Ecole Polytechnique

Study on multiple methods for forecast and backcast time series

Group 9 : Victor BOSSARD, Aubry d'ANDOQUE, Yelan MO, Liwenhan XIE

2019-12-17

1. Summary

In the project, our group tried on various methods to forecast a time series concerning energy loads and additional temperature information. These methods include traditional time series analysis, like MA, VAR, and more advanced techniques like NNAR, STL, ARIMA. Their strengths and weaknesses are discussed in detail. With early exploratory data analysis, we derived insights of the data pattern and this information guides our selection of features. We also explored how we can leverage PCA to impute the missing values, instead of the naive seasonal average.

2. Background

2.a Problem Formulation

The problem is from a Kaggle Challenge: Global Energy Forecasting Competition 2012 - Load Forecasting[1], where the task is to **backcast missing values and forecast future behaviors**. The dataset concerns the hourly energy loads (unit: kWatt) for a US utility with 20 unknown zones from the year 2004 to 2008. And there is additional information about the hourly temperature (unit: ℉) observed in 11 stations, whose relationship with the given zones remains unclear. In general, eight entire weeks in the load history is missing for every zone, including four in 2005 and four in 2006, one at each season.

Specifically, the missing periods of energy loads are[2]:

① 2005/03/06 0am - 2005/03/12 11pm;
② 2005/06/20 0am - 2005/06/26 11pm;
③ 2005/09/10 0am - 2005/09/16 11pm;
④ 2005/12/25 0am - 2005/12/31 11pm;
⑤ 2006/02/13 0am - 2006/02/19 11pm;
⑥ 2006/05/25 0am - 2006/05/31 11pm;
⑦ 2006/08/02 0am - 2006/08/08 11pm;
⑧ 2006/11/22 0am - 2006/11/28 11pm.

And the temperature of the prediction period is not given, i.e. 2008/06/30 5am - 2008/07/07 11pm.

As the missing values lie in the middle of the time series and can be predicted by their former periods, in this project, we regarded the backcast problem the same as the prediction problem and focused on investigating different predicting methods, which include:

1) Discuss the effectiveness of each method to the problem.
2) Compare different methods to provide the best result.

We mainly test on the aggregated data: zone 21, where the value is the sum of all 20 zones.

[1] https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting
[2] Date time format: YYYY/MM/DD HH a

2.b. Recap of Time Series Analysis

Before we go down into the problem, it is helpful to review some basic concepts introduced in time series analysis.

Normally, we are interested in the trend, cyclic pattern, seasonality in a time series. And time series analysis comprises methods to extract meaningful characteristics of data, such as statistics, where forecasting use suites of models to predict future values based on observed values. Many models are based on the assumption of stationarity, that the statistical properties, such as mean, standard deviation, coefficient of autocorrelation remain constant over time. When the data does not satisfy the requirement of stationarity, a common approach is to use a high-order difference to achieve stationarity. An important measure is autocorrelation, i.e. $R_{xx}(r) = E(X_t \underline{X_{t+r}})$. A similar concept is cross correlation, where the lagged correlation of two series is concerned.

2.c. Evaluation Metrics

As we plan on trying different methods, we should first derive a set of relevant metrics to compare each of our predictions on. What we will focus on is not especially the complexity of the algorithms or the time taken to run, even though it could be interesting in an industrial context. But we will focus on evaluating our forecast accuracy.

In the most general term, the forecasting error e_i at a given step t_i is the difference of the validation data y_i and the forecast f_i. But as we intend to forecast for a period of a week we need aggregated measures over that time period.

Mean Error $ME = mean(e_i)$
It is the simplest of all metrics as it only averages the errors over the whole forecasting horizon. The pros are that it is easily understood as the expected error at each step may it be positive or negative.

Mean Absolute Error $MAE = mean(|e_i|)$
This metric is as simple as ME but it corrects the effect of the sign. As ME we can emphasize the fact that all errors big or small are treated as equal.

Root Mean Square Error $RMSE = \sqrt{mean(e_i^2)}$
Also known as standard error, it is widely used. We shall note that the use of the square penalize big errors over smaller ones.

The major problem with the three metrics above is that the return a scale-dependent error. Which mean that the score is not easily translatable to time series with other units or with different order of magnitude. For instance, in our case we would obtain greater errors on the 21st aggregated zone just because of the fact that it is its value are one order of magnitude greater than other zones. That is the motivation to also look at Pourcentage and Scales errors below.

Mean Percentage Error & Mean Absolute Percentage Error

$$MPE = mean(\frac{100 \times e_i}{y_i}) \ \& \ MAPE = mean(\frac{100 \times |e_i|}{y_i})$$

They both tell how much the prediction differs from the validation value but in terms of percentage. This means it can translate for another zone or another dataset. They are the more universal metrics.

Mean Absolute Scaled Error

$$MASE = mean(|q_i|) \quad where \quad q_i = \frac{e_i}{MAE(Training\ Sample)}$$

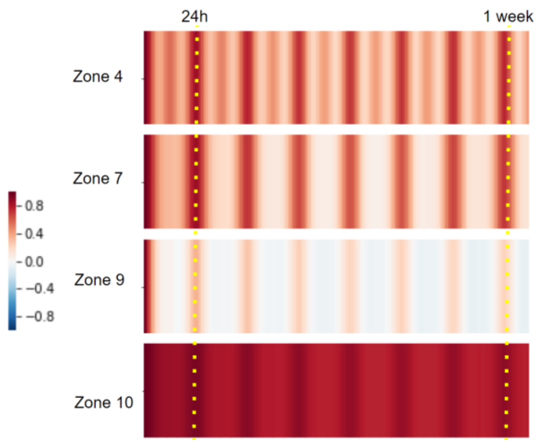
By means of scaling by the it results that this error is not scale dependant. Bit there is a need to define the method used on the training sample to derive the scale factor. In general it is a mean method but in case of seasonal data it could also be the seasonal naïve method described later. As a result, if the MASE is greater than one the forecast is worse than the naïve method used on the training set.

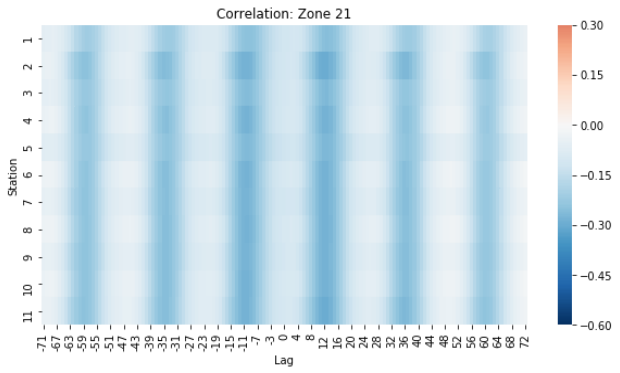
For each forecasting methods we will calculate this metrics. It should be a solid base to choose the most appropriate forecast in our situation.

3. Global Analysis

Relying heavily with data analysis, we obtained an overview of the dataset. Because it is reported that 99% of all color blind people suffer from red/green color blindness, we tried our best to make sure that our plot is friendly for the protanopia or deuteranopia by avoiding ambiguous color schemes.

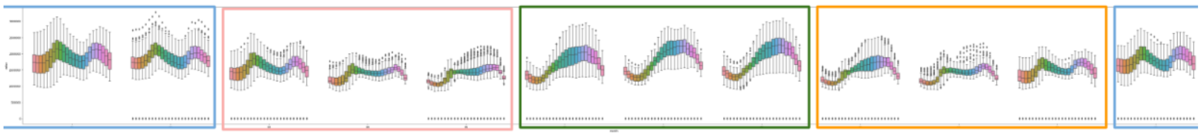
First of all, we visualized the raw time series value of the data. Due to the limited pixel density, it is impossible to see the hourly change of five years in a view. Therefore, we aggregated the data with the day level by sum. Though the range in different zones vary dramatically, the overall trends are highly similar, which seas a strong seasonality. However, zone 9 and zone 10 are abnormal, for the previous one is noisy and highly unstable, and the latter one has an abrupt increase in the last year. This is also reflected in the ACF result (left figure below). While other zones has significant high correlations to the same time of neighboring days, zone 9 has a rather low self correlation in any lags, and zone 10 remains highly correlated to neighboring days, regardless of time. Moreover, investigating into the ACF in years, we discovered that although generally the correlation is going down, one year lag slightly cross the confidence interval, which implies a seasonality in terms of year.





Left: Examples of regular patterns and abnormal autocorrelations .
Right: The correlation of 11 temperature stations and energy loads with lags.

Looking into the seasonality as the raw data suggests, we tried to bin data according to their month, year, hour, etc., enumerating possible granularities. And we observed strong seasonality in terms of hours and months. In particular, as the figure below shows the hourly distribution over different months in zone 21, we may have a better understanding of the data.



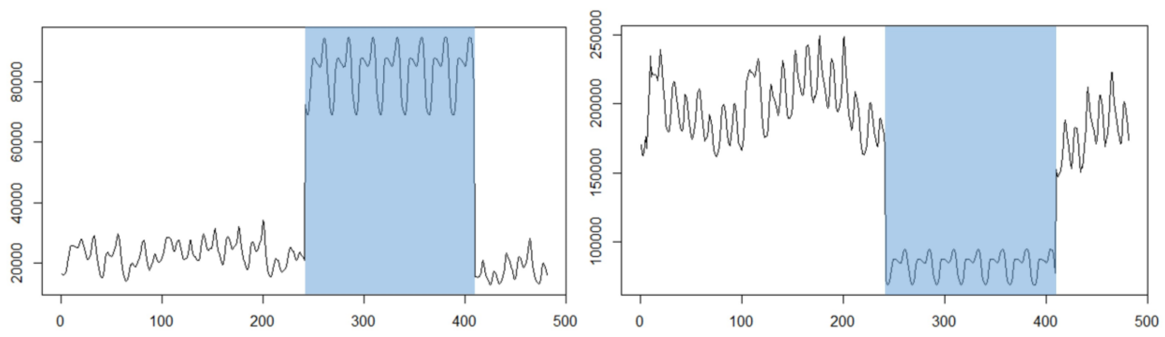
Hourly distribution of energy loads in each month (from Jan to Dec).

We also have a closer look at the weather data. From cross correlation, there exists certain links between the temperature and the energy loads. And the plot suggests that a 12 hour lag has the largest negative correlation. To some extent, the result is not intuitive, because normally we assume that current weather affects current energy consumption. However, we read the relevant paper[1] written by the Kaggle organizer and learned that such a phenomenon is actually the pattern of this data. As the value of all 11 stations are highly correlated, with correlations higher than 95% for each pair, we decided to use the average temperature as an additional attribute for the prediction.

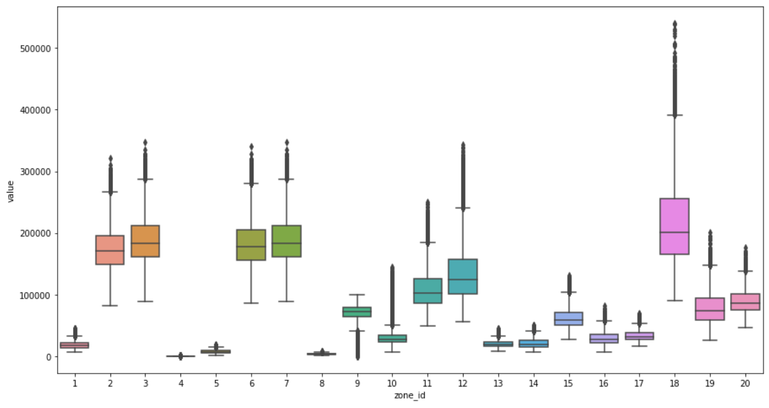
In summary, from a global perspective, the dataset has a strong seasonality in terms of season, month, and hour. And zone 9 and zone 10 are exceptions to be considered independently. Temperatures in different stations are highly correlated and have a 12-hour-lag correlation to the energy load.

4. Missing Values Imputation: PCA

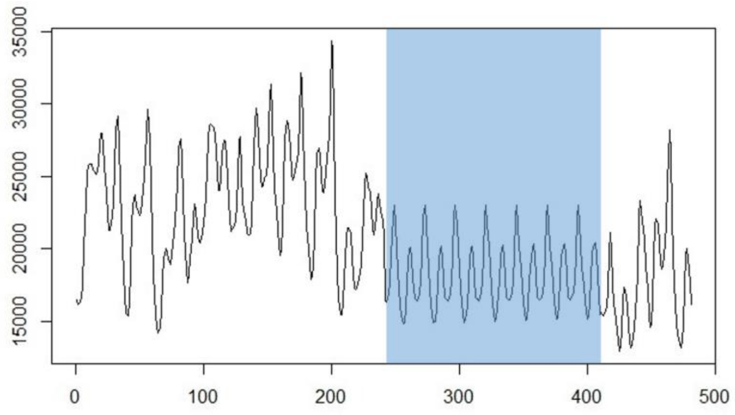
In the dataset offered by Kaggle, continuous data of several weeks are missing. We can use the generally imputation methods as well as the forecast and backcast methods to fulfill the dataset. In this section we will apply the classic imputation methods for missing values. Then in the following sections we will introduce methods for forecasting, apply then on missing values, compare and draw a conclusion.


Left: Imputation result for zone 1.Right: Imputation result for zone 2.
The blue part indicate the predicted values

Firstly we apply directly the iterative PCA algorithm on the raw dataset. Then we observe the result of the first missing part of zone 1 (left). The result is astonishing because it maintains the seasonality of the data. However the weird growth of the predicted data indicate the fallibility of the result. When we take a look at the result of the second zone (right), we can find out the reason of the “weird growth”.



According to the data distribution, the mean value of zone 2 is much higher than zone 1, the distribution of data for each zone varies significantly. We cannot impute directly the raw data while computing them together. We’ve separated the data by zones, then apply the iterative PCA algorithme again. The result of the first zone is quite compelling, for the daily seasonality underlying data is well captured.



5. Classical Methods for forecasting

5.a Simple Methods

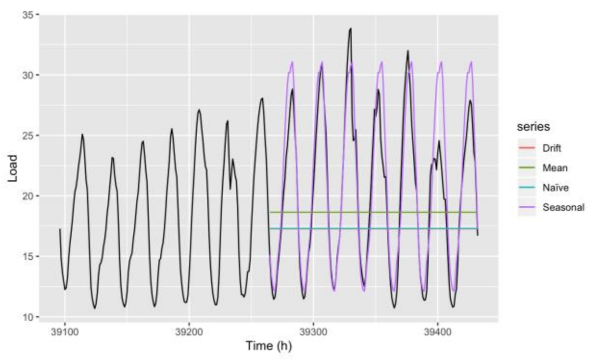
Before jumping into complex models we should try to implement simpler forecastings. First they are a way to familiarize with the data set but also the accuracy we will get will be our ground results to compare other methods to.

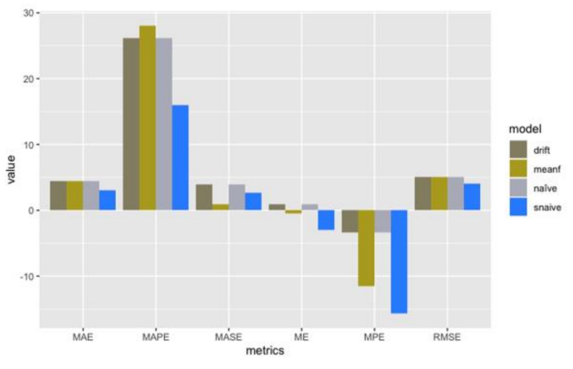
Naïve Method:
Using the last available data point as the forecast value for all the forecasting window.

Mean Method:
Simply using the mean of historical data as a forecast through out our forecasting windows.

Seasonal Naive Method
As we saw in our preliminary exploration of the data, the dataset has a high seasonality component. So it makes sense for use to use that seasonality to forecast. That is to say we use the last period as a whole and repeat it through our forecasting window. To predict a week ahead we would use the last available day seven times.

To those methods we could apply a drift which is simply to add the trend we had over the training data to the forecast to account for global trend.

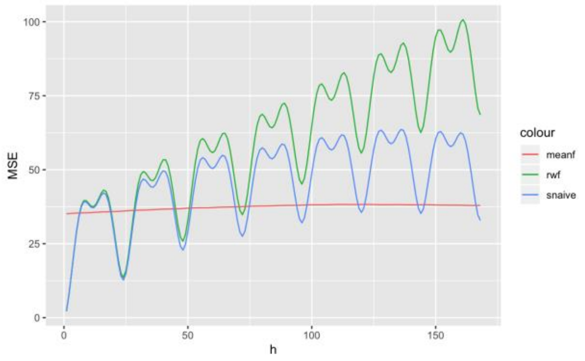


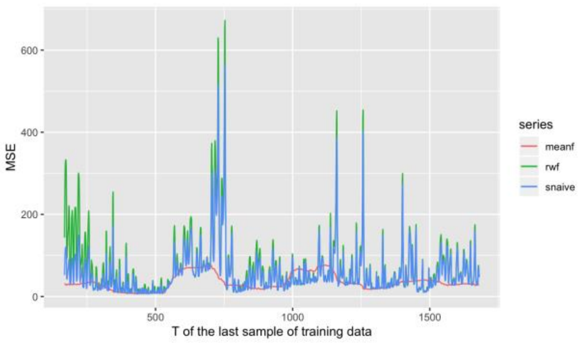


As we guessed, as the seasonality of the data is very high the seasonal naïve method yield the best overall results. We can make more remarks still. The drift variation was applied to the naïve method but, as the overall trend is very low on the dataset, the results are identical. Naïve method get a MASE of 1 because it is its MAE that is used by default as the scaling factor.

Cross-validation
We performed cross-validation in order to derive the best conditions in which we should run our forecasts. We focused on the length of the training set and on the effect on the error of the length of forecast.

Considering the time it took to run on our simpler forecasting methods we decided not to run for the whole dataset and we won't cross-validate on more complex models.



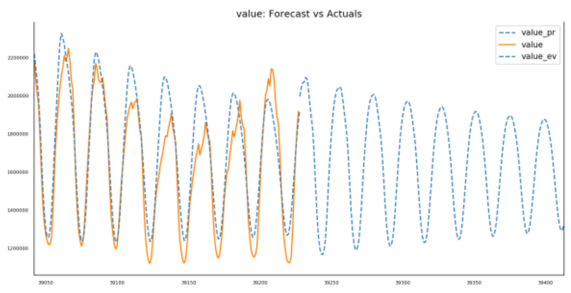


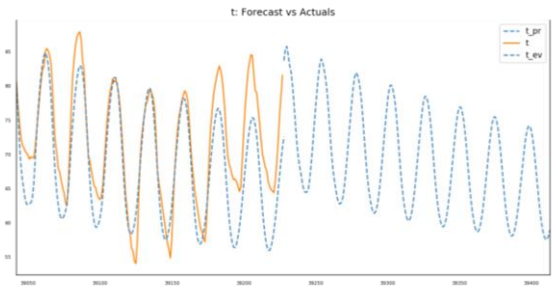
We can still analyze in general terms. It seems like those algorithms are not sensitive to the length of the training set (right figure) as error has no trend but more of a random variation. It appears that some algorithms (in that case the meanf) have a more constant error whatever the length of prediction. Whereas the others rely on the periodicity of the data to get low error.

5.b. Vector Autoregression (VAR)

In order to leverage the temperature information for a priciser forecast, a direct idea is to try vector autoregression. It is an augmented AR model, i.e. $VAR(p)\colon y_t = c + A_1 y_{t-1} + A_2 y_{t-2} + \ldots + A_p y_{t-p} + \epsilon_t$. And it requires that ① $E(\epsilon_t) = 0$; ② $E(\epsilon_t \epsilon_t{}') = \Omega$; ③ $E(\epsilon_t \epsilon_{t-k}{}') = 0$ for any $k \neq 0$.

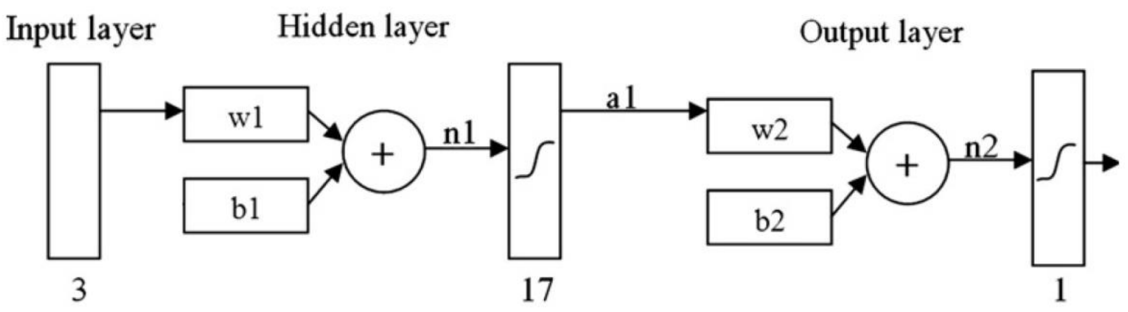
With the Granger's Causality Test, it is confirmed that the temperature and energy loads Granger cause each other. And with Augmented Dickey-Fuller Test, it is confirmed that the two-variate time series is stationary. In accordance with AIC criteria, the model parameter, lag, is set to be 96.





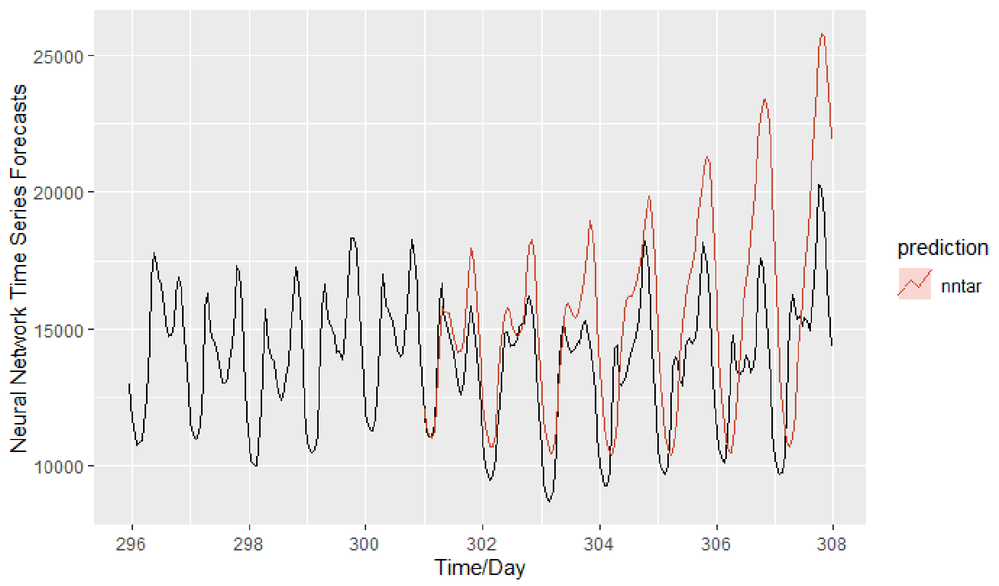
From the result, VAR can well capture the general pattern, though a little rough and with the increasing of time, there is a trend of convergence, which is not good. And the prediction of weather is quite inaccurate. In fact, this kind of iterative matrix multiplication will lead to divergence, convergence, or random walking. However, comparing with other methods with the same predicting period, VAR model is no better than seasonal naive.

5.c. Neural Network Time Series Forecasts (Neural Network Autoregression) (NNAR)



In this project we use feed-forward neural network to predict the time series with seasonality at time t. We use the notation $NNAR(p, P, k)_m$ to indicate p non-seasonal lag inputs, P seasonal lags inputs, m observations per period and k neurons in the hidden layer. So the input is : $(y_{t-1}, y_{t-2}, \dots, y_{t-p}, y_{t-m}, y_{t-2m}, y_{t-Pm})$.

We can remark that $\text{NNAR}(p, P, 0)_m$, which means a neural network without hidden layer, is equivalent to $\text{ARIMA}(p, 0, 0)(P,0,0)_m$, which means an ARIMA model with lag p and seasonal lag P, without differencing and moving average, so an autoregression model.



an example of results obtained by NNAR(30,1,16)[24]

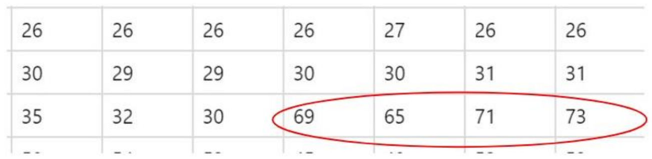
It is shown on the graph that the NNAR well model the seasonality and short-term trend, as well as the features of the time series, as the “grooves” in each period are well predicted.

Neuron Network autoregression with external regressors.

Since the temperature is given, we can use the temperature data as external regressors in our neuron network model.

The temperature data of 11 stations is given, however we don't know the correlation between the 11 stations and 20 zones. It is possible for us to match the zone with the station that correlates the most as input data. At last we chose the mean temperature of the 11 stations, the reasons are as follows:

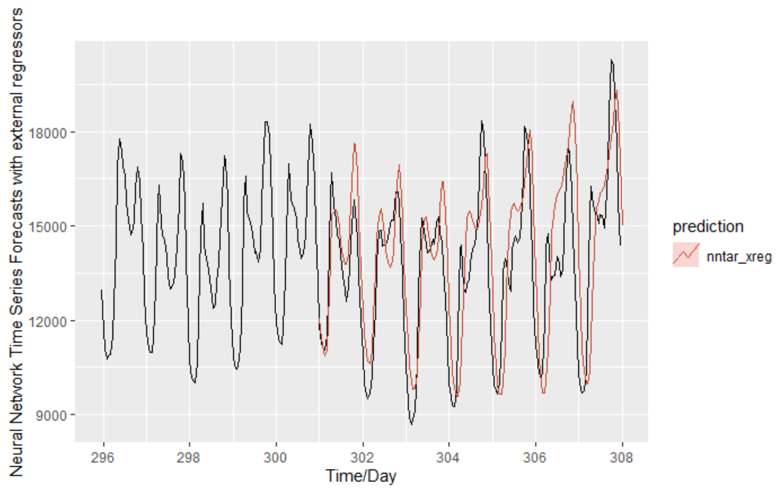
1. The dirty data : we can observe some abnormal values in the temperature data. For example, a sudden rise of temperature at night is not possible for the whole zone, but it can be explained by an approaching man-made hot source. To avoid this case it's better to use the mean temperature of 11 stations.

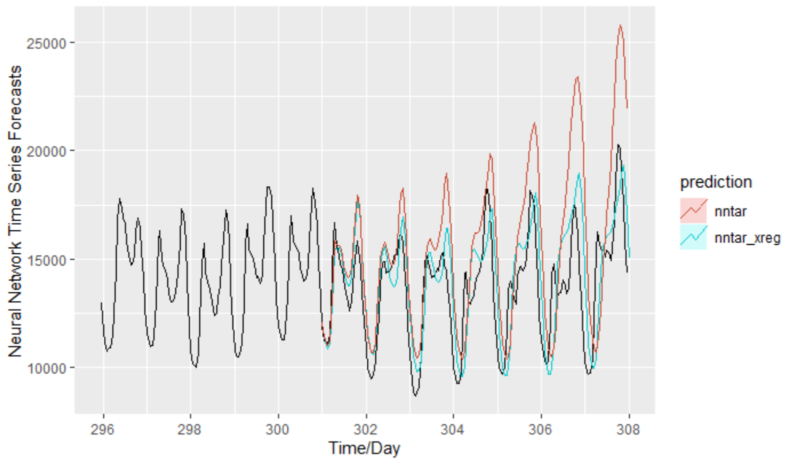


26	26	26	26	27	26	26
30	29	29	30	30	31	31
35	32	30	69	65	71	73

2. Matching the stations with the zones is "businesswise wrong", cause the data from a weather station in California may match very well with a zone in Florida. But, it is just a coincidence.

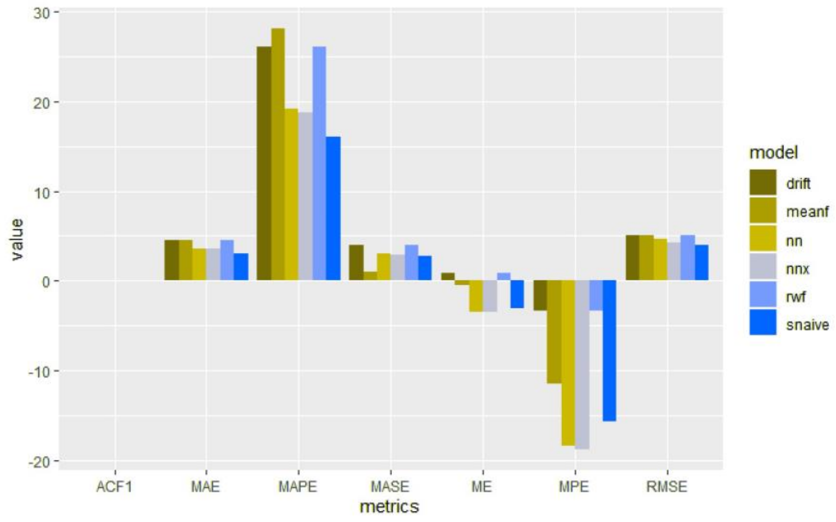
According to the correlation graph we have presented before in the global analysis, the value of energy consuming at t_{-0} is highly correlated with the temperature at t_{-12}, t_{-24}, etc. Thus, we chose the average temperature of the past 48 hours and the temperature 12 hours before as the external regressors.

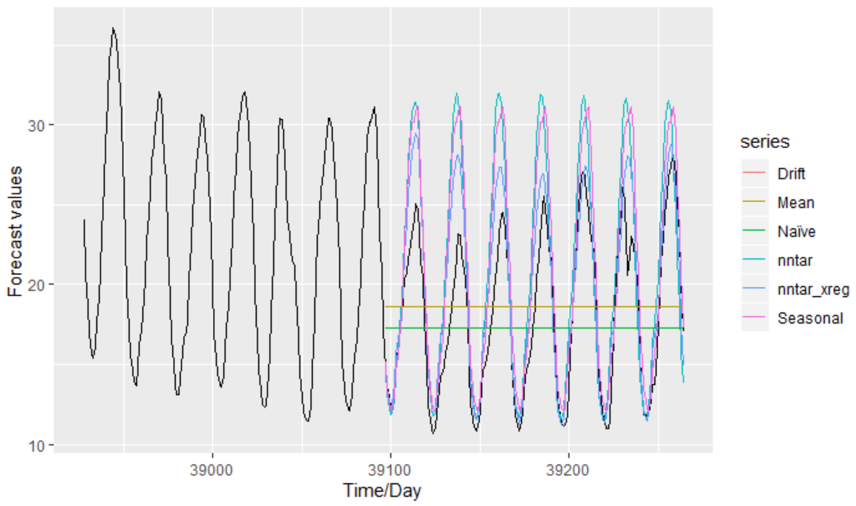




An example of results obtained by NNAR(30,1,16)[24] with ex-regressors

We can see that the neural network autoregression model with the external regressors works better than the one without external regressor. The two NNAR models all have a better performance than most of the simple models that we introduced, except the seasonal naive model. But the result of seasonal naive model is highly depended on the value of the last period.





Comparison of results obtained by the classic methods and neural networks

We should remark that the NNAR method can’t deal with the missing values, but we can use the NNAR with external regressors to fulfill the missing values, then NNAR to forecast, cause the future temperature isn’t given.

6. ARIMA and STLF methods

ARIMA and STL are very well known forecasting methods for times series provided in R. They both handle only univariate time series, but, in the case of ARIMA, one can use external regressors. Our data is a very complex one, with hourly records, both short and long seasonalities, and it's challenging for models such as ARIMA. Thus, an effective time series model must be sufficiently flexible to capture these principal features without imposing too heavy computational or inferential burdens.

6.a. Theory

6.a.i AutoRegressive Integrated Moving Average (ARIMA)

Known from the name, the ARIMA model is the combination of autoregression model (AR), moving average model (MA) and differencing.
For AR model, $y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t$, where $E(\epsilon_t) = 0$, and $E(\varepsilon_t \varepsilon_\tau) = \begin{cases} \sigma^2 & t = \tau \\ 0 & t \neq \tau \end{cases}$. While for MA model, $Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \ldots$, where $\{\epsilon_t\}$ is a series of white noise, satisfying $\sum_{j=0}^{\infty} \theta_j{}^2 < \infty$. Consequently, the ARIMA model can be written as: $y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$.
in which$y_t{}'$ is the difference term.

We denote the model by $ARIMA(p, d, q)$, where p is the order of the autoregressive part, d is the degree of first differencing involved, and q is the order of the moving average part. The equation can be written in backshift notation as

$$\underset{\substack{\uparrow \\ \text{AR}(p)}}{(1 - \phi_1 B - \cdots - \phi_p B^p)} \quad \underset{\substack{\uparrow \\ d \text{ differences}}}{(1 - B)^d y_t} \quad = \quad c + \underset{\substack{\uparrow \\ \text{MA}(q)}}{(1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t}$$

where $y_t{}' = (1 - B)^d y_t$ and $c = \mu(1 - \phi_1 - \cdots - \phi_p)$, $\mu$is the mean of $y_t{}'$.

An $ARIMA(p, d, q)$ model is used for the stationary series, which means time series without seasonality and trend. Before applying the ARIMA model we have to decompose the series to remove its seasonality and trend.

A seasonal ARIMA model is formed by including additional seasonal terms in the classic ARIMA models. It is denoted by $ARIMA(p, d, q)(P, D, Q)_m$ where (P, D, Q) are the seasonal terms.

$$\text{ARIMA} \quad \underset{\substack{\uparrow \\ \text{Non-seasonal part} \\ \text{of the model}}}{\underbrace{(p, d, q)}} \quad \underset{\substack{\uparrow \\ \text{Seasonal part of} \\ \text{of the model}}}{\underbrace{(P, D, Q)_m}} \quad \begin{aligned} &(1 - \Phi_1 B^{\omega} - \Phi_2 B^{2\omega} - \cdots - \Phi_p B^{P\omega})(1 - \varphi_1 B - \varphi_2 B^2 - \cdots \\ &\quad - \varphi_p B^p)(1 - B^{\omega})^D (1 - B)^d Z_t \\ &\quad = (1 - \Theta_1 B^{\omega} - \Theta_2 B^{2\omega} - \cdots - \Theta_Q B^{Q\omega})(1 - \theta_1 B - \theta_2 B^2 - \cdots \\ &\quad\quad - \theta_q B^q)\varepsilon_t \end{aligned}$$

The auto.arima() function in R uses a variation of the Hyndman-Khandakar algorithm [3], which combines unit root tests, minimisation of the AICc and MLE to obtain the best ARIMA model.

6.a.ii Seasonal Trend decomposition procedure based on Loess

STL is a filtering procedure for decomposing a time series into trend, seasonal and remainder components. This procedure has a simple design that consists of a sequence of application of the Loess smoother, therefore it allows fast computation, even for very long time series and large amount of trend and seasonal smoothing.

Let Yt represent our data at time t, assuming an additive decomposition$\exists T, S, R\ as\ \forall t \in T, Yt = Tt + St + Rt$, where Tt is the smoothed trend component, St is the seasonal component and Rt is a remainder component.

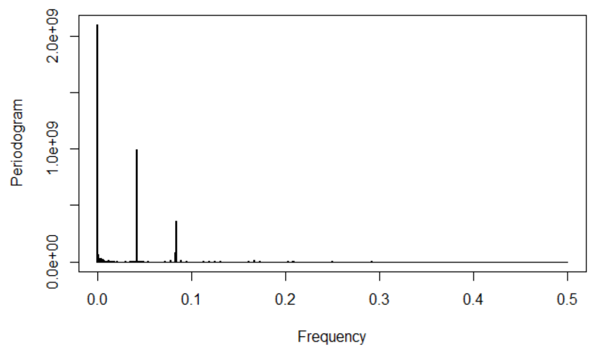
For advanced details about the Loess smoothing our about parametres (e.g. number of loops), you can refer to the original paper in the *Journal of Official Statistics* in 1990[2]. Nevertheless it basically consists in :

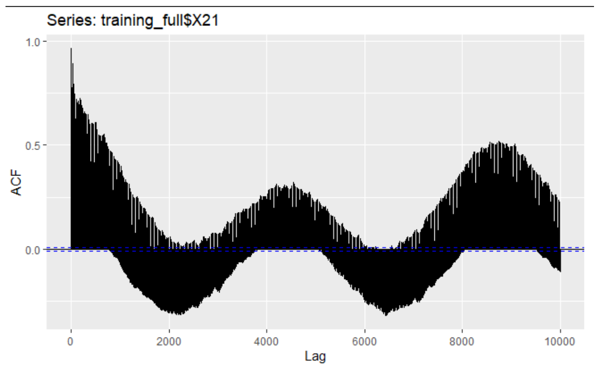
- An inner loop (at k pass) : computes $DT^k = Y - T^k$. Smooth every subseries of DT^k (in our case every day of each year) using loess, store it in C^{k+1}. Filter C^{k+1}using 3 MA and a loess smoothing and store it in L^{k+1}. The seasonal component (k+1 loop) is $S^{k+1} = C^{k+1} - L^{k+1}$, it prevents low frequency power to enter S^{k+1}. $DS^k = Y - S^{k+1}$, deseasonalized series is smoothed by loess and gives T^{k+1}.
- An outer loop : computes $R = Y - T - S$, define $h = 6 * median(|R|)$ and compute the robustness weights $p_v = B(|R_v|/h)$with B the bisquare function. These weights will be used in the loess smoothing at the next outer loop, so in the computation of C and Tin the inner loop.

The stl forecasting computes $\hat{Y} = \hat{S} + \hat{A}$ where $\hat{A} = \hat{R} + \hat{T}$. To compute $\hat{S}$, it assumes that the seasonal component changes extremely slowly, so it is forecast by simply taking the last year of the estimated component (or average). In other words, a seasonal naïve method is used for the seasonal component. To forecast $\hat{A}$,the seasonally adjusted component, any non-seasonal forecasting method may be used (e.g. a non-seasonal ARIMA).

6.b. Test and discussions:

In this section, we will focus on the virtual zone 21. Recall that all zones are characterized by complex seasonality as highlighted here :

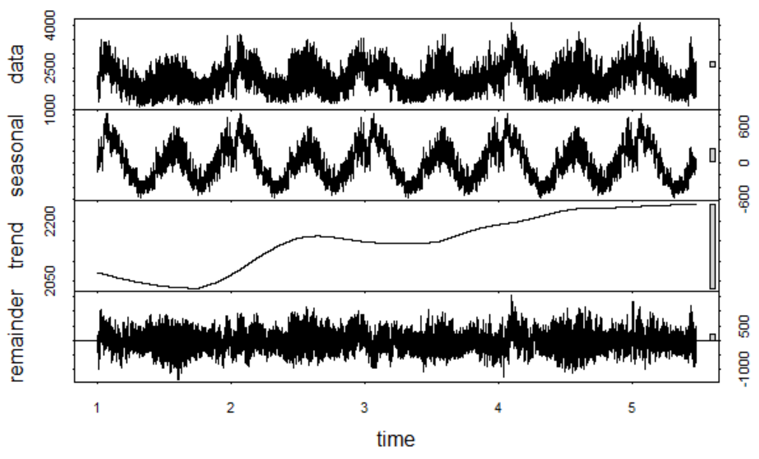




From these graphs, we assume that there are 4 different seasonalities, by order of importance, 4374, 24, 12, 8765. Or in a most relevant manner, half a year seasonality, daily seasonality, half daily seasonality, yearly seasonality.

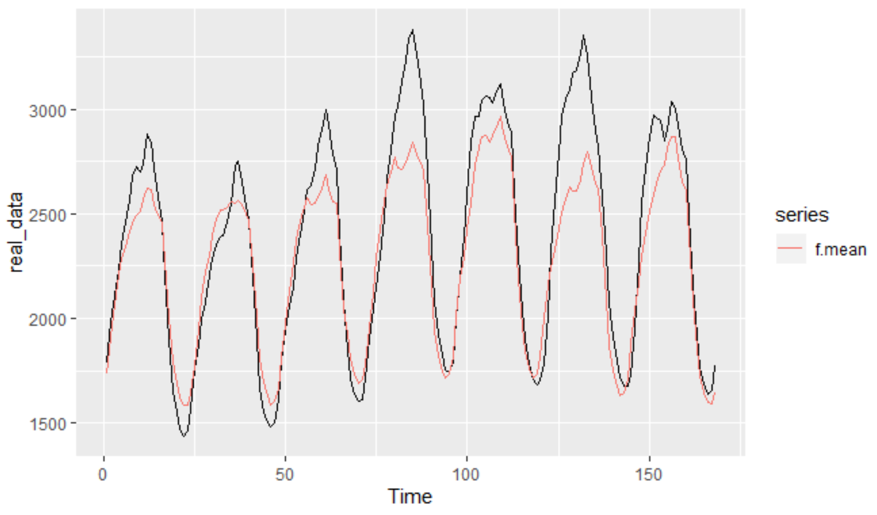
Now let’s create our first R objects, time series. We could use either ts or msts, we will chose the latter as msts objects are more flexible than ts ones. For instance msts can handle multiple seasonalities and non integer frequencies. To create a time series object, you have to choose a “frequency” (which has a different meaning from physics). Our data is recorded each hour, as the longest seasonality is the yearly seasonality, we set the frequency argument to 365*24. Now we apply the stl function to our time series.

To properly analyze this result you have to take care about the different scales. It’s clear that our data is mostly seasonal with a weak trend. You can also see that the remainder power and the seasonal power are equivalent, our data is very noisy. From the remainder, you can spot outliers, there are recognizable with peaks in the remainder plot.



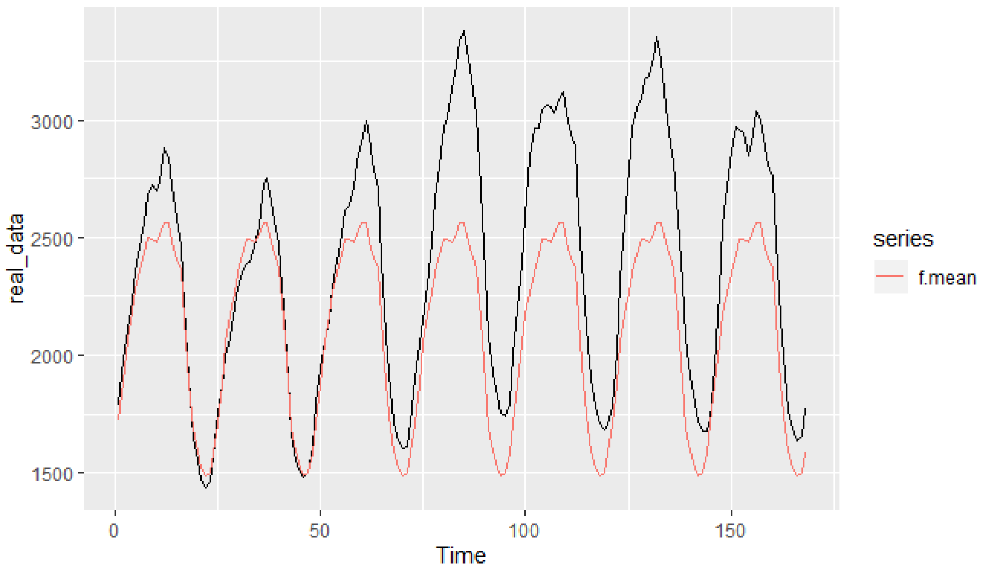
Many studies mention that there is “hidden” seasonalities that acts as outliers to the algorithm, adding noise to it, although they may actually being useful to the prediction. For instance, it's the case of moving events such as Easter or the Chinese New Year, or festivals that can fall in either March or April (for Easter) or in January or February (for the Chinese New Year). It’s very difficult for usual models to catch those patterns, and it’s only possible by using dummy variables as external regressors.

Now it’s time to perform our first forecasting. We use stlf function, with arima method to forecast the non seasonal data (i.e. trend + remainder).

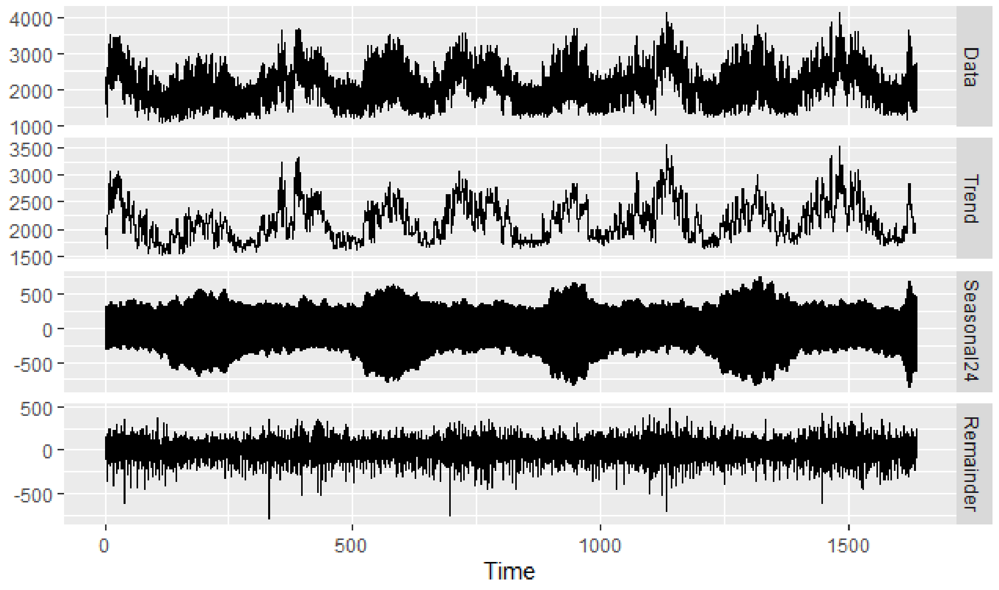


Results seems very accurate, with a MPE of 3.06 and a MAPE of 6.34.

But what if we had worked with the same time series but with a different frequency ? For instance, our data has clearly a daily seasonality. We can create an other time series, with a frequency of 24. Here is the result given by the exact same method.



As you can see, the forecasting is really different with MPE of 10.3 and MAPE 10.95.
It can be explained regarding the trend from the decomposition, we can still see periodicity in the trend, this information is lost (the arima model fitted is non seasonal) !



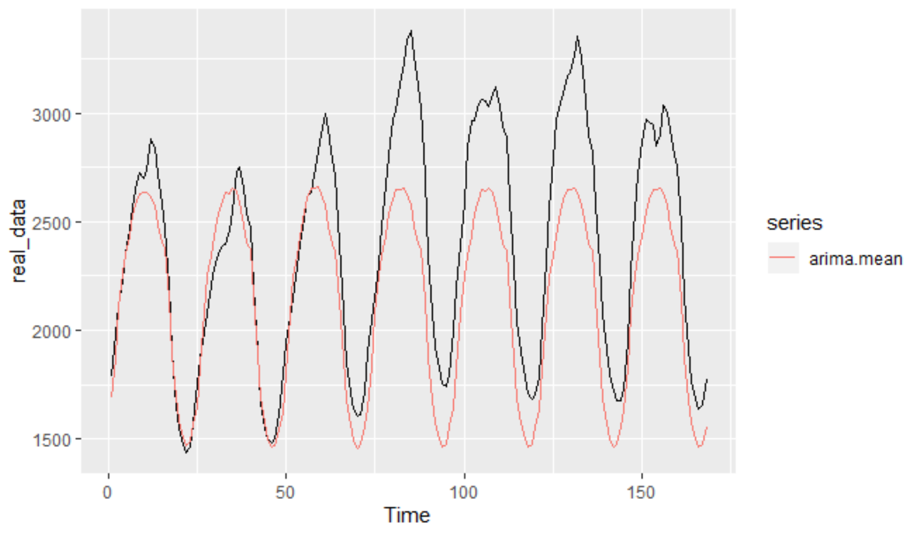
It seems that it is always valuable to set frequency on the longest seasonality observed because it captures both long patterns and short patterns. Moreover, even if seasonal patterns may change a little bit over time, it’s not a big issue since stl handle those slightly changes.

Using a loop I could have tried to find even best results but it sounded a bit like overfitting our data, therefore it’s important to use cross validation on the frequency parametre (didn’t have time to do it). Observing the impact of the frequency parametre, I noticed that the frequency must be a multiple of the smallest seasonality observed, otherwise the forecasting method doesn’t work properly.

Seasonal ARIMA:

Due to the maths behind ARIMA, it takes very long time to fit large data set, and struggles with time series of frequency higher than 280. But as a beginner, I tried first to use the function provided by R, auto.arima on one of my time series with yearly seasonality. Which resulted in many crashes from the Rstudio application.

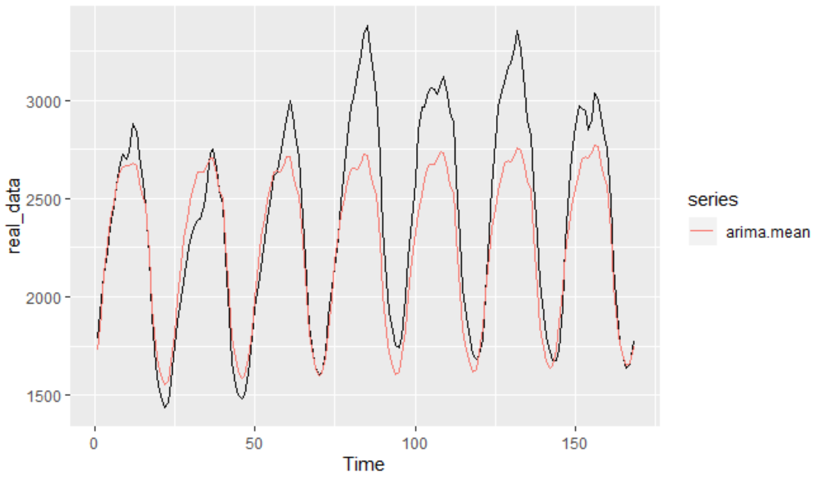
It’s actually possible to use auto.arima function on our data but on time series with daily frequency. To find the best window length of training, I used cross validation, and I chose the one that minimised the MAPE forecasting error. I found out that the mean MAPE error increased a lot with training set shorter than 6 months. On a 6 month training set, it takes approximately three minutes to fit and find that the ARIMA (5,0,0)(2,1,0)[24] model is the best with following forecast.



The accuracy is rather good, with MPE 9.42 and MAPE 10.48. But we can infer from the forecasting that the model does only take into account the daily pattern,not any moving average (neither for seasonal component nor static component). This is only due to the fact that we work with daily time series, so we lost part of the information from the longer seasonalities. How to solve this issue?

An important question with the ARIMA forecasting is the window size of training. Is it valuable to train on the complete dataset? Basically, larger training data sets are effective antidotes to fight against overfitting (as we just did). But in our case, reducing the window training could bring information from the longer seasonalities, indeed, if the window training is shorter than the half yearly seasonality, ARIMA would detect an MA, and then prediction will get better.

Is it overfitting? I would say no, indeed, we used our knowledge of seasonalities lengths to infer that such a window is adapted to this particular forecasting. But we have to remember that it won't be a good setting to predict at another time. Since efficiency of such a method is questionable I preferred to move on to other methods. Still, prediction with a window size of training of one month is illustrated below.



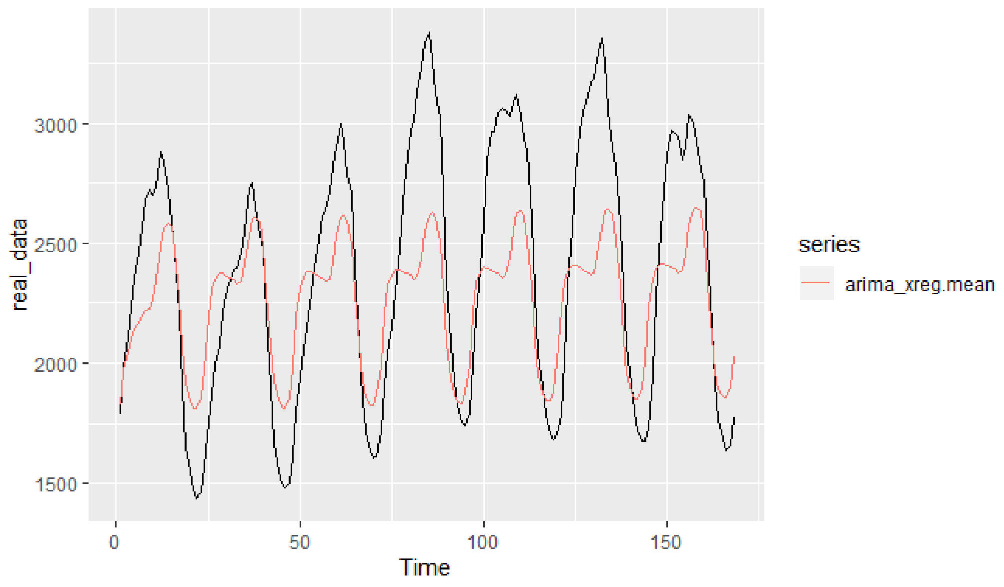
Model fitted ARIMA(3,0,0)(2,1,0)[24] with MPE = 4.3 and MAPE = 7.16.

Dynamic harmonic regression:

I mentioned earlier that seasonal ARIMA had trouble to handle large frequencies, but there is a way to take care of the complex seasonalities we have, we should use a non seasonal ARIMA model together with an external Fourier regressor. The only difference is that the seasonalities will be fixed in time here.

Fourier function returns the fourier terms from the decomposition, and can predict them as well. The higher the order (K), the more "wiggly" the seasonal pattern is allowed to be. With $K=1$, it is a simple sine curve. You can select the value of K by minimizing the AICc value.

But recall that our data has short seasonalities (=24) and larger ones (=365*24), to capture both seasonalities, we should use a fourier external regressor on a mtst object of these two frequencies. Therefore, to compute fourier terms, your training set should be at least one time longer than your longest seasonality. By playing with K, you can now control the complexity of the decomposition of the two different time series. Beware, although the fourier decomposition is quick, fitting an arima model on it is quite long (but at least, it's not crashing your application)



Results obtained with a msts time series of frequency (24,365*24), fitted with an ARIMA xreg = fourier(K= c(12,2)), accuracy of MPE = 2.2, MAPE = 12.38 with

Conclusion on STLF and ARIMA methods:

It appears that, concerning our data, the STLF method is way more efficient than ARIMA methods. This efficiency is expressed in terms of accuracy (stlf with frequency = 365*24 is actually the best forecasting we had so far), in terms of computational time (fitting a model takes at most one minute), and in terms of mathematical complexity.

I actually firstly tried STL methods before ARIMA, because it seems pretty natural to me to capture the seasonalities, remove it from the data, and forecast both parts (seasonal and remainder+trend) independently.

ARIMA methods seems pretty useful in many other cases, and I really tried to dig into this model to get better results, but here we reached the limits of such a model.

And what about forecasting with temperature regressors ?

Forecasting with such models is difficult because we require future values of the predictor variables (here temperature or Fourier terms). Future values of the Fourier terms are easy to compute, but future temperatures are, of course, unknown. If we are only interested in forecasting up to a week ahead, we could use temperature forecasts obtained from a meteorological model. Alternatively, we could use scenario forecasting and plug in possible temperature patterns. Looking closely to the data it appears that the total electricity load is a smoothed transformation of mean temperature. I didn’t assess this statement with statistical arguments but it seems that forecasting the electricity load is easier than forecasting the temperature.

Nevertheless, temperatures were really useful in our case to backcast the missing values in our data. The other group achieved a very accurate backasting using neural networks.

7. Conclusion

Thanks to this challenge we discovered multiple ways of forecasting time-series as well as some inherent problems such as dealing with missing values or badly formatted data. What we can assure is that no method can be applied eyes shut. One must investigate the data in order to detect patterns, seasonality, abnormalities before choosing the appropriate forecasting method.

As a general conclusion after trying all the methods we can say that the more complex the algorithm does not mean it yields better results. Depending on the application context, one might be satisfied with the great results at low cost given by either simple methods or neural networks. But on the other hand if one want to invest time in analyzing the dataset with more care, extracting seasonality and adding external regressors more complex methods exists. They can give great results but require time to understand, to tune the parameters and to compute.

Bibliography

[1] Wang, P., Liu, B., & Hong, T. (2016). Electric load forecasting with recency effect: A big data approach. International Journal of Forecasting, 32(3), 585-597. DOI:10.1016/j.ijforecast.2015.09.006

[2] Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: a seasonal-trend decomposition. Journal of official statistics, 6(1), 3-73.

[3] Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27(1), 1–22. DOI: 10.18637/jss.v027.i03

[4] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.

[5] https://cran.r-project.org/web/views/TimeSeries.html