Time Series Analysis

Kamila Tukhvatullina

```
#install.packages("openxlsx")
library(openxlsx)
```

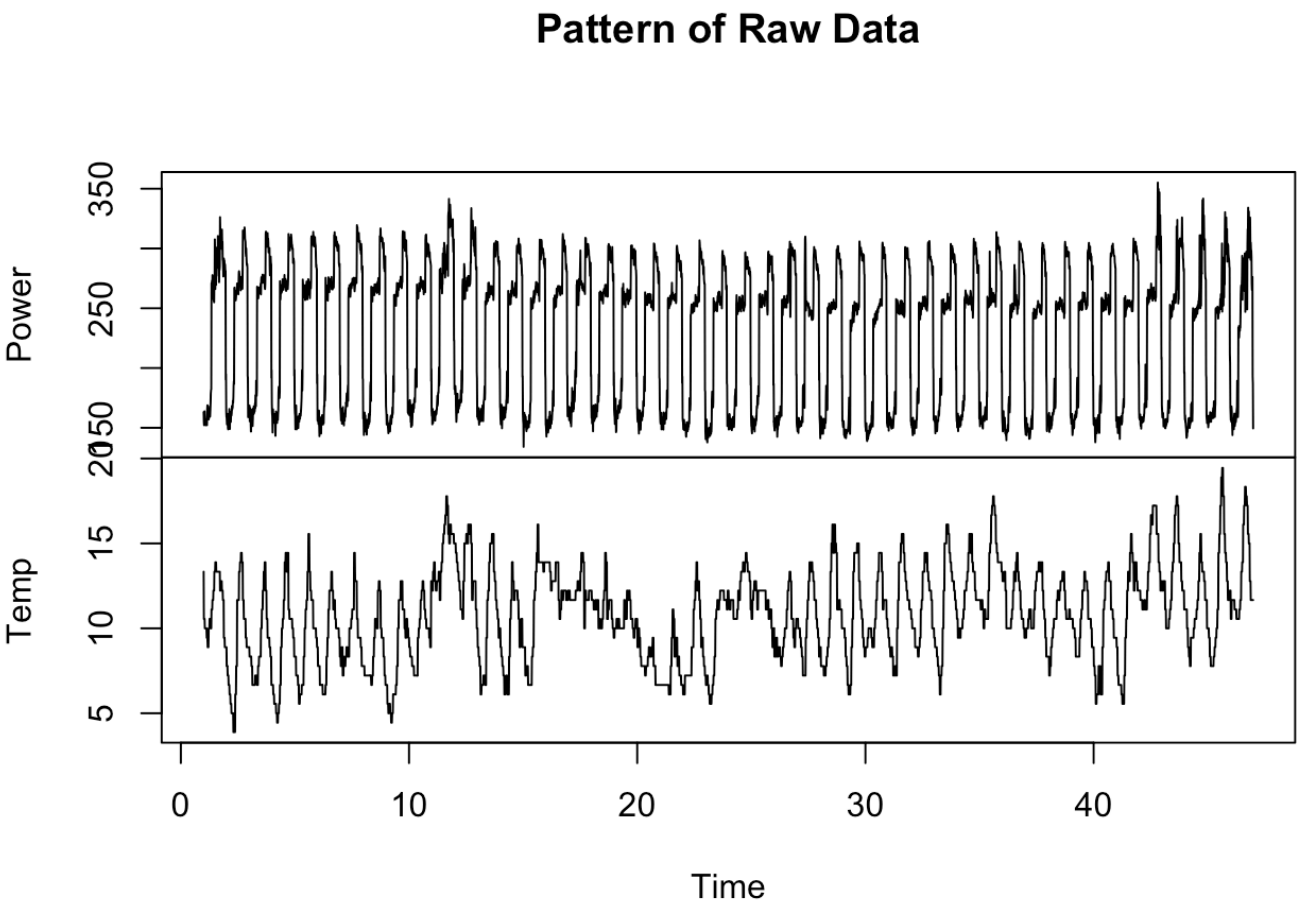
we’re using this library to correctly read the file with dataset

```
elec_train = read.xlsx("/Users/kamila/Downloads/Elec-train.xlsx", colNames = TRUE)
library(data.table)
nms <- c("TimeS","Power", "Temp")
setnames(elec_train, nms)
elec_train = elec_train[,c("Power", "Temp")]
elecc = ts(elec_train[92:4507,], freq = 96)   #this was changed later on as I disc
overed day 1 doesn't have 96 observations
head(elecc)
```

```
##       Power     Temp
## [1,] 163.1 13.33333
## [2,] 154.4 10.55556
## [3,] 152.2 10.55556
## [4,] 158.7 10.55556
## [5,] 163.8 10.55556
## [6,] 158.7 10.00000
```

I’ve chosen frequncy = 96 as it’s the amount of observations we get in a period of 1 day. Let’s make a simple plot and see data that we have:

```
plot.ts(elecc, main = "Pattern of Raw Data", ylab = "electricity", xlab = "Time")
```



- from the first glance we can alredy see a strong periodic pattern, trend is not evident here, might be very minor: to be checked

```
library(forecast)
```

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

```
#another (more beautiful) way of plotting data
library(ggplot2)
autoplot(elecc) +
      ggtitle('Electricity consumption over time') + xlab('time') +
      ylab('consumption')
```

see the pattern of the data, how many are missing

```
#install.packages("mice")
library(mice)
```

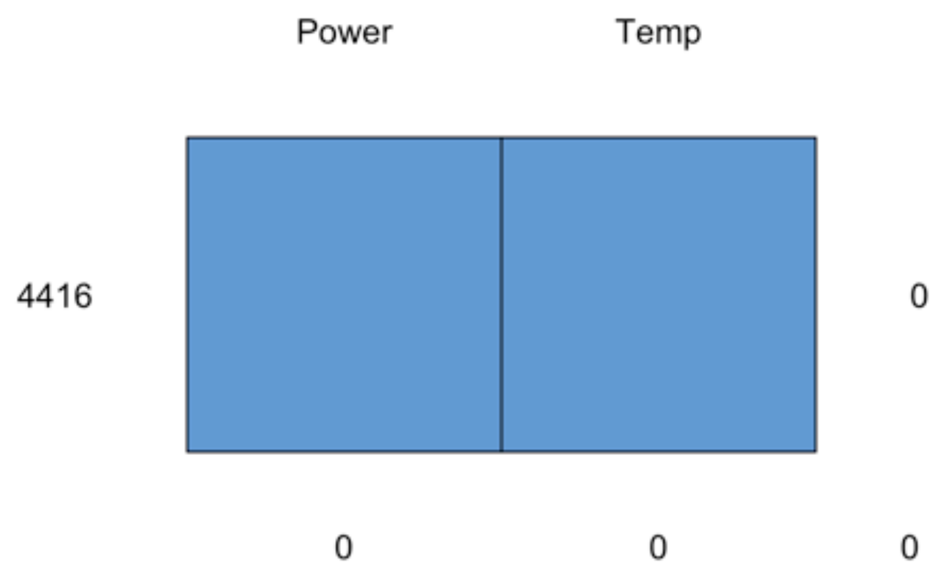
```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
md.pattern(elecc)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```




```
##      Power Temp
## 4416     1    1 0
##          0    0 0
```

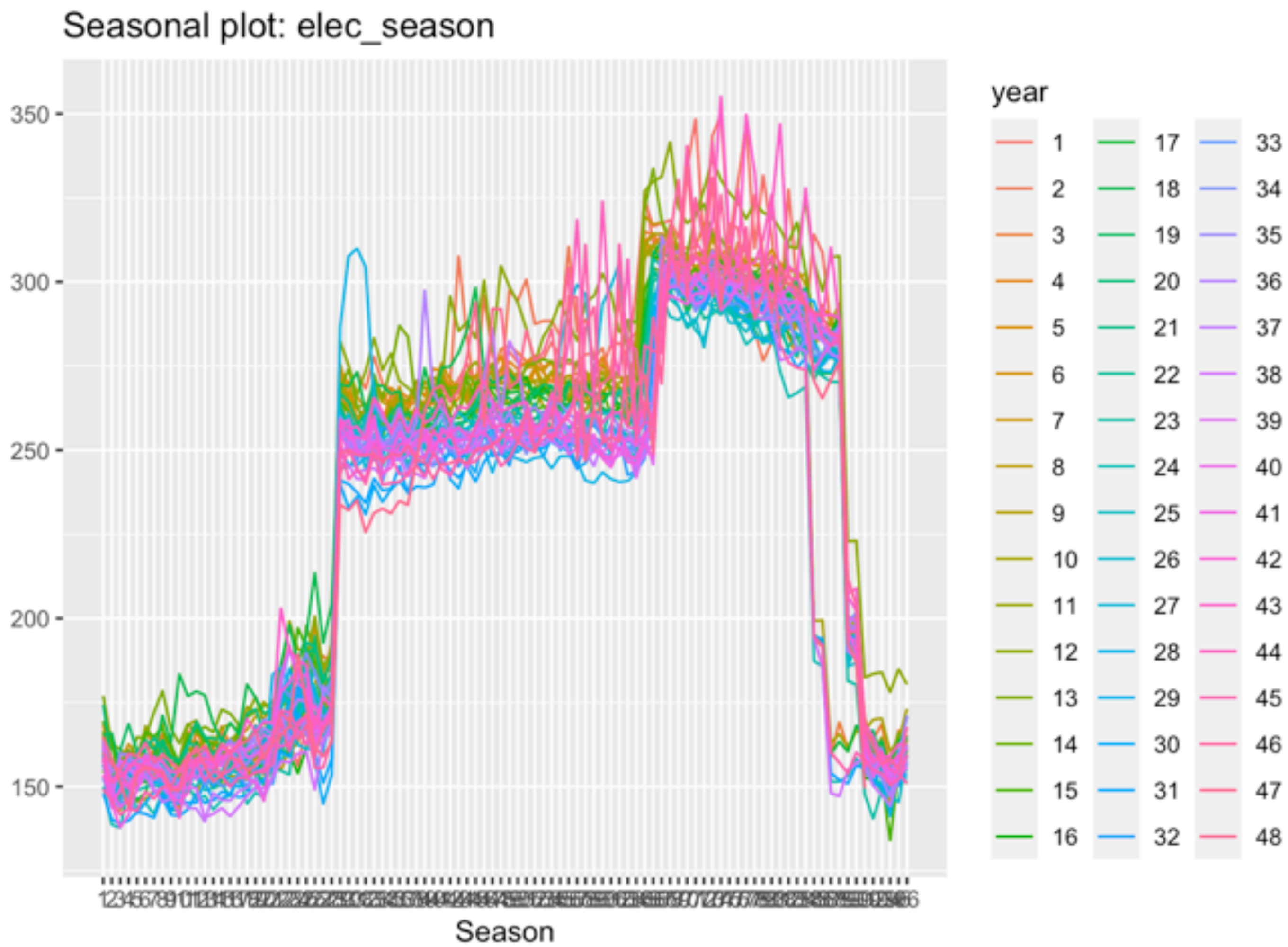
indeed, we’re missing a day of electricity consumption data. We’re not going to use mice library for this, but we will predict the data with time series forecasting techniques. (btw we have reduced dataset to have full periods only, which means we deleted day 1 and the last day with no observations)

```
require(forecast)
```

seasonality plot:

```
elec_season = ts(elec_train$Power, freq = 96)
ggseasonplot(elec_season,hour.labels= TRUE,hour.labels.left=TRUE)
```

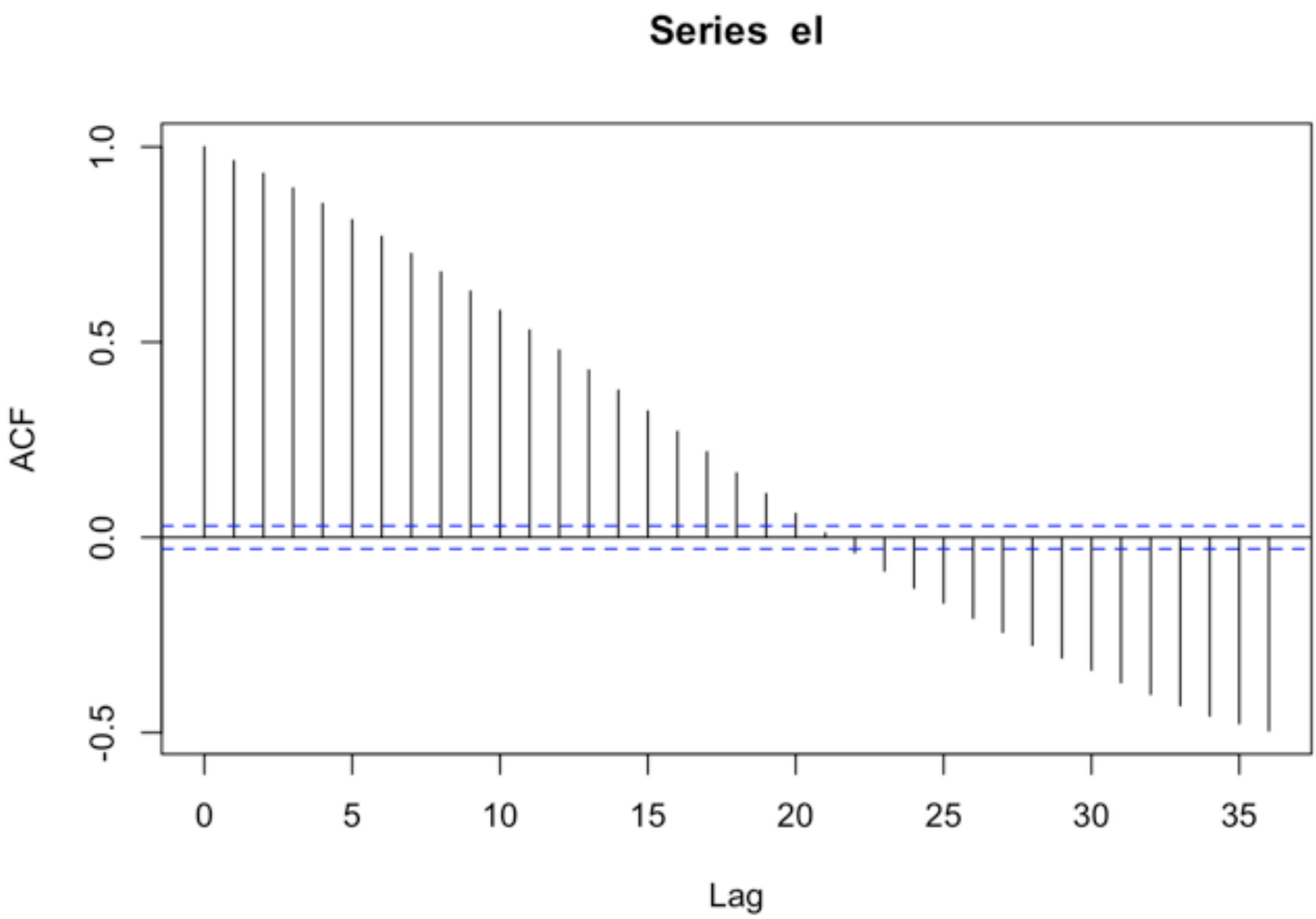
```
## Warning: Removed 96 row(s) containing missing values (geom_path).
```



data is very periodic: electricity consumption does not vary much from day to day

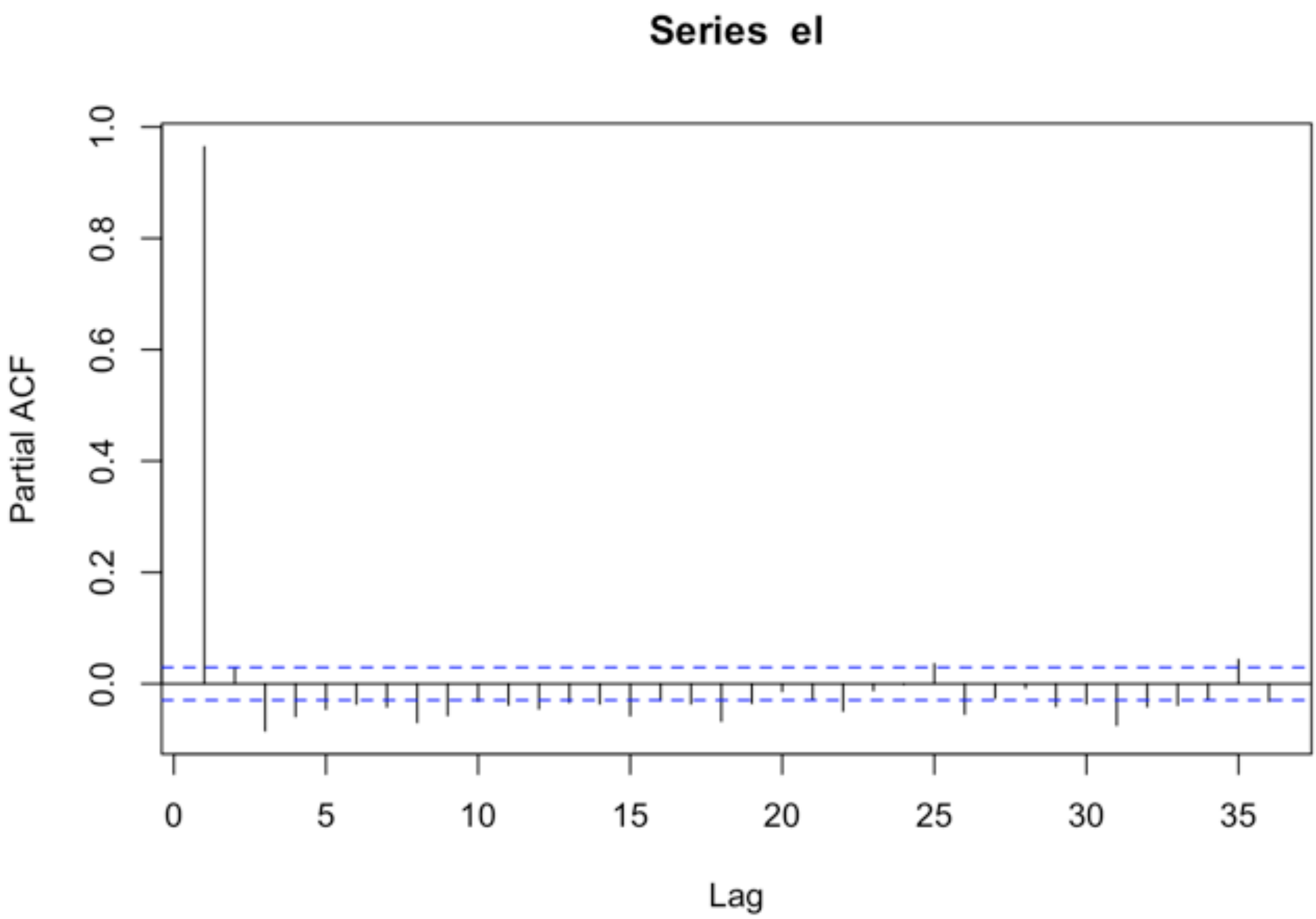
let's see autocorrelation :

```
el = elec_train$Power[92:4507]    #problem of missing values
el_temp = elec_train$Temp[92:4507]
acf(el)
```



there are significatnt autocorrelations almost everywhere, let’s see what partial acf will show:

```
pacf(el)
```



significant autocorrelation of 3d, 4th, 8,9 etc orders.

divide into train and test:

```
el_only = ts(el, start=c(1, 6), freq=96)

serie_train = window(el_only, start=c(1,6), end=c(43,96))
serie_test = window(el_only, start=c(44,1), end=c(47,96))
```

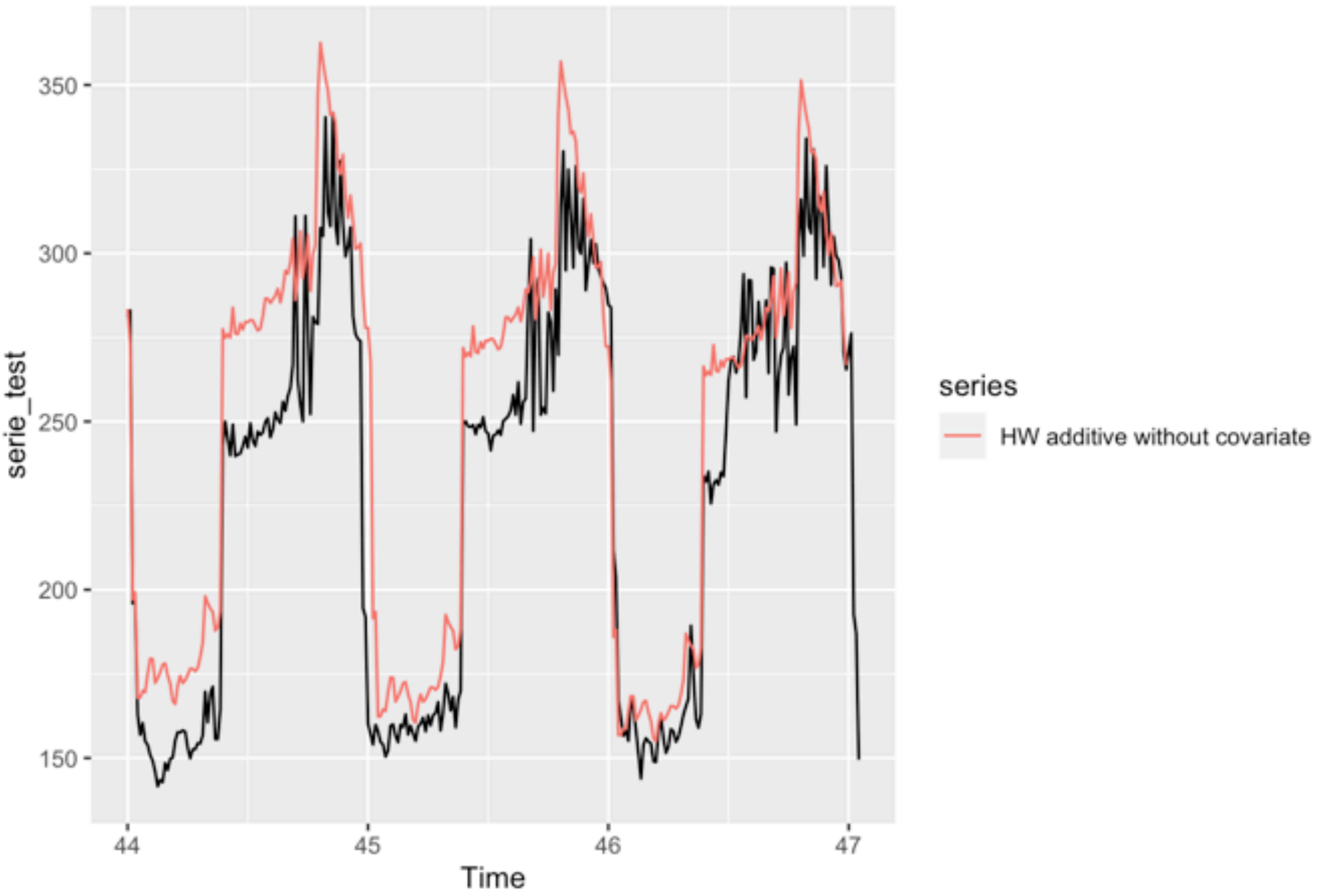
```
## Warning in window.default(x, ...): 'end' value not changed
```

```
#serie_train=window(el_only,start=c(1,1),end=c(4,10))
#serie_test=window(el_only,start=c(297,11),end=c(300,10))

s_train=window(elecc,start=c(1,1),end=c(43,96))
s_test=window(elecc,start=c(44,1),end=c(47,96))
```

```
## Warning in window.default(x, ...): 'end' value not changed
```

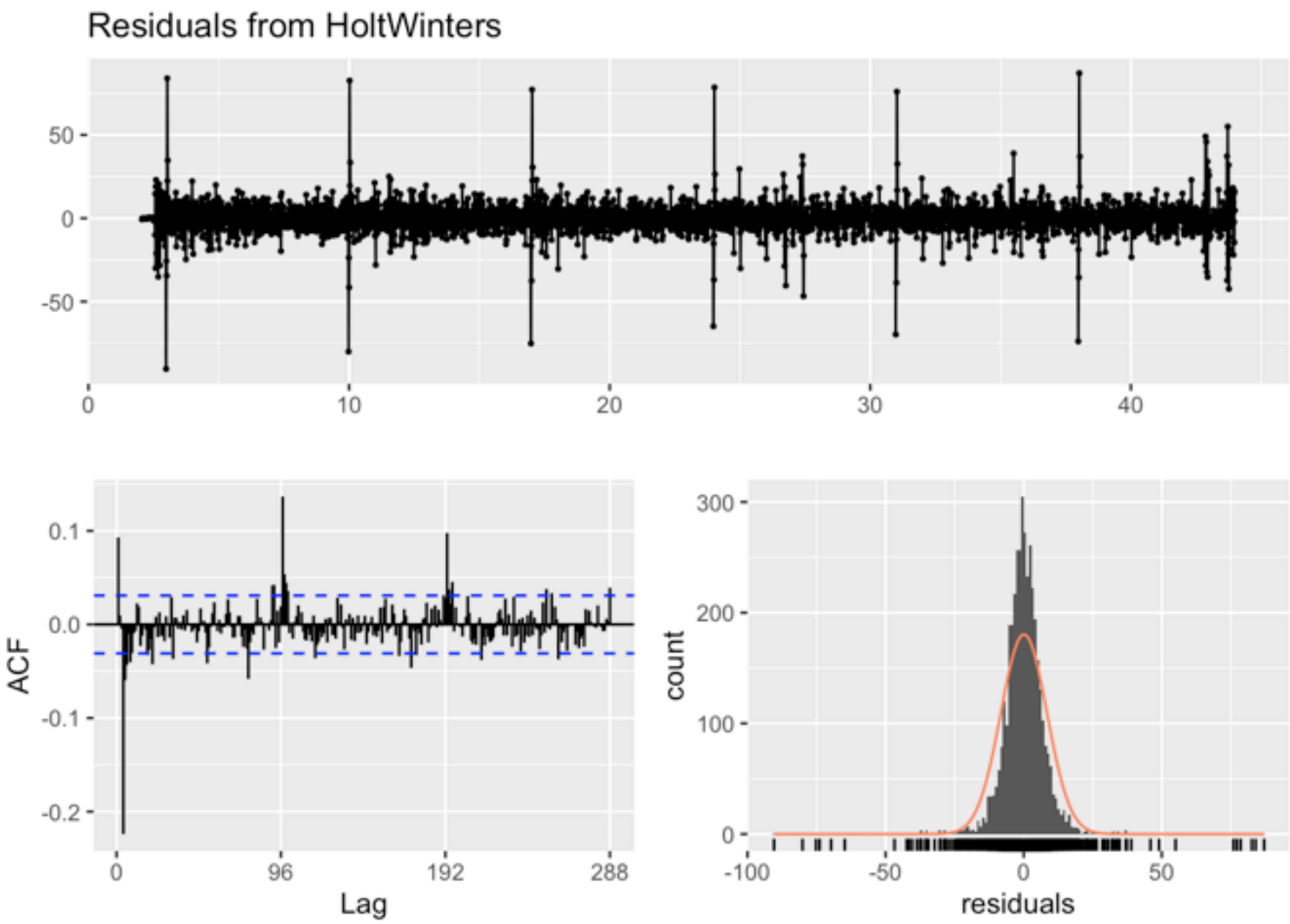
let’s apply H-W exponential smoothing model with no covariate:

```
#fit=hw(serie_train,lambda="auto")
#prev=forecast(fit,h=28)
#autoplot(prev) + autolayer(serie_train, series="true data")+
#autolayer(prev$mean, series="HW forecasts")
#checkresiduals(fit)


fit_hw = HoltWinters(serie_train, alpha=NULL, beta=NULL, gamma=NULL, seasonal='add
itive')

prev=forecast(fit_hw, h=288)
autoplot(serie_test) + autolayer(prev$mean,series="HW additive without covariate")
```




```
checkresiduals(fit_hw)
```

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




```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

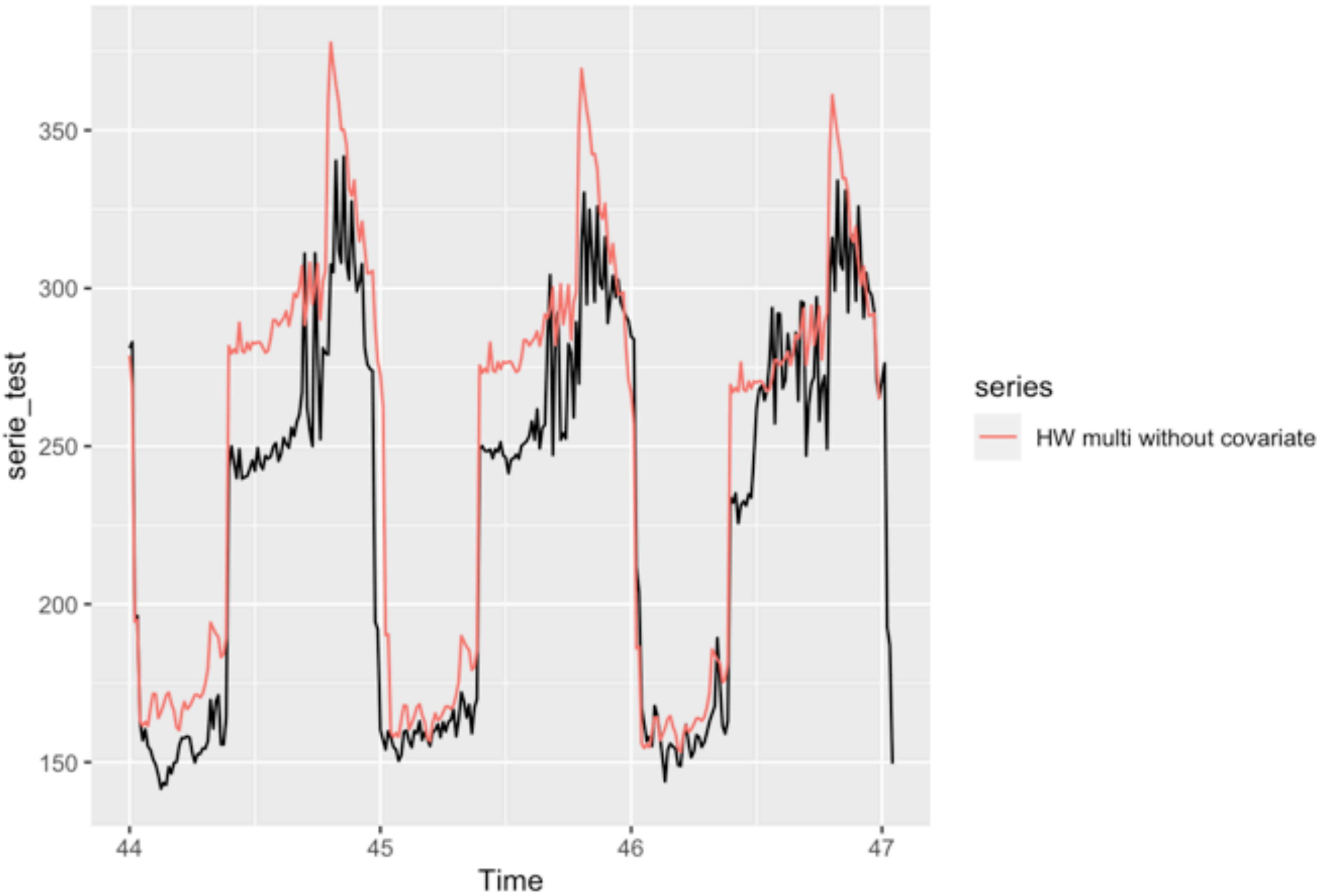
```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
rmse(serie_test, prev$mean)
```

```
## [1] 26.99426
```

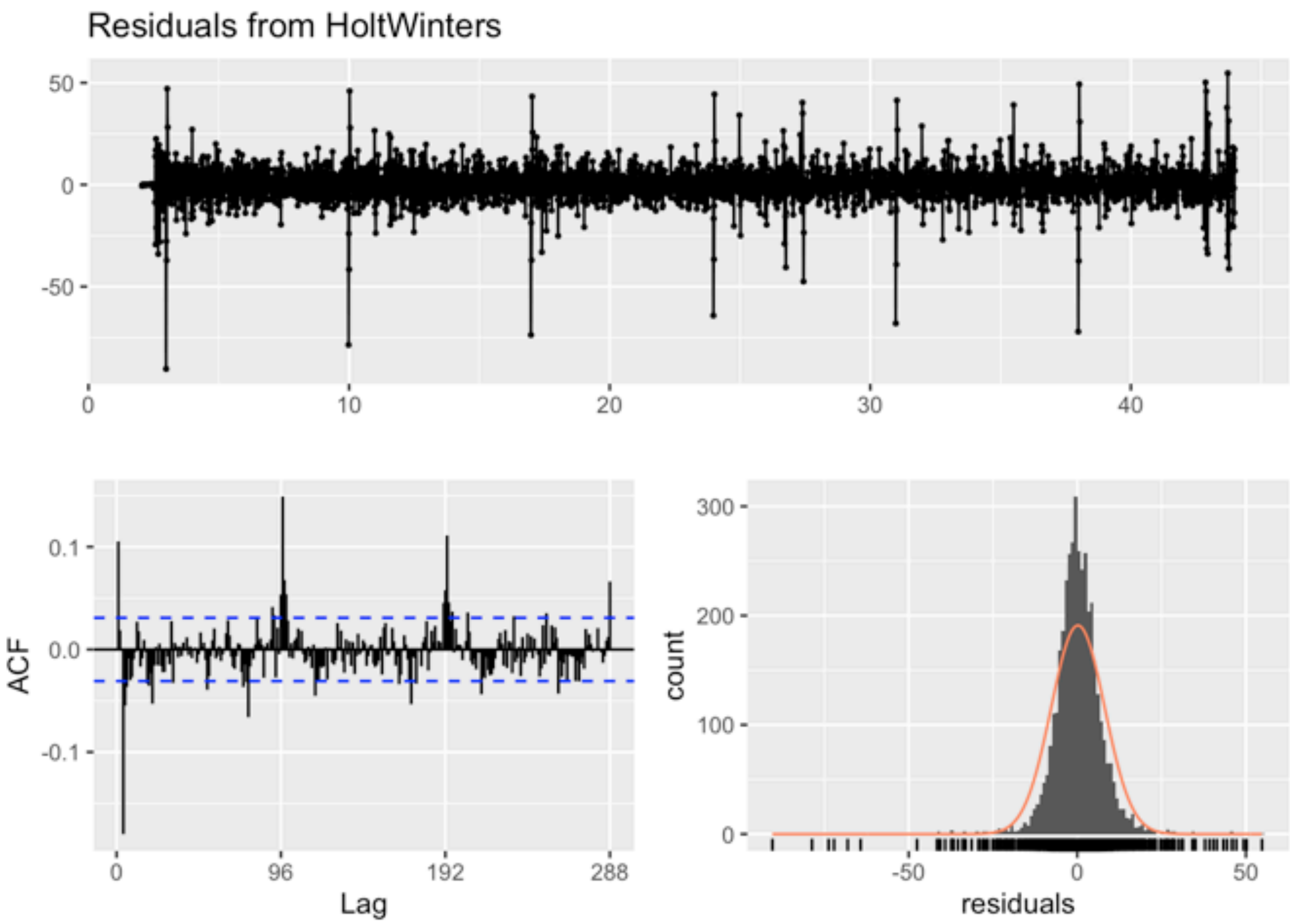
looks not bad! rmse = 26 with quite a big horizon we will try to forecast with multi seasonal holt-winters too:

```
multi_seasonal_hw = HoltWinters(serie_train, alpha=NULL, beta=NULL, gamma=NULL, se
asonal='multi')

prev1=forecast(multi_seasonal_hw, h=288)
autoplot(serie_test) + autolayer(prev1$mean,series="HW multi without covariate")
```




```
checkresiduals(multi_seasonal_hw)
```

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



I'm surprised, it looks like not a bad forecast, although resuduals show there is a lot to improve.

```
#fit_hw$method
#fit_hw$model
```

I used to change frequency and use hw function instead and it has other properties than HoltWinters. A lot of difficulties with horizon and frequency choice

```
#install.packages(Metrics)
#library("Metrics")
rmse(serie_test, prev1$mean)
```

```
## [1] 28.26982
```

it's a bit worse than HW additive

let's see also hw with dumped option:

```
hd=holt(serie_train,h=96,alpha=NULL,beta=NULL,damped=TRUE)
print(sqrt(mean((hd$mean-serie_test)^2)))
```

```
## [1] 79.459
```

that’s a terrible rmse :) we will not use damped version try ses:

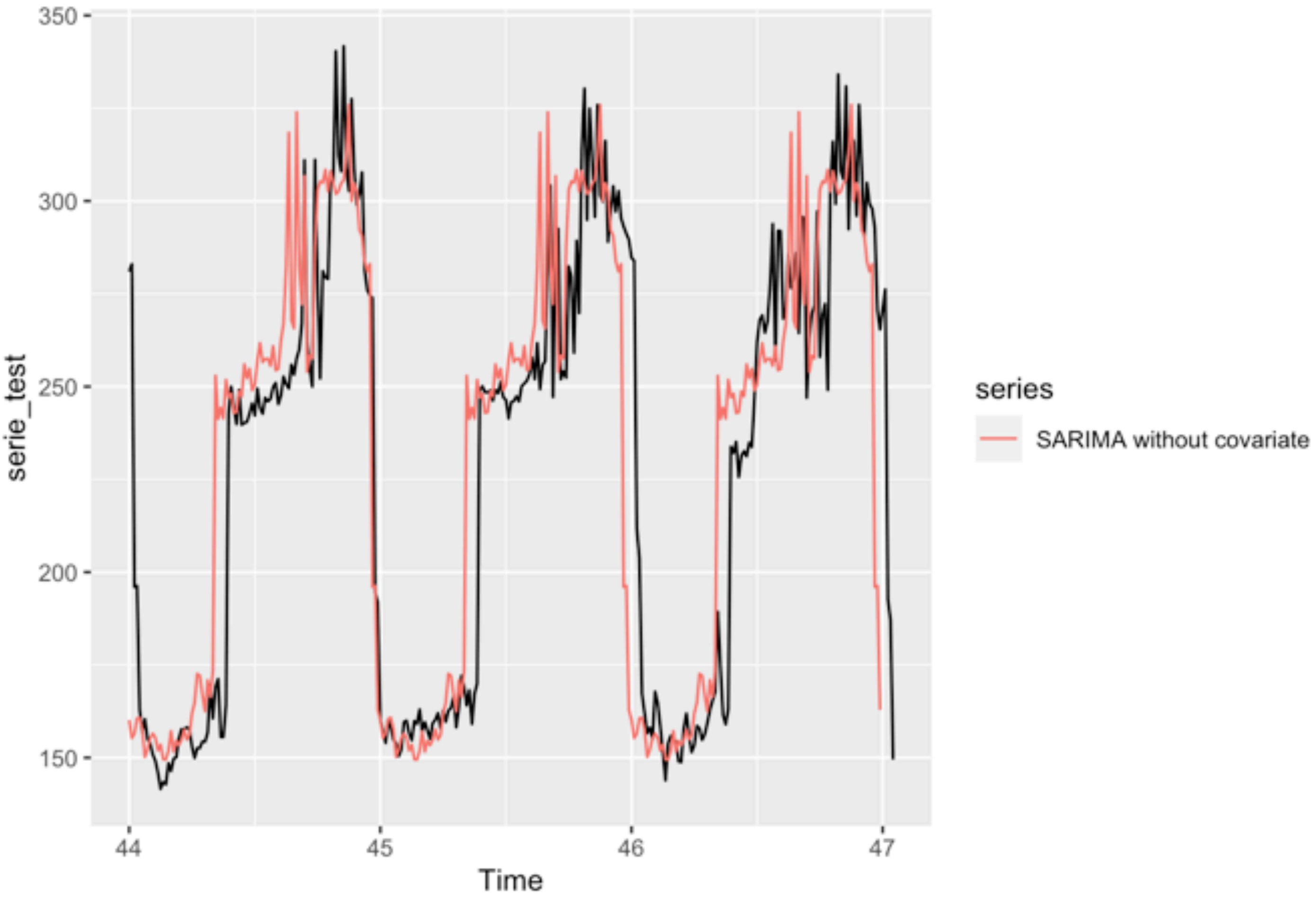
```
SES=ses(serie_train,h=288,alpha=NULL)
print(sqrt(mean((SES$mean-serie_test)^2)))
```

```
## [1] 80.67099
```

much worse than H-W

Let’s see SARIMA model:

```
fit_sarima=auto.arima(s_train[,"Power"])
previ=forecast(fit_sarima,h=288)
autoplot(serie_test)+autolayer(previ$mean,series="SARIMA without covariate")
```

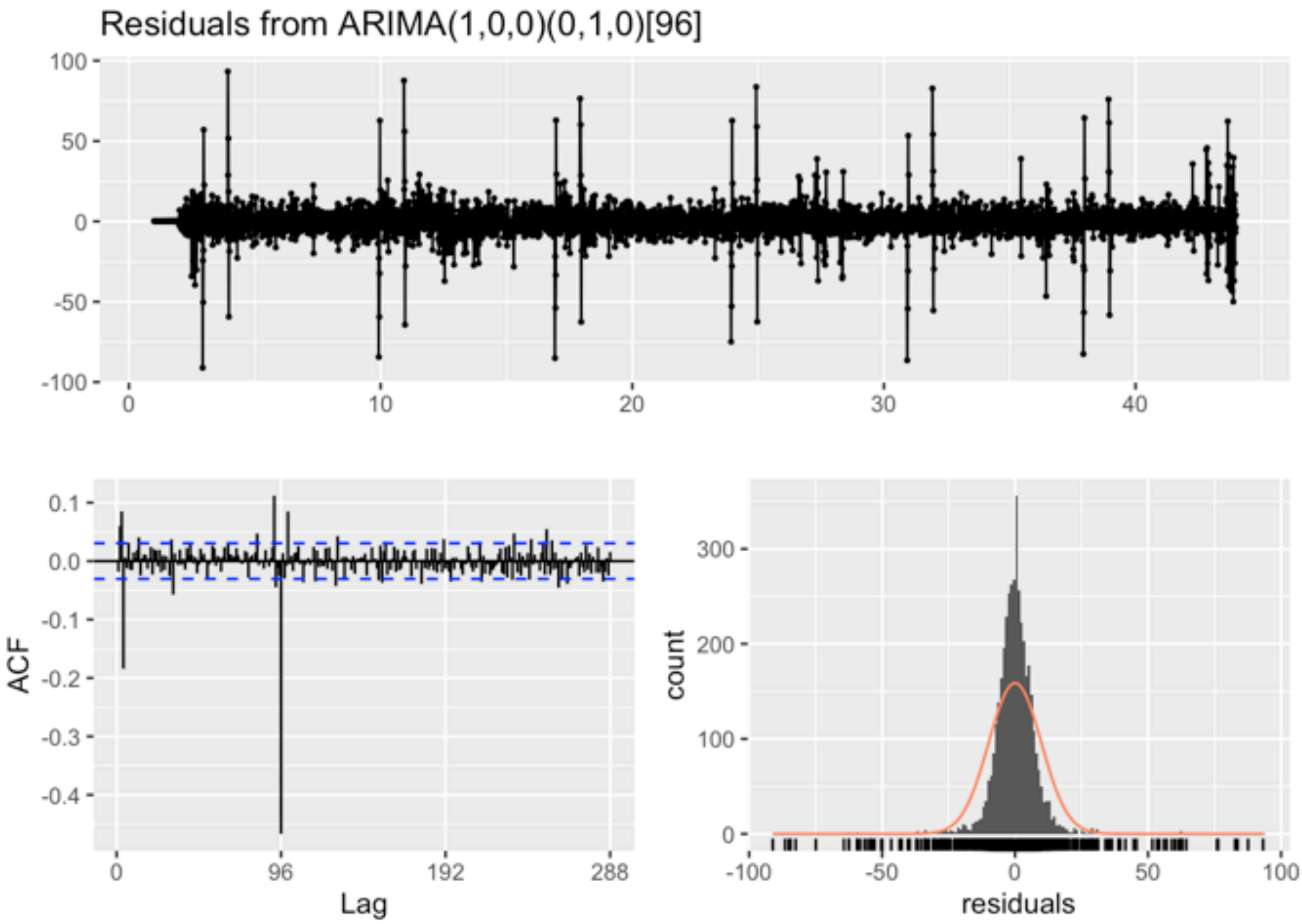



```
print(sqrt(mean((previ$mean-s_test[,"Power"])^2)))
```

```
## [1] 20.44878
```

UPD: I came back here after I found a better manual model with a covariate to try it for power only.

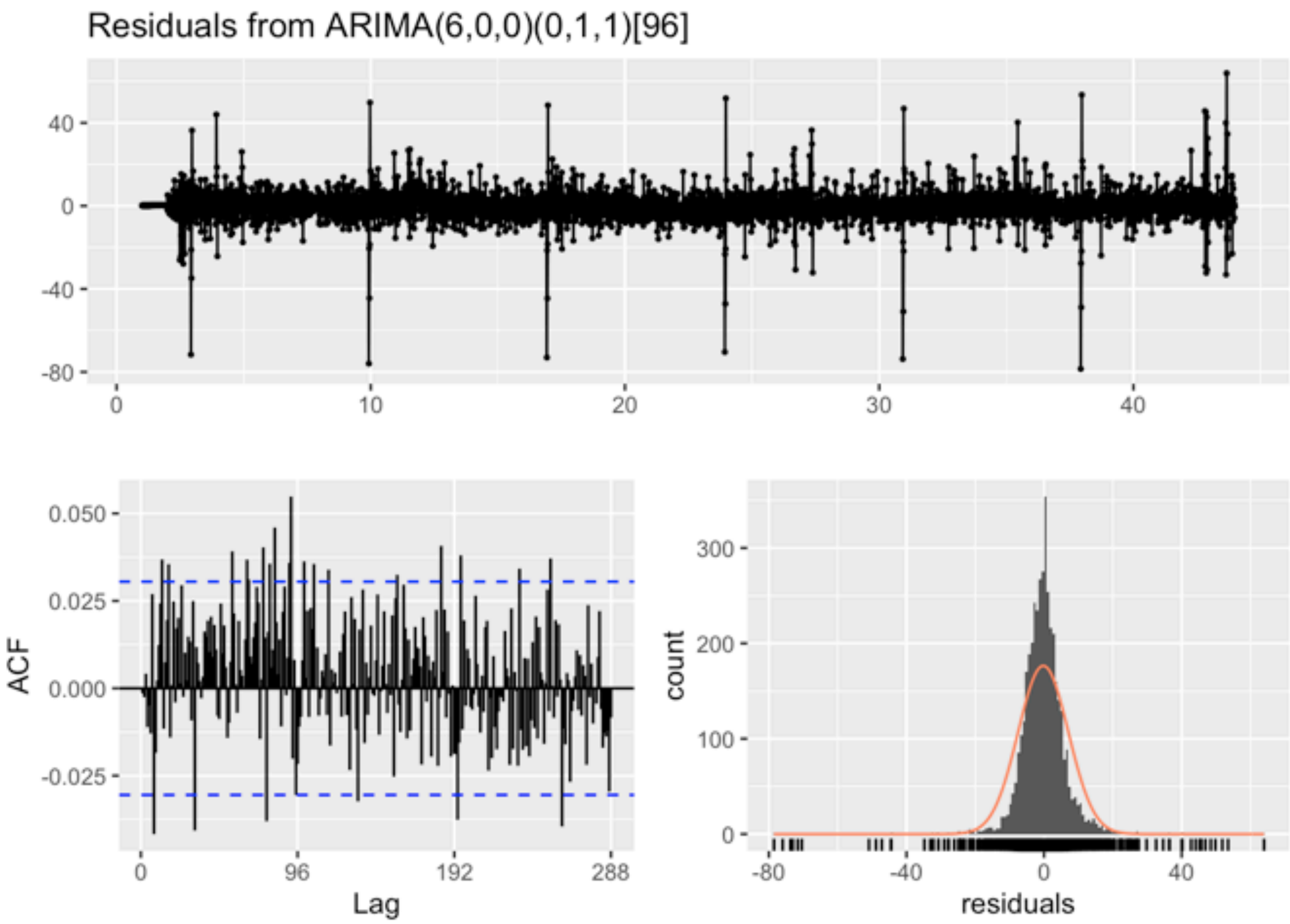
```
checkresiduals(fit_sarima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(0,1,0)[96]
## Q* = 1451.9, df = 191, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 192
```

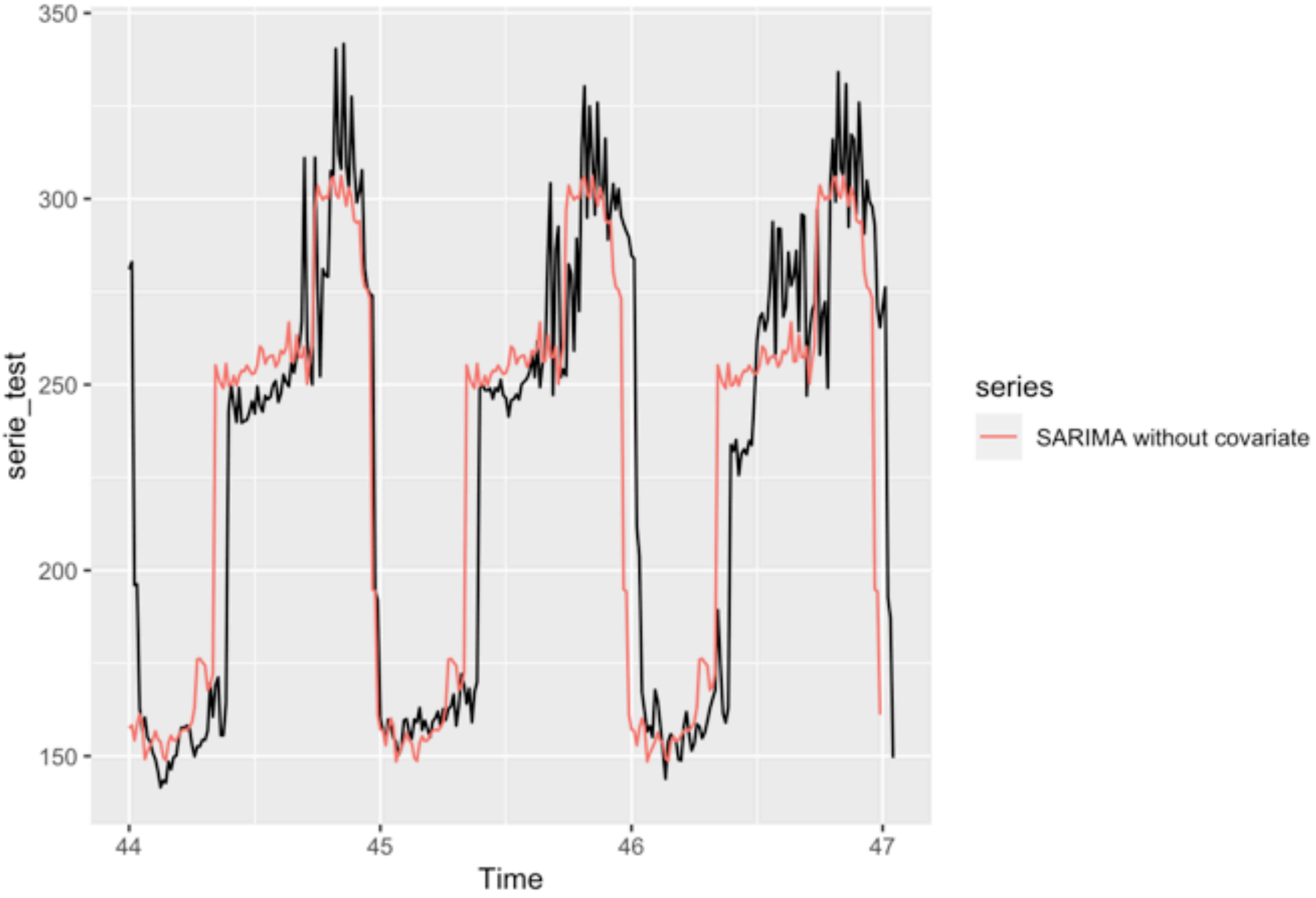
we will need to treat seosonality in addition to treating trend.

```
man_fit_sarima = Arima(s_train[,"Power"], order=c(6,0,0),seasonal = c(0,1,1))
checkresiduals(man_fit_sarima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(6,0,0)(0,1,1)[96]
## Q* = 267.89, df = 185, p-value = 6.457e-05
##
## Model df: 7.   Total lags used: 192
```

```
man_sarima = forecast(man_fit_sarima,h=288)
autoplot(serie_test)+autolayer(man_sarima$mean,series="SARIMA without covariate")
```




```
man_fit_sarima$aic
```

```
## [1] 27714.19
```

```
print(sqrt(mean((man_sarima$mean-s_test[,"Power"])^2)))
```

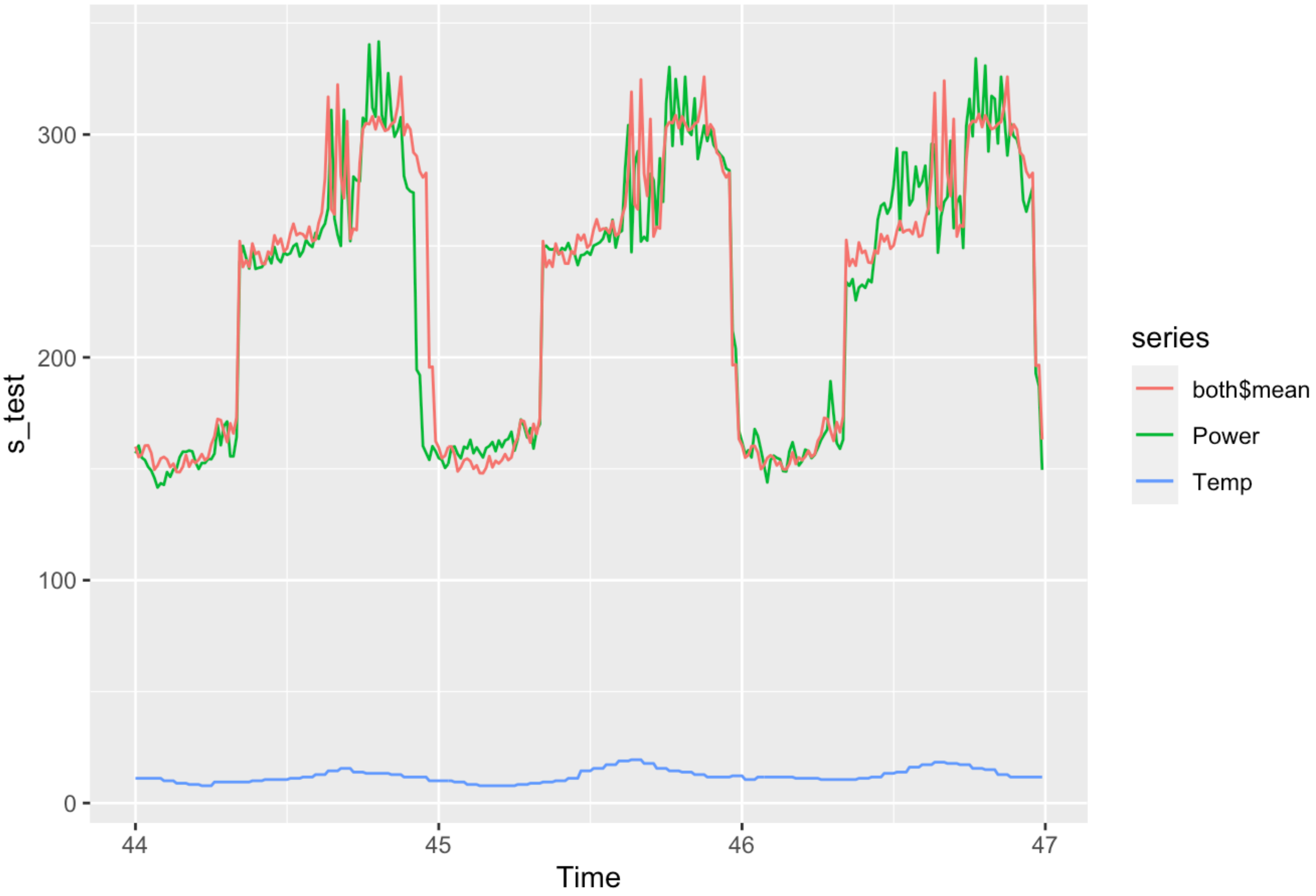
```
## [1] 18.26967
```

this forecast is best.

Let’s introduce the second variable in hope it will do it even better.

We will use a dynamic regression model for forecasting electricity demand, using temperature covariate. The order of the ARIMA model for the residual part is automaticaly selected

```
fit_both=auto.arima(s_train[,"Power"],xreg=s_train[,2])
both=forecast(fit_both,h=288,xreg=s_test[,2])
autoplot(s_test)+autolayer(both$mean)
```




```
print(sqrt(mean((both$mean-s_test[,"Power"])^2)))
```

```
## [1] 20.3865
```

We can see that introducing temperaturre as a covariate slightly improves results of forecast.

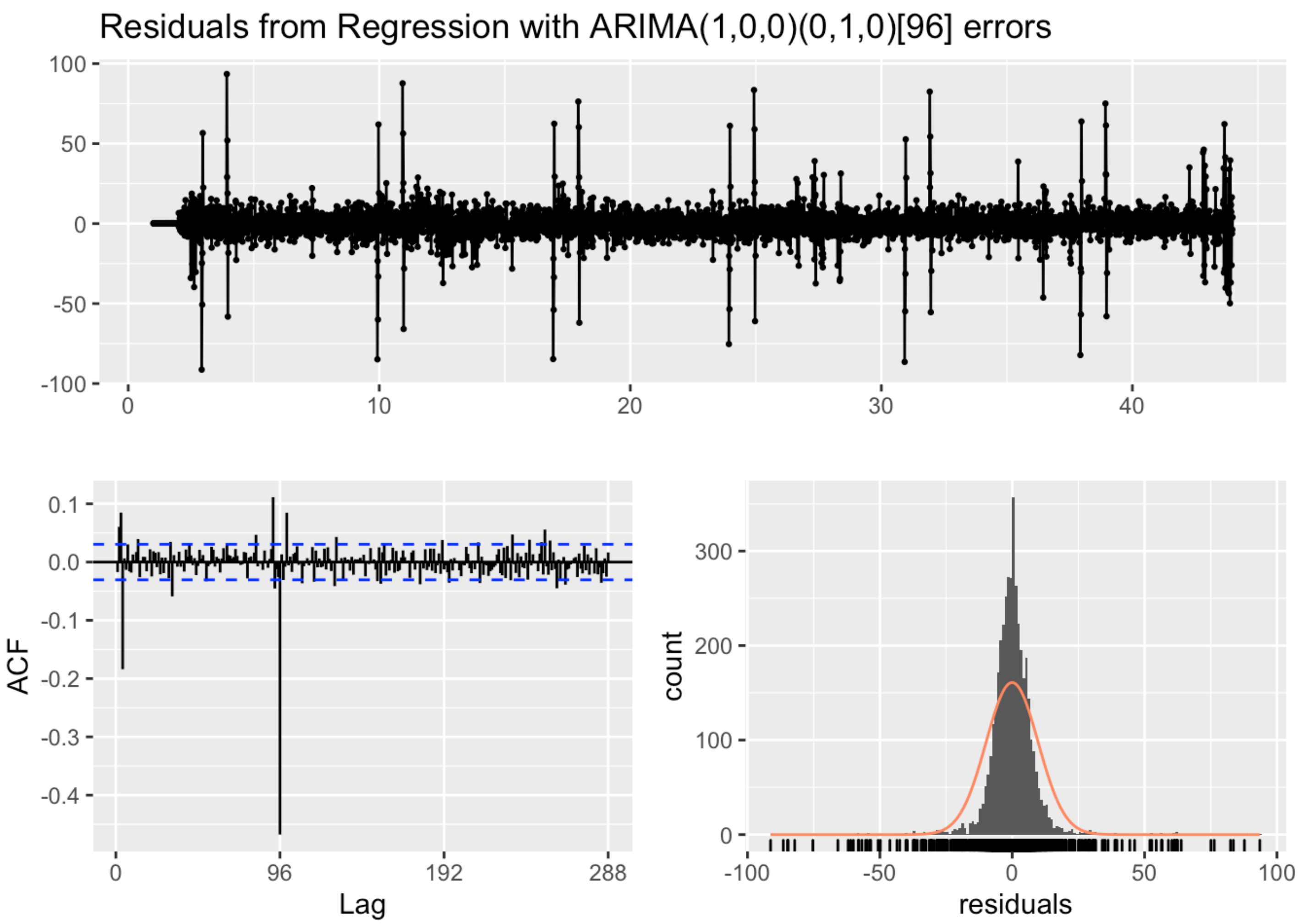
according to RMSE best model is manually chosen SARIMA for now

covariates allows us to improve the forecasting. But if we check the residual, there is still some autocorrelations:

```
summary(fit_both)
```

```
## Series: s_train[, "Power"]
## Regression with ARIMA(1,0,0)(0,1,0)[96] errors
##
## Coefficients:
##          ar1   xreg
##       0.7622  0.4525
## s.e.  0.0102  0.2281
##
## sigma^2 estimated as 99.41:  log likelihood=-14992.75
## AIC=29991.5   AICc=29991.5   BIC=30010.4
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.0406804 9.851609 5.722563 -0.1133479 2.633456 0.7237978
##                    ACF1
## Training set -0.01721292
```

```
checkresiduals(fit_both)
```

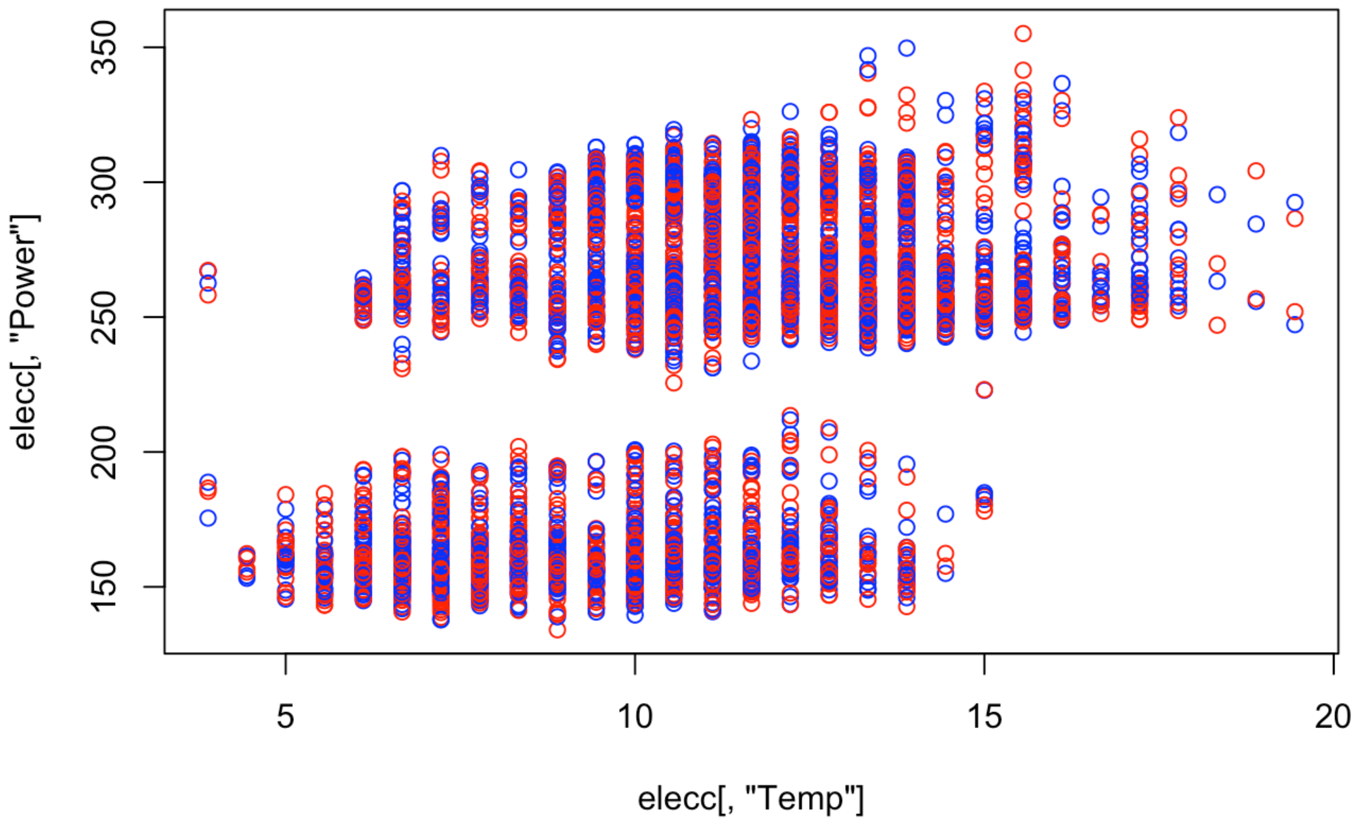


```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,0)(0,1,0)[96] errors
## Q* = 1452.7, df = 190, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 192
```

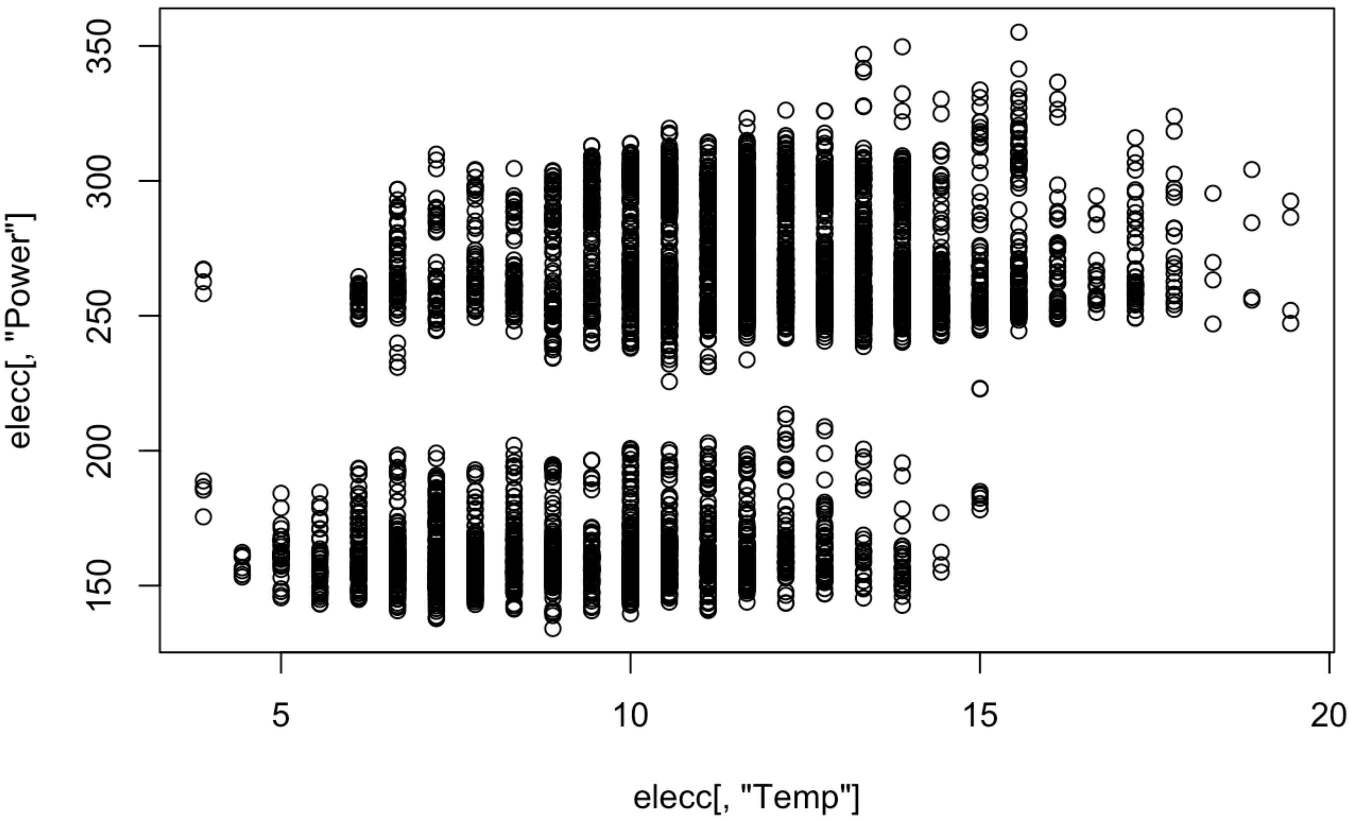
We shall treat data to get read of possible trend if there is any to then apply models.

We can try to find a better model manually. Let's have a look to the relationship between consumption and temperature

```
plot(elecc[,"Temp"],
     elecc[,"Power"], col = c("red", "blue"))
```

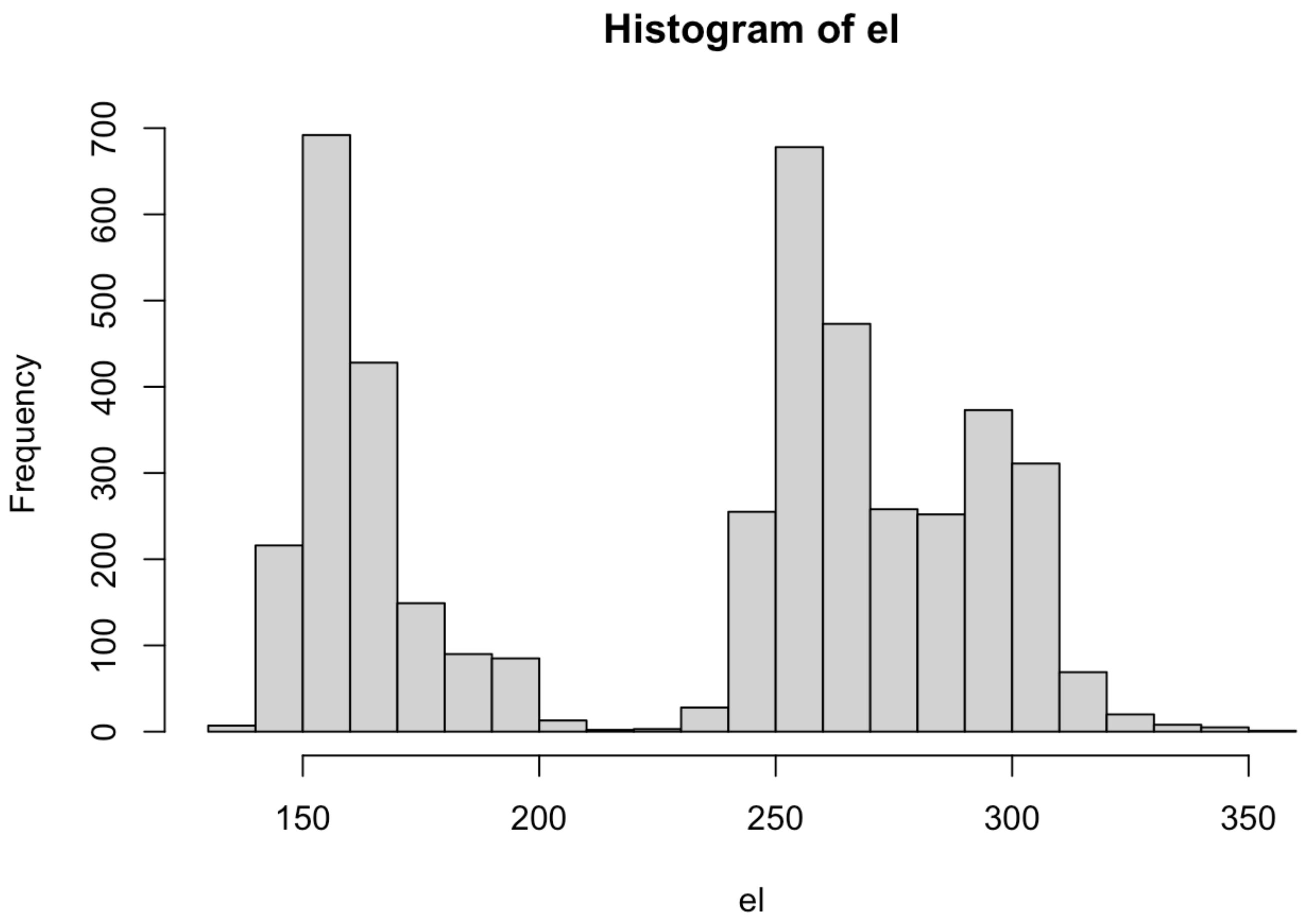



```
 #with clours it's is easier for me to comprehend
plot(elecc[,"Temp"],
     elecc[,"Power"])
```

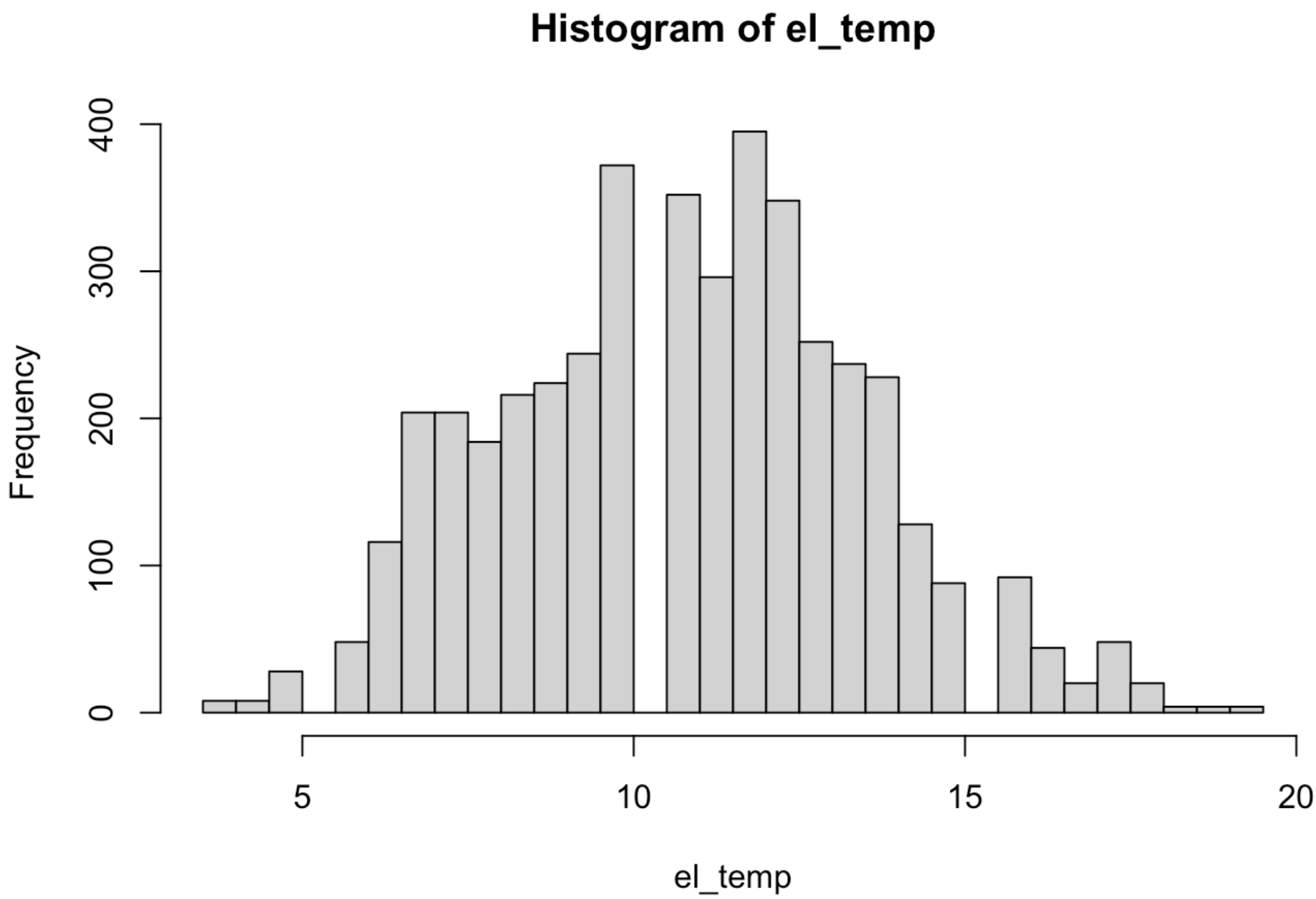


In the class we saw y=x2 and it’s a noticeable bowed shape. In this case it’s not that evident but due to this separation into lower and upper part we can think of sigmoid function that could fit data

```
hist(el, breaks = "scott")
```




```
hist(el_temp, breaks = 'scott')
```



we might think that temperature is distrubuted normally, while electricity has basically 2 different clusters, that have almost equal picks.

while thinking let's see if we can remove any effect of covariate

```
ell=cbind(Power=s_train[,1],Temp=s_train[,2])
fit_manual=tslm(Power~Temp+trend+season,data=s_train)
summary(fit_manual)
```

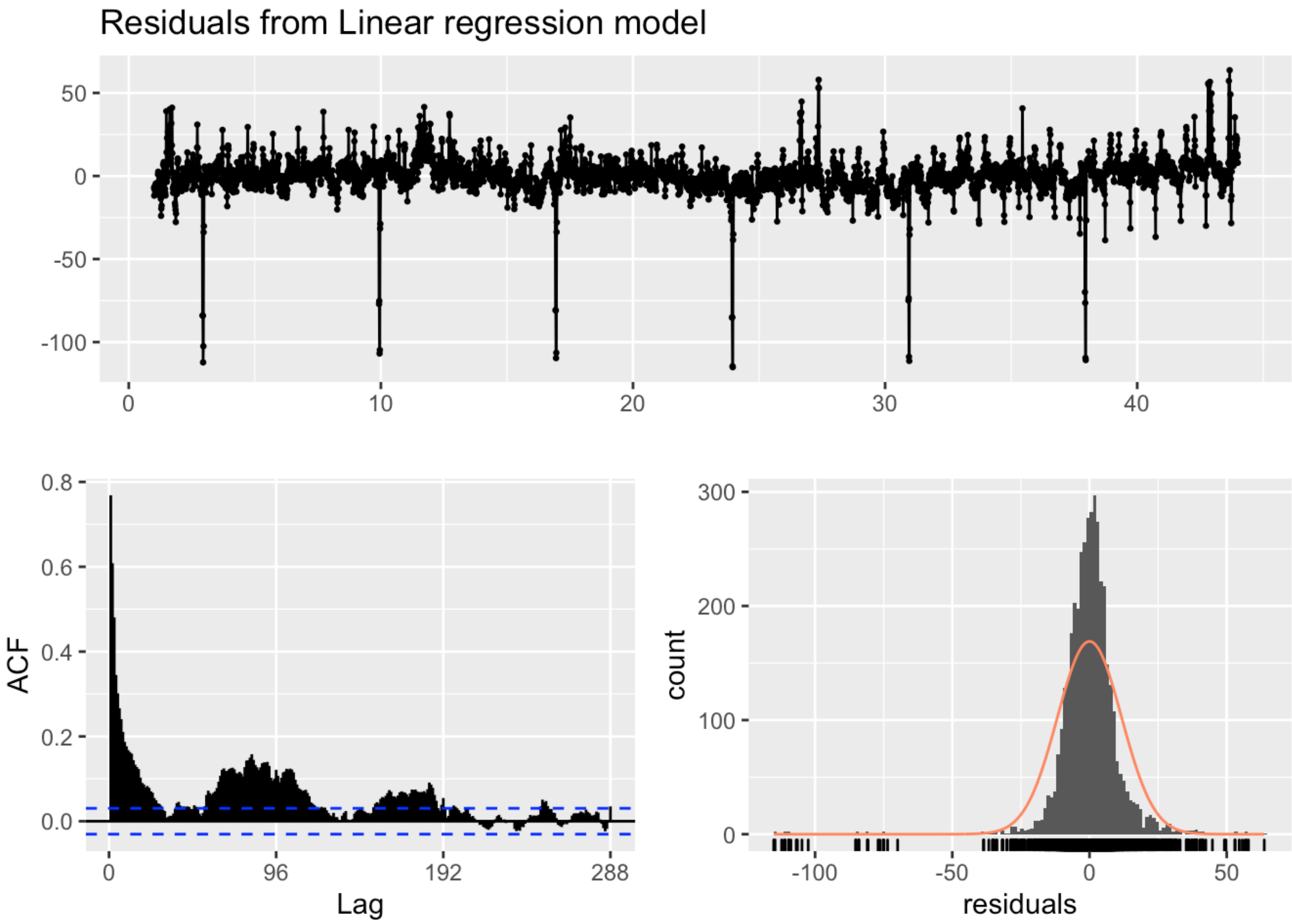
```
##
## Call:
## tslm(formula = Power ~ Temp + trend + season, data = s_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -115.087   -4.825    0.172    4.831   63.688
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.539e+02  2.026e+00  75.967  < 2e-16 ***
## Temp         1.219e+00  9.614e-02  12.684  < 2e-16 ***
## trend       -3.834e-03  1.624e-04 -23.614  < 2e-16 ***
## season2     -5.605e-01  2.553e+00  -0.220  0.82623
```

```
## season3     -6.473e+00  2.553e+00  -2.536  0.01127 *
## season4     -3.110e-01  2.553e+00  -0.122  0.90305
## season5      2.767e+00  2.553e+00   1.084  0.27844
## season6      2.868e+00  2.553e+00   1.123  0.26140
## season7     -6.451e+00  2.553e+00  -2.527  0.01156 *
## season8     -6.373e+00  2.553e+00  -2.496  0.01260 *
## season9     -3.141e+00  2.553e+00  -1.230  0.21868
## season10    -3.590e+00  2.554e+00  -1.406  0.15985
## season11    -9.562e-01  2.554e+00  -0.374  0.70812
## season12    -1.843e+00  2.554e+00  -0.722  0.47053
## season13    -4.997e-01  2.554e+00  -0.196  0.84488
## season14    -4.909e+00  2.555e+00  -1.921  0.05475 .
## season15    -6.384e+00  2.555e+00  -2.499  0.01250 *
## season16    -7.684e-01  2.555e+00  -0.301  0.76359
## season17     8.424e-01  2.555e+00   0.330  0.74163
## season18    -9.376e-02  2.556e+00  -0.037  0.97074
## season19    -9.806e-01  2.556e+00  -0.384  0.70127
## season20     6.186e-01  2.556e+00   0.242  0.80880
## season21     5.456e-01  2.556e+00   0.213  0.83097
## season22     1.432e+00  2.557e+00   0.560  0.57545
## season23     2.441e+00  2.557e+00   0.955  0.33988
## season24     4.254e+00  2.557e+00   1.664  0.09627 .
## season25     3.885e+00  2.557e+00   1.520  0.12869
## season26     7.469e+00  2.557e+00   2.921  0.00351 **
## season27     1.500e+01  2.557e+00   5.865 4.84e-09 ***
## season28     1.679e+01  2.557e+00   6.567 5.78e-11 ***
## season29     1.714e+01  2.557e+00   6.704 2.31e-11 ***
## season30     2.214e+01  2.557e+00   8.656  < 2e-16 ***
## season31     2.150e+01  2.557e+00   8.408  < 2e-16 ***
## season32     1.639e+01  2.557e+00   6.410 1.62e-10 ***
## season33     1.998e+01  2.557e+00   7.811 7.16e-15 ***
## season34     1.039e+02  2.556e+00  40.630  < 2e-16 ***
## season35     1.016e+02  2.556e+00  39.740  < 2e-16 ***
## season36     9.893e+01  2.556e+00  38.699  < 2e-16 ***
## season37     9.849e+01  2.556e+00  38.527  < 2e-16 ***
## season38     1.013e+02  2.553e+00  39.700  < 2e-16 ***
## season39     9.607e+01  2.553e+00  37.634  < 2e-16 ***
## season40     9.828e+01  2.553e+00  38.503  < 2e-16 ***
## season41     9.932e+01  2.553e+00  38.907  < 2e-16 ***
## season42     9.535e+01  2.554e+00  37.338  < 2e-16 ***
## season43     9.632e+01  2.554e+00  37.719  < 2e-16 ***
## season44     9.783e+01  2.554e+00  38.308  < 2e-16 ***
## season45     9.811e+01  2.554e+00  38.419  < 2e-16 ***
## season46     9.990e+01  2.558e+00  39.052  < 2e-16 ***
## season47     9.809e+01  2.558e+00  38.343  < 2e-16 ***
## season48     9.860e+01  2.558e+00  38.543  < 2e-16 ***
## season49     9.922e+01  2.558e+00  38.784  < 2e-16 ***
## season50     9.895e+01  2.566e+00  38.568  < 2e-16 ***
## season51     1.024e+02  2.566e+00  39.897  < 2e-16 ***
## season52     1.013e+02  2.566e+00  39.470  < 2e-16 ***
```

```
## season53     9.953e+01  2.566e+00  38.792  < 2e-16 ***
## season54     1.009e+02  2.571e+00  39.253  < 2e-16 ***
## season55     1.013e+02  2.571e+00  39.397  < 2e-16 ***
## season56     1.008e+02  2.571e+00  39.209  < 2e-16 ***
## season57     9.969e+01  2.571e+00  38.777  < 2e-16 ***
## season58     9.982e+01  2.576e+00  38.742  < 2e-16 ***
## season59     1.004e+02  2.576e+00  38.962  < 2e-16 ***
## season60     1.000e+02  2.576e+00  38.825  < 2e-16 ***
## season61     1.003e+02  2.576e+00  38.926  < 2e-16 ***
## season62     1.012e+02  2.578e+00  39.271  < 2e-16 ***
## season63     9.923e+01  2.578e+00  38.497  < 2e-16 ***
## season64     9.921e+01  2.578e+00  38.489  < 2e-16 ***
## season65     1.003e+02  2.578e+00  38.915  < 2e-16 ***
## season66     9.965e+01  2.572e+00  38.742  < 2e-16 ***
## season67     1.002e+02  2.572e+00  38.944  < 2e-16 ***
## season68     9.832e+01  2.572e+00  38.226  < 2e-16 ***
## season69     9.658e+01  2.572e+00  37.549  < 2e-16 ***
## season70     1.165e+02  2.564e+00  45.442  < 2e-16 ***
## season71     1.295e+02  2.564e+00  50.504  < 2e-16 ***
## season72     1.433e+02  2.564e+00  55.871  < 2e-16 ***
## season73     1.438e+02  2.564e+00  56.098  < 2e-16 ***
## season74     1.416e+02  2.557e+00  55.387  < 2e-16 ***
## season75     1.399e+02  2.557e+00  54.712  < 2e-16 ***
## season76     1.390e+02  2.557e+00  54.370  < 2e-16 ***
## season77     1.401e+02  2.557e+00  54.772  < 2e-16 ***
## season78     1.452e+02  2.556e+00  56.811  < 2e-16 ***
## season79     1.419e+02  2.556e+00  55.529  < 2e-16 ***
## season80     1.407e+02  2.556e+00  55.048  < 2e-16 ***
## season81     1.387e+02  2.556e+00  54.289  < 2e-16 ***
## season82     1.396e+02  2.554e+00  54.673  < 2e-16 ***
## season83     1.377e+02  2.554e+00  53.904  < 2e-16 ***
## season84     1.365e+02  2.554e+00  53.446  < 2e-16 ***
## season85     1.368e+02  2.554e+00  53.556  < 2e-16 ***
## season86     1.355e+02  2.553e+00  53.052  < 2e-16 ***
## season87     1.327e+02  2.553e+00  51.957  < 2e-16 ***
## season88     1.321e+02  2.553e+00  51.745  < 2e-16 ***
## season89     1.307e+02  2.553e+00  51.175  < 2e-16 ***
## season90     1.135e+02  2.553e+00  44.473  < 2e-16 ***
## season91     1.118e+02  2.553e+00  43.799  < 2e-16 ***
## season92     1.071e+02  2.553e+00  41.940  < 2e-16 ***
## season93     1.076e+02  2.553e+00  42.136  < 2e-16 ***
## season94     3.122e+01  2.553e+00  12.230  < 2e-16 ***
## season95     3.305e+01  2.553e+00  12.947  < 2e-16 ***
## season96     3.290e+00  2.553e+00   1.289  0.19751
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.84 on 4030 degrees of freedom
## Multiple R-squared:  0.9582, Adjusted R-squared:  0.9572
## F-statistic: 952.1 on 97 and 4030 DF,  p-value: < 2.2e-16
```

there's a trend and temperature look very significant. So many seasons that we need to treat before modelling data.

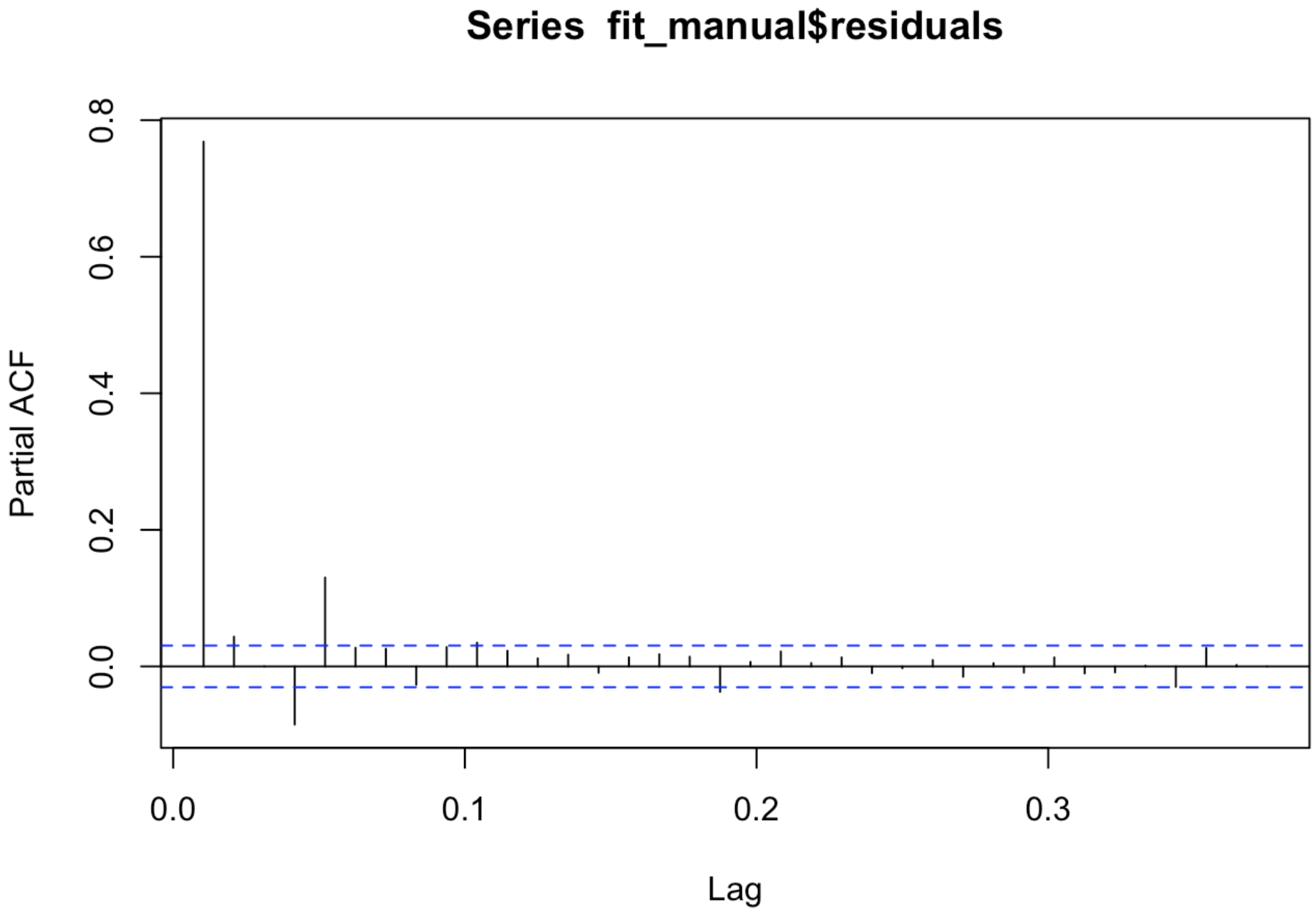
```
checkresiduals(fit_manual)
```




```
##
##   Breusch-Godfrey test for serial correlation of order up to 192
##
## data:  Residuals from Linear regression model
## LM test = 2611.8, df = 192, p-value < 2.2e-16
```

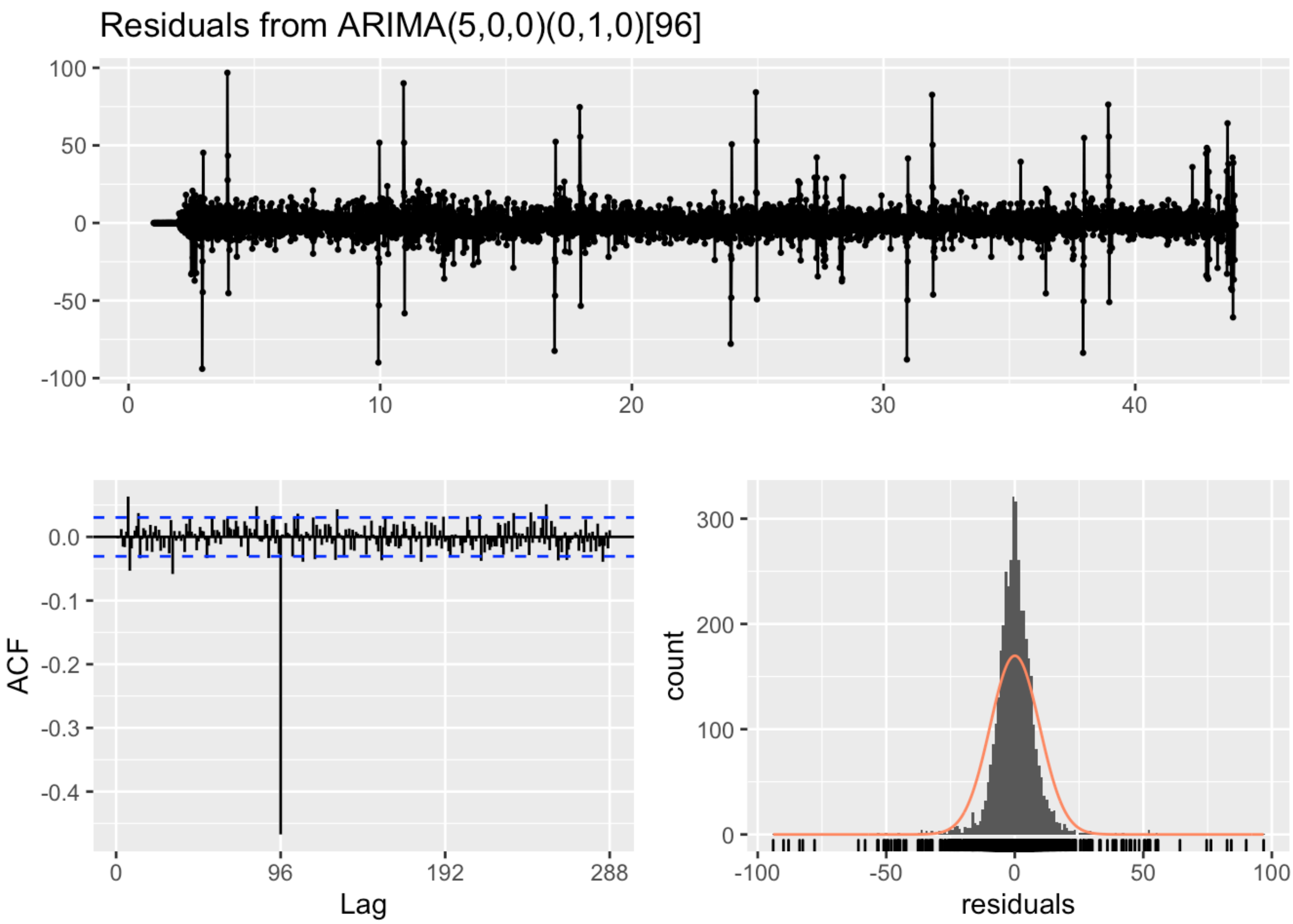
Variance is too big, we shall address seasonality. We'll use Box Cox and log transformations

```
plot(pacf(fit_manual$residuals))
```



PACF and SCF look like those of an AR5 model: exponential deacrease of the ACF and significant PCA at lag 5.We can see it's very periodic (for ACF): picks at 96, 192, 288 - it corresponds to our chosen frequency.This ACF suggest a seasonnal MA1 We can test it:

```
tmp=fit_manual$residuals
fit3=Arima(tmp,order=c(5,0,0),seasonal = c(0,1,0))
checkresiduals(fit3)
```

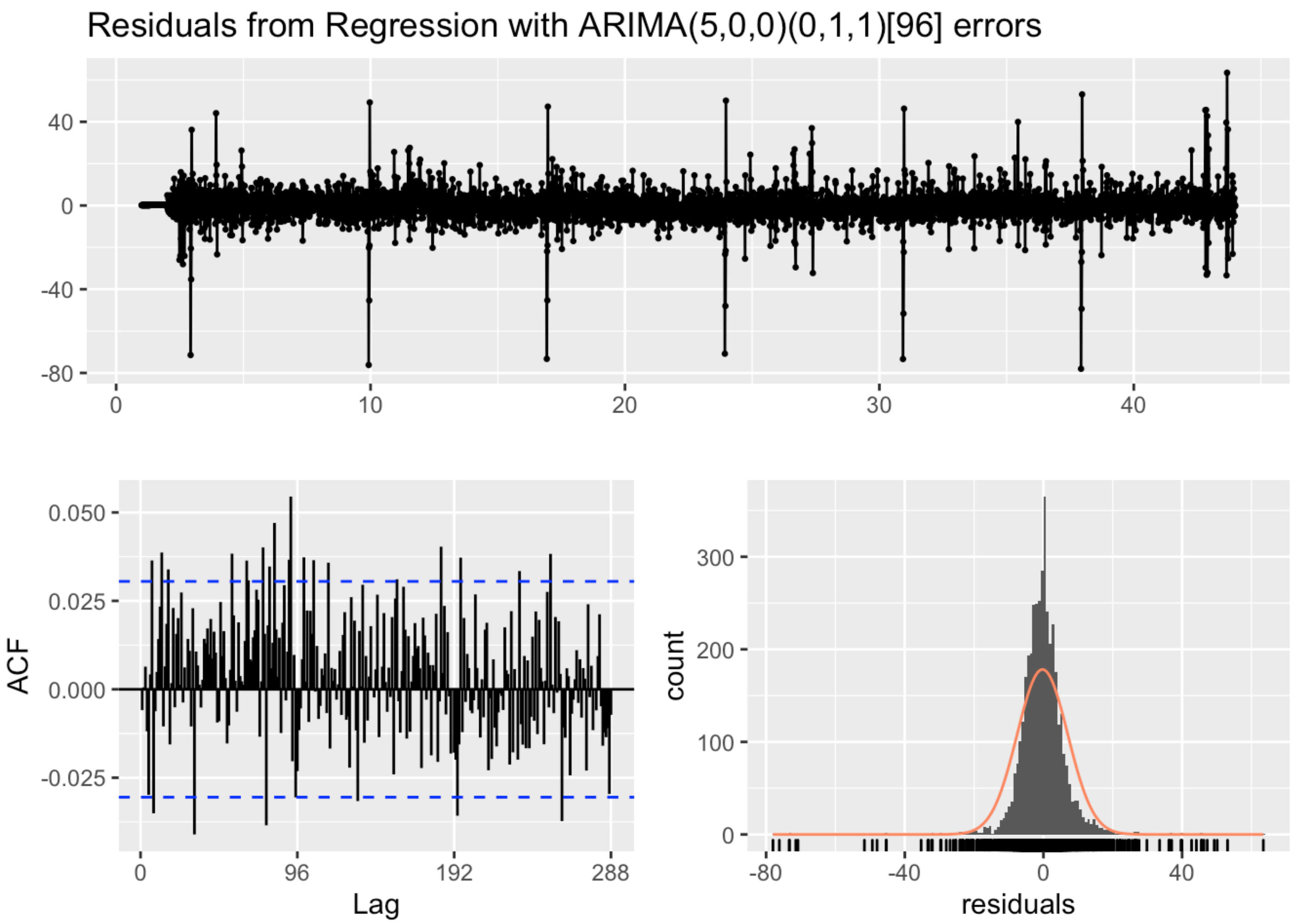



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,0)(0,1,0)[96]
## Q* = 1211, df = 187, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 192
```

It definitely looks better, but still there are significant ACF that we can address.

Residual have significant ACF at periodic lag (96). We will add a second order MA in the seasonal pattern:

```
man_fit = Arima(s_train[,"Power"],xreg=s_train[,2], order=c(5,0,0),seasonal = c(0,
1,1))
checkresiduals(man_fit)
```



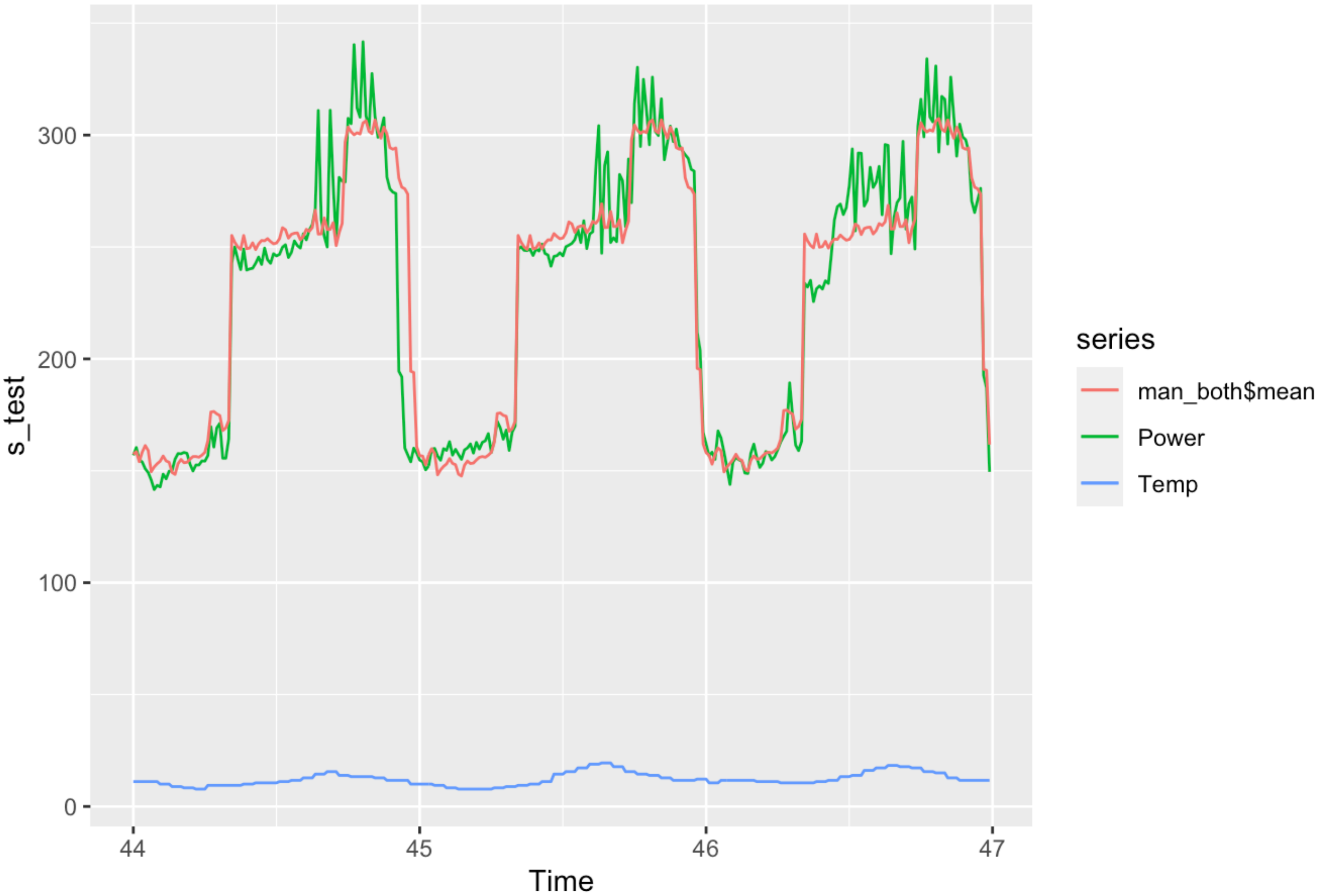

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(5,0,0)(0,1,1)[96] errors
## Q* = 266.13, df = 185, p-value = 8.625e-05
##
## Model df: 7.   Total lags used: 192
```

```
man_fit$aic
```

```
## [1] 27712.66
```

this AIC is better than the one obtained with auto.arima. We can suggest it will perform better in forecast.

```
man_both=forecast(man_fit,h=288,xreg=s_test[,2])
autoplot(s_test)+autolayer(man_both$mean)
```



looks good to me. Let's see RMSE of the obrained model:

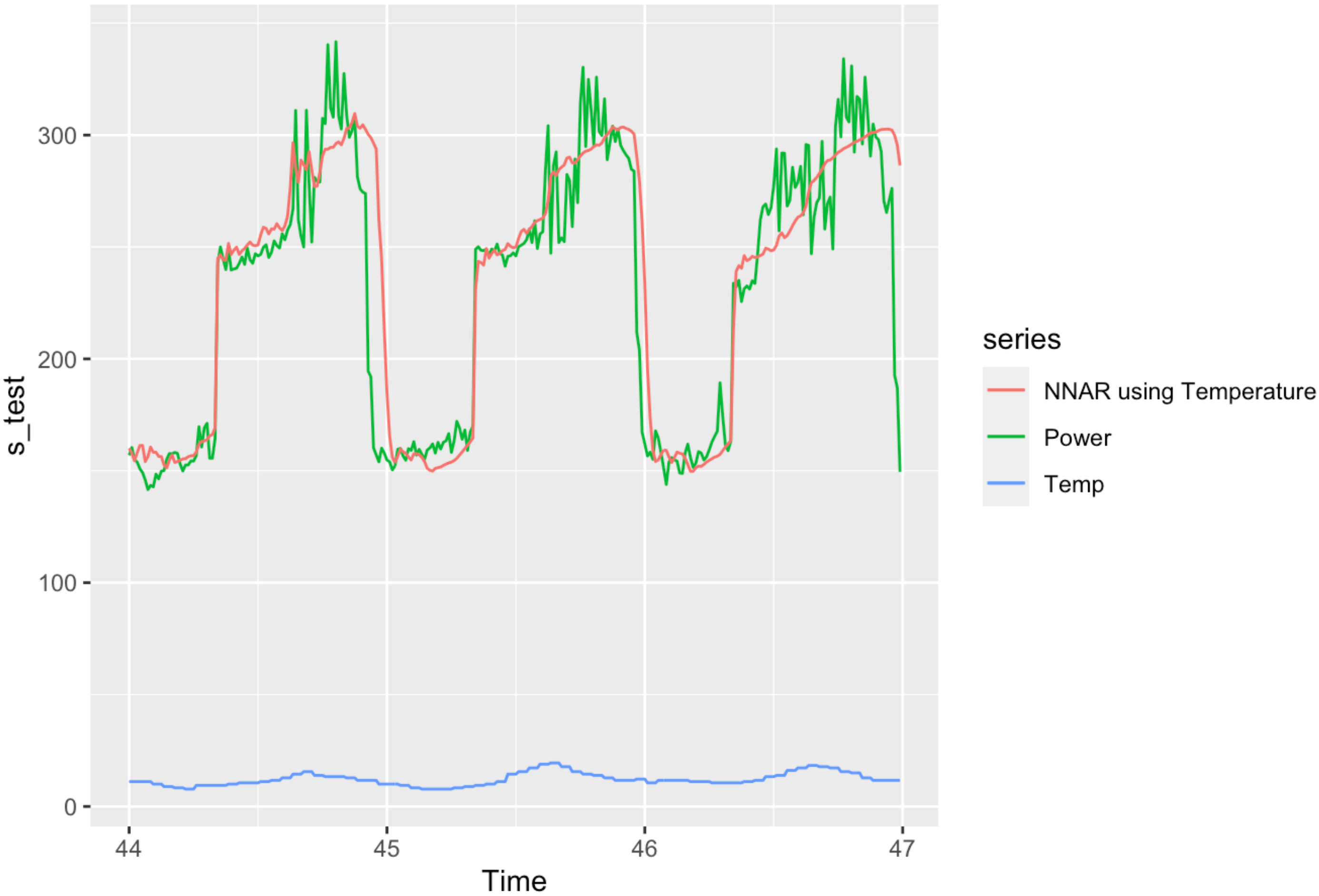
```
print(sqrt(mean((man_both$mean-s_test[,"Power"])^2)))
```

```
## [1] 18.11793
```

Great! It is better than auto-arima.

We will try NNAR and it will be the last model:

```
fit_NN=nnetar(s_train[,"Power"],xreg=s_train[,2])
prevNN=forecast(fit_NN,h=96,xreg=s_test[,2])
autoplot(s_test)+autolayer(prevNN$mean,series="NNAR using Temperature")
```



RMSE:

```
print(sqrt(mean((prevNN$mean-s_test[,"Power"])^2)))
```

```
## [1] 27.39518
```

this forecast is worse than the one we obtained manually, We will produce a forecast using model called man_fit for Y with covariates and a model man_fit_sarima for univariate case.

We obtain forecast for the case with Temp as covariate

```
new_day <- elec_train$Temp[4508:4603]
my_forecast=forecast(man_fit,h=96,xreg=new_day)
my_forecast
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 44.00000       157.7340 148.2500 167.2180 143.2294 172.2385
## 44.01042       158.4851 146.6937 170.2765 140.4517 176.5185
## 44.02083       154.0566 140.8283 167.2849 133.8257 174.2875
## 44.03125       158.3692 144.0628 172.6755 136.4895 180.2488
## 44.04167       161.2805 146.7024 175.8586 138.9852 183.5758
## 44.05208       158.7010 143.8911 173.5109 136.0513 181.3508
```

```
## 44.06250       149.2463 134.2898 164.2027 126.3724 172.1202
## 44.07292       151.3852 136.3487 166.4218 128.3889 174.3816
## 44.08333       152.8215 137.7009 167.9421 129.6965 175.9465
## 44.09375       154.2589 139.0885 169.4293 131.0577 177.4601
## 44.10417       156.5917 141.3867 171.7968 133.3377 179.8458
## 44.11458       154.4061 139.1751 169.6371 131.1123 177.6999
## 44.12500       153.8069 138.5612 169.0527 130.4905 177.1233
## 44.13542       149.8821 134.6256 165.1387 126.5493 173.2150
## 44.14583       149.0312 133.7674 164.2950 125.6872 172.3751
## 44.15625       153.8286 138.5602 169.0971 130.4775 177.1797
## 44.16667       155.6973 140.4254 170.9692 132.3410 179.0536
## 44.17708       154.5833 139.3092 169.8574 131.2235 177.9430
## 44.18750       154.7730 139.4974 170.0487 131.4109 178.1351
## 44.19792       156.1286 140.8519 171.4053 132.7649 179.4923
## 44.20833       157.3212 142.0438 172.5986 133.9564 180.6860
## 44.21875       157.9903 142.7125 173.2682 134.6248 181.3558
## 44.22917       157.6716 142.3934 172.9498 134.3056 181.0376
## 44.23958       158.5941 143.3157 173.8725 135.2278 181.9604
## 44.25000       159.8843 144.6058 175.1629 136.5178 183.2509
## 44.26042       164.4135 149.1348 179.6922 141.0468 187.7802
## 44.27083       177.2238 161.9450 192.5025 153.8570 200.5906
## 44.28125       177.4589 162.1801 192.7377 154.0920 200.8258
## 44.29167       176.4010 161.1222 191.6799 153.0341 199.7680
## 44.30208       175.3251 160.0462 190.6039 151.9581 198.6920
## 44.31250       168.6435 153.3646 183.9223 145.2765 192.0105
## 44.32292       169.7900 154.5111 185.0689 146.4230 193.1570
## 44.33333       173.1589 157.8800 188.4378 149.7919 196.5259
## 44.34375       255.4657 240.1868 270.7446 232.0986 278.8327
## 44.35417       252.4563 237.1774 267.7352 229.0893 275.8234
## 44.36458       250.3859 235.1070 265.6648 227.0189 273.7530
## 44.37500       249.2635 233.9846 264.5424 225.8965 272.6306
## 44.38542       255.4399 240.1610 270.7188 232.0729 278.8070
## 44.39583       249.5638 234.2849 264.8426 226.1967 272.9308
## 44.40625       249.8632 234.5843 265.1421 226.4961 273.2302
## 44.41667       252.1901 236.9112 267.4690 228.8230 275.5571
## 44.42708       248.8533 233.5744 264.1321 225.4862 272.2203
## 44.43750       251.3780 236.0991 266.6569 228.0109 274.7450
## 44.44792       252.8936 237.6147 268.1725 229.5265 276.2606
## 44.45833       252.7578 237.4789 268.0367 229.3908 276.1249
## 44.46875       254.6923 239.4134 269.9712 231.3252 278.0593
## 44.47917       253.4007 238.1218 268.6796 230.0336 276.7677
## 44.48958       252.3836 237.1047 267.6625 229.0165 275.7506
## 44.50000       252.7304 237.4515 268.0093 229.3634 276.0975
## 44.51042       255.6339 240.3550 270.9128 232.2668 279.0009
## 44.52083       260.5471 245.2682 275.8260 237.1800 283.9142
## 44.53125       259.6563 244.3774 274.9352 236.2892 283.0233
## 44.54167       255.8878 240.6089 271.1667 232.5207 279.2548
## 44.55208       257.9206 242.6417 273.1995 234.5535 281.2876
## 44.56250       258.4071 243.1282 273.6860 235.0400 281.7741
## 44.57292       258.5423 243.2635 273.8212 235.1753 281.9094
```

```
## 44.58333       255.4910 240.2121 270.7699 232.1240 278.8581
## 44.59375       257.3629 242.0840 272.6418 233.9959 280.7300
## 44.60417       260.4653 245.1864 275.7442 237.0982 283.8323
## 44.61458       259.6117 244.3328 274.8906 236.2447 282.9788
## 44.62500       261.2374 245.9585 276.5163 237.8704 284.6045
## 44.63542       268.6193 253.3404 283.8982 245.2523 291.9864
## 44.64583       258.0183 242.7394 273.2972 234.6512 281.3853
## 44.65625       258.1022 242.8233 273.3811 234.7351 281.4692
## 44.66667       265.2375 249.9586 280.5164 241.8705 288.6046
## 44.67708       259.4713 244.1924 274.7502 236.1042 282.8383
## 44.68750       259.4932 244.2143 274.7721 236.1261 282.8603
## 44.69792       262.4474 247.1685 277.7263 239.0804 285.8145
## 44.70833       252.2467 236.9678 267.5256 228.8797 275.6138
## 44.71875       258.6383 243.3594 273.9172 235.2713 282.0054
## 44.72917       262.7708 247.4919 278.0497 239.4038 286.1379
## 44.73958       299.1243 283.8454 314.4032 275.7572 322.4913
## 44.75000       305.8350 290.5561 321.1139 282.4680 329.2021
## 44.76042       302.8006 287.5217 318.0795 279.4336 326.1677
## 44.77083       301.4080 286.1291 316.6869 278.0410 324.7751
## 44.78125       302.3352 287.0563 317.6141 278.9681 325.7022
## 44.79167       301.7641 286.4852 317.0430 278.3971 325.1312
## 44.80208       306.7032 291.4243 321.9821 283.3362 330.0703
## 44.81250       307.6927 292.4138 322.9716 284.3256 331.0597
## 44.82292       302.9891 287.7102 318.2680 279.6220 326.3561
## 44.83333       301.9320 286.6531 317.2109 278.5649 325.2990
## 44.84375       308.0469 292.7680 323.3258 284.6799 331.4140
## 44.85417       303.3855 288.1066 318.6644 280.0184 326.7525
## 44.86458       299.8862 284.6073 315.1651 276.5191 323.2532
## 44.87500       304.8040 289.5251 320.0829 281.4370 328.1711
## 44.88542       302.5513 287.2724 317.8302 279.1842 325.9184
## 44.89583       296.3300 281.0512 311.6089 272.9630 319.6971
## 44.90625       295.6309 280.3520 310.9098 272.2639 318.9980
## 44.91667       296.1677 280.8888 311.4466 272.8006 319.5347
## 44.92708       282.1381 266.8592 297.4169 258.7710 305.5051
## 44.93750       278.0230 262.7441 293.3019 254.6559 301.3900
## 44.94792       277.2881 262.0092 292.5670 253.9211 300.6552
## 44.95833       274.8738 259.5949 290.1527 251.5067 298.2408
## 44.96875       196.0950 180.8161 211.3739 172.7279 219.4620
## 44.97917       195.5223 180.2434 210.8012 172.1553 218.8894
## 44.98958       162.3571 147.0782 177.6360 138.9900 185.7241
```

and for univariate model:

```
my_forecast_uni = forecast(man_fit_sarima, h=96)
my_forecast_uni
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 44.00000       157.4670 147.9823 166.9518 142.9613 171.9727
## 44.01042       158.3195 146.5379 170.1011 140.3011 176.3379
```

```
## 44.02083       154.2243 140.9795 167.4691 133.9682 174.4805
## 44.03125       158.3993 144.0479 172.7506 136.4508 180.3478
## 44.04167       161.2748 146.6403 175.9094 138.8932 183.6565
## 44.05208       158.8154 143.9778 173.6529 136.1233 181.5075
## 44.06250       149.1658 134.1742 164.1574 126.2382 172.0935
## 44.07292       151.3115 136.2260 166.3970 128.2403 174.3828
## 44.08333       152.7258 137.5357 167.9160 129.4945 175.9571
## 44.09375       154.3277 139.0572 169.5982 130.9735 177.6819
## 44.10417       156.6818 141.3550 172.0086 133.2415 180.1221
## 44.11458       154.4646 139.0938 169.8353 130.9570 177.9721
## 44.12500       153.8485 138.4496 169.2474 130.2980 177.3991
## 44.13542       149.7376 134.3192 165.1561 126.1572 173.3181
## 44.14583       148.8613 133.4284 164.2943 125.2587 172.4640
## 44.15625       153.6472 138.2040 169.0903 130.0289 177.2654
## 44.16667       155.4936 140.0428 170.9445 131.8636 179.1237
## 44.17708       154.1624 138.7058 169.6191 130.5235 177.8013
## 44.18750       154.3398 138.8789 169.8006 130.6944 177.9851
## 44.19792       155.6878 140.2238 171.1517 132.0377 179.3378
## 44.20833       156.8664 141.4003 172.3326 133.2130 180.5199
## 44.21875       157.2559 141.7882 172.7236 133.6001 180.9118
## 44.22917       156.9341 141.4652 172.4030 133.2765 180.5918
## 44.23958       157.8624 142.3927 173.3322 134.2035 181.5214
## 44.25000       159.1375 143.6672 174.6079 135.4776 182.7974
## 44.26042       163.2078 147.7370 178.6787 139.5473 186.8684
## 44.27083       175.9860 160.5149 191.4572 152.3250 199.6471
## 44.28125       176.2341 160.7627 191.7055 152.5727 199.8955
## 44.29167       175.1868 159.7152 190.6583 151.5251 198.8485
## 44.30208       174.4419 158.9702 189.9136 150.7800 198.1038
## 44.31250       167.7597 152.2879 183.2315 144.0977 191.4218
## 44.32292       168.8877 153.4159 184.3596 145.2256 192.5499
## 44.33333       172.2559 156.7840 187.7278 148.5937 195.9181
## 44.34375       255.1766 239.7047 270.6486 231.5143 278.8389
## 44.35417       252.1841 236.7121 267.6561 228.5218 275.8464
## 44.36458       250.1033 234.6313 265.5753 226.4409 273.7656
## 44.37500       248.9869 233.5149 264.4589 225.3245 272.6492
## 44.38542       255.5730 240.1010 271.0450 231.9106 279.2353
## 44.39583       249.7056 234.2336 265.1776 226.0432 273.3680
## 44.40625       250.0180 234.5460 265.4901 226.3556 273.6805
## 44.41667       252.3419 236.8698 267.8139 228.6794 276.0043
## 44.42708       249.7141 234.2421 265.1861 226.0517 273.3765
## 44.43750       252.2259 236.7539 267.6980 228.5635 275.8884
## 44.44792       253.7509 238.2788 269.2229 230.0884 277.4133
## 44.45833       253.6058 238.1338 269.0779 229.9434 277.2683
## 44.46875       255.1238 239.6518 270.5958 231.4614 278.7862
## 44.47917       253.8268 238.3548 269.2989 230.1644 277.4893
## 44.48958       252.8125 237.3405 268.2845 229.1501 276.4749
## 44.50000       253.1654 237.6934 268.6375 229.5030 276.8279
## 44.51042       255.3462 239.8742 270.8183 231.6838 279.0087
## 44.52083       260.2550 244.7830 275.7271 236.5926 283.9174
## 44.53125       259.3765 243.9045 274.8485 235.7141 283.0389
```

```
## 44.54167       255.6065 240.1345 271.0785 231.9441 279.2689
## 44.55208       257.2073 241.7353 272.6793 233.5449 280.8697
## 44.56250       257.6999 242.2278 273.1719 234.0374 281.3623
## 44.57292       257.8178 242.3458 273.2898 234.1554 281.4802
## 44.58333       254.7873 239.3152 270.2593 231.1248 278.4497
## 44.59375       256.0544 240.5824 271.5264 232.3920 279.7168
## 44.60417       259.1377 243.6657 274.6098 235.4753 282.8002
## 44.61458       258.2882 242.8161 273.7602 234.6257 281.9506
## 44.62500       259.8872 244.4151 275.3592 236.2247 283.5496
## 44.63542       266.7302 251.2582 282.2022 243.0678 290.3926
## 44.64583       256.2154 240.7434 271.6875 232.5530 279.8779
## 44.65625       256.3006 240.8285 271.7726 232.6381 279.9630
## 44.66667       263.3586 247.8866 278.8306 239.6962 287.0210
## 44.67708       257.3361 241.8640 272.8081 233.6736 280.9985
## 44.68750       257.3674 241.8954 272.8394 233.7050 281.0298
## 44.69792       260.2654 244.7933 275.7374 236.6029 283.9278
## 44.70833       250.1356 234.6636 265.6076 226.4732 273.7980
## 44.71875       256.4312 240.9592 271.9033 232.7688 280.0937
## 44.72917       260.6572 245.1852 276.1293 236.9948 284.3197
## 44.73958       296.9452 281.4732 312.4172 273.2828 320.6076
## 44.75000       303.6008 288.1288 319.0729 279.9384 327.2633
## 44.76042       301.1218 285.6498 316.5938 277.4594 324.7842
## 44.77083       299.7201 284.2480 315.1921 276.0576 323.3825
## 44.78125       300.6296 285.1576 316.1017 276.9672 324.2921
## 44.79167       300.0694 284.5974 315.5415 276.4070 323.7319
## 44.80208       304.9567 289.4847 320.4288 281.2943 328.6192
## 44.81250       305.9053 290.4333 321.3774 282.2429 329.5678
## 44.82292       301.2443 285.7722 316.7163 277.5818 324.9067
## 44.83333       300.1819 284.7098 315.6539 276.5194 323.8443
## 44.84375       306.2390 290.7669 321.7110 282.5765 329.9014
## 44.85417       301.5992 286.1271 317.0712 277.9367 325.2616
## 44.86458       298.1192 282.6472 313.5912 274.4568 321.7816
## 44.87500       302.9854 287.5134 318.4574 279.3230 326.6478
## 44.88542       300.4148 284.9427 315.8868 276.7523 324.0772
## 44.89583       294.2270 278.7550 309.6991 270.5646 317.8895
## 44.90625       293.5288 278.0568 309.0008 269.8664 317.1912
## 44.91667       294.0479 278.5758 309.5199 270.3854 317.7103
## 44.92708       280.3699 264.8979 295.8420 256.7075 304.0324
## 44.93750       276.2753 260.8032 291.7473 252.6128 299.9377
## 44.94792       275.5023 260.0302 290.9743 251.8398 299.1647
## 44.95833       273.1097 257.6377 288.5817 249.4473 296.7721
## 44.96875       194.8725 179.4005 210.3445 171.2101 218.5349
## 44.97917       194.3113 178.8393 209.7834 170.6489 217.9738
## 44.98958       161.1644 145.6924 176.6364 137.5020 184.8268
```

```
a <- my_forecast$mean
a
```

```
## Time Series:
## Start = c(44, 1)
## End = c(44, 96)
## Frequency = 96
##  [1] 157.7340 158.4851 154.0566 158.3692 161.2805 158.7010 149.2463 151.3852
##  [9] 152.8215 154.2589 156.5917 154.4061 153.8069 149.8821 149.0312 153.8286
## [17] 155.6973 154.5833 154.7730 156.1286 157.3212 157.9903 157.6716 158.5941
## [25] 159.8843 164.4135 177.2238 177.4589 176.4010 175.3251 168.6435 169.7900
## [33] 173.1589 255.4657 252.4563 250.3859 249.2635 255.4399 249.5638 249.8632
## [41] 252.1901 248.8533 251.3780 252.8936 252.7578 254.6923 253.4007 252.3836
## [49] 252.7304 255.6339 260.5471 259.6563 255.8878 257.9206 258.4071 258.5423
## [57] 255.4910 257.3629 260.4653 259.6117 261.2374 268.6193 258.0183 258.1022
## [65] 265.2375 259.4713 259.4932 262.4474 252.2467 258.6383 262.7708 299.1243
## [73] 305.8350 302.8006 301.4080 302.3352 301.7641 306.7032 307.6927 302.9891
## [81] 301.9320 308.0469 303.3855 299.8862 304.8040 302.5513 296.3300 295.6309
## [89] 296.1677 282.1381 278.0230 277.2881 274.8738 196.0950 195.5223 162.3571
```

```
b <- my_forecast_uni$mean
b
```

```
## Time Series:
## Start = c(44, 1)
## End = c(44, 96)
## Frequency = 96
##  [1] 157.4670 158.3195 154.2243 158.3993 161.2748 158.8154 149.1658 151.3115
##  [9] 152.7258 154.3277 156.6818 154.4646 153.8485 149.7376 148.8613 153.6472
## [17] 155.4936 154.1624 154.3398 155.6878 156.8664 157.2559 156.9341 157.8624
## [25] 159.1375 163.2078 175.9860 176.2341 175.1868 174.4419 167.7597 168.8877
## [33] 172.2559 255.1766 252.1841 250.1033 248.9869 255.5730 249.7056 250.0180
## [41] 252.3419 249.7141 252.2259 253.7509 253.6058 255.1238 253.8268 252.8125
## [49] 253.1654 255.3462 260.2550 259.3765 255.6065 257.2073 257.6999 257.8178
## [57] 254.7873 256.0544 259.1377 258.2882 259.8872 266.7302 256.2154 256.3006
## [65] 263.3586 257.3361 257.3674 260.2654 250.1356 256.4312 260.6572 296.9452
## [73] 303.6008 301.1218 299.7201 300.6296 300.0694 304.9567 305.9053 301.2443
## [81] 300.1819 306.2390 301.5992 298.1192 302.9854 300.4148 294.2270 293.5288
## [89] 294.0479 280.3699 276.2753 275.5023 273.1097 194.8725 194.3113 161.1644
```

```
write(a, file = "myforecast",
      ncolumns = 1,
      append = FALSE, sep = " ")  #we have imported our forecasts
```