Sonnellini-Assignment

Andrea Sonnellini

01 ottobre 2020

Overview

This report details the analysis I performed to forecast the energy consumption for 2/17/2010.

To achieve this goal, I tried to run all the algos we saw during the class.

- exp. smoothing
- SARIMA
- NN
- Dynamic Regression
- VAR

The main difficulty I experienced is that the high number of datapoints (1 every 15 minutes) prevented me from being able to use "in the standard way" some R functions like Arima or hw with the values I wanted to set for certain parameters. In particular there were some long-range (i.e. weekly) correlations which I could not model with SARIMA because the values of p,q,P,Q and the periodicity supported by the Arima R function cannot be too high.

To overcome this problem, I tried initially to perform the following actions:

1) downsample the time series - I tried to alternative approaches:

1A) Compute for each hour the average of 4 points in that hour

OR

1B) For each hour consider only the first point (the one at xx:15)

2) Run the algos on the downsampled timeseries

3) Make a forecast for the downsampled timeseries

4) Finally recover the "missing" points via interpolation

Unfortunately the above procedure led to poor results in terms of error on the test set, so I dropped this idea.

This PDF document does not report those attempts given that eventually I did not rely on them.

Therefore, to overcome the issues mentioned above when using SARIMA, I did not consider long-range correlations and I considered only "low"" values p,q,P,Q. The drawback is that eventually the residuals of the models I obtained were not white noise - neverhteless I could not do find a better way to perform this analysis.

Finally, the most accurate algo to perform predictions (based on test error) turned out to be a manually chosen SARIMA. This holds both for the case where we predict using the temperature or not.

Data Pre-processing and Exploratory analysis

Load the packages.

```
library(forecast)
library(ggplot2)
library(openxlsx)
```

```r
set.seed(1986) # I set a seed to ensure to always obtain the same results even in case I will use some
```

Load the data.

The file Elec-train.xlsx contains electricity consumption (kW) and outdoor air temperature for one building. These quantities are measured every 15 minutes, from 1/1/2010 1:15 to 2/16/2010 23:45 (47 days). In addition, outdoor air temperature are available for 2/17/2010. The goal is to forecast electricity consumption (kW) for 2/17/2010.

My assumption is that we will predict the next 96 values of electricity consumption starting from 2/17/2010 at 00:00 until 2/17/2010 23:45.

```r
# Check the content of the file - NOTE: the file must not be open at the same time on Excel
read.xlsx(xlsxFile = "D:/BIG_DATA/DSTI/OneDrive - Data ScienceTech Institute/2020-09-Time-series/assignm
```

```
##                Power.(kW) Temp.(C°)
## 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
## 1/1/2010 2:45       157.7  10.55556
## 1/1/2010 3:00       163.2  10.55556
## 1/1/2010 3:15       151.7  10.00000
```

```r
elCons.data =  read.xlsx(xlsxFile = "D:/BIG_DATA/DSTI/OneDrive - Data ScienceTech Institute/2020-09-Time

str(elCons.data)
```

```
## 'data.frame':    4603 obs. of  2 variables:
##  $ Power.(kW): num  165 152 147 154 154 ...
##  $ Temp.(C°) : num  10.6 10.6 10.6 10.6 10.6 ...
```

I rename the columns in a simpler way.

```r
colnames(elCons.data) = c("Power","Temp")
```

Inspect head and tail of the dataframe

```r
head(elCons.data)
```

```
##                Power     Temp
## 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
```

```r
tail(elCons.data)
```

```
##                  Power     Temp
## 2/17/2010 22:30    NA 13.88889
## 2/17/2010 22:45    NA 13.88889
## 2/17/2010 23:00    NA 13.88889
## 2/17/2010 23:15    NA 12.77778
## 2/17/2010 23:30    NA 12.77778
```

```
## 2/17/2010 23:45    NA 12.77778
```

```
tail(head(elCons.data, n = dim(elCons.data)[1] -96*2))
```

```
##                  Power     Temp
## 2/15/2010 22:30 289.6 11.66667
## 2/15/2010 22:45 284.7 11.66667
## 2/15/2010 23:00 283.9 11.66667
## 2/15/2010 23:15 211.9 12.22222
## 2/15/2010 23:30 203.8 12.22222
## 2/15/2010 23:45 167.2 12.22222
```

The last 96 rows of the dataset refer to 17 Feb 2010, so the column power has no values.

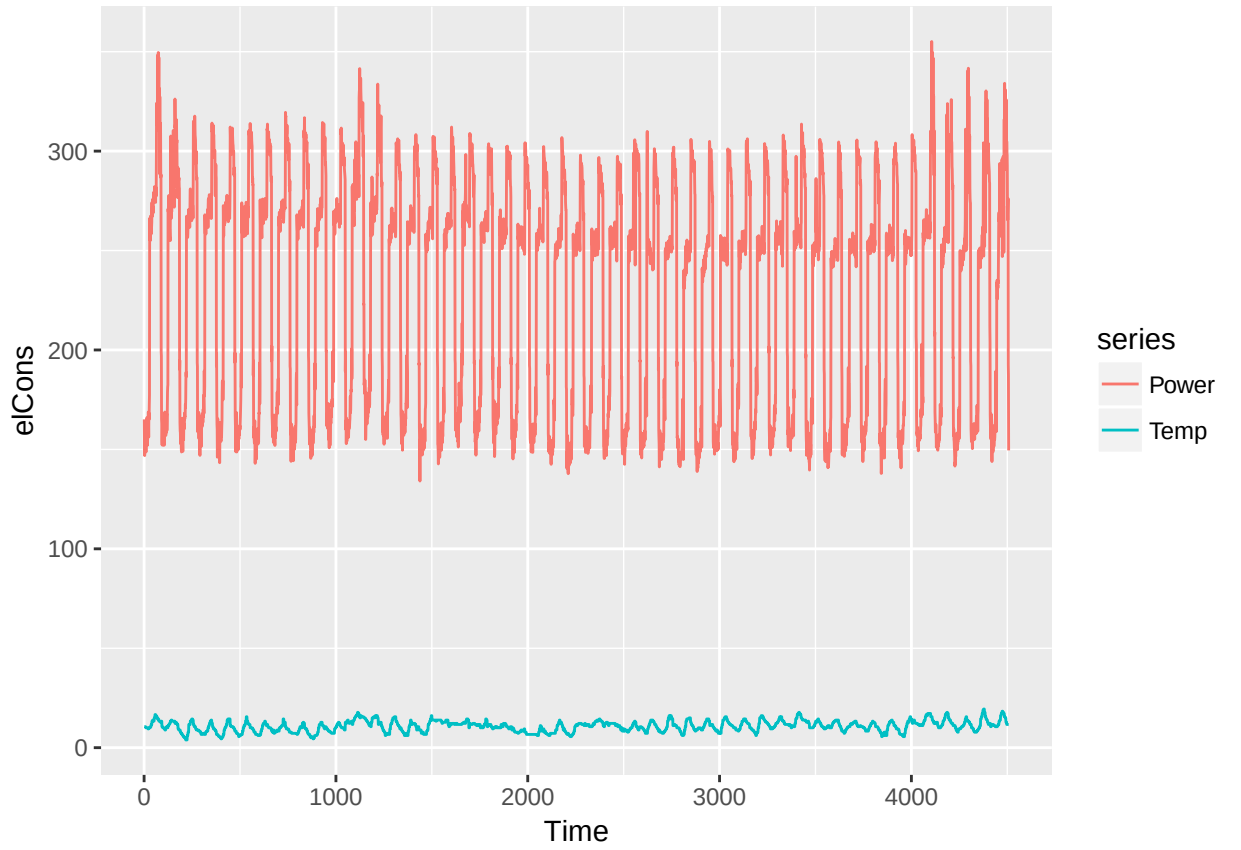
I convert the dataframe in ts format and split the dataframe in data before 17 Feb (elCons) and data for the 17 Feb (elCons.future).

The rest of the analysis will be performed on data before 17 Feb (elCons) only.

```
elCons = ts(head(elCons.data, n = dim(elCons.data)[1] -96))# before 17 Feb 2010

elCons.future = ts(tail(elCons.data, n = 96)) # @ 17 Feb 2010
```

Visual inspection

```
autoplot(elCons)
```



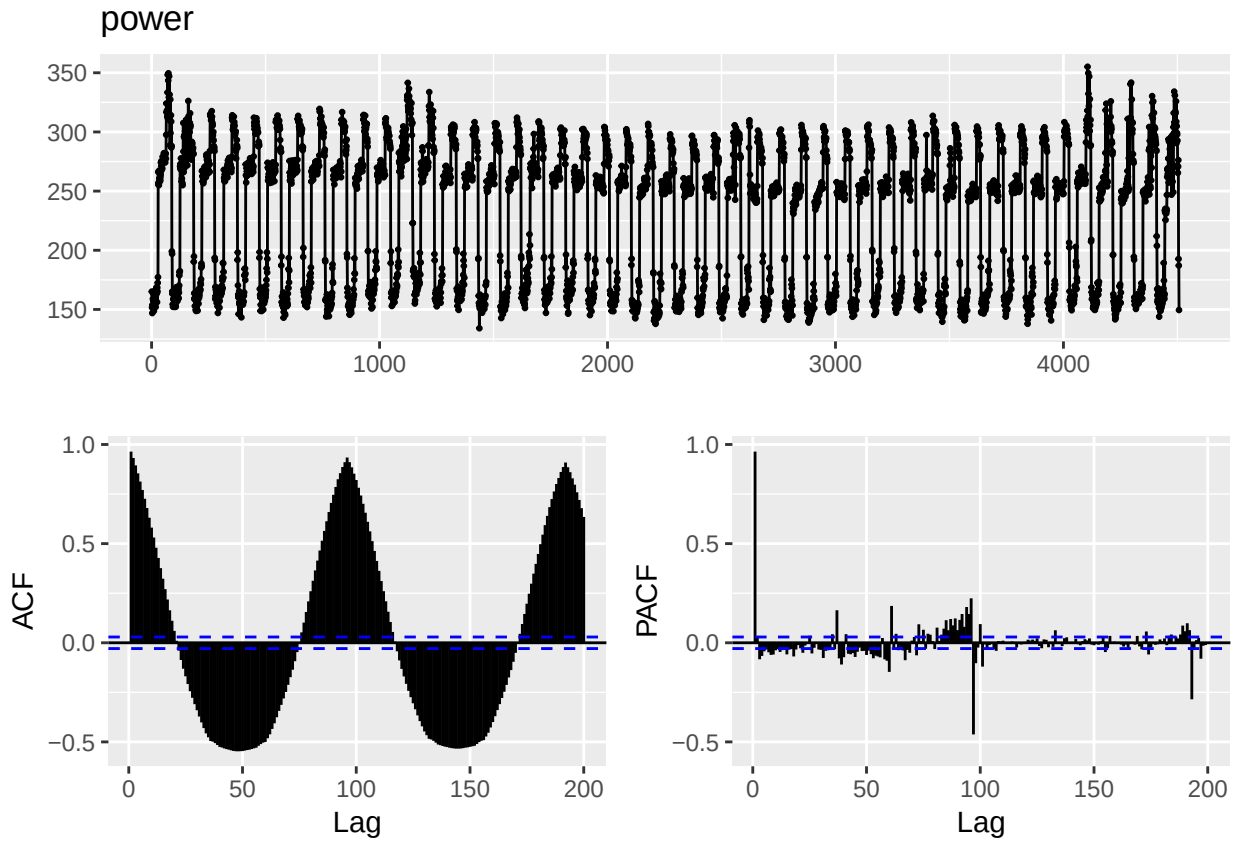
Focusing on Power, from the above there seems to be a seasonality, most likely with a period of 96 (number of points which cover 1 day). Apparently no trend.

The variance of Power is quite constant except in the last part (approx after 4000) where it seems to be higher.

Check seasonality

I will double check the seasonality period looking at acf - given that the seasonality most likely is 96, I force R to compute ACF till lag = 200.

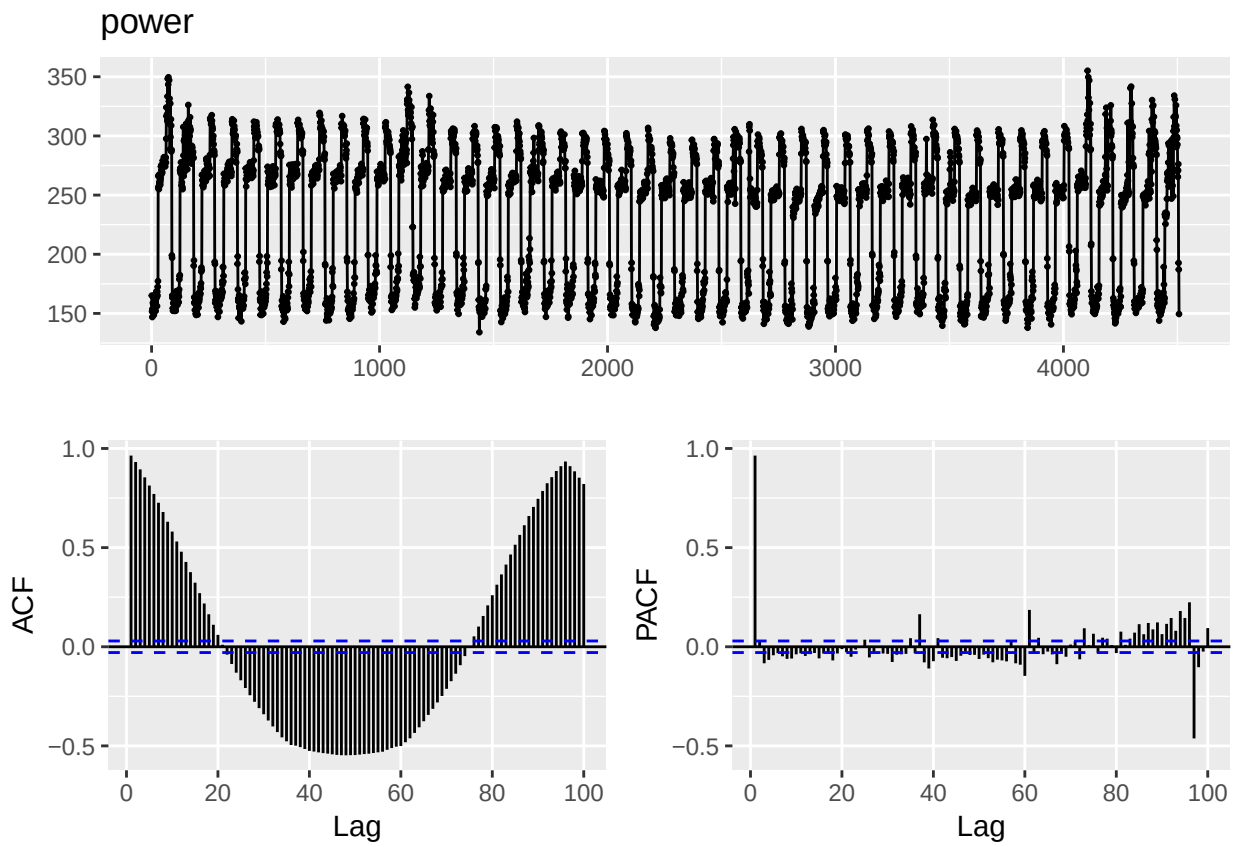
```
power = elCons[,"Power"]

ggtsdisplay(power, lag.max = 200)
```



The above confirms a seasonality of 96 and no trend.

I zoom in till lag = 100 to ensure seasonal period is 96.

```
ggtsdisplay(power, lag.max = 100) #acf confirms seasonality is 96
```



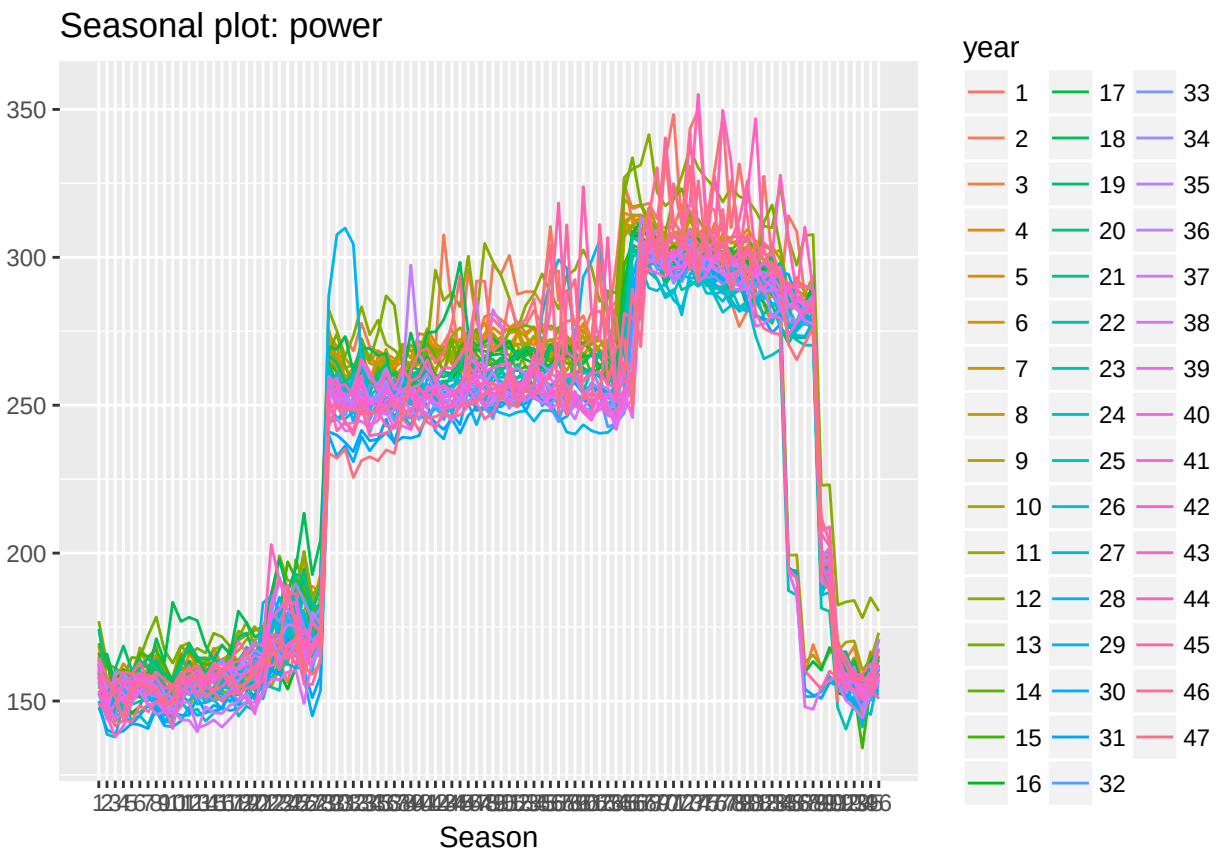
Seasonality seems indeed to be 96.

We set frequency = 96 in the timeseries.

```
power = ts(power, frequency = 96)
```

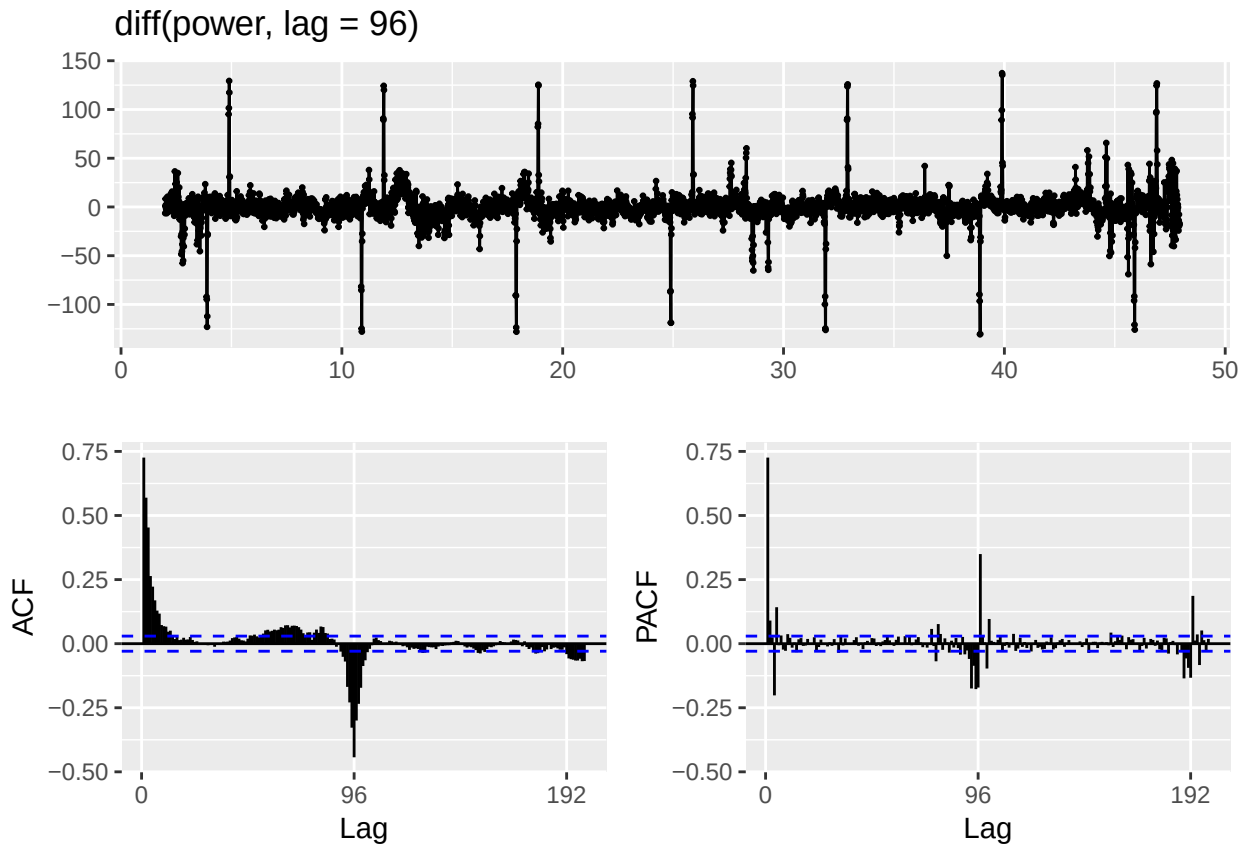
And check on a season plot

```
ggseasonplot(power)
```



Next, I try to differentiate with a lag of 96 and re-check the residuals, the acf and pacf

```
ggtsdisplay(diff(power, lag = 96), lag.max = 200) #acf confirms seasonality is 96
```



The remaining noise does seem to still have some periodic spikes, i.e. it is not stationary.

Autocorrelations as well are relevant, meaning that there is something more to be modeled beyond the "simple" daily seasonality.

I will also perform a test on residuals to check whether they are white noise or not. In the below Box.test I set lag = 100 to cover also the seasonal period.

```
Box.test( diff(power, lag = 96), lag = 100,type = "Ljung-Box")
```

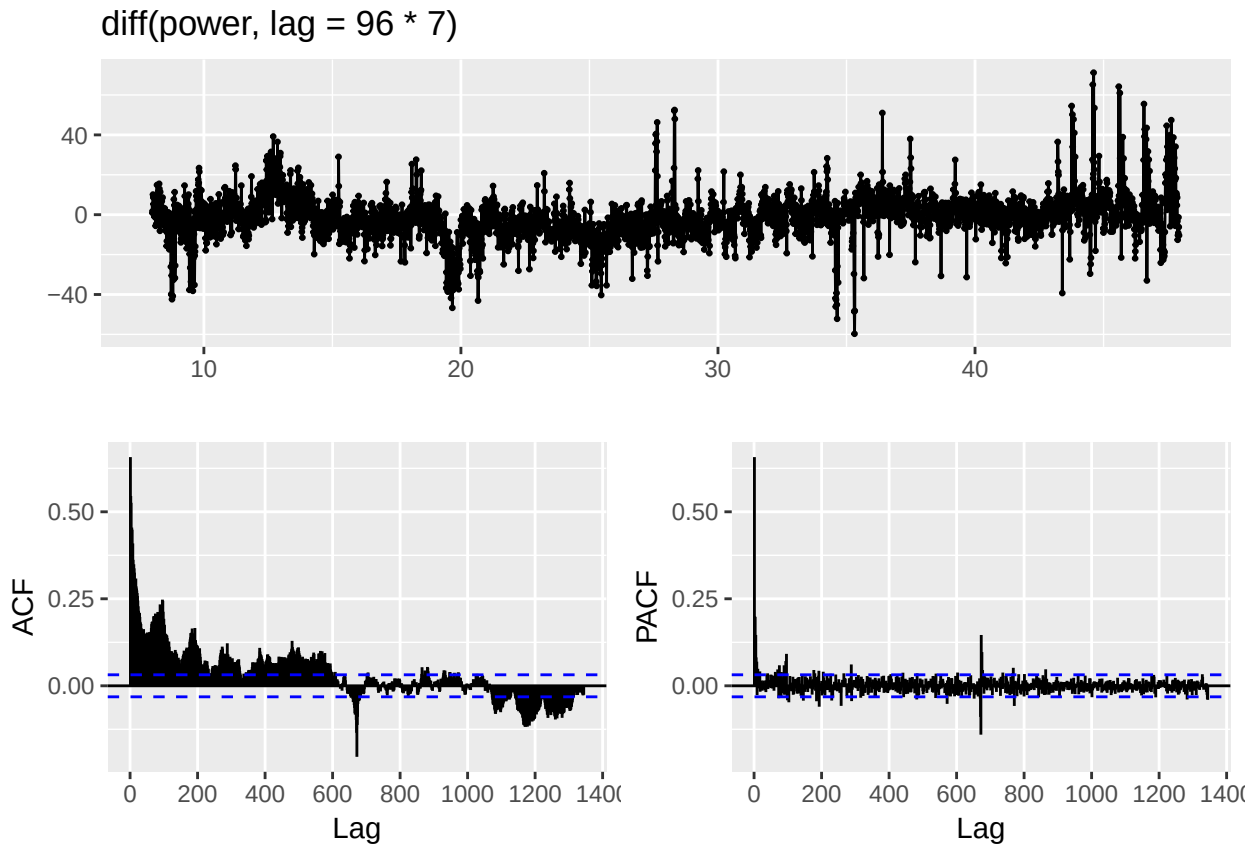
```
##
##  Box-Ljung test
##
## data:  diff(power, lag = 96)
## X-squared = 8600.1, df = 100, p-value < 2.2e-16
```

The above test confirms that, upon removing daily seasonality, we do not obtain white noise.

Check weekly seasonality

We try now to differentiate wrt the weekly seasonality (96*7), and check if this helps to have a stationary noise.

```
ggtsdisplay(diff(power, lag = 96*7), lag.max = 96*7*2)
```



The differentiated by 96*7series seems more stationary than before indeed. This is something to keep in mind when building the SARIMA models.

Train and test set

In next sections we will train different models on a train set and compare their performances based on RMSE on a test set.

The the test set will be of size 96, i.e. the same size that of the window of time we want to forecast.

```
power.train = head(power, n = length(power)[1] - 96)
power.test = tail(power, n = 96)
```

Models without considering Temperature

Exponential smoothing

Given the seasonality of data, we can run a hw seasonal model with additive effect.

```
#hw.seas = hw(power.train, seasonal = "additive")# returns error

#pred.hw.seas = forecast(hw.seas, h = 96)

#autoplot(power.test) + autolayer(pred.hw.seas, series = "HW seasonal")
```

The function "hw" returns the error "Error in ets(x,"AAA", alpha = alpha, beta = beta, gamma = gamma, phi = phi, : Frequency too high".

Looking online I found the following article https://robjhyndman.com/hyndsight/longseasonality/ from the author of the function hw, Hyndman, that explains that hw supports seasonilities up to 24 (because higher seasonalities requires to estimate too many parameters). So we cannot use hw on the raw data.

A possible solution could be to refactor the dataset in order to have datapoints with the same value for one hour. This would reduce the number of datapoints, leading to a seasonality of 24.

As discussed in the overview section however this approach turned out to give non-accurate results.

SARIMA

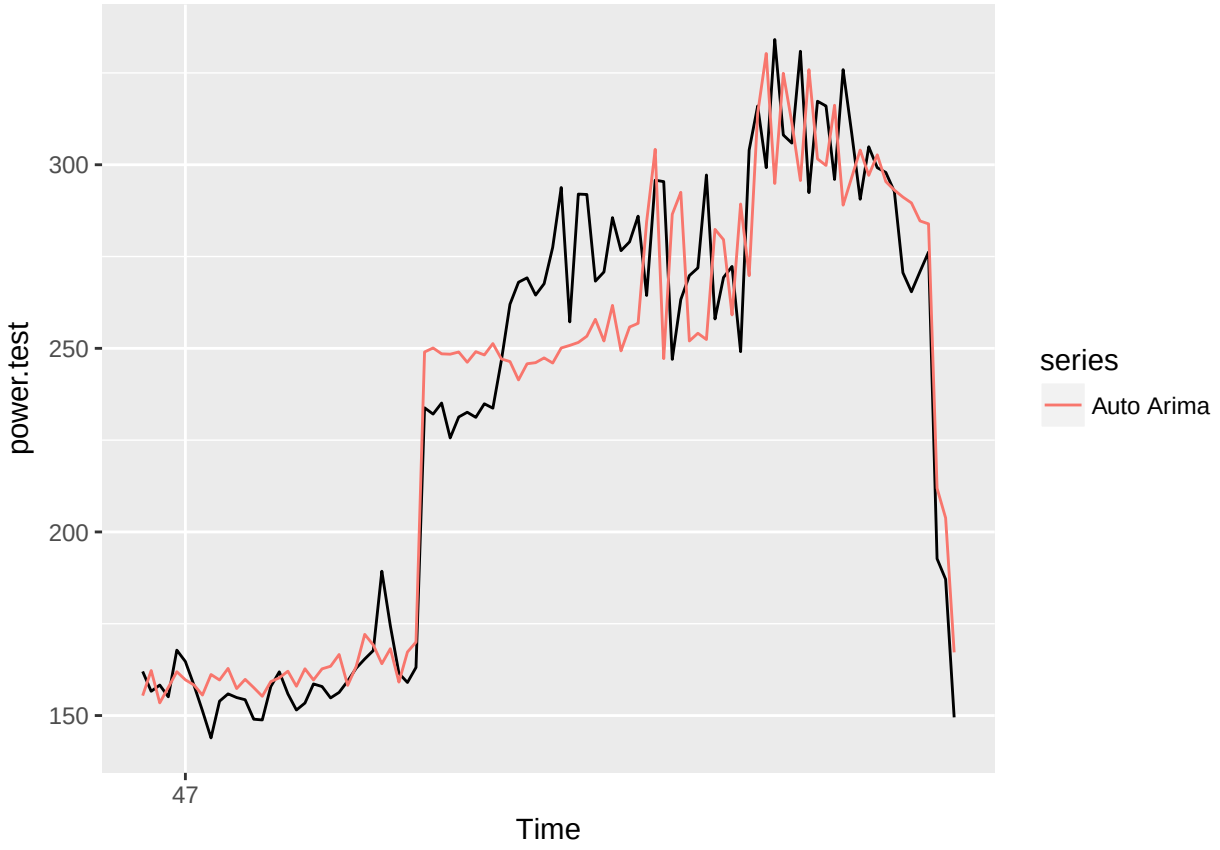
Automatic SARIMA

First attempt is to run an automatic procedure for SARIMA.

```
auto.sarima = auto.arima(power.train)

print(auto.sarima)
```

```
## Series: power.train
## ARIMA(5,0,4)(0,1,0)[96]
##
## Coefficients:
##          ar1      ar2      ar3      ar4     ar5     ma1     ma2     ma3
##       0.0768  -0.0980  -0.0169  -0.0033  0.2530  0.6403  0.6243  0.6255
## s.e.  0.1768   0.0716   0.0563   0.0536  0.0553  0.1774  0.1098  0.0813
##          ma4
##       0.2925
## s.e.  0.0948
##
## sigma^2 estimated as 112.1:  log likelihood=-16300.12
## AIC=32620.25   AICc=32620.3   BIC=32683.94
```

```
pred.auto.sarima = forecast(auto.sarima, h = 96)

autoplot(power.test) + autolayer(pred.auto.sarima$mean, series = "Auto Arima")
```



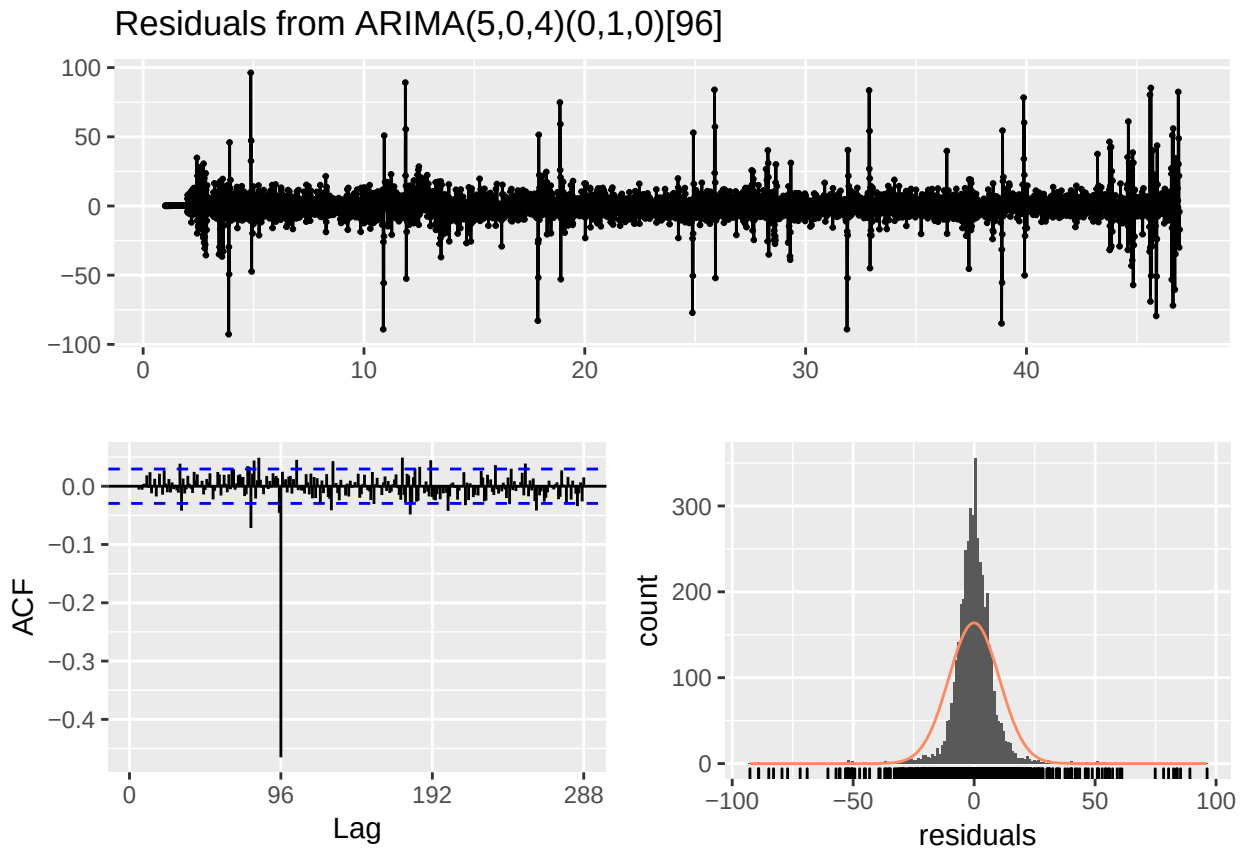

```
RMSE.func = function(predictions, realData){

  sqrt( mean( (predictions[["mean"]] - realData )^2)   )

}
```

```
RMSE.auto.sarima = RMSE.func(pred.auto.sarima, power.test)
```

The automatic selection has returned an ARIMA(5,0,4)(0,1,0)[96] with a RMSE of 19.78435.

Check the residuals

```
checkresiduals(auto.sarima, test = "LB", lag = 100)
```

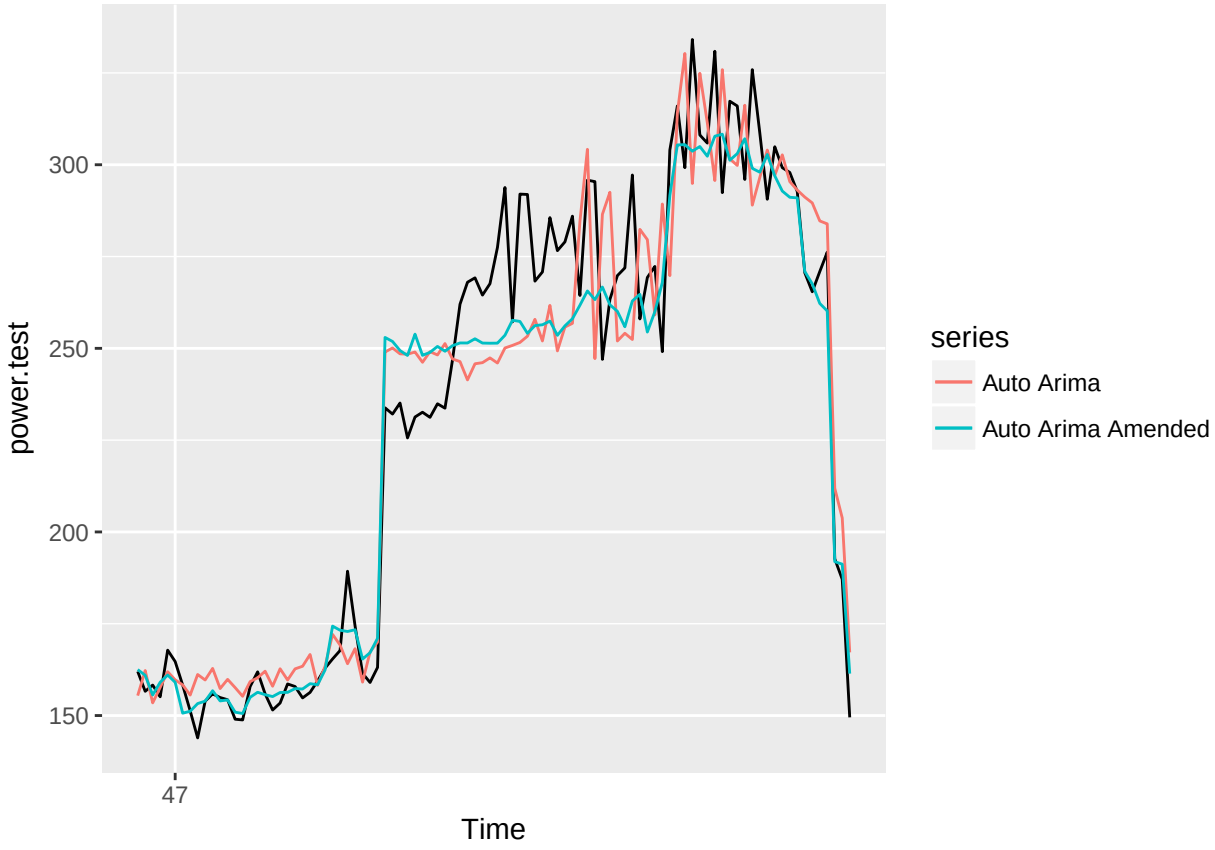



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,4)(0,1,0)[96]
## Q* = 1127.7, df = 91, p-value < 2.2e-16
##
## Model df: 9.   Total lags used: 100
```

The noise seems not to be stationary.

Looking at the ACF we see a spike at lag = 96 ==> this seems to suggest we could add to the automatically selected SARIMA a MA1 seasonal term.

```
auto.sarima.amended = Arima( power.train ,order = c(5,0,4), seasonal = c(0,1,1))

pred.auto.sarima.amended = forecast(auto.sarima.amended, h = 96)

autoplot(power.test) + autolayer(pred.auto.sarima$mean, series = "Auto Arima") + autolayer(pred.auto.sa
```

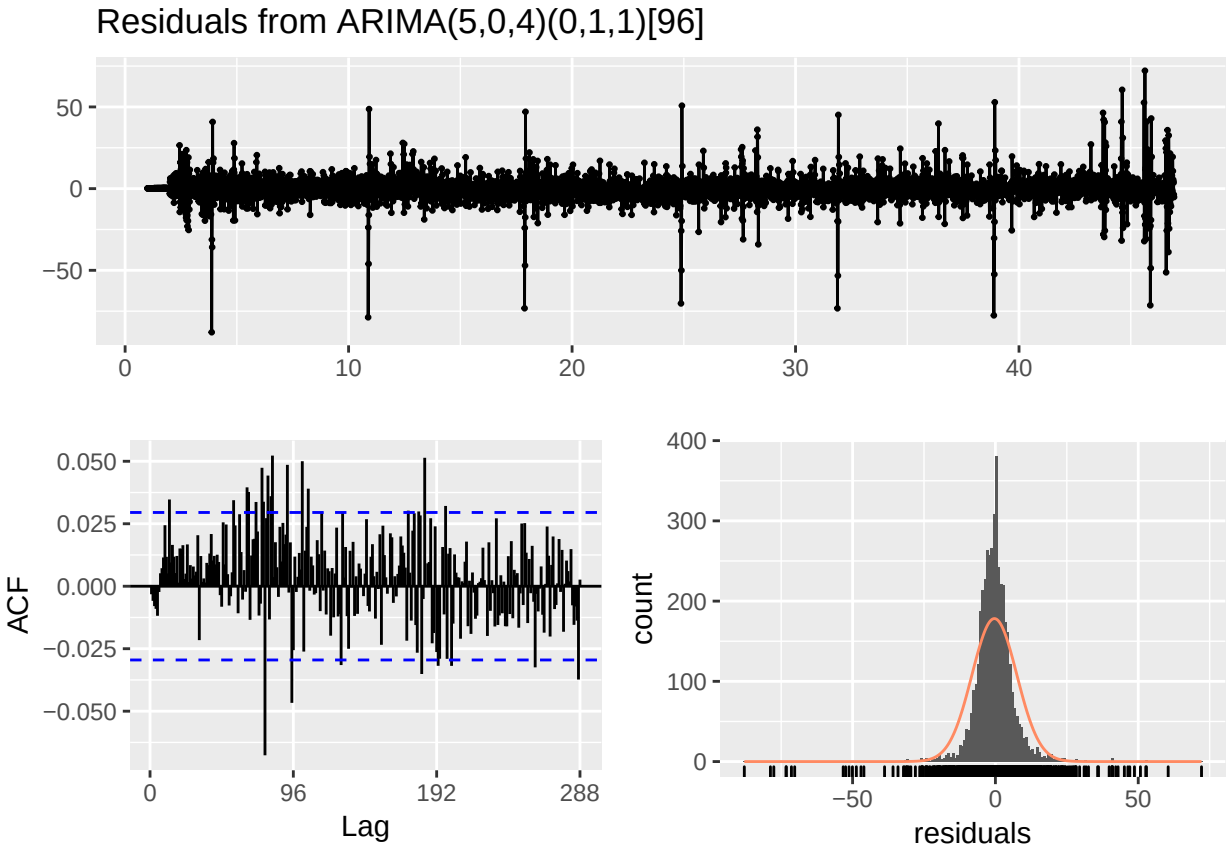



```
RMSE.auto.sarima.amended = RMSE.func(pred.auto.sarima.amended, power.test)
RMSE.auto.sarima.amended
```

```
## [1] 15.0636
```

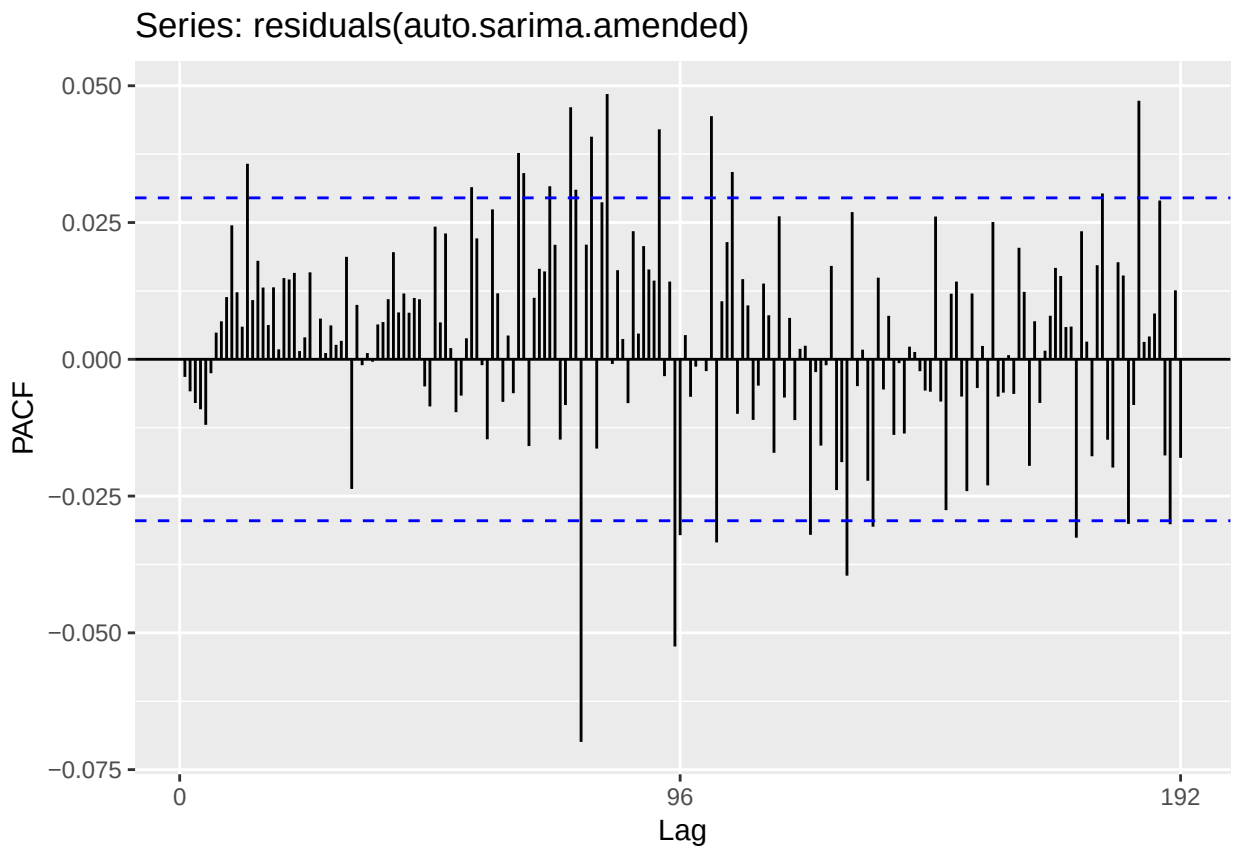
$\text{RMSE.auto.sarima.amended} = 15.0636$

```
checkresiduals(auto.sarima.amended, test = "LB", lag = 100)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,4)(0,1,1)[96]
## Q* = 180.75, df = 90, p-value = 4.723e-08
##
## Model df: 10.   Total lags used: 100
```

```
ggPacf( residuals(auto.sarima.amended))
```



The noise seems still to show some seasonality. Spike in pacf at 184 We could try to add a RA184

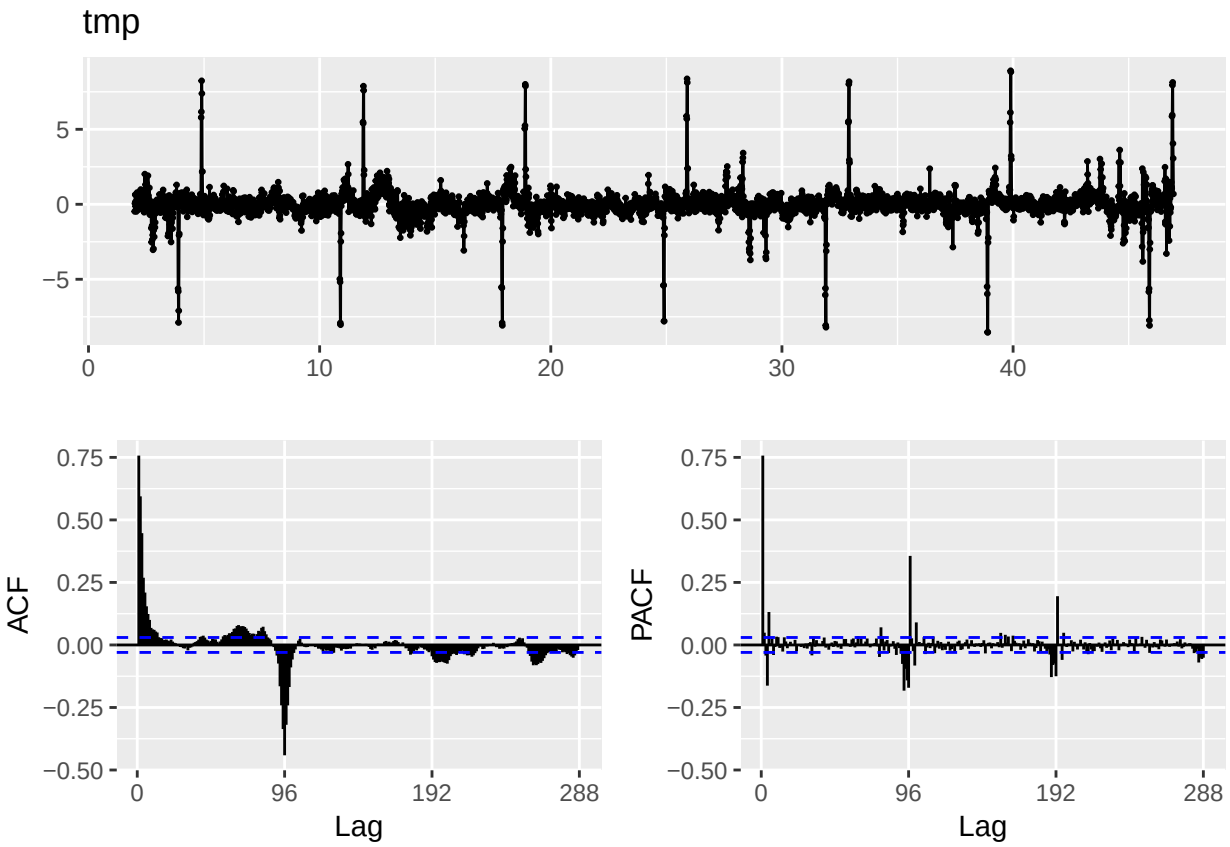
```
#auto.sarima.amended.2 = Arima( power.train ,order = c(184,0,4), seasonal = c(0,1,1))

#pred.auto.sarima.amended.2 = forecast(auto.sarima.amended.2, h = 96)
```

R returns an error (cannot process more than 100 parameters), too many parameters for ARIMA(184,0,4)(0,1,1)[96]

So we have to stick to SARIMA(5,0,4)(0,1,1)[96]

Last attempt is to use a BoxCox transformation, given that the last days in power.train show a variance which is slightly greater than previously.

```
lambda=BoxCox.lambda(power.train)
AP_BC=BoxCox(power.train,lambda)
tmp=diff(AP_BC,lag=96)
ggtsdisplay(tmp)
```



I do not see much improvements compared to the case without Using BoxCox.lambda

```
fit3=auto.arima(power.train,lambda = "auto")
prev=forecast(fit3,h=96)
```

```
RMSE.sarima.auto.lambda = RMSE.func(prev, power.test)
RMSE.sarima.auto.lambda
```

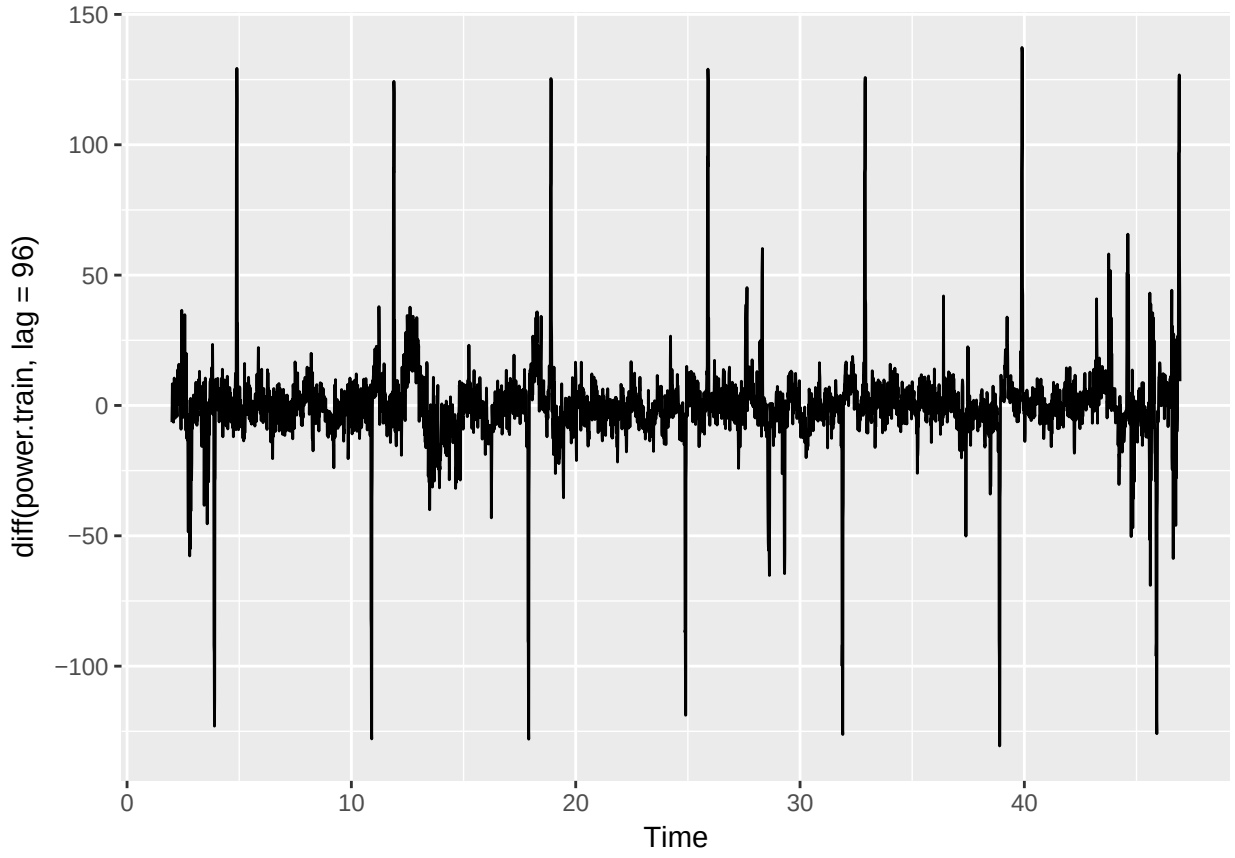
```
## [1] 19.77547
```

I do not see any improvement in using BoxCox transformation, so I will not futher use it.

Manual SARIMA

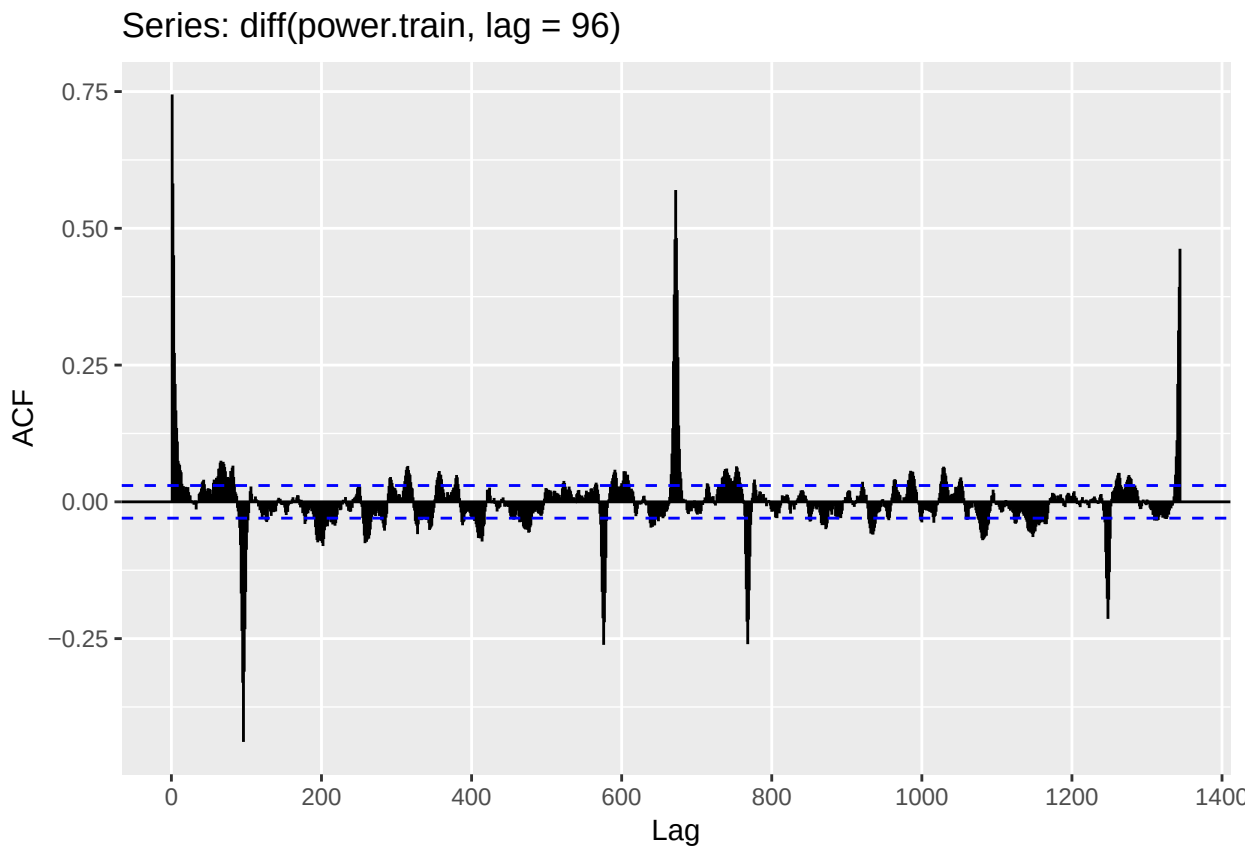
Manual SARIMA considering the daily seasonality only

```
autoplot(diff(power.train, lag = 96), lag.max = 200)
```



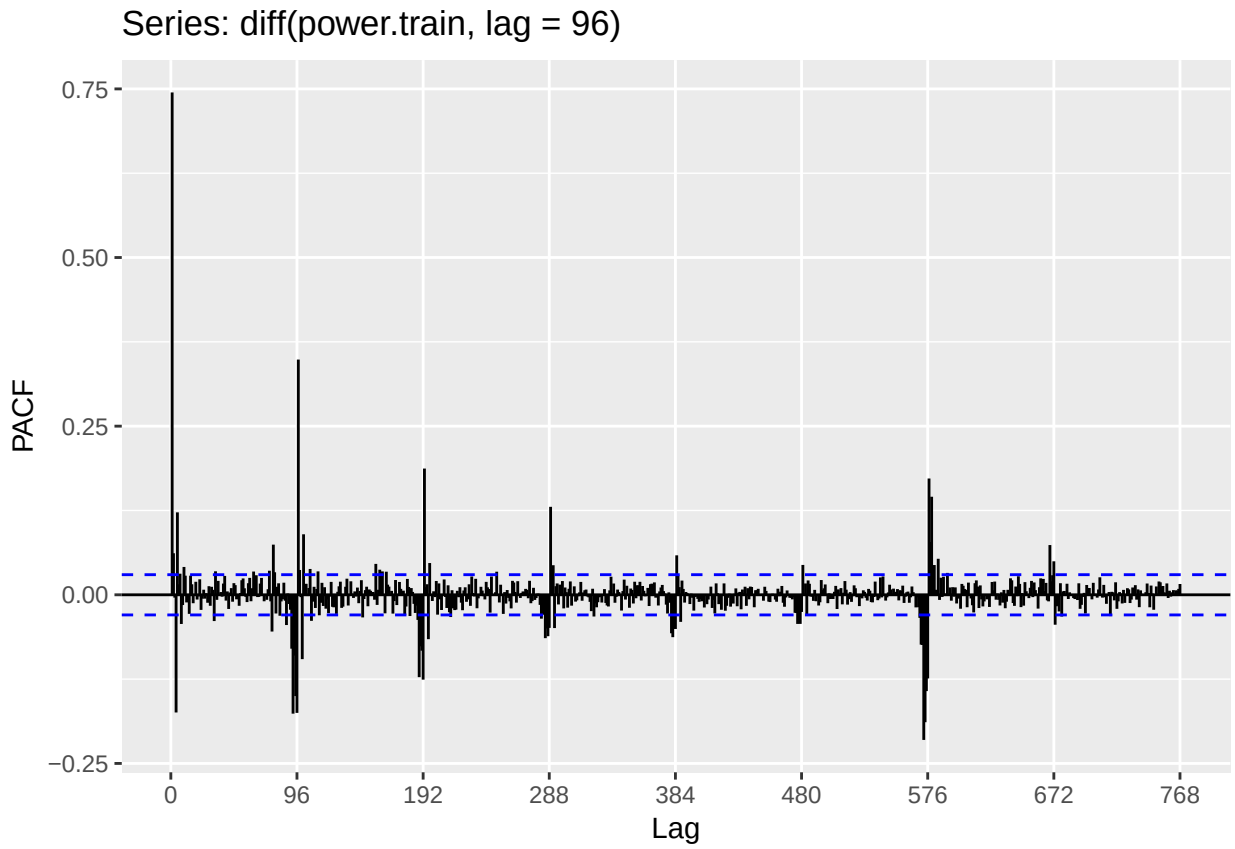
As already mentioned previously, we see 2 periodic spikes for 7 times. This seems to point to a weekly trend.

```
ggAcf(diff(power.train, lag = 96), lag.max = 96*7*2)
```



From the above graph we see a seasonal weekly effect.

```
ggPacf(diff(power.train, lag = 96),lag.max = 96*8)
```

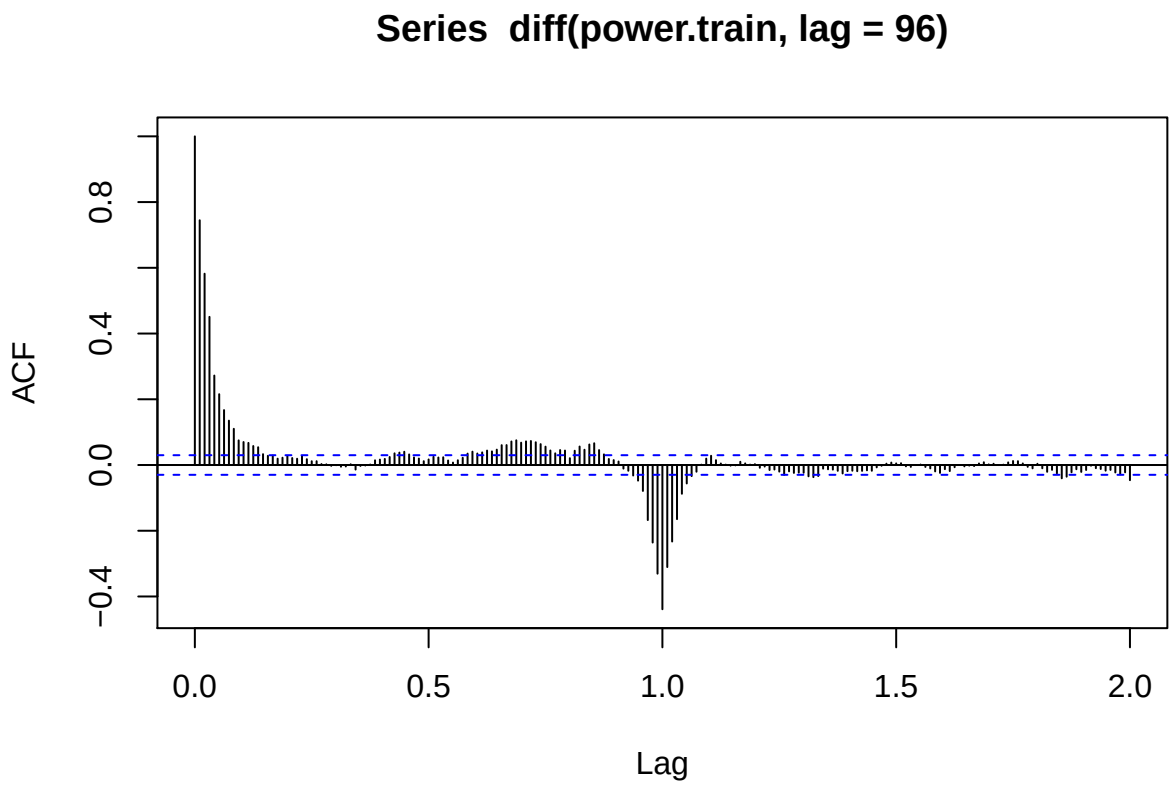


The above seems to suggest to check whether it is possible to build a SARIMA considering this weekly seasonality.

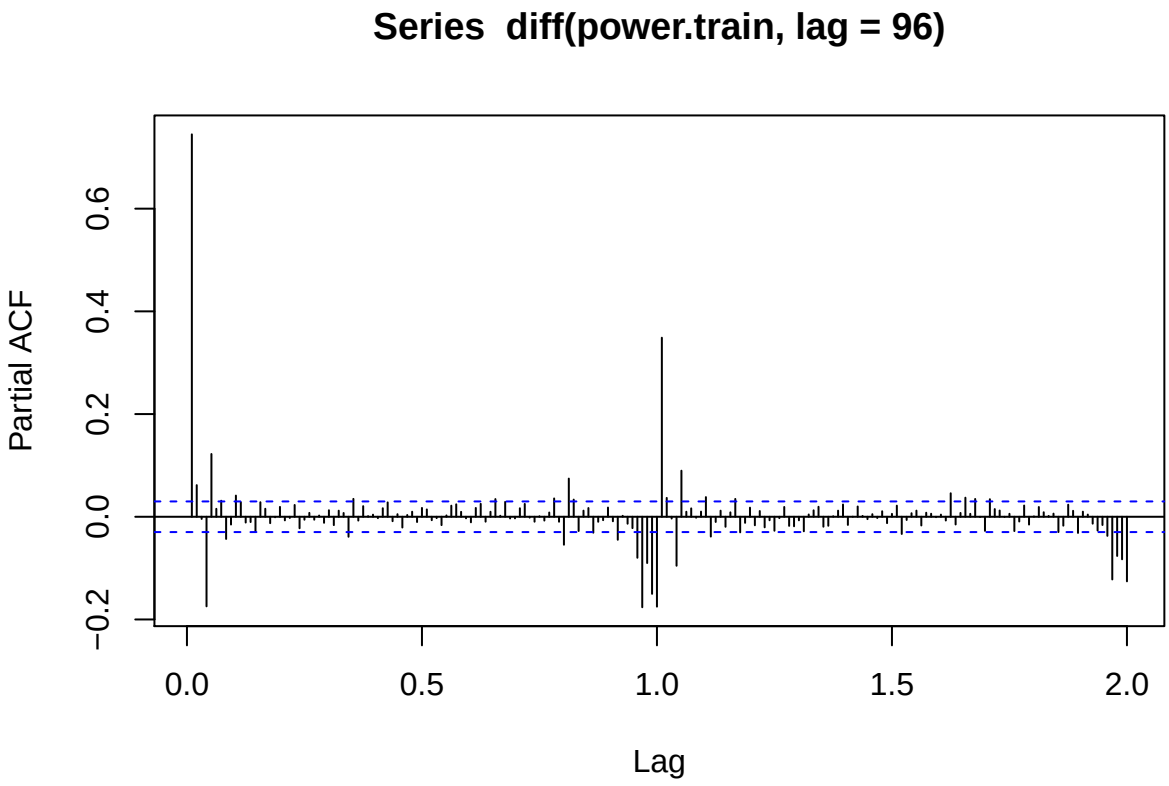
We will end this chapter trying to run a SARIMA that considers a daily periodicity, while in the next chapter we will consider weekly periodicity.

From the previous ACF and PACF we could see that in principle we should choose very high orders for p,q,P,Q but these are not supported by Arima functions. So we will limit our exploration of the ACF and PCF to lags within 2 days max.

```
acf(diff(power.train, lag = 96), lag.max = 96*2)
```

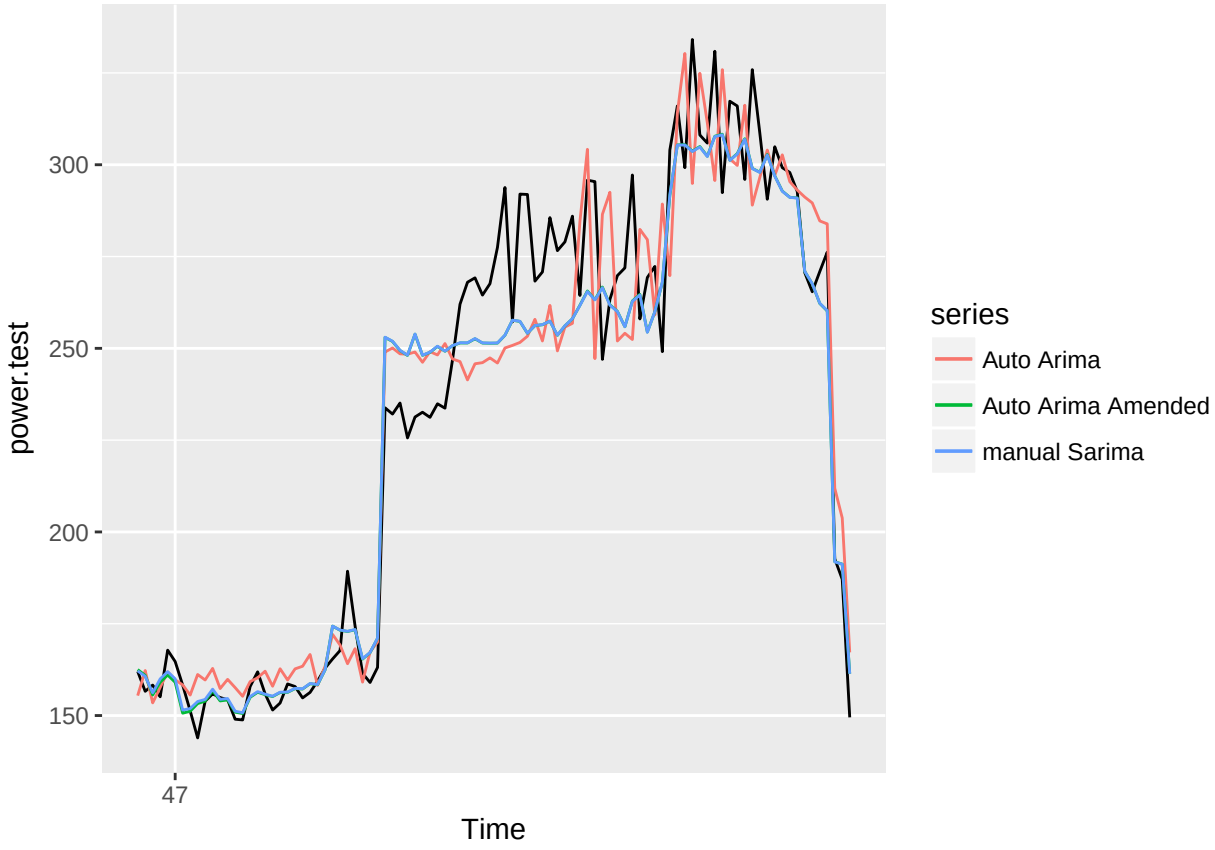



```
pacf(diff(power.train, lag = 96), lag.max = 96*2)
```



the above seems to suggest a non seasonal AR5 along with MA1.

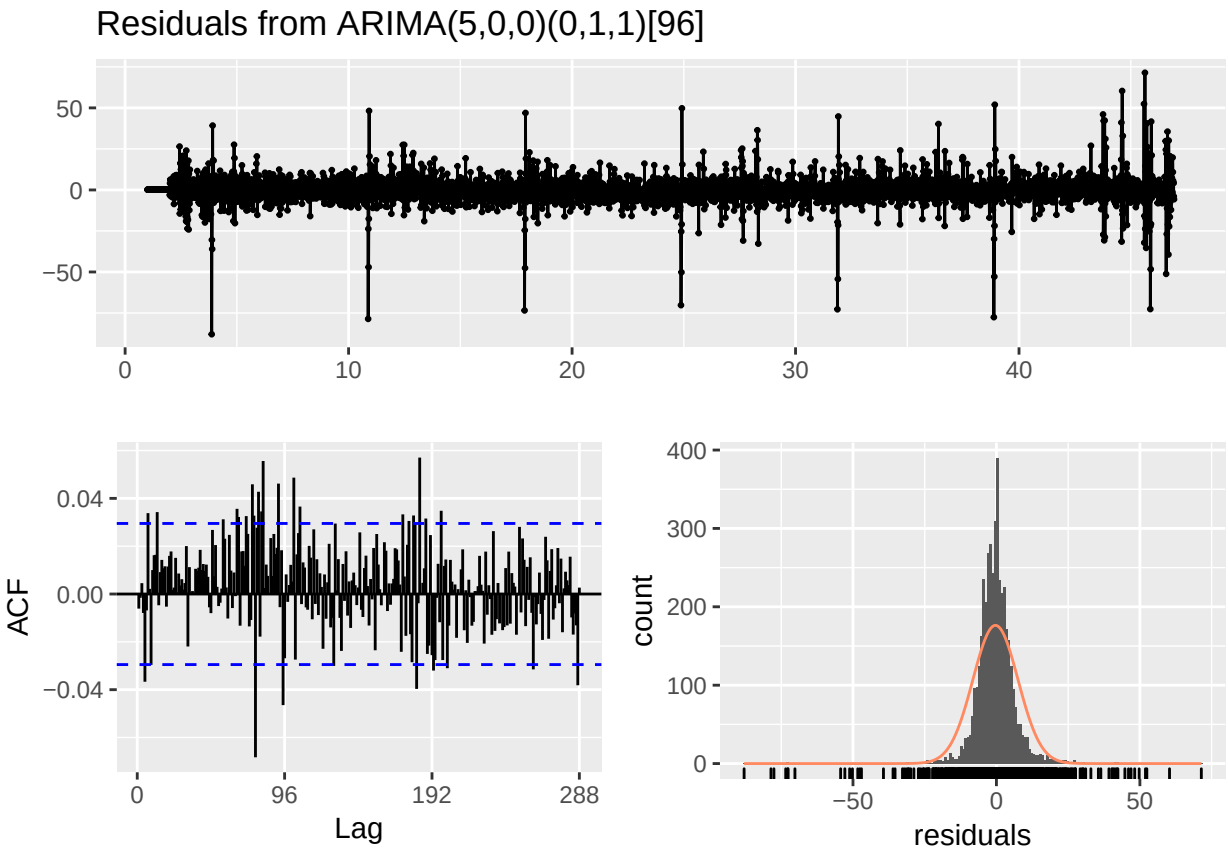
```
manual.sarima = Arima(power.train, order = c(5,0,0), seasonal = c(0,1,1))

pred.manual.sarima = forecast(manual.sarima, h = 96)

autoplot(power.test) + autolayer(pred.auto.sarima$mean, series = "Auto Arima") + autolayer(pred.auto.sa
autolayer(pred.manual.sarima$mean, series = "manual Sarima" )
```




```
RMSE.manual.sarima = RMSE.func(pred.manual.sarima, power.test)
RMSE.manual.sarima
```

```
## [1] 15.06212
```

```
checkresiduals(manual.sarima, test = "LB", lag = 100)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,1)[96]
## Q* = 183.93, df = 94, p-value = 8.683e-08
##
## Model df: 6.   Total lags used: 100
```

The noise is not white, but I do not see how to improve this without running high and non-supported values of p,q,P,Q.

Manual SARIMA considering the weekly seasonality

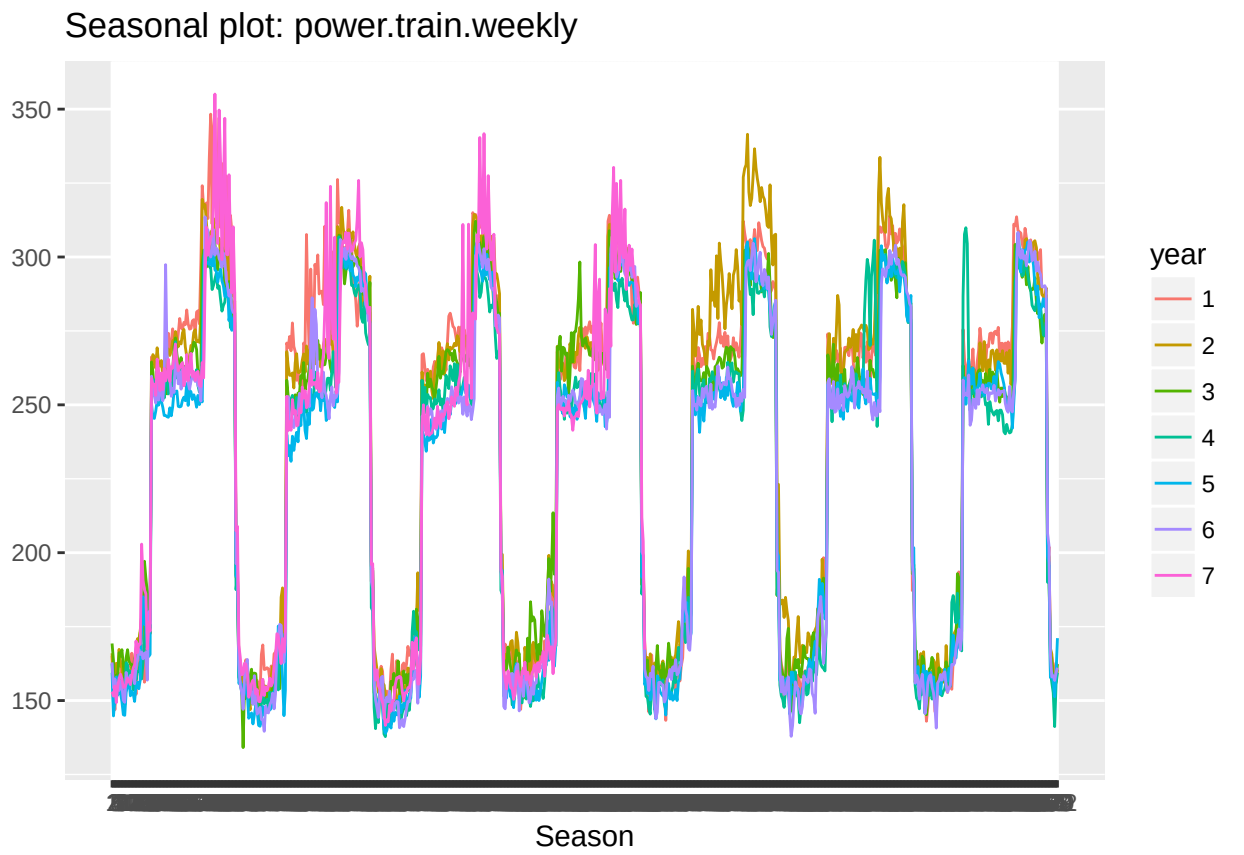
We see in the ACF a periodicity of 96 * 7, i.e. a weekly effect.

In light of the above, I try to set a frequency of 96*7.

```
power.train.weekly = ts(power.train, frequency = 672)
power.test.weekly = ts(power.test, frequency = 672)
```

Check new seasonality:

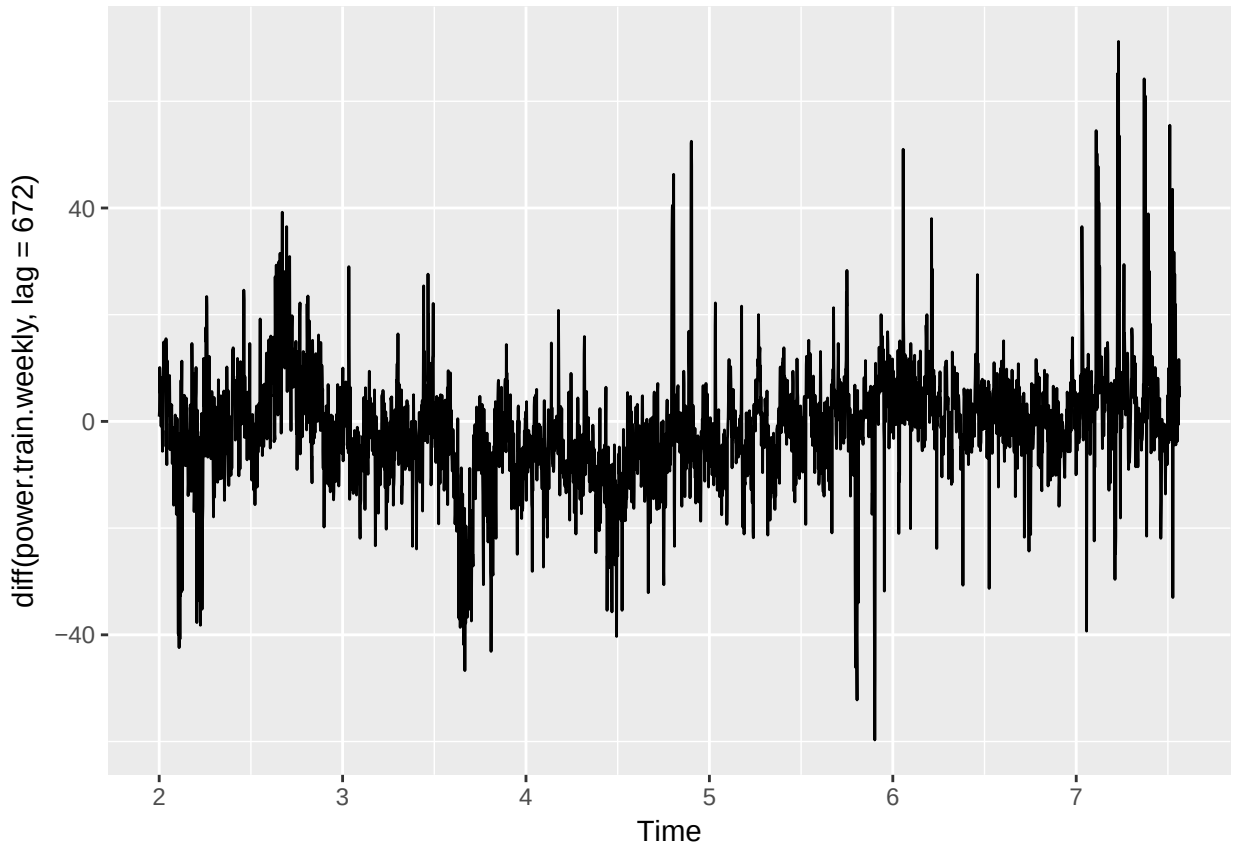
```
ggseasonplot(power.train.weekly)
```



From the above seasonal plot it is not so clear whether we should look at daily or weekly periodicity.

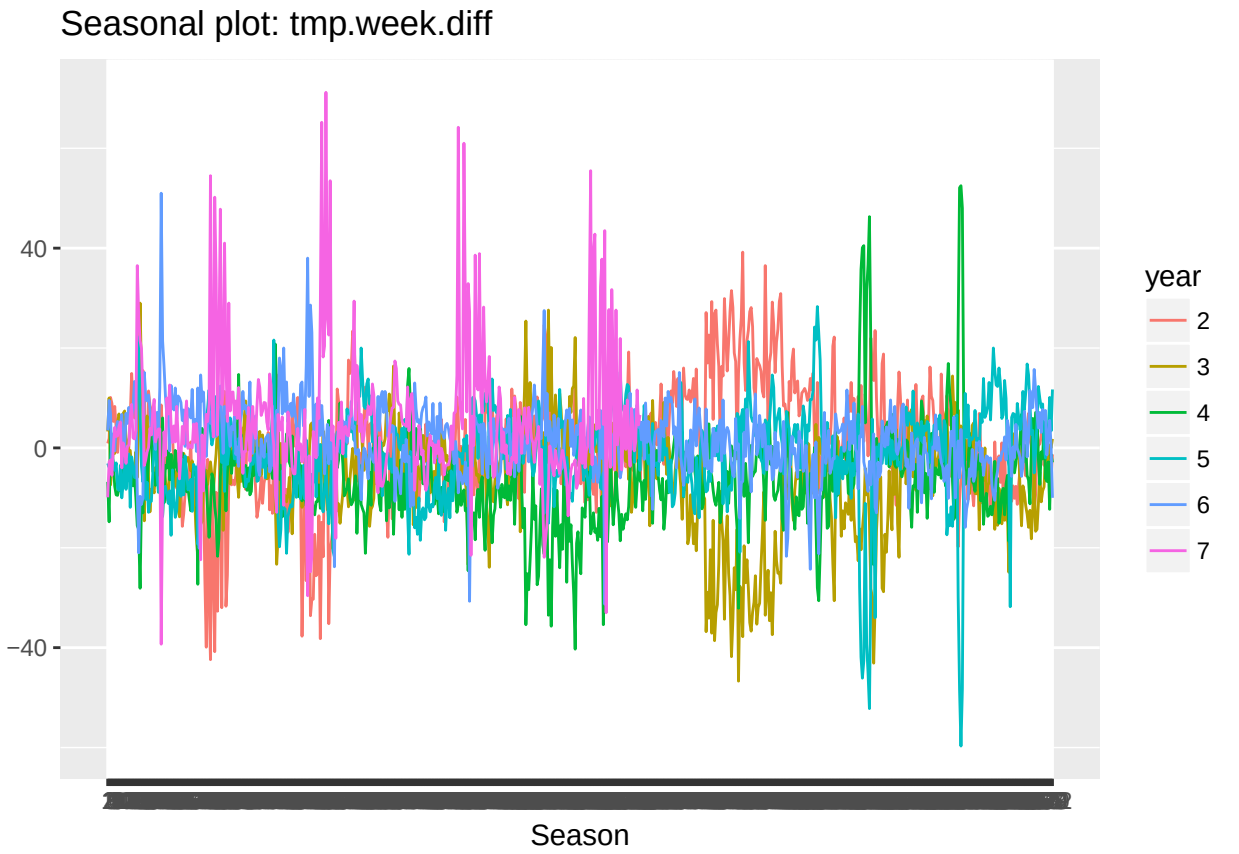
Try to differentiate the series by 97*6

```
autoplot(diff(power.train.weekly, lag = 672))
```

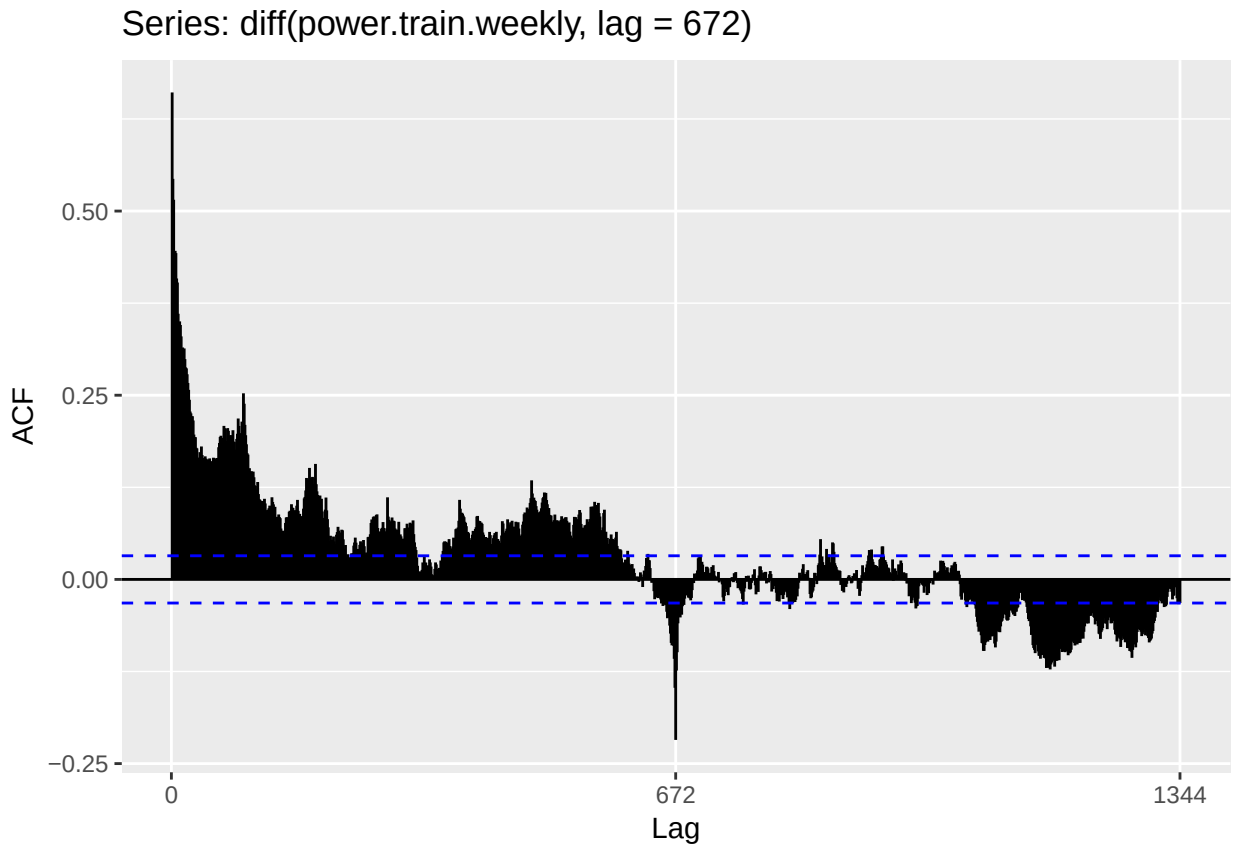


The noise does not show anymore the seasoneal weekly spikes.

```
tmp.week.diff = diff(power.train.weekly, lag = 672)

ggseasonplot(tmp.week.diff)
```




```
ggAcf(diff(power.train.weekly, lag = 672))
```



The noise seems to be more stationary than before, even though there are still correlations to be catpured.

So I would like to test a SARIMA model taking into account the weekly seasonality. Based on acf and pacf:

- acf:
  - non-seasonal: exp decrease
  - seasonal: has exponential decrease
- pacf:
  - non seasonal: 0 after a p which is larger than 100 ==> too high, R will throw an error
  - seasonal: 0 after 672

Also ARIMA in R does not support such a big frequency of 96*7.

In light of that, we cannot build a SARIMA with a weekly periodicity. we have to accept the manual SARIMA we found, i.e. SARIMA(5,0,0)(0,1,1)[96].

Seasonal NN

```
fit=nnetar(power.train)
print(fit)
```

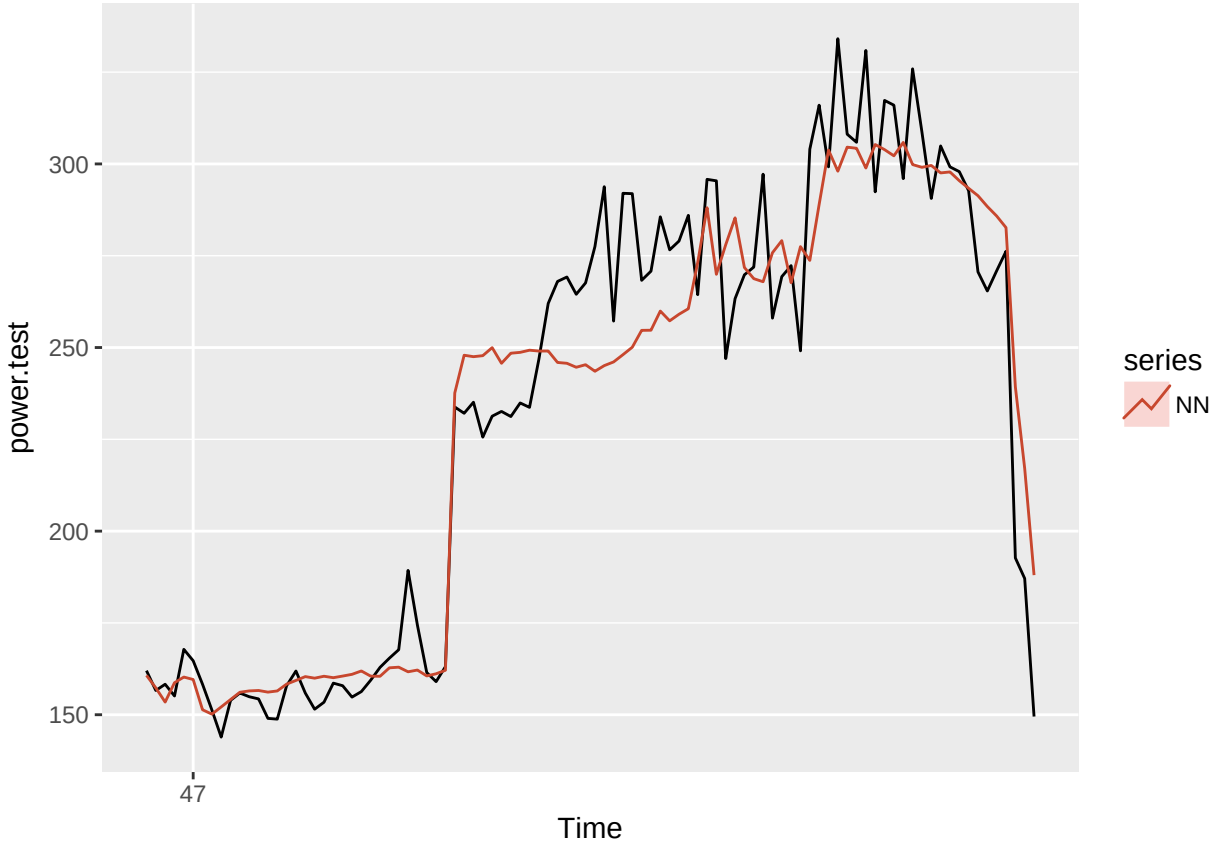
```
## Series: power.train
## Model:  NNAR(11,1,6)[96]
## Call:   nnetar(y = power.train)
##
## Average of 20 networks, each of which is
## a 12-6-1 network with 85 weights
## options were - linear output units
```

```
##
## sigma^2 estimated as 78.11
```

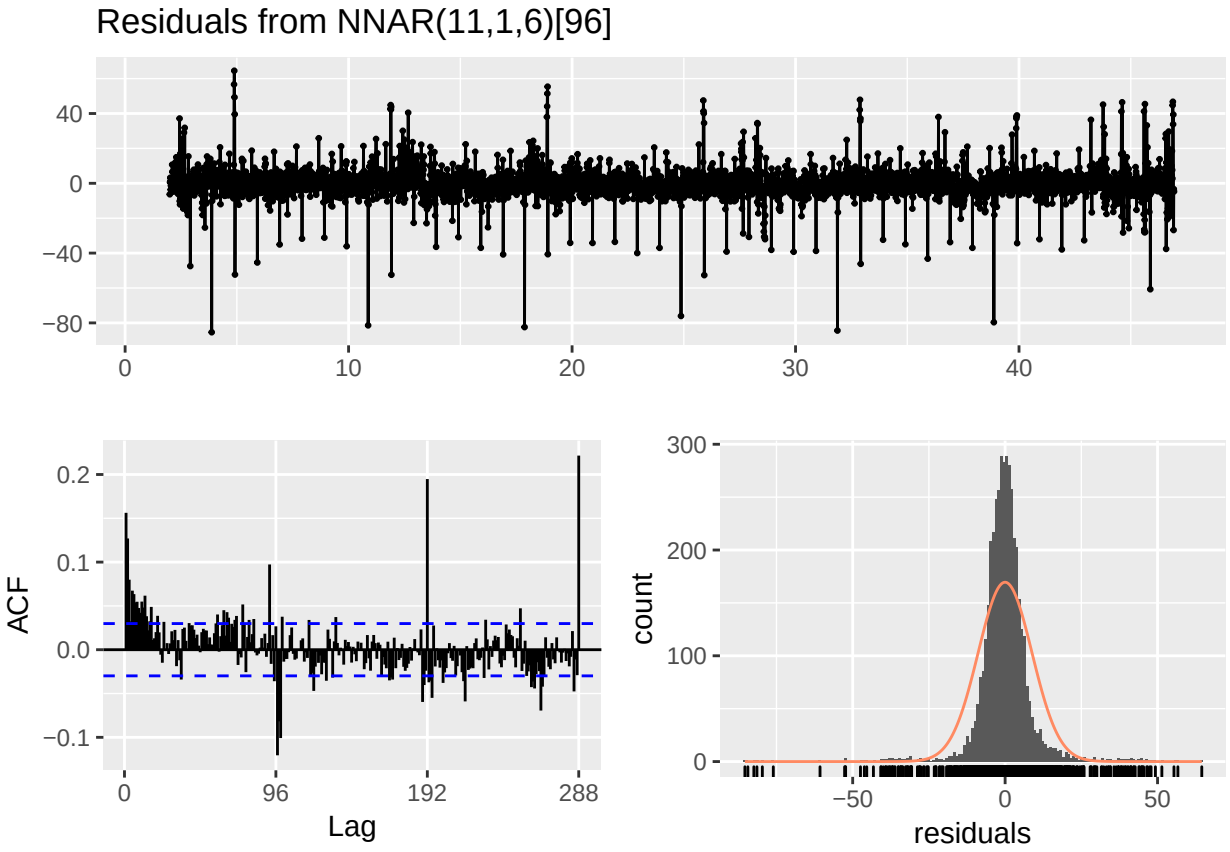
```
pred.NN = forecast(fit, h = 96)
RMSE.NN = RMSE.func(pred.NN, power.test)
RMSE.NN
```

```
## [1] 18.06576
```

```
autoplot(power.test) + autolayer(pred.NN, series = "NN")
```




```
checkresiduals(fit)
```



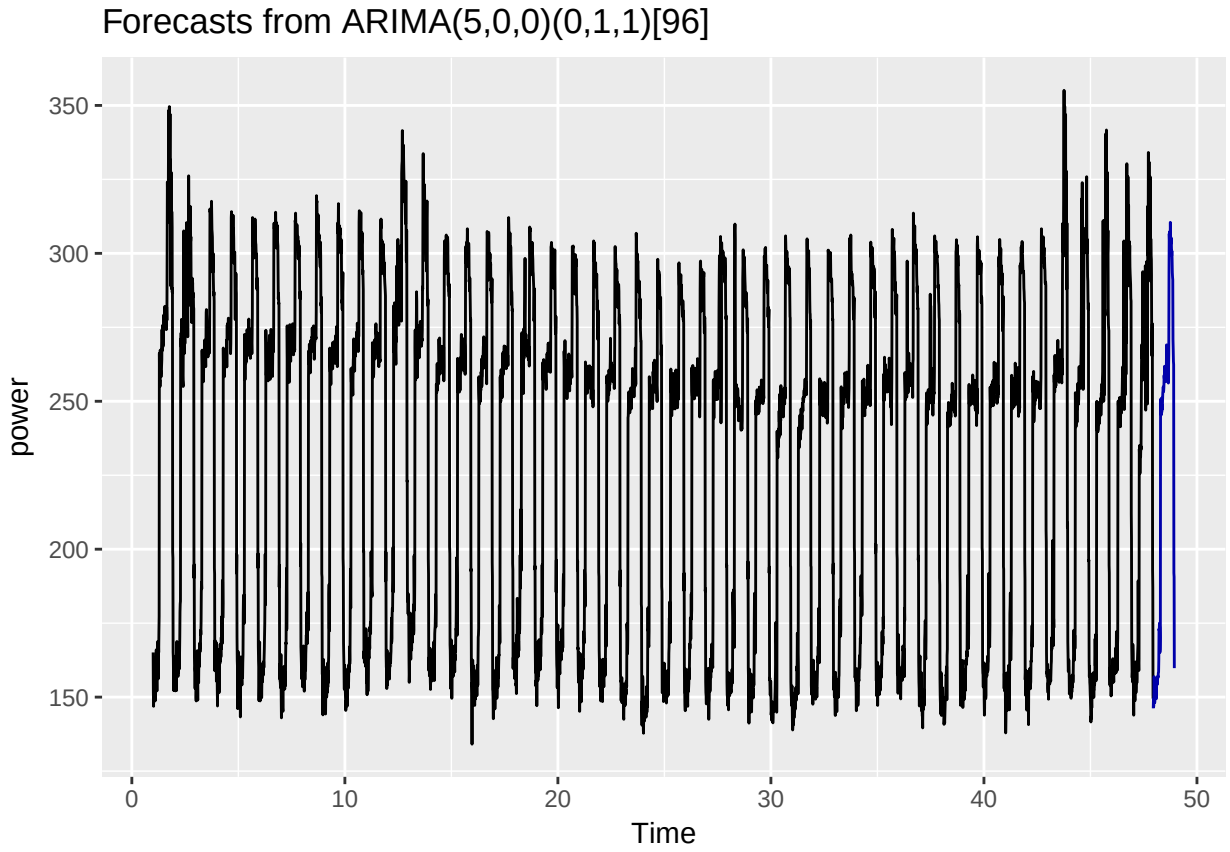
Prediction with the best model

Based on the what we found so far, the best model without considering temperature is the SARIMA

Arima(power.train, order = c(5,0,0), seasonal = c(0,1,1)) with periodicity of 96.

So I will retrain it on the entire dataset and then will predict the values of power for 2/17/2010 at 00:00 until 2/17/2010 23:45.

```
final.model = Arima( power, order = c(5,0,0), seasonal = c(0,1,1) )

pred.final.model = forecast(final.model, h = 96)


autoplot(pred.final.model, PI = FALSE)
```




```
setwd("D:/BIG_DATA/DSTI/OneDrive - Data ScienceTech Institute/2020-09-Time-series/assignment")

#write.csv(pred.final.model$mean, file = "AndreaSonnellini.csv", row.names = FALSE, col.names = c("powe
```

Forecast using temperature

```
temp = elCons[,"Temp"]

temp.ts = ts(temp, frequency = 96)

temp.ts.train = head(temp.ts, n = length(temp.ts) - 96)

temp.ts.test = tail(temp.ts, n = 96)
```

Auto dynamic regression model with temp

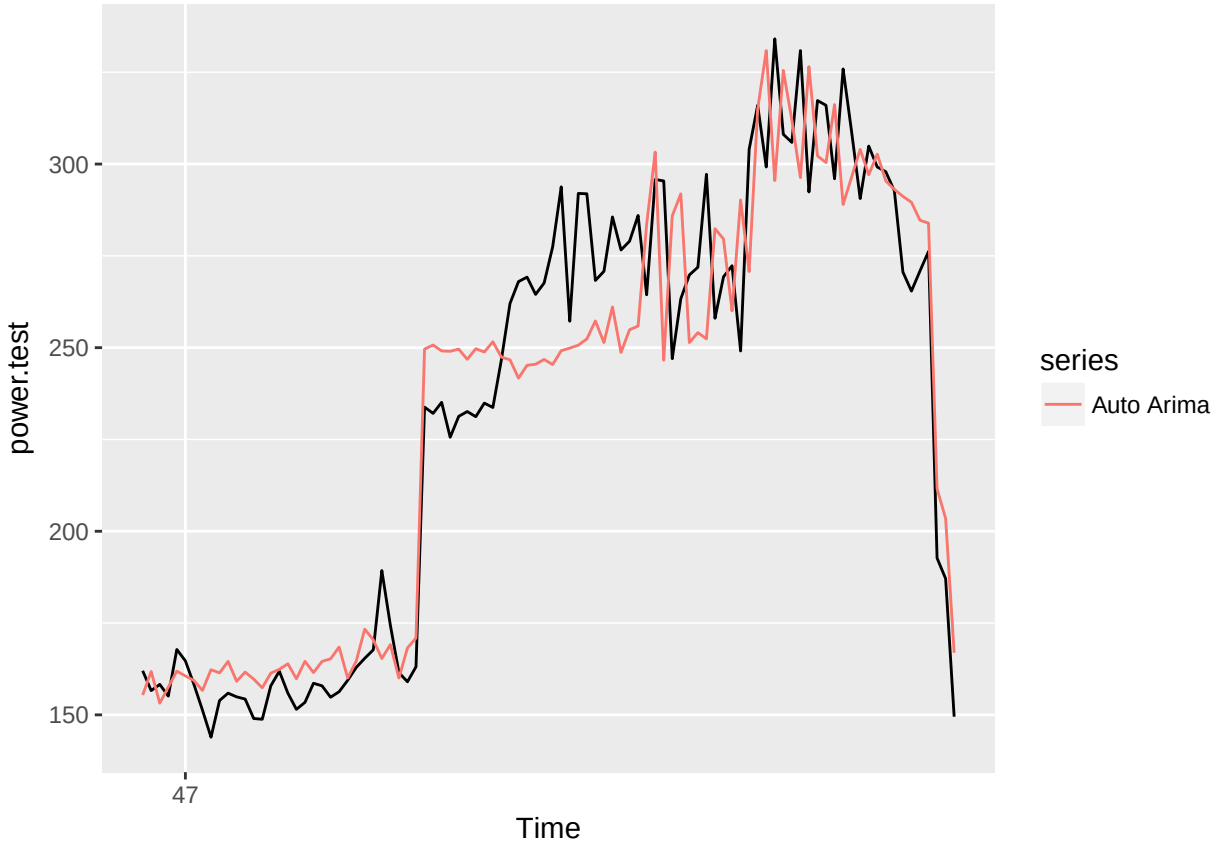
```
auto.sarima.temp = auto.arima(power.train, xreg = temp.ts.train)
```

```
print(auto.sarima.temp)
```

```
## Series: power.train
## Regression with ARIMA(5,0,4)(0,1,0)[96] errors
##
## Coefficients:
##          ar1      ar2      ar3      ar4     ar5     ma1     ma2     ma3
```

```
##       0.0642  -0.0991  -0.0192  -0.0094  0.2539  0.6496  0.6304  0.6293
## s.e.  0.1754   0.0705   0.0557   0.0535  0.0542  0.1758  0.1090  0.0802
##          ma4    xreg
##       0.2999  0.5531
## s.e.  0.0936  0.2265
##
## sigma^2 estimated as 111.9:  log likelihood=-16297.21
## AIC=32616.42   AICc=32616.48   BIC=32686.49
```

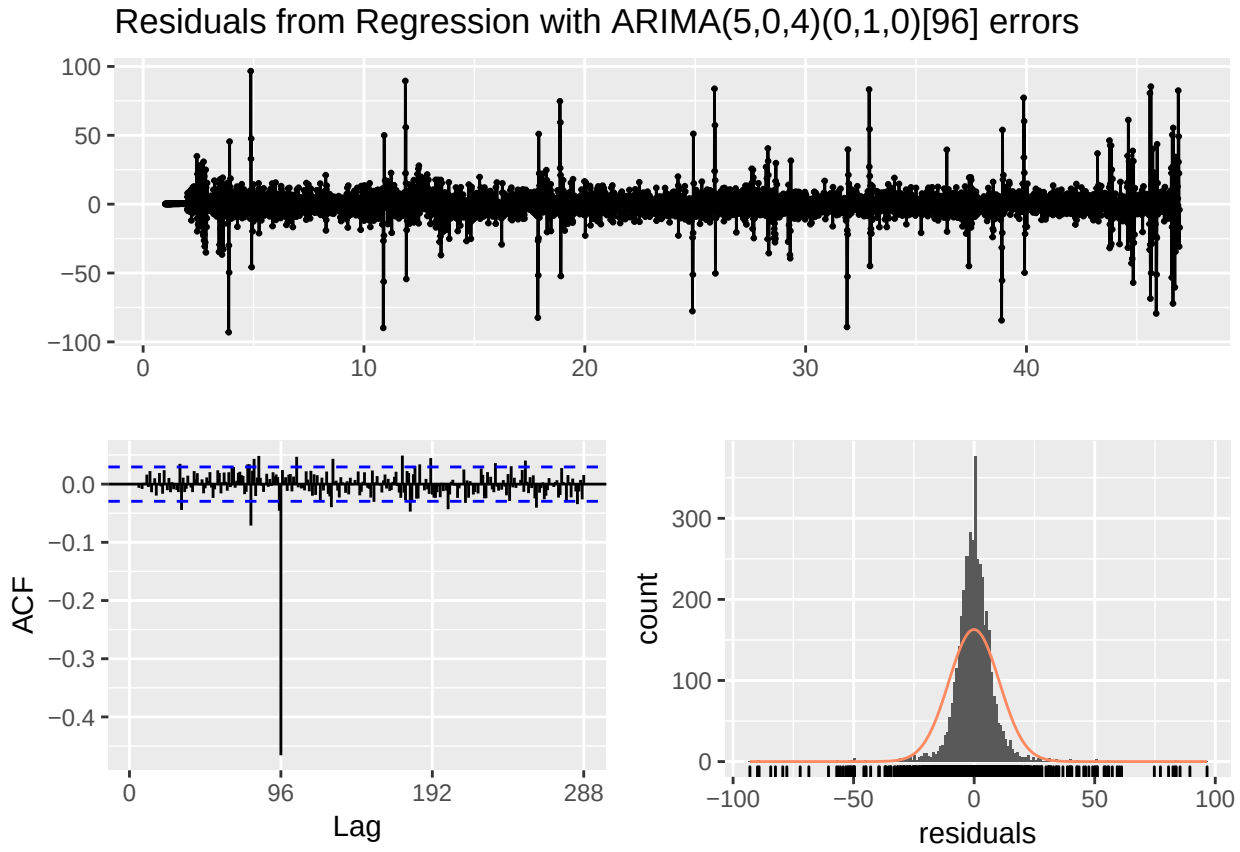
```
pred.auto.sarima.temp = forecast(auto.sarima.temp, h = 96,xreg = temp.ts.test)

autoplot(power.test) + autolayer(pred.auto.sarima.temp$mean, series = "Auto Arima")
```




```
RMSE.auto.sarima.temp = RMSE.func(pred.auto.sarima.temp, power.test)
RMSE.auto.sarima.temp
```

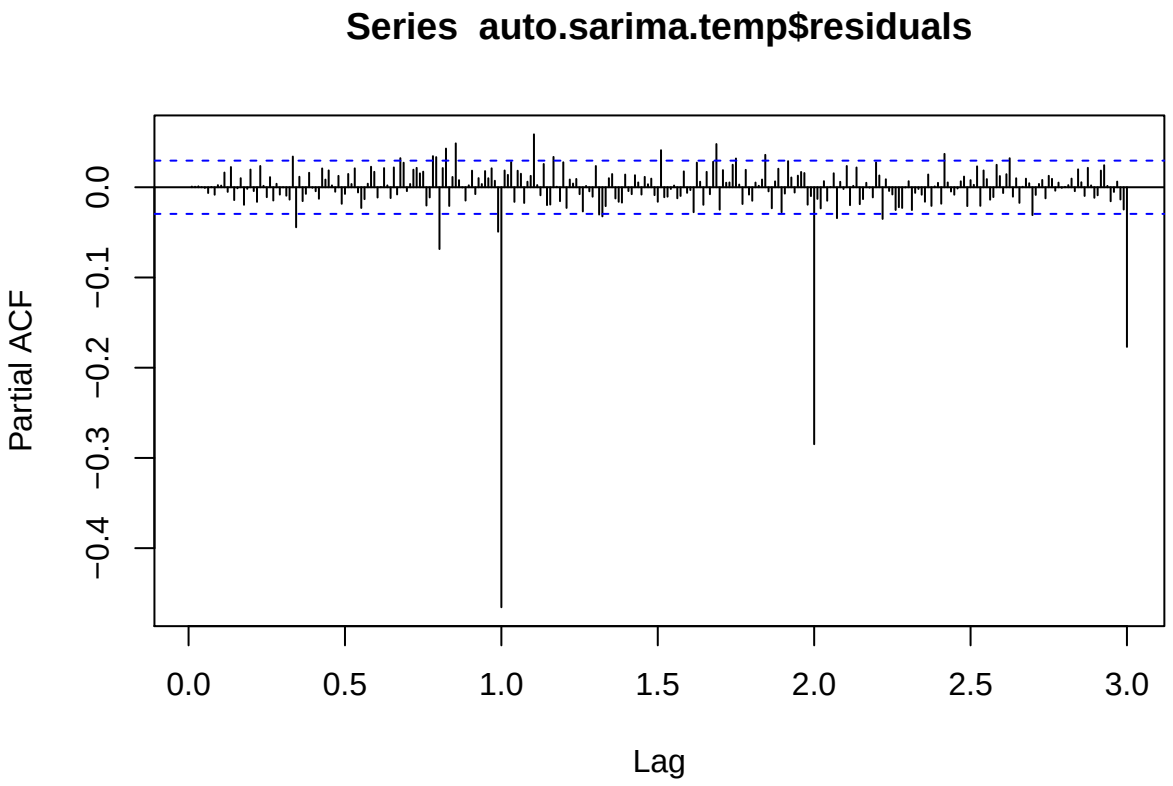
```
## [1] 20.03835
```

```
checkresiduals(auto.sarima.temp)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,4)(0,1,0)[96] errors
## Q* = 1279.4, df = 182, p-value < 2.2e-16
##
## Model df: 10.   Total lags used: 192
```

```
plot(pacf(auto.sarima.temp$residuals,lag.max = 288))
```



Like for the case without temperature, the residuals seems not to be white noise.

Looking at the ACF and PACF we see we could add an MA1 seasonal term.

```
auto.sarima.temp.amended = Arima( power.train, xreg = temp.ts.train ,order = c(5,0,4), seasonal = c(0,1

pred.auto.sarima.temp.amended = forecast(auto.sarima.temp.amended, xreg = temp.ts.test, h = 96)
```

```
RMSE.auto.sarima.temp.amended = RMSE.func(pred.auto.sarima.temp.amended, power.test)
RMSE.auto.sarima.temp.amended
```

```
## [1] 14.80175
```

Timeseries regression model & manually chosen dynamic model

```
tsrm.temp = tslm(power.train ~ temp.ts.train + trend + season)
summary(tsrm.temp)
```

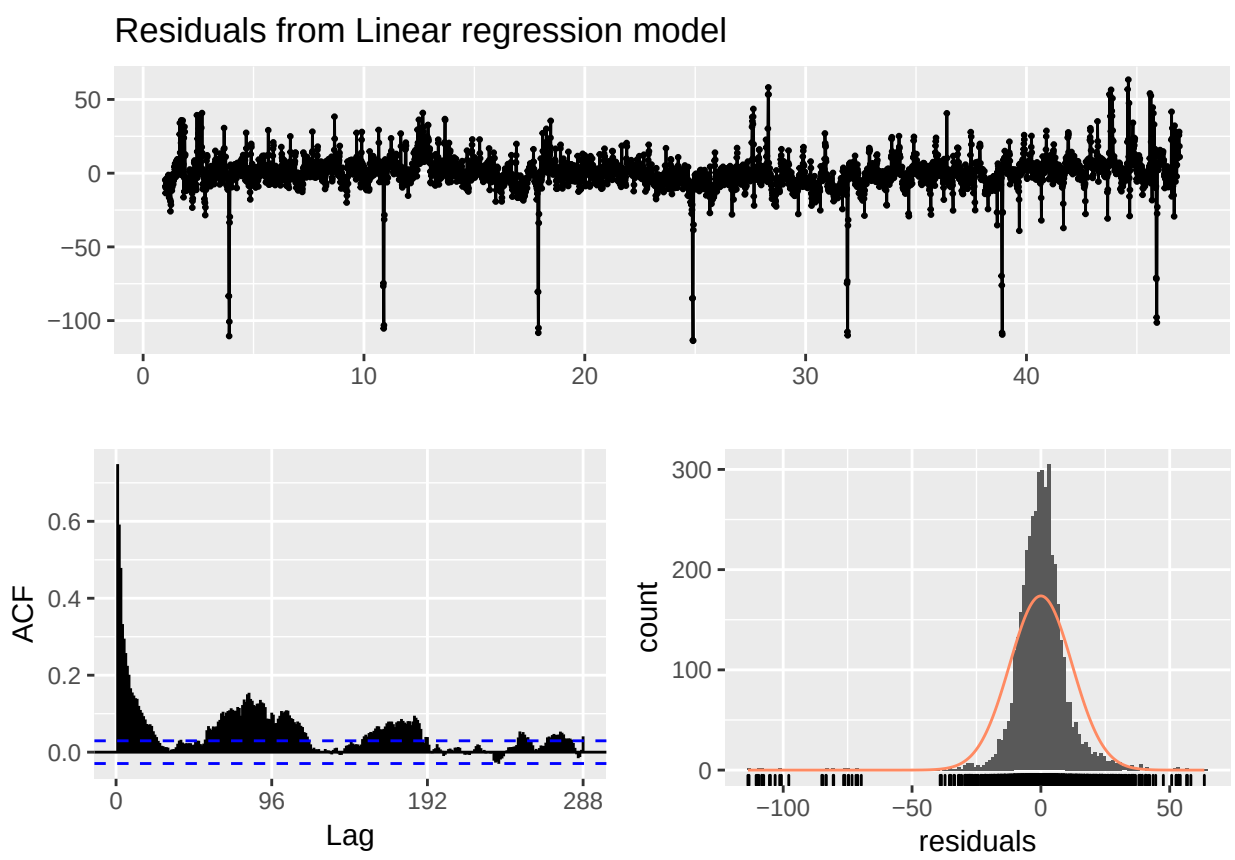
```
##
## Call:
## tslm(formula = power.train ~ temp.ts.train + trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -113.691   -4.999    0.046    4.847   63.435
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    1.563e+02  2.006e+00  77.888  < 2e-16 ***
## temp.ts.train  1.278e+00  9.568e-02  13.356  < 2e-16 ***
## trend         -3.697e-03  1.507e-04 -24.530  < 2e-16 ***
## season2       -9.151e+00  2.555e+00  -3.581 0.000346 ***
## season3       -9.314e+00  2.555e+00  -3.645 0.000271 ***
## season4       -5.991e+00  2.555e+00  -2.344 0.019101 *
## season5       -6.371e+00  2.556e+00  -2.493 0.012704 *
## season6       -3.587e+00  2.556e+00  -1.404 0.160529
## season7       -4.620e+00  2.556e+00  -1.808 0.070697 .
## season8       -3.114e+00  2.556e+00  -1.219 0.223063
## season9       -7.453e+00  2.556e+00  -2.916 0.003563 **
## season10      -8.834e+00  2.556e+00  -3.456 0.000553 ***
## season11      -3.300e+00  2.556e+00  -1.291 0.196717
## season12      -1.638e+00  2.556e+00  -0.641 0.521728
## season13      -2.429e+00  2.557e+00  -0.950 0.342155
## season14      -3.345e+00  2.557e+00  -1.308 0.190852
## season15      -1.763e+00  2.557e+00  -0.689 0.490562
## season16      -2.115e+00  2.557e+00  -0.827 0.408009
## season17      -1.148e+00  2.557e+00  -0.449 0.653386
## season18      -3.316e-01  2.557e+00  -0.130 0.896828
## season19       1.687e+00  2.557e+00   0.660 0.509357
## season20       1.211e+00  2.557e+00   0.473 0.635917
## season21       4.666e+00  2.557e+00   1.825 0.068071 .
## season22       1.229e+01  2.557e+00   4.805 1.60e-06 ***
## season23       1.359e+01  2.557e+00   5.316 1.11e-07 ***
## season24       1.383e+01  2.557e+00   5.410 6.63e-08 ***
## season25       1.887e+01  2.557e+00   7.379 1.90e-13 ***
## season26       1.766e+01  2.557e+00   6.909 5.61e-12 ***
## season27       1.323e+01  2.557e+00   5.174 2.40e-07 ***
## season28       1.693e+01  2.557e+00   6.622 3.99e-11 ***
## season29       1.007e+02  2.556e+00  39.385  < 2e-16 ***
## season30       9.851e+01  2.556e+00  38.533  < 2e-16 ***
## season31       9.591e+01  2.556e+00  37.515  < 2e-16 ***
## season32       9.534e+01  2.556e+00  37.293  < 2e-16 ***
## season33       9.837e+01  2.556e+00  38.490  < 2e-16 ***
## season34       9.304e+01  2.556e+00  36.403  < 2e-16 ***
## season35       9.517e+01  2.556e+00  37.240  < 2e-16 ***
## season36       9.608e+01  2.556e+00  37.593  < 2e-16 ***
## season37       9.236e+01  2.559e+00  36.088  < 2e-16 ***
## season38       9.326e+01  2.559e+00  36.441  < 2e-16 ***
## season39       9.463e+01  2.559e+00  36.977  < 2e-16 ***
## season40       9.499e+01  2.559e+00  37.119  < 2e-16 ***
## season41       9.668e+01  2.566e+00  37.674  < 2e-16 ***
## season42       9.480e+01  2.566e+00  36.942  < 2e-16 ***
## season43       9.549e+01  2.566e+00  37.210  < 2e-16 ***
## season44       9.611e+01  2.566e+00  37.454  < 2e-16 ***
## season45       9.570e+01  2.576e+00  37.155  < 2e-16 ***
## season46       9.905e+01  2.576e+00  38.458  < 2e-16 ***
## season47       9.800e+01  2.576e+00  38.050  < 2e-16 ***
## season48       9.640e+01  2.576e+00  37.427  < 2e-16 ***
## season49       9.789e+01  2.582e+00  37.909  < 2e-16 ***
## season50       9.815e+01  2.582e+00  38.010  < 2e-16 ***
## season51       9.785e+01  2.582e+00  37.894  < 2e-16 ***
## season52       9.657e+01  2.582e+00  37.399  < 2e-16 ***
```

```
## season53       9.680e+01  2.590e+00  37.378  < 2e-16 ***
## season54       9.740e+01  2.590e+00  37.611  < 2e-16 ***
## season55       9.767e+01  2.590e+00  37.714  < 2e-16 ***
## season56       9.835e+01  2.590e+00  37.978  < 2e-16 ***
## season57       9.803e+01  2.592e+00  37.815  < 2e-16 ***
## season58       9.803e+01  2.592e+00  37.816  < 2e-16 ***
## season59       9.709e+01  2.592e+00  37.453  < 2e-16 ***
## season60       9.698e+01  2.592e+00  37.410  < 2e-16 ***
## season61       9.651e+01  2.586e+00  37.325  < 2e-16 ***
## season62       9.830e+01  2.586e+00  38.016  < 2e-16 ***
## season63       9.641e+01  2.586e+00  37.284  < 2e-16 ***
## season64       9.460e+01  2.586e+00  36.585  < 2e-16 ***
## season65       1.142e+02  2.575e+00  44.335  < 2e-16 ***
## season66       1.267e+02  2.575e+00  49.223  < 2e-16 ***
## season67       1.392e+02  2.575e+00  54.058  < 2e-16 ***
## season68       1.413e+02  2.575e+00  54.891  < 2e-16 ***
## season69       1.396e+02  2.566e+00  54.410  < 2e-16 ***
## season70       1.384e+02  2.566e+00  53.953  < 2e-16 ***
## season71       1.380e+02  2.566e+00  53.775  < 2e-16 ***
## season72       1.377e+02  2.566e+00  53.686  < 2e-16 ***
## season73       1.437e+02  2.563e+00  56.043  < 2e-16 ***
## season74       1.407e+02  2.563e+00  54.877  < 2e-16 ***
## season75       1.380e+02  2.563e+00  53.849  < 2e-16 ***
## season76       1.371e+02  2.563e+00  53.487  < 2e-16 ***
## season77       1.384e+02  2.561e+00  54.038  < 2e-16 ***
## season78       1.349e+02  2.561e+00  52.694  < 2e-16 ***
## season79       1.345e+02  2.561e+00  52.539  < 2e-16 ***
## season80       1.347e+02  2.561e+00  52.602  < 2e-16 ***
## season81       1.323e+02  2.558e+00  51.696  < 2e-16 ***
## season82       1.305e+02  2.558e+00  51.005  < 2e-16 ***
## season83       1.292e+02  2.558e+00  50.492  < 2e-16 ***
## season84       1.275e+02  2.558e+00  49.841  < 2e-16 ***
## season85       1.103e+02  2.557e+00  43.154  < 2e-16 ***
## season86       1.085e+02  2.557e+00  42.444  < 2e-16 ***
## season87       1.028e+02  2.557e+00  40.210  < 2e-16 ***
## season88       1.032e+02  2.557e+00  40.373  < 2e-16 ***
## season89       2.833e+01  2.556e+00  11.081  < 2e-16 ***
## season90       2.995e+01  2.556e+00  11.715  < 2e-16 ***
## season91       5.943e-01  2.556e+00   0.233 0.816160
## season92      -2.636e+00  2.570e+00  -1.026 0.305112
## season93      -3.087e+00  2.570e+00  -1.201 0.229657
## season94      -8.937e+00  2.570e+00  -3.478 0.000510 ***
## season95      -3.033e+00  2.570e+00  -1.180 0.237916
## season96       6.070e-03  2.570e+00   0.002 0.998115
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.26 on 4313 degrees of freedom
## Multiple R-squared:  0.9555, Adjusted R-squared:  0.9545
## F-statistic: 955.7 on 97 and 4313 DF,  p-value: < 2.2e-16
```

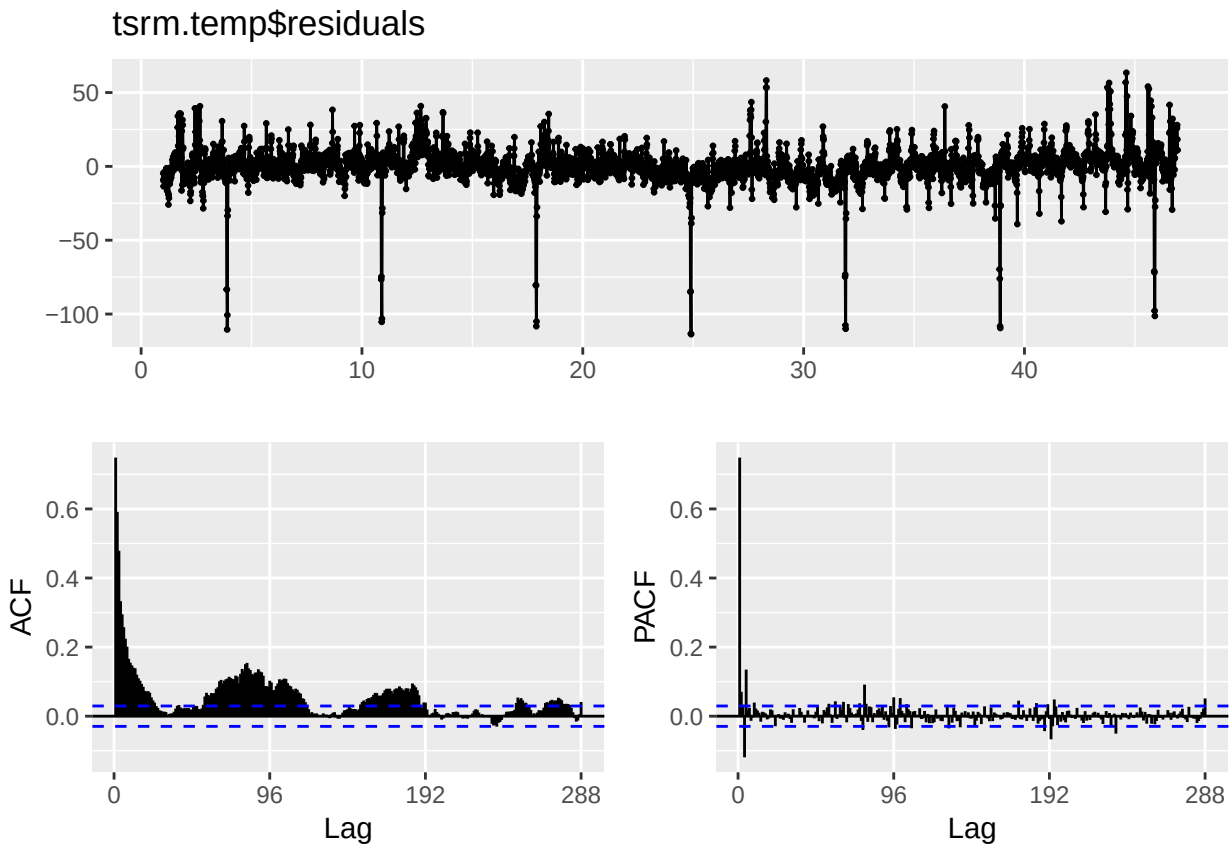
Check if the linear model above is ok in terms of residuals being white noise.

```
checkresiduals(tsrm.temp,test="LB",plot=TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 11282, df = 94, p-value < 2.2e-16
##
## Model df: 98.   Total lags used: 192
```

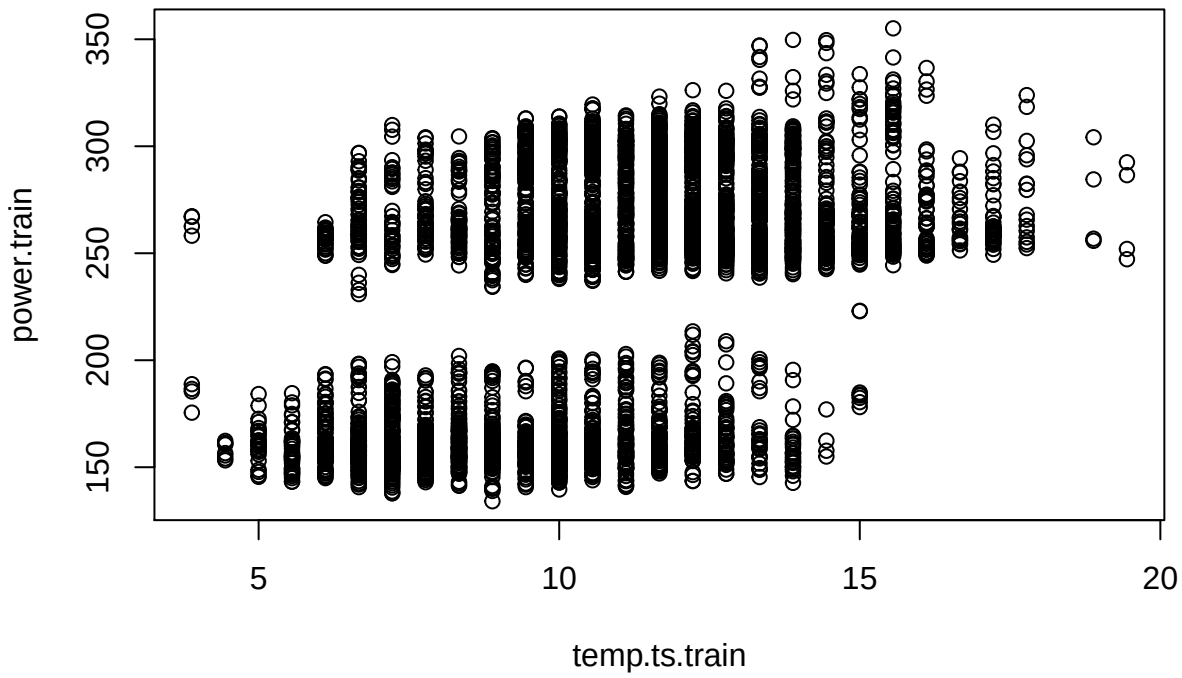
```
ggtsdisplay(tsrm.temp$residuals)
```



There are still some correlations to be captured.

First let's check the relationship between power and temperature

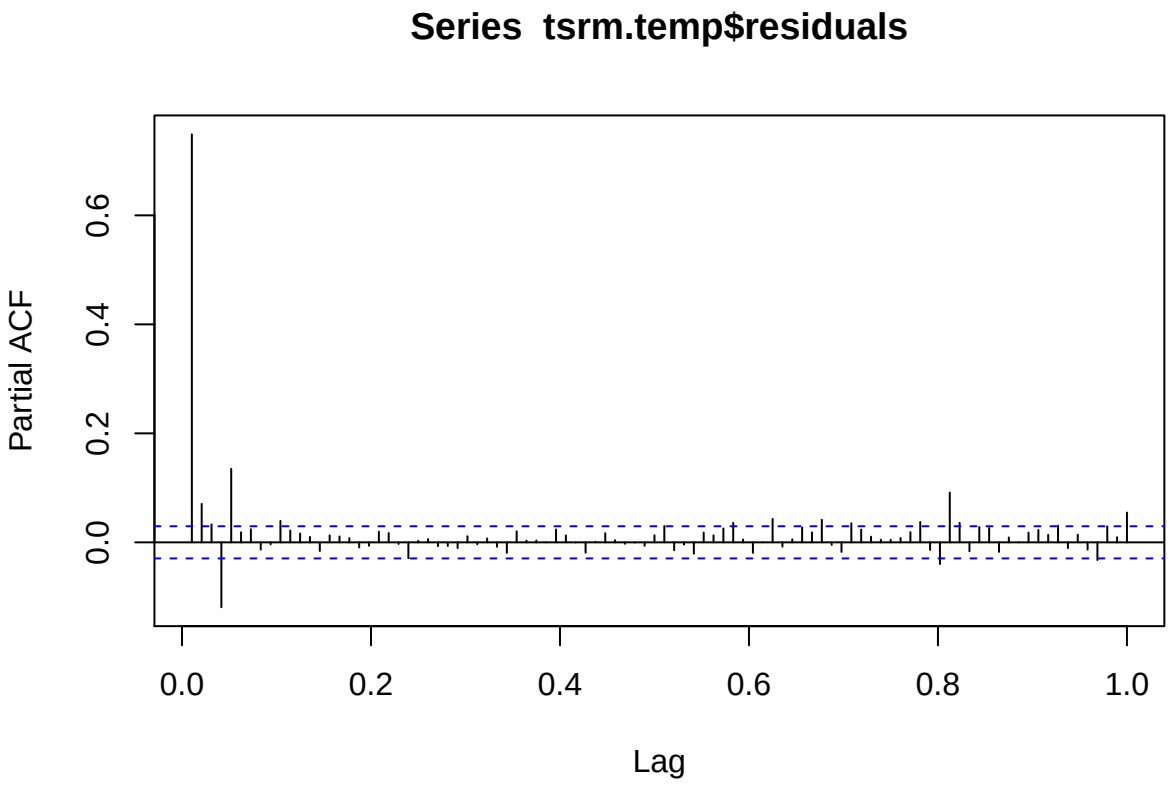
```
plot(temp.ts.train, power.train)
```



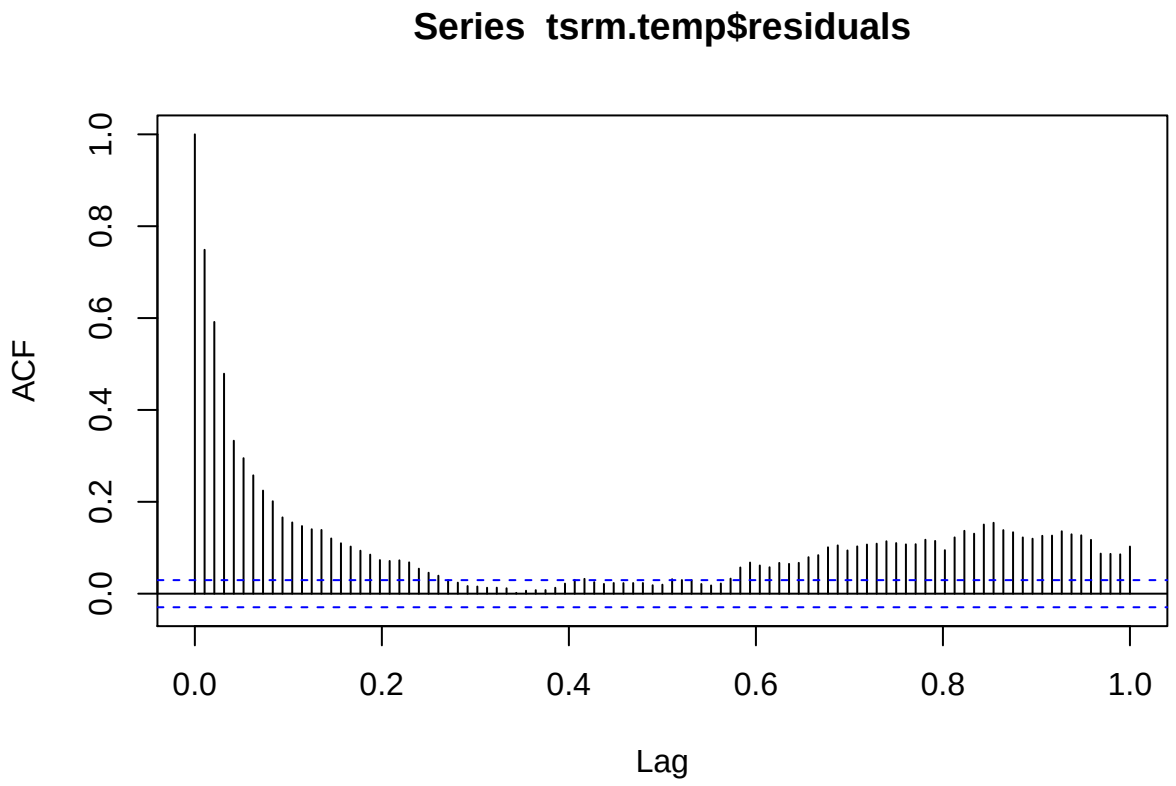
No clear relationship between power and temperature. They really seems to be uncorrelated.

Checking again the residuals, zooming on pacf

```
pacf(tsrm.temp$residuals, lag.max = 96)
```



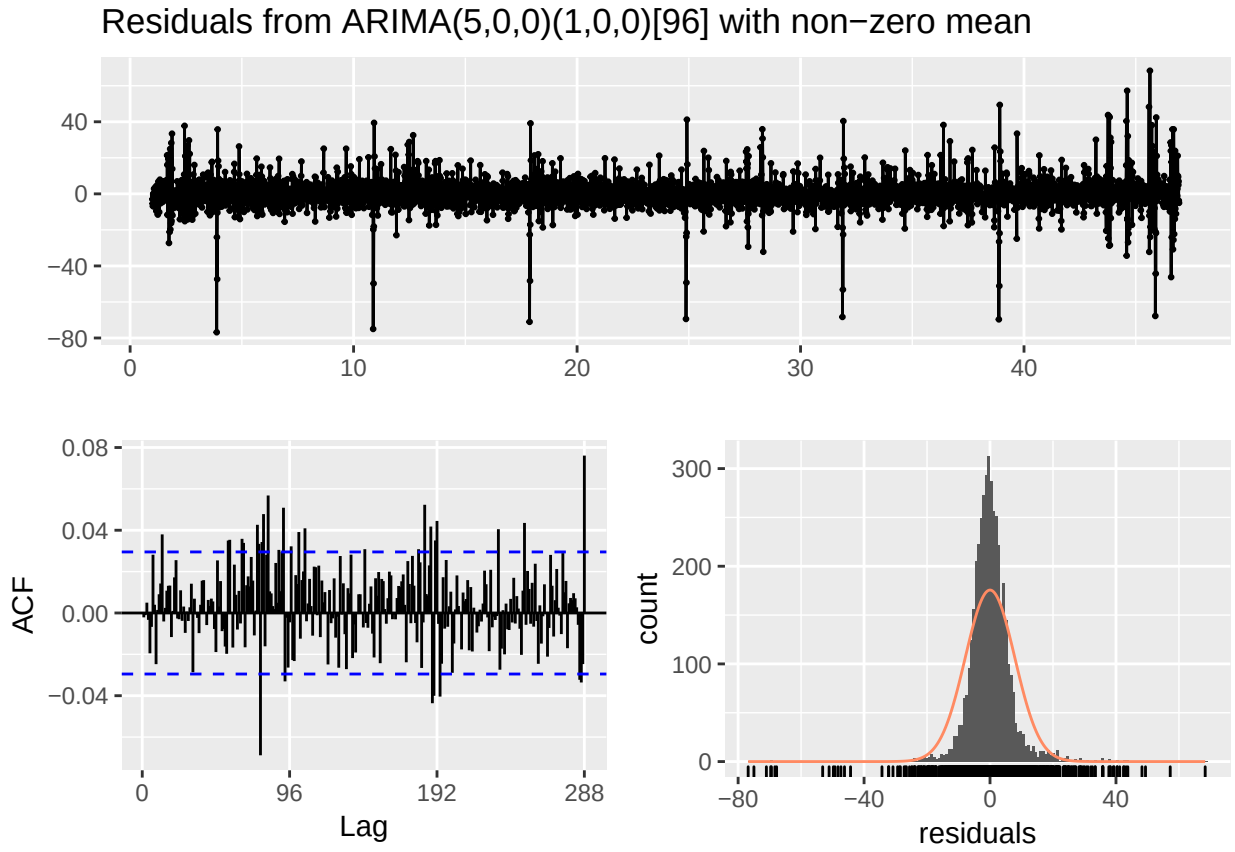

```
acf(tsrm.temp$residuals, lag.max = 96)
```



PACF non seasonal = 0 after 78, but I will consider only short-term correlations and choose 5. ACF non seasonal = exp decrease

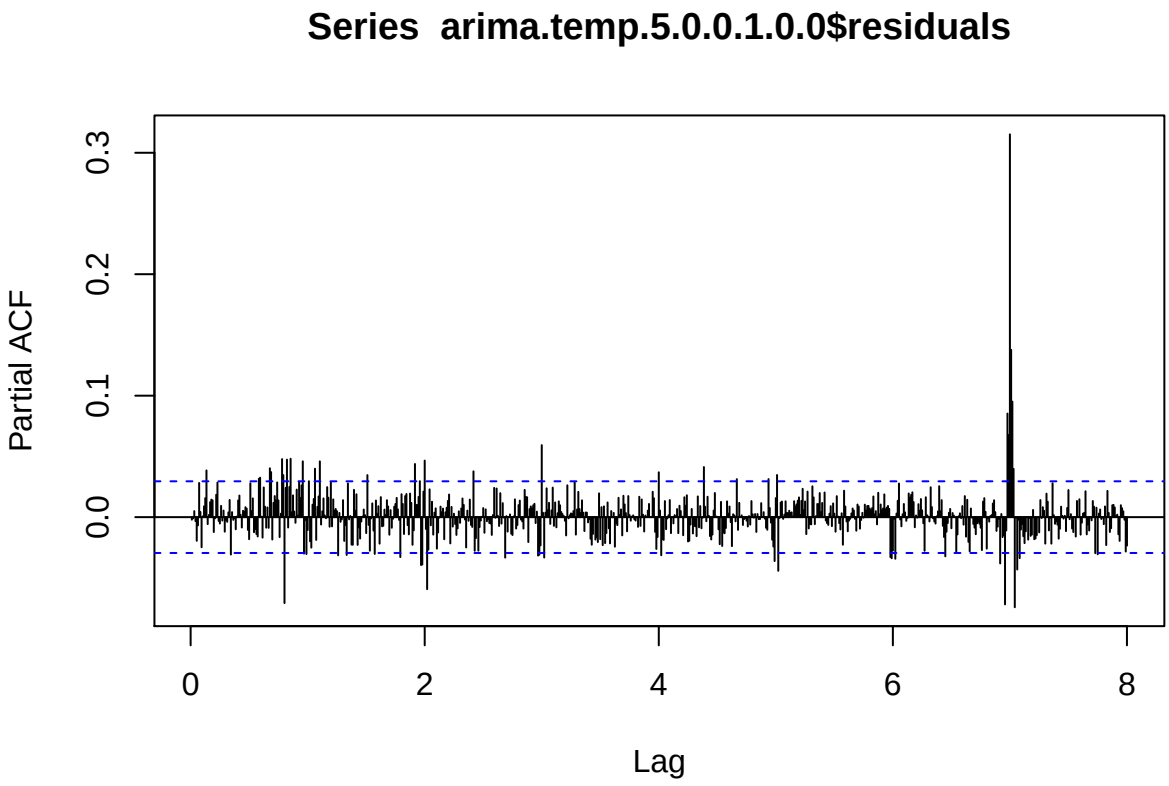
PACF seasonal = 1

```
res.arima.temp = tsrm.temp$residuals

arima.temp.5.0.0.1.0.0 = Arima( res.arima.temp, order = c(5,0,0), seasonal = c(1,0,0))

checkresiduals(arima.temp.5.0.0.1.0.0)
```

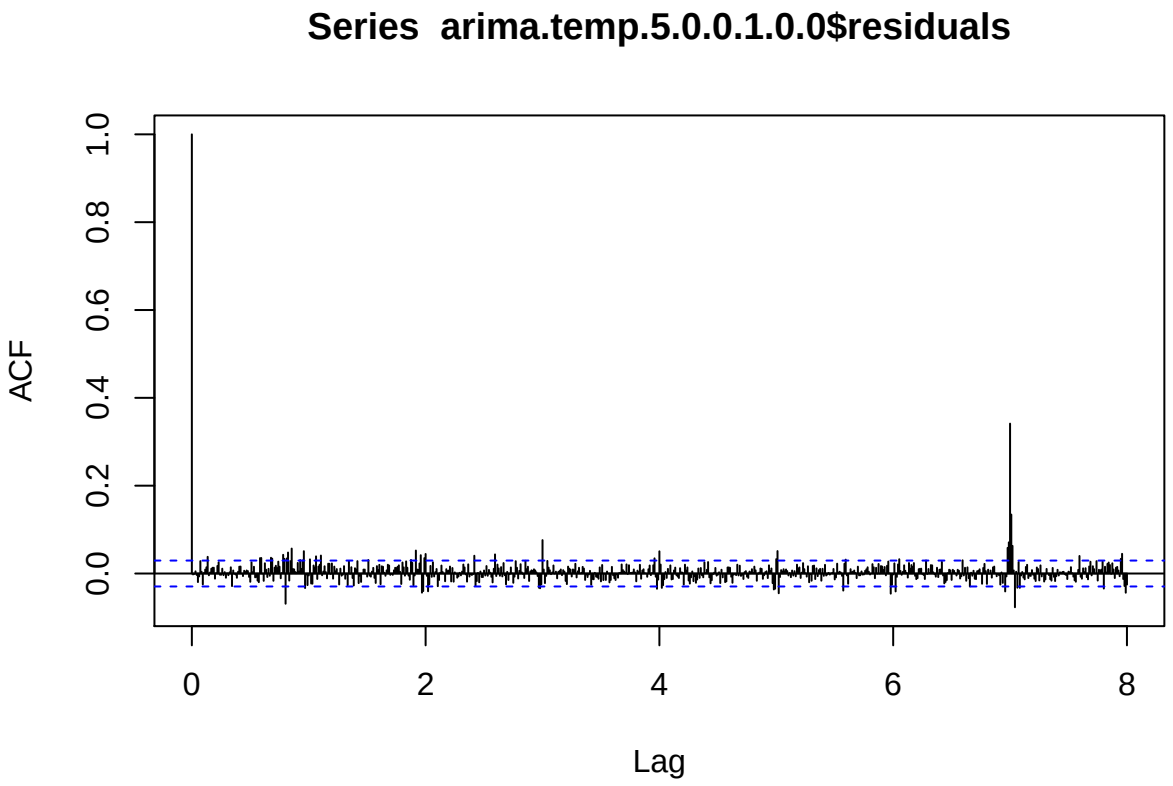



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(1,0,0)[96] with non-zero mean
## Q* = 348.83, df = 185, p-value = 3.653e-12
##
## Model df: 7.   Total lags used: 192
```

```
pacf(arima.temp.5.0.0.1.0.0$residuals, lag.max = 96*8)
```



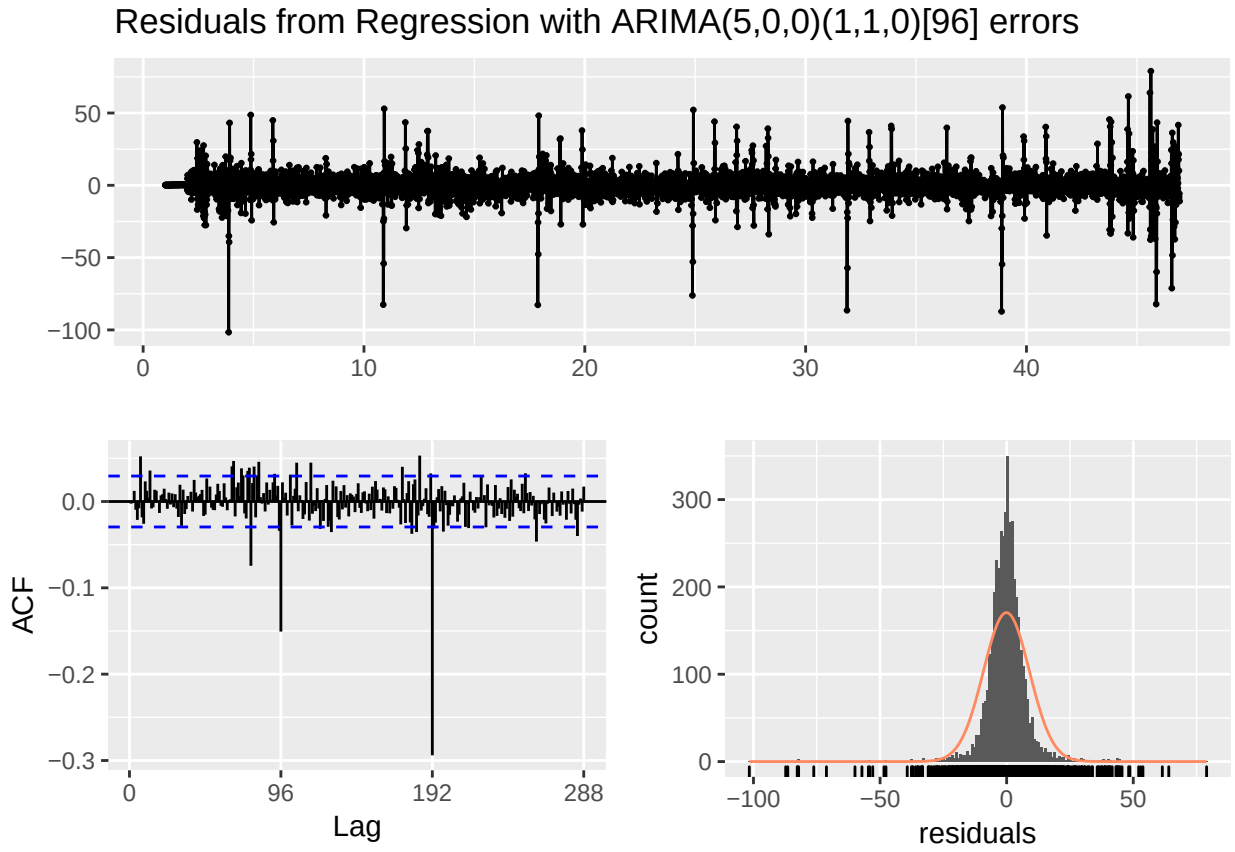

```
acf(arima.temp.5.0.0.1.0.0$residuals, lag.max = 96*8)
```



Based on the above residuals I would add an MA7 or AR7 seasonal term but this will lead to an error in R related to the fact that ARIMA implementation supports max a lag of 350, while with an MA7 will mean a lag 96*7 $>$ 350.

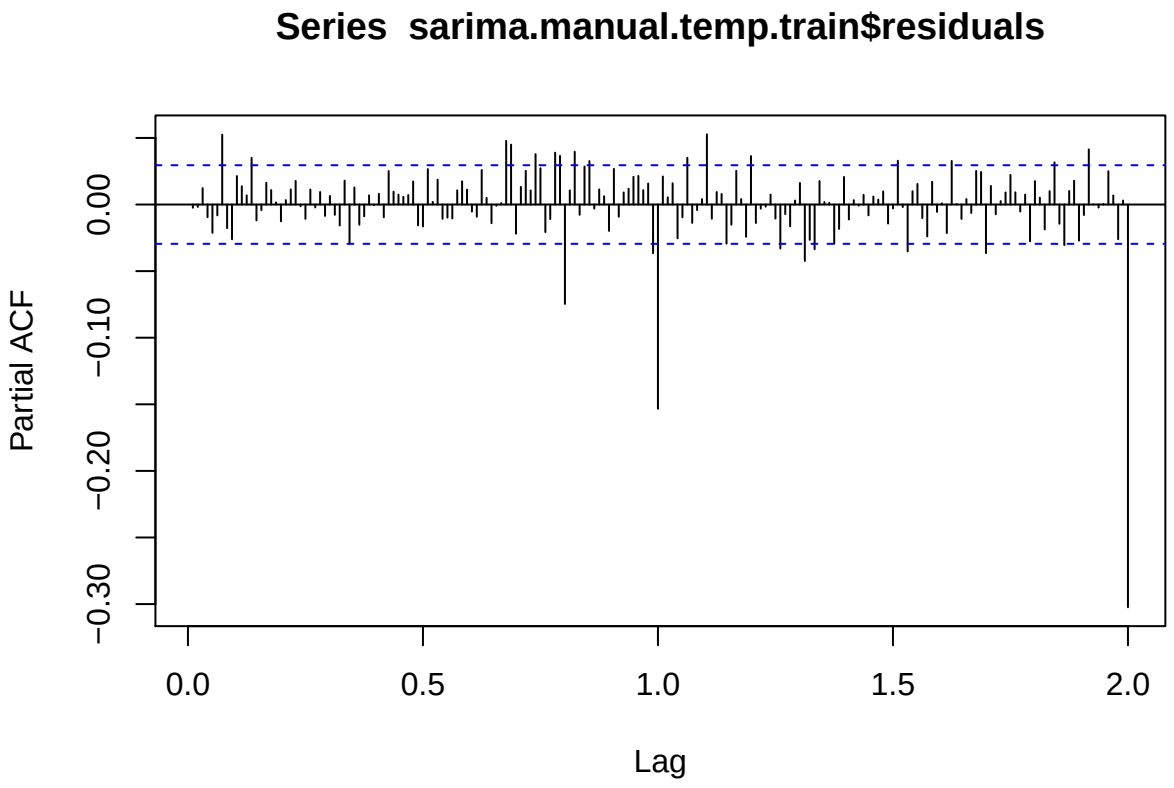
So the best SARIMA model we can get is eventually a SARIMA (5.0.0)(1.1.0) [96] with regression on temperature.

```
sarima.manual.temp.train = Arima( power.train, xreg = temp.ts.train ,order = c(5,0,0), seasonal = c(1,1

checkresiduals(sarima.manual.temp.train)
```



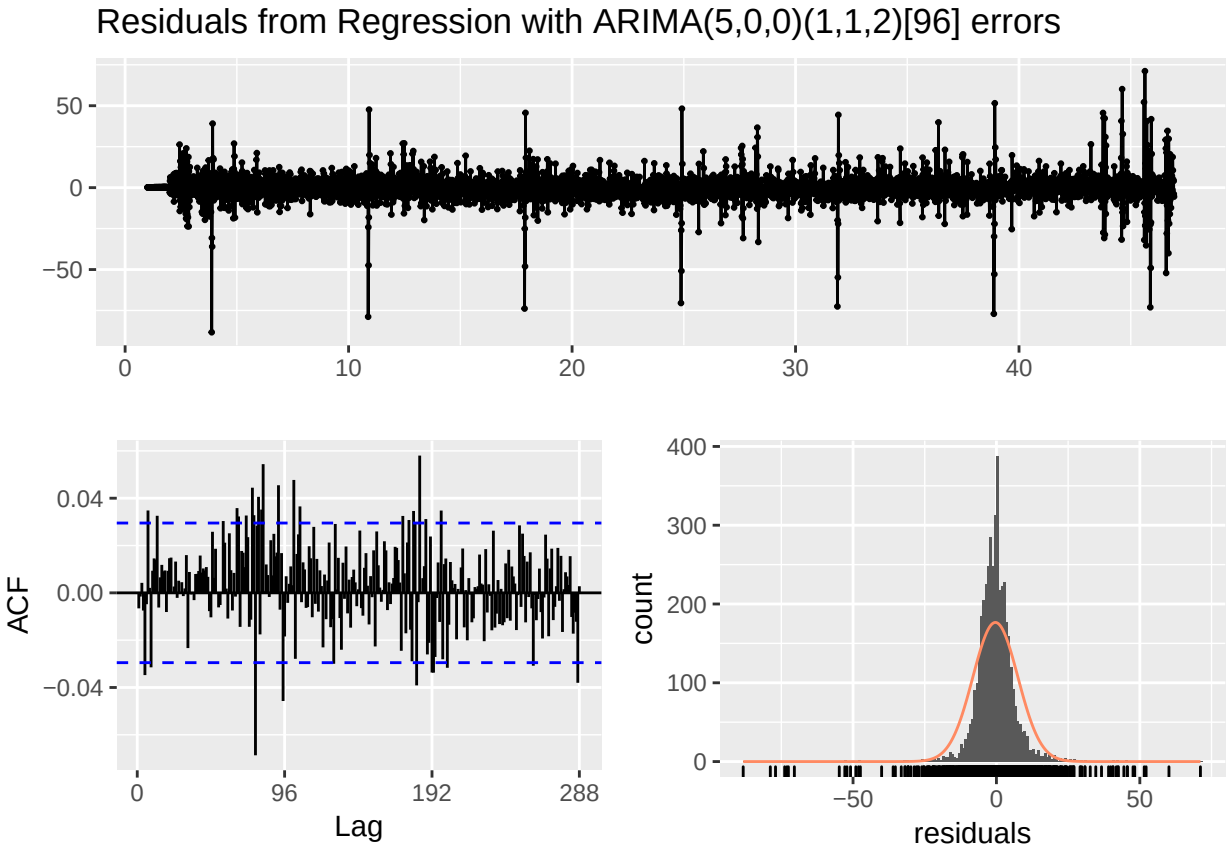

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(1,1,0)[96] errors
## Q* = 836.78, df = 185, p-value < 2.2e-16
##
## Model df: 7.   Total lags used: 192
```

```
pacf(sarima.manual.temp.train$residuals, lag.max = 96*2)
```



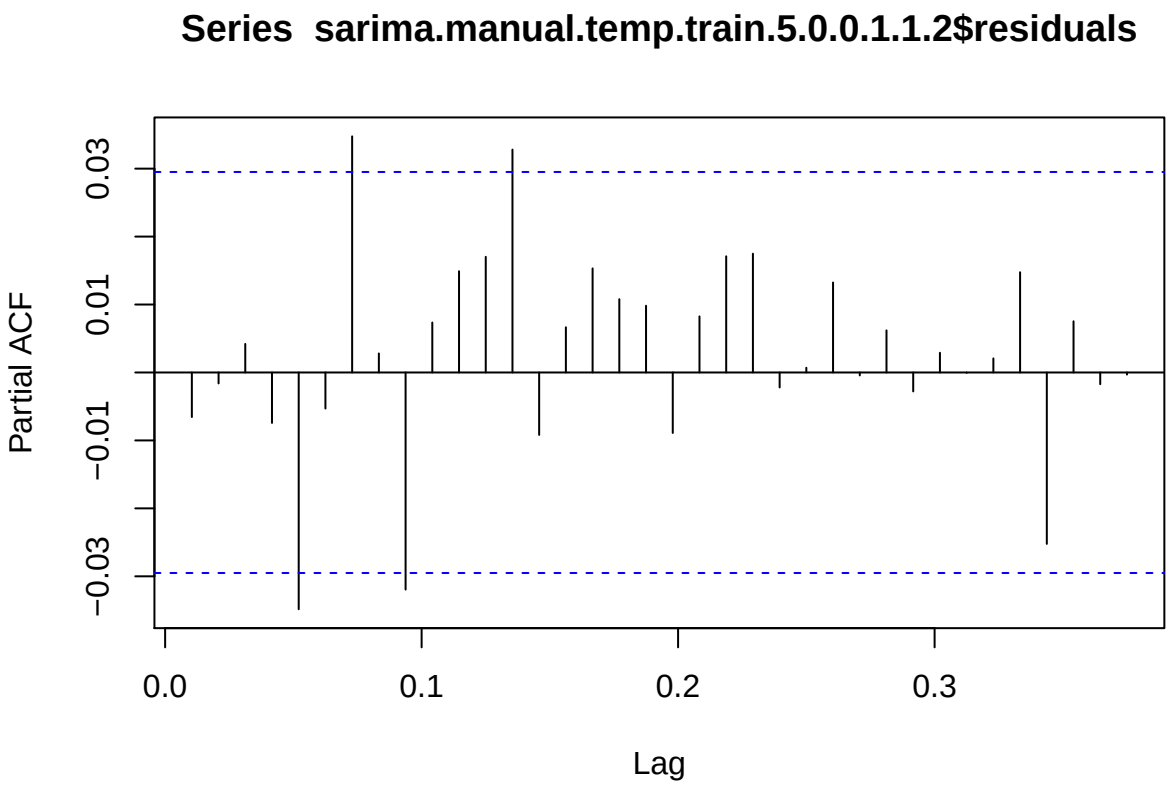
From the above it looks like we could introduce a MA2 seasonal term

```
sarima.manual.temp.train.5.0.0.1.1.2 = Arima( power.train, xreg = temp.ts.train ,order = c(5,0,0), seas

checkresiduals(sarima.manual.temp.train.5.0.0.1.1.2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(1,1,2)[96] errors
## Q* = 324.84, df = 183, p-value = 5.252e-10
##
## Model df: 9.   Total lags used: 192
```

```
pacf(sarima.manual.temp.train.5.0.0.1.1.2$residuals)
```



Residuals still seem not to be white noise, but I do not see room for further improvement, meaning that we should go increase p,q,P,Q but this would cause issues with R Arima function.

```
pred.arima.manual.temp.5.0.0.1.1.2 = forecast(sarima.manual.temp.train.5.0.0.1.1.2, xreg = temp.ts.test
```

```
RMSE.arima.manual.temp.5.0.0.1.1.2 = RMSE.func(pred.arima.manual.temp.5.0.0.1.1.2, power.test)
RMSE.arima.manual.temp.5.0.0.1.1.2
```

```
## [1] 14.79897
```

Seasonal NN with Temperature

Seasonal NN

```
fit.temp=nnetar(power.train, xreg = temp.ts.train)
print(fit.temp)
```

```
## Series: power.train
## Model:  NNAR(11,1,7)[96]
## Call:   nnetar(y = power.train, xreg = temp.ts.train)
##
## Average of 20 networks, each of which is
## a 13-7-1 network with 106 weights
## options were - linear output units
##
## sigma^2 estimated as 73.24
```

```
pred.NN.temp = forecast(fit.temp, h = 96, xreg = temp.ts.test)
RMSE.NN.temp = RMSE.func(pred.NN.temp, power.test)
RMSE.NN.temp
```

```
## [1] 18.03994
```

Var Method

```
library(vars)
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: urca
```

```
## Loading required package: lmtest
```

```
VARselect(cbind(power.train, temp.ts.train), lag.max=96, type="const")
```

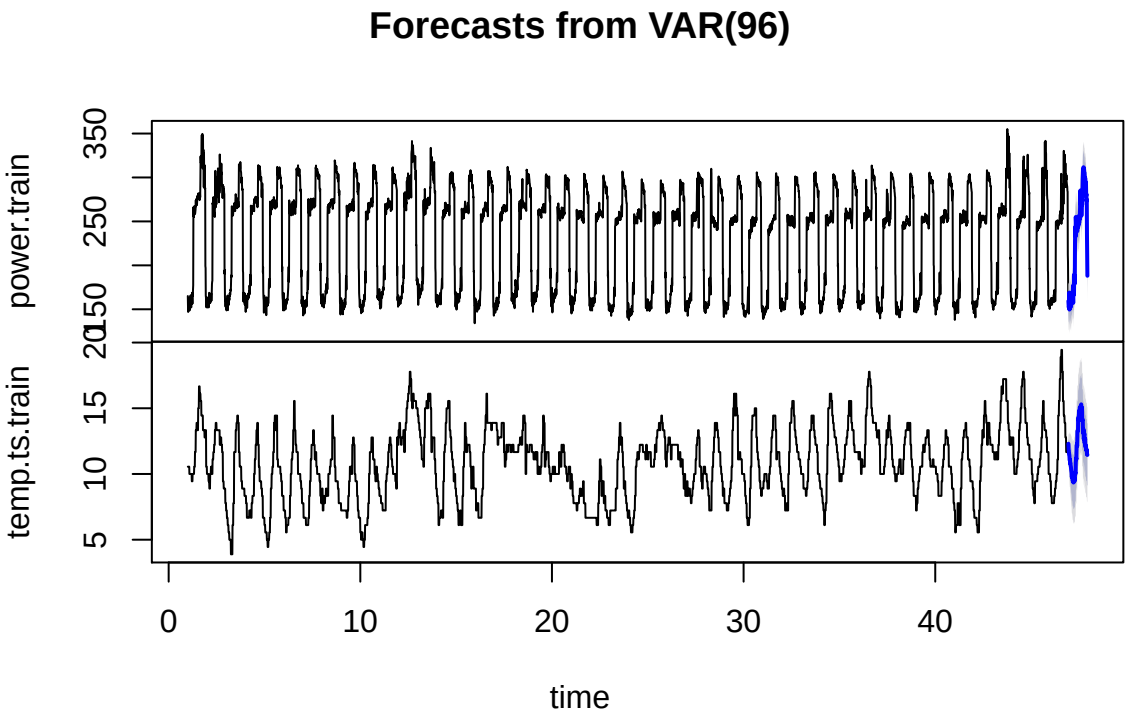
```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     96     96     96     96
##
## $criteria
##                1         2         3         4         5         6
## AIC(n)  3.917390  3.918841  3.911309  3.907782  3.800475  3.800990
## HQ(n)   3.920518  3.924053  3.918606  3.917165  3.811943  3.814543
## SC(n)   3.926248  3.933603  3.931976  3.934354  3.832952  3.839371
## FPE(n) 50.269090 50.342040 49.964295 49.788408 44.722437 44.745458
##                7         8         9        10        11        12
## AIC(n)  3.800666  3.792820  3.750097  3.749562  3.747681  3.745969
## HQ(n)   3.816304  3.810543  3.769905  3.771456  3.771659  3.772032
## SC(n)   3.844953  3.843011  3.806193  3.811563  3.815587  3.819779
## FPE(n) 44.730987 44.381369 42.525191 42.502489 42.422598 42.350030
##               13        14        15        16        17        18
## AIC(n)  3.725897  3.725207  3.722768  3.720893  3.710262  3.706056
## HQ(n)   3.754046  3.755440  3.755087  3.755296  3.746750  3.744630
## SC(n)   3.805613  3.810827  3.814293  3.818323  3.813596  3.815296
## FPE(n) 41.508482 41.479817 41.378797 41.301271 40.864528 40.693029
##               19        20        21        22        23        24
## AIC(n)  3.704203  3.703795  3.681561  3.678174  3.674121  3.672899
## HQ(n)   3.744862  3.746539  3.726390  3.725088  3.723120  3.723984
## SC(n)   3.819348  3.824844  3.808516  3.811033  3.812884  3.817568
## FPE(n) 40.617707 40.601135 39.708399 39.574133 39.414049 39.365952
##               25        26        27        28        29        30
## AIC(n)  3.648014  3.643445  3.639889  3.637142  3.602372  3.597602
```

```
## HQ(n)   3.701184  3.698699  3.697228  3.696567  3.663881  3.661197
## SC(n)   3.798588  3.799923  3.802272  3.805430  3.776565  3.777700
## FPE(n) 38.398434 38.223367 38.087705 37.983256 36.685261 36.510719
##               31        32        33        34        35        36
## AIC(n)  3.586185  3.579994  3.554556  3.549706  3.540594  3.536116
## HQ(n)   3.651864  3.647759  3.624406  3.621641  3.614614  3.612221
## SC(n)   3.772187  3.771901  3.752368  3.753423  3.750216  3.751643
## FPE(n) 36.096242 35.873484 34.972459 34.803264 34.487611 34.333538
##               37        38        39        40        41        42
## AIC(n)  3.481044  3.474294  3.466595  3.460826  3.437045  3.435699
## HQ(n)   3.559234  3.554569  3.548955  3.545272  3.523575  3.524315
## SC(n)   3.702476  3.701631  3.699836  3.699973  3.682096  3.686655
## FPE(n) 32.493852 32.275285 32.027766 31.843558 31.095219 31.053439
##               43        44        45        46        47        48
## AIC(n)  3.435893  3.431987  3.422496  3.422978  3.424735  3.422525
## HQ(n)   3.526594  3.524773  3.517367  3.519934  3.523776  3.523652
## SC(n)   3.692754  3.694753  3.691167  3.697553  3.705215  3.708911
## FPE(n) 31.059481 30.938423 30.646196 30.660995 30.714934 30.647180
##               49        50        51        52        53        54
## AIC(n)  3.406611  3.406766  3.408285  3.407334  3.390186  3.390407
## HQ(n)   3.509822  3.512062  3.515666  3.516801  3.501738  3.504043
## SC(n)   3.698901  3.704961  3.712385  3.717339  3.706096  3.712221
## FPE(n) 30.163331 30.168044 30.213934 30.185260 29.672086 29.678660
##               55        56        57        58        59        60
## AIC(n)  3.390746  3.388765  3.346430  3.344637  3.340181  3.333633
## HQ(n)   3.506468  3.506572  3.466322  3.466614  3.464243  3.459780
## SC(n)   3.718465  3.722389  3.685959  3.690070  3.691519  3.690877
## FPE(n) 29.688774 29.630045 28.401884 28.351034 28.225027 28.040864
##               61        62        63        64        65        66
## AIC(n)  3.290356  3.290630  3.285546  3.286888  3.272410  3.272258
## HQ(n)   3.418588  3.420947  3.417948  3.421375  3.408982  3.410915
## SC(n)   3.653504  3.659682  3.660504  3.667750  3.659177  3.664930
## FPE(n) 26.853248 26.860641 26.724478 26.760401 26.375802 26.371837
##               67        68        69        70        71        72
## AIC(n)  3.266534  3.268205  3.250139  3.250931  3.251399  3.251234
## HQ(n)   3.407277  3.411033  3.395052  3.397929  3.400482  3.402401
## SC(n)   3.665111  3.672687  3.660526  3.667223  3.673596  3.679335
## FPE(n) 26.221379 26.265283 25.795081 25.815564 25.827710 25.823483
##               73        74        75        76        77        78
## AIC(n)  3.234639  3.235884  3.234278  3.235583  3.226487  3.224988
## HQ(n)   3.387892  3.391222  3.391701  3.395091  3.388080  3.388666
## SC(n)   3.668645  3.675795  3.680094  3.687304  3.684113  3.688519
## FPE(n) 25.398536 25.430232 25.389481 25.422698 25.192570 25.154895
##               79        80        81        82        83        84
## AIC(n)  3.224892  3.224463  3.221118  3.221528  3.222544  3.221604
## HQ(n)   3.390655  3.392312  3.391052  3.393547  3.396647  3.397792
## SC(n)   3.694327  3.699804  3.702363  3.708678  3.715598  3.720563
## FPE(n) 25.152535 25.141824 25.057925 25.068267 25.093809 25.070302
##               85        86        87        88        89        90
## AIC(n)  3.205122  3.201607  3.189486  3.186515  3.152022  3.147153
## HQ(n)   3.383396  3.381966  3.371930  3.371044  3.338636  3.335852
## SC(n)   3.709986  3.712376  3.706160  3.709094  3.680505  3.681542
## FPE(n) 24.660552 24.574093 24.278101 24.206153 23.385516 23.272022
##               91        92        93        94        95        96
```

```
## AIC(n)  3.135598  3.121640  3.075194  3.028065  2.990475  2.903087
## HQ(n)   3.326383  3.314510  3.270148  3.225105  3.189599  3.104296
## SC(n)   3.675892  3.667839  3.627297  3.586073  3.554387  3.472904
## FPE(n) 23.004739 22.685944 21.656427 20.659546 19.897427 18.232505
```

```
var <- VAR(cbind(power.train, temp.ts.train), p=96,type = "const")
#summary(var)
```

```
fcst <- forecast(var,h=96)
plot(fcst)
```




```
serial.test(var, lags.pt=96, type="PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var
## Chi-squared = 658.62, df = 0, p-value < 2.2e-16
```

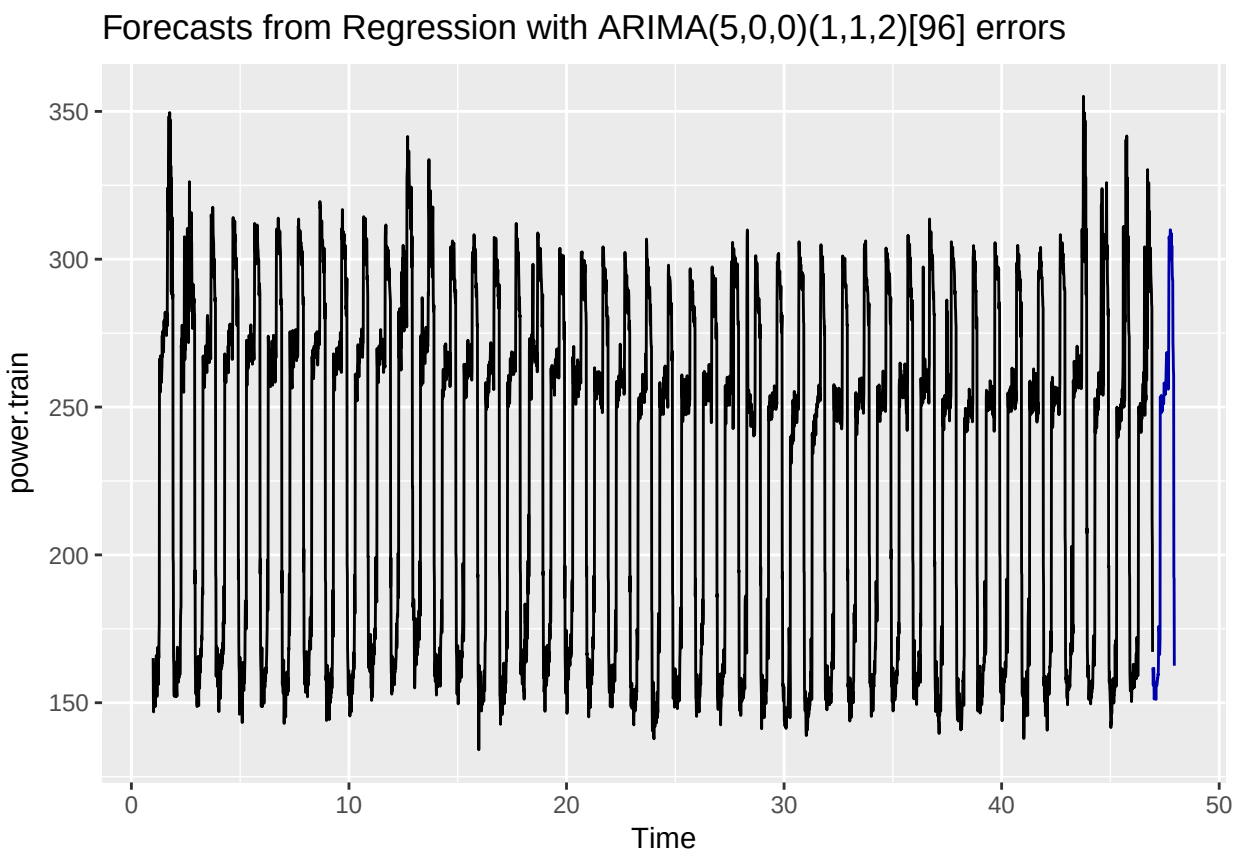
The noise is not white.

```
RMSE.func(fcst$forecast$power.train,power.test)
```

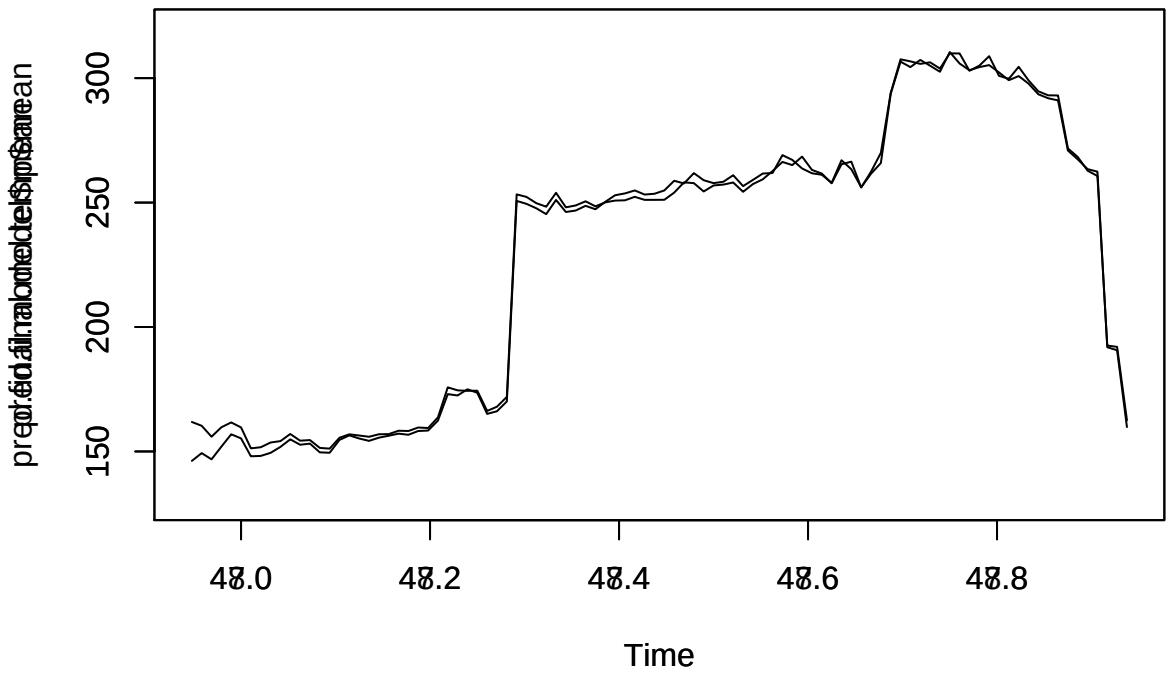
```
## [1] 17.94055
```

Prediction with the best model

```
final.model.temp = Arima( power.train, xreg = temp.ts.train ,order = c(5,0,0), seasonal = c(1,1,2))

temp.assignment = tail(elCons.data, n= 96)

temp.ts.assignment = ts(temp.assignment[,"Temp"], frequency = 96)

pred.final.model.temp = forecast(final.model.temp, xreg = temp.ts.assignment, h = 96)

autoplot(pred.final.model.temp, PI = FALSE)
```




```
plot(pred.final.model.temp$mean, ylim = c(130,320) )
par(new=TRUE)
plot(pred.final.model$mean, ylim = c(130,320))
```



Merge predictions without and with temperature

```
#library("xlsx")

setwd("D:/BIG_DATA/DSTI/OneDrive - Data ScienceTech Institute/2020-09-Time-series/assignment")

#write.xlsx(pred.final.model$mean, file = "AndreaSonnellini_temp.xlsx")

write.csv(pred.final.model.temp$mean, file = "AndreaSonnellini_temp.csv", row.names = FALSE, col.names =
```

```
## Warning in write.csv(pred.final.model.temp$mean, file =
## "AndreaSonnellini_temp.csv", : attempt to set 'col.names' ignored
```

```
notTemp = pred.final.model
```

```
finalComparison= cbind.data.frame(PowerNoTemp = notTemp$mean, PowerTemp = pred.final.model.temp$mean)

head(finalComparison)
```

```
##   PowerNoTemp PowerTemp
## 1    146.2063  161.8483
## 2    149.3343  160.3127
## 3    146.8370  155.9211
## 4    151.9251  159.7166
## 5    156.9012  161.6722
## 6    155.2494  159.6542
```

```
#write.csv(finalComparison, file = "AndreaSonnellini-final.csv", row.names = FALSE, sep = ";")
```