Time Series Exam

Electricity consumption forcasting

Moussa Griche, DS-S22 Student

2023-01-12

Contents

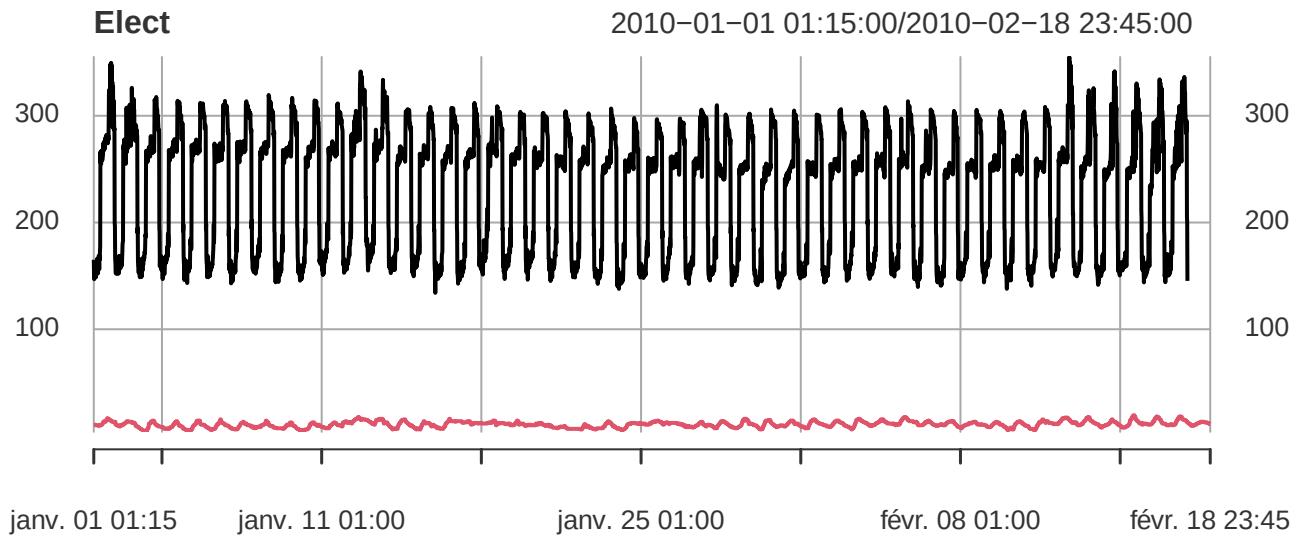
I- Analysing the Time serie

I-1- Uploading Data

```
Elect_data <- read_excel("./Elec-train.xlsx")
Elect_data$Timestamp = as.POSIXct(Elect_data$Timestamp, format="%m/%d/%Y %H:%M")
Elect <- xts(Elect_data[,c(2,3)], Elect_data$Timestamp)
```

I-2- Ploting the time series

```
plot(Elect)
```



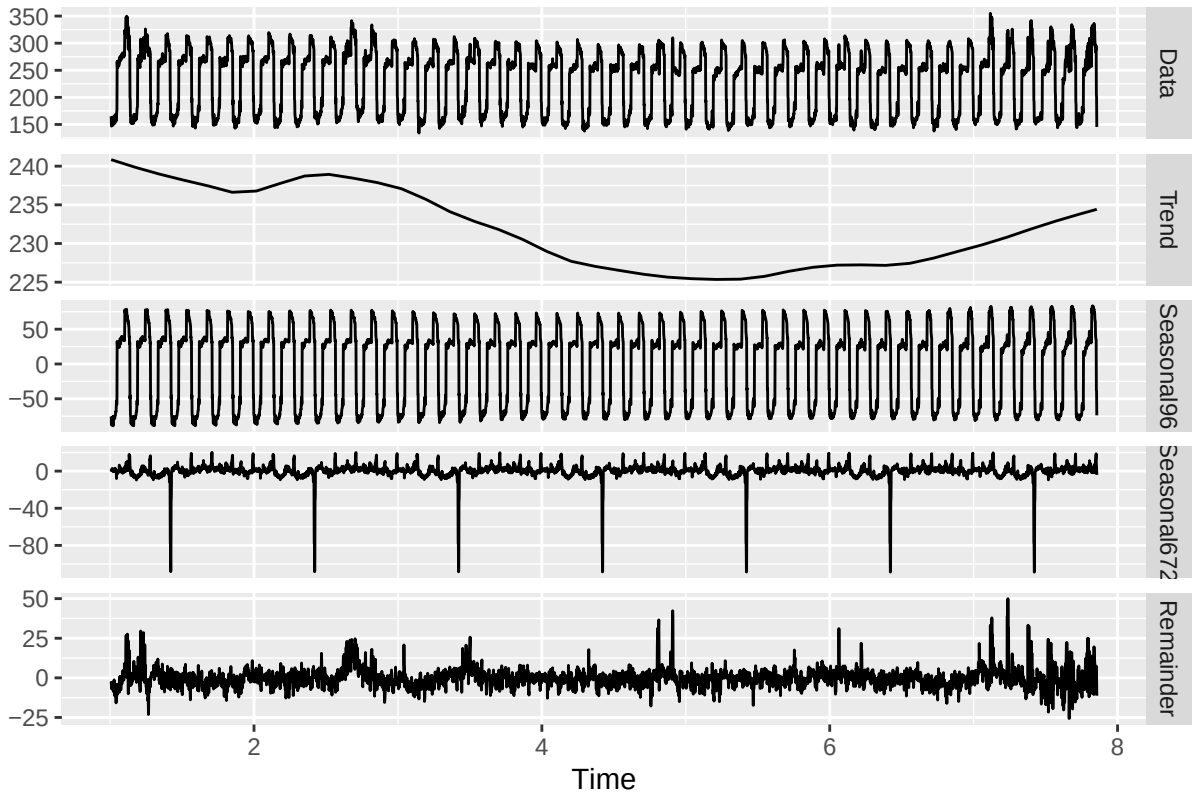
I-3- Analysing the trend and seasonality:

We could see that is possible to have a daily and weekly seasonality. Let's try to test the seasonality of this time series.

To analyse the double seasonality of the electricity consumption (kW), we make the frequency of the time series equal to 96 and 96*7. Like this, the time series became a daily time series.

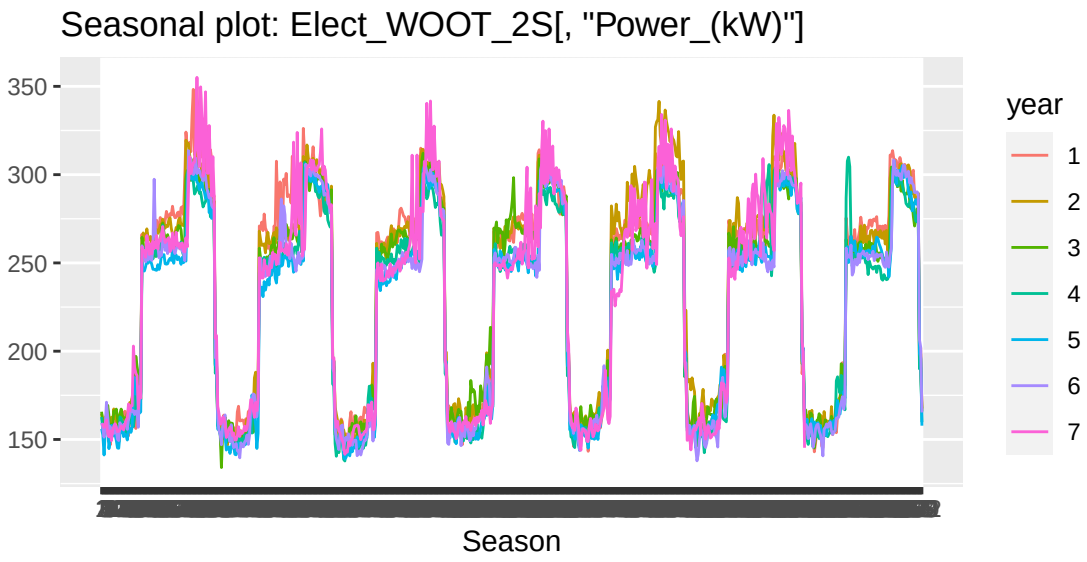
```
Elect_WOOT <- xts(Elect_data[,c(2,3)], Elect_data$Timestamp)
Elect_WOOT <- window(Elect_WOOT, start = "2010-01-01 01:15:00", end = "2010-02-17 23:45:00")
Elect_WOOT_2S <- msts(Elect_WOOT, seasonal.periods = c(96,672), start=c(1,6))
```

```
Elect_WOOT_D_decompose <- mstl(Elect_WOOT_2S[,"Power_(kW)"])
autoplot(Elect_WOOT_D_decompose)
```



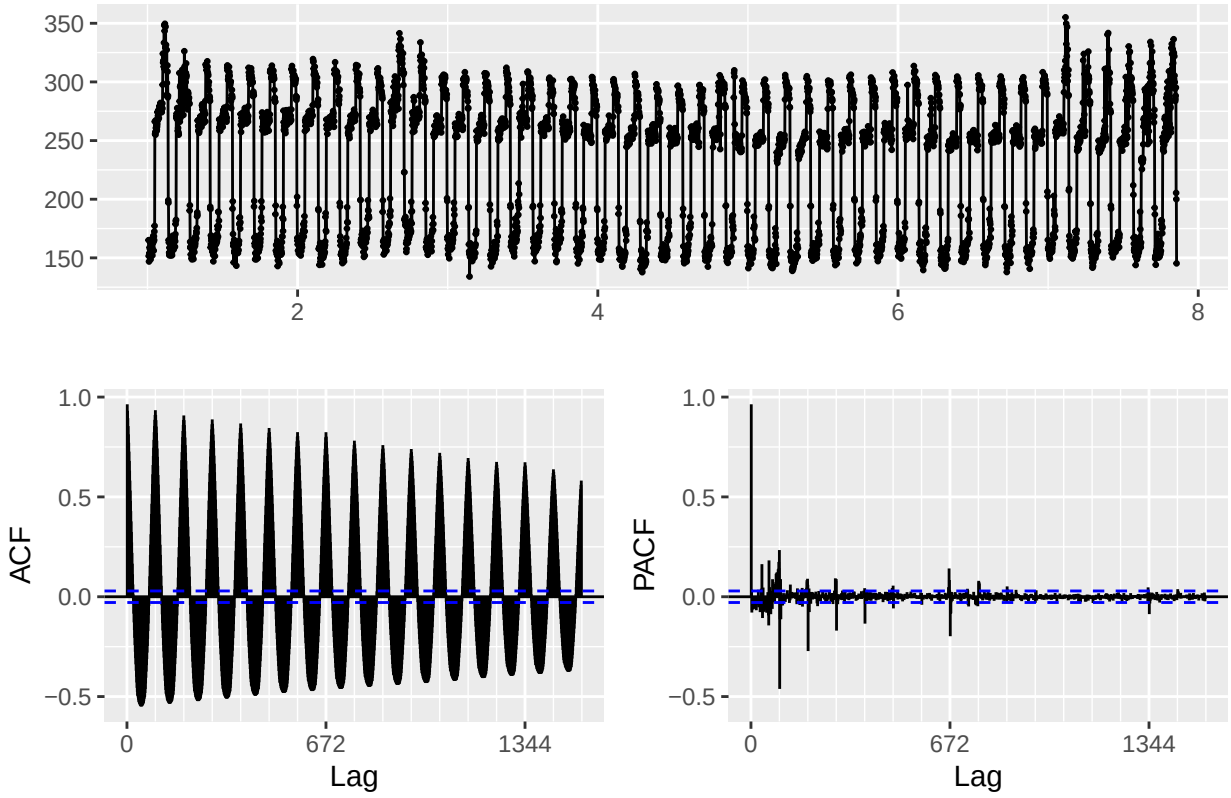
We can see from the decompose plot that the electricity consumption (kW) time series has a trend and the double seasonality is present.

```
ggseasonplot(Elect_WOOT_2S[,"Power_(kW)"])
```



The ggseasonal plot confirm that this time series hase a daily and weekly seasonality.

```
ggtsdisplay(Elect_WOOT_2S[,"Power_(kW)"])
```



The ACF plot confirm what we have already seen (the electricity consumption (kW) time series has a seasonality)

I-4- Spelliting the electricity consumption (kW) time serie into train and test set

As we have seen that the electricity consumption (kW) time series has a daily and weekly seasonality, we take a test set equal to a week observations. Like this, we test over all days week and capt all the seasonality.

Since we should forecast one day electricity consumption, we start with fitting models with only daily seasonality and then we see if there is some differences with models fitted with double seasonality time series.

```
#Splitting data to training dataset and testing dataset
Elect_WOOT_D <- ts(Elect_WOOT, frequency = 96, start=c(1,6))

Elect_D.ts.train <- window(Elect_WOOT_D, start=c(1,6), end=c(41,96))

Elect_D.ts.test <- window(Elect_WOOT_D, start=c(42,1), end=c(48,96))
```

II- Simple seasonality forecast:

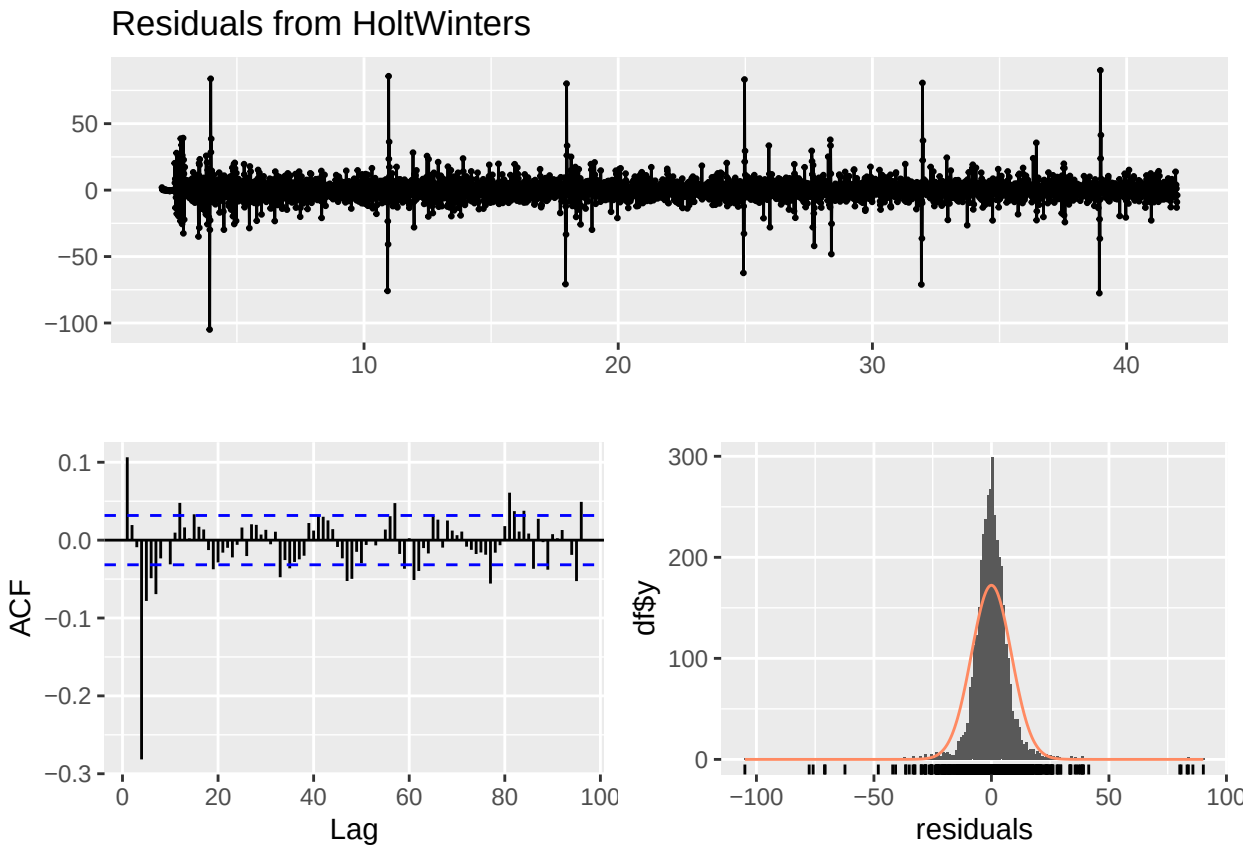
II-1- Fitting Electricity Consumption (kW) models without using outdoor temperature:

II-1-1- Forecasting with exponential smoothing

II-1-1-1 Holt Winters with Additive Seasonality

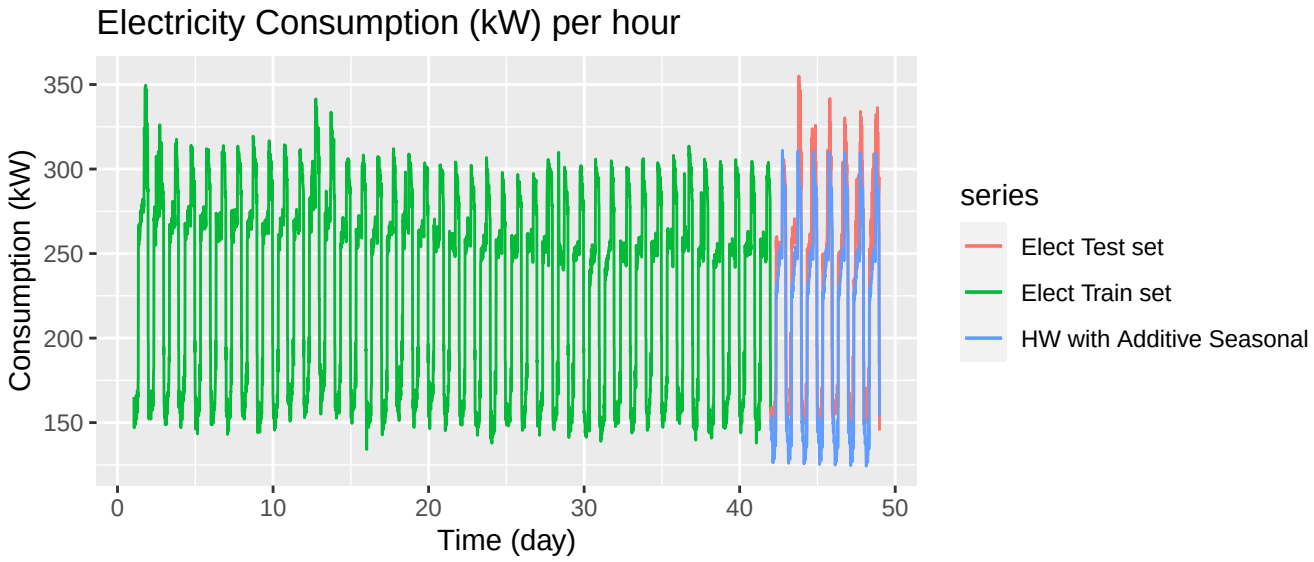
```
Elect_WOOT_D_hw <- HoltWinters(Elect_D.ts.train[,"Power_(kW)"], alpha=NULL, beta = NULL, gamma = NULL)
checkresiduals(Elect_WOOT_D_hw, lag.max=96)
```

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




```
Elect_WOOT_D_hw_forecast <- forecast :: forecast(Elect_WOOT_D_hw, h=672)
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_hw_forecast$mean, series='HW with Additive Seasonal')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_hw_RMSE <- sqrt(mean((Elect_WOOT_D_hw_forecast$mean - Elect_D.ts.test[,"Power_(kW)"])**2))

cat("The RMSE of HW model with Additive Seasonal equale to:", Elect_WOOT_hw_RMSE, "\n")
```

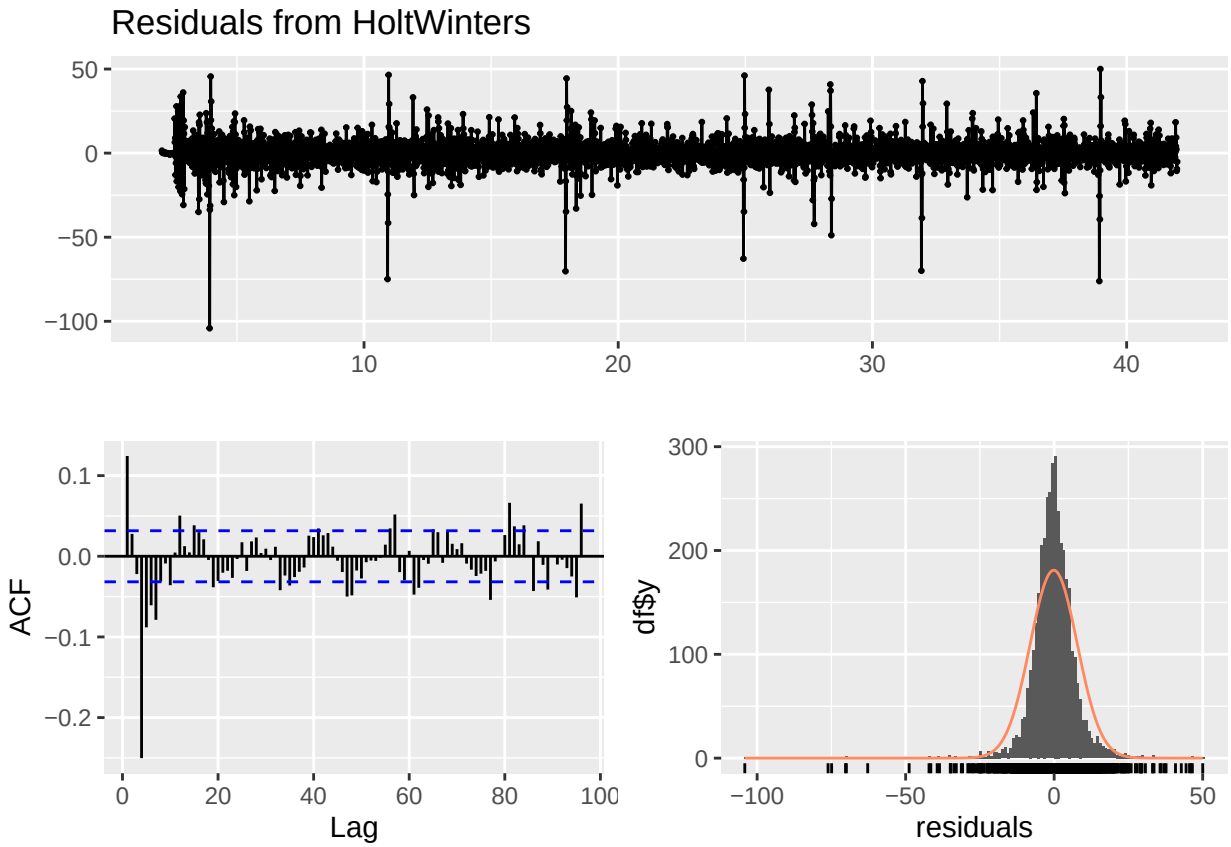
```
## The RMSE of HW model with Additive Seasonal equale to: 23.57358
```

We see from the Holt Winters model with Additive seasonality, that the residuals are not white noise (residuals are not independents).

II-1-1-2- Holt Winters with Multiplicative Seasonality

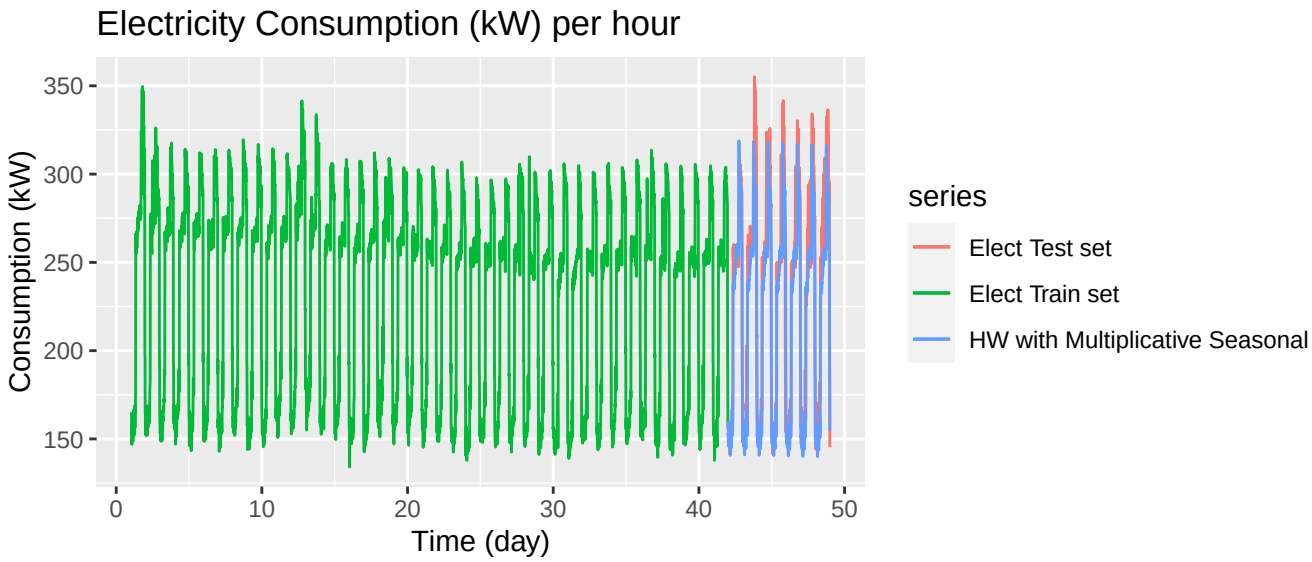
```
Elect_WOOT_D_hw_MS <- HoltWinters(Elect_D.ts.train[,"Power_(kW)"], seasonal = "multiplicative", alpha=NULL, beta
checkresiduals(Elect_WOOT_D_hw_MS, lag.max=96)
```

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




```
Elect_WOOT_D_hw_MS_forecast <- forecast :: forecast(Elect_WOOT_D_hw_MS, h=672)
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_hw_MS_forecast$mean, series='HW with Multiplicative Seasonal')+
  ggtitle ('Electricity Consumption (kW) per hour') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_hw_MS_RMSE <- sqrt(mean((Elect_WOOT_D_hw_MS_forecast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))

cat("The RMSE of HW model with Multiplicative Seasonal equale to:", Elect_WOOT_D_hw_MS_RMSE, "\n")
```

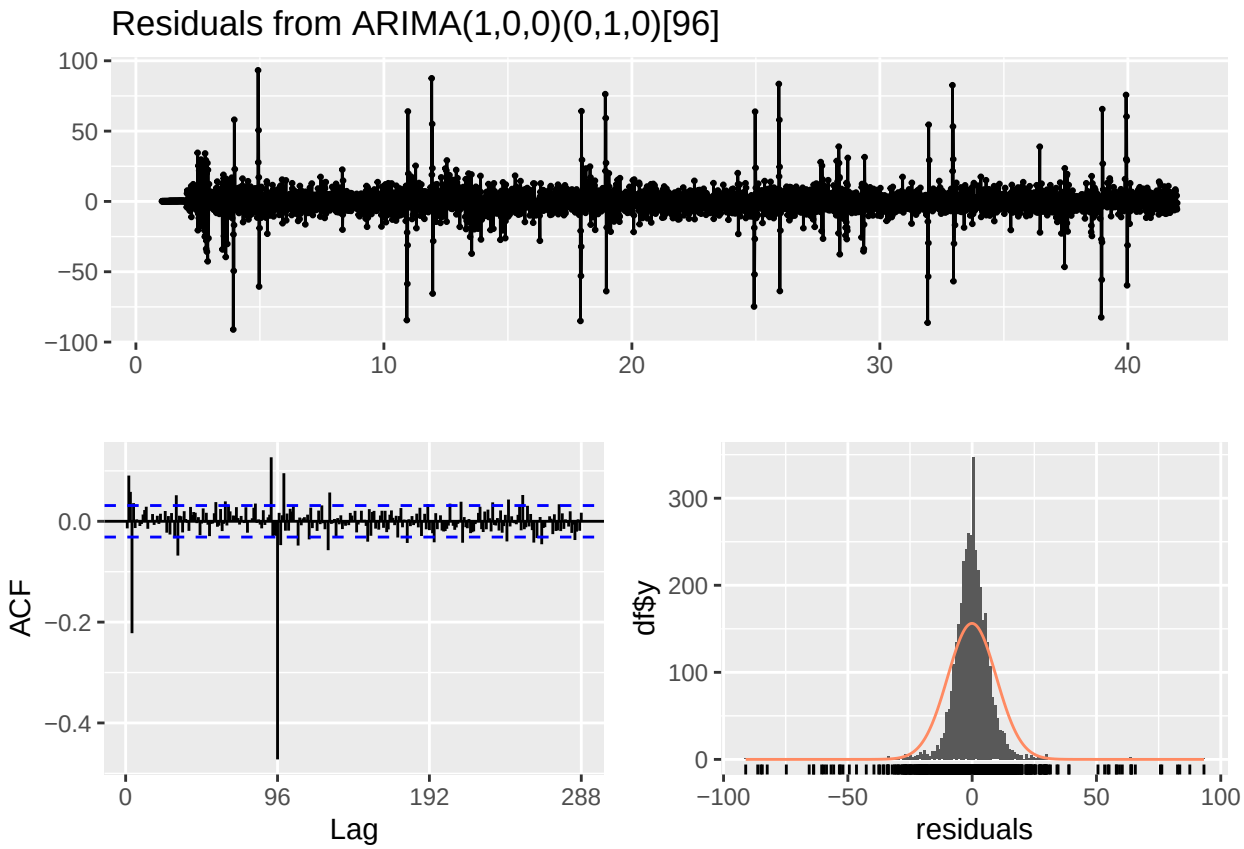
```
## The RMSE of HW model with Multiplicative Seasonal equale to: 17.95675
```

With the Multiplicative Seasonal in Holt Winters model, the residuals still not white noise.

II-1-2- Forecasting Auto SARIMA Model:

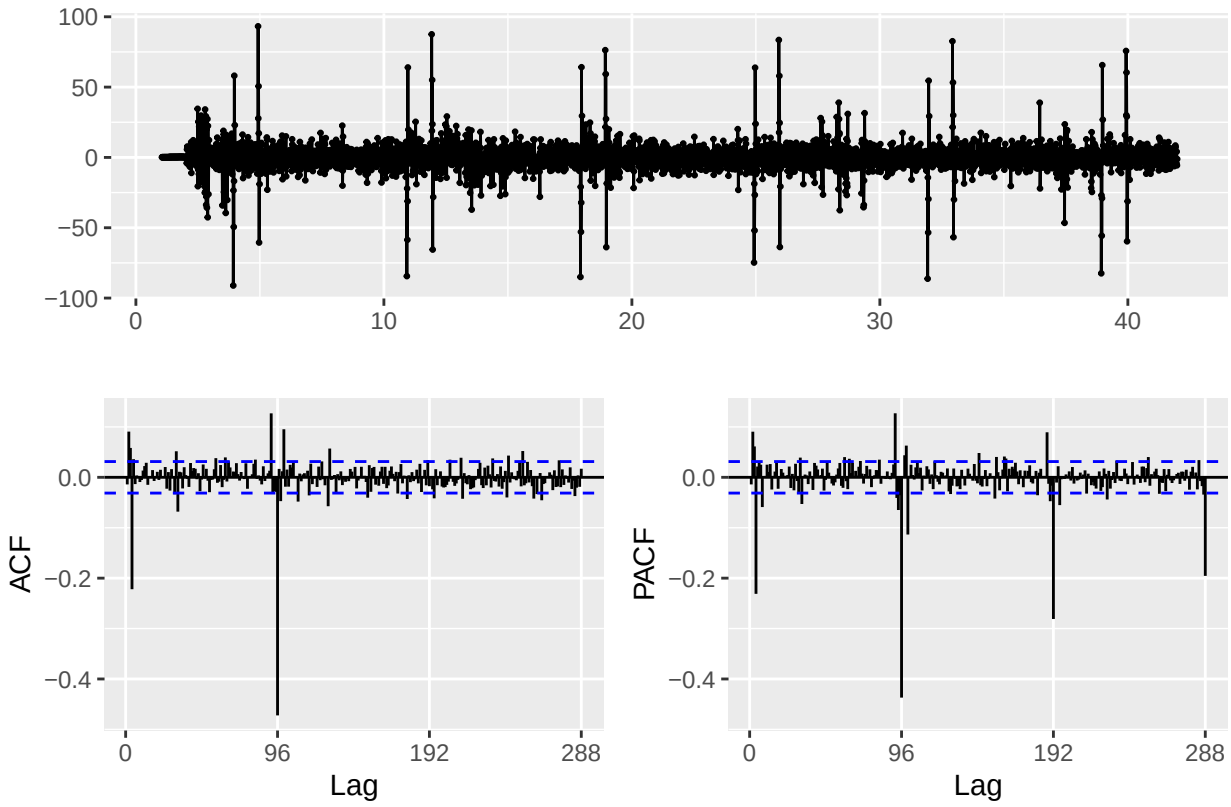
II-1-2-1- Forecasting Auto SARIMA Model:

```
Elect_WOOT_D_auto.arima <- auto.arima(Elect_D.ts.train[,"Power_(kW)"])
checkresiduals(Elect_WOOT_D_auto.arima)
```

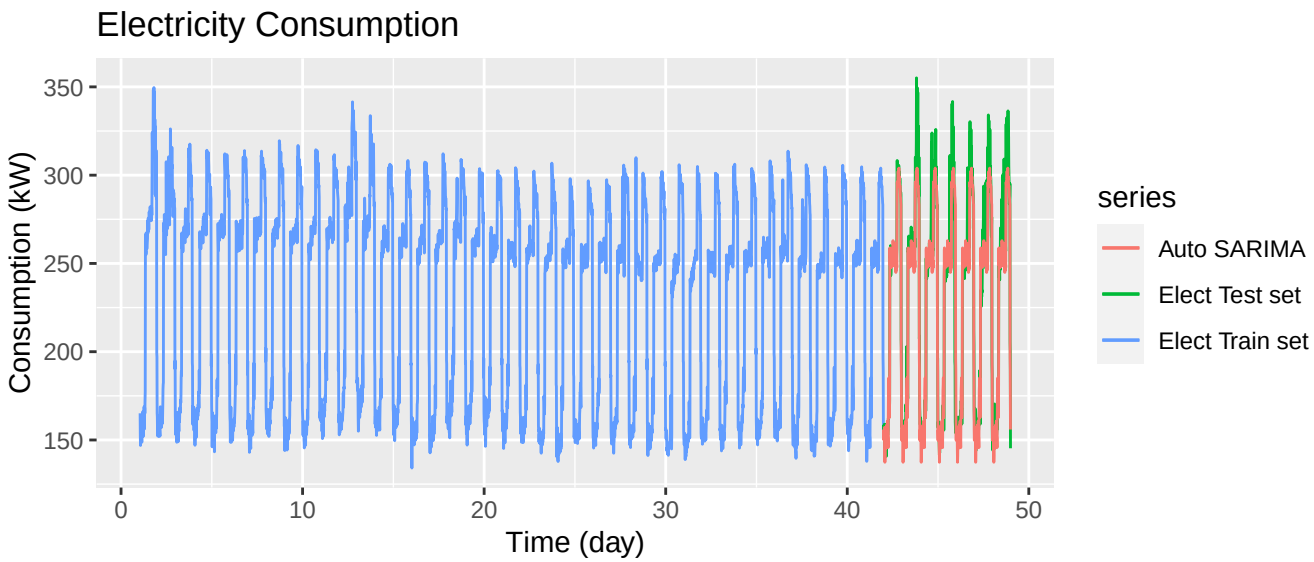



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 1531.9, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

```
ggtsdisplay(Elect_WOOT_D_auto.arima$residuals)
```




```
Elect_WOOT_D_auto.arima_forcast <- forecast::forecast(Elect_WOOT_D_auto.arima, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_auto.arima_forcast$mean, series='Auto SARIMA') +
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_auto.arima_RMSE <- sqrt(mean((Elect_WOOT_D_auto.arima_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^
Elect_WOOT_D_auto.arima_AIC <- Elect_WOOT_D_auto.arima$aicc

cat("The RMSE of Auto_Arima model equal to:", Elect_WOOT_D_auto.arima_RMSE,"\n")
```

```
## The RMSE of Auto_Arima model equal to: 17.22069
```

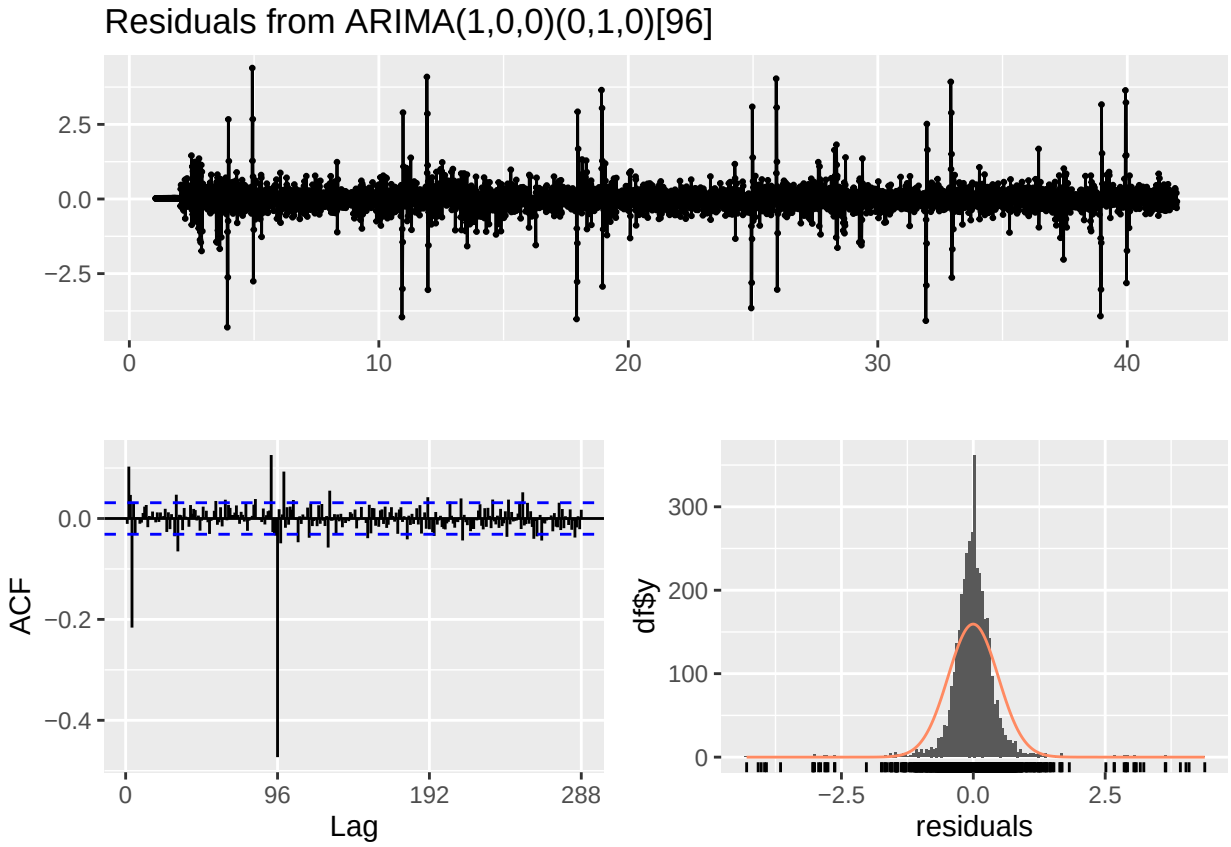
```
cat("The AIC of Auto_Arima model equal to:", Elect_WOOT_D_auto.arima_AIC,"\n")
```

```
## The AIC of Auto_Arima model equal to: 28481.47
```

The Auto SARIMA Model is no so good because we can see from the ACF and PACF that the residuals are correlated and Ljung-Box test gives a p-value ««0.05. So, the residuals for the Auto SARIMA model are not white noise.

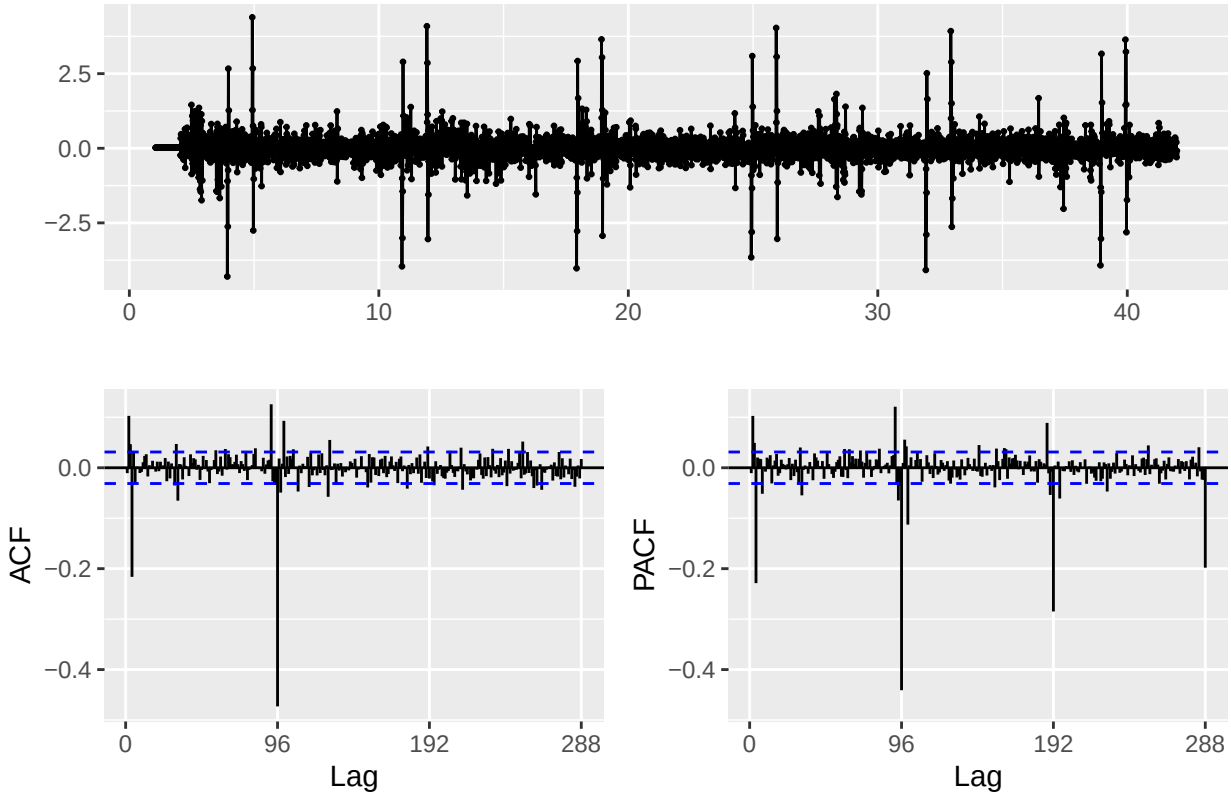
II-1-2-2- Forecasting Auto SARIMA Model with Box-Cox transformation :

```
Elect_WOOT_D_auto.arima_BC <- auto.arima(Elect_D.ts.train[,"Power_(kW)"], lambda = 'auto')
checkresiduals(Elect_WOOT_D_auto.arima_BC)
```

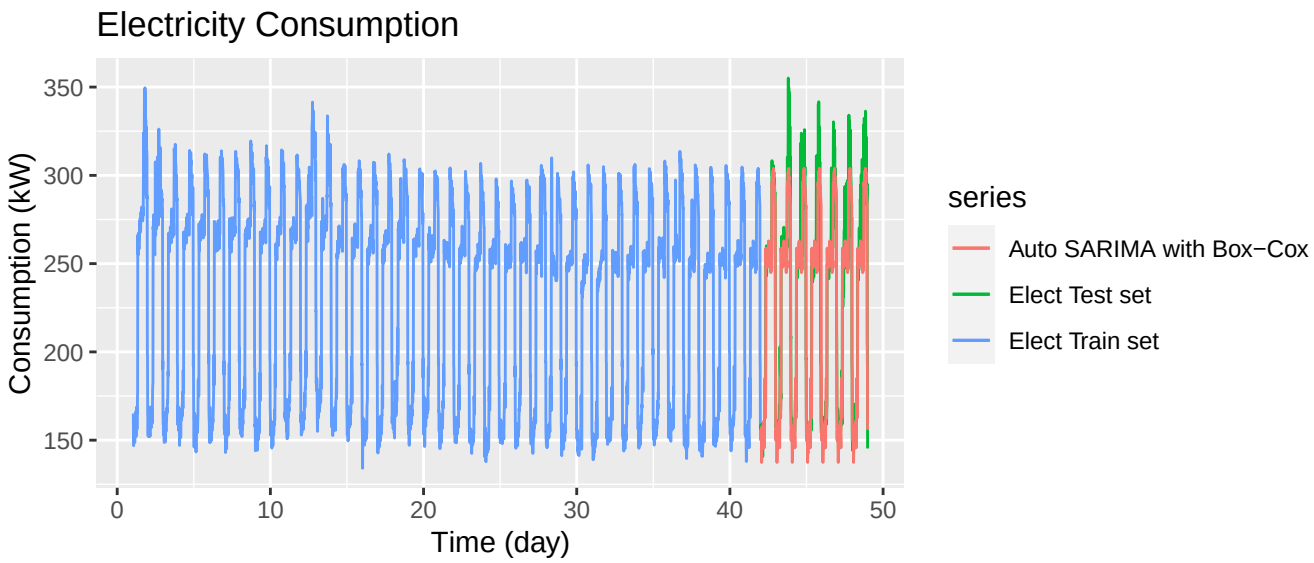



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 1528, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

```
ggtsdisplay(Elect_WOOT_D_auto.arima_BC$residuals)
```




```
Elect_WOOT_D_auto.arima_BC_forcast <- forecast::forecast(Elect_WOOT_D_auto.arima_BC, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_auto.arima_BC_forcast$mean, series='Auto SARIMA with Box-Cox')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_auto.arima_BC_RMSE <- sqrt(mean((Elect_WOOT_D_auto.arima_BC_forcast$mean-Elect_D.ts.test[,"Power_(k
Elect_WOOT_D_auto.arima_BC_AIC <- Elect_WOOT_D_auto.arima$aicc

cat("The RMSE of Auto SARIMA model with Box-Cox Transformation equal to:", Elect_WOOT_D_auto.arima_BC_RMSE,"\n")
```

```
## The RMSE of Auto SARIMA model with Box-Cox Transformation equal to: 17.22011
```

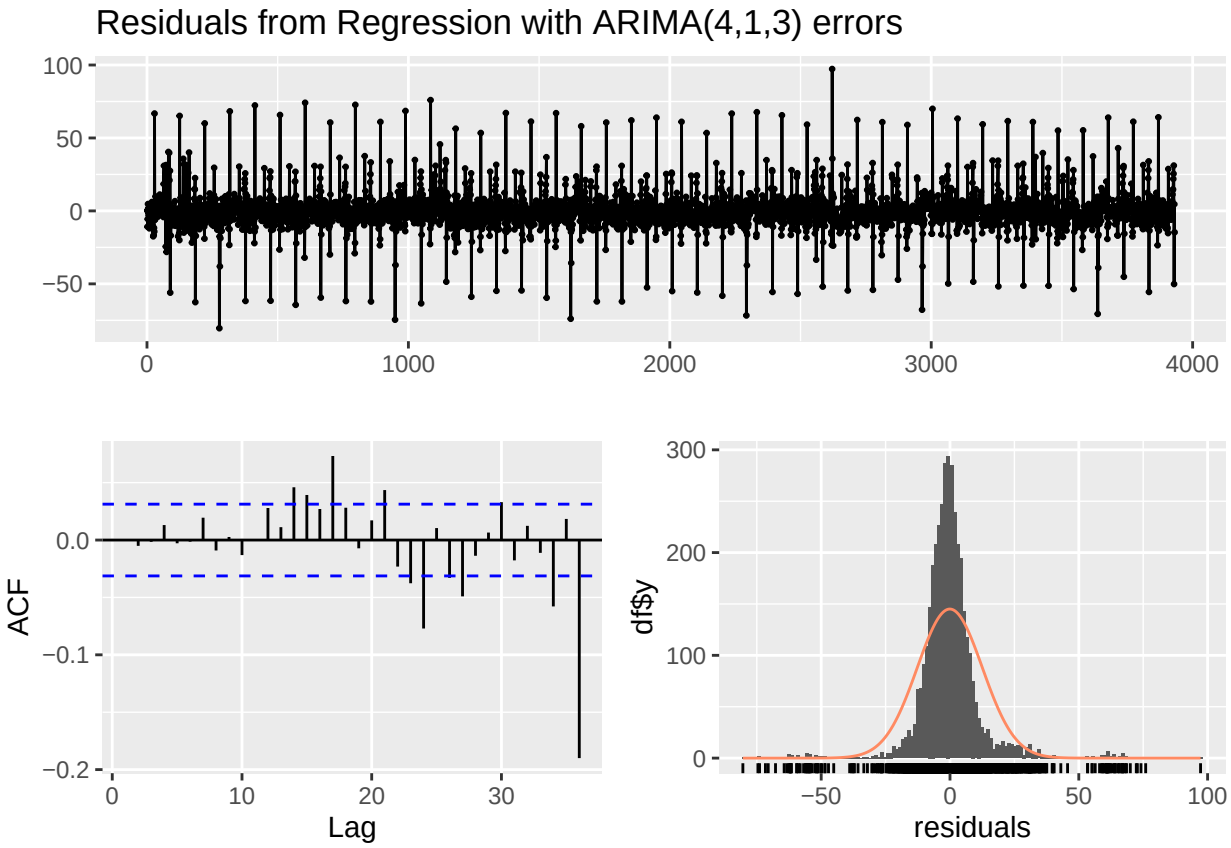
```
cat("The AIC of Auto SARIMA model with Box-Cox Transformation equal to:", Elect_WOOT_D_auto.arima_BC_AIC,"\n")
```

```
## The AIC of Auto SARIMA model with Box-Cox Transformation equal to: 28481.47
```

Here, the idea is to add a Box Cox transformation to the Auto SARIMA Model to get better model. But the results show that we still have the same problem of the residuals.

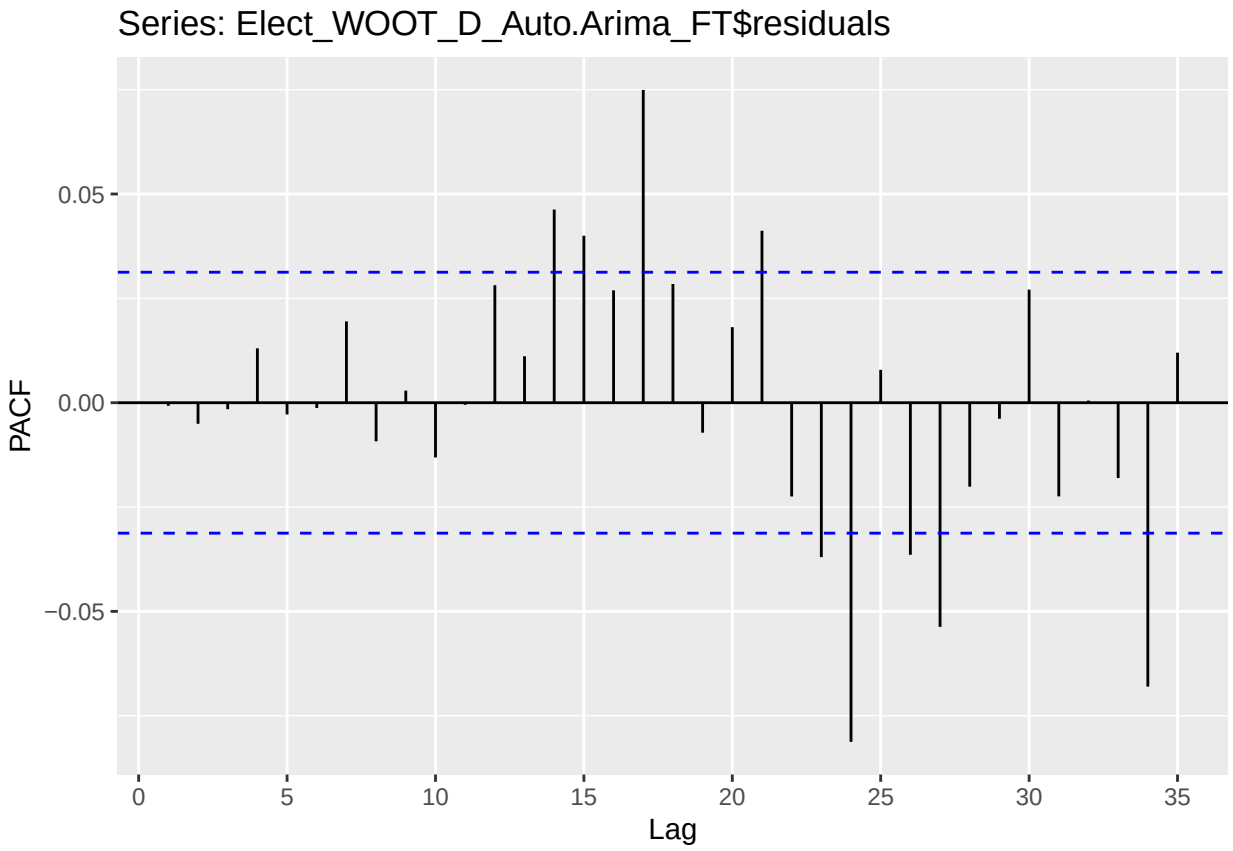
II-1-2-3- Forecasting Auto SARIMA Model with Fourrier transformation :

```
Elect.train_TF <- window(Elect, start = "2010-01-01 01:15:00", end = "2010-02-10 23:45:00")
Elect.test_TF <- window(Elect, start = "2010-02-11 01:15:00", end = "2010-02-17 23:45:00")

Elect_WOOT_D_Auto.Arima_FT <- auto.arima(ts(Elect.train_TF[,"Power_(kW)"]), xreg= fourier(ts(Elect.train_TF[,"Po

checkresiduals(Elect_WOOT_D_Auto.Arima_FT)
```

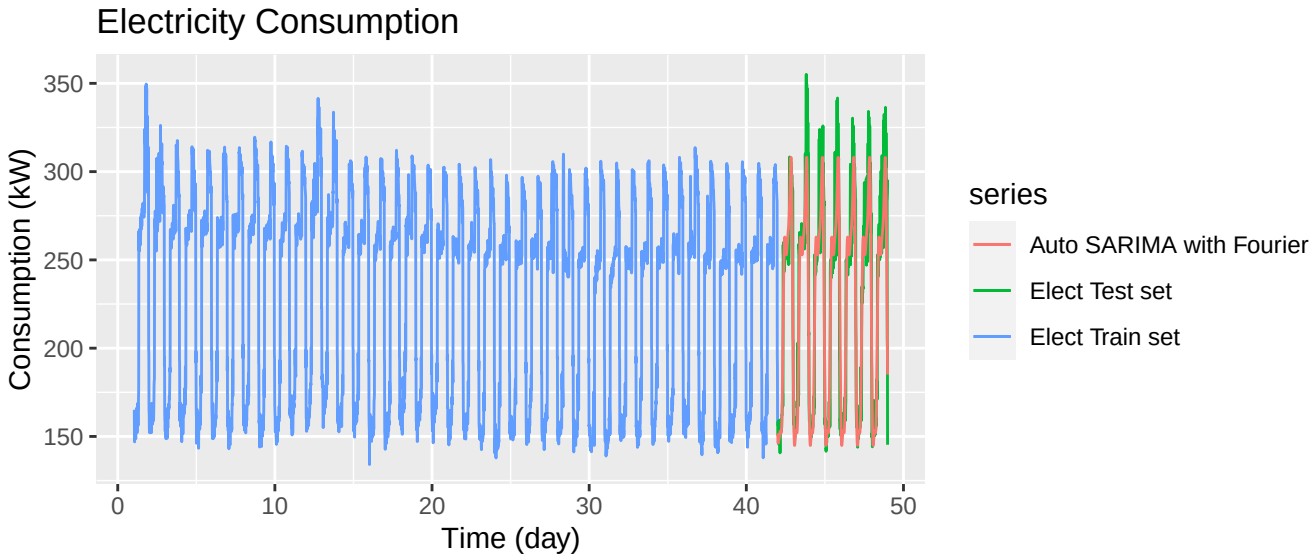



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(4,1,3) errors
## Q* = 3.3462, df = 3, p-value = 0.3413
##
## Model df: 7.   Total lags used: 10
```

```
ggPacf(Elect_WOOT_D_Auto.Arima_FT$residuals)
```




```
Elect_WOOT_D_auto.arima_FT_forcast <- forecast::forecast(Elect_WOOT_D_Auto.Arima_FT, xreg=
                                fourier(ts(Elect.test_TF[,"Power_(kW)"], frequency= 96), K=4, h=672))
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(ts(Elect_WOOT_D_auto.arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series='Auto SARIMA w
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_auto.arima_FT_RMSE <-sqrt(mean(as.numeric(ts(Elect_WOOT_D_auto.arima_FT_forcast$mean, frequency = 9
Elect_WOOT_D_auto.arima_FT_AIC = Elect_WOOT_D_Auto.Arima_FT$aicc

cat("The RMSE of Auto SARIMA model with Fourrier Transformation equale to:", Elect_WOOT_D_auto.arima_FT_RMSE, "\
```

```
## The RMSE of Auto SARIMA model with Fourrier Transformation equale to: 19.51704
```

```
cat("The AIC of Auto SARIMAl with Fourrier Transformation equale to:", Elect_WOOT_D_auto.arima_FT_AIC, "\n")
```

```
## The AIC of Auto SARIMAl with Fourrier Transformation equale to: 31125.75
```

When we add a Fourier Transformation to Auto SARIMA Model, the residuals seems become good (Ljung-Box test =0.34 > 0.05), but we see that we have a problem of over fitting because the RMSE on the Test Set get bigger.

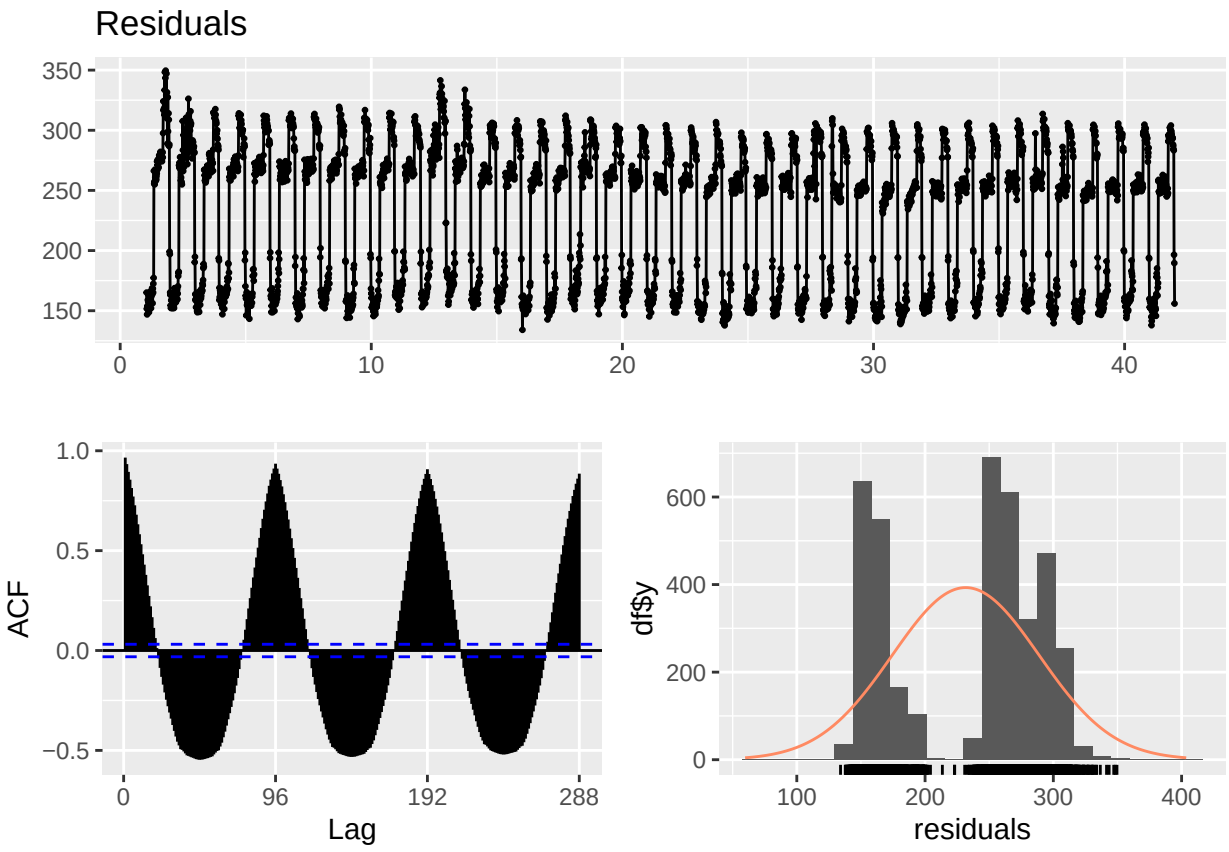
II-1-3- Forecasting Manuel SARIMA Model:

II-1-3-1- Forecasting Manuel SARIMA Model:

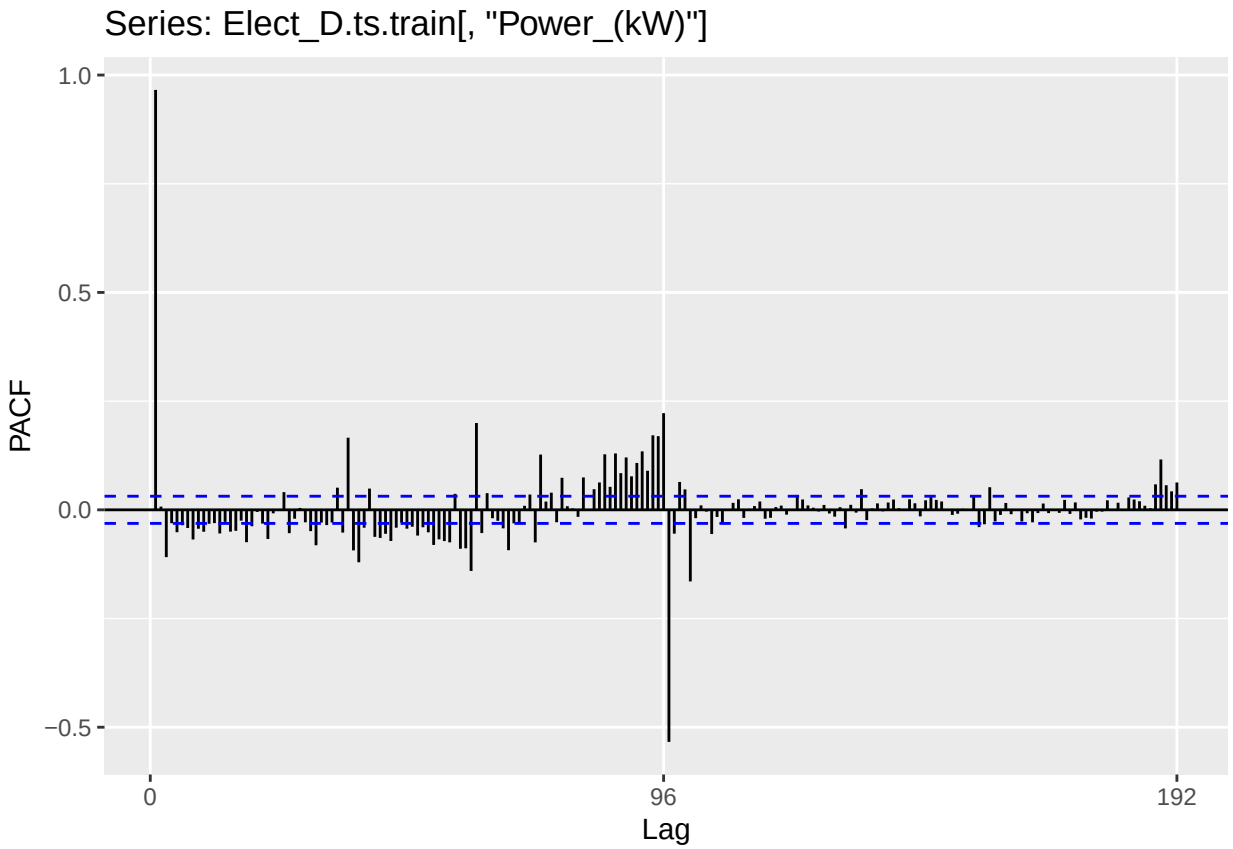
The first step here is to solve the problem of seasonality. The electricity consumption (kW) time series has a daily seasonality (96). We try to do a diff with lag=96 to this time series to remove the daily seasonality.

```
checkresiduals(Elect_D.ts.train[,"Power_(kW)"])
```

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



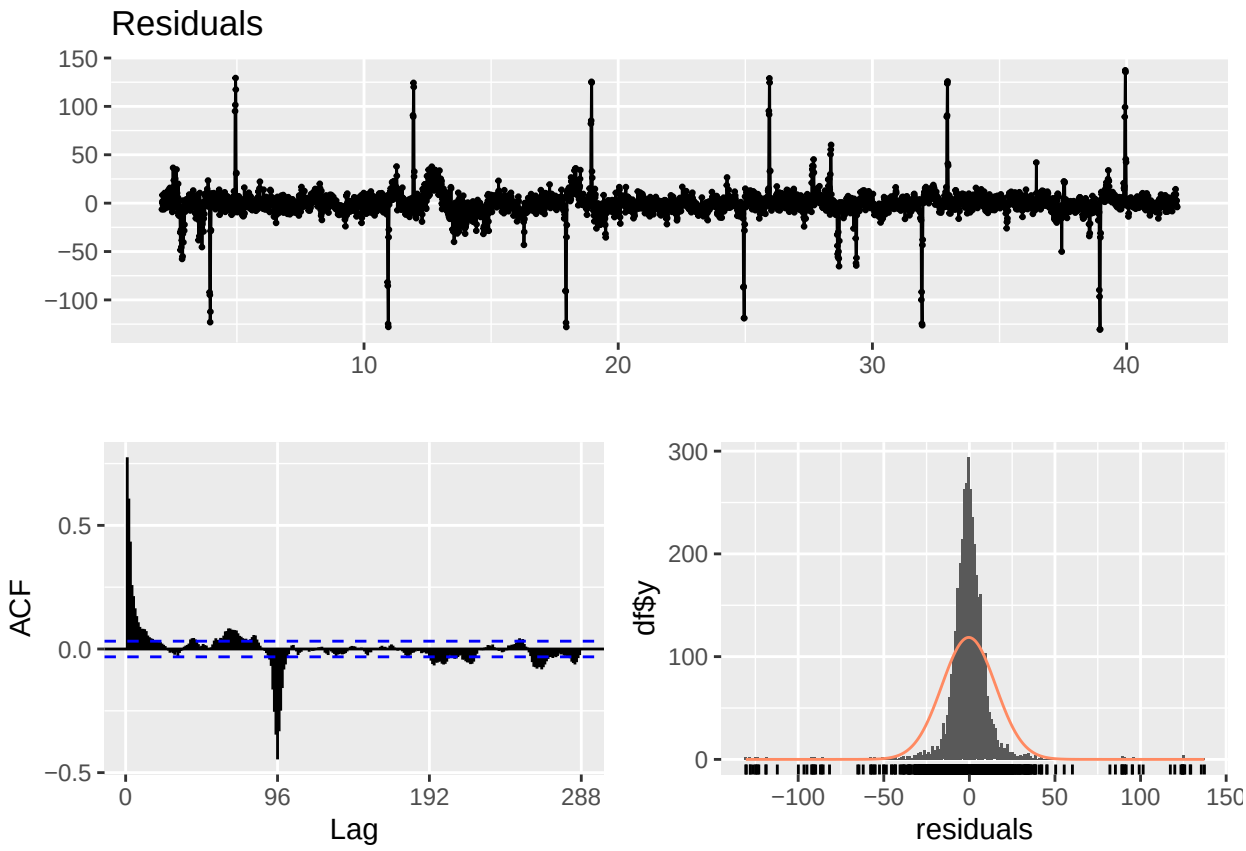

```
ggPacf(Elect_D.ts.train[,"Power_(kW)"])
```



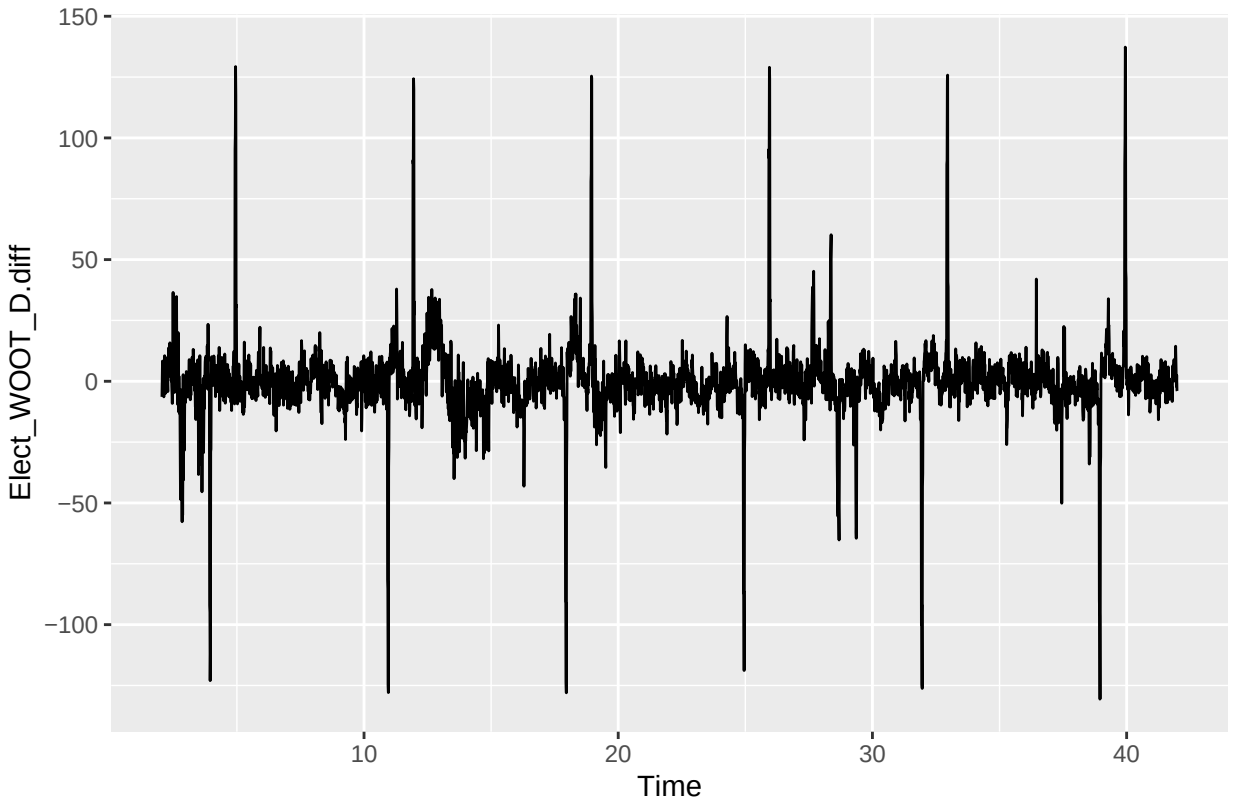

```
Elect_WOOT_D.diff <- diff(Elect_D.ts.train[,"Power_(kW)"] ,lag = 96, differences = 1)
```

```
checkresiduals(Elect_WOOT_D.diff)
```

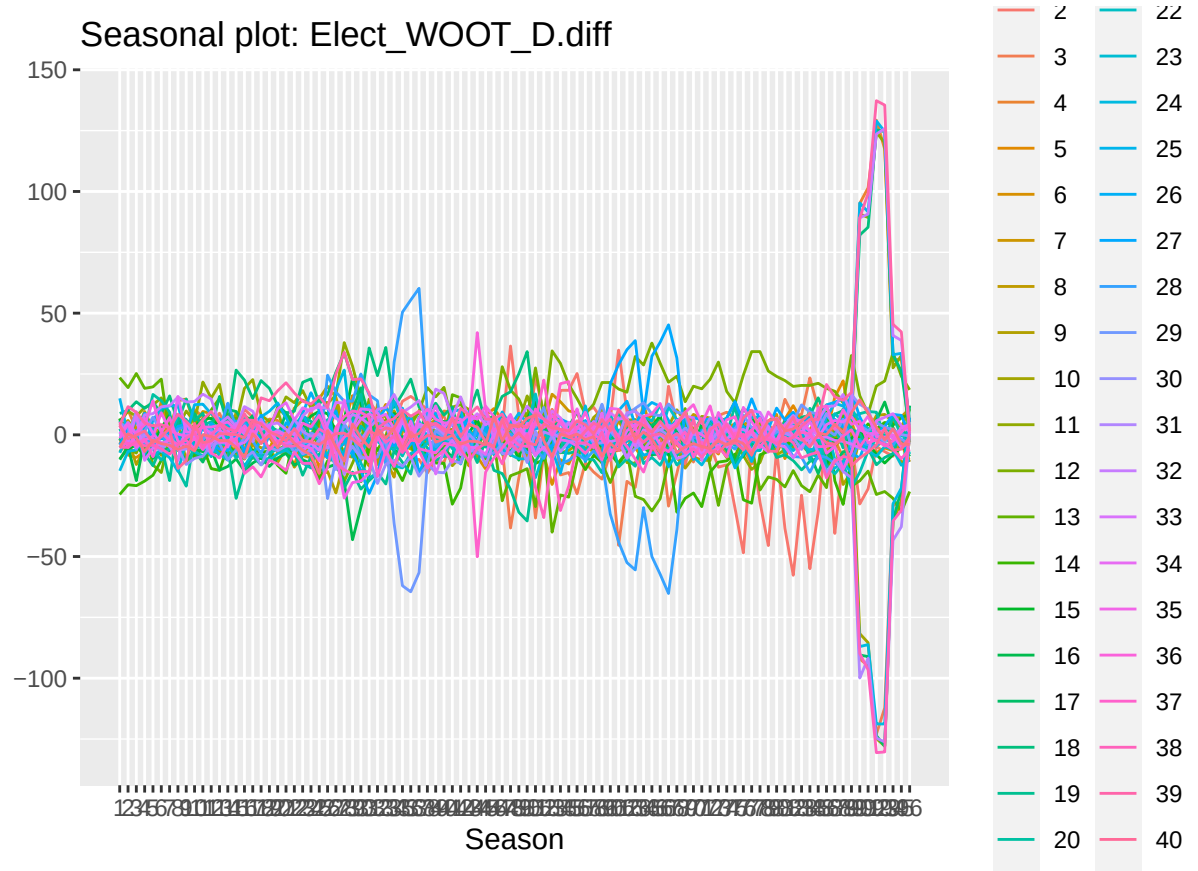
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



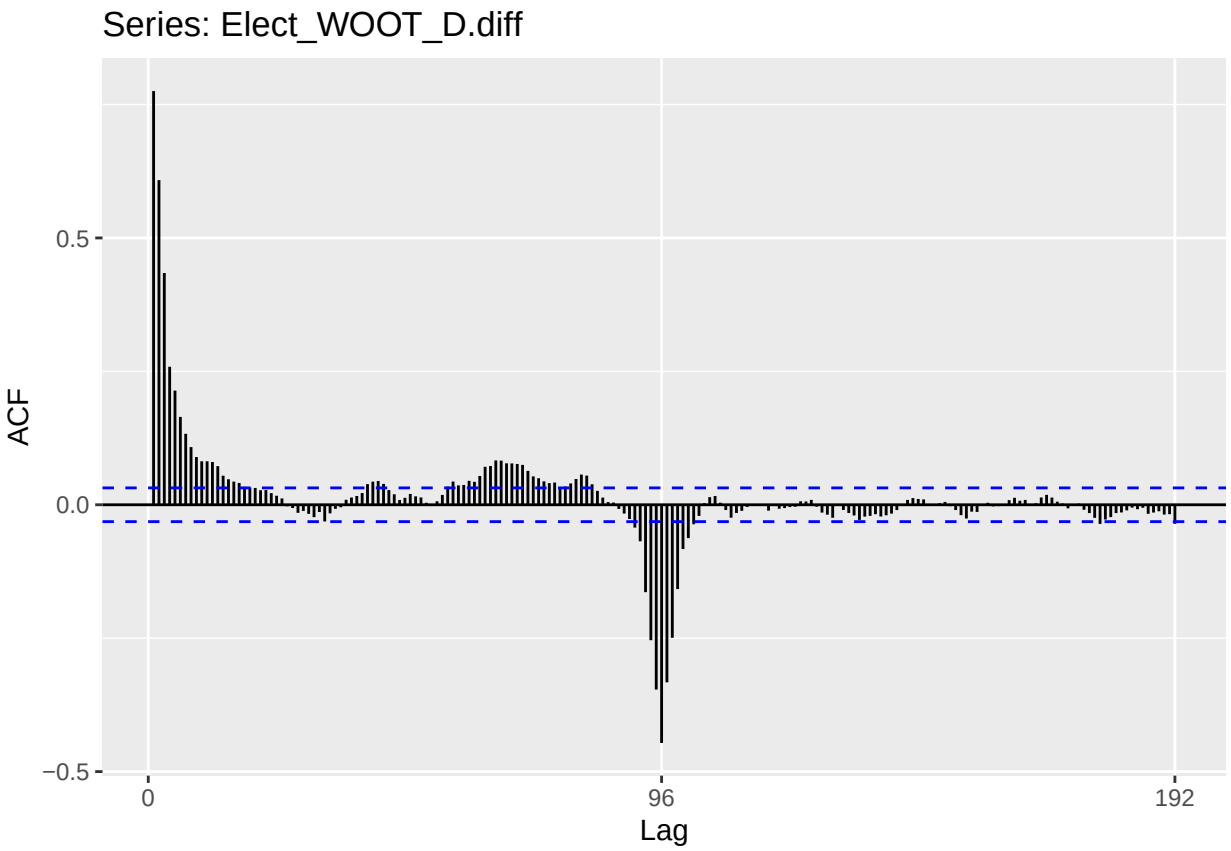

```
par(mfrow=c(2,2))
autoplot(Elect_WOOT_D.diff)
```



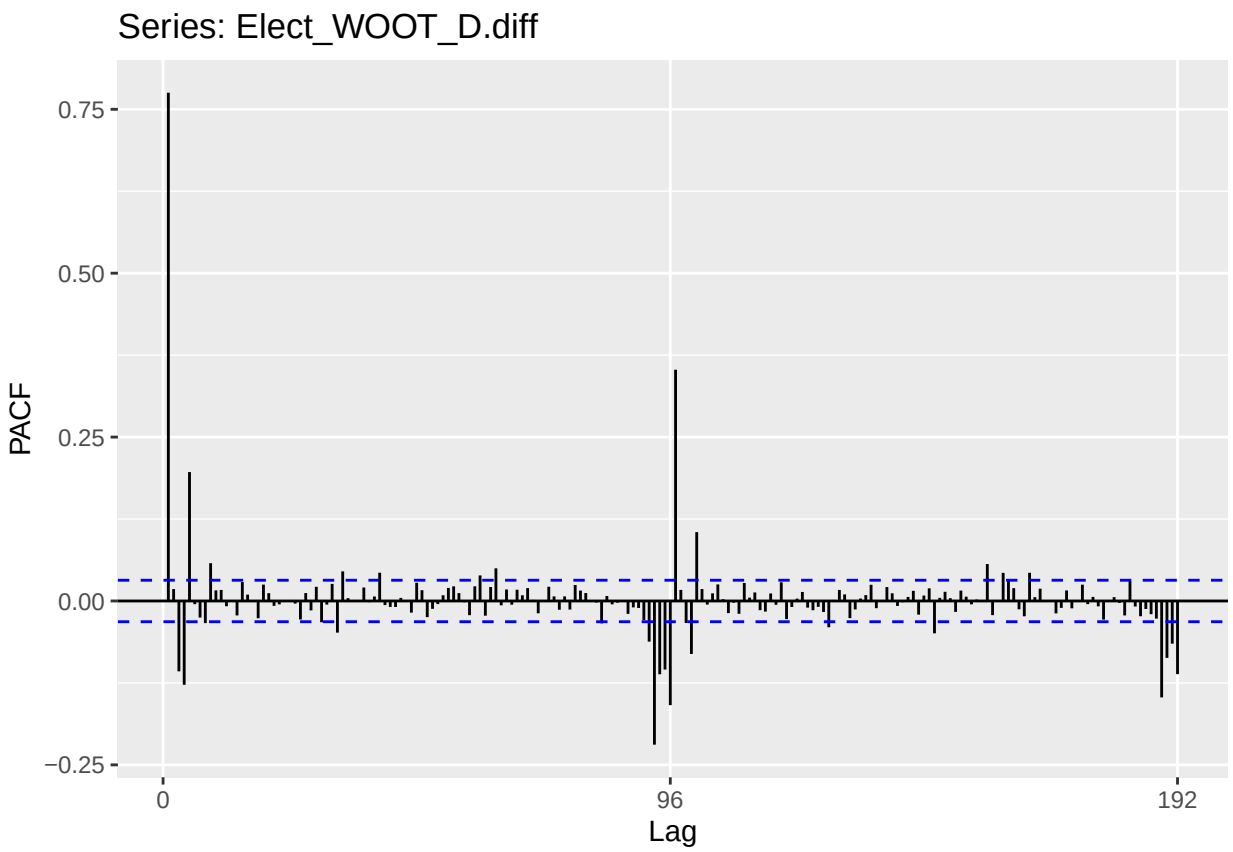
```
ggseasonplot(Elect_WOOT_D.diff)
```




```
ggAcf(Elect_WOOT_D.diff)
```

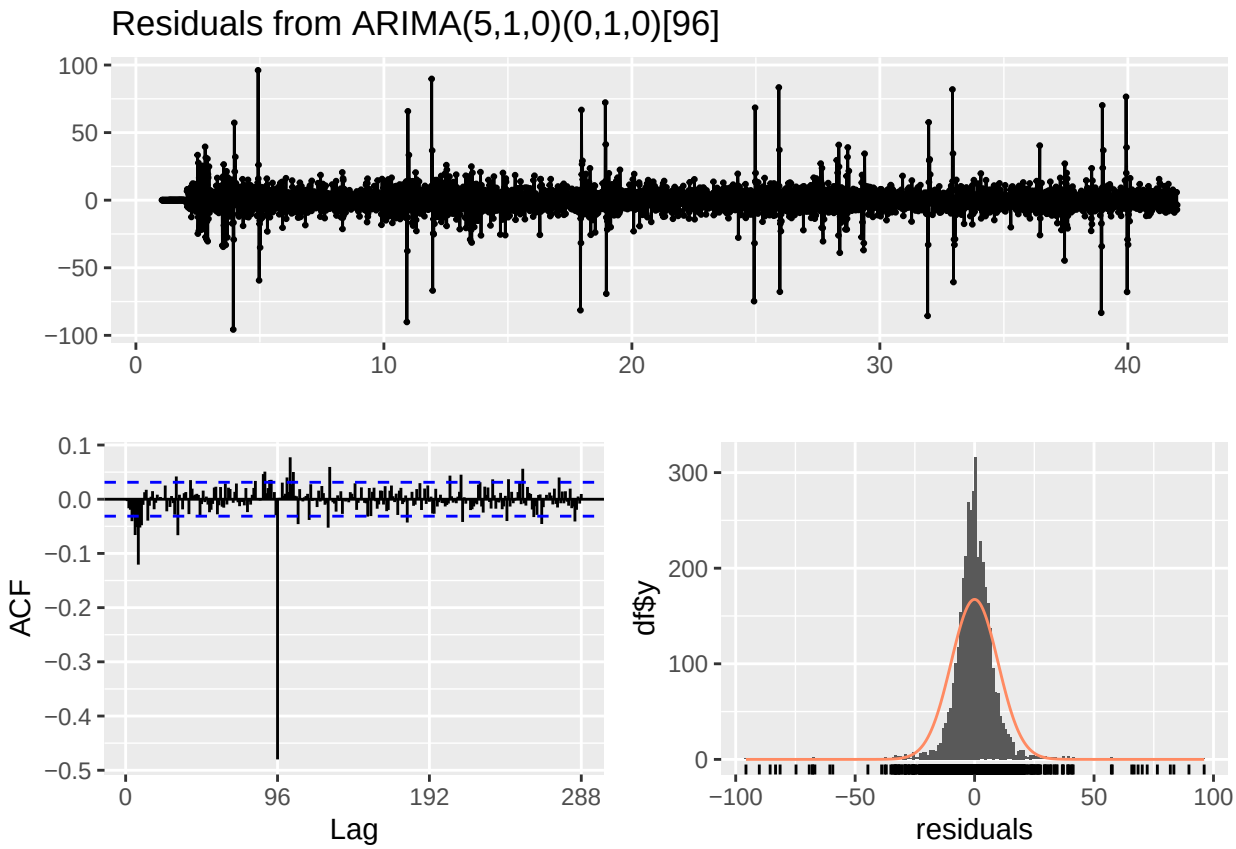


```
ggPacf(Elect_WOOT_D.diff)
```



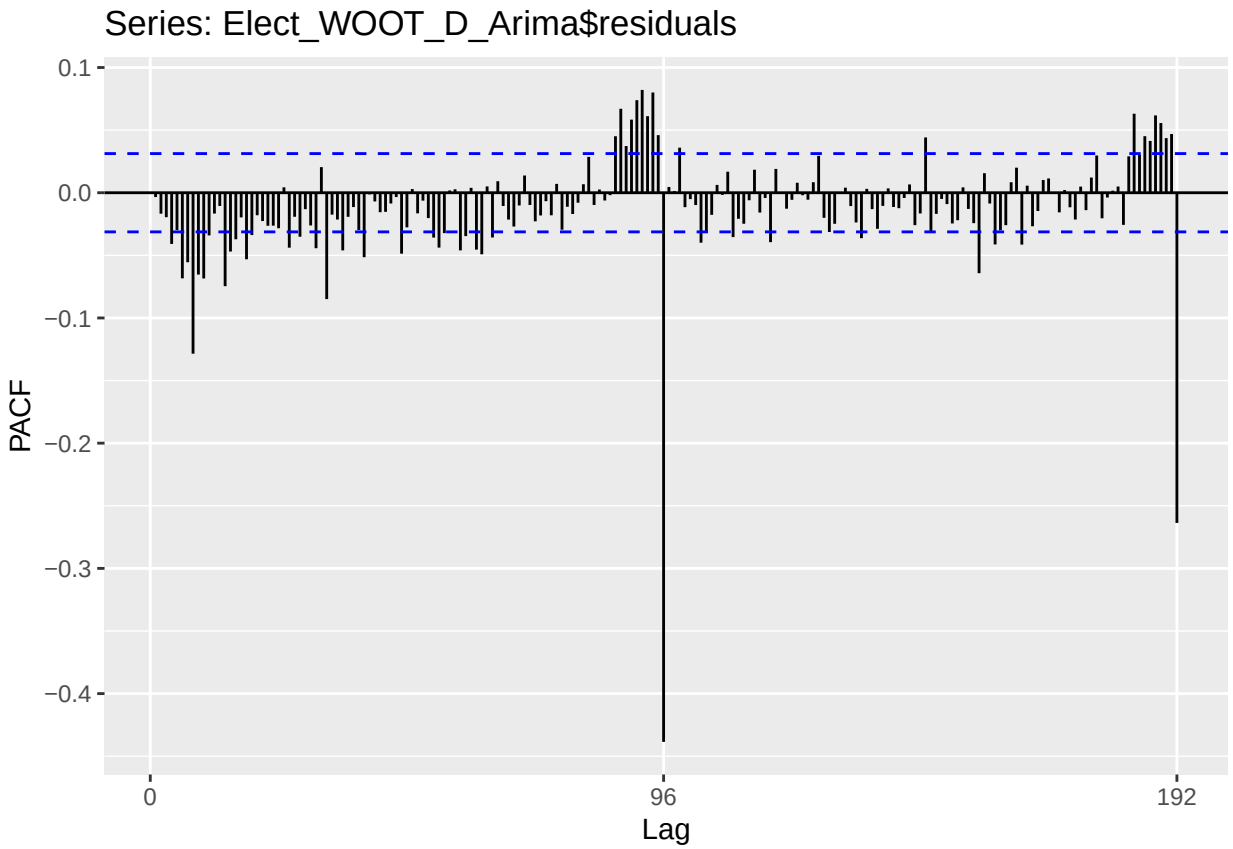
When we apply a diff on the time series with lag =96, we see that we have an ACF that decrease quickly but there is always a pic on lag 96. We can start with seasonal =(0,1,0) in SARIMA function. We see also on a significant PACF at lag=5. So, we can start with AR5, order=c(5,0,0).

```
Elect_WOOT_D_Arima=Arima(Elect_D.ts.train[,"Power_(kW)"], order=c(5,1,0), seasonal=c(0,1,0))
checkresiduals(Elect_WOOT_D_Arima)
```



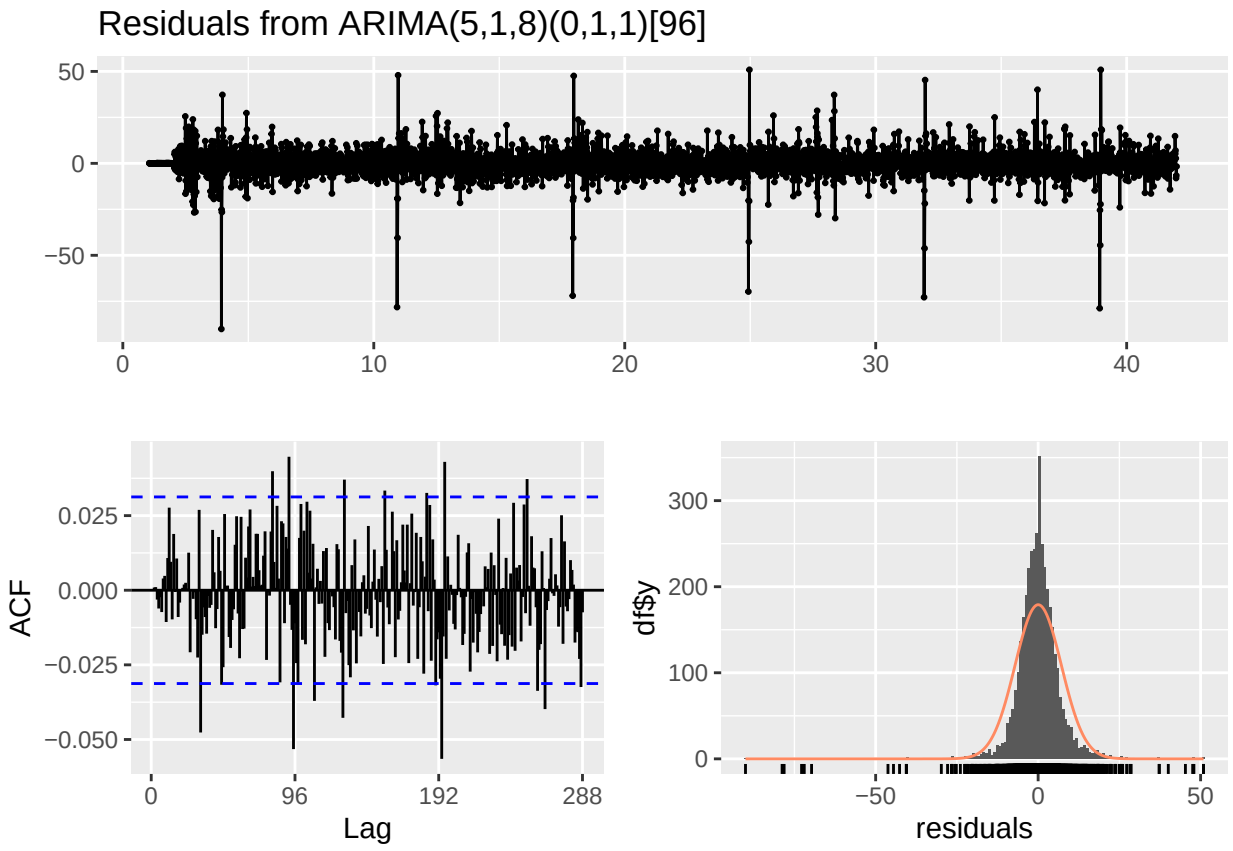

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,1,0)(0,1,0)[96]
## Q* = 1355.6, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

```
ggPacf(Elect_WOOT_D_Arima$residuals)
```



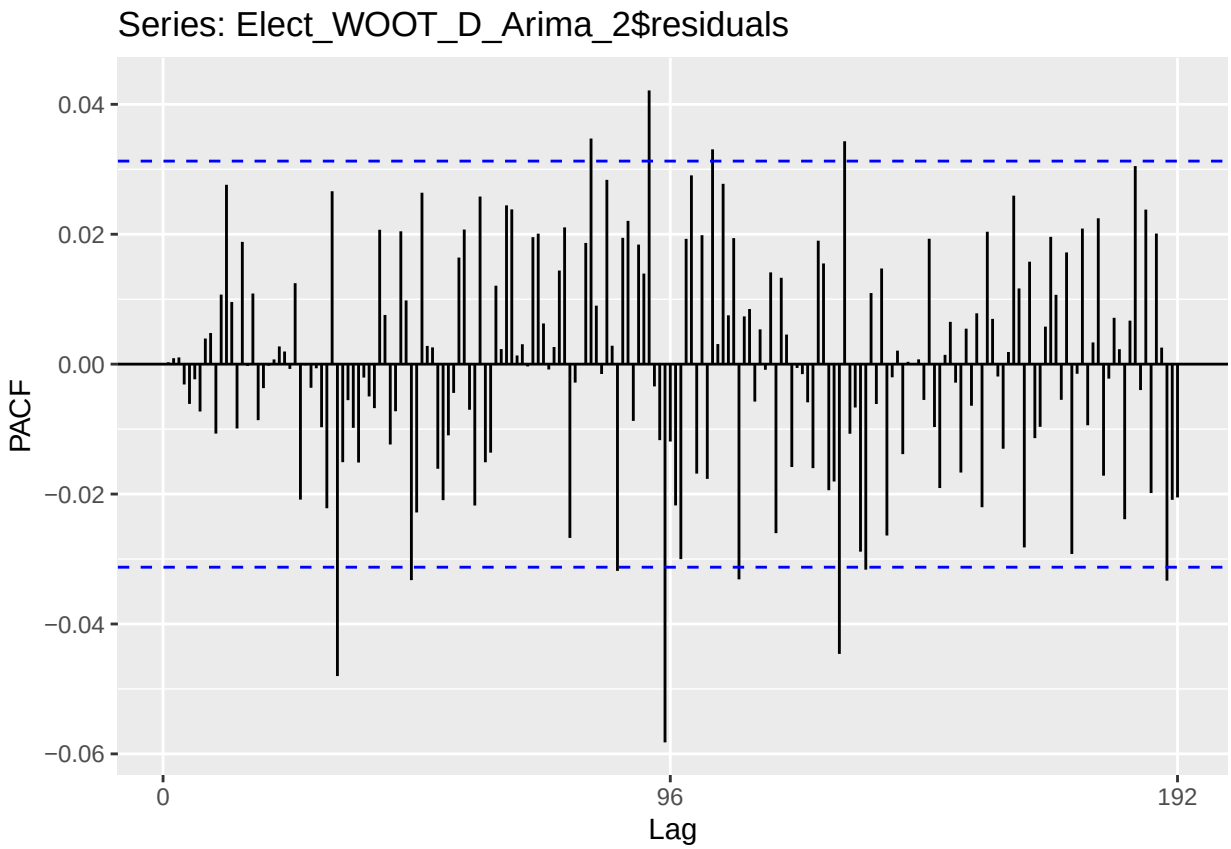
The correspondent residuals are still correlated. We see on the ACF and PACF that we have a significant ACF at lag=96 and significant PACF at lag 8. Thus, we can make the seasonal=(0,1,1) and order=(5,1,8)

```
Elect_WOOT_D_Arima_2=Arima(Elect_D.ts.train[,"Power_(kW)"], order=c(5,1,8), seasonal=c(0,1,1))
checkresiduals(Elect_WOOT_D_Arima_2)
```



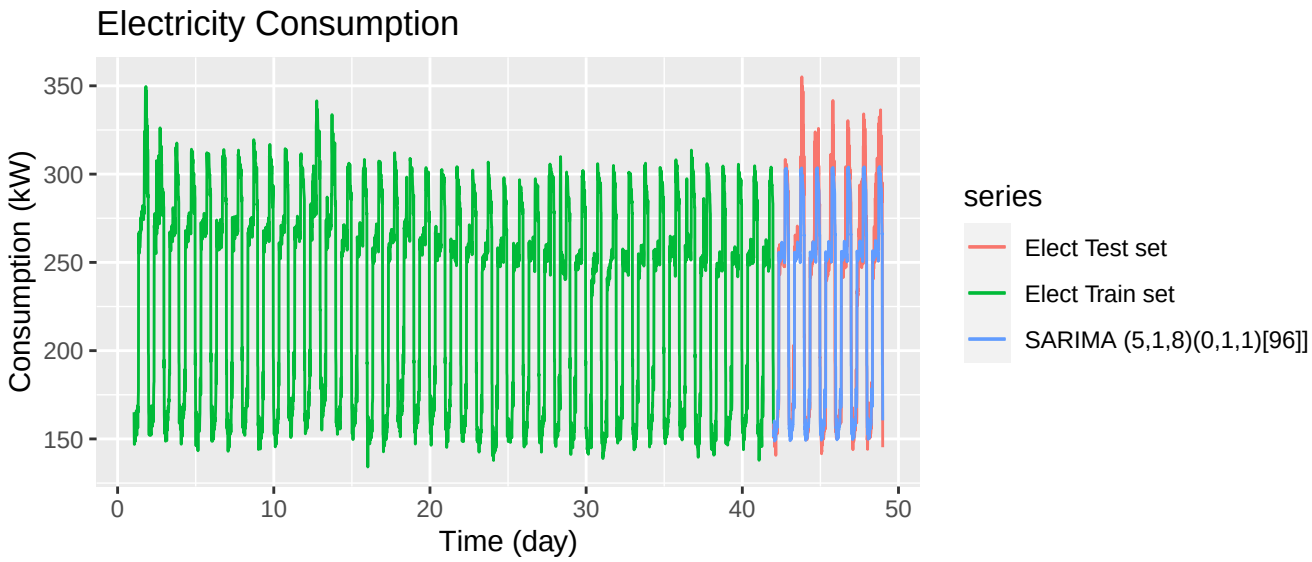
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,1,8)(0,1,1)[96]
## Q* = 229.74, df = 178, p-value = 0.005407
##
## Model df: 14.   Total lags used: 192
```

```
ggPacf(Elect_WOOT_D_Arima_2$residuals)
```



Here we can see that the residuals of this SARIMA model look better. They are not totally uncorrelated but the correlation is less than before. Let's look on the forecast on the test set and the RMSE.

```
Elect_WOOT_D_Arima_2_forcast <- forecast::forecast(Elect_WOOT_D_Arima_2, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_Arima_2_forcast$mean, series='SARIMA (5,1,8)(0,1,1)[96]]')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_Arima_2_RMSE <- sqrt(mean((Elect_WOOT_D_Arima_2_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOOT_D_Arima_2_AIC = Elect_WOOT_D_Arima$aicc

cat("The RMSE of SARIMA (5,1,8)(0,1,1)[96] model  equal to:", Elect_WOOT_D_Arima_2_RMSE, "\n")
```

```
## The RMSE of SARIMA (5,1,8)(0,1,1)[96] model  equal to: 15.87536
```

```
cat("The AIC of SARIMA (5,1,8)(0,1,1)[96] model  equal to:", Elect_WOOT_D_Arima_2_AIC, "\n")
```

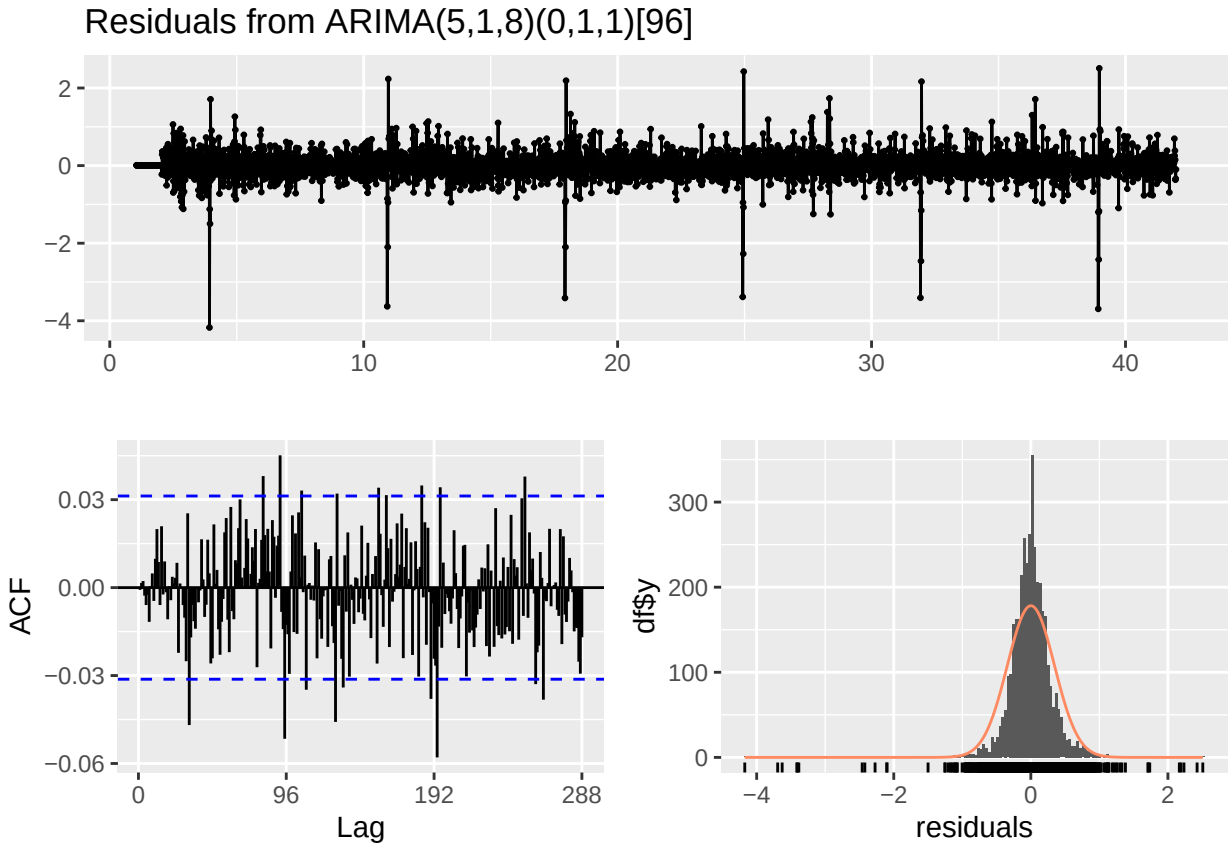
```
## The AIC of SARIMA (5,1,8)(0,1,1)[96] model  equal to: 28505.58
```

The RMSE on the test set is better that shows that we have less over fitting.

II-1-3-2- Forecasting Manuel SARIMA Model with Box-Cox transformation :

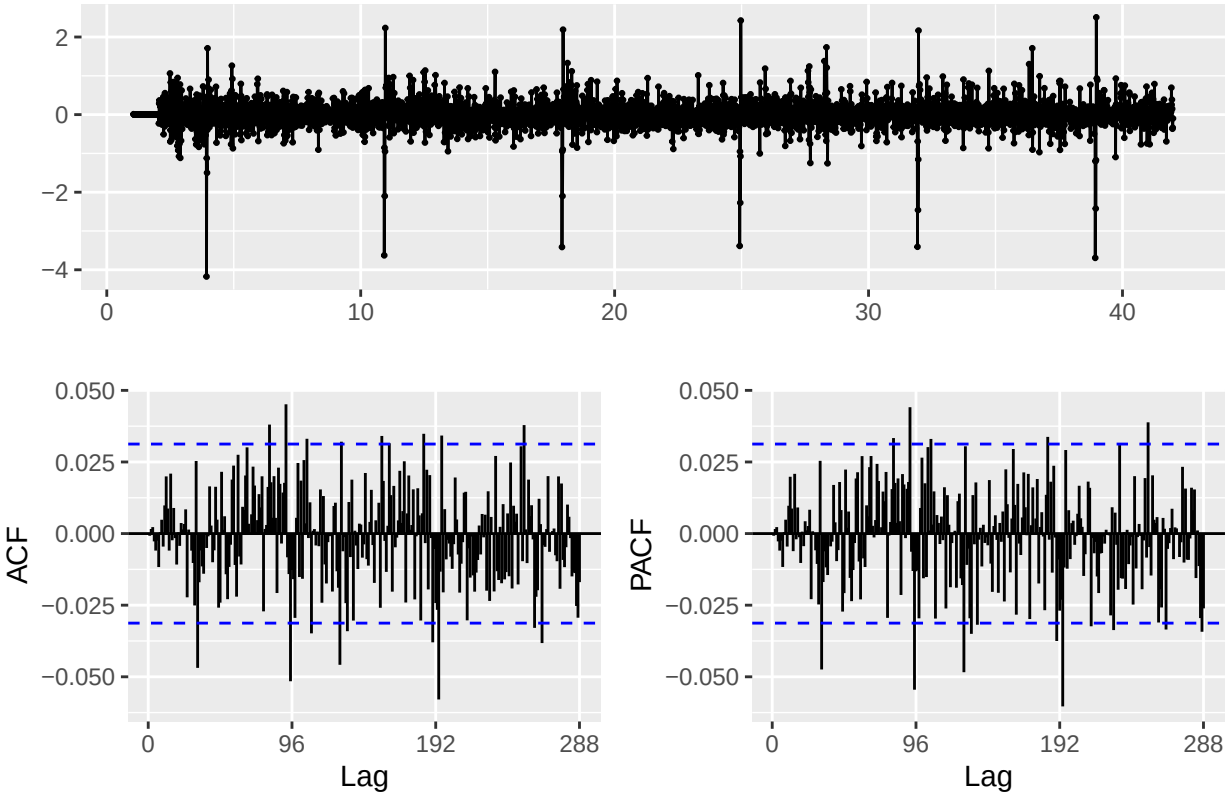
Lets see what's happen when we add a Box-Cox transformation to the SARIMA Model.

```
Elect_WOOT_D_Arima_2_BC=Arima(Elect_D.ts.train[,"Power_(kW)"], order=c(5,1,8), seasonal=c(0,1,1), lambda = "auto
checkresiduals(Elect_WOOT_D_Arima_2_BC)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,1,8)(0,1,1)[96]
## Q* = 221.76, df = 178, p-value = 0.01437
##
## Model df: 14.   Total lags used: 192
```

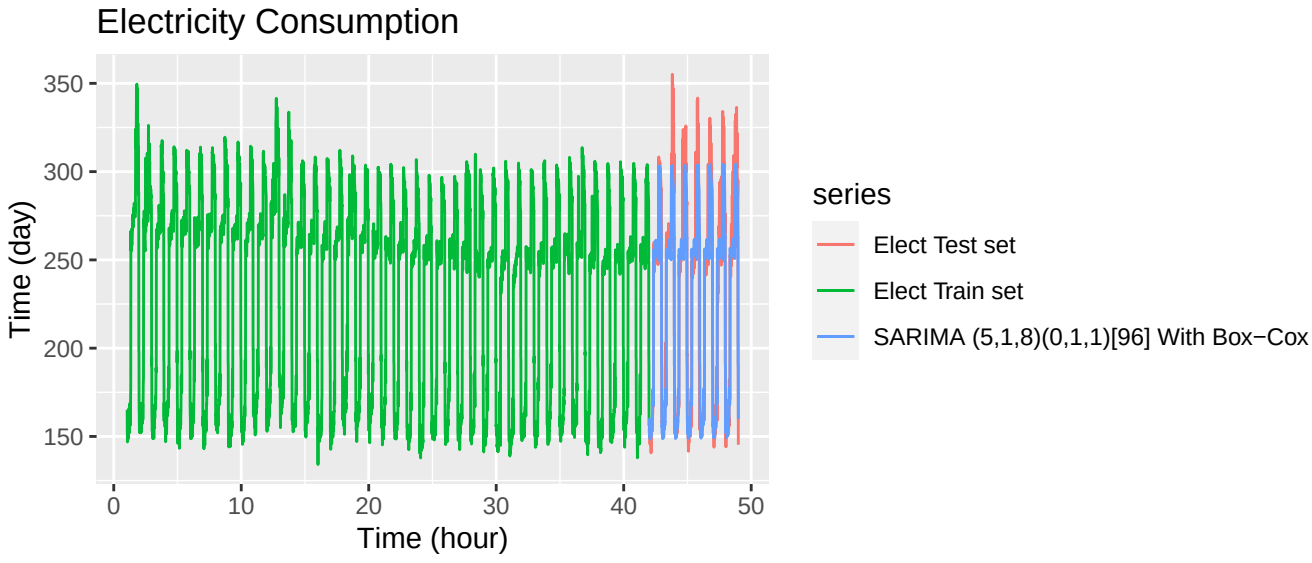
```
ggtsdisplay(Elect_WOOT_D_Arima_2_BC$residuals)
```



Whith the Box-Cox transformation, the residuals seems better but stay have some correlation. The Ljung-Box test gives a P-value equal to 0.015 which is < 0.05 but better then the models already seen.

Lets take a look on the forecast and the RMSE with SARIMA and Box-Cox transformation.

```
Elect_WOOT_D_Arima_2_BC_forcast <- forecast::forecast(Elect_WOOT_D_Arima_2_BC, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_Arima_2_BC_forcast$mean, series='SARIMA (5,1,8)(0,1,1)[96] With Box-Cox')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (hour)') +
  ylab('Time (day)')
```



```
Elect_WOOT_D_Arima_BC_RMSE <- sqrt(mean((Elect_WOOT_D_Arima_2_BC_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2)
Elect_WOOT_D_Arima_BC_AIC = Elect_WOOT_D_Arima_2_BC$aicc

cat("The RMSE of SARIMA (5,1,8)(0,1,1)[96] model with Box-Cox transformation equal to:", Elect_WOOT_D_Arima_BC_R
```

```
## The RMSE of SARIMA (5,1,8)(0,1,1)[96] model with Box-Cox transformation equal to: 15.88234
```

```
cat("The AIC of SARIMA (5,1,8)(0,1,1)[96] model with Box-Cox transformation equal to:", Elect_WOOT_D_Arima_BC_AI
```

```
## The AIC of SARIMA (5,1,8)(0,1,1)[96] model with Box-Cox transformation equal to: 2955.701
```

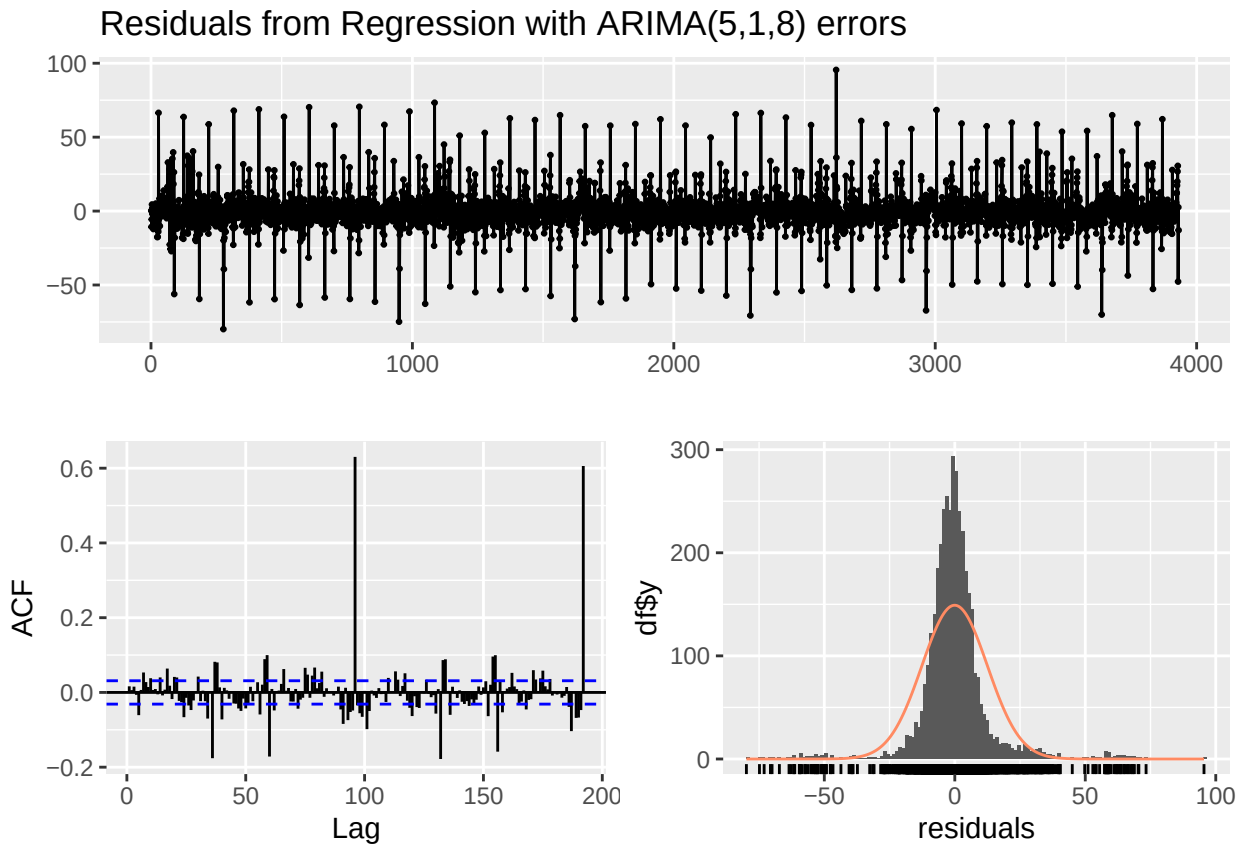
The forecast and the RMSE with Box-Cox seem the same as the SARIMA model without Box-Cox transformation.

II-1-3-3- Forecasting Manuel SARIMA Model with Fourrier transformation :

What will be the result if we add a Fourier transformation to the SARIMA Model.

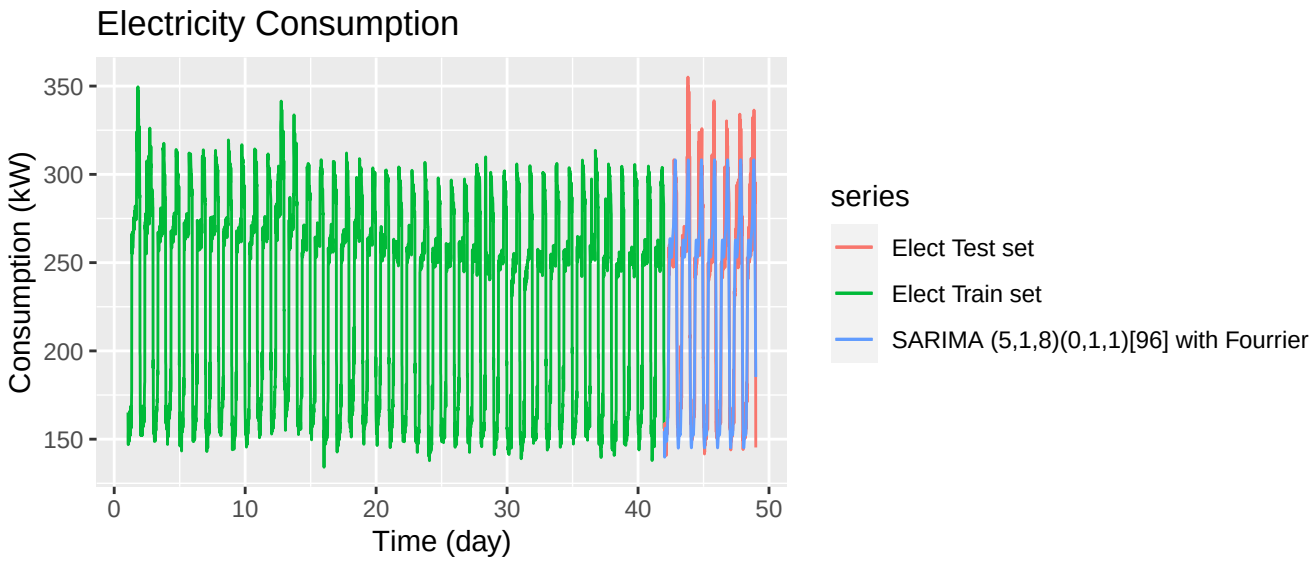
```
Elect_WOOT_D_Arima_FT <- Arima(ts(Elect.train_TF[,"Power_(kW)"]), order=c(5,1,8), seasonal=c(0,1,1), xreg=
                               fourier(ts(Elect.train_TF[,"Power_(kW)"], frequency= 96), K=4))
```

```
checkresiduals(Elect_WOOT_D_Arima_FT, lag.max=192)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,8) errors
## Q* = 45.216, df = 3, p-value = 8.325e-10
##
## Model df: 13.   Total lags used: 16
```

```
Elect_WOOT_D_Arima_FT_forcast <- forecast::forecast(Elect_WOOT_D_Arima_FT, xreg=
                                 fourier(ts(Elect.test_TF[,"Power_(kW)"], frequency= 96), K=4, h=672))
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(ts(Elect_WOOT_D_Arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series='SARIMA (5,1,8)(0,1
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_Arima_FT_RMSE <- sqrt(mean(as.numeric(ts(Elect_WOOT_D_Arima_FT_forcast$mean, frequency = 96, start
Elect_WOOT_D_Arima_FT_AIC = Elect_WOOT_D_Arima_FT$aicc

cat("The RMSE of SARIMA (5,1,8)(0,1,1)[96] model with Fourier transformation equal to:", Elect_WOOT_D_Arima_FT_R
```

```
## The RMSE of SARIMA (5,1,8)(0,1,1)[96] model with Fourier transformation equal to: 19.47428
```

```
cat("The AIC of SARIMA (5,1,8)(0,1,1)[96] model with Fourier transformation equal to:", Elect_WOOT_D_Arima_FT_AI
```

```
## The AIC of SARIMA (5,1,8)(0,1,1)[96] model with Fourier transformation equal to: 31081.26
```

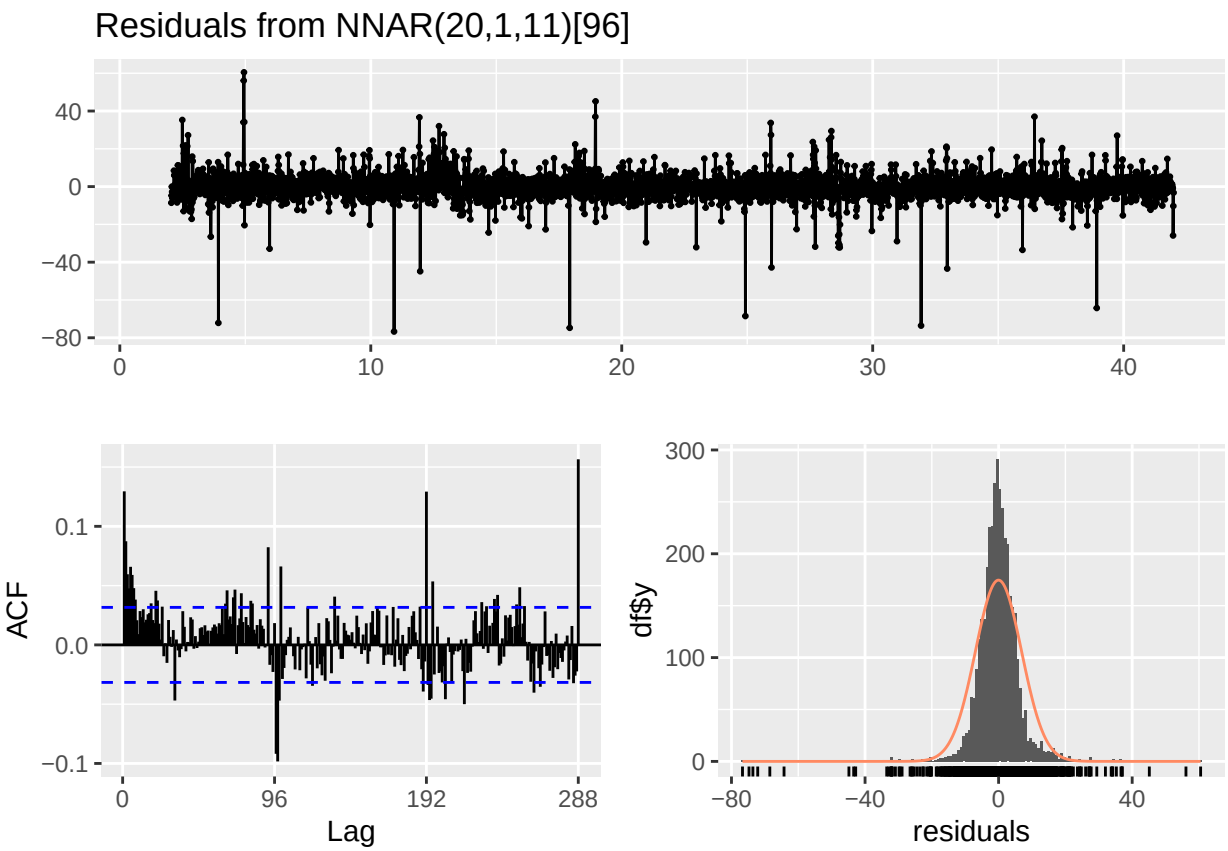
We can see that Fourier transformation has a negative impact on the SARIMA model because the residuals of the SARIMA model with Fourier transformation are very correlated and the RMSE on the test set is worse (over fitting more present here).

II-1-4- Forecasting with Neural Network:

II-1-4-1- Forecasting with Neural Network:

```
Elect_WOOT_D_NN=nnetar(Elect_D.ts.train[,"Power_(kW)"])
checkresiduals(Elect_WOOT_D_NN)
```

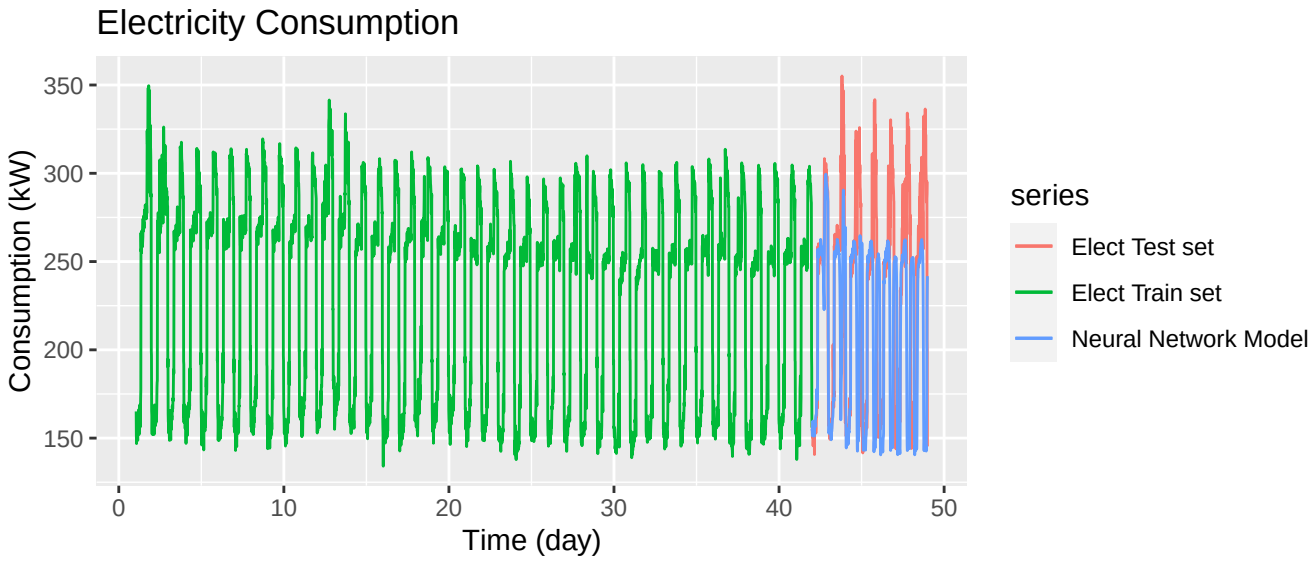
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



With Neural Network AutoRegressive, we get a model with residuals very correlated.

What about the forecast and the RMSE with Neural Network AutoRegressiv model?

```
Elect_WOOT_D_NN_forcast <- forecast::forecast(Elect_WOOT_D_NN, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_NN_forcast$mean, series='Neural Network Model')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_NN_RMSE <- sqrt(mean((Elect_WOOT_D_NN_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOOT_D_NN_AIC = Elect_WOOT_D_NN$aicc
```

```
cat("The RMSE of Neural Network model equal to:", Elect_WOOT_D_NN_RMSE, "\n")
```

```
## The RMSE of Neural Network model equal to: 68.65859
```

```
cat("The AIC of Neural Network model equal to:", Elect_WOOT_D_NN_AIC, "\n")
```

```
## The AIC of Neural Network model equal to:
```

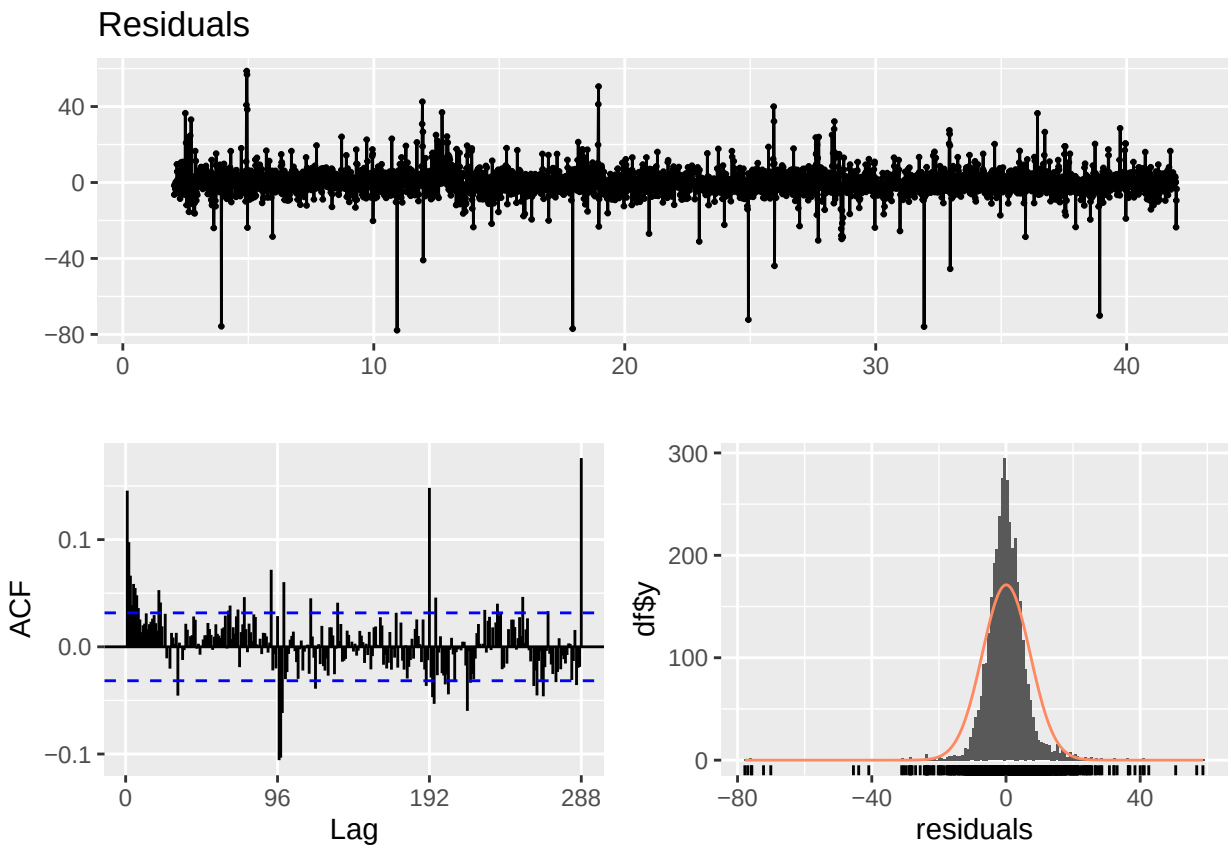
We can see that the forecast using Neural Network AutoRegressiv model are not good and the RMSE on the test set confirm that because the RMSE is so high and shows the presence of over fitting.

II-1-4-2- Forecasting with Neural Network and Box-Cox transformation:

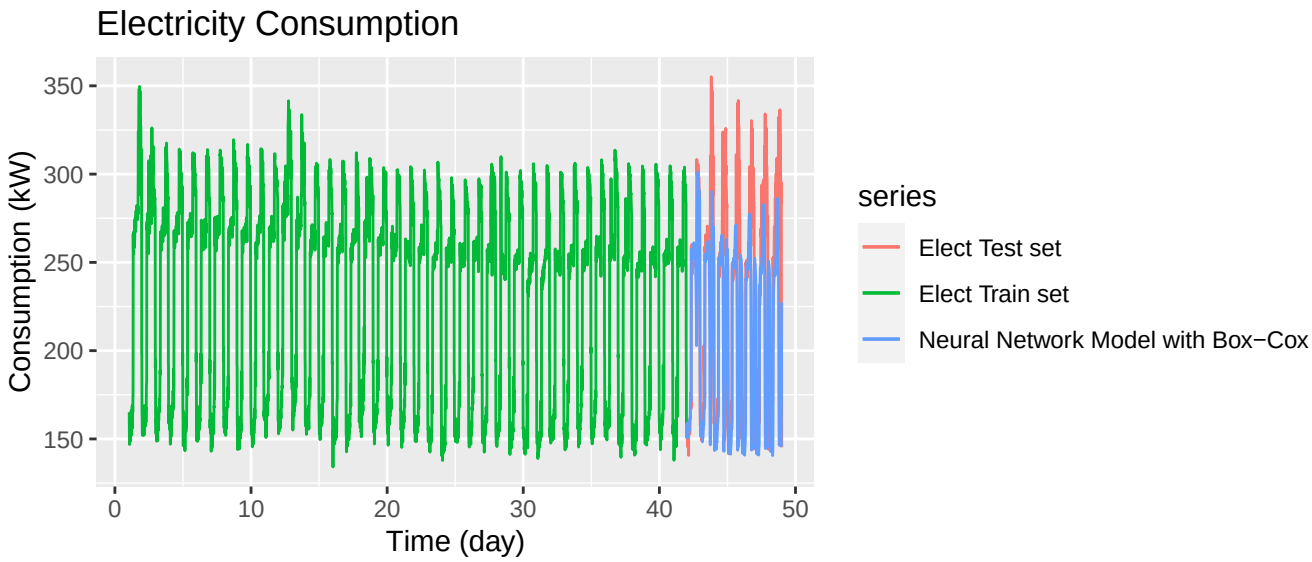
```
Elect_WOOT_D_NN_BC=nnetar(Elect_D.ts.train[,"Power_(kW)"], lambda='auto')
```

```
checkresiduals(Elect_WOOT_D_NN_BC$residuals)
```

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




```
Elect_WOOT_D_NN_BC_forcast <- forecast::forecast(Elect_WOOT_D_NN_BC, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(Elect_WOOT_D_NN_BC_forcast$mean, series='Neural Network Model with Box-Cox')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_NN_BC_RMSE <- sqrt(mean((Elect_WOOT_D_NN_BC_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOOT_D_NN_BC_AIC = Elect_WOOT_D_NN_BC$aicc

cat("The RMSE of Neural Network model with Box-Cox transformation equal to:", Elect_WOOT_D_NN_BC_RMSE, "\n")
```

```
## The RMSE of Neural Network model with Box-Cox transformation equal to: 66.18553
```

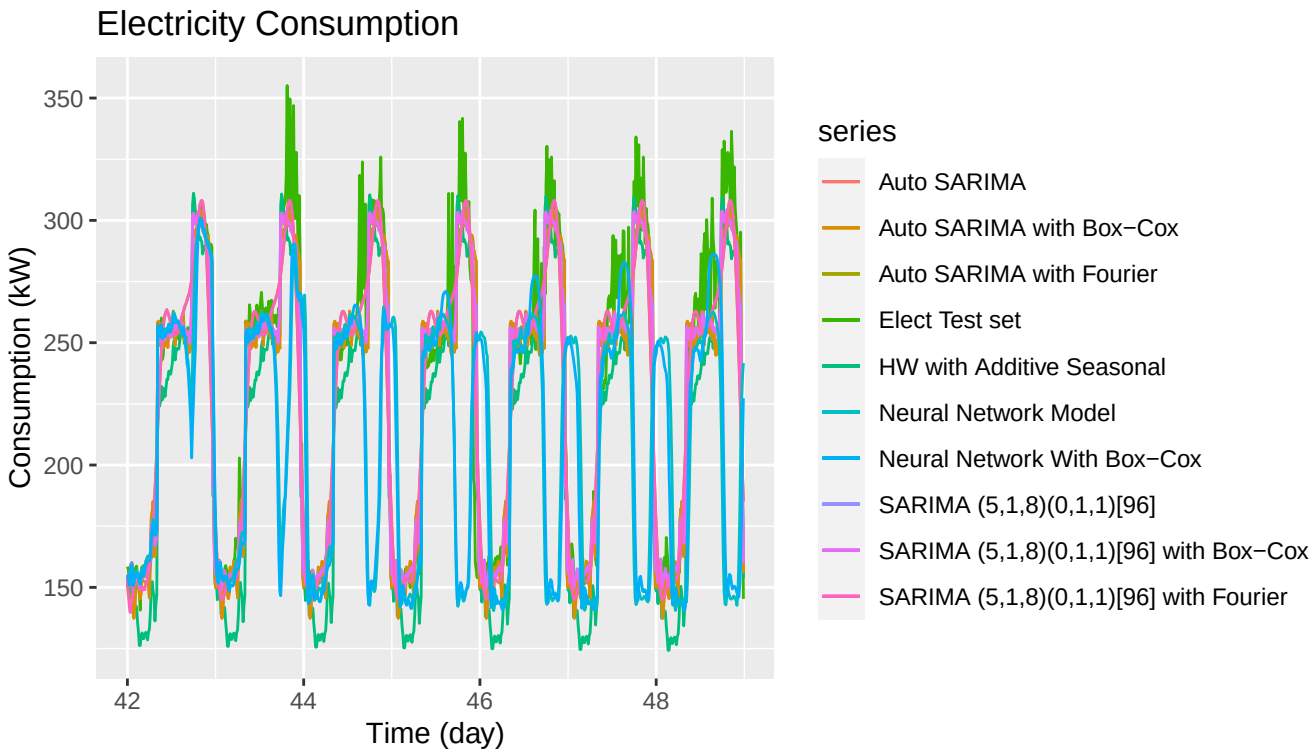
```
cat("The AIC of Neural Network model with Box-Cox transformation equal to:", Elect_WOOT_D_NN_BC_AIC, "\n")
```

```
## The AIC of Neural Network model with Box-Cox transformation equal to:
```

When we add a Cox-Box transformation to NNAR model, the result don't change a lot and the model still have residuals that are noisy and an over fitting problem.

II-1-5- Ploting all the models

```
autoplot(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D_hw_forecast$mean, series='HW with Additive Seasonal') +
  autolayer(Elect_WOOT_D_auto.arima_forcast$mean, series='Auto SARIMA') +
  autolayer(Elect_WOOT_D_auto.arima_BC_forcast$mean, series='Auto SARIMA with Box-Cox') +
  autolayer(ts(Elect_WOOT_D_auto.arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series='Auto SARIMA w
  autolayer(Elect_WOOT_D_Arima_2_forcast$mean, series='SARIMA (5,1,8)(0,1,1)[96]') +
  autolayer(Elect_WOOT_D_Arima_2_BC_forcast$mean, series='SARIMA (5,1,8)(0,1,1)[96] with Box-Cox') +
  autolayer(ts(Elect_WOOT_D_Arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series='SARIMA (5,1,8)(0,1
  autolayer(Elect_WOOT_D_NN_forcast$mean, series='Neural Network Model') +
  autolayer(Elect_WOOT_D_NN_BC_forcast$mean, series='Neural Network With Box-Cox')+
  #autolayer(Elect_WOOT_D_TBATS_forcast$mean, series='TBATS Model')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```



II-1-6- Comparison of the RMSE, AIC and Ljung Box test

```
results <- data.frame(
  Model_without_covariate = c("Holt Winters Model (Additive Seasonality)","Holt Winters Model (Multiplicative Se
  RMSE = c(Elect_WOOT_hw_RMSE, Elect_WOOT_D_hw_MS_RMSE, Elect_WOOT_D_auto.arima_RMSE, Elect_WOOT_D_auto.arima_BC
  AICC = c("-","-", Elect_WOOT_D_auto.arima_AIC, Elect_WOOT_D_auto.arima_BC_AIC, Elect_WOOT_D_auto.arima_FT_AIC,
  Ljung_Box_test = c("-","-","2.2e-16","2.2e-16","2.2e-16",0.005407,0.01437,"8.325e-10","-","-")

)

results
```

```
##                                      Model_without_covariate     RMSE
## 1                       Holt Winters Model (Additive Seasonality) 23.57358
## 2                 Holt Winters Model (Multiplicative Seasonality) 17.95675
## 3                                             Auto SARIMA Model  17.22069
## 4                   Auto SARIMA Model with Box-Cox transformation 17.22011
## 5                  Auto SARIMA Model with Fourrier transformation 19.51704
## 6                               SARIMA Model (5,1,8)(0,1,1)[96]') 15.87536
## 7    SARIMA Model (5,1,8)(0,1,1)[96] with Box-Cox Transformation 15.88234
## 8  SARIMA Model (5,1,8)(0,1,1)[96] with Fourrier Transformation 19.47428
## 9                                           Neural Network Model 68.65859
## 10               Neural Network Model with Box-Cox Transformation 66.18553
##                 AICC Ljung_Box_test
## 1                  -              -
## 2                  -              -
## 3  28481.4681068343        2.2e-16
## 4  28481.4681068343        2.2e-16
## 5  31125.7548633708        2.2e-16
## 6  28505.5834461697       0.005407
## 7  2955.70128856105        0.01437
## 8  31081.2639278966      8.325e-10
## 9                  -              -
## 10                 -              -
```

II-2- Fitting Electricity Consumption (kW) models with using outdoor temperature: (with covariate)

Now, we will try to fit a time series model for the Electricity consumption time series with Temperature as covariate. We will do the same job as the analyse without covariate.

II-2-1- Forecasting with exponential smoothing with covariate

II-2-1-1- Holt Winters with Additive Seasonality

II-2-1-2- Holt Winters with Multiplicative Seasonality

As we have a high frequency time series (frequency = 96), the "hw" function don't work for such time series (hw needs a time series with a frequency less then 24). For this reason, we have used "HoltWinters" function to deal with the high frequency problem.

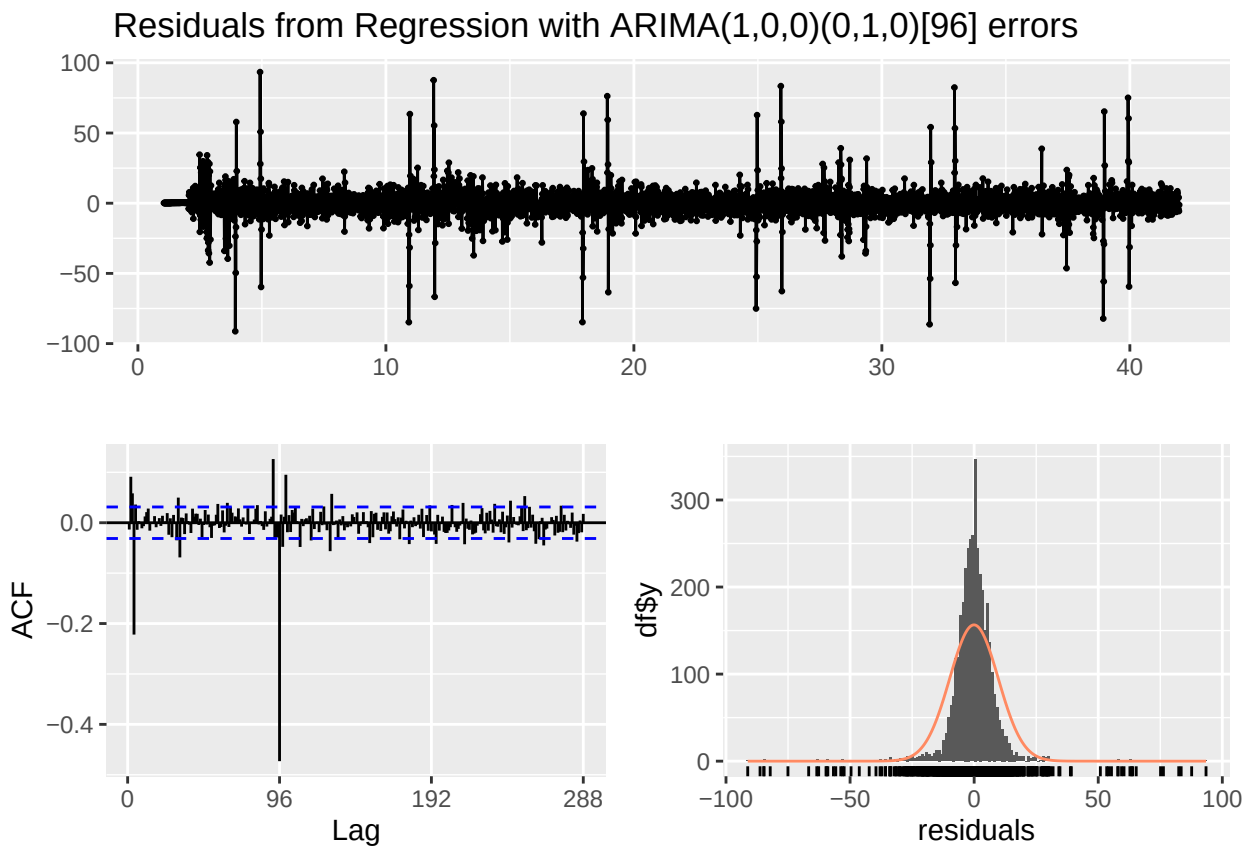
But, the limit of "HoltWinters" function is that it don't accept a covariate (xreg is not accepted in "HoltWinters" function in R).

Thus, we will not fit Holt Winters models for the Electricity consumption time series with Temperature as covariate.

II-2-2- Forecasting Auto SARIMA Model with covariate

II-2-2-1- Forecasting Auto SARIMA Model with covariate

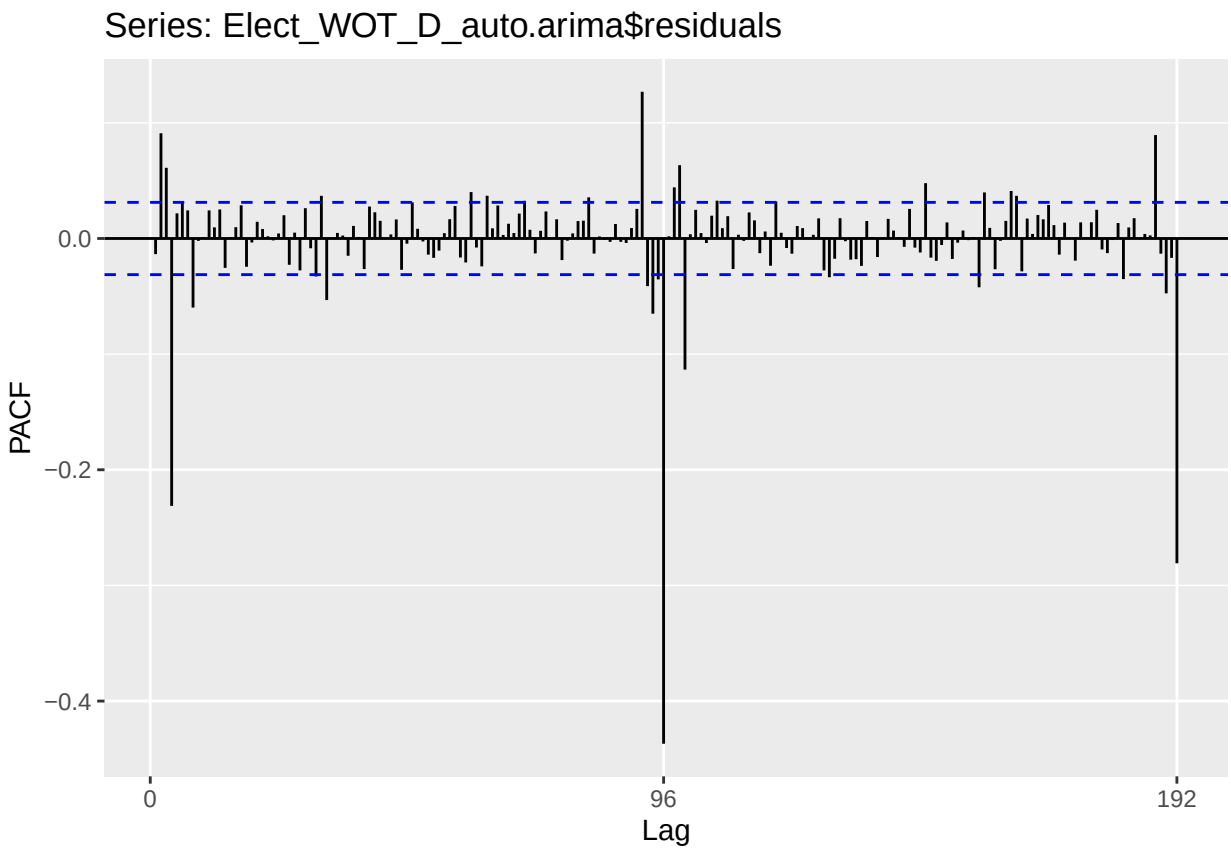
```
Elect_WOT_D_auto.arima <- auto.arima(Elect_D.ts.train[,"Power_(kW)"], xreg= Elect_D.ts.train[,"Temp_(C°)"])
checkresiduals(Elect_WOT_D_auto.arima)
```



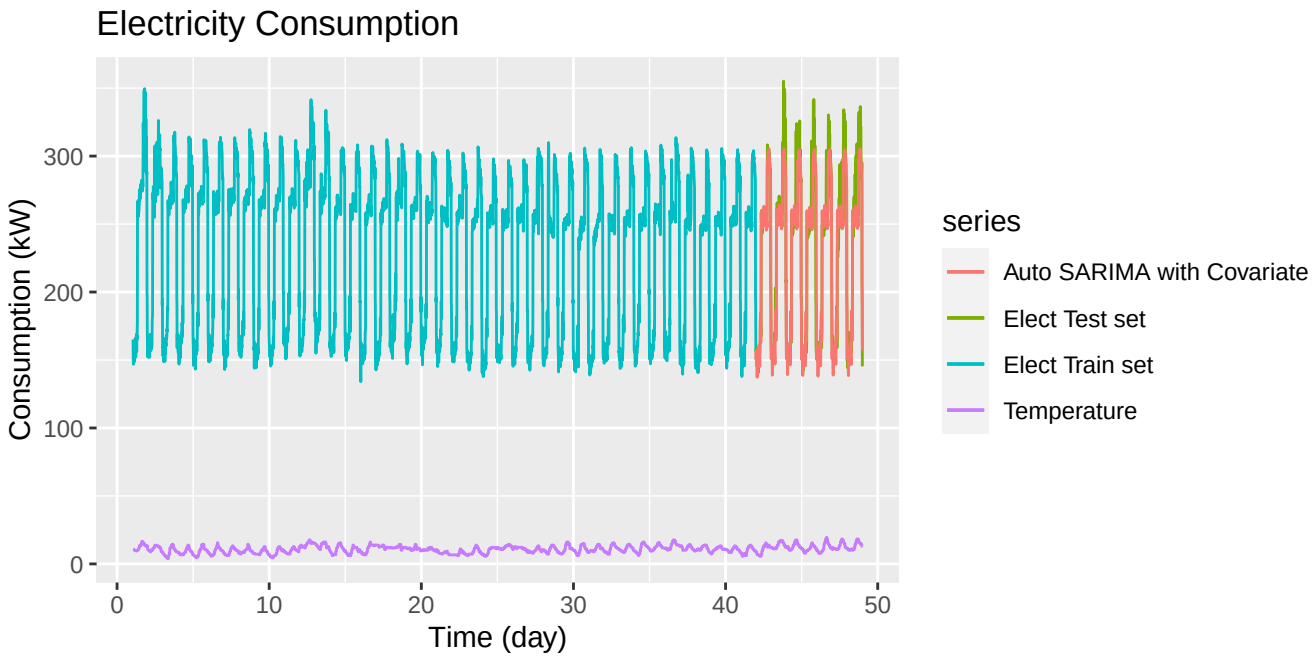

```
##
##  Ljung-Box test
##
```

```
## data:  Residuals from Regression with ARIMA(1,0,0)(0,1,0)[96] errors
## Q* = 1534.6, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

```
ggPacf(Elect_WOT_D_auto.arima$residuals)
```




```
Elect_WOT_D_auto.arima_forcast <- forecast :: forecast(Elect_WOT_D_auto.arima , xreg= Elect_D.ts.test[,"Temp_(C°

autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_auto.arima_forcast$mean, series='Auto SARIMA with Covariate')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_auto.arima_RMSE <- sqrt(mean((Elect_WOT_D_auto.arima_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2)
Elect_WOT_D_auto.arima_AIC = Elect_WOT_D_auto.arima$aicc

cat("The RMSE of Auto SARIMA model with Covariate equal to:", Elect_WOT_D_auto.arima_RMSE, "\n")
```

```
## The RMSE of Auto SARIMA model with Covariate equal to: 16.87297
```

```
cat("The AIC of Auto SARIMA model with Covariate equal to:", Elect_WOT_D_auto.arima_AIC, "\n")
```

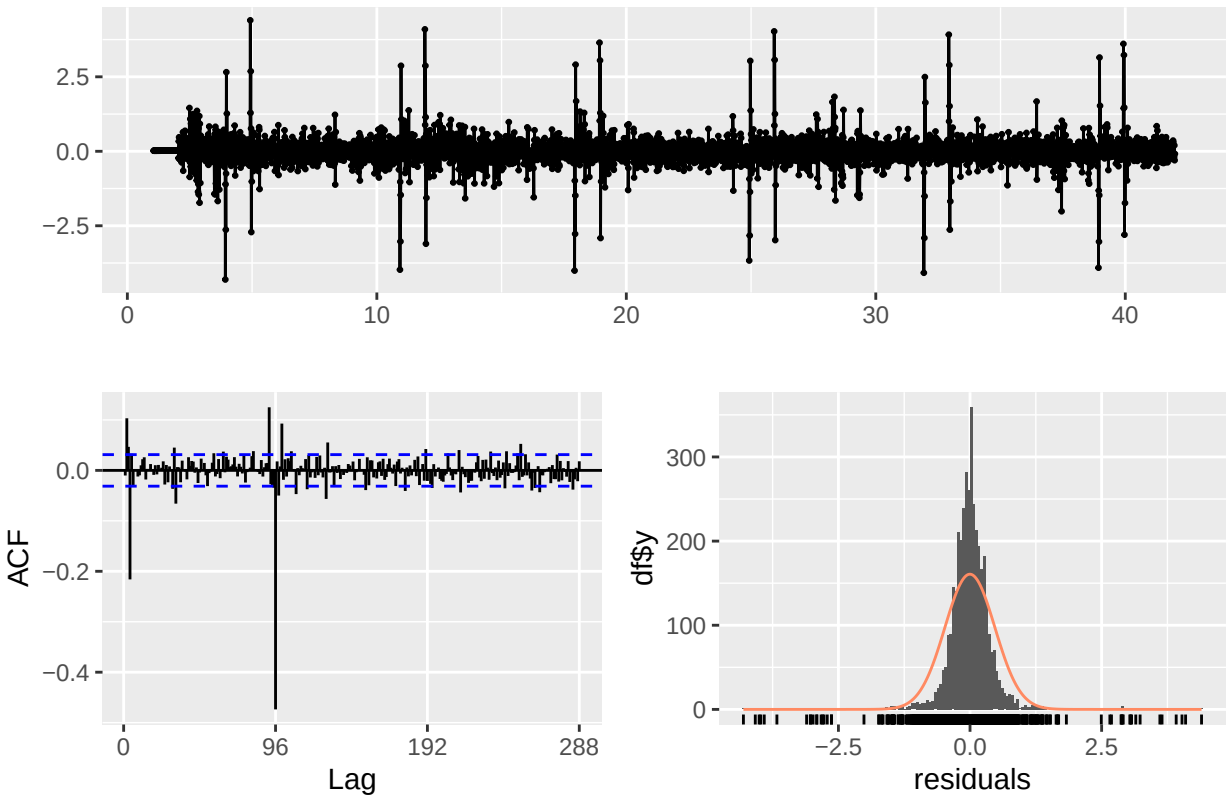
```
## The AIC of Auto SARIMA model with Covariate equal to: 28481.55
```

When we use auto arime function for the electricity consumption with covariate, the choses model is ARIMA(1,0,0)(0,1,0)[96]. The residuals of this model are noisy because they are autocorrelated and not independent (Ljung-Box test P-value «« < 0.05). Also, the RMSE of this model on a test set is not so good.

II-2-2-2- Forecasting Auto SARIMA Model with covariate and Cox-Box transformation

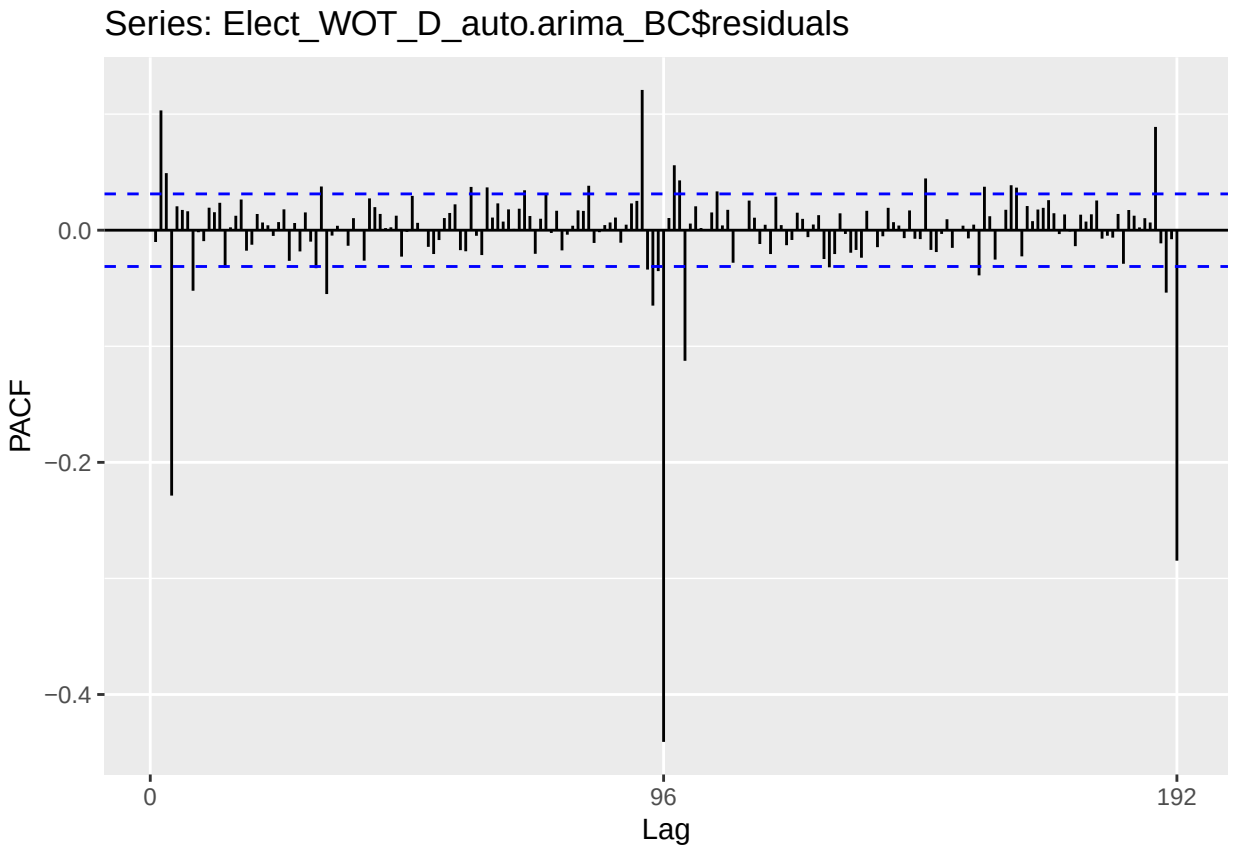
Let's try to add a Box-Cox transformation to the auta arima model to see if we get a better model.

```
Elect_WOT_D_auto.arima_BC <- auto.arima(Elect_D.ts.train[,"Power_(kW)"], xreg= Elect_D.ts.train[,"Temp_(C°)"], l
checkresiduals(Elect_WOT_D_auto.arima_BC)
```

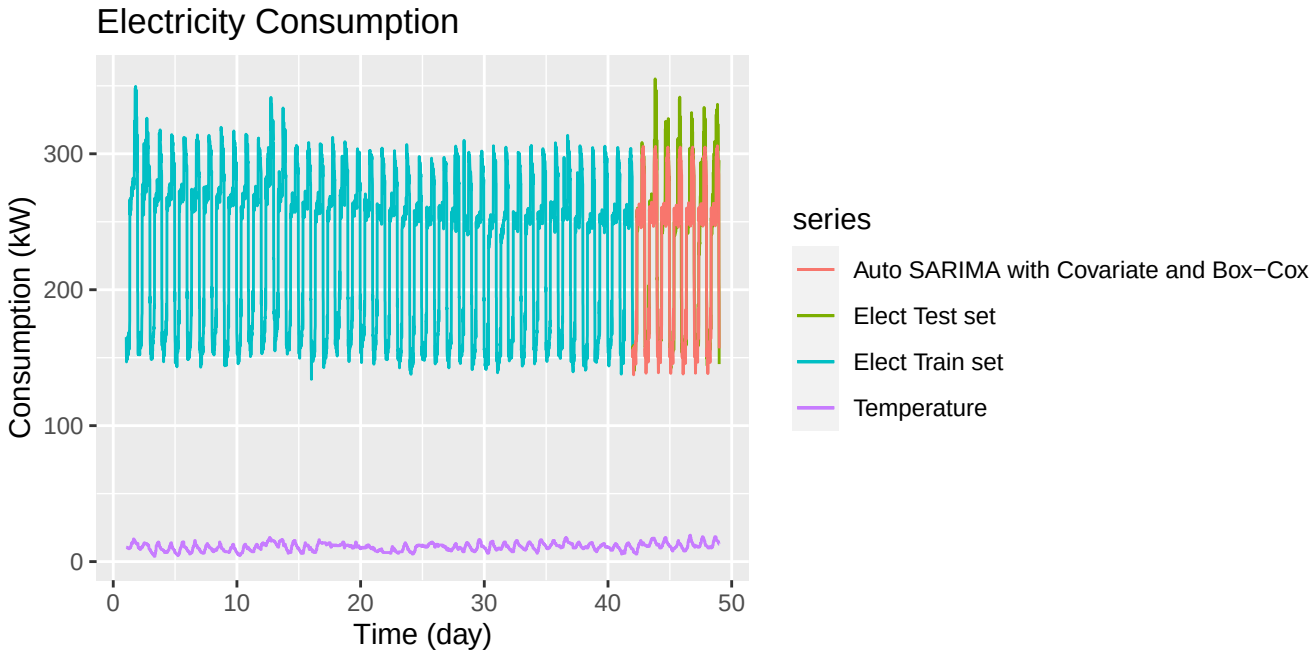



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0)(0,1,0)[96] errors
## Q* = 1530.7, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

```
ggPacf(Elect_WOT_D_auto.arima_BC$residuals)
```




```
Elect_WOT_D_auto.arima_BC_forcast <- forecast :: forecast(Elect_WOT_D_auto.arima_BC , xreg= Elect_D.ts.test[,"Te

autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_auto.arima_BC_forcast$mean, series='Auto SARIMA with Covariate and Box-Cox')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```



```
Elect_WOT_D_auto.arima_BC_RMSE <- sqrt(mean((Elect_WOT_D_auto.arima_BC_forcast$mean-Elect_D.ts.test[,"Power_(kW)
Elect_WOT_D_auto.arima_BC_AIC = Elect_WOT_D_auto.arima_BC$aicc

cat("The RMSE of Auto SARIMA model with Covariate and Box-Cox transformation equale to:", Elect_WOT_D_auto.arima
```

```
## The RMSE of Auto SARIMA model with Covariate and Box-Cox transformation equale to: 16.82584
```

```
cat("The AIC of Auto SARIMA model with Covariate and Box-Cox transformation equale to:", Elect_WOT_D_auto.arima_
```

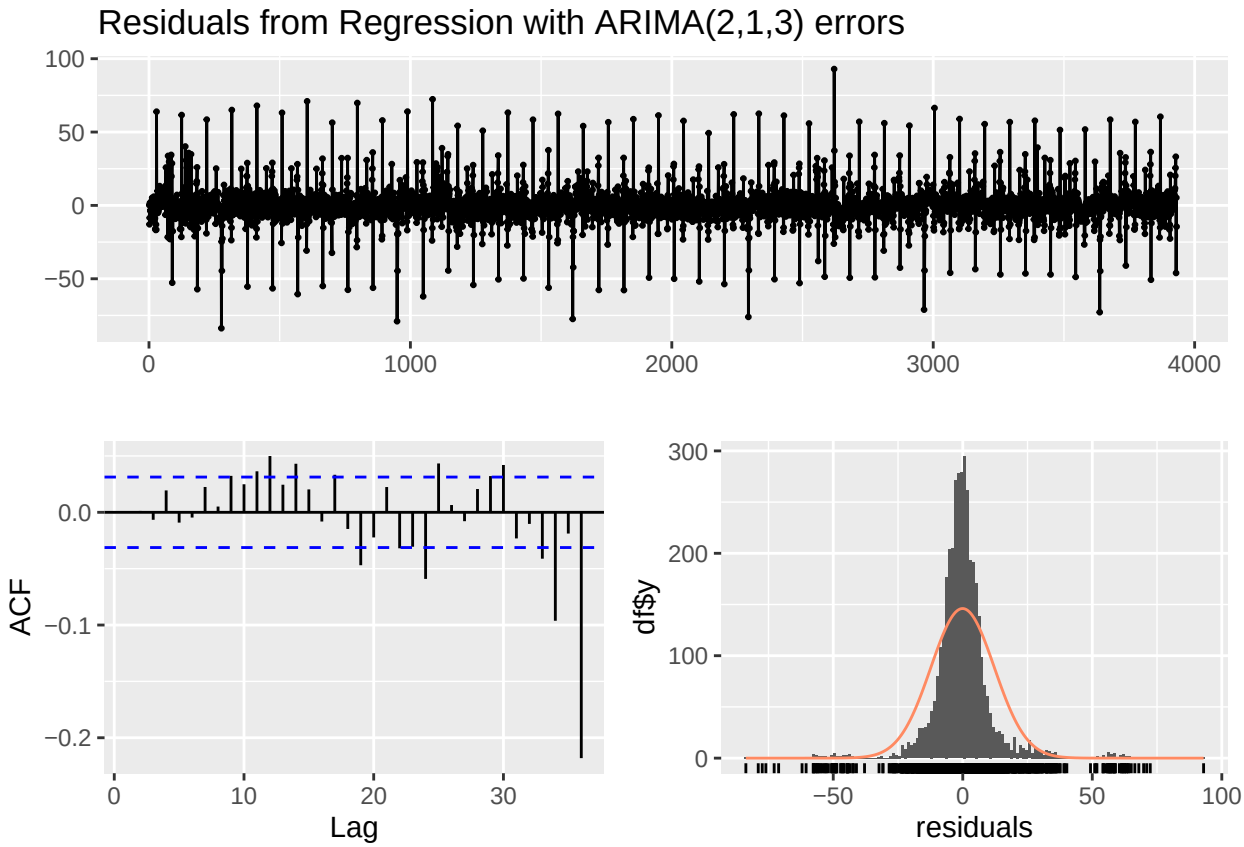
```
## The AIC of Auto SARIMA model with Covariate and Box-Cox transformation equale to: 5267.29
```

With a Box-Cox transformation in auto arima model, we don"t see any change comparing to the first auto arima model. We still have the problem of noisy residuals and RMSE not so good.

II-2-2-3- Forecasting Auto SARIMA Model with covariate and Fourier transformation

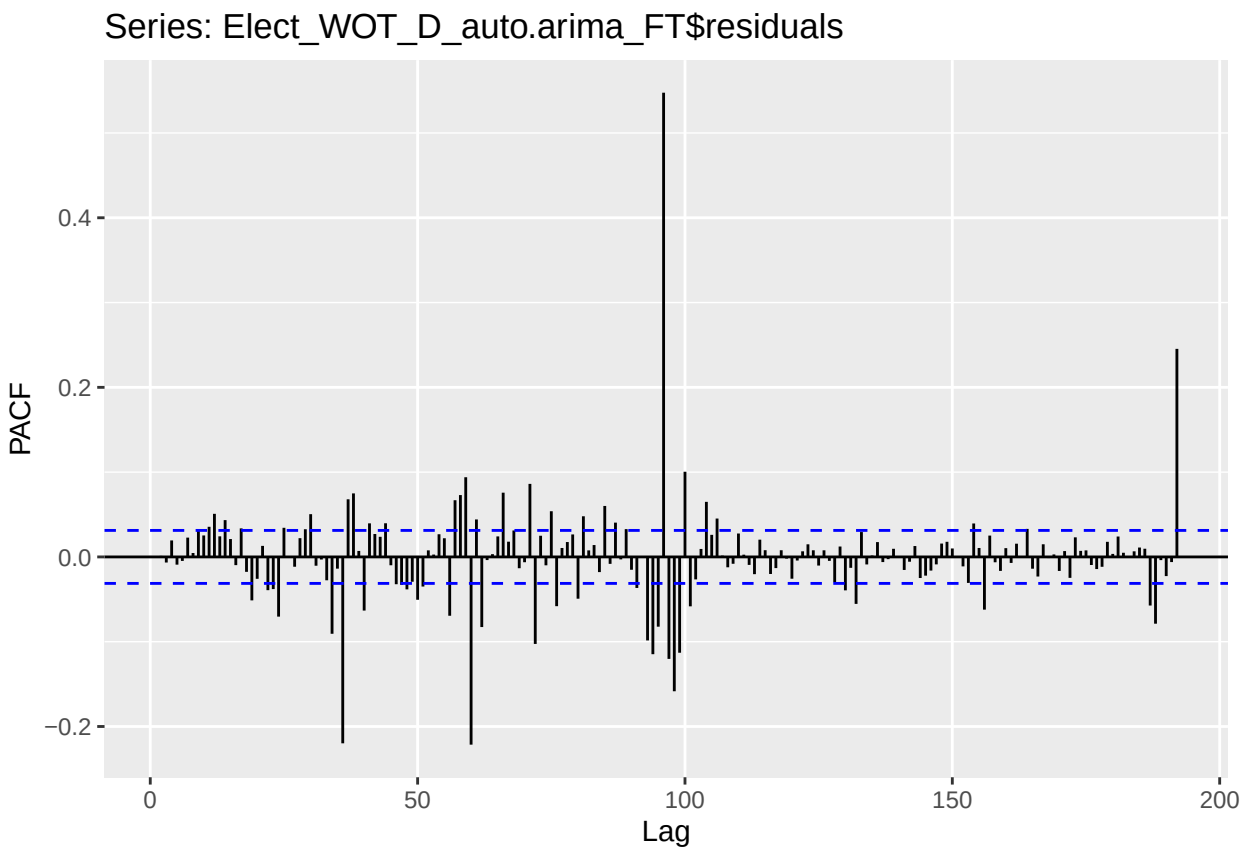
What will be the result if we add a Fourier transformation to auto arima model?

```
Elect_WOT_D_auto.arima_FT <- auto.arima(ts(Elect.train_TF[,"Power_(kW)"]),  xreg= cbind(Elect_D.ts.train[,"Temp_

checkresiduals(Elect_WOT_D_auto.arima_FT)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,3) errors
## Q* = 10.732, df = 5, p-value = 0.05696
##
## Model df: 5.   Total lags used: 10
```

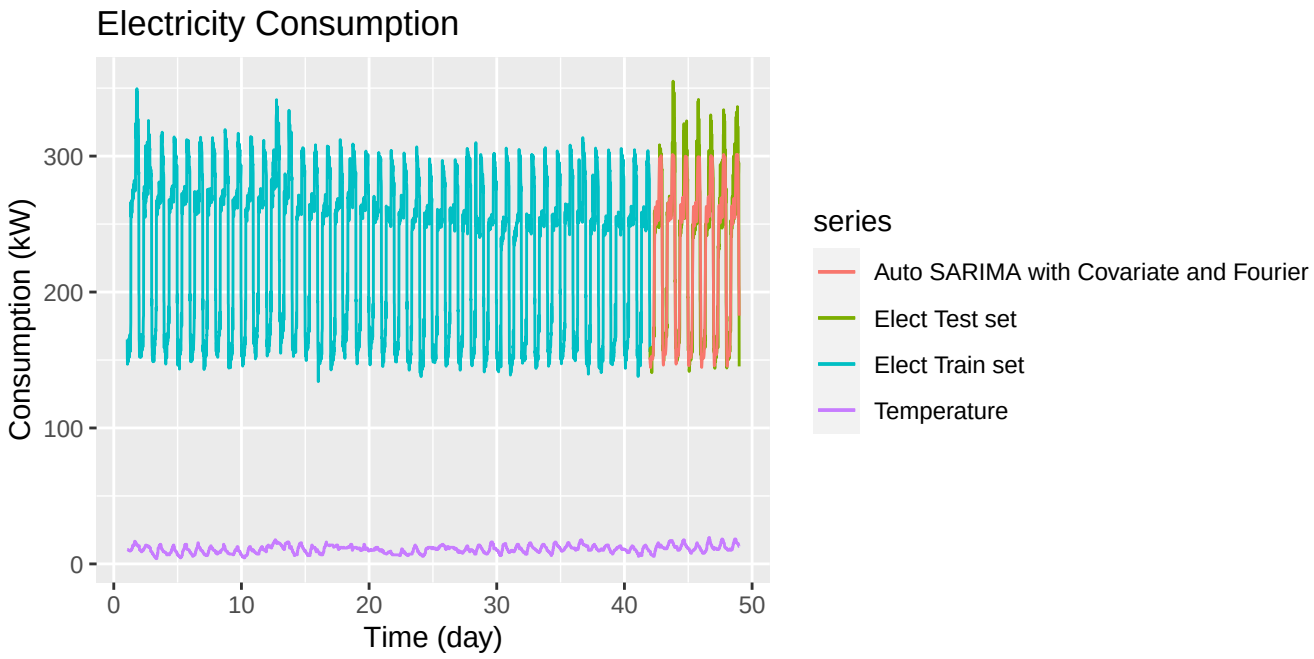
```
ggPacf(Elect_WOT_D_auto.arima_FT$residuals, lag.max = 192)
```




```
Elect_WOT_D_auto.arima_FT_forcast <- forecast :: forecast(Elect_WOT_D_auto.arima_FT, xreg= cbind(Elect_D.ts.test
```

```
## Warning in forecast.forecast_ARIMA(Elect_WOT_D_auto.arima_FT, xreg =
## cbind(Elect_D.ts.test[, : xreg contains different column names from the xreg
## used in training. Please check that the regressors are in the same order.
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(ts(Elect_WOT_D_auto.arima_FT_forcast$mean,frequency = 96, start = c(42, 1)), series='Auto SARIMA wit
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_auto.arima_FT_RMSE <- sqrt(mean(as.numeric(ts(Elect_WOT_D_auto.arima_FT_forcast$mean, frequency = 96
Elect_WOT_D_auto.arima_FT_AIC = Elect_WOT_D_auto.arima_FT$aicc

cat("The RMSE of Auto SARIMA model with Covariate and Fourier transformation equal to:", Elect_WOT_D_auto.arima_
```

```
## The RMSE of Auto SARIMA model with Covariate and Fourier transformation equal to: 18.56354
```

```
cat("The AIC of Auto SARIMA model with Covariate and Fourier transformation equal to:", Elect_WOT_D_auto.arima_F
```

```
## The AIC of Auto SARIMA model with Covariate and Fourier transformation equal to: 30881.87
```

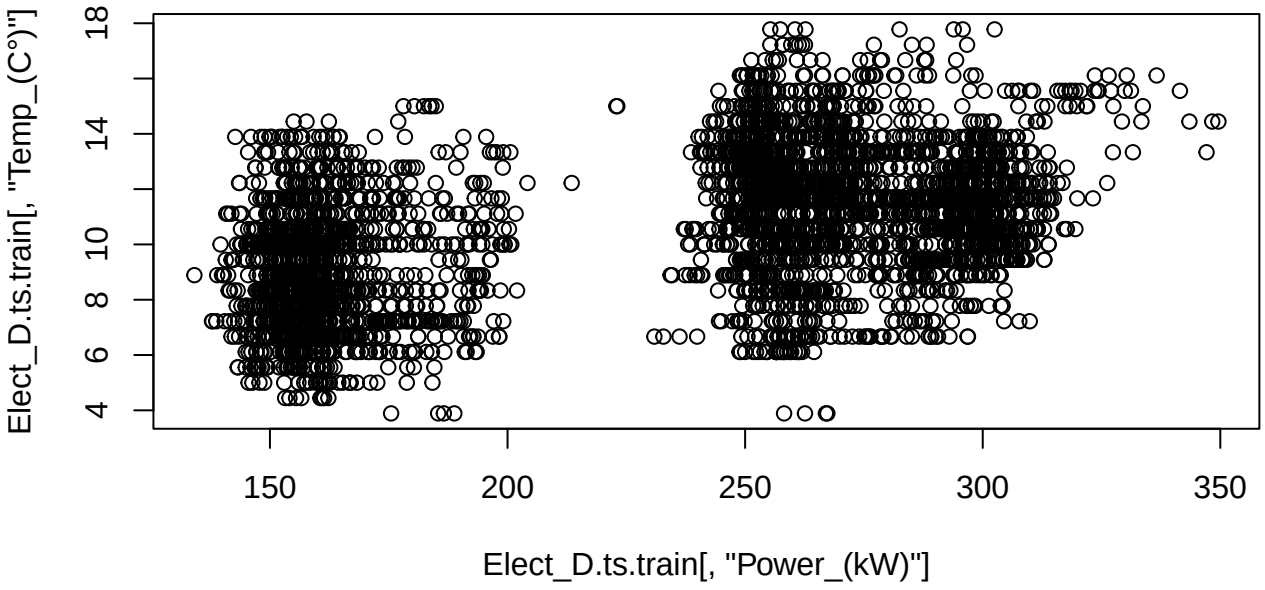
The Fourier transformation in auto arima model has a positive impact on the independence of the residuals because now we have significant Ljung-Box test P-value ($0.056 > 0.05$) but the residuals show a small autocorrelation in the ACF and PACF. But, with a Fourier transformation in auto arima model, the RMSE on test set is worse and the over fitting problem is more present here.

II-2-3- Forecasting Time Series Linear Model

In this section, we try to modeling a Manuel SARIMA.

For that, we start by looking for the relation between the electricity consumption and the temperature.

```
plot(Elect_D.ts.train[,"Power_(kW)"],Elect_D.ts.train[,"Temp_(C°)"])
```




```
data <- read_excel("./Elec-train.xlsx")
data <- data[2:3932,]
data$Times = as.POSIXct(data$Timestamp, format="%m/%d/%Y %H:%M")
data$weekday <- weekdays(data$Times)
data$hour <- as.numeric(format(data$Times, "%H"))
data <- transform(data,workday=ifelse(data$weekday %in% c("samedi","dimanche"),0,1))
data <- transform(data,day_night=ifelse(data$hour <=8 & data$hour>0 ,0,1))
```

```
Elect_WOT_D.train_tslm <- cbind(Consumption = Elect_D.ts.train[,"Power_(kW)"],
                                Square_Consumption = Elect_D.ts.train[,"Power_(kW)"]**2,
                                Temperature = Elect_D.ts.train[,"Temp_(C°)"],
                                Square_Temperature = Elect_D.ts.train[,"Temp_(C°)"]**2,
                                workday = data$workday,
                                Day_Night = data$day_night
                                )
```

Here, we use tslm function to to fit a linear model for this time series with variables like Square_Consumption, Temperature, Square_Temperature, workday, Day_Night, trend and seasonal.

The idea is to fit different models with differents variables and the select the better Linear model.

```
Elect_WOT_D_tslm_fit <- tslm(Consumption~Square_Consumption+Square_Temperature+trend+season, data=Elect_WOT_D.tr
```

```
Elect_WOT_D_tslm_fit_1 <- tslm(Consumption~Square_Consumption+Temperature+Square_Temperature+trend+season, data=
```

```
Elect_WOT_D_tslm_fit_2 <- tslm(Consumption~Square_Consumption+Temperature+Square_Temperature+workday+trend+seasc
```

```
Elect_WOT_D_tslm_fit_3 <- tslm(Consumption~Square_Consumption+Temperature+Square_Temperature+workday+Day_Night+t
```

We compare the models using CV function to get a comparaison of the principal

```
CV(Elect_WOT_D_tslm_fit)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    4.8597141 6193.0253638 6198.2995152 6820.6902764    0.9985656
```

```
CV(Elect_WOT_D_tslm_fit_1)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    4.8459025 6181.4914817 6186.8725211 6815.4330433    0.9985702
```

```
CV(Elect_WOT_D_tslm_fit_2)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    4.6104485 5984.6141241 5990.1031523 6624.8323349    0.9986404
```

```
CV(Elect_WOT_D_tslm_fit_3)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    4.6104485 5984.6141241 5990.1031523 6624.8323349    0.9986404
```

From the results above, we see that the "Elect_WOT_D_tslm_fit_2" and "Elect_WOT_D_tslm_fit_3" models are the best. We select "Elect_WOT_D_tslm_fit_2" because it has less variables (it is simpal comparing with the second one).

```
summary(Elect_WOT_D_tslm_fit_2)
```

```
##
## Call:
## tslm(formula = Consumption ~ Square_Consumption + Temperature +
##     Square_Temperature + workday + trend + season, data = Elect_WOT_D.train_tslm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.9544  -0.7307   0.0439   1.0767  10.5480
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.058e+02  6.216e-01 170.233  < 2e-16 ***
## Square_Consumption   2.001e-03  6.209e-06 322.243  < 2e-16 ***
## Temperature          3.104e-01  9.471e-02   3.277 0.001057 **
## Square_Temperature -1.579e-02  4.404e-03  -3.586 0.000340 ***
## workday              1.080e+00  7.661e-02  14.098  < 2e-16 ***
## trend                2.550e-05  3.382e-05   0.754 0.450976
## season2             -3.722e-01  4.726e-01  -0.788 0.430986
## season3             -2.730e+00  4.728e-01  -5.773 8.39e-09 ***
## season4             -2.958e-01  4.726e-01  -0.626 0.531433
## season5              8.739e-01  4.727e-01   1.849 0.064554 .
## season6              9.852e-01  4.699e-01   2.097 0.036076 *
## season7             -2.654e+00  4.700e-01  -5.646 1.76e-08 ***
## season8             -2.746e+00  4.700e-01  -5.841 5.61e-09 ***
## season9             -1.404e+00  4.699e-01  -2.987 0.002835 **
## season10            -1.712e+00  4.701e-01  -3.642 0.000274 ***
## season11            -6.491e-01  4.700e-01  -1.381 0.167347
## season12            -9.591e-01  4.700e-01  -2.040 0.041369 *
## season13            -3.969e-01  4.700e-01  -0.844 0.398466
## season14            -2.256e+00  4.703e-01  -4.798 1.66e-06 ***
## season15            -2.849e+00  4.704e-01  -6.056 1.53e-09 ***
## season16            -6.997e-01  4.702e-01  -1.488 0.136797
## season17            -9.936e-02  4.702e-01  -0.211 0.832658
## season18            -5.411e-01  4.706e-01  -1.150 0.250309
## season19            -8.627e-01  4.706e-01  -1.833 0.066852 .
## season20            -2.478e-01  4.706e-01  -0.527 0.598448
## season21            -3.852e-01  4.706e-01  -0.819 0.413046
## season22            -8.459e-02  4.708e-01  -0.180 0.857423
```

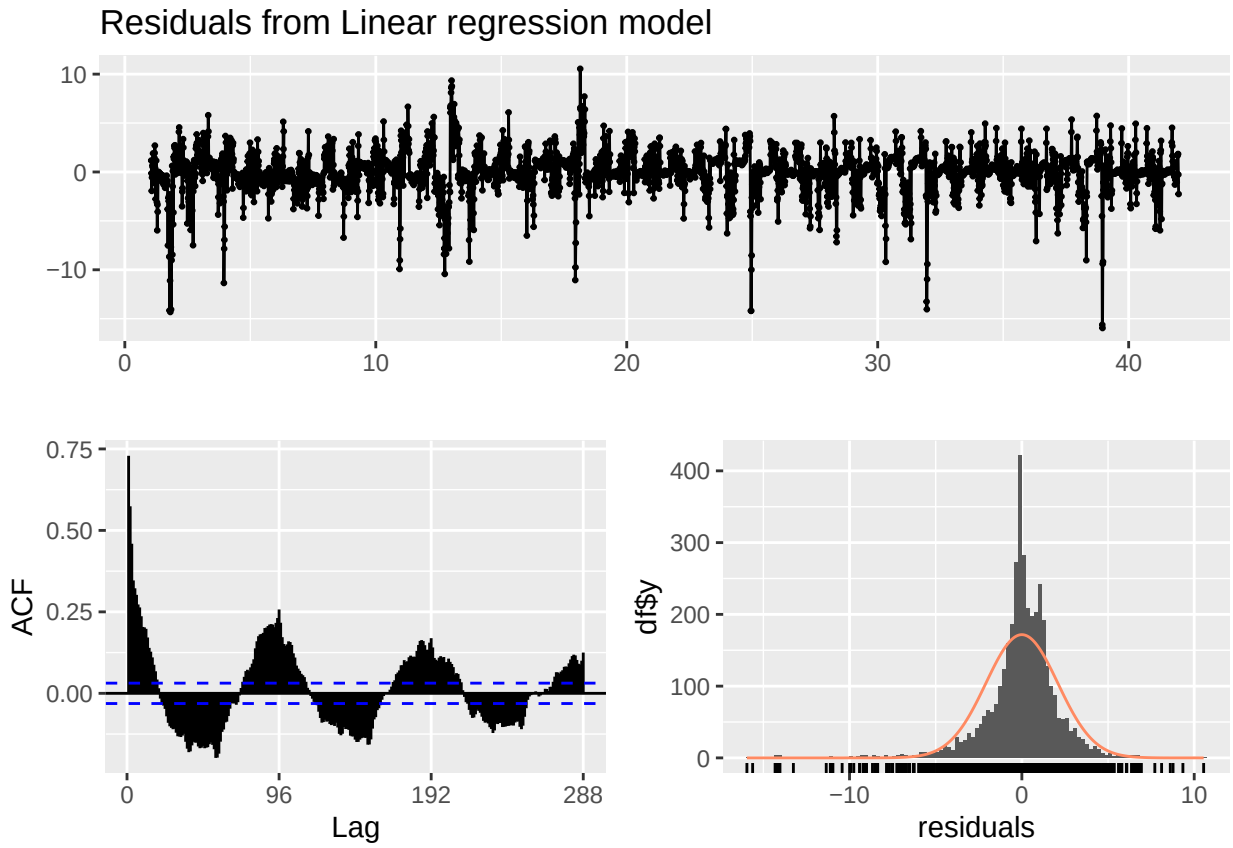
```
## season23           1.686e-01  4.708e-01   0.358 0.720245
## season24           9.560e-01  4.709e-01   2.030 0.042403 *
## season25           7.489e-01  4.709e-01   1.590 0.111820
## season26           1.971e+00  4.710e-01   4.186 2.90e-05 ***
## season27           4.240e+00  4.716e-01   8.990  < 2e-16 ***
## season28           4.787e+00  4.719e-01  10.143  < 2e-16 ***
## season29           4.941e+00  4.720e-01  10.468  < 2e-16 ***
## season30           6.590e+00  4.730e-01  13.933  < 2e-16 ***
## season31           6.220e+00  4.729e-01  13.152  < 2e-16 ***
## season32           4.895e+00  4.720e-01  10.371  < 2e-16 ***
## season33           6.007e+00  4.726e-01  12.711  < 2e-16 ***
## season34           1.646e+01  5.426e-01  30.337  < 2e-16 ***
## season35           1.647e+01  5.392e-01  30.550  < 2e-16 ***
## season36           1.656e+01  5.348e-01  30.961  < 2e-16 ***
## season37           1.658e+01  5.343e-01  31.033  < 2e-16 ***
## season38           1.650e+01  5.386e-01  30.634  < 2e-16 ***
## season39           1.668e+01  5.308e-01  31.414  < 2e-16 ***
## season40           1.662e+01  5.344e-01  31.105  < 2e-16 ***
## season41           1.659e+01  5.357e-01  30.968  < 2e-16 ***
## season42           1.668e+01  5.304e-01  31.454  < 2e-16 ***
## season43           1.670e+01  5.316e-01  31.424  < 2e-16 ***
## season44           1.656e+01  5.342e-01  31.004  < 2e-16 ***
## season45           1.664e+01  5.344e-01  31.131  < 2e-16 ***
## season46           1.652e+01  5.386e-01  30.672  < 2e-16 ***
## season47           1.655e+01  5.357e-01  30.892  < 2e-16 ***
## season48           1.642e+01  5.368e-01  30.589  < 2e-16 ***
## season49           1.646e+01  5.379e-01  30.598  < 2e-16 ***
## season50           1.651e+01  5.382e-01  30.681  < 2e-16 ***
## season51           1.632e+01  5.437e-01  30.018  < 2e-16 ***
## season52           1.644e+01  5.422e-01  30.317  < 2e-16 ***
## season53           1.646e+01  5.396e-01  30.495  < 2e-16 ***
## season54           1.633e+01  5.433e-01  30.047  < 2e-16 ***
## season55           1.640e+01  5.437e-01  30.160  < 2e-16 ***
## season56           1.642e+01  5.426e-01  30.258  < 2e-16 ***
## season57           1.649e+01  5.415e-01  30.449  < 2e-16 ***
## season58           1.654e+01  5.429e-01  30.461  < 2e-16 ***
## season59           1.651e+01  5.433e-01  30.383  < 2e-16 ***
## season60           1.652e+01  5.428e-01  30.431  < 2e-16 ***
## season61           1.642e+01  5.426e-01  30.253  < 2e-16 ***
## season62           1.648e+01  5.429e-01  30.360  < 2e-16 ***
## season63           1.648e+01  5.419e-01  30.407  < 2e-16 ***
## season64           1.655e+01  5.420e-01  30.529  < 2e-16 ***
## season65           1.647e+01  5.417e-01  30.399  < 2e-16 ***
## season66           1.643e+01  5.410e-01  30.365  < 2e-16 ***
## season67           1.636e+01  5.419e-01  30.193  < 2e-16 ***
## season68           1.654e+01  5.378e-01  30.760  < 2e-16 ***
## season69           1.641e+01  5.380e-01  30.499  < 2e-16 ***
## season70           1.355e+01  5.753e-01  23.560  < 2e-16 ***
## season71           1.205e+01  6.031e-01  19.977  < 2e-16 ***
## season72           1.058e+01  6.300e-01  16.797  < 2e-16 ***
## season73           1.067e+01  6.288e-01  16.971  < 2e-16 ***
## season74           1.138e+01  6.218e-01  18.293  < 2e-16 ***
## season75           1.159e+01  6.187e-01  18.739  < 2e-16 ***
## season76           1.167e+01  6.170e-01  18.908  < 2e-16 ***
## season77           1.164e+01  6.178e-01  18.850  < 2e-16 ***
## season78           1.046e+01  6.315e-01  16.564  < 2e-16 ***
## season79           1.131e+01  6.215e-01  18.190  < 2e-16 ***
## season80           1.169e+01  6.183e-01  18.903  < 2e-16 ***
## season81           1.194e+01  6.147e-01  19.425  < 2e-16 ***
## season82           1.189e+01  6.149e-01  19.329  < 2e-16 ***
## season83           1.244e+01  6.090e-01  20.429  < 2e-16 ***
```

```
## season84            1.240e+01  6.090e-01  20.356  < 2e-16 ***
## season85            1.263e+01  6.064e-01  20.825  < 2e-16 ***
## season86            1.306e+01  6.015e-01  21.720  < 2e-16 ***
## season87            1.317e+01  5.992e-01  21.979  < 2e-16 ***
## season88            1.337e+01  5.967e-01  22.414  < 2e-16 ***
## season89            1.374e+01  5.918e-01  23.209  < 2e-16 ***
## season90            1.379e+01  5.614e-01  24.566  < 2e-16 ***
## season91            1.388e+01  5.590e-01  24.825  < 2e-16 ***
## season92            1.240e+01  5.515e-01  22.492  < 2e-16 ***
## season93            1.241e+01  5.532e-01  22.431  < 2e-16 ***
## season94            8.894e+00  4.744e-01  18.750  < 2e-16 ***
## season95            9.430e+00  4.750e-01  19.853  < 2e-16 ***
## season96            1.014e+00  4.697e-01   2.159 0.030896 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.113 on 3830 degrees of freedom
## Multiple R-squared:  0.9987, Adjusted R-squared:  0.9986
## F-statistic: 2.887e+04 on 100 and 3830 DF,  p-value: < 2.2e-16
```

We see here that all used variables are significante.

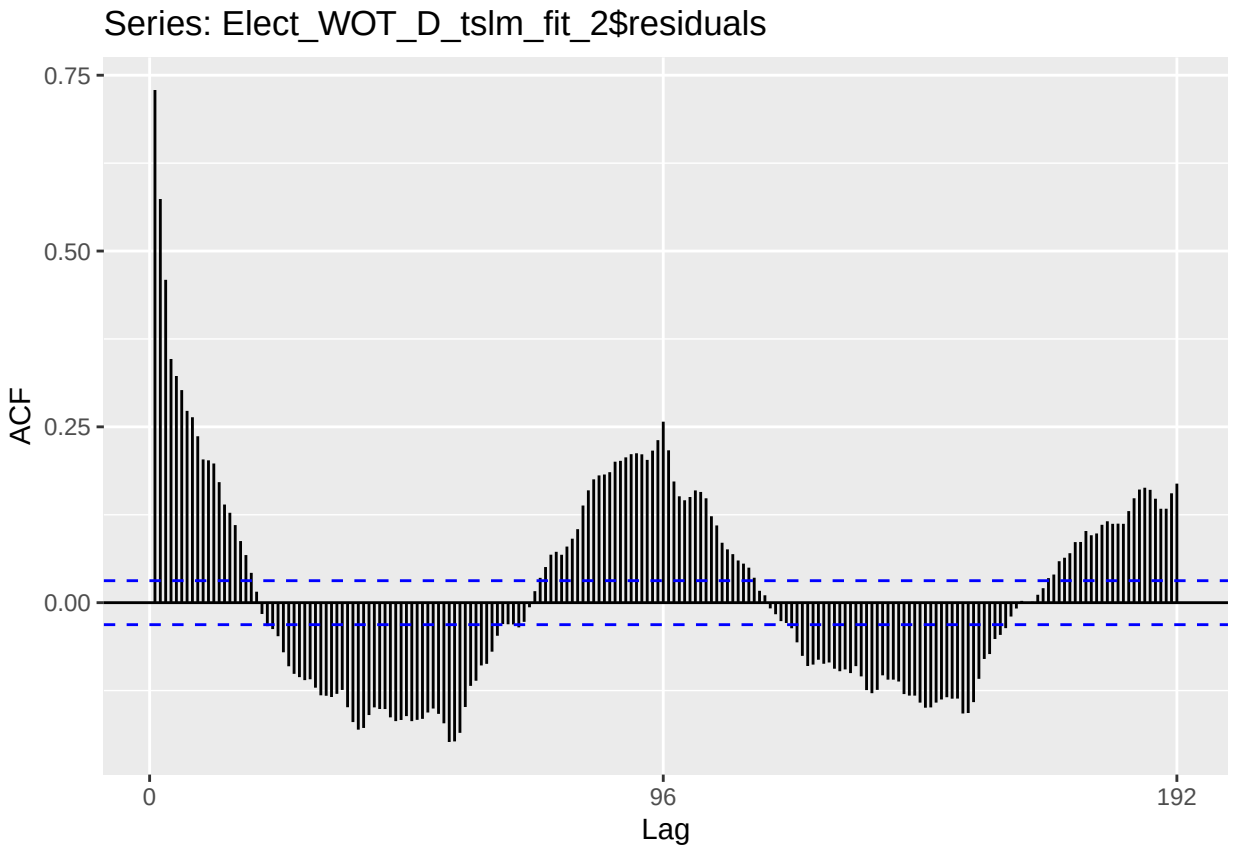
now, we take a look on the residuals of this linear model.

```
checkresiduals(Elect_WOT_D_tslm_fit_2)
```

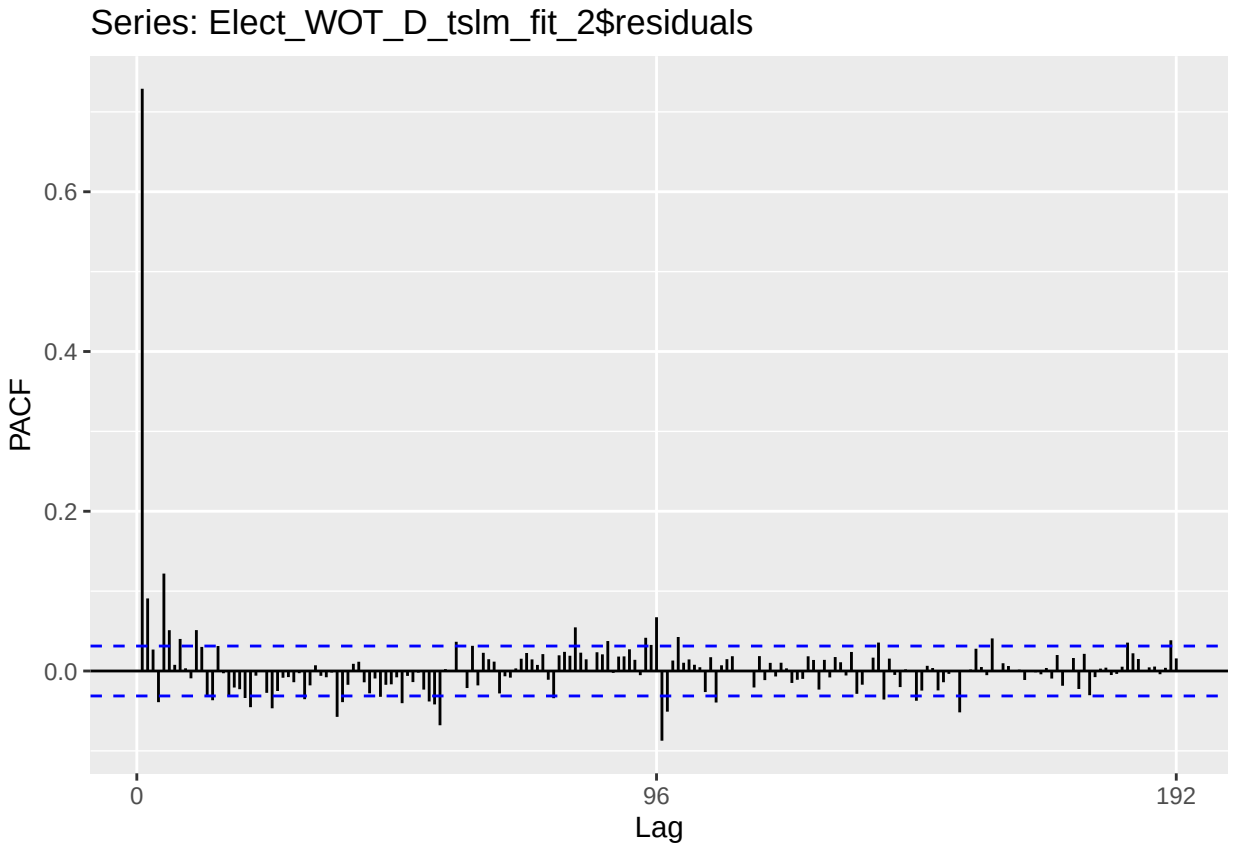



```
##
##  Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 2328.1, df = 192, p-value < 2.2e-16
```

```
par(mfrow=c(1,2))
ggAcf(Elect_WOT_D_tslm_fit_2$residuals)
```




```
ggPacf(Elect_WOT_D_tslm_fit_2$residuals)
```

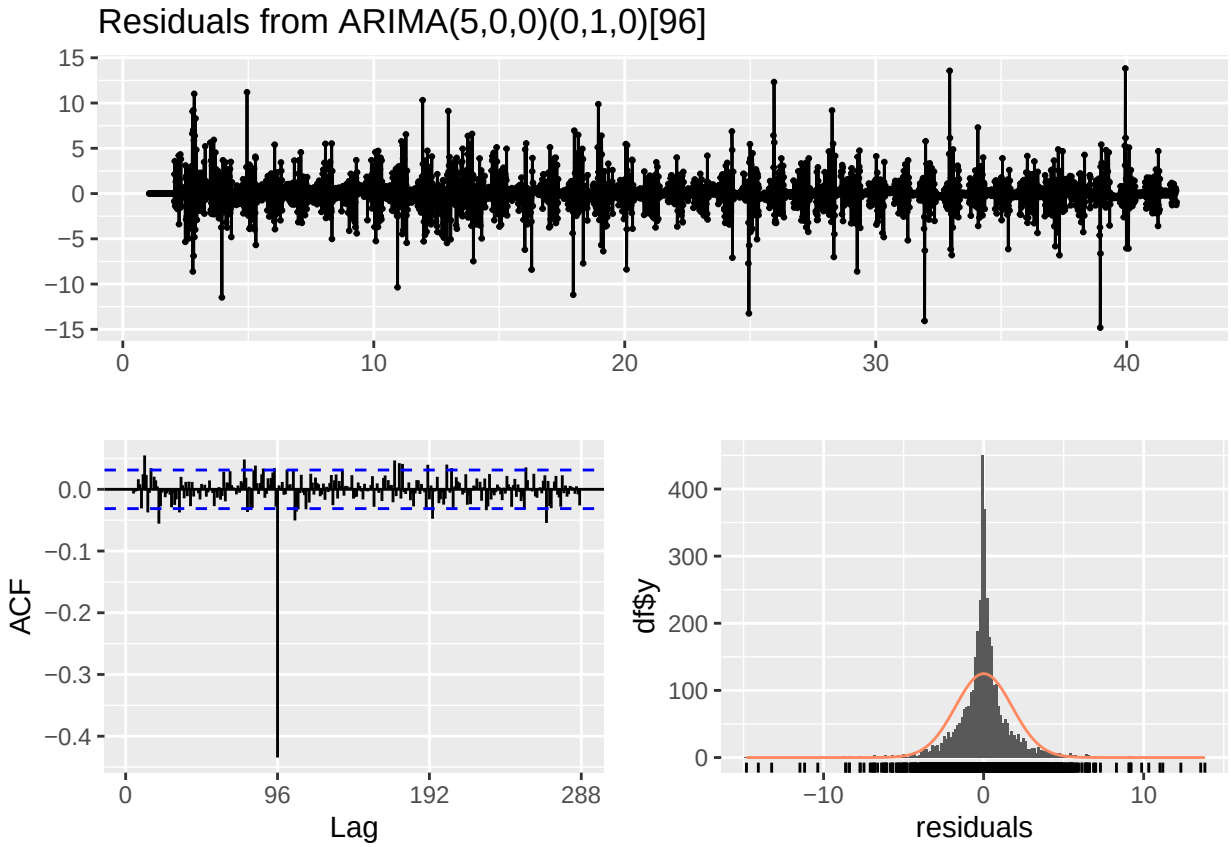


From the ACF, we

see a seasonalit and from the PACF we wee a significant PACF at lag = 5.

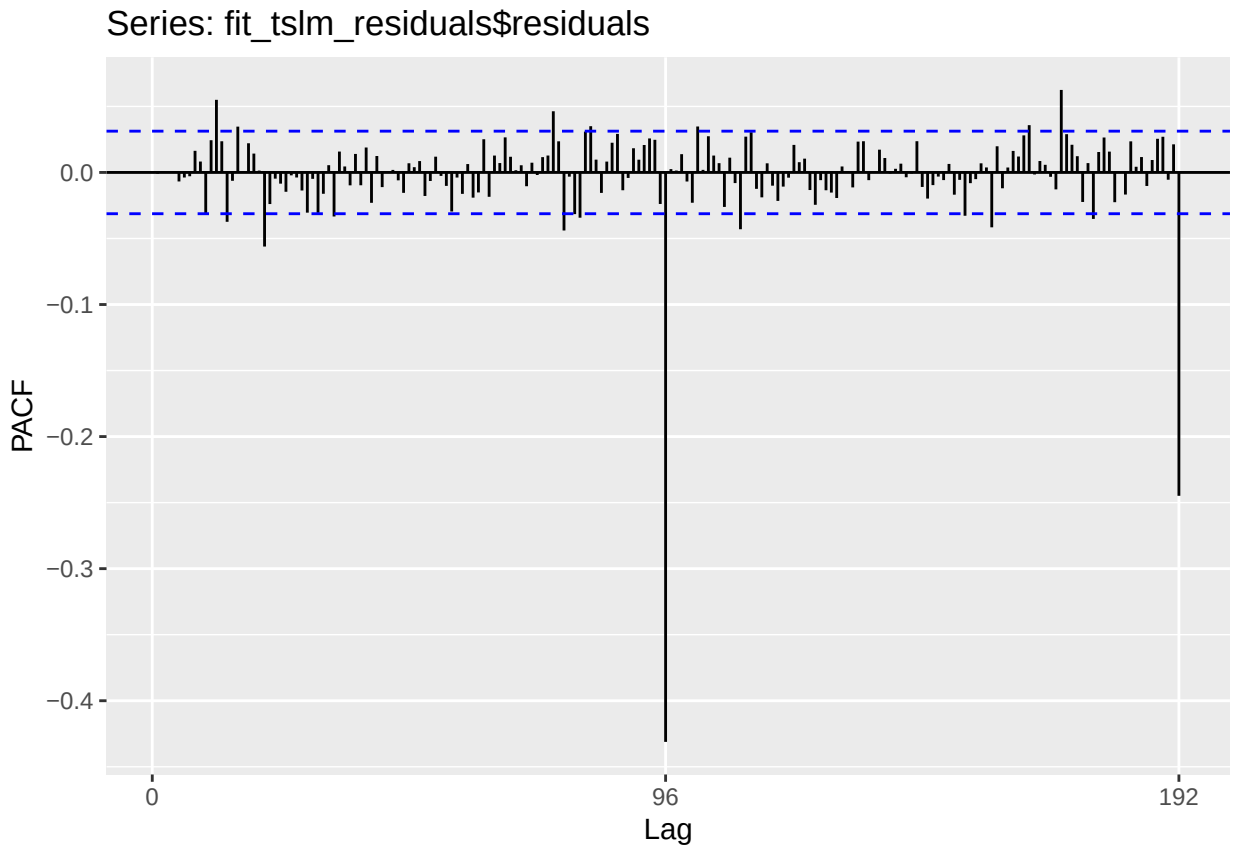
So, we can start to fit these residuals with (5,0,0)(0,1,0) model.

```
fit_tslm_residuals=Arima(Elect_WOT_D_tslm_fit_2$residuals,order=c(5,0,0), seasonal = c(0,1,0))
checkresiduals(fit_tslm_residuals)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,0)[96]
## Q* = 1036.6, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

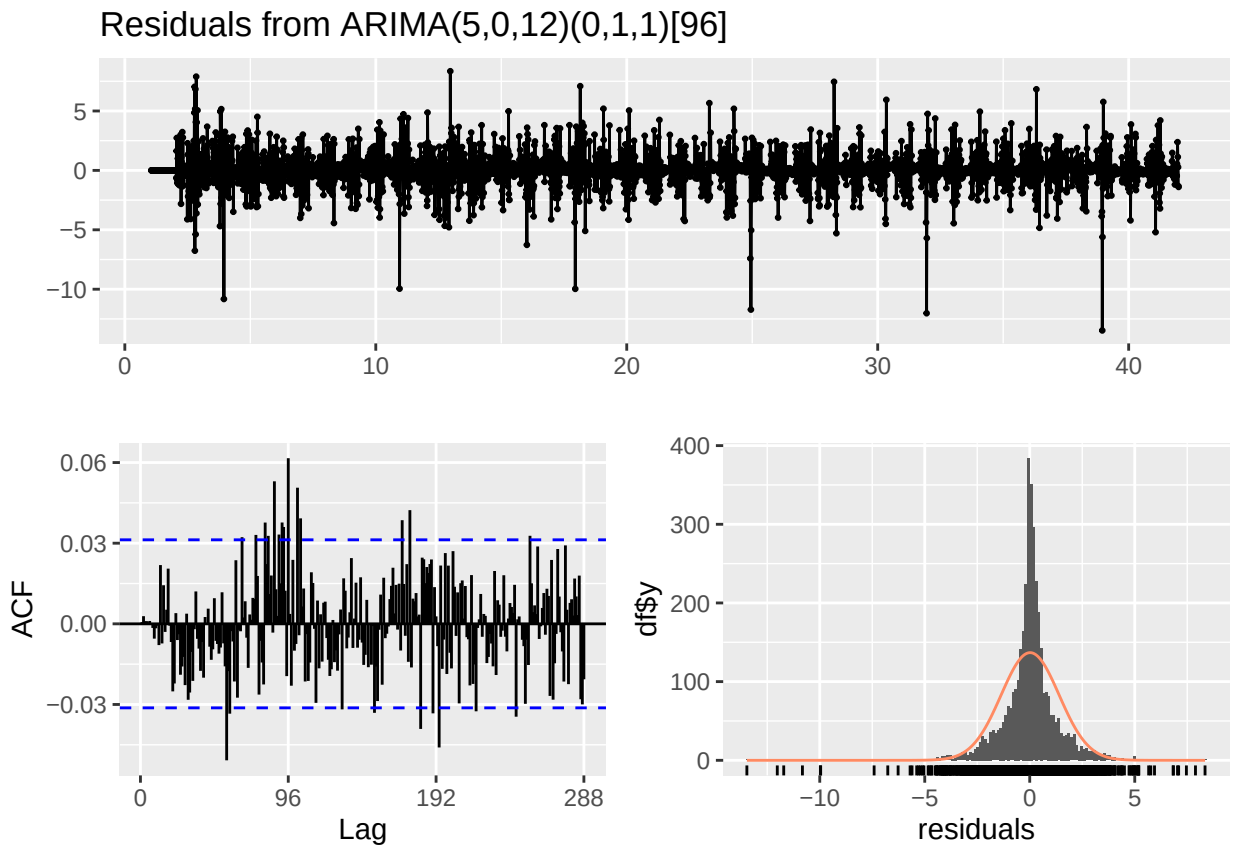
```
ggPacf(fit_tslm_residuals$residuals)
```



7.441e-05 fit_tslm_residua

seasonal = c(0,1,1))

```
fit_tslm_residuals_1=Arima(Elect_WOT_D_tslm_fit_2$residuals,order=c(5,0,12), seasonal = c(0,1,1))
checkresiduals(fit_tslm_residuals_1)
```

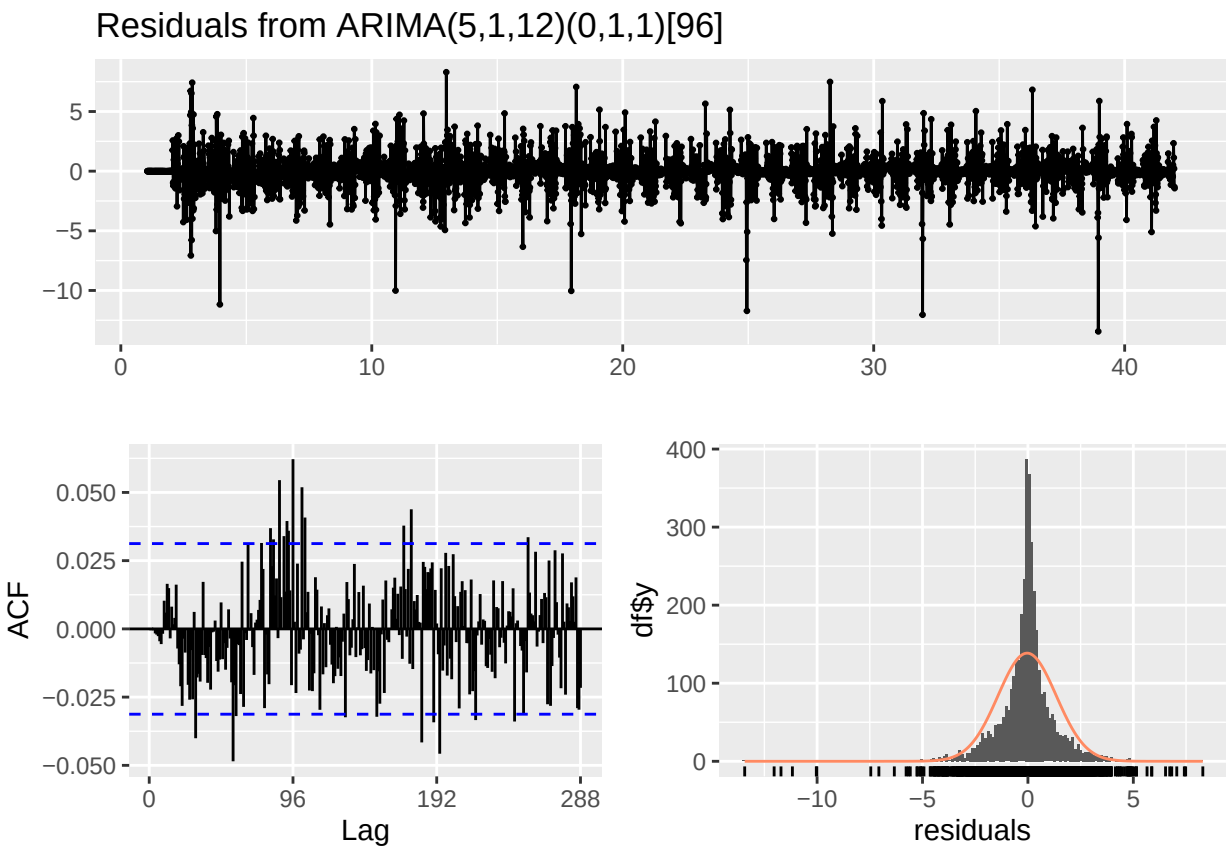


##

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,12)(0,1,1)[96]
## Q* = 253.83, df = 174, p-value = 7.441e-05
##
## Model df: 18.   Total lags used: 192
```

Form the ACF we can see that a pic at lag 96 still present and from the PACF, we see a significant PACF at lag 12. We can try this model (5,1,12)(0,1,1) to fit these residuals

```
fit_tslm_residuals_2=Arima(Elect_WOT_D_tslm_fit_2$residuals,order=c(5,1,12), seasonal = c(0,1,1))
checkresiduals(fit_tslm_residuals_2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,1,12)(0,1,1)[96]
## Q* = 265.6, df = 174, p-value = 9.417e-06
##
## Model df: 18.   Total lags used: 192
```

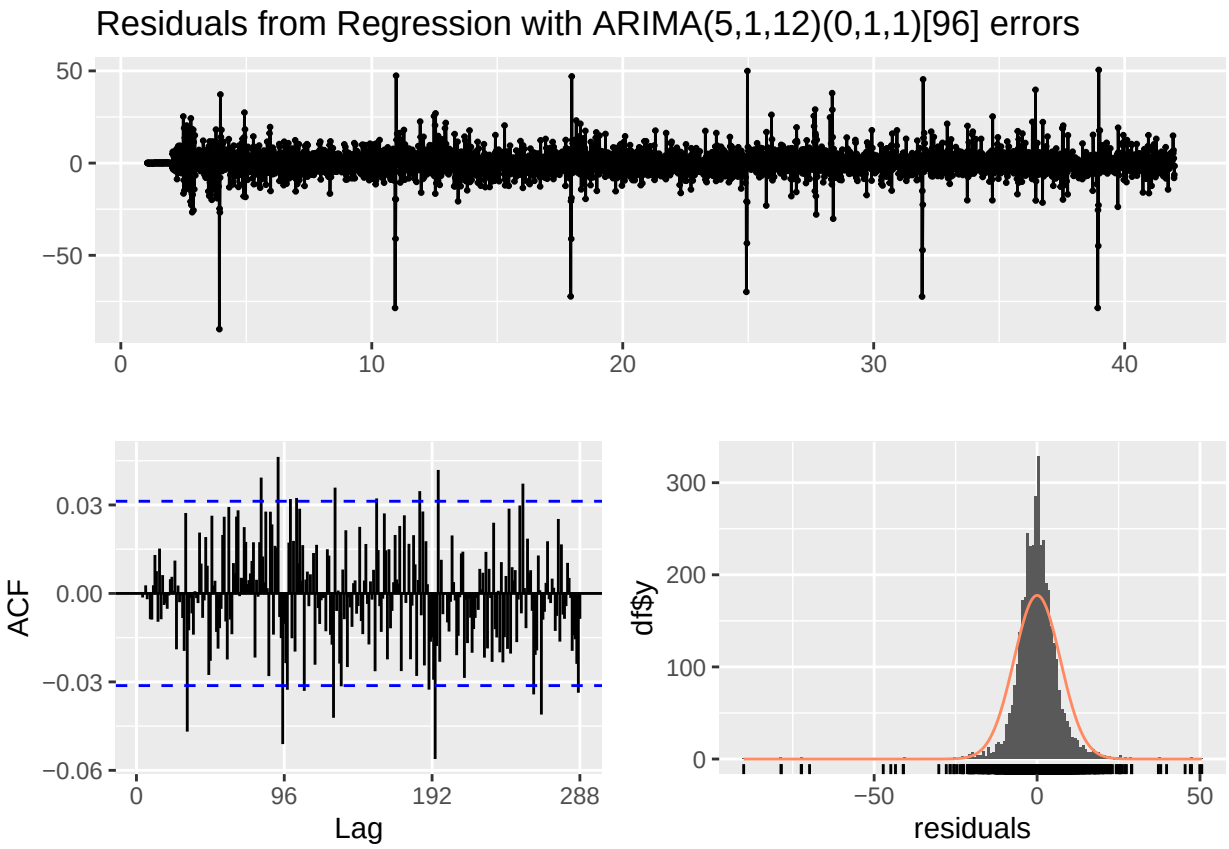
Now we can see less autocorrelation and the Ljung-Box test P-value is bigger than befor. It's not perfect but we can start to fit a manuel SARIMA for the electricity consumption time series with covariate with the model (5,1,12)(0,1,1).

II-2-4- Forecasting Manuel SARIMA Model with covariate

II-2-4-1- Forecasting Manuel SARIMA Model with covariate

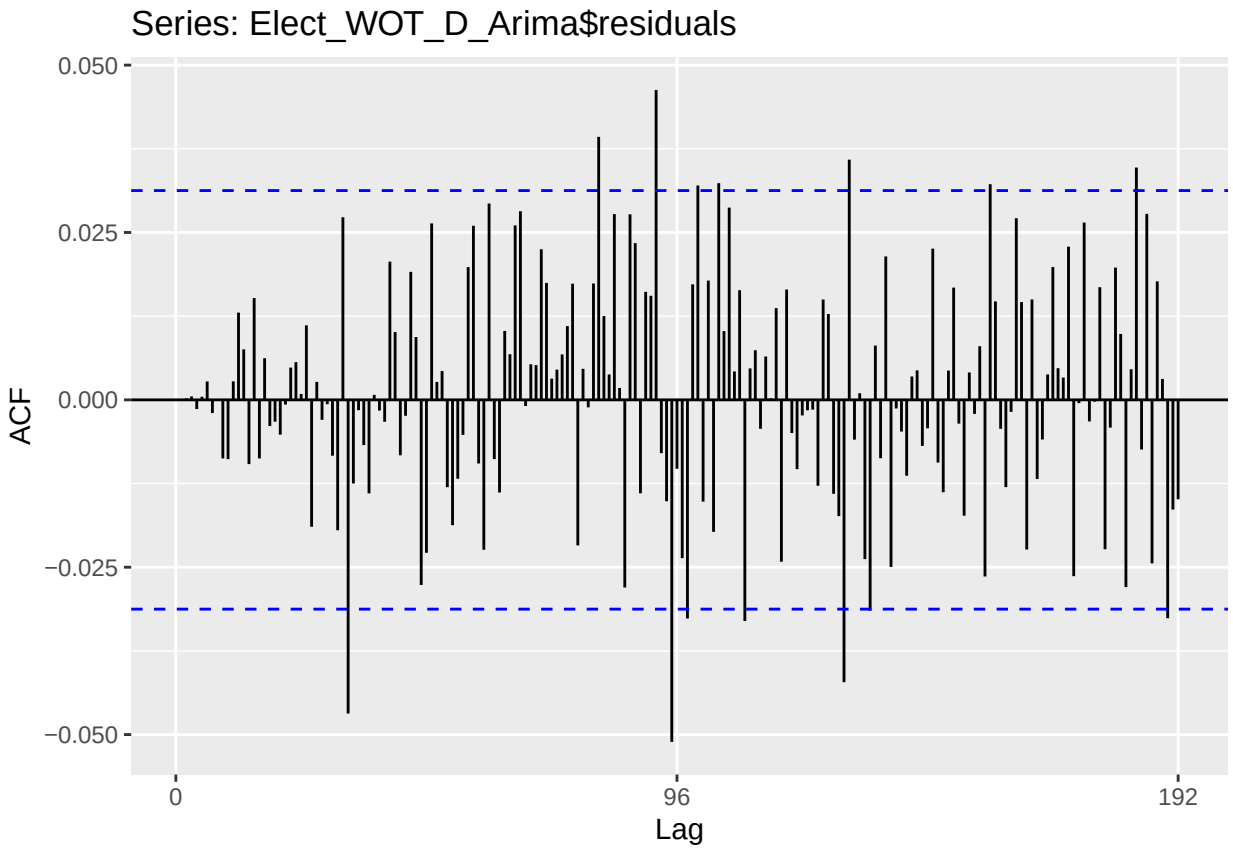
```
Elect_WOT_D_Arima <- Arima(Elect_D.ts.train[,"Power_(kW)"], order=c(5,1,12),  seasonal=c(0,1,1), xreg= Elect_D.t
```

```
checkresiduals(Elect_WOT_D_Arima)
```

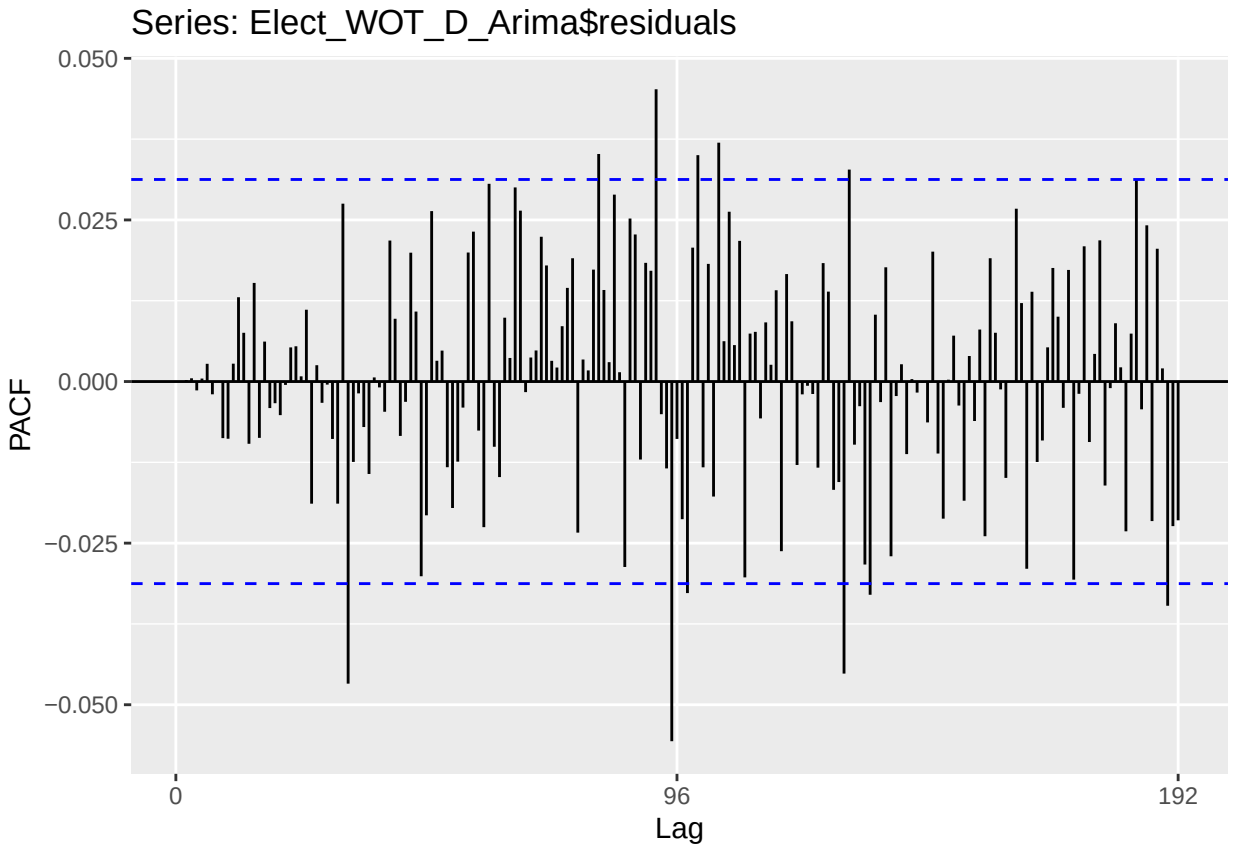



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,12)(0,1,1)[96] errors
## Q* = 227.29, df = 174, p-value = 0.004109
##
## Model df: 18.   Total lags used: 192
```

```
ggAcf(Elect_WOT_D_Arima$residuals)
```



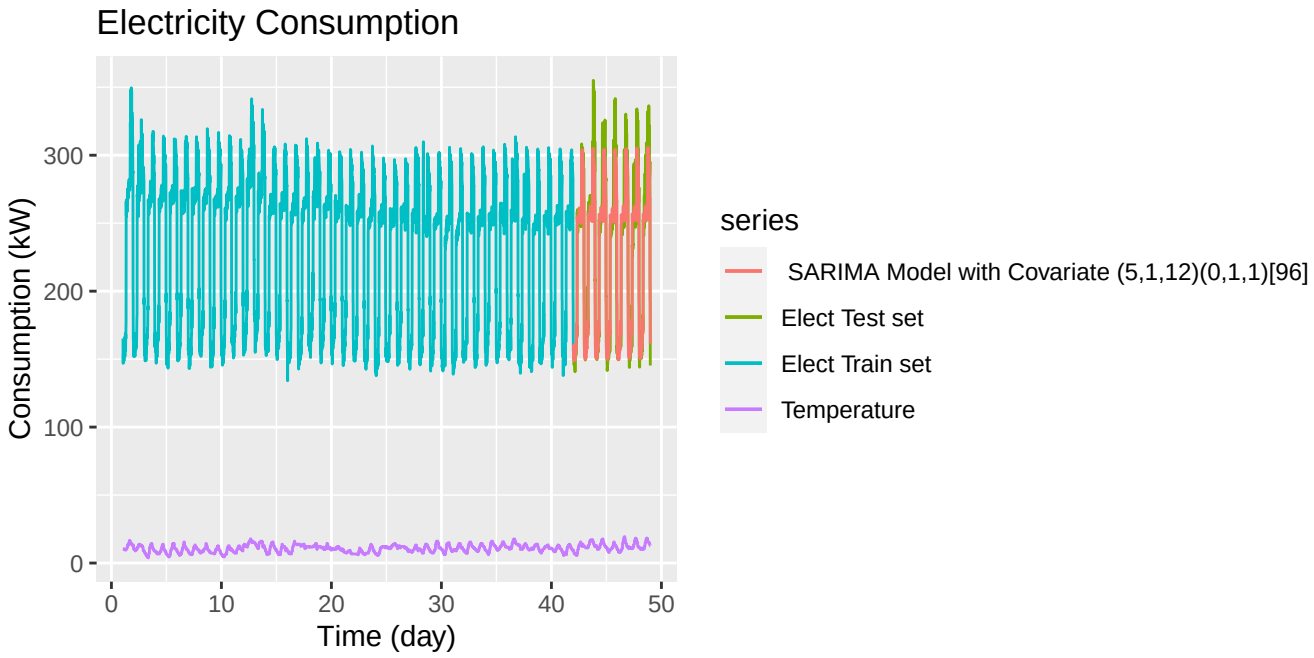

```
ggPacf(Elect_WOT_D_Arima$residuals)
```



The ACF and PACF whow less autocorrelation and Ljung-Box test P-value is better (0.004). This model is not perfect but it seems better than the others models.

Now, we look at the RMSE.

```
Elect_WOT_D_Arima_forcast <- forecast :: forecast(Elect_WOT_D_Arima , xreg= Elect_D.ts.test[,"Temp_(C°)"] , h=67

autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_Arima_forcast$mean, series=' SARIMA Model with Covariate (5,1,12)(0,1,1)[96]')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_Arima_RMSE <- sqrt(mean((Elect_WOT_D_Arima_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOT_D_Arima_AIC = Elect_WOT_D_Arima$aicc

cat("The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] equal to:", Elect_WOT_D_Arima_RMSE, "\n")
```

```
## The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] equal to: 15.58481
```

```
cat("The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] equal to:", Elect_WOT_D_Arima_AIC, "\n")
```

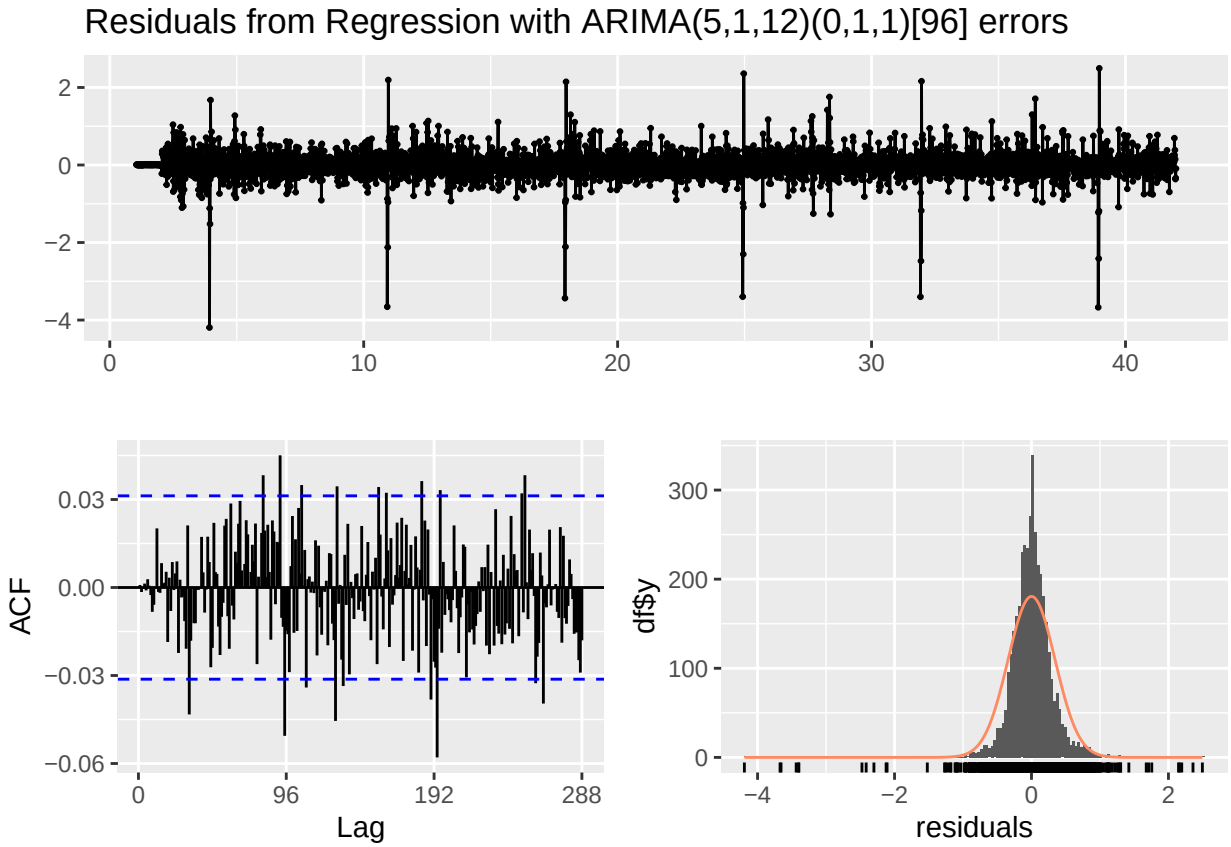
```
## The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] equal to: 26206.09
```

The RMSE on a test set is better then before.

II-2-4-2- Forecasting Manuel SARIMA Model with covariate and Cox-Box transformation

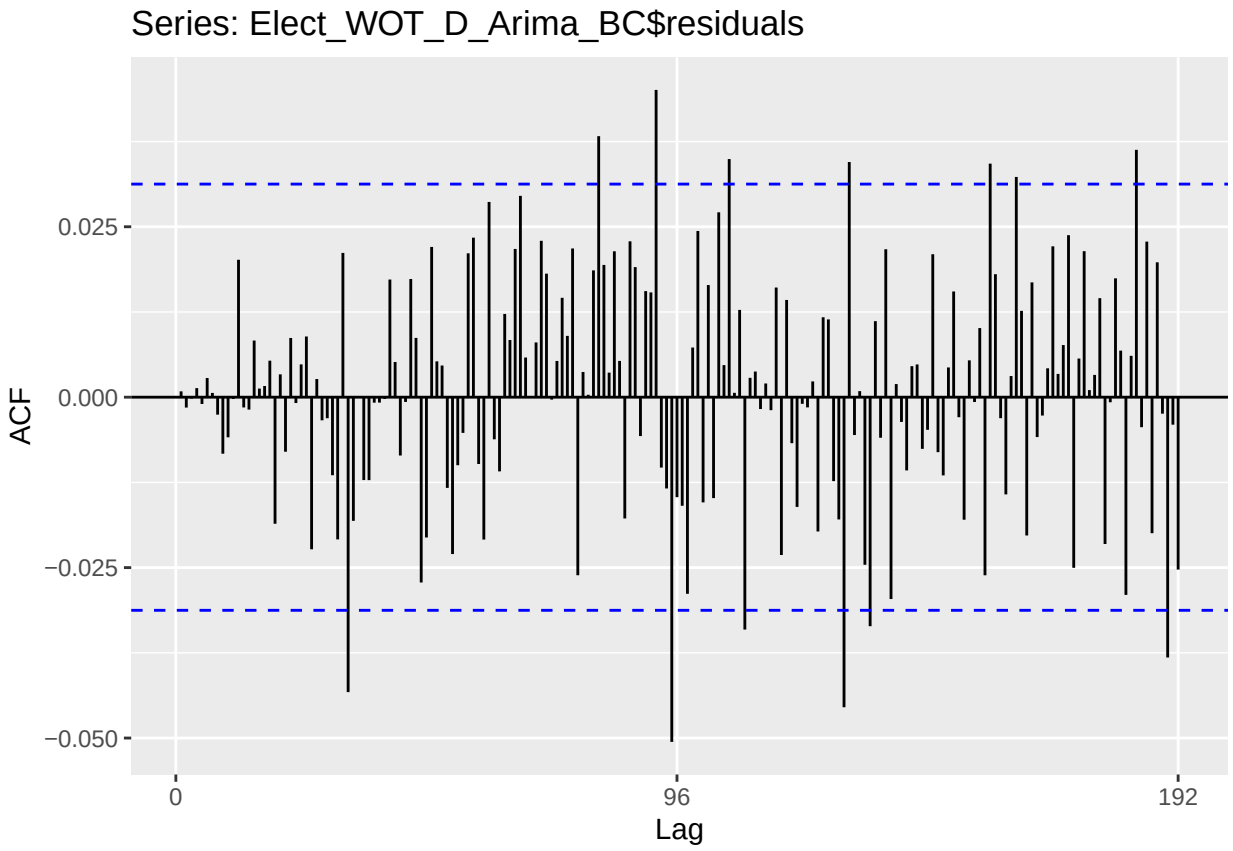
```
Elect_WOT_D_Arima_BC <- Arima(Elect_D.ts.train[,"Power_(kW)"], order=c(5,1,12),  seasonal=c(0,1,1), xreg= Elect_
```

```
checkresiduals(Elect_WOT_D_Arima_BC)
```

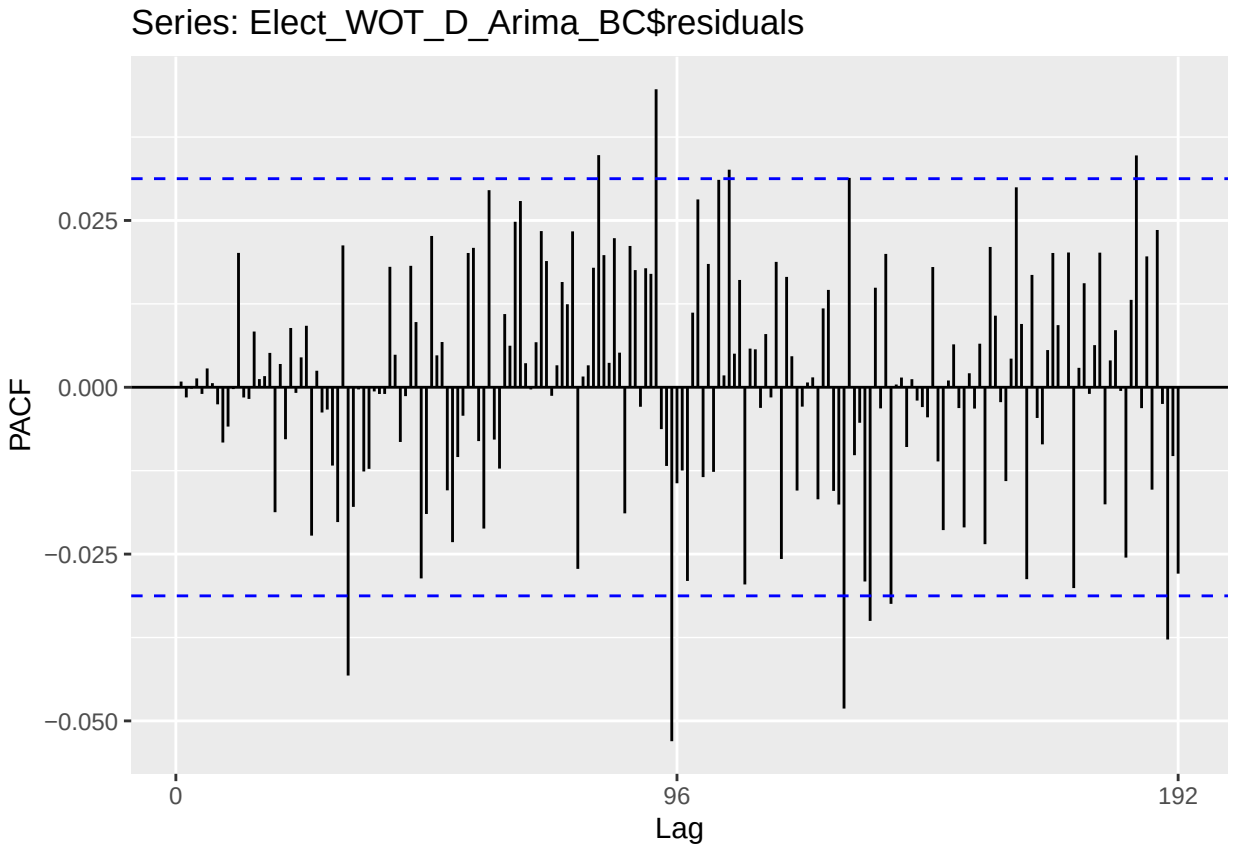



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,12)(0,1,1)[96] errors
## Q* = 219.48, df = 174, p-value = 0.01107
##
## Model df: 18.   Total lags used: 192
```

```
ggAcf(Elect_WOT_D_Arima_BC$residuals)
```

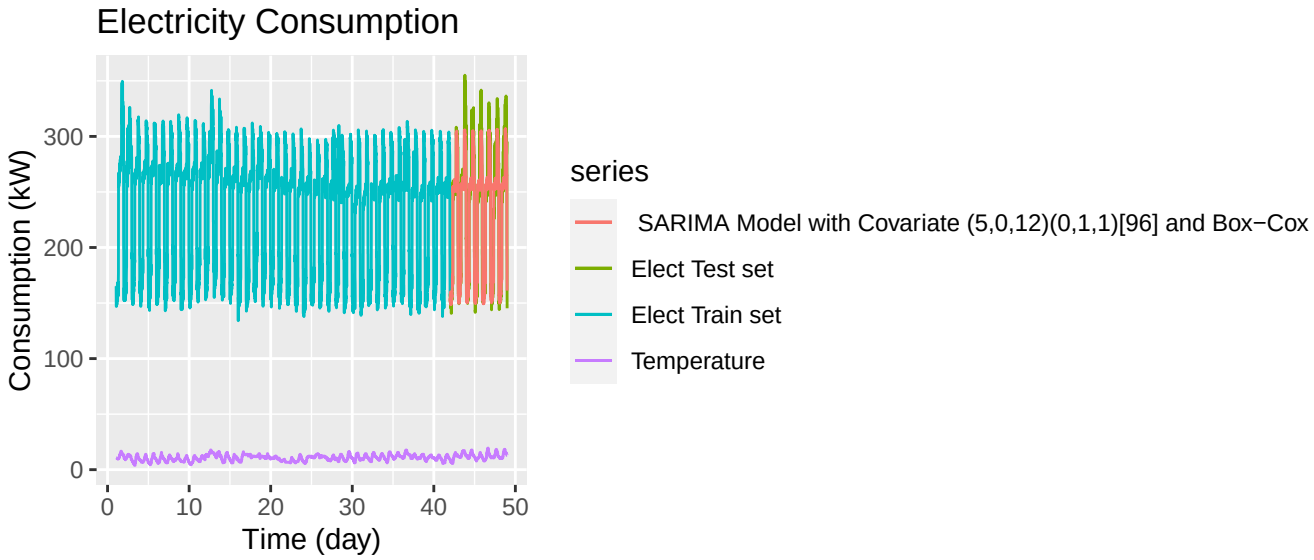



```
ggPacf(Elect_WOT_D_Arima_BC$residuals)
```




```
Elect_WOT_D_Arima_BC_forcast <- forecast :: forecast(Elect_WOT_D_Arima_BC , xreg= Elect_D.ts.test[,"Temp_(C°)"]
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_Arima_BC_forcast$mean, series=' SARIMA Model with Covariate (5,0,12)(0,1,1)[96] and Box-
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_Arima_BC_RMSE <- sqrt(mean((Elect_WOT_D_Arima_BC_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOT_D_Arima_BC_AIC = Elect_WOT_D_Arima_BC$aicc

cat("The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Box-Cox Transformation equal to:", Elect_WO
```

```
## The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Box-Cox Transformation equal to: 15.40838
```

```
cat("The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Box-Cox Transformation equal to:", Elect_WOT
```

```
## The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Box-Cox Transformation equal to: 2958.875
```

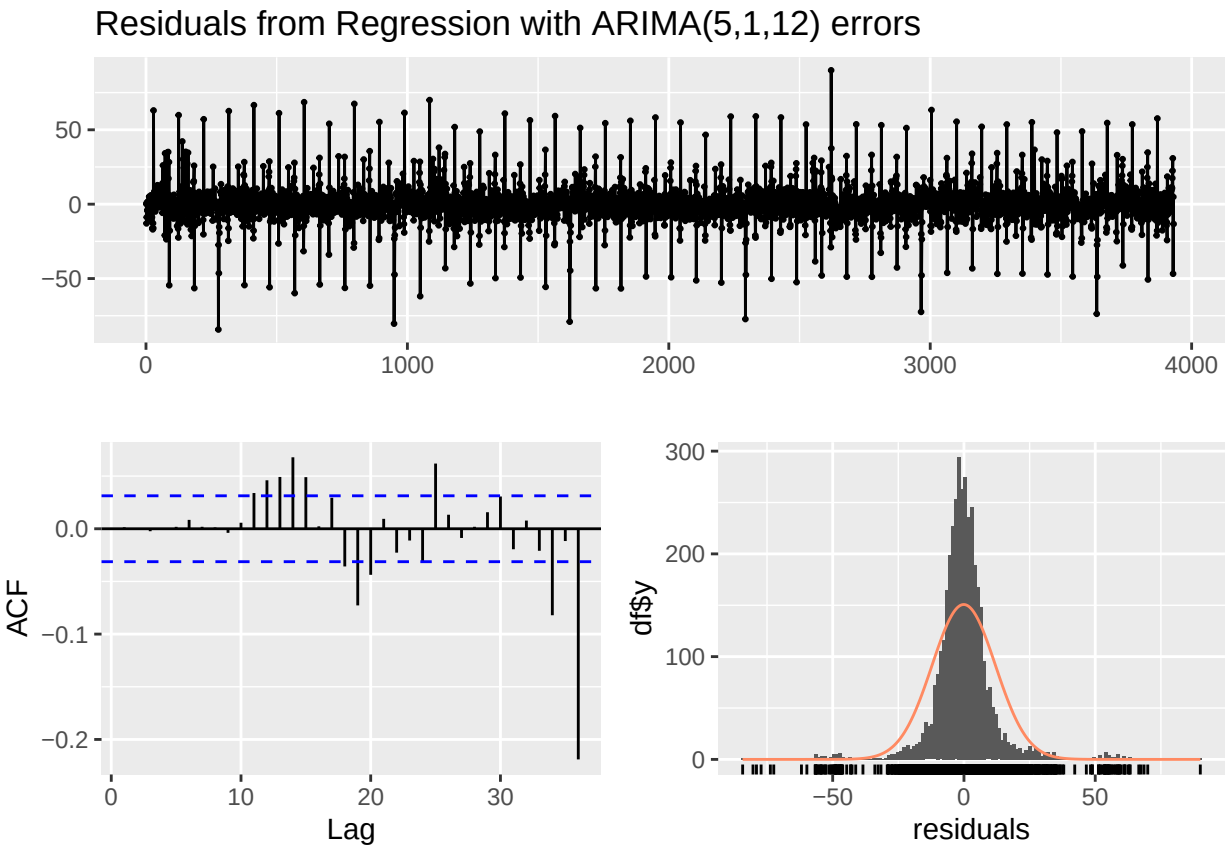
When we add a Box-Cox transformation to the SARIMA model the result get better. We have nowe less autocorrelation between residuals and the Ljung-Box test P-value is significante at 90% (alpha = 10%).

Also, the RMSE on a test set is the smallest comparing with previous models.

II-2-4-3- Forecasting Manuel SARIMA Model with covariate and Fourrier transformation

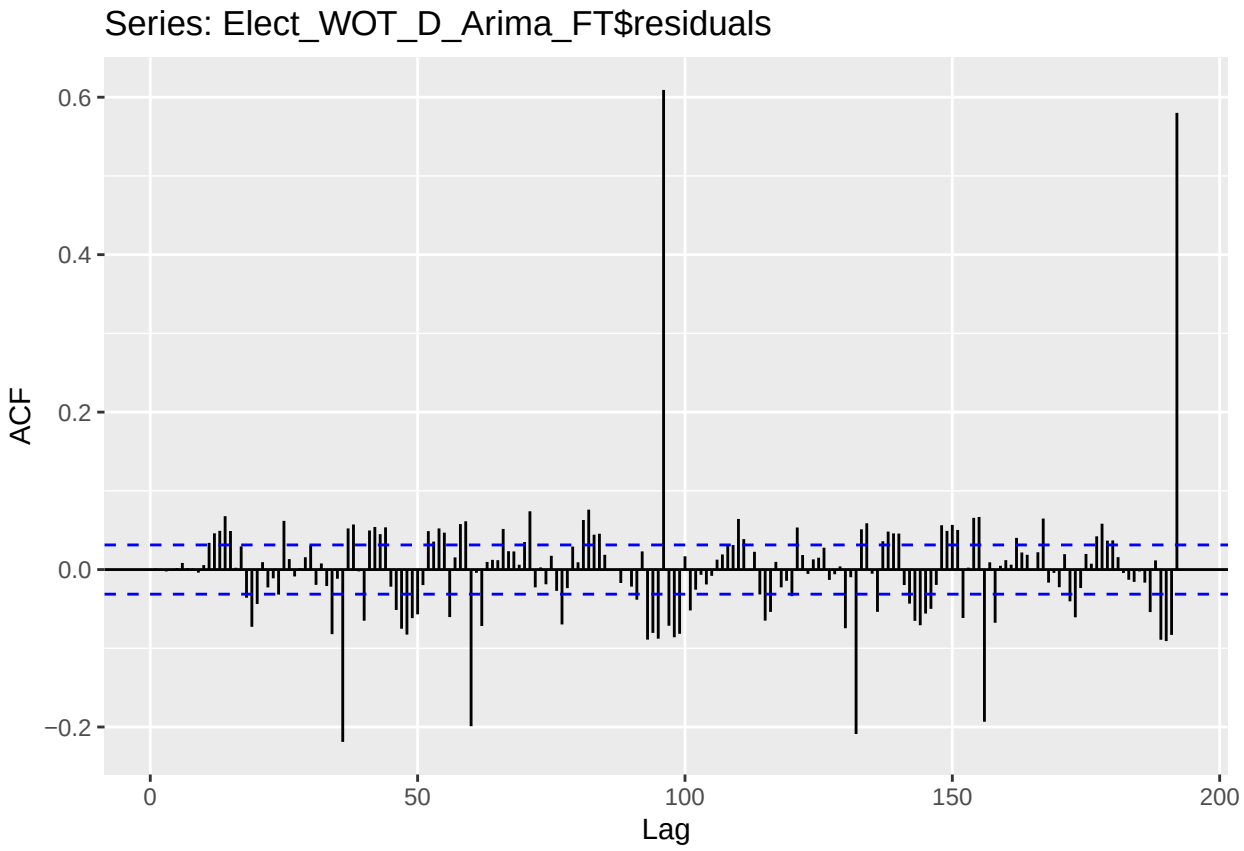
```
Elect_WOT_D_Arima_FT <- Arima(ts(Elect.train_TF[,"Power_(kW)"]), order=c(5,1,12),  seasonal=c(0,0,0),  xreg= cbi
```

```
checkresiduals(Elect_WOT_D_Arima_FT)
```

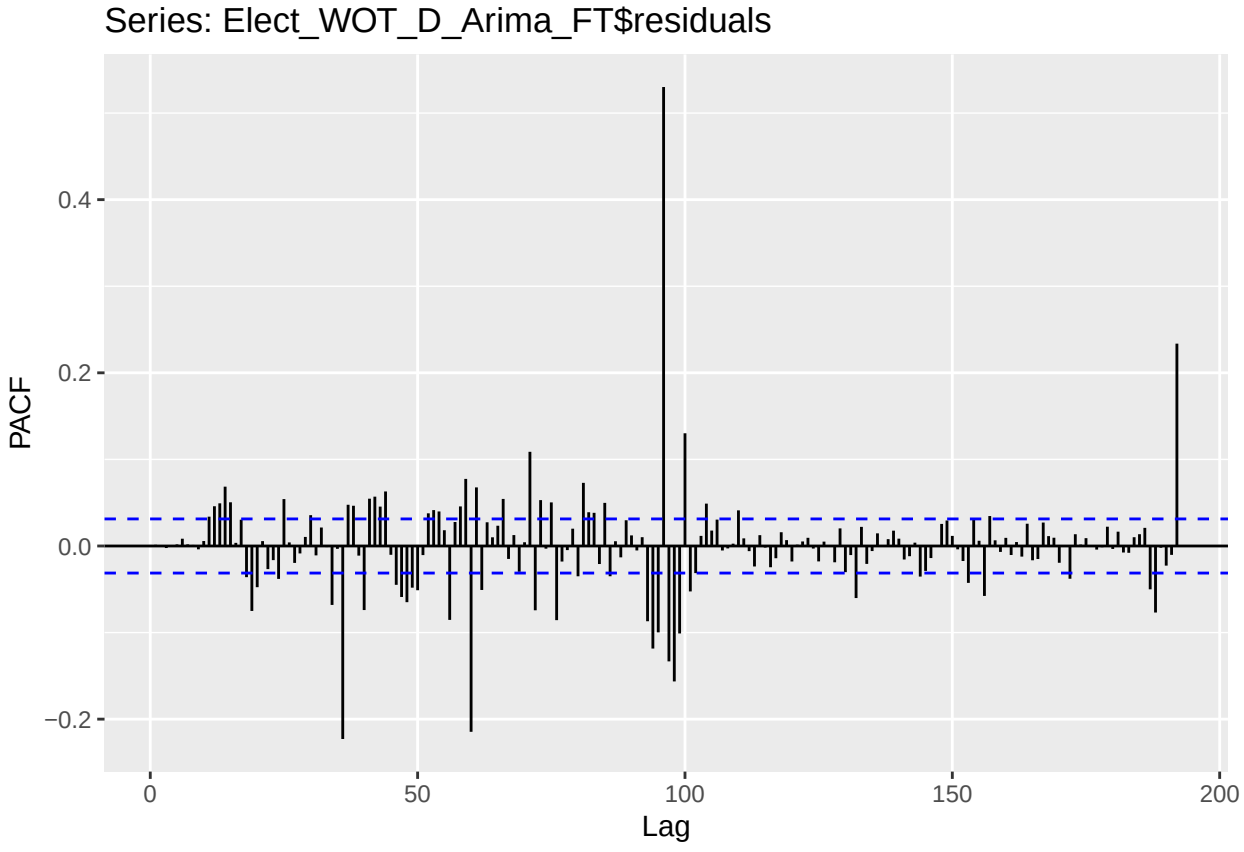



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,12) errors
## Q* = 87.538, df = 3, p-value < 2.2e-16
##
## Model df: 17.   Total lags used: 20
```

```
ggAcf(Elect_WOT_D_Arima_FT$residuals, lag.max = 192)
```




```
ggPacf(Elect_WOT_D_Arima_FT$residuals, lag.max = 192)
```

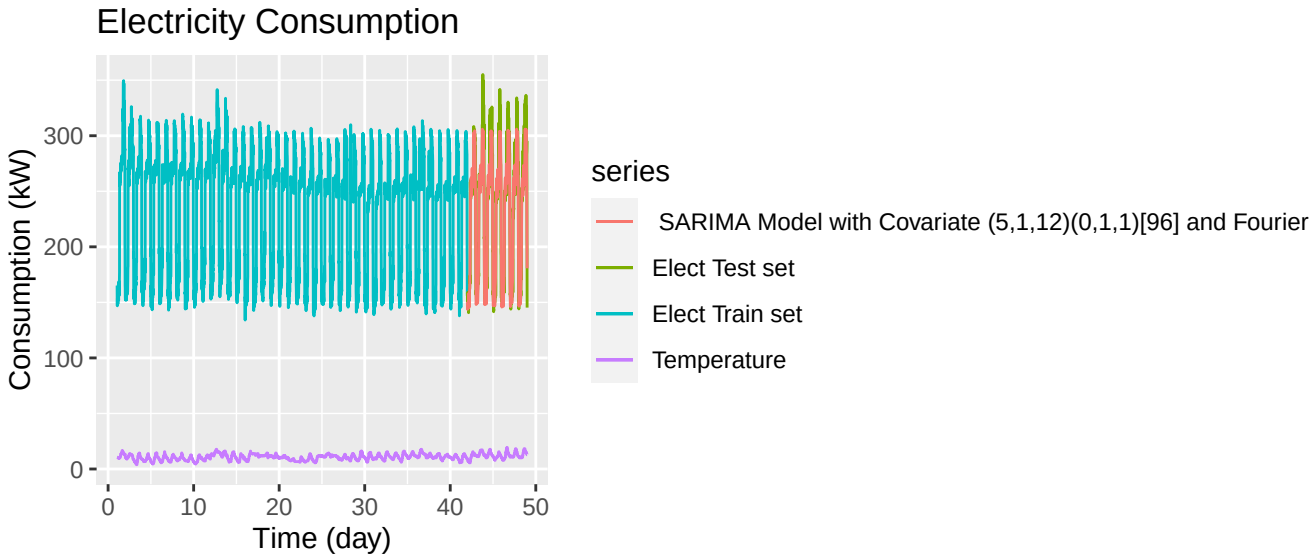



```
Elect_WOT_D_Arima_FT_forcast <- forecast :: forecast(Elect_WOT_D_Arima_FT , xreg= cbind(Elect_D.ts.test[,"Temp_(
```

```
## Warning in forecast.forecast_ARIMA(Elect_WOT_D_Arima_FT, xreg =
```

```
## cbind(Elect_D.ts.test[, : xreg contains different column names from the xreg
## used in training. Please check that the regressors are in the same order.
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(ts(Elect_WOT_D_Arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series=' SARIMA Model with
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_Arima_FT_RMSE <- sqrt(mean((ts(Elect_WOT_D_Arima_FT_forcast$mean, frequency = 96, start = c(42, 1))-
Elect_WOT_D_Arima_FT_AIC = Elect_WOT_D_Arima_FT$aicc

cat("The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Fourier Transformation equal to:", Elect_WO
```

```
## The RMSE of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Fourier Transformation equal to: 18.11876
```

```
cat("The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Fourier Transformation equal to:", Elect_WOT
```

```
## The AIC of SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Fourier Transformation equal to: 30814.51
```

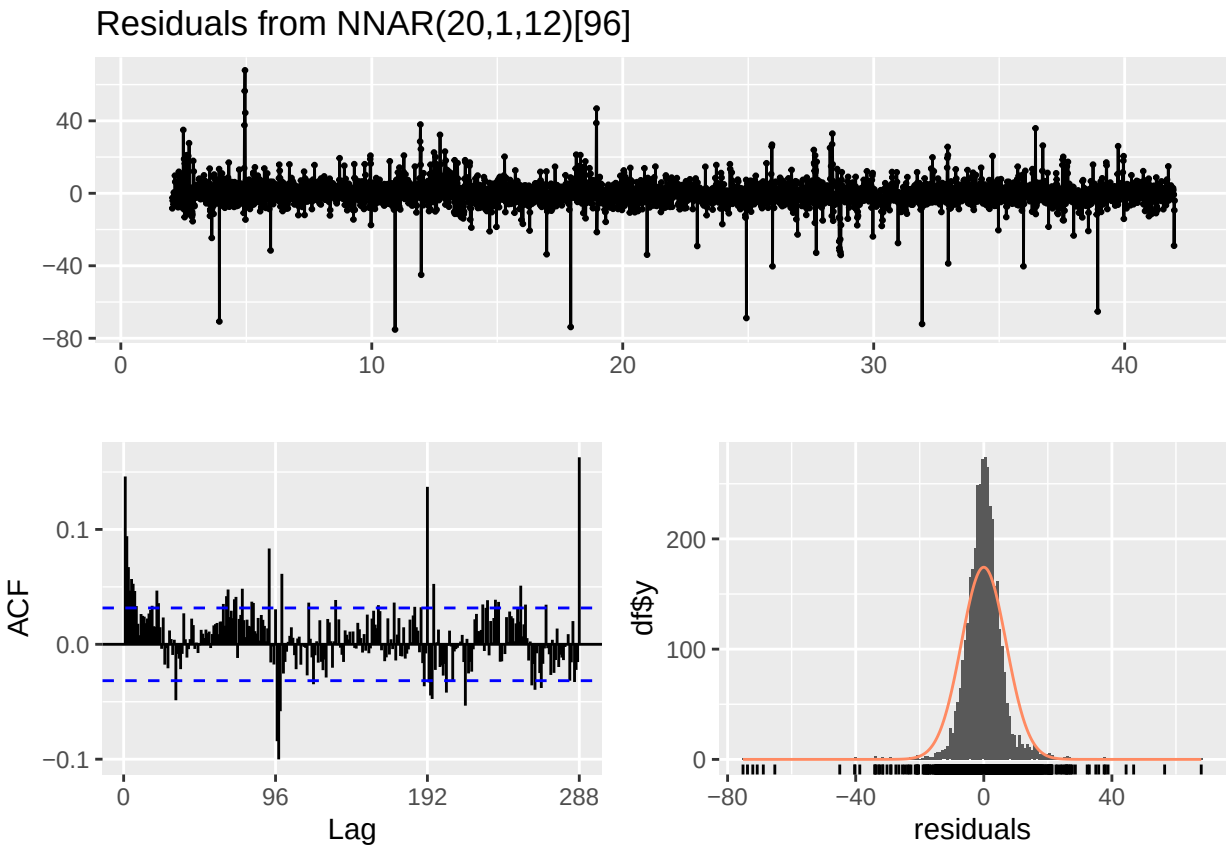
The Fourier transformation didn't add a positive effect on the SARIMA model.

II-2-5- Forecastring NNAR model with covariates

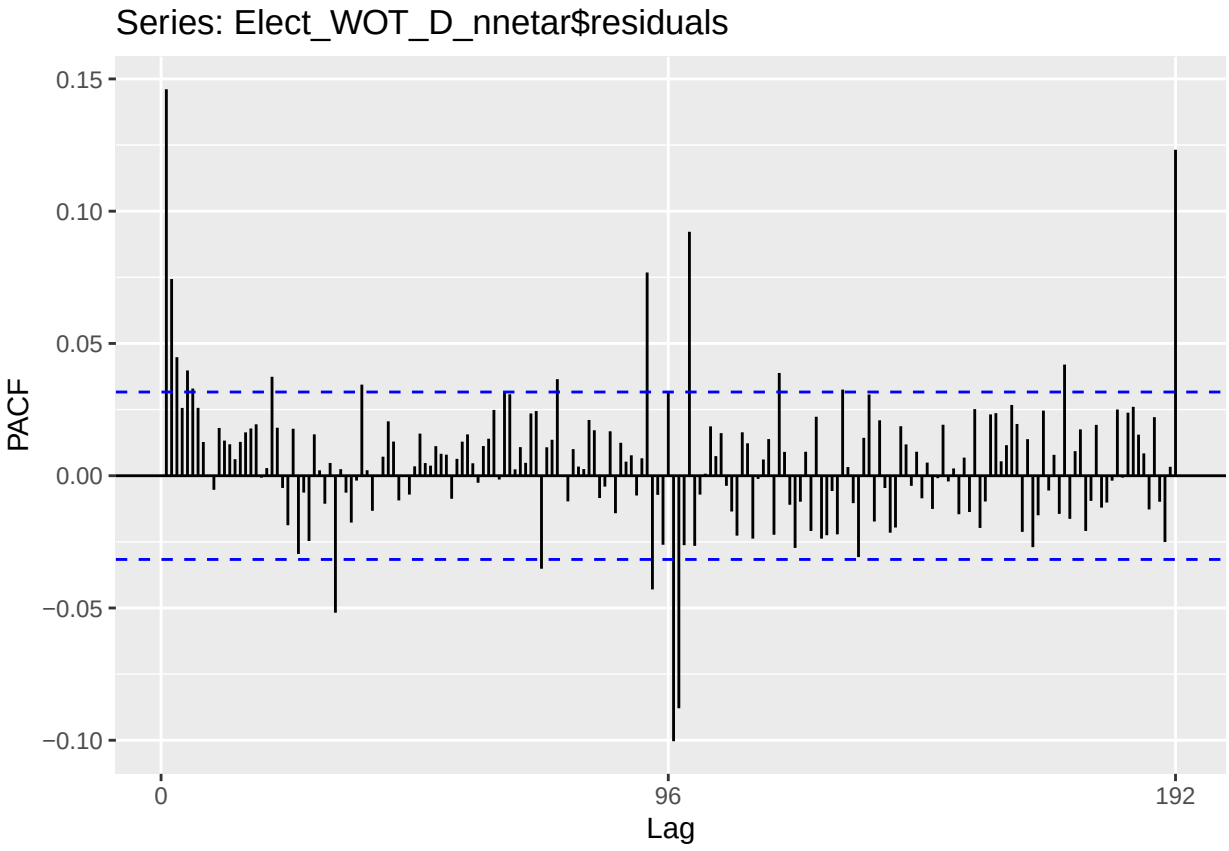
II-2-5-1- Forecastring NNAR model with covariates

```
Elect_WOT_D_nnetar <- nnetar(Elect_D.ts.train[,"Power_(kW)"], xreg= Elect_D.ts.train[,"Temp_(C°)"])
checkresiduals(Elect_WOT_D_nnetar)
```

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

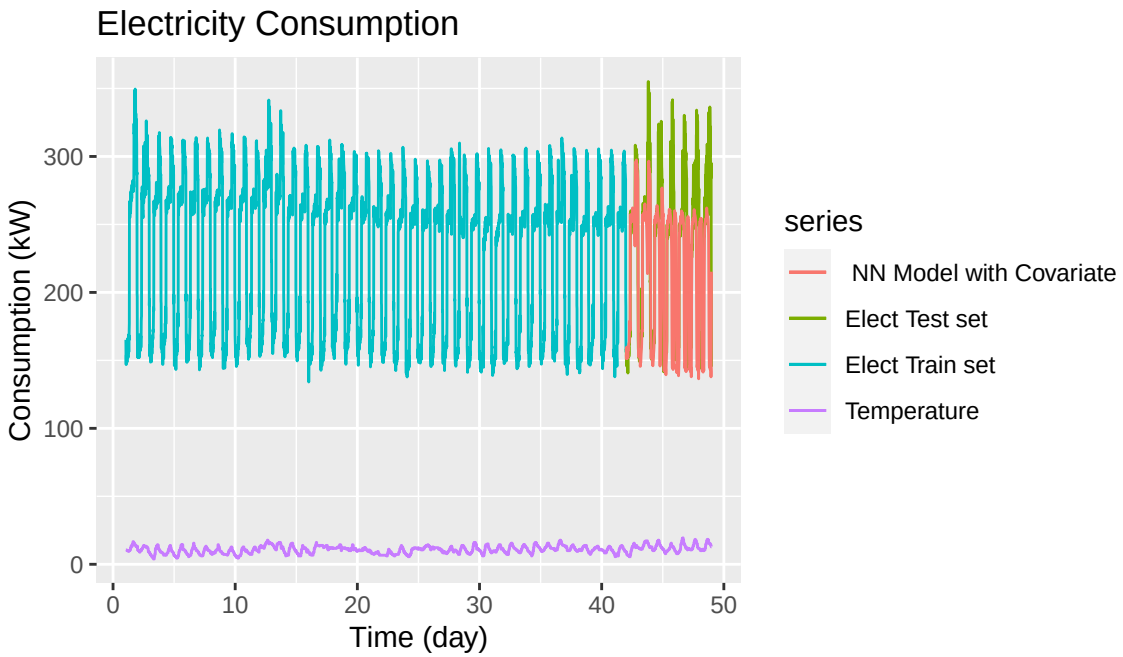



```
ggPacf(Elect_WOT_D_nnetar$residuals)
```




```
Elect_WOT_D_nnetar_forcast <- forecast :: forecast(Elect_WOT_D_nnetar , xreg= Elect_D.ts.test[,"Temp_(C°)"] , h=
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_nnetar_forcast$mean, series=' NN Model with Covariate')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_NN_RMSE <- sqrt(mean((Elect_WOT_D_nnetar_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOT_D_NN_AIC = Elect_WOT_D_nnetar$aicc

cat("The RMSE of Neural Network with Covariate equale to:", Elect_WOT_D_NN_RMSE, "\n")
```

```
## The RMSE of Neural Network with Covariate equale to: 66.37661
```

```
cat("The AIC of Neural Network Model with Covariate equale to:", Elect_WOT_D_NN_AIC, "\n")
```

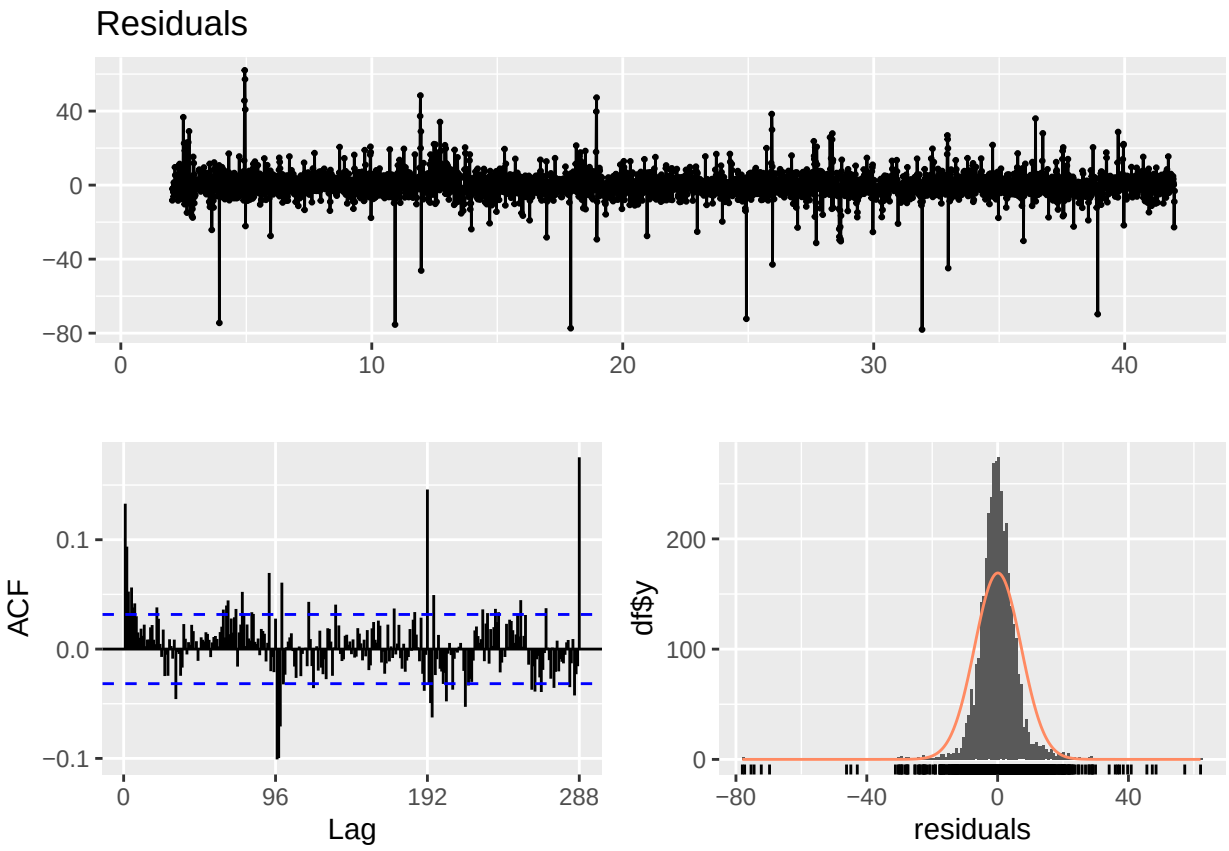
```
## The AIC of Neural Network Model with Covariate equale to:
```

The NNAR model shows a noisy residuals and a very bad RMSE.

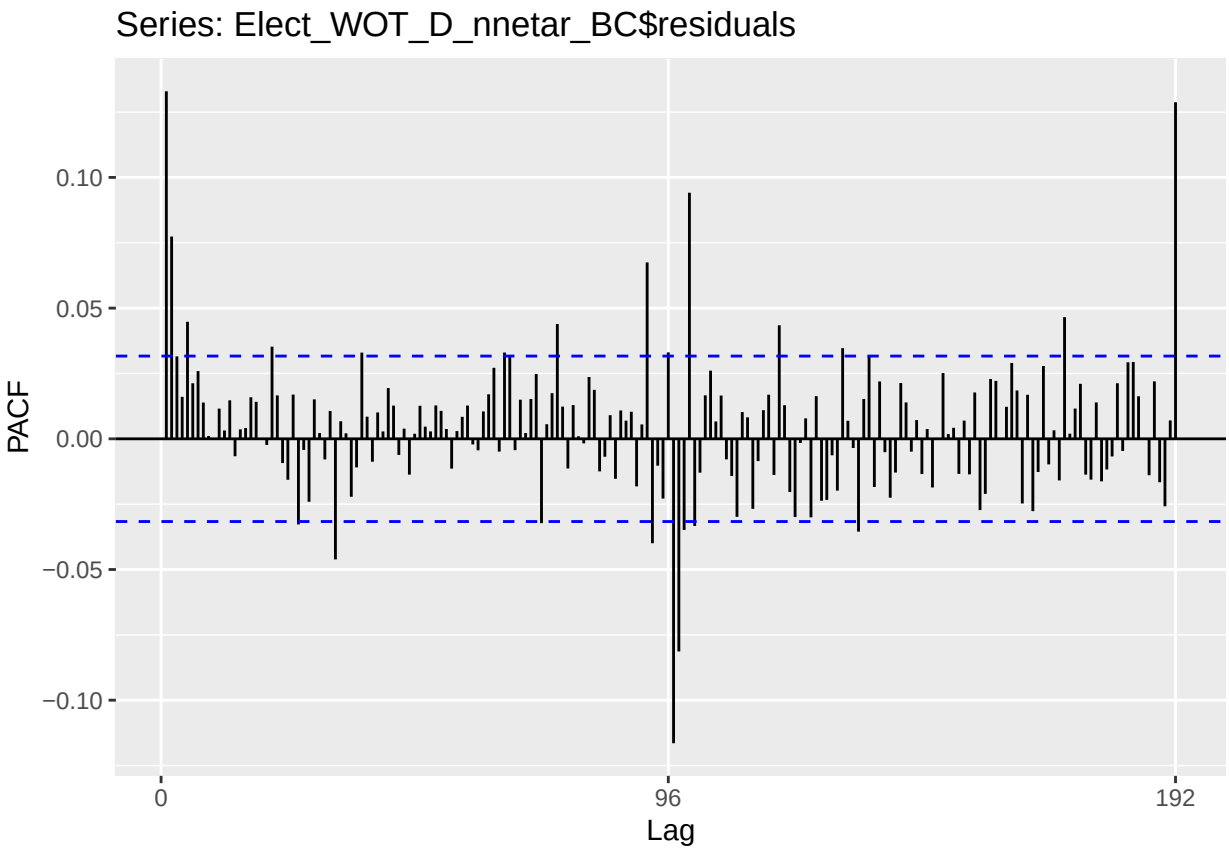
II-2-5-2- Forecastring NNAR model with covariates and Box-Cox Transformation

```
Elect_WOT_D_nnetar_BC <- nnetar(Elect_D.ts.train[,"Power_(kW)"], xreg= Elect_D.ts.train[,"Temp_(C°)"], lambda =
checkresiduals(Elect_WOT_D_nnetar_BC$residuals)
```

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

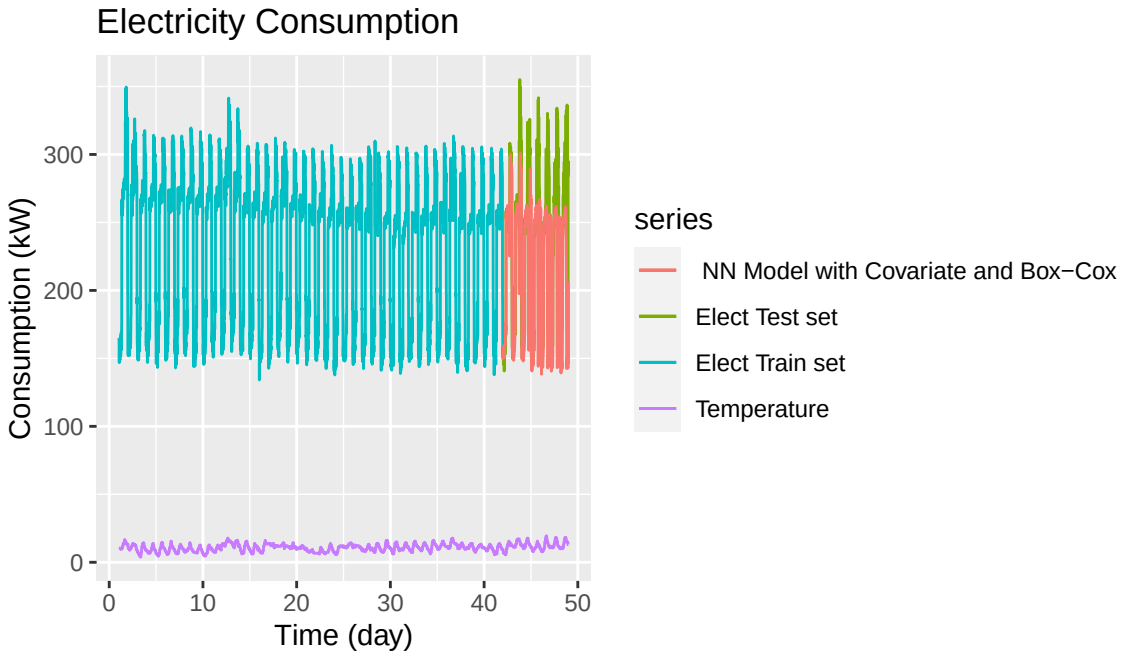



```
ggPacf(Elect_WOT_D_nnetar_BC$residuals)
```




```
Elect_WOT_D_nnetar_BC_forcast <- forecast :: forecast(Elect_WOT_D_nnetar_BC , xreg= Elect_D.ts.test[,"Temp_(C°)"
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(Elect_WOT_D_nnetar_BC_forcast$mean, series=' NN Model with Covariate and Box-Cox')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_NN_BC_RMSE <- sqrt(mean((Elect_WOT_D_nnetar_BC_forcast$mean-Elect_D.ts.test[,"Power_(kW)"])^2))
Elect_WOT_D_NN_BC_AIC = Elect_WOT_D_nnetar_BC$aicc

cat("The RMSE of Neural Network with Covariate and Box-Cox Transformation equale to:", Elect_WOT_D_NN_BC_RMSE, '
```

```
## The RMSE of Neural Network with Covariate and Box-Cox Transformation equale to: 63.12776
```

```
cat("The AIC of Neural Network Model with Covariate and Box-Cox Transformation equale to:", Elect_WOT_D_NN_BC_AI
```

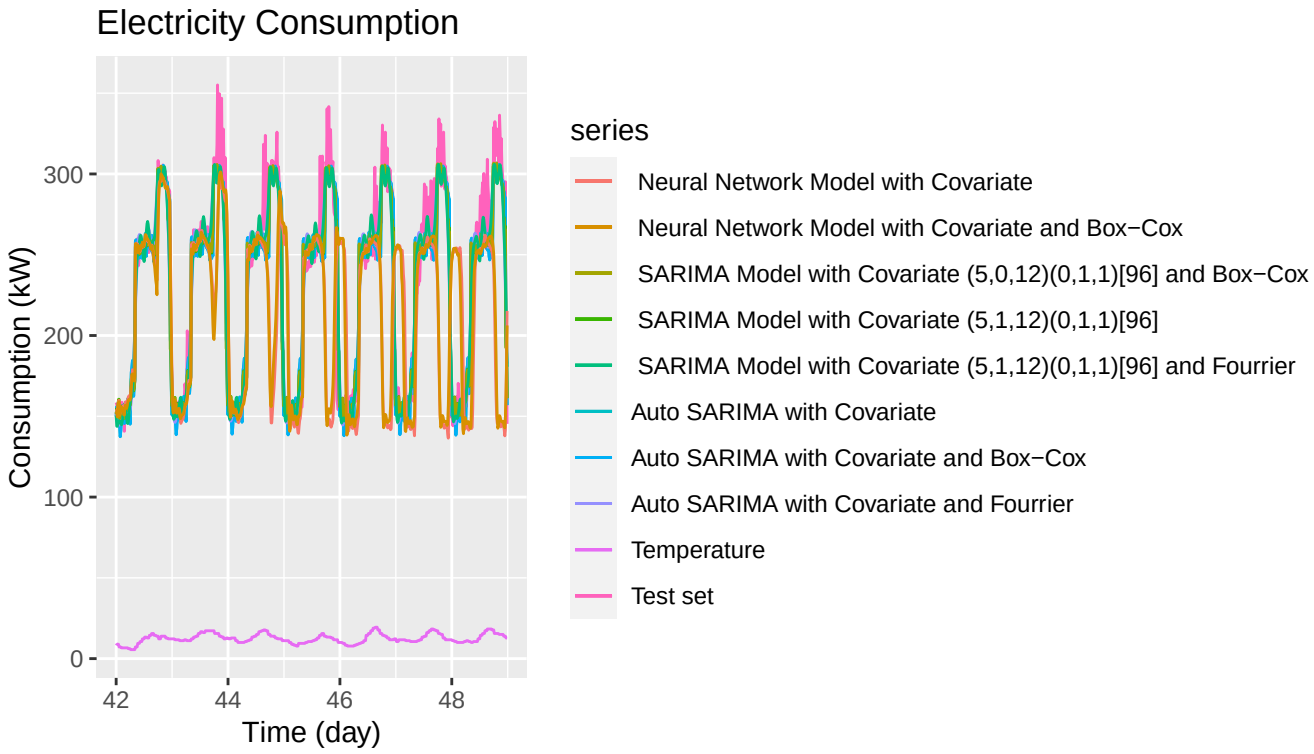
```
## The AIC of Neural Network Model with Covariate and Box-Cox Transformation equale to:
```

The NNAR model with Box-Cox transformation is almost the same as the NNAR without BOX-Cox transformation. This model is not good since the residuals are noisy ans RMSE is bad.

II-2-6- Ploting all the models

```
autoplot(Elect_D.ts.test[,"Power_(kW)"], series='Test set') +
  autolayer(Elect_D.ts.test[,"Temp_(C°)"], series='Temperature') +

  autolayer(Elect_WOT_D_auto.arima_forcast$mean, series='Auto SARIMA with Covariate')+
  autolayer(Elect_WOT_D_auto.arima_BC_forcast$mean, series='Auto SARIMA with Covariate and Box-Cox')+
  autolayer(ts(Elect_WOT_D_auto.arima_FT_forcast$mean,frequency = 96, start = c(42, 1)), series='Auto SARIMA wit
  autolayer(Elect_WOT_D_Arima_forcast$mean, series=' SARIMA Model with Covariate (5,1,12)(0,1,1)[96]')+
  autolayer(Elect_WOT_D_Arima_BC_forcast$mean, series=' SARIMA Model with Covariate (5,0,12)(0,1,1)[96] and Box-
  autolayer(ts(Elect_WOT_D_Arima_FT_forcast$mean, frequency = 96, start = c(42, 1)), series=' SARIMA Model with
  autolayer(Elect_WOT_D_nnetar_forcast$mean, series=' Neural Network Model with Covariate')+
```

```
  autolayer(Elect_WOT_D_nnetar_BC_forcast$mean, series=' Neural Network Model with Covariate and Box-Cox')+

  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```



II-2-7- Comparison of the RMSE, AIC and Ljung Box test

```
results_Covariat <- data.frame(
  Model_without_covariate = c("Holt Winters Model with Covariate (Additive Seasonality)","Holt Winters Model wit
  RMSE = c("-", "-", Elect_WOT_D_auto.arima_RMSE, Elect_WOT_D_auto.arima_BC_RMSE, Elect_WOT_D_auto.arima_FT_RMSE
  AICC = c("-","-", Elect_WOT_D_auto.arima_AIC, Elect_WOT_D_auto.arima_BC_AIC, Elect_WOT_D_auto.arima_FT_AIC, El
  Ljung_Box_test = c("-","-","2.2e-16","2.2e-16",0.05696, 0.0009174,0.01107 ,"2.2e-16" ,"-","-","-")

)

results_Covariat
```

```
##                                                  Model_without_covariate
## 1                      Holt Winters Model with Covariate (Additive Seasonality)
## 2                Holt Winters Model with Covariate (Multiplicative Seasonality)
## 3                                          Auto SARIMA Model with Covariate
## 4                   Auto SARIMA Model with Covariate and Box-Cox transformation
## 5                  Auto SARIMA Model with Covariate and Fourrier transformation
## 6                              SARIMA Model with Covariate (5,1,12)(0,1,1)[96]')
## 7  SARIMA Model with Covariate (5,1,12)(0,1,1)[96][4] and Box-Cox Transformation
## 8    SARIMA Model with Covariate (5,1,12)(0,1,1)[96] and Fourrier Transformation
## 9                                                        Neural Network Model
## 10                            Neural Network Model with Box-Cox Transformation
## 11                                                               TBATS Model
##                RMSE             AICC Ljung_Box_test
## 1                 -                -              -
## 2                 -                -              -
```

```
## 3  16.8729710467494 28481.5500137868        2.2e-16
## 4  16.8258373711971 5267.29008680568        2.2e-16
## 5  18.5635372043124 30881.8735605881         0.05696
## 6  15.5848080671078 26206.0948516872       0.0009174
## 7  15.4083770027205 2958.87484868275         0.01107
## 8   18.118763169949 30814.5091644661        2.2e-16
## 9  66.3766063672239                -               -
## 10 63.1277551889114                -               -
## 11                 -                -               -
```

II- Double seasonality forecast:

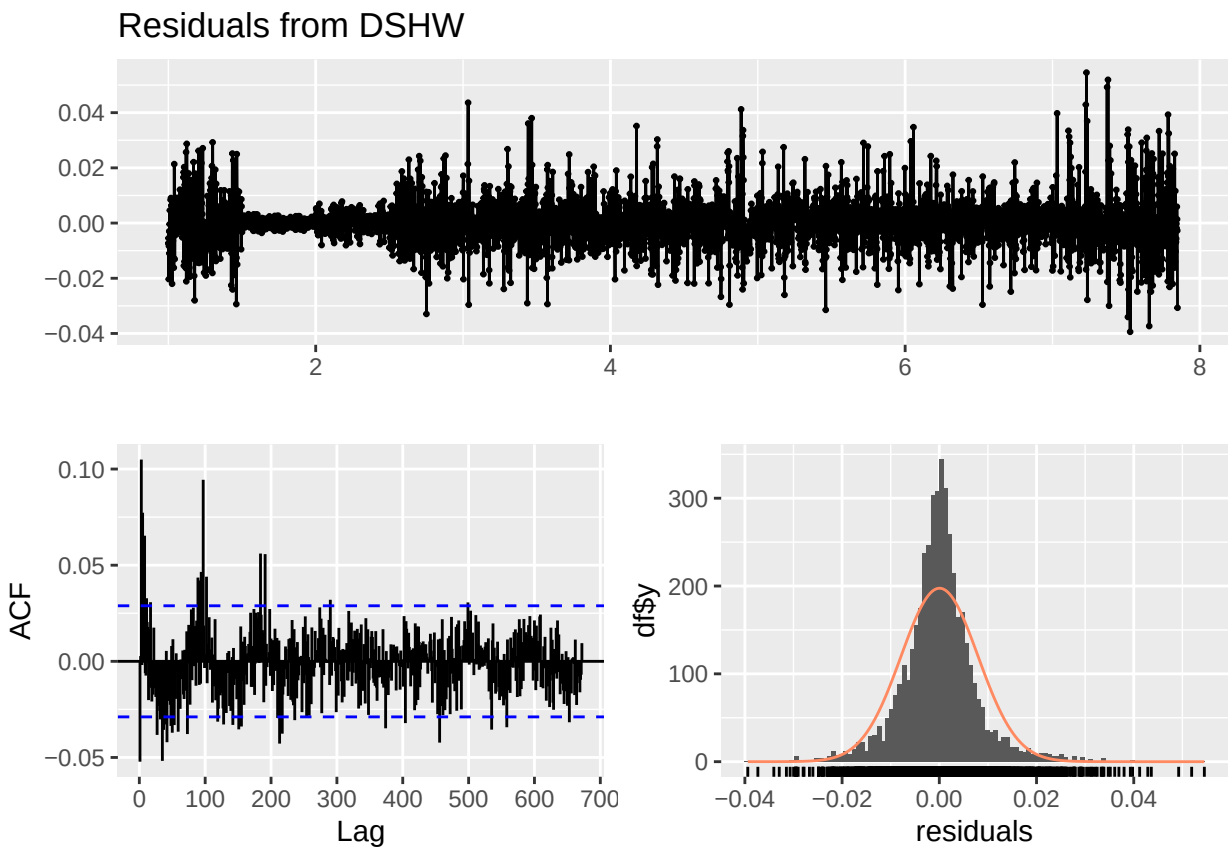
II-1. Fitting Electricity Consumption (kW) models without using outdoor temperature:

II-1-1 Forecast Double Seasonality Holt Winter Model:

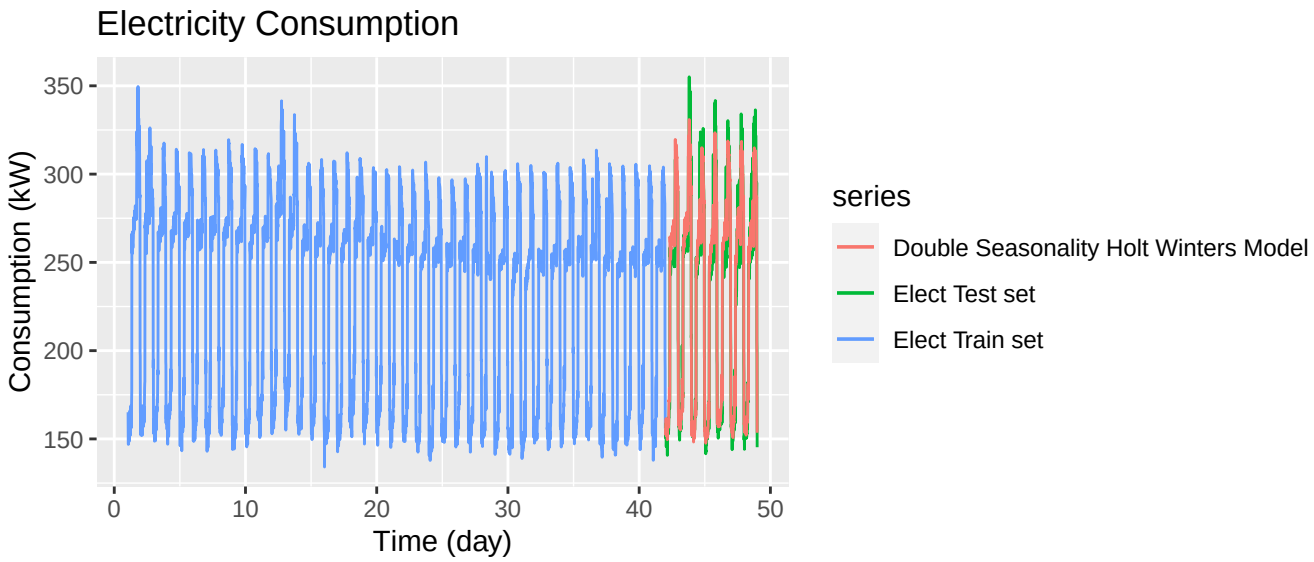
```
Elect_WOOT_D_DSHW = dshw(Elect_WOOT[,"Power_(kW)"], period1=96, period2 = 672, h=672, lambda = 'auto')
```

```
checkresiduals(Elect_WOOT_D_DSHW , lag.max=672)
```

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




```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(ts(Elect_WOOT_D_DSHW$mean, frequency = 96, start=c(42,1)), series='Double Seasonality Holt Winters M
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_DSHW_RMSE <- sqrt(mean((as.numeric(Elect_WOOT_D_DSHW$mean) - as.numeric(Elect_D.ts.test[,"Power_(kW
Elect_WOOT_D_DSHW_AIC = Elect_WOOT_D_DSHW$aicc

cat("The RMSE of Double Seasonality Holt Winters Model equal to:", Elect_WOOT_D_DSHW_RMSE, "\n")
```

```
## The RMSE of Double Seasonality Holt Winters Model equal to: 9.798622
```

```
cat("The AIC of Double Seasonality Holt Winters Model equal to:", Elect_WOOT_D_DSHW_AIC, "\n")
```

```
## The AIC of Double Seasonality Holt Winters Model equal to:
```

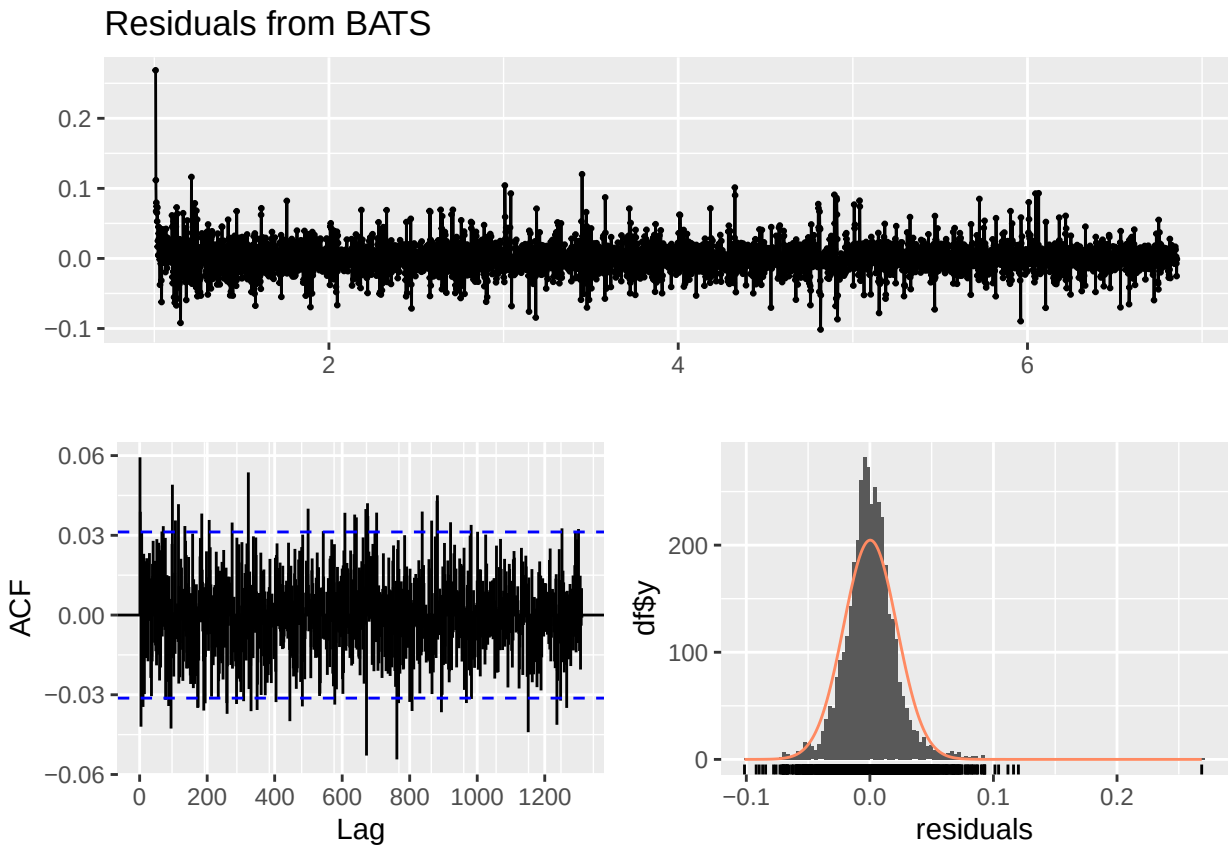
II-1-2. Forecasting BATS model

BATS is a time series model that is useful for handling data with multiple seasonal patterns, i.e., the data that changes over time. The TBATS is preferred over BATS as the Trigonometric seasonality (TBATS) can deal with complex and high frequency.

```
Elect_2S.ts.train <- window(Elect_WOOT_2S, start=c(1,6), end=c(6,576))

Elect_2S.ts.test <- window(Elect_WOOT_2S, start=c(6,577), end=c(7,576))
```

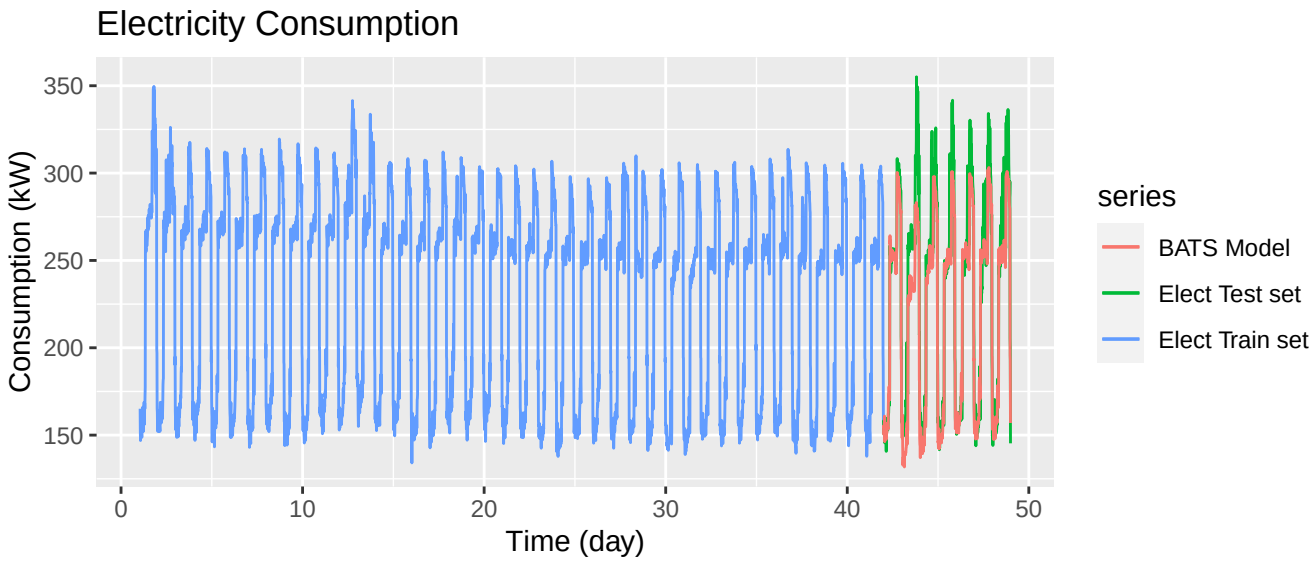
```
Elect_WOOT_D_S2_BATS = bats(Elect_2S.ts.train[,"Power_(kW)"])
```

```
checkresiduals(Elect_WOOT_D_S2_BATS)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from BATS
## Q* = 952.65, df = 777, p-value = 1.49e-05
##
## Model df: 9.   Total lags used: 786
```

```
Elect_WOOT_D_S2_BATS_forcast <- forecast::forecast(Elect_WOOT_D_S2_BATS, h=672)
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set')+
  autolayer(ts(Elect_WOOT_D_S2_BATS_forcast$mean, frequency=96, start=c(42,1)), series='BATS Model')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOOT_D_S2_BATS_RMSE <- sqrt(mean((Elect_WOOT_D_S2_BATS_forcast$mean-Elect_2S.ts.test[,"Power_(kW)"])^2))
Elect_WOOT_D_S2_BATS_AIC = Elect_WOOT_D_S2_BATS$aicc

cat("The RMSE of BAST model equal to:", Elect_WOOT_D_S2_BATS_RMSE, "\n")
```

```
## The RMSE of BAST model equal to: 17.9726
```

```
cat("The AIC of BAST model equal to:", Elect_WOOT_D_S2_BATS_AIC, "\n")
```

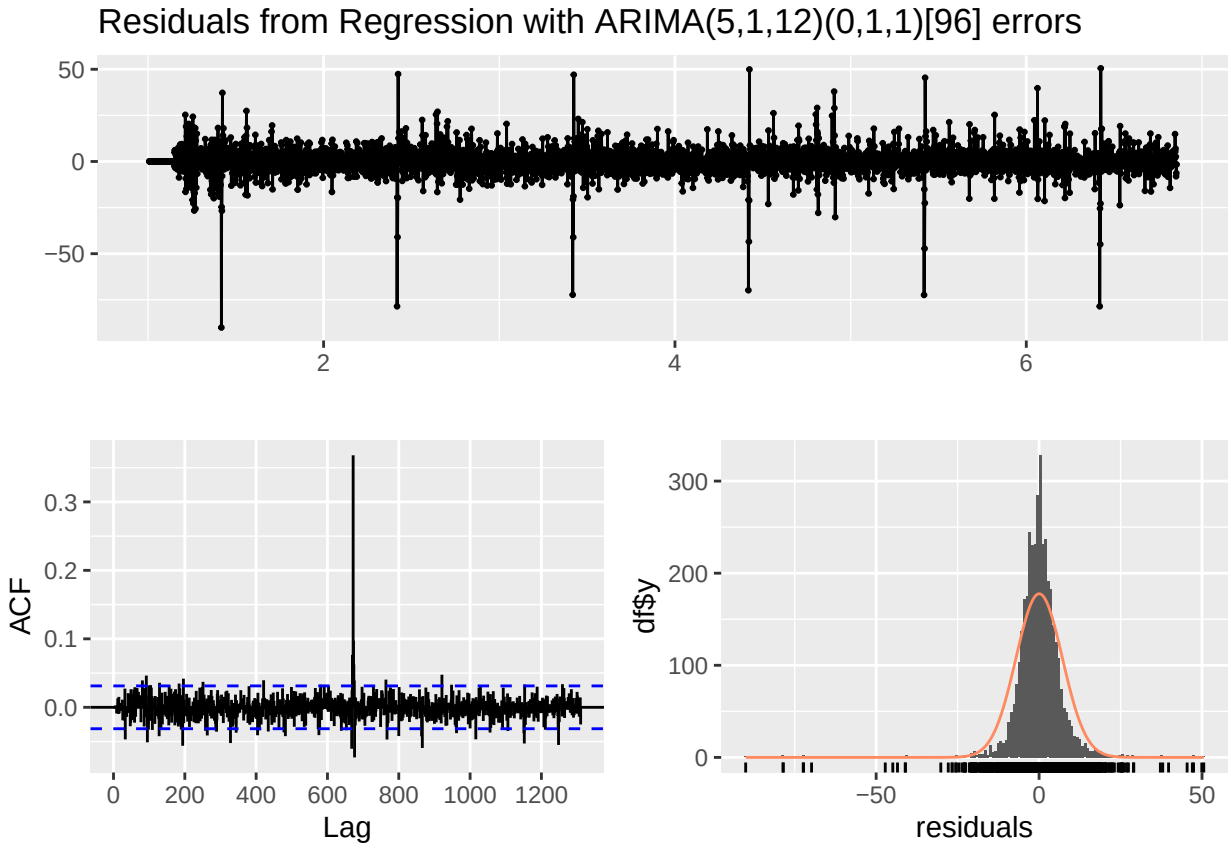
```
## The AIC of BAST model equal to:
```

II-2. Fitting Electricity Consumption (kW) models with using outdoor temperature: (with covariate)

II-2-1. Forecasting Manuel SARIMA model

```
Elect_WOT_D_2S_Arima_BC <- Arima(Elect_2S.ts.train[,"Power_(kW)"], order=c(5,1,12), seasonal=list(order=c(0,1,1)
```

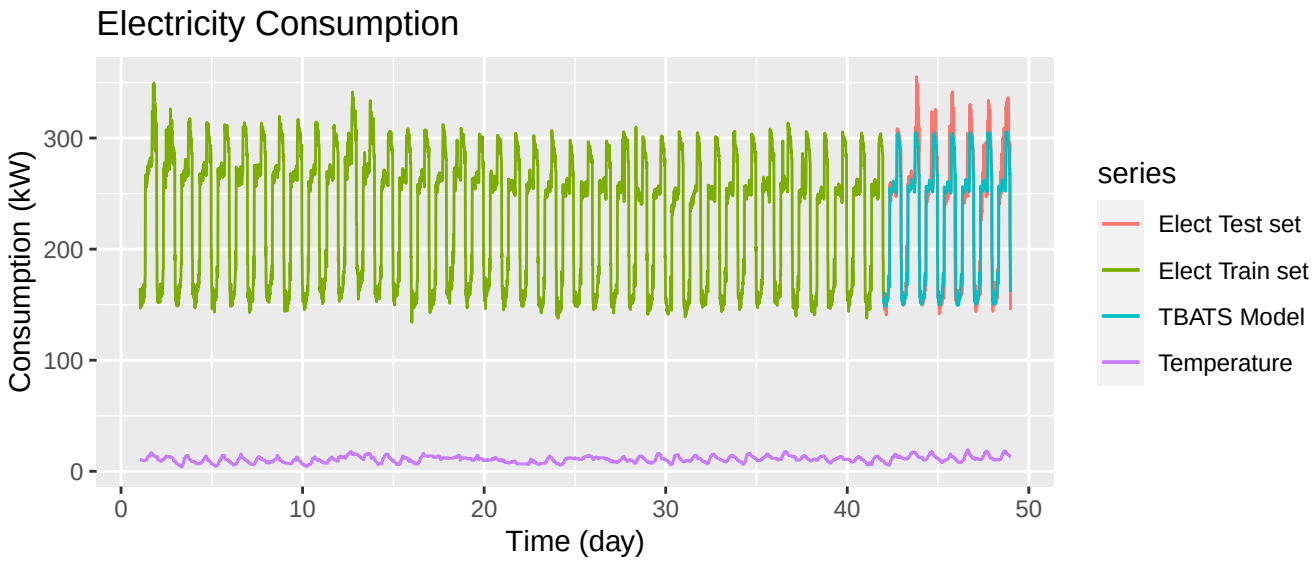
```
checkresiduals(Elect_WOT_D_2S_Arima_BC)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,1,12)(0,1,1)[96] errors
## Q* = 1634.2, df = 768, p-value < 2.2e-16
##
## Model df: 18.   Total lags used: 786
```

```
Elect_WOT_D_2S_Arima_BC_forcast <- forecast::forecast(Elect_WOT_D_2S_Arima_BC , xreg= Elect_2S.ts.test[,"Temp_(C
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(ts(Elect_WOT_D_2S_Arima_BC_forcast$mean,frequency=96, start=c(42,1)), series='TBATS Model')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_2S_Arima_BC_RMSE <- sqrt(mean((ts(Elect_WOT_D_2S_Arima_BC_forcast$mean, frequency=96, start = c(42,
Elect_WOT_D_2S_Arima_BC_AIC = Elect_WOT_D_2S_Arima_BC$aicc

cat("The RMSE of Manuel SARIMA with double seasonality model equale to:", Elect_WOT_D_2S_Arima_BC_RMSE, "\n")
```

```
## The RMSE of Manuel SARIMA with double seasonality model equale to: 15.58481
```

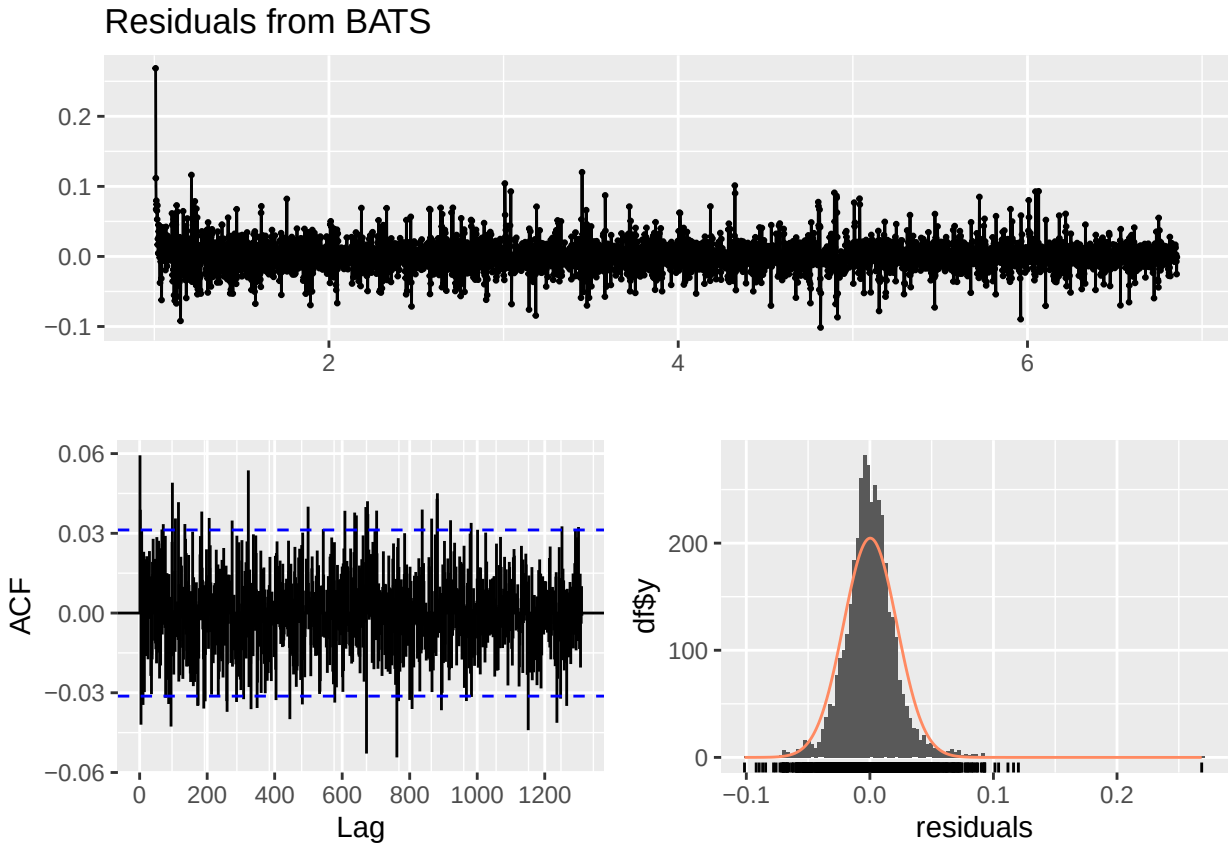
```
cat("The AIC of SARIMA with double seasonality model equale to:", Elect_WOT_D_2S_Arima_BC_AIC, "\n")
```

```
## The AIC of SARIMA with double seasonality model equale to: 26206.09
```

II-2-2. Forecasting BATS model

```
Elect_WOT_D_2S_BATS = bats(Elect_2S.ts.train[,"Power_(kW)"], xreg= Elect_2S.ts.train[,"Temp_(C°)"])
```

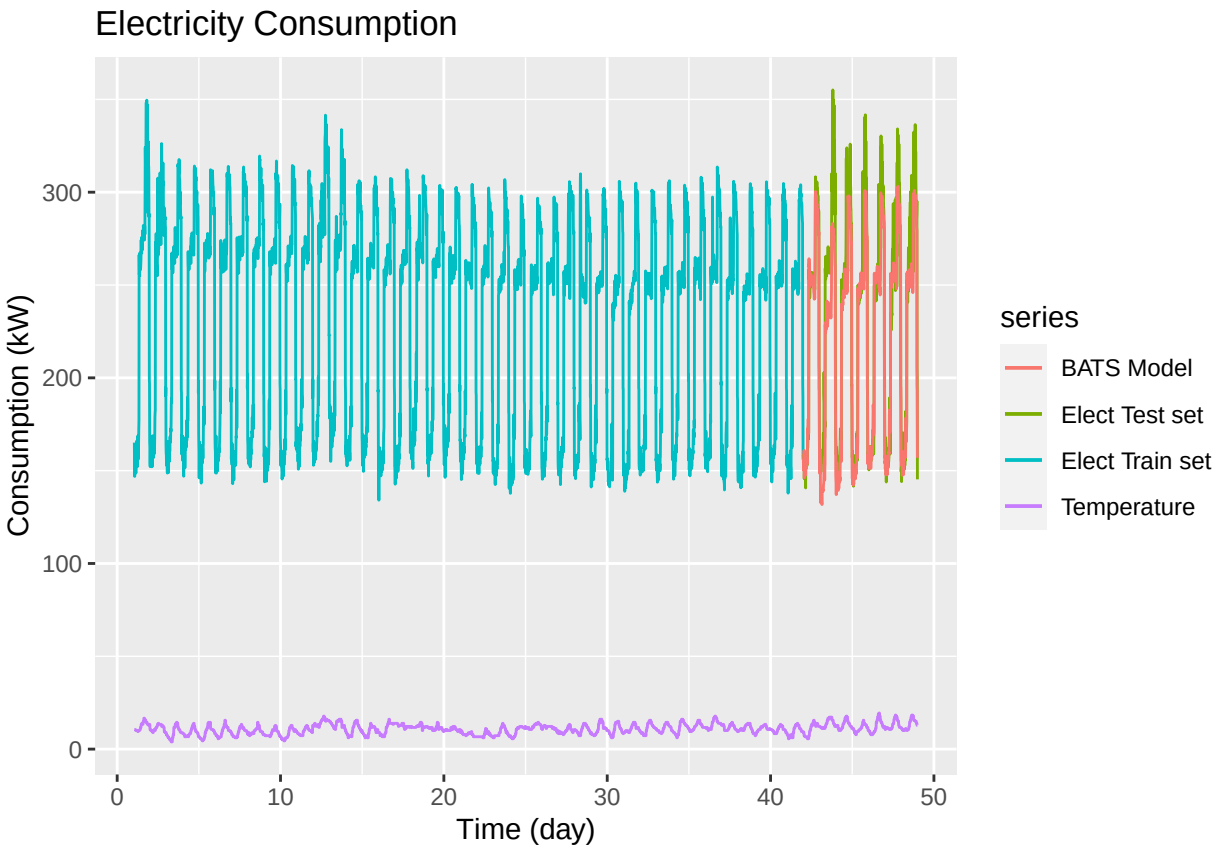
```
checkresiduals(Elect_WOT_D_2S_BATS)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from BATS
## Q* = 952.65, df = 777, p-value = 1.49e-05
##
## Model df: 9.   Total lags used: 786
```

```
Elect_WOT_D_2S_BATS_forcast <- forecast::forecast(Elect_WOT_D_2S_BATS, h=672, xreg = Elect_2S.ts.test[,"Temp_(C°
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(ts(Elect_WOT_D_2S_BATS_forcast$mean,frequency=96, start=c(42,1)), series='BATS Model')+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_2S_BATS_RMSE <- sqrt(mean((Elect_WOT_D_2S_BATS_forcast$mean-Elect_2S.ts.test[,"Power_(kW)"])^2))
Elect_WOT_D_2S_BATS_AIC = Elect_WOT_D_2S_BATS$aicc

cat("The RMSE of TBAST model equale to:", Elect_WOT_D_2S_BATS_RMSE, "\n")
```

```
## The RMSE of TBAST model equale to: 17.9726
```

```
cat("The AIC of TBAST model equale to:", Elect_WOT_D_2S_BATS_AIC, "\n")
```

```
## The AIC of TBAST model equale to:
```

II-2-3. Forecasting Prophet model

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

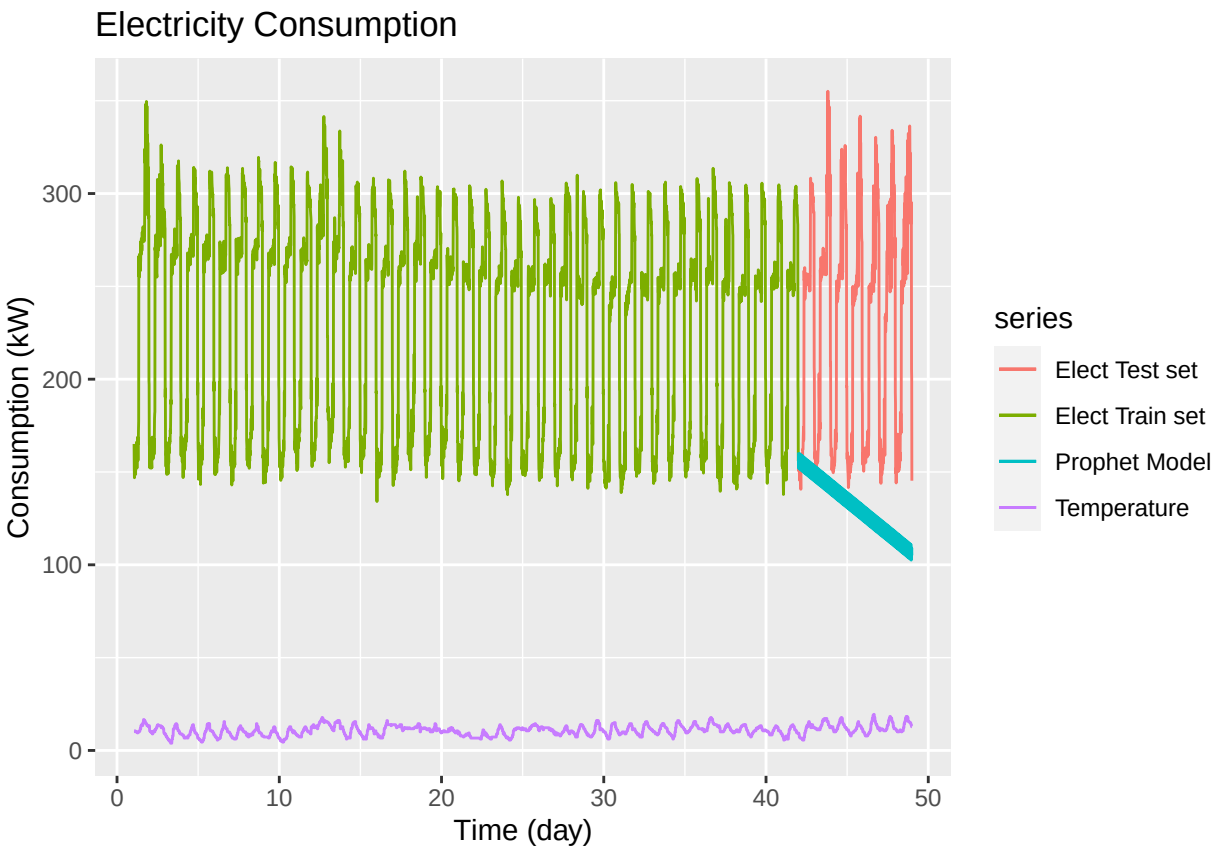
First, we should prepare a df for the prophet function.

```
ds = Elect_data$Timestamp[1:3931]
y = Elect_data$`Power_(kW)`[1:3931]
xreg = Elect_data$`Temp_(C°)`[1:3931]
Prophet_df <- data.frame(ds=ds, "y"=y, "xreg"=xreg)
```

```
Elect_WOT_D_2S_Prophet <- prophet(Prophet_df, yearly.seasonality = FALSE, weekly.seasonality = 672, daily.seasor
```

```
future <- make_future_dataframe(Elect_WOT_D_2S_Prophet, periods = 672)

Elect_WOT_D_2S_Prophet_forcast <- predict(Elect_WOT_D_2S_Prophet, future)
```

```
autoplot(Elect_D.ts.train[,"Power_(kW)"], series="Elect Train set") +
  autolayer(Elect_D.ts.test[,"Power_(kW)"], series='Elect Test set') +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series='Temperature') +
  autolayer(ts(Elect_WOT_D_2S_Prophet_forcast$yhat[3932:4603], frequency =96, start=c(42,1)), series='Prophet Mo
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```




```
Elect_WOT_D_2S_Prophet_RMSE <- sqrt(mean((as.numeric(Elect_WOT_D_2S_Prophet_forcast$yhat[3932:4603]) - as.numeri
Elect_WOT_D_2S_Prophet_AIC = Elect_WOT_D_2S_Prophet$aicc

cat("The RMSE of TBAST model equale to:", Elect_WOT_D_2S_Prophet_RMSE, "\n")
```

```
## The RMSE of TBAST model equale to: 119.1687
```

```
cat("The AIC of TBAST model equale to:", Elect_WOT_D_2S_Prophet_AIC, "\n")
```

```
## The AIC of TBAST model equale to:
```

III- Best Models' selection

III-1- Univariate Best model

```
Univ_best_model <- data.frame(
  Model = c("Simple Seasonality Model", "Double Seasonality Model"),
  Train_RMSE = c(9.677965,6.933779),
  Test_RMSE = c(Elect_WOOT_D_Arima_BC_RMSE, Elect_WOOT_D_DSHW_RMSE),
  Ljung_Box_test = c(0.01107,"8.156e-05"),
```

```
  AICc = c(Elect_WOOT_D_Arima_BC_AIC,"-")

)

Univ_best_model
```

```
##                       Model Train_RMSE Test_RMSE Ljung_Box_test             AICc
## 1 Simple Seasonality Model   9.677965 15.882338        0.01107 2955.70128856105
## 2 Double Seasonality Model   6.933779  9.798622      8.156e-05                -
```

From what we have got as models for the Electricity consumption (Kwh) time series without Temperature, we ca say that the best model to forecast a daily electricity consumption is the Holt Winters model with Double seasonality because this model doesn't show a lot of over fitting problem (RMSE on test set is not so high from the RMSE on train set). The problem with this model is that the residuals are independent white noise .

If we chose the SARIMA Model (5,1,8)(0,1,1)[96], the residuals are white noise at 90% (alpha=10%) but un over fitting probel is present.

The principal criteria to chose is to avoid the over fitting problem.

```
Elect_WOOT_D_Chosen_Model <- dshw(Elect_WOOT_D[,"Power_(kW)"], period1=96, period2 = 672, h=672, lambda = 'auto'
```

III-1- Bivariate Best model

```
Biv_best_model <- data.frame(
  Model = c("Simple Seasonality Model", "Double Seasonality Model"),
  Train_RMSE = c(7.003965,"-"),
  Test_RMSE = c(Elect_WOT_D_Arima_BC_RMSE, Elect_WOT_D_2S_BATS_RMSE),
  Ljung_Box_test = c(0.01107,"1.49e-05"),
  AICc = c(Elect_WOT_D_Arima_BC_AIC,"46513.68")
)

Biv_best_model
```

```
##                       Model Train_RMSE Test_RMSE Ljung_Box_test             AICc
## 1 Simple Seasonality Model   7.003965  15.40838        0.01107 2958.87484868275
## 2 Double Seasonality Model          -  17.97260       1.49e-05         46513.68
```

For the forecast with covariate, the best model is a SARIMA (5,1,12)(0,1,1)[96] because the RMSE on test set is the lowest and the Ljung_Box_test P-value is significant at 90% (alpha = 10%). Also the AICc of this SARIMA Model is the lowest comparing with all others models. But this model seems having a small problem of overfitting because Test_RMSE is high than Train_RMSE.

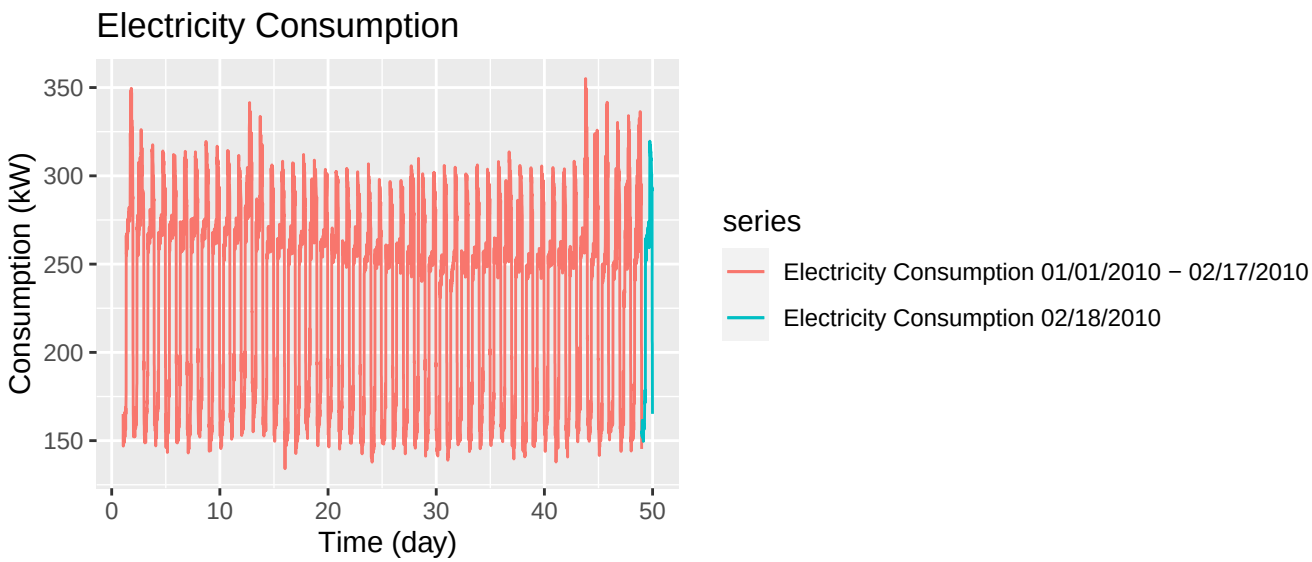
```
Elect_WOT_D_Chosen_Model <- Arima(Elect_WOOT_D[,"Power_(kW)"], order=c(5,1,12),  seasonal=c(0,1,1), xreg= Elect_
```

IV- Forecasting the february 18th Electricity consumption

IV-1- Forecasting the february 18th Electricity consumption Without Temperture

```
#Forecasting 02/18/2010 electricity consumption

Elect_WOOT_D_Chosen_Model_forecast = forecast :: forecast(Elect_WOOT_D_Chosen_Model, h = 96)

#Plotting prediction results
autoplot(Elect_WOOT_D[,"Power_(kW)"], series="Electricity Consumption 01/01/2010 - 02/17/2010") +
```

```
  autolayer(ts(Elect_WOOT_D_Chosen_Model_forecast$mean, frequency=96, start=c(49,1)), series='Electricity Consum
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```

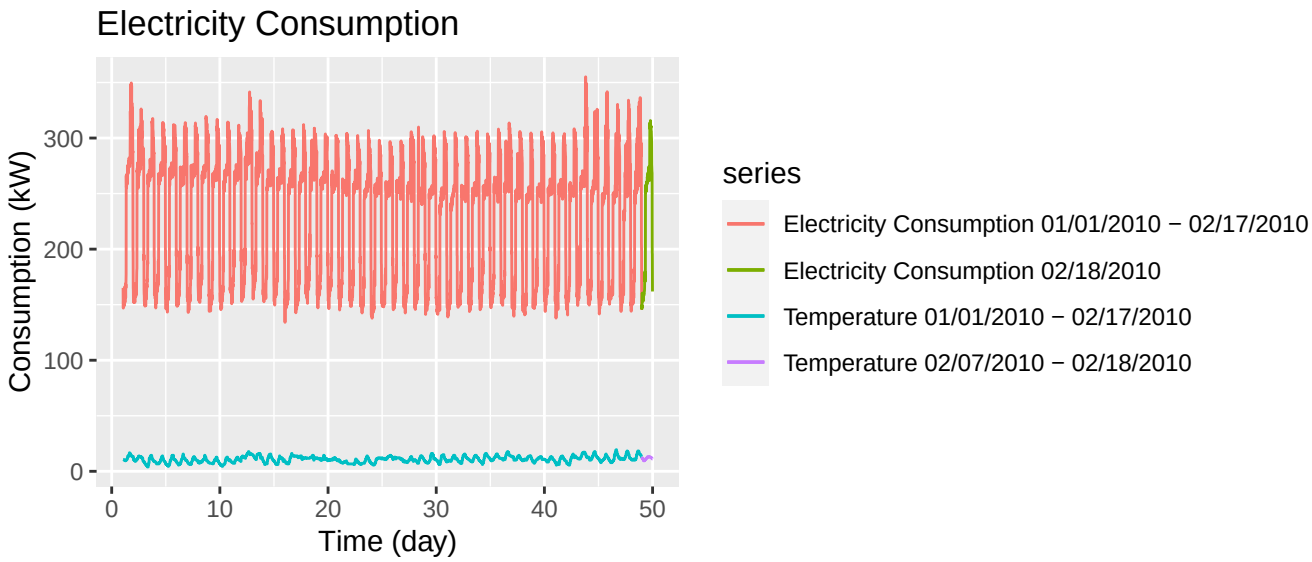


IV-2- Forecasting the february 18th Electricity consumption With Temperture

```
# February 18th Temperature for prediction

Temp_Pred <- ts(window(Elect[,"Temp_(C°)"], start = "2010-02-18 00:00:00", end = "2010-02-18 23:45:00"), frequer


#Forecasting 02/18/2010 electricity consumption
Elect_WOT_D_Chosen_Model_forecast = forecast :: forecast(Elect_WOT_D_Chosen_Model, xreg= Temp_Pred, h = 96)
```

```
## Warning in forecast.forecast_ARIMA(Elect_WOT_D_Chosen_Model, xreg = Temp_Pred, :
## xreg contains different column names from the xreg used in training. Please
## check that the regressors are in the same order.
```

```
#Plotting prediction results
autoplot(Elect_WOOT_D[,"Power_(kW)"], series="Electricity Consumption 01/01/2010 - 02/17/2010") +
  autolayer(Elect_WOOT_D[,"Temp_(C°)"], series="Temperature 01/01/2010 - 02/17/2010") +
  autolayer(Elect_WOT_D_Chosen_Model_forecast$mean, series='Electricity Consumption 02/18/2010')+
  autolayer(Temp_Pred, series="Temperature 02/07/2010 - 02/18/2010")+
  ggtitle ('Electricity Consumption') +
  xlab('Time (day)') +
  ylab('Consumption (kW)')
```



IV-3- Saving the forecast values

```
#Save prediction in Data Frame
Pred_DataFrame <- data.frame("18th_Elect_Pred_WOOT" = as.numeric(Elect_WOOT_D_Chosen_Model_forecast$mean) , "18t

#Save prediction results to xlsx file
write_xlsx(Pred_DataFrame, path  = "./Moussa_GRICHE.xlsx")
```

V- Conclusion

This time series represents some specificities such as high frequency and double seasonality (daily and weekly). To model this time series with correct predictions, I decided to choose the model with less overfitting instead of a model with white noise residuals.