Projet_TimeSeries

Fatimetou Haidara

2023-01-26

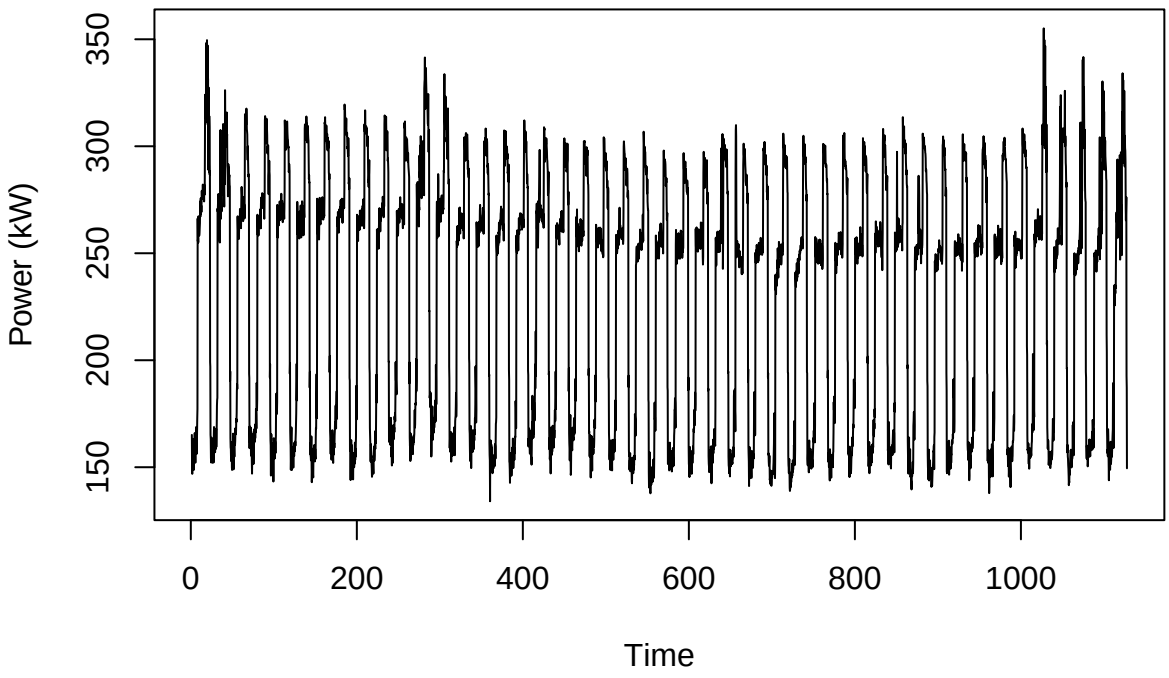
```
#Loading the data
data <- read_excel("C:/Users/f_ati/Documents/Master2/Times series/Projet/Timeseries/Elec-train.xlsx")
data
```

```
## # 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
```

```
#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=60/15)
tail(elec_power)
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 1126             265.4 270.9
## 1127 276.2 192.7 187.1 149.5
```

```
plot(elec_power)
```




```
mean(elec_power)
```

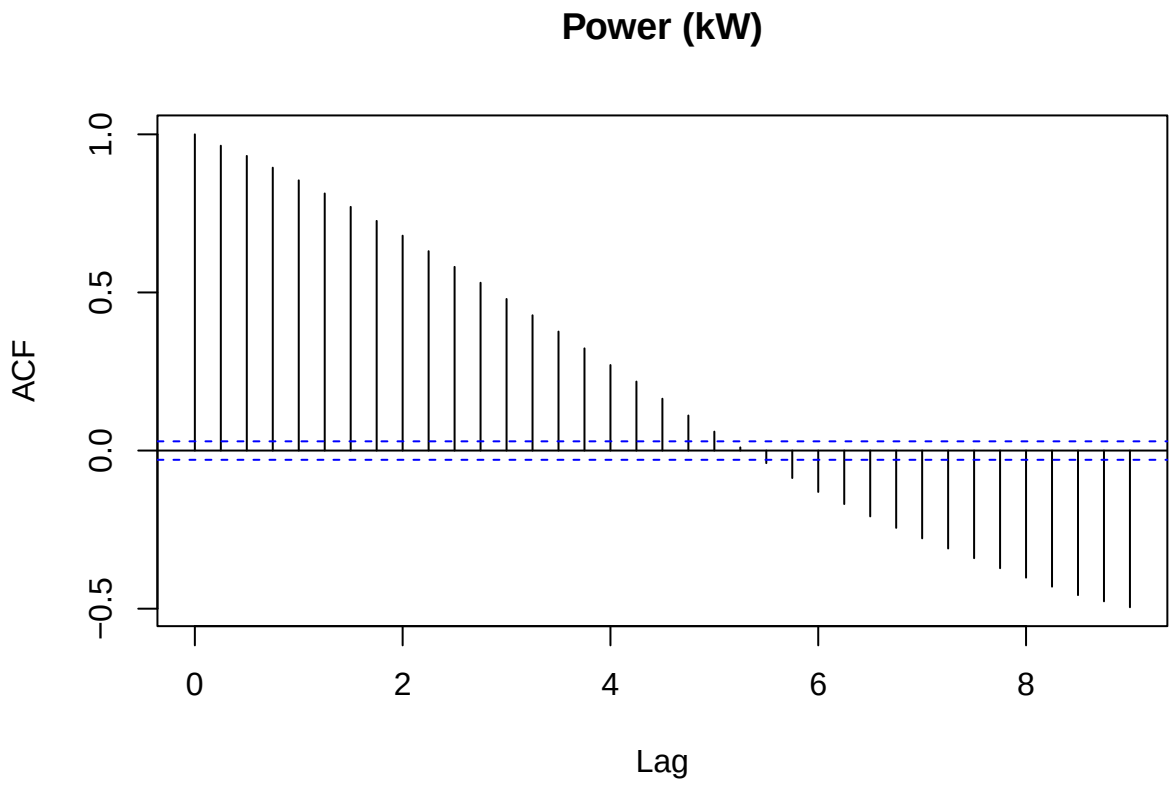
```
## [1] 231.5873
```

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

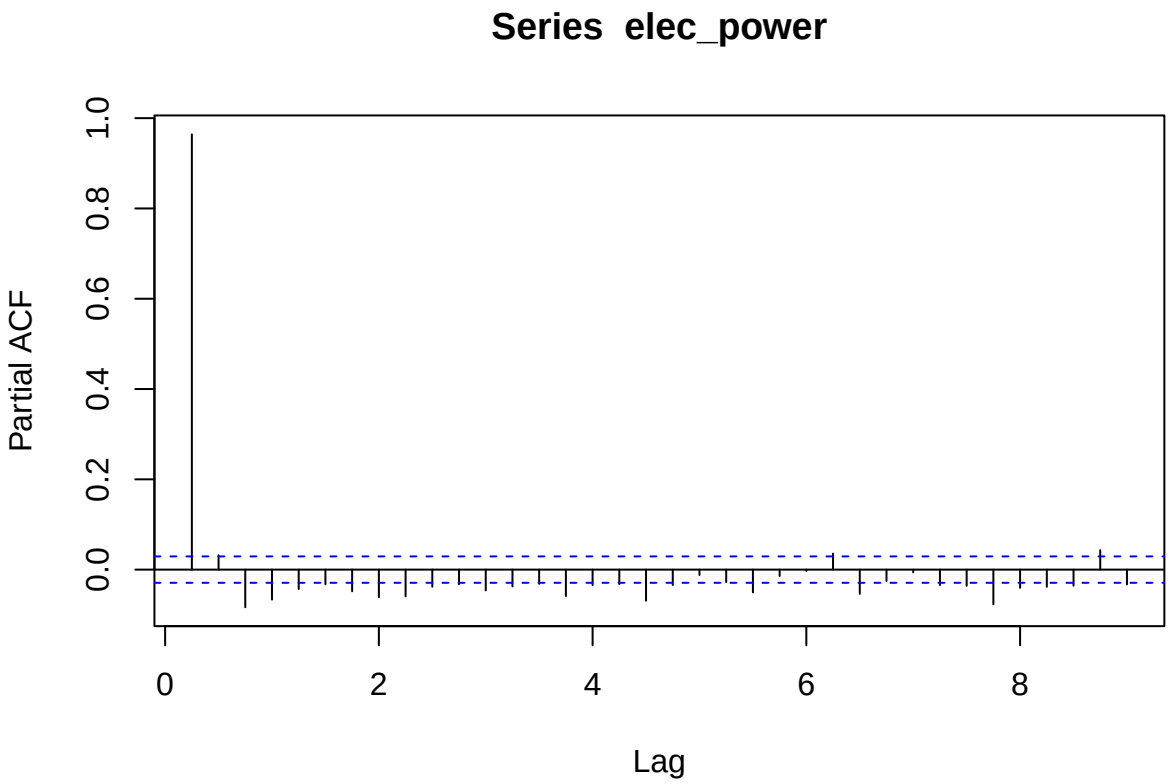
```
tmp=acf(elec_power,type="cor",plot = FALSE)
tmp$acf[1:3,1,1]
```

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

```
plot(tmp)
```



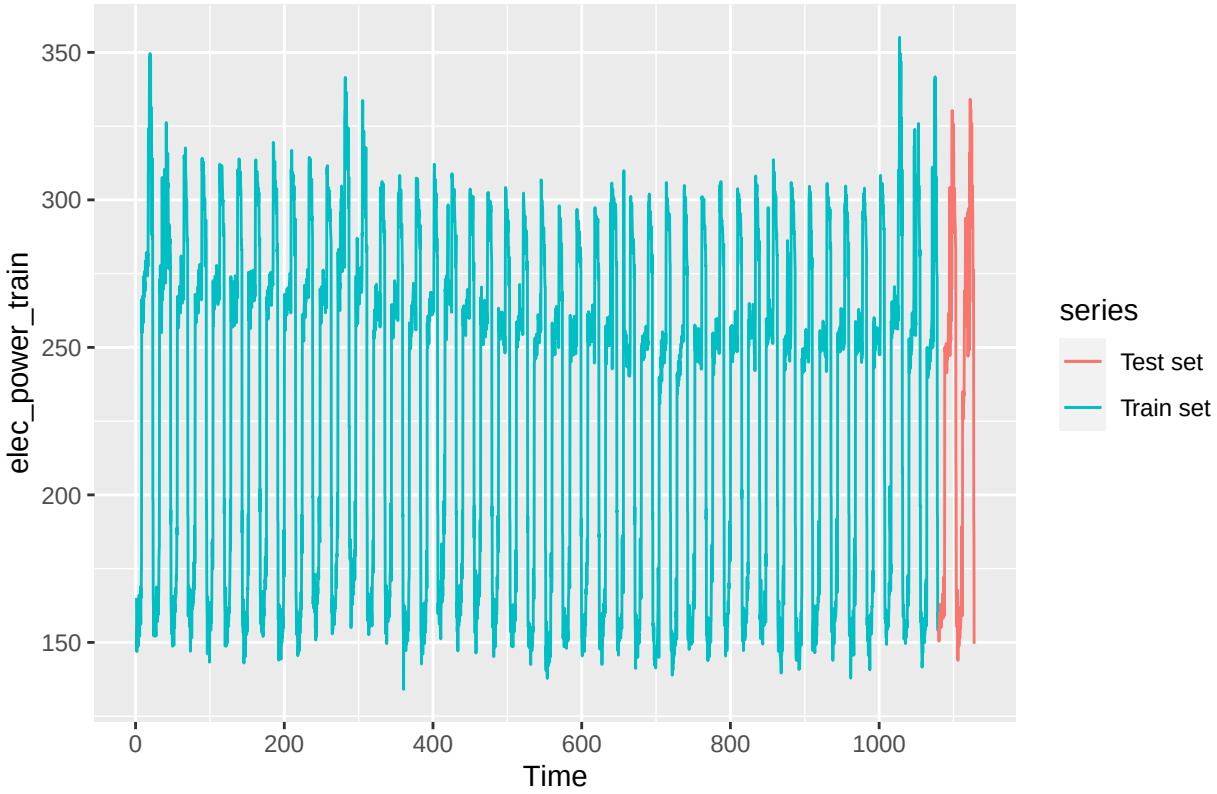

```
pacf(elec_power)
```



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,4315)
elec_power_test=tail(elec_power,192)
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.

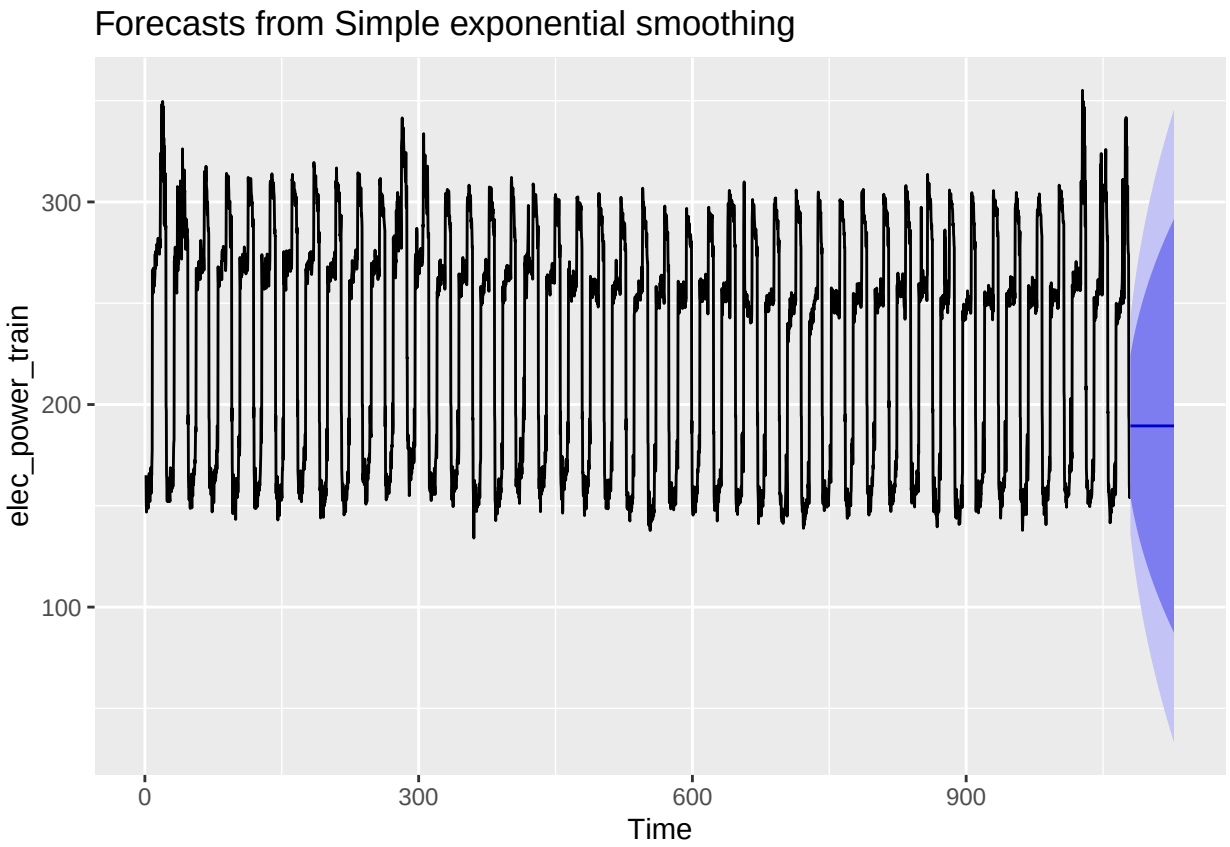
Lissage exponentiel simple (SES)

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=192, alpha = .2)
round(accuracy(SES,elec_power_test),2)
```

```
##                 ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set  0.04 27.11 16.09 -1.47  7.80 1.00 0.83        NA
## Test set     43.23 73.25 64.31 12.49 25.84 3.98 0.94      3.46
```

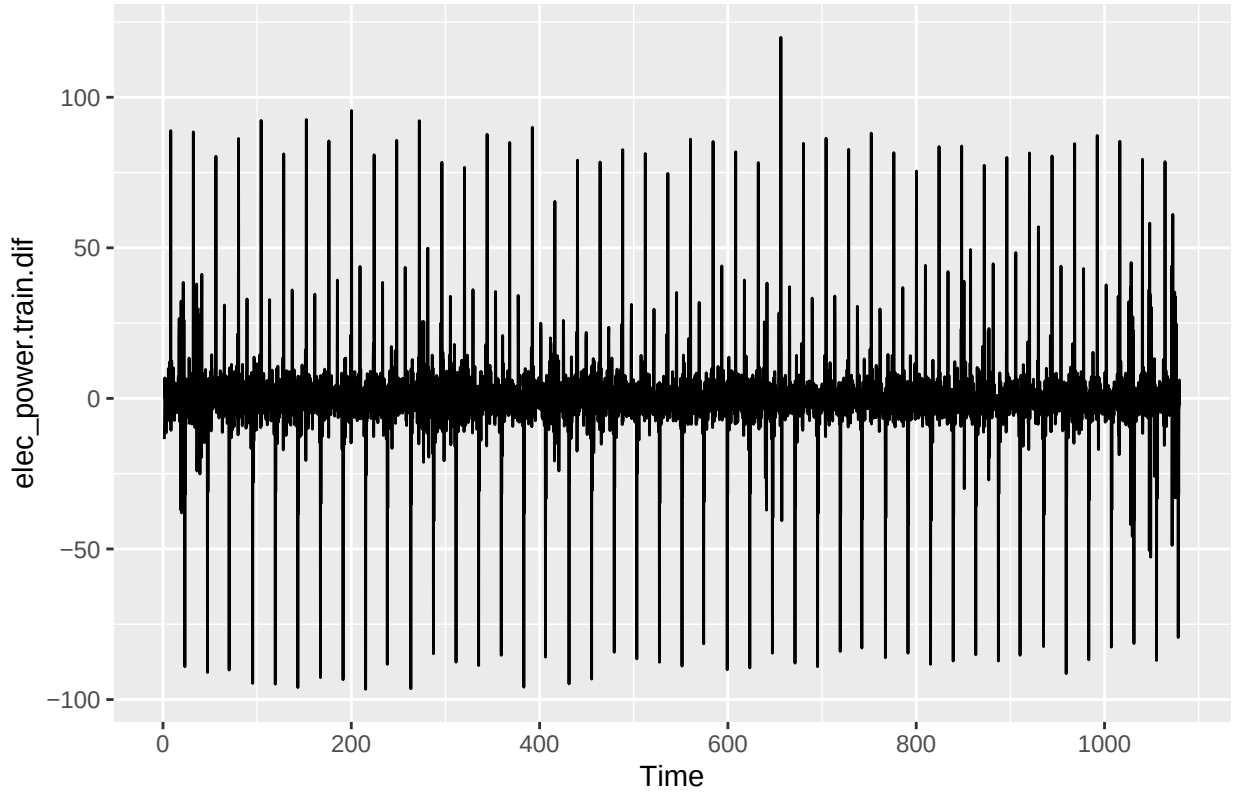
```
autoplot(SES)
```



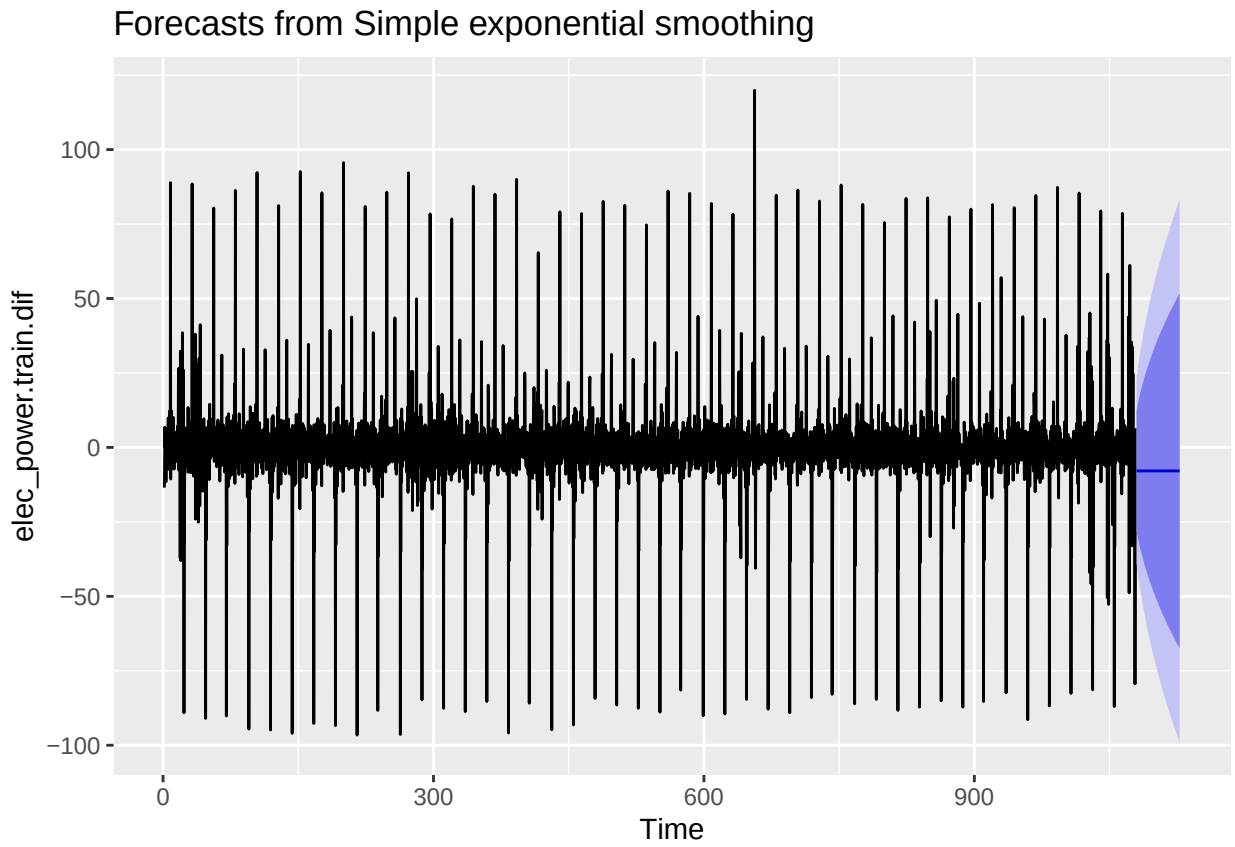
nous pouvons remarquer qu'une estimation plate est projetée vers l'avenir par notre modèle de prévision. Par conséquent, nous pouvons dire que d'après les données, il ne capture pas la tendance actuelle.

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=192 ,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)
```

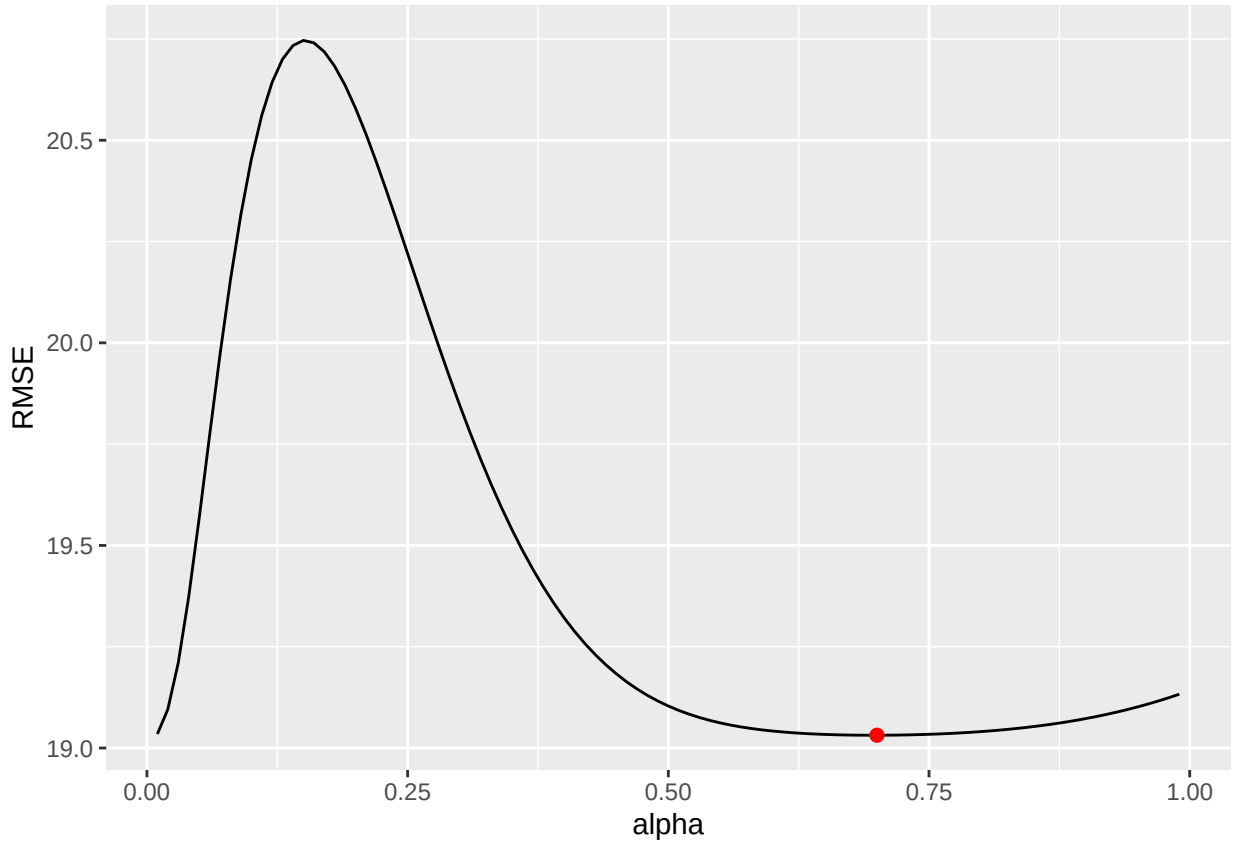
```
##                       ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -0.006594403 15.81522  8.484059      NaN      Inf 0.7476761
## Test set      7.861902173 20.57925 14.675011 7.054971 397.5033 1.2932671
##                    ACF1 Theil's U
## Training set -0.1612523        NA
## Test set     -0.2381809 0.8725684
```

```
# 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 = 192)
  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")
```



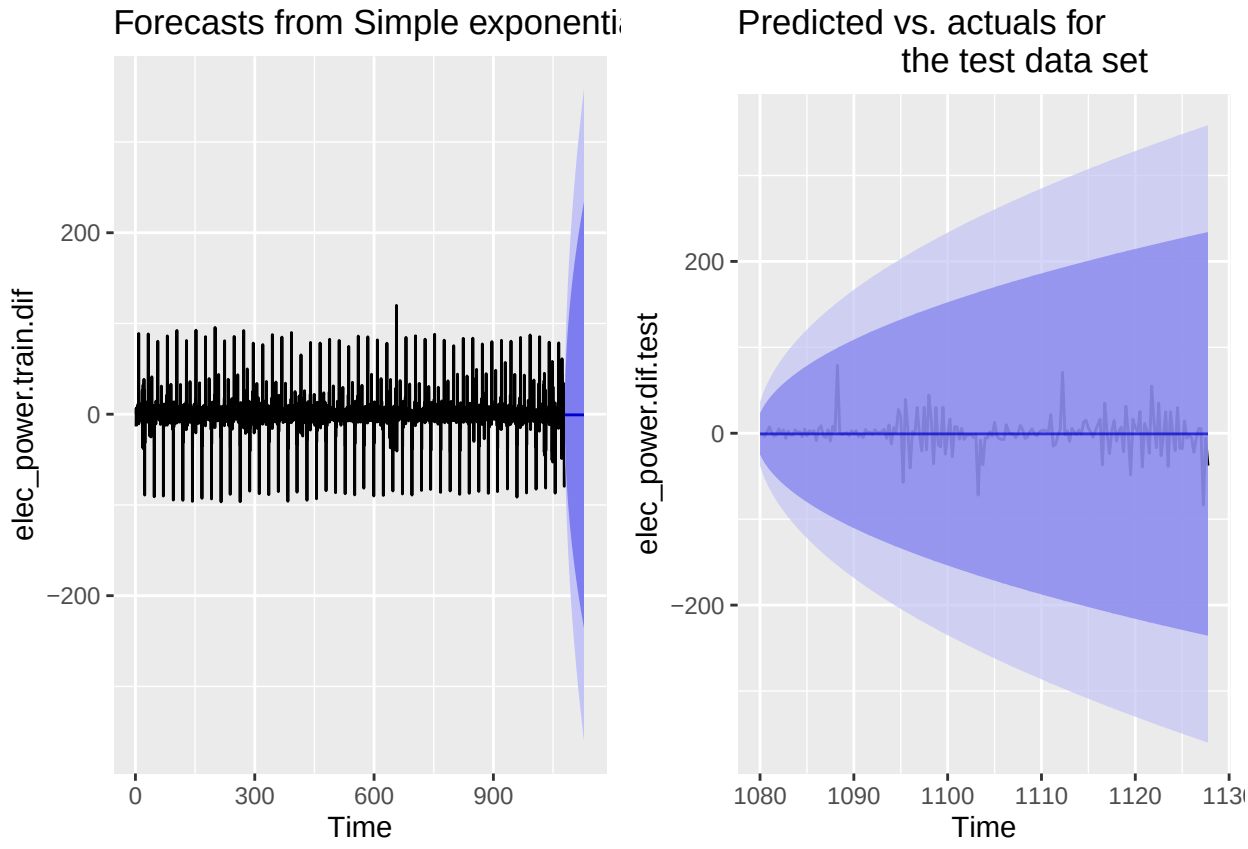
Nous remarquerons que environs 0,7 minimisera le plus.

Maintenant, nous allons essayer de réajuster notre modèle de prévision pour le SES avec alpha = 0,7.

```
# refit model with alpha = .7
SES.opt=ses(elec_power.train.dif,h=192 ,alpha = .7)
round(accuracy(SES.opt,elec_power.dif.test),2)
```

```
##                ME  RMSE   MAE  MPE   MAPE MASE  ACF1 Theil's U
## Training set 0.00 18.84 10.31  NaN    Inf 0.91 -0.41        NA
## Test set     0.71 19.03 11.60 91.3 108.73 1.02 -0.24      0.98
```

```
# plotting results
p1 <- autoplot(SES.opt) +
  theme(legend.position = "bottom")
p2 <- autoplot(elec_power.dif.test) +
  autolayer(SES.opt, alpha = .7) +
  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

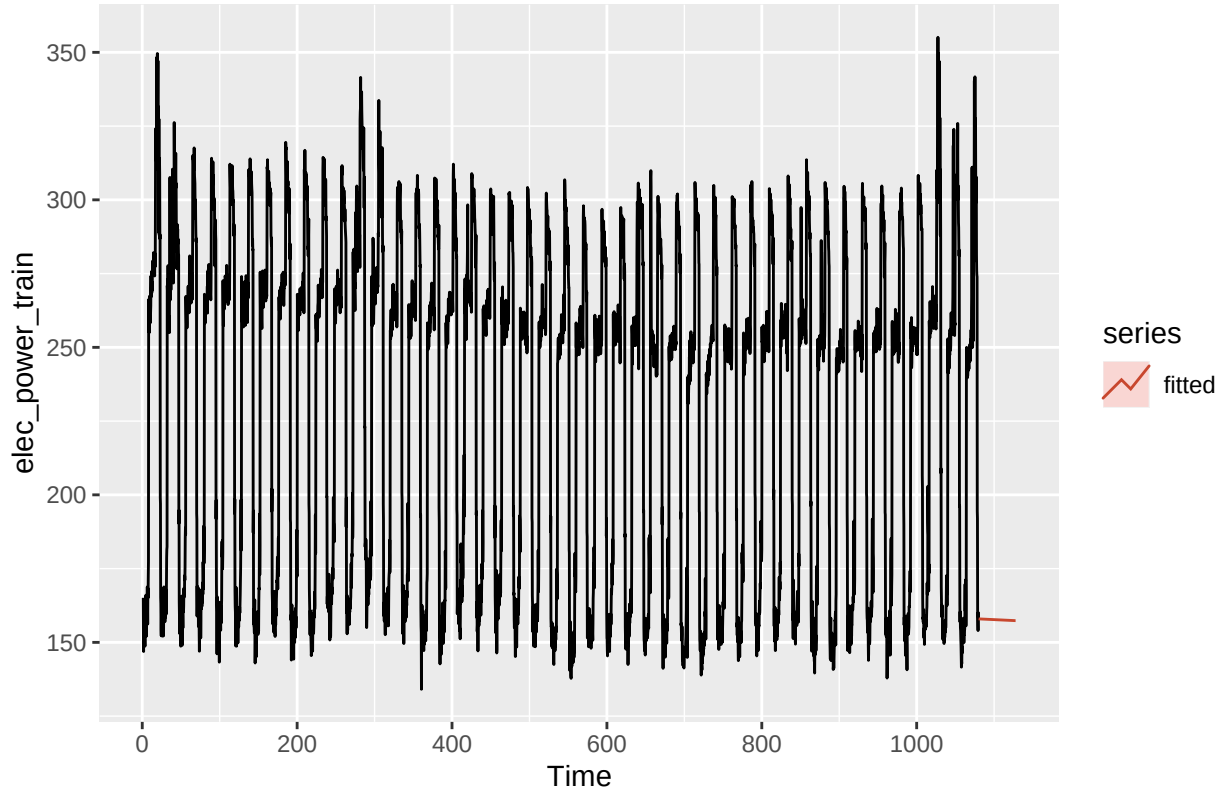
Nous avons vu qu'en SES, nous avons dû supprimer les tendances à long terme pour améliorer le modèle. Mais dans la méthode de Holt, nous pouvons appliquer un lissage exponentiel tout en capturant les tendances

dans les données. Cependat Il s'agit d'une technique qui fonctionne avec des données présentant une tendance mais pas de saisonnalité.

```
# Forecasting with a Holt
HOLT=holt(elec_power_train,h=192,alpha=NULL,beta=NULL)
round(accuracy(HOLT,elec_power_test),2)
```

```
##                    ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set -0.01 15.14  7.04 -0.24  3.22 0.44 0.00        NA
## Test set     75.03 95.58 76.13 27.17 27.90 4.71 0.94      4.31
```

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



Le modèle n'est pas adapté

Holt-Winter's Seasonal 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.

Le modèle additif est utilisé lorsque la tendance saisonnière des données a la même ampleur ou est constante tout au long de la période, tandis que le modèle multiplicatif est utilisé si l'ampleur de la tendance saisonnière des données augmente avec le temps.

```
#Additive seasonal Holt-Winters
fit1=hw(elec_power_train,  seasonal = "additive",h=192)

#Multiplicative seasonal Holt-Winters
fit2 = hw(elec_power_train, seasonal='multiplicative',h=192)
```

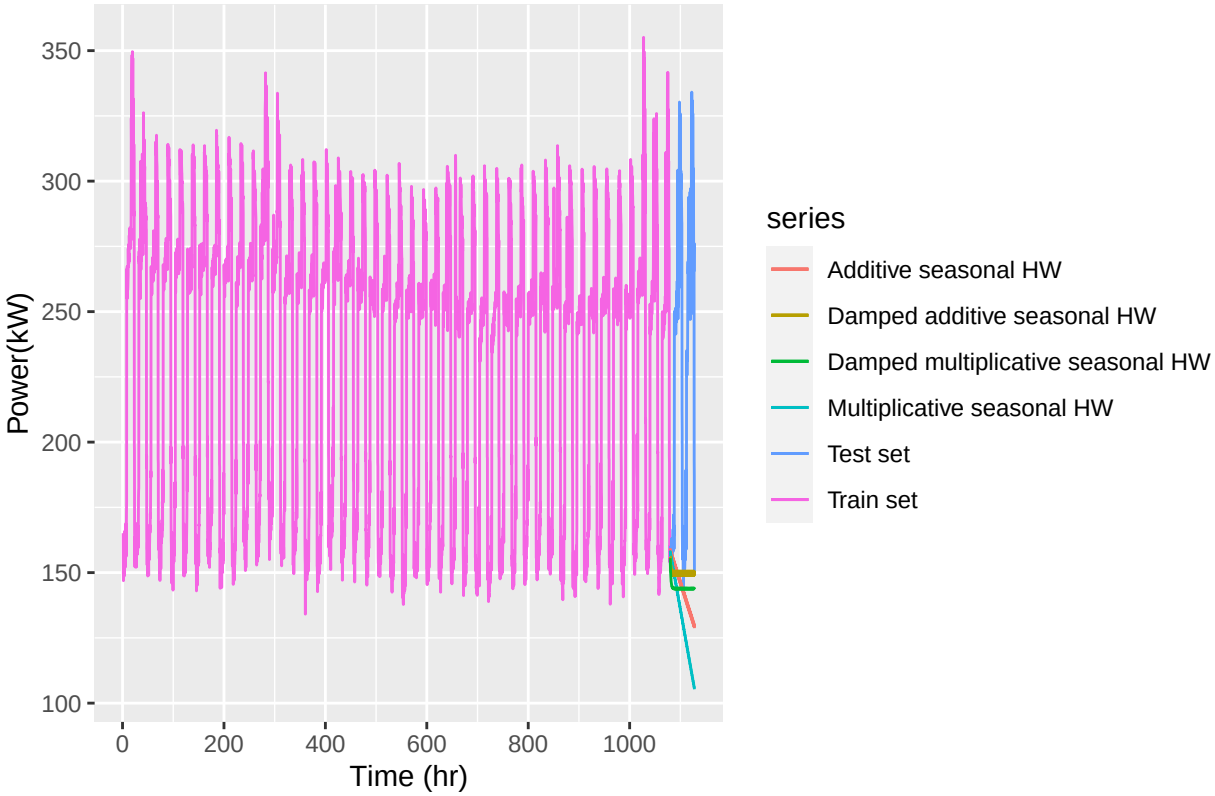
Damping Method

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.

```
#Damped additive seasonal Holt-Winters
fit3 = hw(elec_power_train, seasonal='additive',h=192,damped=TRUE)

#Damped multiplicative seasonal Holt-Winters
fit4 = hw(elec_power_train,seasonal='multiplicative',h=192,damped=TRUE)
```

Plot all models

```
autoplot(elec_power_train,series="Train set") +
  autolayer(elec_power_test,series='Test set')+
  autolayer(fit1$mean,series='Additive seasonal HW')+
  autolayer(fit2$mean,series='Multiplicative seasonal HW')+
  autolayer(fit3$mean,series='Damped additive seasonal HW')+
  autolayer(fit4$mean,series='Damped multiplicative seasonal HW')+
  xlab('Time (hr)') +
  ylab('Power(kW)')
```



RMSE

```
cat('Additive seasonal: RMSE Test set = ',sqrt(mean((fit1$mean-elec_power_test)^2)),'\n')
```

```
## Additive seasonal: RMSE Test set =  108.8472
```

```
cat('Multiplicative seasonal: RMSE Test set =',sqrt(mean((fit2$mean-elec_power_test)^2)),'\n')
```

```
## Multiplicative seasonal: RMSE Test set = 121.1528
```

```
cat('Damped additive seasonal: RMSE Test set=',sqrt(mean((fit3$mean-elec_power_test)^2)),'\n')
```

```
## Damped additive seasonal: RMSE Test set= 101.9794
```

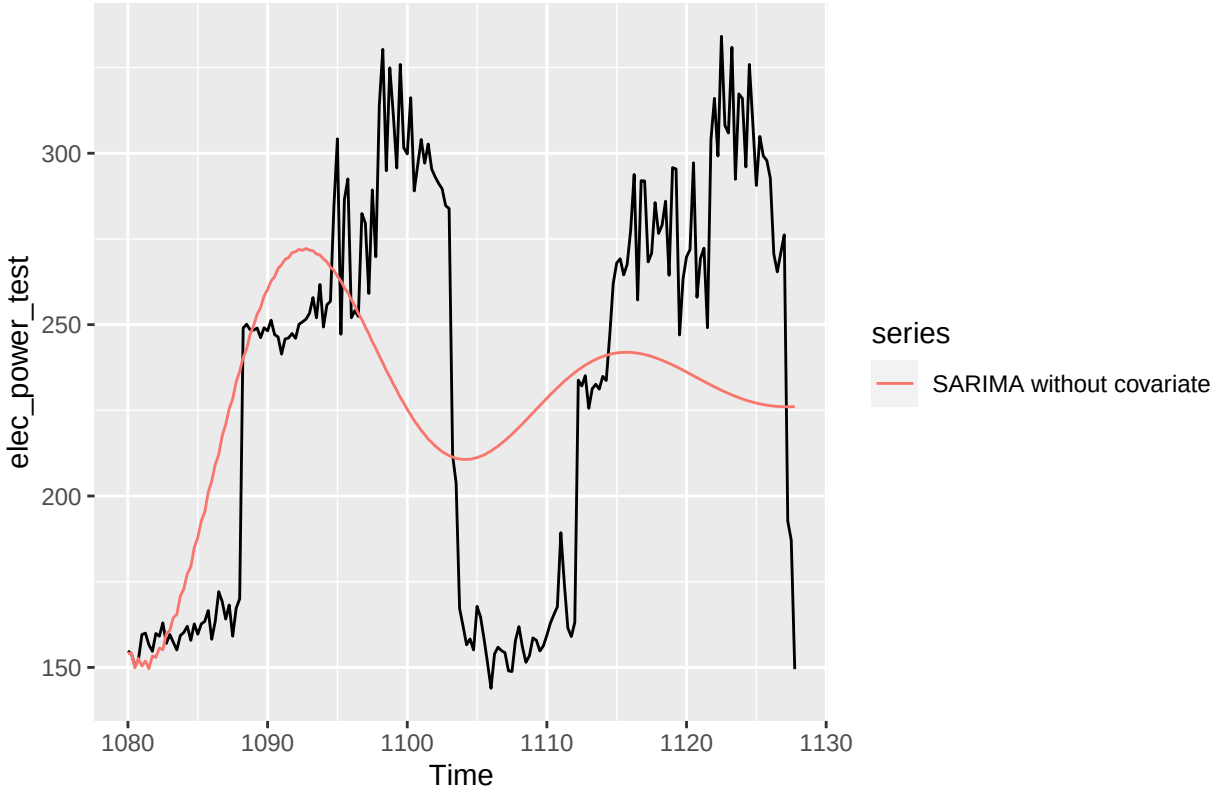
```
cat('Damped multiplicative seasonal: RMSE Test set=',sqrt(mean((fit4$mean-elec_power_test)^2)),'\n')
```

```
## Damped multiplicative seasonal: RMSE Test set= 106.6134
```

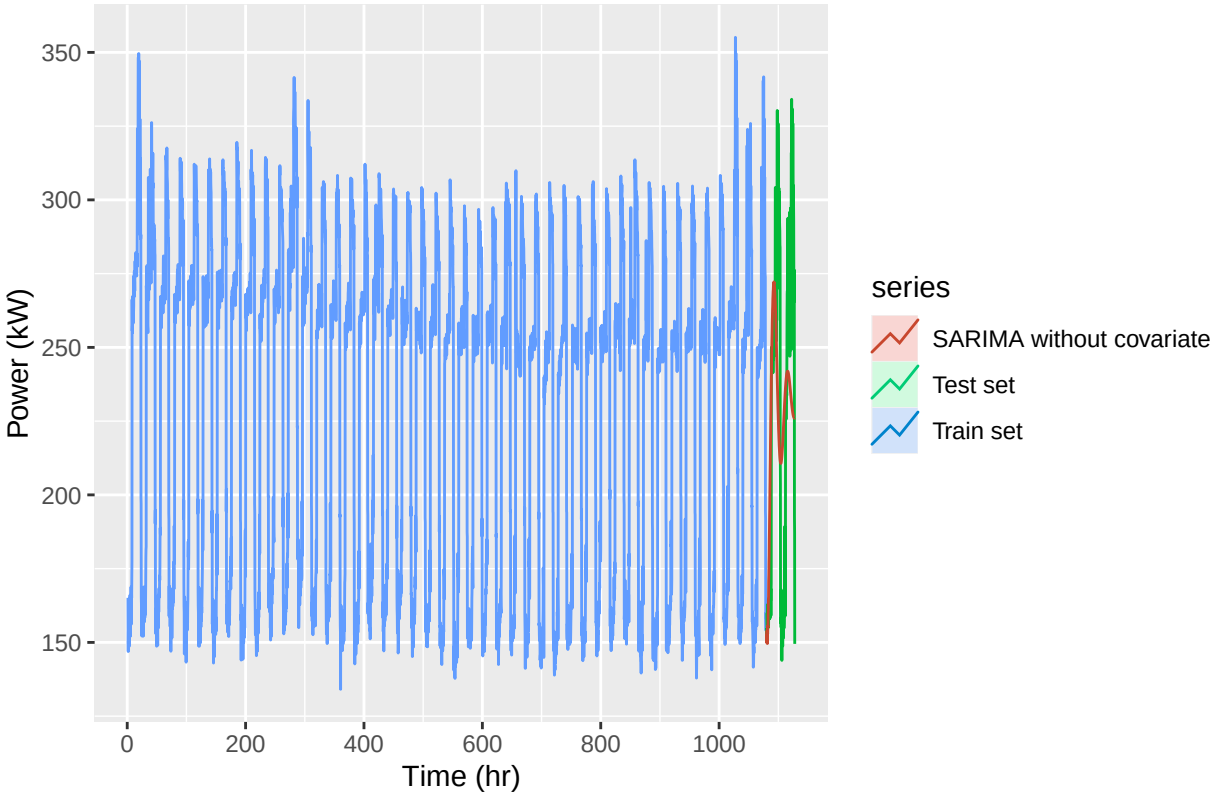
On conclue que pour l'instant le meilleur modèle est SES.opt , présentant l'erreur la plus faible avec un RMSE=19.03.

Forecasting with ARMA models

```
#SARIMA model.

elec_power_arima=auto.arima(elec_power_train)
prev_arima=forecast(elec_power_arima,h=192)
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)
```

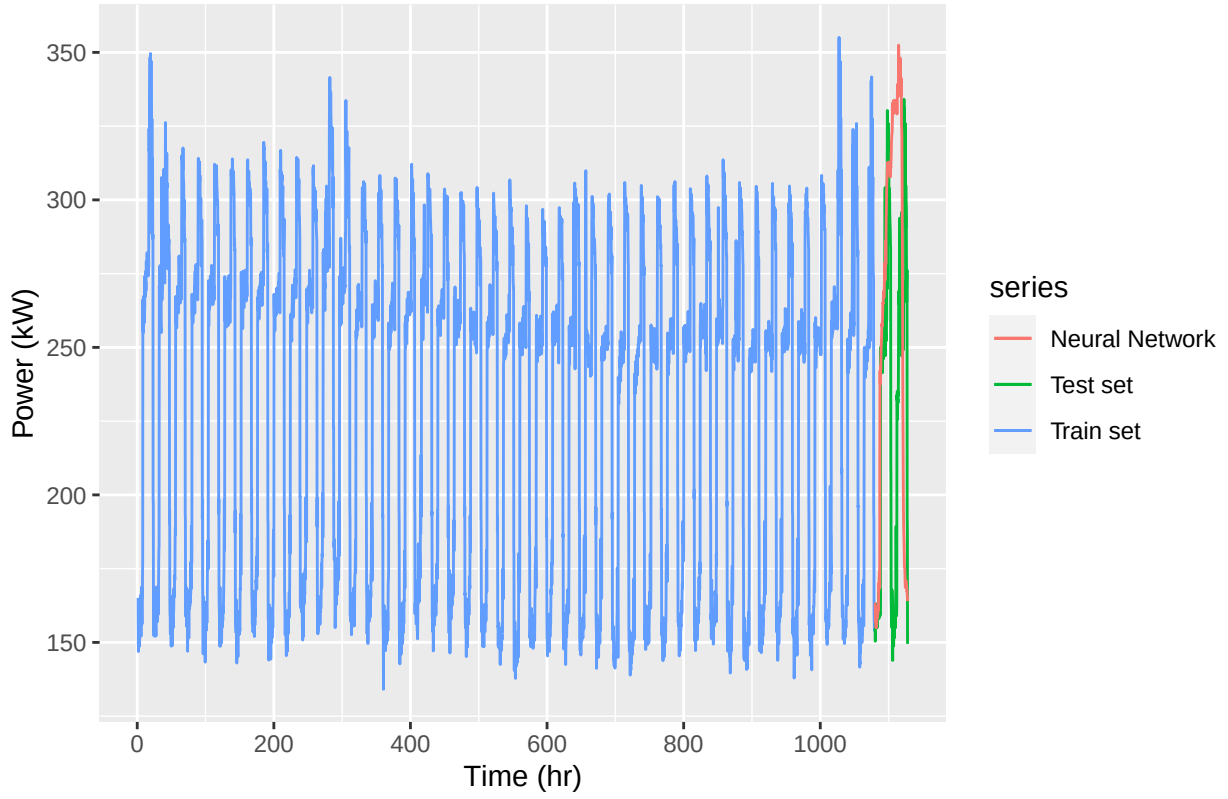
```
##                ME  RMSE   MAE   MPE  MAPE MASE ACF1 Theil's U
## Training set 0.01 14.52  7.62 -0.40  3.46 0.47 0.00        NA
## Test set     5.81 50.20 41.59 -2.58 18.72 2.57 0.92       2.9
```

Les résultats ne sont pas meilleurs avec ce modèle

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 = 192)
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)')
```



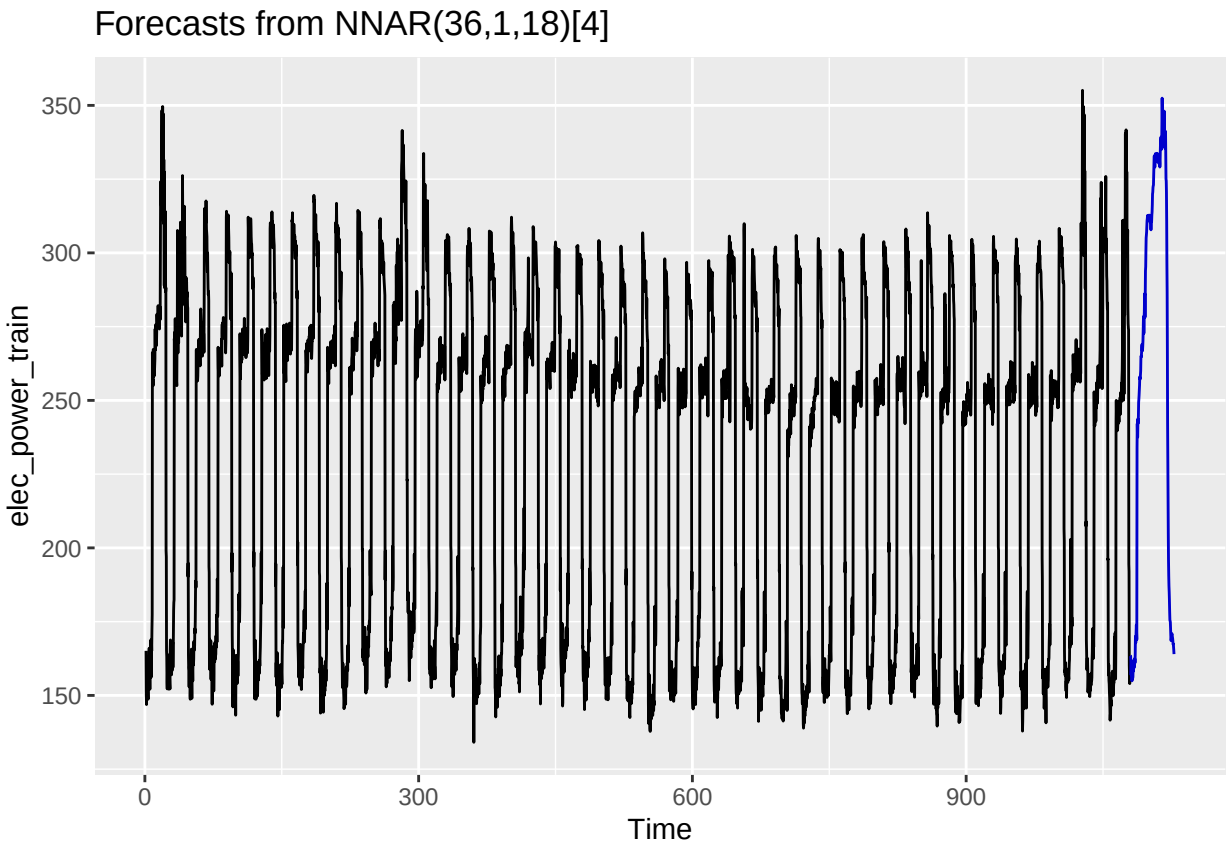

```
round(accuracy(pred_elec_power_nn, elec_power_test),2)
```

```
##                 ME  RMSE   MAE    MPE  MAPE MASE ACF1 Theil's U
## Training set   0.0  7.02  4.45  -0.13  2.00 0.28 0.01        NA
## Test set     -33.5 91.51 66.17 -21.73 32.77 4.10 0.97      6.26
```

```
print(elec_power_nn)
```

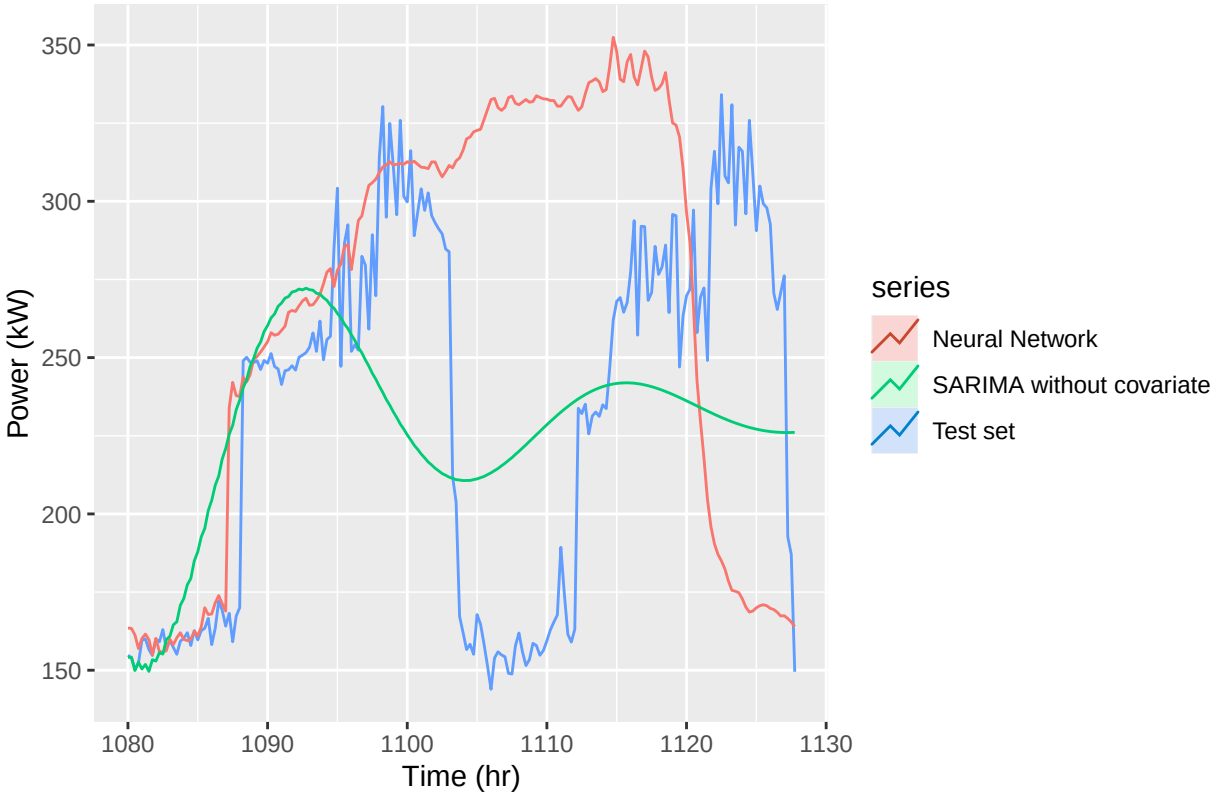
```
## Series: elec_power_train
## Model:  NNAR(36,1,18)[4]
## Call:   nnetar(y = elec_power_train)
##
## Average of 20 networks, each of which is
## a 36-18-1 network with 685 weights
## options were - linear output units
##
## sigma^2 estimated as 49.25
```

```
pred_elec_power_nn %>% forecast(h=192) %>% autoplot()
```



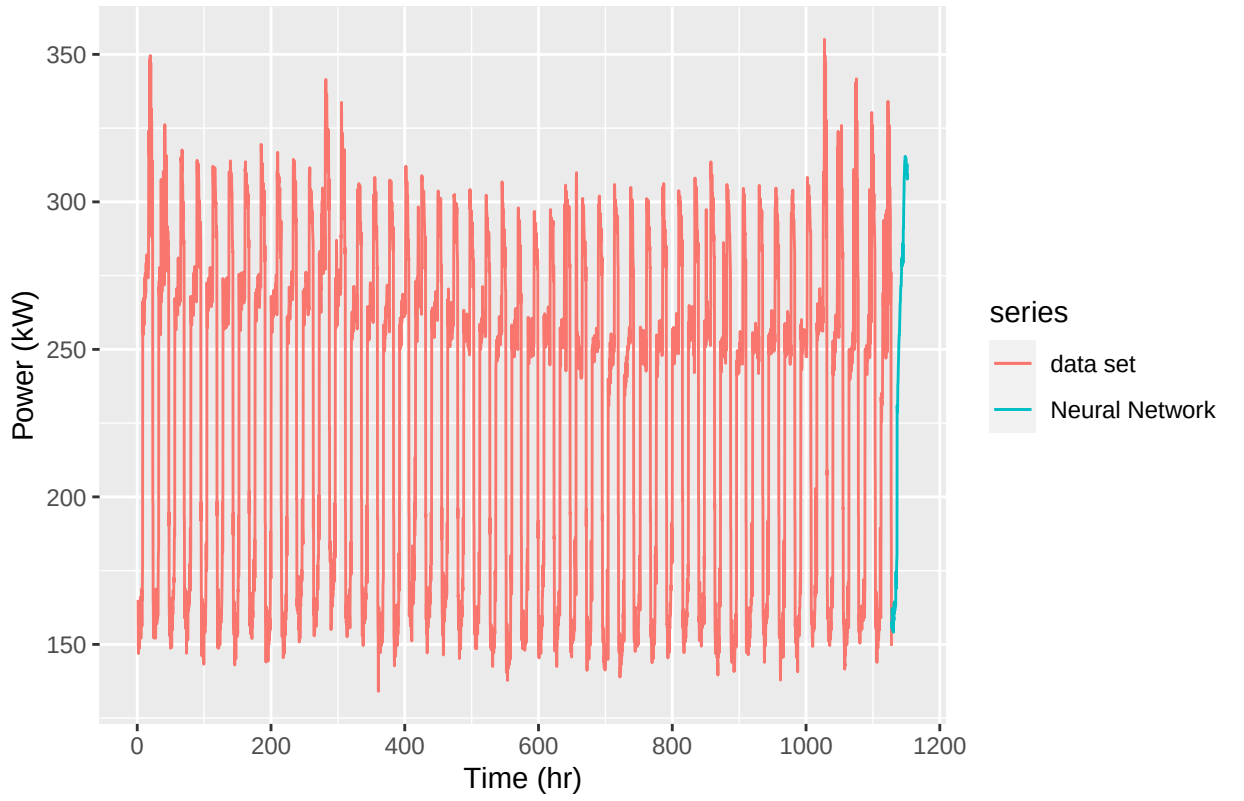
Les prévisions sont moins efficaces qu'avec les modèles ou SARIMA.

```
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)')
```



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

```
elec_power_pred = nnetar(elec_power)
pred_elec_power_pred = forecast(elec_power_pred, h = 96)
autoplot(elec_power,series="data set") +

  autolayer(pred_elec_power_pred$mean,series='Neural Network')+
  xlab('Time (hr)') +
  ylab('Power (kW)')
```



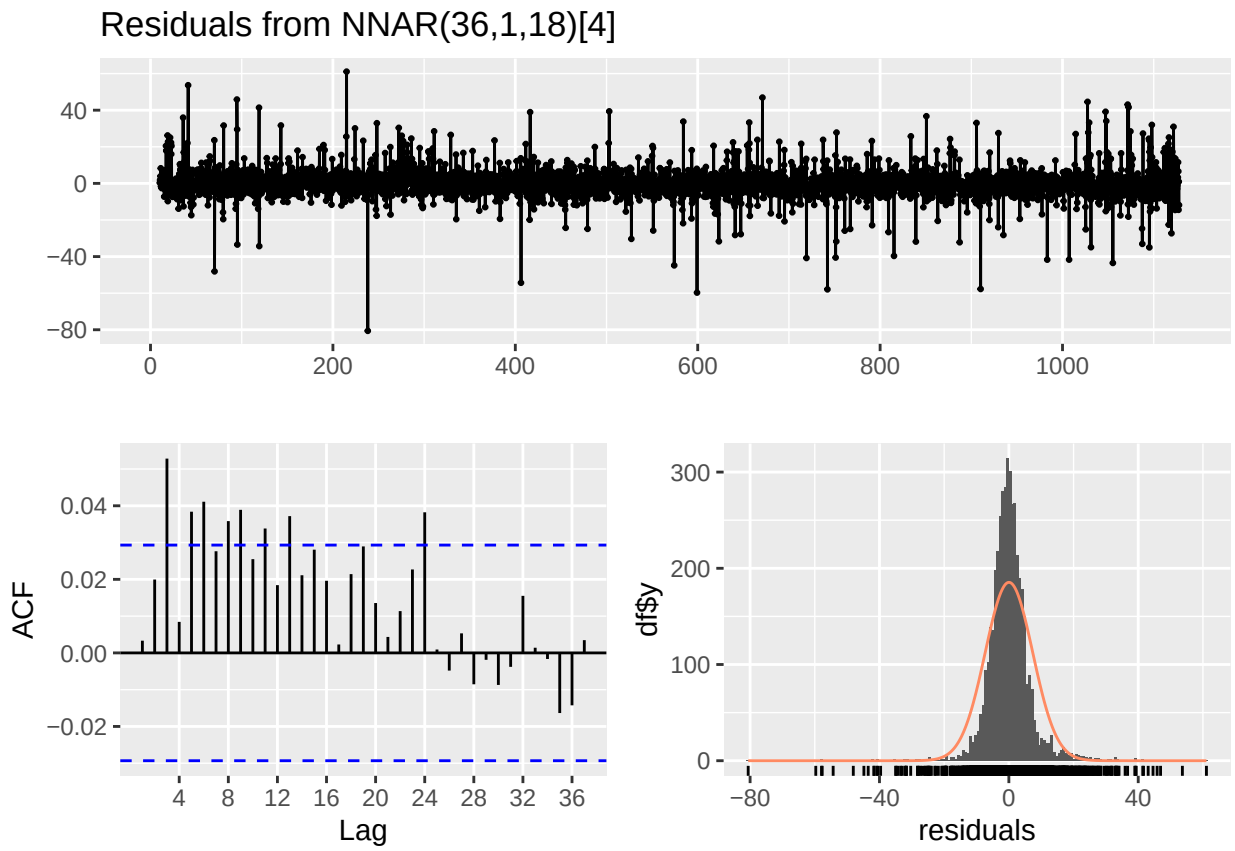

```
# Results
Pred = print(pred_elec_power_pred)
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 1128 156.7791 157.5305 155.5460 162.1317
## 1129 159.7060 160.1784 154.7191 156.0015
## 1130 159.5955 156.2688 155.4688 154.1379
## 1131 159.2708 158.9910 158.2536 164.0464
## 1132 160.6999 164.3967 161.4885 161.8050
## 1133 161.6432 164.5954 162.8008 164.3068
## 1134 163.2817 169.3182 174.6921 171.6016
## 1135 169.0070 173.1097 177.8139 178.9326
## 1136 180.1819 230.8660 229.4437 228.6958
## 1137 230.5976 238.1554 238.2551 241.2198
## 1138 244.5804 246.2994 249.1246 251.8188
## 1139 252.7511 254.9528 254.9522 256.3534
## 1140 258.7591 261.1005 263.4560 265.6339
## 1141 265.7992 268.6942 270.3827 271.4395
## 1142 271.5085 271.5104 274.4170 275.3139
## 1143 280.3734 280.8800 279.5855 278.2607
## 1144 281.6578 280.4057 285.9086 285.5169
## 1145 279.2497 286.5919 291.2270 291.2325
## 1146 297.4068 300.8612 302.7100 307.4684
## 1147 308.9821 310.9865 313.4814 314.0250
## 1148 314.6733 315.4193 314.9945 314.8720
## 1149 315.1519 314.4218 314.5013 312.9340
```

```
## 1150 311.5844 311.7941 310.6957 311.9759
## 1151 312.7300 308.6508 307.7192 309.6575
```

```
#Checking model

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

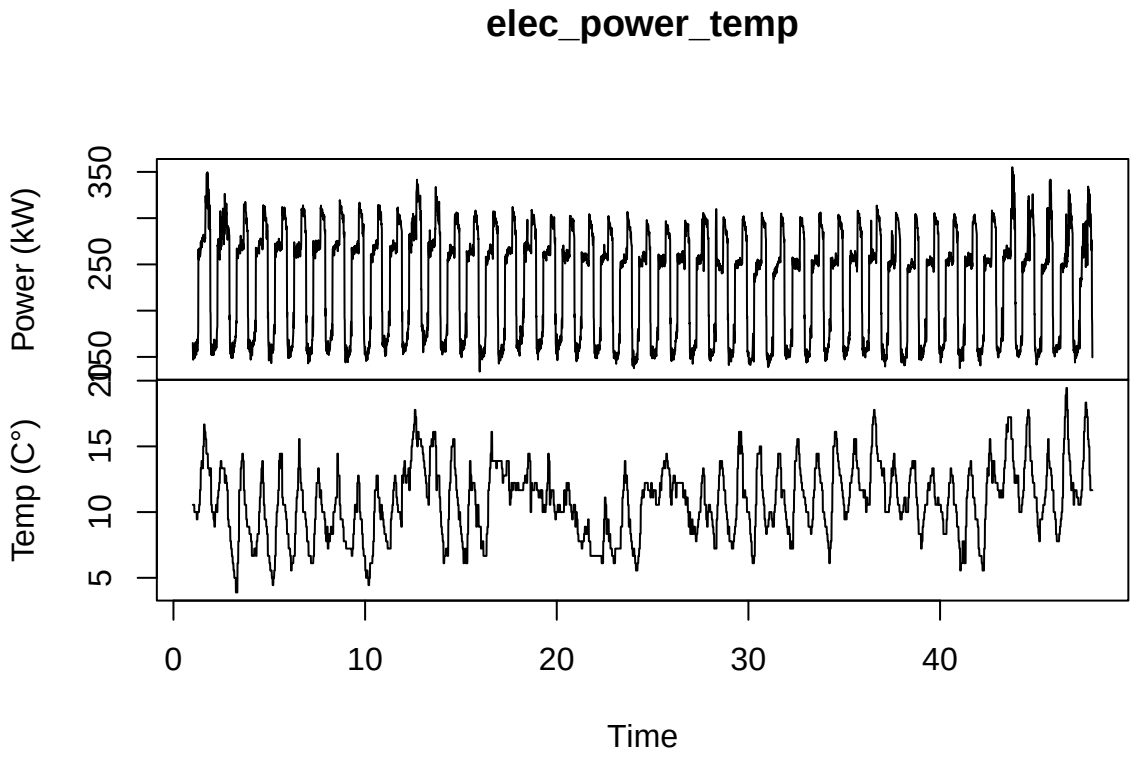
```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```



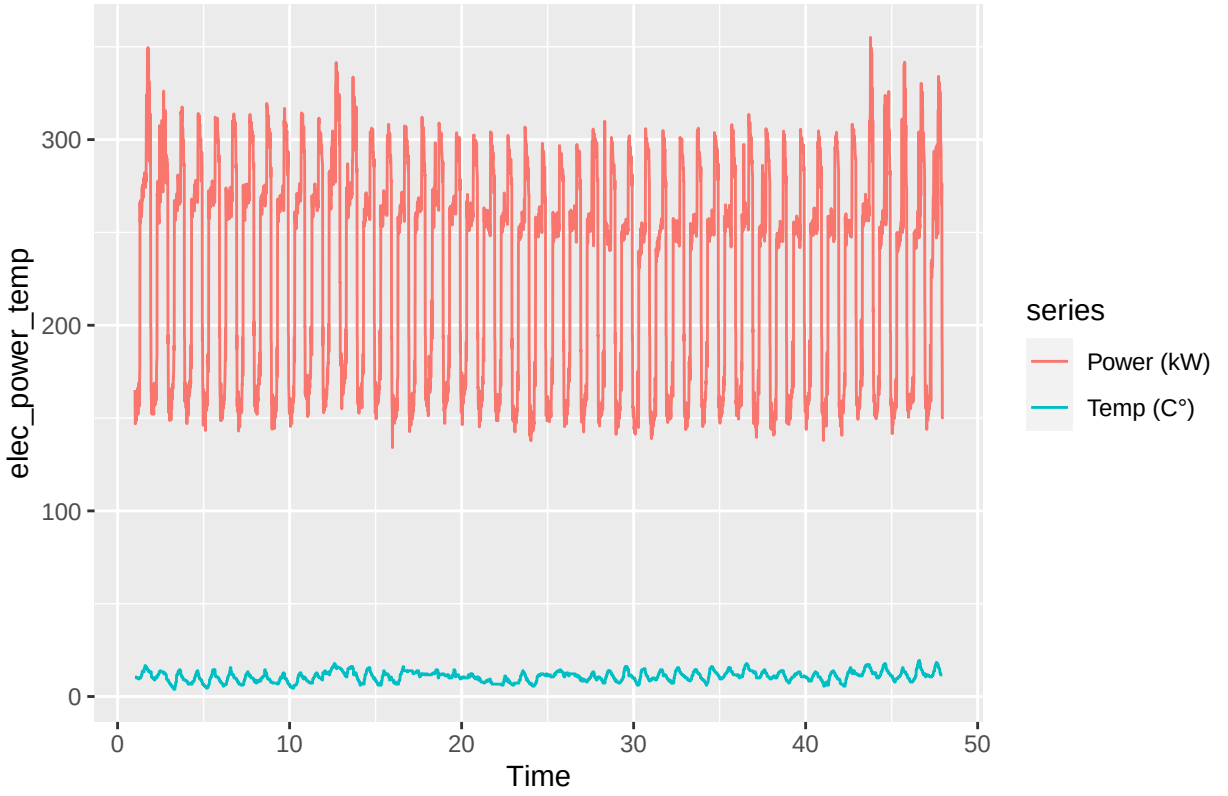

```
#write_csv(Pred,file="Pred_sans_temperature.csv")
#write.xlsx(Pred, file, sheetName = "Sheet1",
 # col.names = TRUE, row.names = TRUE, append = FALSE)
```

Part 2: Forecast electricity consumption by using outdoor temperature

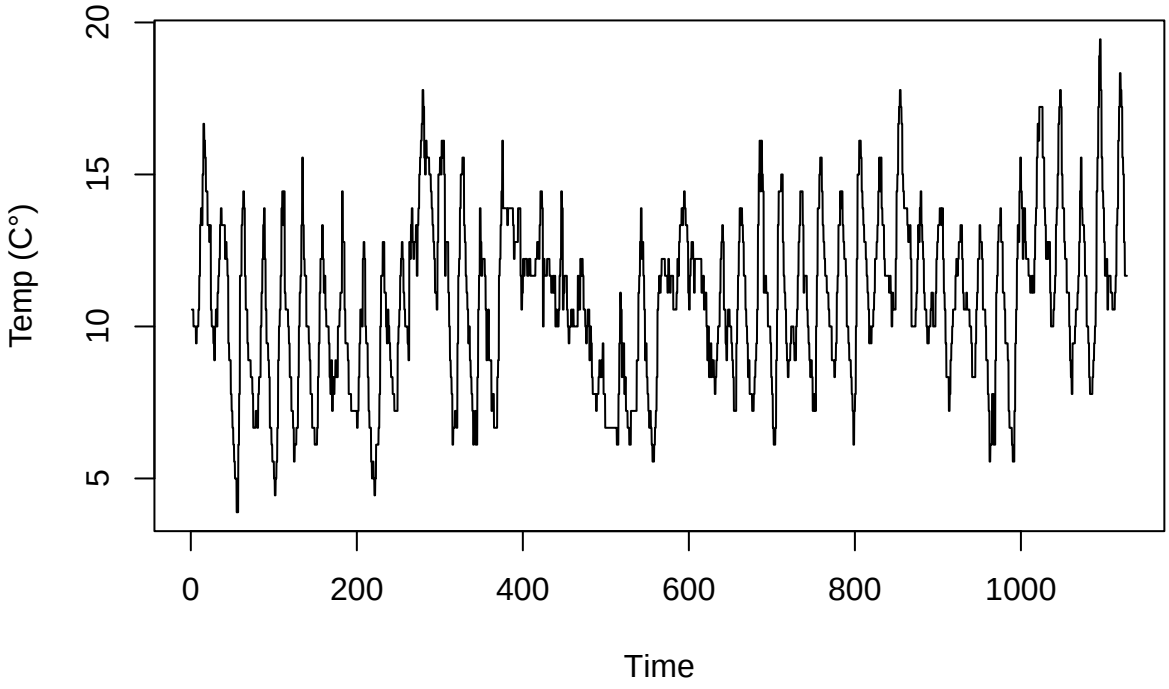
```
elec_power_temp<-ts(data[1:4507,2:3],start=c(1,2),freq=96)
plot(elec_power_temp)
```



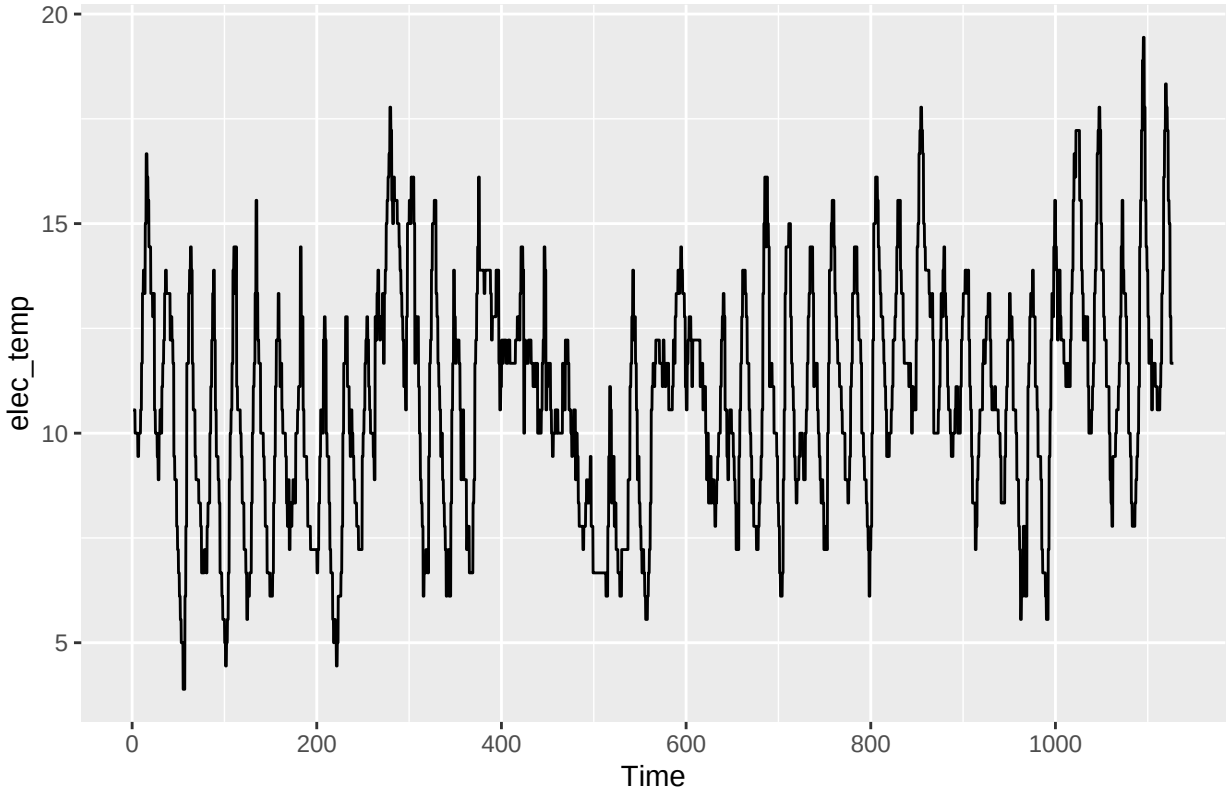

```
autoplot(elec_power_temp)
```




```
elec_temp<-ts(data[1:4507,3],start=c(1,2),freq=4)
plot(elec_temp)
```



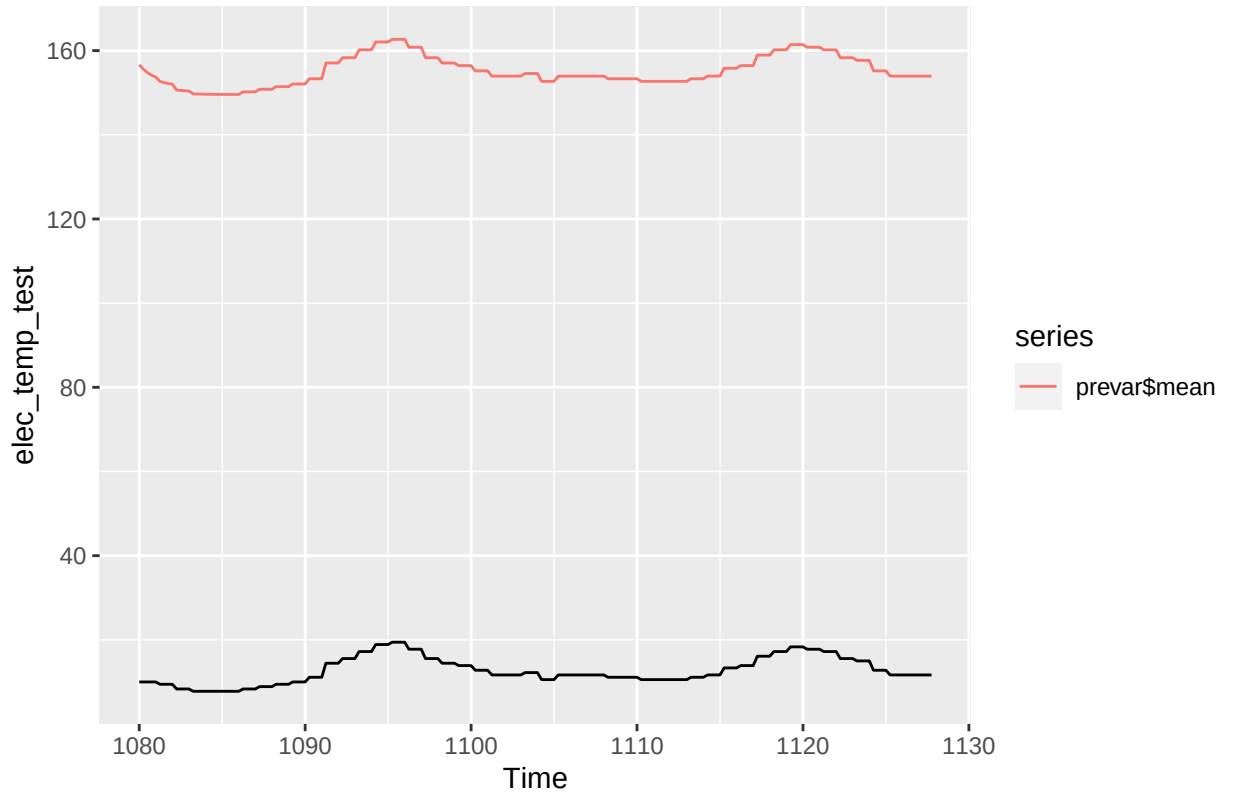

```
autoplot(elec_temp)
```




```
#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,4315)
elec_temp_test=tail(elec_temp,192)
```

Forecasting SARIMA model:

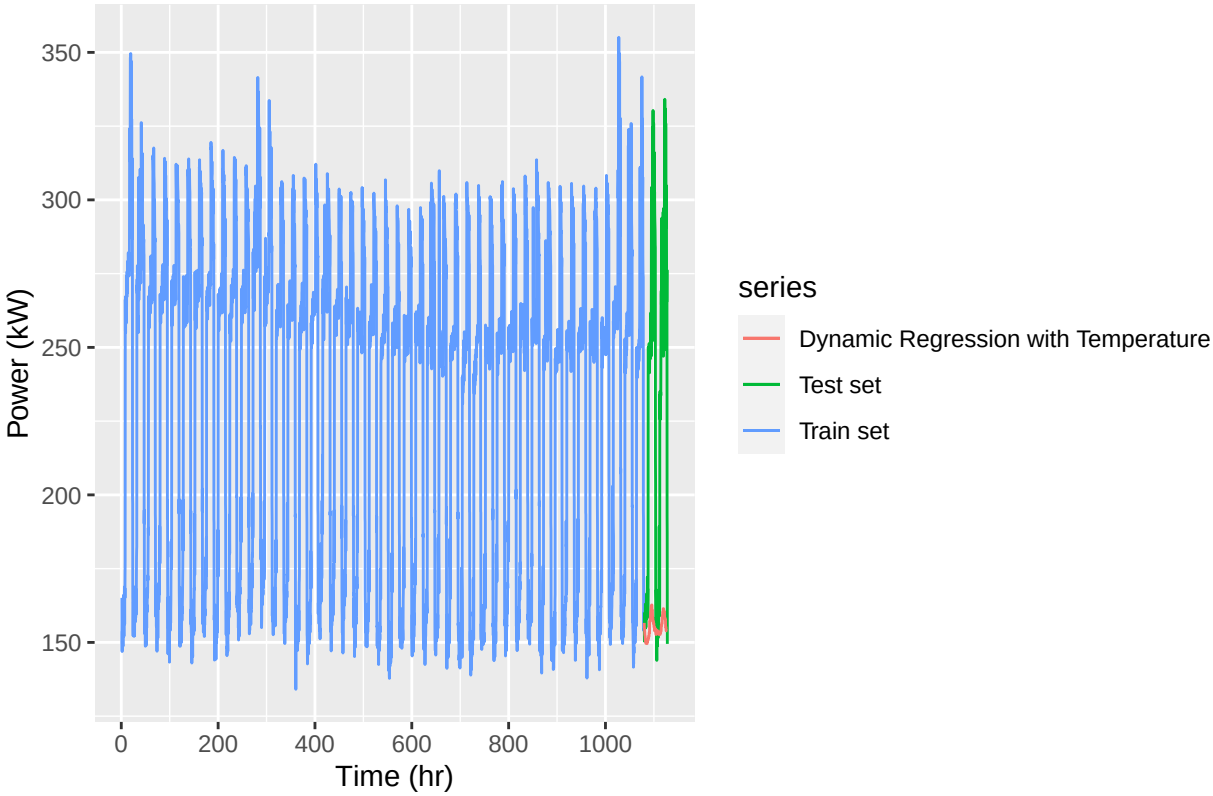
```
elec_power_temp_train_ar=auto.arima(elec_power_train,xreg=elec_temp_train)
prevar=forecast(elec_power_temp_train_ar,h=192,xreg=elec_temp_test)
autoplot(elec_temp_test)+autolayer(prevar$mean)
```




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

```
## [1] 96.23036
```

```
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)')
```

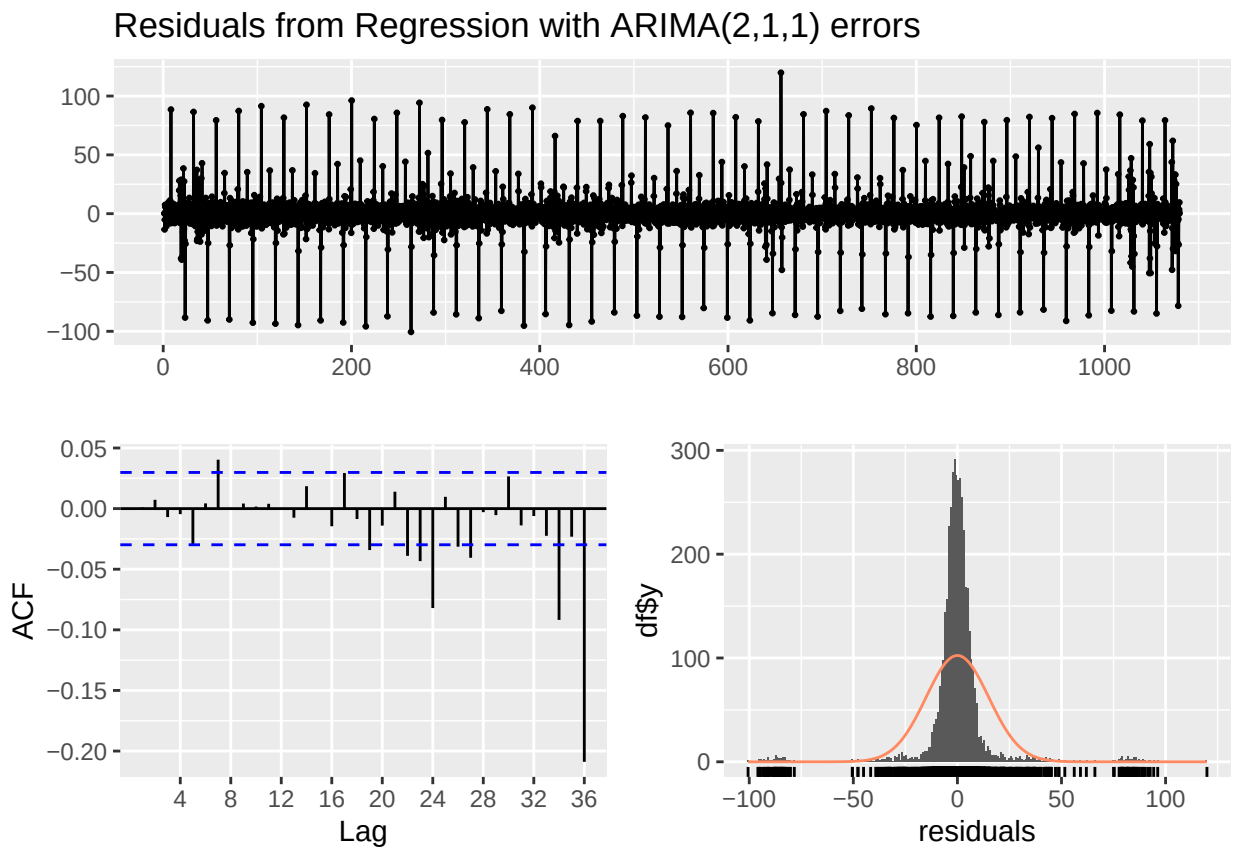


vérifions le résidu, il y a encore des autocorrélations :

```
summary(elec_power_temp_train_ar)
```

```
## Series: elec_power_train
## Regression with ARIMA(2,1,1) errors
##
## Coefficients:
##          ar1     ar2      ma1  Temp (C°)
##       0.6563  0.0895  -0.6940     1.1238
## s.e.  0.0951  0.0159   0.0951     0.4967
##
## sigma^2 = 227.2:  log likelihood = -17822.58
## AIC=35655.16   AICc=35655.17   BIC=35687
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE   MAPE      MASE
## Training set -0.002298962 15.06354 7.119902 -0.1702121 3.2395 0.4407687
##                      ACF1
## Training set 0.0008762523
```

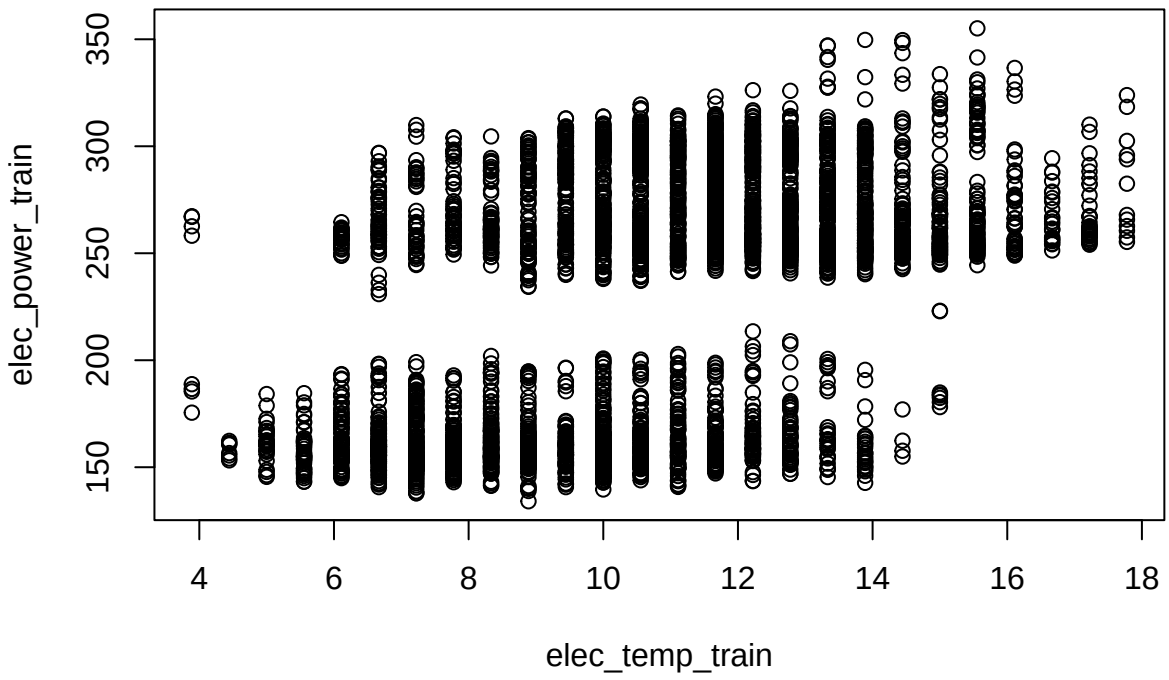
```
#autocorrelation of residuals
checkresiduals(elec_power_temp_train_ar,test="LB",plot=TRUE)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,1) errors
## Q* = 11.306, df = 5, p-value = 0.04564
##
## Model df: 3.   Total lags used: 8
```

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

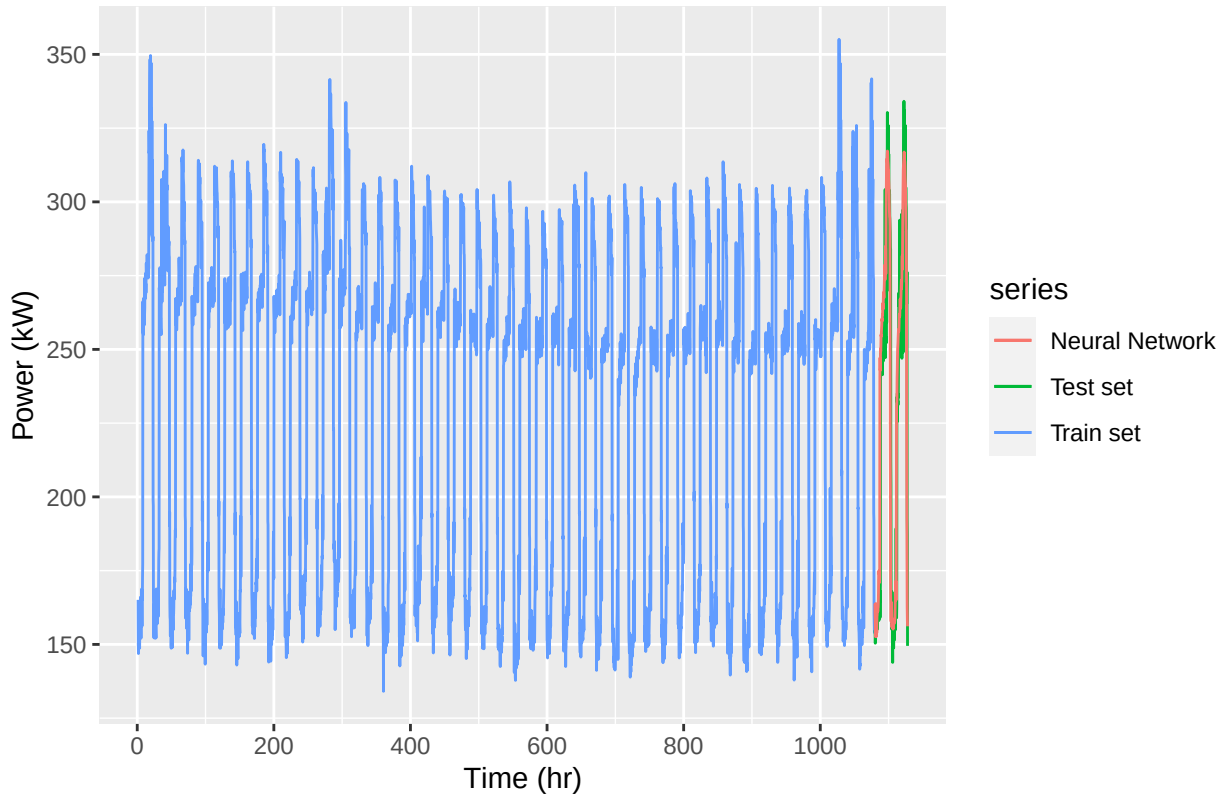
```
plot(elec_temp_train,elec_power_train)
```



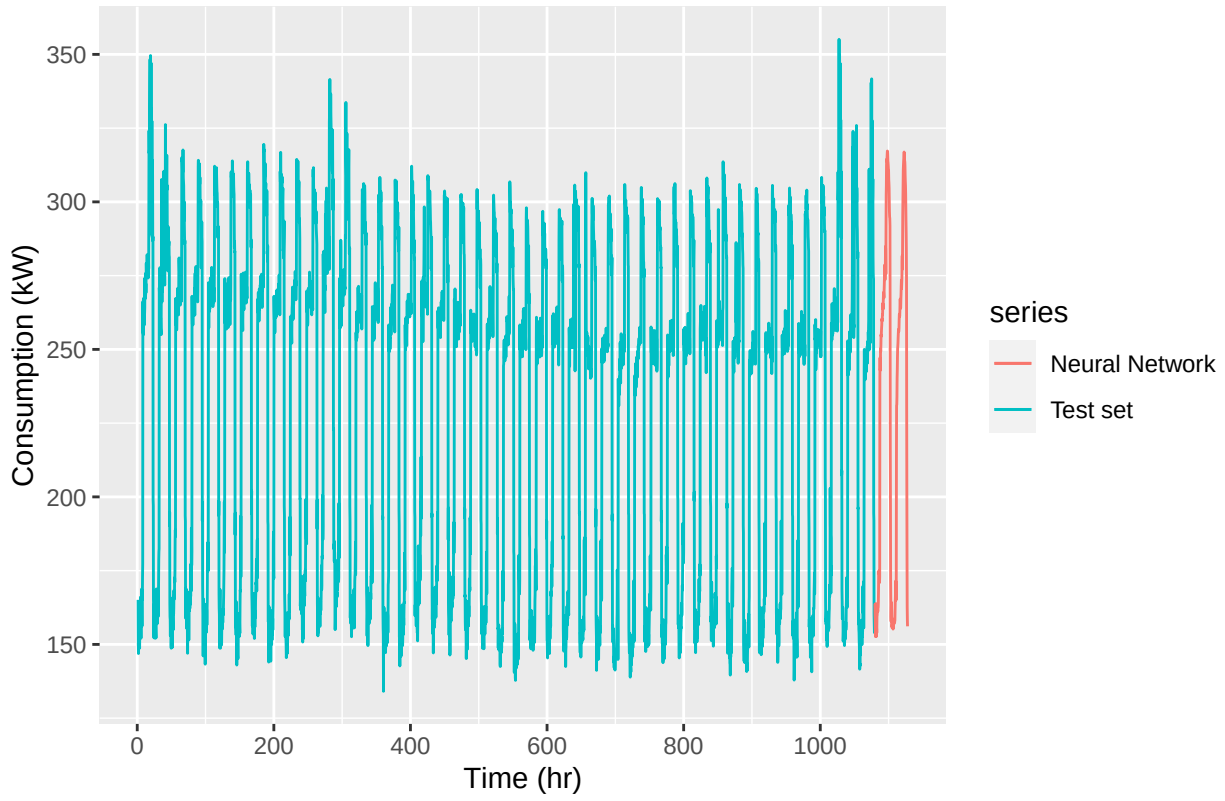
Forecasting with Neural Network

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 = 192)
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('Consumption (kW)')
```




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

```
## [1] 25.77891
```

Le RSME est le plus bas,C'est donc le meilleur modèle.