Time Series Analysis Exam

Electricity Consumption Prediction

Mariano CRIMI

Basic exploration and wrangling

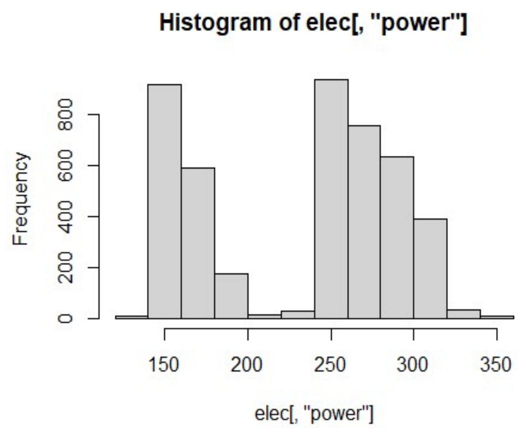
We read in the data from the working directory. We do a basic axploration to understand how the daily data is composed.

```
data <- read_excel("Elec-train.xlsx")
data$Timestamp <- strptime(data$Timestamp, "%m/%d/%Y %H:%M")
data <- mutate(data, Day = as.Date(Timestamp))
data %>% group_by(Day) %>% summarise(no_rows = length(Day))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 48 x 2
##    Day        no_rows
##    <date>       <int>
##  1 2010-01-01      91
##  2 2010-01-02      96
##  3 2010-01-03      96
##  4 2010-01-04      96
##  5 2010-01-05      96
##  6 2010-01-06      96
##  7 2010-01-07      96
##  8 2010-01-08      96
##  9 2010-01-09      96
## 10 2010-01-10      96
## # ... with 38 more rows
```

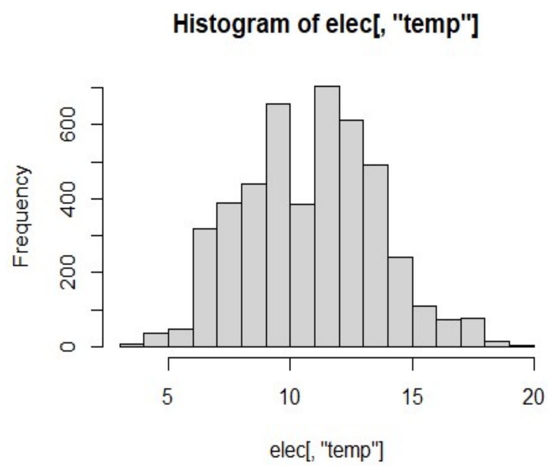
We have 48 days worth of data, with 96 samples each day (every 15 mins), with the exception of day 1, for which we only have 91 minutes. The series starts in the 6th quarter that day.

With this information we can construct the time series object:

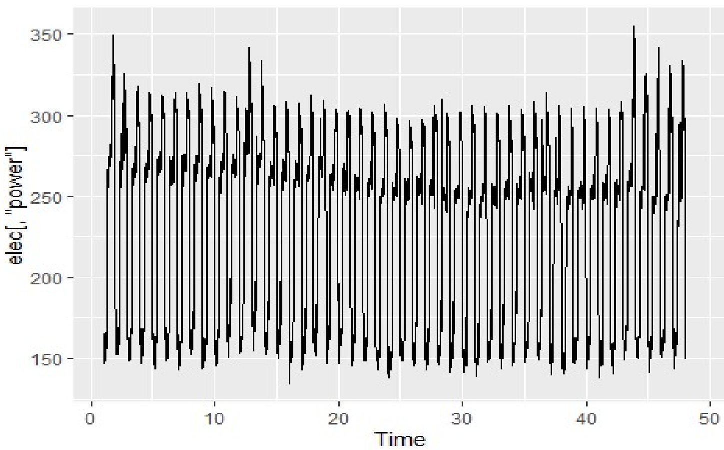
```
#convert data into a time series object
elec <- ts(data %>% select(2, 3), start=c(1,6),end=c(48,96), frequency=96)
#setup column names
colnames(elec) <- c("power","temp")

#Plot histograms
hist(elec[,"power"])
```



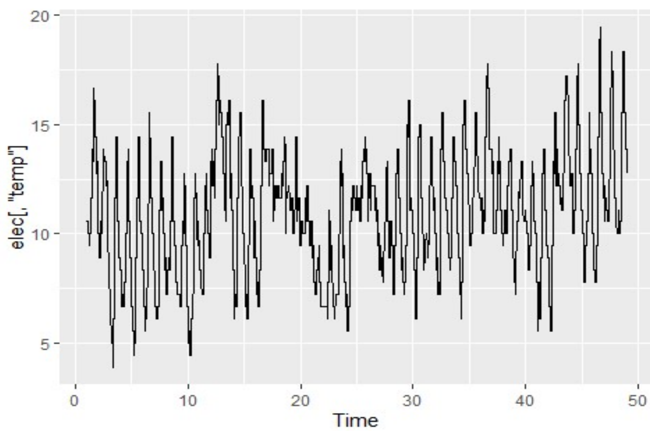

```r
hist(elec[,"temp"])
```




```r
#Plot raw data
autoplot(elec[,"power"])
```



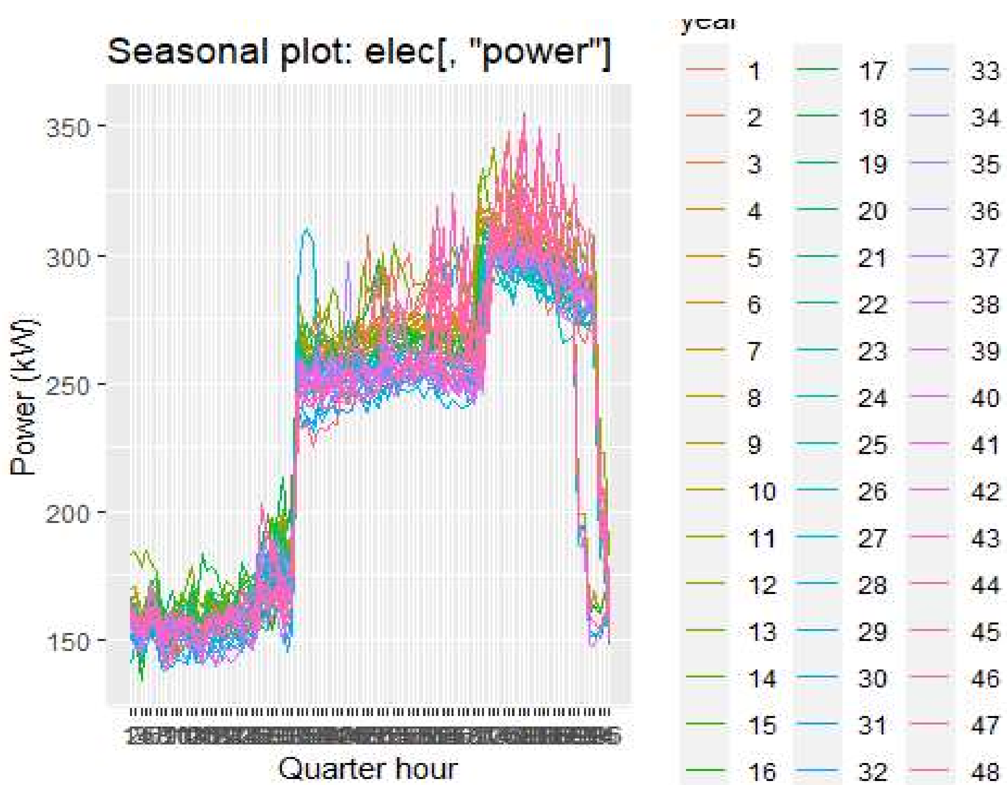
```r
autoplot(elec[,"temp"])
```



Now I look at the data from a seasonal perspective:

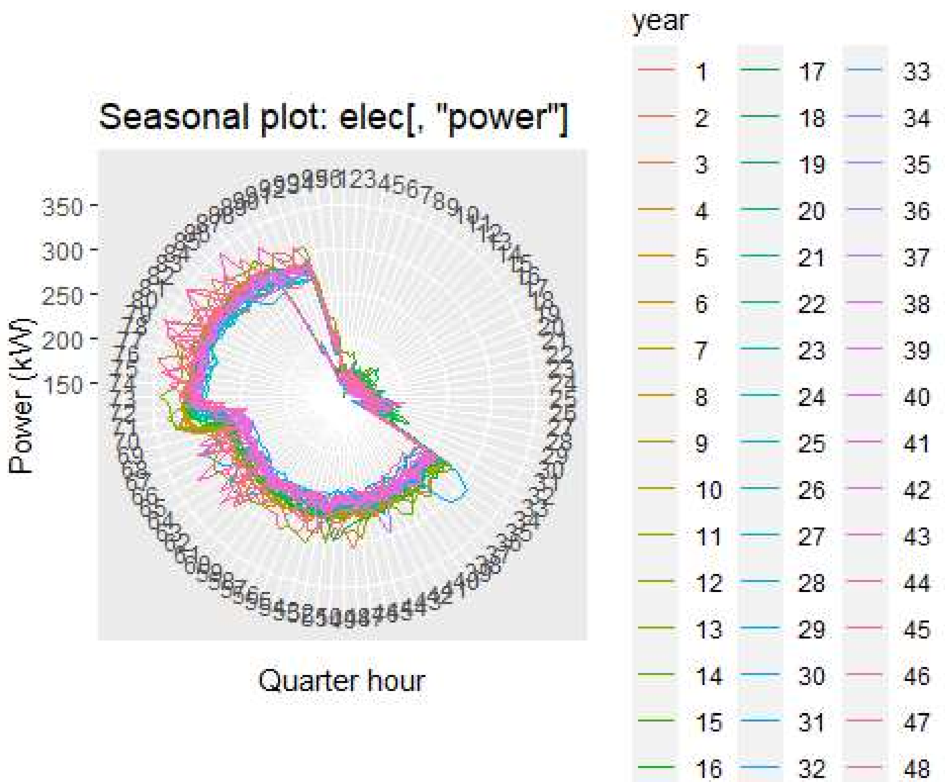
```r
#Seasonal plot
ggseasonplot(elec[,"power"], ylab= 'Power (kW)', xlab = 'Quarter hour')
```

```
## Warning: Removed 96 row(s) containing missing values (geom_path).
```

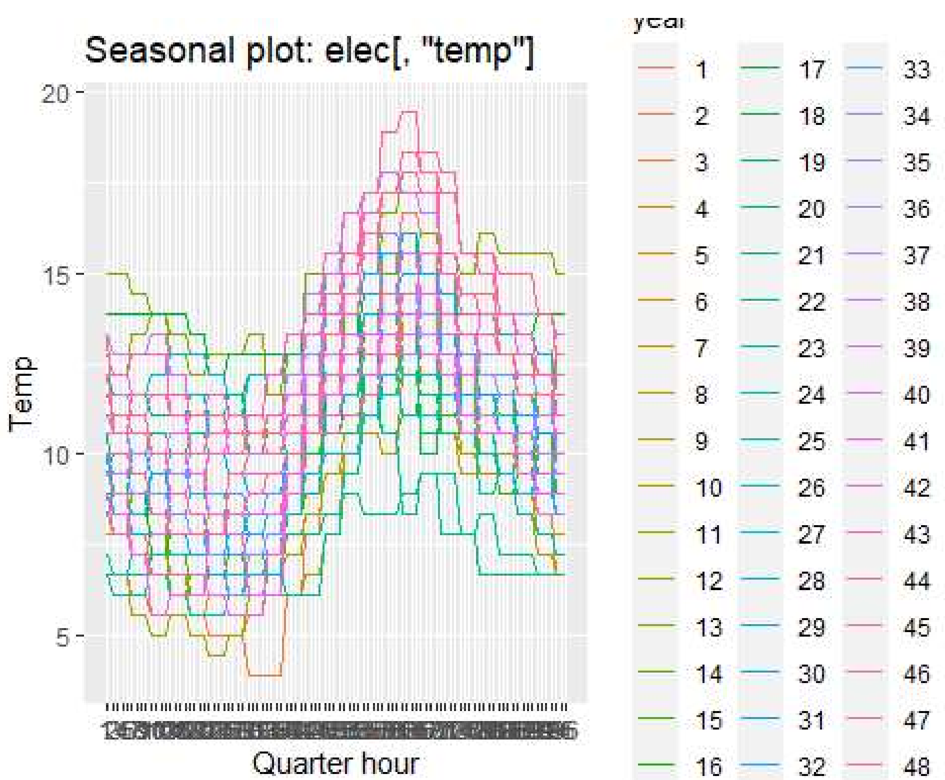



```r
ggseasonplot(elec[,"power"], ylab= 'Power (kW)', xlab = 'Quarter hour',
polar=TRUE)
```

```
## Warning: Removed 97 row(s) containing missing values (geom_path).
```

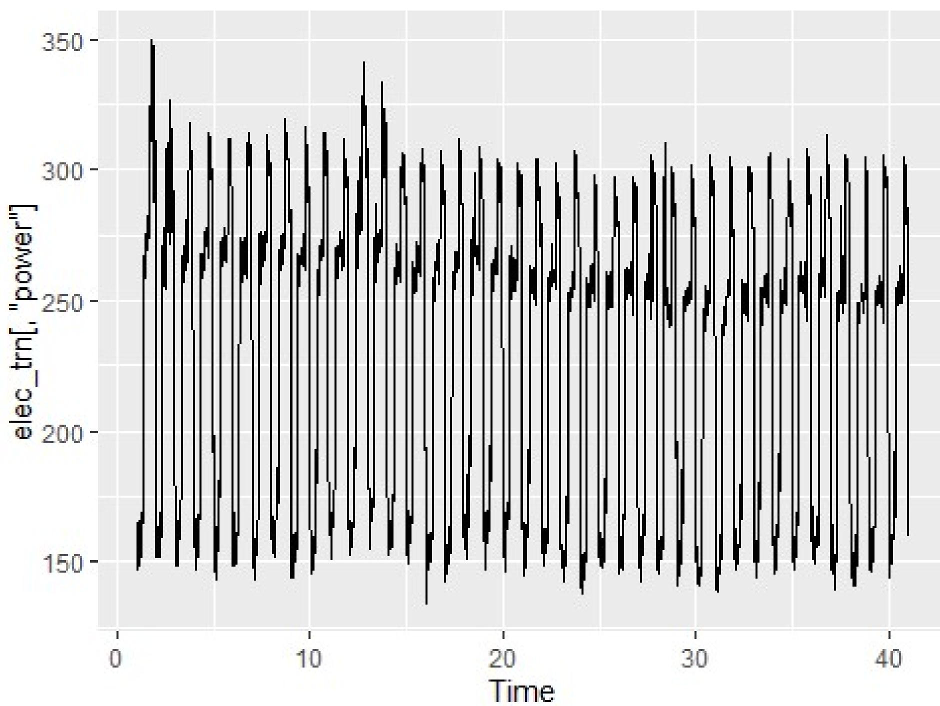



```r
ggseasonplot(elec[,"temp"], ylab= 'Temp', xlab = 'Quarter hour')
```

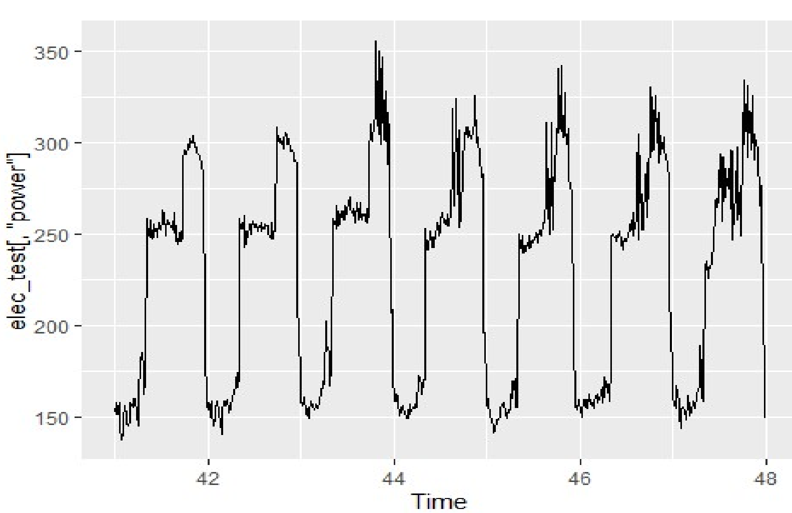


I then proceed to split the data into training and validation, in order to be able to choose the best model.

```
#We take days 1 to 40 as our trining data
elec_trn <- window(elec, start=c(1, 6), end=c(40,96))
autoplot(elec_trn[,"power"])
```

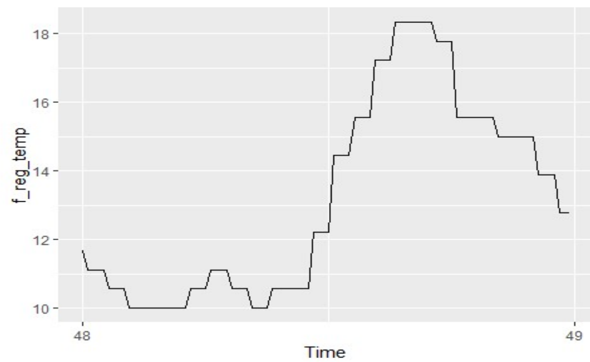



```
#We take days 40 to 47 as our validation data
elec_test <- window(elec, start=c(41, 1), end=c(47,96))
autoplot(elec_test[,"power"])
```



We take the temprature of day 48, so that we can include them as regressors in the final forecast

```
f_reg_temp <- window(elec[,"temp"], start=c(48, 1), end=c(48,96))
autoplot(f_reg_temp)
```



And finally I set up the horizon for the validation process.

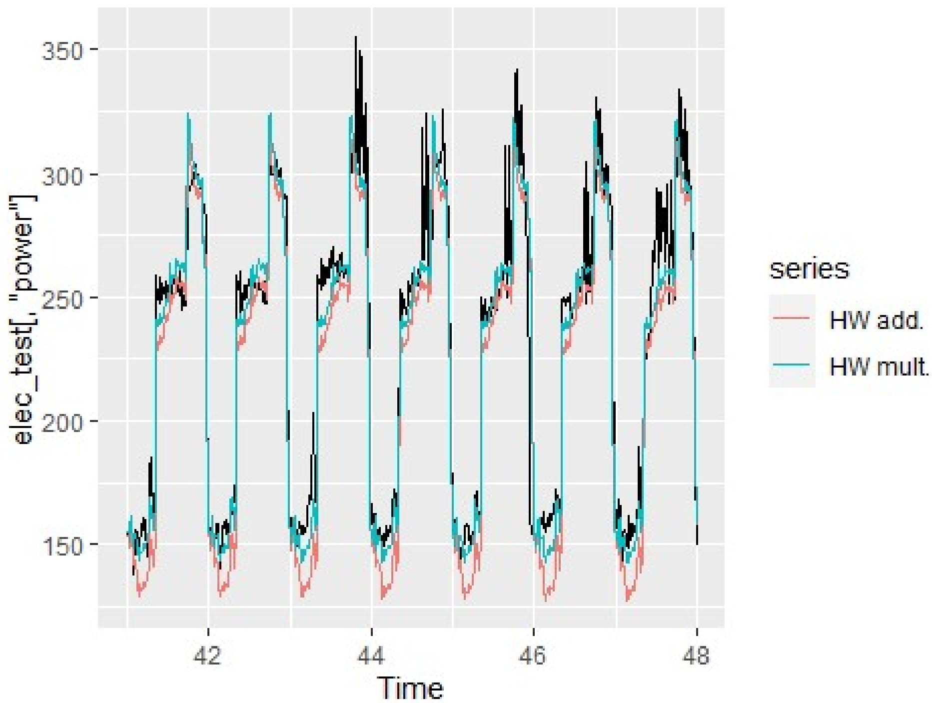
```
test_h = 7*96
```

Modelization (without regressors)

Holt-Winters

We start our analyis using exponential smoothing, trying both the additive and multiplicative seasonal factors and letting the function choose the optimal hyperparameters alpha, beta and gamma.

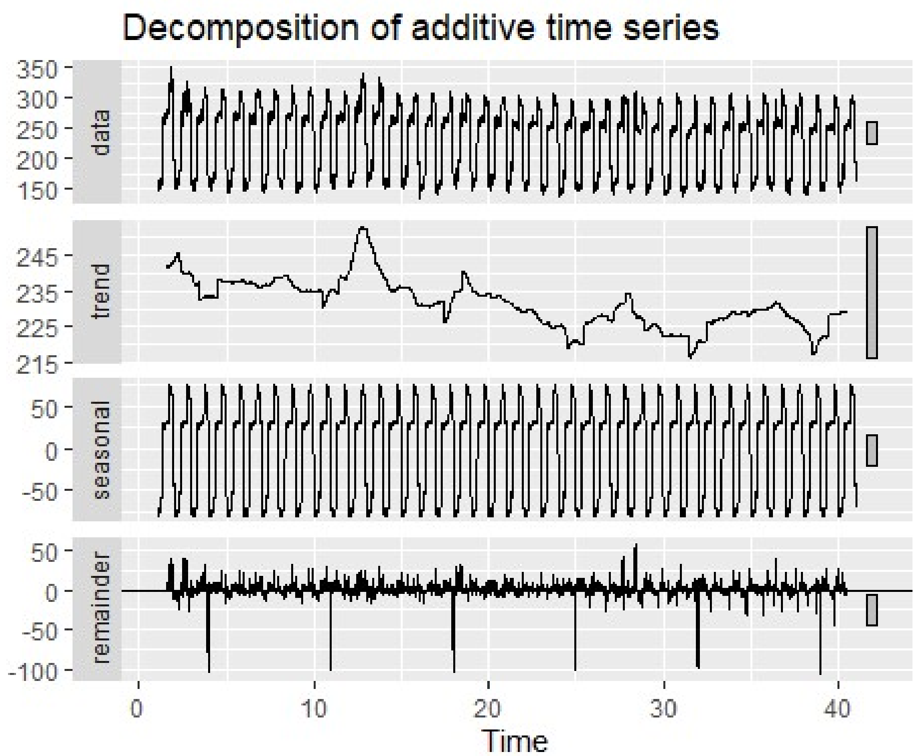
```
#Setup and predict with additive seasonal effect
hw_fitAdd <- HoltWinters(elec_trn[,"power"],alpha=NULL,beta=NULL,gamma=NULL,
seasonal = "additive")
hw_fitAdd_pred <- predict(hw_fitAdd,n.ahead=test_h)

#Setup and predict with multiplicative seasonal effect
hw_fitMult <- HoltWinters(elec_trn[,"power"],alpha=NULL,beta=NULL,gamma=NULL,
seasonal = "multiplicative")
hw_fitMult_pred <- predict(hw_fitMult,n.ahead=test_h)

#Plot predictions against ground trouth
autoplot (elec_test[,"power"]) +
  autolayer(hw_fitAdd_pred, series='HW add.',PI=FALSE) +
  autolayer(hw_fitMult_pred, series='HW mult.',PI=FALSE)

## Warning: Ignoring unknown parameters: PI

## Warning: Ignoring unknown parameters: PI
```



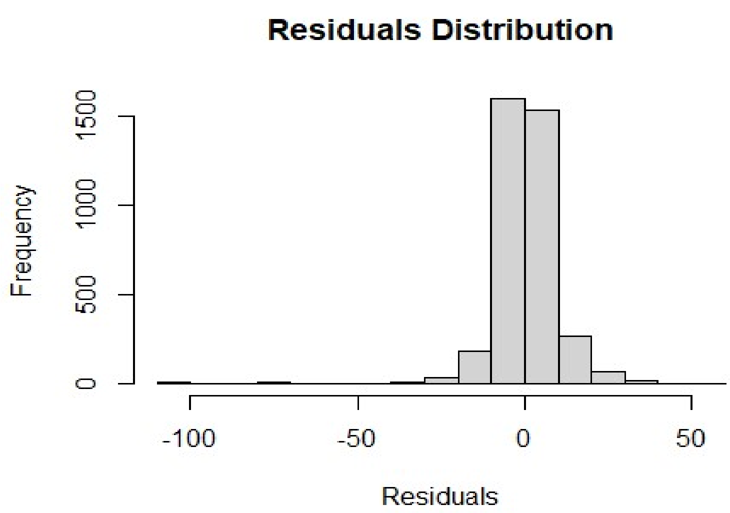

```
#Calculate Errors
rmse(elec_test[,"power"],hw_fitAdd_pred)

## [1] 20.19866

rmse(elec_test[,"power"],hw_fitMult_pred)

## [1] 15.34826

hw_fitAdd$test_rmse <- rmse(elec_test[,"power"],hw_fitAdd_pred)
hw_fitMult$test_rmse <- rmse(elec_test[,"power"],hw_fitMult_pred)
```

The HW with multiplicative effect fit is not too bad, but we move forward to explore possible ARIMA models. First we decompose the serie to see if there's an stochastic part to be modeled

```
elec_decomp  <- decompose(elec_trn[,"power"])
autoplot(elec_decomp) + xlab('Time')
```




```
hist(elec_decomp$random, main= "Residuals Distribution", xlab = "Residuals")
```



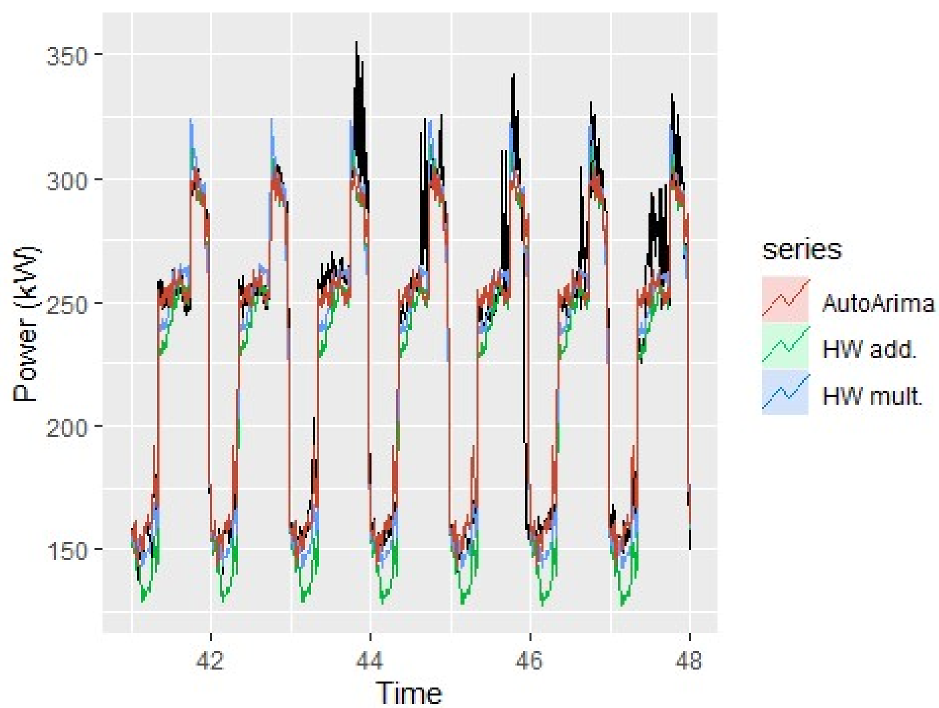

```
Box.test(elec_decomp$random,type="Ljung-Box")

##
##  Box-Ljung test
##
## data:  elec_decomp$random
## X-squared = 2044.5, df = 1, p-value < 2.2e-16
```

We see that we can we null hypothesis that the residuals can be defined as white noise so we proceed with an attempt to model them.

ARIMA

I first start by trying an automatic ARIMA,

```
autoarima_fit <- auto.arima(elec_trn[,"power"])
autoarima_pred  <- forecast(autoarima_fit, h=test_h)

autoplot (elec_test[,"power"], ylab = 'Power (kW)') +
  autolayer(hw_fitAdd_pred, series='HW add.',PI=FALSE) +
  autolayer(hw_fitMult_pred, series='HW mult.',PI=FALSE) +
  autolayer(autoarima_pred, series='AutoArima',PI=FALSE)

## Warning: Ignoring unknown parameters: PI

## Warning: Ignoring unknown parameters: PI
```




```
rmse(elec_test[,"power"],autoarima_pred$mean)

## [1] 15.37603

autoarima_fit$test_rmse <- rmse(elec_test,autoarima_pred$mean)
checkresiduals(autoarima_fit, plot= FALSE)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 1495.9, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

We see that the autorima picks up the seasonality and correctly differentiates to remove it. Then it finds an AR model for the non sesonal part.

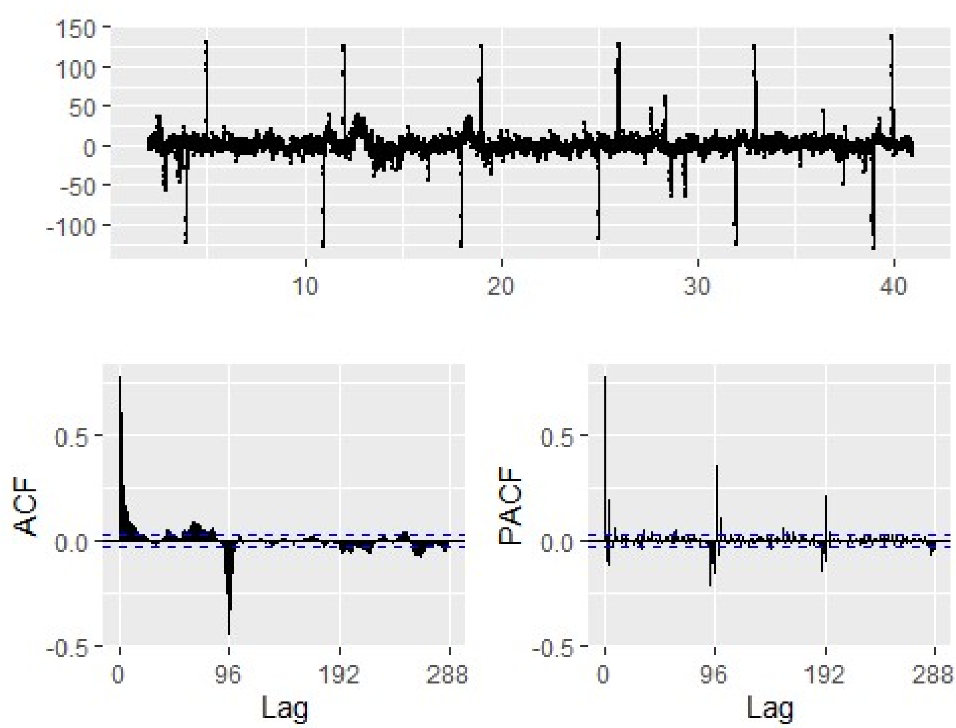
The RMSE is better but quite similar to the one obtained by testing the HW multiplicative. It seems it would be worth exploring if another parameters for p,d and q would perform better.

I now attempt a manual ARIMA. We first diffenciate our series by season in order to remove it:

```
elec_diff_96 <- diff(elec_trn[,"power"], lag=96)
```

We then plot PACF and ACF for the differentiated series

```
ggtsdisplay(elec_diff_96)
```



The series seems aproximately stationary, but we pefform the root unit test.

```
elec_diff_96 %>% ur.kpss() %>% summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 9 lags.
##
## Value of test-statistic is: 0.0725
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

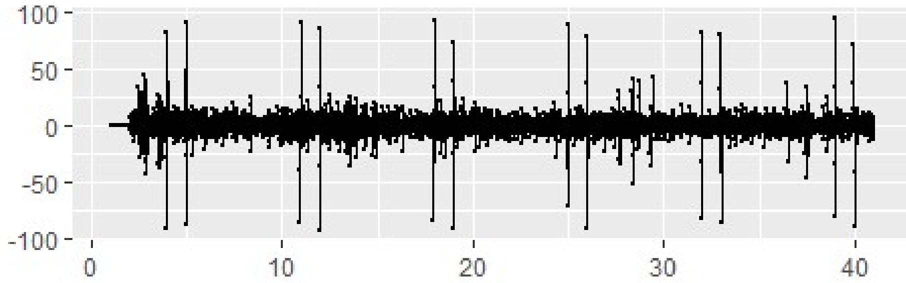
Indeed, the series is stationary but we see that the test statistic is much bigger than the 1% critical value. We are tempted to differentiate the series again:

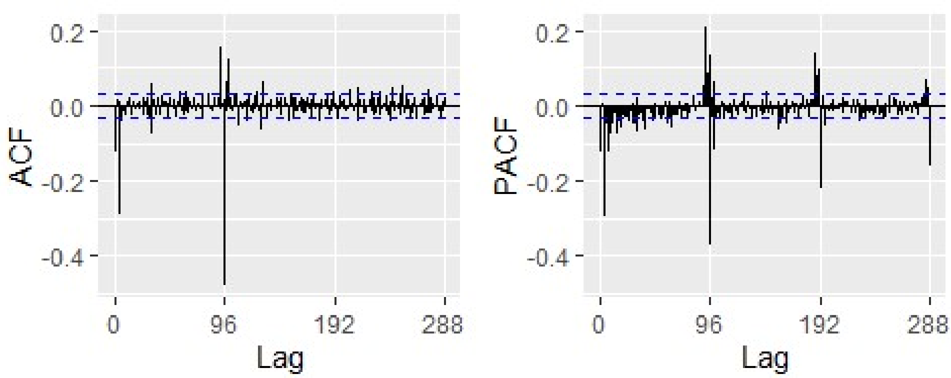
```
elec_diff_96 %>% diff() %>% ur.kpss() %>% summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 9 lags.
##
## Value of test-statistic is: 0.0015
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

This time it seems like the differentiation really improve the significance value of the test, so we imagine that the model would benefit from 1 seasonal differentiation and 1 non seasonal differentiation. So we are tempted to try a non seasonal differentiation

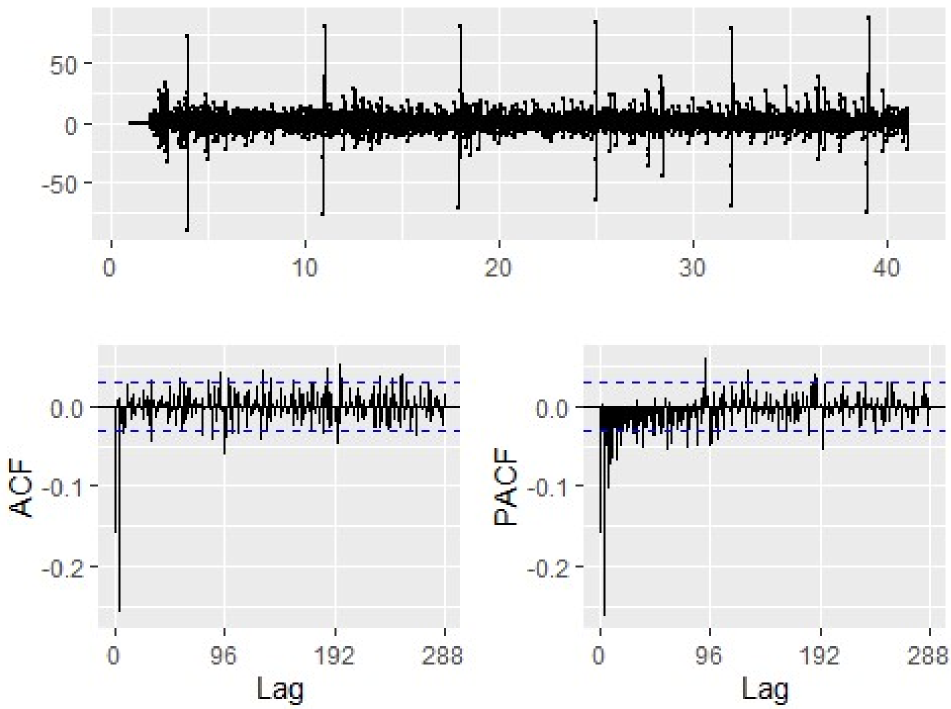
```
arima_fit <- Arima(elec_trn[,"power"], order= c(0,1,0), seasonal=c(0,1,0))
arima_fit %>% residuals() %>% ggtsdisplay()
```





We now see a clear seasonal patterrn suggestive of an MA(1)96 with decay on the PACF and a single signicant value at 96

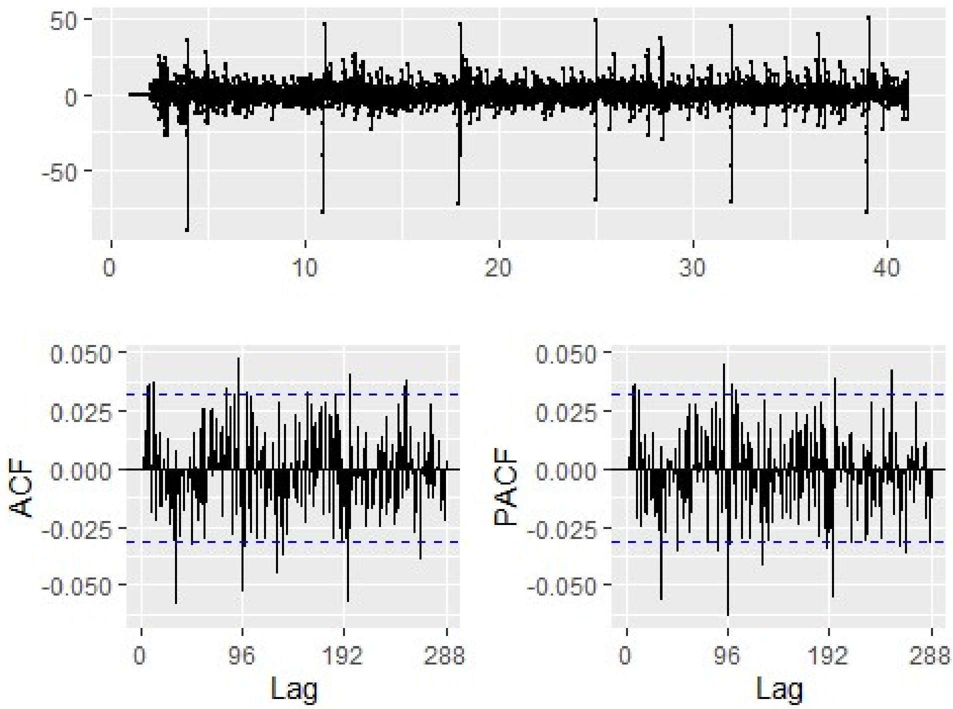
```
arima_fit <- Arima(elec_trn[,"power"], order= c(0,1,0), seasonal=c(0,1,1))
arima_fit %>% residuals() %>% ggtsdisplay()
```



It seems like this model capture the seasonal correlations well, but we still have quite significant lags in the non-seasonal part.

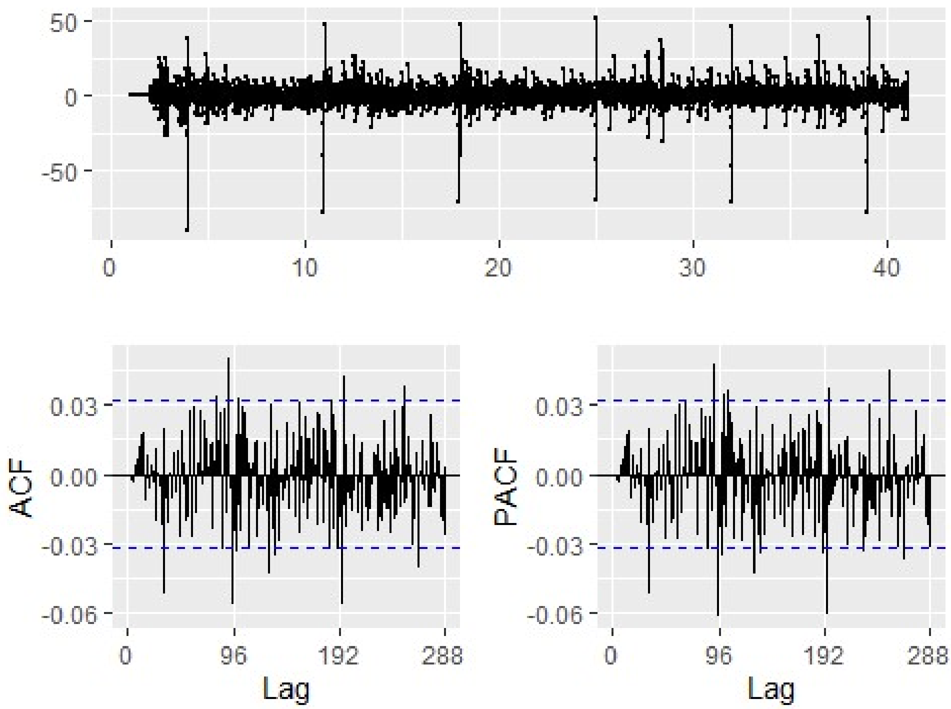
There seems to be significance pattern at lag 4, but we have also a slightly significative lag at 7 with decay on the PACF. I incline towards MA(7) to capture that last one.

```
arima_fit <- Arima(elec_trn[,"power"], order= c(0,1,7), seasonal=c(0,1,1))
arima_fit %>% residuals() %>% ggtsdisplay()
```



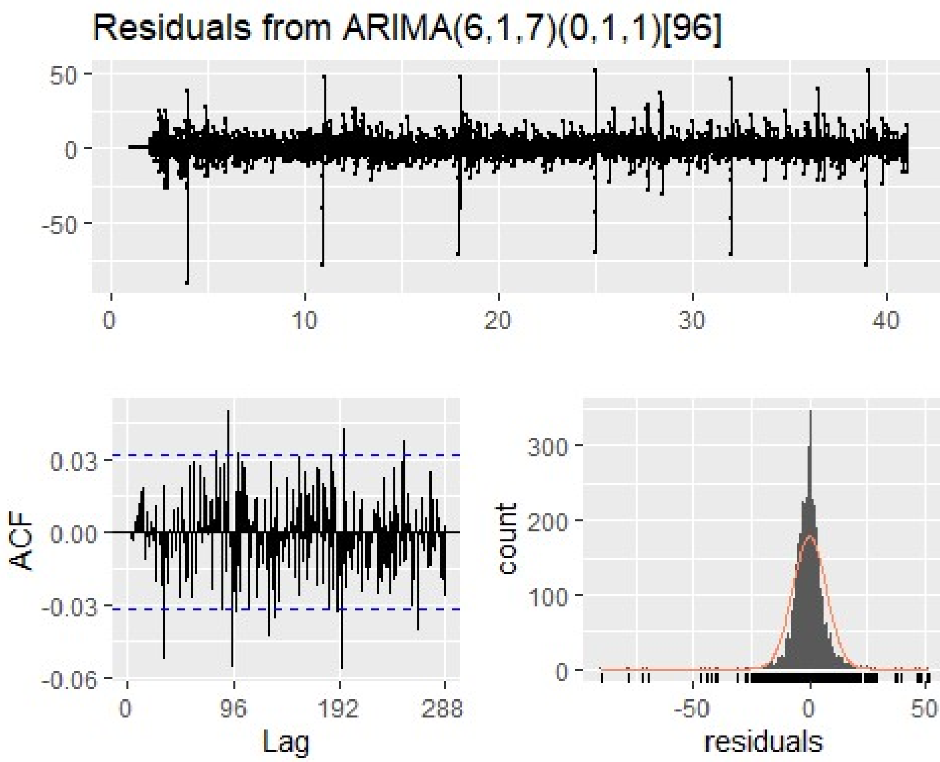
We see significance at lag 6 and 7. We can try an AR(6) for the nonseasonal part for simplicity.

```
arima_fit <- Arima(elec_trn[,"power"], order= c(6,1,7), seasonal=c(0,1,1))
arima_fit %>% residuals() %>% ggtsdisplay()
```



I proceed to evaluate this last model in terms of RMSE and Boxtext:

```
arima_pred  <- forecast(arima_fit, h=test_h)
rmse(elec_test[,"power"],arima_pred$mean)

## [1] 14.54987

checkresiduals(arima_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(6,1,7)(0,1,1)[96]
## Q* = 232.4, df = 178, p-value = 0.003825
##
## Model df: 14.   Total lags used: 192
```

It seems that we captured the initial lag correlations. There's still significant correlations, even at lag 95 but we chose not to model them as it would make the model too complicated.

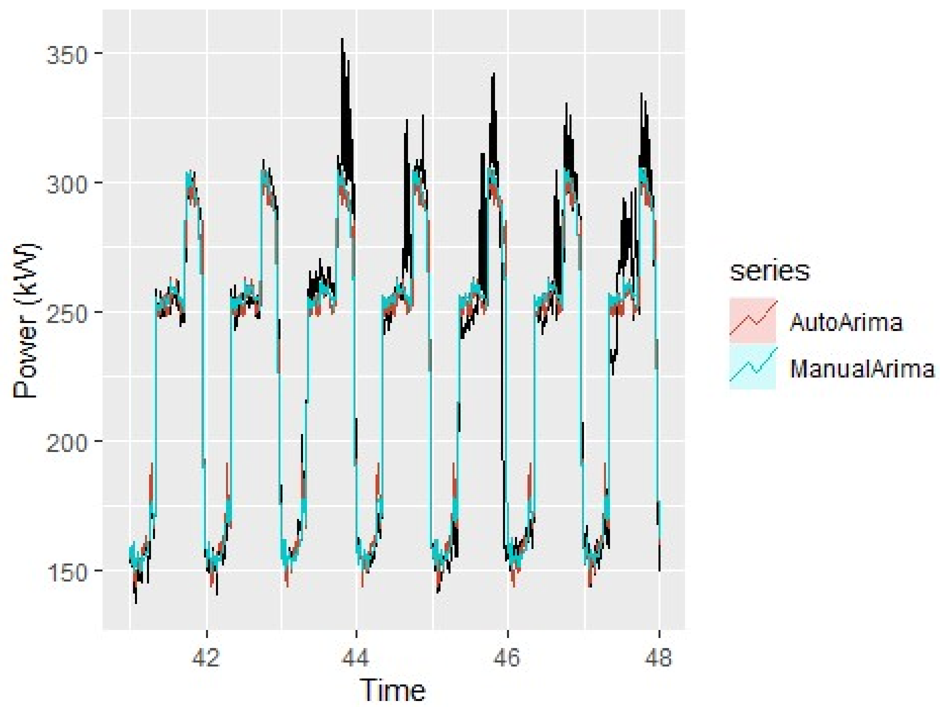
RMSE is quite good and the Box test is a little bit more acceptable. We proceed with this one as the manual baseline.

We store the RMSE for further comparison.

```
arima_pred$test_rmse <- rmse(elec_test[,"power"],arima_pred$mean)
```

We then plot the manual and auto SARIMA models:

```
autoplot (elec_test[,"power"], ylab = 'Power (kW)') +
  autolayer(autoarima_pred, series='AutoArima',PI=FALSE) +
  autolayer(arima_pred, series='ManualArima',PI=FALSE)
```



Neural networks

We now try some neural network models.

We first attempt an automatic fit:

```
#Auto
auto_nn_fit=nnetar(elec_trn[,"power"])
auto_nn_pred  <- forecast(auto_nn_fit, h=test_h)
auto_nn_pred$test_rmse = rmse(elec_test[,"power"],auto_nn_pred$mean)
auto_nn_pred$test_rmse

## [1] 66.4517
```

RMSE is quite bad.

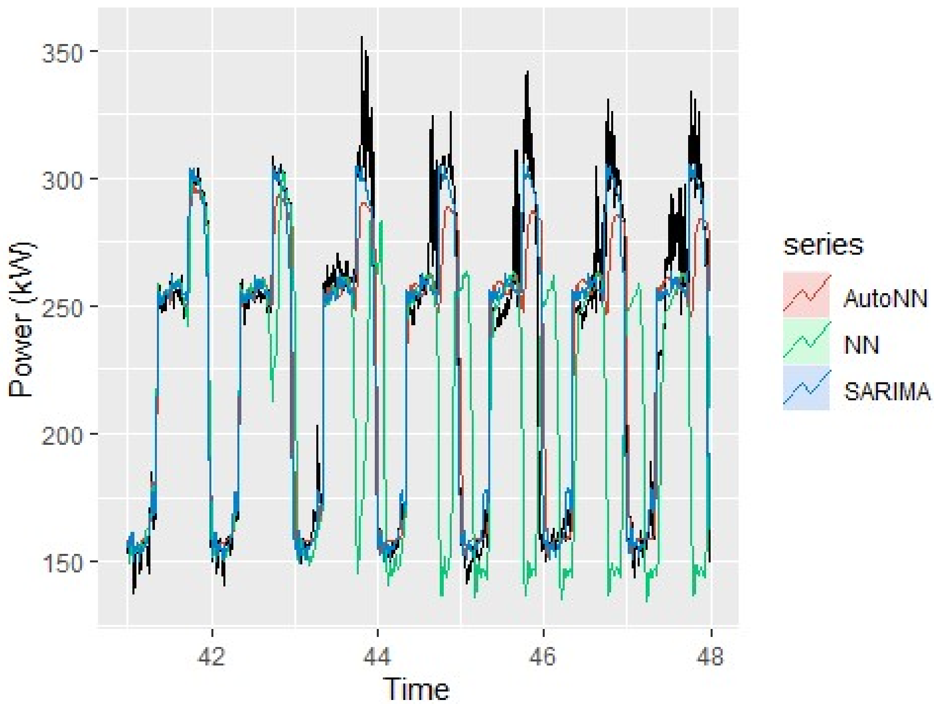
We now attempt a manual fitting.

We force the non-sesonal lag to 7 and the sesonal lag to 96

```
#Manual
nn_fit=nnetar(elec_trn[,"power"], p=7, q=96)
nn_pred  <- forecast(nn_fit, h=test_h)
nn_fit$test_rmse = rmse(elec_test[,"power"],nn_pred$mean)
nn_fit$test_rmse

## [1] 22.90166
```

RMSE is also quite bad. We plot the series for comparison:

```
autoplot (elec_test[,"power"], ylab = 'Power (kW)') +
  autolayer(nn_pred, series='AutoNN',PI=FALSE) +
  autolayer(auto_nn_pred, series='NN',PI=FALSE) +
  autolayer(arima_pred, series='SARIMA',PI=FALSE)
```



In both cases we have worst errors than with the manual arima, so we lean towards this model for the prediction without regressors

```
summary(arima_fit)
```

```
## Series: elec_trn[, "power"]
## ARIMA(6,1,7)(0,1,1)[96]
##
## Coefficients:
##          ar1      ar2     ar3     ar4     ar5     ar6      ma1     ma2
ma3
##       0.3768  -0.8440  0.2438  0.0781  0.1233  0.0737  -0.6262  0.8745  -
0.5230
## s.e.  1.0878   0.2373  0.8519  0.1744  0.0643  0.2423   1.0888  0.5048
0.8466
##          ma4      ma5      ma6     ma7     sma1
##      -0.3580  -0.0526  -0.3437  0.0805  -0.8504
## s.e.  0.4559   0.3892   0.2478  0.2835   0.0094
##
## sigma^2 estimated as 53.22:  log likelihood=-12788.2
```

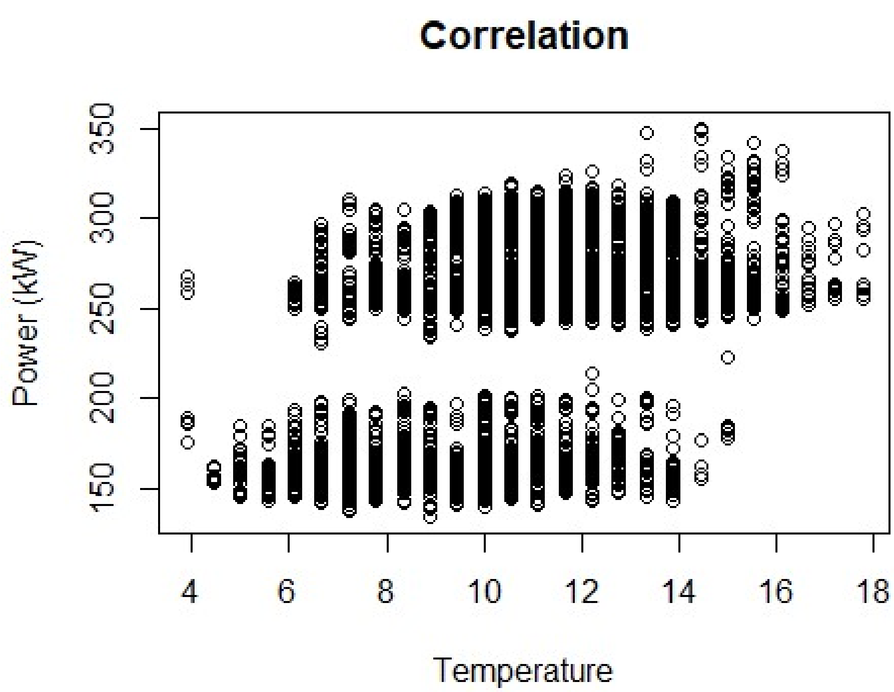
```
## AIC=25606.39   AICc=25606.52   BIC=25699.79
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE    MAPE      MASE
## Training set 0.02477745 7.189082 4.547619 -0.04375849 2.08998 0.5638977
##                       ACF1
## Training set -0.0003040997
```

Modelization with regressors

Initial Analysis

I now attempt to integrate the temperature regressor, for which I investigate the correlation between temperature and power:

```
plot(elec_trn[,"temp"],elec_trn[,"power"], ylab = 'Power (kW)',
xlab="Temperature", main='Correlation')
```




```
cor(elec_trn[,"temp"], elec_trn[,"power"], method=c("pearson", "kendall",
"spearman"))

## [1] 0.4576452
```

There seems to be a valuable a correlation between the temperature and the power consumptions.

Neural Networks

Having identify a correlation, we start by integrating them to the neural network model:

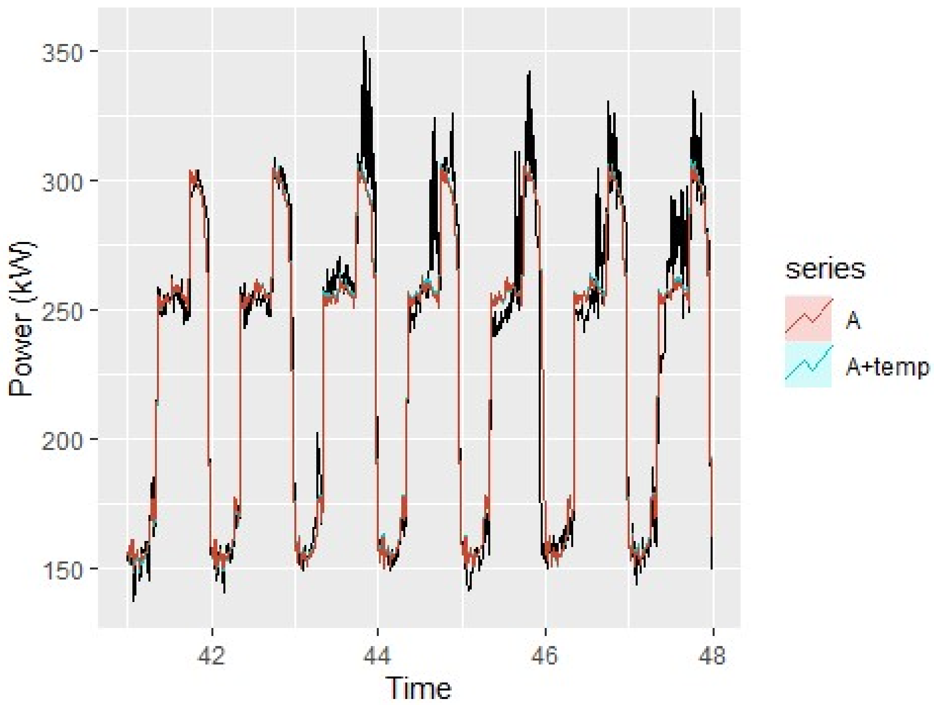
```
nn_reg_fit <- nnetar(elec_trn[,"power"], p=7, q=96, xreg=elec_trn[,"temp"])
nn_reg_pred  <- forecast(nn_reg_fit, h=test_h, xreg=elec_test[,"temp"] )
nn_reg_fit$test_rmse <- rmse(elec_test[,"power"],nn_reg_pred$mean)
nn_reg_fit$test_rmse

## [1] 21.97418
```

Still not very convincing. I come back to our prefered ARIMA model and introduce the regressors:

```
arima_reg_fit  <- Arima(elec_trn[,"power"], order= c(6,1,7),
seasonal=c(0,1,1), xreg=elec_trn[,'temp'])
arima_reg_pred <- forecast(arima_reg_fit,h=test_h,xreg=elec_test[,'temp'])
arima_reg_fit$test_rms <- rmse(elec_test["power"],arima_reg_pred$mean)

checkresiduals(arima_reg_fit, plot=FALSE)

##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(6,1,7)(0,1,1)[96] errors
## Q* = 232.41, df = 177, p-value = 0.00328
##
## Model df: 15.   Total lags used: 192

rmse(elec_test[,"power"],arima_reg_pred$mean)

## [1] 14.30219
```

The regressors seem to slightly improve the model. Next I plot the comparisson with our best model with and without regressors.

```
autoplot (elec_test[,"power"], ylab = 'Power (kW)') +
  autolayer(arima_reg_pred, series='A+temp',PI=FALSE) +
  autolayer(arima_pred, series='A',PI=FALSE)
```



It seems also that the SARIMA fits a better model than Neural Networks when considering regressors.

Full training

I now proceed re-train our SARIMA models using the full dataset:

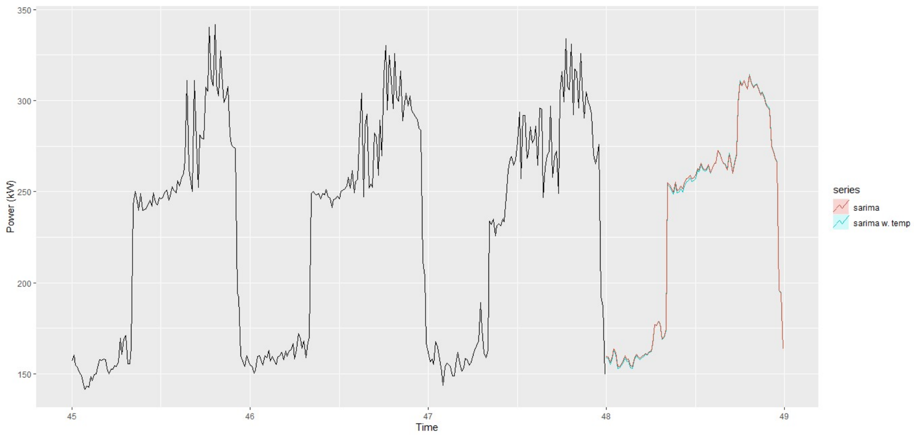
```
#Retraining with full dataset
f_arima_fit <- Arima(elec[,"power"], order= c(6,1,7), seasonal=c(0,1,1))
f_arima_pred <-forecast(f_arima_fit,h=96)

#Get the temperature for the day 48
f_temp <- window(elec[,"temp"], start=c(48, 1), end=c(48,96))

#Predict day 48
f_arima_reg_fit  <- Arima(elec[,"power"], order= c(6,1,7), seasonal=c(0,1,1),
xreg=elec[,'temp'])
f_arima_reg_pred <- forecast(f_arima_reg_fit,h=96,xreg=f_temp)
```

Final prediction

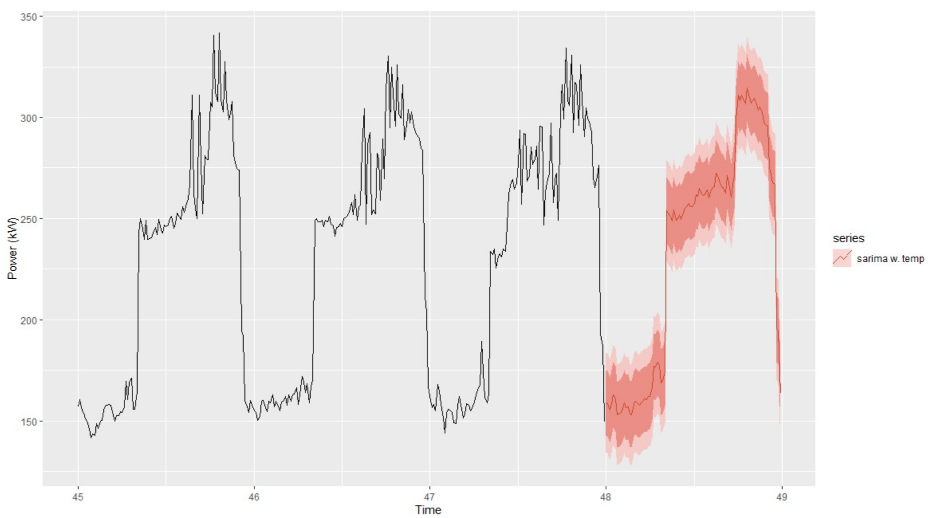
Et voilà les prédictions:

```
autoplot(window(elec[,"power"], start=c(40, 1), end=c(47,96)), ylab = 'Power
(kW)') +
```

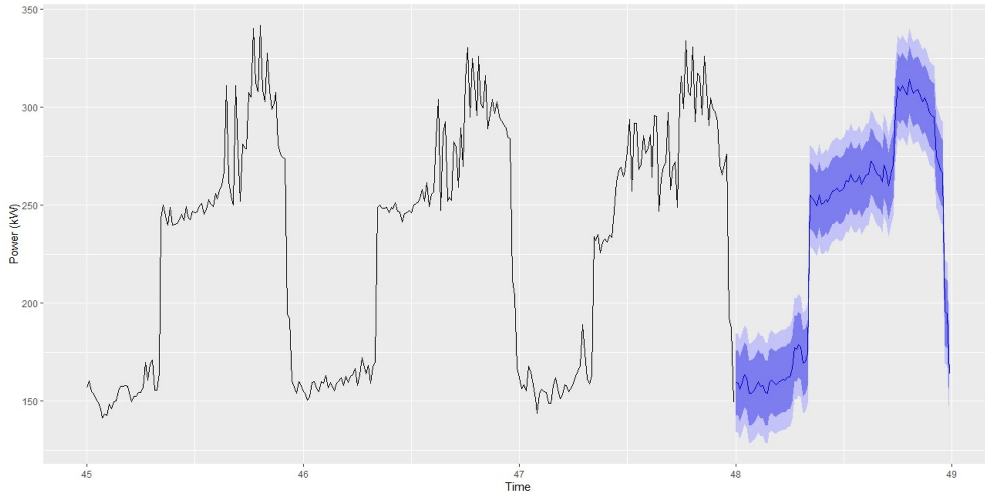
```
autolayer(f_arima_reg_pred, series='sarima w. temp',PI=FALSE) +
autolayer(f_arima_pred, series='sarima',PI=FALSE)
```



«`Temp`» Regressed series:



Non «`temp`» regressed series:



f_arima_reg_pred («temp» regressed)

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 48.00000       158.9555 142.8802 175.0307 134.3705 183.5404
## 48.01042       158.6094 142.4917 174.7271 133.9595 183.2593
## 48.02083       155.4048 139.2562 171.5534 130.7077 180.1019
## 48.03125       158.7069 142.5255 174.8884 133.9595 183.4543
## 48.04167       162.8035 146.6085 178.9984 138.0354 187.5715
## 48.05208       160.9309 144.7237 177.1380 136.1441 185.7176
## 48.06250       153.0796 136.8618 169.2974 128.2767 177.8826
## 48.07292       153.2632 137.0378 169.4886 128.4486 178.0778
## 48.08333       154.3684 138.1345 170.6022 129.5408 179.1959
## 48.09375       156.0648 139.8251 172.3046 131.2282 180.9014
## 48.10417       158.8808 142.6362 175.1254 134.0368 183.7248
## 48.11458       156.5545 140.3046 172.8043 131.7025 181.4064
## 48.12500       156.9005 140.6455 173.1555 132.0407 181.7603
## 48.13542       153.4844 137.2243 169.7444 128.6168 178.3520
## 48.14583       153.1083 136.8438 169.3727 128.2340 177.9826
## 48.15625       158.1403 141.8719 174.4087 133.2599 183.0207
## 48.16667       159.8817 143.6090 176.1544 134.9948 184.7686
## 48.17708       158.8274 142.5501 175.1046 133.9335 183.7212
## 48.18750       157.8600 141.5784 174.1417 132.9593 182.7607
## 48.19792       159.0563 142.7706 175.3421 134.1495 183.9632
## 48.20833       159.7521 143.4625 176.0417 134.8394 184.6649
## 48.21875       160.9074 144.6137 177.2010 135.9883 185.8264
## 48.22917       160.5064 144.2083 176.8045 135.5806 185.4321
## 48.23958       162.0047 145.7024 178.3070 137.0725 186.9370
## 48.25000       162.1335 145.8272 178.4397 137.1951 187.0718
## 48.26042       166.3992 150.0891 182.7094 141.4550 191.3435
## 48.27083       177.0205 160.7062 193.3348 152.0700 201.9710
## 48.28125       176.5871 160.2685 192.9056 151.6300 201.5442
## 48.29167       178.9504 162.6277 195.2731 153.9869 203.9139
## 48.30208       177.3273 161.0006 193.6539 152.3578 202.2968
## 48.31250       168.9612 152.6306 185.2918 143.9857 193.9367
```

```
## 48.32292       169.9317 153.5970 186.2664 144.9499 194.9135
## 48.33333       173.8609 157.5219 190.1998 148.8726 198.8492
## 48.34375       254.0579 237.7148 270.4010 229.0633 279.0525
## 48.35417       252.8300 236.4830 269.1770 227.8294 277.8306
## 48.36458       250.9927 234.6417 267.3437 225.9860 275.9994
## 48.37500       248.7467 232.3916 265.1018 223.7337 273.7597
## 48.38542       254.1102 237.7509 270.4695 229.0908 279.1296
## 48.39583       249.2382 232.8748 265.6015 224.2125 274.2638
## 48.40625       249.8506 233.4832 266.2179 224.8189 274.8823
## 48.41667       251.7328 235.3614 268.1041 226.6950 276.7706
## 48.42708       249.7162 233.3408 266.0917 224.6721 274.7603
## 48.43750       252.6230 236.2434 269.0026 227.5725 277.6734
## 48.44792       255.2311 238.8475 271.6148 230.1745 280.2878
## 48.45833       255.8947 239.5071 272.2824 230.8320 280.9575
## 48.46875       257.3697 240.9781 273.7614 232.3008 282.4386
## 48.47917       255.7261 239.3303 272.1218 230.6509 280.8012
## 48.48958       256.0213 239.6214 272.4212 230.9398 281.1028
## 48.50000       257.3157 240.9117 273.7196 232.2280 282.4033
## 48.51042       261.8104 245.4024 278.2183 236.7166 286.9042
## 48.52083       260.9810 244.5690 277.3929 235.8810 286.0809
## 48.53125       264.9353 248.5193 281.3514 239.8291 290.0415
## 48.54167       262.0612 245.6411 278.4814 236.9488 287.1737
## 48.55208       261.2267 244.8025 277.6509 236.1080 286.3453
## 48.56250       261.7917 245.3635 278.2199 236.6670 286.9165
## 48.57292       264.2451 247.8129 280.6774 239.1142 289.3761
## 48.58333       260.0162 243.5799 276.4526 234.8791 285.1534
## 48.59375       262.9473 246.5069 279.3877 237.8039 288.0907
## 48.60417       265.4283 248.9838 281.8727 240.2787 290.5778
## 48.61458       265.6634 249.2149 282.1118 240.5077 290.8191
## 48.62500       272.4815 256.0290 288.9340 247.3196 297.6434
## 48.63542       271.0458 254.5892 287.5023 245.8776 296.2139
## 48.64583       267.6924 251.2318 284.1530 242.5180 292.8667
## 48.65625       265.8007 249.3360 282.2653 240.6202 290.9811
## 48.66667       265.2397 248.7711 281.7084 240.0531 290.4263
## 48.67708       262.3982 245.9255 278.8709 237.2054 287.5910
## 48.68750       271.1382 254.6614 287.6149 245.9392 296.3372
## 48.69792       267.5802 251.0994 284.0610 242.3750 292.7854
## 48.70833       260.4831 243.9983 276.9679 235.2717 285.6944
## 48.71875       266.3463 249.8574 282.8351 241.1288 291.5638
## 48.72917       271.0742 254.5813 287.5671 245.8505 296.2978
## 48.73958       298.7775 282.2806 315.2744 273.5477 324.0074
## 48.75000       311.2170 294.7160 327.7179 285.9809 336.4530
## 48.76042       308.4332 291.9282 324.9382 283.1910 333.6754
## 48.77083       311.1123 294.6033 327.6213 285.8639 336.3606
## 48.78125       308.8613 292.3483 325.3743 283.6068 334.1158
## 48.79167       306.5455 290.0284 323.0626 281.2848 331.8062
## 48.80208       314.4608 297.9397 330.9819 289.1939 339.7276
## 48.81250       309.9404 293.4153 326.4655 284.6674 335.2134
## 48.82292       307.2716 290.7424 323.8007 281.9924 332.5507
## 48.83333       308.4303 291.8972 324.9635 283.1450 333.7156
```

```
## 48.84375       309.4018 292.8646 325.9390 284.1104 334.6933
## 48.85417       306.4213 289.8801 322.9626 281.1237 331.7190
## 48.86458       303.3907 286.8455 319.9359 278.0870 328.6945
## 48.87500       305.0136 288.4643 321.5628 279.7037 330.3235
## 48.88542       302.4121 285.8588 318.9653 277.0960 327.7281
## 48.89583       298.1919 281.6346 314.7492 272.8697 323.5141
## 48.90625       296.6382 280.0769 313.1995 271.3099 321.9666
## 48.91667       295.7638 279.1984 312.3291 270.4293 321.0982
## 48.92708       275.3135 258.7442 291.8828 249.9729 300.6541
## 48.93750       271.8184 255.2451 288.3917 246.4716 297.1651
## 48.94792       267.8548 251.2775 284.4322 242.5019 293.2077
## 48.95833       266.8532 250.2718 283.4346 241.4942 292.2122
## 48.96875       195.6793 179.0939 212.2647 170.3141 221.0445
## 48.97917       194.5470 177.9576 211.1364 169.1757 219.9183
## 48.98958       163.7026 147.1092 180.2960 138.3252 189.0800
```

f_arima_pred (non «temp» regressed)

```
##           Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 48.00000       159.6136 143.2033 176.0240 134.5162 184.7111
## 48.01042       159.3407 142.8805 175.8008 134.1670 184.5143
## 48.02083       156.1609 139.6635 172.6584 130.9303 181.3916
## 48.03125       159.4896 142.9527 176.0264 134.1986 184.7805
## 48.04167       163.5774 147.0224 180.1323 138.2587 188.8960
## 48.05208       161.9018 145.3302 178.4734 136.5577 187.2459
## 48.06250       154.0494 137.4629 170.6360 128.6825 179.4163
## 48.07292       154.2504 137.6524 170.8485 128.8659 179.6350
## 48.08333       155.3666 138.7558 171.9774 129.9625 180.7707
## 48.09375       157.2020 140.5818 173.8222 131.7836 182.6203
## 48.10417       160.0060 143.3779 176.6342 134.5755 185.4366
## 48.11458       157.6822 141.0454 174.3191 132.2384 183.1261
## 48.12500       158.0323 141.3871 174.6775 132.5757 183.4890
## 48.13542       154.4226 137.7690 171.0763 128.9531 179.8922
## 48.14583       154.0504 137.3892 170.7115 128.5693 179.5314
## 48.15625       159.0795 142.4114 175.7476 133.5879 184.5711
## 48.16667       160.8156 144.1401 177.4910 135.3127 186.3185
## 48.17708       159.5618 142.8786 176.2449 134.0471 185.0765
## 48.18750       158.5978 141.9071 175.2884 133.0716 184.1239
## 48.19792       159.8005 143.1028 176.4983 134.2635 185.3375
## 48.20833       160.4964 143.7919 177.2009 134.9490 186.0438
## 48.21875       161.3181 144.6064 178.0298 135.7598 186.8764
## 48.22917       160.9145 144.1954 177.6336 135.3449 186.4842
## 48.23958       162.4166 145.6902 179.1430 136.8358 187.9974
## 48.25000       162.5523 145.8190 179.2856 136.9609 188.1437
## 48.26042       166.5073 149.7671 183.2474 140.9054 192.1092
## 48.27083       177.1194 160.3721 193.8666 151.5066 202.7321
## 48.28125       176.6859 159.9314 193.4405 151.0620 202.3098
## 48.29167       179.0458 162.2841 195.8076 153.4110 204.6807
## 48.30208       177.7817 161.0131 194.5504 152.1363 203.4272
## 48.31250       169.4259 152.6503 186.2014 143.7699 195.0819
```

```
## 48.32292        170.3823 153.5997 187.1650 144.7155 196.0492
## 48.33333        174.3058 157.5159 191.0957 148.6279 199.9837
## 48.34375        255.0260 238.2290 271.8229 229.3373 280.7147
## 48.35417        253.8043 237.0004 270.6082 228.1050 279.5036
## 48.36458        251.9646 235.1537 268.7754 226.2546 277.6745
## 48.37500        249.7168 232.8989 266.5348 223.9961 275.4376
## 48.38542        255.3529 238.5278 272.1779 229.6212 281.0846
## 48.39583        250.4839 233.6518 267.3161 224.7415 276.2264
## 48.40625        251.1057 234.2666 267.9447 225.3526 276.8588
## 48.41667        252.9867 236.1407 269.8327 227.2230 278.7505
## 48.42708        251.5537 234.7006 268.4068 225.7791 277.3282
## 48.43750        254.4525 237.5923 271.3127 228.6671 280.2379
## 48.44792        257.0627 240.1956 273.9299 231.2666 282.8589
## 48.45833        257.7281 240.8540 274.6023 231.9214 283.5349
## 48.46875        258.9451 242.0640 275.8262 233.1277 284.7625
## 48.47917        257.2956 240.4074 274.1837 231.4674 283.1238
## 48.48958        257.5867 240.6915 274.4820 231.7478 283.4257
## 48.50000        258.8839 241.9817 275.7861 233.0342 284.7336
## 48.51042        262.6319 245.7228 279.5411 236.7716 288.4923
## 48.52083        261.8161 244.9000 278.7322 235.9451 287.6871
## 48.53125        265.7527 248.8295 282.6758 239.8709 291.6344
## 48.54167        262.8765 245.9463 279.8067 236.9840 288.7690
## 48.55208        261.8544 244.9172 278.7915 235.9512 287.7575
## 48.56250        262.4235 245.4794 279.3676 236.5097 288.3373
## 48.57292        264.8641 247.9130 281.8152 238.9396 290.7885
## 48.58333        260.6402 243.6822 277.5983 234.7051 286.5754
## 48.59375        263.0803 246.1153 280.0454 237.1345 289.0262
## 48.60417        265.5608 248.5888 282.5328 239.6043 291.5173
## 48.61458        265.7976 248.8186 282.7766 239.8305 291.7648
## 48.62500        272.5925 255.6065 289.5784 246.6147 298.5703
## 48.63542        270.8327 253.8397 287.8256 244.8442 296.8211
## 48.64583        267.4857 250.4858 284.4856 241.4866 293.4848
## 48.65625        265.5994 248.5925 282.6062 239.5896 291.6091
## 48.66667        265.0424 248.0286 282.0561 239.0220 291.0627
## 48.67708        261.9208 244.9000 278.9415 235.8898 287.9518
## 48.68750        270.6378 253.6101 287.6655 244.5961 296.6794
## 48.69792        267.0848 250.0502 284.1195 241.0326 293.1371
## 48.70833        259.9982 242.9566 277.0398 233.9353 286.0610
## 48.71875        265.6522 248.6036 282.7007 239.5787 291.7256
## 48.72917        270.4049 253.3494 287.4604 244.3208 296.4890
## 48.73958        298.0980 281.0355 315.1604 272.0033 324.1927
## 48.75000        310.5086 293.4392 327.5779 284.4032 336.6139
## 48.76042        308.3040 291.2277 325.3802 282.1880 334.4199
## 48.77083        310.9702 293.8870 328.0534 284.8437 337.0967
## 48.78125        308.7221 291.6320 325.8123 282.5850 334.8592
## 48.79167        306.4121 289.3150 323.5091 280.2644 332.5598
## 48.80208        314.1947 297.0907 331.2986 288.0364 340.3529
## 48.81250        309.6813 292.5704 326.7922 283.5125 335.8501
## 48.82292        307.0153 289.8975 324.1331 280.8359 333.1947
## 48.83333        308.1662 291.0415 325.2909 281.9762 334.3562
```

```
## 48.84375        309.0098 291.8782 326.1414 282.8092 335.2103
## 48.85417        306.0293 288.8907 323.1678 279.8182 332.2404
## 48.86458        303.0052 285.8598 320.1506 276.7835 329.2268
## 48.87500        304.6249 287.4726 321.7772 278.3927 330.8571
## 48.88542        301.6185 284.4593 318.7777 275.3758 327.8613
## 48.89583        297.4019 280.2358 314.5680 271.1486 323.6552
## 48.90625        295.8511 278.6781 313.0241 269.5872 322.1149
## 48.91667        294.9766 277.7968 312.1565 268.7023 321.2510
## 48.92708        274.9848 257.7980 292.1716 248.6999 301.2697
## 48.93750        271.4924 254.2987 288.6860 245.1970 297.7878
## 48.94792        267.5258 250.3253 284.7263 241.2198 293.8317
## 48.95833        266.5265 249.3191 283.7339 240.2100 292.8430
## 48.96875        195.8236 178.6093 213.0379 169.4966 222.1505
## 48.97917        194.6970 177.4758 211.9181 168.3595 221.0345
## 48.98958        163.8526 146.6246 181.0806 137.5046 190.2006
```