REVSTAT – Statistical Journal
Volume 21, Number 2, April 2023, 283–302
https://doi.org/10.57805/revstat.v21i2.400

Random Forests for Time Series

Authors: Benjamin Goehry ✉
– Laboratoire de Mathématiques d'Orsay, CNRS, Université Paris-Saclay,
Faculté des Sciences d'Orsay, Bâtiment 307, 91405 Orsay, France
benjamin.goehry@gmail.com

Hui Yan
– EDF Lab,
7 bd Gaspard Monge, 91120 Palaiseau, France
hui.yan@edf.fr

Yannig Goude
– EDF Lab & Laboratoire de Mathématiques d'Orsay, CNRS,
Université Paris-Saclay, Orsay, France
yannig.goude@edf.fr

Pascal Massart
– Laboratoire de Mathématiques d'Orsay, CNRS,
Université Paris-Saclay, Orsay, France
pascal.massart@math.u-psud.fr

Jean-Michel Poggi
– University Paris & Laboratoire de Mathématiques d'Orsay, CNRS,
Université Paris-Saclay, Orsay, France
jean-michel.poggi@math.u-psud.fr

Received: July 2020 Revised: November 2021 Accepted: November 2021

Abstract:

- Random forests are a powerful learning algorithm. However, when dealing with time series, the time-dependent structure is lost, assuming the observations are independent. We propose some variants of random forests for time series. The idea is to replace standard bootstrap with a dependent block bootstrap to subsample time series during tree construction. We present numerical experiments on electricity load forecasting. The first, at a disaggregated level and the second at a national level focusing on atypical periods. For both, we explore a heuristic for the choice of the block size. Additional experiments with generic time series data are also available.

Keywords:

- *block bootstrap; random forests; regression; time series.*

AMS Subject Classification:

- 62M10, 62P30.

✉ Corresponding author.

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

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

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. A binary tree can be associated to the input space partitioning. Each node corresponds to a test matching how the input space was cut. An illustration is given in Figure 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.

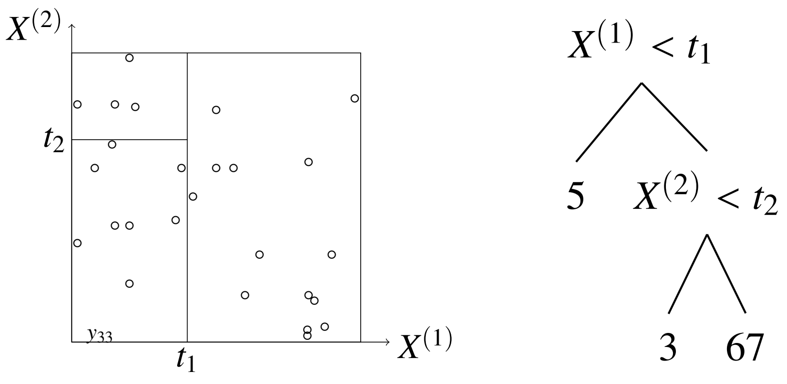

Figure 1: A partitioning of $[0,1]^2$ and the associated binary tree.

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 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 dependence structure. We illustrate this phenomenon for a dataset from [16] which is described in Section 3.1. We observe in Figure 2 the original load over the month of January. Using the standard bootstrap we obtain the series in Figure 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 Figure 4.

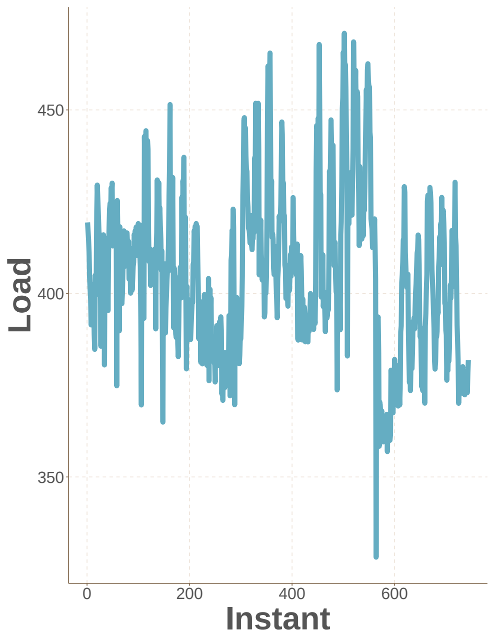

Figure 2: Original load hourly sampled.

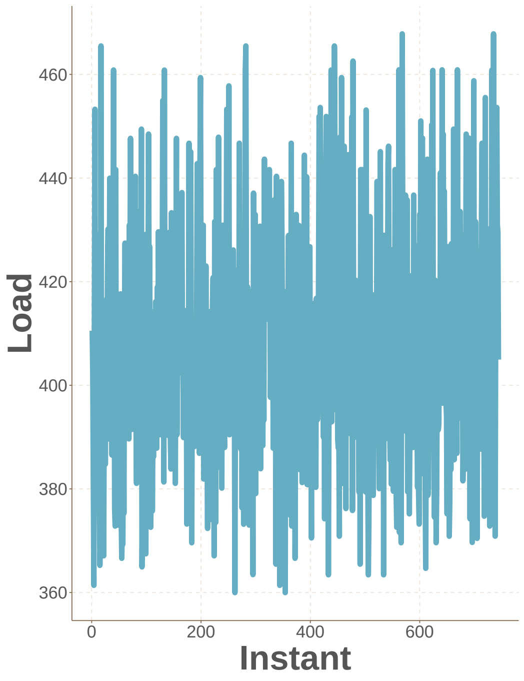

Figure 3: Bootstrapped load.

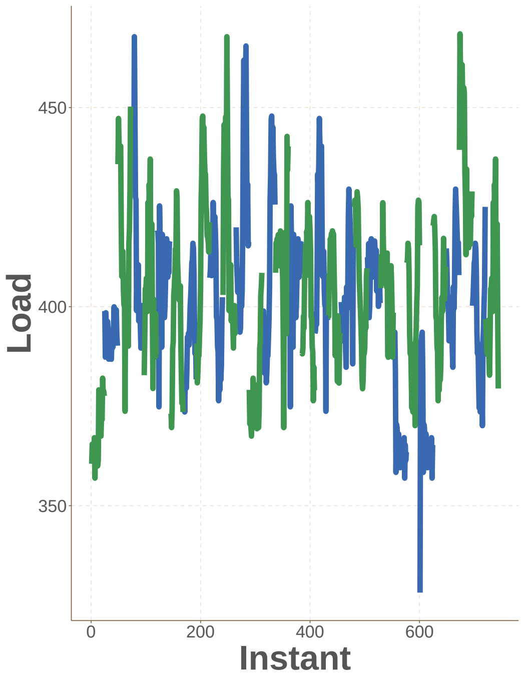

Figure 4: Block bootstrapped load with block size of 24h.

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. A companion paper [19] explores experimentally the value of bagging for time series forecasting. More generally, we refer to the special issue presented in [20] for more details about the recent developments in bootstraps methods for dependent data.

Our strategy is mainly motivated by the results on random forests in the time-dependent case in [14], proven using a block decomposition on the entries $(X_i, Y_i)_{1\leq i\leq n}$. The proofs rely on a lemma from [21] that shows that the blocks are close to being independent, under the condition that the block length is well-chosen. But it should be noted that after obtaining the bootstrap sample, the procedure to build a tree is unchanged and flipping the data after bootstrap will not change the resulting tree. The data are, in that sense and at this stage, considered to be exchangeable since the splitting criterion is unchanged and since it does not take into account the time dependence between the observations. We then try to make the data, before this stage, as much compatible with the underlying independence hypothesis.

A typical example of weak dependence is the m-dependent case, for which considering block bootstrap of length at least m allows to recover exchangeability. In the general weak dependence case, it is reasonable to consider that performing block bootstrap with a suitably chosen block-length could make the data more compatible with the exchangeable hypothesis. The aim of this work is to show that, based on the theoretical work in [14], 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 dependence structure. This intuition is supported by the experiment reported in Appendix 2 (with time shuffling) illustrating that preserving the temporal structure is, at least empirically, beneficial.

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 [22]: 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 in the following way

$$B_b = \big((X_{(b-1)l_n+1}, Y_{(b-1)l_n+1}), ..., (X_{bl_n}, Y_{bl_n})\big), \qquad b = 1, ..., B.$$

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

Moving block bootstrap. [23] and [24] 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 $B_{i,l_n} = ((X_i, Y_i), ..., (X_{i+l_n-1}, Y_{i+l_n-1}))$ the block of size l_n beginning with the observation (X_i, Y_i) for $i \in \{1, ..., n-l+1\}$. The procedure then consists to draw randomly K indices $(I_j)_{1\leq k\leq K}$ uniformly on the set $\{1, ..., n-l_n+1\}$ and associate one block to each index, $(B_{I_k})_{1\leq k\leq K}$. The bootstrap set is then defined as $\mathcal{D}_n^\star = (B_{I_1}, ..., B_{I_K})$.

Circular block bootstrap. When studying the moving block bootstrap we can note that less weight is given to the endpoints 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 [25] introducing the so-called *circular block bootstrap*. The idea is to wrap the time series writing $X_i := X_{i_n}$ where $i_n = i \bmod n$, $X_0 := X_n$ and then use the same procedure as in the moving block bootstrap where the index I is drawn uniformly on the set $\{1, ..., n\}$ 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 variants recalled in Section 2.1.

Note that the proposed algorithm only considers the dependence during the bootstrapping phase, directly on the entries $(X_i, Y_i)_{1 \le i \le n}$. Once the bootstrap sample is drawn the splitting is done as in the independent case. The adapted algorithm is found in Algorithm 1 underlining the modification with respect to the original random forest procedure.

```
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 $\tau_n$
for $j \leftarrow 1$ to $M$ do
    Construct the $j$th tree:
    • Draw $\alpha_n \le n$ observations $\underline{\text{using a block bootstrap variant with parameter } l_n}$.
    • 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.

Note that here, we consider the bootstrap directly on the entries $(X_i, Y_i)_{1 \le i \le n}$, and thus keeping the black box design of the random forests. Even if the time series nature of the data is forgotten after the bootstrap step, it should be noted that to include the time as a dependent variable could provide an indirect way to weakly take into account, at some extent, the temporal nature of the data. Works on blocks bootstraps in the forecasting literature presented in Section 1 use generally the block bootstrap on the residuals after removing trends and seasonality. However, using such a procedure in our experiments (by bootstrapping the residuals of a pilot random forest) led to worse performance and further explain our approach.

2.3. Block 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 m-th 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 j-th variable are randomly permuted in the OOB sample and we 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 m-th tree, we denote by $\widetilde{errOOB_m^j}$ the OOB_m sample's error when the j-th variable is permuted, then the importance of the variable $X^{(j)}$ is defined by

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

The higher the increase in the prediction error after the permutation of the j-th variable in the out-of-bag observations, the more important the variable is. However, if the permutation of $X^{(j)}$ does not 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 dependence 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 does not 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 l_n and denote by B_m^* the blocks in the out-of-observations for the m-th tree. In order to compute the importance of the j-th 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 j-th variable denoted by $\overline{errOOB_m^j}$. The block variable importance for the j-th variable is then defined by

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

The out-of-bag observations stemming from the block bootstrap with parameter l_n are not necessarily composed of blocks of the size l_n. 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 as follows. The three following cases are exclusive. First, 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. Second, 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. Finally, 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.

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.

We run the experiments by implementing the extra features we propose in this paper as an extension of the R package *ranger* [26], 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`. Additional experiments with time series data are performed and the results can be found in the same github repository as our modified R package, omitted here for brevity reasons.

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 Figure 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.

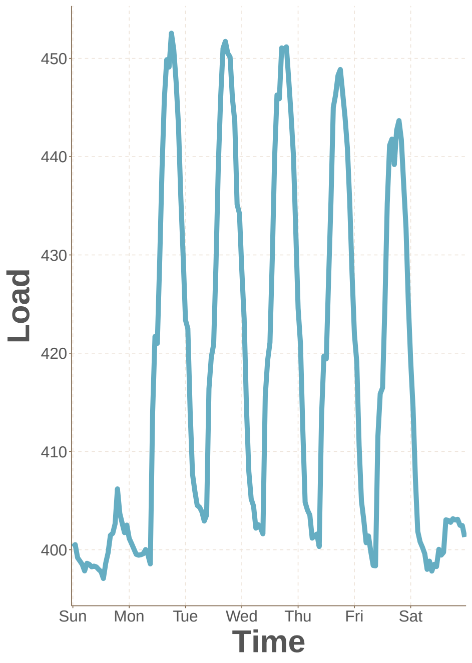

Figure 5: Weekly profile hourly sampled of the UnivLab Patrick dataset.

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 [27] in which they also considered random forests with a similar model for the same kind of problem. This results in the model described in (1.1) with X_t of the form

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

where:

- Temp_t corresponds to the temperature at instant t;
- $\mathrm{Schedule}_t$ take three values: Regular, Break, Holiday;
- Hour_t corresponds to the hour of the day at instant t;
- $\mathrm{InstantWeek}_t$ corresponds to the hour in the month;
- $\mathrm{DayType}_t$ corresponds to the day of the week;
- Time_t corresponds to the day of the year divided by 366.

The selected value for m_{try} according to the best performance on the validation set for the standard random forest 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 Figure 6. For the sake of comparison, the baseline $Y_t = Y_{t-24}$ has a RMSE of 19.43 on the test set. 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 Figure 7. We can find the same kind of figures for each m_{try} from 1 to 8 in Appendix 1 from Figures 15 to 22. 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.

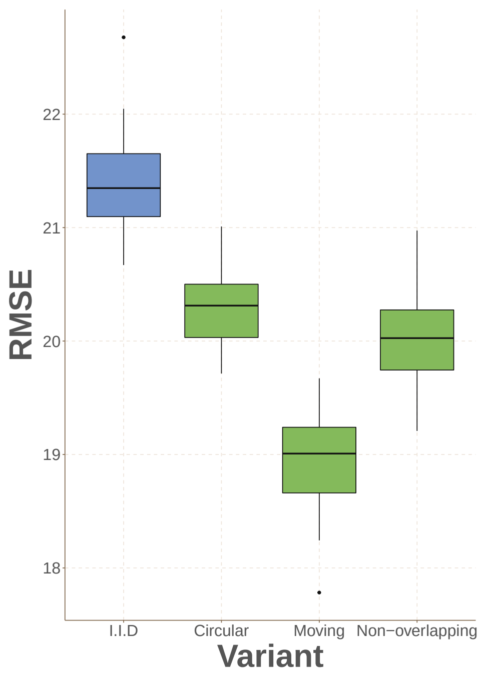

Figure 6: Performance of the different variants for $m_{try} = 2$, evaluated on the month of December of the UnivLab Patrick dataset.

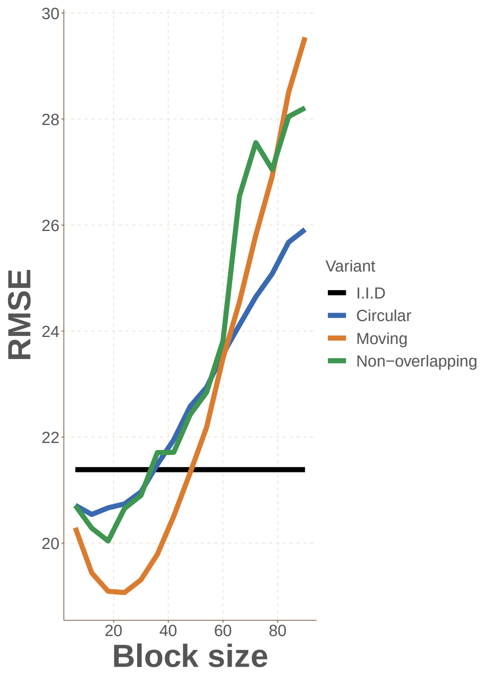

Figure 7: Performance of the variants for $m_{try} = 2$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

One may wonder if the block bootstrap mechanism really helps to take into account time dependence or if it is another underlying mechanism. In order to illustrate this point, we shuffled the instances in the training set. If it was another mechanism at play, we would have the same results as before. The results after shuffling the training set can be found in Appendix 2. We can clearly see that once the training set does not have the dependence structure, using the block bootstrap variants has basically the same behaviour as the standard random forests, regardless of the block length, and thus further confirms that the block bootstrap random forests take into account the dependence structure.

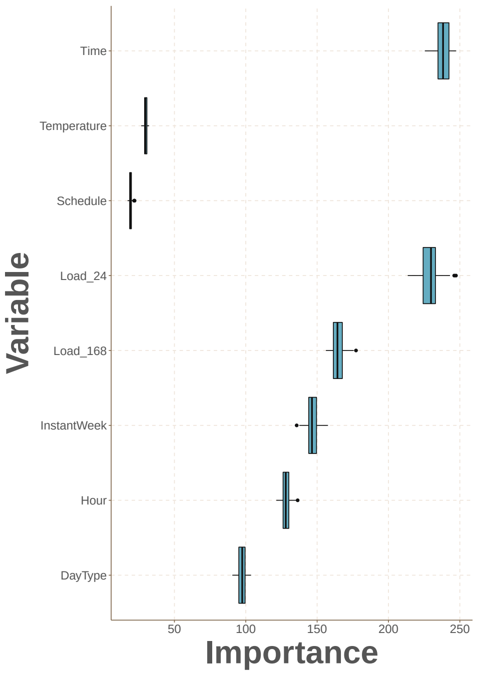

Figure 8: Variable importance moving bootstrap variant under the standard permutation on the UnivLab Patrick dataset.

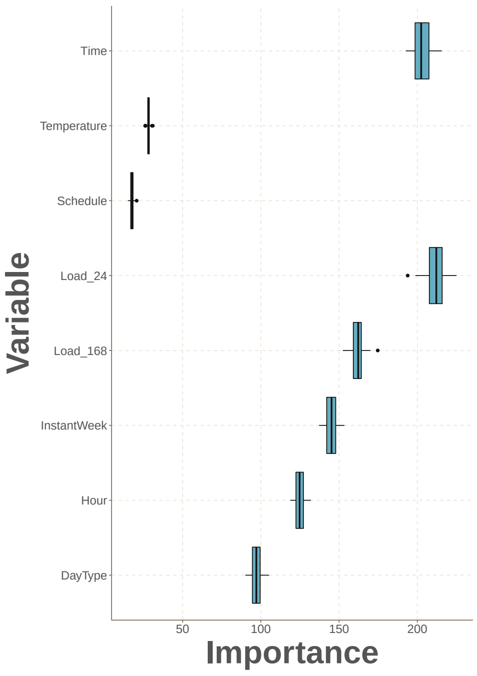

Figure 9: Block moving bootstrap variant importance with block size of 24h on the UnivLab Patrick dataset.

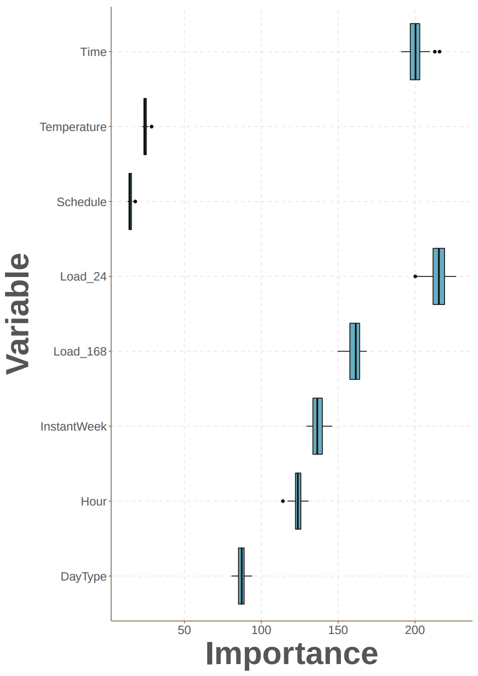

Figure 10: Variable importance non-overlapping variant under the standard permutation on the UnivLab Patrick dataset.

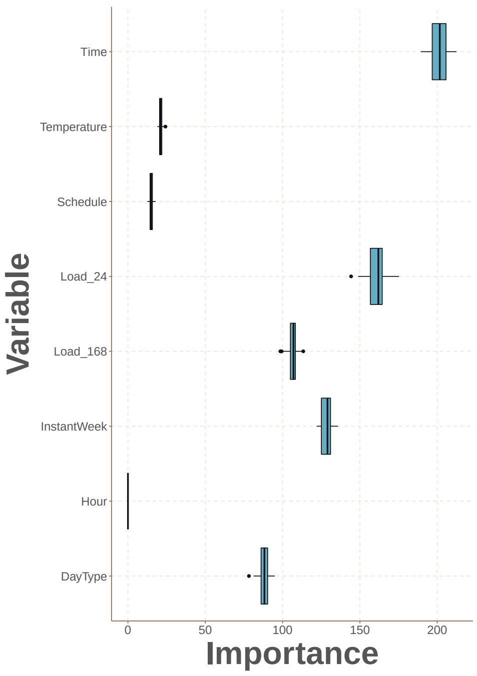

Figure 11: Block non-overlapping variant importance with block size of 24h on the UnivLab Patrick dataset.

Computing the variable importance for blocks of size 24 hours we obtain Figures 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.

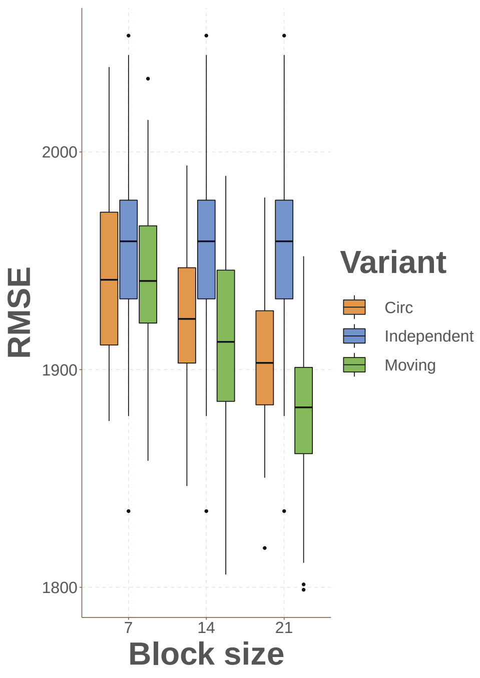

Figure 12: Performances evaluated on April 2016 on the French load forecasting problem of the different variants for three block length values.

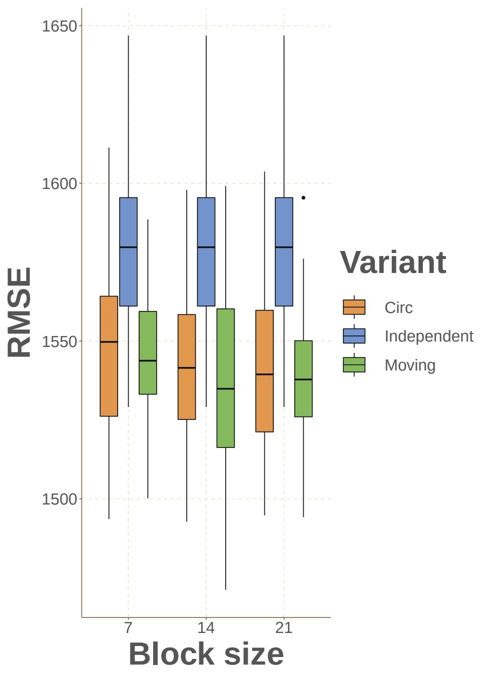

Figure 13: Performances evaluated on October 2016 on the French load forecasting problem of the different variants for three block length values.

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 (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.

The results are found, respectively for April and October 2016, in Figures 12 and 13. We observe that, for both months, 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.

3.3. Supplementary experiments

Further experiments are carried out with two forecasting competition data sets: quarterly and monthly series from M3 [28] (2184 series, 756 quarterly data and 1428 monthly data) and M4 [29] (4151 series, 402 quarterly data and 3749 monthly data) competitions to assess the performance of the proposed variants. Our main objective here is to compare the performance of the standard random forest and the block-bootstrap variants extensively on general time series data, instead of accessing how competitive the random forest algorithm itself is for these two data sets. Note that both stationary and non-stationary data are included in the data set whereas random forest cannot extrapolate and thus performs poorly on non-stationary data comparing to other baseline time series methods in the literature such as ARIMA models. The metrics we use for evaluation here are the normalized RMSE (NRMSE $= \frac{RMSE}{\sigma(serie_{train})}$, where σ is the standard deviation of the training part of the series), and the normalized difference in MAPE (NdMAPE $= \frac{\Delta(MAPE)}{MAPE_{i.i.d.}}$) where $\Delta(MAPE) = MAPE_{i.i.d.} - MAPE_{variant}$. Higher values indicate better results with variants.

As described in (1.1), let us denote by Y_t the series to be predicted at step t. Unlike the load forecasting application, only the frequency and time features are used. For monthly data, the frequency feature ranges from 1 to 12, which corresponds to the month. For quarterly, this feature is thus 1 to 4, and 1 stands for the first quarter. By regressing on time, we aim at estimating the trend and the seasonality components of each series. Including lags as explanatory variable would be a natural choice in time series forecasting tasks, here we choose not to do that to stay as far as possible from the exchangeability of the data.

We keep all other hyper parameters of the random forest identical to the standard i.i.d. version to compare the obtained results with those from the block bootstrap variants.

The only hyper parameter remains to be tuned is thus the block size. To be able to choose the block size automatically, we propose to set a general auto-correlation threshold for all series, to determine for each of them, the largest lag as the block length.

Better performance is achieved as shown in Figure 14 with the moving block variant on the monthly series (the same for the M4 data set). A Wilcoxon signed rank test confirms the gain with respect to the standard i.i.d. forest. We also observe in Table 1 that in general, higher auto-correlation thresholds lead to better results.

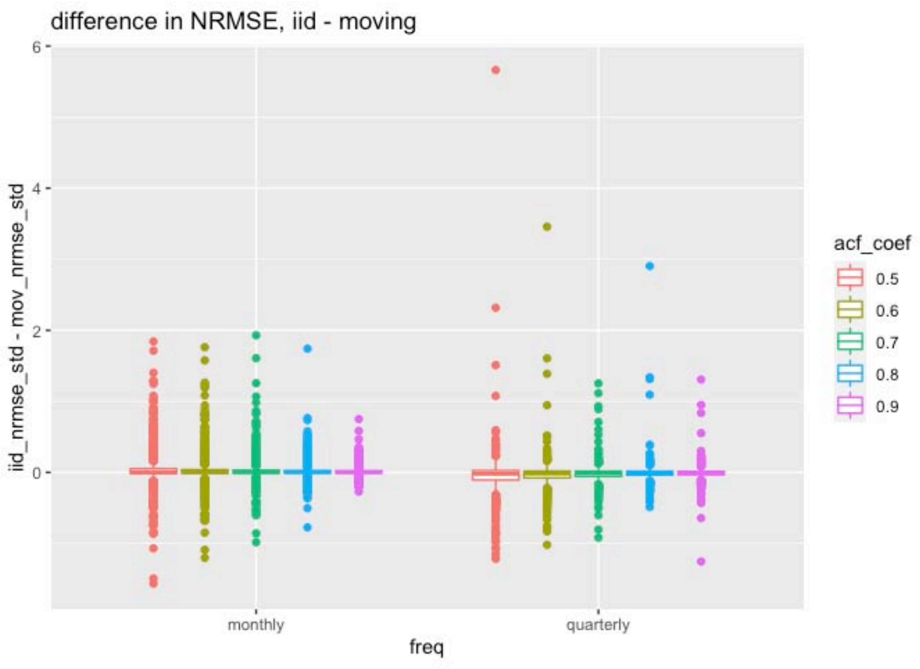

Figure 14: Difference in NRMSE of the standard random forest (i.i.d.) and the moving block variant (moving), for monthly and quarterly data, with different auto-correlation threshold values from 0.5 to 0.9, from the M3 data set.

Table 1: The percentage of cases where the block bootstrap variant outperforms the i.i.d. in terms of NdMAPE.

acf_coef	**M3**	**M4**
0.5	0.581	0.488
0.6	0.567	0.496
0.7	0.589	0.505
0.8	0.586	0.515
0.9	0.572	0.515

We choose to present our major results with a restricted number of graphs and statistics to conserve space. All the codes and other supporting materials can be found in the same GitHub repository as our implemented variants under the sub-directory benchmark_Mcomp.

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 [22, 23, 24, 25] 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 performed a detailed study on one specific field of application and an automatic extensive study was conducted on the time series of the M3 and M4 competitions. We illustrated the potential value of the random forests variants. We also showed that it could be useful to develop an 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.

APPENDIX 1

Performance of the variants for each given m_{try} from 1 to 8, when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset can be found from Figures 15 to 22.

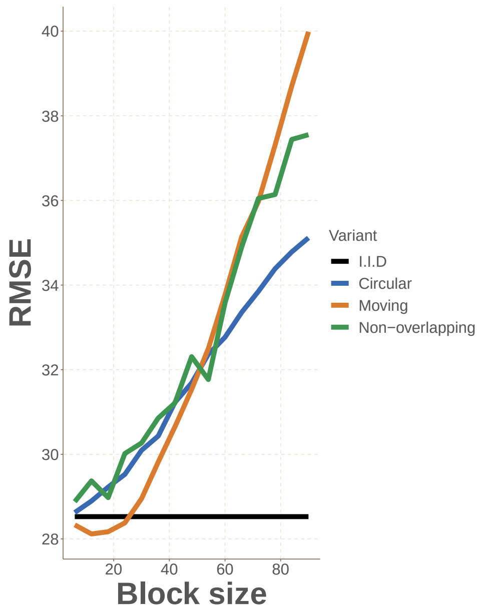

Figure 15: Performance of the variants for $m_{try}=1$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

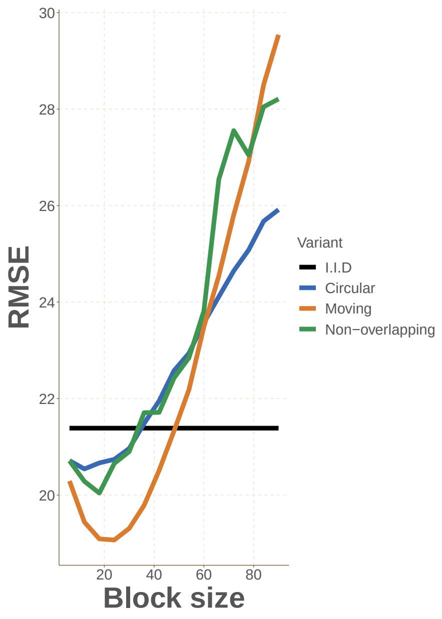

Figure 16: Performance of the variants for $m_{try}=2$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

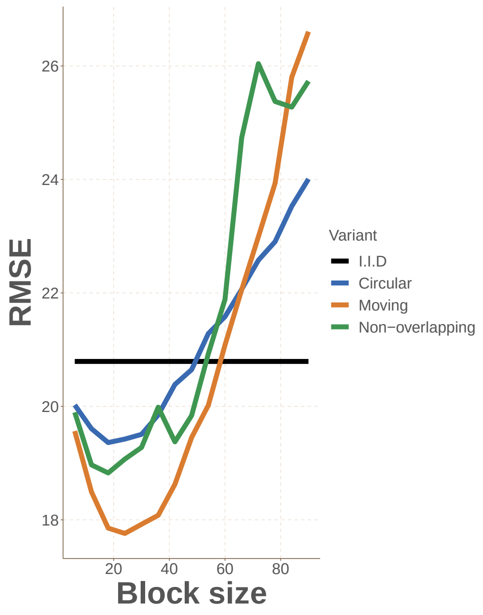

Figure 17: Performance of the variants for $m_{try}=3$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

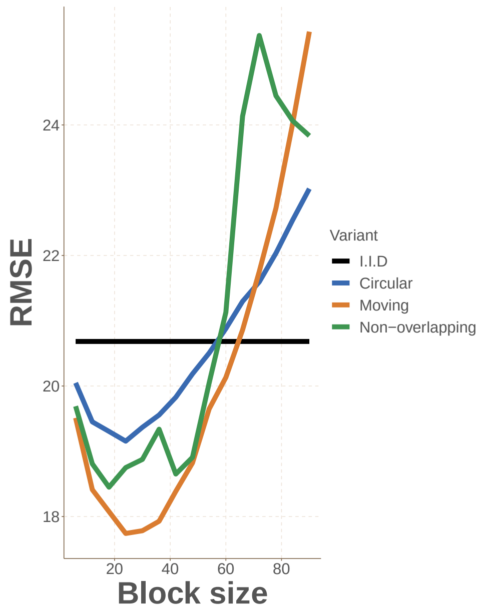

Figure 18: Performance of the variants for $m_{try}=4$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

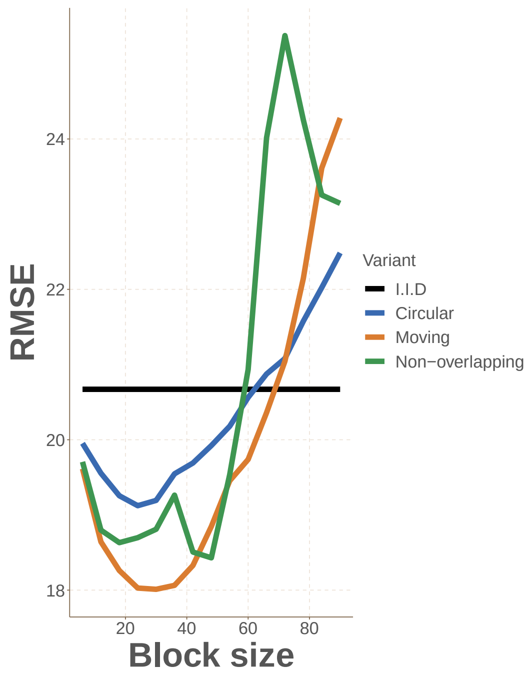

Figure 19: Performance of the variants for $m_{try} = 5$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

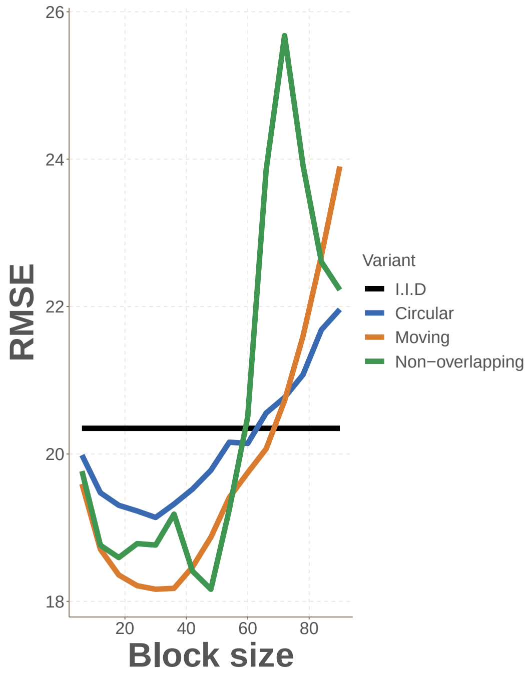

Figure 20: Performance of the variants for $m_{try} = 6$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

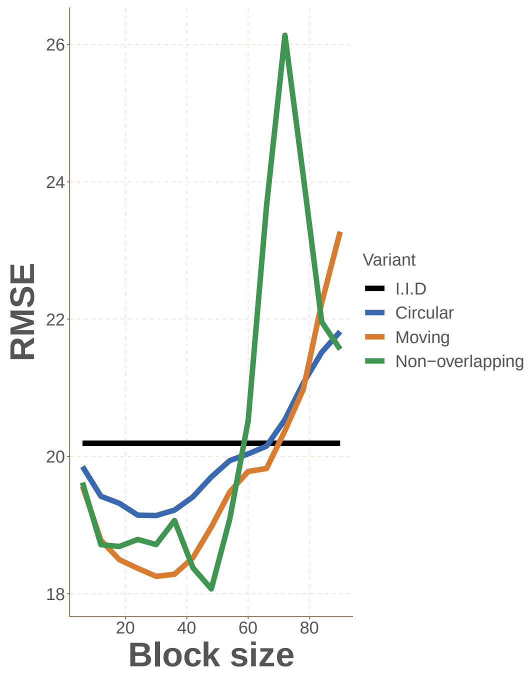

Figure 21: Performance of the variants for $m_{try} = 7$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

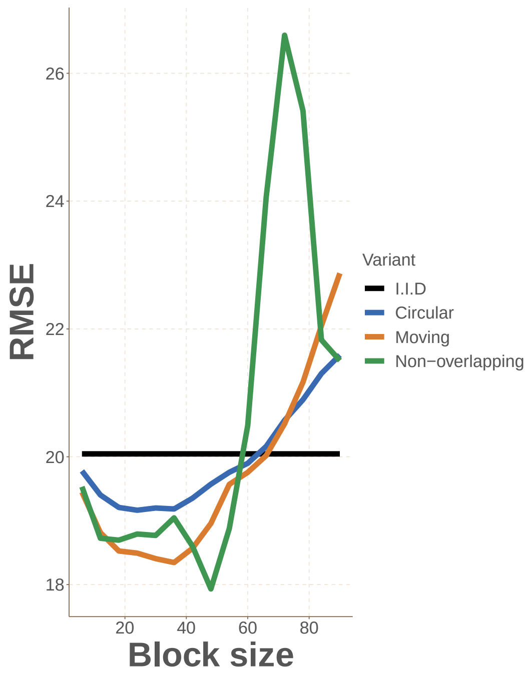

Figure 22: Performance of the variants for $m_{try} = 8$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

APPENDIX 2

Performance of the variants, when the observations in the training set are shuffled beforehand, for m_{try} equal to 1 and 2, when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset can be found from Figures 23 to 24. We have similar results for m_{try} from 3 to 8.

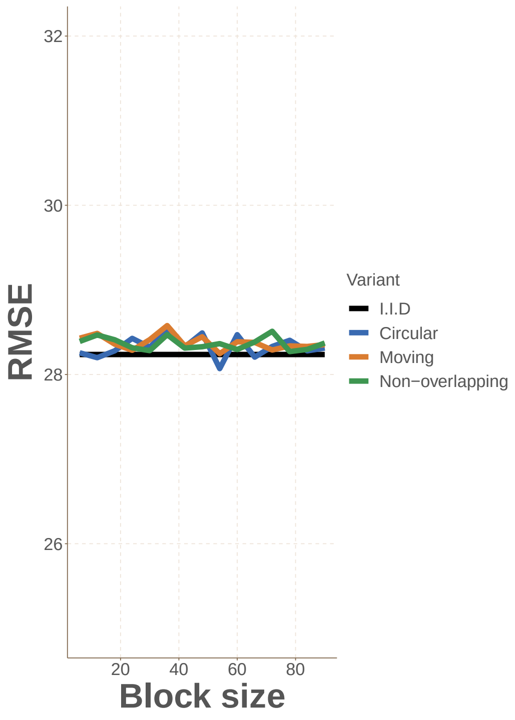

Figure 23: Performance of the variants when training set is shuffled for $m_{try}=1$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

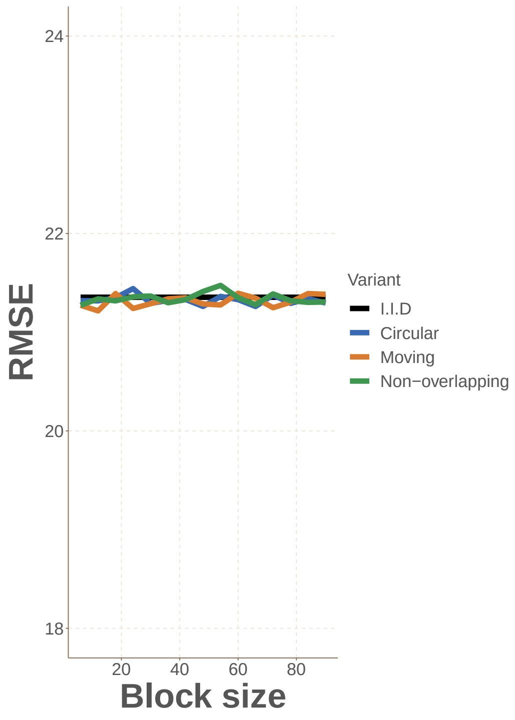

Figure 24: Performance of the variants when training set is shuffled for $m_{try}=2$ when the block size changes, evaluated on the month of December of the UnivLab Patrick dataset.

ACKNOWLEDGMENTS

The authors would like to thank the reviewer for their helpful comments and suggestions.

REFERENCES

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

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

[3] Svetnik, V.; Liaw, A.; Tong, C.; Culberson, J.C.; Sheridan, R.P. and Feuston, B.P. (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling, *Journal of Chemical Information and Computer Sciences*, **43**(6), 1947–1958.

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

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

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

[7] Kane, M.J.; Price, N.; Scotch, M. and Rabinowitz, P. (2014). Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks, *BMC Bioinformatics*, **15**(1), 276.

[8] Dudek, G. (2015). *Short-term load forecasting using random forests.* In "Intelligent Systems' 2014", Springer, pp. 821–828.

[9] Lahouar, A. and Ben Hadj Slama, J. (2015). *Random forests model for one day ahead load forecasting.* In "IREC2015 – The Sixth International Renewable Energy Congress", 1–6.

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

[11] Moon, J.; Kim, Y.; Son, M. and Hwang, E. (2018). Hybrid short-term load forecasting scheme using random forest and multilayer perceptron, *Energies*, **11**, 3283.

[12] Breiman, L.; Friedman, J.; Stone, C.J. and Olshen, R.A. (1984). *Classification and Regression Trees*, The Wadsworth and Brooks–Cole Statistics-Probability Series, Taylor & Francis.

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

[14] Goehry, B. (2020). Random forests for time-dependent processes, *ESAIM: PS*, **24**, 801–826.

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

[16] Miller, C. and Meggers, F. (2017). The Building Data Genome Project: an open, public data set from non-residential building electrical meters, *Energy Procedia*, **122**, 439–444.

[17] Cordeiro, C. and Neves, M. (2009). Forecasting time series with BOOT. EXPOS procedure, *REVSTAT – Statistical Journal*, **7**(2), 135–149.

[18] Bergmeir, C.; Hyndman, R.J. and Benítez, J.M. (2016). Bagging exponential smoothing methods using STL decomposition and Box–Cox transformation, *International Journal of Forecasting*, **32**(2), 303–312.

[19] Petropoulos, F.; Hyndman, R.J. and Bergmeir, C. (2018). Exploring the sources of uncertainty: why does bagging for time series forecasting work?, *European Journal of Operational Research*, **268**(2), 545–554.

[20] Cavaliere, G.; Politis, D.N. and Rahbek, A. (2015). Recent developments in bootstrap methods for dependent data, *Journal of Time Series Analysis*, **36**(3), 269–271.

[21] Yu, B. (1994). Rates of convergence for empirical processes of stationary mixing sequences, *The Annals of Probability*, **22**(1), 94–116.

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

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

[24] Liu, R.Y. and Singh, K. (1992). *Moving blocks jackknife and bootstrap capture weak dependence.* In "Exploring the Limits of Bootstrap (East Lansing, MI, 1990), Wiley Ser. Probab. Math. Statist.", Wiley, New York, pp. 225–248.

[25] Politis, D.N. and Romano, J.P. (1992). *A circular block-resampling procedure for stationary data.* In "Exploring the Limits of Bootstrap (East Lansing, MI, 1990), Wiley Ser. Probab. Math. Statist.", Wiley, New York, pp. 263–270.

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

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

[28] Makridakis, S. and Hibon, M. (2000). The M3-Competition: results, conclusions and implications, *International Journal of Forecasting*, (6), 451–476.

[29] Makridakis, S.; Spiliotis, E. and Assimakopoulos, V. (2018). The M4-Competition: results, findings, conclusion and way forward, *International Journal of Forecasting*, (34), 802–808.