DATA SCIENCETECH INSTITUTE

EXAM

Time Series Analysis

Done by: Mohamed Abid

Date: June 22, 2022

I. Preparation of data

1. Load packages and Initialize

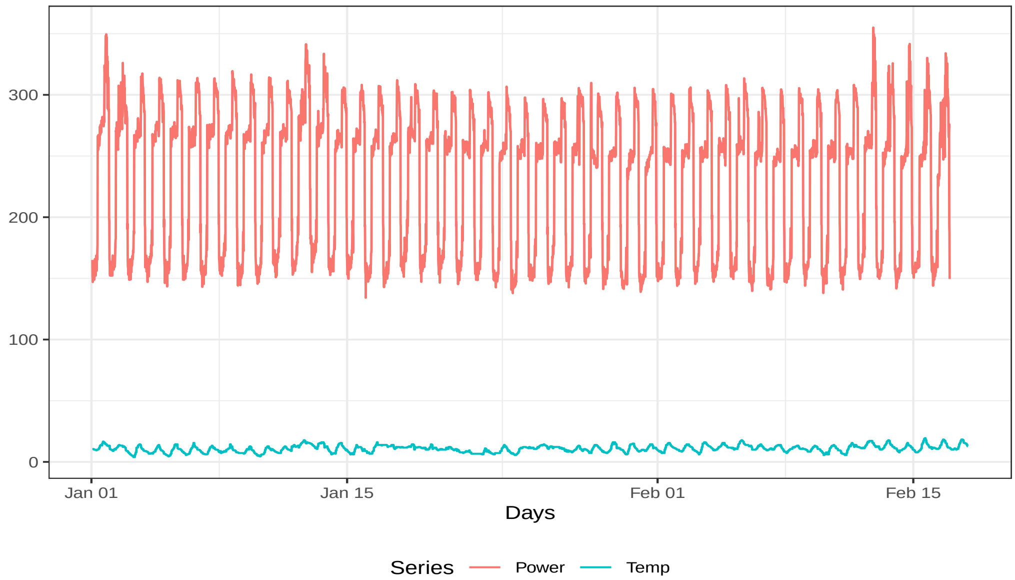
```
# Load packages
library(openxlsx)
library(xts)
library(Matrix)
library(ggplot2)
library(forecast)
library(keras)
library(vars)
# Change default theme
theme_set(theme_bw(10))
# RMSE function
RMSE.val <- function(X.true, X.test){
  return( sqrt( mean( (X.true - X.test)^2 ) ) )
}
```

2. Load and split data

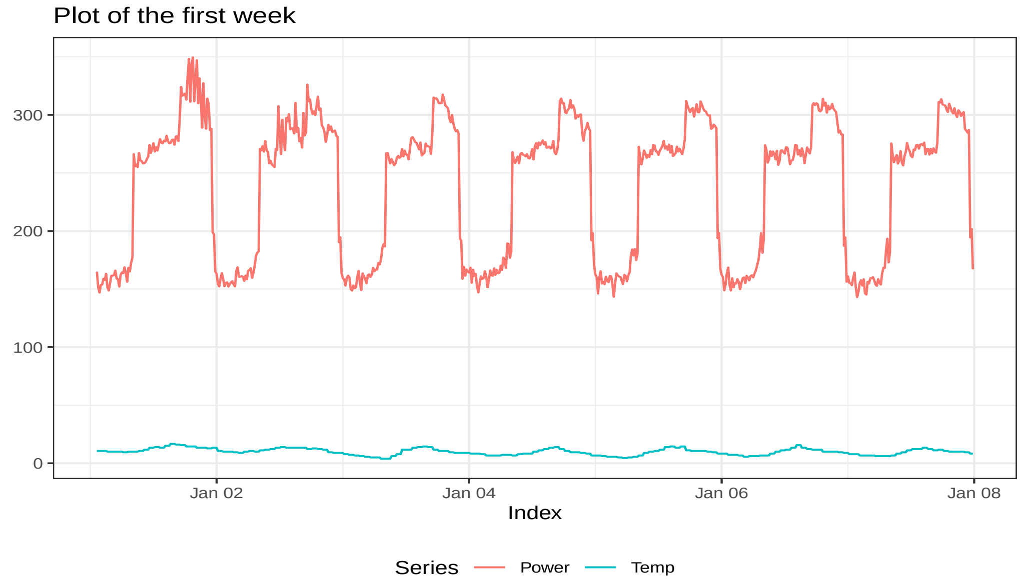
```
# Load data
Elec = read.xlsx('Elec-train.xlsx', 1)
# Convert data to 'xts' object
colnames(Elec) = c('Timestamp', 'Power', 'Temp')
xts_Elec = xts(Elec[, c('Power', 'Temp')], order.by = strptime(Elec[, "Timestamp"],
                                                   format = "%m/%d/%Y  %H:%M"))
# Split data
xts_Elec.train = xts_Elec['/2010-02-15']
xts_Elec.test  = xts_Elec['2010-02-16']
```

3. Plot data

```
# Plot all data
autoplot(xts_Elec[, c('Power', 'Temp')], facets = NULL) + labs(x = "Days") +
  theme(legend.position="bottom")
```



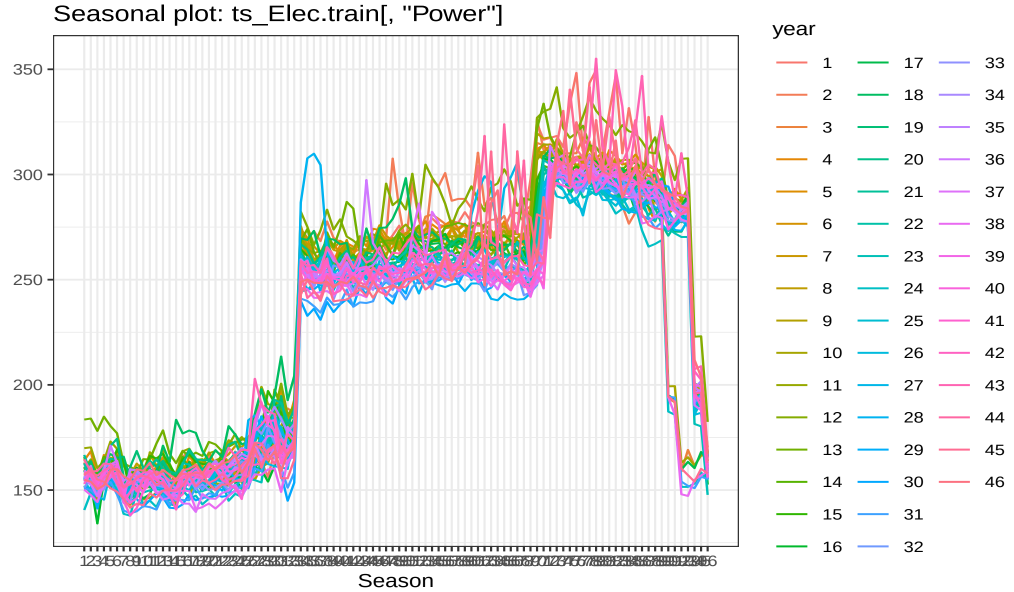
```
# plot of the first week
autoplot(xts_Elec['2010-01-01/2010-01-07', c('Power', 'Temp')], facets = NULL,
         main = "Plot of the first week") +
  theme(legend.position="bottom")
```



It's clear that the time series presents a one day periodic pattern for the "Power" data.

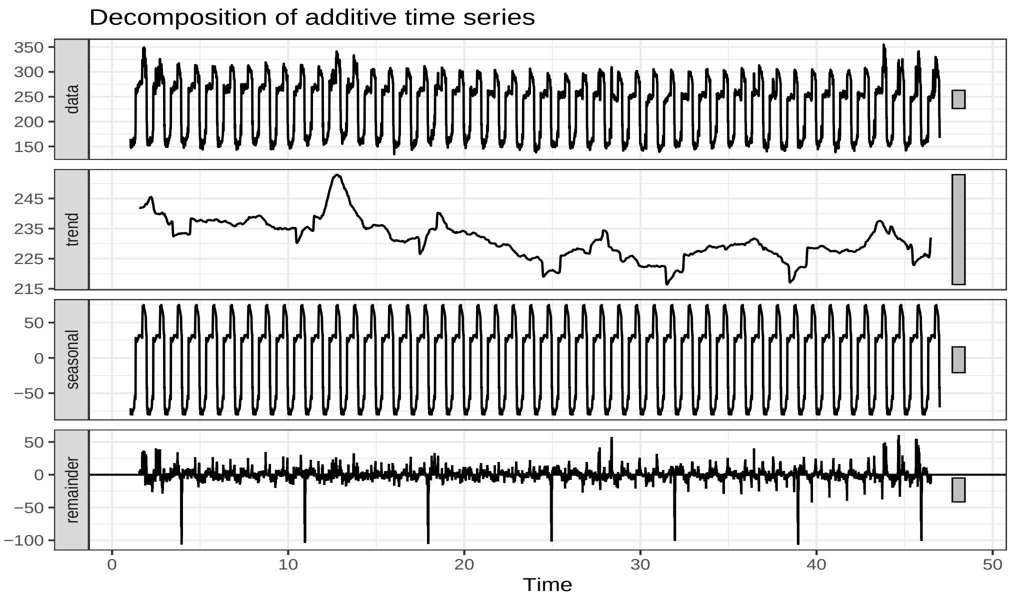
⇒ We create a time series with frequency equal to 96 (number of observations per day).

```
# Create time series
ts_Elec.train = ts(coredata(xts_Elec.train), frequency = 96, start = c(1,6))
ts_Elec.test = ts(coredata(xts_Elec.test), frequency = 96, start = c(47,1))
# Seasonal plot
ggseasonplot(ts_Elec.train[, 'Power'])
```



The above figure confirms our hypothesis about the one-day periodicity of the time series.

```
# Decompose of additive time series
components_power = decompose(ts_Elec.train[, 'Power'])
autoplot(components_power)
```



⇒ From the above figure, we can see the time series presents both seasonal and trend patterns.

We have a data with frequency equal to 96 which is a high frequency.

• Using frequency = 96,

- We cannot use the function “hw()”, which accepts at maximum a frequancy equal to 24.
- We can use the function “HoltWinters()” but we cannot apply the damping effect or Box-Cox.
- We can use Auto ARIMA and NNET or search a best SARIMA model.

I applied all this techniques but I couldn’t obtain a good model that presents uncorrelated residuals.

• I followed a second approach in which the data is downsampled to obtain 4 time series, each with a frequency equal to 24. We proceed into the following steps:

First step: We split the data into 4 parts, by taking one value from each hour to obtain:

- Data 1 contains the observations of minute 00 from each hour.
- Data 2 contains the observations of minute 15 from each hour.
- Data 3 contains the observations of minute 30 from each hour.
- Data 4 contains the observations of minute 45 from each hour.

Second step: We forecast each data using the models we saw during the course. We obtain :

- Data 1 is used to predict the “Power” values at minute 00 of each hour.
- Data 2 is used to predict the “Power” values at minute 15 of each hour.
- Data 3 is used to predict the “Power” values at minute 30 of each hour.
- Data 4 is used to predict the “Power” values at minute 45 of each hour.

Third step: By appropriately combining information from the forecasted data, we obtain the 96 predicted values for the next day.

Fourth step: We compute the RMSE of each model and choose the best one to be applied on all data to perform the forecast.

We will use the “xts” package (eXtensible Time Series) which will help us split and concatenate data with respect to the time of each observation.

II. Forecast without using outdoor temperature

1. Split data into 4 parts

```
lxts_Elec.train = list(); lxts_Elec.test = list()
lts_Elec.train = list(); time.Elec.test = list()
for (i in 1:4){
  # Create a list of xts object.
  lxts_Elec.train[[i]] = xts_Elec.train[.indexmin(xts_Elec.train) == 15*(i-1)]
  lxts_Elec.test[[i]] = xts_Elec.test[.indexmin(xts_Elec.test) == 15*(i-1)]
  # Create list of time series object
  if (i == 1)
    lts_Elec.train[[i]] = ts(coredata(lxts_Elec.train[[i]]), frequency = 24, start = c(1,3))
  else
    lts_Elec.train[[i]] = ts(coredata(lxts_Elec.train[[i]]), frequency = 24, start = c(1,2))
  # Prepare time of each forecasted data part to create the xts object
  time.Elec.test[[i]] = time(lxts_Elec.test[[i]])
}
# Verification of data
for (i in 1:4){
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  print(tail(lxts_Elec.train[[i]], 3))
  print(tail(lts_Elec.train[[i]], 3))
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##                     Power     Temp
## 2010-02-15 21:00:00 304.0 12.77778
## 2010-02-15 22:00:00 293.1 11.66667
## 2010-02-15 23:00:00 283.9 11.66667
## Time Series:
## Start = c(46, 22)
## End = c(46, 24)
## Frequency = 24
##          Power     Temp
## 46.87500 304.0 12.77778
## 46.91667 293.1 11.66667
## 46.95833 283.9 11.66667
```

```
## * --- Data 2 (Minute 15 of each hour) --- *
##                     Power     Temp
## 2010-02-15 21:15:00 297.1 11.66667
## 2010-02-15 22:15:00 291.2 11.66667
## 2010-02-15 23:15:00 211.9 12.22222
## Time Series:
## Start = c(46, 22)
## End = c(46, 24)
## Frequency = 24
##          Power     Temp
## 46.87500 297.1 11.66667
## 46.91667 291.2 11.66667
## 46.95833 211.9 12.22222
```

```
## * --- Data 3 (Minute 30 of each hour) --- *
##                     Power     Temp
## 2010-02-15 21:30:00 302.7 11.66667
## 2010-02-15 22:30:00 289.6 11.66667
## 2010-02-15 23:30:00 203.8 12.22222
## Time Series:
## Start = c(46, 22)
## End = c(46, 24)
## Frequency = 24
##          Power     Temp
## 46.87500 302.7 11.66667
## 46.91667 289.6 11.66667
## 46.95833 203.8 12.22222
```

```
## * --- Data 4 (Minute 45 of each hour) --- *
##                     Power     Temp
## 2010-02-15 21:45:00 295.4 11.66667
## 2010-02-15 22:45:00 284.7 11.66667
## 2010-02-15 23:45:00 167.2 12.22222
## Time Series:
## Start = c(46, 22)
## End = c(46, 24)
## Frequency = 24
##          Power     Temp
## 46.87500 295.4 11.66667
## 46.91667 284.7 11.66667
## 46.95833 167.2 12.22222
```

2. Additive seasonal HoltWinters

```
fit4.hw1.add1 = list(); prev4.hw1.add1 = list(); lxts_prev4.hw1.add1 = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
```

```
  # Additive seasonal HoltWinters
  fit4.hw1.add1[[i]] = HoltWinters(lts_Elec.train[[i]][, 'Power'])
  prev4.hw1.add1[[i]] = forecast(fit4.hw1.add1[[i]], h = 24)
  lxts_prev4.hw1.add1[[i]] = xts(prev4.hw1.add1[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.hw1.add1 = do.call(rbind, lxts_prev4.hw1.add1)
# Compute the RMSE
rmse4.hw1.add1 = RMSE.val(xts_prev4.hw1.add1, xts_Elec.test[, 'Power'])
cat ('RMSE of Additive seasonal HoltWinters =', rmse4.hw1.add1, '\n')
```

```
## RMSE of Additive seasonal HoltWinters = 15.22545
```

3. Multiplicative seasonal HoltWinters

```
fit4.hw1.mult1 = list(); prev4.hw1.mult1 = list(); lxts_prev4.hw1.mult1 = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # Multiplicative seasonal HoltWinters
  fit4.hw1.mult1[[i]] = HoltWinters(lts_Elec.train[[i]][, 'Power'], seasonal = "multiplicative")
  prev4.hw1.mult1[[i]] = forecast(fit4.hw1.mult1[[i]], h = 24)
  lxts_prev4.hw1.mult1[[i]] = xts(prev4.hw1.mult1[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.hw1.mult1 = do.call(rbind, lxts_prev4.hw1.mult1)
# Compute the RMSE
rmse4.hw1.mult1 = RMSE.val(xts_prev4.hw1.mult1, xts_Elec.test[, 'Power'])
cat ('RMSE of Multiplicative seasonal HoltWinters =', rmse4.hw1.mult1, '\n')
```

```
## RMSE of Multiplicative seasonal HoltWinters = 16.67767
```

4. Additive seasonal HW

```
fit4.hw2.add1 = list(); lxts_fit4.hw2.add1 = list()
fit4.hw2.add2 = list(); lxts_fit4.hw2.add2 = list()
fit4.hw2.add3 = list(); lxts_fit4.hw2.add3 = list()
fit4.hw2.add4 = list(); lxts_fit4.hw2.add4 = list()
# Forecast and convert the result to xts object
for (i in 1:4){
  # Additive seasonal HW
  fit4.hw2.add1[[i]] = hw(lts_Elec.train[[i]][,'Power'], h = 24)
  lxts_fit4.hw2.add1[[i]] = xts(fit4.hw2.add1[[i]]$mean, order.by = time.Elec.test[[i]])
  # Additive seasonal HW + damping
  fit4.hw2.add2[[i]] = hw(lts_Elec.train[[i]][,'Power'], h = 24, damped = T)
  lxts_fit4.hw2.add2[[i]] = xts(fit4.hw2.add2[[i]]$mean, order.by = time.Elec.test[[i]])
  # Additive seasonal HW + Box-Cox
  fit4.hw2.add3[[i]] = hw(lts_Elec.train[[i]][,'Power'], h = 24, lambda = 'auto')
  lxts_fit4.hw2.add3[[i]] = xts(fit4.hw2.add3[[i]]$mean, order.by = time.Elec.test[[i]])
  # Additive seasonal HW + damping + Box-Cox
  fit4.hw2.add4[[i]] = hw(lts_Elec.train[[i]][,'Power'], h = 24, damped = T, lambda = 'auto')
  lxts_fit4.hw2.add4[[i]] = xts(fit4.hw2.add4[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_fit4.hw2.add1 = do.call(rbind, lxts_fit4.hw2.add1)
xts_fit4.hw2.add2 = do.call(rbind, lxts_fit4.hw2.add2)
xts_fit4.hw2.add3 = do.call(rbind, lxts_fit4.hw2.add3)
```

```
xts_fit4.hw2.add4 = do.call(rbind, lxts_fit4.hw2.add4)
# Compute the RMSE
rmse4.hw2.add1 = RMSE.val(xts_fit4.hw2.add1, xts_Elec.test[, 'Power'])
rmse4.hw2.add2 = RMSE.val(xts_fit4.hw2.add2, xts_Elec.test[, 'Power'])
rmse4.hw2.add3 = RMSE.val(xts_fit4.hw2.add3, xts_Elec.test[, 'Power'])
rmse4.hw2.add4 = RMSE.val(xts_fit4.hw2.add4, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of Additive seasonal HW =', rmse4.hw2.add1, '\n')
```

```
## RMSE of Additive seasonal HW = 15.75758
```

```
cat ('RMSE of Additive seasonal HW with damping =', rmse4.hw2.add2, '\n')
```

```
## RMSE of Additive seasonal HW with damping = 15.37618
```

```
cat ('RMSE of Additive seasonal HW with Box-Cox =', rmse4.hw2.add3, '\n')
```

```
## RMSE of Additive seasonal HW with Box-Cox = 15.898
```

```
cat ('RMSE of Additive seasonal HW with damping and Box-Cox =', rmse4.hw2.add4, '\n')
```

```
## RMSE of Additive seasonal HW with damping and Box-Cox = 16.12223
```

5. Multiplicative seasonal HW

```
fit4.hw2.mult1 = list(); lxts_fit4.hw2.mult1 = list()
fit4.hw2.mult2 = list(); lxts_fit4.hw2.mult2 = list()
# Forecast and convert the result to xts object
for (i in 1:4){
  # Multiplicative seasonal HW
  fit4.hw2.mult1[[i]] = hw(lts_Elec.train[[i]][, 'Power'], h = 24, seasonal = "multiplicative")
  lxts_fit4.hw2.mult1[[i]] = xts(fit4.hw2.mult1[[i]]$mean, order.by = time.Elec.test[[i]])
  # Multiplicative seasonal HW + damping
  fit4.hw2.mult2[[i]] = hw(lts_Elec.train[[i]][, 'Power'], h = 24, seasonal = "multiplicative",
                           damped = T)
  lxts_fit4.hw2.mult2[[i]] = xts(fit4.hw2.mult2[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_fit4.hw2.mult1 = do.call(rbind, lxts_fit4.hw2.mult1)
xts_fit4.hw2.mult2 = do.call(rbind, lxts_fit4.hw2.mult2)
# Compute the RMSE
rmse4.hw2.mult1 = RMSE.val(xts_fit4.hw2.mult1, xts_Elec.test[, 'Power'])
rmse4.hw2.mult2 = RMSE.val(xts_fit4.hw2.mult2, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of Multiplicative seasonal HW =', rmse4.hw2.mult1, '\n')
```

```
## RMSE of Multiplicative seasonal HW = 16.88001
```

```
cat ('RMSE of Multiplicative seasonal HW with damping =', rmse4.hw2.mult2, '\n')
```

```
## RMSE of Multiplicative seasonal HW with damping = 16.84145
```

6. Auto ARIMA

```
fit4.autoArima1 = list(); prev4.autoArima1 = list(); lxts_prev4.autoArima1 = list()
fit4.autoArima2 = list(); prev4.autoArima2 = list(); lxts_prev4.autoArima2 = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # Auto ARIMA
```

```
  fit4.autoArima1[[i]] = auto.arima(lts_Elec.train[[i]][, 'Power'])
  prev4.autoArima1[[i]] = forecast(fit4.autoArima1[[i]], h = 24)
  lxts_prev4.autoArima1[[i]] = xts(prev4.autoArima1[[i]]$mean, order.by = time.Elec.test[[i]])
  # Auto ARIMA + Box-Cox
  fit4.autoArima2[[i]] = auto.arima(lts_Elec.train[[i]][, 'Power'], lambda = 'auto')
  prev4.autoArima2[[i]] = forecast(fit4.autoArima2[[i]], h = 24)
  lxts_prev4.autoArima2[[i]] = xts(prev4.autoArima2[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.autoArima1 = do.call(rbind, lxts_prev4.autoArima1)
xts_prev4.autoArima2 = do.call(rbind, lxts_prev4.autoArima2)
# Compute the RMSE
rmse4.autoArima1 = RMSE.val(xts_prev4.autoArima1, xts_Elec.test[, 'Power'])
rmse4.autoArima2 = RMSE.val(xts_prev4.autoArima2, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of Auto ARIMA =', rmse4.autoArima1, '\n')
```

```
## RMSE of Auto ARIMA = 17.6711
```

```
cat ('RMSE of Auto ARIMA with Box-Cox =', rmse4.autoArima2, '\n')
```

```
## RMSE of Auto ARIMA with Box-Cox = 17.08156
```

7. NNET

```
fit4.nnet1 = list(); prev4.nnet1 = list(); lxts_prev4.nnet1 = list()
fit4.nnet2 = list(); prev4.nnet2 = list(); lxts_prev4.nnet2 = list()
  # Forecast and convert the results to xts objects.
for (i in 1:4){
  # NNET
  fit4.nnet1[[i]] = nnetar(lts_Elec.train[[i]][, 'Power'])
  prev4.nnet1[[i]] = forecast(fit4.nnet1[[i]], h = 24)
  lxts_prev4.nnet1[[i]] = xts(prev4.nnet1[[i]]$mean, order.by = time.Elec.test[[i]])
  # NNET + Box-Cox
  fit4.nnet2[[i]] = nnetar(lts_Elec.train[[i]][, 'Power'], lambda = 'auto')
  prev4.nnet2[[i]] = forecast(fit4.nnet2[[i]], h = 24)
  lxts_prev4.nnet2[[i]] = xts(prev4.nnet2[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.nnet1 = do.call(rbind, lxts_prev4.nnet1)
xts_prev4.nnet2 = do.call(rbind, lxts_prev4.nnet2)
# Compute the RMSE
rmse4.nnet1 = RMSE.val(xts_prev4.nnet1, xts_Elec.test[, 'Power'])
rmse4.nnet2 = RMSE.val(xts_prev4.nnet2, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of NNET =', rmse4.nnet1, '\n')
```

```
## RMSE of NNET = 16.22362
```

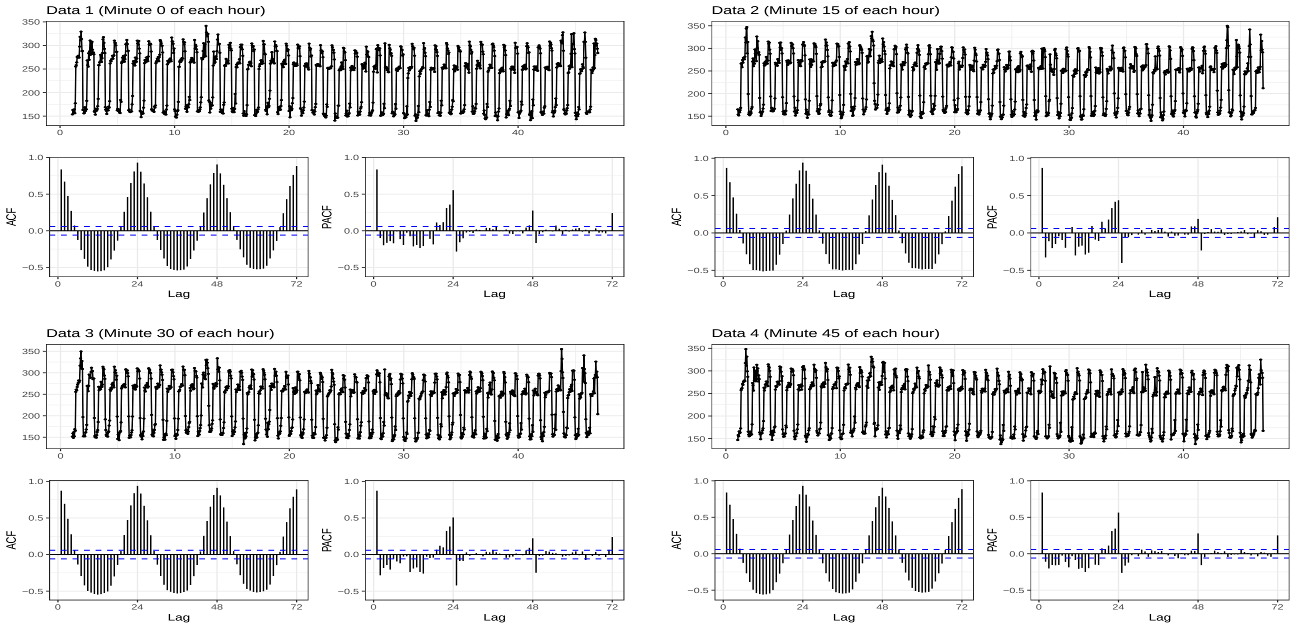
```
cat ('RMSE of NNET with Box-Cox =', rmse4.nnet2, '\n')
```

```
## RMSE of NNET with Box-Cox = 16.95131
```

8. SARIMA

Plot

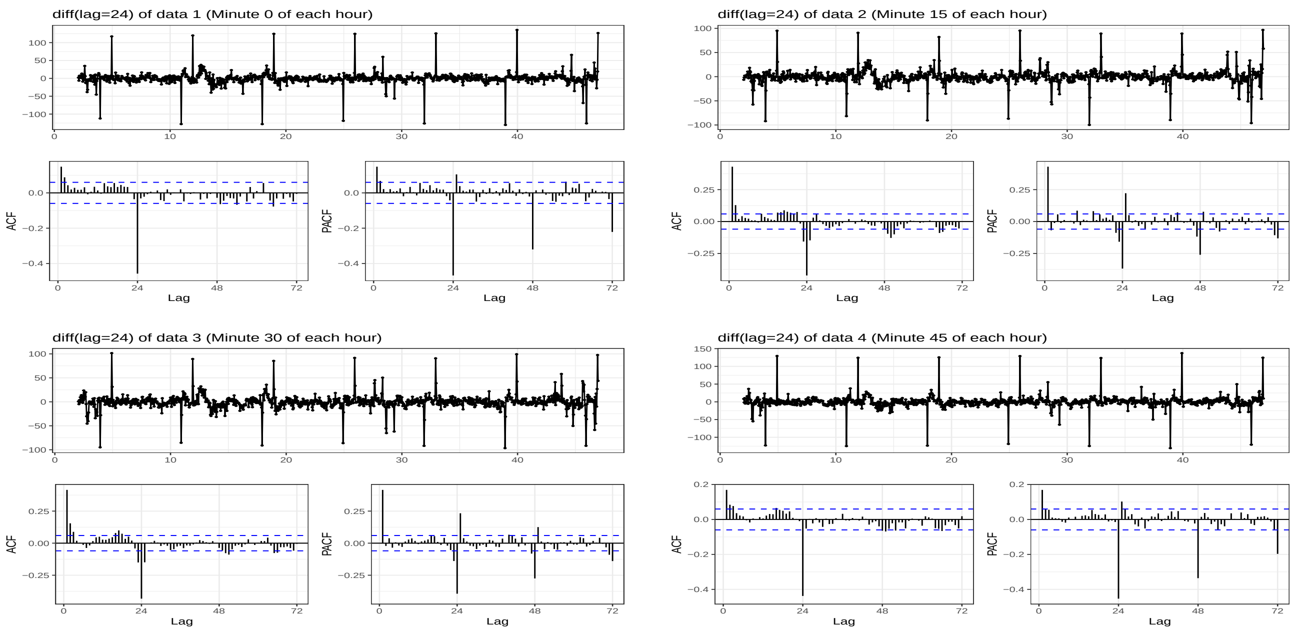
```
for (i in 1:4){
  titleOfPlot = paste('Data ', i, ' (Minute ', 15*(i-1), ' of each hour)' , sep='')
  ggtsdisplay(lts_Elec.train[[i]][, 'Power'], main = titleOfPlot)
}
```



Differencing : lag=24

We remove the seasonality by applying the difference operation with lag = 24.

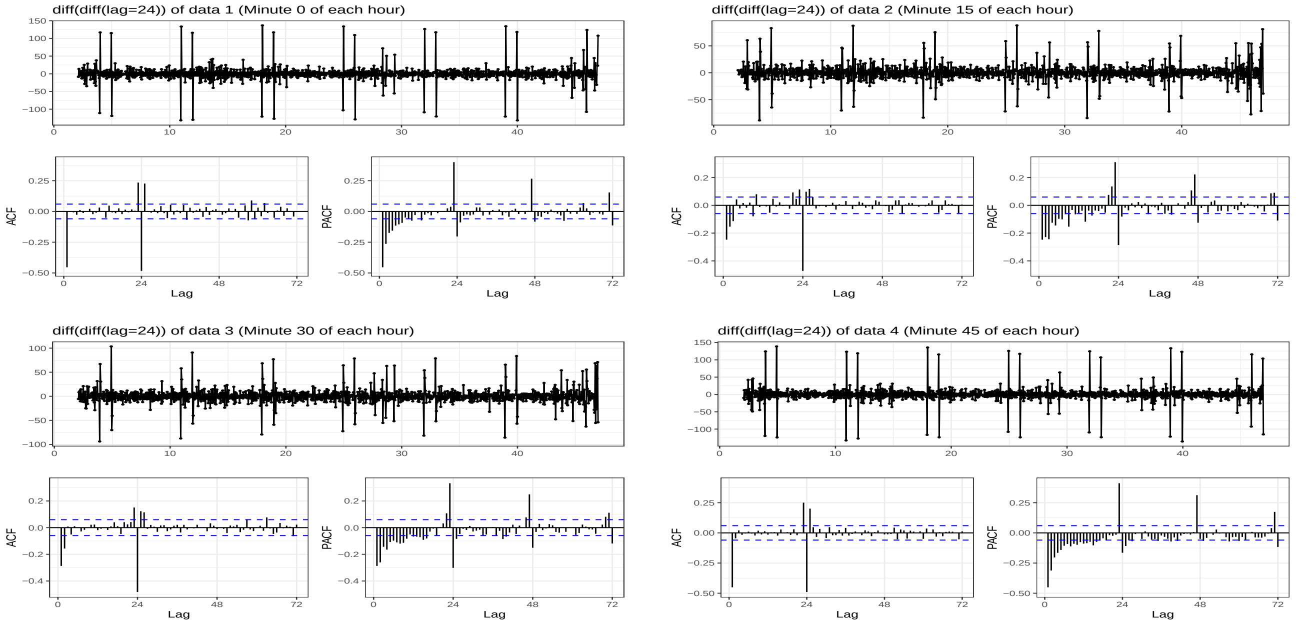
```
power.diff0 = list()
for (i in 1:4){
  titleOfPlot = paste('diff(lag=24) of data ', i, ' (Minute ', 15*(i-1), ' of each hour)' , sep='')
  power.diff0[[i]] = diff(lts_Elec.train[[i]][, 'Power'], lag = 24)
  ggtsdisplay(power.diff0[[i]], main = titleOfPlot)
}
```



Differencing : lag(lag=24)

We remove the trend by applying the difference operation to the previous result with lag = 1.

```
power.diff1 = list()
for (i in 1:4){
  titleOfPlot = paste('diff(diff(lag=24)) of data ', i, ' (Minute ', 15*(i-1), ' of each hour)' ,
                      sep='')
  power.diff1[[i]] = diff(power.diff0[[i]])
  ggtsdisplay(power.diff1[[i]], main = titleOfPlot)
}
```



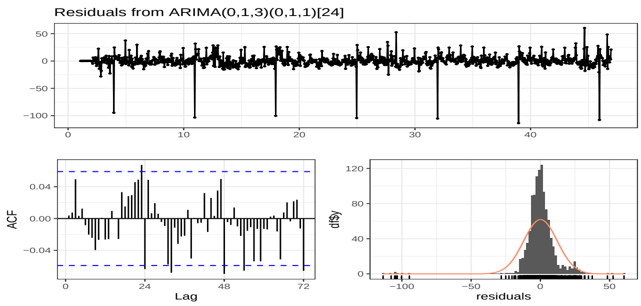
From these graphs, we can see:
- An exponential decrease on the "pacf" graph for all data.
- A significant ACF at lag = 24 for all data suggesting a seasonal MA_1.
- A significant ACF at lag = 1, lag = 3, lag = 2, lag = 1 for data 1, 2, 3, 4 respectively, suggesting a non-seasonal MA_3.

$\Rightarrow$ We propose the model $\text{SARIMA}_{(0,1,3)(0,1,1)[24]}$

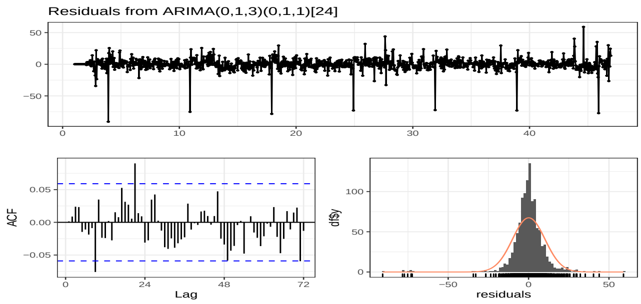
$\textbf{SARIMA}_{(0,1,3)(0,1,1)[24]}$

```
fit4.arima11 = list()
# Create the model
for (i in 1:4){
  # SARIMA(0,1,3)(0,1,1)[24]
  fit4.arima11[[i]] = Arima(lts_Elec.train[[i]][, 'Power'], order = c(0,1,3), seasonal = c(0,1,1))
  # Check residuals
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.arima11[[i]])
}
```

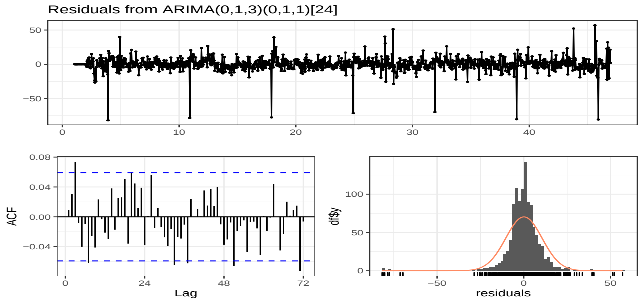
```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 57.263, df = 44, p-value = 0.0866
##
## Model df: 4.   Total lags used: 48
```



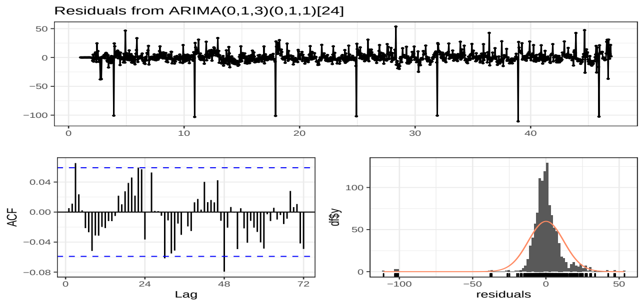
```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 47.588, df = 44, p-value = 0.3288
##
## Model df: 4.   Total lags used: 48
```




```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 61.639, df = 44, p-value = 0.04056
##
## Model df: 4.   Total lags used: 48
```




```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 57.703, df = 44, p-value = 0.08056
##
## Model df: 4.   Total lags used: 48
```



We have p-value = 0.04056 for data 3, and we can see there is still significant ACF at lag = 7 in data 3.

$\Rightarrow$ We propose the model $\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$

$\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$

```
fit4.arima21 = list(); fit4.arima22 = list();
# Create the model
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  fit4.arima21[[i]] = Arima(lts_Elec.train[[i]][, 'Power'], order = c(0,1,7), seasonal = c(0,1,1))
  # SARIMA(0,1,7)(0,1,1)[24] + Box-Cox
  fit4.arima22[[i]] = Arima(lts_Elec.train[[i]][, 'Power'], order = c(0,1,7), seasonal = c(0,1,1),
                          lambda = 'auto')
  # Check residuals of SARIMA(0,1,7)(0,1,1)[24]
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.arima21[[i]], plot = F)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 48.524, df = 40, p-value = 0.167
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 43.77, df = 40, p-value = 0.3145
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 44.469, df = 40, p-value = 0.2891
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 41.299, df = 40, p-value = 0.4137
##
## Model df: 8.   Total lags used: 48
```

⇒ We have p-value > 0.05 for all data.

⇒ We will use the model $\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$ to forecast data.

```
prev4.arima21 = list(); lxts_prev4.arima21 = list()
prev4.arima22 = list(); lxts_prev4.arima22 = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  prev4.arima21[[i]] = forecast(fit4.arima21[[i]], h = 24)
  lxts_prev4.arima21[[i]] = xts(prev4.arima21[[i]]$mean, order.by = time.Elec.test[[i]])
  # SARIMA(0,1,7)(0,1,1)[24] + Box-Cox
  prev4.arima22[[i]] = forecast(fit4.arima22[[i]], h = 24)
  lxts_prev4.arima22[[i]] = xts(prev4.arima22[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.arima21 = do.call(rbind, lxts_prev4.arima21)
xts_prev4.arima22 = do.call(rbind, lxts_prev4.arima22)
# Compute the RMSE
rmse4.arima21 = RMSE.val(xts_prev4.arima21, xts_Elec.test[, 'Power'])
rmse4.arima22 = RMSE.val(xts_prev4.arima22, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of SARIMA(0,1,7)(0,1,1)[24] =', rmse4.arima21, '\n')
```

```
## RMSE of SARIMA(0,1,7)(0,1,1)[24] = 14.68489
```

```
cat ('RMSE of SARIMA(0,1,7)(0,1,1)[24] with Box-Cox =', rmse4.arima22, '\n')
```

```
## RMSE of SARIMA(0,1,7)(0,1,1)[24] with Box-Cox = 14.70515
```

9. Choose the model

Model summary

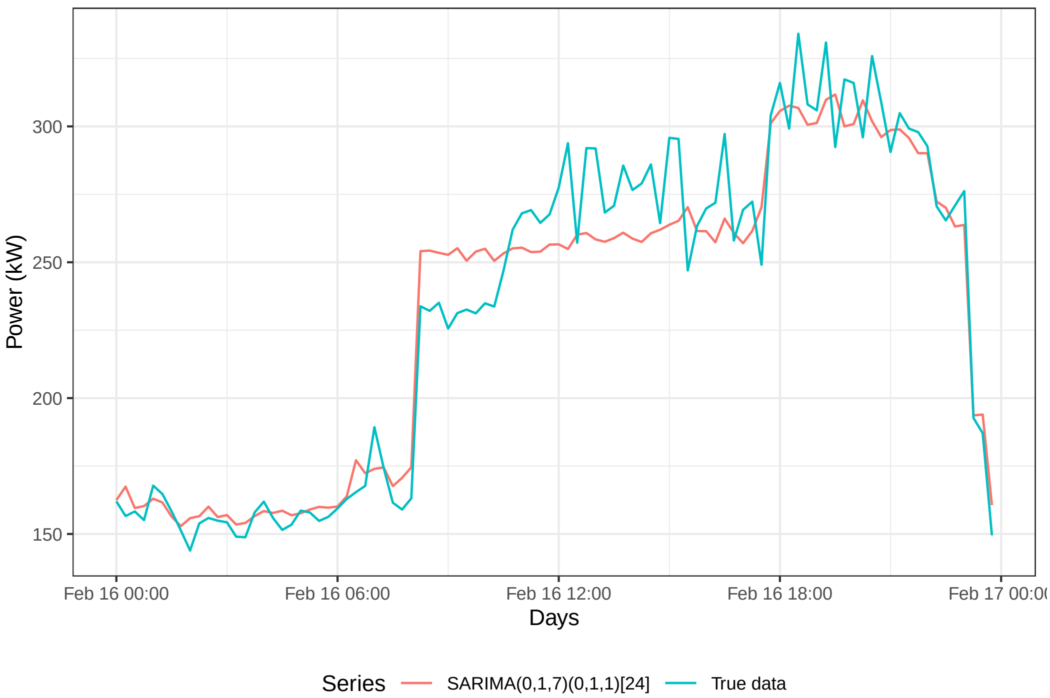
```
df = data.frame(RMSE = c(rmse4.hw1.add1, rmse4.hw1.mult1, rmse4.hw2.add2, rmse4.hw2.mult2,
                         rmse4.autoArima2, rmse4.nnet1, rmse4.arima21))
rownames(df) = c("Additive seasonal HoltWinters", "Multiplicative seasonal HoltWinters",
                 "Additive seasonal HW + damping","Multiplicative seasonal HW + damping",
                 "Auto ARIMA + Box-Cox", "NNET", "SARIMA(0,1,7)(0,1,1)[24]")
print(df)
```

```
##                                         RMSE
## Additive seasonal HoltWinters       15.22545
## Multiplicative seasonal HoltWinters 16.67767
## Additive seasonal HW + damping      15.37618
## Multiplicative seasonal HW + damping 16.84145
## Auto ARIMA + Box-Cox                17.08156
## NNET                                16.22362
## SARIMA(0,1,7)(0,1,1)[24]            14.68489
```

⇒ The best model is $\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$, we will apply it for prediction using all data.

Plot

```
xts_prevAndTest = cbind(xts_prev4.arima21, xts_Elec.test[, 'Power'])
names(xts_prevAndTest) = c("SARIMA(0,1,7)(0,1,1)[24]", "True data")
autoplot(xts_prevAndTest, facets = NULL) + labs(x = 'Days', y = 'Power (kW)') +
  theme(legend.position="bottom")
```



10. Forecast using all data

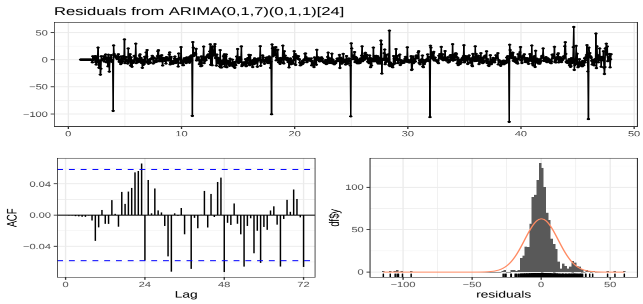
Prepare data

```
# Initialization
xts_Elec.fit = xts_Elec['/2010-02-16']
xts_Elec.prev = xts_Elec['2010-02-17']
lxts_Elec.fit = list(); lts_Elec.fit = list()
lxts_Elec.prev = list(); lts_Elec.prev = list()
time.Elec.prev = list()
for (i in 1:4){
  # Create a list of xts object.
  lxts_Elec.fit[[i]] = xts_Elec.fit[.indexmin(xts_Elec.fit) == 15*(i-1)]
  lxts_Elec.prev[[i]] = xts_Elec.prev[.indexmin(xts_Elec.prev) == 15*(i-1)]
  # Create list of time series object
  if (i == 1)
    lts_Elec.fit[[i]] = ts(coredata(lxts_Elec.fit[[i]]), frequency = 24, start = c(1,3))
  else
    lts_Elec.fit[[i]] = ts(coredata(lxts_Elec.fit[[i]]), frequency = 24, start = c(1,2))
  lts_Elec.prev[[i]] = ts(coredata(lxts_Elec.prev[[i]]), frequency = 24, start = c(48,1))
  # Prepare time of each forecasted data part
  time.Elec.prev[[i]] = time(lxts_Elec.prev[[i]])
}
```

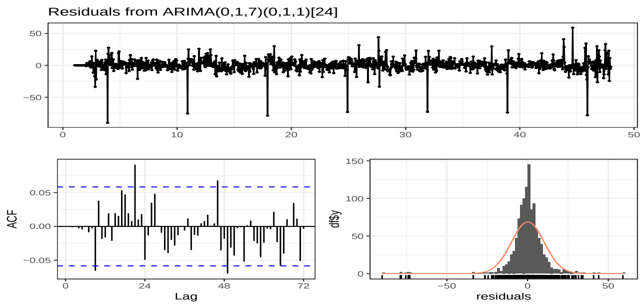
Forecast using all data and check the residuals

```
fit.all = list(); prev.all = list(); lxts_prev.all = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  fit.all[[i]] = Arima(lts_Elec.fit[[i]][, 'Power'], order = c(0,1,7), seasonal = c(0,1,1))
  prev.all[[i]] = forecast(fit.all[[i]], h = 24)
  lxts_prev.all[[i]] = xts(prev.all[[i]]$mean, order.by = time.Elec.prev[[i]])
  # Check residuals of SARIMA(0,1,7)(0,1,1)[24]
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit.all[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev.all = do.call(rbind, lxts_prev.all)
```

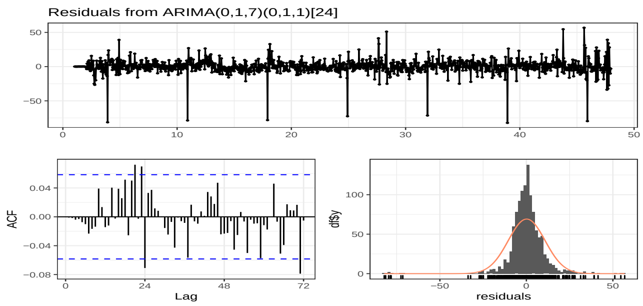
```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 55.541, df = 40, p-value = 0.05205
##
## Model df: 8.   Total lags used: 48
```



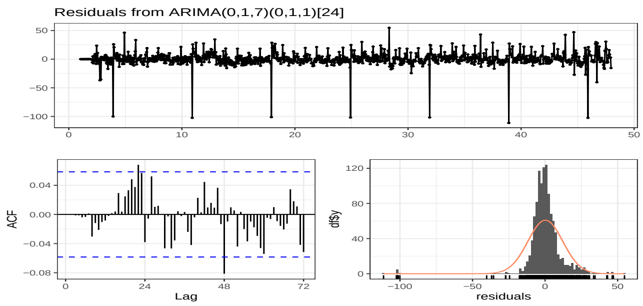

```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 47.397, df = 40, p-value = 0.1964
##
## Model df: 8.   Total lags used: 48
```




```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 49.548, df = 40, p-value = 0.1432
##
## Model df: 8.   Total lags used: 48
```



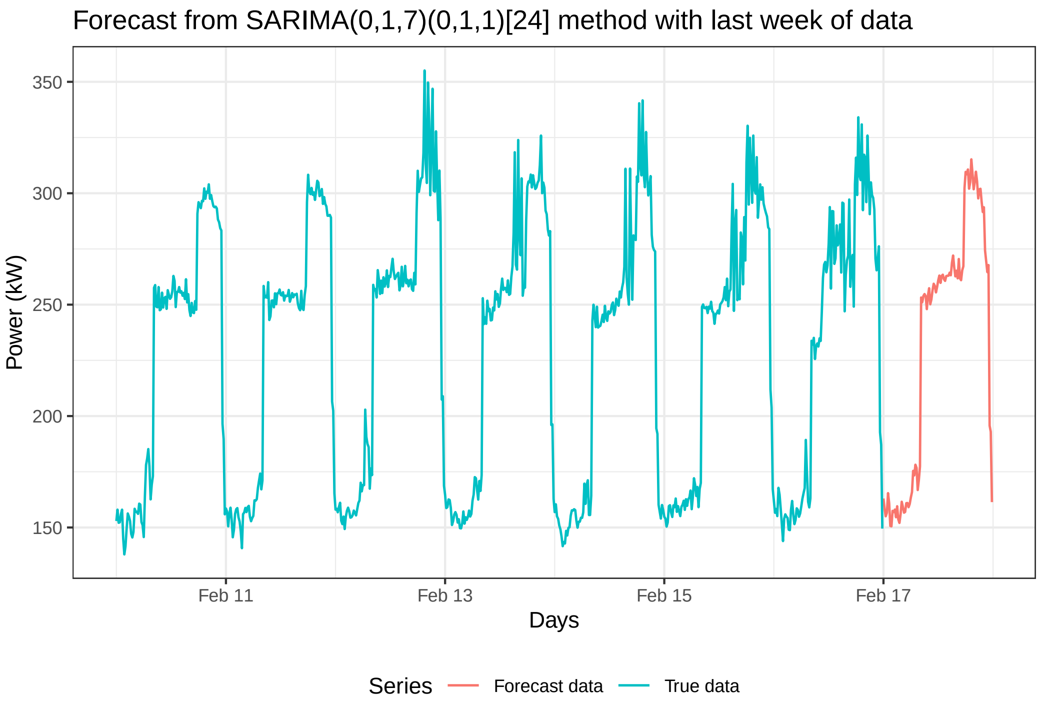

```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 45.816, df = 40, p-value = 0.2435
##
## Model df: 8.   Total lags used: 48
```



For all data parts, the residuals are approximatively Gaussian and the mean is approximatively equal to 0 and the p-value > 0.05.

Plot the forecasted time series

```
xts_prevAndTest = cbind(xts_prev.all, xts_Elec.fit['2010-02-10/2010-02-16', 'Power'])
names(xts_prevAndTest) = c("Forecast data", "True data")
autoplot(xts_prevAndTest, facets = NULL) + labs(x = 'Days', y = 'Power (kW)') +
  ggtitle("Forecast from SARIMA(0,1,7)(0,1,1)[24] method with last week of data") +
  theme(legend.position="bottom")
```



Save forecasted data

```
write.xlsx(xts_prev.all, "MohamedAbid.xlsx", startCol = 1, startRow = 1, colNames = F)
```

III. Forecast using outdoor temperature

1. Prepare data

```
lts_Elec.test = list()
for (i in 1:4){
  lts_Elec.test[[i]] = ts(coredata(lxts_Elec.test[[i]]), frequency = 24, start = c(47,1))
}
```

2. Time series regression models

We will use only the covariate in the function "tslm()" without specifying "trend" and "season". After that by looking at the residuals we will introduce the neccessary corrections.

```
fit4.tslm = list()
for (i in 1:4){
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  fit4.tslm[[i]] = tslm(Power ~ Temp, data = lts_Elec.train[[i]])
  print(summary(fit4.tslm[[i]])$coefficients)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 121.90648  6.2528623 19.49611 6.044955e-73
## Temp         10.13939  0.5619213 18.04414 5.878409e-64
## * --- Data 2 (Minute 15 of each hour) --- *
##               Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 125.906242  6.2258309 20.2232 1.393582e-77
## Temp          9.836683  0.5594398 17.5831 3.466375e-61
## * --- Data 3 (Minute 30 of each hour) --- *
##              Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 122.12009  6.2151792 19.64868 6.427208e-74
## Temp         10.16157  0.5584826 18.19495 7.007388e-65
## * --- Data 4 (Minute 45 of each hour) --- *
##              Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 120.27623  6.3047106 19.0772 2.525370e-70
## Temp         10.25806  0.5665277 18.1069 2.406676e-64
```

⇒ The feature 'Temperature' is significant.

```
for (i in 1:4){
  # Check residuals
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.tslm[[i]], test = 'LB', plot = F)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 6340.1, df = 46, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 48
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 7016.6, df = 46, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 48
```

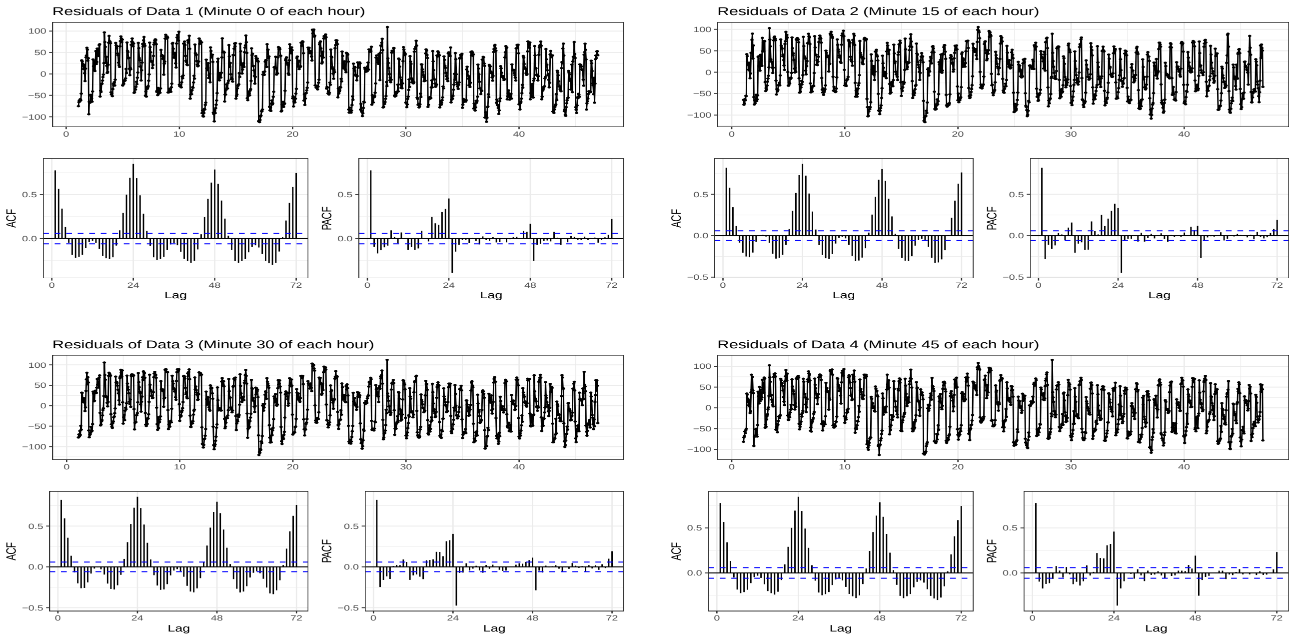
```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 6985.3, df = 46, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 48
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 6395.6, df = 46, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 48
```

The p-value < 0.05 for all data ⇒ the residuals are correlated.

⇒ We have to design a model for the residuals using SARIMA.

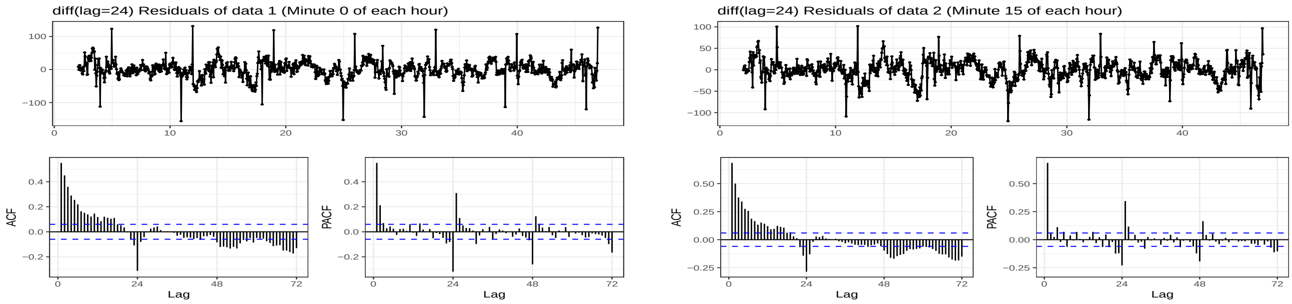
3. SARIMA

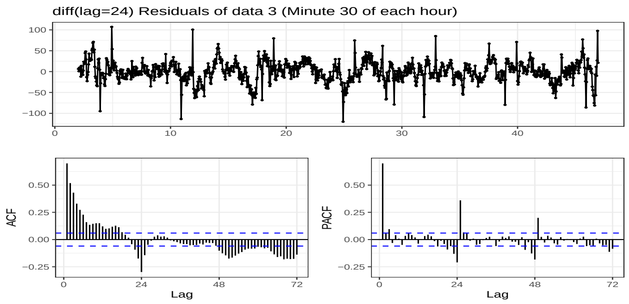
```
fit4.resd = list()
for (i in 1:4){
  titleOfPlot = paste('Residuals of Data ', i, ' (Minute ', 15*(i-1), ' of each hour)' , sep='')
  fit4.resd[[i]] = fit4.tslm[[i]]$residuals
  ggtsdisplay(fit4.resd[[i]], main = titleOfPlot)
}
```

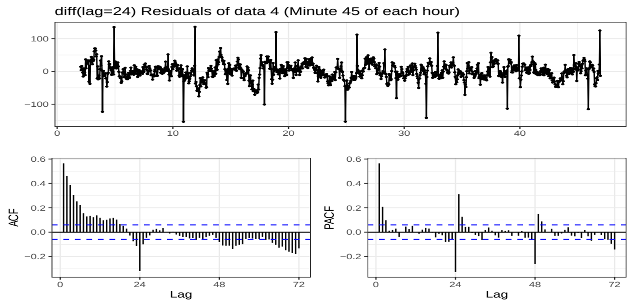


Differencing the residuals : lag=24

```
resd.diff0 = list()
for (i in 1:4){
  titleOfPlot = paste('diff(lag=24) Residuals of data ', i, ' (Minute ', 15*(i-1), ' of each hour)' ,
                      sep='')
  resd.diff0[[i]] = diff(fit4.resd[[i]], lag = 24)
  ggtsdisplay(resd.diff0[[i]], main = titleOfPlot)
}
```

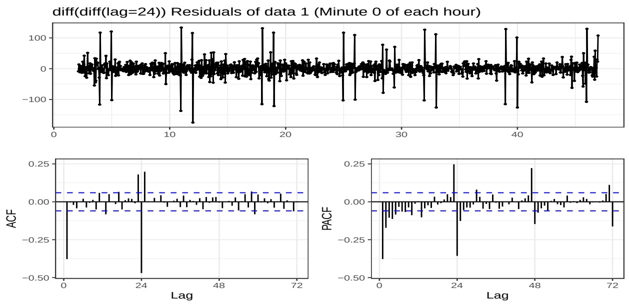


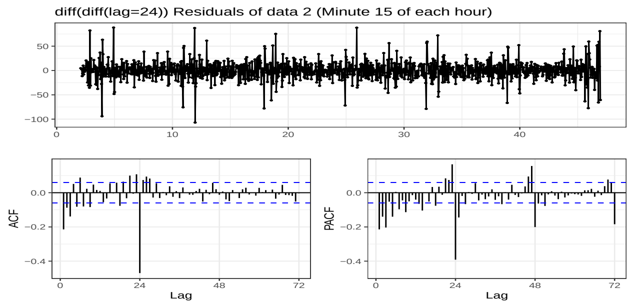


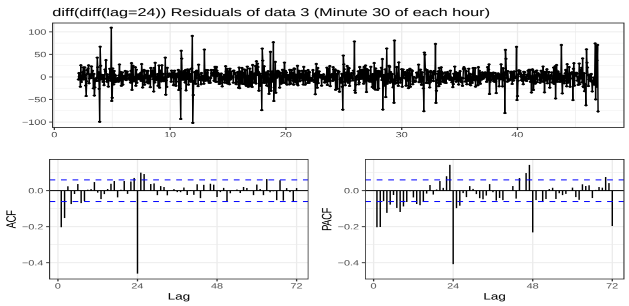


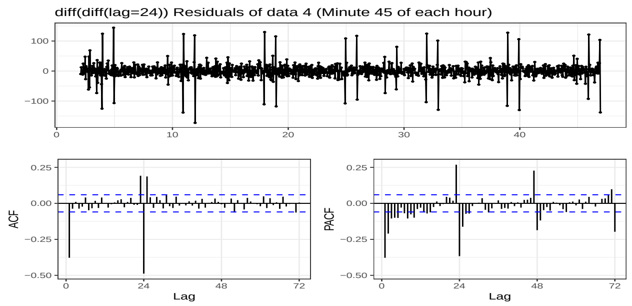
Differencing the residuals : lag(lag=24)

```
resd.diff1 = list()
for (i in 1:4){
  titleOfPlot = paste('diff(diff(lag=24)) Residuals of data ', i, ' (Minute ', 15*(i-1),
                      ' of each hour)' , sep='')
  resd.diff1[[i]] = diff(resd.diff0[[i]])
  ggtsdisplay(resd.diff1[[i]], main = titleOfPlot)
}
```









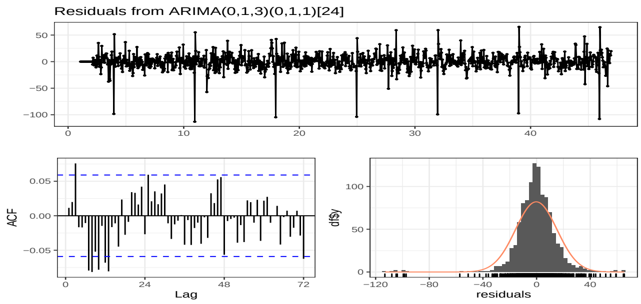
From these graphs, we can see:
- An exponential decrease on the "pacf" graph for all data.
- A significant ACF at lag = 24 for all data suggesting a seasonal MA_1.
- A significant ACF at lag = 1, lag = 3, lag = 2, lag = 1 for data 1, 2, 3, 4 respectively, suggesting a non-seasonal MA_3.

$\Rightarrow$ We propose the model $SARIMA_{(0,1,3)(0,1,1)[24]}$ for the residuals.

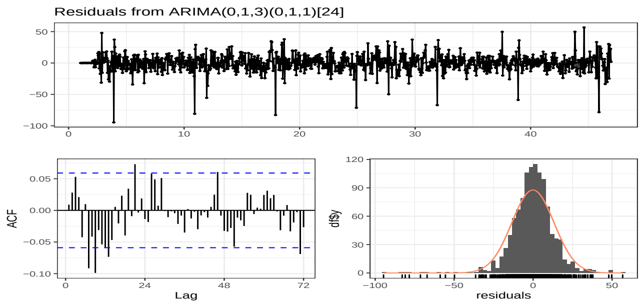
$SARIMA_{(0,1,3)(0,1,1)[24]}$ for the residuals

```
fit4.arima1.resd = list()
for (i in 1:4){
  # Fit with SARIMA(0,1,3)(0,1,1)[24] on the residuals
  fit4.arima1.resd[[i]] = Arima(fit4.resd[[i]], order = c(0,1,3), seasonal = c(0,1,1))
  # Check residuals
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.arima1.resd[[i]])
}
```

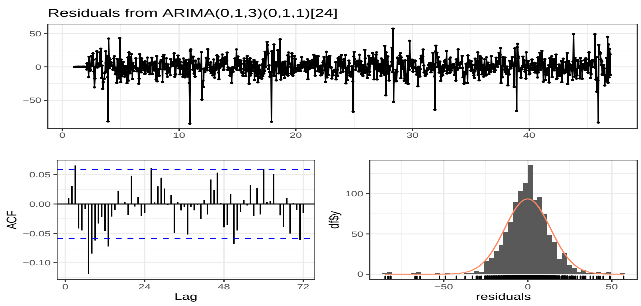
```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 85.642, df = 44, p-value = 0.0001722
##
## Model df: 4.   Total lags used: 48
```



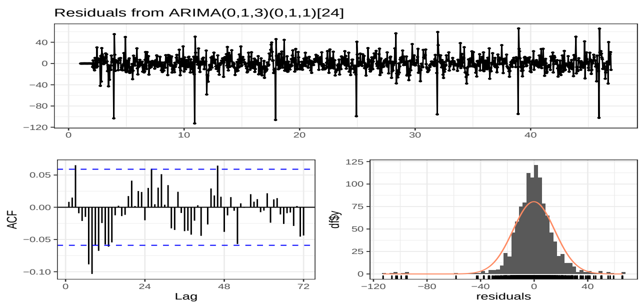

```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 76.213, df = 44, p-value = 0.001846
##
## Model df: 4.   Total lags used: 48
```




```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 77.2, df = 44, p-value = 0.001458
##
## Model df: 4.   Total lags used: 48
```




```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[24]
## Q* = 82.82, df = 44, p-value = 0.0003595
##
## Model df: 4.   Total lags used: 48
```



For all data, we have p-value < 0.05, and we can see after lag $= 3$ there is a significant ACF at lag $= 7$.

$\Rightarrow$ We propose the model $\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$ for the residuals.

$\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$ for the residuals

```
fit4.arima2.resd = list()
for (i in 1:4){
  # Fit with SARIMA(0,1,7)(0,1,1)[24] on the residuals
  fit4.arima2.resd[[i]] = Arima(fit4.resd[[i]], order = c(0,1,7), seasonal = c(0,1,1))
  # Check residuals
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.arima2.resd[[i]], plot = F)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 45.695, df = 40, p-value = 0.2474
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 42.985, df = 40, p-value = 0.3446
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 32.239, df = 40, p-value = 0.8038
##
## Model df: 8.   Total lags used: 48
```

```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,7)(0,1,1)[24]
## Q* = 38.736, df = 40, p-value = 0.5271
##
## Model df: 8.   Total lags used: 48
```

$\Rightarrow$ We have p-value > 0.05 for all data.
$\Rightarrow$ We will use the model $\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$ to forecast data with covariates.

$\text{SARIMA}_{(0,1,7)(0,1,1)[24]}$ for data with covariates

```
fit4.arima11.cov = list(); prev4.arima11.cov = list(); lxts_prev4.arima11.cov = list()
fit4.arima12.cov = list(); prev4.arima12.cov = list(); lxts_prev4.arima12.cov = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  fit4.arima11.cov[[i]] = Arima(lts_Elec.train[[i]][, 'Power'], order = c(0,1,7),
                                seasonal = c(0,1,1), xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.arima11.cov[[i]] = forecast(fit4.arima11.cov[[i]], xreg = lts_Elec.test[[i]][, 'Temp'],
                                    h = 24)
  lxts_prev4.arima11.cov[[i]] = xts(prev4.arima11.cov[[i]]$mean, order.by = time.Elec.test[[i]])
  # SARIMA(0,1,7)(0,1,1)[24] + Box-Cox
  fit4.arima12.cov[[i]] = Arima(lts_Elec.train[[i]][, 'Power'], lambda = 'auto', order = c(0,1,7),
                                seasonal = c(0,1,1), xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.arima12.cov[[i]] = forecast(fit4.arima12.cov[[i]], xreg = lts_Elec.test[[i]][, 'Temp'],
                                    h = 24)
  lxts_prev4.arima12.cov[[i]] = xts(prev4.arima12.cov[[i]]$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.arima11.cov = do.call(rbind, lxts_prev4.arima11.cov)
xts_prev4.arima12.cov = do.call(rbind, lxts_prev4.arima12.cov)
# Compute the RMSE
rmse4.arima11.cov = RMSE.val(xts_prev4.arima11.cov, xts_Elec.test[, 'Power'])
rmse4.arima12.cov = RMSE.val(xts_prev4.arima12.cov, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of SARIMA(0,1,7)(0,1,1)[24] using covariates =', rmse4.arima11.cov, '\n')
```

```
## RMSE of SARIMA(0,1,7)(0,1,1)[24] using covariates = 14.5565
```

```
cat ('RMSE of SARIMA(0,1,7)(0,1,1)[24] with Box-Cox using covariates =', rmse4.arima12.cov, '\n')
```

```
## RMSE of SARIMA(0,1,7)(0,1,1)[24] with Box-Cox using covariates = 14.55678
```

```
for (i in 1:4){
  # Check residuals
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit4.arima11.cov[[i]], plot = F)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 47.911, df = 39, p-value = 0.155
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 44.983, df = 39, p-value = 0.2357
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 44.552, df = 39, p-value = 0.2496
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 40.756, df = 39, p-value = 0.3931
##
## Model df: 9.   Total lags used: 48
```

$\Rightarrow$ For all data, the p-value > 0.05. This model is correct and we can use it.

4. Auto ARIMA with Covariates

```
fit4.autoArima1.cov = list(); prev4.autoArima1.cov = list(); lxts_prev4.autoArima1.cov = list()
fit4.autoArima2.cov = list(); prev4.autoArima2.cov = list(); lxts_prev4.autoArima2.cov = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  fit4.autoArima1.cov[[i]] = auto.arima(lts_Elec.train[[i]][, 'Power'],
                                         xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.autoArima1.cov[[i]] = forecast(fit4.autoArima1.cov[[i]],
                                        xreg = lts_Elec.test[[i]][, 'Temp'], h = 24)
  lxts_prev4.autoArima1.cov[[i]] = xts(prev4.autoArima1.cov[[i]]$mean,
                                        order.by = time.Elec.test[[i]])
  # SARIMA(0,1,7)(0,1,1)[24] + Box-Cox
  fit4.autoArima2.cov[[i]] = auto.arima(lts_Elec.train[[i]][, 'Power'], lambda = 'auto',
                                         xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.autoArima2.cov[[i]] = forecast(fit4.autoArima2.cov[[i]],
                                        xreg = lts_Elec.test[[i]][, 'Temp'], h = 24)
  lxts_prev4.autoArima2.cov[[i]] = xts(prev4.autoArima2.cov[[i]]$mean,
                                        order.by = time.Elec.test[[i]])
```

```
}
# Concatenate the 4 parts of forecasted data
xts_prev4.autoArima1.cov = do.call(rbind, lxts_prev4.autoArima1.cov)
xts_prev4.autoArima2.cov = do.call(rbind, lxts_prev4.autoArima2.cov)
# Compute the RMSE
rmse4.autoArima1.cov = RMSE.val(xts_prev4.autoArima1.cov, xts_Elec.test[, 'Power'])
rmse4.autoArima2.cov = RMSE.val(xts_prev4.autoArima2.cov, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of Auto ARIMA using covariates =', rmse4.autoArima1.cov, '\n')
```

```
## RMSE of Auto ARIMA using covariates = 17.85662
```

```
cat ('RMSE of Auto ARIMA with Box-Cox using covariates =', rmse4.autoArima2.cov, '\n')
```

```
## RMSE of Auto ARIMA with Box-Cox using covariates = 16.12468
```

5. NNET with Covariates

```
fit4.nnet1.cov = list(); prev4.nnet1.cov = list(); lxts_prev4.nnet1.cov = list()
fit4.nnet2.cov = list(); prev4.nnet2.cov = list(); lxts_prev4.nnet2.cov = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24]
  fit4.nnet1.cov[[i]] = nnetar(lts_Elec.train[[i]][, 'Power'],
                               xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.nnet1.cov[[i]] =  forecast(fit4.nnet1.cov[[i]],
                                   xreg = lts_Elec.test[[i]][, 'Temp'], h = 24)
  lxts_prev4.nnet1.cov[[i]] = xts(prev4.nnet1.cov[[i]]$mean,
                                  order.by = time.Elec.test[[i]])
  # SARIMA(0,1,7)(0,1,1)[24] + Box-Cox
  fit4.nnet2.cov[[i]] = nnetar(lts_Elec.train[[i]][, 'Power'], lambda = 'auto',
                               xreg = lts_Elec.train[[i]][, 'Temp'])
  prev4.nnet2.cov[[i]] = forecast(fit4.nnet2.cov[[i]],
                                  xreg = lts_Elec.test[[i]][, 'Temp'], h = 24)
  lxts_prev4.nnet2.cov[[i]] = xts(prev4.nnet2.cov[[i]]$mean,
                                  order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.nnet1.cov = do.call(rbind, lxts_prev4.nnet1.cov)
xts_prev4.nnet2.cov = do.call(rbind, lxts_prev4.nnet2.cov)
# Compute the RMSE
rmse4.nnet1.cov = RMSE.val(xts_prev4.nnet1.cov, xts_Elec.test[, 'Power'])
rmse4.nnet2.cov = RMSE.val(xts_prev4.nnet2.cov, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of NNET using covariates =', rmse4.nnet1.cov, '\n')
```

```
## RMSE of NNET using covariates = 16.95093
```

```
cat ('RMSE of NNET with Box-Cox using covariates =', rmse4.nnet2.cov, '\n')
```

```
## RMSE of NNET with Box-Cox using covariates = 19.13964
```

6. Vectoriel Auto-Regressive models

Select the best VAR$_p$ model

```
for (i in 1:4){
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  print(VARselect(lts_Elec.train[[i]], lag.max = 7, type = "both", season = 24))
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      1      2
##
## $criteria
##                1         2         3         4         5         6         7
## AIC(n)  4.501248  4.488496  4.489289  4.493797  4.498039  4.503271  4.507242
## HQ(n)   4.594522  4.588679  4.596382  4.607798  4.618950  4.631092  4.641971
## SC(n)   4.747749  4.753257  4.772310  4.795077  4.817579  4.841070  4.863300
## FPE(n) 90.131313 88.989708 89.060815 89.463716 89.844733 90.316731 90.676870
##
## * --- Data 2 (Minute 15 of each hour) --- *
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                1         2         3         4         5         6         7
## AIC(n)  4.245767  4.249480  4.247368  4.249425  4.254292  4.260599  4.257696
## HQ(n)   4.338969  4.349586  4.354377  4.363338  4.375110  4.388320  4.392322
## SC(n)   4.492089  4.514048  4.530182  4.550485  4.573598  4.598151  4.613495
## FPE(n) 69.810667 70.070708 69.923230 70.067647 70.410032 70.856019 70.651301
##
## * --- Data 3 (Minute 30 of each hour) --- *
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      1      1      1      1
##
## $criteria
##                1         2         3         4         5         6         7
## AIC(n)  4.272868  4.275453  4.278853  4.283470  4.287274  4.293254  4.291419
## HQ(n)   4.366070  4.375559  4.385862  4.397383  4.408091  4.420975  4.426044
## SC(n)   4.519190  4.540021  4.561667  4.584530  4.606580  4.630806  4.647217
## FPE(n) 71.728494 71.914517 72.159787 72.494162 72.770963 73.208045 73.074456
##
## * --- Data 4 (Minute 45 of each hour) --- *
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      3      2      1      3
##
## $criteria
##                1         2         3         4         5         6         7
## AIC(n)  4.499497  4.486951  4.486066  4.489825  4.494335  4.500443  4.500018
## HQ(n)   4.592699  4.587056  4.593076  4.603739  4.615152  4.628164  4.634643
## SC(n)   4.745819  4.751519  4.768880  4.790886  4.813641  4.837995  4.855816
## FPE(n) 89.973639 88.852291 88.774249 89.109133 89.512484 90.061636 90.024180
```

The “AIC” selected is equal to 2, 1, 1, and 3 for data 1, 2, 3 and 4, respectively.
⇒ We will start by choosing “$p = 2$” for the VAR_p model.

Estimate and check residuals of a VAR_2 model

```
var = list()
for (i in 1:4){
  # Estimation of a VAR2
  var[[i]] = VAR(lts_Elec.train[[i]], p = 2, type = "both", season = 24, exogen = NULL)
  # Check residuals
```

```
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  print(serial.test(var[[i]], lags.pt = 10, type = "PT.asymptotic"))
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
## Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var[[i]]
## Chi-squared = 36.623, df = 32, p-value = 0.2628
```

```
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var[[i]]
## Chi-squared = 44.525, df = 32, p-value = 0.06954
```

```
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var[[i]]
## Chi-squared = 44.158, df = 32, p-value = 0.07462
```

```
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var[[i]]
## Chi-squared = 35.578, df = 32, p-value = 0.3035
```

$\Rightarrow$ For all data, the p-value> 0.05, we can choose this model for fprecasting.

Forecast usin the VAR_2 model

```
prev4.var = list(); lxts_prev4.var = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  prev4.var[[i]] = forecast(var[[i]], h = 24)
  lxts_prev4.var[[i]] = xts(prev4.var[[i]]$forecast$Power$mean, order.by = time.Elec.test[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev4.var = do.call(rbind, lxts_prev4.var)
# Compute the RMSE
rmse4.var = RMSE.val(xts_prev4.var, xts_Elec.test[, 'Power'])
# Print the RMSE
cat ('RMSE of VAR =', rmse4.var, '\n')
```

```
## RMSE of VAR = 16.85849
```

7. Choose the model

Model summary

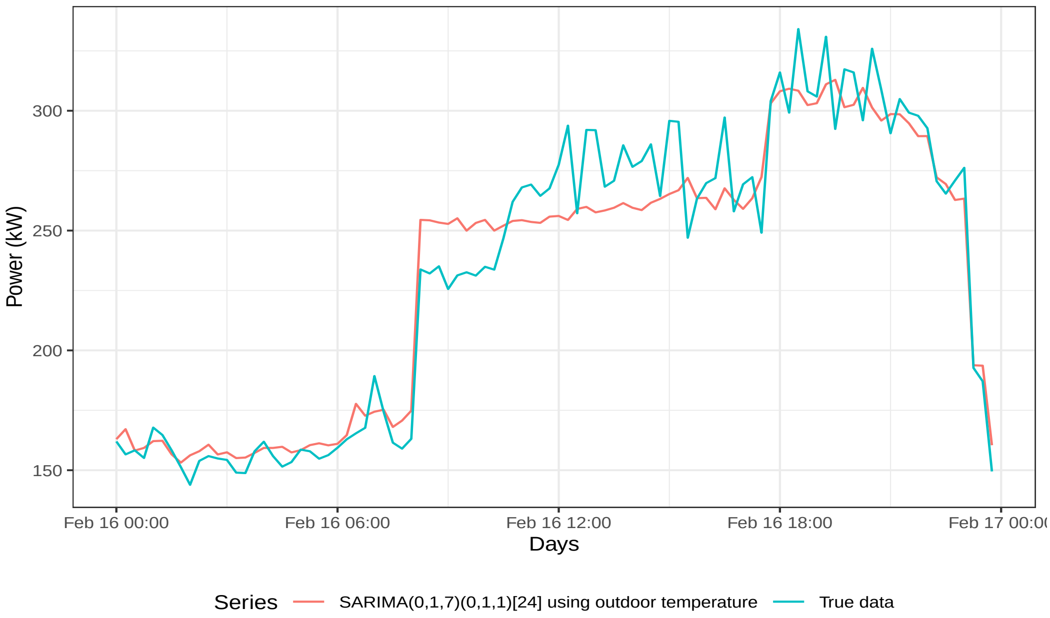
```
df = data.frame(RMSE = c(rmse4.arima11.cov, rmse4.autoArima2.cov, rmse4.nnet1.cov, rmse4.var))
rownames(df) = c("SARIMA(0,1,7)(0,1,1)[24] using temperature",
                 "Auto ARIMA + Box-Cox using temperature","NNET using temperature",
                 "Vectoriel Auto-Regressive models")
print(df)
```

```
##                                                RMSE
## SARIMA(0,1,7)(0,1,1)[24] using temperature 14.55650
## Auto ARIMA + Box-Cox using temperature     16.12468
## NNET using temperature                     16.95093
## Vectoriel Auto-Regressive models           16.85849
```

$\Rightarrow$ The best model is $SARIMA_{(0,1,7)(0,1,1)[24]}$, we will apply it for prediction using all data.

Plot

```
xts_prevAndTest = cbind(xts_prev4.arima11.cov, xts_Elec.test[, 'Power'])
names(xts_prevAndTest) = c("SARIMA(0,1,7)(0,1,1)[24] using outdoor temperature", "True data")
autoplot(xts_prevAndTest, facets = NULL) + labs(x = 'Days', y = 'Power (kW)') +
  theme(legend.position="bottom")
```



8. Forecast using all data

Forecast

```
fit.cov.all = list(); prev.cov.all = list(); lxts_prev.cov.all = list()
# Forecast and convert the results to xts objects.
for (i in 1:4){
  # SARIMA(0,1,7)(0,1,1)[24] using covariates
  fit.cov.all[[i]] = Arima(lts_Elec.fit[[i]][, 'Power'], xreg = lts_Elec.fit[[i]][, 'Temp'],
                           order = c(0,1,7), seasonal = c(0,1,1))
  prev.cov.all[[i]] = forecast(fit.cov.all[[i]], xreg = lts_Elec.prev[[i]][, 'Temp'], h = 24)
  lxts_prev.cov.all[[i]] = xts(prev.cov.all[[i]]$mean, order.by = time.Elec.prev[[i]])
}
# Concatenate the 4 parts of forecasted data
xts_prev.cov.all = do.call(rbind, lxts_prev.cov.all)
```

Check residuals for the selected model

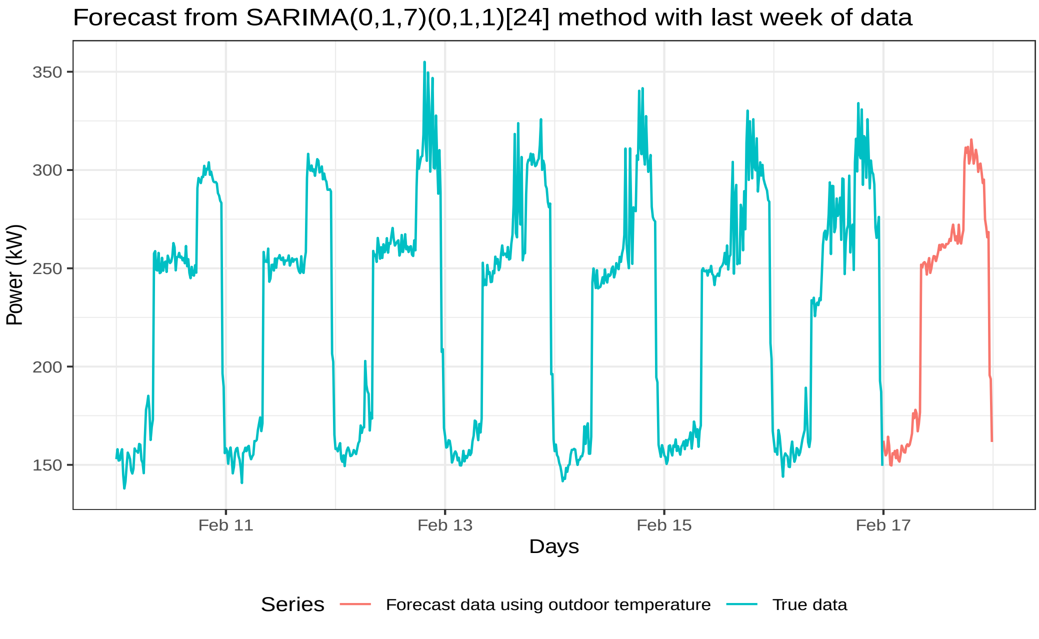
```
for (i in 1:4){
  cat ('* --- Data', i, '(Minute', 15*(i-1), 'of each hour) --- *\n')
  checkresiduals(fit.cov.all[[i]], plot = F)
}
```

```
## * --- Data 1 (Minute 0 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 53.888, df = 39, p-value = 0.05676
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 2 (Minute 15 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 48.54, df = 39, p-value = 0.1407
```

```
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 3 (Minute 30 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 49.336, df = 39, p-value = 0.1242
##
## Model df: 9.   Total lags used: 48
##
## * --- Data 4 (Minute 45 of each hour) --- *
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,7)(0,1,1)[24] errors
## Q* = 44.96, df = 39, p-value = 0.2364
##
## Model df: 9.   Total lags used: 48
```

Plot the forecasted data

```
xts_prevAndTest = cbind(xts_prev.cov.all, xts_Elec.fit['2010-02-10/2010-02-16', 'Power'])
names(xts_prevAndTest) = c("Forecast data using outdoor temperature", "True data")
autoplot(xts_prevAndTest, facets = NULL) +
  labs(x = 'Days', y = 'Power (kW)') +
  ggtitle("Forecast from SARIMA(0,1,7)(0,1,1)[24] method with last week of data") +
  theme(legend.position="bottom")
```



Save forecasted data with covariates

```
wb = loadWorkbook('MohamedAbid.xlsx')
writeData(wb, 1, xts_prev.cov.all, startCol = 2, startRow = 1, colNames = F)
saveWorkbook(wb, 'MohamedAbid.xlsx', overwrite = T)
```