Random Forests for Time Series

Benjamin Goehry, Hui Yan, Yannig Goude, Pascal Massart, Jean-Michel Poggi

Random Forests for Time Series

Benjamin Goehry*[1], Hui Yan†[2], Yannig Goude‡[1,2], Pascal Massart§[2], and Jean-Michel Poggi¶[1,3]

[1]Laboratoire de Mathématiques d'Orsay, Université Paris-Saclay, France
[2]EDF Lab, France
[3]University Paris, France

Abstract

Random forests were introduced in 2001 by Breiman and have since become a popular learning algorithm, for both regression and classification. However, when dealing with time series, random forests do not integrate the time-dependent structure, implicitly supposing that the observations are independent. We propose some variants of the random forests designed for time series. The idea is to replace the standard bootstrap with a dependent bootstrap (i.e block bootstrap) to subsample time series during the tree construction phase to take time dependence into account. We then present two numerical experiments on electricity load forecasting. The first one, at a disaggregated level, is based on an application to load forecasting of a building and illustrate how the variants may perform. The second one is at a more aggregated level (French national forecasting) but focusing on atypical periods. For both, we explore a heuristic for the choice of the block size, the new parameter.

Keywords— Block bootstrap, Random forests, Time series
AMS Subject Classification— 62M10, 62P30.

1 Introduction

Random forests were introduced in 2001 by Breiman in [1] and are since then one of the most popular algorithms in machine learning [2]. The popularity comes from the

*benjamin.goehry@gmail.com
†hui.yan@edf.fr
‡yannig.goude@edf.fr
§pascal.massart@math.u-psud.fr
¶Jean-Michel.Poggi@math.u-psud.fr

wide range of applications in which they are known to perform well on even high dimensional, are fast to compute and easy to tune. Successful applications can be cited: chemo-informatics [3], ecology [4, 5], 3D object recognition [6] and time series prediction [7, 8, 9, 10, 11].

Suppose that we have a random sequence $(X_t, Y_t)_{t\in\mathbb{Z}} \in \mathcal{X}\times\mathcal{Y}$ such that

$$Y_t = f(X_t) + \epsilon_t \tag{1.1}$$

and the error ϵ_t is such that $\mathbb{E}[\epsilon_t|U_t] = 0$. The purpose of random forests is to estimate, by only observing a training sample $\mathcal{D}_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$, the regression function

$$\forall x \in \mathcal{X}, f(x) = \mathbb{E}[Y_t|X_t = x].$$

Random forests can be related to two main sources, regression trees [12] and bagging [13]. Regression trees are constructed by a recursive partitioning of the input space based on some criterion to estimate the regression function f. At each step of the tree construction, a split is selected (a variable and a location on the variable) based on the evaluation of the criterion among all the admissible splits based on all the variables. The cell is cut in two on the selected split and the previous step is reiterated on the new cells. A tree is then a piecewise constant decomposition of the input space. We can associated to the input space partitioning a binary tree where each node corresponds to a test matching how the input space was cut. An illustration is given in fig. 1 of a partitioning in the two-dimensional space and its associated binary tree. The principle of bagging (short form of bootstrap aggregating) is to create M randomly generated training sets by randomly sampling α_n observations with or without replacement from the set $\mathcal{D}_n$ and to construct on each set a predictor. Once the predictors are constructed, the bagging prediction for a new observation x is an aggregation, generally the empirical mean, of the predictions given by the M predictors for the point x. This procedure aims to improve stability and accuracy of the base predictor. In the context of random forests the predictors are regression trees. In order to explain the random forest procedure we then have to explicit the construction of one tree.

The first step is the bootstrap/subsampling: α_n points are selected with or without replacement among the n realisations. Then a tree is constructed based on these α_n selected points. At each node of the tree the best split (the variable and the location on this variable) is determined by minimising the intra-node variance. This is commonly called the CART criterion introduced in [12]. Instead of minimising this criterion among all the admissible splits based on all the variables the choice of inputs is restricted to a random subset of fixed size m_{try}. This procedure is then iterated on each node produced after binary splitting until stopping conditions are met. The first stopping rule is when the variance in a node is equal to zero. Since this is rarely the case a second condition is that the number of observations in a node must be greater than a given threshold.

Even if the theoretical settings of random forests was until recently restricted to the i.i.d case, a theoretical study extending it to the time-dependent case is proposed in [14]. In addition, applications on time series could be found, as previously cited, in [7, 10], in electricity load forecasting [8], [9], [11].

The bootstrap step determines which observations are chosen to construct a tree. The original bootstrap which we call standard (or i.i.d) bootstrap from [15] consists of randomly

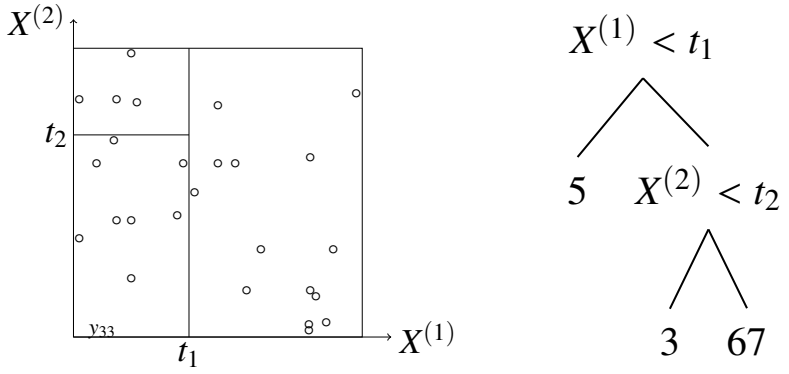

Figure 1: A partitioning of $[0,1]^2$ and the associated binary tree.

drawing α_n observations among the n with or without replacement. Note that we use here an abuse of language, the bootstrap is standardly defined as drawing n observations among the n observations with replacement. The goal of this bootstrap is to replicate the distribution of $\mathcal{D}_n$. However, this is adapted to the case of independent and identically distributed observations. When the data has an underlying dependence structure as for time series the i.i.d hypothesis is not verified anymore and using the standard bootstrap destroys the dependent structure. We illustrate this phenomenon for a dataset from [16] which is described in section 3.1. We observe in fig. 2 the original load over the month of January. Using the standard bootstrap we obtain the series in fig. 3 and immediately note that the structure we had in the original series is all gone. By contrast, using a moving block bootstrap, described in section 2, using a block length of 24 hours we recover similar patterns as in the original series of fig. 4.

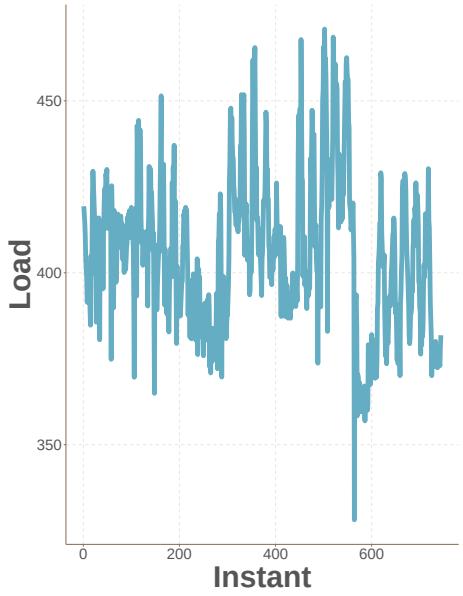

Figure 2: Original load hourly sampled.

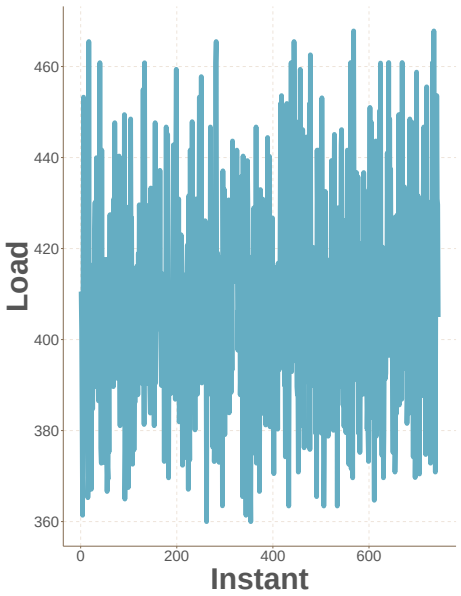

Figure 3: Bootstraped load

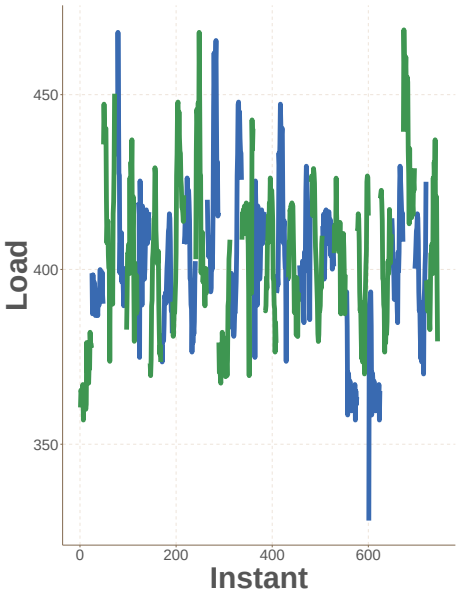

Figure 4: Block bootstrapped load with block size of 24h

We list here a few papers using blocks bootstrap in the forecasting literature. The first one is [17] in which they use a sieve bootstrap to perform bagging with exponential smoothing models. They use exponential smoothing to decompose the data, then fit an autoregressive model to the residuals, and generate new residuals from this AR process. Finally, they fit the exponential smoothing model that was used for decomposition to all bootstrapped series. Another work is from [18] who propose a method of bagging which is as follows. After applying a Box-Cox transformation to the data, the series is decomposed into trend, seasonal and remainder components. The remainder component is then bootstrapped using the moving block bootstrap, defined in section 2, the trend and seasonal components are added back, and the Box-Cox transformation is inverted. For each one of these bootstrapped time series, a model among several exponential smoothing models is chosen, using the bias-corrected AIC. Then, point forecasts are calculated using all the different models and the resulting forecasts are combined using the median. We refer to [19] for more details about the recent developments in bootstraps methods for dependent data.

The aim of this work is to show that the forecasting performance could be improved by replacing the bootstrap step by what we call block bootstrap variants, to subsample time series during the tree construction phase and thereby keep the dependent structure. Since random forests were already introduced in this introduction. The next section presents the different block bootstrap variants, the new algorithm and a new way to compute the variable importance. We then present two numerical experiments. The first one is based on an application to load forecasting of a building from the dataset described in [16] and see how the variants may perform. The second one on the French national forecasting problem and explore a heuristic on the choice of the new parameter.

2 Random forests for time series

2.1 Block bootstrap variants

Non-overlapping block bootstrap A first variant is found in [20]: the *non-overlapping block bootstrap*. The idea is to construct a number of non-overlapping blocks and then to draw uniformly, with replacement, among the constructed blocks. More precisely, let l_n be the size of a block and $B \geq 1$ the greatest integer such that $l_n B \leq n$. The blocks are then constructed the following way

$$B_b = \left(X_{(b-1)l_n+1}, \ldots, X_{bl_n}\right) \; b = 1, \ldots, B.$$

The bootstrap set $\mathcal{D}_n^\star$ is then obtained by drawing K blocks, $\left(B_1^\star, \ldots, B_K^\star\right)$, uniformly with replacement in the collection of non-overlapping blocks$\left(B_b\right)_{1 \leq b \leq B}$ for a suitably chosen K.

Moving block bootstrap [21] and [22] introduced the so-called *moving block bootstrap*. The idea is, instead of picking randomly one observation among the n observations as for the standard bootstrap, the moving block bootstrap pick randomly a block of l_n consecutive observations. Repeating this step and concatenating all the selected blocks, we

get a new time series with a preserved structure at least in each block. More precisely, let us denote by $B_{i,l_n} = ((X_i, Y_i), \ldots, (X_{i+l_n-1}, Y_{i+l_n-1})$ the block of size l_n beginning with the observation (X_i, Y_i) for $i \in \{1, \ldots, n-l+1\}$. The procedure then consists to draw randomly K indices $(I_j)_{1\leq k\leq K}$ uniformly on the set $\{1, \ldots, n-l_n+1\}$ and associate one block to each index, $(B_{I_k})_{1\leq k\leq K}$. The bootstrap set is then defined as $\mathcal{D}_n^\star = (B_{I_1}, \ldots, B_{I_K})$.

Circular block bootstrap When studying the moving block bootstrap we can note that less weight is given to certain parts of the time series which also leads in theory to non negligible bias when computing the mean. A way to correct this issue is given in [23] introducing the so-called *circular block bootstrap*. The idea is to wrap the time series writing $X_i := X_{i_n}$ where $i_n = i \mod n, X_0 := X_n$ and then use the same procedure as in the moving block bootstrap where the index I is drawn uniformly on the set $\{1, \ldots, n\}$ instead.

Note that in each above variant, taking $l_n = 1$ we recover the standard bootstrap of [15]. For a given number of selected observations in each tree α_n the number of blocks K is such that $K = \frac{\alpha_n}{l_n}$.

2.2 Proposed random forest for time series

Our proposition in order to incorporate the dependence structure is by replacing the first step for the construction of a random tree in the random forest building procedure, namely replacing the standard bootstrap step with one of the block bootstrap variant recalled in section 2.1. The adapted algorithm is found in algorithm 1 underlining the modification with respect to the original random forest procedure.

2.3 Bloc permutation importance

Random forests can be used to rank with respect to a decreasing order of importance the variables. One way to measure the significance of a variable is the *Mean Decrease Accuracy* introduced in [1] which stems from the idea that if a variable is not important, then permuting its value should not change the prediction accuracy.

For each tree, we have access to the so-called *out-of-bag* observations denoted by OOB_m, composed of the observations not included in the bootstrap sample $\mathcal{D}_n^m$ used to construct the mth tree. The OOB_m sample can then be used to estimate the out-of-bag error denoted by $errOOB_m$. In order to compute the importance of the variable $X^{(j)}$, the values of the jth variable are randomly permuted in the OOB sample and compute for each tree an out-of-bag error estimation for the permuted observations. The importance of the variable $X^{(j)}$ is then obtained by averaging the difference between the out-of-bag error before and after permutation. More formally, if, for the mth tree, we denote by $\widetilde{errOOB}_m^j$ the OOB_m sample's error when the jth variable is permuted, then the importance of the variable $X^{(j)}$ is defined by

$$VI\left(X^{(j)}\right) = \frac{1}{M}\sum_{m=1}^{M}\left(\widetilde{errOOB}_m^j - errOOB_m\right).$$

input: $((X_1, Y_1), \ldots, (X_n, Y_n))$
parameters: $M, \alpha_n, m_{try}, \tau_n, \underline{l_n}$
stopping criteria: the variance in the node is zero or the number of observations in a node is below the threshold τ_n
for $j \leftarrow 1$ *to* M **do**
Construct the jth tree:

- Draw $\alpha_n \leq n$ observations
  using a block bootstrap variant with parameter l_n.
- Repeat recursively on each resulting node the following steps until a stopping criterion is met:
  - At each node, select randomly m_{try} variables
  - Select the best split using the variance criterion among the previously chosen variables.
  - Cut according to the chosen split.

end
output for a new observation x: mean of the M predictions given by the trees for x.

Algorithm 1: Random forest for time series

The higher the increase in the prediction error after the permutation of the jth variable in the out-of-bag observations, the more important the variable is. However, if the permutation of $X^{(j)}$ doesn't change much the error prediction then the importance of the considered variable is small.

In the case of dependent observations we are faced with the same issue as in the construction of the random forests, namely the permutation of variable in the out-of-bag observations does not preserve the dependent structure. In the case where block instead of standard bootstrap is used in the random forest we introduce a new variable importance computation: the *block (permutation) variable importance*. However, using a block bootstrap variant doesn't necessarily lead to a out-of-bag observations with constant number of consecutive observations but we solve this issue in the following. Let us first suppose that the out-of-bag observations can be separated in blocks of size of the block size parameter in the forest l_n and denote by B_m^* the blocks in the out-of-observations for the mth tree. In order to compute the importance of the jth variable, the permutation of the considered variable is done by only permuting the blocks in B_m^* and preserving the structure in each block. We can then compute a block permuted out-of-bag error estimation for the jth variable denoted by $\overline{errOOB_m^j}$. The block variable importance for the jth variable is then defined by

$$VI\left(X^{(j)}\right) = \frac{1}{M}\sum_{m=1}^{M}\left(\overline{errOOB_m^j} - errOOB_m\right).$$

The out-of-bag observations stemming from the block bootstrap with parameter l_n is not necessarily composed of blocks of the size l_n but the non-overlapping block bootstrap. In order to obtain an OOB sample which has the same block size as in the construction of the random forest we adapt the obtained out-of-bag observations to get a new set of blocks of out-of-bag observations. The construction of the latter is as it follows. If a block of consecutive observations in the out-of-bag observations is of the right length l_n we add it to the block out-of-bag observations and if the length is larger than l_n and less than $2l_n$ we draw a random subset of consecutive observations of length l_n. If a block of consecutive observations in the out-of-observations has a length less than l_n then the block is not kept. Then the block out-of-bag observations is composed of the kept block observations of length l_n and satisfies the conditions to compute the block permutation variable importance as previously defined.

3 Numerical experiments

We consider two experiments in this work. One regarding the performance the variants may attain on a real world application of load forecasting, at a disaggregated level, on one of the building dataset from [16], which is composed of different building loads with hourly observations. The other regarding the choice of the block length parameter, this time on the French national load forecasting problem, at a more aggregated level but focusing on atypical periods.

We run the experiments by implementing the extra features we propose in this paper as an extension of the R package *ranger* [24], and thus inherit the availability in both

C++ and R. Our R package *rangerts* is freely available from the github repository `https://github.com/hyanworkspace/rangerts`. In the following experiments, the results are obtained over 50 runs. The parameters of the random forest are set to default except for the m_{try} parameter which is optimised on a validation set and the block size parameter for which we carry out an in-depth analysis in section 3.2.

3.1 First load forecasting application: On the performance and variable importance

This experiment is based on the so-called building loads, a collection of 507 whole buildings electrical meters made publicly available. We refer to the paper [16] for a complete description of the collection. We consider one specific building in the building data genome project called *UnivLab Patrick*. This building belongs to the college laboratory category located in the New York time zone and has an area of around 7054 square meters. We have access to its electricity load from the 1st January 2015 to the 31th December 2015 with a sampling rate of one observation per hour. The weekly profile is found in fig. 5. We see a clear daily trend as well as a clear distinction between the week and the end of the week due to less activity. We also have access to exogenous variables: the temperature as well as to the schedule of the building, indicating if a day is ordinary, a break or a holiday. We decompose the year in three parts: the training set is composed of the observations from the 1st January to the 31st October, the validation set corresponds to the month of November and the test set corresponds to the month of December.

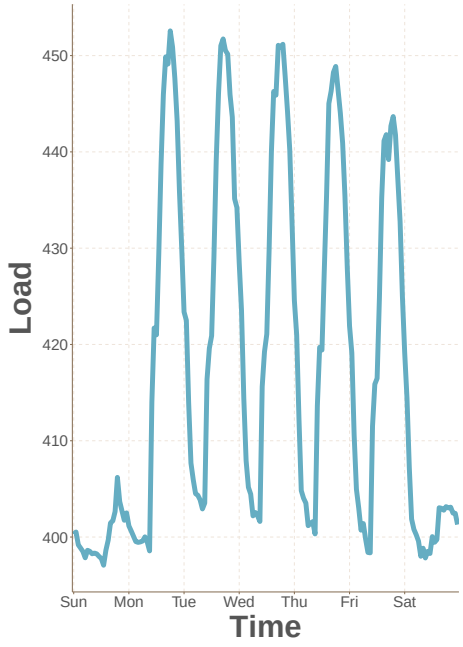

Figure 5: Weekly profile hourly sampled of the UnivLab Patrick dataset.

Let us denote by Y_t the system load of the building at hour t. In this experiment, we aim to forecast at a horizon of 24 hours. Based on the weekly profile, having hourly sampled observations, the chosen model is inspired by [25] in which they also considered random forests with a similar model for the same kind of problem. This results in the model described in eq. (1.1) with X_t of the form

$$X_t = (Y_{t-24}, Y_{t-168}, \text{Temp}_t, \text{Schedule}_t, \text{Hour}_t, \text{InstantWeek}_t, \text{DayType}_t, \text{Toy}_t) \tag{3.1}$$

where

- Temp_t corresponds to the temperature at instant t;
- Schedule_t take three values: Regular, Break, Holiday;
- Hour_t corresponds to the hour of the day at instant t;
- InstantWeek_t corresponds to the hour in the month;
- DayType_t corresponds to the day of the week;
- Toy_t corresponds to the day of the year divided by 366.

The selected value for m_{try} according to the performance on the validation set is $m_{try} = 2$. For this parameter we computed the different variants varying the block size parameters multiple of 6 hours up to 90 hours. We first optimise the performances on the validation set, looking for the best block size value minimising the RMSE and then plug it in for the

test set. The performance are resumed in fig. 6. We observe an improvement for the three variants with an improvement up to 11% for the mean RMSE compared to the standard random forest. We also show the evolution of the performance according to the block size parameter in fig. 7. We observe for the three variants a similar pattern in the evolution of the performance, namely a decrease for which the three variants performs better than the standard random forest and then an increase. We note that, even if the performance get worse when the block size is large, we also have a large window for which the performance is far better for these three variants with an optimal block size parameter of around 24 hours also corresponding to the forecasting horizon and the main seasonality of the data.

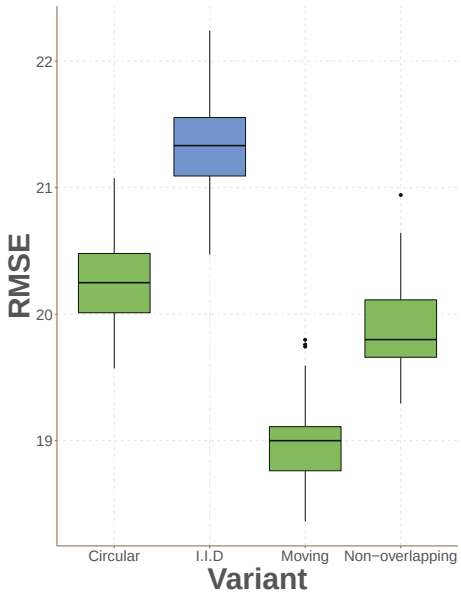

Figure 6: Performance of the different variants for $m_{try} = 2$, evaluated on the month of December of the UnivLab Patrick dataset.

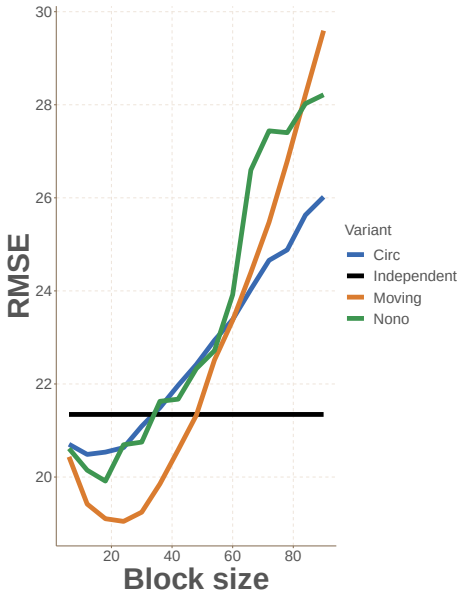

Figure 7: Performance of the variants for $m_{try} = 2$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

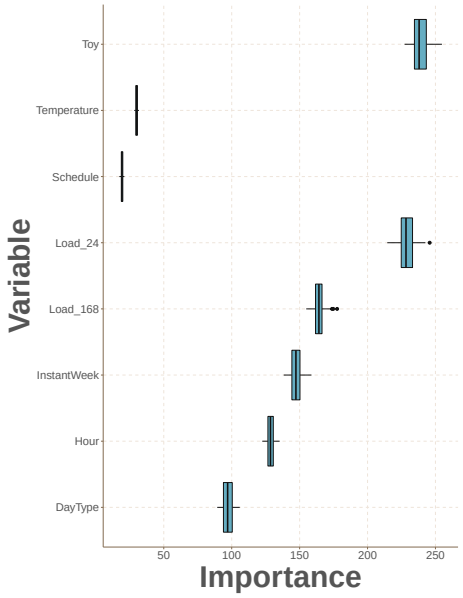

Figure 8: Variable importance moving bootstrap variant under the standard permutation on the UnivLab Patrick dataset.

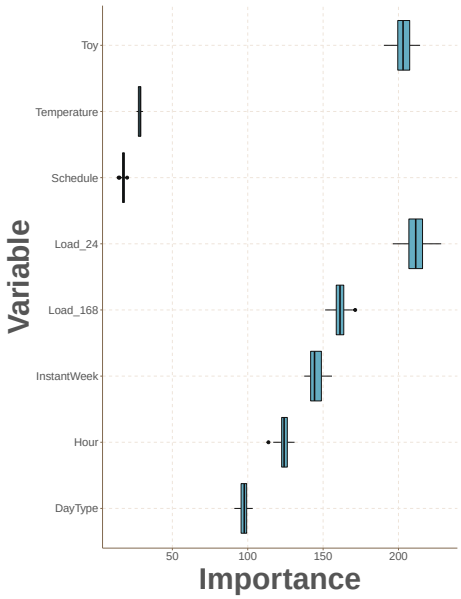

Figure 9: Block moving bootstrap variant importance with block size of 24h on the UnivLab Patrick dataset.

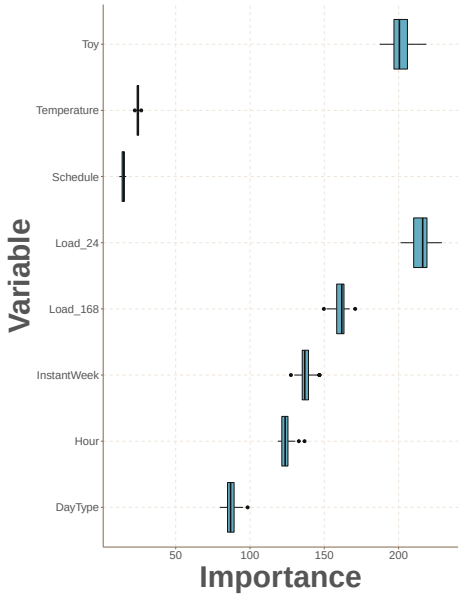

Figure 10: Variable importance non-overlapping variant under the standard permutation on the UnivLab Patrick dataset.

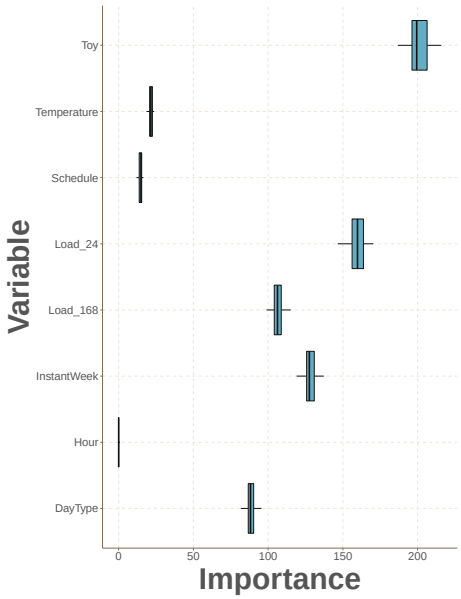

Figure 11: Block non-overlapping variant importance with block size of 24h on the UnivLab Patrick dataset.

Computing the variable importance for blocks of size 24 hours we obtain figs. 8 to 11. We observe that the difference between the standard variable importance and the block variable importance is essentially noticeable for the non-overlapping block bootstrap variant. The most evident difference is for the variable *Hour* for which the importance is set to zero using the block variable importance. Since the blocks are of length 24 hours and always beginning at the same time, permuting the blocks will not change the out-of-bag error since each permutation is replaced by an identical copy and thus the output from this procedure for the variable *Hour*.

3.2 Second load forecasting application: On the block length choice

We discuss here the choice of the block length parameter, found in every block bootstrap variant. In the previous experiment, we notice that the optimal choice for the block length was 24 hours, corresponding to the daily step and seasonality in the dataset. However, the last experiment is done by optimising the block length on the validation set error. It would be interesting to choose this parameter more wisely in order to avoid unnecessary computations and we think that it should be proportional to the (minimal) seasonality in the dataset. The block bootstrap aims to build blocks that preserve the dependency in them but that the blocks are independent to a certain extent. In the case of seasonal trends, the intuition would consequently be to choose blocks correlated to basic seasonal components.

We illustrate this with another dataset, on the French national load with goal to forecast at a 24 hours horizon as well, having a longer span of time and thus having more stable results.

We consider the French electricity load of the year 2015 as the training set with a sampling rate of one observation per day at noon. The test set for this experiment are the months April and October of the year 2016, corresponding to the transition between summer and winter season, a particularly difficult period to forecast. We observed in various experiments that the random forests for time series variants work the best when it is "difficult" to forecast. This typically corresponds to the shoulder seasons in the load forecasting field. We use here the model described in eq. (3.1) as well without the variables *Hour* and *InstantWeek*. Since the observations are daily occurrences, the minimal seasonality would be the week. Hence, we consider three values for the block length parameter: 7, 14 and 21 days. The selected value for m_{try} is 3 corresponding to the worst case scenario, in the sense that for another value of m_{try} the block bootstrap variants are doing better than shown in this example. Note that for this example we removed the non-overlapping block bootstrap variant. We have found that this variant needs more observations to get consistent results, providing less diversity in the trees due to its construction.

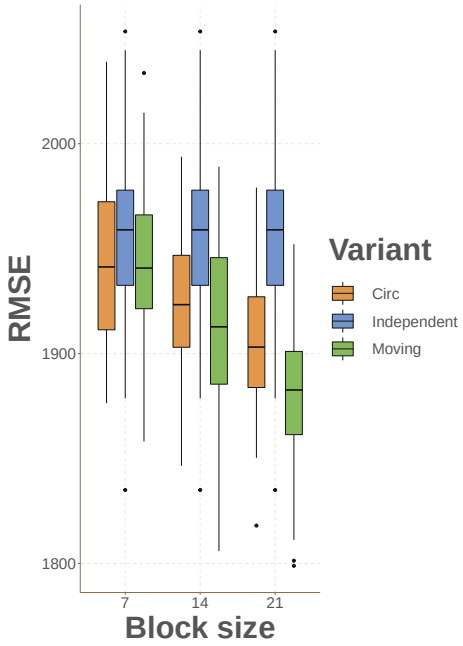

Figure 12: Performances evaluated on April 2016 on the French load forecasting problem of the different variants for three block length values.

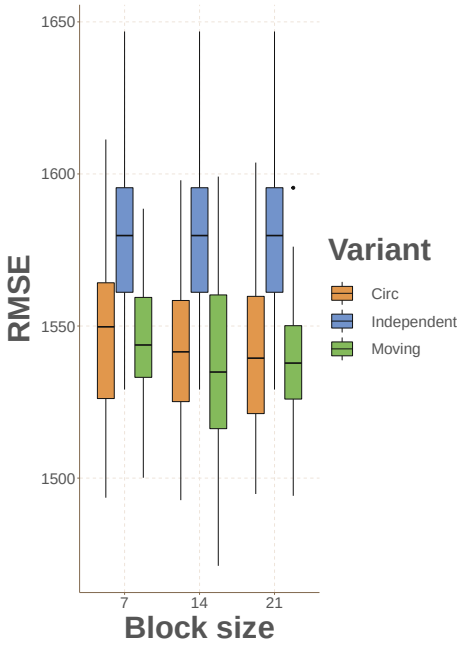

Figure 13: Performances evaluated on October 2016 on the French load forecasting problem of the different variants for three block length values.

The results are found, respectively for April and October 2016, in figs. 12 and 13. We observe that, for both month, we have a consistent improvement of the performance in comparison to the standard random forest for each choice of block length. We even note significant improvement in the performance when taking twice or thrice the seasonality for April. However, taking larger values than these would lead to a diversity problem in the trees as mentioned before and thus have less consistent performance. This concludes that the heuristic for the block length parameter choice would be to take the smallest seasonality up to a multiplying factor of two or three.

4 Conclusion and Perspectives

We introduced a new variant of random forests taking into account the temporal dependency of the observations and showed that we can improve significantly the performance on forecasting tasks when choosing the right block length. A variant of the variable importance based on the block bootstrap mechanism is also introduced. The non-overlapping variant seems to be mistaken regarding the importance of the variables, forgetting some variables fundamental to the forecasting problem as the hour variable in our first application, and thus we do not advise to use this variant for this purpose. However, both moving and circular variants seem to perform much better than the standard random forests when the block length is well-chosen, and we showed that a good heuristic for the block length choice is correlated to a multiple of the smallest seasonality.

This work is mainly methodological, a first perspective would be to prove theoretical results on the random forests variants under time-dependent observations hypotheses. Consistency of random forests is proven under stationary and $\beta-$mixing hypotheses in [14] when trees are not fully grown and the observations are subsampled. The previously cited works regarding the block bootstrap as [20, 21, 22, 23] also show consistency of some estimators, generally under less restrictive hypotheses. It would be interesting to prove similar results on the variants by adapting and combining the previous proof techniques.

We have considered one specific field of application to illustrate the potential value of the random forests variants. Of course, a more extensive study of other time series forecasting context could be considered for future works. It could also be useful to develop a more adaptive and automatic way to choose the block length parameter. Finally, it could be interesting to explore more deeply under which conditions (input variables, etc.) the variants work, going well beyond the scope of this paper.

References

[1] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[2] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, The Journal of Machine Learning Research 15 (1) (2014) 3133–3181.

[3] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, B. P. Feuston, Random forest: a classification and regression tool for compound classification and qsar mod-

eling, Journal of chemical information and computer sciences 43 (6) (2003) 1947–1958.

[4] D. R. Cutler, T. C. Edwards, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, J. J. Lawler, Random forests for classification in ecology, Ecology 88 (11) (2007) 2783–2792.

[5] A. M. Prasad, L. R. Iverson, A. Liaw, Newer classification and regression tree techniques: bagging and random forests for ecological prediction, Ecosystems 9 (2) (2006) 181–199.

[6] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, Real-time human pose recognition in parts from single depth images, Communications of the ACM 56 (1) (2013) 116–124.

[7] M. J. Kane, N. Price, M. Scotch, P. Rabinowitz, Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks, BMC bioinformatics 15 (1) (2014) 276.

[8] G. Dudek, Short-term load forecasting using random forests, in: Intelligent Systems' 2014, Springer, 2015, pp. 821–828.

[9] A. Lahouar, J. Ben Hadj Slama, Random forests model for one day ahead load forecasting, in: IREC2015 The Sixth International Renewable Energy Congress, 2015, pp. 1–6.

[10] A. Fischer, L. Montuelle, M. Mougeot, D. Picard, Statistical learning for wind power: a modeling and stability study towards forecasting, Wind Energy 20 (12) (2017) 2037–2047.

[11] J. Moon, Y. Kim, M. Son, E. Hwang, Hybrid short-term load forecasting scheme using random forest and multilayer perceptron, Energies 11 (2018).

[12] L. Breiman, J. Friedman, C. Stone, R. Olshen, Classification and Regression Trees, The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis, 1984.

[13] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[14] B. Goehry, Random forests for time-dependent processes, to appear, ESAIM: Probability and Statistics (2020). doi:10.1051/ps/2020015.

[15] B. Efron, Bootstrap methods: Another look at the jackknife, Ann. Statist. 7 (1) (1979) 1–26.

[16] C. Miller, F. Meggers, The building data genome project: An open, public data set from non-residential building electrical meters, Energy Procedia 122 (2017) 439 – 444.

[17] C. Cordeiro, M. Neves, Forecasting time series with boot. expos procedure, REVSTAT-Statistical Journal 7 (2) (2009) 135–149.

[18] C. Bergmeir, R. J. Hyndman, J. M. Benítez, Bagging exponential smoothing methods using stl decomposition and box–cox transformation, International journal of forecasting 32 (2) (2016) 303–312.

[19] G. Cavaliere, D. N. Politis, A. Rahbek, P. Bertail, S. Clémençon, J. Tressou, et al., Recent developments in bootstrap methods for dependent data, Journal of Time Series Analysis 36 (3) (2015) 462–480.

[20] E. Carlstein, The use of subseries values for estimating the variance of a general statistic from a stationary sequence, Ann. Statist. 14 (3) (1986) 1171–1179.

[21] H. R. Kunsch, The jackknife and the bootstrap for general stationary observations, Ann. Statist. 17 (3) (1989) 1217–1241.

[22] R. Y. Liu, K. Singh, Moving blocks jackknife and bootstrap capture weak dependence, in: Exploring the limits of bootstrap (East Lansing, MI, 1990), Wiley Ser. Probab. Math. Statist. Probab. Math. Statist., Wiley, New York, 1992, pp. 225–248.

[23] D. N. Politis, J. P. Romano, A circular block-resampling procedure for stationary data, in: Exploring the limits of bootstrap (East Lansing, MI, 1990), Wiley Ser. Probab. Math. Statist. Probab. Math. Statist., Wiley, New York, 1992, pp. 263–270.

[24] M. Wright, A. Ziegler, ranger: A fast implementation of random forests for high dimensional data in c++ and r, Journal of Statistical Software, Articles 77 (2017) 1–17.

[25] B. Goehry, Y. Goude, P. Massart, J.-M. Poggi, Aggregation of multi-scale experts for bottom-up load forecasting, IEEE Transactions on Smart Grid 11 (3) (2019) 1895–1904.