Projet TimeSeries

Fatimetou Haidara

2023-01-30

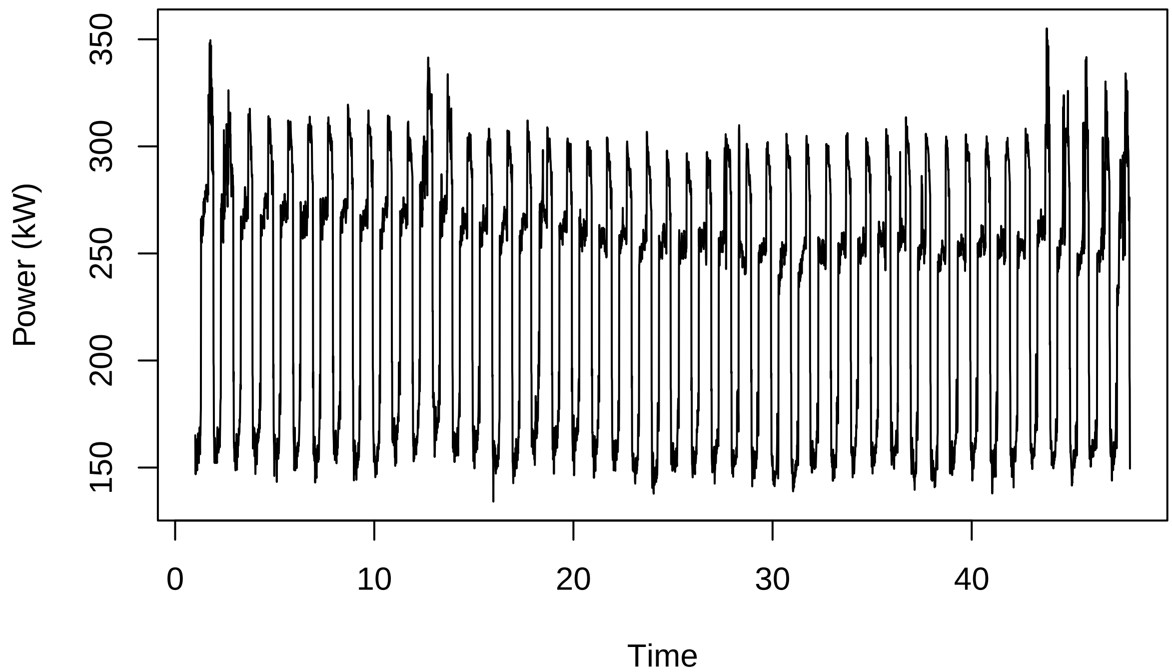
```r
#Loading the data
data <- read_excel("C:/Users/f_ati/Documents/Master2/Times series/Projet/Timeseries/Elec-train.xlsx")
data
```

```
## # A tibble: 4,603 x 3
##    Timestamp          `Power (kW)` `Temp (C°)`
##    <chr>                     <dbl>       <dbl>
##  1 40179.052083333336         165.        10.6
##  2 1/1/2010 1:30              152.        10.6
##  3 1/1/2010 1:45              147.        10.6
##  4 1/1/2010 2:00              154.        10.6
##  5 1/1/2010 2:15              154.        10.6
##  6 1/1/2010 2:30              159         10.6
##  7 1/1/2010 2:45              158.        10.6
##  8 1/1/2010 3:00              163.        10.6
##  9 1/1/2010 3:15              152.        10
## 10 1/1/2010 3:30              149.        10
## # ... with 4,593 more rows
```

```r
#These quantities are measured every 15 minutes, 1h =60/15.
#from 1/1/2010 1:15 to 2/16/2010 23:45.
elec_power<-ts(data[1:4507,2],start=c(1,2),freq=24*60/15)
tail(elec_power)
```

```
## Time Series:
## Start = c(47, 87)
## End = c(47, 92)
## Frequency = 96
##      Power (kW)
## [1,]      265.4
## [2,]      270.9
## [3,]      276.2
## [4,]      192.7
## [5,]      187.1
## [6,]      149.5
```

```r
plot(elec_power)
```




```r
mean(elec_power)
```

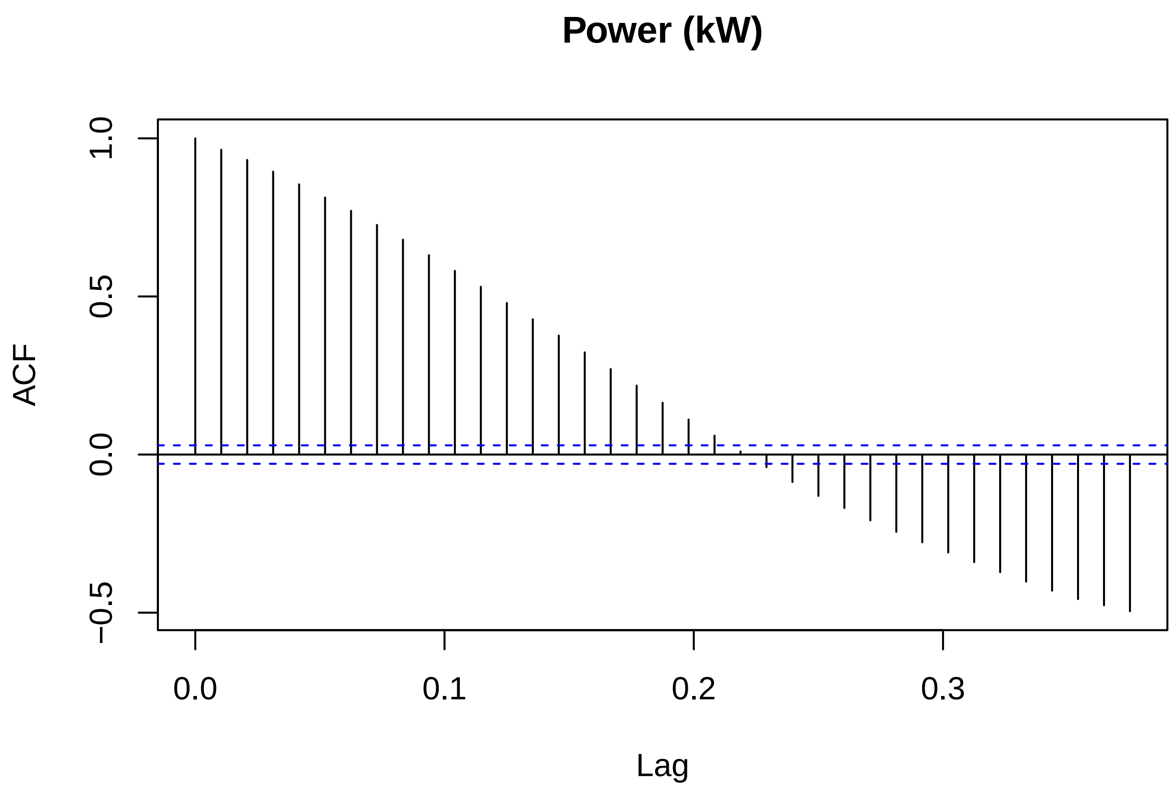
```
## [1] 231.5873
```

L'auto-corrélation nous montre qu'il y a un modèle saisionnier dans les données.

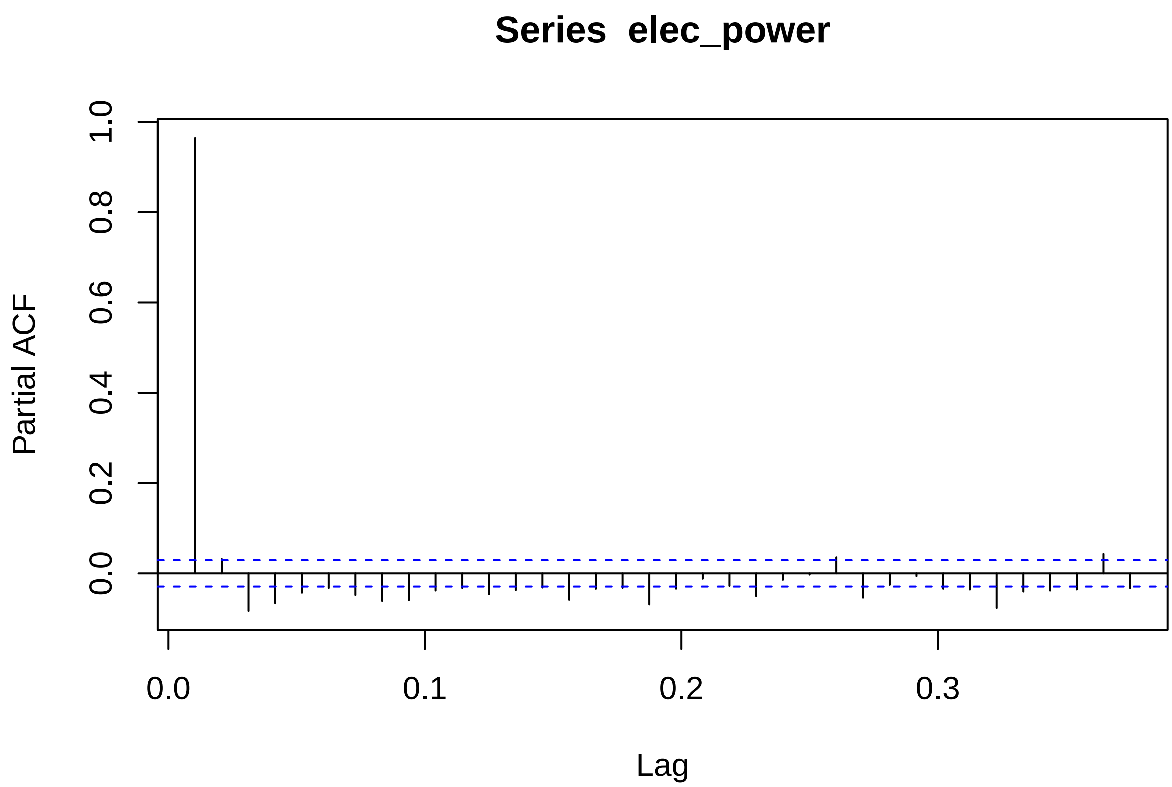
```r
tmp=acf(elec_power,type="cor",plot = FALSE)
tmp$acf[1:3,1,1]
```

```
## [1] 1.0000000 0.9641389 0.9317837
```

```r
plot(tmp)
```



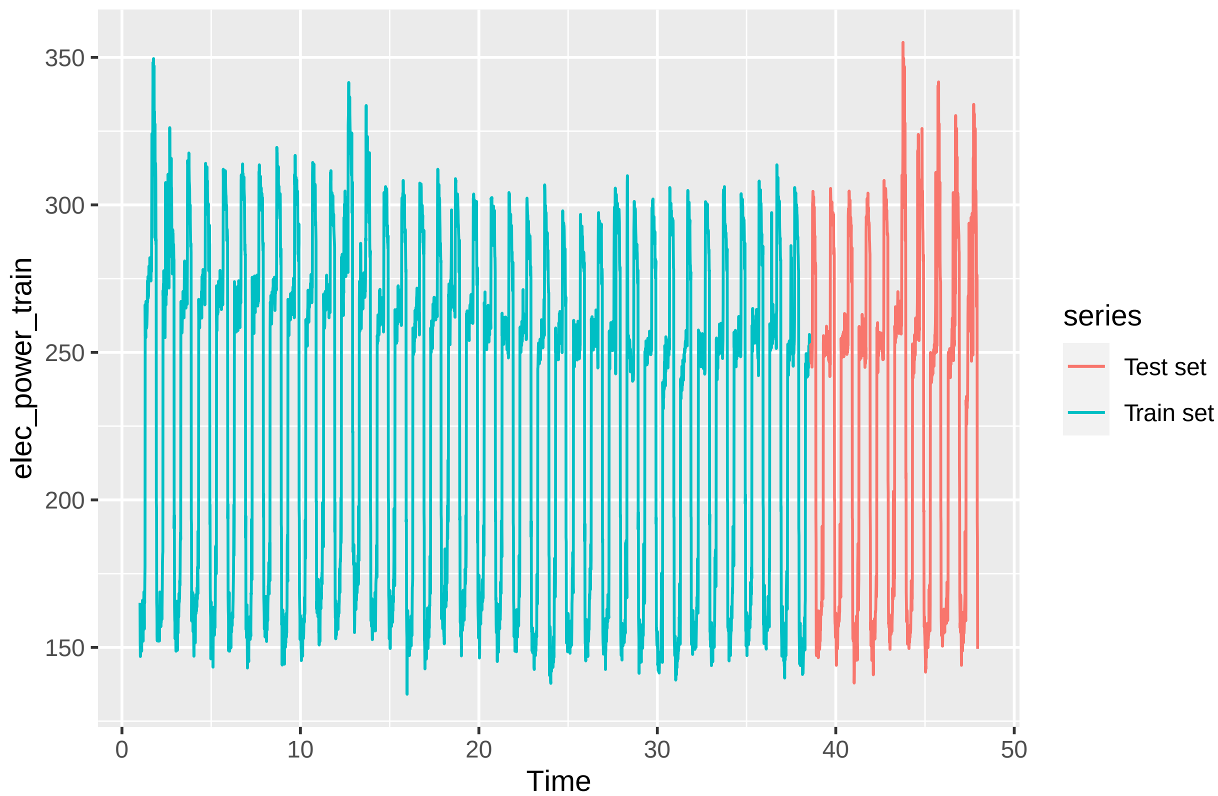

```
pacf(elec_power)
```



Le graphique saisonnier nous le confirme.

We split the serie into train and test

```
#We need to make two sets of data: the train one (80%) and the test one (20%) in order to evaluate the 
elec_power_train=head(elec_power,3607)
elec_power_test=tail(elec_power,900)
autoplot(elec_power_train,series="Train set")+
autolayer(elec_power_test,series='Test set')
```



########################### FORECAST ##################################

On commence par les modèles qui ne tiennent pas compte des tendances saisonnières.

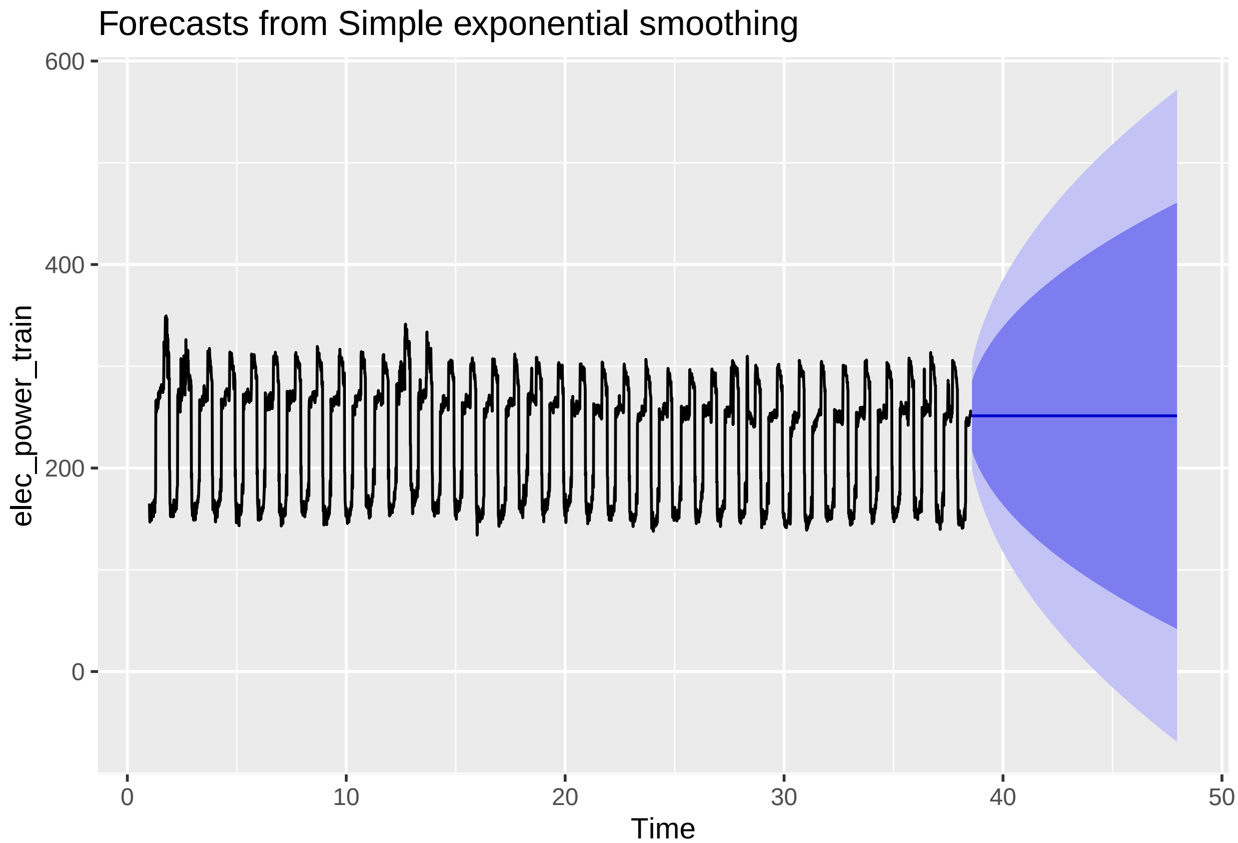
Lissage exponentiel simple (SES)

La technique de lissage exponentiel simple est utilisée pour les données qui n'ont pas de tendance ou de modèle saisonnier.

```
SES=ses(elec_power_train,h=900, alpha = .2)
round(accuracy(SES,elec_power_test),2)
```

```
##                   ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set    0.13 26.88 15.86  -1.41  7.69 1.99 0.83        NA
## Test set      -20.16 60.98 47.98 -16.59 25.95 6.03 0.96      4.81
```

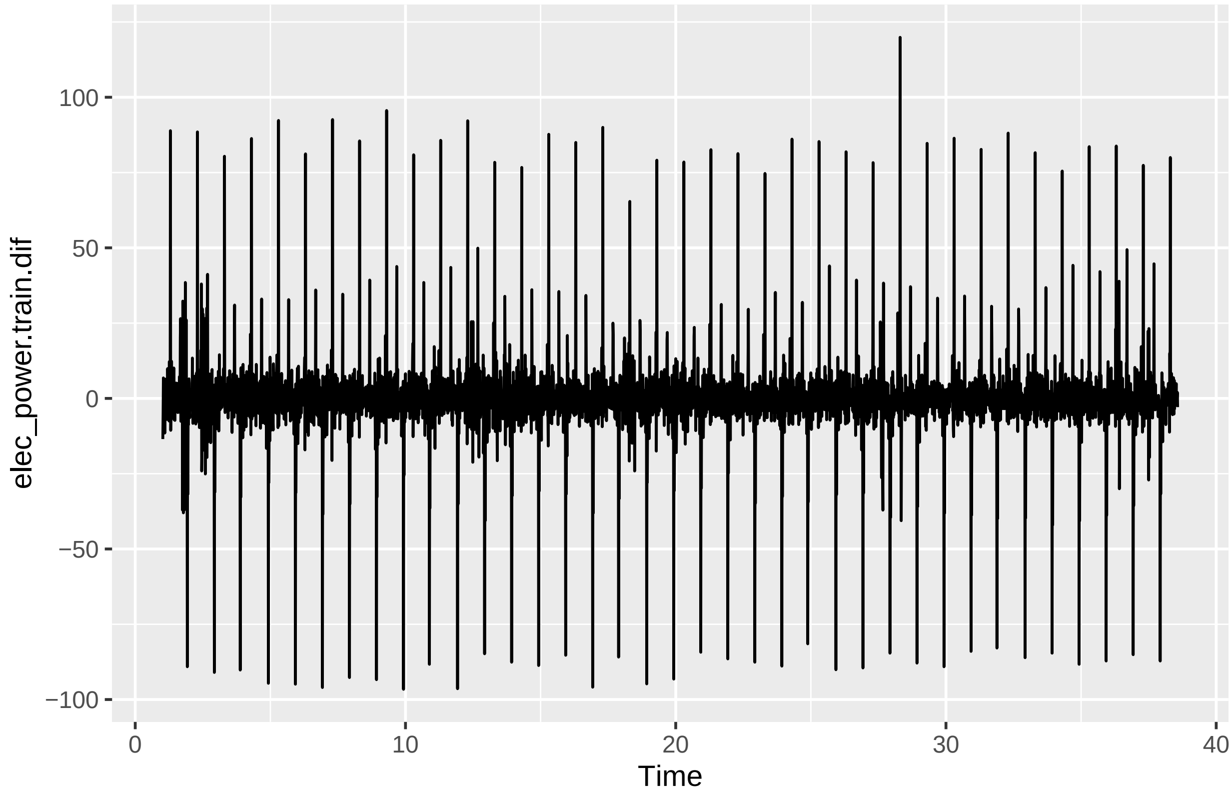
```
autoplot(SES)
```



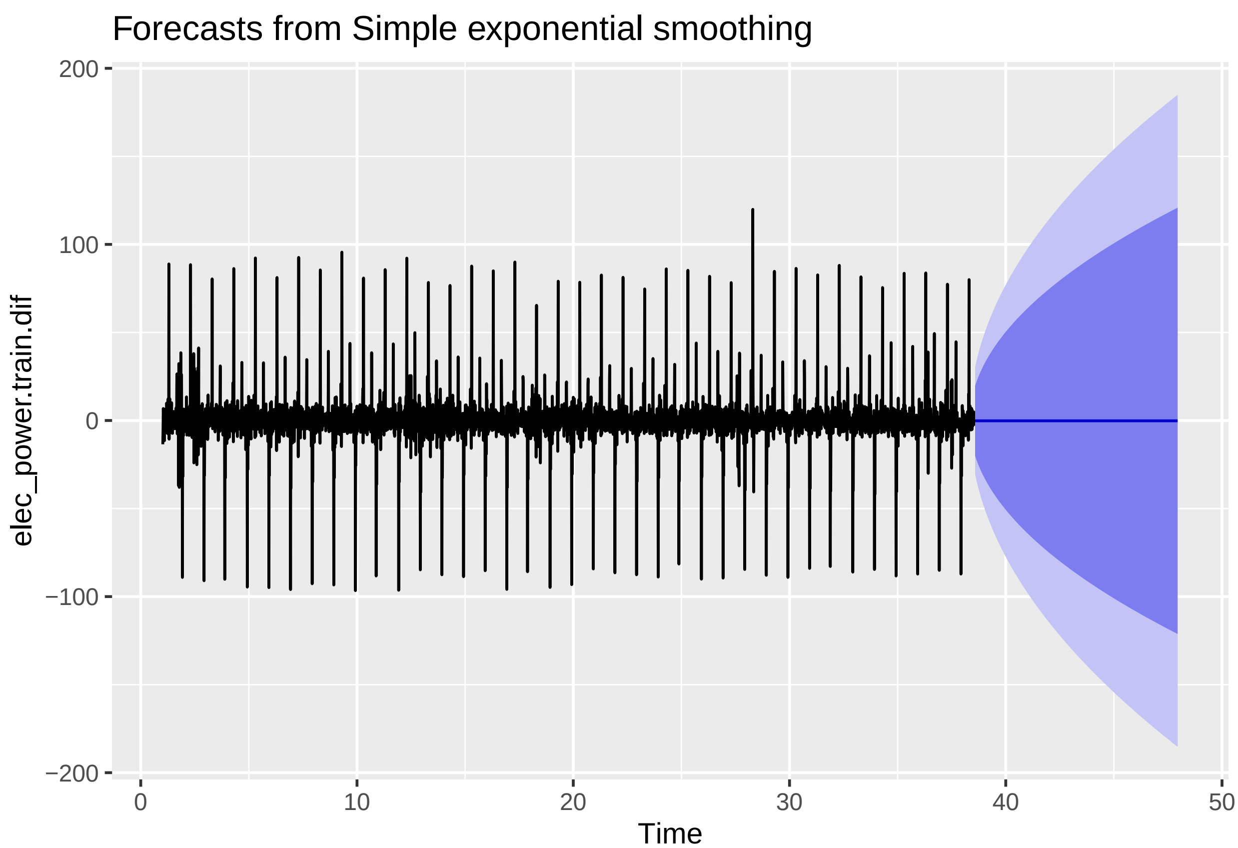
Nous pouvons remarquer qu'une estimation plate est projetée vers l'avenir par notre modèle de prévision. Par conséquent, nous pouvons dire que d'après les données, il ne capture pas la tendance actuelle.

Par conséquent, pour corriger cela, nous utiliserons la fonction diff() pour supprimer la tendance des données.

```
# removing the trend
elec_power.train.dif <- diff(elec_power_train)
autoplot(elec_power.train.dif)
```




```
# reapplying SES on the filtered data
SES.diff=ses(elec_power.train.dif,h=900 ,alpha = .2)

autoplot(SES.diff)
```



Afin de comprendre les performances de notre modèle, nous devons comparer nos prévisions avec notre ensemble de données de test.Puisque notre ensemble de données train a été différencié, nous devons également créer un ensemble de test différencié.

Ici, nous allons créer un ensemble de test différencié et ensuite comparer notre prévision .Le paramètre de lissage, « alpha », contrôle le poids accordé à l'observation la plus récente. Nous définissons la valeur de alpha entre 0,02 et 0,99 en utilisant la boucle. Nous essayons de comprendre quel niveau minimisera le test RMSE.

```
# removing trend from test set
elec_power.dif.test <- diff(elec_power_test)
accuracy(SES.diff, elec_power.dif.test)
```

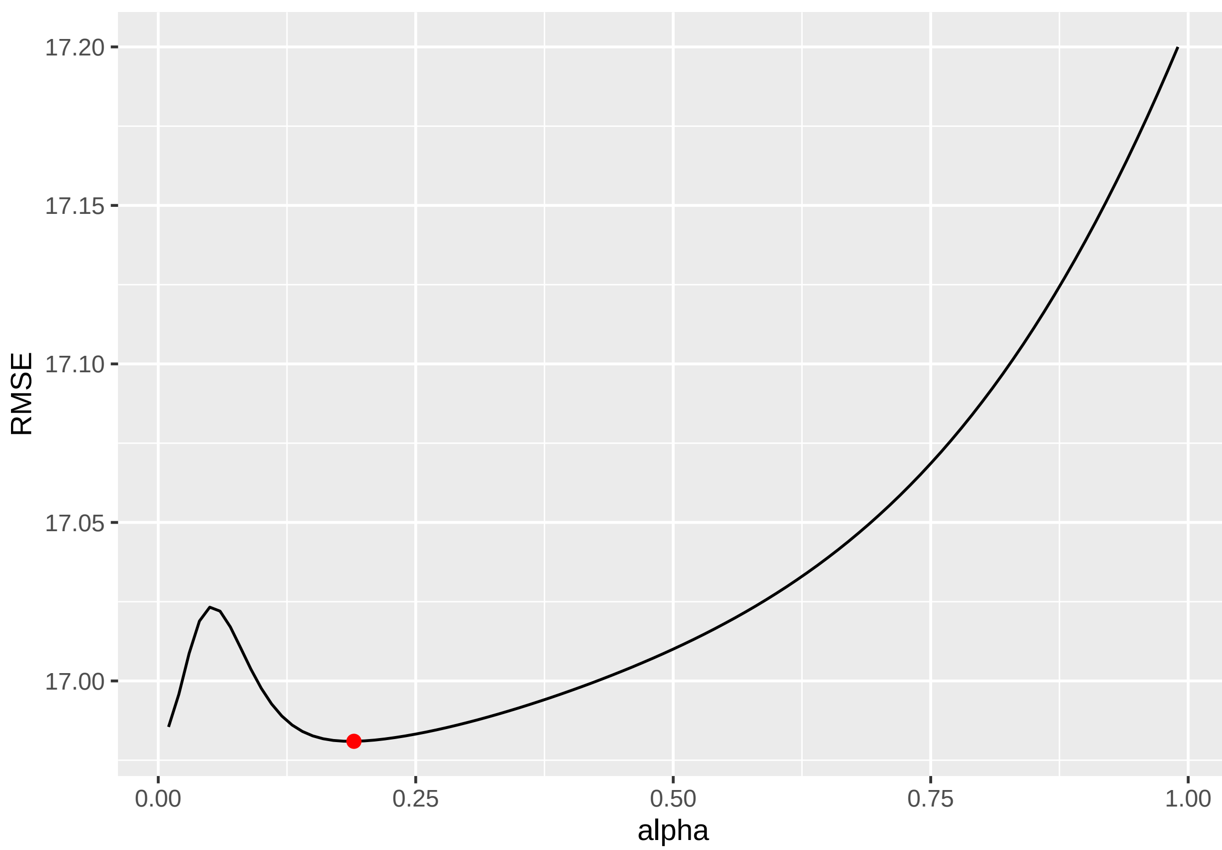
```
##                        ME     RMSE      MAE MPE MAPE     MASE       ACF1
## Training set 0.002796698 15.53155 8.312289 NaN  Inf 1.330100 -0.1568283
## Test set     0.068116291 16.98109 8.647432 Inf  Inf 1.383728 -0.1074126
##              Theil's U
## Training set        NA
## Test set           NaN
```

```
# comparing our model
alpha <- seq(.01, .99, by = .01)
RMSE <- NA
for(i in seq_along(alpha)) {
  fit <- ses(elec_power.train.dif, alpha = alpha[i],
             h = 900)
  RMSE[i] <- accuracy(fit,
```

```
                      elec_power.dif.test)[2,2]
}

# convert to a data frame and
# identify min alpha value
alpha.fit <- data_frame(alpha, RMSE)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
```

```
alpha.min <- filter(alpha.fit,
                    RMSE == min(RMSE))

# plot RMSE vs. alpha
ggplot(alpha.fit, aes(alpha, RMSE)) +
  geom_line() +
  geom_point(data = alpha.min,
             aes(alpha, RMSE),
             size = 2, color = "red")
```



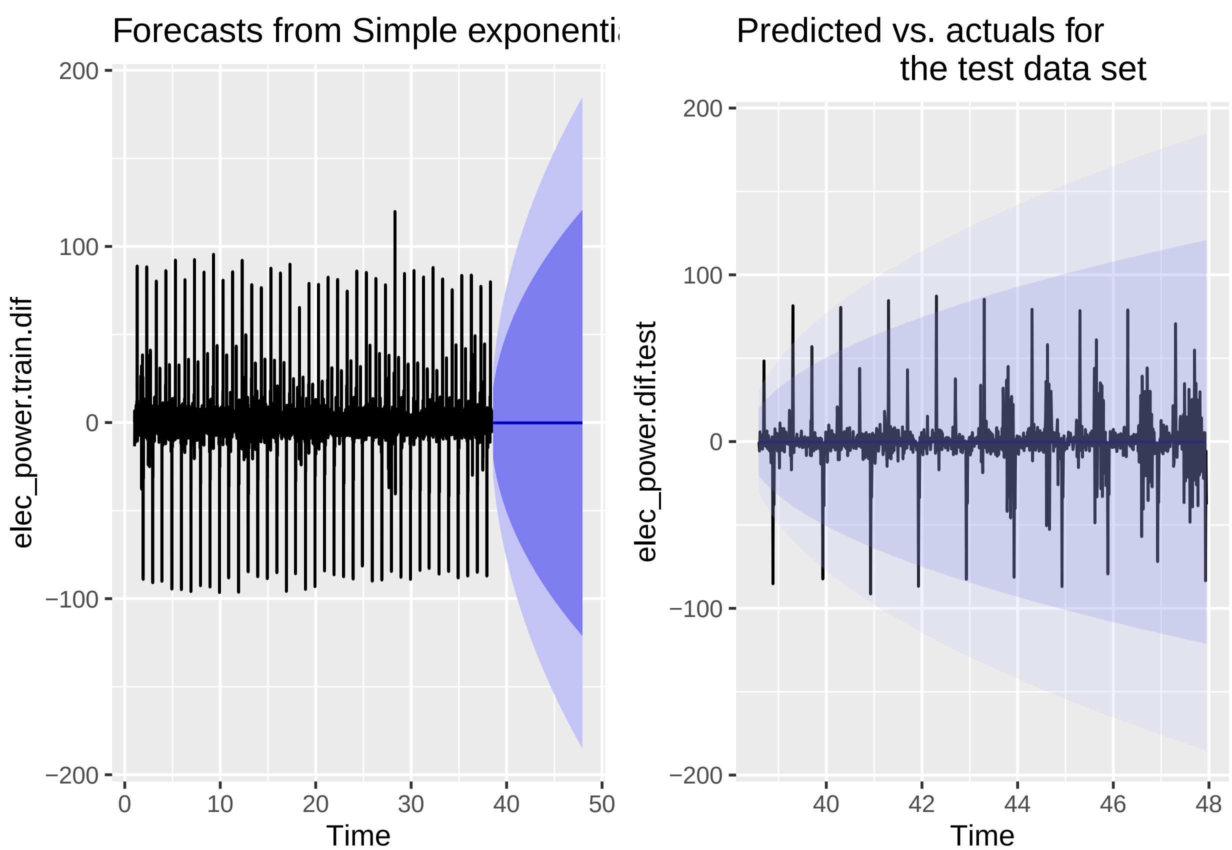
Nous remarquerons que environs 0,2 minimisera le plus.

Maintenant, nous allons essayer de réajuster notre modèle de prévision pour le SES avec alpha = 0,2.

```r
# refit model with alpha = .7
SES.opt=ses(elec_power.train.dif,h=900 ,alpha = .2)
round(accuracy(SES.opt,elec_power.dif.test),2)
```

```
##                ME  RMSE  MAE MPE MAPE MASE  ACF1 Theil's U
## Training set 0.00 15.53 8.31 NaN  Inf 1.33 -0.16        NA
## Test set     0.07 16.98 8.65 Inf  Inf 1.38 -0.11       NaN
```

```r
# plotting results
p1 <- autoplot(SES.opt) +
  theme(legend.position = "bottom")
p2 <- autoplot(elec_power.dif.test) +
  autolayer(SES.opt, alpha = .2) +
  ggtitle("Predicted vs. actuals for
                 the test data set")

gridExtra::grid.arrange(p1, p2,
                        nrow = 1)
```



L'intervalle de confiance prédit de notre modèle est beaucoup plus étroit.

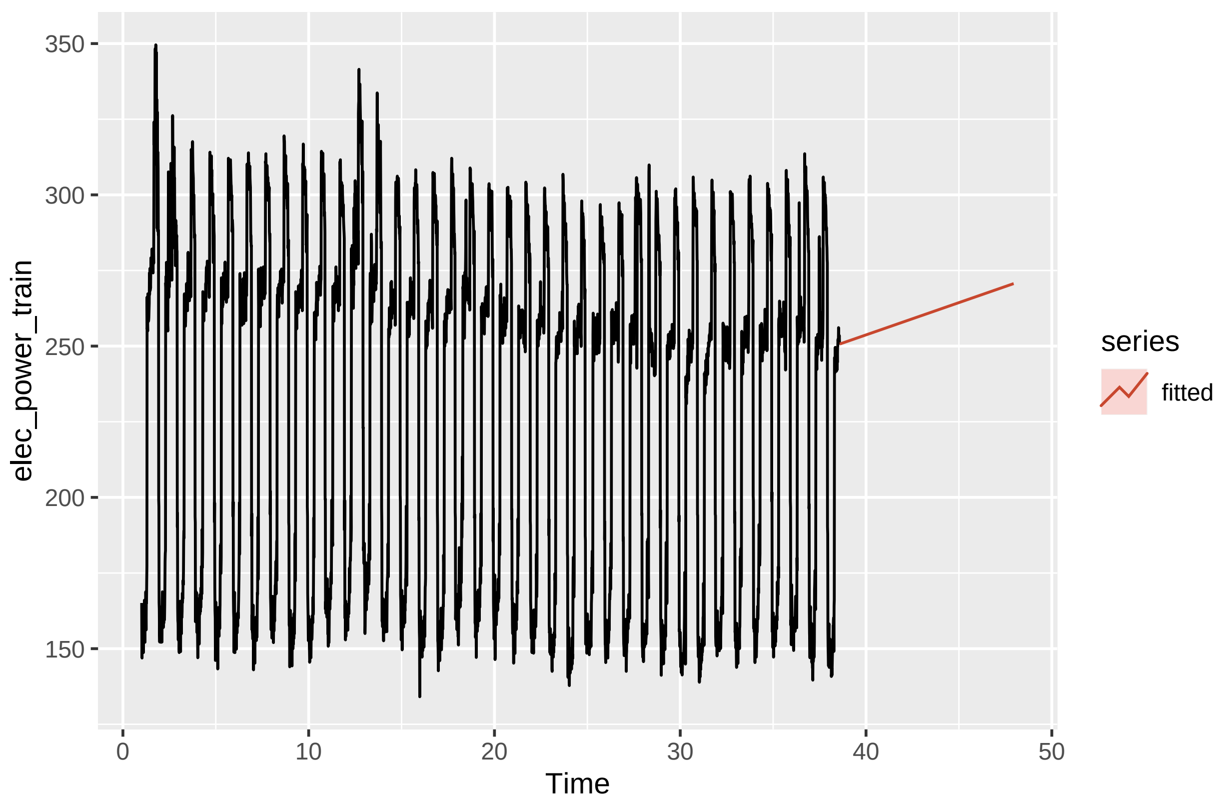
Méthode de Holt

Nous avons vu qu'en SES, nous avons dû supprimer les tendances à long terme pour améliorer le modèle. Mais dans la méthode de Holt, nous pouvons appliquer un lissage exponentiel tout en capturant les tendances dans les données. Cependat Il s'agit d'une technique qui fonctionne avec des données présentant une tendance mais pas de saisonnalité.

```r
# Forecasting with a Holt
HOLT=holt(elec_power_train,h=900,alpha=NULL,beta=NULL)
round(accuracy(HOLT,elec_power_test),2)
```

```
##                  ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set   0.00 14.89  6.89  -0.23  3.18 0.87 0.00        NA
## Test set     -29.51 64.57 49.54 -20.89 27.52 6.23 0.95      5.23
```

```r
autoplot(elec_power_train) + autolayer(HOLT,series='fitted',PI=FALSE)
```



Le modèle n'est pas adapté avec un RMSE de 64.57

Holt-Winter's Seasonal Method et Damping Method

cette methode est utilisée pour les données présentant à la fois des tendances et des tendances saisonnières. Cette méthode peut être implémentée soit en utilisant la structure additive, soit en utilisant la structure multiplicative en fonction de l'ensemble de données.

La méthode d'amortissement utilise le coefficient d'amortissement phi pour estimer de manière plus prudente les tendances prévues. La valeur de phi se situe entre 0 et 1. Si nous croyons que notre modèle additif et multiplicatif va être une ligne plate, alors il y a de fortes chances qu'il soit amorti.

Cependantla fréquence des données est trop élevée pour la fonction ets(). La suggestion de Rob Hyndman est de modéliser la saisonnalité en utilisant des termes de Fourier, et éventuellement en utilisant ARIMA pour les résidus.

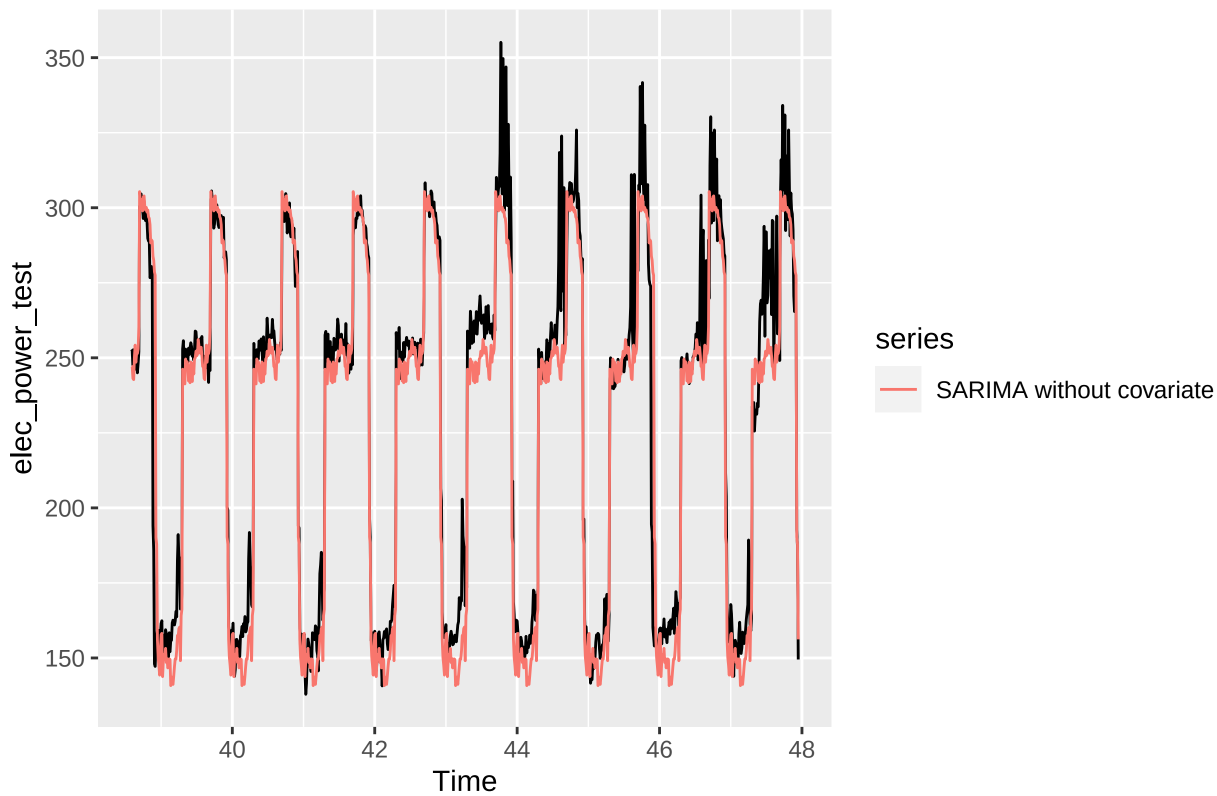
```
#Additive seasonal Holt-Winters
#fit1=hw(elec_power_train,  seasonal = "additive",h=900)

#Multiplicative seasonal Holt-Winters
#fit2 = hw(elec_power_train, seasonal='multiplicative',h=900)
```

```
#Damped additive seasonal Holt-Winters
#fit3 = hw(elec_power_train, seasonal='additive',h=900,damped=TRUE)

#Damped multiplicative seasonal Holt-Winters
#fit4 = hw(elec_power_train,seasonal='multiplicative',h=900,damped=TRUE)
```

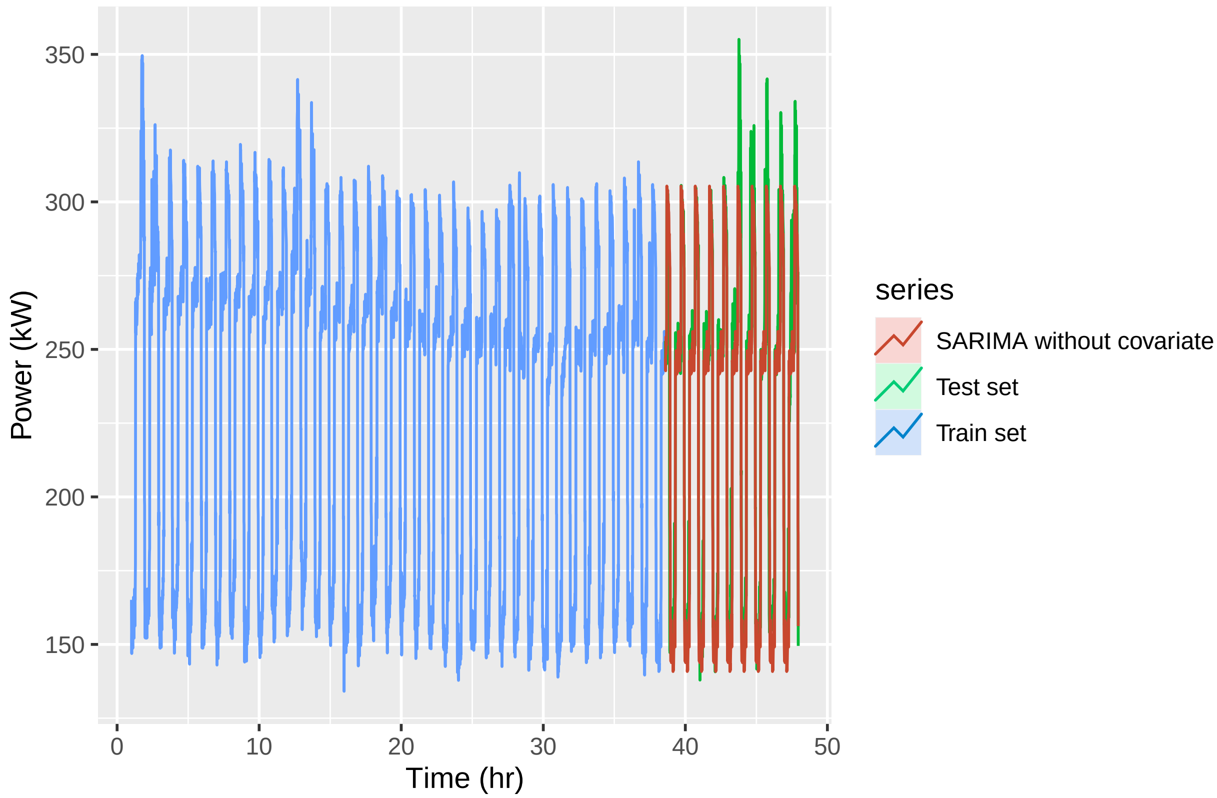
Pour l'instant le meilleur modèle est SES.opt , présentant l'erreur la plus faible avec un RMSE=16.98.

Forecasting with ARMA models

```
#SARIMA model.

elec_power_arima=auto.arima(elec_power_train)
prev_arima=forecast(elec_power_arima,h=900)
autoplot(elec_power_test)+
  autolayer(prev_arima$mean,series="SARIMA without covariate")
```




```
autoplot(elec_power_train,series="Train set") +
  autolayer(elec_power_test,series='Test set')+
  autolayer(prev_arima,series='SARIMA without covariate',PI=FALSE)+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```




```
round(accuracy(prev_arima, elec_power_test),2)
```

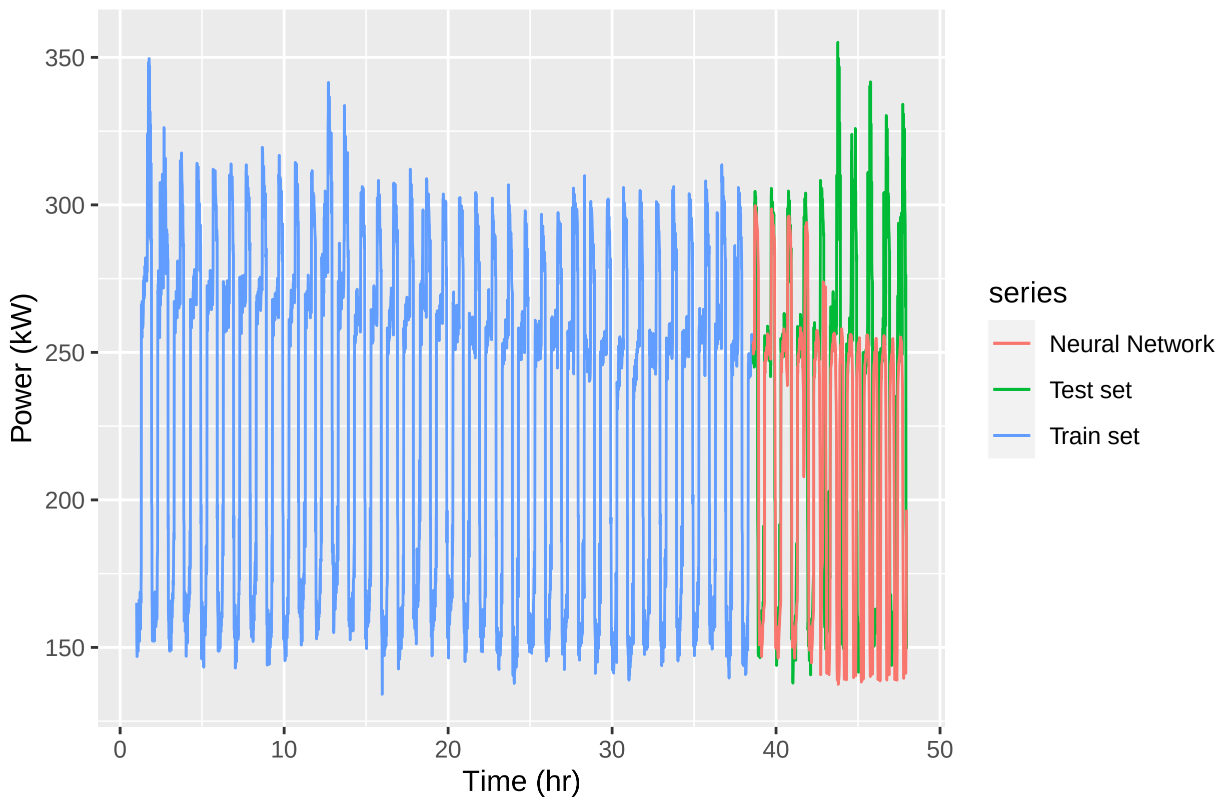
```
##                  ME  RMSE   MAE   MPE MAPE MASE ACF1 Theil's U
## Training set -0.10  9.37  5.78 -0.13 2.66 0.73 0.00        NA
## Test set      5.85 16.98 10.36  2.52 4.82 1.30 0.68      1.11
```

Les résultats sont un peu plus meilleurs.

Forecasting with Neural Network

We van automatically select the best NNAR(p,P,k)T:

```
elec_power_nn = nnetar(elec_power_train)
pred_elec_power_nn = forecast(elec_power_nn, h = 900)
autoplot(elec_power_train,series="Train set") +
  autolayer(elec_power_test,series='Test set')+
  autolayer(pred_elec_power_nn$mean,series='Neural Network')+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```




```
round(accuracy(pred_elec_power_nn, elec_power_test),2)
```

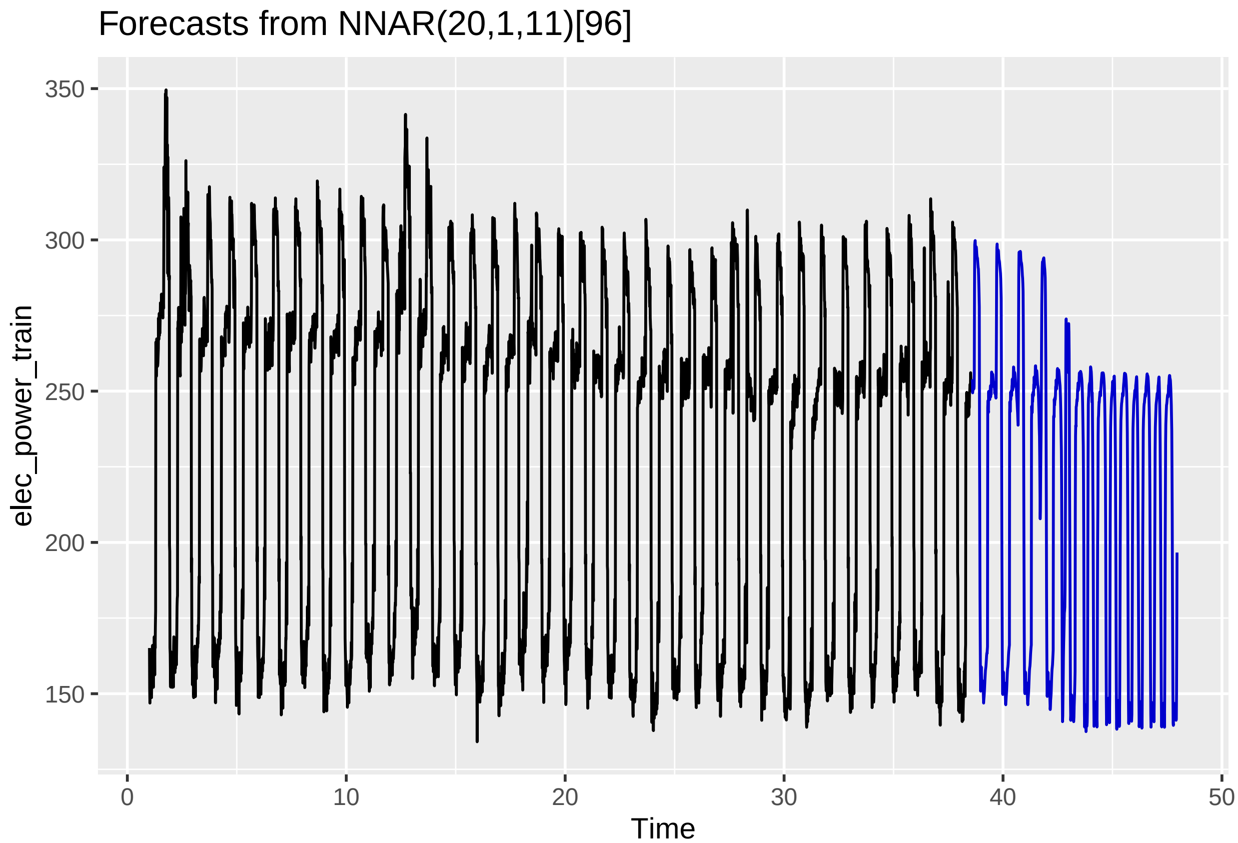
```
##                 ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set -0.04  6.94  4.38 -0.14  2.00 0.55 0.12        NA
## Test set     16.35 66.76 41.88  2.58 18.56 5.27 0.96      3.86
```

On a pas pas de meilleur resultat avec le Neural Network.

```
print(elec_power_nn)
```

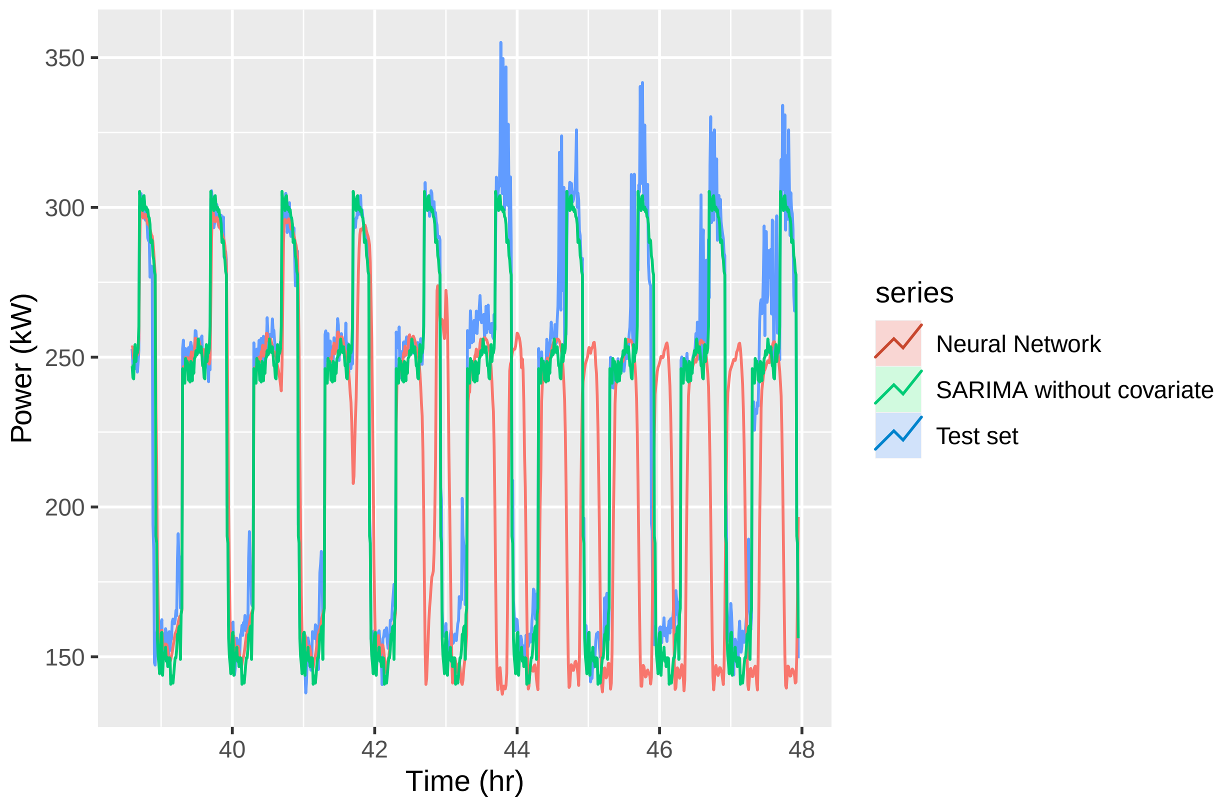
```
## Series: elec_power_train
## Model:  NNAR(20,1,11)[96]
## Call:   nnetar(y = elec_power_train)
##
## Average of 20 networks, each of which is
## a 21-11-1 network with 254 weights
## options were - linear output units
##
## sigma^2 estimated as 48.1
```

```
pred_elec_power_nn %>% forecast(h=900) %>% autoplot()
```



Les prévisions sont moins efficaces qu'avec les modèles SARIMA.

```
autoplot(elec_power_test,series='Test set') +

  autolayer(pred_elec_power_nn$mean,series='Neural Network')+
  autolayer(prev_arima,series='SARIMA without covariate',PI=FALSE)+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```

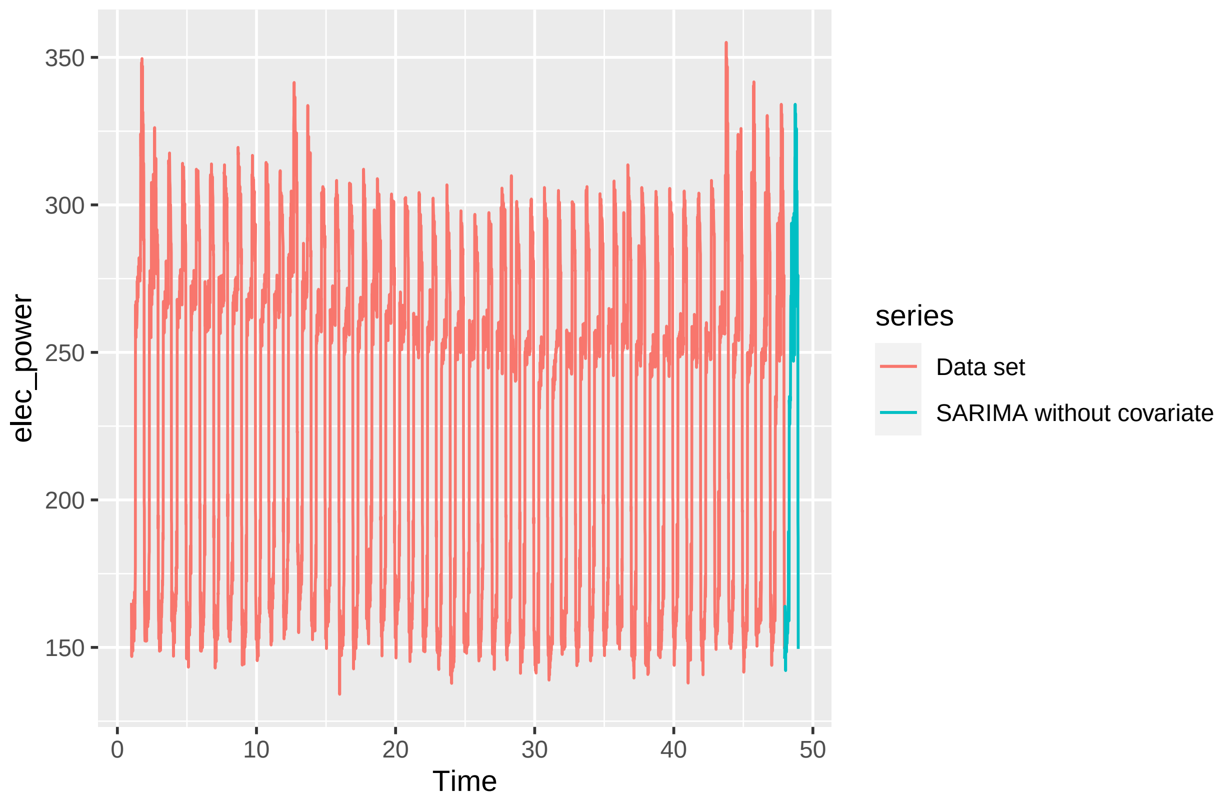


On conclue que le meilleur modèle est SARIMA , présentant l'erreur la plus faible avec un RMSE=16.98.

Nous allons maintenant prévoir la consommation d'électricité (kW) pour le 17/02/2010 avec 96 observations.

```
#Forecast 17/02/2010

elec_power_arima_pred=auto.arima(elec_power)
prev_arima_pred=forecast(elec_power_arima_pred,h=96)
autoplot(elec_power,series="Data set")+
  autolayer(prev_arima_pred$mean,series="SARIMA without covariate",PI=FALSE)
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, :
## Ignoring unknown parameters: ‘PI‘
```



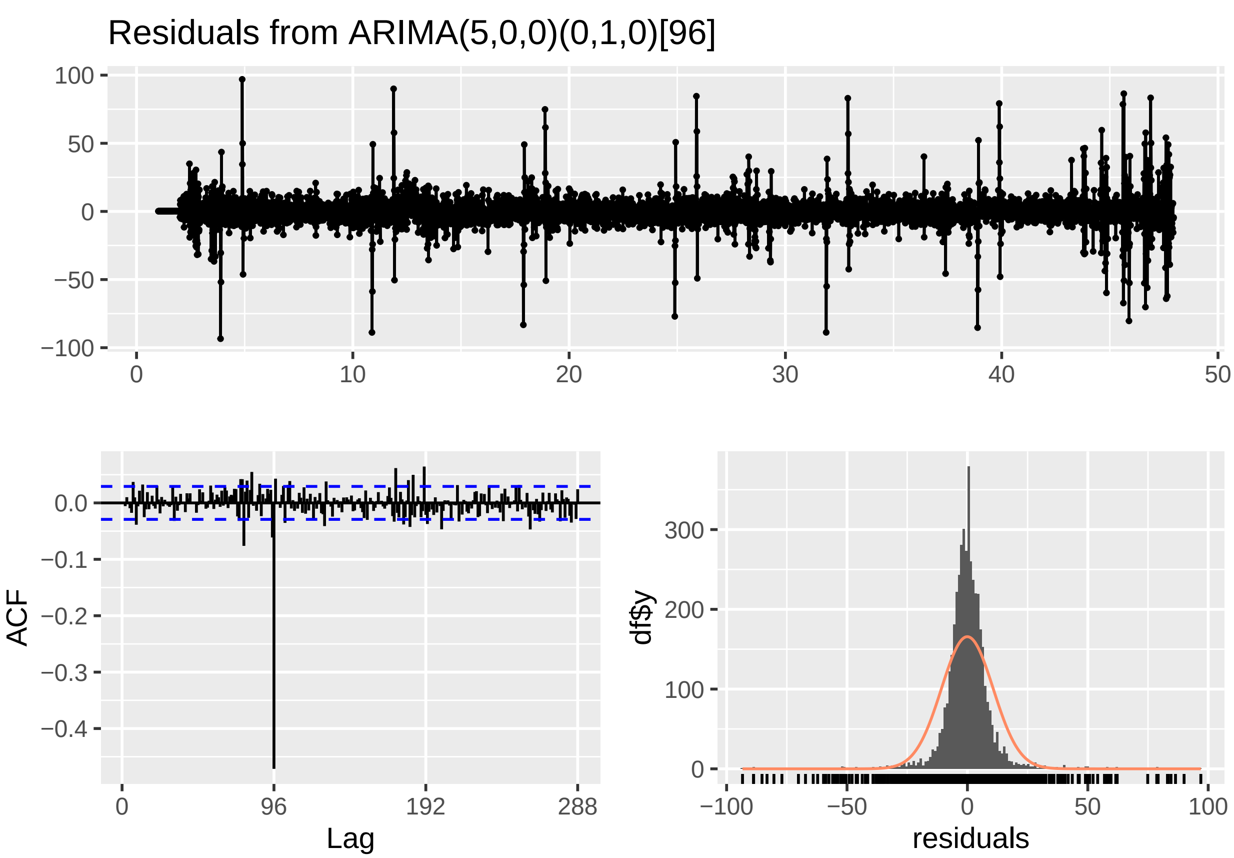

```
# Results
Pred = print(prev_arima_pred)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47.95833       146.1747 132.2315 160.1180 124.8504 167.4991
## 47.96875       146.6443 129.6879 163.6007 120.7117 172.5769
## 47.97917       150.0067 131.4245 168.5888 121.5877 178.4257
## 47.98958       149.2703 129.0698 169.4707 118.3763 180.1642
## 48.00000       163.9390 143.4779 184.4001 132.6464 195.2315
## 48.01042       161.1339 140.4530 181.8149 129.5051 192.7627
## 48.02083       155.7695 134.9295 176.6094 123.8975 187.6414
## 48.03125       149.2982 128.4246 170.1718 117.3748 181.2216
## 48.04167       142.1431 121.2090 163.0773 110.1272 174.1591
## 48.05208       152.6909 131.7266 173.6553 120.6288 184.7531
## 48.06250       154.8982 133.9214 175.8750 122.8170 186.9794
## 48.07292       154.1044 133.1115 175.0972 121.9986 186.2101
## 48.08333       153.7430 132.7447 174.7412 121.6290 185.8570
## 48.09375       148.5244 127.5227 169.5261 116.4051 180.6437
## 48.10417       148.4431 127.4383 169.4479 116.3190 180.5672
## 48.11458       157.6337 136.6279 178.6395 125.5081 189.7593
## 48.12500       161.6748 140.6681 182.6815 129.5478 193.8018
## 48.13542       155.7363 134.7290 176.7436 123.6084 187.8642
## 48.14583       151.3711 130.3636 172.3787 119.2429 183.4994
## 48.15625       153.2953 132.2876 174.3031 121.1668 185.4239
## 48.16667       158.5241 137.5162 179.5319 126.3953 190.6528
## 48.17708       157.8383 136.8303 178.8462 125.7094 189.9671
```

```
## 48.18750       154.7519 133.7439 175.7599 122.6230 186.8808
## 48.19792       156.2643 135.2563 177.2723 124.1353 188.3932
## 48.20833       159.3707 138.3627 180.3787 127.2417 191.4997
## 48.21875       162.8779 141.8698 183.8859 130.7489 195.0069
## 48.22917       165.3830 144.3750 186.3910 133.2540 197.5120
## 48.23958       167.6863 146.6782 188.6943 135.5573 199.8153
## 48.25000       189.2897 168.2817 210.2978 157.1607 221.4188
## 48.26042       174.3919 153.3839 195.3999 142.2629 206.5209
## 48.27083       161.4936 140.4856 182.5016 129.3646 193.6226
## 48.28125       158.9952 137.9872 180.0032 126.8662 191.1242
## 48.29167       163.0962 142.0881 184.1042 130.9672 195.2252
## 48.30208       233.7970 212.7890 254.8051 201.6680 265.9261
## 48.31250       232.0977 211.0897 253.1058 199.9687 264.2268
## 48.32292       235.0982 214.0902 256.1062 202.9692 267.2272
## 48.33333       225.5986 204.5906 246.6067 193.4696 257.7276
## 48.34375       231.2989 210.2909 252.3070 199.1699 263.4279
## 48.35417       232.5992 211.5911 253.6072 200.4701 264.7282
## 48.36458       231.1994 210.1913 252.2074 199.0703 263.3284
## 48.37500       234.8995 213.8915 255.9075 202.7705 267.0285
## 48.38542       233.6996 212.6916 254.7076 201.5706 265.8286
## 48.39583       246.7997 225.7917 267.8077 214.6707 278.9287
## 48.40625       261.9998 240.9917 283.0078 229.8707 294.1288
## 48.41667       267.9998 246.9918 289.0079 235.8708 300.1288
## 48.42708       269.1999 248.1918 290.2079 237.0708 301.3289
## 48.43750       264.4999 243.4919 285.5079 232.3709 296.6289
## 48.44792       267.5999 246.5919 288.6079 235.4709 299.7289
## 48.45833       277.4999 256.4919 298.5080 245.3709 309.6289
## 48.46875       293.7999 272.7919 314.8080 261.6709 325.9290
## 48.47917       257.2000 236.1919 278.2080 225.0709 289.3290
## 48.48958       292.0000 270.9919 313.0080 259.8710 324.1290
## 48.50000       291.9000 270.8919 312.9080 259.7710 324.0290
## 48.51042       268.3000 247.2919 289.3080 236.1710 300.4290
## 48.52083       270.8000 249.7920 291.8080 238.6710 302.9290
## 48.53125       285.6000 264.5920 306.6080 253.4710 317.7290
## 48.54167       276.6000 255.5920 297.6080 244.4710 308.7290
## 48.55208       279.0000 257.9920 300.0080 246.8710 311.1290
## 48.56250       286.0000 264.9920 307.0080 253.8710 318.1290
## 48.57292       264.4000 243.3920 285.4080 232.2710 296.5290
## 48.58333       295.8000 274.7920 316.8080 263.6710 327.9290
## 48.59375       295.4000 274.3920 316.4080 263.2710 327.5290
## 48.60417       247.0000 225.9920 268.0080 214.8710 279.1290
## 48.61458       263.3000 242.2920 284.3080 231.1710 295.4290
## 48.62500       269.8000 248.7920 290.8080 237.6710 301.9290
## 48.63542       271.9000 250.8920 292.9080 239.7710 304.0290
## 48.64583       297.2000 276.1920 318.2080 265.0710 329.3290
## 48.65625       258.0000 236.9920 279.0080 225.8710 290.1290
## 48.66667       269.3000 248.2920 290.3080 237.1710 301.4290
## 48.67708       272.3000 251.2920 293.3080 240.1710 304.4290
## 48.68750       249.1000 228.0920 270.1080 216.9710 281.2290
## 48.69792       304.0000 282.9920 325.0080 271.8710 336.1290
## 48.70833       316.0000 294.9920 337.0080 283.8710 348.1290
## 48.71875       299.2000 278.1920 320.2080 267.0710 331.3290
## 48.72917       334.1000 313.0920 355.1080 301.9710 366.2290
## 48.73958       308.1000 287.0920 329.1080 275.9710 340.2290
```

```
## 48.75000        305.9000 284.8920 326.9080 273.7710 338.0290
## 48.76042        330.9000 309.8920 351.9080 298.7710 363.0290
## 48.77083        292.4000 271.3920 313.4080 260.2710 324.5290
## 48.78125        317.3000 296.2920 338.3080 285.1710 349.4290
## 48.79167        316.0000 294.9920 337.0080 283.8710 348.1290
## 48.80208        296.0000 274.9920 317.0080 263.8710 328.1290
## 48.81250        325.9000 304.8920 346.9080 293.7710 358.0290
## 48.82292        308.7000 287.6920 329.7080 276.5710 340.8290
## 48.83333        290.6000 269.5920 311.6080 258.4710 322.7290
## 48.84375        304.9000 283.8920 325.9080 272.7710 337.0290
## 48.85417        299.2000 278.1920 320.2080 267.0710 331.3290
## 48.86458        297.9000 276.8920 318.9080 265.7710 330.0290
## 48.87500        292.7000 271.6920 313.7080 260.5710 324.8290
## 48.88542        270.6000 249.5920 291.6080 238.4710 302.7290
## 48.89583        265.4000 244.3920 286.4080 233.2710 297.5290
## 48.90625        270.9000 249.8920 291.9080 238.7710 303.0290
## 48.91667        276.2000 255.1920 297.2080 244.0710 308.3290
## 48.92708        192.7000 171.6920 213.7080 160.5710 224.8290
## 48.93750        187.1000 166.0920 208.1080 154.9710 219.2290
## 48.94792        149.5000 128.4920 170.5080 117.3710 181.6290
```

```
#Checking model

checkresiduals(prev_arima_pred,test="LB",plot=TRUE)
```



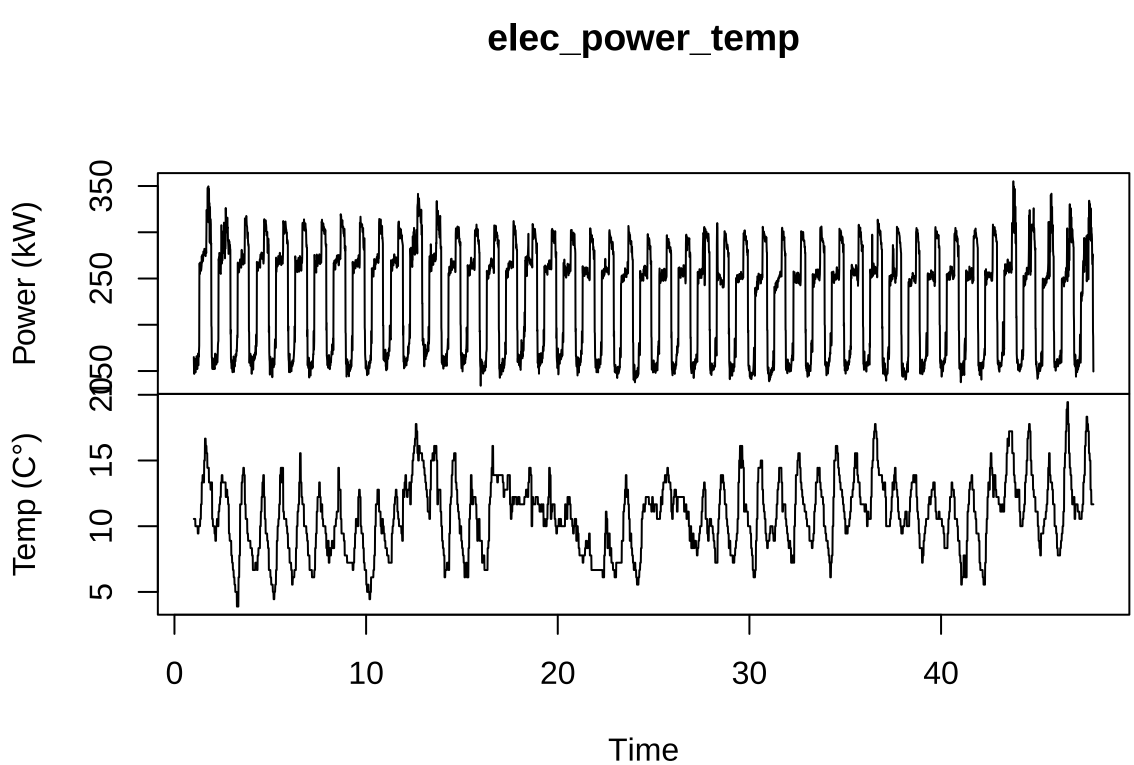

```
##
```

```
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,0)[96]
## Q* = 1421.1, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

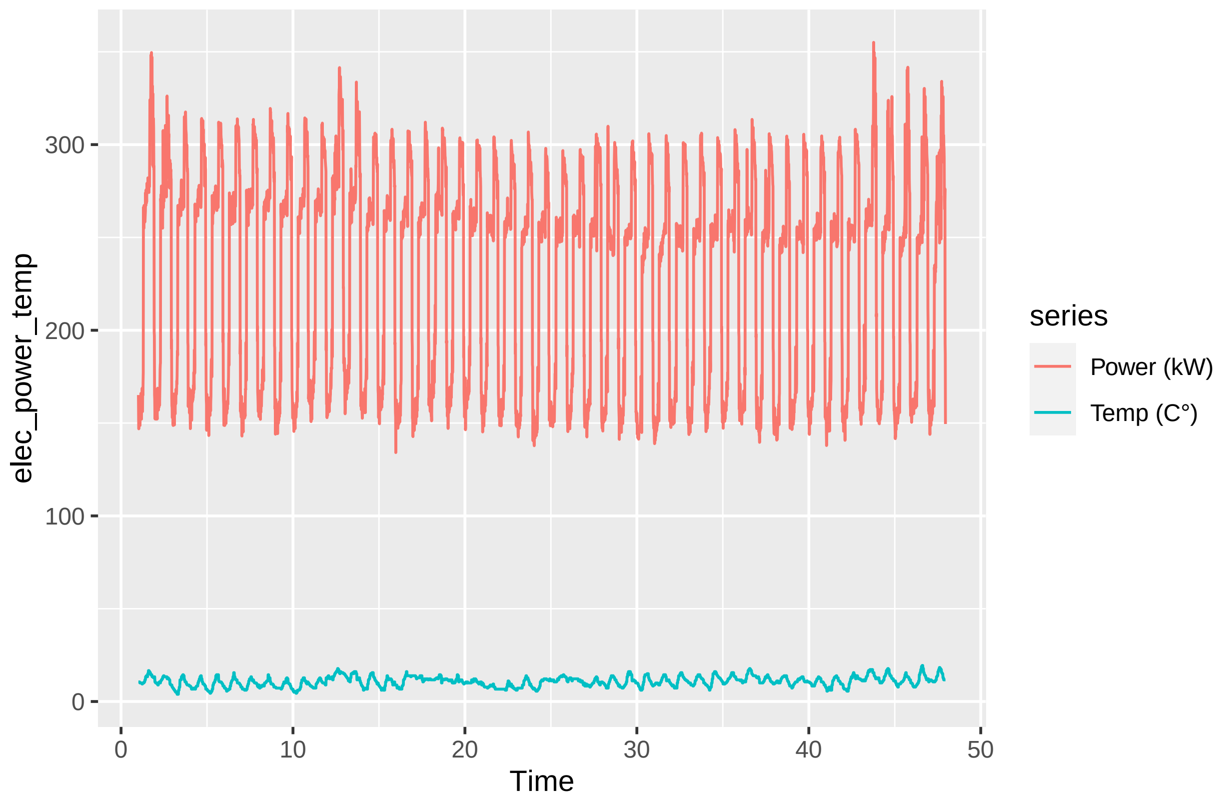
```
#write_csv(Pred,file="Pred_sans_temperature.csv")
#write.xlsx(Pred, file, sheetName = "Sheet1",
 # col.names = TRUE, row.names = TRUE, append = FALSE)
```

Part 2: Forecast electricity consumption by using outdoor temperature

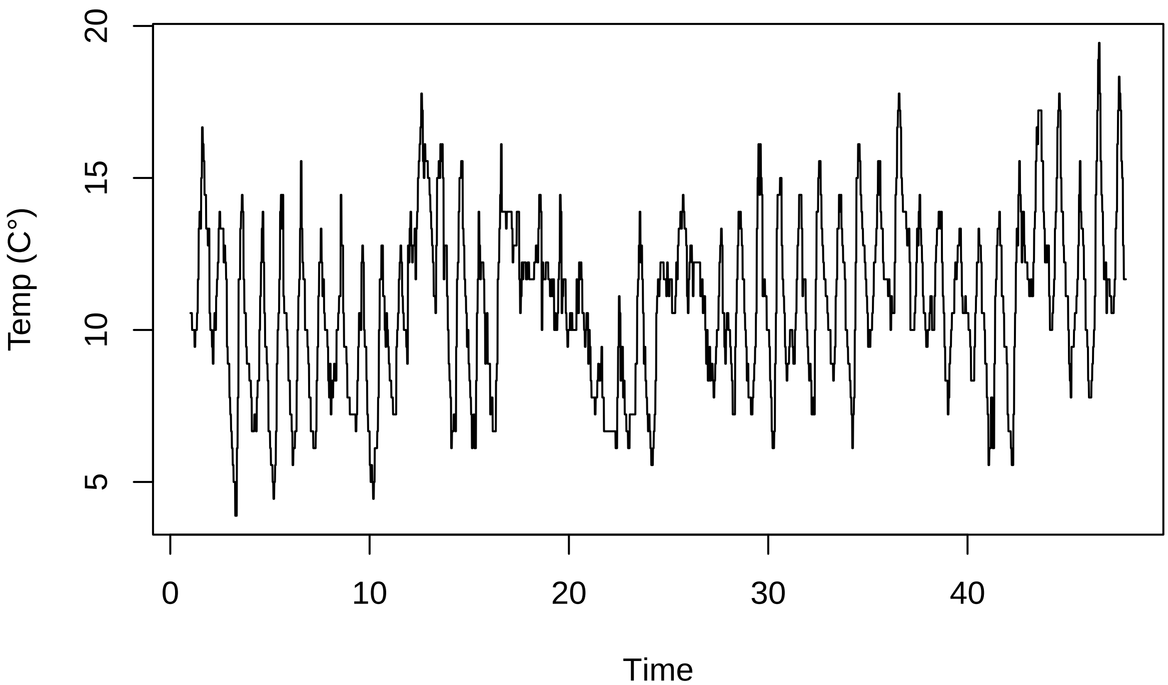
```
elec_power_temp<-ts(data[1:4507,2:3],start=c(1,2),freq=96)
plot(elec_power_temp)
```



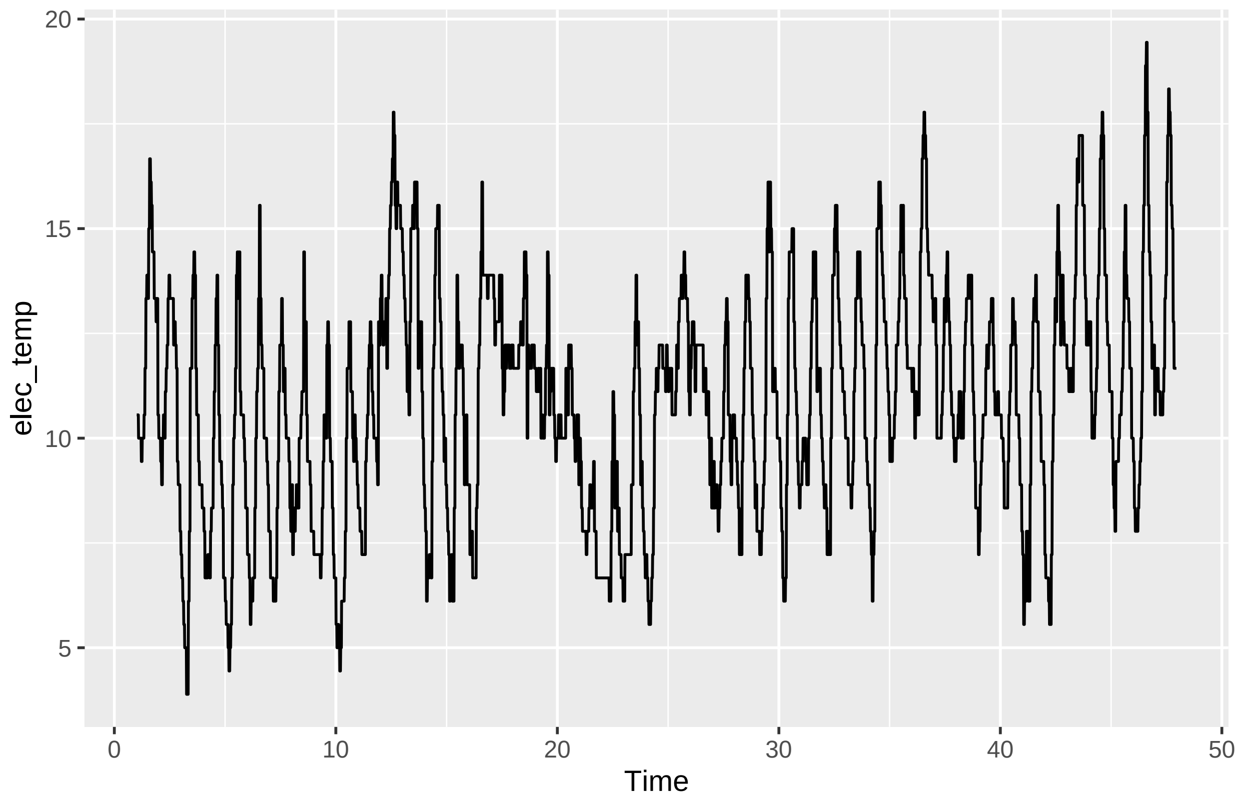

```
autoplot(elec_power_temp)
```




```
elec_temp<-ts(data[1:4507,3],start=c(1,2),freq=96)
plot(elec_temp)
```



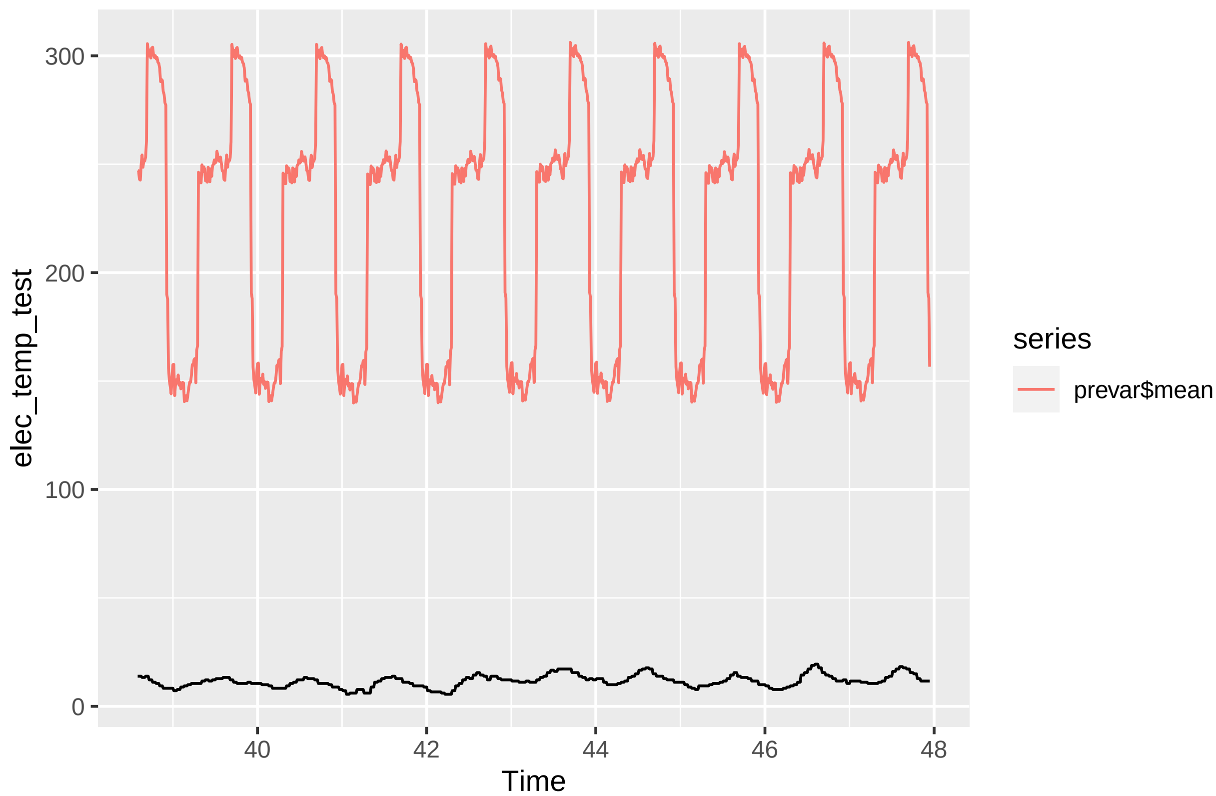

```
autoplot(elec_temp)
```




```
#We need to make two sets of data: the train one (80%) and the test one (20%) in order to evaluate the 
elec_temp_train=head(elec_temp,3607)
elec_temp_test=tail(elec_temp,900)
```

Forecasting SARIMA model:

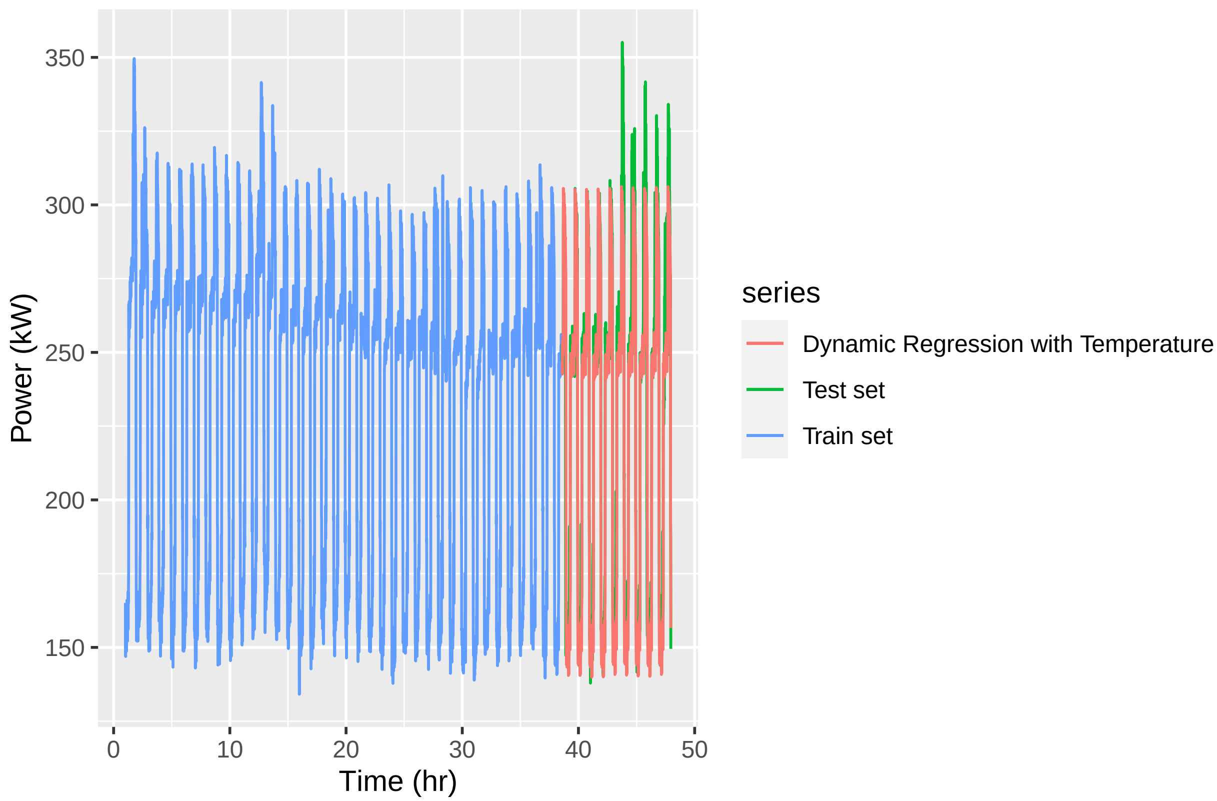
```
elec_power_temp_train_ar=auto.arima(elec_power_train,xreg=elec_temp_train)
prevar=forecast(elec_power_temp_train_ar,h=900,xreg=elec_temp_test)
autoplot(elec_temp_test)+autolayer(prevar$mean)
```




```r
#RMSE
print(sqrt(mean((prevar$mean-elec_power_test)^2)))
```

```
## [1] 16.9125
```

```r
autoplot(elec_power_train,series="Train set") +
  autolayer(elec_power_test,series='Test set')+
  autolayer(prevar$mean,series='Dynamic Regression with Temperature')+

  xlab('Time (hr)') +
  ylab('Power (kW)')
```

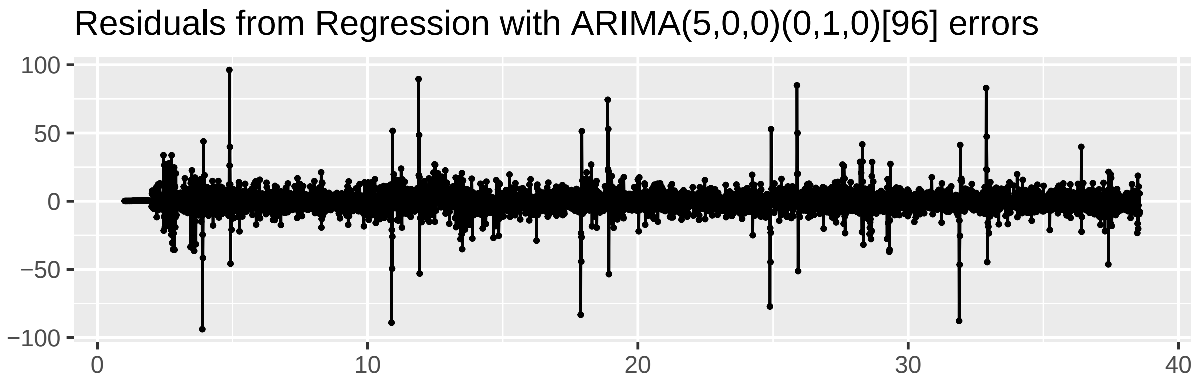


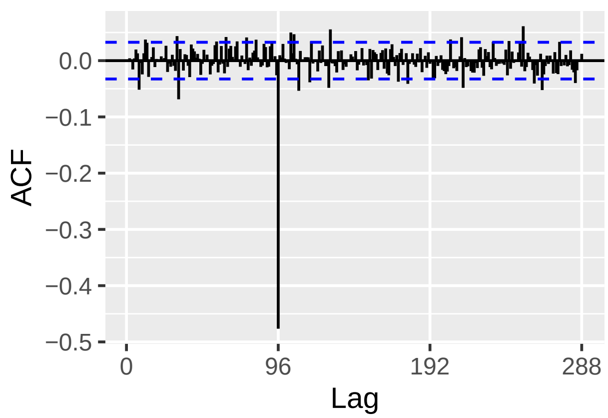
vérifions le résidu, il y a encore des autocorrélations :

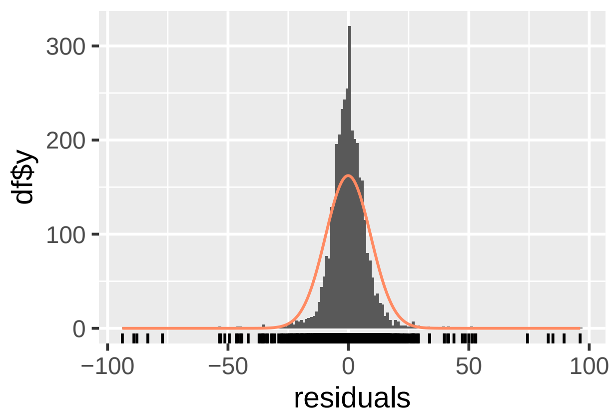
```
summary(elec_power_temp_train_ar)
```

```
## Series: elec_power_train
## Regression with ARIMA(5,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1     ar2      ar3      ar4     ar5  Temp (C°)
##       0.7584  0.1095  -0.0178  -0.2645  0.1867     0.1834
## s.e.  0.0166  0.0205   0.0206   0.0205  0.0166     0.2356
##
## sigma^2 = 90.28:  log likelihood = -12884.28
## AIC=25782.56   AICc=25782.6   BIC=25825.71
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1041197 9.366215 5.776051 -0.1287148 2.654515 0.7262754
##                      ACF1
## Training set 0.0007459103
```

```
#autocorrelation of residuals
checkresiduals(elec_power_temp_train_ar,test="LB",plot=TRUE)
```



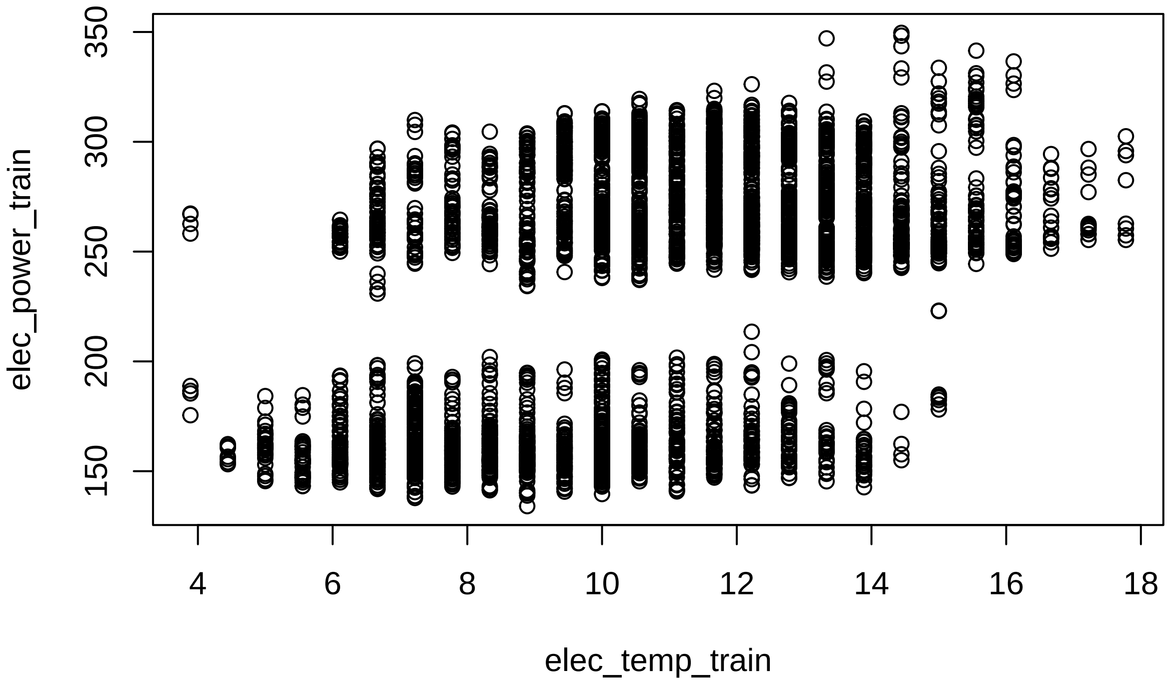





```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,0)[96] errors
## Q* = 1119.9, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

Nous pouvons essayer de trouver un meilleur modèle manuellement. Regardons la relation entre le Power et la Temp

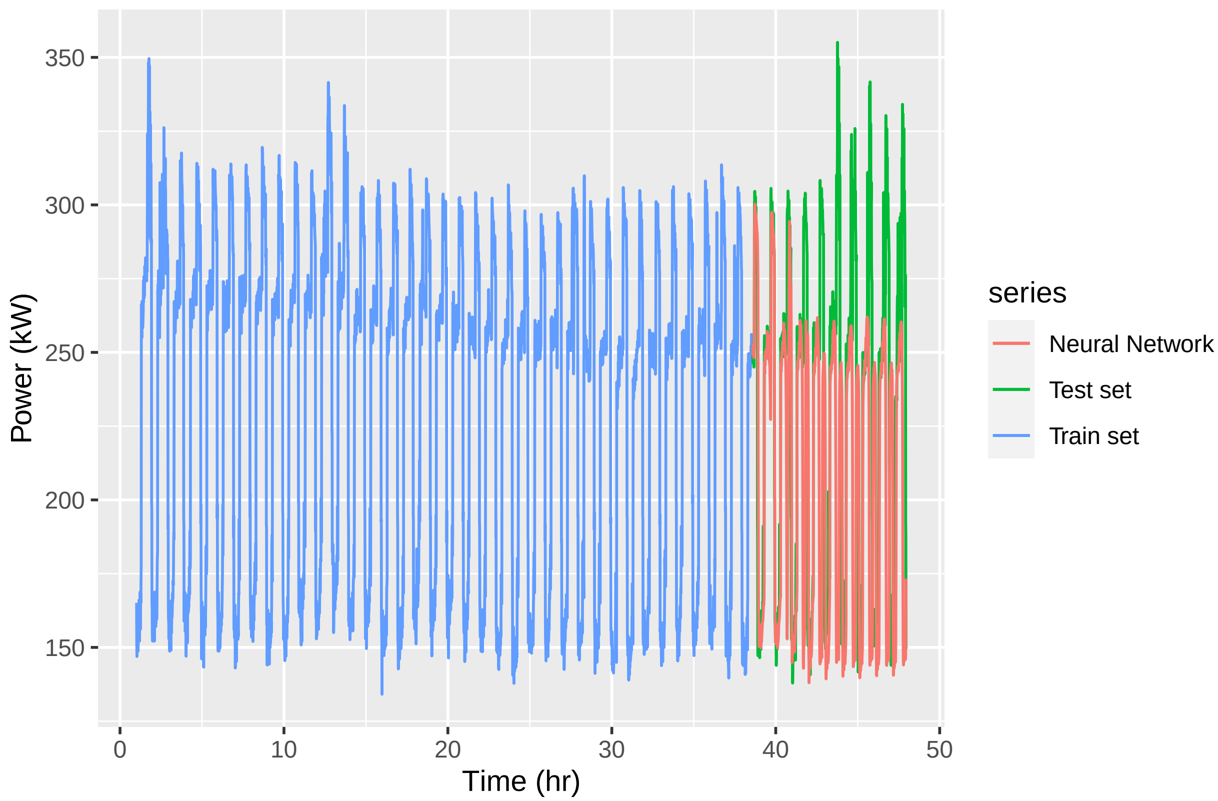
```
plot(elec_temp_train,elec_power_train)
```



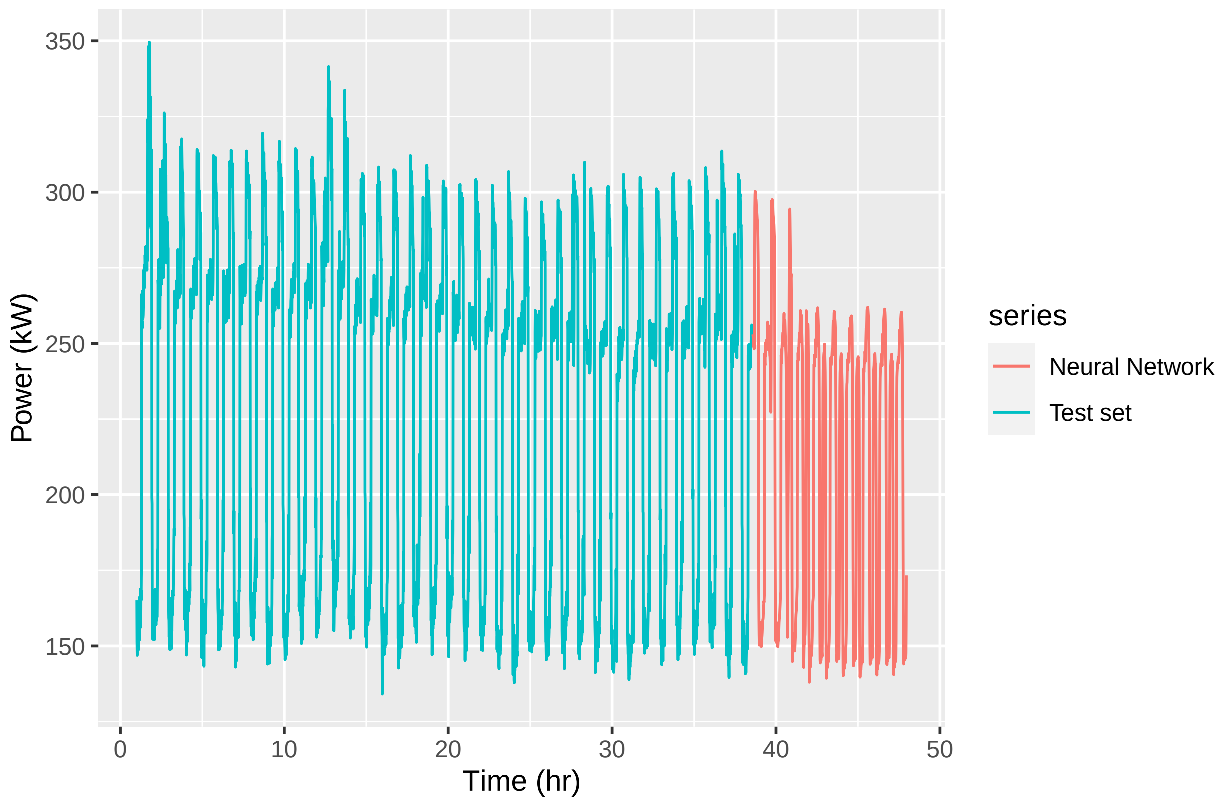
Forecasting with Neural Network

We van automatically select the best NNAR(p,P,k)T:

```
elec_power_temp_nn = nnetar(elec_power_train,xreg = elec_temp_train)
pred_elec_power_temp_nn = forecast(elec_power_temp_nn,xreg = elec_temp_test, h = 900)
autoplot(elec_power_train,series="Train set") +
  autolayer(elec_power_test,series='Test set')+
  autolayer(pred_elec_power_temp_nn$mean,series='Neural Network')+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```




```
autoplot(elec_power_train,series='Test set') +
  autolayer(pred_elec_power_temp_nn$mean,series='Neural Network')+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```

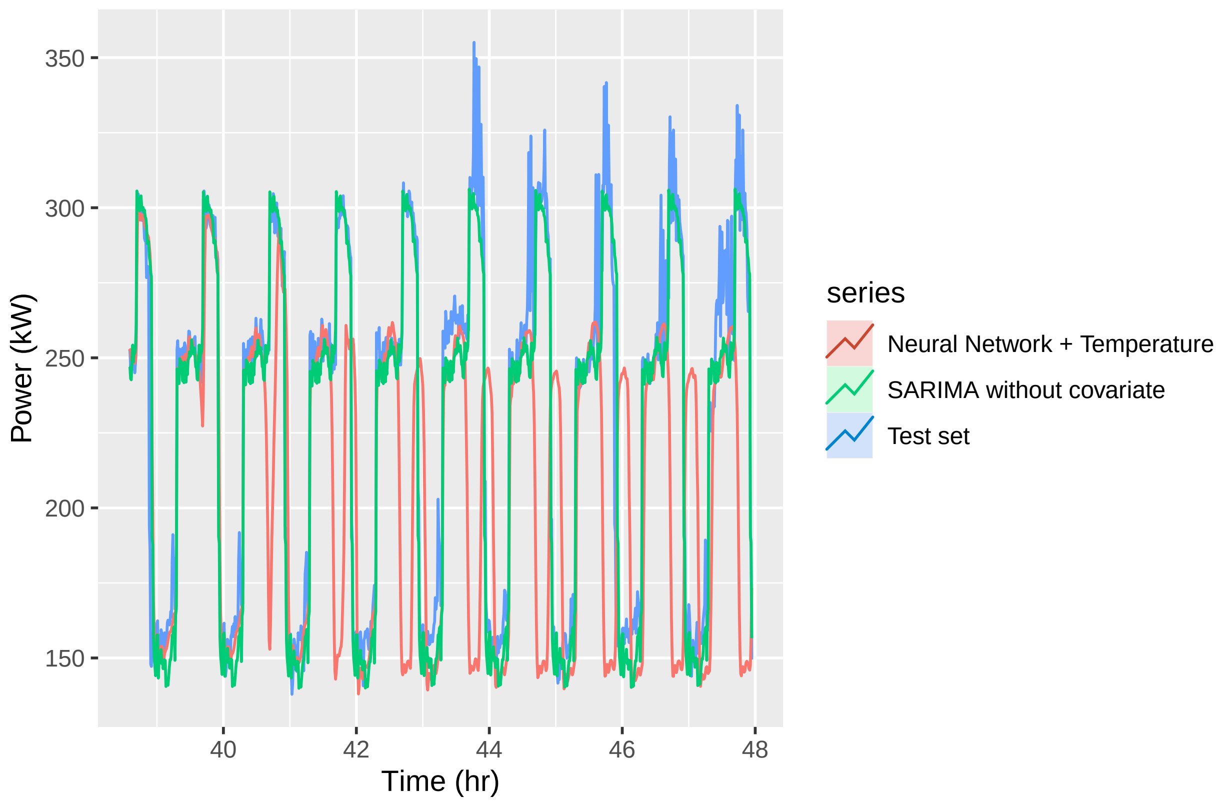



```r
#RMSE
print(sqrt(mean((pred_elec_power_temp_nn$mean-elec_power_test)^2)))
```

```
## [1] 65.96226
```

C'est donc SARIMA le meilleur modèle avec un RMSE de 16.912.

```r
#We can zoom in the prediction.

autoplot(elec_power_test,series='Test set') +
  autolayer(pred_elec_power_temp_nn$mean,series='Neural Network + Temperature')+
  autolayer(prevar,series='SARIMA without covariate',PI=FALSE)+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```



Nous allons maintenant prévoir la consommation d'électricité (kW) pour le 17/02/2010 en fonction de la température avec 96 observations.

```
temp_17 <- ts(data[4508:4603,3], frequency = 96, start=c(1,2))
head(temp_17)
```

```
## Time Series:
## Start = c(1, 2)
## End = c(1, 7)
## Frequency = 96
##      Temp (C°)
## [1,]  11.66667
## [2,]  11.11111
## [3,]  11.11111
## [4,]  11.11111
## [5,]  11.11111
## [6,]  10.55556
```

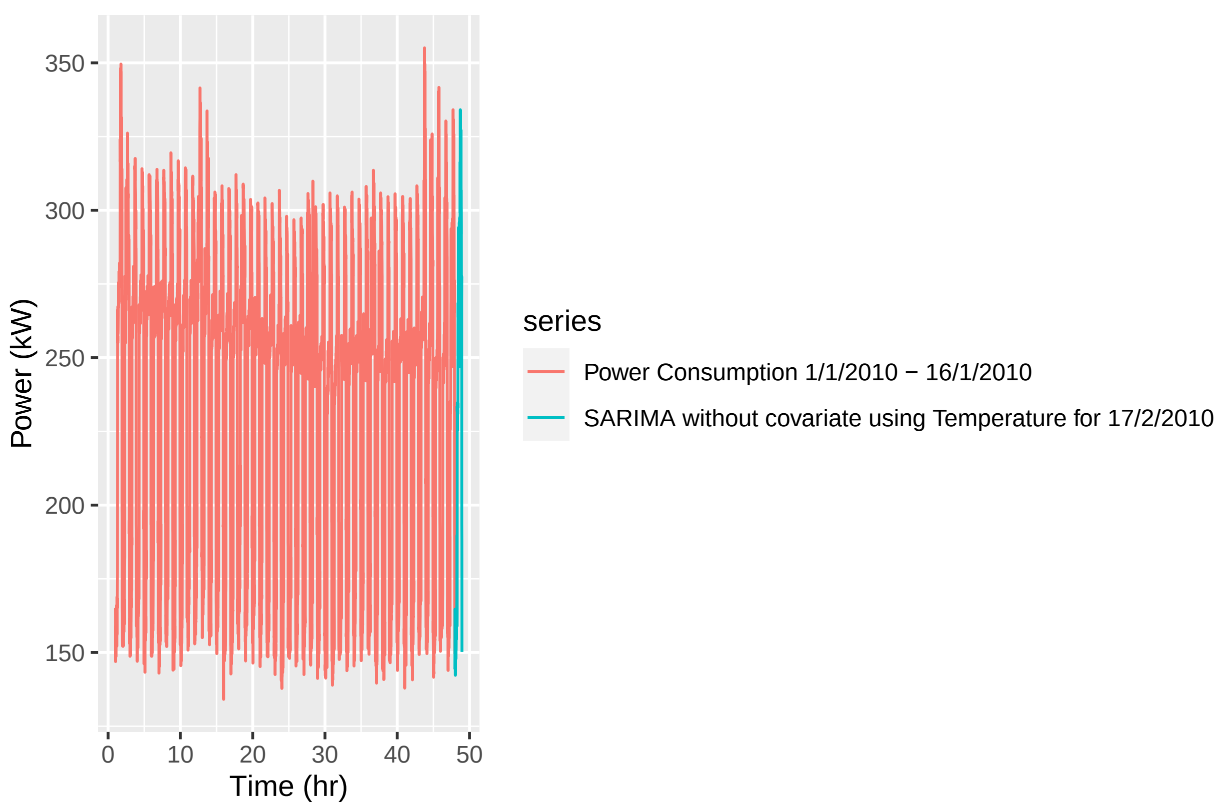
```
tail(temp_17)
```

```
## Time Series:
## Start = c(1, 92)
## End = c(2, 1)
## Frequency = 96
##      Temp (C°)
```

```
## [1,]  13.88889
## [2,]  13.88889
## [3,]  13.88889
## [4,]  12.77778
## [5,]  12.77778
## [6,]  12.77778
```

```
elec_power_temp_train_ar_pred=auto.arima(elec_power,xreg=elec_temp)
prevar17=forecast(elec_power_temp_train_ar_pred,h=96,xreg=temp_17)
autoplot(elec_power,series="Power Consumption 1/1/2010 - 16/1/2010") +
  autolayer(prevar17$mean,series="SARIMA without covariate using Temperature for 17/2/2010",PI=FALSE) +
  xlab('Time (hr)') +
  ylab('Power (kW)')
```

```
## Warning in ggplot2::geom_line(ggplot2::aes_(x = ~timeVal, y = ~seriesVal, :
## Ignoring unknown parameters: ‘PI‘
```



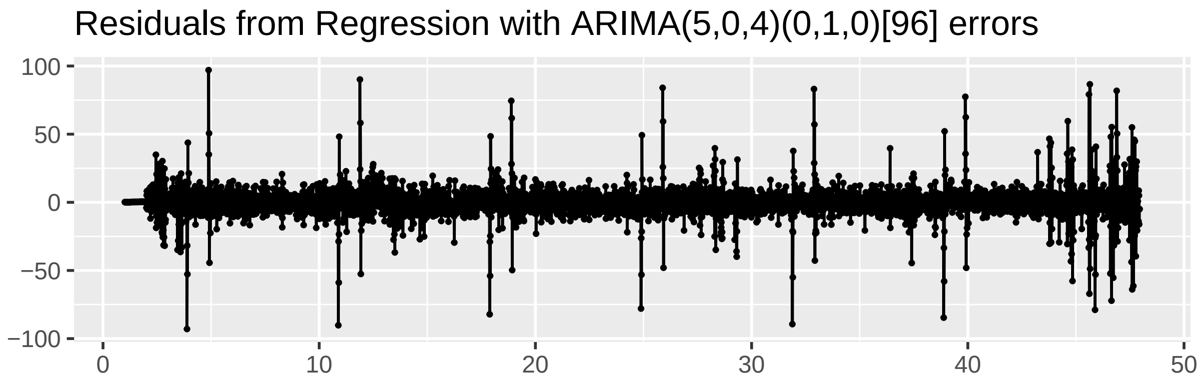

```
#Pred results
Pred_T = print(prevar17)
```

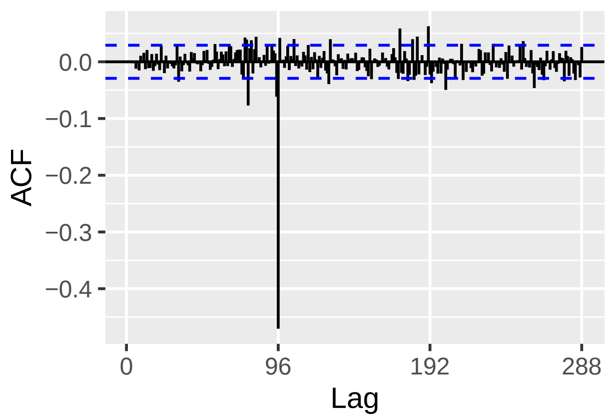
```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47.95833       144.3176 130.4109 158.2242 123.0492 165.5860
## 47.96875       145.3003 128.4172 162.1834 119.4798 171.1208
## 47.97917       150.7041 132.2321 169.1760 122.4537 178.9545
```

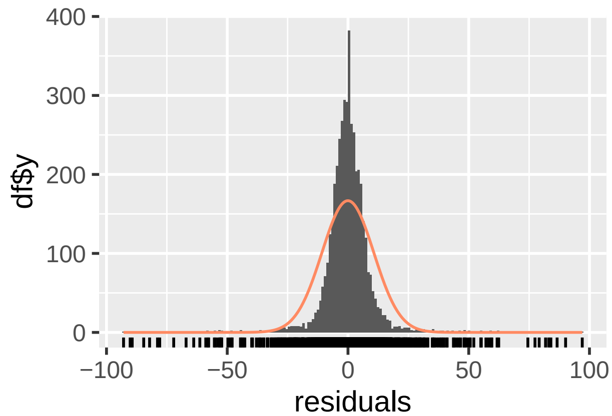
```
## 47.98958       150.8001 130.7058 170.8944 120.0685 181.5316
## 48.00000       164.7166 144.3646 185.0685 133.5910 195.8421
## 48.01042       159.7337 139.1897 180.2777 128.3144 191.1530
## 48.02083       154.6955 134.0187 175.3722 123.0731 186.3178
## 48.03125       148.8742 128.1431 169.6053 117.1687 180.5797
## 48.04167       142.2673 121.4306 163.1040 110.4003 174.1342
## 48.05208       152.3274 131.4767 173.1781 120.4389 184.2158
## 48.06250       153.8391 132.9835 174.6947 121.9432 185.7350
## 48.07292       153.0982 132.2341 173.9623 121.1893 185.0071
## 48.08333       152.9200 132.0540 173.7861 121.0082 184.8319
## 48.09375       147.8133 126.9401 168.6866 115.8904 179.7362
## 48.10417       147.7420 126.8676 168.6164 115.8174 179.6667
## 48.11458       156.6476 135.7732 177.5220 124.7229 188.5722
## 48.12500       160.6629 139.7879 181.5378 128.7374 192.5884
## 48.13542       155.1519 134.2769 176.0270 123.2263 187.0775
## 48.14583       150.8051 129.9296 171.6806 118.8788 182.7314
## 48.15625       152.7445 131.8689 173.6202 120.8180 184.6711
## 48.16667       157.8784 137.0028 178.7541 125.9519 189.8050
## 48.17708       157.4905 136.6149 178.3662 125.5639 189.4171
## 48.18750       154.4478 133.5721 175.3235 122.5212 186.3744
## 48.19792       155.9666 135.0909 176.8423 124.0399 187.8932
## 48.20833       159.0772 138.2014 179.9529 127.1505 191.0038
## 48.21875       163.2207 142.3449 184.0964 131.2940 195.1473
## 48.22917       165.7060 144.8303 186.5818 133.7794 197.6327
## 48.23958       168.0257 147.1499 188.9014 136.0990 199.9523
## 48.25000       189.6339 168.7582 210.5097 157.7073 221.5606
## 48.26042       174.4044 153.5286 195.2801 142.4777 206.3311
## 48.27083       161.4986 140.6229 182.3743 129.5719 193.4253
## 48.28125       158.9914 138.1156 179.8671 127.0647 190.9180
## 48.29167       163.0975 142.2218 183.9733 131.1708 195.0242
## 48.30208       233.4696 212.5939 254.3453 201.5429 265.3963
## 48.31250       231.7699 210.8942 252.6457 199.8432 263.6966
## 48.32292       234.7683 213.8926 255.6441 202.8416 266.6950
## 48.33333       225.2653 204.3896 246.1410 193.3386 257.1920
## 48.34375       230.9670 210.0913 251.8427 199.0403 262.8937
## 48.35417       232.2688 211.3931 253.1446 200.3421 264.1955
## 48.36458       230.8688 209.9931 251.7446 198.9422 262.7955
## 48.37500       234.5684 213.6926 255.4441 202.6417 266.4950
## 48.38542       233.0354 212.1596 253.9111 201.1087 264.9620
## 48.39583       246.1357 225.2600 267.0115 214.2091 278.0624
## 48.40625       261.3365 240.4608 282.2123 229.4099 293.2632
## 48.41667       267.3365 246.4608 288.2123 235.4099 299.2632
## 48.42708       268.5364 247.6606 289.4121 236.6097 300.4631
## 48.43750       263.8360 242.9602 284.7117 231.9093 295.7627
## 48.44792       266.9360 246.0603 287.8118 235.0093 298.8627
## 48.45833       276.8364 255.9606 297.7121 244.9097 308.7630
## 48.46875       294.1320 273.2563 315.0077 262.2053 326.0587
## 48.47917       257.5319 236.6562 278.4077 225.6052 289.4586
## 48.48958       292.3318 271.4561 313.2075 260.4051 324.2585
## 48.50000       292.2318 271.3560 313.1075 260.3051 324.1585
## 48.51042       267.9682 247.0924 288.8439 236.0415 299.8948
## 48.52083       270.4682 249.5924 291.3439 238.5415 302.3949
## 48.53125       285.2681 264.3924 306.1439 253.3415 317.1948
## 48.54167       276.2681 255.3924 297.1438 244.3414 308.1948
```

```
## 48.55208       279.0000 258.1242 299.8757 247.0733 310.9266
## 48.56250       286.0000 265.1243 306.8757 254.0733 317.9267
## 48.57292       264.4000 243.5243 285.2758 232.4733 296.3267
## 48.58333       295.8000 274.9243 316.6757 263.8733 327.7267
## 48.59375       295.4000 274.5243 316.2757 263.4733 327.3267
## 48.60417       247.0000 226.1242 267.8757 215.0733 278.9267
## 48.61458       263.3000 242.4243 284.1757 231.3733 295.2267
## 48.62500       269.8000 248.9243 290.6757 237.8733 301.7267
## 48.63542       272.2319 251.3561 293.1076 240.3052 304.1586
## 48.64583       297.5319 276.6561 318.4076 265.6052 329.4586
## 48.65625       258.3319 237.4561 279.2076 226.4052 290.2586
## 48.66667       269.6319 248.7561 290.5076 237.7052 301.5586
## 48.67708       272.6319 251.7561 293.5076 240.7052 304.5586
## 48.68750       249.4319 228.5561 270.3076 217.5052 281.3586
## 48.69792       304.3319 283.4561 325.2076 272.4052 336.2586
## 48.70833       316.3319 295.4561 337.2076 284.4052 348.2586
## 48.71875       299.2000 278.3243 320.0757 267.2733 331.1267
## 48.72917       334.1000 313.2243 354.9757 302.1733 366.0267
## 48.73958       308.1000 287.2243 328.9757 276.1733 340.0267
## 48.75000       305.9000 285.0243 326.7757 273.9733 337.8267
## 48.76042       331.2319 310.3561 352.1076 299.3052 363.1586
## 48.77083       292.7319 271.8561 313.6076 260.8052 324.6586
## 48.78125       317.6319 296.7561 338.5076 285.7052 349.5586
## 48.79167       316.3319 295.4561 337.2076 284.4052 348.2586
## 48.80208       297.3275 276.4517 318.2032 265.4008 329.2542
## 48.81250       327.2275 306.3517 348.1032 295.3008 359.1542
## 48.82292       310.0275 289.1517 330.9032 278.1008 341.9542
## 48.83333       291.9275 271.0517 312.8032 260.0008 323.8542
## 48.84375       306.8912 286.0155 327.7670 274.9645 338.8179
## 48.85417       301.1912 280.3155 322.0670 269.2645 333.1179
## 48.86458       299.8912 279.0155 320.7670 267.9645 331.8179
## 48.87500       294.6912 273.8155 315.5670 262.7645 326.6179
## 48.88542       271.9275 251.0517 292.8032 240.0008 303.8542
## 48.89583       266.7275 245.8517 287.6032 234.8008 298.6542
## 48.90625       272.2275 251.3517 293.1032 240.3008 304.1542
## 48.91667       277.5275 256.6517 298.4032 245.6008 309.4542
## 48.92708       193.3637 172.4880 214.2395 161.4371 225.2904
## 48.93750       187.7637 166.8880 208.6395 155.8371 219.6904
## 48.94792       150.1637 129.2880 171.0395 118.2371 182.0904
```

```
#Checking model

checkresiduals(prevar17,test="LB",plot=TRUE)
```








```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,4)(0,1,0)[96] errors
## Q* = 1365.2, df = 183, p-value < 2.2e-16
##
## Model df: 9.   Total lags used: 192
```

```
#write_csv(Pred_T,file="Pred_with_temperature.csv")
#write.xlsx(Pred_T, file, sheetName = "Sheet1",
 # col.names = TRUE, row.names = TRUE, append = FALSE)
```

Conclusion:

Le meilleur modèle de prévision est SARIMA avec prise en compte de la température.