TIME SERIES ANALYSIS

Course poject

Joanne ADAM
Data ScienceTech Institut
joanne.adam@edu.dsti.institute

Dataset

We consider the dataset from the file `Elec_30_11_train.xlsx` which contains measurements of electricity consumption (in kW) and the Outdoor Air Temperature (OAT, in degree Fahrenheit) for one building. Temperature measurements in degrees Celsius are also presented in the dataset but as they are linearly related to the measurements in degrees Fahrenheit, these measurements are of no interest. Measurements are done every 15 minutes from the 1^{st} of January 2010 at 1:15am until the 29^{th} of November 2010 at 23:45pm which correspond to a total of 32059 measurements. There is no missing data in the dataset. The aim of this study is to predict the electricity consumption for the day of the 30^{st} of November.

Exploratory Data Analysis

Outliers

Figue 1 (left) presents the datasets and show several unusual down spikes that could be considered as outliers (*e.g.* with an electricity consumption lower than 120 kW). These values are remplaced by the average value of electricity consumption (see figure 1 right and R code below). Some statistics of the dataset are presented in table 1.

```
meani = mean(as.numeric(unlist(data[1:nrow(data),"Power (kW)"])), na.rm=TRUE)
for (index in which(z[,"Power (kW)"]<120))
  {data[index,"Power (kW)"] = meani}
```

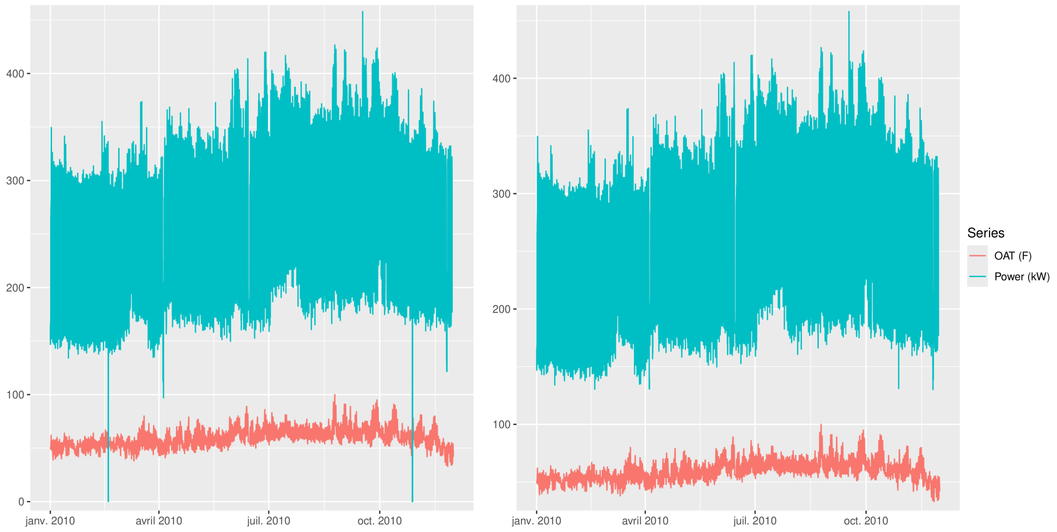

Figure 1: Original dataset (left) and dataset with removed outliers (right).

	Minimum	Maximum	Mean	Median	Variance
Power (kW)	0.0	457.9	262.3	276.5	66.0
OAT (F)	33.0	100.0	59.0	58.0	8.8

Table 1: Basic statistics of the dataset.

Trend and seasonality

Using moving averages, the `decompose` R function returns the trend and seasonal components of the time series. Figure 2 presents the components of the electricty power time series.

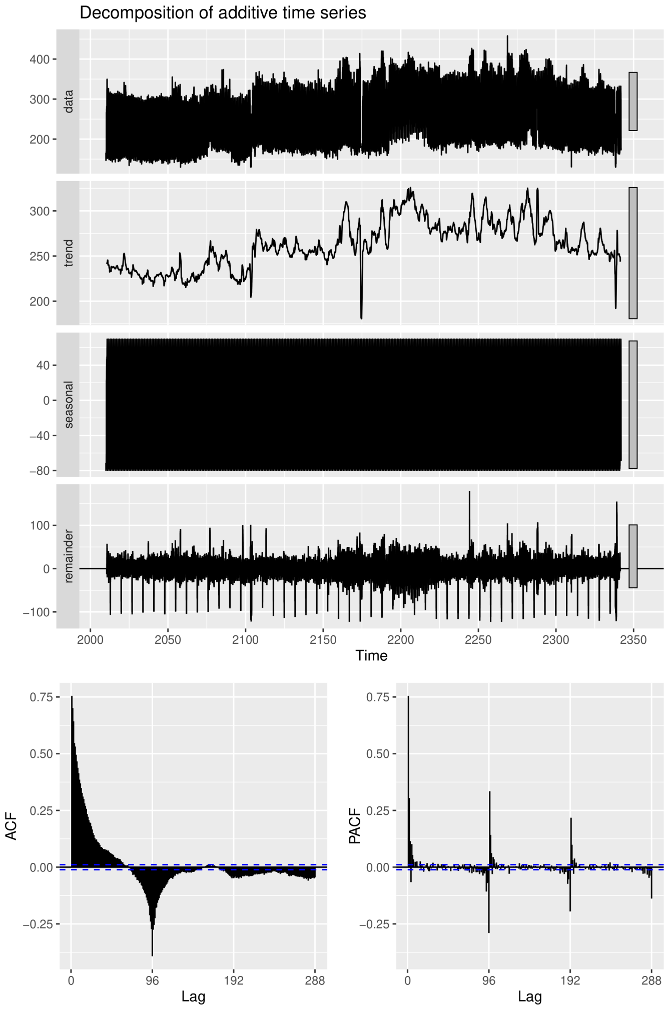

Figure 2: Trend and seasonal components of the electricity consumption time series.

The decomposition shows linear trend that varies in time (figure 2 TREND and ACF frames) and a significant seasonality of 96 lag (figure 2, PACF frame) that corresponds to one day.

Noise

The remainder of the decomposition of the electricity power time series (figure 2 REMAINDER frame) shows periodic fluctuations with relatively large amplitudes. These patterns seems to imply multi seasonality.

Covariate

The cross-correlation test (see R code below) that test the zero-correlation between the electricity power time series and the covariate Outdoor Air Temperature show p-values always null. We reject the null hypothesis, there is no zero correlation. In other words, there is high correlation between the two variables. Because of this high correlation, the covariate data might no convey additional information. Using covariates for modelling and forecasting might not be significant. In this analysis modelling and forecasting is done with and without the covariate for comparison.

```
cc.test(data[,"OAT (F)"], data[,"Power (kW)"], max.lag=10, plot=FALSE)
```

Modelling

In order to model and forecast the electricity comsumption, several methods have been applied. Due to the high frequency of seasonality of the dataset, Holt-Winters method (exponential smoothing) did not provide suitable results. Similarly, Random Forest algorithm modelling did show very high computational time. Models using Random Forest algorithm were not investigated further and will not be discussed in this report. R-code for this algothm and everthing presented in this report are available in the attached R script files.

This report will present models and forecasts using SARIMA, Neural Network and XGBoost methods computed with or without covariate dataset.

Dataset splitting

The dataset is split into a training and a testing set. The last period of Electricity power is taken as the testing test while the rest of the dataset is the training set. R-software code is presented in listing 1 below and datasets are presented in figure 3.

```
# Split in train/test set
v <- ts(data[,1:3], start=c(2010,1), frequency=96)
v_start <- start(v)
v_end   <- end(v)

# remove 2 last periods for training set:
v_train <- window(v, start=v_start , end=c(v_end[1],v_end[2]-2*96))
# start after 2 to last period for testing set:
v_test  <- window(v, start=c(v_end[1], v_end[2]-2*96+1), end=v_end)
```

Listing 1: Dataset splitting into train/test sets

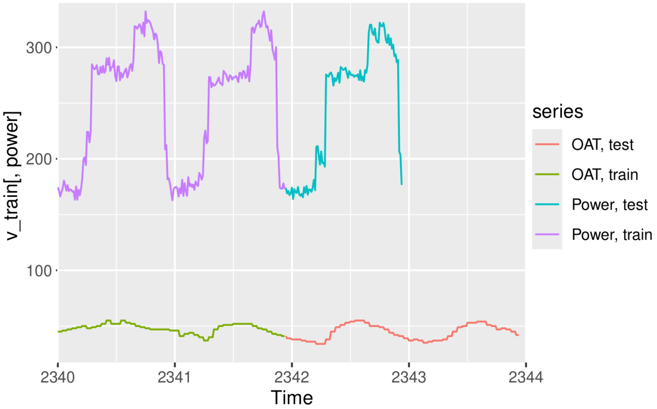

Figure 3: Training and testing sets of the variables Power and OAT (Outer Air Temperature).

SARIMA

Figure 2 shows high seasonality in the dataset therefore ARIMA methodology would not be appropriate for this study. Seasonality ARIMA (SARIMA) method is applied. Due to the relatively long computation time (more than one hour per computation), grid-seach is not implemented to search for the best parameters. See listing 2 for R-code. Figures 4 and 5 presents the best SARIMA model with prediction for fitting with and without considering the OAT covariate.

Table 2 shows few statistics of the models and show little improvement in the fitting when considering OAT covariates.

	With OAT covariate	Without OAT covariate
RMSE	8.31	8.60
p-value	$2.2e^{-16}$	$2.2e^{-16}$
Computation time	1h10	56 min

Table 2: Statistics of SARIMA modelling.

Noise residuals of this model (figure 4, p-values in table 2) still show significant signals that are not taken into account with this model. Increasing the values of the parameters seems to be necessary however, updating the parameters led to computational failure.

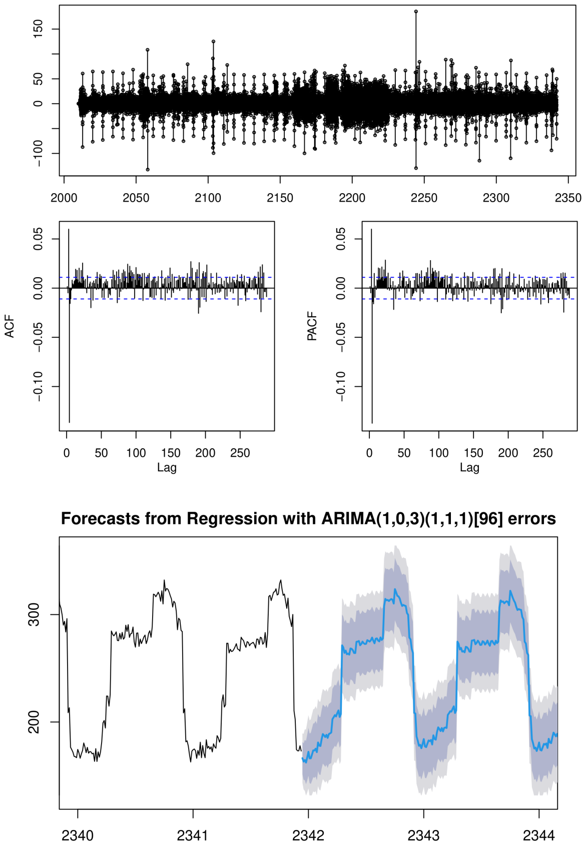

Figure 4: Modelling and forecasting using SARIMA algorithm with (1,0,3)(1,1,1)[96] parameters and OAT covariates.

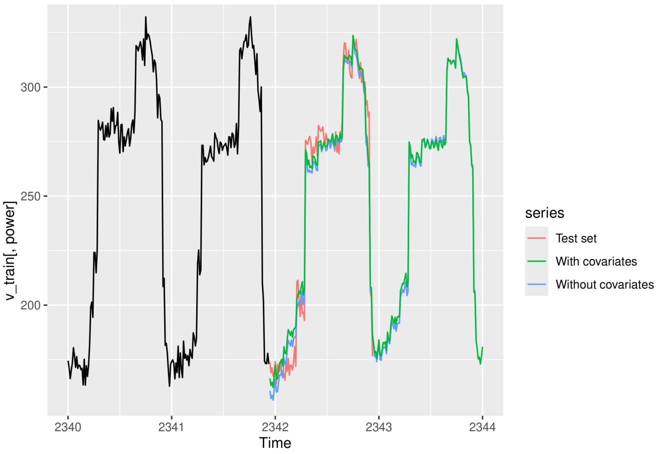

Figure 5: Testing set and forecast of the Electricity consumption using SARIMA algorith with (1,0,3)(1,1,1)[96] parameters with (green line) and without OAT covariate (blue line).

```r
if ( covariates )
  {
  start.time <- Sys.time()
  fit = Arima(v_train[,power], xreg=v_train[,oat], order=c(1,0,3), seasonal=c(1,1,1))
  end.time <- Sys.time()
  cat("timing :", (end.time-start.time), "\n")

  checkresiduals(fit)
  tsdisplay(fit$residuals)

  fc <- forecast(fit, xreg=v_train[,oat], h=192)

  rmse = calc_rmse(v_test[,power], fc$mean)
  cat("RMSE SARIMA (with covariates) :", rmse, "\n")
  }
else
  {
  start.time <- Sys.time()
  fit <- Arima(v_train[,power], order=c(1,0,3), seasonal=c(1,1,1))
  end.time <- Sys.time()
  cat("timing :", (end.time-start.time), "\n")

  checkresiduals(fit)
  tsdisplay(fit$residuals)

  fc <- forecast(fit, h=192)

  rmse = calc_rmse(v_test[,power], fc$mean)
  cat("RMSE SARIMA (without covariates) :", rmse, "\n")
```

Listing 2: R code for SARIMA modelling and forecasting

Neural-Network

Grid-search analysis is performed to search for the best parameters *i.e.* the set of parameters that minimize the root-mean-square-error between the data and the model (see attached R script files for complete code). Results show that parameters (7,3,6) are best-suited for Neural Network modelling of the Electricity consumption and parameters (4,2,4) are best-suited when considering outer-air temperature data as a covariate. Modeling and forecasting using these parameters are presented in figures 6 and 7. Listing 3 presents the piece of R code used to model and forecast the electricity consumption.

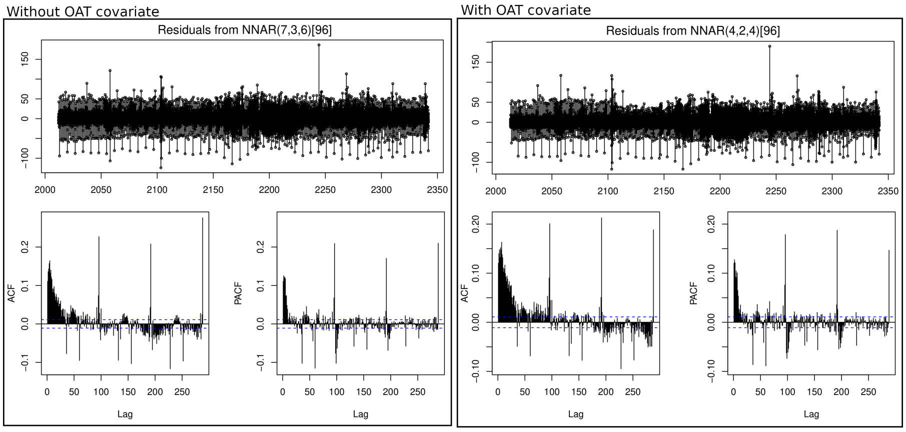

Figure 6: Modelling using Neural Network algorithm with (7,3,6)[96] parameters when not considering OAT covariates and (4,2,4)[96] when considering OAT covariates.

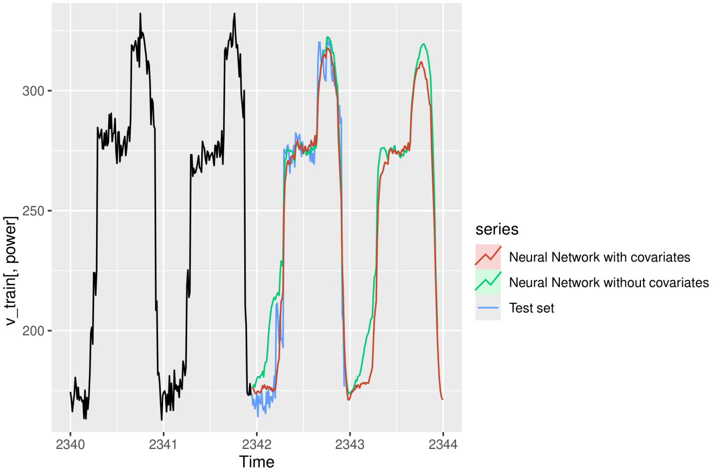

Figure 7: Testing set (blue line) and forecast of the electricity consumption using Neural Network algorith with (red line) and without OAT covariate (green line).

```
if ( covariates )
  {
  start.time <- Sys.time()
  fit  = nnetar(v_train[,power], xreg=v_train[,oat], p=4, P=2)
  end.time <- Sys.time()
  cat("timing :", (end.time-start.time), "\n")

  checkresiduals(fit)
  tsdisplay(fit$residuals)

  prev = forecast(fit, xreg=v_train[,oat], h=192)

  rmse = calc_rmse(prev$mean, v_test[,power])
  cat("RMSE NN (with covariates) :", rmse, "\n")
  }
else
  {
  start.time <- Sys.time()
  fit  = nnetar(v_train[,power], p=10, P=3)
  end.time <- Sys.time()
  cat("timing :", (end.time-start.time), "\n")

  checkresiduals(fit)
  tsdisplay(fit$residuals)

  prev = forecast(fit, h=192)

  rmse = calc_rmse(prev$mean, v_test[,power])
  cat("RMSE NN (without covariates) :", rmse, "\n")
```

Listing 3: R code for Neural Network method modelling and forecasting

Residuals, auto-correlation and partial auto-correlation functions (figure 6) and values of residuals (table 3) show that these models are not able to fully explain the data. We do observe seasons in the cross-correlation functions that differ from the main season. These two additional seasons have been identified with lags of 36 and 60 which corresponds to 9 and 15 hours respectively. Multi-season modelling (*e.g.* using arima/fourier modelling, see attached R-scripts for examples) could help doing modellisation and forecasting for this dataset with several seasons.

	With OAT covariate	Without OAT covariate
RMSE	9.92	10.38
p-value	2.2e^{-16}	2.2e^{-16}
Computation time	1h00	57 min

Table 3: Statistics (RMSE: Root-Mean-Square Error and p-value) and computational time of Neural Network method modelling with and without considering Outer Air Temperature (OAT) as covariate.

XGBoost

Modelling and forecasting using XGBoost algorithm without considering OAT covariate are presented in figure 8. See listing 4 for R code. XGBoost algorithm show good fitting model with a RMSE of 7.90 and a computation time of 6 minutes.

```r
start.time <- Sys.time()
data = as.vector(v_train[,power])[1:97]
# Creating the dataset for XGBoost algorithm:
for (i in 1:(length(as.vector(v_train[,power]))-97) )
  {
  data = rbind(data, as.vector(v_train[,power])[(i+1):(i+97)])
  }

# Compute the model:
model <- xgboost(data=data[,1:96], label=data[,97], max_depth=10, eta=0.1,
  nrounds=100, nthread=4, objective="reg:squarederror", verbose=FALSE)
end.time <- Sys.time()
cat("timing :", (end.time-start.time), "\n")

# Make forecast over two seasons:
pred    = rep(NULL, 192)
newdata = tail(v_train[,power],96)
for (t in 1:192)
  {
  pred[t] = predict(model, matrix(newdata,1,96))
  newdata = c(newdata[-1],pred[t])
  }

# Convert the forecast into a time series:
ts_xgboost = ts(pred, start=start(v_test), end=end(v_test), frequency=96)

rmse = calc_rmse(ts_xgboost, v_test[,power])
cat("RMSE XGBoost (without covariates) :", rmse, "\n")
```

Listing 4: R code for XGBoost modelling and forecasting

Discussion and concluding remarks

Table 4 presents the root-mean-square-error and computation time for each of the models presented in this report. Results show that adding Outer-Air-Temperature covariate does not improve significantly the fit to the model.

Model computed using XGBoost methodology as the minimum residual error and the faster computation time. The forecast using this model is attached in a separate file, as the prefered model.

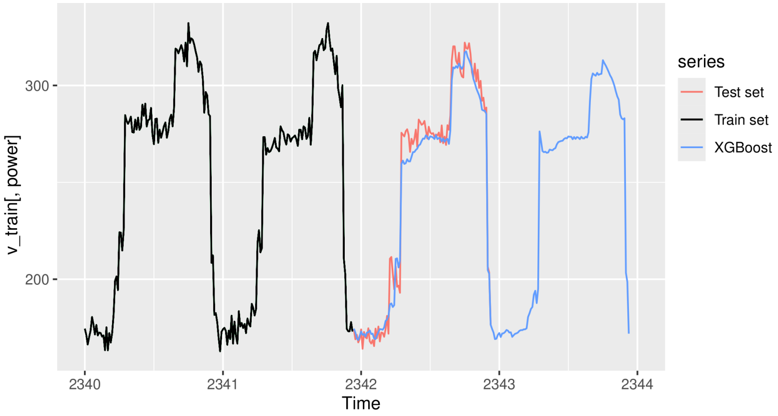

Figure 8: Modelling and forecast using XGBoost algorithm without considering the OAT covariate.

	RMSE	Computation time
SARIMA (without covariates)	8.60	56 min
SARIMA (with covariates)	8.31	1h16
Neural Network (without covariates)	14.93	1h10
Neural Network (with covariates)	10.29	40 min
XGBoost (without covariates)	7.90	6 min

Table 4: Root mean square error (RMSE) and computation time of modelling using different algorithms.

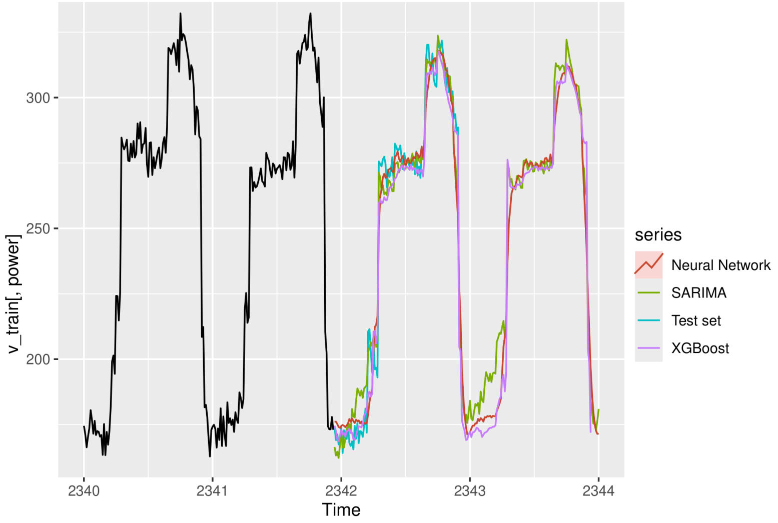

Figure 9: Modelling and forecast of electricity consumption using Outer-Air Temperature as a covariate.

All R codes are available on Github (https://github.com/JoanneAB/TimeSeriesForecasting).