Article:
Zhao, X, Barber, S, Taylor, CC et al. (2021) Interval forecasts based on regression trees for streaming data. Advances in Data Analysis and Classification, 15 (1). pp. 5-36. ISSN: 1862-5347

https://doi.org/10.1007/s11634-019-00382-7

Advances in Data Analysis and Classification manuscript No.
(will be inserted by the editor)

Interval Forecasts based on Regression Trees for Streaming Data

Xin Zhao[1,2] · **Stuart Barber**[2] · **Charles C Taylor**[2] · **Zoka Milan**[3]

Received: date / Accepted: date

Abstract In forecasting, we often require interval forecasts instead of just a specific point forecast. To track streaming data effectively, this interval forecast should reliably cover the observed data and yet be as narrow as possible. To achieve this, we propose two methods based on regression trees: one ensemble method and one method based on a single tree. For the ensemble method, we use weighted results from the most recent models, and for the single-tree method, we retain one model until it becomes necessary to train a new model. We propose a novel method to update the interval forecast adaptively using root mean square prediction errors calculated from the latest data batch. We use wavelet-transformed data to capture long time variable information and conditional inference trees for the underlying regression tree model. Results show that both methods perform well, having good coverage without the intervals being excessively wide. When the underlying data generation mechanism changes, their performance is initially affected but can recover relatively quickly as time proceeds. The method based on a single tree performs the best in computational (CPU) time compared to the ensemble method. When compared to ARIMA and GARCH modelling, our methods achieve better or similar coverage and width but require considerably less CPU time.

Keywords Ctree · MODWT · wavelet · Liver Transplantation · data stream

Mathematics Subject Classification (2000) MSC 62M10

Xin Zhao
Tel.: +86-18753341710
E-mail: mmxinzhao@hotmail.com
1. School of Mathematics, Southeast University, Nanjing, 210096 China.
2. School of Mathematics, University of Leeds, Leeds LS2 9JT, U.K.
3. King's College Hospital Trust, London SE5 9RS, U.K.

Acknowledgements

Xin Zhao is grateful for the financial support of the China Scholarship Council (CSC) during this research, which was completed during her PhD studies at the University of Leeds.

Advances in Data Analysis and Classification manuscript No.
(will be inserted by the editor)

Interval Forecasts based on Regression Trees for Streaming Data

Received: date / Accepted: date

Abstract In forecasting, we often require interval forecasts instead of just a specific point forecast. To track streaming data effectively, this interval forecast should reliably cover the observed data and yet be as narrow as possible. To achieve this, we propose two methods based on regression trees: one ensemble method and one method based on a single tree. For the ensemble method, we use weighted results from the most recent models, and for the single-tree method, we retain one model until it becomes necessary to train a new model. We propose a novel method to update the interval forecast adaptively using root mean square prediction errors calculated from the latest data batch. We use wavelet-transformed data to capture long time variable information and conditional inference trees for the underlying regression tree model. Results show that both methods perform well, having good coverage without the intervals being excessively wide. When the underlying data generation mechanism changes, their performance is initially affected but can recover relatively quickly as time proceeds. The method based on a single tree performs the best in computational (CPU) time compared to the ensemble method. When compared to ARIMA and GARCH modelling, our methods achieve better or similar coverage and width but require considerably less CPU time.

Keywords Ctree · MODWT · wavelet · Liver Transplantation · data stream

Mathematics Subject Classification (2000) MSC 62M10

1 Introduction

In data stream analysis, we build models to capture information hidden in the data, either for description or prediction. Streaming regression trees have been widely developed to capture such information. For many applications, forecasting the response value at a given time in the future is the primary task; usually this is done simply as a point forecast. However, sometimes an interval forecast is also required and this is often more useful than the point forecast as the interval includes some information about uncertainty. We use the term "interval forecast" to refer to an interval that should usually include the observed value of the streaming variable at the specified time. The interval construction method we use is inspired by Appice and Ceci (2006), who use count-based and normal distribution-based procedures. In this paper, we employ a count-based procedure using quantiles for interval construction, rather than relying on any distributional assumption or approximation.

Our interest in this problem is motivated by a real data example. During surgery, a patient's heart rate is continuously monitored. If we can reliably predict undesirable heart rates, even just a minute ahead, surgeons have some time for preparation, which could potentially save lives. In this circumstance, point prediction for an exact heart rate value is of limited use. Surgeons pay more attention to the range of the heart rate (whether in the normal range or not), and a method designed for interval forecast is more directly applicable for range monitoring. We also apply our method to a financial time series. Other potential applications include industrial process monitoring, exchange rates, social media activity or customer behaviour data.

Forecasting can be based on statistical time series models like ARIMA or GARCH, but these models require structural and distributional assumptions. In reality, these assumptions may not be satisfied, and cannot be verified for future data. Data mining methods like decision trees do not need such assumptions. Moreover, trees are often simpler to interpret than complex parametric models and can provide simple decision rules to use when this is required. Since trees are generally not based on an underlying statistical or probabilistic model, there is no distributional means of constructing confidence intervals based on regression trees. However, the set of observations in each terminal node can then be used to construct an interval estimate. For these reasons, we use a tree as our regression model. We compare our tree-based forecasts with those produced by ARIMA and GARCH models.

We use the coverage rate (the proportion of test values falling into the interval forecast) of the interval forecast as the main measure of how well our methods are performing. If almost all the observed values are in the interval forecasts, we regard the method as successful. We also consider interval width as a secondary measure of performance (narrow intervals being more useful than wider ones). Computational load is also of obvious practical importance.

The rest of this paper is arranged as follows. In Section 2, we review key concepts in regression tree modelling of streaming data. We introduce our methodology in Section 3, and apply it to simulated data in Section 4 with

model-based ARIMA and GARCH forecasting methods for comparison, as well as exploring the effects of noise and drift. After that we apply it to medical and financial data sets followed by comparisons with ARIMA and ARMA-GARCH in Section 5 and Section 6 respectively. Some concluding comments appear in Section 7. All computations were performed in R (R Core Team 2014), using the packages partykit (Hothorn and Zeileis 2015) for regression trees, waveslim (Whitcher 2013) for wavelet decomposition, forecast (Hyndman 2017; Hyndman and Khandakar 2008) for ARIMA, rugarch (Ghalanos 2014) and fGarch (Wuertz et al. 2019) for GARCH and ARMA-GARCH.

2 Related work

The first consideration is how to utilize long term variable information. Since we are forecasting in the context of streaming data, current data can depend on old data. Building models using only the most recent data seems unwise, so storing old information in an efficient way is important. One way of doing this is via a wavelet transform, which can pick out long term averages and short term fluctuations. Instead of using the standard decimated discrete wavelet transform (DWT), we use the maximal overlap discrete wavelet transform (MODWT; see, for example, Percival and Walden 2000), as it is not constrained by time series length T_n and each time point is represented at all resolution levels of the MODWT. We have previously found that wavelet transformed variables can give better classification performance than the original untransformed data (Zhao et al. 2018).

The second issue is how to deal with streaming data. Various regression tree models have been designed for streaming data, many of which are implemented in the Massive Online Analysis (MOA) open source framework for data stream mining (Bifet et al. 2010). In MOA, the Hoeffding Tree family (Domingos and Hulten 2000; Jin and Agrawal 2003; Bifet et al. 2009; Pfahringer et al. 2007) is implemented, along with some more recently developed methods (Duarte et al. 2016; Ikonomovska et al. 2015; 2011). However, performance of these methods is assessed by accuracy of the predicted values as point estimates while we are more interested in performance based on interval forecast coverage. Without considering trees, there are other methods which use neural networks to obtain prediction intervals such as Shrestha and Solomatine (2006) and Quan et al. (2014). These methods can be applied to streaming data as well, but lack the straightforward interpretation of tree-based models.

Thirdly, we must decide how to detect and respond to concept drift. The data generation mechanism can be referred to as "concept". This concept can remain stable or change over time, for example, when the data generation background changes. If it is stable, models built now can still be used for prediction indefinitely. If it is not stable, we say concept drift happens, either gradually or abruptly. In this scenario, statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways. This leads

to poor prediction performance, as the model built based on the old concept is no longer suitable for prediction of the target variable.

Concept drift detection tools are generally based on prediction accuracy. The Drift Detection Method (DDM) (Gama et al. 2004) monitors the probability of error at time t, denoted as p_t with standard deviation s_t. The Early Drift Detection Methodology (EDDM Baena-García et al. 2006) was developed as an extension of DDM, and is more suitable for slow moving gradual drifts, where DDM previously failed (Sethi and Kantardzic 2017). The Statistical Test of Equal Proportions (STEPD) (Nishida and Yamauchi 2007) computes the accuracy of a batch of recent predictions and compares it with the overall accuracy from the beginning of the stream, using a chi-squared test to check for deviation. An incremental approach, Concept Drift Detection (ECDD), was proposed by Ross et al. (2012). Some window based methods like the Adaptive Windowing (ADWIN) algorithm (Bifet and Gavaldà 2007; Yoshida et al. 2011) and SeqDrift2 (Sobhani and Beigy 2011) detect whether sub-windows have significant differences in terms of predictive accuracy.

We wish to base our response to concept drift on the coverage of our interval forecasts rather than the point prediction accuracy. An interval forecast which covers most of the data suggests that the model in use is a good one which should continue to be used as long as it remains effective. The three simplest ways for a time series to change are to change mean value with variance fixed; change variance with mean value fixed; or change both mean value and variance. When variance alone changes, we can widen or shrink our interval forecast to cover future data more efficiently. But when the mean value changes with or without variance change, we need to consider building a new model. For an interval forecast from a regression tree, we use the sample quantiles of the observations falling into the relevant terminal node of the tree. So when the mean of the data changes substantially, our tree will no longer be able to produce suitable forecasts, being biased towards historical rather than current values. Our criterion is that when too much test data falls outside our forecast intervals, then we build a new model with the most recent batch of data to update or replace the current model.

Finally, when concept drift has happened, how should we respond? There are many ways to improve the model when concept drift is detected. Some authors rebuild or update the model using the new batch of data (Gholipour et al. 2013), while other remove some poorly performing nodes and grow new ones with the latest batch (Domingos and Hulten 2000; Bifet and Gavaldà 2009). In ensemble methods, some drop the worst model and replace it with a new model but with the other models unchanged. Inspired by these ideas, we develop two methods, one ensemble method and one based on a single tree.

3 Methodology

In this section, we first introduce regression trees and the maximal overlap discrete wavelet transform (MODWT). Next, we propose two methods to create

interval forecasts for streaming data: a single-tree and an ensemble approach. We then describe an approach which adaptively adjusts the interval forecasts according to the changing characteristics of the data, and discuss criteria for model retraining. Finally, we define our measures of model performance.

3.1 Background

3.1.1 Outline

We use regression tree models for forecasting time series data $\{y_t\}$, denoting the predicted value h steps ahead at time t as

$$\hat{y}_{t+h} = f(y_t, y_{t-1}, \ldots, y_{t-\gamma+1}).$$

This prediction uses the last γ observations to predict y_{t+h}. We assume that we have initial training data consisting of observations up to time T, of which the last h will not be used for prediction within the training set. A schematic matrix representation is

$$A = \begin{bmatrix} y_1 & y_2 & \cdots & y_\gamma \\ y_2 & y_3 & \cdots & y_{\gamma+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{T-h-\gamma+1} & y_{T-h-\gamma+2} & \cdots & y_{T-h} \end{bmatrix} \longrightarrow \begin{bmatrix} y_{h+\gamma} \\ y_{h+\gamma+1} \\ \vdots \\ y_T \end{bmatrix}. \tag{1}$$

One thing to note is we are not using all of the matrix A to predict each future y_t, but using each line to predict each time point. The matrix just represents a batch of data. Since we wish to use the wavelet transform of the left matrix A, we require a longer time range. So we use longer previous data for the wavelet transform and then truncate the series of wavelet coefficients to have the same time range as A; we refer to the matrix of wavelet coefficient series as W. Then we can use each line in W to predict $y_{h+\gamma}, \ldots, y_T$.

3.1.2 Regression trees

The regression tree model we use is the Conditional Inference Tree (ctree) method (Hothorn et al. 2006b;a), although our approaches could be applied to other regression tree methods. Ctree estimates a regression relationship by binary recursive partitioning in a conditional inference framework. The algorithm works as follows:

1. The association between each potential predictor and the response is quantified by computing a p-value. The variable with the lowest p-value is selected for splitting, unless the lowest p-value exceeds a pre-specified threshold and splitting is stopped. The threshold used in this paper is 0.05.
2. Implement a binary split in the selected input variable. The split itself can be established by any split criterion, including those used in CART, CHAID and so on.

3. Recursively repeat steps 1) and 2) on each data subset formed by the splits made so far until all nodes are sufficiently homogeneous that no further splitting is indicated.

The method of computing the p-values used depends on the nature of the response and potential predictor variables, and can include those used for the Spearman correlation test, the Wilcoxon-Mann-Whitney test, the Kruskal-Wallis test or permutation tests based on ANOVA statistics or correlation coefficients (Hothorn et al. 2006b). These p-values are commonly modified with a multiple testing correction, such as Bonferroni adjustment, which we have used in our analyses.

3.1.3 Wavelet transform

Here, we give a brief introduction to the wavelet tools we use in our method. For more details, see, for example, Percival and Walden (2000). Define the Haar scaling function ϕ and wavelet function ψ as

$$\phi(\tau) = \begin{cases} 1 & \tau \in [0,1) \\ 0 & \text{else,} \end{cases} \qquad \text{and} \qquad \psi(\tau) = \begin{cases} 1 & \tau \in [0,1/2) \\ -1 & \tau \in [1/2,1) \\ 0 & \text{else.} \end{cases}$$

By using dilation and translation, we obtain the non-decimated scaling function and wavelet at location l and resolution level j:

$$\phi_{j,l}(\tau) = 2^{j/2}\phi\big(2^j(\tau - l)\big) \qquad \text{and} \qquad \psi_{j,l}(\tau) = 2^{j/2}\psi\big(2^j(\tau - l)\big),$$

where $j = 0, 1, \ldots, J$, $J = \lfloor \log_2 T \rfloor$, and $l = 1, 2, \ldots, T$. Note that $\phi_{j,l}$ and $\psi_{j,l}$ are compactly supported on $I_{j,l} = \big[2^{-j}l, 2^{-j}(l+1)\big)$.

In the MODWT, the data are represented in terms of the functions $\phi_{j,l}$ and $\psi_{j,l}$. We compute scaling coefficients $s_{j,l}$ of our time series $y_1, \ldots, y_T$ as

$$s_{j,l} = \langle y_\cdot, \phi_{j,l} \rangle = \sum_{t=1}^{T} y_t \phi_{j,l}(t/T) = 2^{j/2} \sum_{t/T \in I_{j,l}} y_t,$$

and wavelet coefficients $d_{j,l}$ as

$$d_{j,l} = \langle y_\cdot, \psi_{j,l} \rangle = s_{j-1,l} - s_{j,l}.$$

Hence, coefficients of $\phi_{j,l}$ and $\psi_{j,l}$ represent local averages and contrasts, respectively, in the interval $I_{j,l}$. To interpret these coefficients, note that when j is small the corresponding scaling and wavelet functions are highly localized at that fine scale, representing brief transient effects. Conversely, when j is large, they represent lower frequency activity at a coarse scale.

Many other wavelet functions exist, but we have found the Haar basis to be effective in our analysis and their simplicity eases interpretation of the results. Other wavelet basis functions have more complex shapes, and while they may give better raw performance in particular data sets they lack the simple interpretation of the Haar wavelet.

After applying the wavelet transform, the scaling coefficients s and wavelet coefficients d are treated as variables which can be used for classification. We then have the $T \times 2J$ wavelet transformed data matrix

$$W = \text{MODWT}(A) = \left[\begin{array}{ccc|ccc} W_1^{d_1} & \cdots & W_1^{d_J} & W_1^{s_1} & \cdots & W_1^{s_J} \\ W_2^{d_1} & \cdots & W_2^{d_J} & W_2^{s_1} & \cdots & W_2^{s_J} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ W_T^{d_1} & \cdots & W_T^{d_J} & W_T^{s_1} & \cdots & W_T^{s_J} \end{array}\right],$$

and we use the columns of W as predictor variables in our regression trees.

3.2 Proposed Methods

We propose two forecasting methods: an ensemble model and a single-tree model. Initially, at time t_0, we set $T = t_0$ in Equation (1) and obtain our starting model. We then use data y_t for $t = t_0 - h + 1, \ldots, t_0$ to obtain predictions for $t = t_0 + 1, \ldots, t_0 + h$. From then on, forecasting is done continuously, but after each batch of data, we review and update our models.

We now describe the stages of our method. As some of the steps for the ensemble model and single-tree model are the same, these steps are described together, with only the differences stated separately. At Stage 1, we use the initial t_0 observations for initial training and forecasting. We regard each subsequent batch of h observations afterwards as a new stage in the process, so at Stage k, we have data y_t, $t = 1, 2, \ldots, t_0 + kh$. At each Stage $k = 1, 2, 3, \ldots$, we go through the following steps. (Some details of the model and forecast updating process are deferred to the following section.)

1. $(k+1)^{st}$ prediction
   - $(k+1)^{st}$ *prediction (ensemble method)*
     For each new observation at Stage k, continuously use the ensemble to get a weighted forecast for new data at Stage $k+1$. Denote a prediction by the vector $P = (L, \hat{y}, U)$, including the predicted value $\hat{y}$ and forecast interval limits L, U. Let $P_{(1)}$ be the forecast from the first (oldest) model (a single tree), $P_{(2)}$ the next oldest and so on. Hence our ensemble is initiated by a single tree which is then joined by further trees. Let s and m be the current and maximum number of trees in the model, respectively. Then the combined forecast from an ensemble of sm models is $\bar{P}_{(s)}$, defined iteratively by

$$\begin{aligned} \bar{P}_{(2)} &= k_1 P_{(1)} + k_2 P_{(2)}, \\ \bar{P}_{(3)} &= k_1 \bar{P}_{(2)} + k_2 P_{(3)}, \\ &\vdots \\ \bar{P}_{(s)} &= k_1 \bar{P}_{(s-1)} + k_2 P_{(s)}, \end{aligned}$$

     where $k_1 + k_2 = 1$. Generally, $k_2 > k_1$ to ensure that the most recent tree is given more weight than older ones.

- $(k+1)^{st}$ *prediction (single method)*
  For each new observation at Stage k, use the latest model to get a prediction P_t for $t \in S_{k+1} = \{t_0 + kh + 1, \ldots, t_0 + (k+1)h\}$ until time reaches $t_0 + kh$.

2. k^{th} RMSE calculation
   Calculate the root mean squared error (defined later in Equation (2)) between the observed data from Stage k and the point forecasts of these data which were made in the previous Stage $k-1$.

3. k^{th} model updating
   If a new model was created at Stage $k-1$, update this latest model trained or updated from the last Stage $k-1$ by using the k^{th} RMSE as described later in §3.3.

4. $(k+1)^{st}$ model training
   - $(k+1)^{st}$ *model training (ensemble method):*
     We use data at Stage k to train a new model. If the total amount of data observed until now is greater than some upper limit T_Ω, then we use the latest T_Ω observations to emphasize more recent information and make the models more responsive to concept drift.
   - $(k+1)^{st}$ *model training (single method):*
     The current model, possibly based only on the original data, may become less relevant, showing bias or excess uncertainty due to concept drift, so we choose to train a new model when necessary. However, a new model is only trained when deemed necessary and otherwise we continue with the existing model. The decision on whether to train a new model is based on the coverage of our forecasts of the most recent batch of data. For the latest h observations, let

$$p_y = \max\left\{\frac{1}{h}\sum_{t\in S_k} I[\hat{y}_t > (y_t + \delta)], \frac{1}{h}\sum_{t\in S_k} I[\hat{y}_t < (y_t - \delta)]\right\},$$

     which will increase if our predictions are systematically above or below the observed values by a tolerance δ, or if the variance of either the predictions or the data-generation process increases. If p_y exceeds a threshold of, say, 90%, then we train a new model using these h observations to replace the old model.

This outlines the ongoing process of model updating which will continue as long as data are being observed. We now give more details of the construction and updating of our forecast intervals, which is largely the same for both ensemble and single-tree methods.

3.3 Construction and updating of interval forecasts

3.3.1 Forecast interval construction

For each terminal node, we use the 0.025 and 0.975 quantiles of the values in that node as an initial interval $[L', U']$. To make our forecast interval less sensitive to overfitting the training data, we enlarge it to

$$[L, U] = [L' - \alpha(U' - L'), U' + \alpha(U' - L')].$$

The initial shrinkage of the intervals serves to protect against any outliers in the original data leading to an unreasonably large interval. Subsequently enlarging the interval serves to reduce the influence of minor change in the distribution of future data. α can be used to trade off coverage and width. Larger values of the tuning parameter α lead to wider intervals, usually with correspondingly higher coverage but big width, while a much small alpha will have little effect. In that case, a range around [1, 3] of α is suggested. This tuning parameter is only applied to new models.

3.3.2 Updating forecast intervals

We adaptively adjust the width of our interval forecasts, depending on the coverage of the most recent point forecasts. For a general Stage k, comprising time points $t \in S_k$, the root mean squared error of the point forecasts of the h observations is

$$\text{RMSE}_k = \sqrt{\frac{1}{h} \sum_{t \in S_k} (\hat{y}_t - y_t)^2}. \tag{2}$$

Denote the forecast interval for observation y_t by $[L_t, U_t]$. Let p_U and p_L be the proportions of intervals with $y_t < U_t$ and $y_t > L_t$ respectively:

$$p_U = \frac{1}{h} \sum_{t \in S_k} I\{y_t < U_t\}, \qquad p_L = \frac{1}{h} \sum_{t \in S_k} I\{y_t > L_t\}.$$

We now use these empirical coverage rates to choose when to increase or decrease L or U by $\beta \cdot \text{RMSE}_k$ to adapt for the forecasting performance at Stage k, since we assume that using the current data to adapt the existing model will improve future prediction. Here, β is a tuning parameter; we have found values of $\beta \in [2, 3]$ to be effective. We set lower and upper target coverage a and b and try to maintain coverage rate in the range (a, b). Typical values would be $a = 0.95, b = 0.99$.

If $p_U < a$, meaning that the proportion of observations where $U_t > y_t$ does not even reach our minimum desired coverage, then we increase future values of U by $\beta \cdot \text{RMSE}_k$. Similarly, if $p_L < a$, we decrease L by $\beta \cdot \text{RMSE}_k$.

If $p_U > b$, meaning that the observation is below the upper limit of the forecast interval more often than we intended, we decrease future forecasts U by $\beta \cdot \text{RMSE}_k$, constrained by $U_t \geqslant \hat{y}_t$. We also do not decrease U_t if the

	$p_U < a$	$p_U > b$
$p_L < a$	$U \to U + \beta \cdot \text{RMSE}_k$ $L \to L - \beta \cdot \text{RMSE}_k$	$U \to U - \beta \cdot \text{RMSE}_k$ $L \to L - \beta \cdot \text{RMSE}_k$
$p_L > b$	$U \to U + \beta \cdot \text{RMSE}_k$ $L \to L + \beta \cdot \text{RMSE}_k$	$U \to U - \beta \cdot \text{RMSE}_k$ $L \to L + \beta \cdot \text{RMSE}_k$

Table 1 Situations where $[L, U]$ is adapted to recent forecasting performance, subject to the constraints $L_t < \hat{y}_t < U_t$, $p_U > a$, and $p_l > a$.

adjusted p_U would go below a while calibrating on Stage k data. Similarly, if $p_L > b$, we increase L by $\beta \cdot \text{RMSE}_k$ subject to the constraint $L_t < \hat{y}_t$. In total, there are four combinations of how $[L, U]$ might be updated, which are summarized in Table 1.

3.4 Performance measurement

To measure the performance of our interval forecasts, we use the coverage or proportion of observations in $[L, U]$:

$$\text{coverage} = \frac{1}{T - t_0} \sum_{t=1}^{T-t_0} I\{y_t \in [L_t, U_t]\}. \tag{3}$$

However if two methods have similar coverage, then we use the one which has lower mean forecast interval width:

$$\text{width} = \frac{1}{T - t_0} \sum_{t=1}^{T-t_0} (U_t - L_t). \tag{4}$$

If one method has both higher coverage and lower width than its competitors, then it is dominant. Often, no method is dominant and then the choice of "best" method requires a judgement as to the trade-off between coverage and width. Whether coverage or width is prioritized depends on the requirement of the data analysis. Although there are methods which combine both coverage and width in one criterion, such as the one proposed by Khosravi et al. (2011), choosing the "best" combination still requires user choice of balance between coverage and width.

4 Simulation study

In this section, we compare the performance of our tree-based models with that of parametric models when forecasting simulated data. We use ARIMA and GARCH models both for simulating data and forecasting. We then explore how the performance of our tree-based methods behaves under different drift types and noise levels.

4.1 ARIMA simulation

4.1.1 Method

Our first parametric model is the ARIMA model defined by

$$\left(1-\sum_{k=1}^{p}\varphi_k B^k\right)(1-B)^d y_t = \left(1+\sum_{k=1}^{q}\vartheta_k B^k\right)\epsilon_t, \tag{5}$$

where B is the lag operator, the φ_k and ϑ_k are the parameters of the autoregressive and moving average parts of the model respectively, and the ϵ_t are innovation terms assumed to be independently normally distributed with mean zero and variance σ_ϵ^2. We fix $p = q = 1$ and either fix $d = 0$ (no trend) or randomly sample $d \in \{0, 1\}$ (trend is possible). For each simulation, we generate a time series composed of ten time series segments (each of length 3000), where each segment has independently generated parameters:

$$Y = \{Y^1, Y^2, \ldots, Y^{10}\}, \tag{6}$$

$$\{Y^i\} \sim \text{ARIMA}(\varphi^i, \vartheta^i, \sigma^i, d^i), \;\; i = 1, 2, \ldots, 10. \tag{7}$$

The parameters $\varphi^i, \vartheta^i, \sigma^i$, and d are sampled from distributions which we index by case j; $\varphi^i, \vartheta^i \sim U(a_j, b_j)$, $\sigma^i \sim U(c_j, d_j)$, and $d^i \sim U\{0, 1\}$ if trend is possible or $d^i = 0$ if trend does not exist. By choosing one distribution j, we generate parameters φ_i, ϑ_i, σ_i, and d_i ten times, so we get nine change points in each time series.

In each replicate, we generate a time series Y of length 30000, and apply all methods separately for forecasting. For each case, we conduct 20 replicate simulations. The parameters we found to work well for tree-based methods are $\alpha = \beta = 2$, $(a, b) = (0.95, 0.99)$, $h = 100$, $t_0 = 1311$, $\gamma = 12$, $\delta = 2$, $T_\Omega = 1000$, and $m = 3$.

We employ a similar ARIMA forecasting approach to our regression tree methods for comparison. Instead of using wavelet transformed variables, we use the original untransformed variable in ARIMA (and, later, for our GARCH simulation), as our ARIMA approach is univariate and employing the wavelet transform converts the univariate predictor into a multivariate predictor with variables at each resolution level. (Extension to multivariate models could be considered in future work.) We use the function auto.arima in the R package forecast (Hyndman 2017; Hyndman and Khandakar 2008) to find the best ARIMA model according to AICc (small sample size corrected AIC). This function conducts a search over possible models within the order constraints provided: $p, q \in \{0, 1, \ldots, 6\}$ and $d \in \{0, 1, 2\}$. Allowing higher orders incurs a higher computational load. The prediction interval levels we choose for single ARIMA and ensemble ARIMA are set to be equal to the empirical coverage results from single tree and ensemble tree methods, which we denote by c, to obtain results comparable to those obtained by using regression trees. There are circumstances when the coverage can not meet this specified prediction interval level c, even though model updating is performed, especially when the

model order has been incorrectly identified. As in the regression tree methods, we use the first $t_0 = 1311$ observations to train our initial model. By using data from $t - \gamma + 1$ to t, we can predict the value at time $t + 1$

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-\gamma+1}),$$

where $\gamma = 1000$. Then we recursively use this predicted $\hat{y}_{t+1}$ to predict $\hat{y}_{t+2}$

$$\hat{y}_{t+2} = f(\hat{y}_{t+1}, y_t, y_{t-1}, \cdots, y_{t-\gamma+2}),$$

and iterate to get the predicted value $h = 100$ observations ahead as $\hat{y}_{t+100}$.

We allow our ARIMA forecasts to respond to concept drift by updating and retraining. For updating, we keep p, q, and d fixed and re-estimate the parameters by using the most recent $\gamma = 10h$ observations. Retraining allows p, q and d to be re-estimated. There are two criteria for model retraining and updating. The first is the empirical coverage; retraining if the method fails to achieve lower coverage $c - 0.3$ and $c - 0.15$ for updating and retraining respectively, or exceeds the upper coverage $b = 0.99$. The second criterion, to avoid excessively frequent fitting, is that no updating or retraining will occur for at least 100 new observations after an update or retraining.

When retraining, instead of using h observations as in the regression model, we use $10h$ observations to make the ARIMA model robust to short-term trends. So in the ensemble method, there is no need to train a new model after every 100 observations unlike the ensemble tree method which uses only 100 observations to train a new model. We choose to train a new model so as to leave out the oldest model or we update the most recent model. For the ensemble model, we choose up to $m = 3$ most recent models and weight predictions in the same way as that in the ensemble tree method. Larger values of m may lead to better performance but at the cost of greater computation time. In all applications we have considered, $m = 3$ has given a reasonable balance between computational speed, performance and interpretability. Other criteria are the same as in the tree methods.

4.1.2 Results

Results from 20 simulations are shown in Table 2 and details of one realization from case 4 are shown in Figure 1.

Note that tree methods take around 1/3 of the computation time of ARIMA in both single and ensemble methods, including the time taken to compute the wavelet transform.

When comparing point estimation accuracy of tree methods with ARIMA, the ARIMA methods have substantially larger RMSE in cases 6 and 8. The other cases have similar RMSE values for tree-based and ARIMA methods. Overall, we conclude that tree methods are better than ARIMA methods in terms of point estimation accuracy.

When comparing tree methods with ARIMA in coverage and width, it is clear that the ensemble tree method is better than or similar to single ARIMA

method and ensemble ARIMA method in most situations except cases 4, 6 and 8. But for these cases, methods with higher coverage are at the cost of bigger width. If we allowed a wider interval for tree methods, they might achieve the same coverages as that of ARIMA. For the tree methods, the single tree method has equal performance to that of the ensemble tree method except in cases 5 and 7 where the single tree method has similar coverage but noticeably wider intervals.

For the widthsd (the standard deviation of width within each simulation), tree methods are somewhat better than ARIMA. For a time series, when trend disappears, ARIMA will still forecast with trend before retraining thus the forecast intervals can be extremely wide. When there is no trend, but a trend is falsely detected by ARIMA, intervals can be wide as well, but tree methods, which consider trend in a different way, can avoid such situations.

When the data-generation distribution changes from one time series segment to the next, as shown in Equation (6), the forecast intervals from the tree methods can react much more quickly than ARIMA. Examples of ARIMA being slow to respond are shown in the circled areas of Figure 1; the ARIMA interval forecasts are predictably ineffective when moving from a segment with trend ($d = 1$) to one without ($d = 0$).

In conclusion, the ensemble tree method is better than the others when there is no time efficiency requirement, although the extent to which it is better depends on the situation. When time is critical, the single tree method is suggested. When the ARIMA effect is strong and trend possible, the single ARIMA method is suggested, but at the expense of wide intervals.

Table 2 The mean and standard deviation (sd) of coverage, width, widthsd and time for all methods across 20 simulations. Widthsd is the standard deviation of width within each simulation. The sd in width(sd) is the standard deviation of mean width across 20 simulations.

φ, ϑ		U(0.1, 0.2)				U(0.8, 0.9)			
σ	white noise	U(0.2, 0.5)		U(4, 5)		U(0.2, 0.5)		U(4, 5)	
trend possible		no	yes	no	yes	no	yes	no	yes
case	0	1	2	3	4	5	6	7	8
Single tree method									
coverage(%)	99.39(0.2)	98.13(1.2)	76.39(8.5)	99.36(0.2)	73.56(7.4)	94.70(1.5)	69.16(10.5)	94.80(1.4)	64.04(9.0)
width	5.52(0.1)	3.21(0.4)	8.15(2.9)	12.24(0.4)	28.92(9.2)	11.87(1.4)	54.99(22.2)	43.11(3.5)	223.54(74.1)
widthsd	0.00(0.0)	0.28(0.2)	7.95(2.9)	0.06(0.1)	29.34(10.2)	3.30(0.8)	74.09(29.6)	11.43(2.0)	272.60(98.0)
time	60.11(2.4)	62.40(12.6)	67.66(15.6)	60.10(2.2)	65.85(10.2)	60.01(1.9)	66.19(2.8)	60.23(1.2)	67.92(2.4)
RMSE	1.00(0.0)	0.62(0.0)	7.30(2.1)	2.22(0.0)	28.36(6.6)	2.28(0.2)	69.30(19.6)	8.55(0.7)	271.36(71.0)
Ensemble tree method									
coverage(%)	99.32(0.0)	99.22(0.1)	75.26(9.8)	99.31(0.1)	73.18(8.9)	94.34(0.6)	69.21(9.4)	93.96(0.8)	66.23(9.1)
width	5.45(0.0)	3.33(0.2)	8.19(2.5)	12.07(0.1)	30.12(6.7)	9.48(0.4)	52.46(18.0)	34.78(0.6)	227.17(65.7)
widthsd	0.16(0.0)	0.41(0.1)	7.67(2.3)	0.53(0.1)	27.21(5.5)	2.23(0.3)	69.00(17.7)	7.23(0.6)	280.70(67.1)
time	229.22(5.1)	239.27(45.4)	252.85(54.3)	229.08(6.1)	249.91(7.0)	258.50(60.1)	251.89(18.3)	238.74(5.9)	248.28(7.6)
RMSE	1.00(0.0)	0.62(0.0)	8.80(2.9)	2.22(0.0)	33.544(8.0)	2.21(0.1)	84.41(25.0)	8.18(0.3)	332.20(93.6)
Single ARIMA method									
coverage(%)	99.00(0.0)	97.29(1.5)	76.57(7.4)	99.01(0.1)	74.73(6.1)	95.10(1.6)	74.07(7.3)	95.26(1.6)	71.31(6.6)
width	6.34(0.6)	3.04(0.3)	17.86(6.6)	12.88(1.0)	67.01(32.9)	11.85(1.9)	181.16(63.4)	43.06(5.8)	702.23(171.2)
widthsd	6.12(1.7)	1.87(0.9)	62.03(38.1)	10.18(4.0)	233.29(180.0)	10.47(2.0)	456.84(327.3)	36.96(6.1)	1481.35(686.4)
time	173.51(5.8)	176.09(45.3)	162.13(40.1)	174.69(5.7)	150.16(6.7)	189.73(34.1)	177.86(39.0)	179.23(12.1)	162.46(9.7)
RMSE	1.01(0.0)	0.62(0.0)	8.49(2.6)	2.23(0.0)	33.99(12.8)	2.15(0.1)	583.08(439.7)	7.91(0.3)	2755.65(2082.3)
Ensemble ARIMA method									
coverage(%)	99.05(0.1)	98.19(0.8)	77.41(7.5)	99.05(0.1)	77.04(6.5)	95.56(0.7)	77.92(5.4)	95.26(0.9)	76.84(5.1)
width	6.34(0.6)	3.33(0.3)	17.34(6.0)	13.20(1.2)	67.40(33.9)	11.90(1.3)	174.41(57.5)	42.13(4.4)	718.73(155.0)
widthsd	5.26(1.6)	2.08(0.9)	52.64(32.0)	9.73(4.0)	202.60(160.7)	9.43(1.4)	370.22(257.9)	32.44(4.4)	1316.99(531.5)
time	680.86(31.9)	710.16(147.7)	668.94(169.9)	805.04(571.7)	636.70(62.5)	816.13(301.1)	669.54(38.2)	691.57(29.6)	662.62(23.2)
RMSE	1.01(0.0)	0.62(0.0)	8.25(2.5)	2.23(0.0)	35.06(14.9)	2.14(0.1)	462.14(378.3)	7.86(0.3)	2665.37(2029.0)

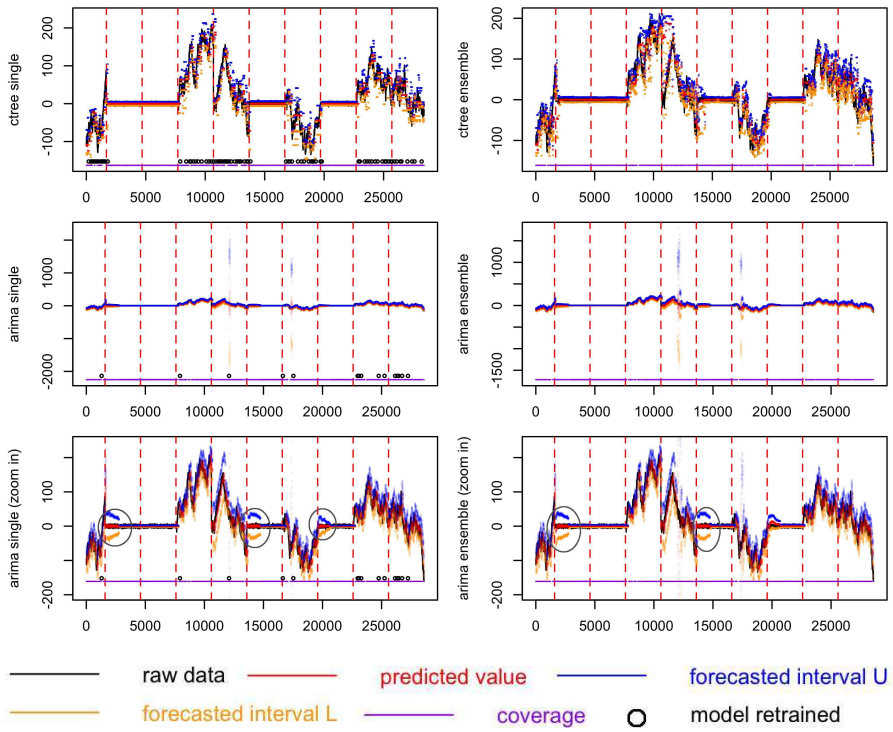


Fig. 1 Simulation results (from one realisation of case 4). Left and right columns show single-model and ensemble results respectively. From top to bottom, results are for ctree, ARIMA, and ARIMA with restricted vertical range. X-axis is time and Y-axis is the values of the time series. In this plot, φ and ϑ both follow $U(0.1, 0.2)$, σ follows $U(4, 5)$ with trend possible. Red dashed vertical lines represent the time when distribution changes. The time when model is retrained is labeled as black circles, slightly above the purple line. The label is the same for the rest of such figures. ARIMA has extremely wide intervals in some cases and its intervals react slowly when the data distribution changes as shown in the circled areas. Colours are displayed in the electronic ADAC version; the printed version is restricted to black/white displays.

4.2 GARCH simulation

4.2.1 Method

We now simulate data from the GARCH model $y_t = \sigma_{t|t-1}\epsilon_t$, where ϵ_t is Gaussian white noise with unit variance and

$$\sigma_{t|t-1} = w + \sum_{k=1}^{r} \xi_k y_{t-k}^2 + \sum_{k=1}^{s} \zeta_k \sigma_{t-k|t-k-1}.$$

In this simulation, we choose both r and s to be 1. However, when we identify the GARCH model, we allow a maximum of 3 for both r and s. The data simu-

Table 3 The mean and standard deviation (sd) of coverage, width, widthsd and time for all methods applied to GARCH data across 20 simulations. Widthsd is the standard deviation of width within each simulation. The sd in width(sd) is the standard deviation of mean width across 20 simulations. Here, w follows uniform distribution $U(0, 2)$.

ξ	0	U(0, 0.2)	U(0, 0.2)	U(0.6, 0.8)	U(0.3, 0.5)
ζ	0	U(0, 0.2)	U(0.6, 0.8)	U(0, 0.2)	U(0.3, 0.5)
case	0	1	2	3	4
Single tree method					
coverage(%)	97.02(1.5)	96.59(1.4)	95.71(0.8)	95.49(1.4)	99.56(0.2)
width	5.37(0.9)	11.08(1.9)	9.90(1.2)	10.48(2.2)	5.82(0.4)
widthsd	1.62(0.4)	4.19(1.2)	4.72(2.1)	4.93(3.1)	0.15(0.2)
time	36.06(5.1)	35.58(2.6)	34.27(0.4)	34.99(1.5)	37.73(6.3)
RMSE	1.14(0.1)	2.47(0.4)	2.35(0.3)	2.49(0.5)	1.00(0.0)
Ensemble tree method					
coverage(%)	98.91(0.2)	98.71(0.2)	97.45(0.2)	97.92(0.2)	99.34(0.1)
width	6.00(0.5)	12.70(1.8)	10.93(1.1)	12.07(1.7)	5.46(0.0)
widthsd	1.74(0.3)	4.96(1.7)	4.28(1.2)	5.20(2.2)	0.16(0.0)
time	234.86(8.2)	235.40(11.7)	236.11(12.3)	236.91(12.8)	241.12(17.3)
RMSE	1.15(0.1)	2.47(0.4)	2.36(0.3)	2.49(0.5)	1.00(0.0)
Single GARCH method					
coverage(%)	96.24(1.6)	95.72(1.2)	95.12(0.9)	94.36(1.1)	99.77(0.0)
width	4.88(0.8)	9.88(1.4)	9.74(1.3)	9.45(1.9)	6.18(0.0)
widthsd	1.48(0.2)	3.96(1.4)	6.10(2.3)	5.15(3.3)	0.31(0.0)
time	998.04(371.3)	962.33(314.4)	986.34(149.1)	841.51(131.8)	2456.56(648.7)
RMSE	1.15(0.1)	2.46(0.4)	2.36(0.3)	2.49(0.5)	1.00(0.0)
Ensemble GARCH method					
coverage(%)	99.06(0.8)	98.40(0.9)	96.31(0.7)	96.22(0.7)	99.77(0.0)
width	6.50(0.9)	12.77(2.0)	11.10(1.4)	10.98(2.2)	6.18(0.0)
widthsd	1.97(0.3)	5.07(1.3)	6.99(2.7)	6.01(3.9)	0.30(0.0)
time	3080.11(1261.3)	2645.60(1168.9)	1764.12(110.1)	2099.51(988.5)	3049.03(340.8)
RMSE	1.15(0.1)	2.46(0.4)	2.36(0.3)	2.49(0.5)	1.00(0.0)

lation process is analogous to the earlier ARIMA simulation. The functions we use include garchAuto, ugarchfit, ugarchforecast in the R packages rugarch (Ghalanos 2014) and fGarch (Wuertz et al. 2019). The criterion used for GARCH model selection is AIC. As for our ARIMA simulation, model-based forecasts use the original variable rather than wavelet-transformed variables.

4.2.2 Results

The results are shown in Table 3 and one specific example is shown in Figure 2. In Table 3, we can see that the forecast intervals from the single tree and single GARCH methods are slightly less responsive to concept drift than their ensemble counterparts in terms of interval coverage. Generally, these methods share similar performance with higher coverage at the cost of higher width. For point estimation, all the methods considered have similar RMSE values.

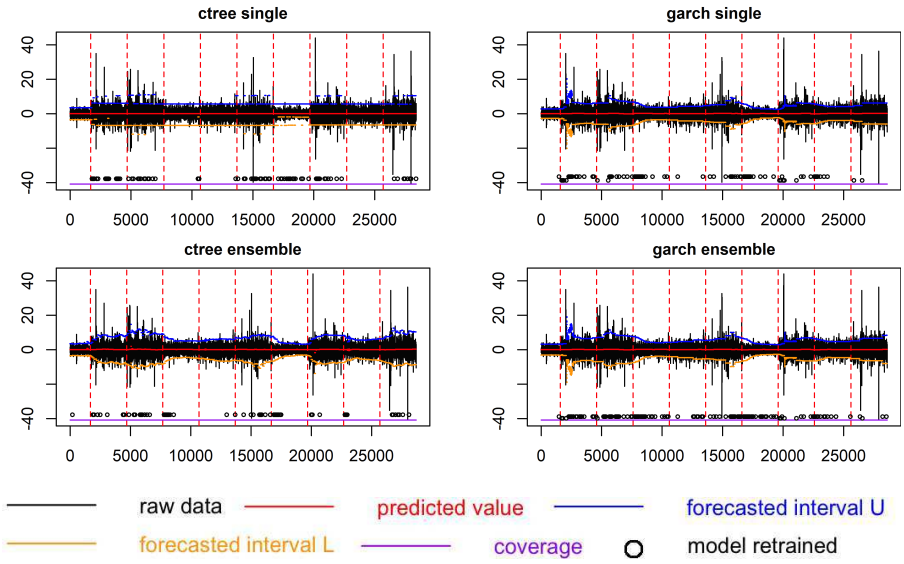


Fig. 2 GARCH simulation results (from one realisation of case 4). In this plot, ξ and ζ both follow $U(0.3, 0.5)$. Red dashed vertical lines represent the time when distribution changes. X-axis is time and Y-axis is the values of the time series.

4.3 Performance of tree methods under different noise levels and drift types

In order to explore how our tree methods perform under different noise levels and drift types, we simulate AR(1) data undergoing different forms of concept drift. As in Equation (5), the noise level is specified by the white noise variance σ_ϵ^2. We consider drift represented by changes in the autoregressive parameter φ and different mean levels and use the four types of drift proposed by Krawczyk and Cano (2018) describing severity and speed of changes. Specification of our data structures and example realizations are shown in Table 4.

When our methods are applied to one simulated data realization of each type, the results are shown in Figure 3. It shows that the single-tree method retrains the model soon after concept drift happens for sudden, gradual and recurring changes. For incremental change, where the concept does not change sharply, the single-tree method finds only two change points, and is not as responsive as for other change types. The ensemble tree method can generally find the change point as it will always train a new model for the new data, but it reacts to sudden changes a bit more slowly than the single-tree method as the ensemble method remembers long term information and takes time to forget old information.

When the simulations were conducted 50 times under each each type of drift for $\sigma_\epsilon = 1, 2, \ldots, 5$, the results are shown in Figures 4 and 5. The change points are the same as in Figure 3. The point when change drift happens, is also the point when coverage drops suddenly.

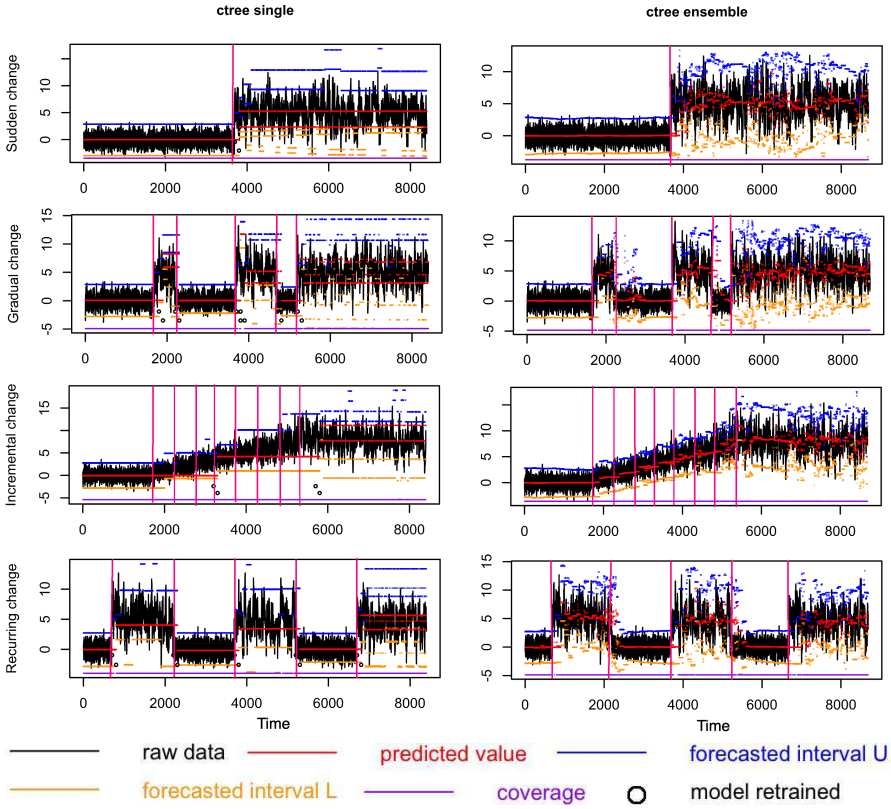


Fig. 3 The performance of single-tree and ensemble methods under different drift types with $\sigma_\epsilon = 1$. The red vertical lines are the time points when concept drift happens.

When $\sigma_\epsilon = 1$, the single tree method can generally detect different types of drift, in many cases responding to concept drift more quickly than the ensemble method. The main exception is for the recurring concept drift, where the ensemble method "remembers" a concept which has been seen previously when it recurs. When drift happens, especially for sudden, gradual, and recurring drifts, the coverage obviously drops but quickly recovers. The AR(1) process with more pronounced autoregression (higher φ) usually has lower coverage and wider intervals than weakly correlated data. When σ_ϵ increases, the frequency of model retraining increases even when drift does not occur (up to $\sigma_\epsilon = 4$) but reduces again at $\sigma_\epsilon = 5$; qualitatively similar results were seen when we increased the noise levels for different drift types.

In conclusion, in the presence of drift, the performance of tree methods initially drops, but soon recovers. When the noise level increases, the coverage performance decreases slightly, but good coverage is maintained at the cost of much wider intervals.

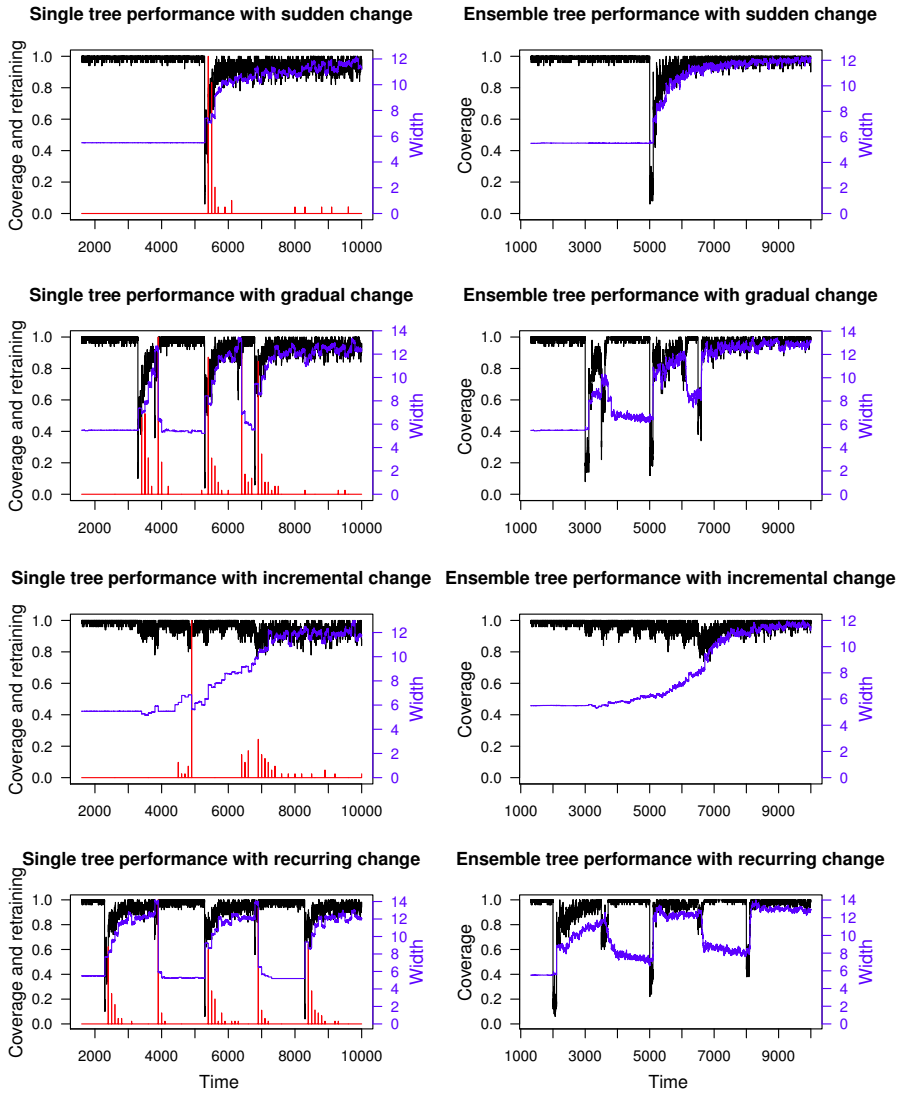
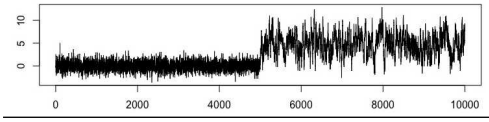
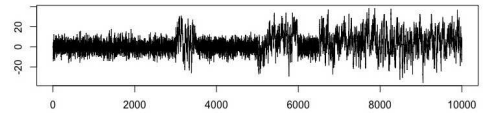
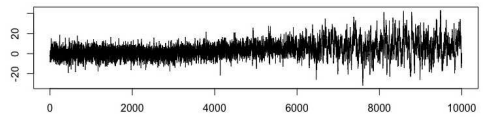
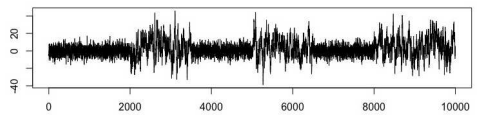


Fig. 4 The performance of single-tree and ensemble methods under different drift types with $\sigma_\epsilon = 1$, averaged over 50 replications at each time point. Black and blue lines are averaged coverage and width, respectively. Red lines are the proportion of replicates where retraining occurred. As the ensemble method always trains a new model when a new data batch arrives, there is no retraining line for the ensemble method.

Table 4 Four types of drift with simulated examples based on AR(1) processes with autoregressive parameter φ. All AR processes have white noise variance σ_ϵ^2; in these examples, $\sigma_\epsilon^2 = 1$.

Drift type and example	Specification
Sudden drift 	$y = [y_1, y_2]$, y_1: $\varphi = 0.1$, mean 0, length 5000, y_2: $\varphi = 0.9$, mean 5, length 5000.
Gradual drift 	$y = [y_1, y_2, y_3, y_4, y_5, y_6]$, y_1, y_3, y_5: $\varphi = 0.1$, mean 0, lengths 3000, 1500, 500, y_2, y_4, y_6: $\varphi = 0.9$, mean 5, lengths 500, 1000, 3500.
Incremental drift 	$y = [y_1, y_2, \ldots, y_9]$, y_i,: $\varphi = i/10$, mean $i - 1$, lengths 3000, 500, ..., 500, 3500.
Recurring drift 	$y = [y_1, y_2, y_3, y_4, y_5, y_6]$, y_1, y_3, y_5: $\varphi = 0.1$; y_2, y_4, y_6: $\varphi = 0.9$, all with mean 5, lengths 2000, 1500, ..., 1500, 2000.

5 Forecasting heart rate in surgery

The data we use comes from 325 patients undergoing Liver Transplantation (LT) operation, a high-risk treatment for patients with end-stage liver disease. The data, which was recorded using a LIDCO monitor on patients undergoing LT between September 2004 and December 2011, is provided by St James's University Hospital, Leeds, UK. For details, refer to Milan et al. (2016). The variable we use is the heart rate (beats/min), as it is one of the most important variables to be monitored on an ongoing basis during surgery. We aim to predict heart rate $h = 100$ heart beats ahead (in our data, each heart beat takes around 0.5 to 1 seconds), so that the operating team can have enough time for preparation if heart rate is forecast to go out of a normal range, when the patient might be in danger. In our illustration, we use previously cleaned data (Zhao et al. 2018).

5.1 Tree-based forecasting

When we forecast heart rate using our ensemble and single tree methods, we found parameter values $\alpha = \beta = 2$, $(a, b) = (0.95, 0.99)$, $h = 100$, $t_0 = 1311$

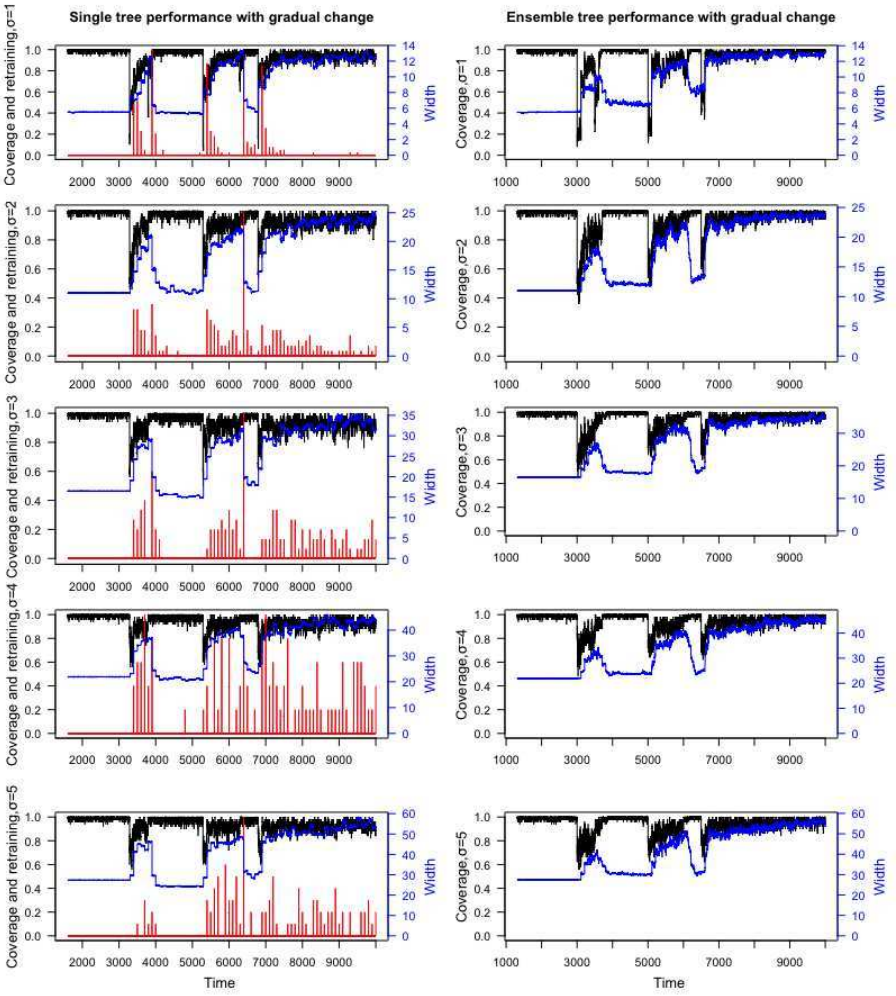


Fig. 5 The performance of single-tree and ensemble methods on data subject to gradual drift under different noise levels $\sigma_\epsilon = 1$ (top) to $\sigma_\epsilon = 5$ (bottom), averaged over 50 replications. Black, blue and red lines have the same meaning as in Figure 4. Change in performance as noise increases is similar with other types of drift.

and $\gamma = 12$, $\delta = 2$, $T_\Omega = 8000$, and $m = 3$ to work well across all patients. We use two patients' data as examples in Figure 6. Ensemble and single tree methods have coverage $(77.7\%, 85.9\%)$, widths $(7.6, 6.8)$, and computing times $(134, 12)$ for patient 1 and coverage $(86.6\%, 83.6\%)$, widths $(10.0, 7.3)$, and computing times $(491, 41)$ for patient 2. From the results, we can see that both methods have good performance. Generally the forecast intervals cover the observed values without being excessively wide, only failing to include the observed data when there are episodes of high volatility.

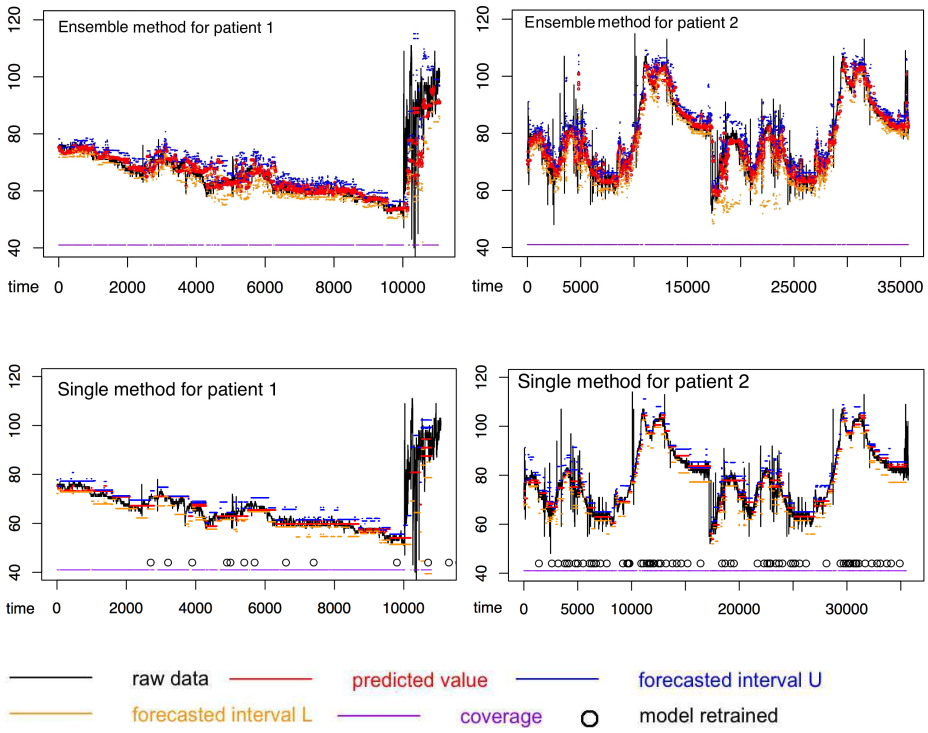


Fig. 6 Data and monitoring forecasts using regression trees for liver transplantation patients 1 and 2.

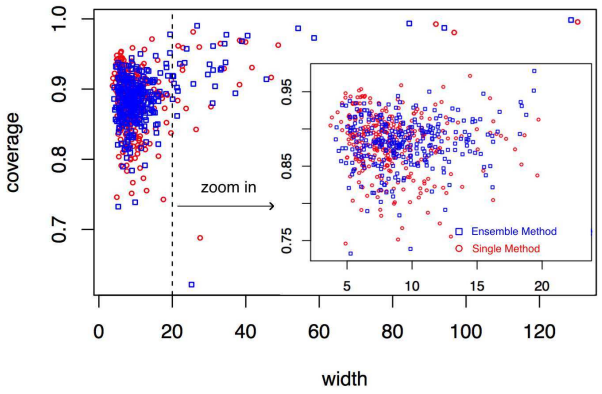


Fig. 7 Summary results of forecasts for all 325 patients. Each point represents mean coverage and width for one patient. The inset plot is a zoom in of the larger plot.

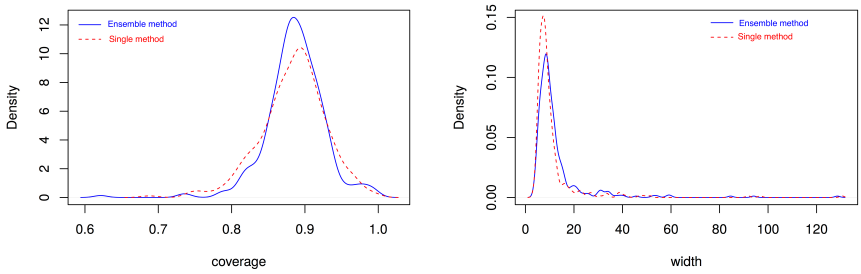


Fig. 8 Kernel density plots of coverage (left) and mean width (right) for the full set of 325 patients.

Table 5 The relative performances of ensemble and single tree methods as percentages of the 325 patients.

		Narrower intervals	
		ensemble	single
Higher coverage	ensemble	5.23%	44.62%
	single	17.54%	32.62%

Summary results for all 325 patients are shown in Table 6 and Figures 7 and 8, including a detailed view of those patients whose mean interval width was below 20. We can see the results of single and ensemble tree methods are generally similar, with the ensemble tree method being a little higher in coverage but having wider intervals and being substantially more computationally expensive. The ensemble tree method has a lower density for width around 10 and the coverage density is a little higher when coverage is around 0.88.

Coverage and width are closely related and inspecting them separately does not tell the entire story, so we now consider them together. Results of some selected patients are shown in Figure 9, with the results of the ensemble and single tree methods for a single patient joined by a dashed line for clarity. For example, for patient 34, the ensemble tree method has better coverage at the cost of wider intervals. But for patients 45 and 46, the single tree method has better coverage with narrower intervals. For patient 26, they share nearly the same coverage, but the ensemble tree method has a smaller width.

We consider each method in terms of coverage and width. One method is superior if it gives higher coverage and narrower intervals, inferior if it has lower coverage and wider intervals, and the two methods represent a trade-off if one has higher coverage and the other has narrower intervals. To compare our methods, we show the proportion of patients in each category in Table 5. It is clear that the single-tree method is better as it has substantially more patients where there is both higher coverage and narrower intervals (32.62%). The ensemble tree method is only superior in about 5% of patients. When neither

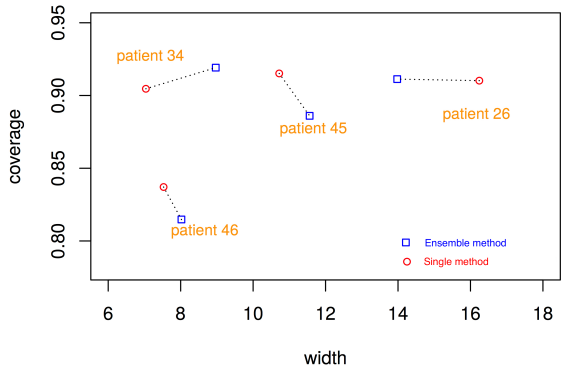


Fig. 9 Coverage and mean interval width of ensemble and single tree methods for selected patients 26, 34, 45 and 46.

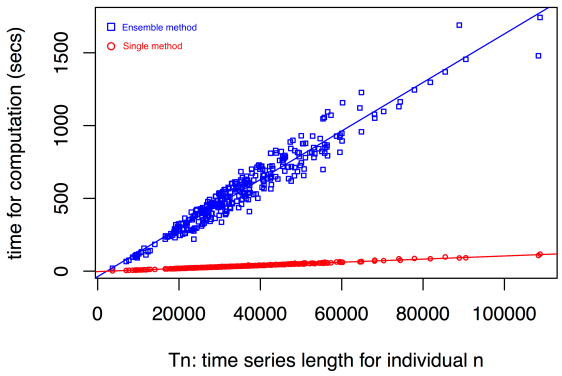


Fig. 10 The computational time (in seconds) for ensemble and single tree methods for all 325 patients. The fitted linear regression slope coefficients are 0.016 and 0.001 for ensemble and single tree respectively.

method is clearly better, the ensemble tree method is more likely to have higher coverage, while the single tree method is more likely to have narrower intervals. Ignoring the interval width, each method has better coverage in about 50% of cases.

As shown in Figure 10, the computation time has a roughly linear relationship with time series length T_n. The fitted regression slope coefficients of 0.016 (ensemble tree method) and 0.001 (single-tree method) indicate that ensemble method takes around 16 times as long as the single-tree method.

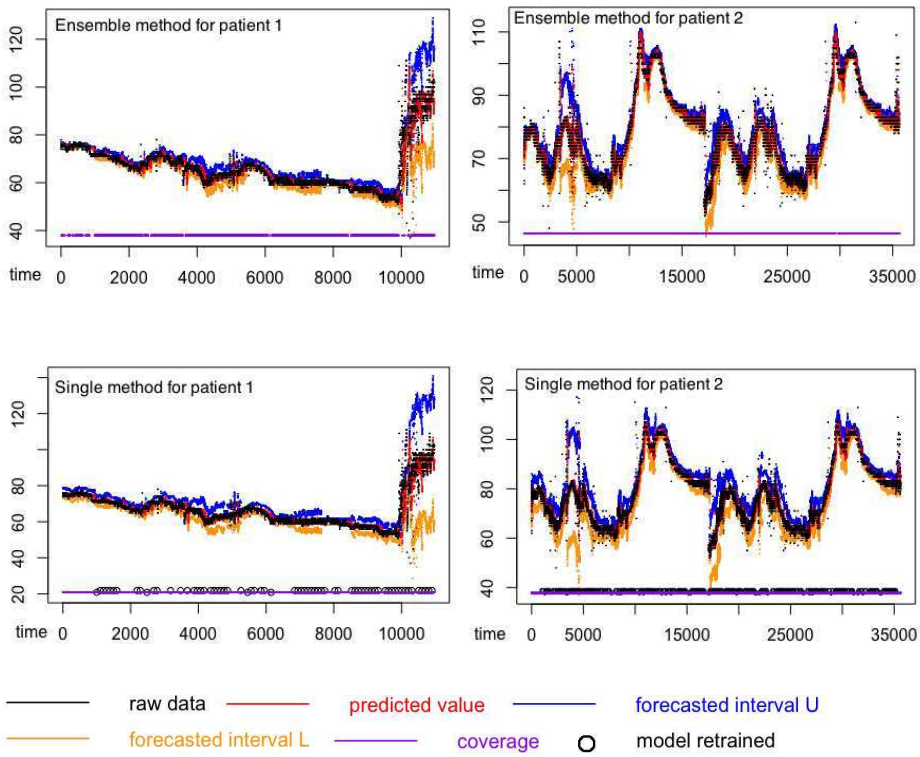


Fig. 11 Data and monitoring forecasts using ARIMA models for liver transplantation patients 1 and 2.

However, since new observations arrive at intervals of about 0.5–1 second, the computation can easily be done online in real time for these data.

The variables used in the trees include both scaling coefficients and wavelet coefficients at high and low resolution levels using both short and long lag information. That means heart rate prediction needs both averages and contrasts with long and short time interval information. In our case, averages were taken over time spans between 2 and 256 heartbeats. Prediction by only using short term information may not have a good performance.

5.2 Comparison to ARIMA

We use the ARIMA model to forecast heart rate to compare its performance with that from trees. The parameters are the same as those in the simulation. Results are shown in Figure 11 and Table 6. Ensemble ARIMA and single ARIMA methods have coverage $(86.82\%, 90.46\%)$, widths $(9.98, 10.56)$, and computing times $(240, 65)$ for patient 1 and coverage $(90.12\%, 91.19\%)$, widths $(10.94, 9.93)$, and computing times $(814, 269)$ for patient 2.

Table 6 Tree and ARIMA results: mean and standard deviation (sd) of coverage, width and time for each Method over 325 patients.

Method	coverage		width		time	
	mean	sd	mean	sd	mean	sd
Tree						
single	0.8837	0.0447	11.05	11.72	35.25	16.81
ensemble	0.8861	0.0405	12.53	12.08	553.71	267.53
ARIMA						
single	0.8853	0.0286	10.93	8.32	466.07	256.33
ensemble	0.8732	0.0300	10.38	7.956	1032.20	525.58

Table 7 The relative performances of ARIMA methods as percentages of the 325 patients.

		Narrower intervals	
		ensemble	single
Higher coverage	ensemble	5.03%	7.38 %
	single	79.87%	7.71%

For ARIMA, we can not distinguish whether the ensemble or single method is better in terms of coverage and width. The single method is a little bit higher in both coverage and width, but clearly less computationally demanding. But they all face a problem: when concept drift in the trend occurs, the old model cannot predict accurately (especially when d is not 0), as shown in the results for patient 2. But for tree-based models, such situations do not occur as these models do not consider trend in the same way. The results of a pairwise are shown in Table 7.

The single-model method (7.71%) is superior slightly more often than the ensemble method (5.03%). When neither method is clearly better, the ensemble method is more likely to have narrower intervals, while the single-model method is more likely to have higher coverage. In practice, we would choose the single-model method as the performances are comparable but the single-model approach is less computationally demanding.

Table 8 The relative performances of single-forecast approaches for regression trees and ARIMA as percentages of the 325 patients.

		Narrower intervals	
		ARIMA	Ctree
Higher coverage	ARIMA	12.62%	38.77 %
	Ctree	33.85%	14.77%

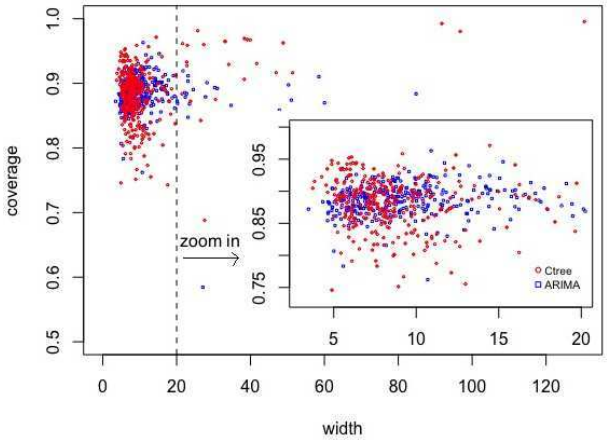


Fig. 12 The relative performances of single-forecast approaches for regression trees and ARIMA across all 325 patients.

Now we compare the single regression tree and single-model ARIMA approaches in Table 8 and Figure 12. The Ctree-based approach is superior slightly more often than the ARIMA method (14.77% vs. 12.62%). So roughly, their performance is similar in coverage and width although the regression tree method is somewhat higher in standard deviation as shown in Table 6. However, in terms of computational time, the regression tree method obviously outperforms ARIMA. In conclusion, we choose the single tree-based method.

6 Forecasting stock price

In order to compare tree-based methods with more sophisticated time series models, in this section, we compare tree-based model with ARMA-GARCH in stock price forecasting. The stock we choose is Shanghai Stock Exchange Composite Stock Price Index (SSE Index), which is an index of all stocks traded at the Shanghai Stock Exchange. It is a relatively long time series with both stationary and non-stationary phases as well as many distribution change points. The time series dates from 19th, December, 1990 to 1st, June, 2018, making a total of 6713 closing price observations excluding non-trading days like weekends and holidays. Since the time series x_t is not stationary, we do first order differencing and take logs to reduce variance fluctuation, so the variable we use is

$$y_t = \log(x_{t+1}) - \log(x_t).$$

The parameters we choose for trees are $\alpha = \beta = 2$, $(a, b) = (0.95, 0.99)$, $h = 100$, $t_0 = 1311$, $\gamma = 12$, $\delta = 2$, $T_\Omega = 1000$, and $m = 3$. For ARMA-GARCH, the maximum value for p, q, r and s are all 2, with $\gamma = 300$. Results are shown in Table 9 and Figure 13 and ensemble ARMA-GARCH is better than single-tree method with slightly higher coverage and lower width. For

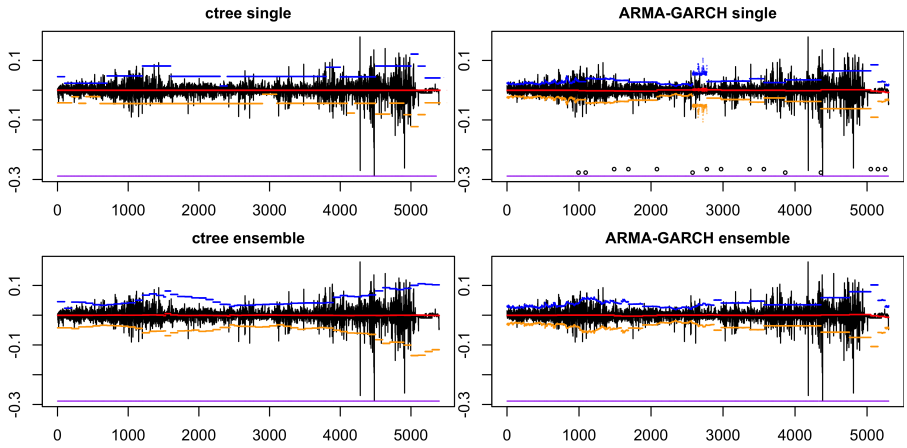


Fig. 13 The performances of single and ensemble methods for regression trees and ARMA-GARCH on SSE Index data.

the other methods, the higher coverage of the ensemble-tree method is at the cost of wider intervals. But in terms of computational time, the ensemble tree method is suggested.

Table 9 The performance of tree methods and ARMA-GARCH methods on SSE Index data.

Method	coverage (%)	width (×100)	widthsd (×100)	time
Tree				
single	93.58%	9.84	3.4	6.06
ensemble	96.70%	11.15	4.7	38.45
ARMA-GARCH				
single	90.93%	7.25	3.1	302.36
ensemble	93.94%	8.40	3.1	598.86

7 Conclusion

We have proposed two tree-based methods to deal with forecasting in a streaming data context. In contrast to many alternative methods, we pay more attention to forecast intervals than point forecasts, although the point forecasts are essential to adaptively adjust our forecast intervals for the current model's accuracy. This adaptation is accomplished by updating the forecast interval by using root mean square prediction error calculated from the most recent batch of data, so as to update the model for future prediction. The interval is

not necessarily symmetric, being initially based on quantiles but then adapted to allow for the performance of both point and interval forecasts, constrained to ensure that the interval forecast will always include the point forecast. An interesting possibility for future research would be to move the entire interval in the direction of the overall mean of the newly-observed data. Another possible topic for future investigation is effective choice of the ensemble size m.

Rather than fixing a time interval of historical data to use in forecasting, we use a wavelet transform to capture long term variable information. The maximal overlap wavelet transform decomposes the original time series into different resolution levels to capture averages and fluctuations over a range of time scales. This means that the tree construction algorithms can select whichever aspects of the information in the data are most useful. Moreover, we gain the benefit of allowing long time spans of historical data to be used in the models without requiring they all be present as separate explanatory variables, so that the information we use will not be constrained to the most recent batch. If variables from coarser resolution levels are used, it means the time series has long term correlation, and vice versa. For example, if variable heart rate on resolution level 4 is used in the tree building process, then it means the heart rate time series has a lag effect of length $2^4 = 16$.

When applied to real and simulated data, tree-based methods generally perform similarly or better than those based on ARIMA, GARCH or ARMA-GARCH except when the ARIMA effect is strong and trends may be present. For ARIMA model-based forecasts, when a trend disappears, forecasts will still assume the trend is present before retraining thus the forecast intervals can be extremely wide. When there is no trend, but a trend is falsely detected by the ARIMA model, intervals can be wide. But tree methods, which consider trend in a different way, can avoid such situations. Because of this risk of incorrectly estimating model structure, it is suggested that we use tree methods even when the ARIMA effect appears to be strong.

When the ARIMA effect is strong without trend, the ensemble tree method is generally slightly better than the single-tree method in terms of the balance of coverage and interval width. When time efficiency is required, the single tree method is suggested, otherwise the ensemble tree method is preferred.

The R code implementations of our methods ctreeensemble.R and ctreeone.R are contained in the 'Supplementary Material' sections of the electronic journal version on the ADAC website.

References

A. Appice and M. Ceci. Mining tolerance regions with model trees. In *International Symposium on Methodologies for Intelligent Systems*, pages 560–569. Springer, 2006.

M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno. Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 443–448. SIAM, 2007.

A. Bifet and R. Gavaldà. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer, 2009.

A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Aata Mining*, pages 139–148. ACM, 2009.

A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

P. Domingos and G. Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM, 2000.

J. Duarte, J. Gama, and A. Bifet. Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):30, 2016.

J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Brazilian Symposium on Artificial Intelligence*, pages 286–295. Springer, 2004.

A. Ghalanos. *rugarch: Univariate GARCH models.*, 2014. R package version 1.3-5.

A. Gholipour, M. J. Hosseini, and H. Beigy. An adaptive regression tree for non-stationary data streams. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 815–817. ACM, 2013.

T. Hothorn and A. Zeileis. partykit: A modular toolkit for recursive partytioning in R. *Journal of Machine Learning Research*, 16:3905–3909, 2015. URL `http://jmlr.org/papers/v16/hothorn15a.html`.

T. Hothorn, K. Hornik, M. A. Van De Wiel, and A. Zeileis. A lego system for conditional inference. *The American Statistician*, 60(3):257–263, 2006a.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006b.

R. J. Hyndman. *forecast: Forecasting functions for time series and linear models*, 2017. URL `http://pkg.robjhyndman.com/forecast`. R package version 8.2.

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. URL `http://www.jstatsoft.org/article/view/v027i03`.

E. Ikonomovska, J. Gama, and S. Džeroski. Learning model trees from evolving data streams. *Data Mining and Knowledge Discovery*, 23(1):128–168, 2011.

E. Ikonomovska, J. Gama, and S. Džeroski. Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, 150: 458–470, 2015.

R. Jin and G. Agrawal. Efficient decision tree construction on streaming data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 571–576, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. doi: 10.1145/956750.956821. URL `http://doi.acm.org/10.1145/956750.956821`.

A. Khosravi, S. Nahavandi, and D. Creighton. Prediction interval construction and optimization for adaptive neurofuzzy inference systems. *IEEE Transactions on Fuzzy Systems*, 19(5):983–988, 2011.

B. Krawczyk and A. Cano. Online ensemble learning with abstaining classifiers for drifting and noisy data streams. *Applied Soft Computing*, 68:677–692, 07 2018. doi: 10.1016/j.asoc.2017.12.008.

Z. Milan, C. Taylor, D. Armstrong, P. Davies, S. Roberts, B. Rupnik, and A. Suddle. Does preoperative beta-blocker use influence intraoperative hemodynamic profile and post-operative course of liver transplantation? *Transplantation Proceedings*, 48(1):111–115, 2016.

K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In *International Conference on Discovery Science*, pages 264–269. Springer, 2007.

D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*, volume 4. Cambridge University Press, Cambridge, 2000.

B. Pfahringer, G. Holmes, and R. Kirkby. New options for Hoeffding trees. In *Australasian Joint Conference on Artificial Intelligence*, pages 90–99. Springer, 2007.

H. Quan, D. Srinivasan, and A. Khosravi. Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2):303–315, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL `http://www.R-project.org/`.

G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198, 2012.

T. S. Sethi and M. Kantardzic. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, 82:77–99, 2017.

D. L. Shrestha and D. P. Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural Networks*, 19(2): 225–235, 2006.

P. Sobhani and H. Beigy. New drift detection method for data streams. *Adaptive and Intelligent Systems*, 6943:88–97, 2011.

B. Whitcher. *waveslim: Basic wavelet routines for one-, two- and three-dimensional signal processing*, 2013. URL `http://CRAN.R-project.org/package=waveslim`. R package version 1.7.3.

D. Wuertz, T. Setz, Y. Chalabi, C. Boudt, P. Chausse, and M. Miklovac. *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*, 2019. URL `https://CRAN.R-project.org/package=fGarch`. R package version 3042.83.1.

S.-I. Yoshida, K. Hatano, E. Takimoto, and M. Takeda. Adaptive online prediction using weighted windows. *IEICE Transactions*, 94-D:1917–1923, 2011.

X. Zhao, S. Barber, C. C. Taylor, and Z. Milan. Classification tree methods for panel data using wavelet-transformed time series. *Computational Statistics & Data Analysis*, 127:204–216, 2018.