Time Series Exam DSTI A23

2024-08-18

Introduction

We would like to forecast electricity consumption (kW) for 2/21/2010 for one building by taking into account or not the outdoor air temperature.

An Excel file containing electricity consumption (kW) and outdoor air temperature for one building has been provided. These quantities are measured every 15 minutes, from 1/1/2010 1:15 to 2/20/2010 23:45. In addition, outdoor air temperature are available for 2/21/2010.

We will proceed as displayed below 1. Forecast without using outdoor temperature, 2. Forecast by using outdoor temperature

Install and load packages

Load data and have a first look on them.

```
install.packages("forecast")
```

```
## Installation du package dans 'C:/Users/datas/AppData/Local/R/win-library/4.3'
## (car 'lib' n'est pas spécifié)
```

```
## Warning: impossible de supprimer l'installation précédente du package
## 'forecast'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problème lors de la
## copie de
## C:\Users\datas\AppData\Local\R\win-library\4.3\00LOCK\forecast\libs\x64\forecast.dll
## vers
## C:\Users\datas\AppData\Local\R\win-library\4.3\forecast\libs\x64\forecast.dll:
## Permission denied
```

```
## Warning: 'forecast' restauré
```

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

```
## Installation du package dans 'C:/Users/datas/AppData/Local/R/win-library/4.3'
## (car 'lib' n'est pas spécifié)
```

```
## Warning: impossible de supprimer l'installation précédente du package 'readxl'
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problème lors de la
## copie de
## C:\Users\datas\AppData\Local\R\win-library\4.3\00LOCK\readxl\libs\x64\readxl.dll
## vers C:\Users\datas\AppData\Local\R\win-library\4.3\readxl\libs\x64\readxl.dll:
## Permission denied
```

```
## Warning: 'readxl' restauré
```

```
install.packages("xts")
```

```
## Installation du package dans 'C:/Users/datas/AppData/Local/R/win-library/4.3'
## (car 'lib' n'est pas spécifié)
```

```
## Warning: impossible de supprimer l'installation précédente du package 'xts'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problème lors de la
## copie de
## C:\Users\datas\AppData\Local\R\win-library\4.3\00LOCK\xts\libs\x64\xts.dll vers
## C:\Users\datas\AppData\Local\R\win-library\4.3\xts\libs\x64\xts.dll: Permission
## denied
```

```
## Warning: 'xts' restauré
```

```
install.packages("randomForest")
```

```
## Installation du package dans 'C:/Users/datas/AppData/Local/R/win-library/4.3'
## (car 'lib' n'est pas spécifié)
```

```
## Warning: impossible de supprimer l'installation précédente du package
## 'randomForest'
```

```
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problème lors de la
## copie de
## C:\Users\datas\AppData\Local\R\win-library\4.3\00LOCK\randomForest\libs\x64\randomForest.d
ll
## vers
## C:\Users\datas\AppData\Local\R\win-library\4.3\randomForest\libs\x64\randomForest.dll:
## Permission denied
```

```
## Warning: 'randomForest' restauré
```

```
library(forecast)
```

```
## Warning: le package 'forecast' a été compilé avec la version R 4.3.3
```

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

```
library(readxl)
```

```
## Warning: le package 'readxl' a été compilé avec la version R 4.3.3
```

```
library(xts)
```

```
## Warning: le package 'xts' a été compilé avec la version R 4.3.3
```

```
## Le chargement a nécessité le package : zoo
```

```
## Warning: le package 'zoo' a été compilé avec la version R 4.3.3
```

```
##
## Attachement du package : 'zoo'
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(randomForest)
```

```
## Warning: le package 'randomForest' a été compilé avec la version R 4.3.3
```

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

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

```
data <- read_excel("2023-11-Elec-train.xlsx", sheet = 1, col_types = c("text", "numeric", "nu
meric"))
head(data)
```

Timestamp <chr>	Power (kW) <dbl>	Temp (C°) <dbl>
40179.052083333336	165.1	10.55556

Timestamp <chr>	Power (kW) <dbl>	Temp (C°) <dbl>
1/1/2010 1:30	151.6	10.55556
1/1/2010 1:45	146.9	10.55556
1/1/2010 2:00	153.7	10.55556
1/1/2010 2:15	153.8	10.55556
1/1/2010 2:30	159.0	10.55556

6 rows

There is on data with a different format. Let’s update it.

```
data$Timestamp[data$Timestamp == "40179.052083333336"] <- "1/1/2010 1:15"
# Check the update
head(data)
```

Timestamp <chr>	Power (kW) <dbl>	Temp (C°) <dbl>
1/1/2010 1:15	165.1	10.55556
1/1/2010 1:30	151.6	10.55556
1/1/2010 1:45	146.9	10.55556
1/1/2010 2:00	153.7	10.55556
1/1/2010 2:15	153.8	10.55556
1/1/2010 2:30	159.0	10.55556

6 rows

Update the format of the dates to Timestamp.

```
data$Timestamp <- as.POSIXct(data$Timestamp, format = "%m/%d/%Y %H:%M")
# View the parsed data
head(data)
```

Timestamp <dttm>	Power (kW) <dbl>	Temp (C°) <dbl>
2010-01-01 01:15:00	165.1	10.55556
2010-01-01 01:30:00	151.6	10.55556
2010-01-01 01:45:00	146.9	10.55556
2010-01-01 02:00:00	153.7	10.55556

Timestamp <dttm>	Power (kW) <dbl>	Temp (C°) <dbl>
2010-01-01 02:15:00	153.8	10.55556
2010-01-01 02:30:00	159.0	10.55556

6 rows

Have another and wider look at the data.

```
summary(data)
```

```
##    Timestamp                       Power (kW)      Temp (C°)
##  Min.   :2010-01-01 01:15:00   Min.   :  0.0   Min.   : 3.889
##  1st Qu.:2010-01-14 00:52:30   1st Qu.:162.9   1st Qu.: 9.444
##  Median :2010-01-27 00:30:00   Median :253.0   Median :11.111
##  Mean   :2010-01-27 00:30:00   Mean   :230.8   Mean   :10.946
##  3rd Qu.:2010-02-09 00:07:30   3rd Qu.:277.3   3rd Qu.:12.778
##  Max.   :2010-02-21 23:45:00   Max.   :355.1   Max.   :19.444
##                                NA's   :96
```

There are some 0. Maybe a power cut. It looks exceptionnal. Let's replace them to avoid issues later on (e.g. with in estimation).

```
data$"Power (kW)"[data$"Power (kW)" == 0] <- 12
#Check the update.
summary(data)
```

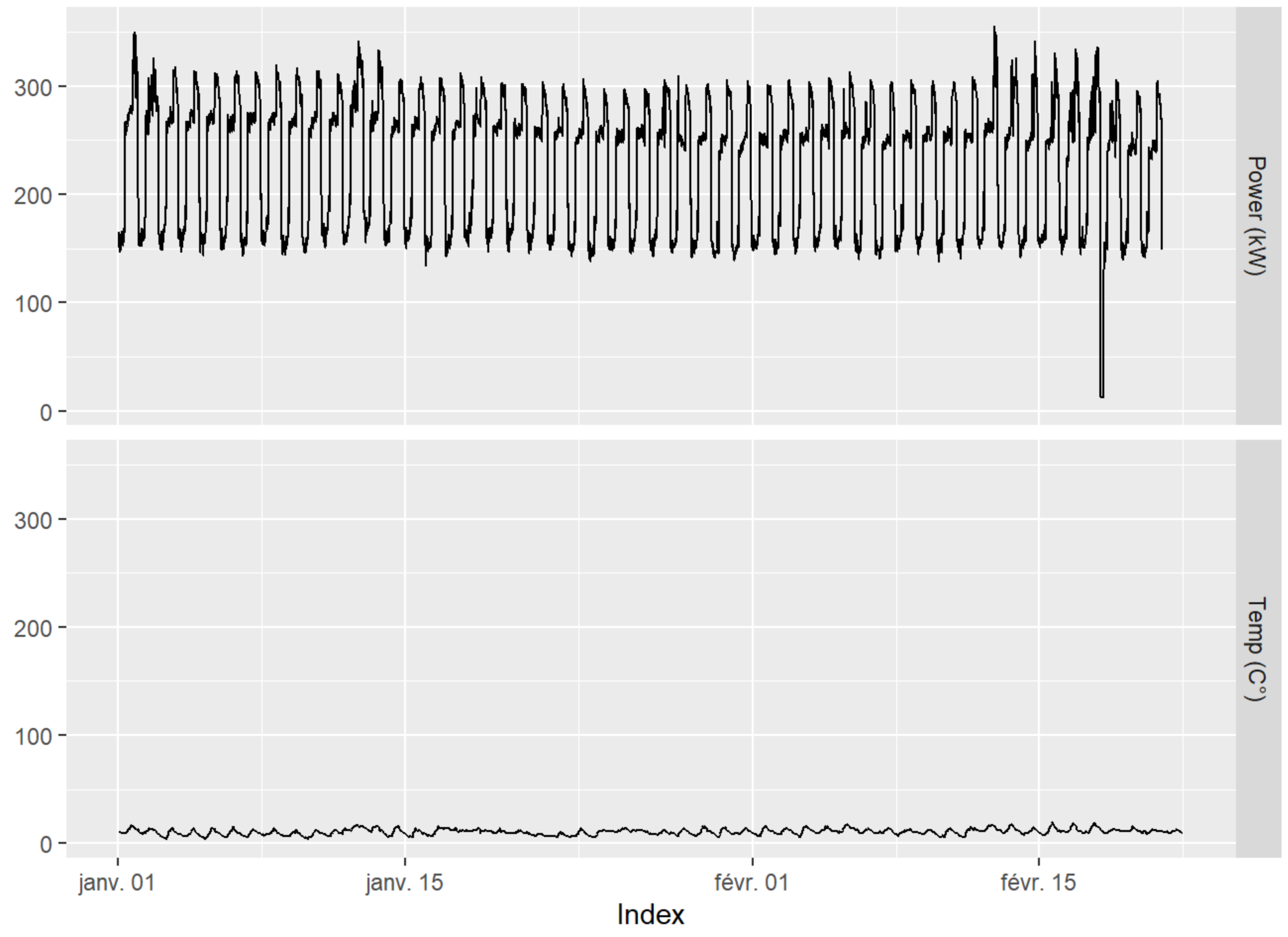
```
##    Timestamp                       Power (kW)      Temp (C°)
##  Min.   :2010-01-01 01:15:00   Min.   : 12.0   Min.   : 3.889
##  1st Qu.:2010-01-14 00:52:30   1st Qu.:162.9   1st Qu.: 9.444
##  Median :2010-01-27 00:30:00   Median :253.0   Median :11.111
##  Mean   :2010-01-27 00:30:00   Mean   :230.8   Mean   :10.946
##  3rd Qu.:2010-02-09 00:07:30   3rd Qu.:277.3   3rd Qu.:12.778
##  Max.   :2010-02-21 23:45:00   Max.   :355.1   Max.   :19.444
##                                NA's   :96
```

```
# Create a sequence of time stamps every 15 minutes
start_time <- as.POSIXct("2010-01-01 01:15")
end_time <- as.POSIXct("2010-02-21 23:45")
time_index <- seq(from = start_time, to = end_time, by = "15 min")
# Observations
observations <- data[,-1]
xts_data <- xts(observations, order.by = time_index)
#Check the structure
str(xts_data)
```

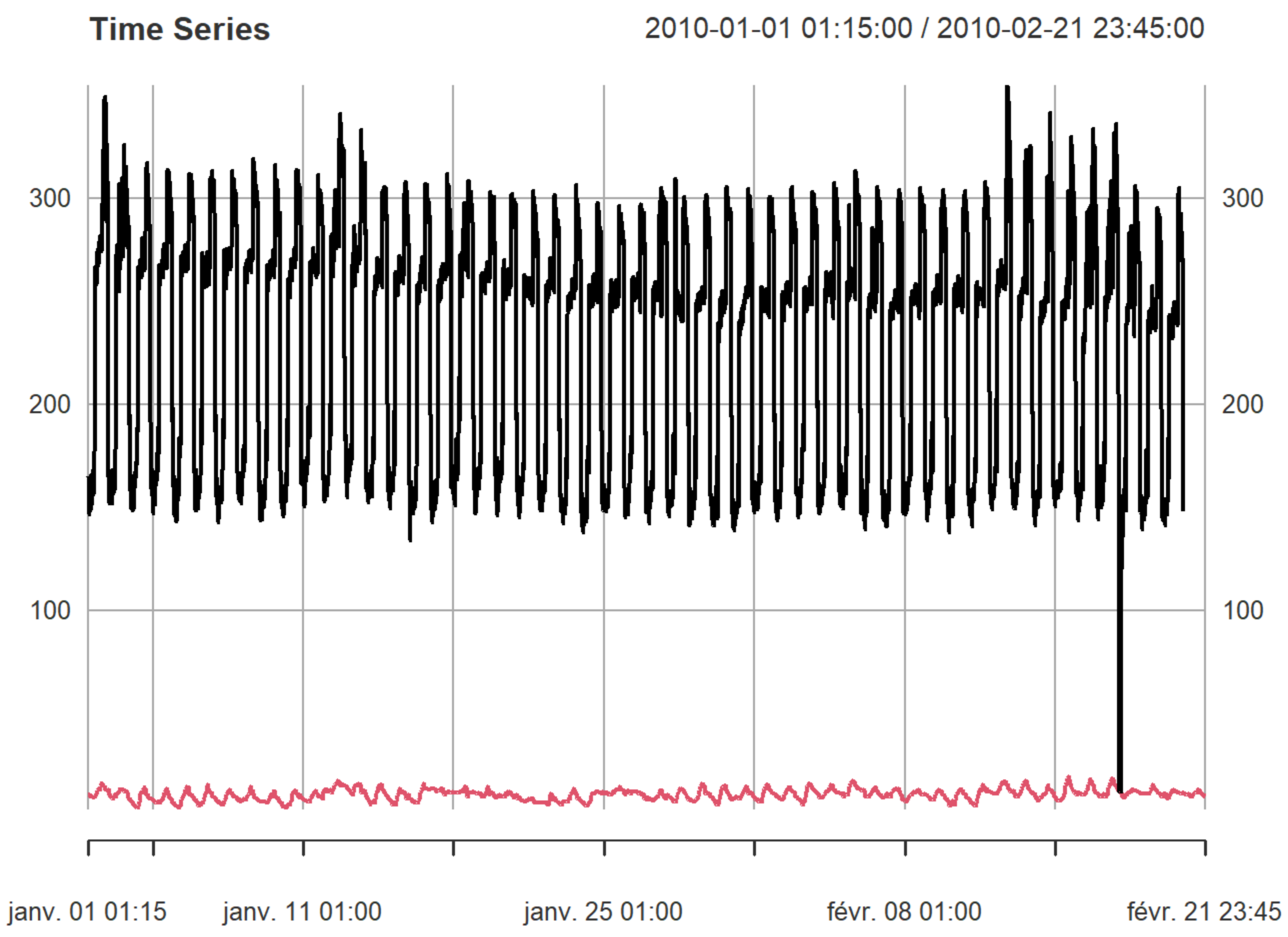
```
## An xts object on 2010-01-01 01:15:00 / 2010-02-21 23:45:00 containing:
##   Data:    double [4987, 2]
##   Columns: Power (kW), Temp (C°)
##   Index:   POSIXct,POSIXt [4987] (TZ: "")
```

Let’s have a look of the data on a graph.

```
autoplot(xts_data)
```



```
plot(xts_data, , main = "Time Series")
```



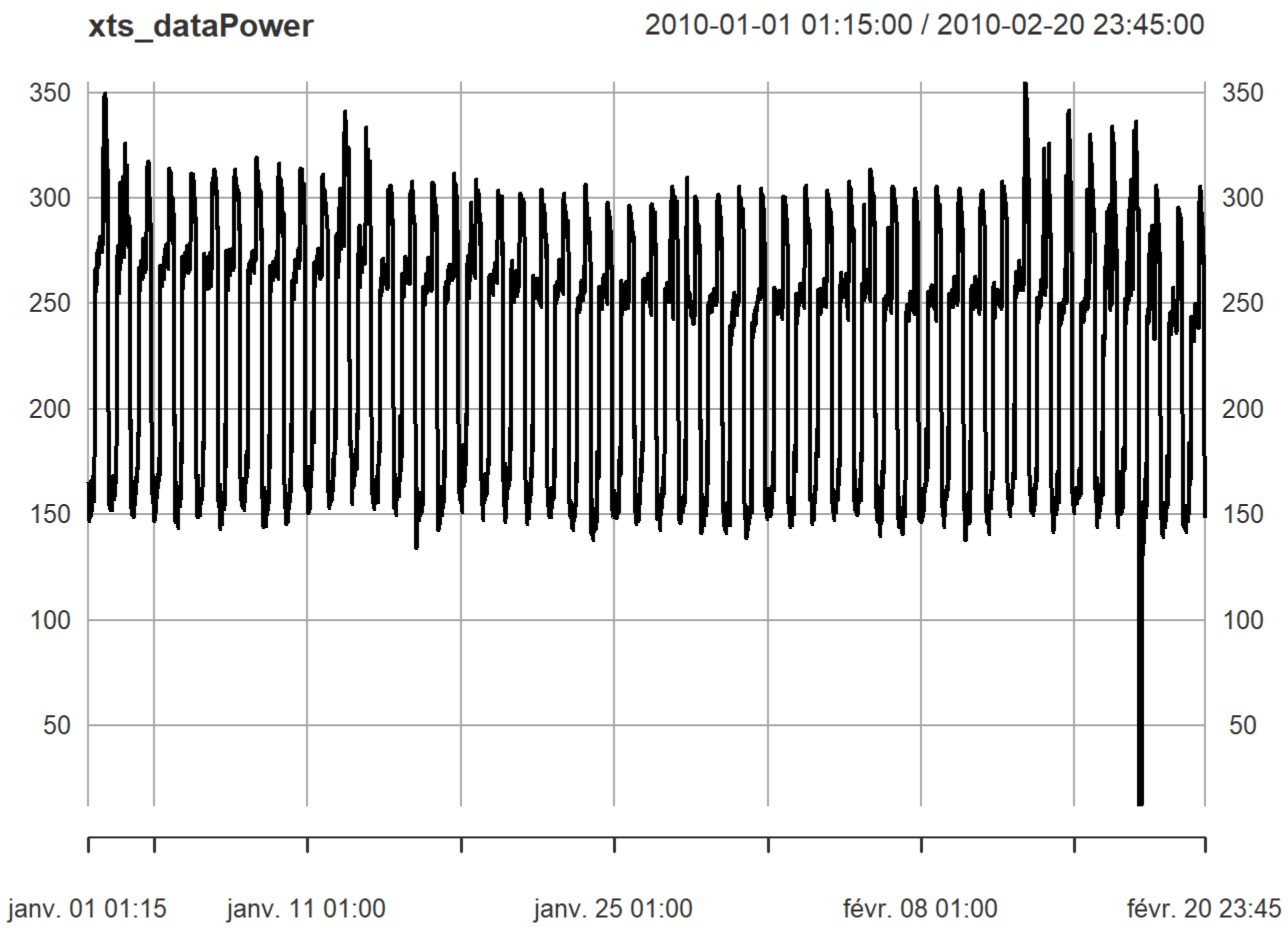
We can see the two Time Series Power (kW) and Temp (C°).

We do not have the last Power (kW). Let’s focus on this Time Series separetly first. The lowest pic should be due to the 0 (replaced by a low value). Maybe a power cut in the building. We can note that thera are some higher picx around the fifteen of february, the first and the 12th of January. It is not easy to know if it is a recurrent pattern with this size of the dataset. The fact that there are more such highest pics at the recent dates may have an increase impact on some forecast methods like Holt Winter where more weights are on recent data. We can see that the most recent pics are more like the ones before these high pics.

Forecast electricity consumption (kW) for 2/21/2010 by using electricity consumption but not outdoor temperature time series.

Create Power (KW) set of specific working variables.

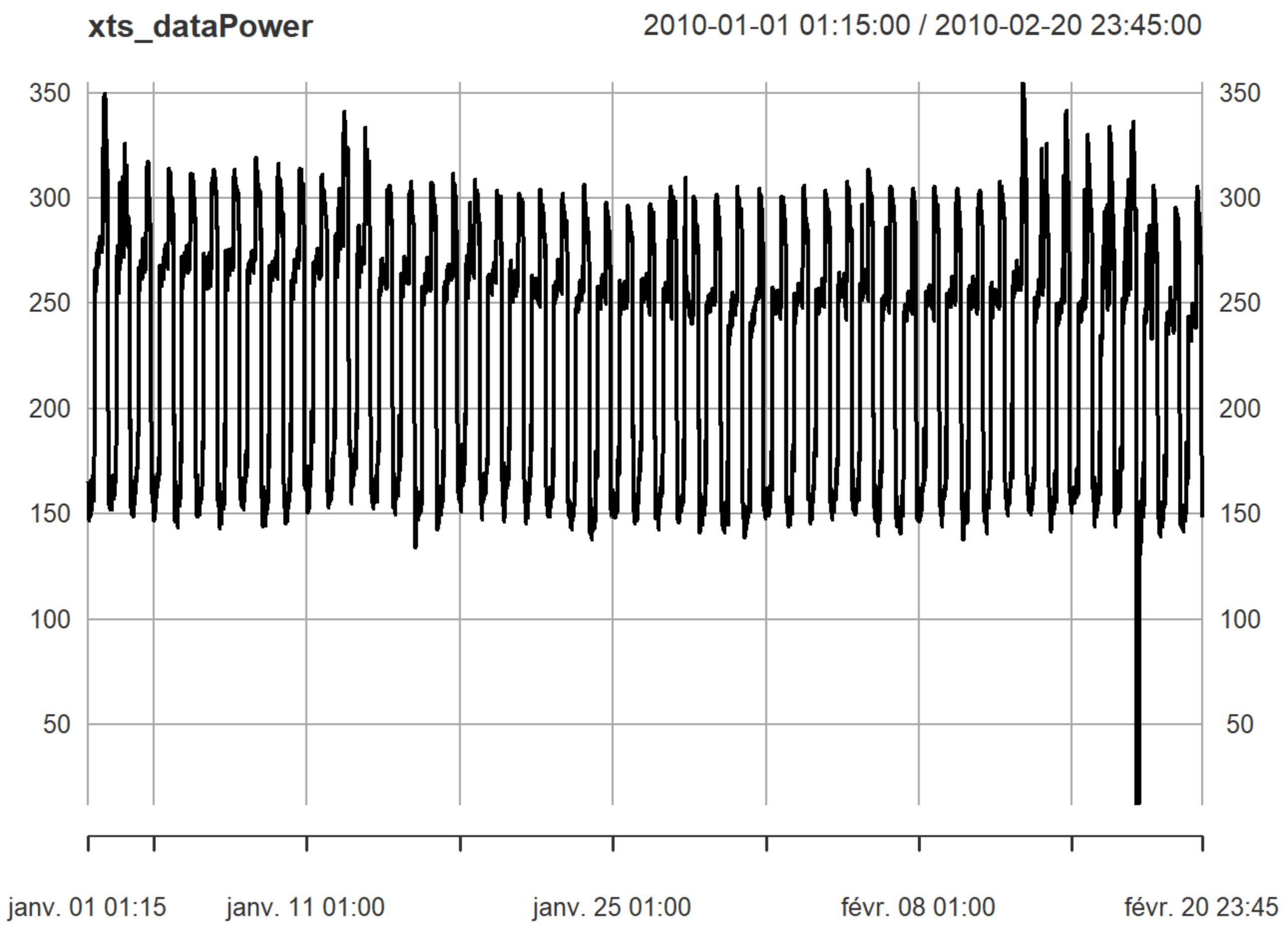
```
observationsPower <- observations[1:(nrow(observations) - 96),1]
end_timePower <- as.POSIXct("2010-02-20 23:45")
time_indexPower <- seq(from = start_time, to = end_timePower, by = "15 min")
xts_dataPower <- xts(observationsPower, order.by = time_indexPower)
head(xts_dataPower)
#Plot the TS
plot(xts_dataPower)
```



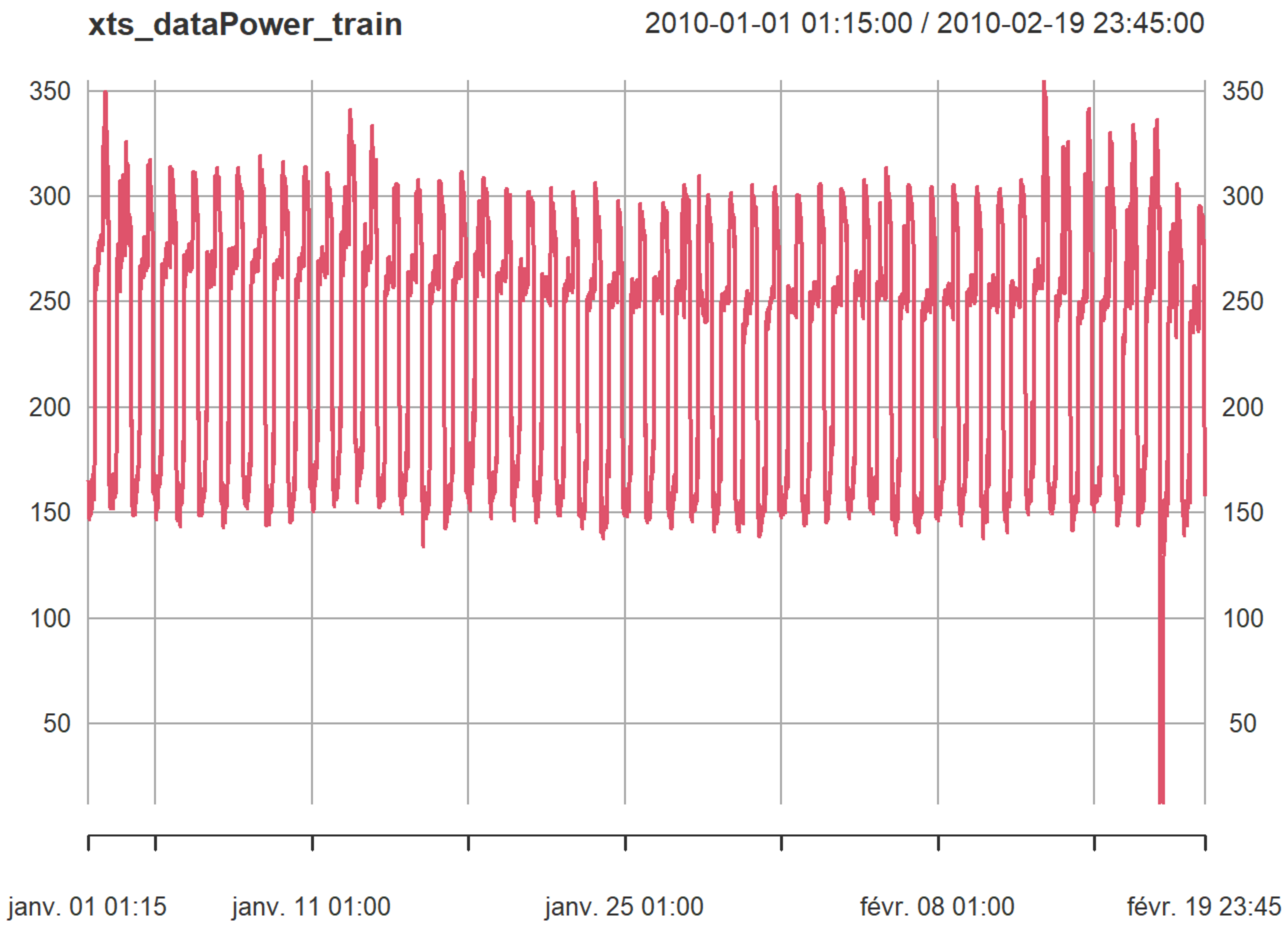
##Split the dataset into train and test datasets. ##Preparation for time series objects.

```
observationsPower_train <- observations[1:(nrow(observations) - 96*2),1]
end_timePower_train <- as.POSIXct("2010-02-19 23:45")

#One period is missing / to forecast for Power (kw)
observationsPower_test <- observations[(nrow(observations) - 96*2 + 1):(nrow(observations)- 9
6),1]
start_timePower_test = end_timePower + (- 96 + 1)*60*15

time_indexPower_train <- seq(from = start_time, to = end_timePower_train, by = "15 min")
xts_dataPower_train <- xts(observationsPower_train, order.by = time_indexPower_train)

time_indexPower_test <- seq(from = start_timePower_test, to = end_timePower, by = "15 min")
xts_dataPower_test <- xts(observationsPower_test, order.by = time_indexPower_test)
```

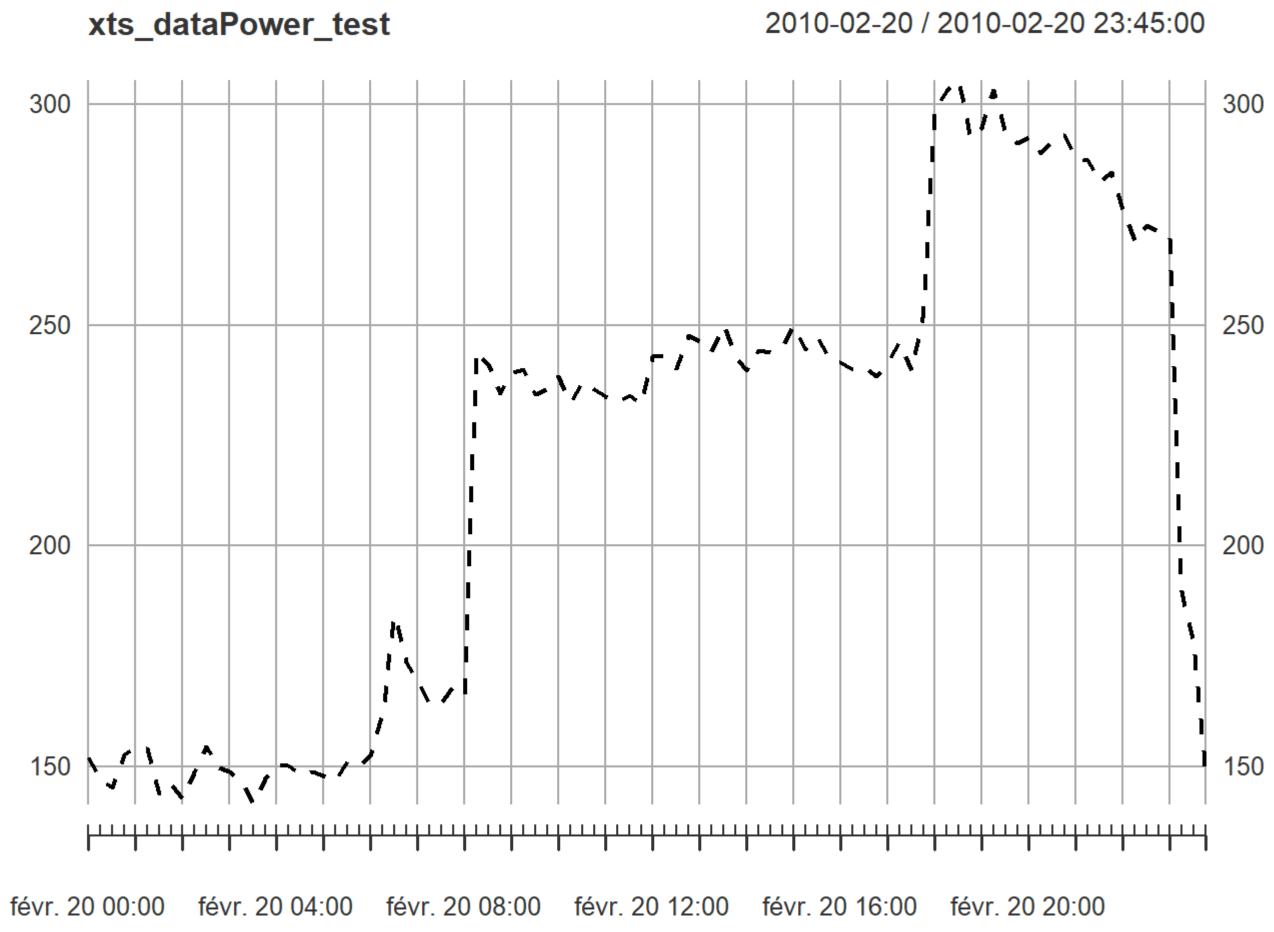
```
#Plot the TS
plot(xts_dataPower)
```



```
plot(xts_dataPower_train,col=2)
```

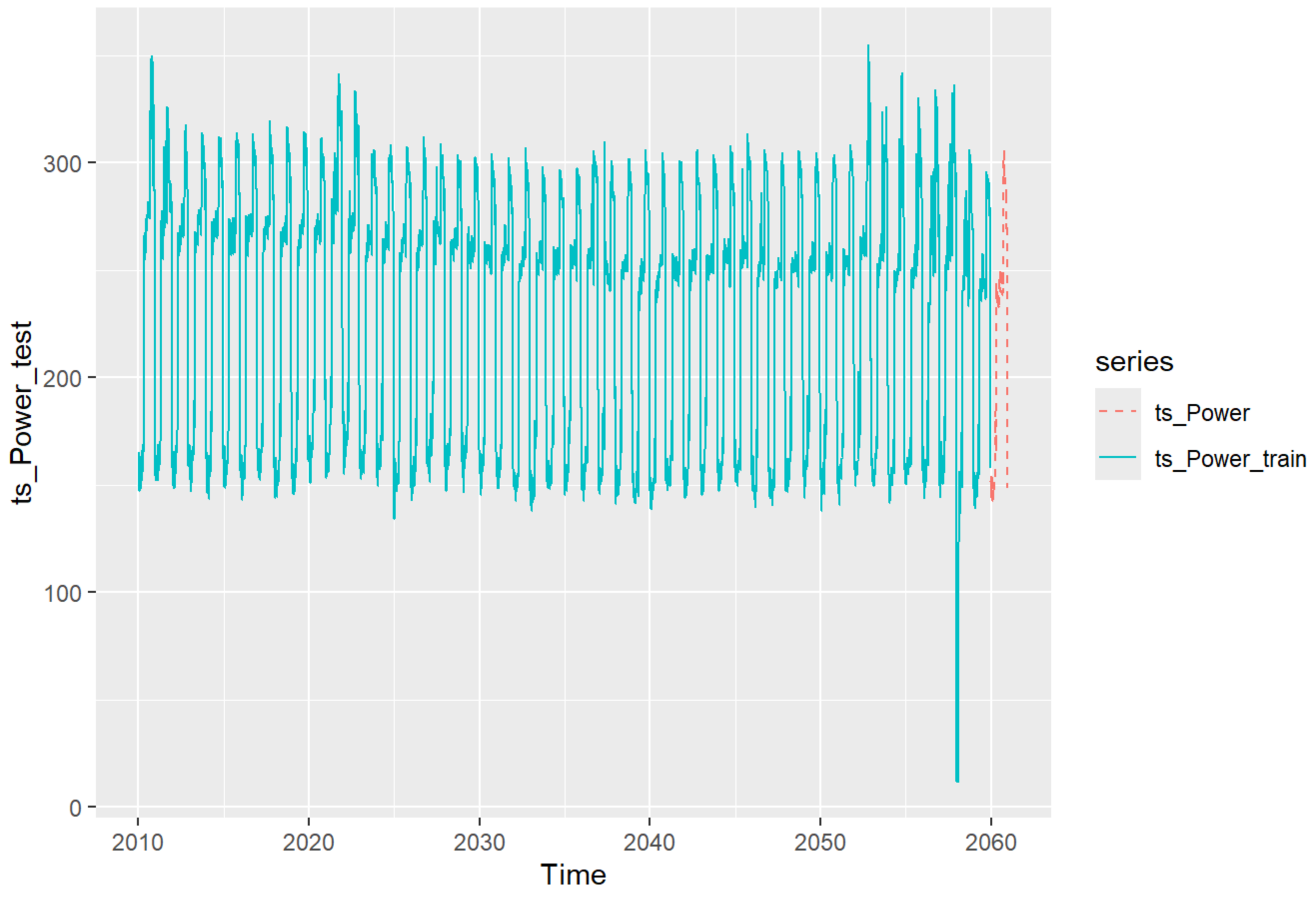


```
plot(xts_dataPower_test,lty=2)
```



Convert xts to ts objects.

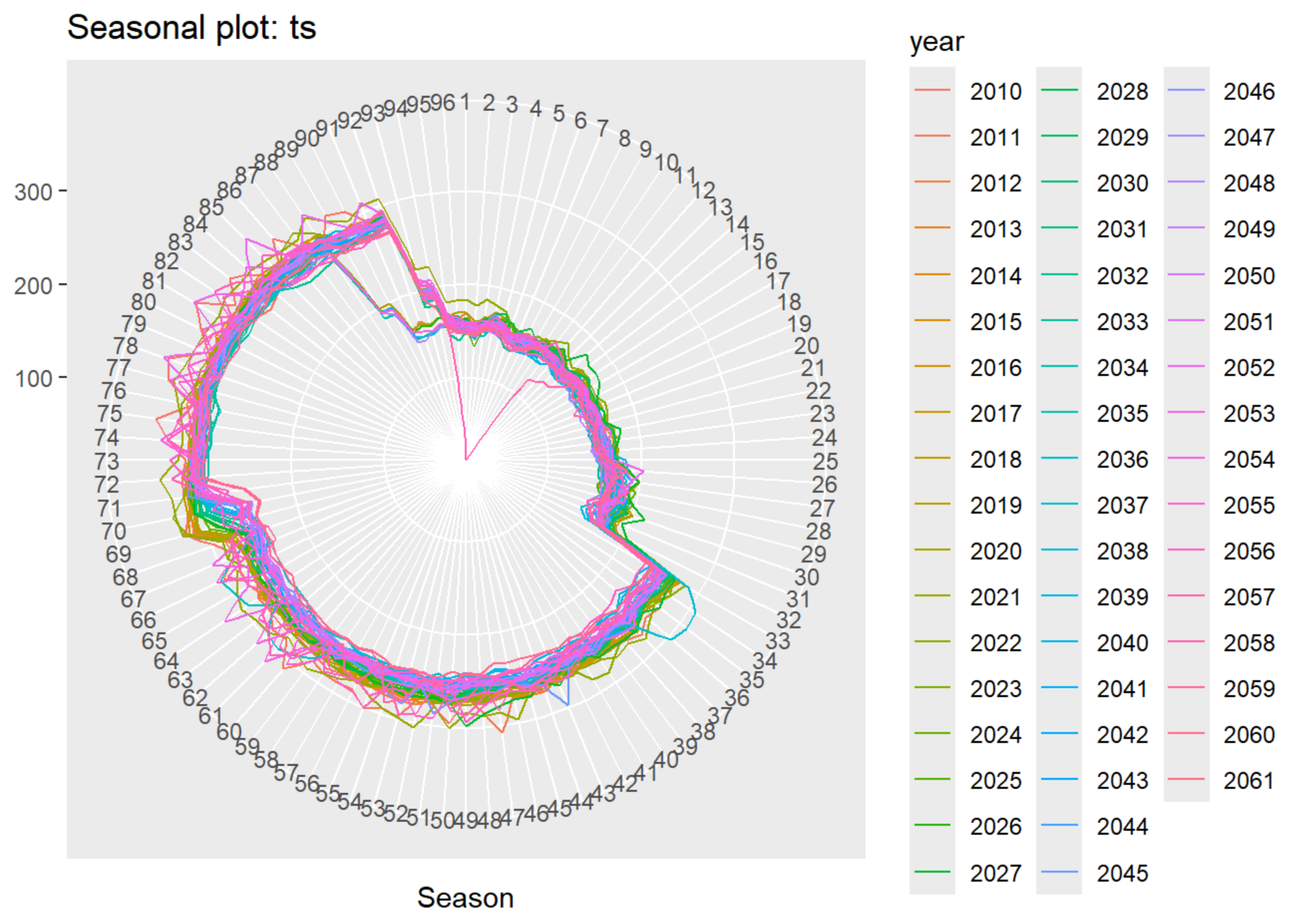
```
frequency = 24*60/15

ts=ts(coredata(xts_data[,1]),start=c(2010,5),frequency=frequency)

ts_Power=ts(coredata(xts_dataPower),start=c(2010,5),frequency=frequency)

ts_Power_train=ts(coredata(xts_dataPower_train),start=c(2010,5),frequency=frequency)
ts_Power_test=ts(coredata(xts_dataPower_test),start=c(2010,5+(nrow(observations)- frequency*
2)),frequency=frequency)

autoplot (ts_Power_test, series = "ts_Power", lty=2)+
  autolayer (ts_Power_train, series = "ts_Power_train")
```



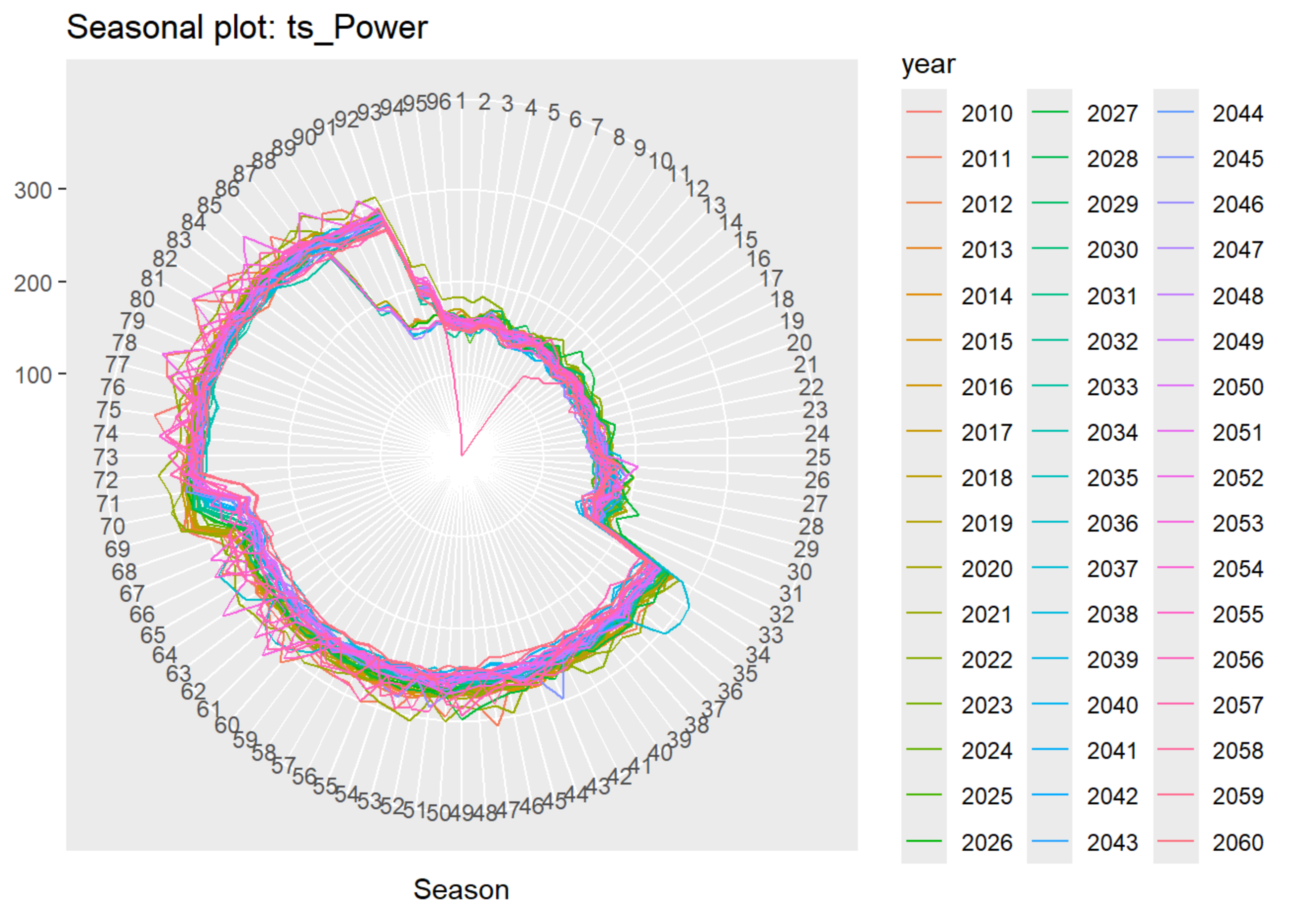
Let’s start to decompose the somponents of the Power Time Series. It seems that we have a seasonal component.

```
ggseasonplot(ts, polar=TRUE)
```

```
## Warning: Removed 97 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

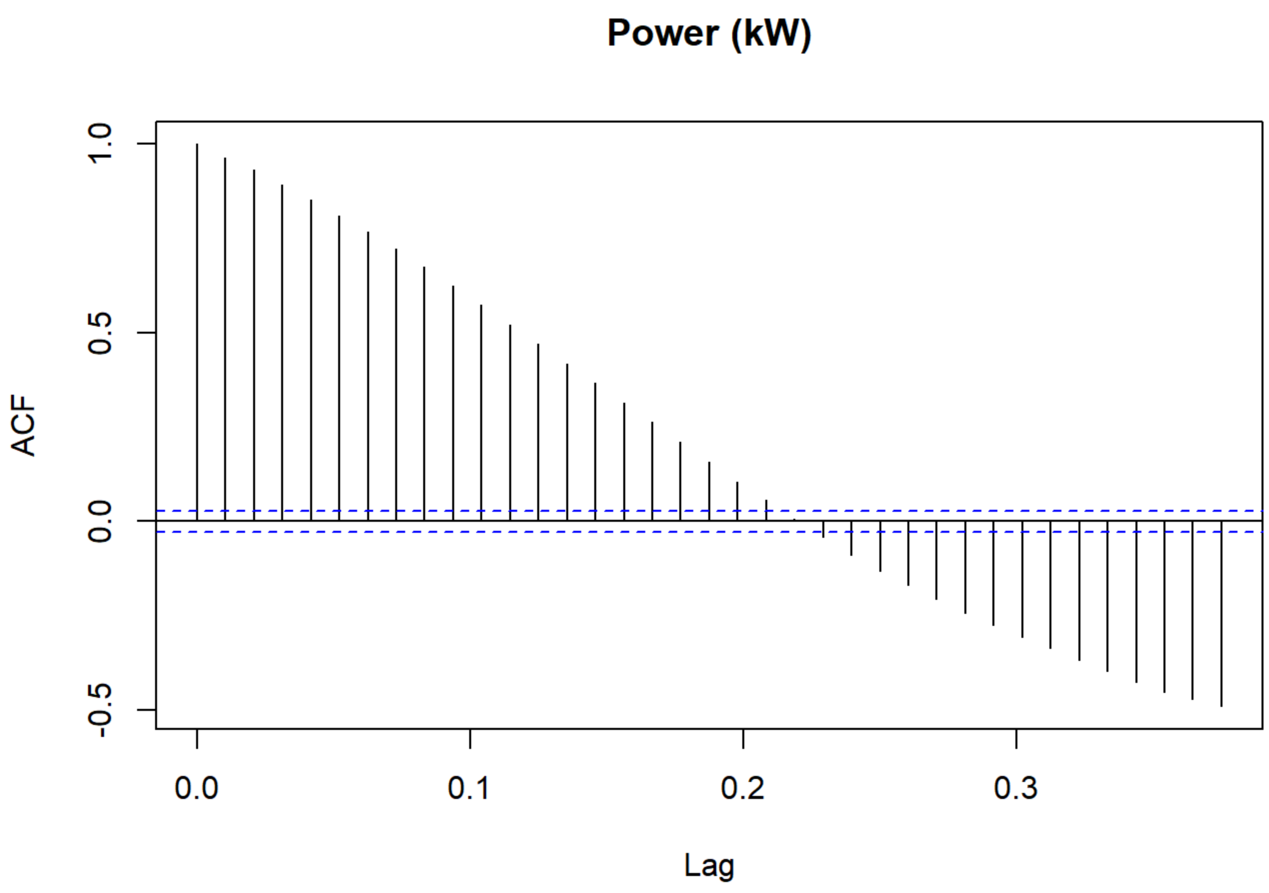


```
ggseasonplot(ts_Power, polar=TRUE)
```



With the ACF, we can check that there is also a trend component.

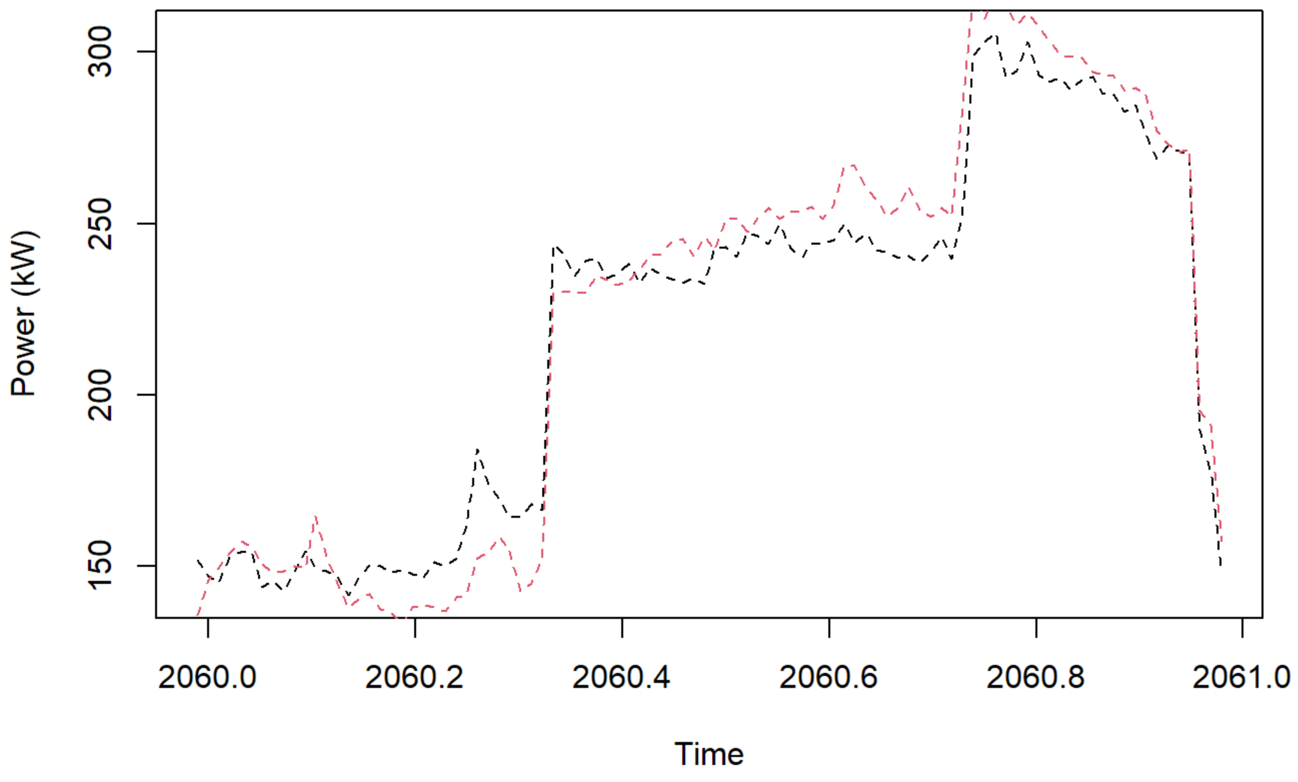
```
acf(ts_Power)
```



FIRST MODELS

Let’s start by looking at Holt Winter model with an additive seasonality without damp then (the amplitude of the seasonnality does not seem to change).

```
model_hw <- HoltWinters(ts_Power_train,seasonal="additive")
prev_hw<-predict(model_hw,n.ahead=frequency)
plot(ts_Power_test, lty=2)
lines(prev_hw, col=2, lty=2)
```



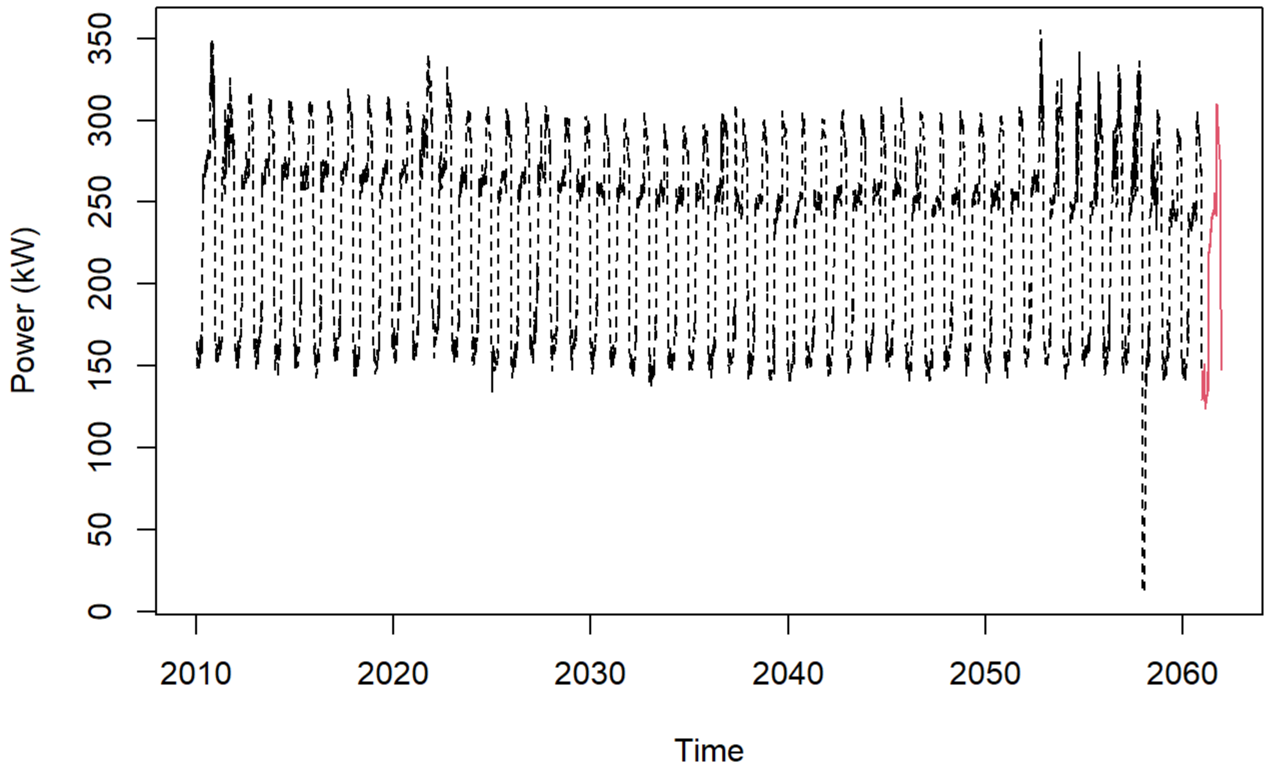
```
cat('HW :',sqrt(mean((prev_hw-ts_Power_test)^2)), '\n')
```

```
## HW : 11.64876
```

We get a similar shape. However some low/high Power are under/over estimated. The RMSE is 11.64876.

Let’s have a quick look on what we would get on all the dataset.

```
hw_model <- HoltWinters(ts_Power,seasonal="additive")
prev<-predict(hw_model,n.ahead=frequency)
plot(ts, lty=2)
lines(prev,col=2)
```

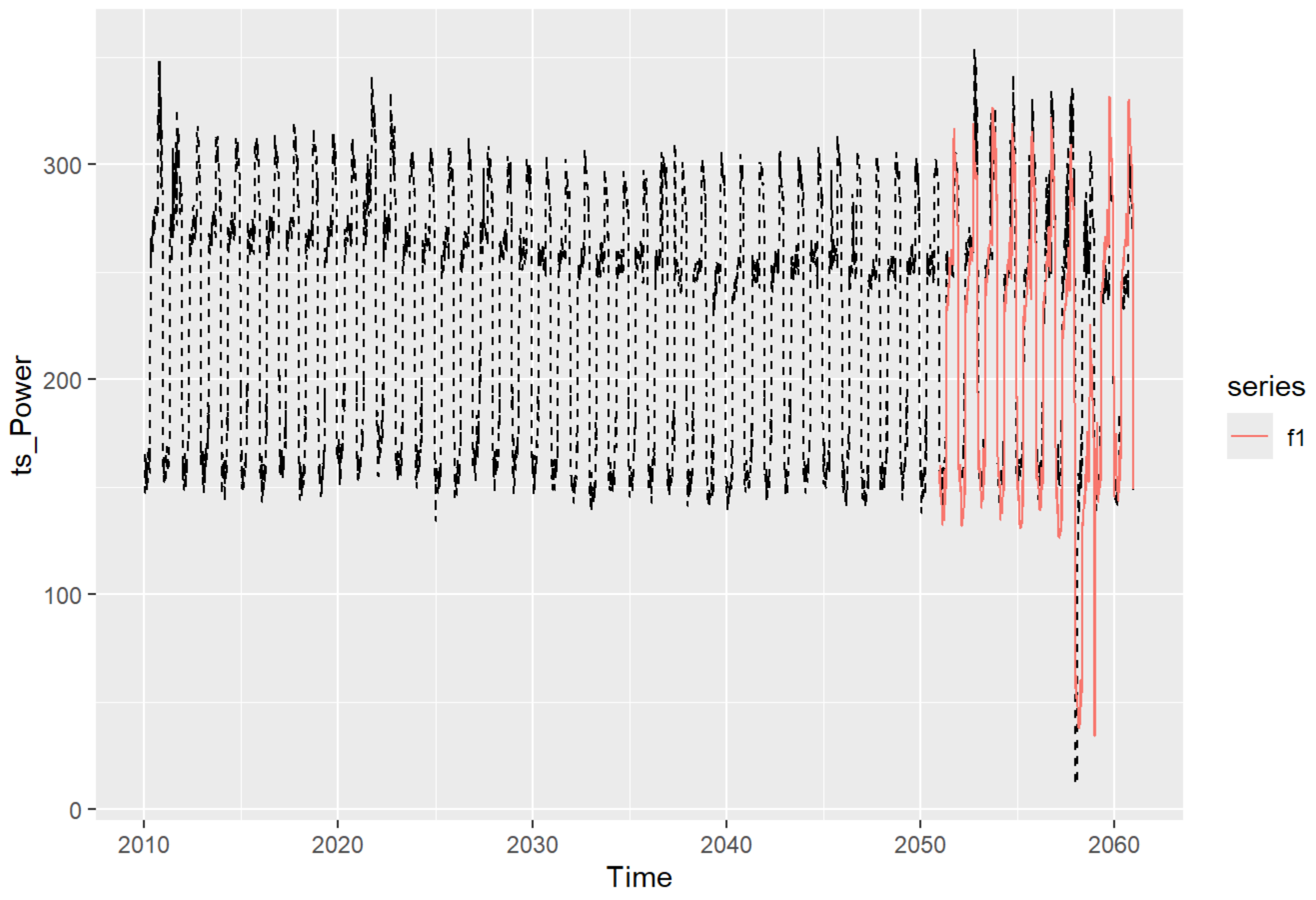


We may have a specific RMSE. By cross validating it is indeed a bit worse (14.84217 instead of 11.64876) but should be more reliable.

```
# Implementation of cross validation

#We extract the final test set training and test set_
start_train = c(2010,5)
end_train = c(2010,5+(nrow(observationsPower_train)))
start_Power_test = c(2010,5+(nrow(observationsPower_train))+1)
end_Power_test = c(2010,5+(nrow(observationsPower_train))+1+96)
ts_Power_train_cv=window(ts_Power,start=start_train, end=end_train)
ts_Power_test_temp=window(ts_Power,start=start_Power_test, end=end_Power_test)
```

```
## Warning in window.default(x, ...): valeur de 'end' inchangée
```

```
#We will now forecast one day using all the previous observations
forecasting_1=NULL
forecasting_2=NULL

nb_iter=10
for (step in 0:(nb_iter-1)){
    start_train = c(2010,5)
    end_train = c(2010,5+(nrow(observationsPower_train)-96*step))
    ts_Power_train_cv=window(ts_Power,start=start_train, end=end_train)
    model_hw_cv <- HoltWinters(ts_Power_train_cv,seasonal="additive")
    forecasting_1=c(predict(model_hw_cv,n.ahead=96),forecasting_1)
}

f_true = window(ts_Power,start=c(2010,5+(nrow(observationsPower_train)-frequency*step)), end=
c(2010,5+(nrow(observationsPower_train)-96*step)+nb_iter*frequency-1))

f1=ts(forecasting_1,start=end_train,frequency = 96)
autoplot (ts_Power, series = "ts_Power", lty=2, col=1) + autolayer (f1)
```



```
cat('RMSE for additive HW without damping:',100*mean(abs(forecasting_1-f_true)/f_true),'\n')
```

```
## RMSE for additive HW without damping: 14.84217
```

SARIMA

SARIMA Auto

Le’s have a look at other, more elaborate models like SARIMA.

First, the auto ARIMA.

```
model_arima_auto = auto.arima (ts_Power_train)
summary(model_arima_auto)
```

```
## Series: ts_Power_train
## ARIMA(5,0,1)(0,1,0)[96]
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ma1
##       0.4028  0.2822  0.1915  -0.2192  0.0669  0.3194
## s.e.  0.1237  0.0868  0.0192   0.0295  0.0319  0.1240
##
## sigma^2 = 136.9:  log likelihood = -18222.12
## AIC=36458.24   AICc=36458.26   BIC=36503.43
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0859759 11.57332 6.668107 -0.829717 3.755572 0.7125138
##                       ACF1
## Training set -0.0004281985
```

ARIMA(5,0,1)(0,1,0)[96] is suggested. ar5 and ma1 are over the interval of 2*se => significant. The RMSE is better than HW (14.84217).

Let’s cross validate to check the reliability.

```
# CV

#We extract the final test set training and test set_
start_train = c(2010,5)
end_train = c(2010,5+(nrow(observationsPower_train)))
start_Power_test = c(2010,5+(nrow(observationsPower_train))+1)
end_Power_test = c(2010,5+(nrow(observationsPower_train))+1+96)
ts_Power_train_cv=window(ts_Power,start=start_train, end=end_train)
ts_Power_test_temp=window(ts_Power,start=start_Power_test, end=end_Power_test)
```

```
## Warning in window.default(x, ...): valeur de 'end' inchangée
```

```
#We will now forecast one day using all the previous observations
forecasting_1=NULL
forecasting_SARIMA_auto=NULL

nb_iter=5
for (step in 0:(nb_iter-1)){
    start_train = c(2010,5)
    end_train = c(2010,5+(nrow(observationsPower_train)-frequency*step))
    ts_Power_train_cv=window(ts_Power,start=start_train, end=end_train)

    model_hw_cv <- HoltWinters(ts_Power_train_cv,seasonal="additive")
    model_SARIMA_auto_cv <- auto.arima (ts_Power_train_cv)

    forecasting_1=c(predict(model_hw_cv,n.ahead=frequency),forecasting_1)
    forecasting_SARIMA_auto=c(predict(model_SARIMA_auto_cv,n.ahead=frequency)$pred,forecastin
g_SARIMA_auto)

}

f_true = window(ts_Power,start=c(2010,5+(nrow(observationsPower_train)-frequency*step)), end=
c(2010,5+(nrow(observationsPower_train)-96*step)+nb_iter*frequency-1))
```

SARIMA auto seems worst now and both have a worse RMSE.

```
cat('RMSE for additive HW without damping:',100*mean(abs(forecasting_1-f_true)/f_true),'\n')
```

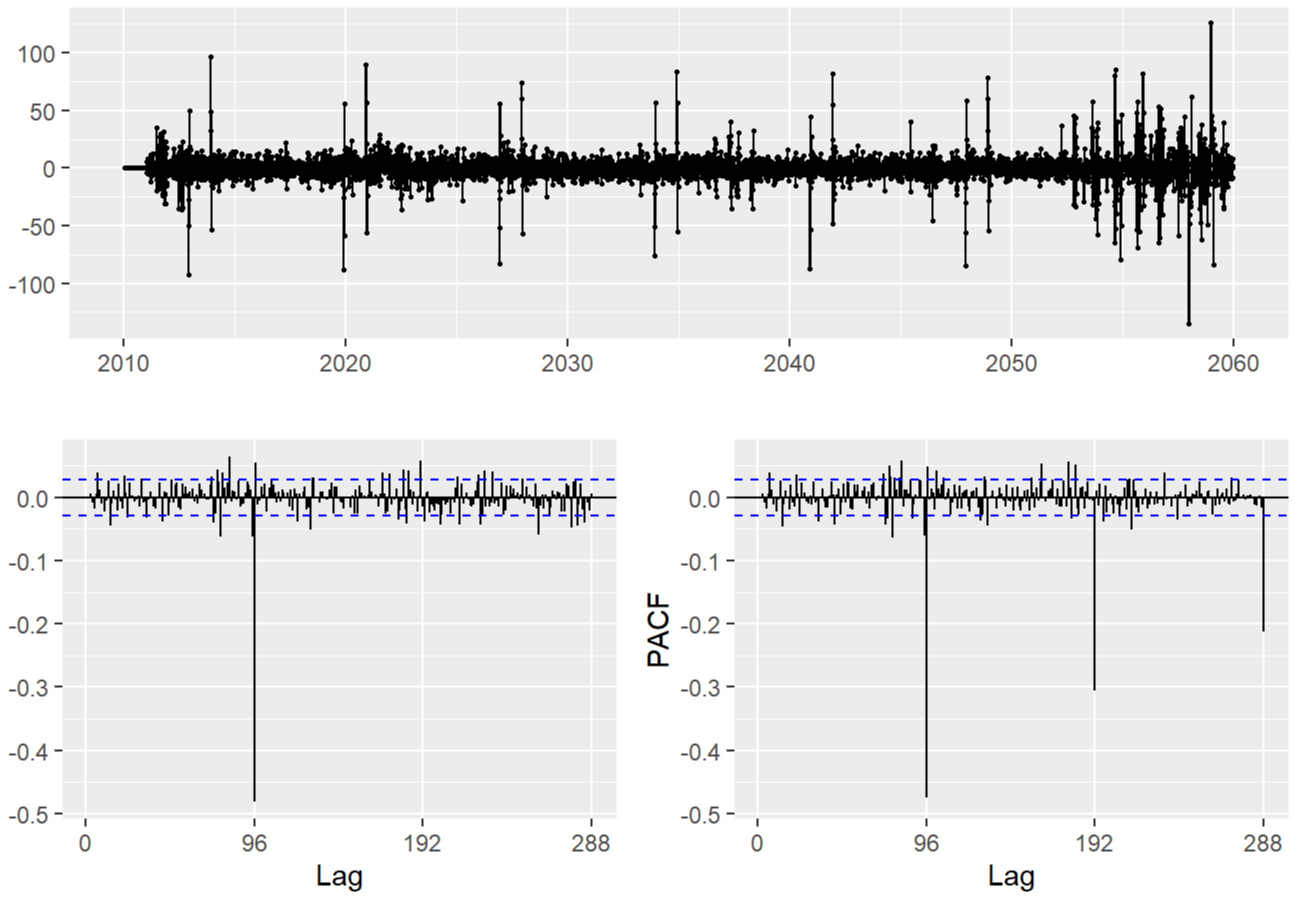
```
## RMSE for additive HW without damping: 22.99119
```

```
cat('RMSE for SARIMA auto:',100*mean(abs(forecasting_SARIMA_auto-f_true)/f_true),'\n')
```

```
## RMSE for SARIMA auto: 28.34863
```

By looking at the SARIMA auto residuals, we can see that we do not get a white noise and it looks like there is still a SMA1 to consider as well(PACF exponential decrease and ACF pic at 96).

```
model_arima_auto %>% residuals() %>% ggtsdisplay()
```

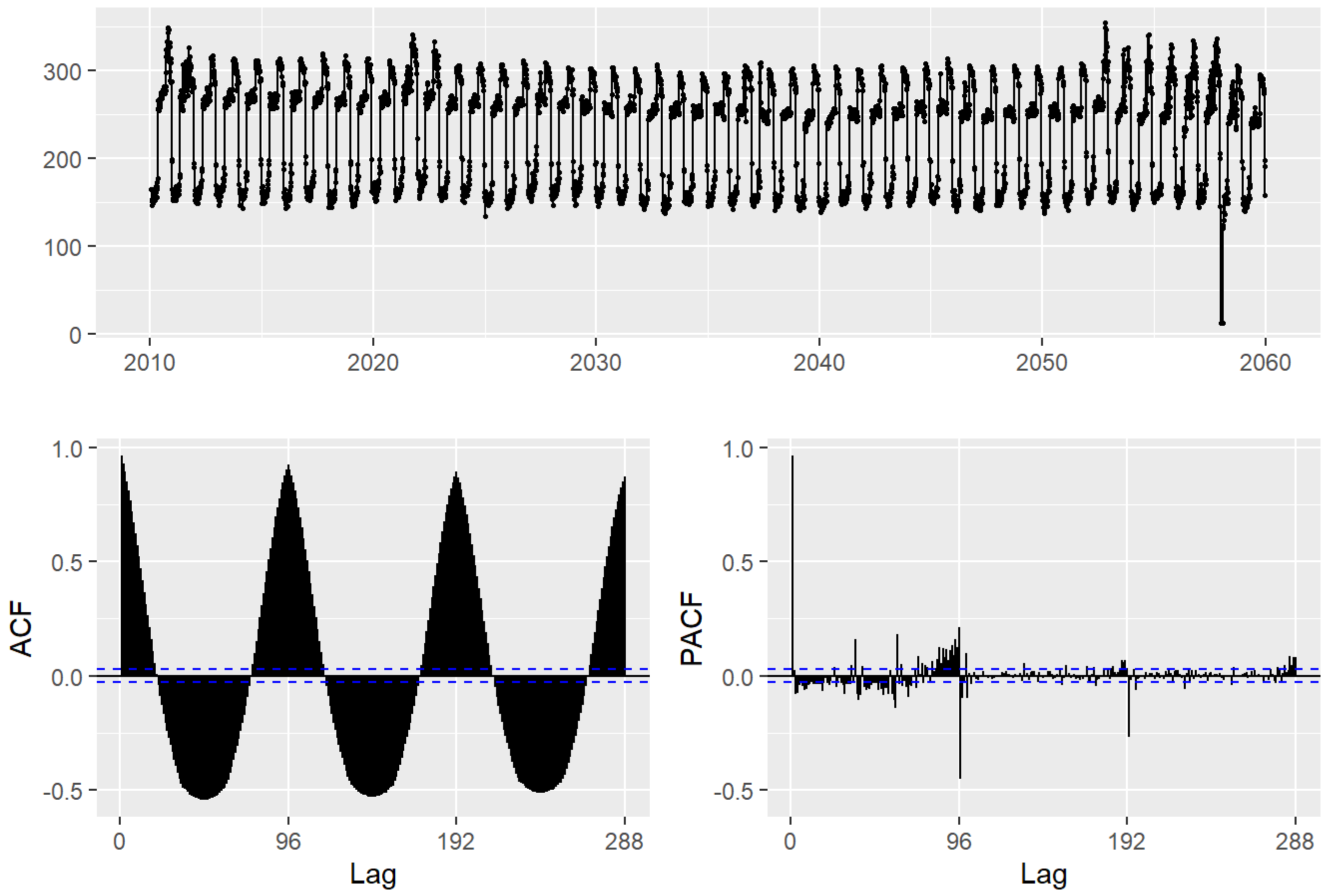


SARIMA Manual

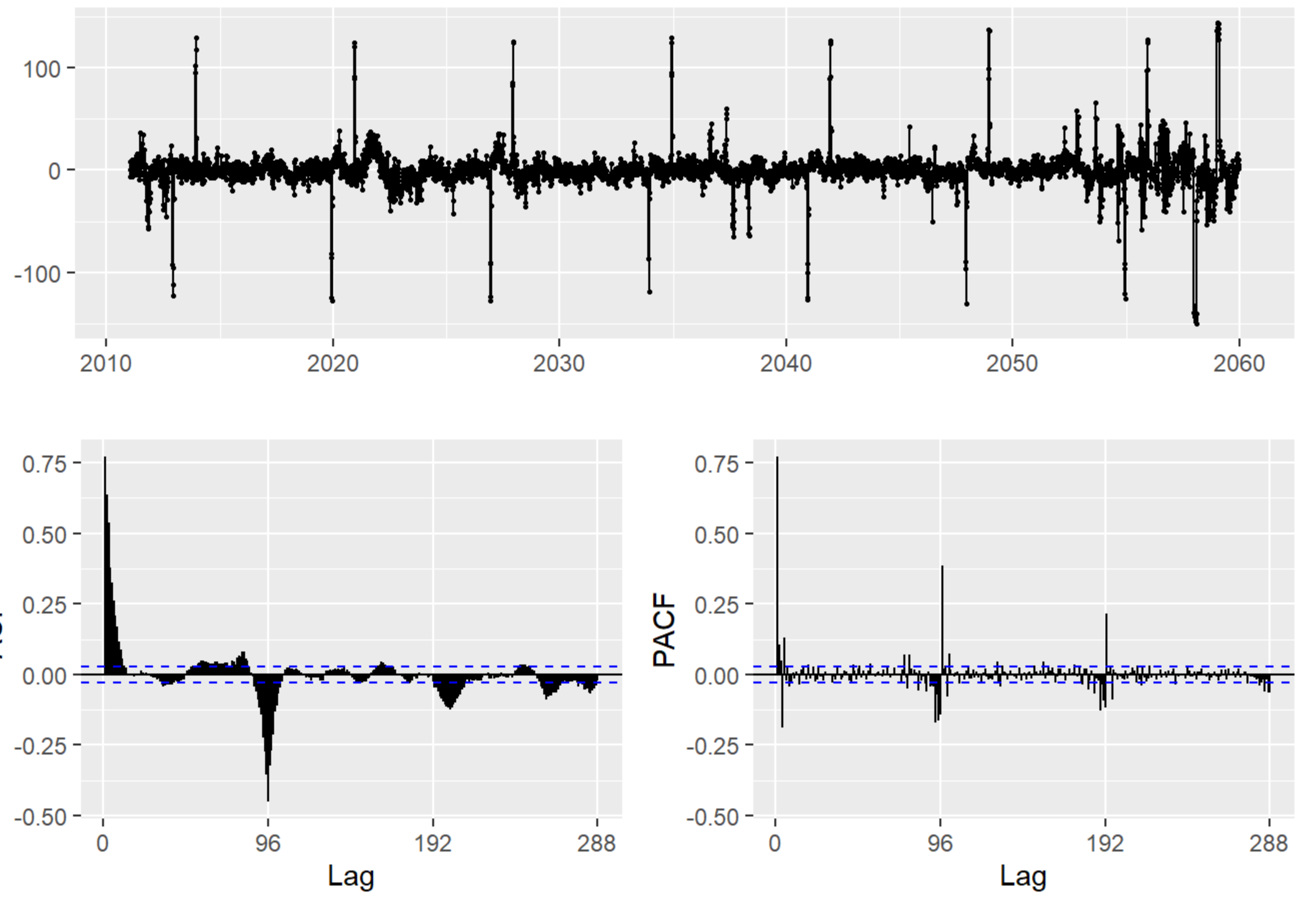
Let’s see if we can do better manually.

We have already noted that we have a season. We can also see it in the ACF. Let’s differentiate to remove it.

```
ggtsdisplay (ts_Power_train)
```



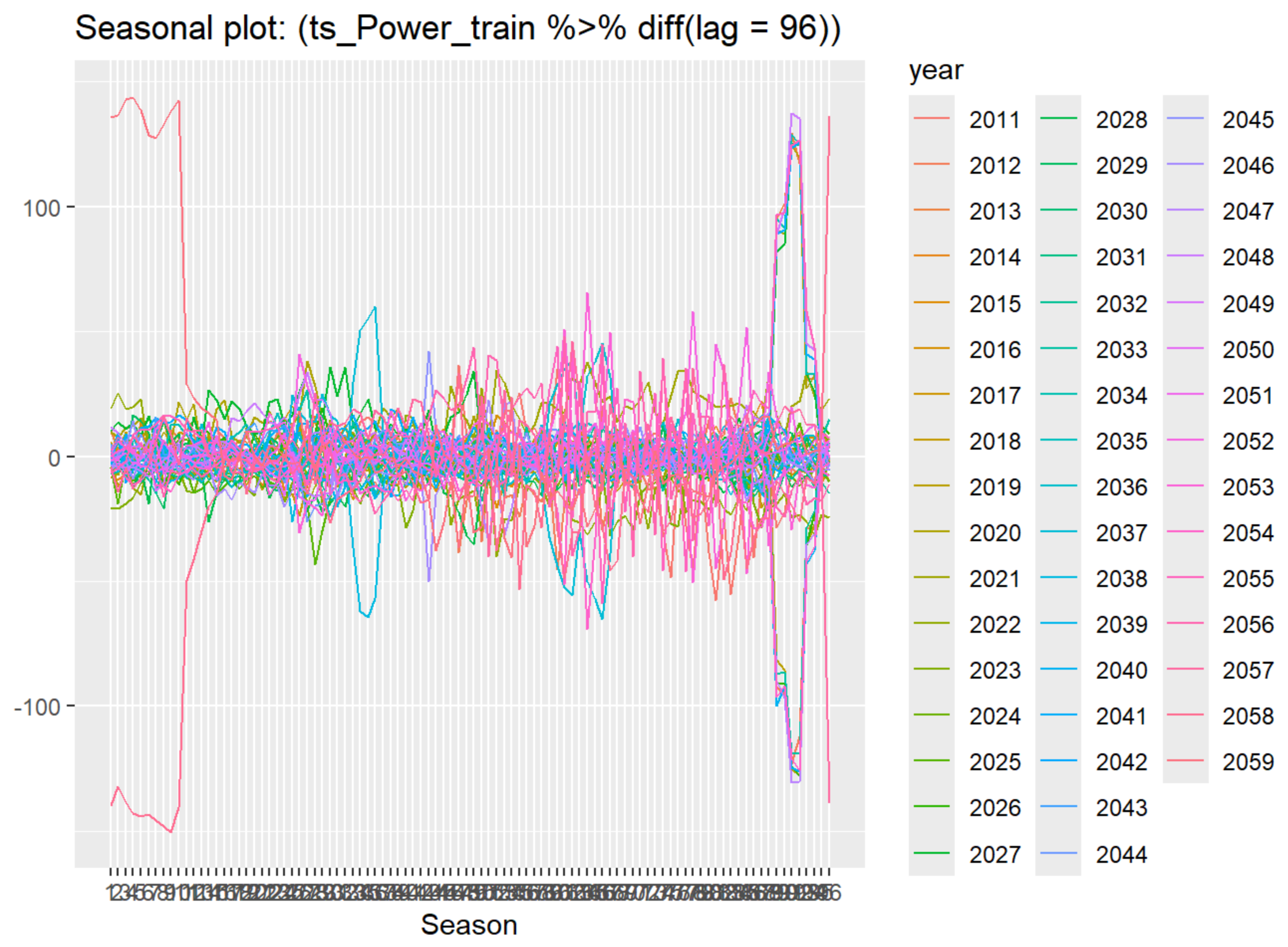
```
ggtsdisplay (ts_Power_train %>% diff(lag=96))
```



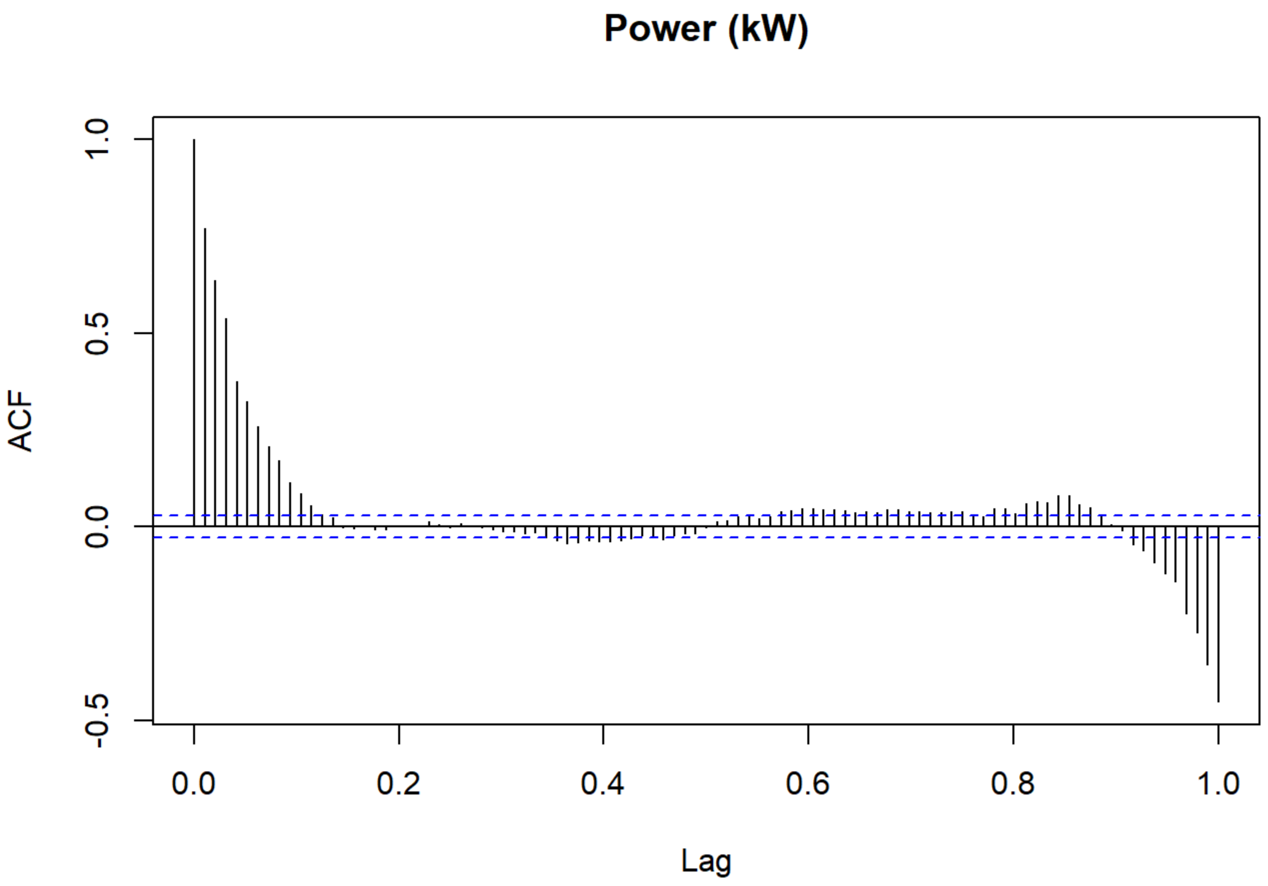
We can see a SMA1 (PACF exponential decrease and a pic in ACF at 96)

We still have several pics but it does not seem to be a seasonal effect.

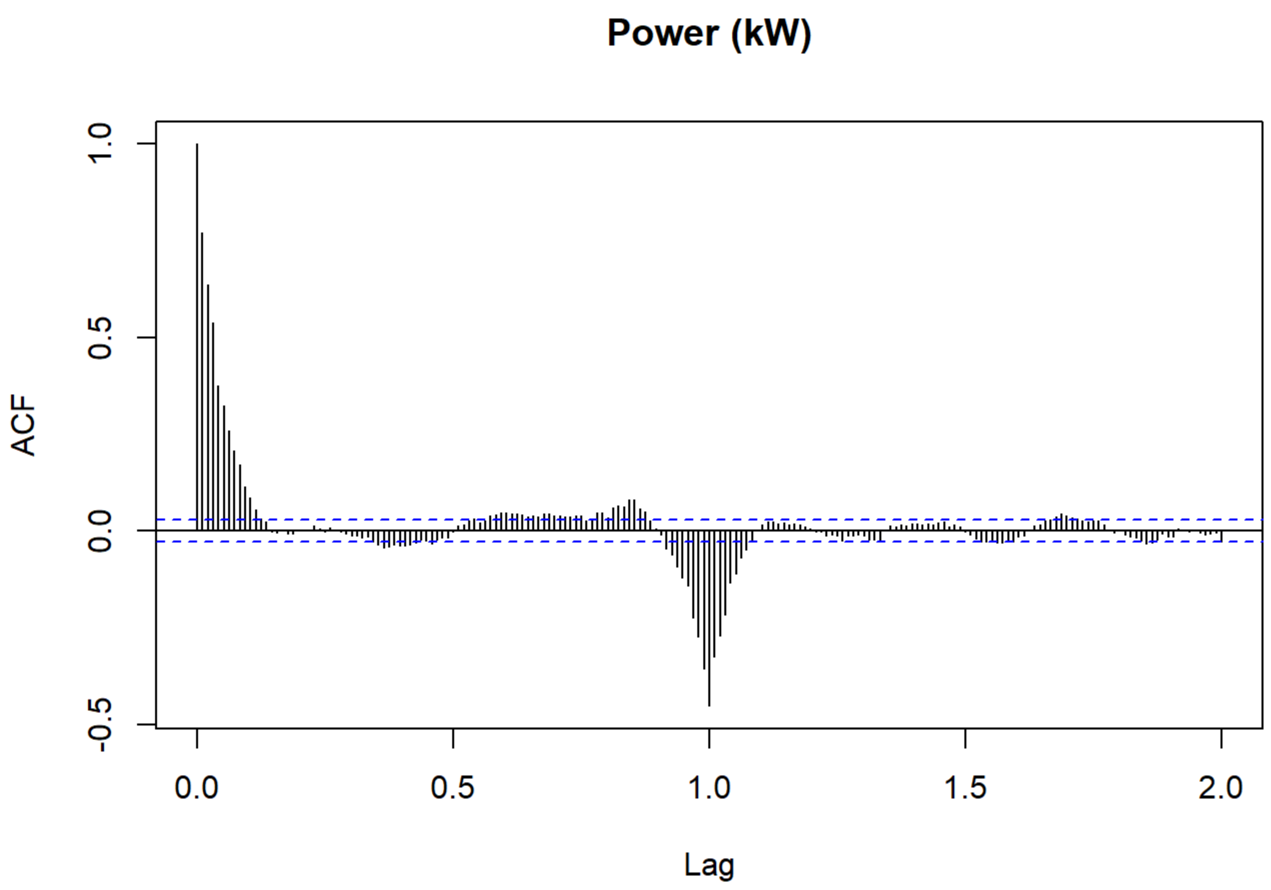
```
ggseasonplot((ts_Power_train %>% diff(lag=96)))
```



```
acf(ts_Power_train %>% diff(lag=96), lag.max=frequency)
```

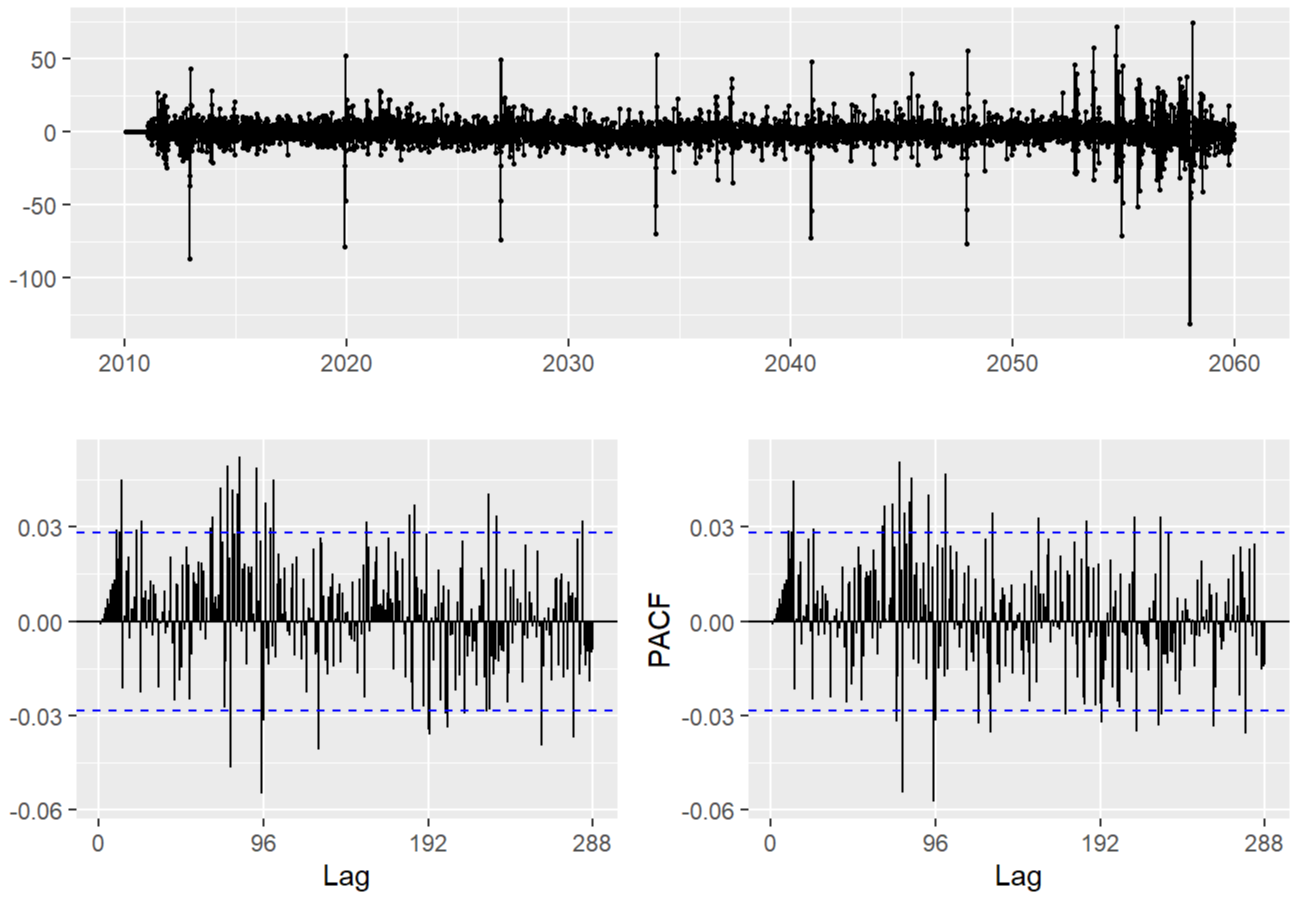


```
acf(ts_Power_train %>% diff(lag=96), lag.max=frequency*2)
```



Let’s try a SARIMA (0,0,12) (0,1,1) then, as we can also see significant autocorrelations at the beginning of the ACF (MA12).

```
#Warning : takes time to run due to the model complexity.
model_arima_man=Arima(ts_Power_train, order=c(0,0,12), seasonal=c(0,1,1))
model_arima_man %>% residuals() %>% ggtsdisplay()
```



```
summary(model_arima_man)
```

```
## Series: ts_Power_train
## ARIMA(0,0,12)(0,1,1)[96]
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5     ma6     ma7     ma8     ma9
##       0.7149  0.5774  0.6269  0.3607  0.3212  0.2833  0.2144  0.2178  0.1515
## s.e.  0.0147  0.0179  0.0197  0.0215  0.0216  0.0213  0.0208  0.0206  0.0206
##         ma10    ma11    ma12     sma1
##       0.0956  0.0564  0.0187  -0.8735
## s.e.  0.0192  0.0172  0.0147   0.0078
##
## sigma^2 = 75.24:  log likelihood = -16882.1
## AIC=33792.19   AICc=33792.28   BIC=33882.56
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.413812 8.574778 5.132866 -0.9619653 3.014349 0.5484672
##                       ACF1
## Training set -0.0009234727
```

ma12 coeff does not seem to be significant. ma11, sma1 yes.

We still get several significant autocorrelations and a pic at 96. 96 looks very high to try to add a MA96 or a AR96 (model complexity/running time).

Let’s look at the RMSE

```
prev_SARIMA_man=forecast(model_arima_man,h=frequency)
cat('SARIMA manual :',sqrt(mean((prev_SARIMA_man$mean-ts_Power_test)^2)), '\n')
```

```
## SARIMA manual : 12.63427
```

We get 12.63427 which is better than SARIMA auto. However we coul dcross validate to check the reliability.

We could also look at Cox-Box (lambda=“auto”) as the series seems to have a slightl higher amplitude at the end but we will stop there to look at other models.

Neurone Network

```
model_nnetar=nnetar(ts_Power_train, lambda="auto")
print(model_nnetar)
```

```
## Series: ts_Power_train
## Model:  NNAR(25,1,14)[96]
## Call:   nnetar(y = ts_Power_train, lambda = "auto")
##
## Average of 20 networks, each of which is
## a 26-14-1 network with 393 weights
## options were - linear output units
##
## sigma^2 estimated as 249057
```

25 lagged values 1 lagged values from the same season 1 hidden layer with 14 neurons

Let’s having an overview with all previous models.

```
model_hw <- HoltWinters(ts_Power_train,seasonal="additive")

plot(ts_Power_test, lty=2)

lines(prev_hw, col=2, lty=2)
cat('HW :',sqrt(mean((prev_hw-ts_Power_test)^2)), '\n')
```

```
## HW : 11.64876
```

```
prev_SARIMA_auto=forecast(model_arima_auto,h=frequency)
lines(prev_SARIMA_auto$mean, col=3, lty=2)
cat('SARIMA auto :',sqrt(mean((prev_SARIMA_auto$mean-ts_Power_test)^2)), '\n')
```

```
## SARIMA auto : 5.862372
```

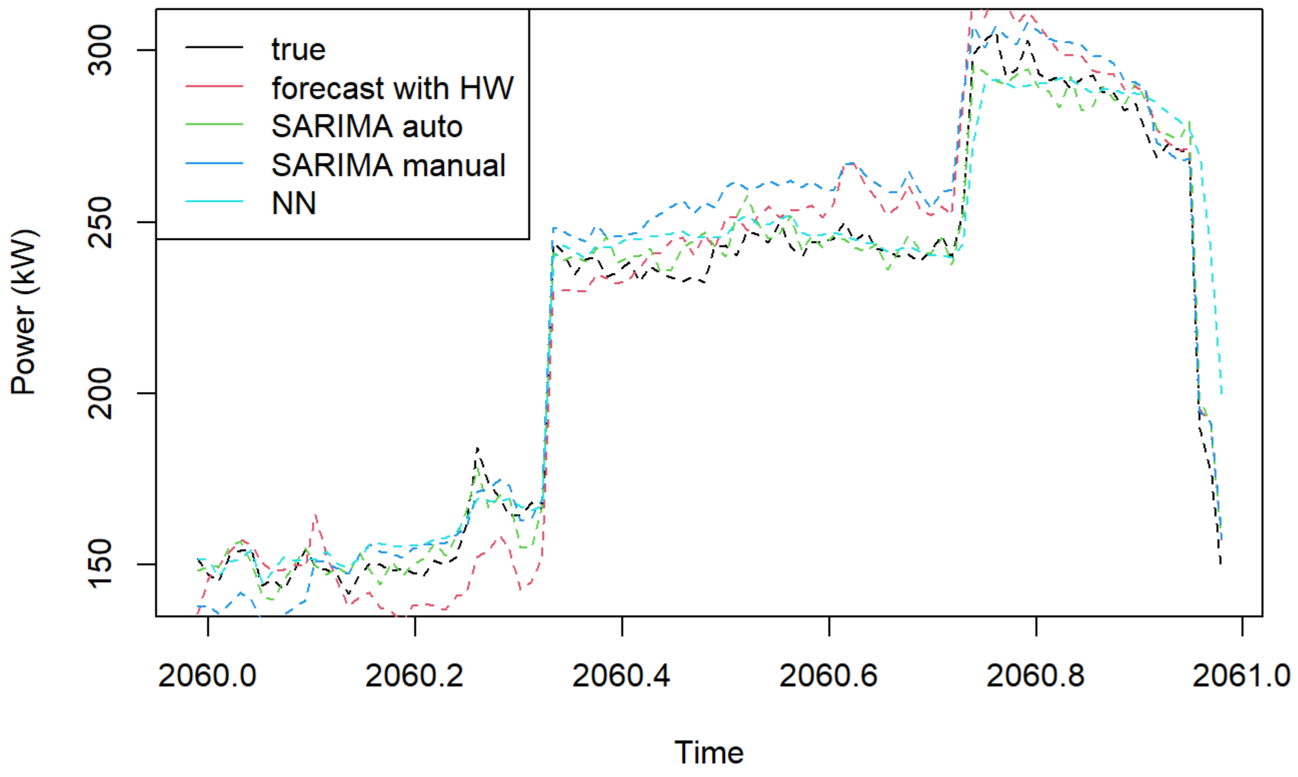
```
prev_SARIMA_man=forecast(model_arima_man,h=frequency)
lines(prev_SARIMA_man$mean, col=4, lty=2)
cat('SARIMA manual :',sqrt(mean((prev_SARIMA_man$mean-ts_Power_test)^2)), '\n')
```

```
## SARIMA manual : 12.63427
```

```
prev_NN=forecast(model_nnetar,h=frequency)
lines(prev_NN$mean, col=5, lty=2)
cat('NN :',sqrt(mean((prev_NN$mean-ts_Power_test)^2)), '\n')
```

```
## NN : 13.59094
```

```
legend('topleft', co=1:5,lty=1, legend=c('true', 'forecast with HW', 'SARIMA auto', 'SARIMA m
anual', 'NN'))
```



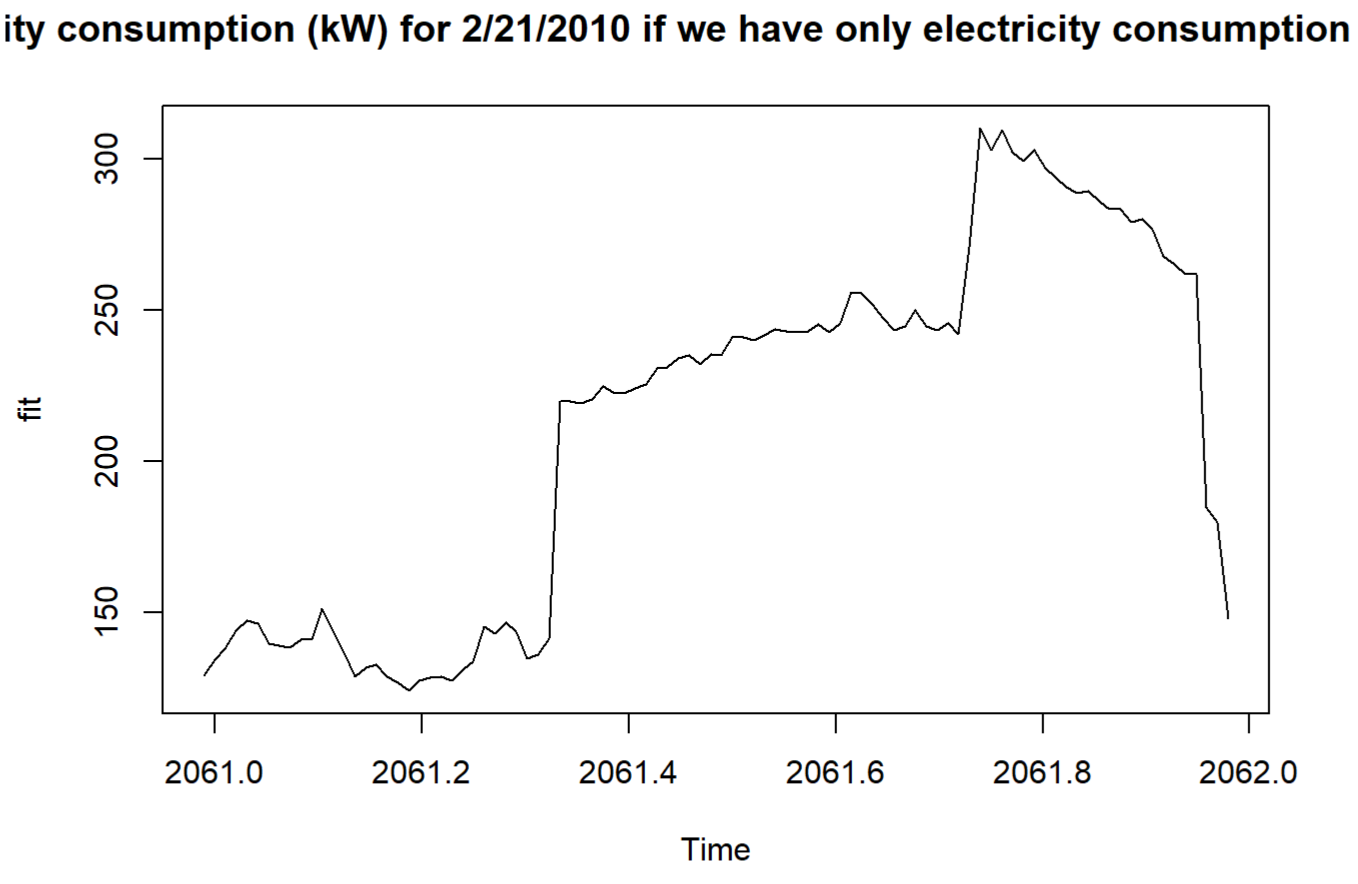
NN has an higher RMSE and does not perform better than the others. The highest pic is also a bit delayed. Indeed we did not introduce any parametric modeling of the periodicity. We did it in HW for example. In NN we just made the link to the “near” covariates.

SARIMA auto gets the lowest RMSE and is the closest to the true. However after cross validation additive HW was better.

Saying that, we should also cross validate the ones we did not (SARIMA manual & NN). It would take a long time due to the compexity of the models but the results will be more reliable.

To end this first part : if we had only the Power consomption time series, we would have taken the best model (HW additive after cross validation here) and applied it to the all time series for the prediction.

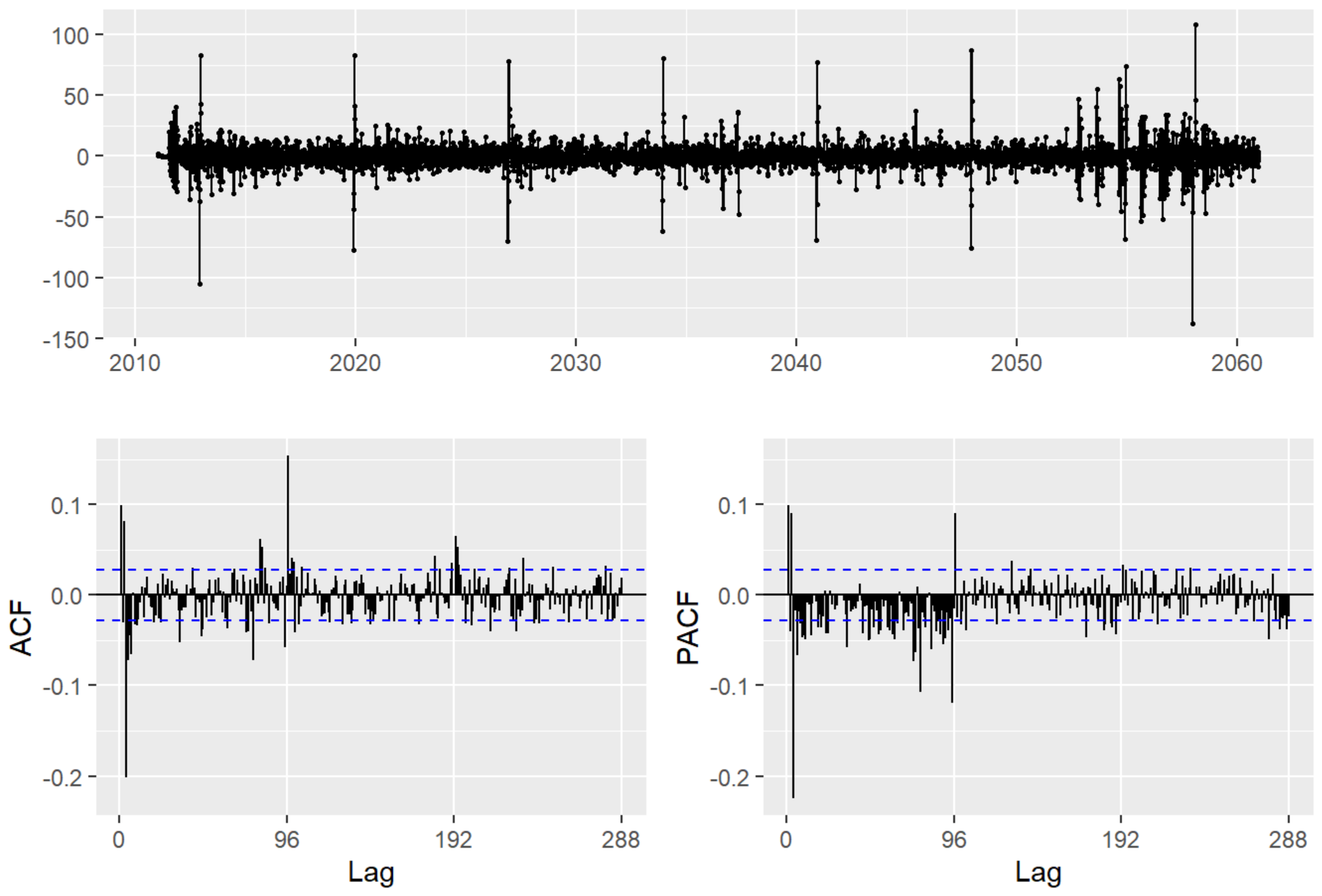
```
#summary(model_hw)
#summary(model_arima_auto)
#summary(prev_SARIMA_man)
#summary(model_nnetar)

model_hw_1TS_final <- HoltWinters(ts_Power,seasonal="additive")
prev_hw_1TS_final<-predict(model_hw_1TS_final,n.ahead=frequency)
plot(prev_hw_1TS_final, main = "electricity consumption (kW) for 2/21/2010 if we have only el
ectricity consumption time series")
```



```
summary(model_hw_1TS_final)
```

```
##              Length Class  Mode
## fitted       19180  mts    numeric
## x             4891  ts     numeric
## alpha            1  -none- numeric
## beta             1  -none- numeric
## gamma            1  -none- numeric
## coefficients    98  -none- numeric
## seasonal         1  -none- character
## SSE              1  -none- numeric
## call             3  -none- call
```

```
model_hw_1TS_final %>% residuals() %>% ggtsdisplay()
```

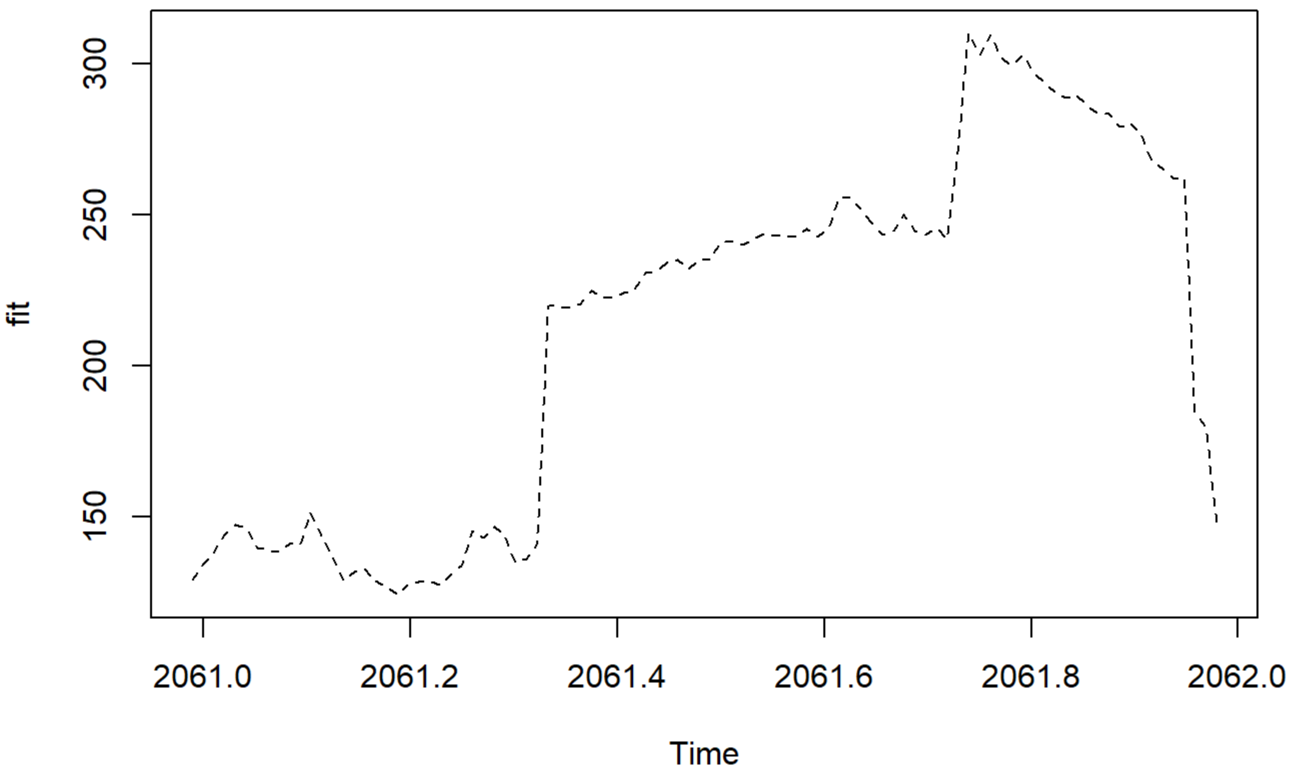


We could hope to get a better situation by taking into account the covariate (outdoor temperature)

Forecast with the best model without using covariates (model HW additive)

```
#Fit the model on all the Power data time series

model_hw_without_cov_final <- HoltWinters(ts_Power,seasonal="additive")
```

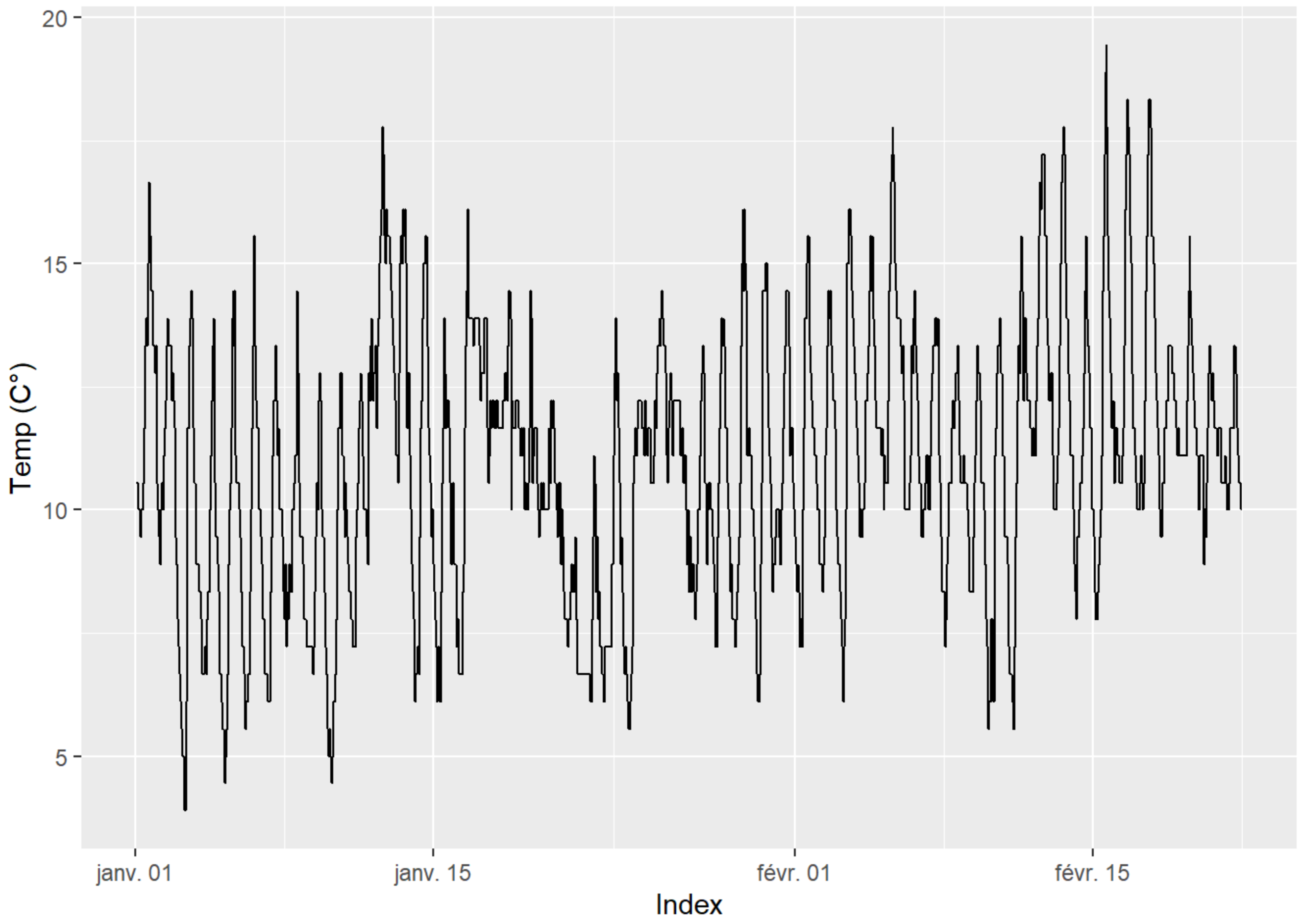
```
prev_hw_without_cov_final<-predict(model_hw_without_cov_final,n.ahead=frequency)
plot(prev_hw_without_cov_final, lty=2)
```



Forecast electricity consumption (kW) for 2/21/2010 by using electricity consumption and outdoor temperature time series.

Let’s prepare and have a look at the electricity Time Series.

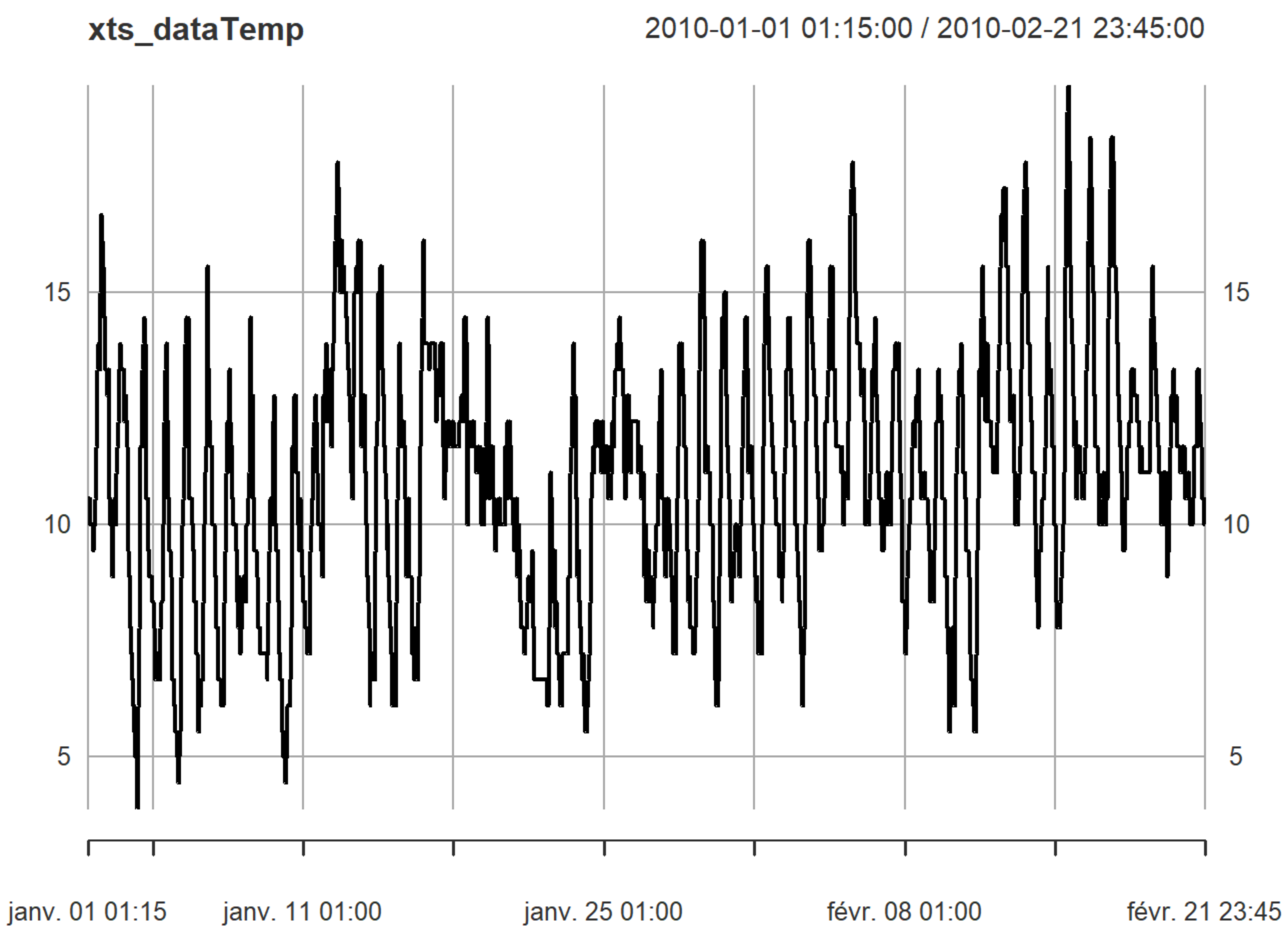
```
observationsTemp <- observations[1:(nrow(observations)),2]
end_timeTemp <- end_time
end_timeTemp_valid = end_Power_test
start_timeTemp_trainValid = start_time
end_timeTemp_trainValid = end_timePower
end_timeTemp_train = end_timePower_train
start_timeTemp_valid = start_timePower_test
end_timeTemp_valid = end_timePower

observationsPower <- observations[1:(nrow(observations) - 96),1]
end_timePower <- as.POSIXct("2010-02-20 23:45")
time_indexPower <- seq(from = start_time, to = end_timePower, by = "15 min")

time_indexTemp <- seq(from = start_time, to = end_timeTemp, by = "15 min")
xts_dataTemp <- xts(observationsTemp, order.by = time_indexTemp)
head(xts_dataTemp)
#Plot the TS
autoplot(xts_dataTemp)
```



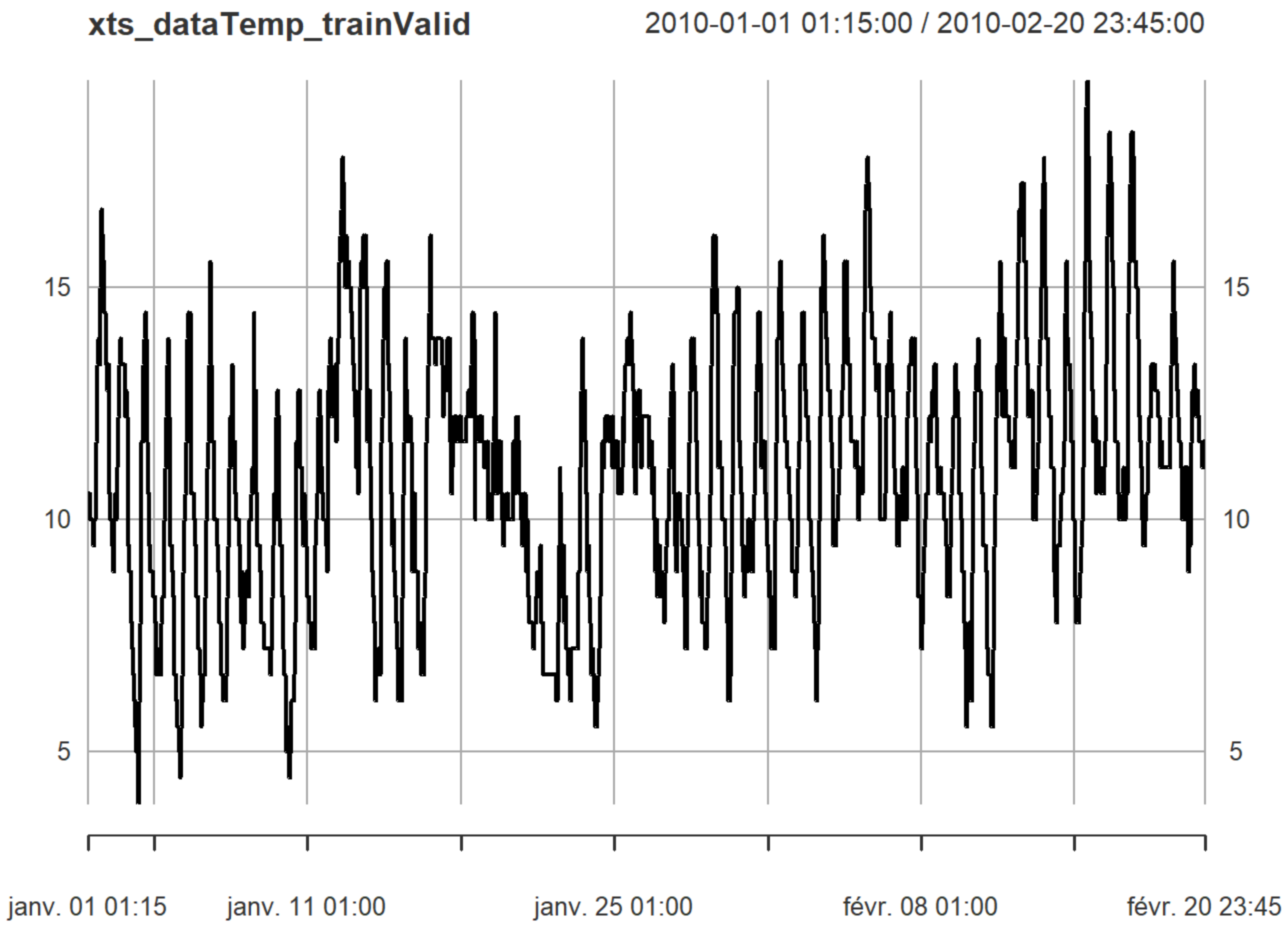
Split the Temperature time series in 4 parts: -trainValid -train -valid -test

```
observationsTemp_trainValid <- observations[1:(nrow(observations) - frequency),2]
observationsTemp_valid <- observations[(nrow(observations) - frequency*2 + 1):(nrow(observati
ons)- frequency),2]
observationsTemp_train <- observations[1:(nrow(observations) - frequency*2),2]

observationsTemp_test <- observations[(nrow(observations) - frequency + 1):(nrow(observation
s)),2]
start_timeTemp_test = end_timeTemp + (- 96 + 1)*60*15

time_indexTemp_trainValid <- seq(from = start_time, to = end_timeTemp_trainValid, by = "15 mi
n")
xts_dataTemp_trainValid <- xts(observationsTemp_trainValid, order.by = time_indexTemp_trainVa
lid)

time_indexTemp_valid <- seq(from = start_timeTemp_valid, to = end_timeTemp_valid, by = "15 mi
n")
xts_dataTemp_valid <- xts(observationsTemp_valid, order.by = time_indexTemp_valid)

time_indexTemp_train <- seq(from = start_time, to = end_timeTemp_train, by = "15 min")
xts_dataTemp_train <- xts(observationsTemp_train, order.by = time_indexTemp_train)

time_indexTemp_test <- seq(from = start_timeTemp_test, to = end_timeTemp, by = "15 min")
xts_dataTemp_test <- xts(observationsTemp_test, order.by = time_indexTemp_test)
```

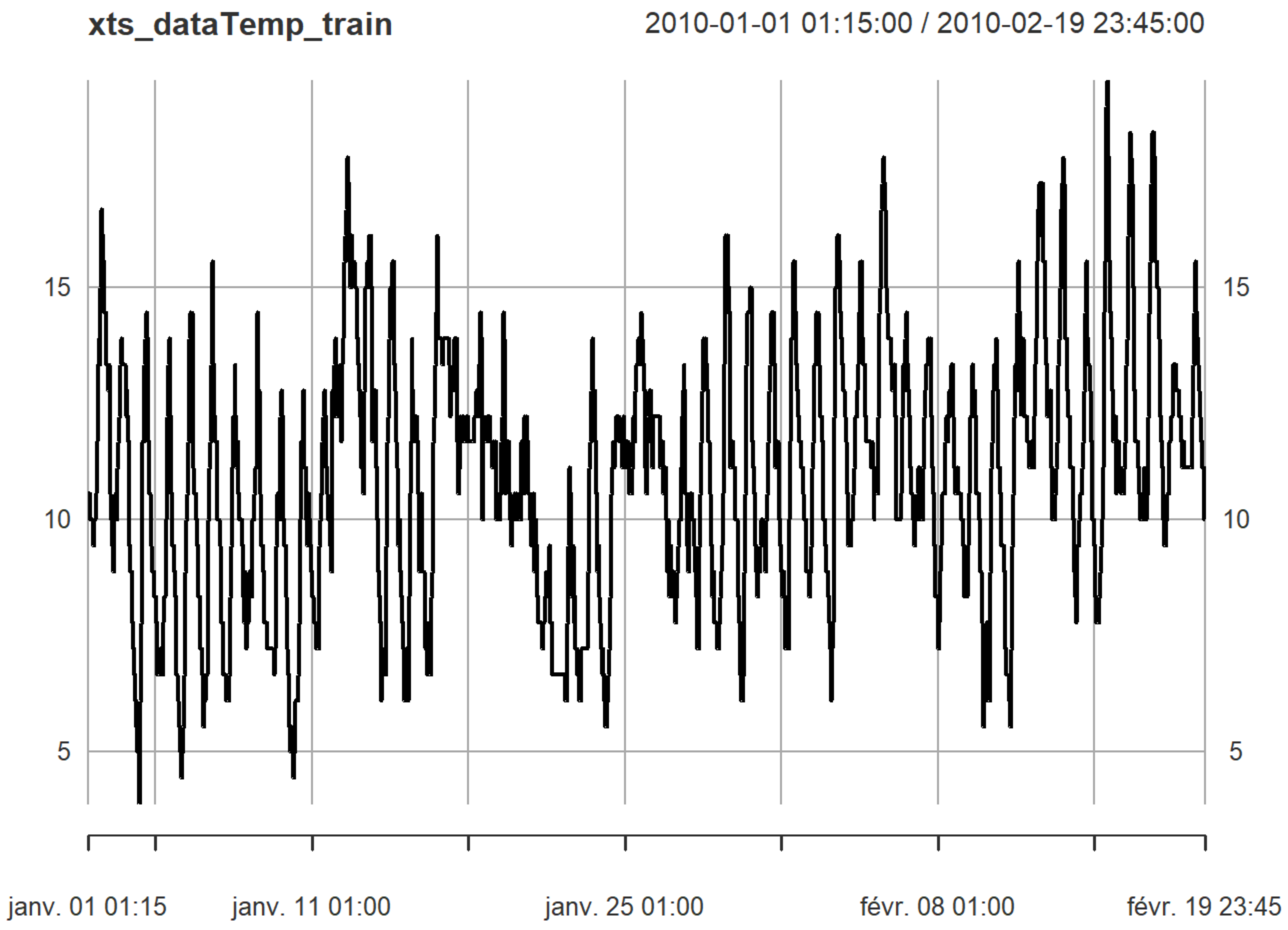
```
#Plot the TS
plot(xts_dataTemp)
```



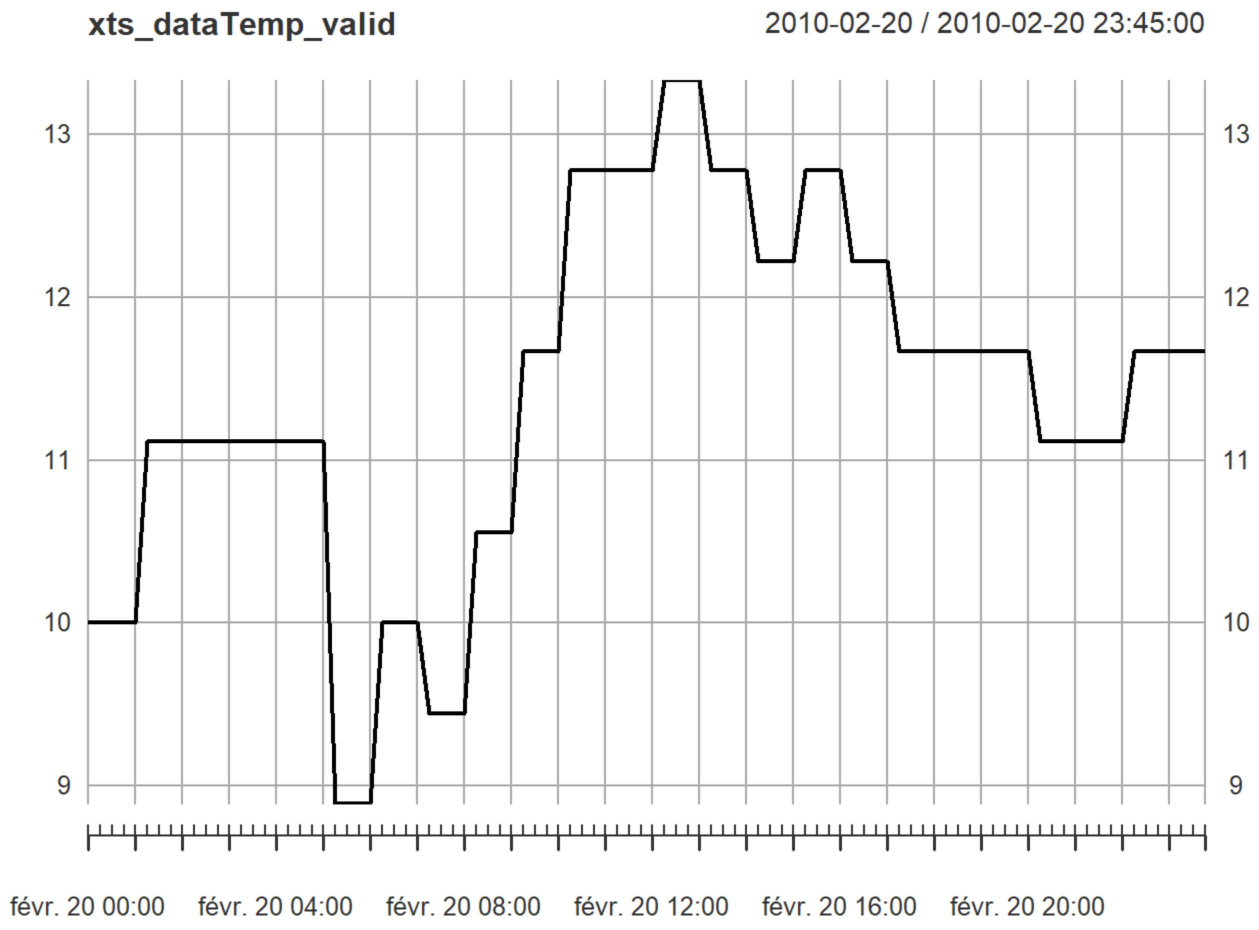
```
plot(xts_dataTemp_trainValid)
```



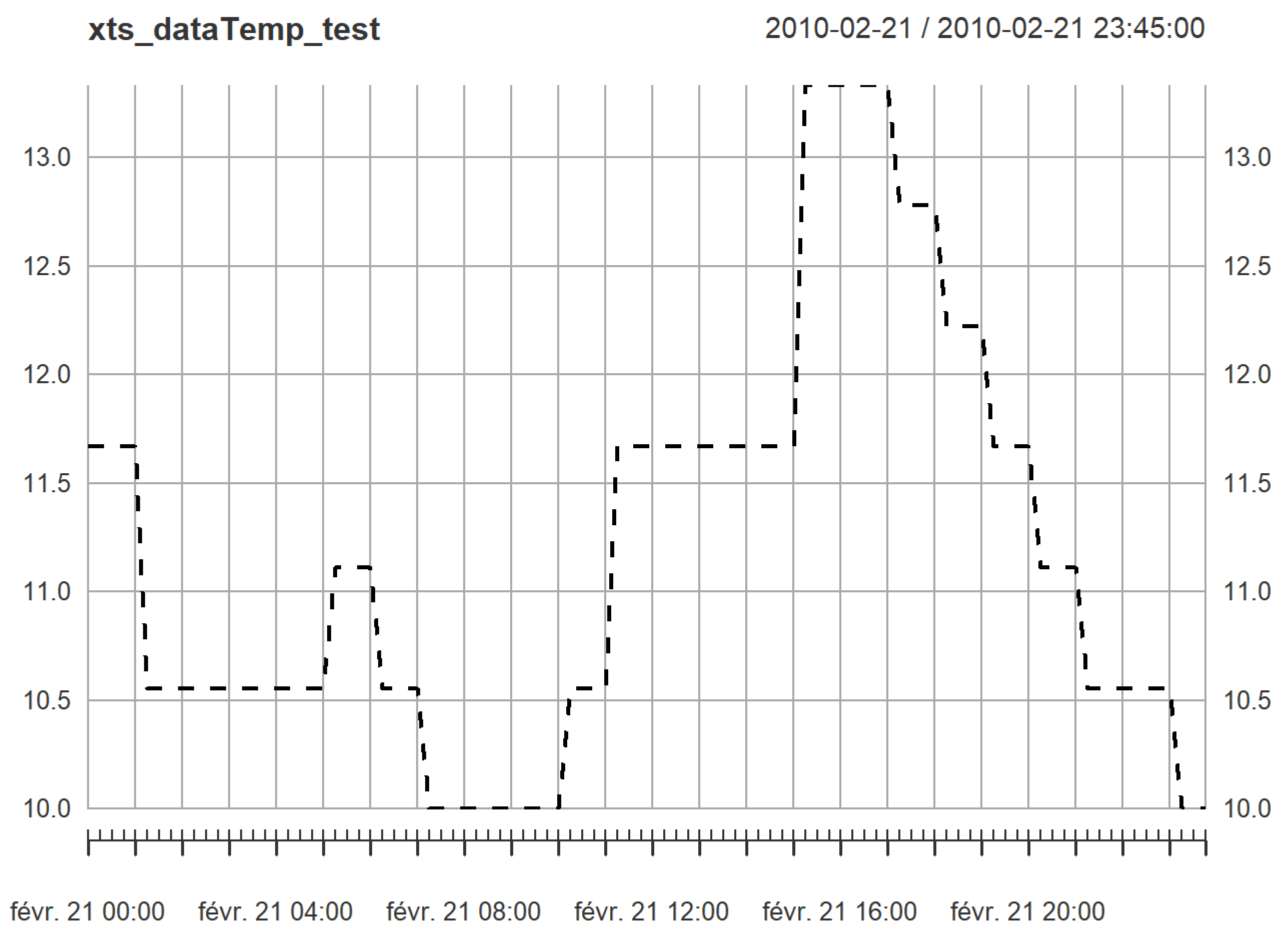
```
plot(xts_dataTemp_train)
```



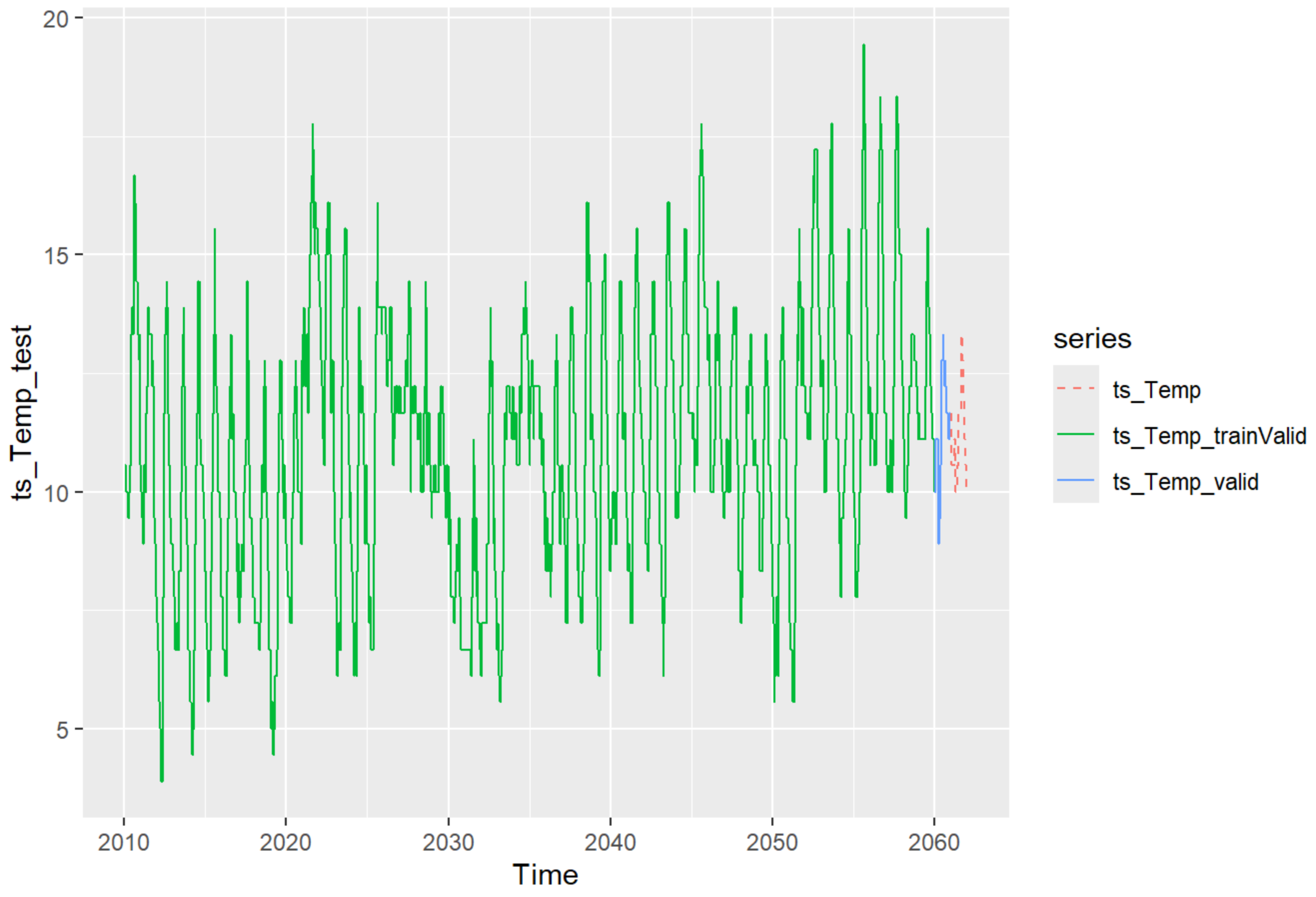
```
plot(xts_dataTemp_valid)
```



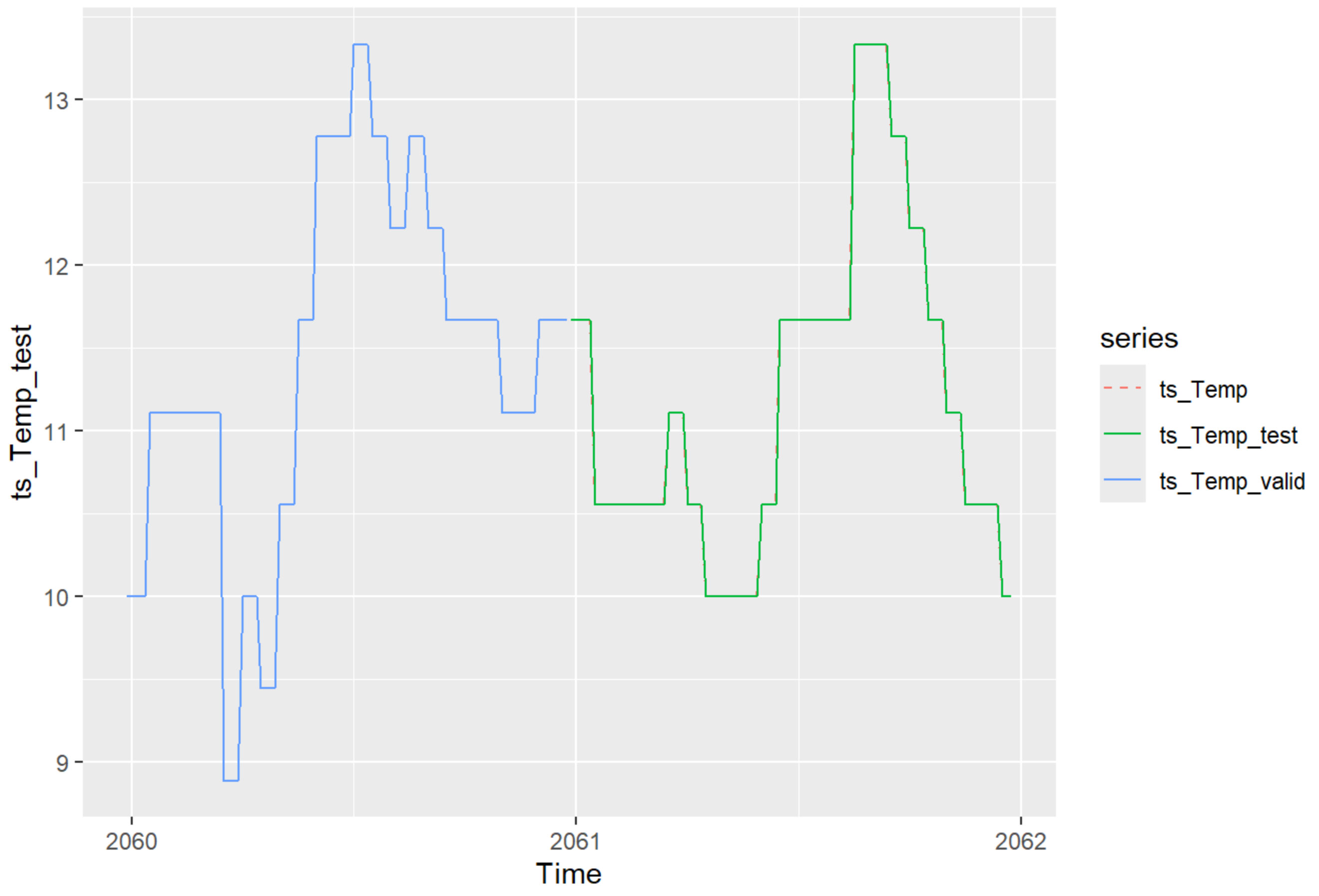
```
plot(xts_dataTemp_test,lty=2)
```



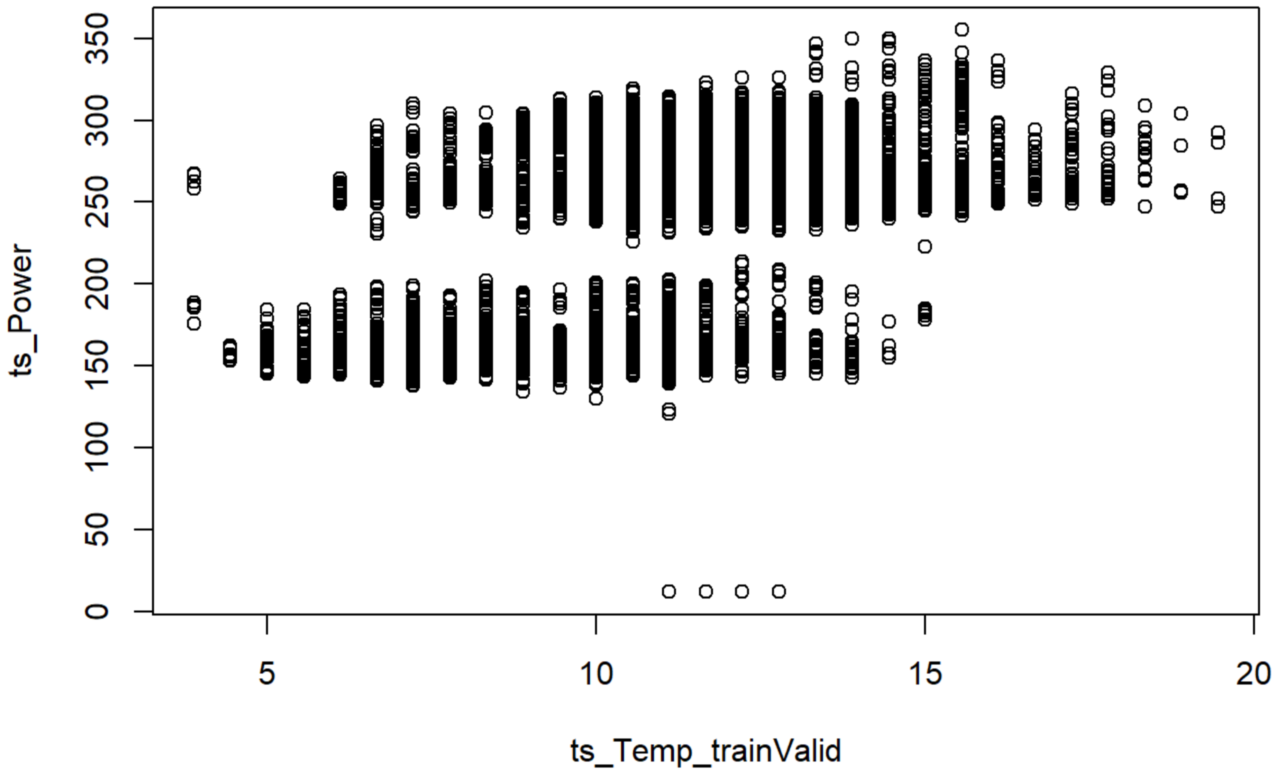
```
ts_Temp=ts(coredata(xts_dataTemp),start=c(2010,5),frequency=frequency)

ts_Temp_trainValid=ts(coredata(xts_dataTemp_trainValid),start=c(2010,5),frequency=frequency)
ts_Temp_train=ts(coredata(xts_dataTemp_train),start=c(2010,5),frequency=frequency)
ts_Temp_valid=ts(coredata(xts_dataTemp_valid),start=c(2010,5+(nrow(observations)- frequency*
2)),frequency=frequency)
ts_Temp_test=ts(coredata(xts_dataTemp_test),start=c(2010,5+(nrow(observations)- frequency)),f
requency=frequency)

autoplot (ts_Temp_test, series = "ts_Temp", lty=2)+
  autolayer (ts_Temp_trainValid, series = "ts_Temp_trainValid")+
  autolayer (ts_Temp_valid, series = "ts_Temp_valid")
```



```
autoplot (ts_Temp_test, series = "ts_Temp", lty=2)+
  autolayer (ts_Temp_valid, series = "ts_Temp_valid")+
  autolayer (ts_Temp_test, series = "ts_Temp_test")
```



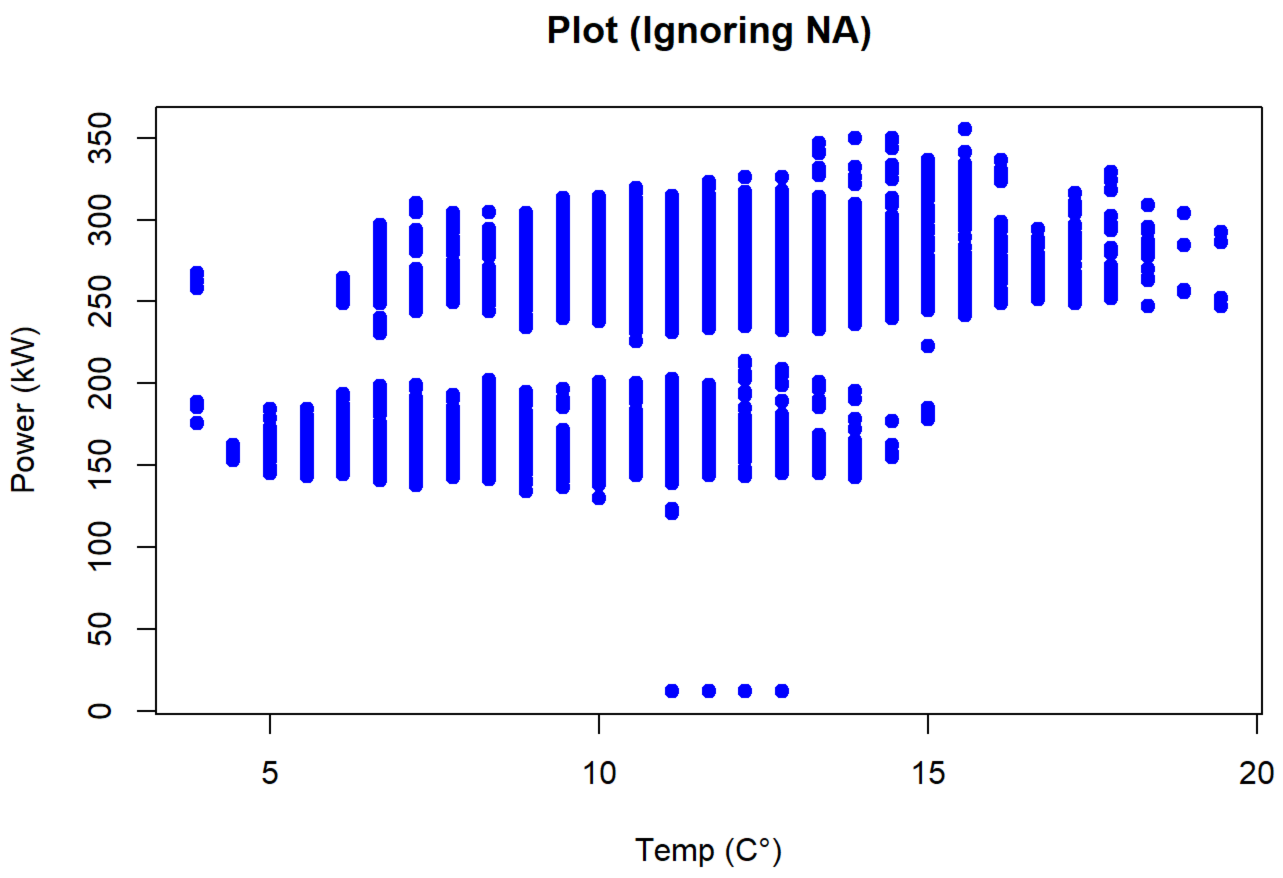
```
plot(ts_Temp_trainValid,ts_Power)
```



By looking at both time series on the same time window, we can see that we may have some dependencies. For exemple, at the highest temperature : the electricity consumption (kW) are the highest.

There are some outliers which should correspond to the 0 Power (maybe an exeptionnal cut in the building) that we updated with a low non zero value.

```
#Plot without the NA
df <- data.frame(
  x = data$"Temp (C°)",
  y = data$"Power (kW)")

df_clean <- df[complete.cases(df), ]

plot(df_clean$x, df_clean$y, main = "Plot (Ignoring NA)", xlab = "Temp (C°)", ylab = "Power
(kW)", pch = 19, col = "blue")
```



We have already studied the forecast without the covariate. Forecasting with covariates.

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

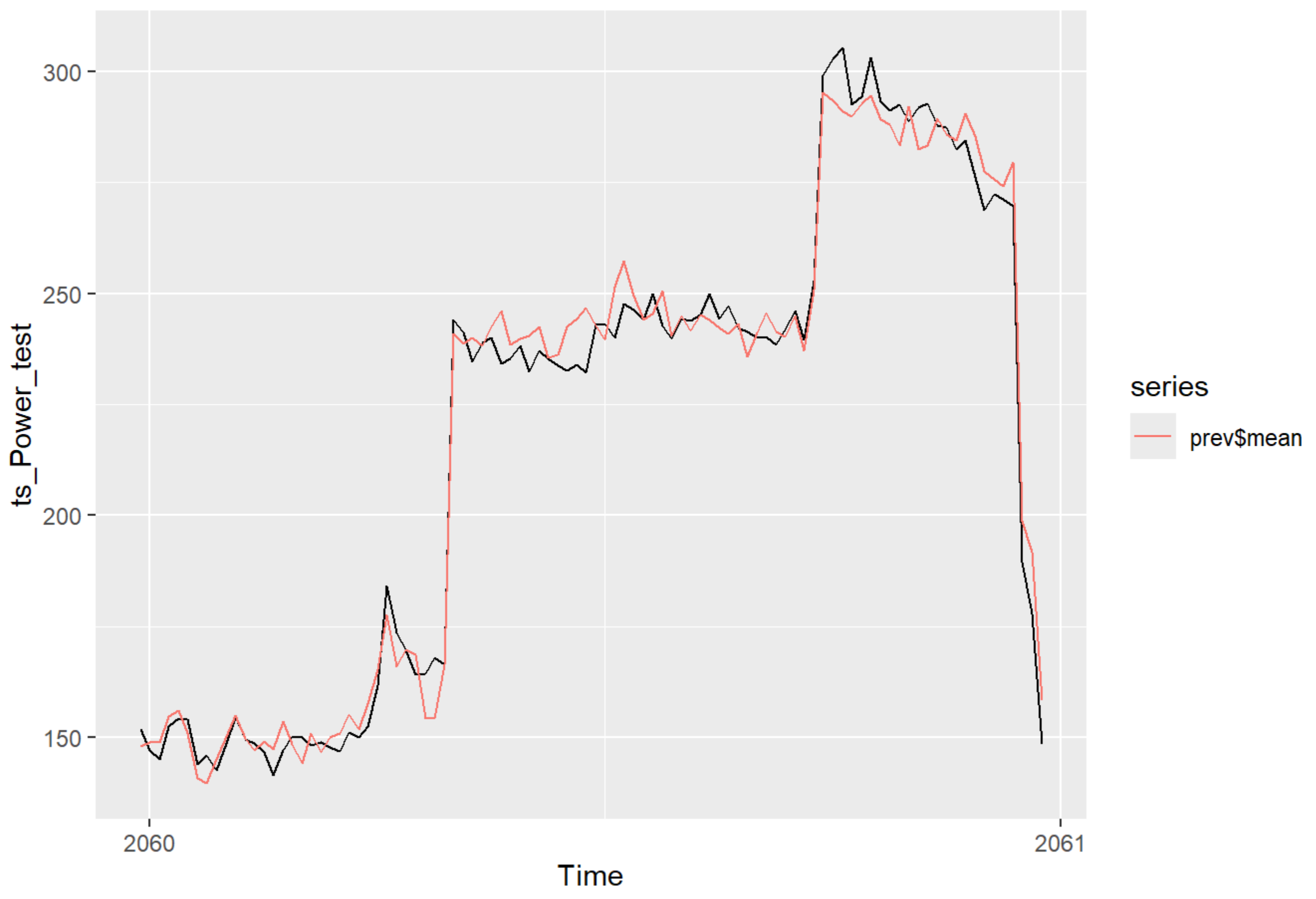
```
model_arima_auto_cov = auto.arima (ts_Power_train, xreg= ts_Temp_train)
summary(model_arima_auto_cov)
```

```
## Series: ts_Power_train
## Regression with ARIMA(5,0,1)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ma1  Temp (C°)
##       0.4135  0.2745  0.1895  -0.2222  0.0688  0.3074     0.4121
## s.e.  0.1261  0.0882  0.0194   0.0296  0.0321  0.1265     0.2635
##
## sigma^2 = 136.8:  log likelihood = -18220.89
## AIC=36457.79   AICc=36457.82   BIC=36509.43
##
## Training set error measures:
##                       ME    RMSE      MAE        MPE     MAPE      MASE
## Training set -0.08623878 11.5703 6.665372 -0.8360915 3.760102 0.7122215
##                       ACF1
## Training set -0.0004989493
```

We get the same model as without covariate : ARIMA(5,0,1)(0,1,0)[96]

ar5 coeff is slightly significant. ma1 is significant.

```
prev=forecast(model_arima_auto_cov, h=frequency, xreg= ts_Temp_valid)
autoplot(ts_Power_test)+autolayer(prev$mean)
```



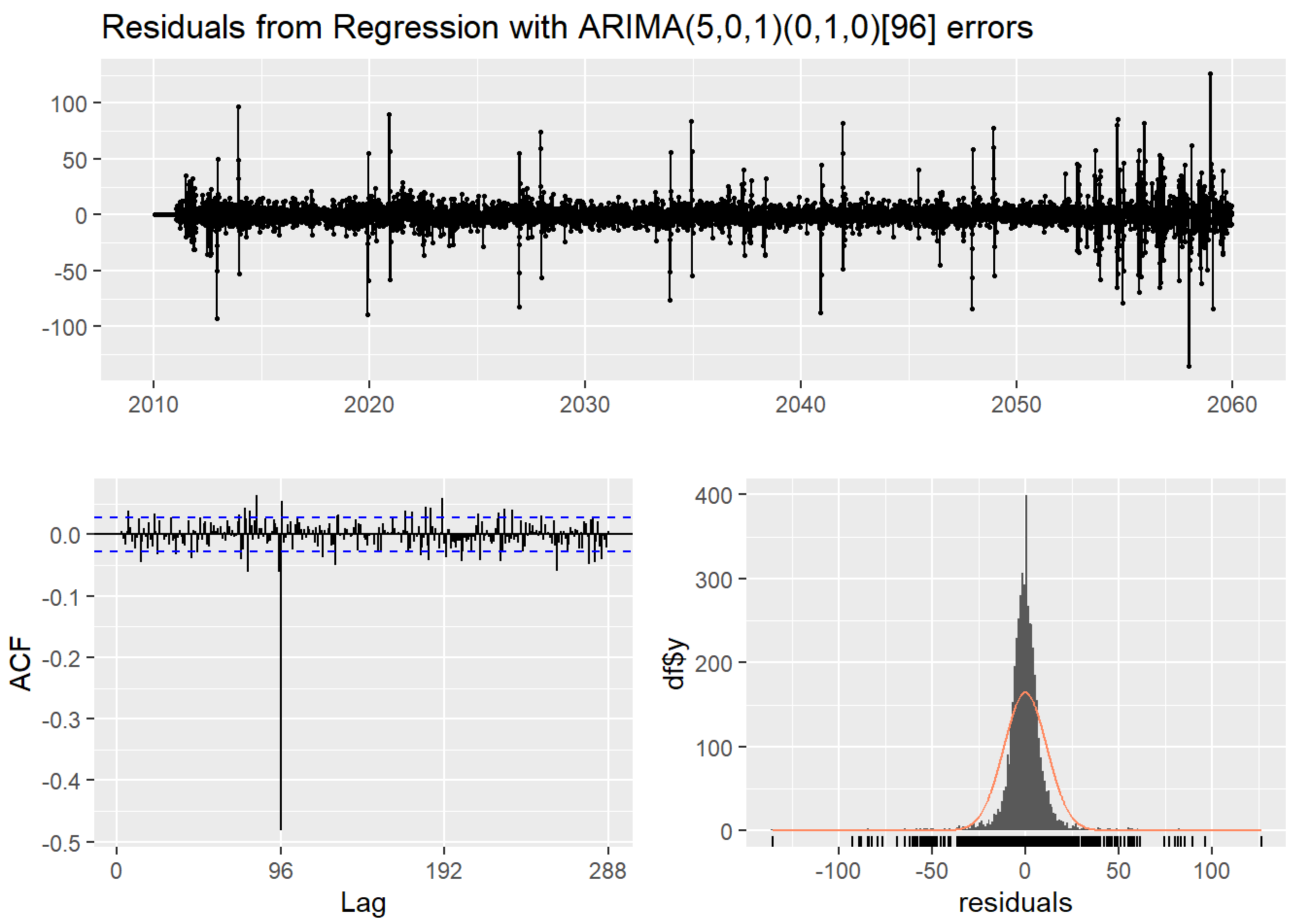
```
cat('SARIMA auto covariates :',sqrt(mean((prev$mean-ts_Power_test)^2)), '\n')
```

```
## SARIMA auto covariates : 5.908364
```

SARIMA auto : 5.908364

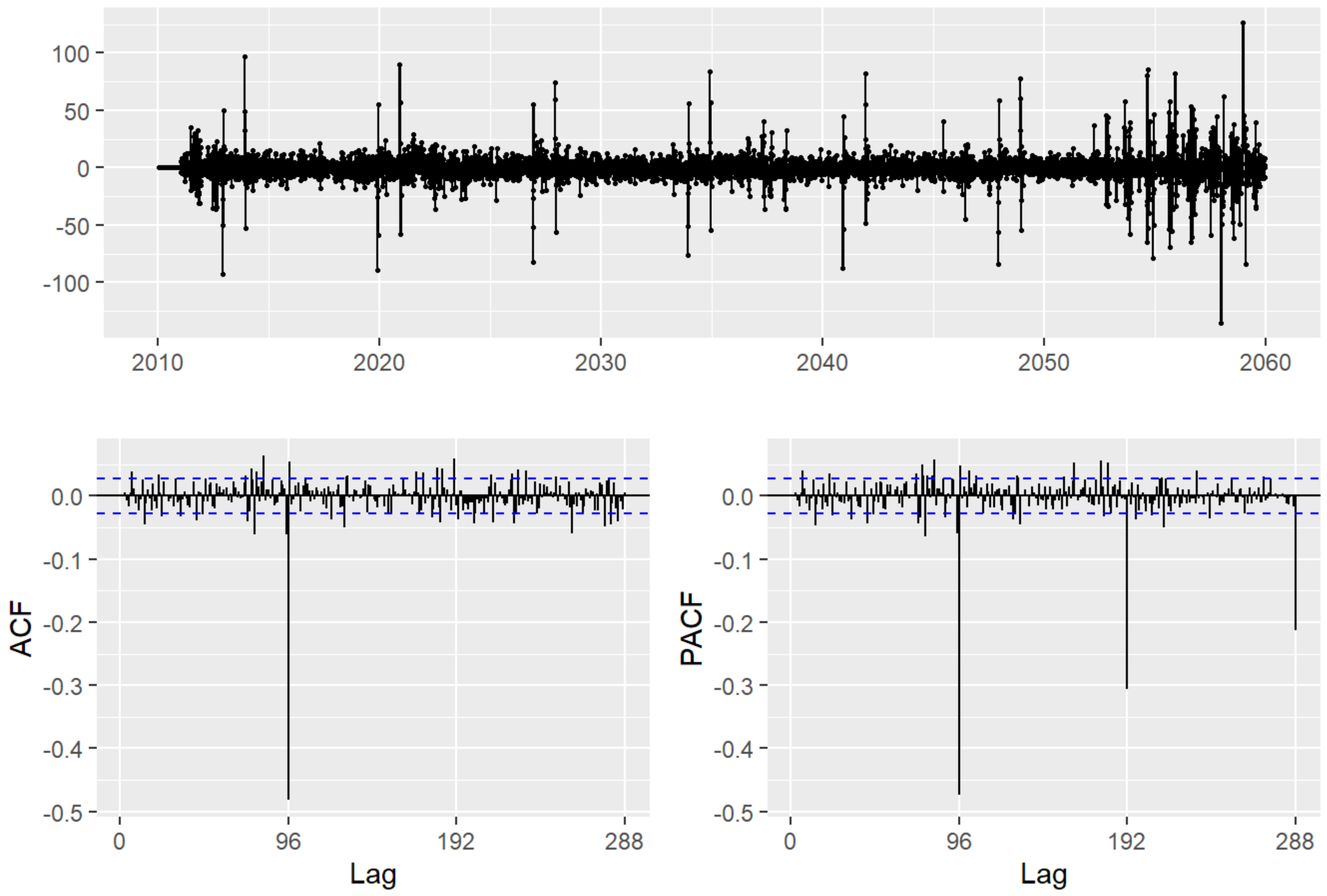
Both curves are close. We get the same model(ARIMA(5,0,1)(0,1,0)[96]) as without taking into account the covariate but let’s look at the residuals.

```
model_arima_auto_cov %>% checkresiduals()
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,1)(0,1,0)[96] errors
## Q* = 1528.3, df = 186, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 192
```

```
model_arima_auto_cov %>% residuals() %>% ggtsdisplay()
```



There is still some autocorrelation. We could try by addind a MA96 but the model would become very complex.

Let’s see if we can get better result manually.

Covariate and manual ARIMA

We have only one covariate. Let’s look at the accuracy of taking into account trend and season as well.

```
merged_ts <- cbind(ts_Power_train,ts_Temp_train)
```

```
model_TSLM_cov = tslm(ts_Power_train ~ ts_Temp_train,data=merged_ts)
summary(model_TSLM_cov)
```

```
##
## Call:
## tslm(formula = ts_Power_train ~ ts_Temp_train, data = merged_ts)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -237.614  -41.458    2.336   43.120  116.223
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   120.9593     3.0946   39.09   <2e-16 ***
## ts_Temp_train  10.0687     0.2747   36.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.68 on 4793 degrees of freedom
## Multiple R-squared:  0.2189, Adjusted R-squared:  0.2187
## F-statistic:  1343 on 1 and 4793 DF,  p-value: < 2.2e-16
```

```
model_TSLM_cov_trendSeason = tslm(ts_Power_train ~ ts_Temp_train+trend+season,data=merged_ts)
summary(model_TSLM_cov_trendSeason)
```

Intercept, temperature but indeed also trend and season (especially from 22 to 95) are significant. Around 22-23 would correspond to the start of the “living day” for people (23*15/60) ~ 5,75 hours after 1h15 ~ 6 h at the morning. We the have increase of the power consumption. Similar reasonning ends up with lower consumption during the night (coeff become negatifs… but they are not significant).

We have positive significant impact of the outdoor temperature on the power consumption.

Negative significant coeff on the intercept. Maybe due to a slighlt decrease that we could see on the general graph.

```
CV(model_TSLM_cov)
```

```
##           CV          AIC         AICc          BIC        AdjR2
## 2.672012e+03 3.783807e+04 3.783807e+04 3.785749e+04 2.187409e-01
```

```
CV(model_TSLM_cov_trendSeason)
```

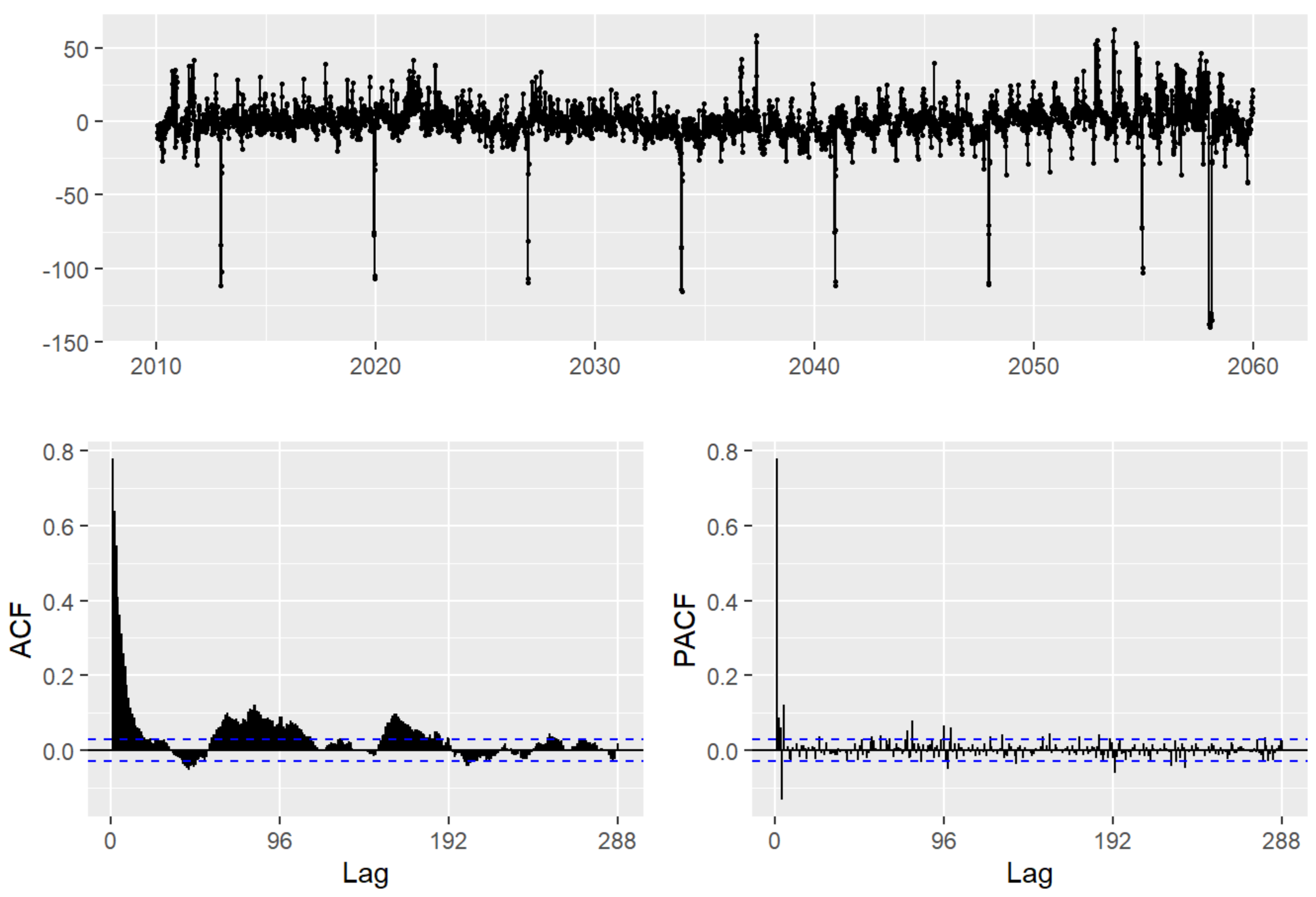
```
##           CV          AIC         AICc          BIC        AdjR2
## 2.031670e+02 2.547995e+04 2.548417e+04 2.612101e+04 9.418046e-01
```

The model with trend and season is way more better (better AdjR2 and lower AIC, AICc, BIC)

```
checkresiduals(model_TSLM_cov_trendSeason,test='LB',plot=FALSE)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 11831, df = 192, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 192
```

```
model_TSLM_cov_trendSeason %>% residuals() %>% ggtsdisplay()
```



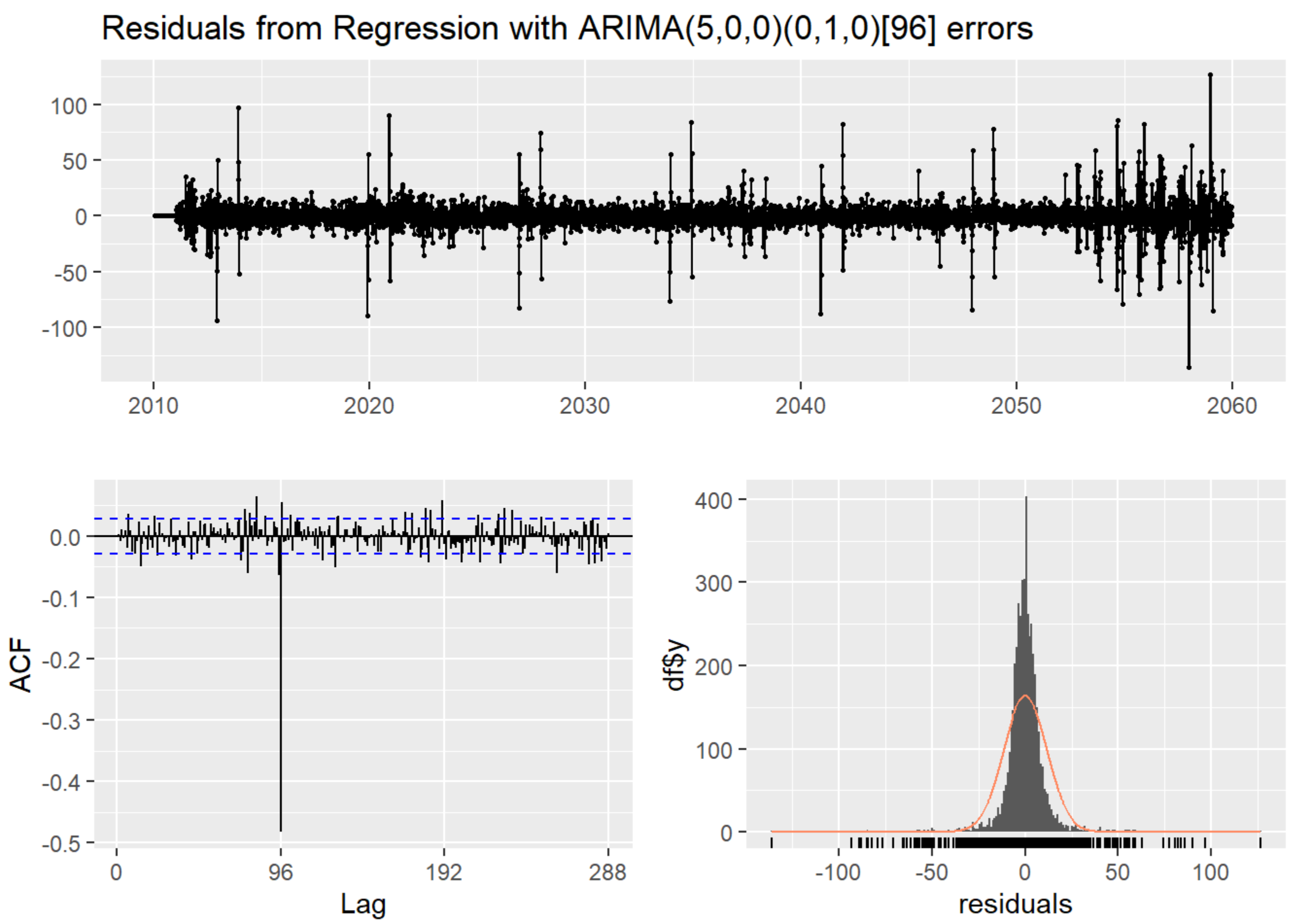
Here the residual are correlated, which means that this regression model (which assumes independent residuals) is not appropriated.

We have significant autocorelations o, ACF and PACF. Let’s chose the simpler one by using the PACF (looks like AR5).

Let’s try an ARIMA (5,0,0) (0,1,0) with covariates. The auto ARIMA had found something close (5,0,1) (0,1,0). The second 1 should get a rid of the season and the trend.

```
model_arima_man_cov=Arima(ts_Power_train, xreg= ts_Temp_train, order=c(5,0,0), seasonal = c
(0,1,0))
summary(model_arima_man_cov)
```

```
checkresiduals(model_arima_man_cov)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,0)[96] errors
## Q* = 1545.2, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

```
prev_arima_man_cov=forecast(model_arima_man_cov,h=frequency, xreg= ts_Temp_valid)
cat('SARIMA manual cov :',sqrt(mean((prev_arima_man_cov$mean-ts_Power_test)^2)), '\n')
```

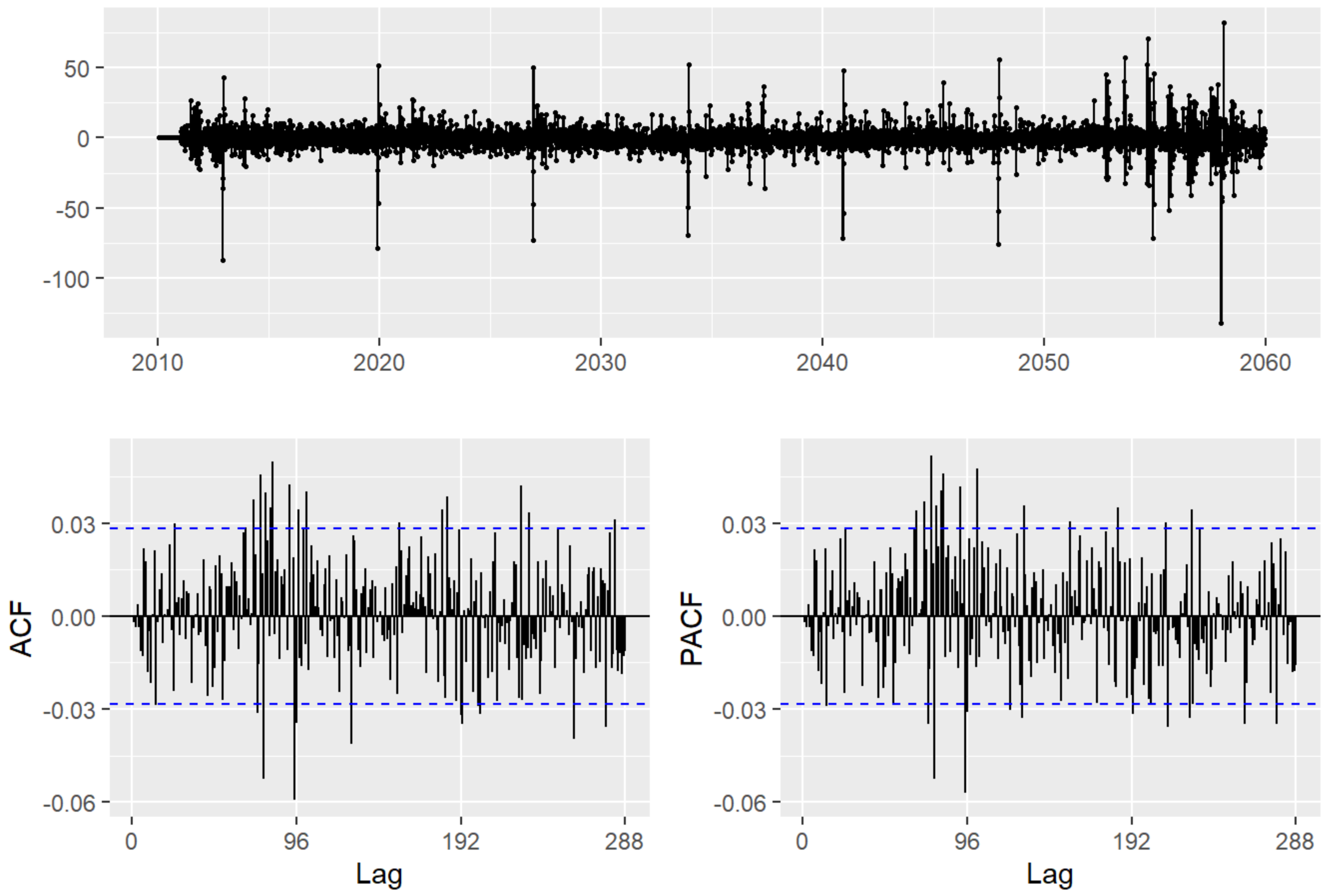
```
## SARIMA manual cov : 5.908758
```

We get SARIMA manual cov : 5.908758. We got SARIMA auto (with covariates) : 5.908364. However, we still have significant autocorelation, especially a SMA1 in the ACF. Let's try order=c(5,0,0), seasonal = c(0,1,1) then.

```
#Takes time to run
model_arima_man_cov=Arima(ts_Power_train, xreg= ts_Temp_train, order=c(5,0,0), seasonal = c
(0,1,1))
summary(model_arima_man_cov)
```

```
## Series: ts_Power_train
## Regression with ARIMA(5,0,0)(0,1,1)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     sma1  Temp (C°)
##       0.7115  0.0689  0.1603  -0.2356  0.1138  -0.8756     0.4967
## s.e.  0.0145  0.0175  0.0174   0.0175  0.0145   0.0078     0.2463
##
## sigma^2 = 74.84:  log likelihood = -16873.55
## AIC=33763.09   AICc=33763.12   BIC=33814.73
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE    MASE
## Training set -0.3670674 8.557835 5.095271 -0.9260798 2.985143 0.54445
##                       ACF1
## Training set -0.002113169
```

```
model_arima_man_cov %>% residuals() %>% ggtsdisplay()
```



AIC, AICc and BIC are better. The RMSE is worst on the validation dataset but after cross validation it could be different. Some autocorelations are still significant and we stil have an SMA1 + a potentially a SAR1 now.

```
prev_arima_man_cov=forecast(model_arima_man_cov,h=frequency, xreg= ts_Temp_valid)
cat('SARIMA manual cov :',sqrt(mean((prev_arima_man_cov$mean-ts_Power_test)^2)), '\n')
```

```
## SARIMA manual cov : 12.15481
```

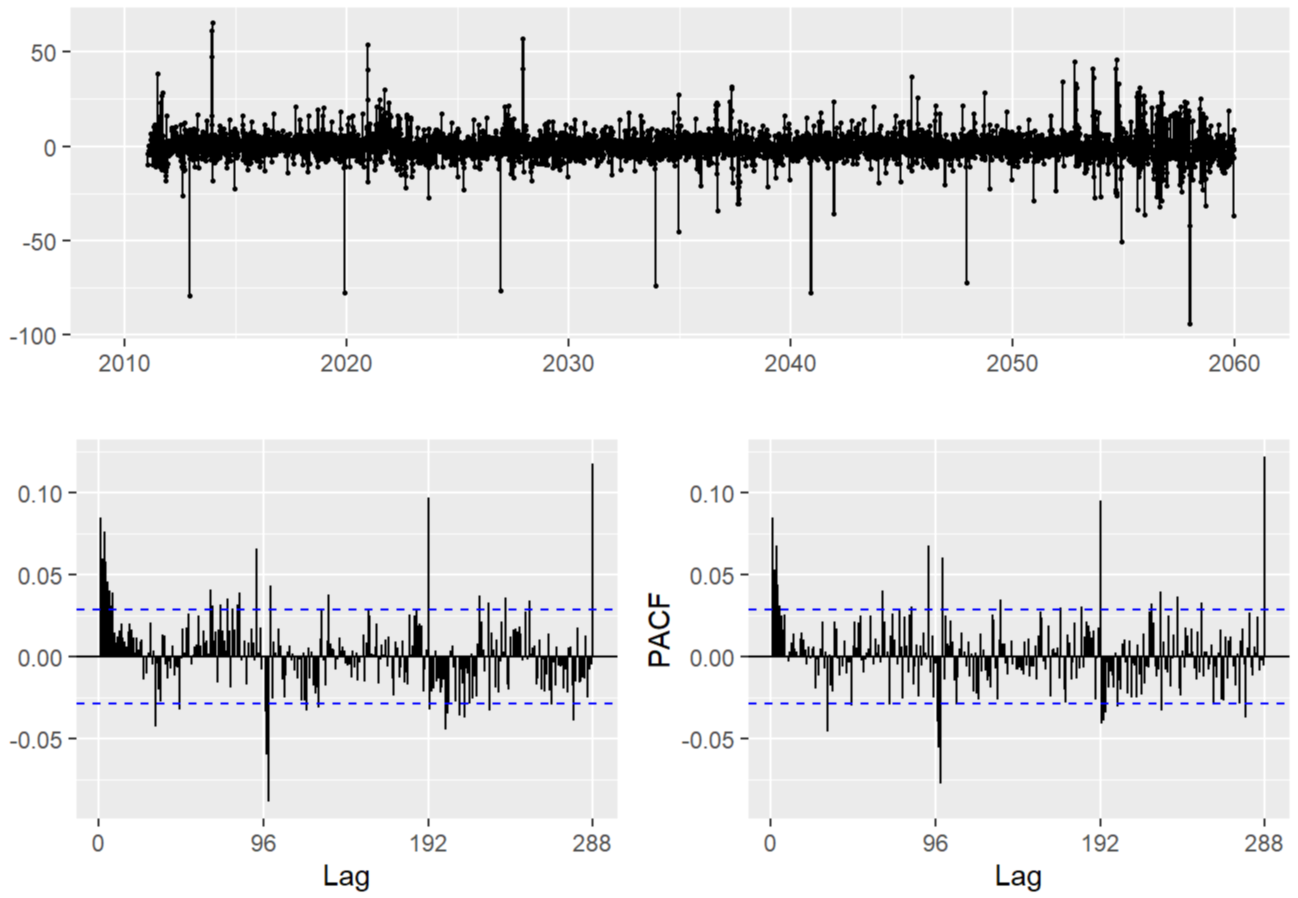
```
model_NN_cov=nnetar(ts_Power_train,xreg=ts_Temp_train)
```

```
summary(model_NN_cov)
```

```
##           Length Class        Mode     
## x         4795   ts           numeric  
## m            1   -none-       numeric  
## p            1   -none-       numeric  
## P            1   -none-       numeric  
## scalex       2   -none-       list     
## scalexreg    2   -none-       list     
## size         1   -none-       numeric  
## xreg      4795   ts           numeric  
## subset    4795   -none-       numeric  
## model       20   nnetarmodels list     
## nnetargs     0   -none-       list     
## fitted    4795   ts           numeric  
## residuals 4795   ts           numeric  
## lags        26   -none-       numeric  
## series       1   -none-       character
## method       1   -none-       character
## call         3   -none-       call     
```

```
model_NN_cov %>% residuals() %>% ggtsdisplay()
```

```
## Warning: Removed 96 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Neurone Network does not look better. The RMSE (NN cov : 14.97342) is not better than the best ARIMA.

```
prev_NN_cov=forecast(model_NN_cov,h=frequency, xreg= ts_Temp_valid)
cat('NN cov :',sqrt(mean((prev_NN_cov$mean-ts_Power_test)^2)), '\n')
```

```
## NN cov : 17.34799
```

Finally, the best model with covariate is the auto ARIMA with covariate (RMSE 5.908364). It is also better than the best model (after cross validation) that does not take into account the outdoor temperature.

Let's forecast now the Power consumption. We previously split the Temperature dataset in train, trainvaliv, valid and test. The test part is the one where we know the future and has to be used as covariate value. Before we need to retain the best model on all the Power dataset + the covariate dataset (traiVal). Then, we will forecast by using this theoretical known future on the Temperature..

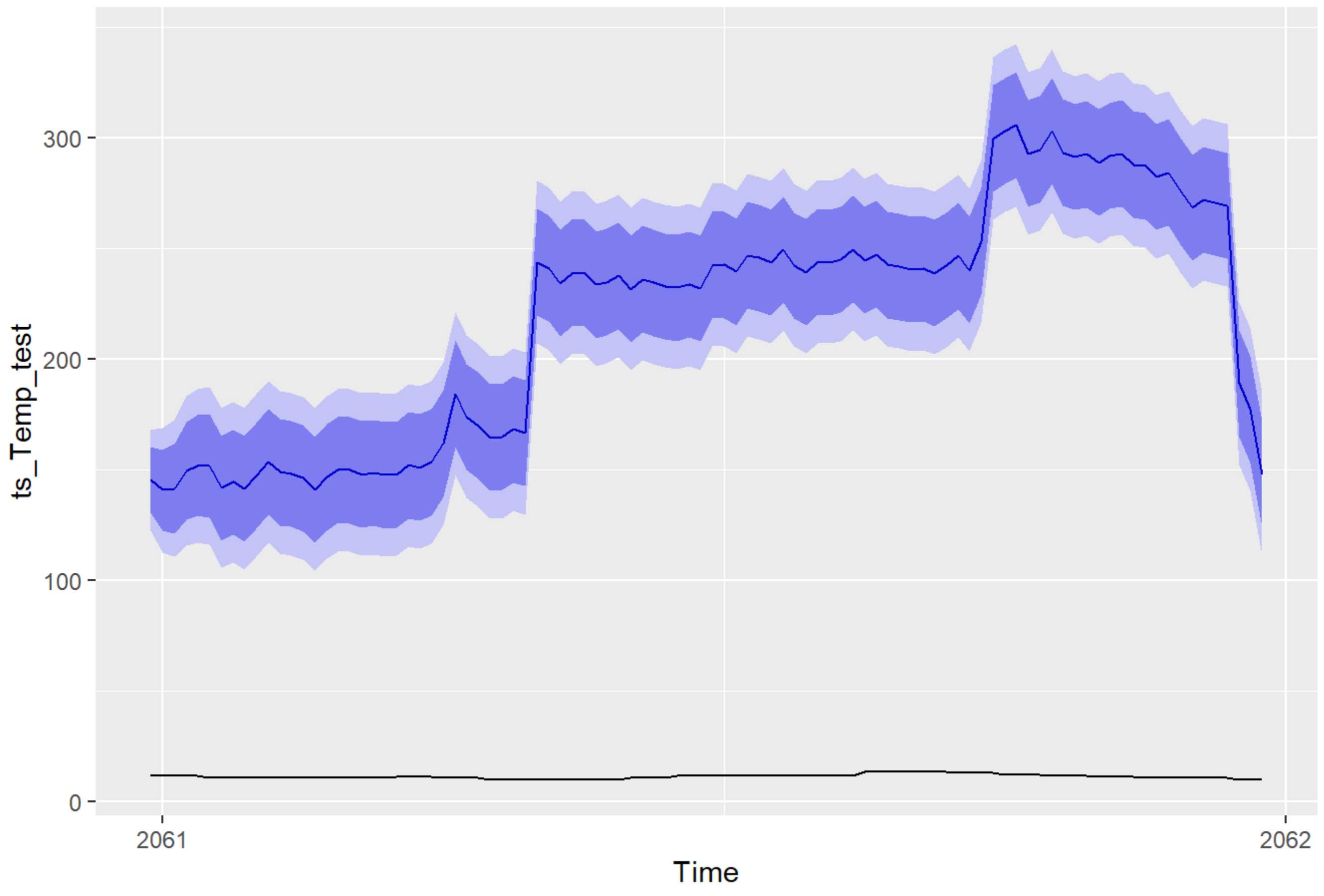
```
model_arima_auto_cov_final=Arima(ts_Power, xreg= ts_Temp_trainValid, order=c(5,0,1), seasonal
= c(0,1,0))
summary(model_arima_auto_cov_final)
```

```
## Series: ts_Power
## Regression with ARIMA(5,0,1)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5     ma1  Temp (C°)
##       0.4136  0.2729  0.1898  -0.2209  0.0685  0.3065     0.4163
## s.e.  0.1242  0.0869  0.0191   0.0292  0.0315  0.1246     0.2598
##
## sigma^2 = 134.7:  log likelihood = -18555.24
## AIC=37126.47   AICc=37126.5   BIC=37178.28
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.08919757 11.48153 6.619749 -0.8233645 3.728019 0.7146069
##                       ACF1
## Training set -0.0004830426
```

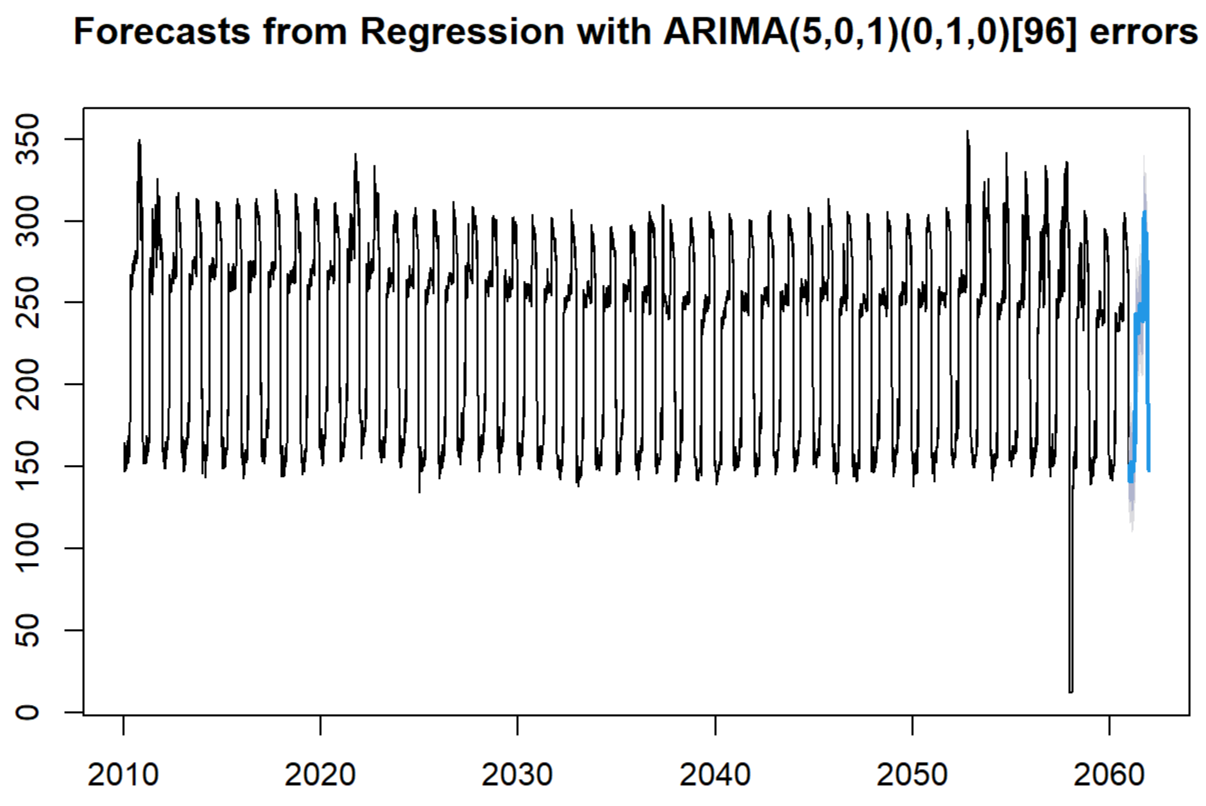
```
prev_arima_auto_cov_final=forecast(model_arima_auto_cov_final , h=frequency, xreg= ts_Temp_te
st)
```

Plot both curves.

```
autoplot(ts_Temp_test)+autolayer(prev_arima_auto_cov_final)
```



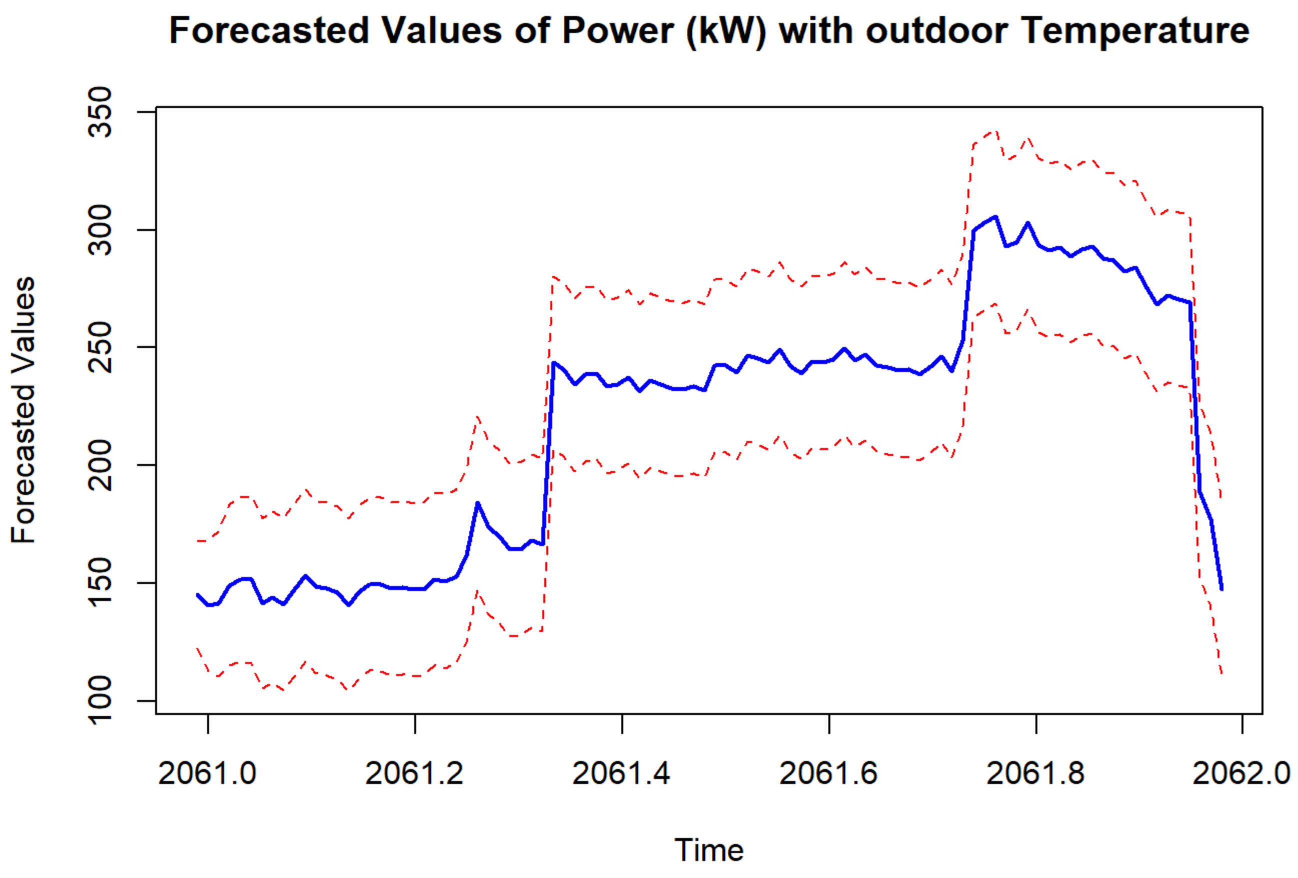
```
plot(prev_arima_auto_cov_final)
```



Exports both Forecasts

```
# Extract forecast values and confidence intervals
forecast_values <- prev_arima_auto_cov_final$mean
lower_bound <- prev_arima_auto_cov_final$lower[, 2]  # 95% lower bound
upper_bound <- prev_arima_auto_cov_final$upper[, 2]  # 95% upper bound
```

```
# Generate the plot
plot(forecast_values, type = "l", col = "blue", lwd = 2, ylim = range(lower_bound, upper_boun
d),
     main = "Forecasted Values of Power (kW) with outdoor Temperature", xlab = "Time", ylab =
"Forecasted Values")

# Add confidence intervals to the plot
lines(lower_bound, col = "red", lty = 2)
lines(upper_bound, col = "red", lty = 2)
```

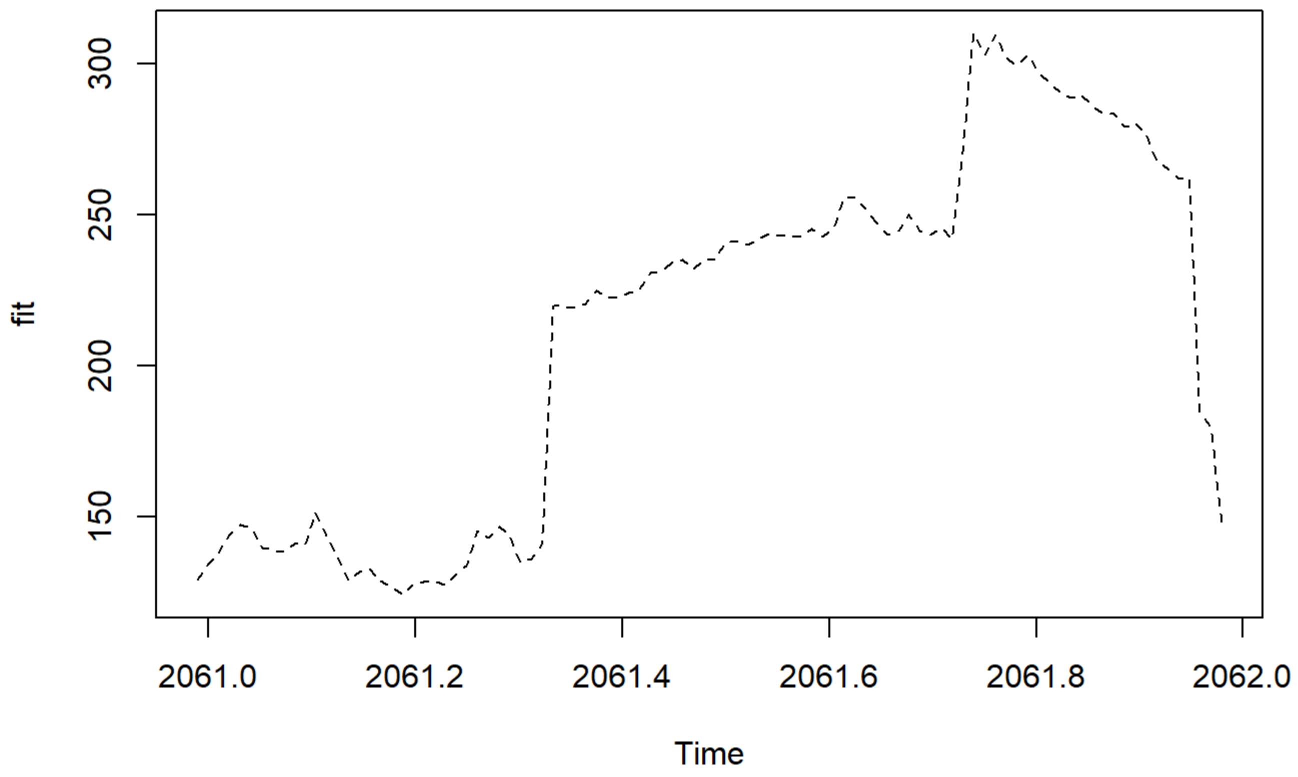


Export the Forecast

```
# Combine forecasted values and xreg values into a data frame
xreg = ts_Temp_test
export_data <- data.frame(
  Forecast = as.numeric(forecast_values),  # Convert forecasted values to numeric
  Xreg1 = xreg
)

# View the data frame before exporting (optional)
print(export_data)
```

Forecast file without covariates

```
prev_hw_without_cov_final<-predict(model_hw_without_cov_final,n.ahead=frequency)
plot(prev_hw_without_cov_final, lty=2)
```



```
# Extract forecast values and confidence intervals
forecast_values_without_cov <- prev_hw_without_cov_final
```

```
# Combine forecasted values and xreg values into a data frame
xreg = ts_Temp_test
export_data <- data.frame(
  ForecastWithout = as.numeric(forecast_values_without_cov),  # Convert forecasted values to
numeric
  ForecastWith = as.numeric(forecast_values),  # Convert forecasted values to numeric
  Xreg1 = xreg
)

# View the data frame before exporting (optional)
print(export_data)
```

```
# Export the data to a CSV file
write.csv(export_data, "forecast_and_xreg.csv", row.names = FALSE)
```