Time series forecasting

12/20/2020

```
library(forecast)
```

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

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

```
dataset <- read.xlsx('elec-train.xlsx')
head(dataset, 25)
```

```
##        Timestamp Power.(kW) Temp.(C°)
## 1  1/1/2010 1:15      165.1 10.555556
## 2  1/1/2010 1:30      151.6 10.555556
## 3  1/1/2010 1:45      146.9 10.555556
## 4  1/1/2010 2:00      153.7 10.555556
## 5  1/1/2010 2:15      153.8 10.555556
## 6  1/1/2010 2:30      159.0 10.555556
## 7  1/1/2010 2:45      157.7 10.555556
## 8  1/1/2010 3:00      163.2 10.555556
## 9  1/1/2010 3:15      151.7 10.000000
## 10 1/1/2010 3:30      148.7 10.000000
## 11 1/1/2010 3:45      155.1 10.000000
## 12 1/1/2010 4:00      161.5 10.000000
## 13 1/1/2010 4:15      161.5 10.000000
## 14 1/1/2010 4:30      162.0 10.000000
## 15 1/1/2010 4:45      166.0 10.000000
## 16 1/1/2010 5:00      159.0 10.000000
## 17 1/1/2010 5:15      158.6 10.000000
## 18 1/1/2010 5:30      152.1 10.000000
## 19 1/1/2010 5:45      162.0 10.000000
## 20 1/1/2010 6:00      164.4 10.000000
## 21 1/1/2010 6:15      163.6  9.444444
## 22 1/1/2010 6:30      168.8  9.444444
## 23 1/1/2010 6:45      163.7  9.444444
## 24 1/1/2010 7:00      156.2  9.444444
## 25 1/1/2010 7:15      168.4 10.000000
```

Although power is measured once every 15 minutes, temperature is measured once per hour at most (sometimes even less often).

```
tail(dataset)
```

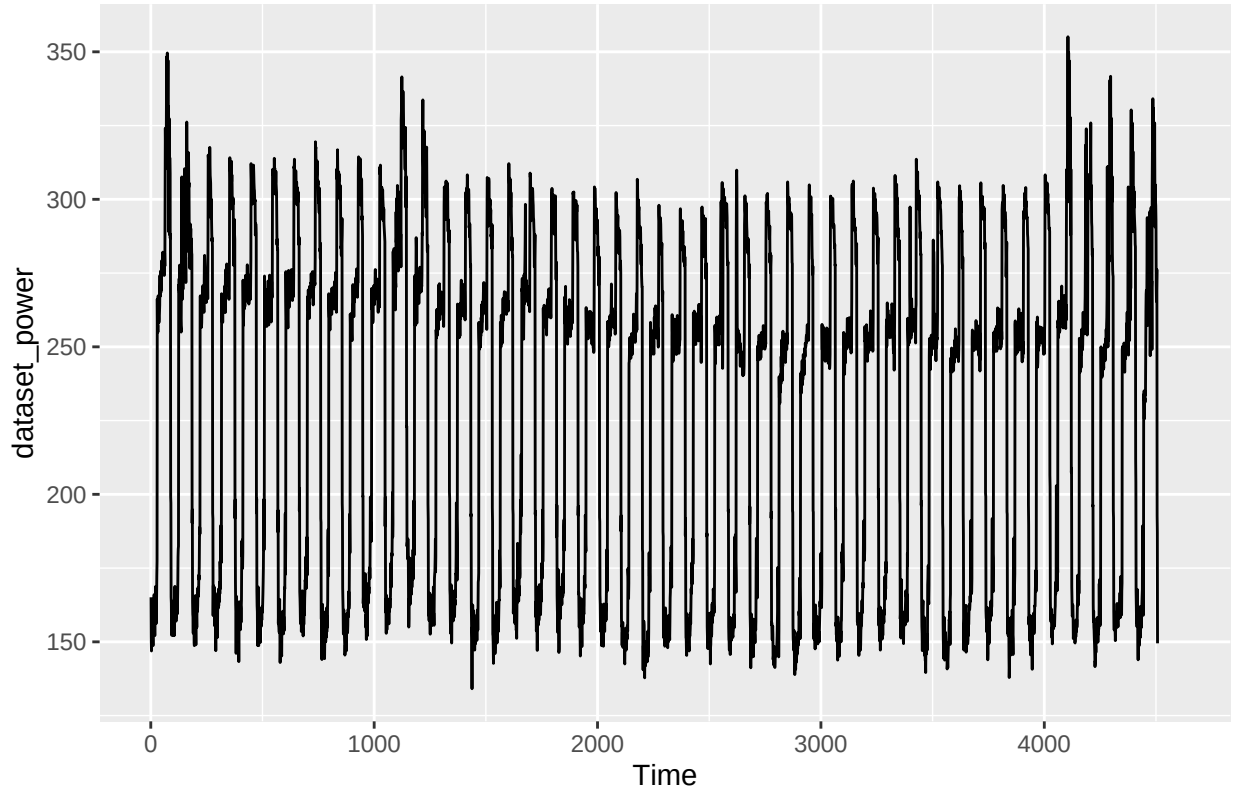
```
##            Timestamp Power.(kW) Temp.(C°)
## 4598 2/17/2010 22:30         NA  13.88889
## 4599 2/17/2010 22:45         NA  13.88889
```

```
## 4600 2/17/2010 23:00          NA  13.88889
## 4601 2/17/2010 23:15          NA  12.77778
## 4602 2/17/2010 23:30          NA  12.77778
## 4603 2/17/2010 23:45          NA  12.77778
```

The last 96 power values are missing, these are the values we're going to forecast (with and without covariant temperatures).

Let's plot the power data as a time series.

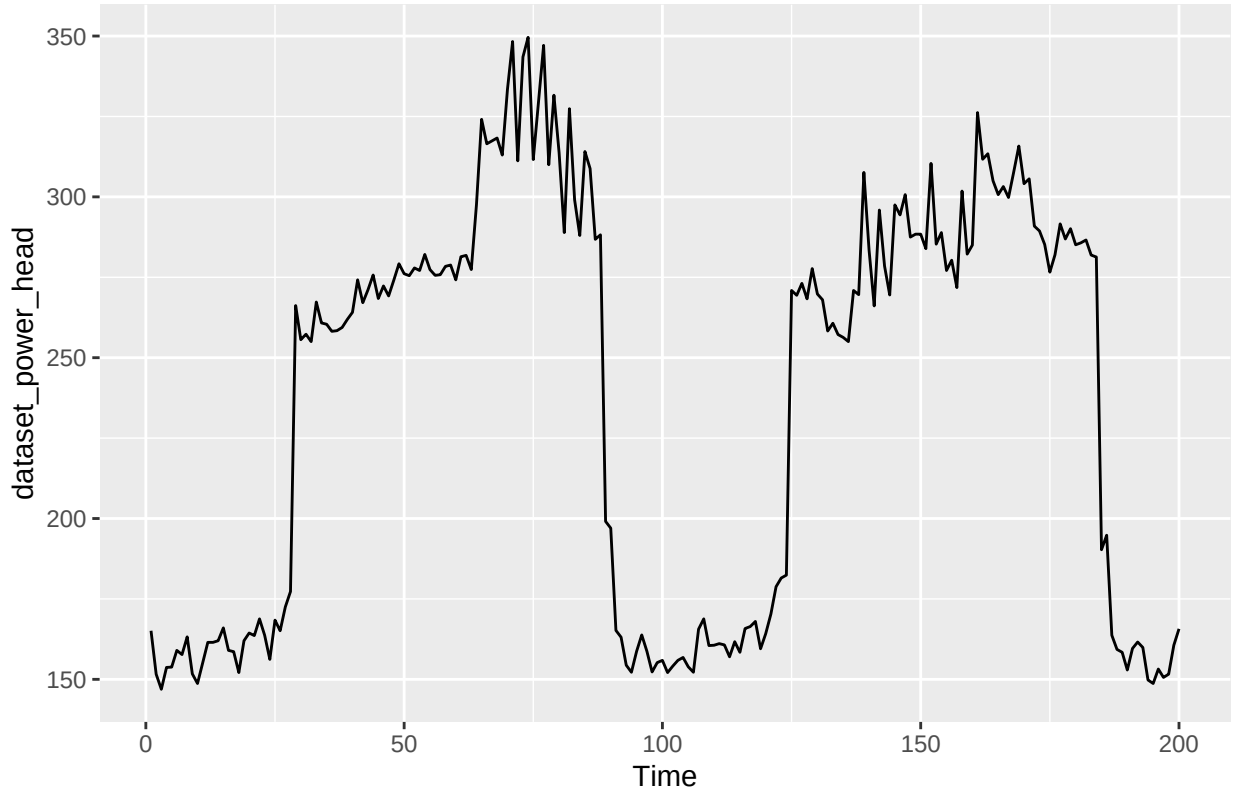
```
dataset_power <- ts(dataset$`Power.(kW)`)
autoplot(dataset_power)
```



It's difficult to see a long term trend here although we may expect a trend based on the month-to-month temperature evolution.

On the other hand we can definitely observe a cyclic pattern so let's visualize the data on a smaller time range.

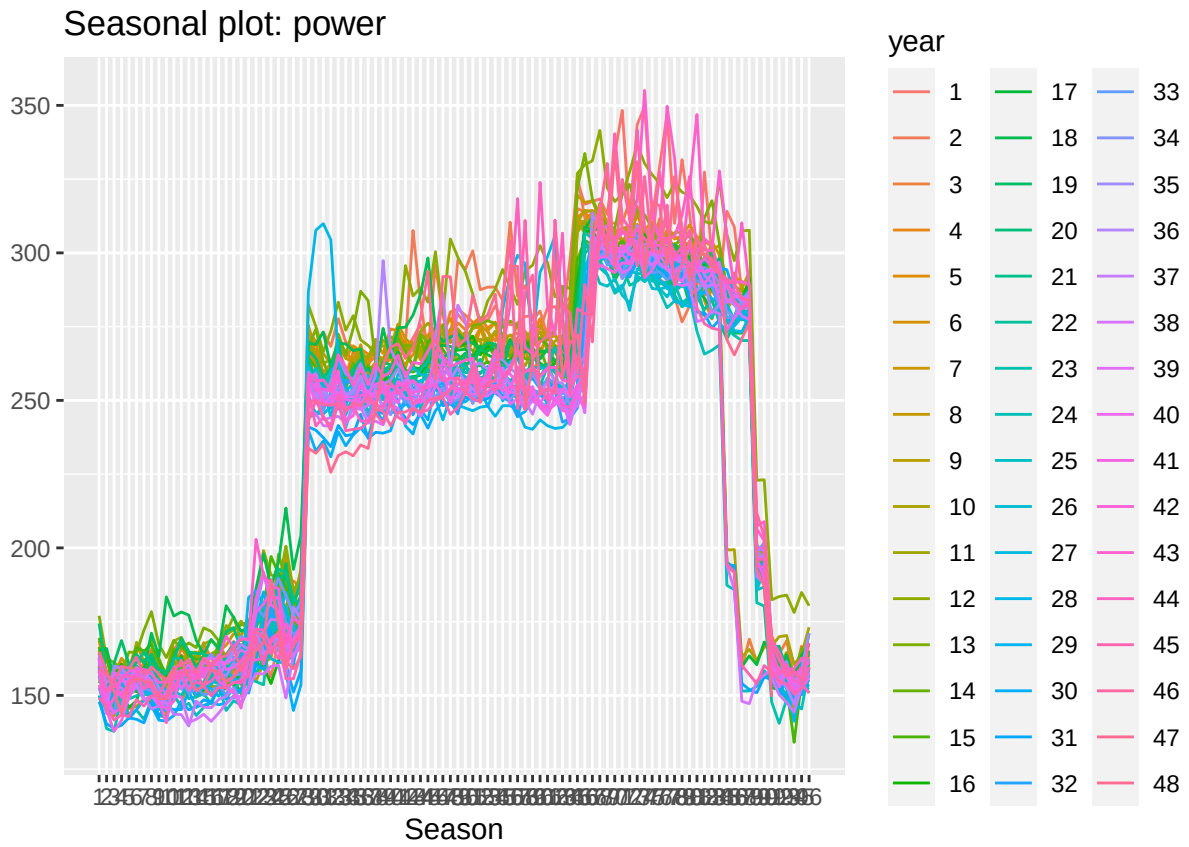
```
dataset_power_head <- head(dataset_power,200)
autoplot(dataset_power_head)
```



Cycles seem to fit 96 timeslots (i.e. 1 day = 24h * 4 * 15min) which makes sense, so let's check the seasonal pattern after extracting the power data as a time series of frequency 96.

```
power <- ts(dataset$`Power.(kW)`, frequency=96)
ggseasonplot(power)
```

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



The daily pattern is confirmed with low power consumption at nighttime, high consumption at daytime with an increase in the evening. This is typical of household power consumption although we may also expect a consumption surge early in the morning.

Now let's separate train and test datasets in order to build and assess our future models.

```
power_train <- window(power, start=c(1,1), end=c(46,91))
power_test <- window(power, start=c(46,92), end=c(47,91))
print(power_test)
```

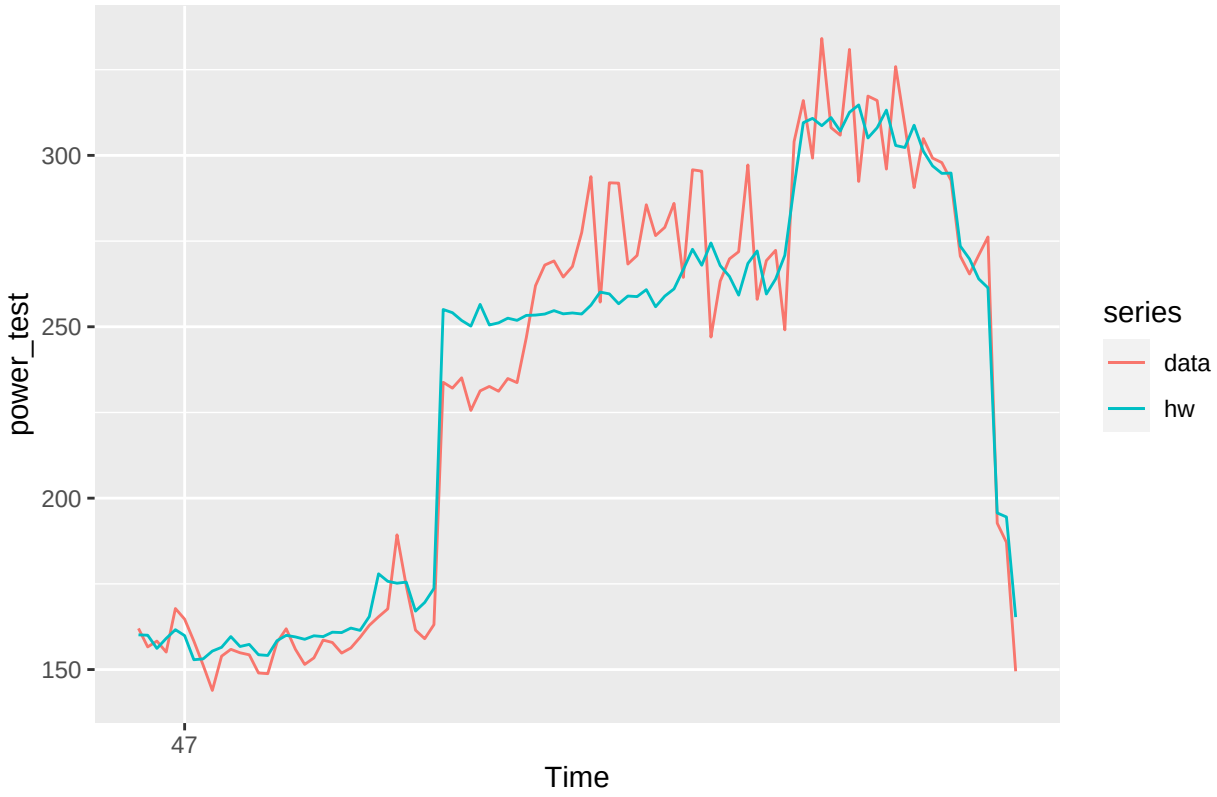
```
## Time Series:
## Start = c(46, 92)
## End = c(47, 91)
## Frequency = 96
##  [1] 162.0 156.6 158.3 155.1 167.8 164.7 158.3 151.3 143.9 153.9 155.9 154.9
## [13] 154.3 149.0 148.8 157.9 161.9 155.9 151.5 153.4 158.6 157.9 154.8 156.3
## [25] 159.4 162.9 165.4 167.7 189.3 174.4 161.5 159.0 163.1 233.8 232.1 235.1
## [37] 225.6 231.3 232.6 231.2 234.9 233.7 246.8 262.0 268.0 269.2 264.5 267.6
## [49] 277.5 293.8 257.2 292.0 291.9 268.3 270.8 285.6 276.6 279.0 286.0 264.4
## [61] 295.8 295.4 247.0 263.3 269.8 271.9 297.2 258.0 269.3 272.3 249.1 304.0
## [73] 316.0 299.2 334.1 308.1 305.9 330.9 292.4 317.3 316.0 296.0 325.9 308.7
## [85] 290.6 304.9 299.2 297.9 292.7 270.6 265.4 270.9 276.2 192.7 187.1 149.5
```

Forecasting power without temperature

Our first model is based on the Holt-Winters function. After playing with the alpha (smoothing factor), beta (double exponential smoothing) and gamma (seasonal component) values, we stop at alpha=0.00002, beta=1

and gamma=0.2.

```
hw_fit <- HoltWinters(power_train, alpha=0.00002, beta=1, gamma=0.2)
hw_prev <- forecast(hw_fit, h=96)
autoplot(power_test, series='data') + autolayer(hw_prev$mean, series='hw')
```




```
cat('RMSE:', sqrt(mean((hw_prev$mean-power_test)^2)))
```

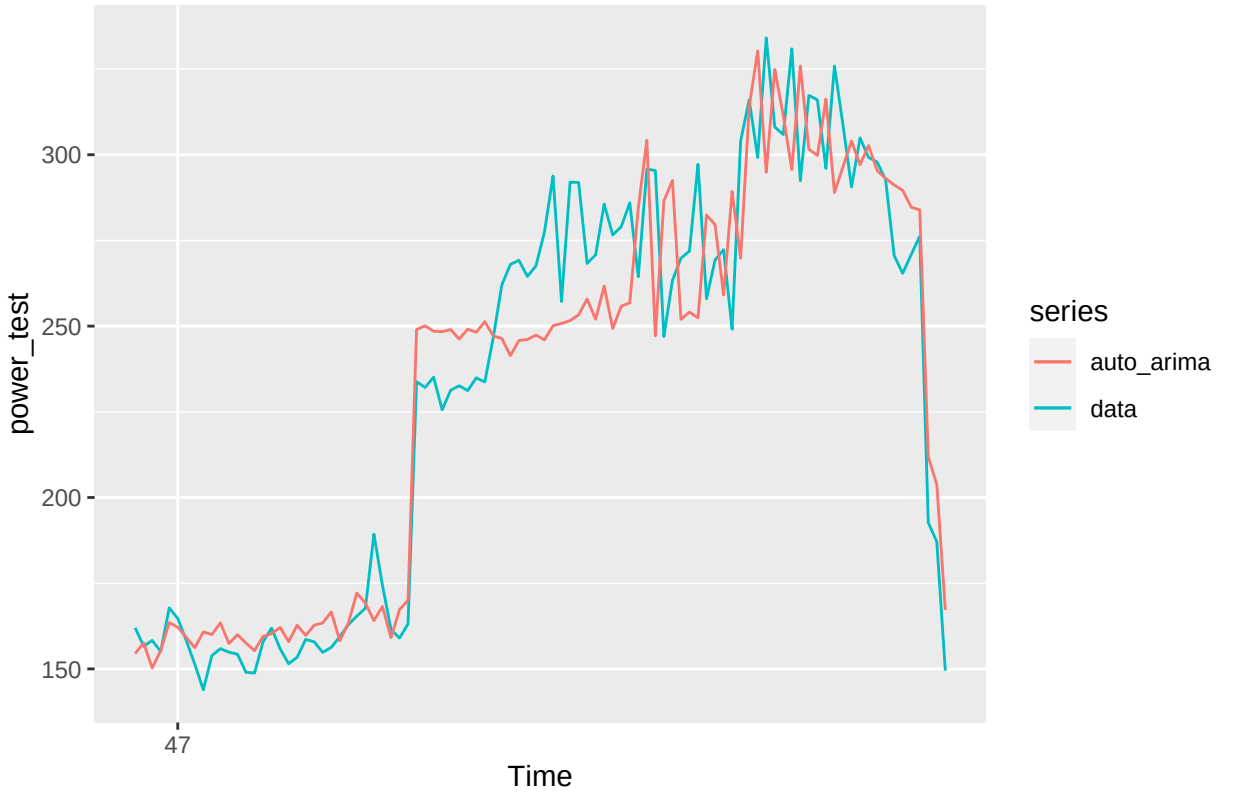
```
## RMSE: 14.52629
```

Our second model is based on automatic ARIMA. Even though the result is coherent, RMSE is worse than Holt-Winters because automatic parameters fail to catch the high noise level between 15 minutes slots.

```
auto_arima_fit <- auto.arima(power_train, lambda='auto')
auto_arima_prev <- forecast(auto_arima_fit, h=96)
print(auto_arima_fit)
```

```
## Series: power_train
## ARIMA(5,0,0)(0,1,0)[96]
## Box Cox transformation: lambda= 0.488894
##
## Coefficients:
##          ar1     ar2     ar3      ar4     ar5
##       0.7367  0.0825  0.0628  -0.2575  0.1327
## s.e.  0.0151  0.0184  0.0184   0.0185  0.0152
##
## sigma^2 estimated as 0.4246:  log likelihood=-4272.49
## AIC=8556.97   AICc=8556.99   BIC=8595.19
```

```
autoplot(power_test, series='data') + autolayer(auto_arima_prev$mean, series='auto_arima')
```

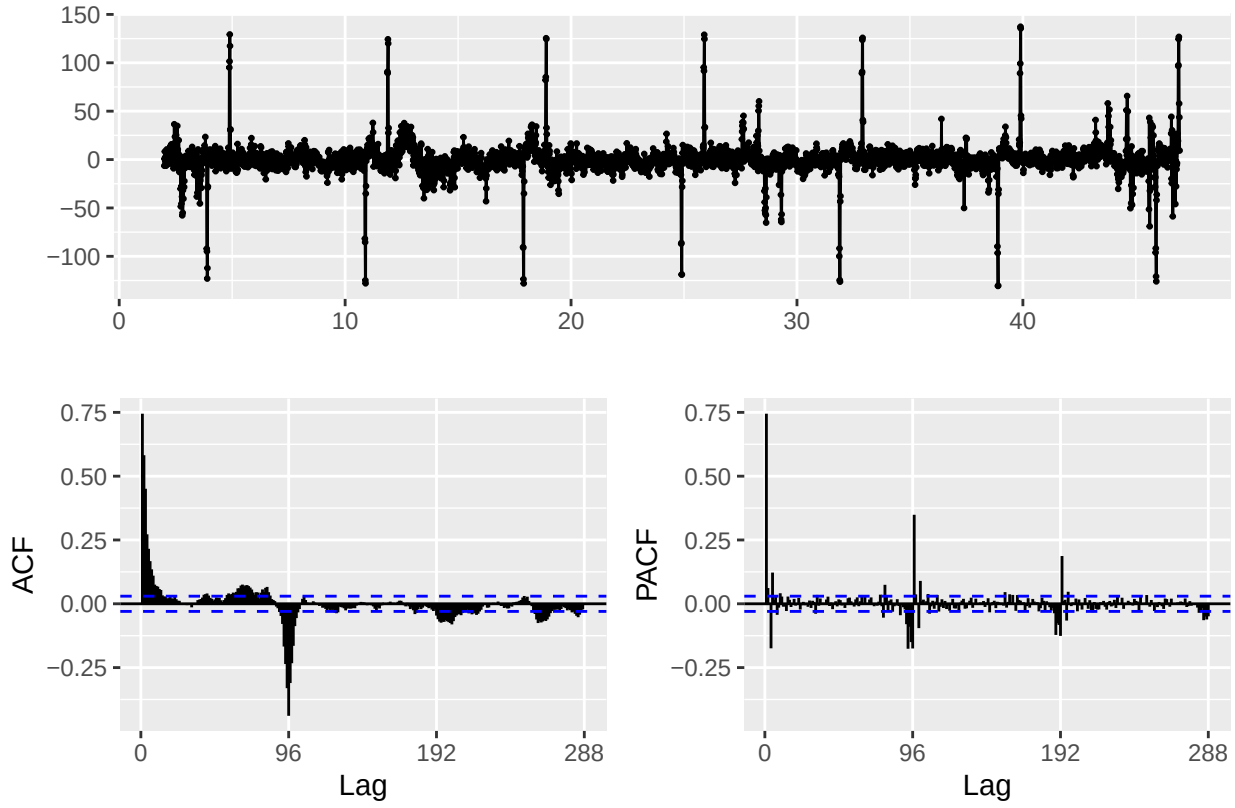



```
cat('RMSE:', sqrt(mean((auto_arima_prev$mean-power_test)^2)))
```

```
## RMSE: 19.78222
```

We try to improve ARIMA by handpicking parameters. In order to select good parameters we first remove the seasonal pattern from the dataset by differentiating with a lag of 96.

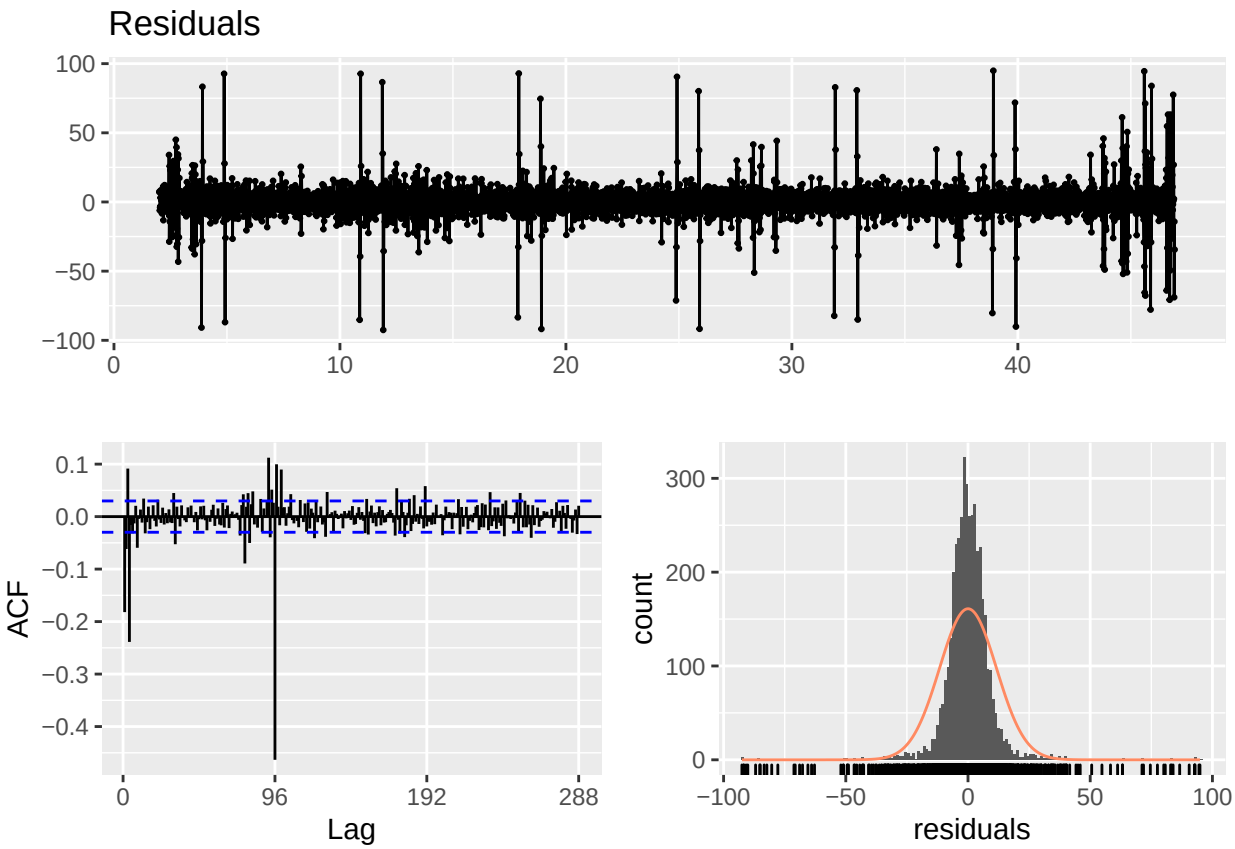
```
diff1 <- diff(power_train, lag=96)
ggtsdisplay(diff1)
```



From the ACF plot we observe a clear trend which we'll remove by differentiating again.

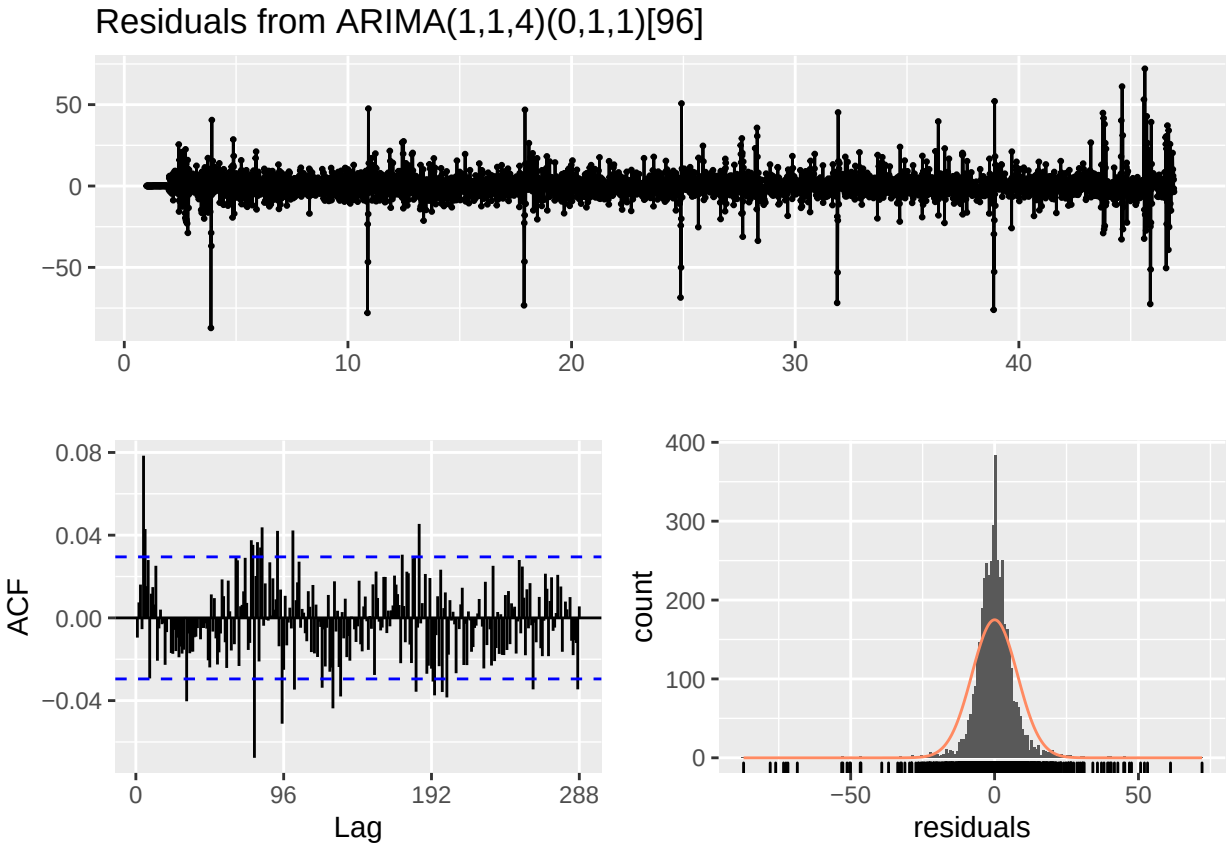
```
diff2 <- diff(diff1)
checkresiduals(diff2)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



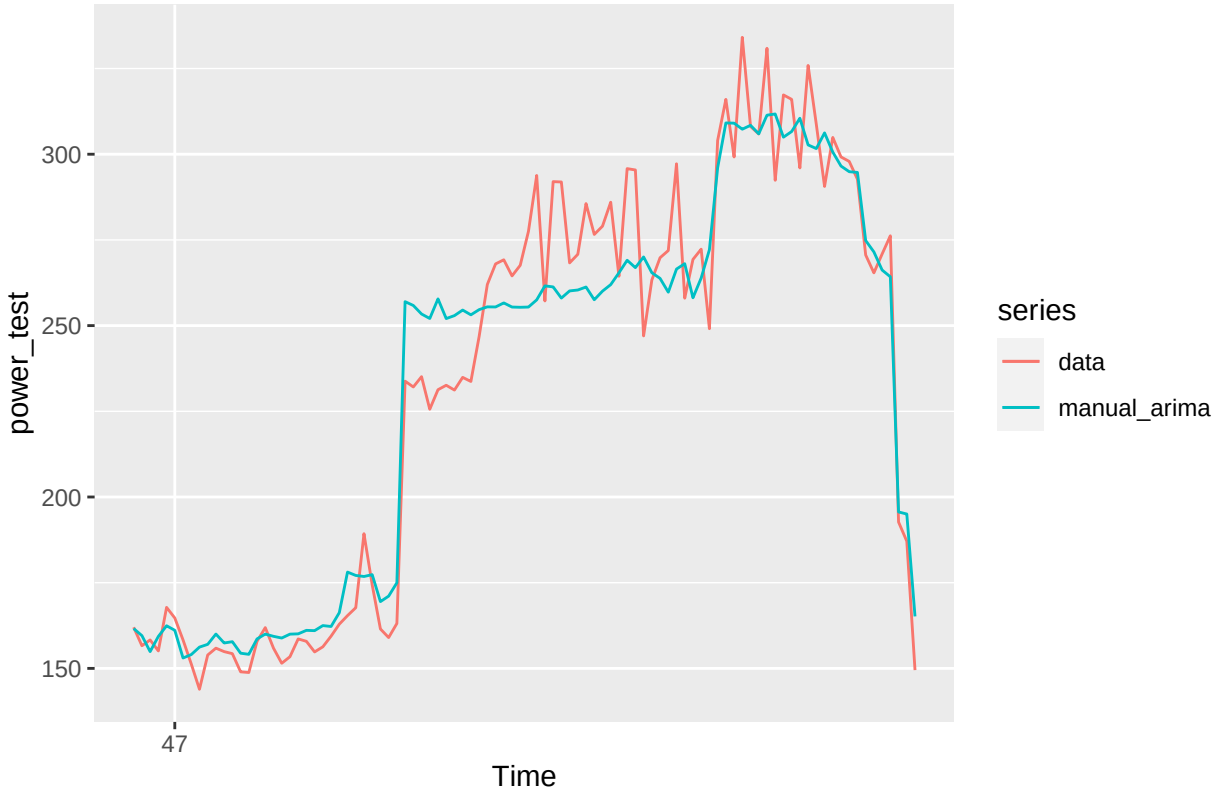
Although residuals are now correctly balanced and most of autocorrelation is included in the +/-5% range, we still observe a seasonal peak at lag 96 plus some bumps at lags 1 and 4. We use these values for our ARIMA model which results in a restrained autocorrelation and a RMSE similar to Holt-Winters.

```
manual_arima_fit <- Arima(power_train, order=c(1,1,4), seasonal=c(0,1,1))
manual_arima_prev <- forecast(manual_arima_fit, h=96)
checkresiduals(manual_arima_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,4)(0,1,1)[96]
## Q* = 337.5, df = 186, p-value = 7.25e-11
##
## Model df: 6.   Total lags used: 192
```

```
autoplot(power_test, series='data') + autolayer(manual_arima_prev$mean, series='manual_arima')
```




```
cat('RMSE:', sqrt(mean((manual_arima_prev$mean-power_test)^2)))
```

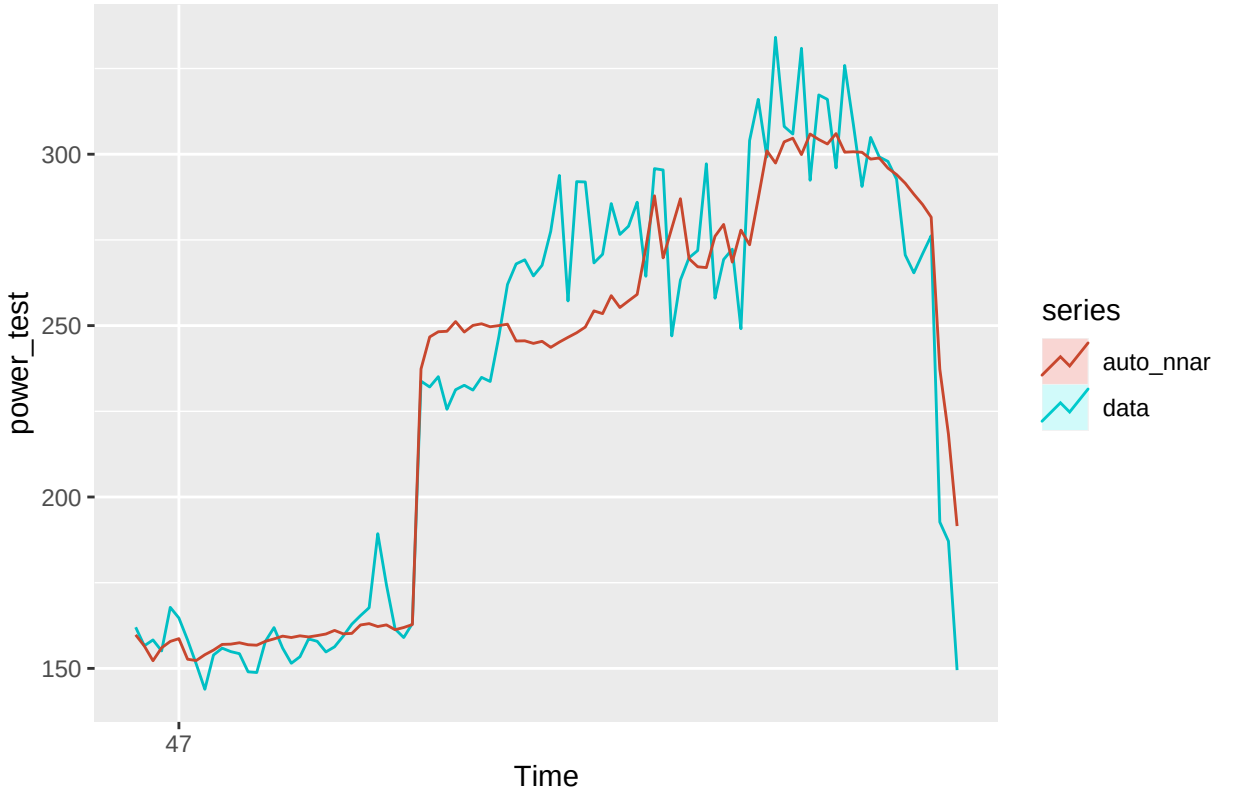
```
## RMSE: 14.4344
```

Our third model leverages the NNAR feed-forward neural network. Automatic model seems acceptable but less precise than the previous one.

```
auto_nnar_fit <- nnetar(power_train, lambda='auto')
auto_nnar_prev <- forecast(auto_nnar_fit, h=96)
print(auto_nnar_fit)
```

```
## Series: power_train
## Model:  NNAR(11,1,6)[96]
## Call:   nnetar(y = power_train, lambda = "auto")
##
## 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 0.3265
```

```
autoplot(power_test, series='data') + autolayer(auto_nnar_prev, series='auto_nnar')
```

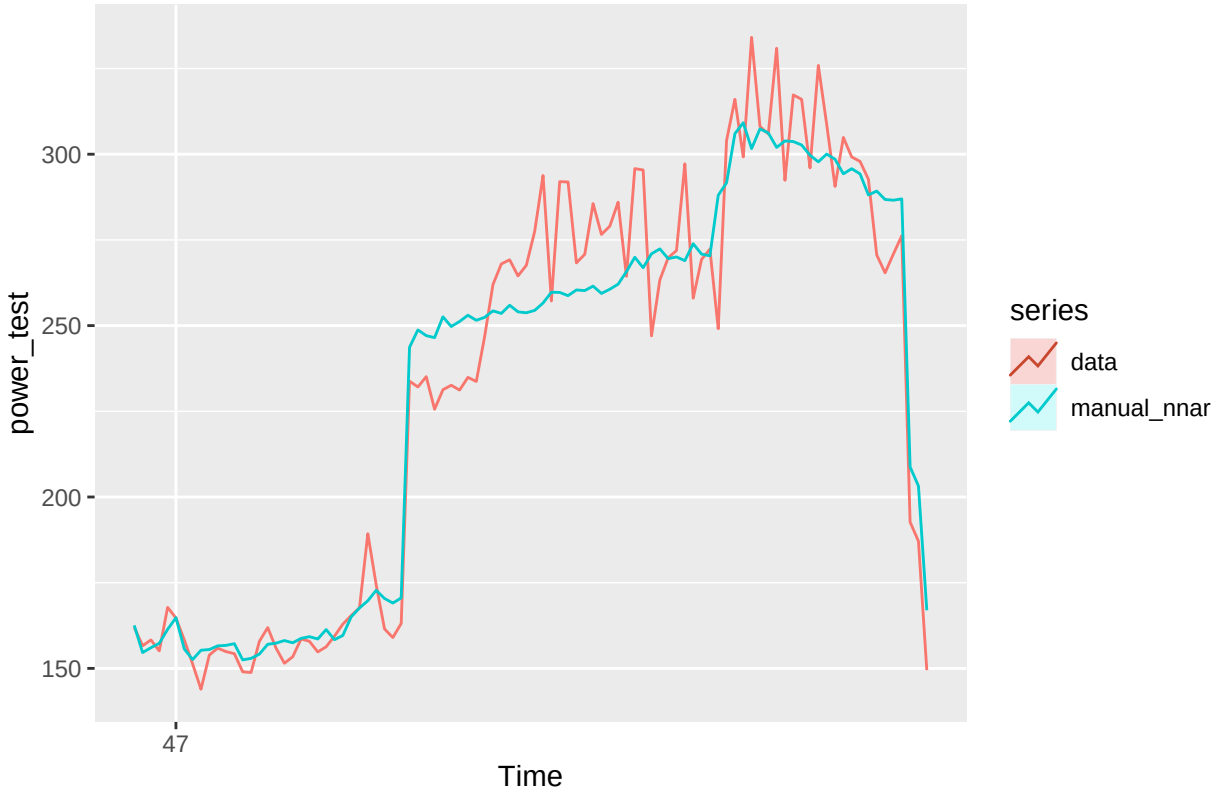



```
cat('RMSE:', sqrt(mean((auto_nnar_prev$mean-power_test)^2)))
```

```
## RMSE: 18.3433
```

We try to improve the NNAR model by changing the p (non-seasonal lags), P (seasonal lags) and size (nodes in the hidden layer) parameters. We increase values until the maximum number of parameters is reached, resulting in a RMSE similar to HW and ARIMA. Lambda is still set automatically.

```
manual_nnar_fit <- nnetar(power_train, 40, 4, 20, lambda='auto')
manual_nnar_prev <- forecast(manual_nnar_fit, h=96)
autoplot(power_test, series='data') + autolayer(manual_nnar_prev, series='manual_nnar')
```

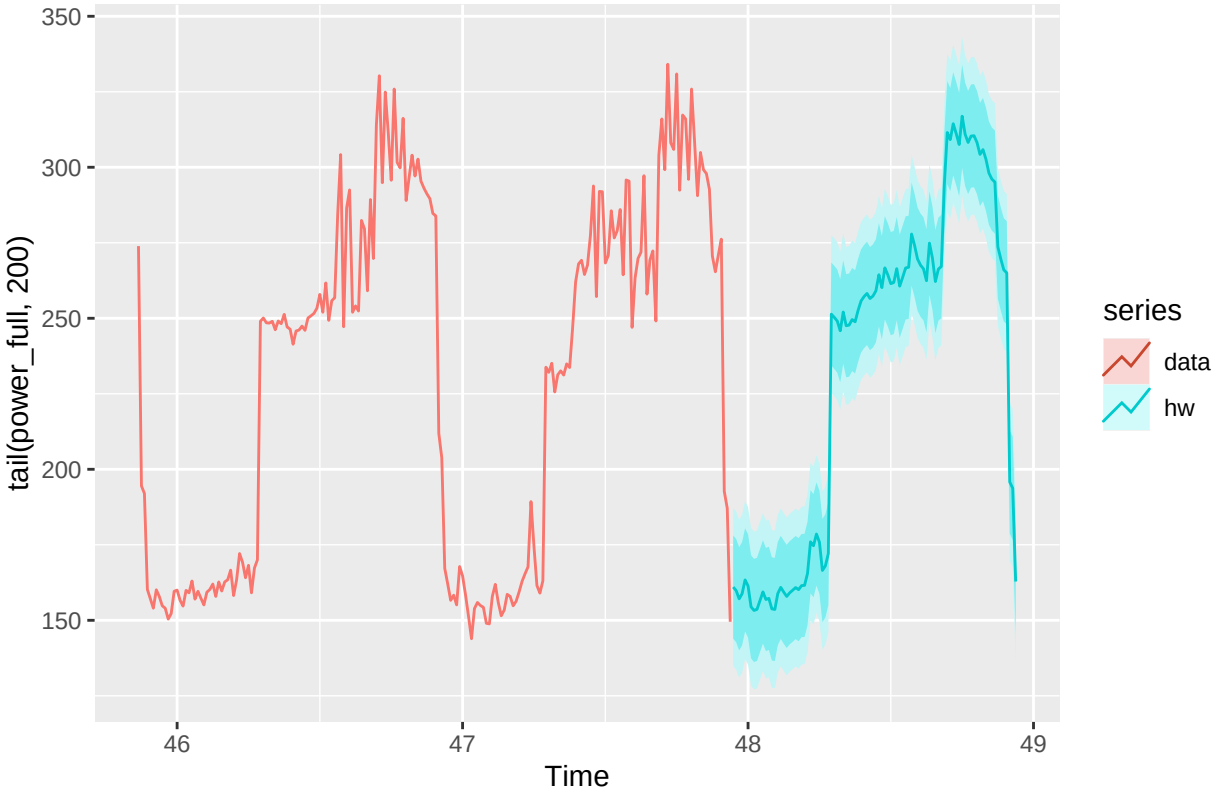



```
cat('RMSE:', sqrt(mean((manual_nnar_prev$mean-power_test)^2)))
```

```
## RMSE: 14.82741
```

Our three final Holt-Winters, ARIMA and NNAR models have very close RMSE values. However, Holt-Winters is faster and may be more resistant to overfitting so we use it in order to forecast the missing day with the full power dataset.

```
power_full <- window(power, start=c(1,1), end=c(47,91))
hw_full <- HoltWinters(power_full, alpha=0.00002, beta=1, gamma=0.2)
F02172010_without_temperature <- forecast(hw_full, h=96)
autoplot(tail(power_full, 200), series='data') + autolayer(F02172010_without_temperature, series='hw')
```




```
print(F02172010_without_temperature$mean)
```

```
## Time Series:
## Start = c(47, 92)
## End = c(48, 91)
## Frequency = 96
##  [1] 161.0025 159.8390 157.1028 158.7952 163.3724 161.3744 154.4675 153.2645
##  [9] 153.6087 156.4914 159.4032 156.8838 157.2615 153.7877 153.5841 158.8423
## [17] 160.9044 159.3321 157.8994 159.1024 159.9553 160.8425 160.1384 161.4926
## [25] 161.5751 165.5184 175.9845 174.6942 178.5714 175.8507 166.5040 168.0328
## [33] 172.1265 251.3714 250.2881 249.0704 245.8562 252.0663 247.5299 247.7510
## [41] 249.5816 248.8344 252.6152 255.7309 257.1756 258.2200 256.5357 257.3639
## [49] 259.0928 264.4188 260.1560 266.7015 264.3837 261.4666 261.8323 266.4200
## [57] 260.6283 263.5999 266.6819 266.8789 277.8920 274.0957 269.6061 267.5878
## [65] 266.3684 262.4198 274.8809 269.9561 262.1653 266.2647 267.1606 294.1551
## [73] 311.5118 309.1503 314.4177 311.1242 307.5354 316.9067 310.9324 308.2013
## [81] 310.3443 310.4495 308.1734 304.2591 305.8486 302.6518 298.0722 296.0744
## [89] 295.1093 273.6439 269.6108 266.0444 265.0361 195.7889 193.7313 162.8399
```

Forecasting power with temperature

```
temperature <- ts(dataset$`Temp.(C°)`, frequency=96)
temperature_train <- window(temperature, start=c(1,1), end=c(46,91))
temperature_test <- window(temperature, start=c(46,92), end=c(47,91))
print(temperature_test)
```

```
## Time Series:
## Start = c(46, 92)
## End = c(47, 91)
## Frequency = 96
##  [1] 12.22222 10.55556 10.55556 10.55556 10.55556 11.66667 11.66667 11.66667
##  [9] 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667
## [17] 11.66667 11.11111 11.11111 11.11111 11.11111 11.11111 11.11111 11.11111
## [25] 11.11111 10.55556 10.55556 10.55556 10.55556 10.55556 10.55556 10.55556
## [33] 10.55556 10.55556 10.55556 10.55556 10.55556 11.11111 11.11111 11.11111
## [41] 11.11111 11.66667 11.66667 11.66667 11.66667 13.33333 13.33333 13.33333
## [49] 13.33333 13.88889 13.88889 13.88889 13.88889 16.11111 16.11111 16.11111
## [57] 16.11111 17.22222 17.22222 17.22222 17.22222 18.33333 18.33333 18.33333
## [65] 18.33333 17.77778 17.77778 17.77778 17.77778 17.22222 17.22222 17.22222
## [73] 17.22222 15.55556 15.55556 15.55556 15.55556 15.00000 15.00000 15.00000
## [81] 15.00000 12.77778 12.77778 12.77778 12.77778 11.66667 11.66667 11.66667
## [89] 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667 11.66667
```

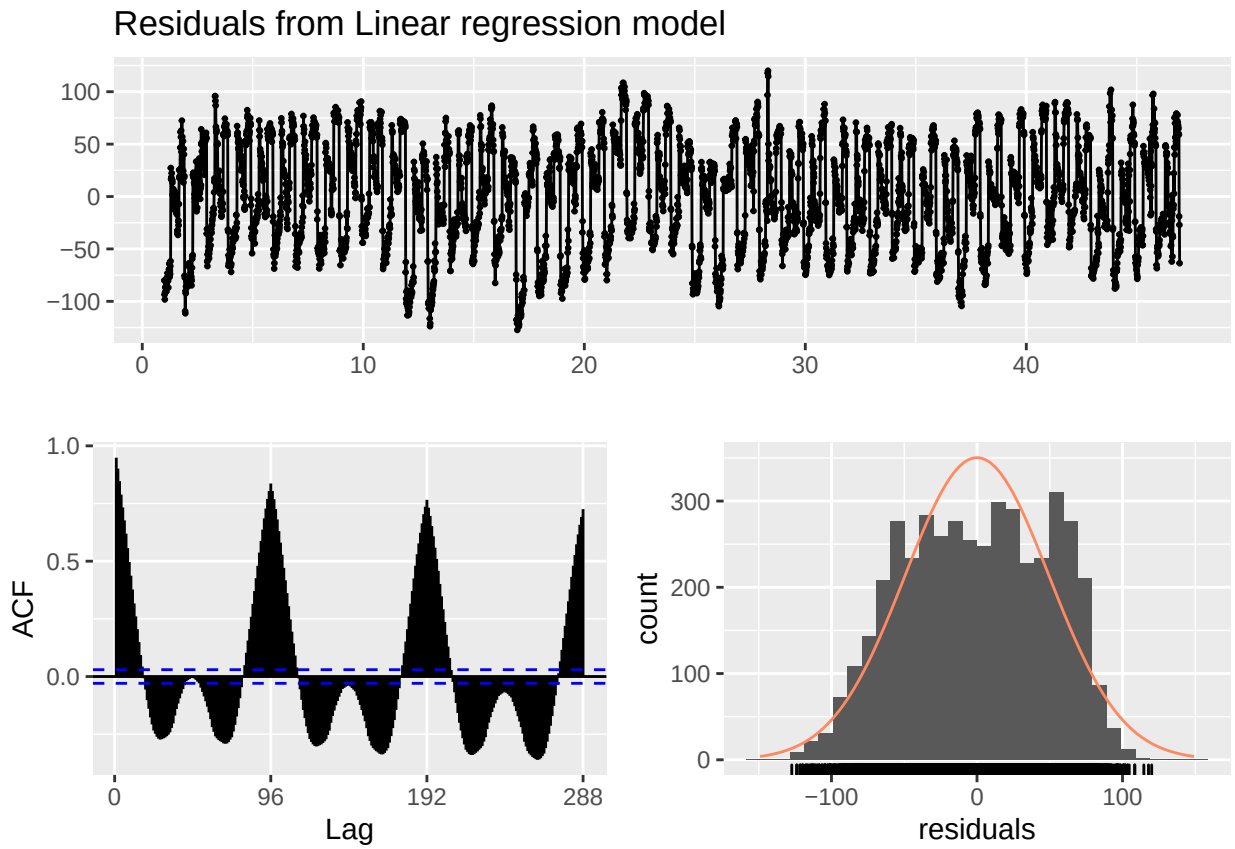
Holt-Winters doesn't take external regressors into account so we can't use it with temperature.

We use the TSLM linear regression method to build a power consumption model based on temperature. As expected there is a correlation between the 2 components (pointed by the low p-value) but temperature is definitely not enough to predict the power consumption. Even after including trend and season components in TSLM we still observe some trend and seasonal patterns in the residuals, so we can't really use TSLM to forecast.

```
tslm_temp <- tslm(power_train~temperature_train+trend)
summary(tslm_temp)
```

```
##
## Call:
## tslm(formula = power_train ~ temperature_train + trend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -127.290  -39.642    1.008   42.279  120.059
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.309e+02  3.147e+00   41.59   <2e-16 ***
## temperature_train  1.082e+01  2.824e-01   38.33   <2e-16 ***
## trend             -7.338e-03  6.013e-04  -12.20   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 49.72 on 4408 degrees of freedom
## Multiple R-squared:  0.2523, Adjusted R-squared:  0.252
## F-statistic: 743.7 on 2 and 4408 DF,  p-value: < 2.2e-16
```

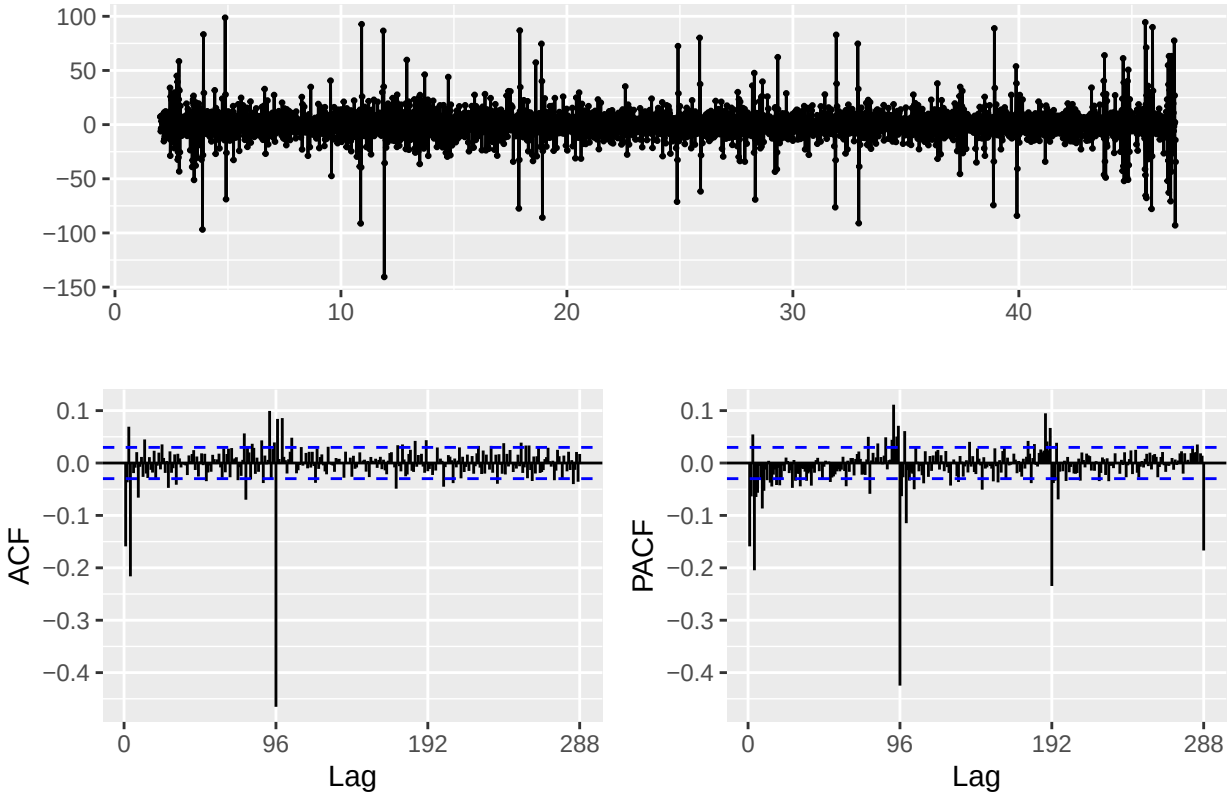
```
checkresiduals(tslm_temp)
```



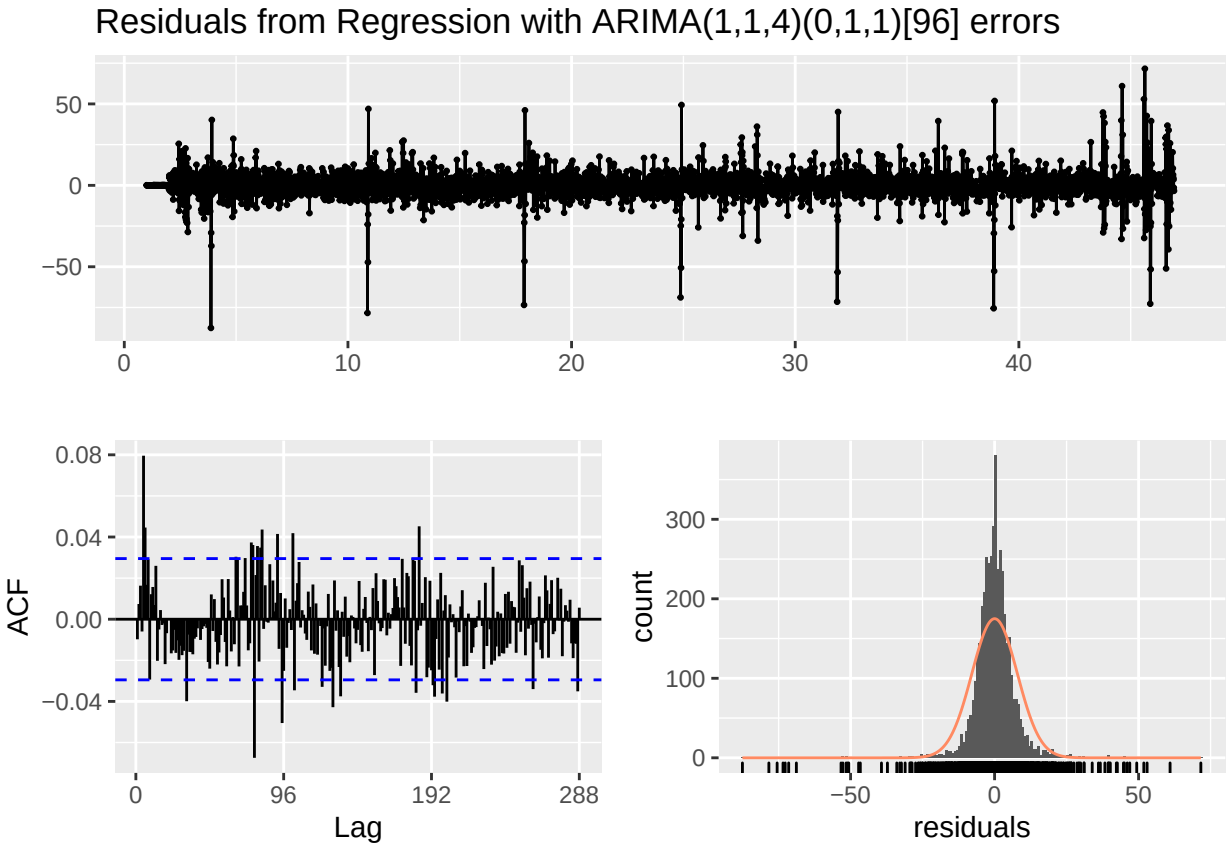

```
##
##  Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 4208.4, df = 192, p-value < 2.2e-16
```

Differentiating the TSLM residuals points at the same parameters used for ARIMA without temperature, so we use the same model as before but this time we add the xreg element then check ACF and residuals again to make sure the main components are included.

```
ggtsdisplay(diff(diff(residuals(tslm_temp), lag=96)))
```



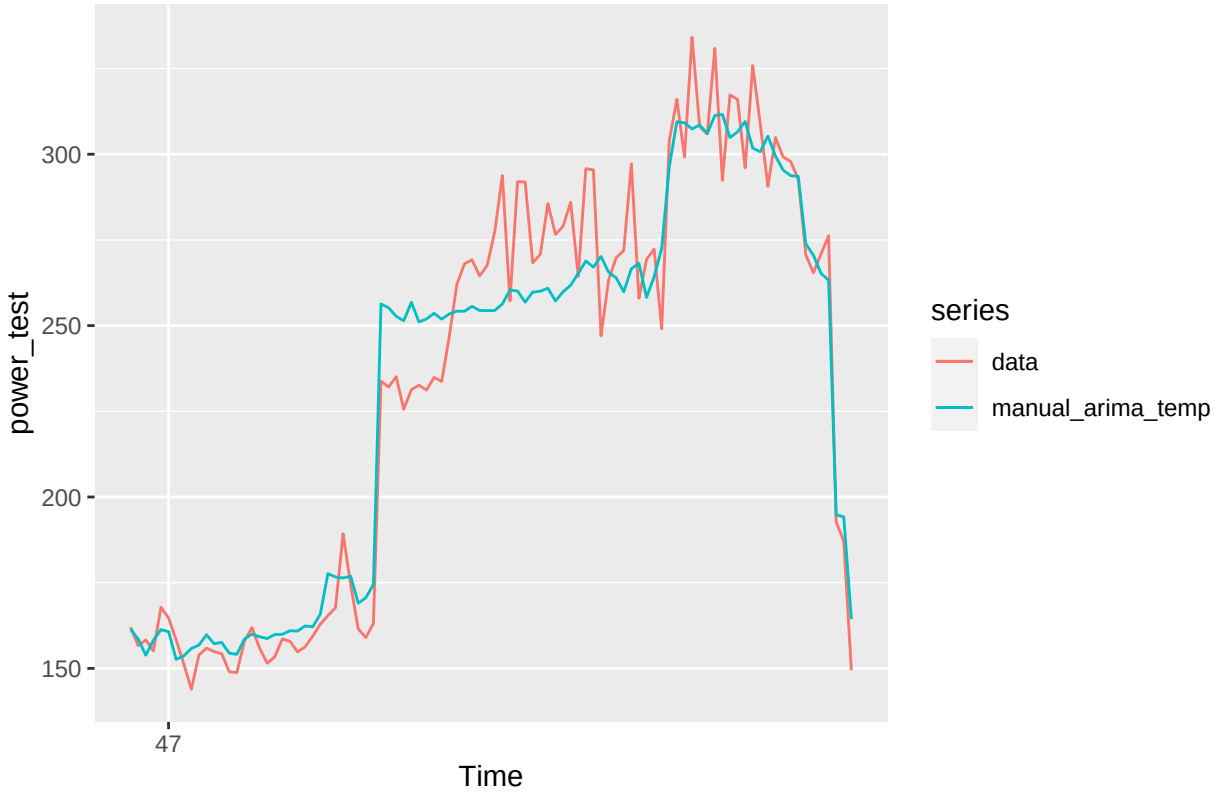

```
manual_arima_temp_fit <- Arima(power_train, order=c(1,1,4), seasonal=c(0,1,1), xreg=temperature_train)
checkresiduals(manual_arima_temp_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,4)(0,1,1)[96] errors
## Q* = 337.19, df = 185, p-value = 5.718e-11
##
## Model df: 7.   Total lags used: 192
```

This results in a very similar ARIMA model as the one without temperature, so temperature doesn't seem to help here.

```
manual_arima_temp_prev <- forecast(manual_arima_temp_fit, h=96, xreg=temperature_test)
autoplot(power_test, series='data') + autolayer(manual_arima_temp_prev$mean, series='manual_arima_temp')
```

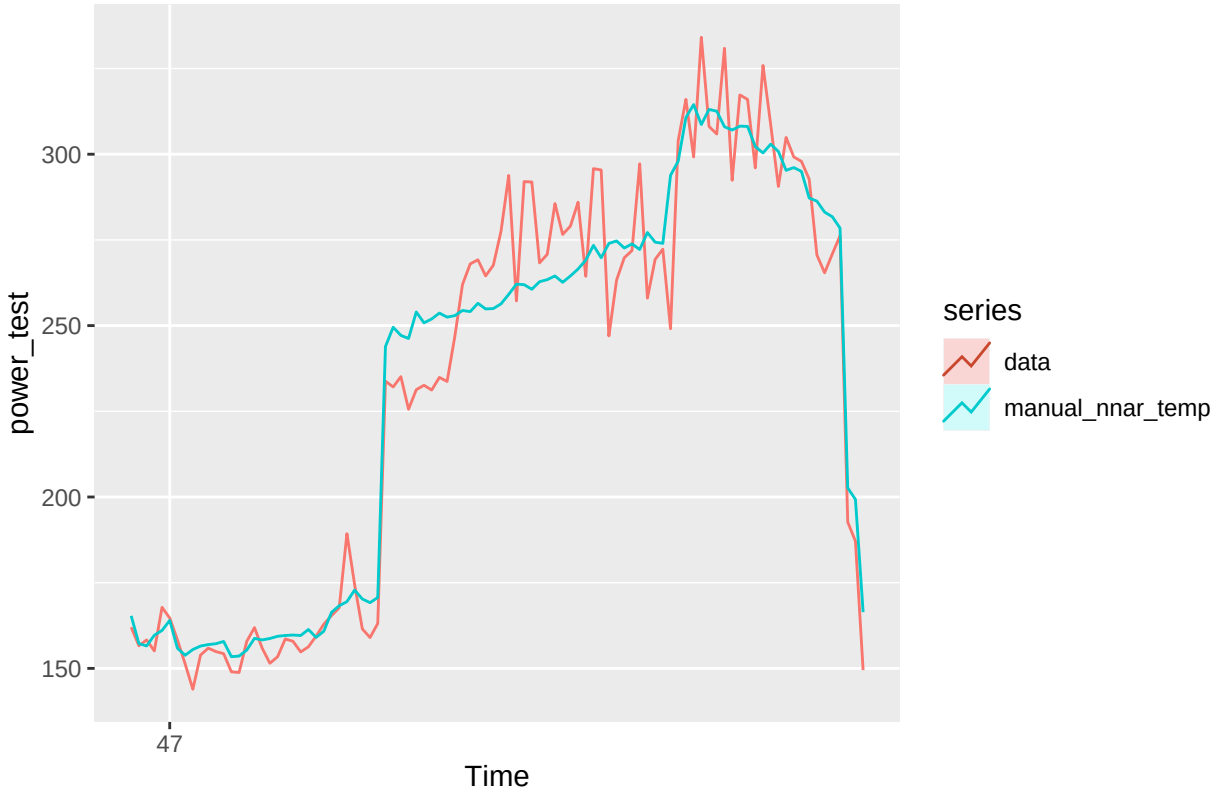



```
cat('RMSE:', sqrt(mean((manual_arima_temp_prev$mean-power_test)^2)))
```

```
## RMSE: 14.4549
```

We do the same for the neural network model: we use the same NNAR parameters as before and add the temperature external regressor. This results in a slightly improved model, although RMSE are still very close.

```
manual_nnar_temp_fit <- nnetar(power_train, 40, 4, 20, lambda='auto', xreg=temperature_train)
manual_nnar_temp_prev <- forecast(manual_nnar_temp_fit, h=96, xreg=temperature_test)
autoplot(power_test, series='data') + autolayer(manual_nnar_temp_prev, series='manual_nnar_temp')
```

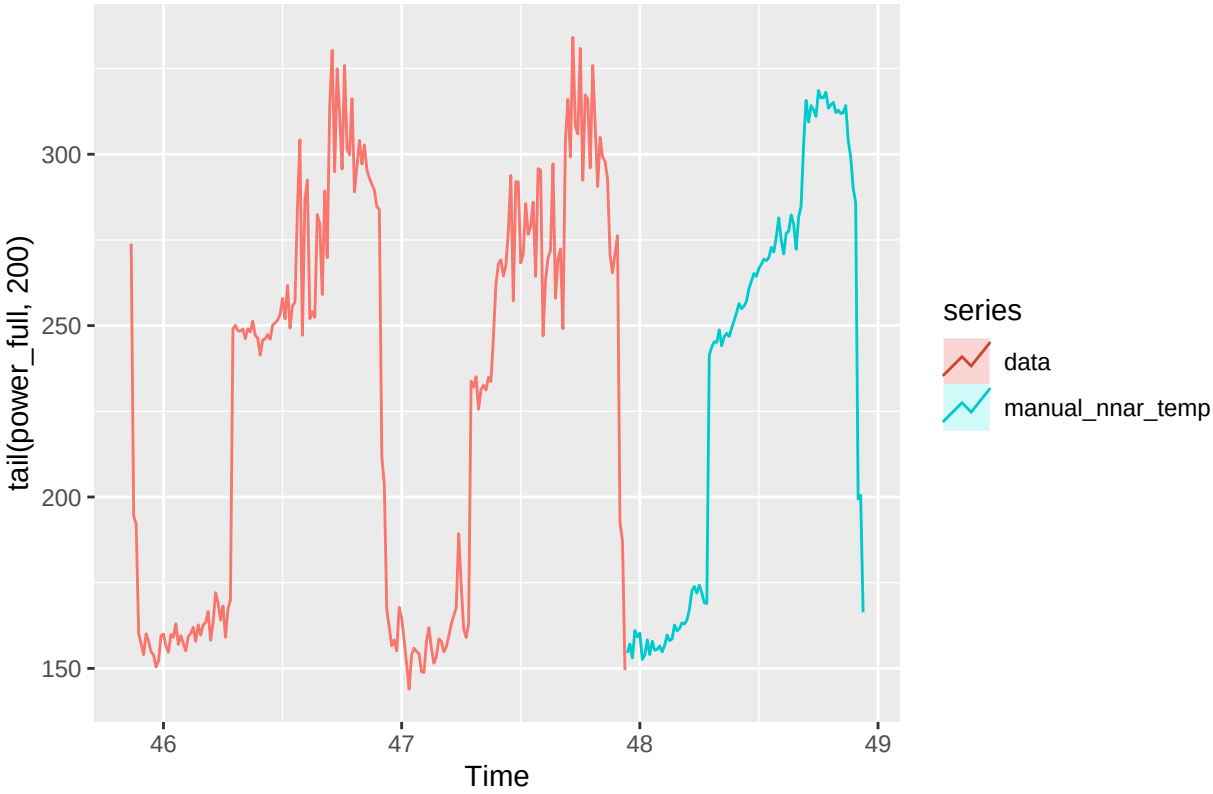



```
cat('RMSE:', sqrt(mean((manual_nnar_temp_prev$mean-power_test)^2)))
```

```
## RMSE: 13.98855
```

Finally, in order to forecast power with temperature, we select NNAR over ARIMA because NNAR is faster and seems to use temperature a little. This may have not been our choice if we had to forecast on a longer time range as our NNAR model may be prone to overfitting due to so many parameters.

```
temperature_full <- window(temperature, start=c(1,1), end=c(47,91))
temperature_try <- window(temperature, start=c(47,92), end=c(48,91))
manual_nnar_temp_full <- nnetar(power_full, 40, 4, 20, lambda='auto', xreg=temperature_full)
F02172010_with_temperature <- forecast(manual_nnar_temp_full, h=96, xreg=temperature_try)
autoplot(tail(power_full, 200), series='data') + autolayer(F02172010_with_temperature, series='manual_nn
```




```
print(F02172010_with_temperature$mean)
```

```
## Time Series:
## Start = c(47, 92)
## End = c(48, 91)
## Frequency = 96
##  [1] 154.4971 157.1197 153.0380 161.0699 159.1799 160.3215 152.6233 153.9028
##  [9] 158.3187 154.0165 157.9004 155.3040 155.5953 156.5335 154.8199 156.7271
## [17] 159.7452 158.0876 158.5763 162.6721 160.9569 161.6456 163.3059 162.9362
## [25] 164.1825 167.3247 172.6875 173.8804 171.9679 174.2792 171.9745 169.1143
## [33] 168.9216 241.4265 243.8246 245.2951 245.1272 248.7197 244.1157 246.7158
## [41] 247.7493 246.8249 249.1979 251.4377 253.8248 256.4277 254.9371 255.7302
## [49] 257.1446 260.9133 263.0268 265.2034 264.3879 266.7402 267.9019 269.4002
## [57] 268.9795 269.8420 272.8221 271.4628 275.9410 281.4471 274.9043 270.9132
## [65] 276.9745 277.5067 282.3049 279.6528 272.2538 281.7015 284.8468 302.6458
## [73] 315.7182 309.3868 314.1754 313.0104 311.0412 318.5758 316.4342 316.4547
## [81] 318.0717 313.4048 314.3874 315.1492 312.1495 312.8478 311.8745 312.1343
## [89] 314.2061 303.9147 299.2629 290.0143 285.8528 199.3650 200.5361 166.3548
```

Let's write our final results in a file.

```
write.xlsx(cbind(F02172010_without_temperature$mean,F02172010_with_temperature$mean),'prediction.xlsx')
```