Time Series Forecasting

Martial KOUASSI

2024-08-19

#Introduction

Welcome to my report on time series forecasting to forecast electricity consumption (kW) of one building for 2/21/2010. I will use the historic electricity consumption data from 1/1/2010 1:15 to 2/20/2010 23:45. Two forecasts should be returned : ##1. The first one without using outdoor temperature, ##2. The second one using outdoor temperature.

I will follow the next steps: - Use Exponential Smoothing models to forcast electricity consumption - Use SARIMA models to forcast electricity consumption .The SARIMA model requires a variation of the data and its own EDA to estimate the model parameters - Use neural network models and other machine learning models to forcast electricity consumption - Use Use LSTM, deep LSTM and Dynamic regression model to forcast electricity consumption - Compare the results from the different models, draw key insights and outline next steps

#INSTALL PACKAGES

```
install.packages("openxlsx")
```

```
## Error in install.packages : Updating loaded packages
```

```
library(openxlsx)
```

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

```
install.packages("readxl")
```

```
## Installing package into 'C:/Users/marti/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

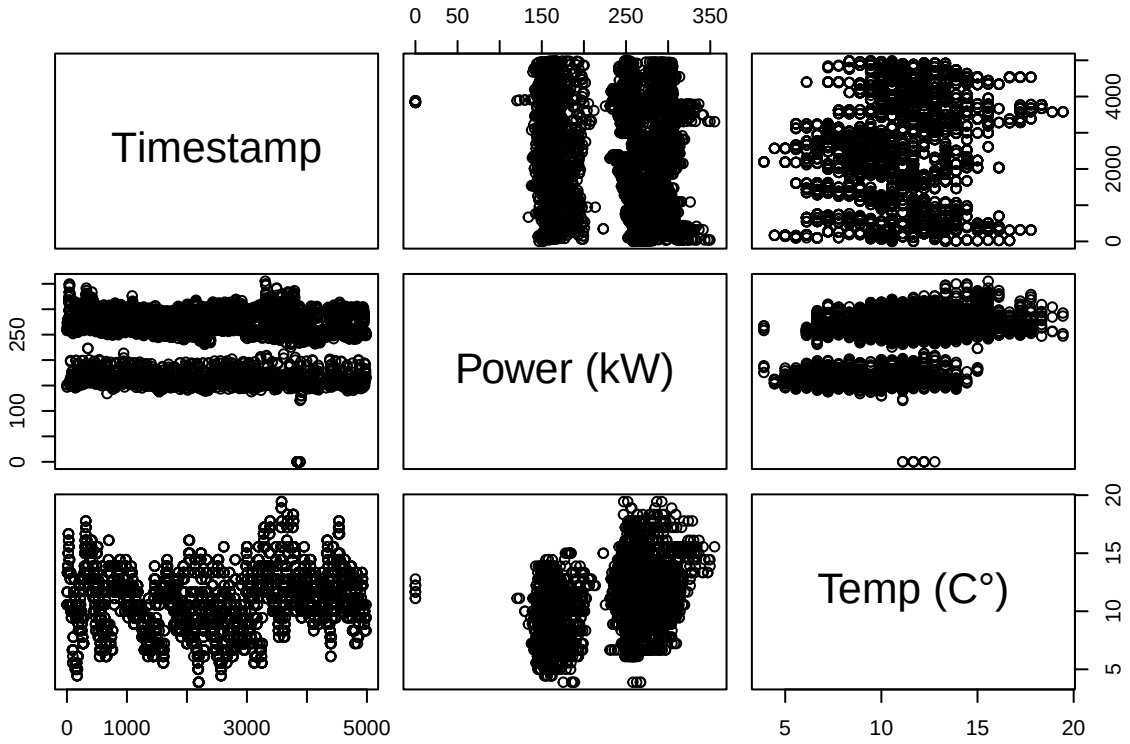
```
## package 'readxl' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\marti\AppData\Local\Temp\RtmpwVGoLX\downloaded_packages
```

```
library(readxl)
```

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

#LOAD AND PLOT THE DATA

```
data = read_excel('2023-11-Elec-train.xlsx')
plot(data)
```



#Part 1: Forecast electricity consumption (kW) for 2/21/2010 without using outdoor temperature

let's plot and create the time series object

```
library(forecast)
```

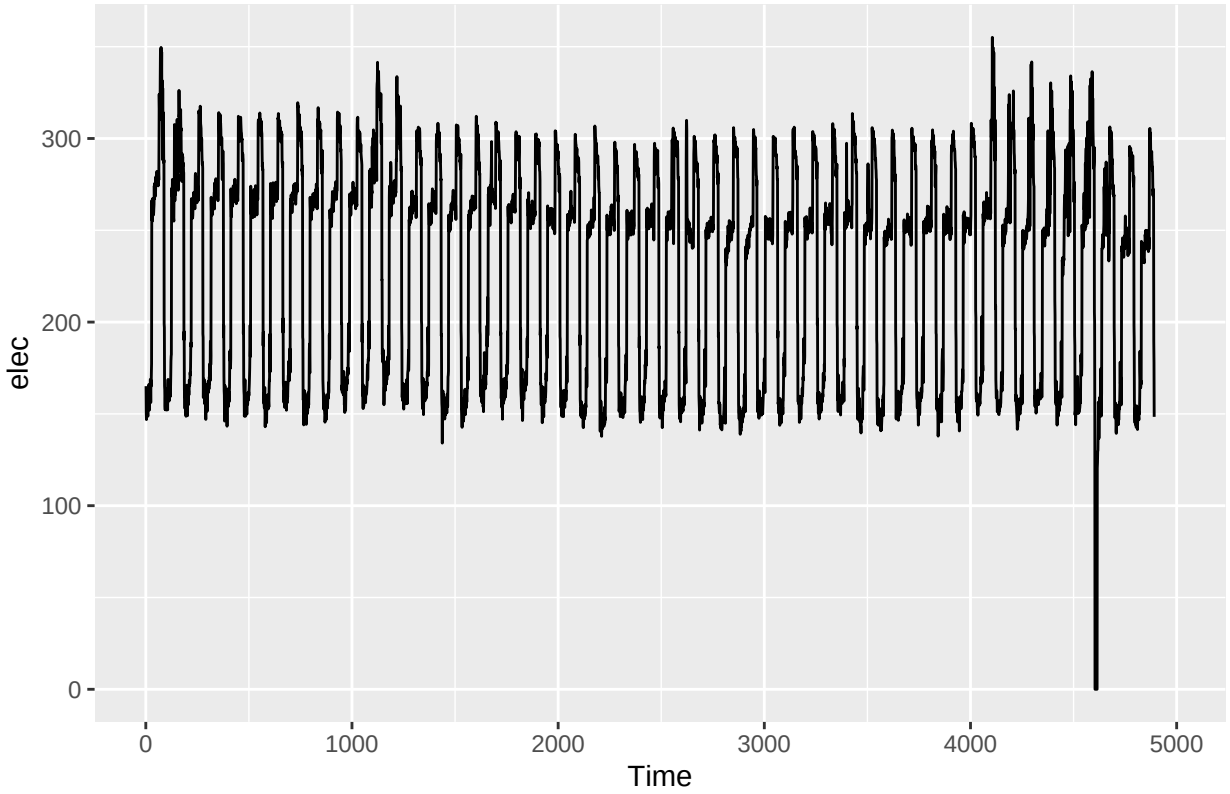
```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## This is forecast 8.22.0
##   Need help getting started? Try the online textbook FPP:
##   http://otexts.com/fpp2/
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
df=data
elec = ts(df$`Power (kW)`)
autoplot(elec)
```



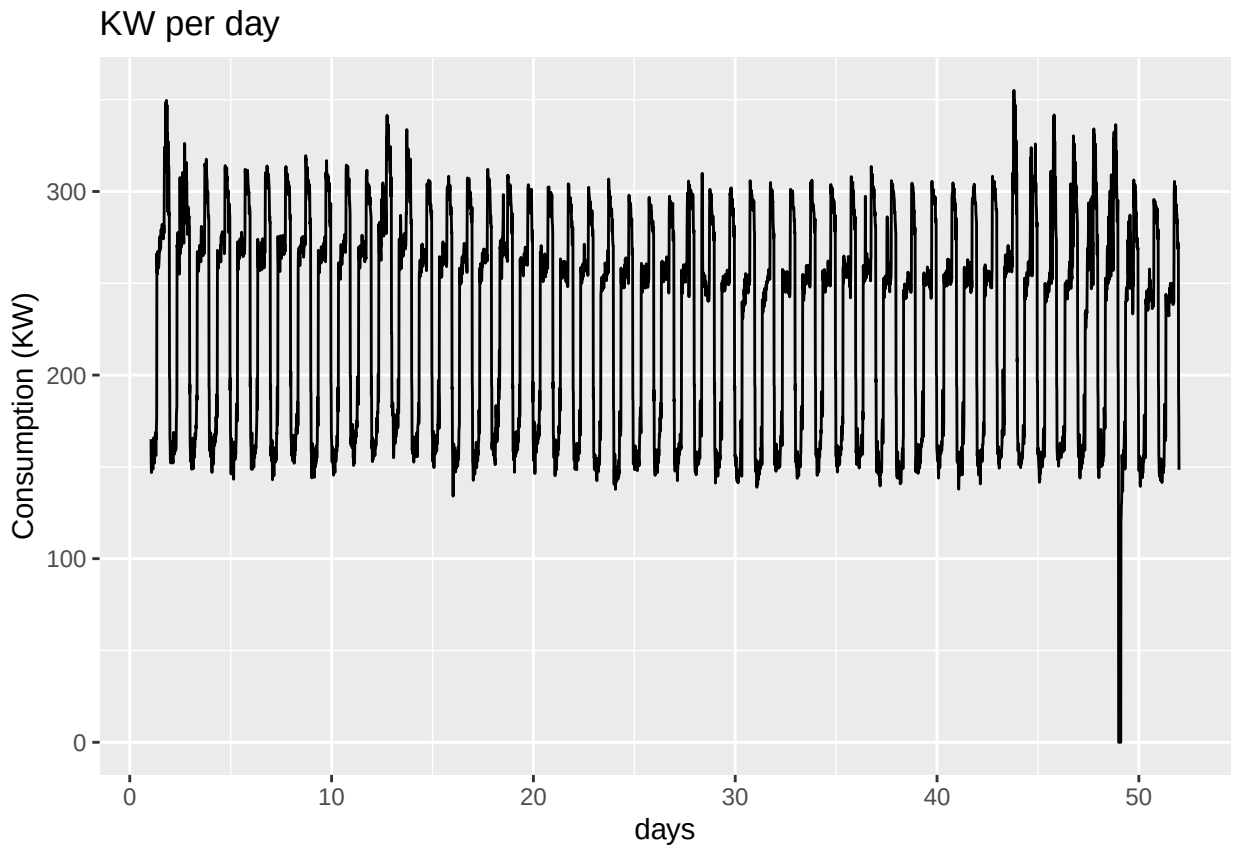
##creation of ts object AS we have observations every 15 mins, we will make it by day. the frequency is 60*24/15=96

```
elec_consum=ts(data[1:4891,2], start=c(1,6), end = c(51,96),frequency = 96,)
tail(elec_consum)
```

```
## Time Series:
## Start = c(51, 91)
## End = c(51, 96)
## Frequency = 96
##      Power (kW)
## [1,]      272.5
## [2,]      271.2
## [3,]      269.8
## [4,]      189.6
## [5,]      177.9
## [6,]      148.4
```

##ploting the data

```
autoplot(elec_consum)+
  ggtitle('KW per day')+
  xlab('days')+
  ylab('Consumption (KW)')
```



As we can see the plot above, it seems there are some outliers. We will try to handle it

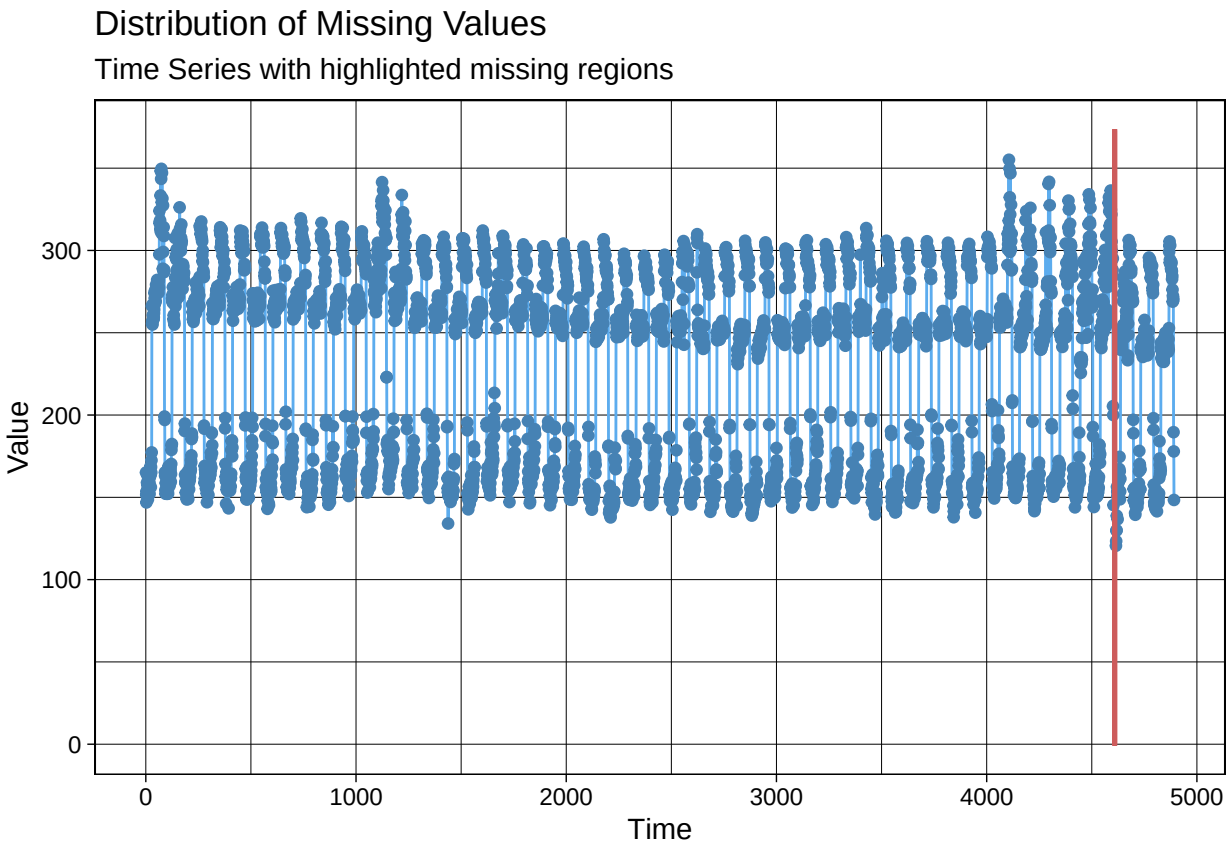
```
# Detect outliers in the time series
outliers = which(elec_consum==0)

# Print the detected outliers
print (outliers)
```

```
##  [1] 4604 4605 4606 4607 4608 4609 4610 4611 4612 4613 4614
```

```
# Replace zero values with NA
elec_consum_NA=elec_consum
elec_consum_NA[outliers] <- NA

library(imputeTS)
```

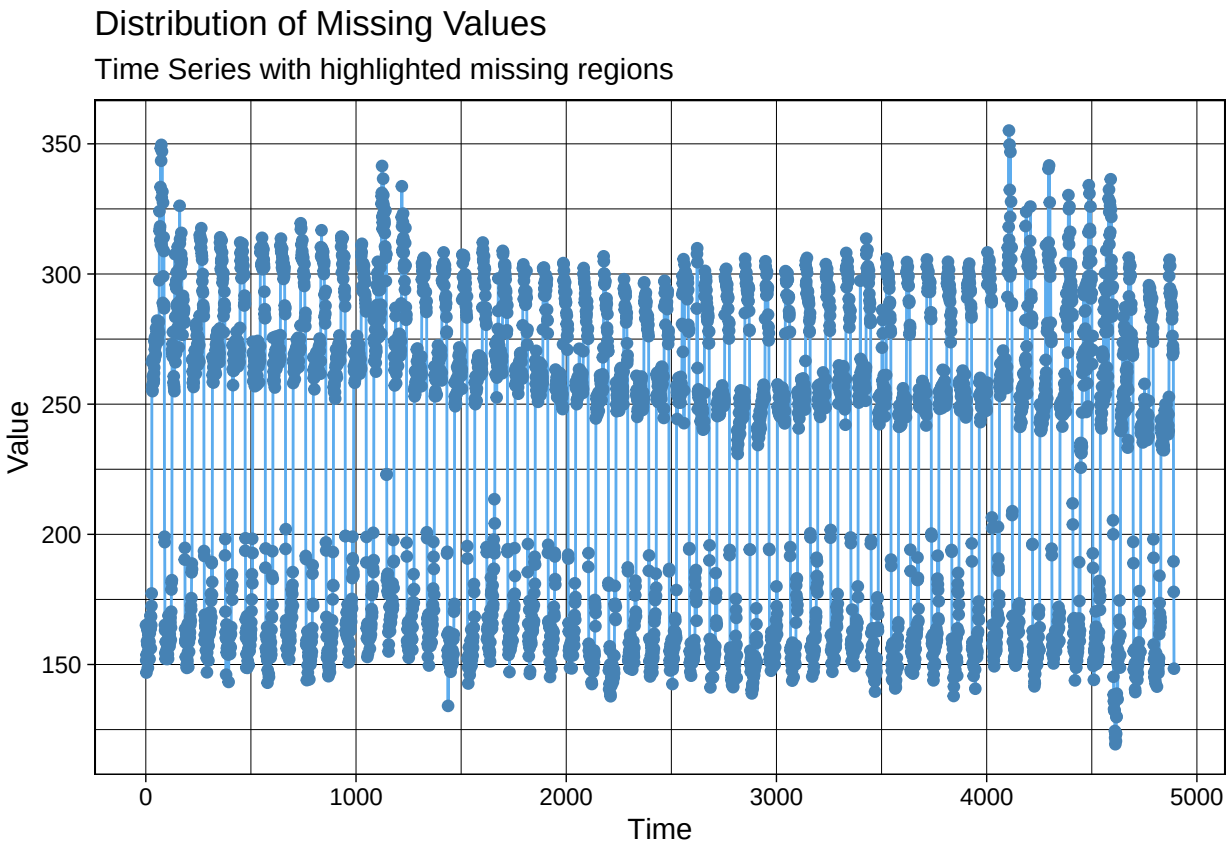
```
## Warning: package 'imputeTS' was built under R version 4.3.3
```

```
# Plot the time series with detected outliers
ggplot_na_distribution(elec_consum_NA)
```

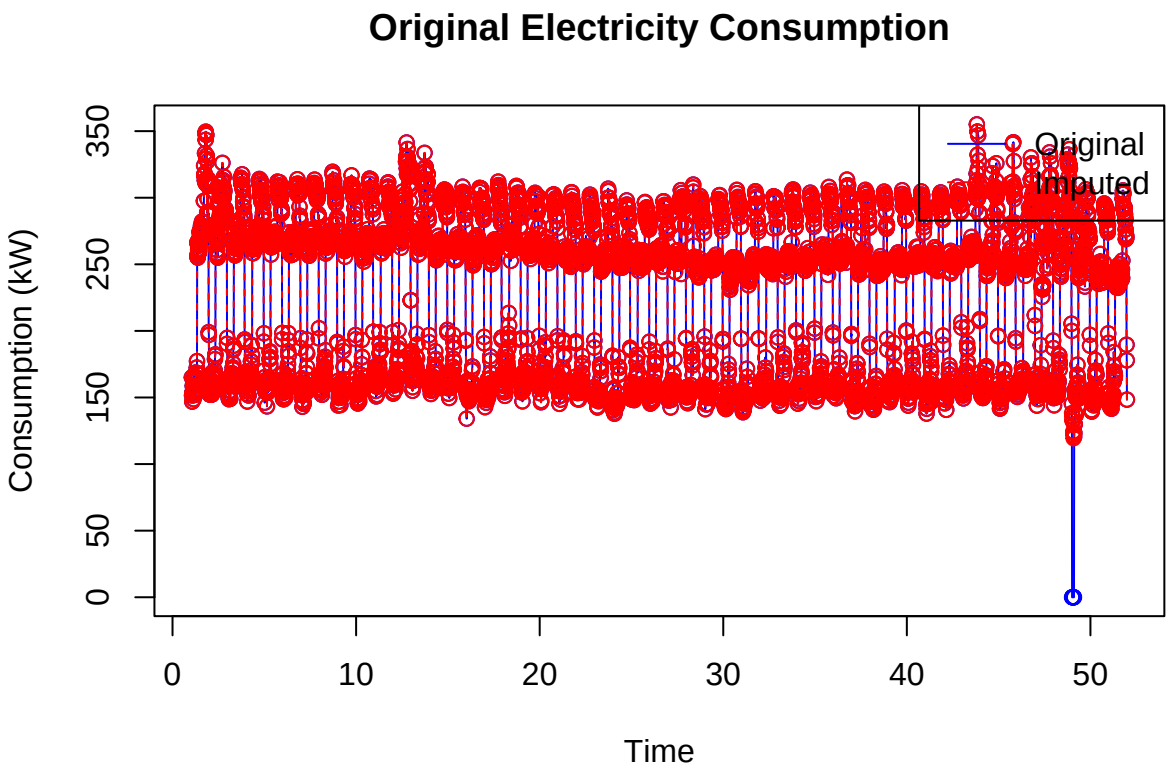


Impute missing values using interpolation

```
elec_consum_i<- na.interp(elec_consum_NA)

#plot after interpolation
ggplot_na_distribution(elec_consum_i)
```




```
# Plot the original and imputed time series
plot.ts(elec_consum, main="Original Electricity Consumption", ylab="Consumption (kW)", col="blue", type=
lines(elec_consum_i, col="red", type="o", lty=2)
legend("topright", legend=c("Original", "Imputed"), col=c("blue", "red"), lty=1:2)
```

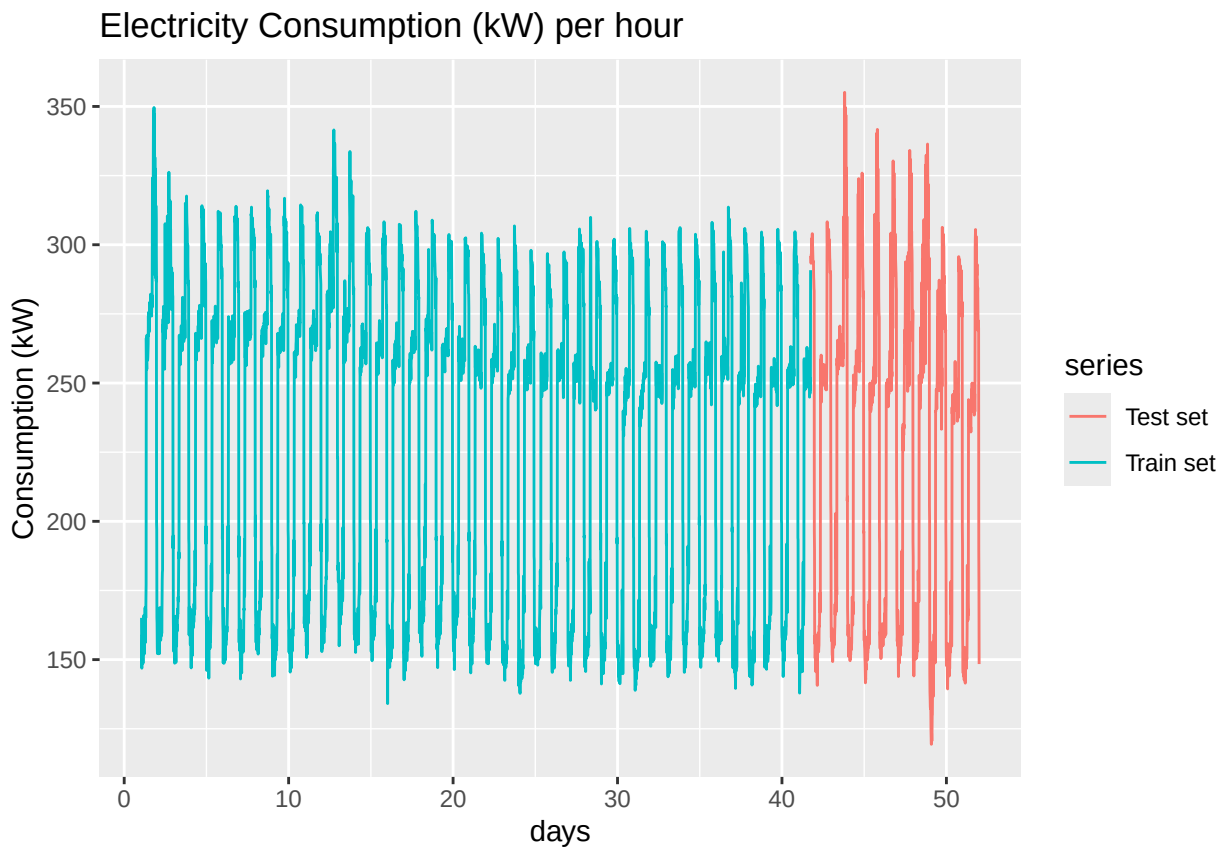



```
# Print the imputed time series
print(elec_consum_i[outliers])
```

```
##  [1] 138.4016 135.9332 132.9196 132.2571 135.4012 132.3401 124.4837 121.8882 119.3849 123.8997 123.1
```

Now we split our time series dataset in two sets: train 80% and test 20% Since I have 4891 observations, I will get: - for the train set : 4891 * 0.8 = 3907 observations (from 1 to 41) - for the test set : 4891 * 0.2 = 984 observations (from 42 to 51)

```
elec_consum_train= window(elec_consum_i, start=c(1,6), end=c(41,72))
elec_consum_test= window(elec_consum_i, start=c(41,73), end=c(51,96))
autoplot(elec_consum_train,series="Train set") +
  autolayer(elec_consum_test,series='Test set')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('days') +
  ylab('Consumption (kW)')
```



Time-series models

Once I've gained enough understanding of the time series data, I'm ready to create different models and check their accuracy on the predictions. In order to compare the results of the models, I will use the Mean Squared Error, which I implemented as follows:

```
rmse = function(y_pred, y_true) {
# Calculate ROOT Mean Squared Error (RMSE)
  #
  # Args:
  #   y_true: vector of true values
  #   y_pred: vector of predicted values
  #
  # Returns:
  #   rmse: RMSE value for the given predicted values



  # Calculate MSE
  rmse = sqrt(mean((y_pred - y_true )^2))
  return(rmse)
}
```

Forecasting with exponential smoothing

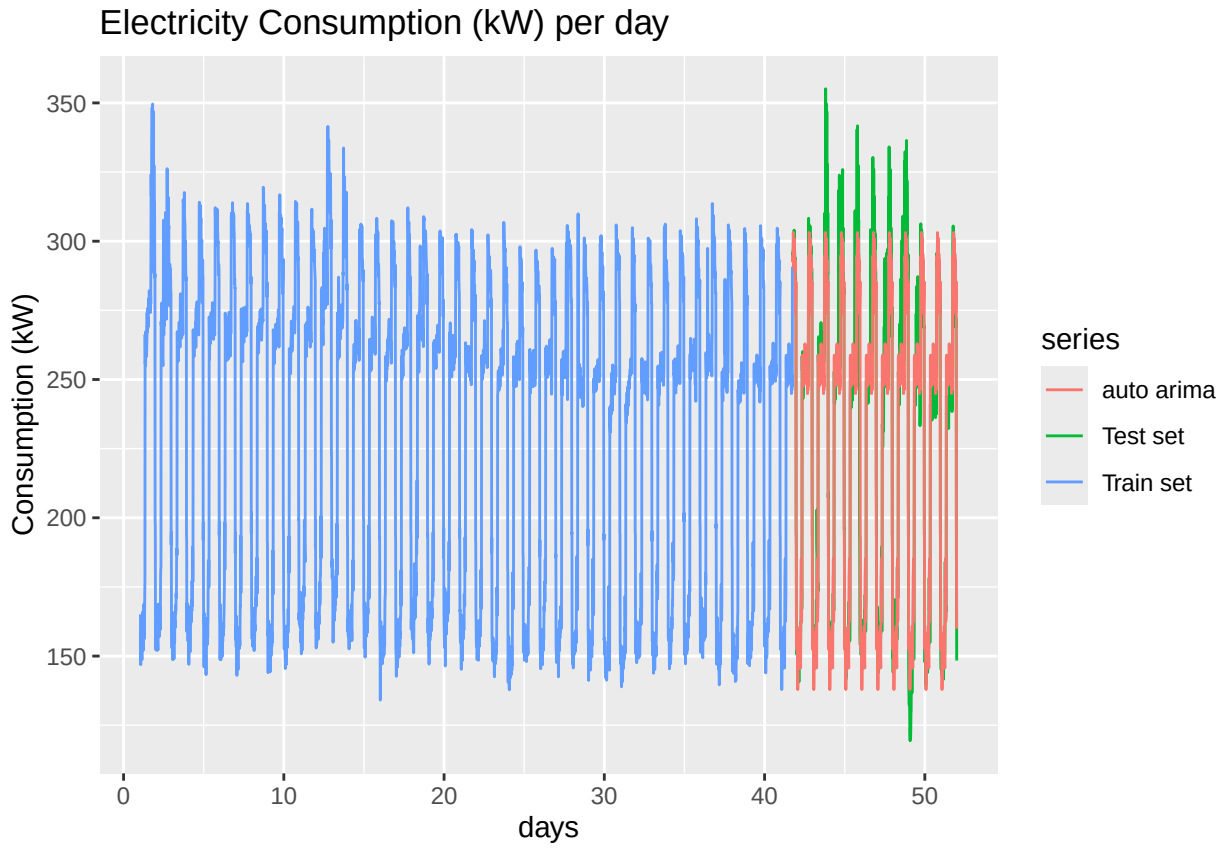
We see a seasonal pattern, probably additive. Because of the frequency limit of 24, we cannot fit a Holt-Winters additive model

** Forcasting with SARIMA models**

*** Automatically***

We choose automaticaly an SARIMA model

```
elec_consum_aut=auto.arima(elec_consum_train,lambda = "auto")
prev_aut=forecast(elec_consum_aut, h=984)


#plot in a graph
autoplot(elec_consum_train,series="Train set") +
  autolayer(elec_consum_test,series='Test set')+
  autolayer(prev_aut$mean,series='auto arima')+
  ggtitle ('Electricity Consumption (kW) per day') +
  xlab('days') +
  ylab('Consumption (kW)')
```



```
summary(elec_consum_aut)
```

```
## Series: elec_consum_train
## ARIMA(5,0,0)(0,1,0)[96]
## Box Cox transformation: lambda= -0.01102751
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5
##       0.7752  0.1390  -0.0687  -0.2499  0.1786
## s.e.  0.0159  0.0199   0.0200   0.0199  0.0159
##
## sigma^2 = 0.001688:  log likelihood = 6760.18
## AIC=-13508.37   AICc=-13508.35   BIC=-13470.89
##
## Training set error measures:
##                       ME     RMSE     MAE        MPE     MAPE     MASE       ACF1
## Training set -0.04130911 9.475386 5.79422 -0.1101576 2.652868 0.725453 0.03228393
```

Let's evaluate the SARIMA (5,0,0)(0,1,0)[96] model by computing the RMSE

```
sarima_model = rmse(prev_aut$mean,elec_consum_test)
sarima_model
```

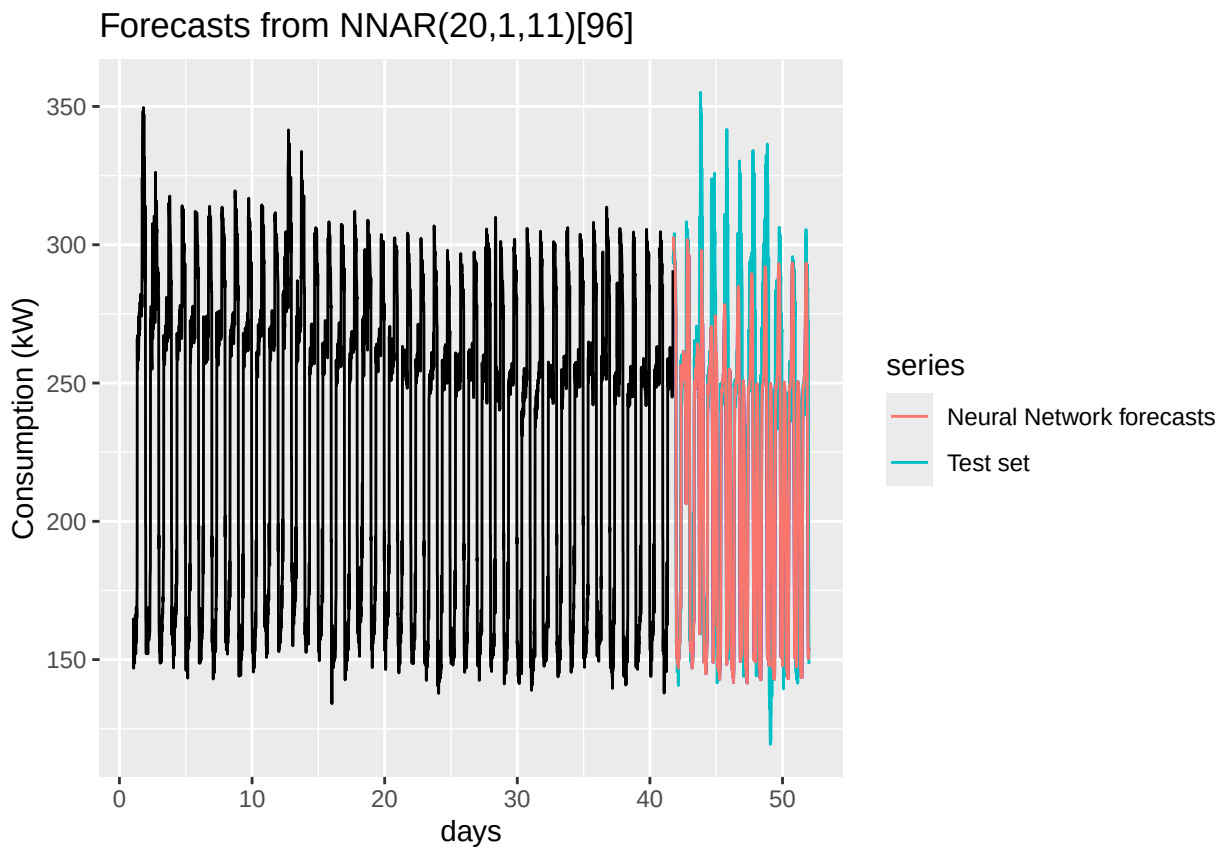
```
## [1] 15.7896
```

The prediction from this model doesn't look bad, but let check if we can improve it.

** Forcasting with Neural Network models**

We can try a auto-regressive neural network

```
elec_consum_NN =nnetar(elec_consum_train, lambda = 'auto')
prev_NN=forecast(elec_consum_NN, h=984)

#plot in a graph
autoplot(prev_NN) +
  autolayer(elec_consum_test,series='Test set')+
  autolayer(prev_NN$mean, series="Neural Network forecasts")+

  xlab('days') +
  ylab('Consumption (kW)')
```




```
summary(elec_consum_NN)
```

```
##           Length Class        Mode
## x         3907   ts           numeric
## m            1   -none-       numeric
## p            1   -none-       numeric
## P            1   -none-       numeric
## scalex       2   -none-       list
## size         1   -none-       numeric
## lambda       1   -none-       numeric
## subset    3907   -none-       numeric
## model       20   nnetarmodels list
## nnetargs     0   -none-       list
## fitted    3907   ts           numeric
## residuals 3907   ts           numeric
## lags        21   -none-       numeric
## series       1   -none-       character
## method       1   -none-       character
## call         3   -none-       call
```

Let's evaluate the Neural Network model by computing the RMSE

```
neural_network = rmse(prev_NN$mean,elec_consum_test)
neural_network
```

```
## [1] 64.90303
```

The score is not better that the previous model (SARIMA), we will try other machine learning models

#** Forcasting with Random forest**

We based our forecast on the 96 previous observations, but that can be optimized (by CV)

```
rfdata=as.vector(elec_consum_train)[1:97]
for (i in 1:(length(as.vector(elec_consum_train))-97)){
rfdata=rbind(rfdata,as.vector(elec_consum_train)[(i+1):(i+97)])
}
```

We fit the model

```
library(randomForest)
```

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

```
## 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
```

```
elec_consum_RF=randomForest(x=rfdata[,-97], y=rfdata[,97])
```

And then sequentially forecast the next 984 values

```
pred=rep(NULL,984)
newdata=tail(elec_consum_train,96)
for (t in 1:984){
pred[t]=predict(elec_consum_RF,newdata=newdata)
newdata=c(newdata[-1],pred[t])
}
prev_RF=ts(pred,start=c(41,73), end=c(51,96),frequency = 96)
```

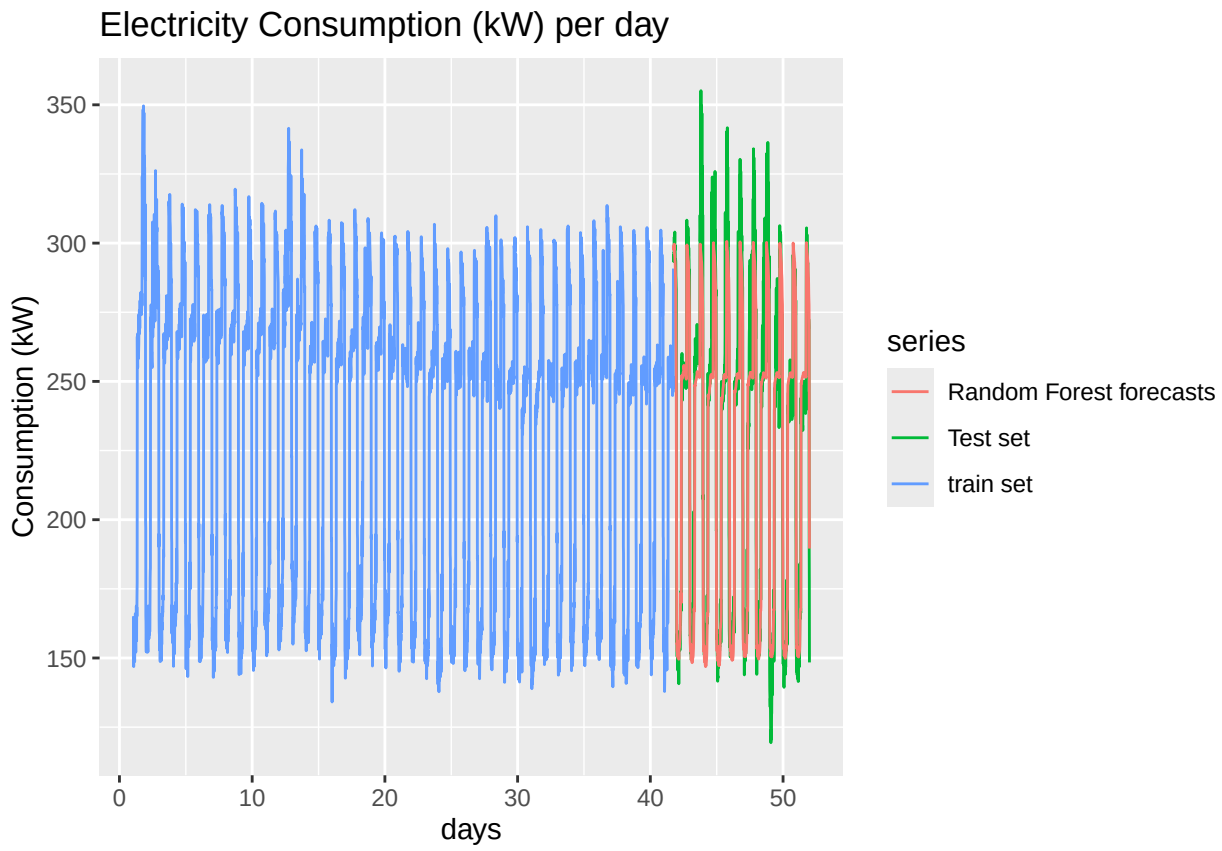
Let's evaluate the Random Forest model by computing the RMSE

```
Random_Forest = rmse(prev_RF,elec_consum_test)
Random_Forest
```

```
## [1] 15.95049
```

the score with random forest model is not bad, but not better than SARIMA.

```
#plot in a graph
autoplot(elec_consum_train, series = 'train set') +
  autolayer(elec_consum_test,series='Test set')+
  autolayer(prev_RF, series="Random Forest forecasts")+
  ggtitle ('Electricity Consumption (kW) per day') +
  xlab('days') +
  ylab('Consumption (kW)')
```



let's try if we get better result with XG BOOST

Forecasting with XG Boost##**

with default values, which should have to be optimized by Cross Validation.

```
install.packages("xgboost")
```

```
## Error in install.packages : Updating loaded packages
```

```
library("xgboost")
```

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

```
model = xgboost(rfdata[,1:96], label = rfdata[,97],
max_depth = 10, eta = .5, nrounds = 200,
nthread = 2, objective = "reg:squarederror")
```

```
## [1]  train-rmse:119.373787
## [2]  train-rmse:60.340326
## [3]  train-rmse:30.893533
## [4]  train-rmse:16.441181
## [5]  train-rmse:9.270089
## [6]  train-rmse:5.851077
## [7]  train-rmse:4.334800
## [8]  train-rmse:3.279405
## [9]  train-rmse:2.695397
## [10] train-rmse:2.367281
## [11] train-rmse:2.163320
## [12] train-rmse:2.055532
## [13] train-rmse:1.924477
## [14] train-rmse:1.799983
## [15] train-rmse:1.618693
## [16] train-rmse:1.530930
## [17] train-rmse:1.415046
## [18] train-rmse:1.291468
## [19] train-rmse:1.193719
## [20] train-rmse:1.148871
## [21] train-rmse:1.095383
## [22] train-rmse:1.017126
## [23] train-rmse:0.939747
## [24] train-rmse:0.880888
## [25] train-rmse:0.850620
## [26] train-rmse:0.789338
## [27] train-rmse:0.724872
## [28] train-rmse:0.690144
## [29] train-rmse:0.649995
## [30] train-rmse:0.610520
## [31] train-rmse:0.548173
## [32] train-rmse:0.528274
## [33] train-rmse:0.485393
## [34] train-rmse:0.472382
## [35] train-rmse:0.423052
## [36] train-rmse:0.403416
## [37] train-rmse:0.391616
## [38] train-rmse:0.357850
## [39] train-rmse:0.331136
## [40] train-rmse:0.307727
## [41] train-rmse:0.268497
## [42] train-rmse:0.256274
## [43] train-rmse:0.243106
## [44] train-rmse:0.227924
## [45] train-rmse:0.212543
## [46] train-rmse:0.184451
## [47] train-rmse:0.166164
## [48] train-rmse:0.155050
## [49] train-rmse:0.147728
## [50] train-rmse:0.130443
## [51] train-rmse:0.125307
## [52] train-rmse:0.118196
## [53] train-rmse:0.112226
## [54] train-rmse:0.107357
```

```
## [55] train-rmse:0.100260
## [56] train-rmse:0.093986
## [57] train-rmse:0.087674
## [58] train-rmse:0.084898
## [59] train-rmse:0.079789
## [60] train-rmse:0.074304
## [61] train-rmse:0.069721
## [62] train-rmse:0.064953
## [63] train-rmse:0.062778
## [64] train-rmse:0.060233
## [65] train-rmse:0.057441
## [66] train-rmse:0.052519
## [67] train-rmse:0.047070
## [68] train-rmse:0.045641
## [69] train-rmse:0.041757
## [70] train-rmse:0.039814
## [71] train-rmse:0.037017
## [72] train-rmse:0.034849
## [73] train-rmse:0.031224
## [74] train-rmse:0.028061
## [75] train-rmse:0.026979
## [76] train-rmse:0.025644
## [77] train-rmse:0.023460
## [78] train-rmse:0.022178
## [79] train-rmse:0.020865
## [80] train-rmse:0.019619
## [81] train-rmse:0.018670
## [82] train-rmse:0.017297
## [83] train-rmse:0.016325
## [84] train-rmse:0.015326
## [85] train-rmse:0.013827
## [86] train-rmse:0.011722
## [87] train-rmse:0.011502
## [88] train-rmse:0.010498
## [89] train-rmse:0.009987
## [90] train-rmse:0.009459
## [91] train-rmse:0.008919
## [92] train-rmse:0.008412
## [93] train-rmse:0.007970
## [94] train-rmse:0.007544
## [95] train-rmse:0.007063
## [96] train-rmse:0.006514
## [97] train-rmse:0.005968
## [98] train-rmse:0.005542
## [99] train-rmse:0.005195
## [100]    train-rmse:0.004886
## [101]    train-rmse:0.004400
## [102]    train-rmse:0.004086
## [103]    train-rmse:0.003777
## [104]    train-rmse:0.003670
## [105]    train-rmse:0.003566
## [106]    train-rmse:0.003319
## [107]    train-rmse:0.003172
## [108]    train-rmse:0.002827
```

```
## [109]    train-rmse:0.002669
## [110]    train-rmse:0.002385
## [111]    train-rmse:0.002229
## [112]    train-rmse:0.002142
## [113]    train-rmse:0.002076
## [114]    train-rmse:0.001884
## [115]    train-rmse:0.001804
## [116]    train-rmse:0.001713
## [117]    train-rmse:0.001624
## [118]    train-rmse:0.001569
## [119]    train-rmse:0.001417
## [120]    train-rmse:0.001337
## [121]    train-rmse:0.001310
## [122]    train-rmse:0.001258
## [123]    train-rmse:0.001198
## [124]    train-rmse:0.001085
## [125]    train-rmse:0.001038
## [126]    train-rmse:0.000990
## [127]    train-rmse:0.000894
## [128]    train-rmse:0.000840
## [129]    train-rmse:0.000743
## [130]    train-rmse:0.000698
## [131]    train-rmse:0.000678
## [132]    train-rmse:0.000650
## [133]    train-rmse:0.000619
## [134]    train-rmse:0.000576
## [135]    train-rmse:0.000550
## [136]    train-rmse:0.000511
## [137]    train-rmse:0.000511
## [138]    train-rmse:0.000511
## [139]    train-rmse:0.000511
## [140]    train-rmse:0.000511
## [141]    train-rmse:0.000511
## [142]    train-rmse:0.000511
## [143]    train-rmse:0.000511
## [144]    train-rmse:0.000511
## [145]    train-rmse:0.000511
## [146]    train-rmse:0.000511
## [147]    train-rmse:0.000511
## [148]    train-rmse:0.000511
## [149]    train-rmse:0.000511
## [150]    train-rmse:0.000511
## [151]    train-rmse:0.000511
## [152]    train-rmse:0.000511
## [153]    train-rmse:0.000511
## [154]    train-rmse:0.000511
## [155]    train-rmse:0.000511
## [156]    train-rmse:0.000511
## [157]    train-rmse:0.000511
## [158]    train-rmse:0.000511
## [159]    train-rmse:0.000511
## [160]    train-rmse:0.000511
## [161]    train-rmse:0.000511
## [162]    train-rmse:0.000511
```

```
## [163]    train-rmse:0.000511
## [164]    train-rmse:0.000511
## [165]    train-rmse:0.000511
## [166]    train-rmse:0.000511
## [167]    train-rmse:0.000511
## [168]    train-rmse:0.000511
## [169]    train-rmse:0.000511
## [170]    train-rmse:0.000511
## [171]    train-rmse:0.000511
## [172]    train-rmse:0.000511
## [173]    train-rmse:0.000511
## [174]    train-rmse:0.000511
## [175]    train-rmse:0.000511
## [176]    train-rmse:0.000511
## [177]    train-rmse:0.000511
## [178]    train-rmse:0.000511
## [179]    train-rmse:0.000511
## [180]    train-rmse:0.000511
## [181]    train-rmse:0.000511
## [182]    train-rmse:0.000511
## [183]    train-rmse:0.000511
## [184]    train-rmse:0.000511
## [185]    train-rmse:0.000511
## [186]    train-rmse:0.000511
## [187]    train-rmse:0.000511
## [188]    train-rmse:0.000511
## [189]    train-rmse:0.000511
## [190]    train-rmse:0.000511
## [191]    train-rmse:0.000511
## [192]    train-rmse:0.000511
## [193]    train-rmse:0.000511
## [194]    train-rmse:0.000511
## [195]    train-rmse:0.000511
## [196]    train-rmse:0.000511
## [197]    train-rmse:0.000511
## [198]    train-rmse:0.000511
## [199]    train-rmse:0.000511
## [200]    train-rmse:0.000511
```

And then sequentially forecast the next 984 values

```
pred_XGB=rep(NULL,984)
newdata=tail(elec_consum_train,96)
for (t in 1:984){
pred_XGB[t]=predict(model,matrix(newdata,1,96))
newdata=c(newdata[-1],pred[t])
}
prev_XGB=ts(pred_XGB,start=c(41,73), end=c(51,96),frequency = 96)
```

Let's evaluate the XG BOOST model by computing the RMSE

```
XG_boost = rmse(prev_XGB,elec_consum_test)
XG_boost
```

```
## [1] 16.79729
```

#First conclusion

let's resume all score we get and choose the best one.

```
cat('Forecasting results : ' , '\n')
```

```
## Forecasting results :
```

```
cat('RMSE with SARIMA model is:', sarima_model ,'\n')
```

```
## RMSE with SARIMA model is: 15.7896
```

```
cat('RMSE with Neural Network model is:', neural_network ,'\n')
```

```
## RMSE with Neural Network model is: 64.90303
```

```
cat('RMSE with Random Forest is:', Random_Forest ,'\n')
```

```
## RMSE with Random Forest is: 15.95049
```

```
cat('RMSE with XGB BOOST model is:', XG_boost ,'\n')
```

```
## RMSE with XGB BOOST model is: 16.79729
```

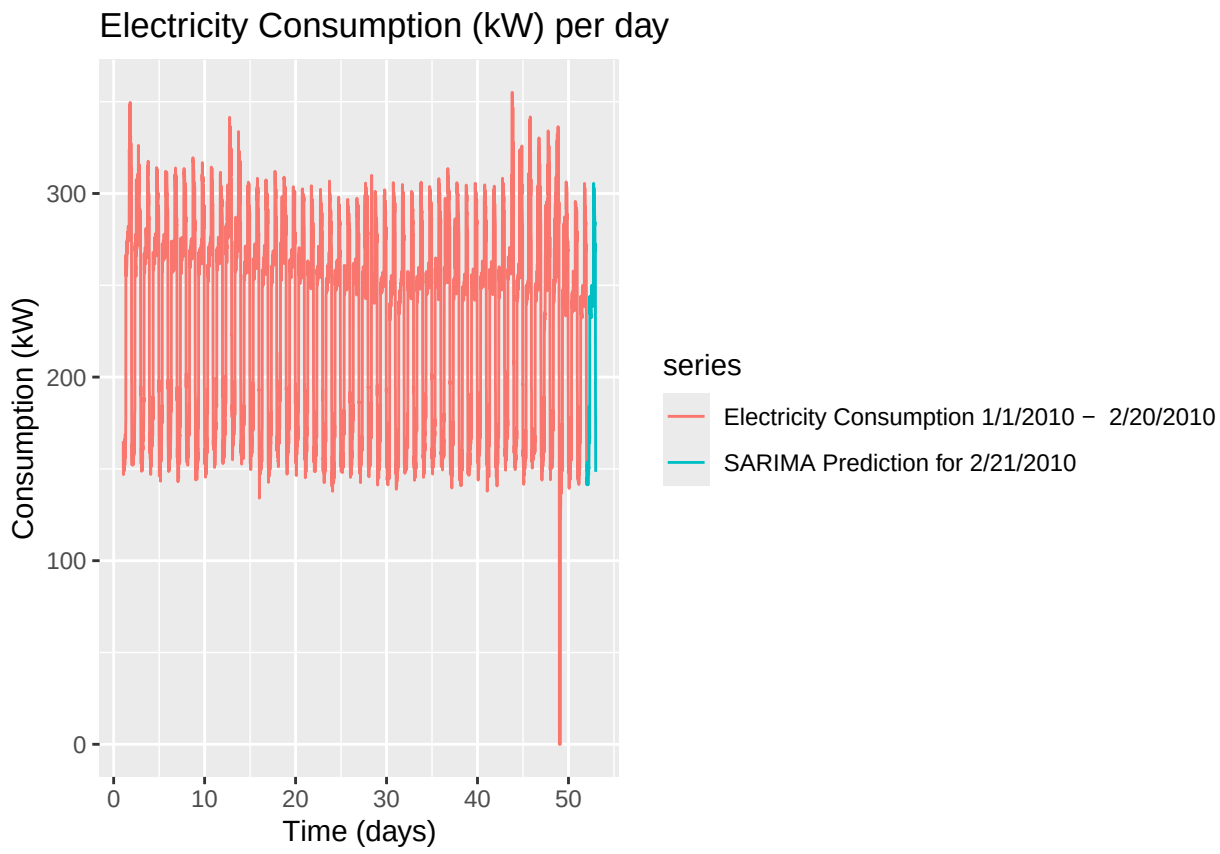
SARIMA model (5,0,0)(0,1,0)[96] is definitely our best model so far.

let's clearly see the prediction

```
autoplot(elec_consum_test,series='Test set') +
  autolayer(prev_aut$mean,series='SARIMA')+
  ggtitle ('Electricity Consumption (kW) per day') +
  xlab('Time (days)') +
  ylab('Consumption (kW)')
```

We will now forecast the electricity consumption (kW) for 2/21/2010 based on the whole previous consumption information (ARIMA (5,0,0)(0,1,0)[96]).
The prediction interval = 24 hr for the entire day of 2/21/2010.So h =(24*60)/15 = 96 observations

```
elec_consum_21 = Arima(elec_consum_i, order=c(5,0,0), seasonal=c(0,1,0),lambda = "auto")
prev_consum_21 = forecast(elec_consum_21, h = 96)
autoplot(elec_consum,series="Electricity Consumption 1/1/2010 -  2/20/2010") +
  autolayer(prev_consum_21$mean,series='SARIMA Prediction for 2/21/2010')+
  ggtitle ('Electricity Consumption (kW) per day') +
  xlab('Time (days)') +
  ylab('Consumption (kW)')
```



#Part 2: Forecast electricity consumption (kW) for 2/21/2010 by using outdoor temperature

let's create the time series object

```
df2 = read_excel('2023-11-Elec-train.xlsx')
```

```
elec_consum_t=ts(df2[1:4891,2:3], start=c(1,6), end = c(51,96),frequency = 96,)
tail(elec_consum_t)
```

```
## Time Series:
## Start = c(51, 91)
## End = c(51, 96)
## Frequency = 96
##          Power (kW) Temp (C°)
## 51.93750      272.5  11.66667
## 51.94792      271.2  11.66667
## 51.95833      269.8  11.66667
## 51.96875      189.6  11.66667
## 51.97917      177.9  11.66667
## 51.98958      148.4  11.66667
```

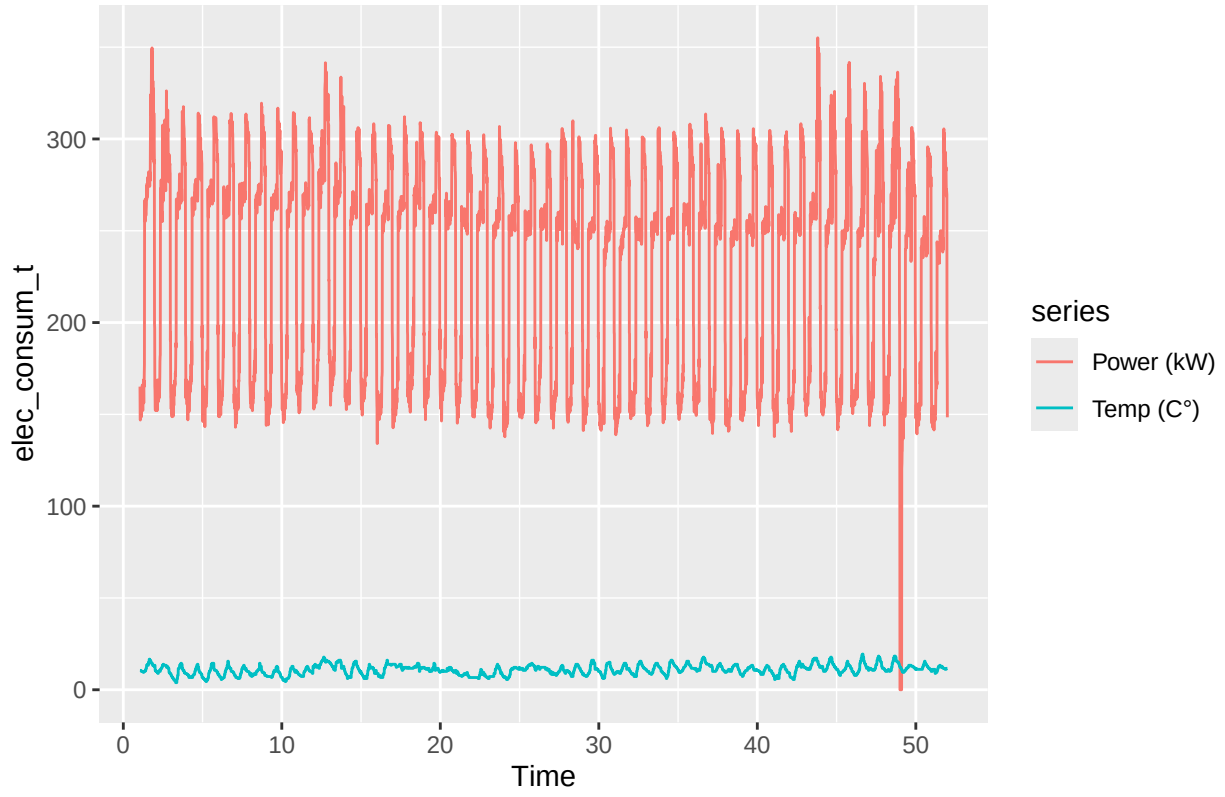
#ploting the ts

```
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.3.3
```

```
## -- Attaching packages ---------------------------------------------------------------------------------

## v fma       2.5     v expsmooth 2.3

## Warning: package 'fma' was built under R version 4.3.3

## Warning: package 'expsmooth' was built under R version 4.3.3

## -- Conflicts ------------------------------------------------------------------------------------------
## x randomForest::margin() masks ggplot2::margin()
```

```
autoplot(elec_consum_t)
```



#HANDLE OUTLIERS

```
outlierst=which(elec_consum_t==0)
elec_consum_t[outlierst]= elec_consum_i[outliers]
elec_consum_t[4604:4614,]
```

```
##       Power (kW) Temp (C°)
##  [1,]   138.4016  12.77778
##  [2,]   135.9332  12.22222
##  [3,]   132.9196  12.22222
##  [4,]   132.2571  12.22222
```

```
##  [5,]   135.4012  12.22222
##  [6,]   132.3401  11.66667
##  [7,]   124.4837  11.66667
##  [8,]   121.8882  11.66667
##  [9,]   119.3849  11.66667
## [10,]   123.8997  11.11111
## [11,]   123.1607  11.11111
```

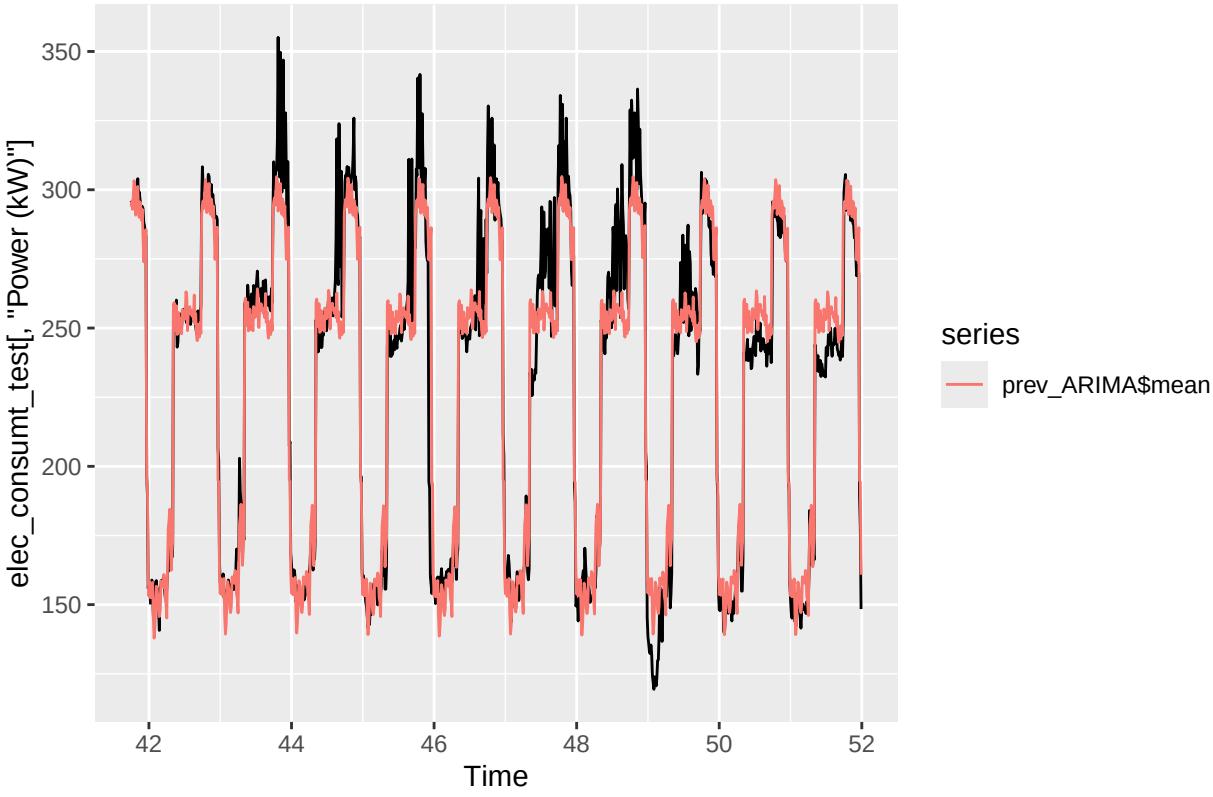
We split into train and test

```
elec_consumt_train= window(elec_consum_t, start=c(1,6), end=c(41,72))
elec_consumt_test= window(elec_consum_t, start=c(41,73), end=c(51,96))
```

#Forecasting with ARIMA model

We will use a dynamic regression model for forecasting electricity consumption, using temperature as external covariates. The order of the ARIMA model for the residual part is automatically selected

```
ARIMA=auto.arima(elec_consumt_train[,"Power (kW)"],xreg=elec_consumt_train[,2])
prev_ARIMA=forecast(ARIMA,h=984,xreg=elec_consumt_test[,2])
autoplot(elec_consumt_test[,"Power (kW)"])+autolayer(prev_ARIMA$mean)
```



let's evaluate this model

```r
arima_score = rmse(prev_ARIMA$mean,elec_consumt_test[,"Power (kW)"])
arima_score
```

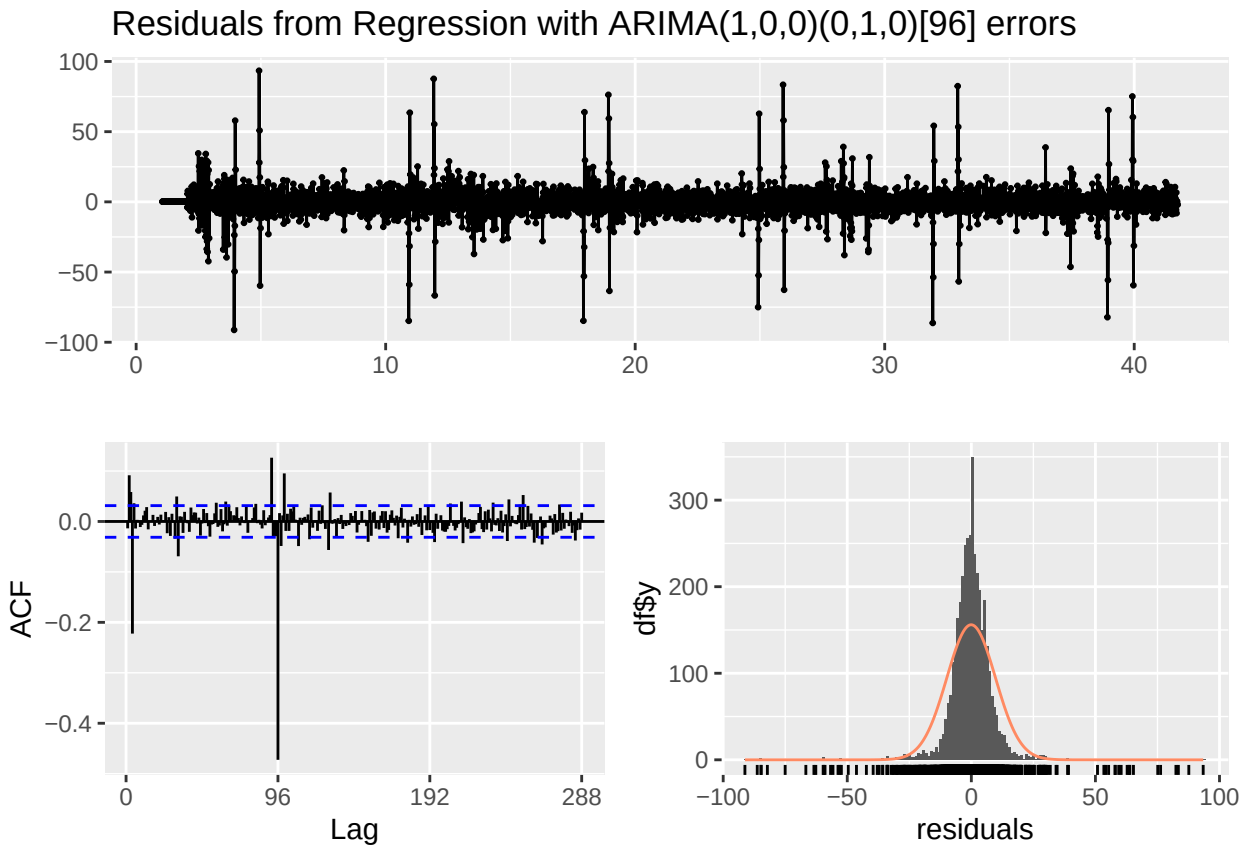
```
## [1] 15.61163
```

we can see that using covariates allows us to improve the forecasting. But if we check the residual, there is still some autocorrelations:

```r
summary(ARIMA)
```

```
## Series: elec_consumt_train[, "Power (kW)"]
## Regression with ARIMA(1,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1    xreg
##       0.7734  0.3283
## s.e.  0.0103  0.2396
##
## sigma^2 = 98.76:  log likelihood = -14158.36
## AIC=28322.72   AICc=28322.73   BIC=28341.46
##
## Training set error measures:
##                       ME     RMSE     MAE        MPE     MAPE      MASE         ACF1
## Training set -0.07249998 9.812342 5.73861 -0.1211996 2.648591 0.7184904 -0.01365387
```

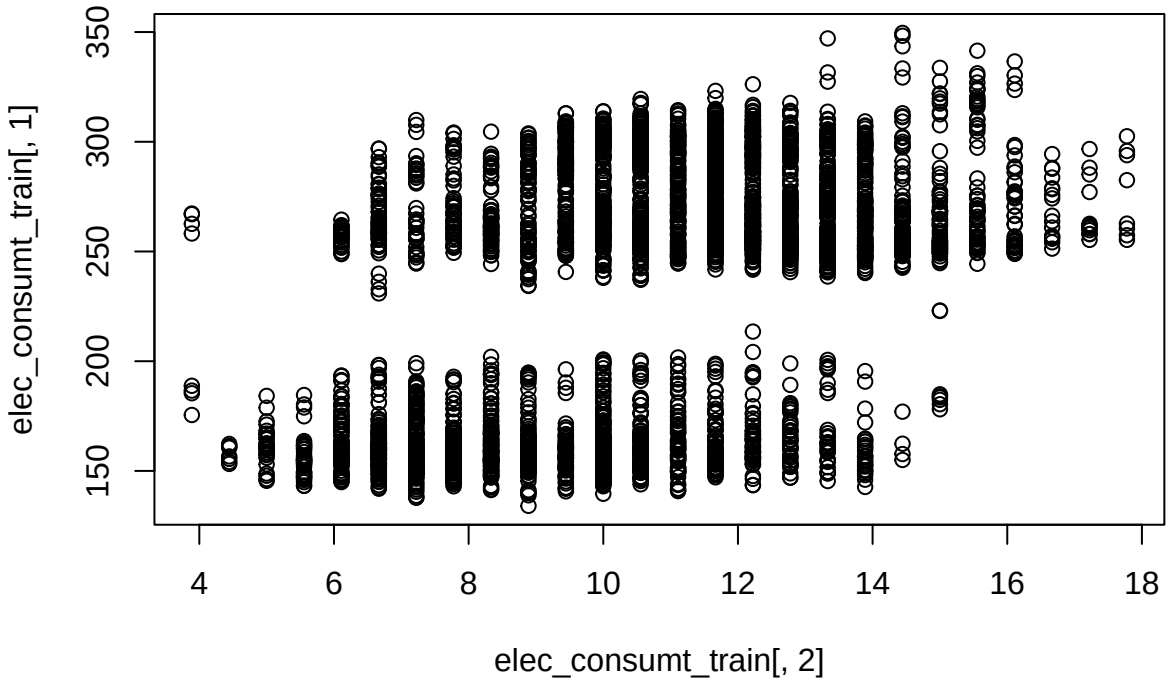
```r
checkresiduals(ARIMA)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0)(0,1,0)[96] errors
## Q* = 1523.5, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

We can try to find a better model manually. Let's have a look to the relationship between consumption and Temperature

```
plot(elec_consumt_train[,2],elec_consumt_train[,1])
```



There seems to be a visible pattern where the electricity consumption varies with temperature. As temperature increases, there is a notable shift in the distribution of power consumption. there are two distinct bands of power consumption values, one centered around 200 kW and the other around 300 kW.

it might correspond to different times of the day—one band representing power consumption during peak hours (e.g., late afternoon to evening) and the other representing off-peak hours. During peak hours, power consumption might be higher due to increased activity (e.g., cooking, cooling, lighting) even if the temperature is the same.

- First incorporate time-dependent patterns and seasonality in our analysis.
- Second explore clustering techniques to identify different operational states or patterns in the data.

incorporate time-dependent patterns and seasonality in our analysis

using outdoor temperature

```
effect_temperature=tslm(elec_consumt_train[,1]~elec_consumt_train[,2]+trend+season,data=elec_consumt_tra
summary(effect_temperature)
```

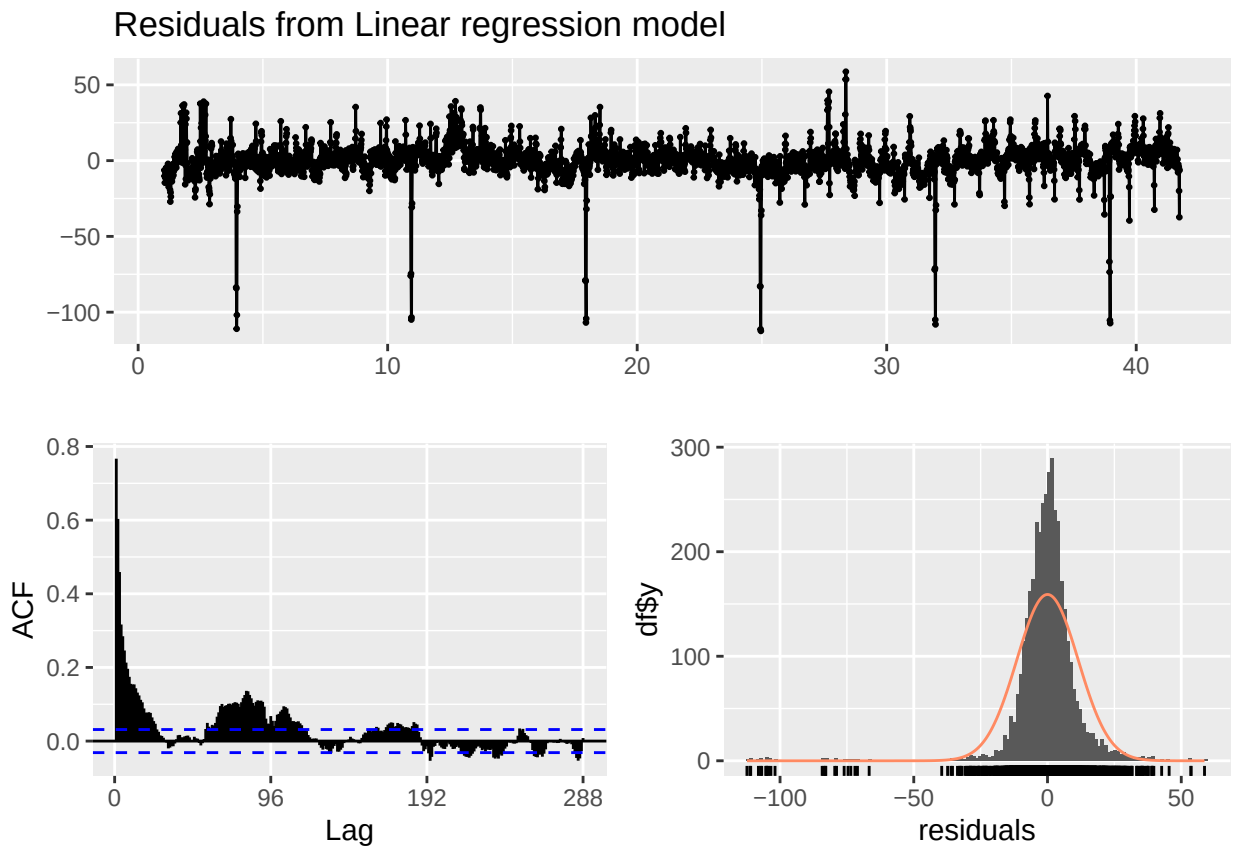
```
##
## Call:
## tslm(formula = elec_consumt_train[, 1] ~ elec_consumt_train[,
##     2] + trend + season, data = elec_consumt_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -112.370   -4.606    0.176    4.654   58.696
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.567e+02  2.070e+00  75.699  < 2e-16 ***
## elec_consumt_train[, 2]  1.096e+00  9.629e-02  11.383  < 2e-16 ***
## trend                   -4.732e-03  1.679e-04 -28.186  < 2e-16 ***
## season2                 -5.182e-01  2.600e+00  -0.199  0.84202
## season3                 -6.678e+00  2.600e+00  -2.569  0.01024 *
## season4                 -3.412e-01  2.600e+00  -0.131  0.89559
## season5                  2.859e+00  2.600e+00   1.100  0.27159
## season6                  3.154e+00  2.584e+00   1.220  0.22239
## season7                 -6.529e+00  2.584e+00  -2.527  0.01155 *
## season8                 -6.766e+00  2.584e+00  -2.618  0.00887 **
## season9                 -3.322e+00  2.584e+00  -1.286  0.19863
## season10                -4.037e+00  2.585e+00  -1.562  0.11835
## season11                -1.196e+00  2.585e+00  -0.463  0.64359
## season12                -2.006e+00  2.585e+00  -0.776  0.43774
## season13                -4.108e-01  2.585e+00  -0.159  0.87370
## season14                -5.115e+00  2.585e+00  -1.978  0.04796 *
## season15                -6.556e+00  2.585e+00  -2.536  0.01125 *
## season16                -9.637e-01  2.585e+00  -0.373  0.70934
## season17                 6.874e-01  2.585e+00   0.266  0.79034
## season18                -2.546e-01  2.586e+00  -0.098  0.92160
## season19                -1.167e+00  2.586e+00  -0.451  0.65189
## season20                 5.159e-01  2.586e+00   0.199  0.84192
## season21                 1.499e-01  2.586e+00   0.058  0.95380
## season22                 1.079e+00  2.587e+00   0.417  0.67667
## season23                 1.898e+00  2.587e+00   0.734  0.46315
## season24                 4.074e+00  2.587e+00   1.575  0.11543
## season25                 3.508e+00  2.587e+00   1.356  0.17523
## season26                 6.991e+00  2.587e+00   2.702  0.00692 **
## season27                 1.400e+01  2.587e+00   5.412 6.61e-08 ***
## season28                 1.605e+01  2.587e+00   6.205 6.06e-10 ***
## season29                 1.632e+01  2.587e+00   6.310 3.12e-10 ***
## season30                 2.192e+01  2.587e+00   8.471  < 2e-16 ***
## season31                 2.134e+01  2.587e+00   8.249  < 2e-16 ***
## season32                 1.623e+01  2.587e+00   6.273 3.95e-10 ***
## season33                 1.991e+01  2.587e+00   7.694 1.80e-14 ***
## season34                 1.040e+02  2.587e+00  40.209  < 2e-16 ***
## season35                 1.018e+02  2.587e+00  39.369  < 2e-16 ***
## season36                 9.899e+01  2.587e+00  38.265  < 2e-16 ***
```

```
## season37                9.866e+01  2.587e+00  38.136  < 2e-16 ***
## season38                1.015e+02  2.584e+00  39.276  < 2e-16 ***
## season39                9.641e+01  2.584e+00  37.319  < 2e-16 ***
## season40                9.882e+01  2.584e+00  38.251  < 2e-16 ***
## season41                9.966e+01  2.584e+00  38.576  < 2e-16 ***
## season42                9.583e+01  2.585e+00  37.077  < 2e-16 ***
## season43                9.664e+01  2.585e+00  37.392  < 2e-16 ***
## season44                9.826e+01  2.585e+00  38.018  < 2e-16 ***
## season45                9.849e+01  2.585e+00  38.105  < 2e-16 ***
## season46                1.006e+02  2.589e+00  38.865  < 2e-16 ***
## season47                9.868e+01  2.589e+00  38.111  < 2e-16 ***
## season48                9.929e+01  2.589e+00  38.348  < 2e-16 ***
## season49                1.001e+02  2.589e+00  38.655  < 2e-16 ***
## season50                9.956e+01  2.596e+00  38.346  < 2e-16 ***
## season51                1.030e+02  2.596e+00  39.666  < 2e-16 ***
## season52                1.021e+02  2.596e+00  39.333  < 2e-16 ***
## season53                1.004e+02  2.596e+00  38.678  < 2e-16 ***
## season54                1.021e+02  2.602e+00  39.239  < 2e-16 ***
## season55                1.024e+02  2.602e+00  39.365  < 2e-16 ***
## season56                1.017e+02  2.602e+00  39.084  < 2e-16 ***
## season57                1.011e+02  2.602e+00  38.844  < 2e-16 ***
## season58                1.012e+02  2.607e+00  38.808  < 2e-16 ***
## season59                1.014e+02  2.607e+00  38.913  < 2e-16 ***
## season60                1.011e+02  2.607e+00  38.784  < 2e-16 ***
## season61                1.009e+02  2.607e+00  38.689  < 2e-16 ***
## season62                1.008e+02  2.608e+00  38.660  < 2e-16 ***
## season63                1.002e+02  2.608e+00  38.422  < 2e-16 ***
## season64                1.003e+02  2.608e+00  38.464  < 2e-16 ***
## season65                1.001e+02  2.608e+00  38.367  < 2e-16 ***
## season66                1.004e+02  2.603e+00  38.588  < 2e-16 ***
## season67                1.010e+02  2.603e+00  38.811  < 2e-16 ***
## season68                9.843e+01  2.603e+00  37.818  < 2e-16 ***
## season69                9.842e+01  2.603e+00  37.813  < 2e-16 ***
## season70                1.193e+02  2.595e+00  45.992  < 2e-16 ***
## season71                1.328e+02  2.595e+00  51.185  < 2e-16 ***
## season72                1.448e+02  2.595e+00  55.816  < 2e-16 ***
## season73                1.443e+02  2.611e+00  55.290  < 2e-16 ***
## season74                1.420e+02  2.604e+00  54.521  < 2e-16 ***
## season75                1.406e+02  2.604e+00  54.004  < 2e-16 ***
## season76                1.398e+02  2.604e+00  53.666  < 2e-16 ***
## season77                1.402e+02  2.604e+00  53.818  < 2e-16 ***
## season78                1.459e+02  2.602e+00  56.063  < 2e-16 ***
## season79                1.419e+02  2.602e+00  54.523  < 2e-16 ***
## season80                1.405e+02  2.602e+00  54.007  < 2e-16 ***
## season81                1.390e+02  2.602e+00  53.416  < 2e-16 ***
## season82                1.391e+02  2.601e+00  53.465  < 2e-16 ***
## season83                1.368e+02  2.601e+00  52.575  < 2e-16 ***
## season84                1.366e+02  2.601e+00  52.527  < 2e-16 ***
## season85                1.356e+02  2.601e+00  52.128  < 2e-16 ***
## season86                1.336e+02  2.600e+00  51.394  < 2e-16 ***
## season87                1.325e+02  2.600e+00  50.978  < 2e-16 ***
## season88                1.314e+02  2.600e+00  50.536  < 2e-16 ***
## season89                1.292e+02  2.600e+00  49.684  < 2e-16 ***
## season90                1.122e+02  2.599e+00  43.145  < 2e-16 ***
```

```
## season91                  1.108e+02  2.599e+00  42.631  < 2e-16 ***
## season92                  1.047e+02  2.599e+00  40.291  < 2e-16 ***
## season93                  1.058e+02  2.599e+00  40.720  < 2e-16 ***
## season94                  2.994e+01  2.599e+00  11.519  < 2e-16 ***
## season95                  3.209e+01  2.599e+00  12.344  < 2e-16 ***
## season96                  2.910e+00  2.599e+00   1.120  0.26294
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.62 on 3809 degrees of freedom
## Multiple R-squared:  0.9598, Adjusted R-squared:  0.9588
## F-statistic: 938.5 on 97 and 3809 DF,  p-value: < 2.2e-16
```

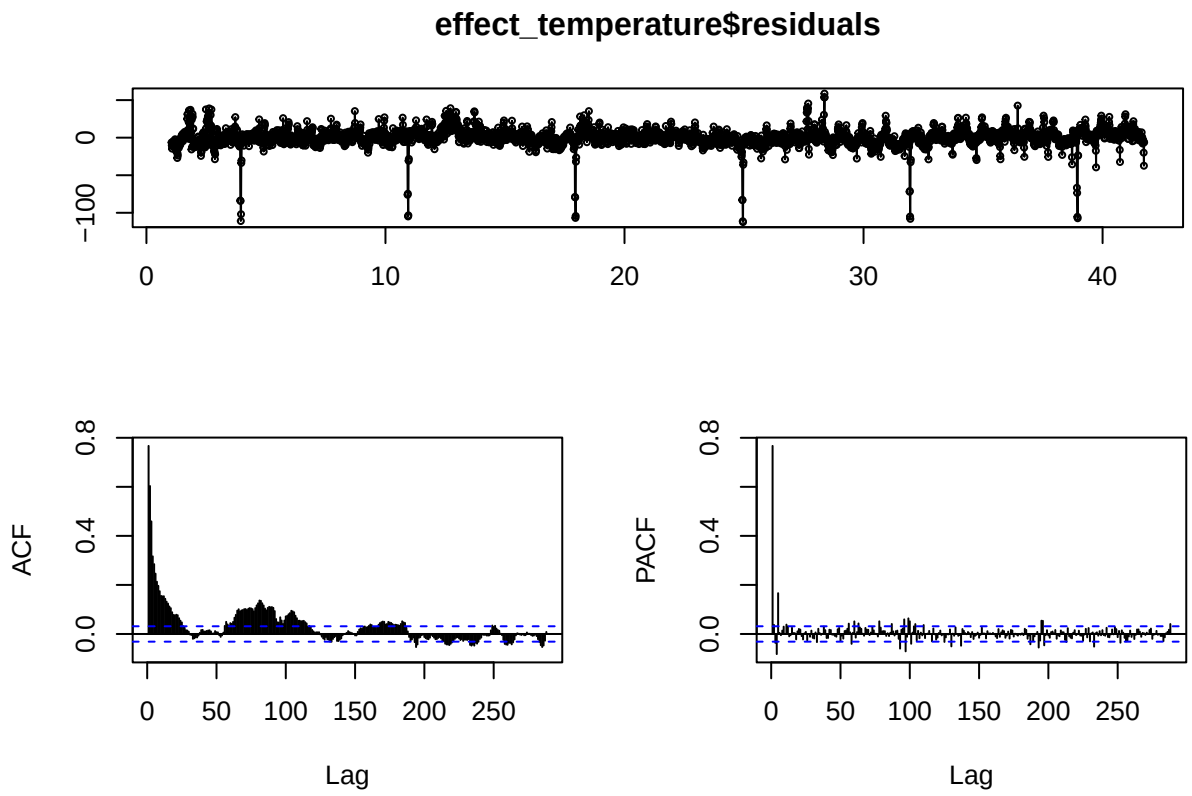
All the feature seems significant. Let's now have a look to the residual

```
checkresiduals(effect_temperature)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 2478.6, df = 192, p-value < 2.2e-16
```

```
tsdisplay(effect_temperature$residuals)
```

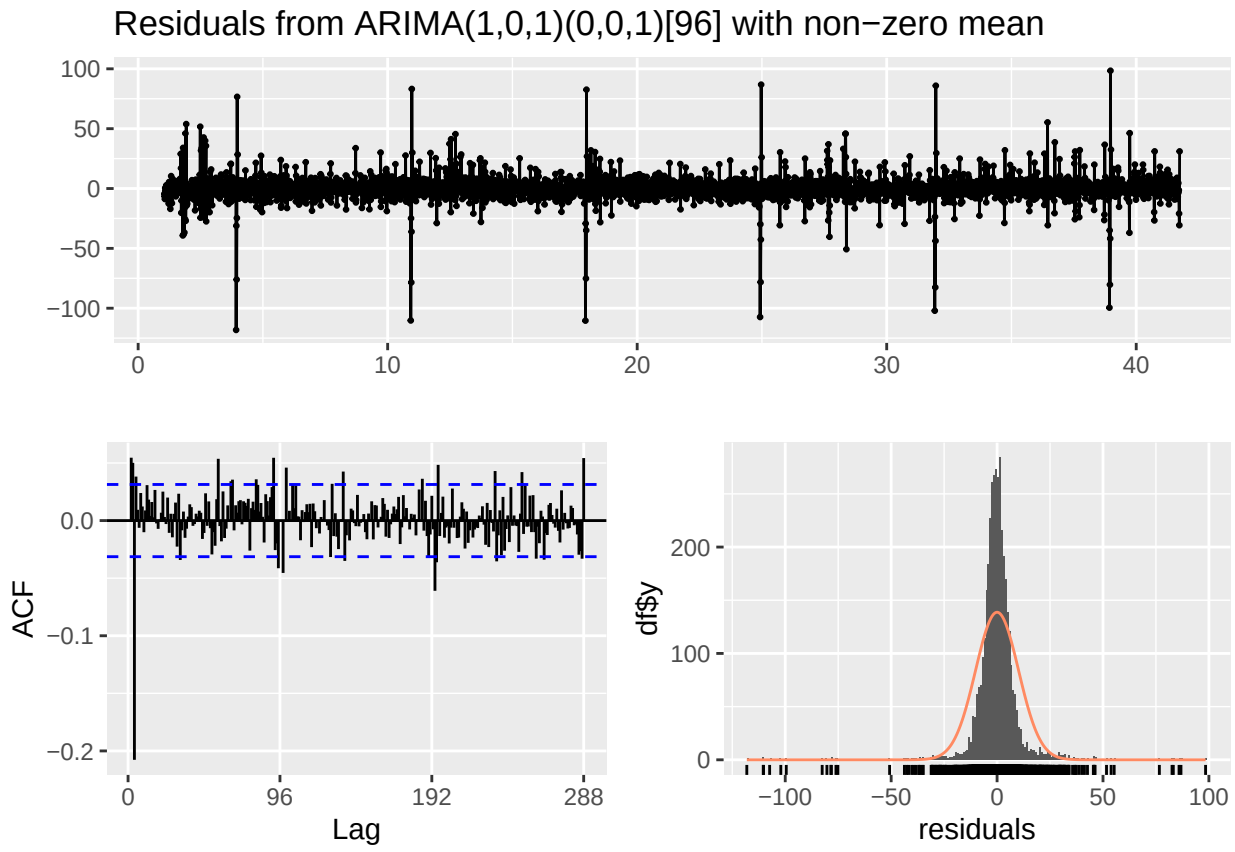


let's try an auto.arima an improve it manually

```
tmp=effect_temperature$residuals
effect_temperature2=auto.arima(tmp,lambda = 'auto')
```

```
## Warning in guerrero(x, lower, upper): Guerrero's method for selecting a Box-Cox parameter (lambda) is
## strictly positive data.
```

```
checkresiduals(effect_temperature2)
```

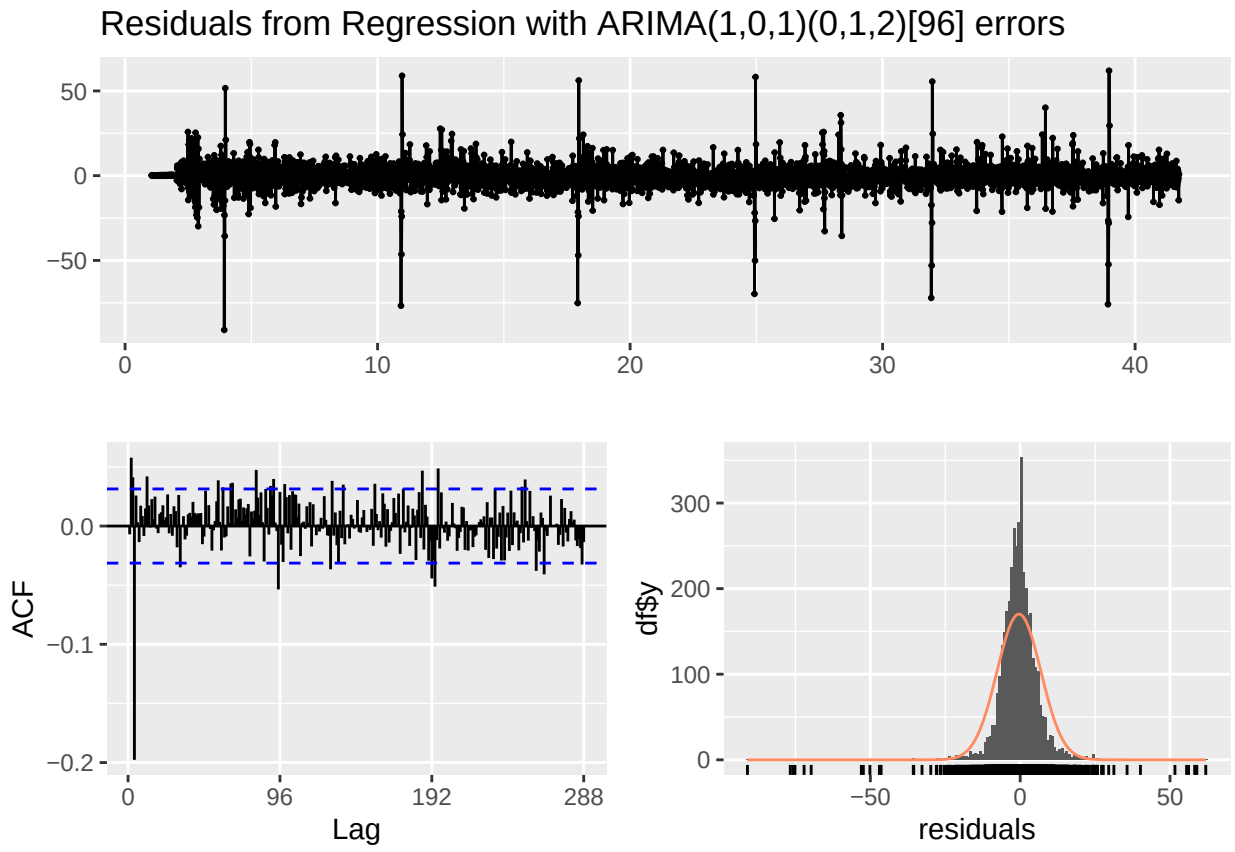



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(0,0,1)[96] with non-zero mean
## Q* = 422.85, df = 189, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 192
```

There is still a seasonal pattern. to improve our model we can try to add seasonal order to make the model more complex but I don't have enough computation power

We fit the model

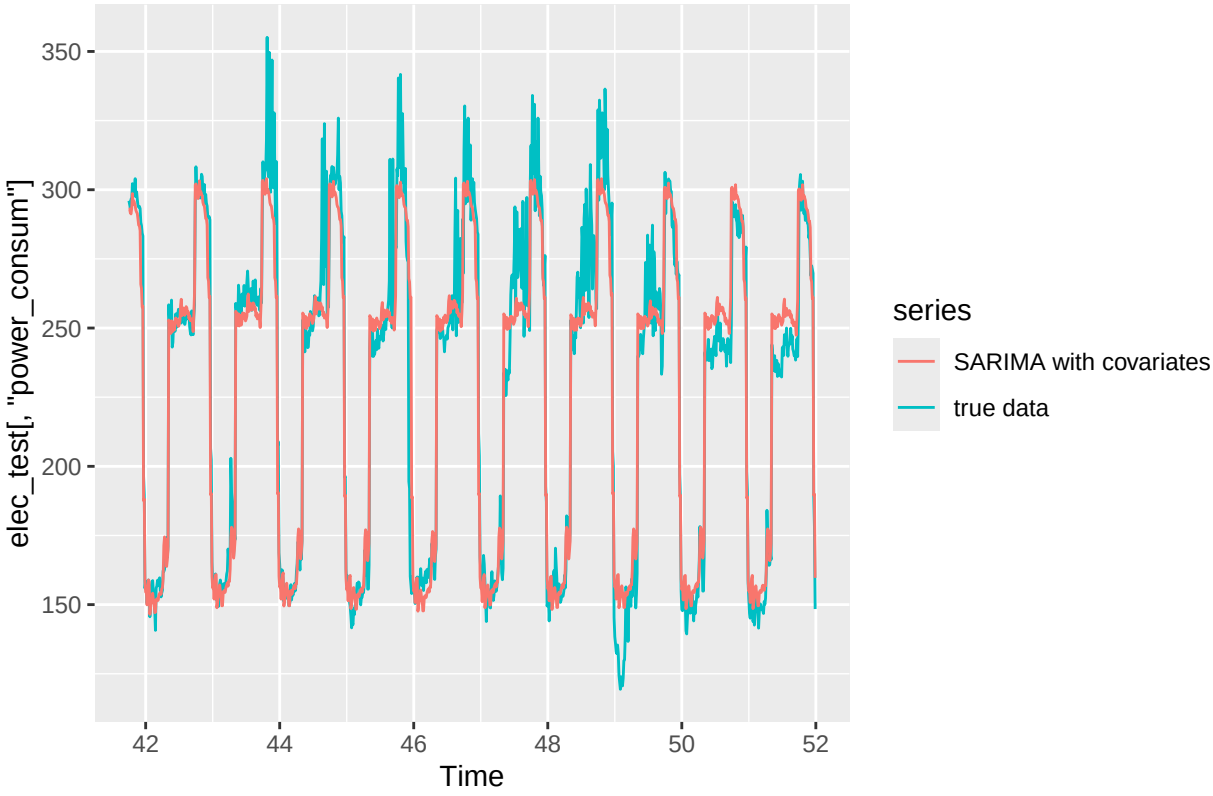
```
fit_eff=Arima(elec_consumt_train[,1],xreg=elec_consumt_train[,2],order=c(1,0,1),seasonal = c(0,1,2))
checkresiduals(fit_eff)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,1)(0,1,2)[96] errors
## Q* = 445.1, df = 188, p-value < 2.2e-16
##
## Model df: 4.   Total lags used: 192
```

We can perform forecasting:

```
elec_test=cbind(power_consum=elec_consumt_test[,1],Temp=elec_consumt_test[,2])
prev_t=forecast(fit_eff,h=984,xreg=elec_consumt_test[,2])
autoplot(elec_test[,"power_consum"], series="true data")+autolayer(prev_t$mean,series="SARIMA with covar
```



let evaluate this model

```
SARIMA_COVARIATES= rmse(prev_t$mean,elec_test[,"power_consum"])
SARIMA_COVARIATES
```

```
## [1] 14.89199
```

The result are better than those obtained with the auto.arima function.

#Second conclusion

let's resume all score we get and choose the best one.

```
cat('Forecasting results whit temperature : ' , '\n')
```

```
## Forecasting results whit temperature :
```

```
cat('RMSE with Auto Arima is:', arima_score,'\n')
```

```
## RMSE with Auto Arima is: 15.61163
```

```
cat('RMSE with SARIMA model is:', SARIMA_COVARIATES ,'\n')
```

```
## RMSE with SARIMA model is: 14.89199
```

The best one is SARIMA model.

We will now forecast the electricity consumption (kW) for 2/21/2010 based on the whole previous consumption information (ARIMA (5,0,0)(0,1,0)[96]).
The prediction interval = 24 hr for the entire day of 2/21/2010.So h =(24*60)/15 = 96 observations

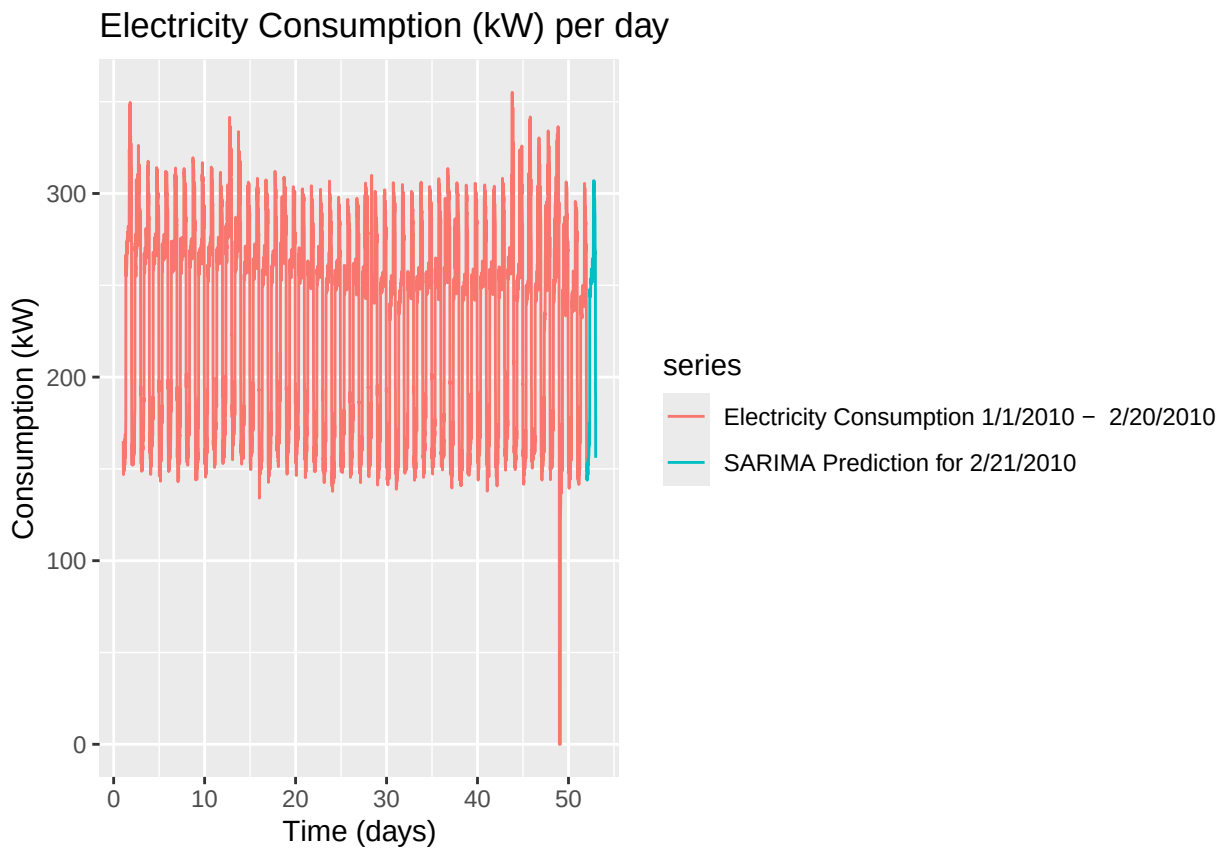
create new ts for forecast

```
datatoforcast= data.frame(
  Timestamp = seq(from = as.POSIXct("2010-02-21 00:00"),  by = "15 min", length.out=96),
  power = data[4892:4987,2], Temp =data[4892:4987,3]
)
head(datatoforcast)
```

```
##             Timestamp Power..kW. Temp..C..
## 1 2010-02-21 00:00:00         NA  11.66667
## 2 2010-02-21 00:15:00         NA  11.66667
## 3 2010-02-21 00:30:00         NA  11.66667
## 4 2010-02-21 00:45:00         NA  11.66667
## 5 2010-02-21 01:00:00         NA  11.66667
## 6 2010-02-21 01:15:00         NA  10.55556
```

```
datatoforcast_ts=ts(datatoforcast[,2:3], start=c(1,1), end = c(1,96),frequency = 96)
```

```
elec_co_21 = Arima(elec_consum_t[,1],xreg=elec_consum_t[,2],order=c(1,0,1),seasonal = c(0,1,2))
prev_co_21 = forecast(elec_co_21, h=96,xreg= datatoforcast_ts[,2])
autoplot(elec_consum,series="Electricity Consumption 1/1/2010 -  2/20/2010") +
  autolayer(prev_co_21$mean,series='SARIMA Prediction for 2/21/2010')+
  ggtitle ('Electricity Consumption (kW) per day') +
  xlab('Time (days)') +
  ylab('Consumption (kW)')
```




```
forecasts = data.frame(
  Timestamp = seq(from = as.POSIXct("2010-02-21 00:00"), by = "15 min", length.out = 96),
  `Forecast without using outdoor temperature` = as.numeric(round(prev_consum_21$mean, 1)),
  `Forecast using outdoor temperature` = as.numeric(round(prev_co_21$mean, 1))
)

# Create a new workbook
wb = createWorkbook()

# Add a worksheet
addWorksheet(wb, "Forecasts")

# Write the data to the worksheet
writeData(wb, "Forecasts", forecasts)


# Save the workbook
saveWorkbook(wb, "Martial_KOUASSI.xlsx", overwrite = TRUE)

# Output message
print("Forecasts saved to Martial_KOUASSI.xlsx")
```

```
## [1] "Forecasts saved to Martial_KOUASSI.xlsx"
```

#CONCLUSION

Our best model is the SARIMA with outdoor temperature. We may be able to reduce the residual autocorrelation and improve the overall accuracy of our forecasts by capturing the underlying patterns in the data.