e+02  2.626e+00  52.052  < 2e-16 ***
## season85     1.357e+02  2.626e+00  51.657  < 2e-16 ***
## season86     1.337e+02  2.625e+00  50.938  < 2e-16 ***
## season87     1.326e+02  2.625e+00  50.525  < 2e-16 ***
## season88     1.315e+02  2.625e+00  50.087  < 2e-16 ***
## season89     1.293e+02  2.625e+00  49.243  < 2e-16 ***
## season90     1.123e+02  2.624e+00  42.776  < 2e-16 ***
## season91     1.109e+02  2.624e+00  42.267  < 2e-16 ***
## season92     1.048e+02  2.624e+00  39.949  < 2e-16 ***
## season93     1.060e+02  2.624e+00  40.374  < 2e-16 ***
## season94     3.007e+01  2.624e+00  11.457  < 2e-16 ***
## season95     3.221e+01  2.624e+00  12.274  < 2e-16 ***
## season96     3.037e+00  2.624e+00   1.157  0.24731
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.66 on 3737 degrees of freedom
## Multiple R-squared:  0.9597, Adjusted R-squared:  0.9586
```

```
## F-statistic: 917.3 on 97 and 3737 DF,  p-value: < 2.2e-16
```

```
lstm_fit_2 = tslm(power~temperature, data=ts_train_2)
summary(lstm_fit_2)
```

```
##
## Call:
## tslm(formula = power ~ temperature, data = ts_train_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -121.572  -42.732    2.868   42.908  112.468
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 125.0210     3.4492   36.25   <2e-16 ***
## temperature  10.0261     0.3146   31.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51 on 3833 degrees of freedom
## Multiple R-squared:  0.2094, Adjusted R-squared:  0.2092
## F-statistic:  1015 on 1 and 3833 DF,  p-value: < 2.2e-16
```

```
#Cross validation of the previous models
CV(lstm_fit_2)
```

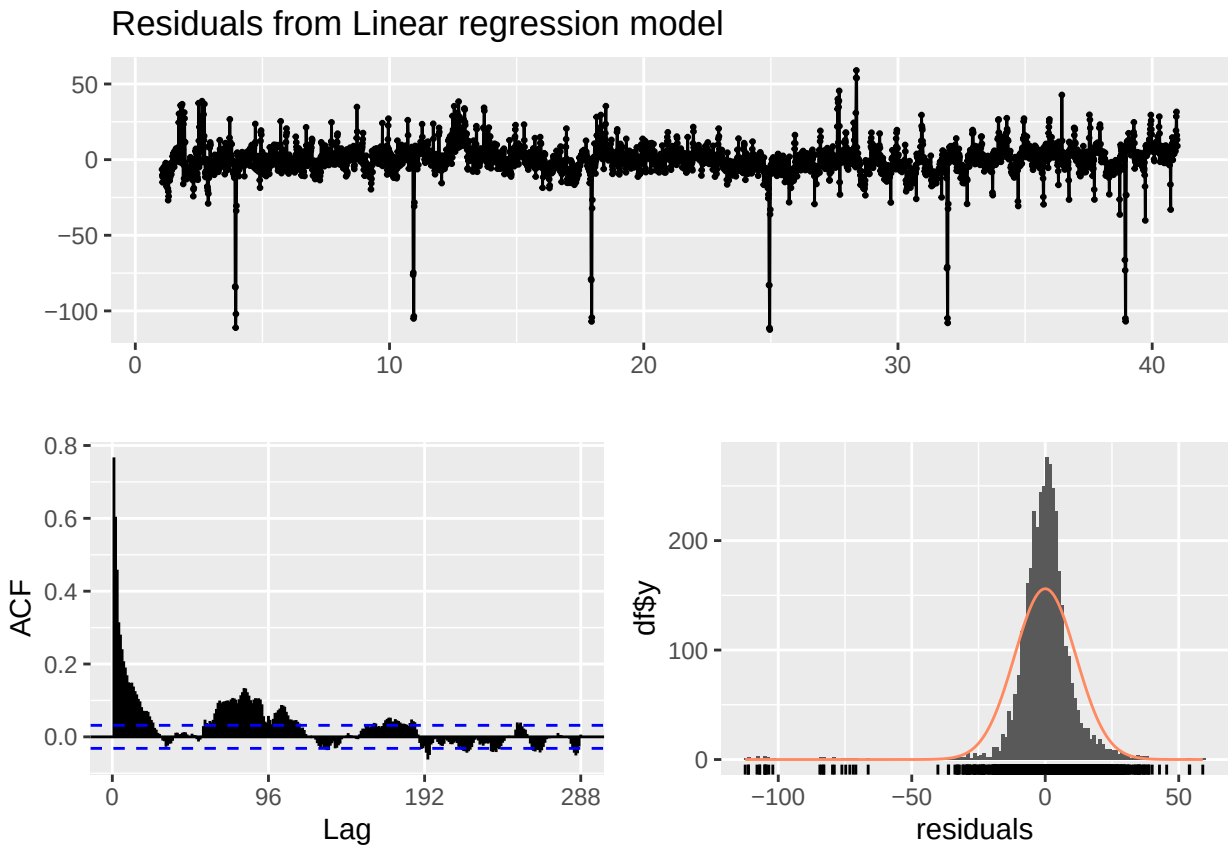
```
##           CV          AIC         AICc          BIC        AdjR2
## 2.601556e+03 3.016053e+04 3.016053e+04 3.017928e+04 2.092329e-01
```

```
CV(lstm_fit)
```

```
##           CV          AIC         AICc          BIC        AdjR2
## 1.395655e+02 1.893885e+04 1.894415e+04 1.955779e+04 9.586455e-01
```

The second regression has the best BIC, so we will proceed with that one. we will use regression with season and trend.

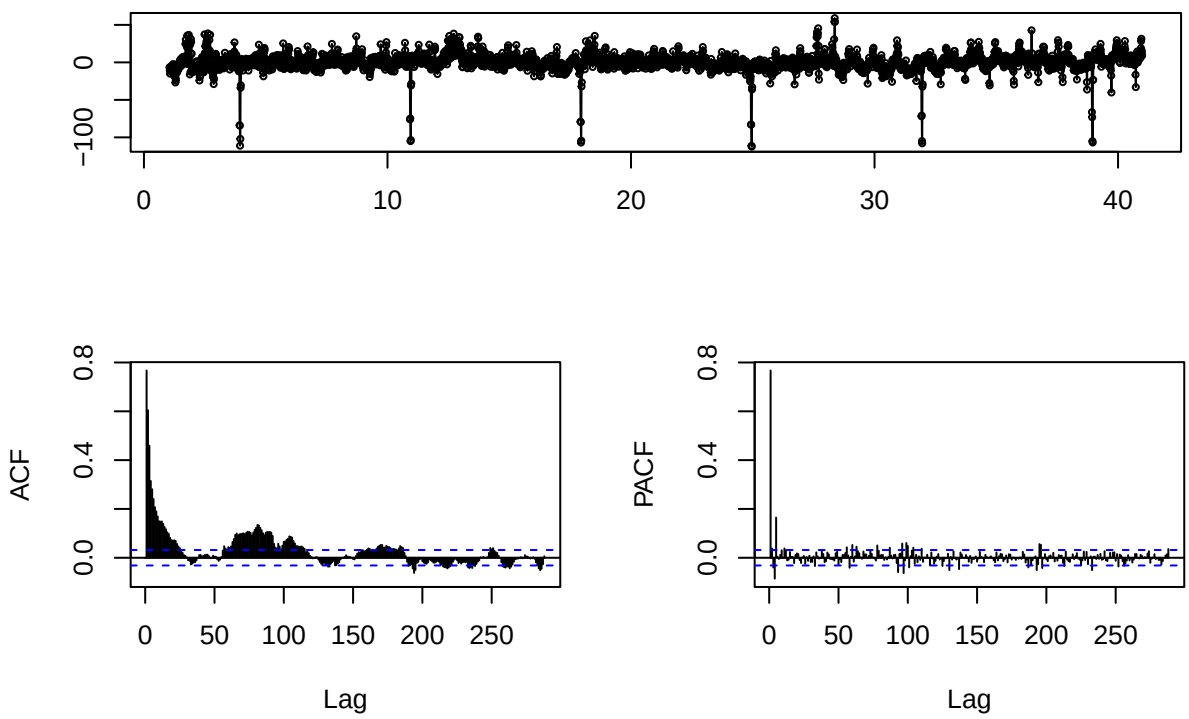
```
#Looking at the residuals
checkresiduals(lstm_fit)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 2432.7, df = 192, p-value < 2.2e-16
```

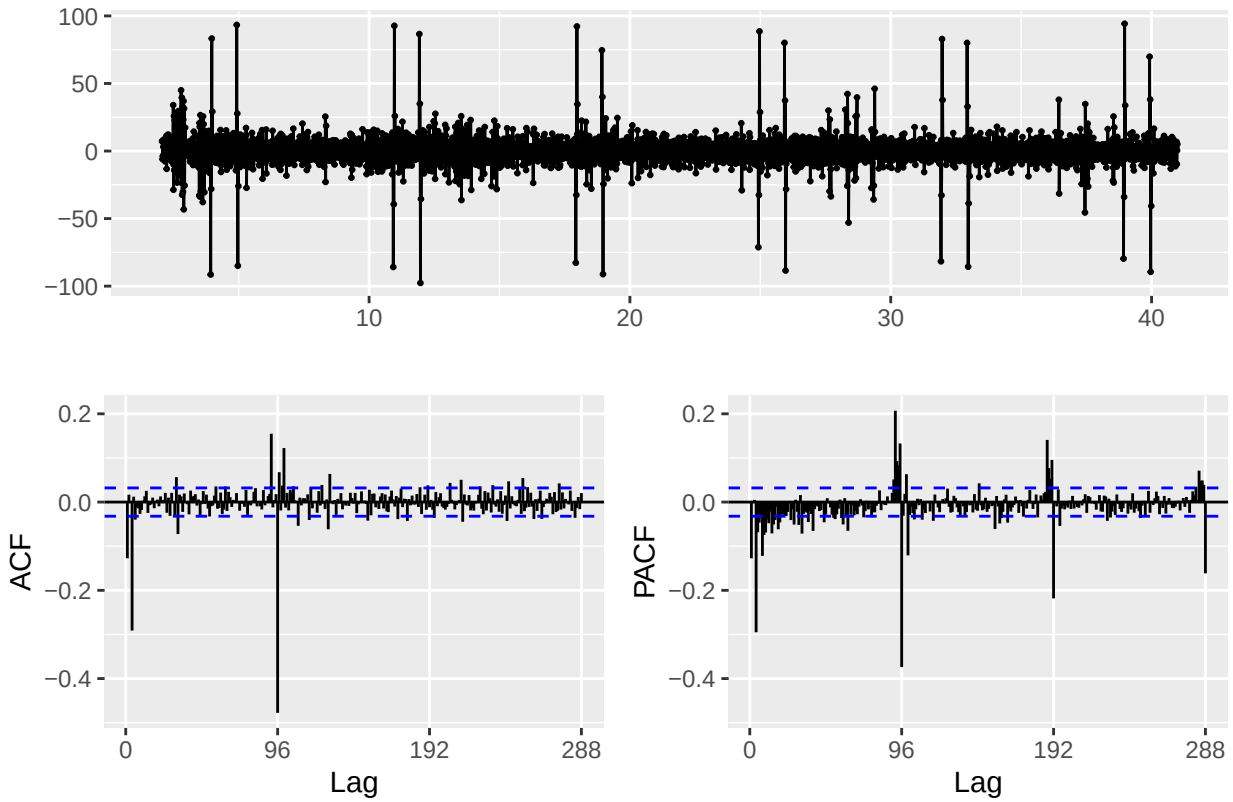
There is a trend and a seasonnal pattern, we will differentiate twice to suppress them as previously.

```
tsdisplay(lstm_fit$residuals)
```



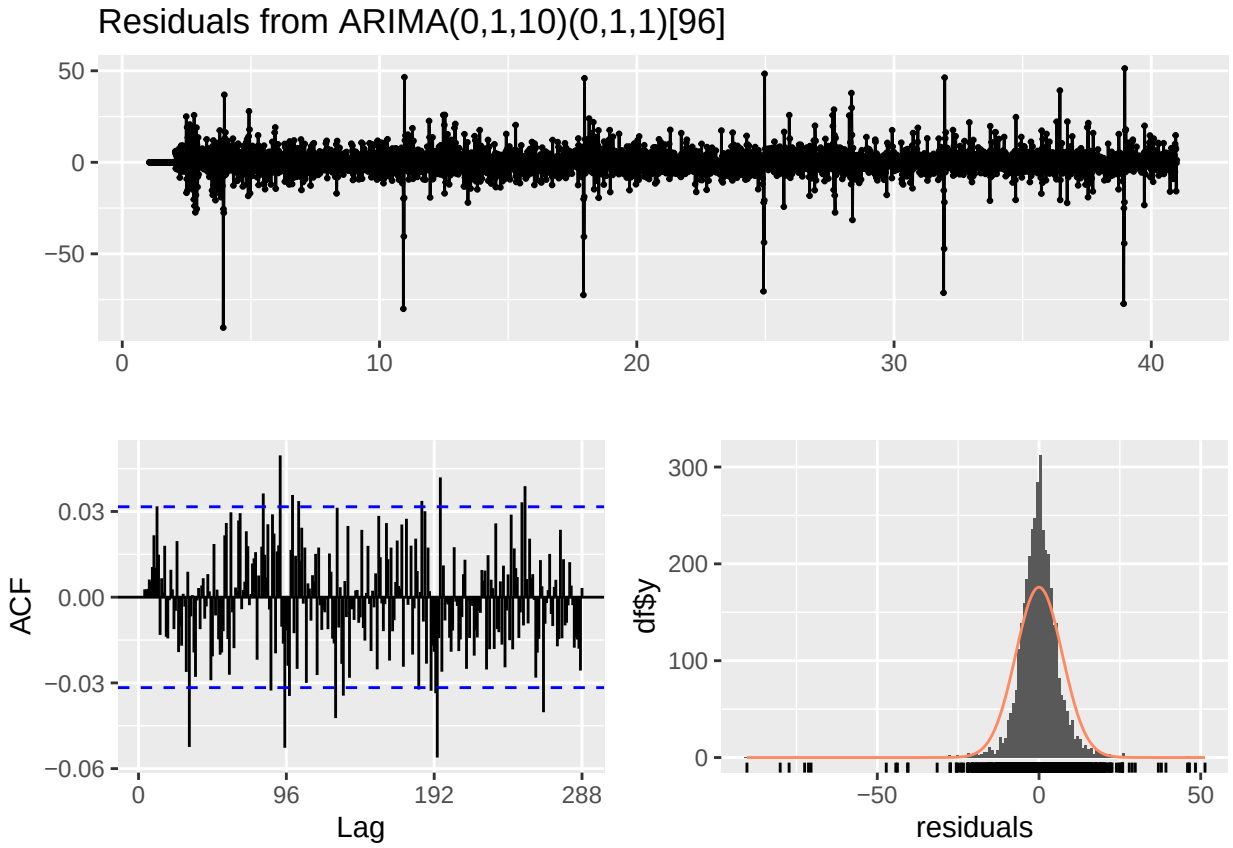

```
# Differentiating the residuals
tmp=diff(diff(lstm_fit$residuals,lag = 96,differences = 1))
```

```
#Let's plot the result
ggtsdisplay(tmp)
```




We got similar result as previously with manual arima, we will try Sarima(0,1,10)(0,1,1)

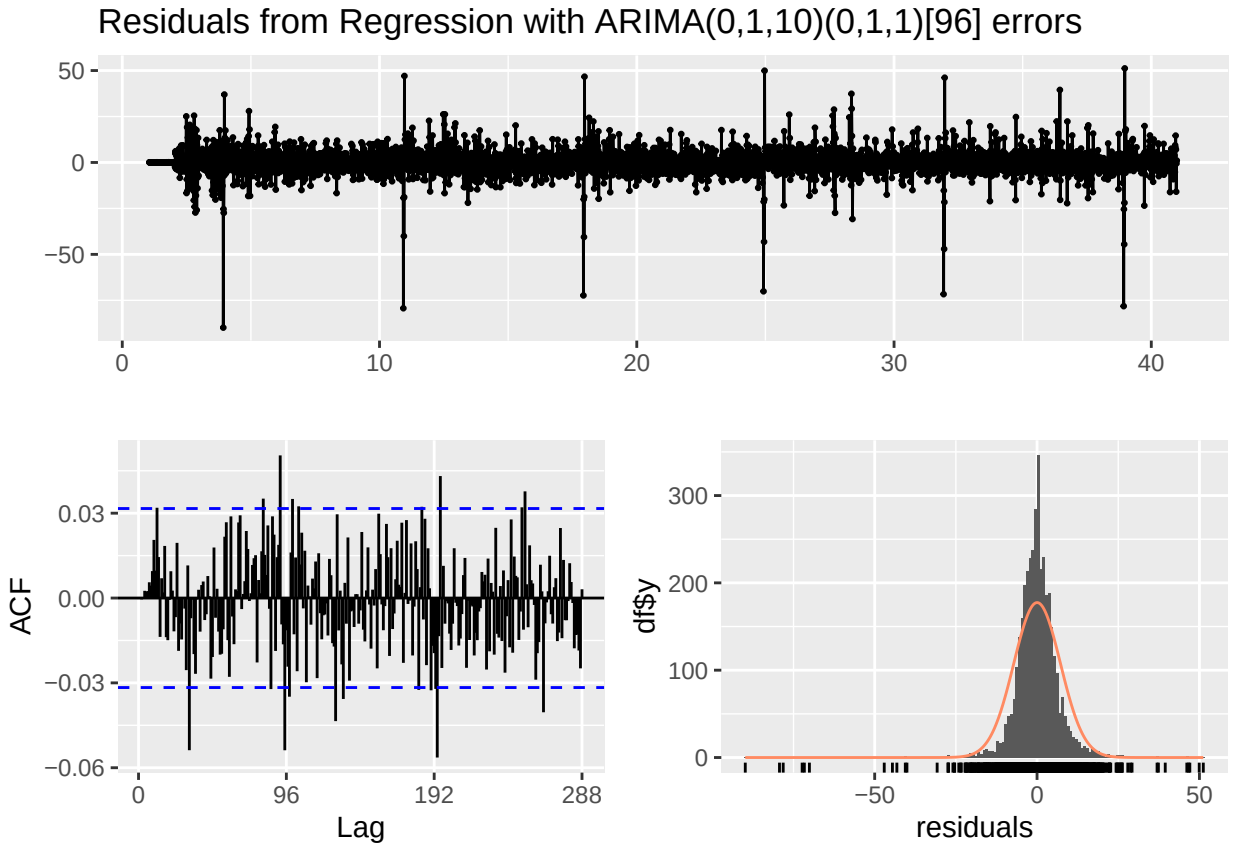
```
#Running the model and checking residuals
manual_arima_with_covariate_fit=Arima(lstm_fit$residuals,order=c(0,1,10), seasonal = c(0,1,1))
checkresiduals(manual_arima_with_covariate_fit,20)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,10)(0,1,1)[96]
## Q* = 11.373, df = 9, p-value = 0.251
##
## Model df: 11.   Total lags used: 20
```

P=0.251, we completely capture all correlation and we can use it on the base model with covariate.

```
#Model with covariate
manual_arima_with_covariate_fit=Arima(ts_train_2[,"power"],xreg=ts_train_2[,"temperature"],
                                      order=c(0,1,10),seasonal = c(0,1,1))
checkresiduals(manual_arima_with_covariate_fit,lag=20)
```

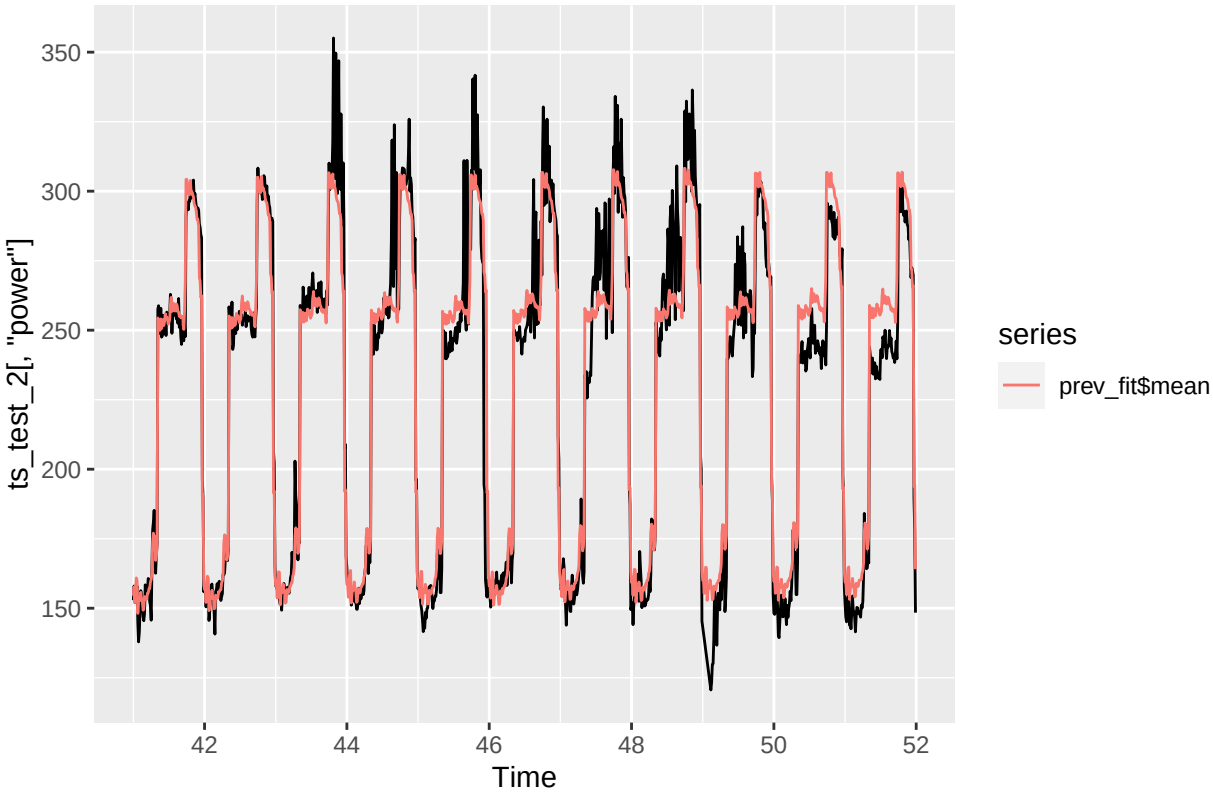



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,10)(0,1,1)[96] errors
## Q* = 11.164, df = 9, p-value = 0.2646
##
## Model df: 11.   Total lags used: 20
```

The p value is 0.2646, we capture all correlation. We can make prediction with the model and compute RMSE.

```
#Forecasting with the build model
prev_fit=forecast(manual_arima_with_covariate_fit,h=96*11,xreg=ts_test_2[,"temperature"])
```

```
#Ploting the results
autoplot(ts_test_2[,"power"])+autolayer(prev_fit$mean)
```




```
#Computing RMSE
RMSE_manual_arima_covariate = sqrt(mean((ts_test-prev_fit$mean)^2))
RMSE_manual_arima_covariate
```

```
## [1] 14.67925
```

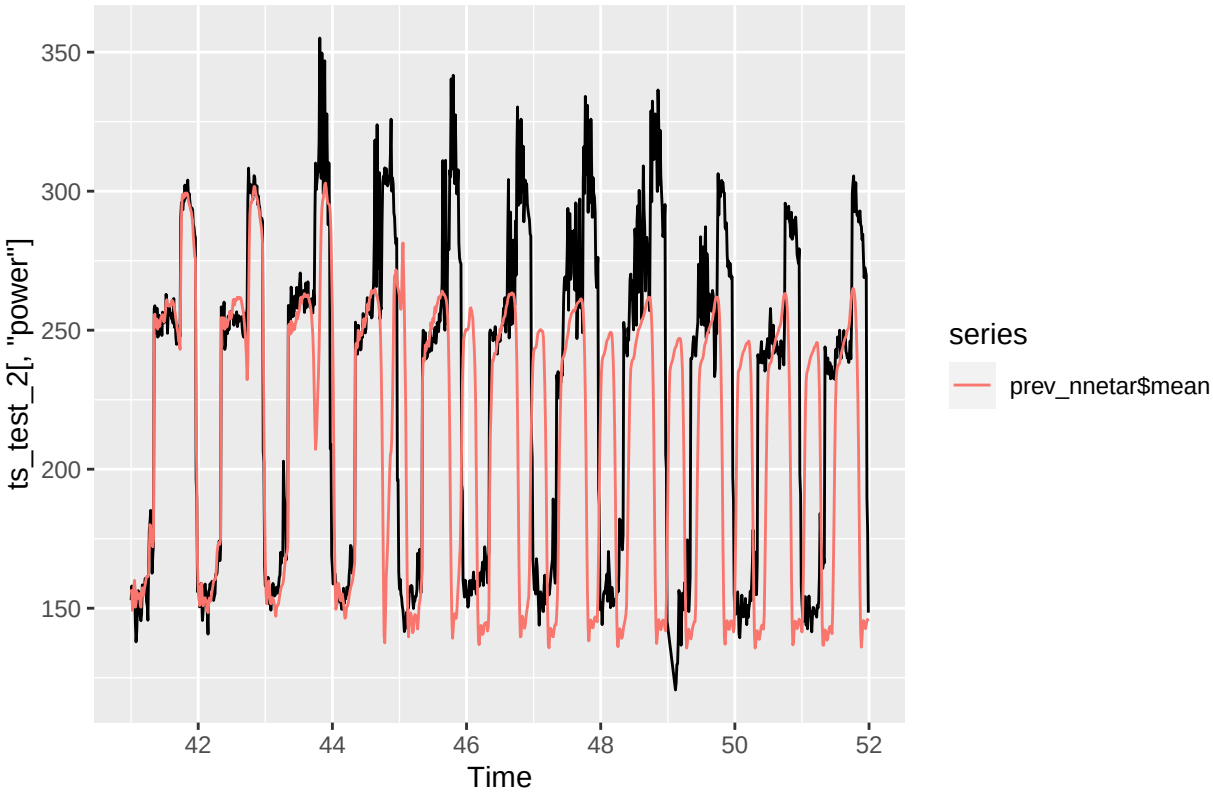
the RMSE is lower than the one of the auto arima with covariate.

# VI-3 NEURAL NETWORK : NNETAR WITH COVARIATE

```
nnetar_fit = nnetar(ts_train_2[,"power"], xreg= ts_train_2[,"temperature"])
print(nnetar_fit)
```

```
## Series: ts_train_2[, "power"]
## Model:  NNAR(20,1,12)[96]
## Call:   nnetar(y = ts_train_2[, "power"], xreg = ts_train_2[, "temperature"])
##
## 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 47.79
```

```
prev_nnetar=forecast(nnetar_fit,xreg=tail(ts_train_2[,"temperature"],96*11))
autoplot(ts_test_2[,"power"])+autolayer(prev_nnetar$mean)
```

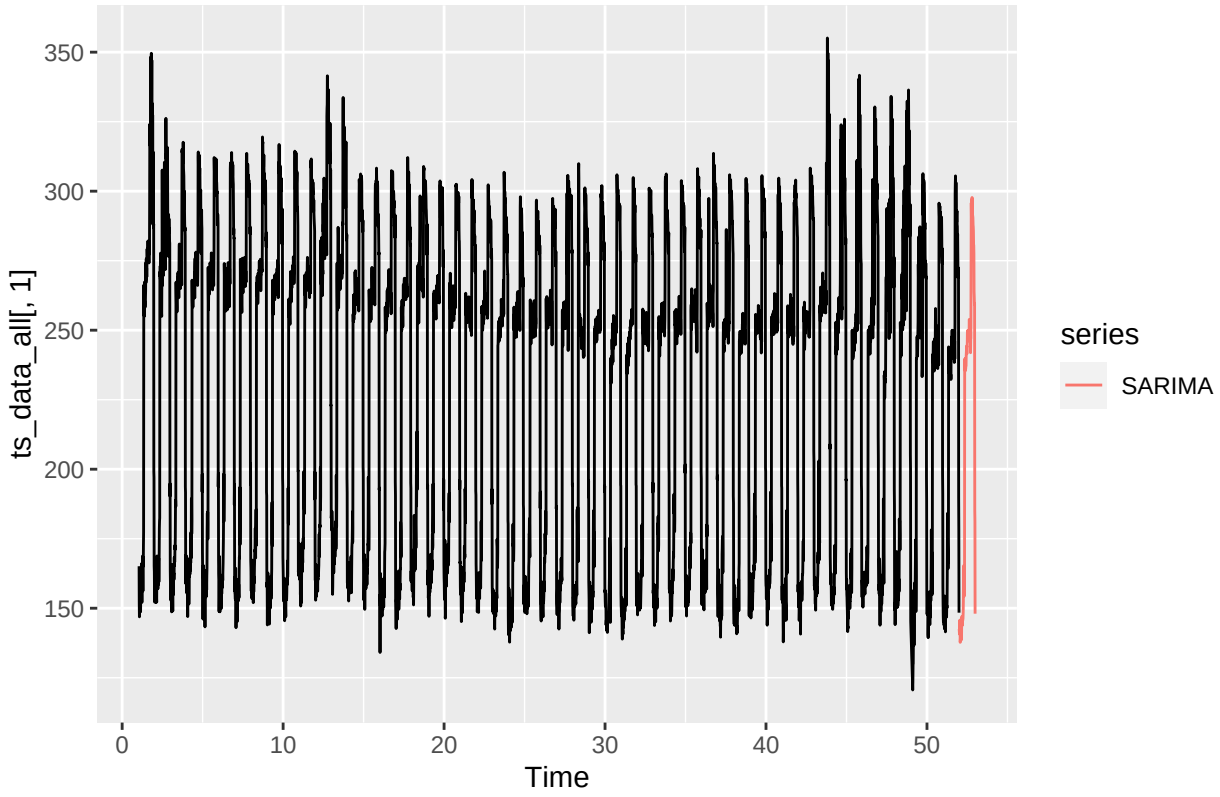



```
RMSE_nnetar = sqrt(mean((ts_test-prev_nnetar$mean)^2))
RMSE_nnetar
```

```
## [1] 65.99621
```

The RMSE is really bad and we can see on the graph that the model doesn't fit well the data. We will then make the final prediction with the manual arima with covariate

```
#Training of the model with the manual arima
fit_manual_arima_covariate=Arima(ts_data_all[,1], order=c(0,1,10), seasonal=c(0,1,1),
                                 xreg=ts_data_all[,2])

#Making the prediction
prev_manual_arima_covariate=forecast(fit_manual_arima_covariate,h=96,xreg=tail(ts_data_all[,2],96))

#Let's see the result
autoplot(ts_data_all[,1])+autolayer(prev_manual_arima_covariate$mean,series="SARIMA")
```




```
# Load the workbook
wb <- loadWorkbook('BOA-THIEMELE.xlsx')

# Add a title in the first cell of the second column (B1)
writeData(wb, sheet = 1, x = "Manual_arima_with_temperature_as_covariate", startCol = 2, startRow = 1, 

# Write the data starting from the second row
writeData(wb, sheet = 1, x = prev_manual_arima_covariate$mean, startCol = 2, startRow = 2, colNames = F
prev_manual_arima_covariate
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 52.00000       142.4897 132.0355 152.9438 126.5015 158.4779
## 52.01042       141.6033 129.0499 154.1566 122.4046 160.8019
## 52.02083       140.7056 127.0694 154.3419 119.8508 161.5604
## 52.03125       143.2921 128.5025 158.0816 120.6734 165.9108
## 52.04167       145.7324 130.6879 160.7769 122.7239 168.7409
## 52.05208       144.9588 129.7301 160.1874 121.6685 168.2490
## 52.06250       137.7387 122.3682 153.1093 114.2316 161.2459
## 52.07292       138.6632 123.2143 154.1120 115.0362 162.2902
## 52.08333       139.6154 124.0844 155.1463 115.8628 163.3679
## 52.09375       142.0495 126.4933 157.6057 118.2583 165.8407
## 52.10417       143.9564 128.3936 159.5192 120.1551 167.7577
## 52.11458       142.6880 127.1186 158.2574 118.8766 166.4993
## 52.12500       142.2234 126.6474 157.7994 118.4020 166.0448
## 52.13542       140.4350 124.8525 156.0176 116.6035 164.2665
## 52.14583       138.6718 123.0826 154.2610 114.8302 162.5134
## 52.15625       144.1493 128.5536 159.7451 120.2977 168.0010
## 52.16667       146.9886 131.3863 162.5909 123.1269 170.8502
## 52.17708       145.4117 129.8028 161.0205 121.5399 169.2834
```

```
## 52.18750       144.7489 129.1335 160.3644 120.8671 168.6307
## 52.19792       143.8374 128.2154 159.4595 119.9456 167.7293
## 52.20833       145.8613 130.2327 161.4899 121.9594 169.7631
## 52.21875       145.5749 129.9397 161.2100 121.6630 169.4868
## 52.22917       145.9673 130.3256 161.6090 122.0454 169.8892
## 52.23958       146.3060 130.6577 161.9543 122.3740 170.2379
## 52.25000       148.6846 133.0298 164.3394 124.7426 172.6266
## 52.26042       152.5890 136.9277 168.2504 128.6370 176.5410
## 52.27083       164.1415 148.4736 179.8094 140.1795 188.1035
## 52.28125       163.6237 147.9492 179.2982 139.6517 187.5957
## 52.29167       165.3690 149.6879 181.0500 141.3869 189.3510
## 52.30208       162.9312 147.2437 178.6188 138.9392 186.9233
## 52.31250       154.3199 138.6258 170.0140 130.3179 178.3219
## 52.32292       155.4215 139.7209 171.1221 131.4095 179.4336
## 52.33333       159.9916 144.2845 175.6988 135.9696 184.0136
## 52.34375       239.6430 223.9293 255.3567 215.6110 263.6750
## 52.35417       238.0564 222.3362 253.7766 214.0144 262.0984
## 52.36458       235.7724 220.0457 251.4991 211.7205 259.8244
## 52.37500       235.3457 219.6124 251.0789 211.2838 259.4076
## 52.38542       239.8644 224.1246 255.6041 215.7925 263.9362
## 52.39583       235.9777 220.2314 251.7239 211.8958 260.0595
## 52.40625       236.1651 220.4123 251.9179 212.0733 260.2569
## 52.41667       237.2780 221.5187 253.0373 213.1762 261.3798
## 52.42708       237.2959 221.5301 253.0617 213.1841 261.4076
## 52.43750       241.1042 225.3319 256.8765 216.9825 265.2259
## 52.44792       242.1544 226.3756 257.9332 218.0228 266.2860
## 52.45833       243.6452 227.8598 259.4305 219.5036 267.7867
## 52.46875       244.3591 228.5673 260.1509 220.2076 268.5106
## 52.47917       241.4863 225.6880 257.2846 217.3249 265.6477
## 52.48958       243.8546 228.0498 259.6594 219.6832 268.0259
## 52.50000       244.0538 228.2425 259.8651 219.8725 268.2351
## 52.51042       248.9750 233.1573 264.7928 224.7838 273.1662
## 52.52083       249.8375 234.0132 265.6618 225.6364 274.0386
## 52.53125       248.9952 233.1644 264.8259 224.7842 273.2062
## 52.54167       249.0705 233.2333 264.9077 224.8496 273.2914
## 52.55208       249.7232 233.8796 265.5669 225.4924 273.9541
## 52.56250       249.2221 233.3720 265.0723 224.9814 273.4629
## 52.57292       249.6352 233.7785 265.4918 225.3845 273.8858
## 52.58333       247.5968 231.7337 263.4599 223.3363 271.8573
## 52.59375       248.8520 232.9824 264.7215 224.5815 273.1224
## 52.60417       246.8523 230.9762 262.7283 222.5720 271.1326
## 52.61458       246.8906 231.0081 262.7731 222.6004 271.1808
## 52.62500       253.8698 237.9808 269.7587 229.5697 278.1698
## 52.63542       253.8120 237.9166 269.7074 229.5021 278.1219
## 52.64583       250.9272 235.0254 266.8291 226.6074 275.2470
## 52.65625       248.4036 232.4953 264.3119 224.0739 272.7332
## 52.66667       245.9835 230.0688 261.8983 221.6440 270.3230
## 52.67708       245.7713 229.8501 261.6925 221.4219 270.1207
## 52.68750       250.9316 235.0040 266.8593 226.5724 275.2908
## 52.69792       245.6895 229.7554 261.6236 221.3204 270.0585
## 52.70833       241.9002 225.9597 257.8407 217.5213 266.2791
## 52.71875       246.8476 230.9007 262.7946 222.4589 271.2364
## 52.72917       247.1206 231.1672 263.0739 222.7220 271.5191
## 52.73958       272.6004 256.6406 288.5602 248.1920 297.0088
```

```
## 52.75000       295.7570 279.7908 311.7233 271.3388 320.1753
## 52.76042       291.2404 275.2677 307.2130 266.8123 315.6684
## 52.77083       296.8830 280.9039 312.8621 272.4451 321.3209
## 52.78125       292.7625 276.7771 308.7480 268.3149 317.2102
## 52.79167       291.0746 275.0827 307.0665 266.6171 315.5321
## 52.80208       297.7261 281.7278 313.7244 273.2588 322.1934
## 52.81250       294.2770 278.2723 310.2817 269.7999 318.7541
## 52.82292       292.1714 276.1602 308.1825 267.6845 316.6583
## 52.83333       291.3546 275.3371 307.3721 266.8579 315.8513
## 52.84375       291.0677 275.0438 307.0917 266.5612 315.5742
## 52.85417       290.4597 274.4294 306.4901 265.9434 314.9760
## 52.86458       287.8402 271.8035 303.8770 263.3142 312.3663
## 52.87500       287.2178 271.1747 303.2610 262.6820 311.7537
## 52.88542       285.8220 269.7725 301.8716 261.2764 310.3677
## 52.89583       280.8251 264.7692 296.8811 256.2697 305.3805
## 52.90625       280.5212 264.4589 296.5835 255.9561 305.0864
## 52.91667       277.4777 261.4090 293.5464 252.9028 302.0527
## 52.92708       263.8961 247.8210 279.9712 239.3114 288.4808
## 52.93750       261.8925 245.8110 277.9739 237.2980 286.4869
## 52.94792       259.2238 243.1360 275.3116 234.6196 283.8280
## 52.95833       259.6510 243.5569 275.7452 235.0371 284.2650
## 52.96875       185.2712 169.1706 201.3718 160.6475 209.8949
## 52.97917       181.2150 165.1080 197.3219 156.5815 205.8484
## 52.98958       148.0358 131.9225 164.1491 123.3926 172.6789
```

```r
# Save the workbook
saveWorkbook(wb, 'BOA-THIEMELE.xlsx', overwrite = TRUE)
```