Evaluating time series forecasting models: an empirical study on performance estimation methods

Vitor Cerqueira[1] [ORCID] **· Luis Torgo[1,2,3] · Igor Mozetič[4]**

Abstract

Performance estimation aims at estimating the loss that a predictive model will incur on unseen data. This process is a fundamental stage in any machine learning project. In this paper we study the application of these methods to time series forecasting tasks. For independent and identically distributed data the most common approach is cross-validation. However, the dependency among observations in time series raises some caveats about the most appropriate way to estimate performance in this type of data. Currently, there is no consensual approach. We contribute to the literature by presenting an extensive empirical study which compares different performance estimation methods for time series forecasting tasks. These methods include variants of cross-validation, out-of-sample (holdout), and prequential approaches. Two case studies are analysed: One with 174 real-world time series and another with three synthetic time series. Results show noticeable differences in the performance estimation methods in the two scenarios. In particular, empirical experiments suggest that blocked cross-validation can be applied to stationary time series. However, when the time series are non-stationary, the most accurate estimates are produced by out-of-sample methods, particularly the holdout approach repeated in multiple testing periods.

Keywords Performance estimation · Model selection · Cross validation · Time series · Forecasting

✉ Vitor Cerqueira
  cerqueira.vitormanuel@gmail.com

  Luis Torgo
  ltorgo@dal.ca

  Igor Mozetič
  igor.mozetic@ijs.si

[1] LIAAD-INESC TEC, Porto, Portugal

[2] University of Porto, Porto, Portugal

[3] Dalhousie University, Halifax, Canada

[4] Jozef Stefan Institute, Ljubljana, Slovenia

1 Introduction

Performance estimation denotes the process of using the available data to estimate the loss that a predictive model will incur in new, yet unseen, observations. Estimating the performance of a predictive model is a fundamental stage in any machine learning project. Practitioners carry out performance estimation to select the most appropriate model and its parameters. Crucially, the process of performance estimation is one of the most reliable approaches to analyse the generalisation ability of predictive models. Such analysis is important not only to select the best model, but also to verify that the respective model solves the underlying predictive task.

Choosing an appropriate performance estimation method usually depends on the characteristics of the data set. When observations are independent and identically distributed (i.i.d.), cross validation is one of the most widely used approaches (Geisser 1975). One of the reasons for its popularity is its efficient use of data (Arlot and Celisse 2010). However, many data sets in real-world applications are not i.i.d., for example, time series. Time series forecasting is an important machine learning problem. This task has a high practical utility in organizations across many domains of application.

When the observations in a data set are not i.i.d., the standard cross-validation approach is not directly applicable. Cross-validation breaks the temporal order of time series observations, which may lead to unrealistic estimates of predictive performance. In effect, when dealing with this type of data sets, practitioners typically apply an out-of-sample (also known as holdout) approach to estimate the performance of predictive models. Essentially, the predictive model is built in the initial part of the data. The subsequent observations are used for testing. Notwithstanding, there are particular scenarios in which cross-validation may be beneficial. For example, when the time series is stationary, or the sample size is small and data efficiency becomes important (Bergmeir et al. 2018).

Several approaches have been developed in recent decades to estimate the performance of forecasting models. However, there is no consensual approach. In this context, we contribute to the literature by carrying out an extensive empirical study which compares several approaches which are often used in practice.

We compare a set of estimation methods which can be broadly split into three categories: out-of-sample (OOS), prequential, and cross-validation (CVAL). OOS approaches are commonly used to estimate the performance of models when the data comprises some degree of temporal dependency. The core idea behind these approaches is to leave the last part of the data for testing. Although this type of approaches do not make a complete use of the available data, they preserve the temporal order of observations. This aspect may be important to cope with the temporal correlation among consecutive observations, and to mimic a realistic deployment scenario. Prequential approaches (Dawid 1984) are also common in incremental or high-frequency data sets such as data streams (Gama et al. 2014). Prequential denotes an evaluation procedure in which an observation (or a set of observations) is first used for testing, and then to re-train or update the model.

CVAL approaches make a more efficient use of the available data as each observation is used to both train and test a model over the different iterations of the procedure. This property may be beneficial in some scenarios in which the sample size is small (Bergmeir et al. 2018). Although the classical K-fold cross validation assumes the data to be i.i.d., some variants of it have been developed which mitigate this problem. In effect, some of these variants have been shown to provide better estimate of performance relative to OOS methods in time series tasks (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018).

The key factor that distinguishes OOS and prequential approaches from CVAL ones is that the former always preserve the temporal order of observations. This means that a model is never tested on past data relative to the training data set. The central research question we address in this paper is the following: How do estimation methods compare with each other in terms of performance estimation ability for different types of time series data? To accomplish this, we applied different estimation methods in two case studies. One is comprised of 174 real-world time series with potential non-stationarities and the other is a stationary synthetic environment (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018).

The results suggest that, as Bergmeir et al. point out (Bergmeir et al. 2018), cross-validation approaches can be applied to stationary time series. However, many real-world phenomena comprise complex non-stationary sources of variation. In these cases, applying holdout in several testing periods shows the best performance.

This paper is an extension to an article already published (Cerqueira et al. 2017). In this work, we substantially increase the experimental setup both in methods and data sets used; we provide additional analyses such as the impact of stationarity; and a more in-depth and critical discussion of the results.

This paper is structured as follows. The literature on performance estimation for time series forecasting tasks is reviewed in Sect. 2. Materials and methods are described in Sect. 3, including the predictive task, time series data sets, performance estimation methodology, and experimental design. The results of the experiments are reported in Sect. 4. A discussion of our results is carried out in Sect. 5. Finally, the conclusions of our empirical study are provided in Sect. 6.

In the interest of reproducibility, the methods and data sets are publicly available.[1]

2 Background

In this section we provide a background to this paper. We review the typical estimation methods used in time series forecasting and explain the motivation for this study.

In general, performance estimation methods for time series forecasting tasks are designed to cope with the dependence between observations. This is typically accomplished by having a model tested on observations future to the ones used for training.

2.1 Out-of-sample (OOS) approaches

When using OOS performance estimation procedures, a time series is split into two parts: an initial fit period in which a model is trained, and a subsequent (temporally) testing period held out for estimating the loss of that model. This simple approach (`Hold-out`) is depicted in Fig. 1. However, within this type of procedure one can adopt different strategies regarding training/testing split point, growing or sliding window settings, and eventual update of the models. In order to produce a robust estimate of predictive performance, (Tashman 2000) recommends employing these strategies in multiple test periods. One might create different sub-samples according to, for example, business cycles (Fildes 1989). For a more general setting one can also adopt a randomized approach. This is similar to random sub-sampling (or repeated holdout) in the sense that they consist of repeating a learning plus testing cycle several times using different, but possibly overlapping

---

[1] https://github.com/vcerqueira/performance_estimation.

Fig. 1 Simple out-of-sample procedure: an initial part of the available observations are used for fitting a predictive model. The last part of the data is held out, where the predictive model is tested
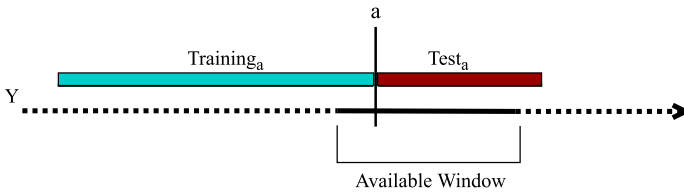

Fig. 2 Example of one iteration of the repeated holdout procedure. A point a is chosen from the available window. Then, a previous part of observations are used for training, while a subsequent part of observations are used for testing
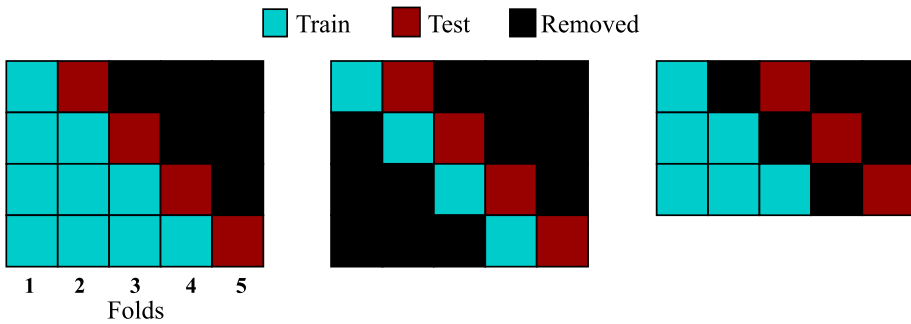

Fig. 3 Variants of prequential approach applied in blocks for performance estimation. This strategy can be applied using a growing window (left, right), or a sliding window (middle). One can also introduce a gap between the training and test sets

data samples (`Rep-Holdout`). This idea is illustrated in Fig. 2, where one iteration of a repeated holdout is shown. A point a is randomly chosen from the available sampling window (constrained by the training and testing sizes) of a time series Y. This point then marks the end of the training set, and the start of the testing set.

2.2 Prequential

OOS approaches are similar to prequential or interleaved-test-then-train evaluation (Dawid 1984). Prequential is typically used in data streams mining. The idea is that each observation is first used to test the model, and then to train the model. This can be applied in blocks of sequential instances (Modha and Masry 1998). In the initial iteration, only the first two blocks are used, the first for training and the second for testing. In the next iteration, the second block is merged with the first, and the third block is used for test. This procedure continues until all blocks are tested (`Preq-Bls`). This procedure is exemplified in the left side of Fig. 3, in which the data is split into 5 blocks.
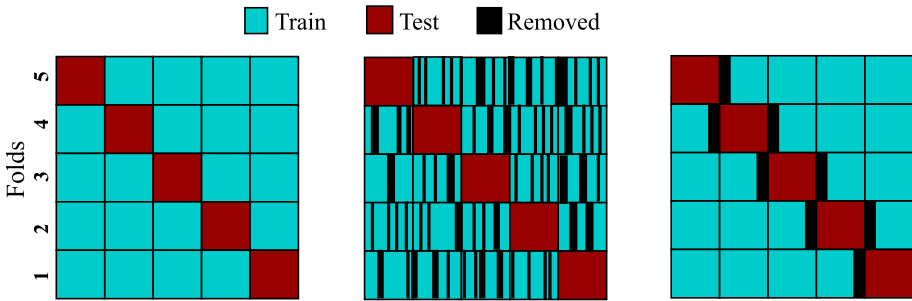
Fig. 4 Variants of cross-validation estimation procedures

A variant of this idea is illustrated in the middle scheme of Fig. 3. Instead of merging the blocks after each iteration (growing window), one can forget the older blocks in a sliding window fashion (`Preq-Sld-Bls`). This idea is typically adopted when past data becomes deprecated, which is common in non-stationary environments. Another variant of the prequential approach is represented in the right side of Fig. 3. This illustrates a prequential approach applied in blocks, where a gap block is introduced (`Preq-Bls-Gap`). The rationale behind this idea is to increase the independence between training and test sets.

The prequential approaches can be regarded as variations of the holdout procedure applied in multiple testing periods (`Rep-Holdout`). The core distinction is that the train and test prequential splits are not randomized, but pre-determined by the number of blocks or repetitions.

2.3 Cross-validation approaches

The typical approach when using K-fold cross-validation is to randomly shuffle the data and split it in K equally-sized folds or blocks. Each fold is a subset of the data comprising t/K randomly assigned observations, where t is the number of observations. After splitting the data into K folds, each fold is iteratively picked for testing. A model is trained on K-1 folds and its loss is estimated on the left out fold (`CV`). In fact, the initial random shuffle of observations before splitting into different blocks is not intrinsic to cross-validation (Geisser 1975). Notwithstanding, the random shuffling is a common practice among data science professionals. This approach to cross-validation is illustrated in the left side of Fig. 4.

2.3.1 Variants designed for time-dependent data

Some variants of K-fold cross-validation have been proposed specially designed for dependent data, such as time series (Arlot and Celisse 2010). However, theoretical problems arise by applying this technique directly to this type of data. The dependency among observations is not taken into account since cross-validation assumes the observations to be i.i.d.. This might lead to overly optimistic estimations and consequently, poor generalisation ability of predictive models on new observations. For example, prior work has shown that cross-validation yields poor estimations for the task of choosing the bandwidth

of a kernel estimator in correlated data (Hart and Wehrly 1986). To overcome this issue and approximate independence between the training and test sets, several methods have been proposed as variants of this procedure. We will focus on variants designed to cope with temporal dependency among observations.

The Blocked Cross-Validation (Snijders 1988) (`CV-Bl`) procedure is similar to the standard form described above. The difference is that there is no initial random shuffling of observations. In time series, this renders K blocks of contiguous observations. The natural order of observations is kept within each block, but broken across them. This approach to cross-validation is also illustrated in the left side of Fig. 4. Since the random shuffle of observations is not being illustrated, the figure for `CV-Bl` is identical to the one shown for `CV`.

The Modified CV procedure (McQuarrie and Tsai 1998) (`CV-Mod`) works by removing observations from the training set that are correlated with the test set. The data is initially randomly shuffled and split into K equally-sized folds similarly to K-fold cross-validation. Afterwards, observations from the training set within a certain temporal range of the observations of the test set are removed. This ensures independence between the training and test sets. However, when a significant amount of observations are removed from training, this may lead to model under-fit. This approach is also described as non-dependent cross-validation (Bergmeir and Benítez 2012). The graph in the middle of Fig. 4 illustrates this approach.

The hv-Blocked Cross-Validation (`CV-hvBl`) proposed by Racine (2000) extends blocked cross-validation to further increase the independence among observations. Specifically, besides blocking the observations in each fold, which means there is no initial randomly shuffle of observations, it also removes adjacent observations between the training and test sets. Effectively, this creates a gap between both sets. This idea is depicted in the right side of Fig. 4.

2.3.2 Usefulness of cross-validation approaches

Recently there has been some work on the usefulness of cross-validation procedures for time series forecasting tasks. Bergmeir and Benítez (2012) present a comparative study of estimation procedures using stationary time series. Their empirical results show evidence that in such conditions cross-validation procedures yield more accurate estimates than an OOS approach. Despite the theoretical issue of applying standard cross-validation, they found no practical problem in their experiments. Notwithstanding, the Blocked cross-validation is suggested for performance estimation using stationary time series.

Bergmeir et al. (2014) extended their previous work for directional time series forecasting tasks. These tasks are related to predicting the direction (upward or downward) of the observable. The results from their experiments suggest that the hv-Blocked CV procedure provides more accurate estimates than the standard out-of-sample approach. These were obtained by applying the methods on stationary time series.

Finally, Bergmeir et al. (2018) present a simulation study comparing standard cross-validation to out-of-sample evaluation. They used three data generating processes and performed 1000 Monte Carlo trials in each of them. For each trial and generating process, a stationary time series with 200 values was created. The results from the simulation suggest that cross-validation systematically yields more accurate estimates, provided that the model is correctly specified.

In a related empirical study (Mozetič et al. 2018), Mozetič et al. compare estimation procedures on several large time-ordered Twitter datasets. They find no significant difference between the best cross-validation and out-of-sample evaluation procedures. However, they do

find that standard, randomized cross-validation is significantly worse than the blocked cross-validation, and should not be used to evaluate classifiers in time-ordered data scenarios.

Despite the results provided by these previous works we argue that they are limited in two ways. First, the used experimental procedure is biased towards cross-validation approaches. While these produce several error estimates (one for each fold), the OOS approach is evaluated in a one-shot estimation, where the last part of the time series is withheld for testing. OOS methods can be applied in several windows for more robust estimates, as recommended by Tashman (2000). By using a single origin, one is prone to particular issues related to that origin.

Second, the results are based on stationary time series, most of them artificial. Time series stationarity is equivalent to identical distribution in the terminology of more traditional predictive tasks. Hence, the synthetic data generation processes and especially the stationary assumption limit interesting patterns that can occur in real-world time series. Our working hypothesis is that in more realistic scenarios one is likely to find time series with complex sources of non-stationary variations.

In this context, this paper provides an extensive comparative study using a wide set of methods for evaluating the performance of univariate time series forecasting models. The analysis is carried out using a real-world scenario as well as a synthetic case study used in the works described previously (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018).

2.4 Related work on performance estimation with dependent data

The problem of performance estimation has also been under research in different scenarios. While we focus on time series forecasting problems, the following works study performance estimation methods in different predictive tasks.

2.4.1 Spatio-temporal dependencies

Geo-referenced time series are becoming more prevalent due to the increase of data collection from sensor networks. In these scenarios, the most appropriate estimation procedure is not obvious as spatio-temporal dependencies are at play. Oliveira et al. (2018) presented an extensive empirical study of performance estimation for forecasting problems with spatio-temporal time series. The results reported by the authors suggest that both cross-validation and out-of-sample methods are applicable in these scenarios. Like previous work in time-dependent domains (Bergmeir and Benítez 2012; Mozetič et al. 2018), Oliveira et al. suggest the use of blocking when using a cross-validation estimation procedure.

2.4.2 Data streams mining

Data streams mining is concerned with predictive models that evolve continuously over time in response to concept drift (Gama et al. 2014). Gama et al. (2013) provide a thorough overview of the evaluation of predictive models for data streams mining. The authors defend the usage of the prequential estimator with a forgetting mechanism, such as a fading factor or a sliding window.

This work is related to ours in the sense that it deals with performance estimation using time-dependent data. Notwithstanding, the paradigm of data streams mining is in line with sequential analysis (Wald 1973). As such, the assumption is that the sample size is not fixed in advance, and predictive models are evaluated as observations are collected. In our

setting, given a time series data set, we want to estimate the loss that a predictive models will incur in unseen observations future to that data set.

3 Materials and methods

In this section we present the materials and methods used in this work. First, we define the prediction task. Second, the time series data sets are described. We then formalize the methodology employed for performance estimation. Finally, we overview the experimental design.

3.1 Predictive task definition

A time series is a temporal sequence of values $Y = \{y_1, y_2, \dots, y_t\}$, where y_i is the value of Y at time i and t is the length of Y. We remark that we use the term time series assuming that Y is a numeric variable, i.e., $y_i \in \mathbb{R}, \forall\, y_i \in Y$.

Time series forecasting denotes the task of predicting the next value of the time series, y_{t+1}, given the previous observations of Y. We focus on a purely auto-regressive modelling approach, predicting future values of time series using its past lags.

To be more precise, we use time delay embedding (Takens 1981) to represent Y in an Euclidean space with embedding dimension p. Effectively, we construct a set of observations which are based on the past p lags of the time series. Each observation is composed of a feature vector $x_i \in \mathbb{X} \subset \mathbb{R}^p$, which denotes the previous p values, and a target vector $y_i \in \mathbb{Y} \subset \mathbb{R}$, which represents the value we want to predict. The objective is to construct a model $f : \mathbb{X} \to \mathbb{Y}$, where f denotes the regression function.

Summarizing, we generate the following matrix:

$$Y_{[n,p]} = \begin{bmatrix} y_1 & y_2 & \cdots & y_{p-1} & y_p & y_{p+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{i-p+1} & y_{i-p+2} & \cdots & y_{i-1} & y_i & y_{i+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{t-p+1} & y_{t-p+2} & \cdots & y_{t-1} & y_t & y_{t+1} \end{bmatrix}$$

Taking the first row of the matrix as an example, the target value is y_{p+1}, while the attributes (predictors) are the previous p values $\{y_1, \dots, y_p\}$.

3.2 Time series data

Two different case studies are used to analyse the performance estimation methods: a scenario comprised of real-world time series and a synthetic setting used in prior works (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018) for addressing the issue of performance estimation for time series forecasting tasks.

3.2.1 Real-world time series

Regarding real-world time series, we use a set of time series from the benchmark database *tsdl* (Hyndman and Yang 2019). From this database, we selected all the univariate time

series with at least 500 observations and which have no missing values. This query returned 149 time series. We also included 25 time series used in previous work by Cerqueira et al. (2019). From the set of 62 time series used by the authors, we selected those with at least 500 observations and which were not originally from the *tsdl* database (which are already retrieved as described above). We refer to the work by Cerqueira et al. (2019) for a description of the time series. In summary, our database of real-world time series comprises 174 time series. The threshold of 500 observations is included so that learning algorithms have enough data to build a good predictive model. The 174 time series represent phenomena from different domains of applications. These include finance, physics, economy, energy, and meteorology. They also cover distinct sampling frequencies, such as hourly or daily. In terms of sample size, the distribution ranges from 506 to 23741 observations. However, we truncated to time series to a maximum of 4000 observations to speed up computations. The database is available online (c.f. footnote 1). We refer to the sources for further information on these time series Hyndman and Yang (2019); Cerqueira et al. (2019).

Stationarity

We analysed the stationarity of the time series comprising the real-world case study. Essentially, a time series is said to be stationary if its characteristics do not depend on the time that the data is observed (Hyndman and Athanasopoulos 2018). In this work we consider a stationarity of order 2. This means that a time series is considered stationary if it has constant mean, constant variance, and an auto-covariance that does not depend on time. Henceforth we will refer a time series as stationary if it is stationary of order 2.

In order to test if a given time series is stationary we follow the wavelet spectrum test described by Nason (2013). This test starts by computing an evolutionary wavelet spectral approximation. Then, for each scale of this approximation, the coefficients of the Haar wavelet are computed. Any large Haar coefficient is evidence of a non-stationarity. An hypothesis test is carried out to assess if a coefficient is large enough to reject the null hypothesis of stationarity. In particular, we apply a multiple hypothesis test with a Bonferroni correction and a false discovery rate (Nason 2013).

In Fig. 5 is shown an example of the application of the wavelet spectrum test to a nonstationary time series. In the graphic, each red horizontal arrow denotes a non-stationarity found by the test. The left-hand side axis denotes the scale of the time series. The
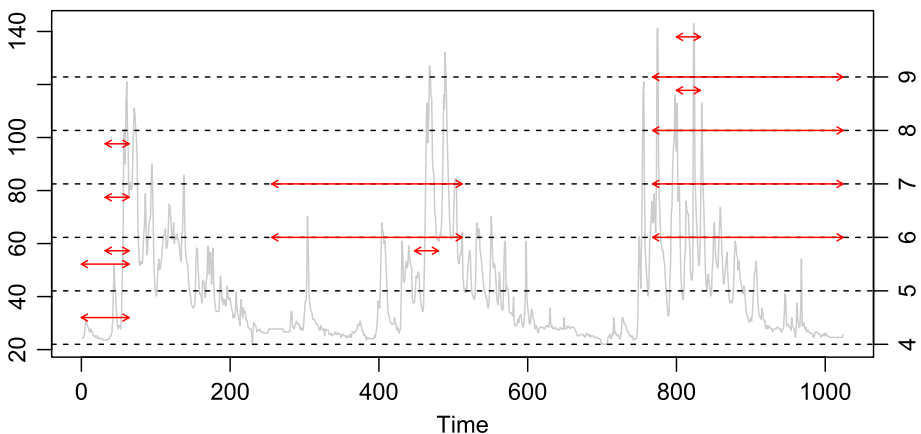

Fig. 5 Application of the wavelet spectrum test to a non-stationary time series. Each red horizontal arrow denote a non-stationarity identified by the test

right-hand axis represents the scale of the wavelet periodogram and where the non-stationarities are found. Finally, the lengths of the arrows denote the scale of the Haar wavelet coefficient whose null hypothesis was rejected. For a thorough description of this method we refer to the work by Nason (2013). Out of the 174 time series used in this work, 97 are stationary, while the remaining 77 are non-stationary.

3.2.2 Synthetic time series

We use three synthetic use cases defined in previous works by Bergmeir et al. (2014, (2018). The data generating processes are all stationary and are designed as follows:

S1   A stable auto-regressive process with lag 3, i.e., the next value of the time series is dependent on the past 3 observations;
S2   An invertible moving average process with lag 1;
S3   A seasonal auto-regressive process with lag 12 and seasonal lag 1.

For the first two cases, S1 and S2, real-valued roots of the characteristic polynomial are sampled from the uniform distribution $[-r; -1.1] \cup [1.1, r]$, where r is set to 5 (Bergmeir and Benítez 2012). Afterwards, the roots are used to estimate the models and create the time series. The data is then processed by making the values all positive. This is accomplished by subtracting the minimum value and adding 1. The third case S3 is created by fitting a seasonal auto-regressive model to a time series of monthly total accidental deaths in the USA (Brockwell and Davis 2013). For a complete description of the data generating process we refer to the work by Bergmeir and Benítez (2012); Bergmeir et al. (2018). Similarly to Bergmeir et al., for each use case we performed 1000 Monte Carlo simulations. In each repetition a time series with 200 values was generated.

3.3 Performance estimation methodology

Performance estimation addresses the issue of estimating the predictive performance of predictive models. Frequently, the objective behind these tasks is to compare different solutions for solving a predictive task. This includes selecting among different learning algorithms and hyper-parameter tuning for a particular one.

Training a learning model and evaluating its predictive ability on the same data has been proven to produce biased results due to overfitting (Arlot and Celisse 2010). In effect, several methods for performance estimation have been proposed in the literature, which use new data to estimate the performance of models. Usually, new data is simulated by splitting the available data. Part of the data is used for training the learning algorithm and the remaining data is used to test and estimate the performance of the model.

For many predictive tasks the most widely used of these methods is K-fold cross-validation (Stone 1974) (c.f. Sect. 2 for a description). The main advantages of this method is its universal splitting criteria and efficient use of all the data. However, cross-validation is based on the assumption that observations in the underlying data are independent. When this assumption is violated, for example in time series data, theoretical problems arise that prevent the proper use of this method in such scenarios. As we described in Sect. 2 several methods have been developed to cope with this issue, from out-of-sample approaches (Tashman 2000) to variants of the standard cross-validation, e.g., block cross-validation (Snijders 1988).
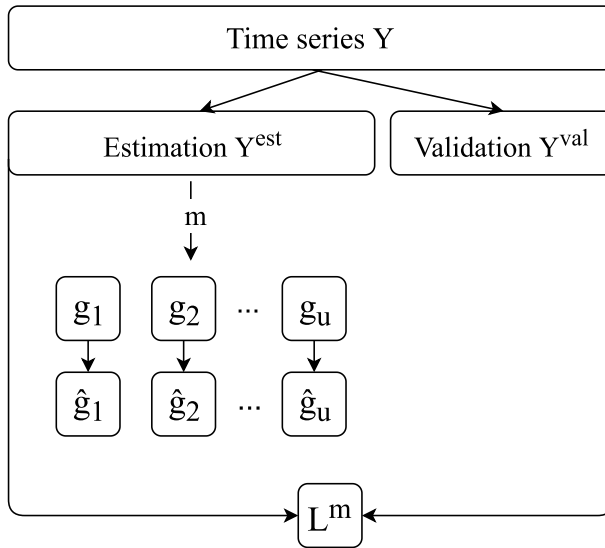
Fig. 6 Experimental comparison procedure (Cerqueira et al. 2017): a time series is split into an estimation set Y^{est} and a subsequent validation set Y^{val}. The first is used to estimate the error $\hat{g}$ that the model m will incur on unseen data, using u different estimation methods. The second is used to compute the actual error L^m incurred by m. The objective is to approximate L^m by $\hat{g}$ as well as possible

Our goal in this paper is to compare a wide set of estimation procedures, and test their suitability for different types of time series forecasting tasks. In order to emulate a realistic scenario we split each time series data in two parts. The first part is used to estimate the loss that a given learning model will incur on unseen future observations. This part is further split into training and test sets as described before. The second part is used to compute the true loss that the model incurred. This strategy allows the computation of unbiased estimates of error since a model is always tested on unseen observations.

The workflow described above is summarised in Fig. 6. A time series Y is split into an estimation set Y^{est} and a subsequent validation set Y^{val}. First, Y^{est} is used to calculate $\hat{g}$, which represents the estimate of the loss that a predictive model m will incur on future new observations. This is accomplished by further splitting Y^{est} into training and test sets according to the respective estimation procedure $g_i, i \in \{1, \ldots, u\}$. The model m is built on the training set and $\hat{g}_i$ is computed on the test set.

Second, in order to evaluate the estimates $\hat{g}_i$ produced by the methods g_i, $i \in \{1, \ldots, u\}$, the model m is re-trained using the complete set Y^{est} and tested on the validation set Y^{val}. Effectively, we obtain L^m, the ground truth loss that m incurs on new data.

In summary, the goal of an estimation method g_i is to approximate L^m by $\hat{g}_i$ as well as possible. In Sect. 3.4.3 we describe how to quantify this approximation.

3.4 Experimental design

The experimental design was devised to address the following research question: How do the predictive performance estimates of cross-validation methods compare to the estimates of out-of-sample approaches for time series forecasting tasks?

Existing empirical evidence suggests that cross-validation methods provide more accurate estimations than traditionally used OOS approaches in stationary time series forecasting (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018) (see Sect. 2). However, many real-world time series comprise complex structures. These include cues from the future that may not have been revealed in the past. In such cases, preserving the temporal order of observations when estimating the predictive ability of models may be an important component.

3.4.1 Embedding dimension and estimation set size

We estimate the optimal embedding dimension (p) (the value which minimises generalisation error) using the method of False Nearest Neighbours (Kennel et al. 1992). This method analyses the behaviour of the nearest neighbours as we increase p. According to Kennel et al. (1992), with a low sub-optimal p many of the nearest neighbours will be false. Then, as we increase p and approach an optimal embedding dimension those false neighbours disappear. We set the tolerance of false nearest neighbours to 1%. Regarding the synthetic case study, we fixed the embedding dimension to 5. The reason for this setup is to try to follow the experimental setup by Bergmeir et al. (2018).

The estimation set (Y^{est}) in each time series is the first 70% observations of the time series – see Fig. 6. The validation period is comprised of the subsequent 30% observations (Y^{val}). These values are typically used when partitioning data sets for performance estimation.

3.4.2 Estimation methods

In the experiments we apply a total of 11 performance estimation methods, which are divided into cross-validation, out-of-sample, and prequential approaches. The cross-validation methods are the following:

> `CV` Standard, randomized K-fold cross-validation;
> `CV-Bl` Blocked K-fold cross-validation;
> `CV-Mod` Modified K-fold cross-validation;
> `CV-hvBl` hv-Blocked K-fold cross-validation;

The out-of-sample approaches are the following:

- `Holdout` A simple OOS approach–the first 70% of Y^E is used for training and the subsequent 30% is used for testing;
- `Rep-Holdout` OOS tested in *nreps* testing periods with a Monte Carlo simulation using 70% of the total observations t of the time series in each test. For each period, a random point is picked from the time series. The previous window comprising 60% of t is used for training and the following window of 10% of t is used for testing.

Finally, we include the following prequetial approaches:

- `Preq-Bls` Prequential evaluation in blocks in a growing fashion;

- `Preq-Sld-Bls` Prequential evaluation in blocks in a sliding fashion–the oldest block of data is discarded after each iteration;
- `Preq-Bls-Gap]` Prequential evaluation in blocks in a growing fashion with a gap block–this is similar to the method above, but comprises a block separating the training and testing blocks in order to increase the independence between the two parts of the data;
- `Preq-Grow` and `Preq-Slide` As baselines we also include the exhaustive prequential methods in which an observation is first used to test the predictive model and then to train it. We use both a growing/landmark window (`Preq-Grow`) and a sliding window (`Preq-Slide`).

We refer to Sect. 2 for a complete description of the methods. The number of folds *K* or repetitions *nreps* in these methods is 10, which is a commonly used setting in the literature. The number of observations removed in `CV-Mod` and `CV-hvBl` (c.f. Sect. 2) is the embedding dimension *p* in each time series.

3.4.3 Evaluation metrics

Our goal is to study which estimation method provides a $\hat{g}$ that best approximates L^m. Let $\hat{g}_i^m$ denote the estimated loss by the learning model *m* using the estimation method *g* on the estimation set, and L^m denote the ground truth loss of learning model *m* on the test set. The objective is to analyze how well $\hat{g}_i^m$ approximates L^m. This is quantified by the absolute predictive accuracy error (APAE) metric and the predictive accuracy error (PAE) (Bergmeir et al. 2018):

$$\text{APAE} = |\hat{g}_i^m - L^m| \tag{1}$$

$$\text{PAE} = \hat{g}_i^m - L^m \tag{2}$$

The APAE metric evaluates the error size of a given estimation method. On the other hand, PAE measures the error bias, i.e., whether a given estimation method is under-estimating or over-estimating the true error.

Another question regarding evaluation is how a given learning model is evaluated regarding its forecasting accuracy. In this work we evaluate models according to root mean squared error (RMSE). This metric is traditionally used for measuring the differences between the estimated values and the actual values.

3.4.4 Learning algorithms

We applied the following learning algorithms:

RBR A rule-based regression algorithm from the *Cubist* R package (Kuhn et al. 2014), which is a variant of the model tree by Quinlan (1993). The main parameter, the number of committees (c.f. Kuhn et al. 2014), was set to 5.;
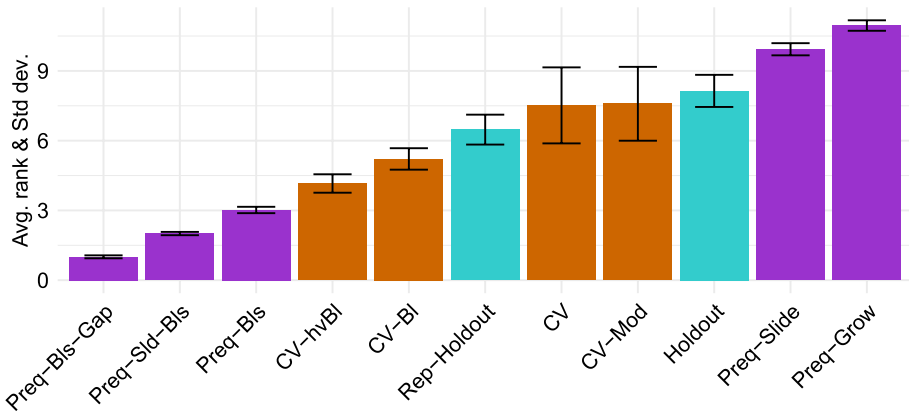
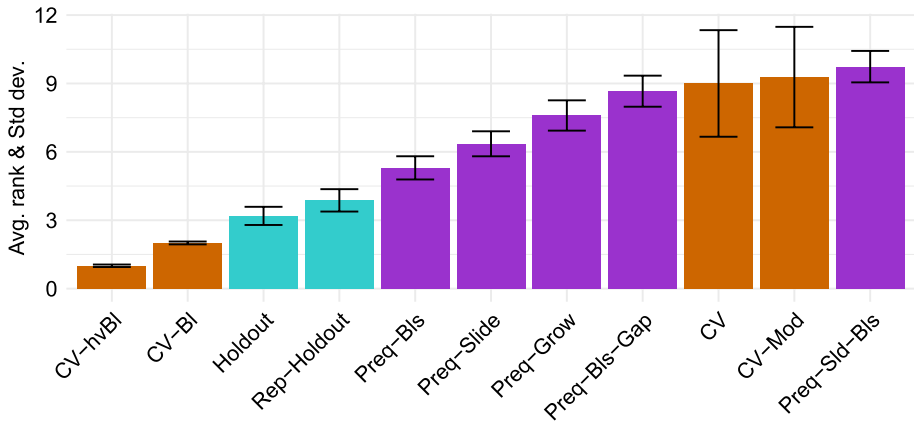**Fig. 7** Average rank and respective standard deviation of each estimation methods in case study S1 using the RBR learning method

RF A Random Forest algorithm, which is an ensemble of decision trees (Breiman 2001). We resort to the implementation from the *ranger* R package (Wright 2015). The number of trees in the ensemble was set to 100.;

GLM A generalized linear model (McCullagh 2019) regression with a Gaussian distribution and a Ridge penalty mixing. We used the implementation of the *glmnet* R package (Friedman et al. 2010).

The remaining parameters of each method were set to defaults. These three learning algorithms are widely used in regression tasks. For time series forecasting in particular, (Cerqueira et al. 2019) showed their usefulness when applied as part of a dynamic heterogeneous ensemble. In particular, the RBR method showed the best performance among 50 other approaches.

4 Empirical experiments

4.1 Results with synthetic case study

In this section we start by analysing the average rank, and respective standard deviation, of each estimation method and for each synthetic scenario (S1, S2, and S3), according to the metric APAE. For example, a rank of 1 in a given Monte Carlo repetition means that the respective method was the best estimator in that repetition. These analyses are reported in Figs. 7, 8, 9. This initial experiment is devised to reproduce the results by Bergmeir et al. (2018). Later, we will analyse how these results compare when using real-world time series.

The results shown by the average ranks corroborate those presented by Bergmeir and Benítez (2012), Bergmeir et al. (2014), Bergmeir et al. (2018). Cross-validation approaches, blocked ones in particular, perform better (i.e., show a lower average rank) relative to the simple out-of-sample procedure `Holdout`. This can be concluded from all three scenarios: S1, S2, and S3.

Fig. 8 Average rank and respective standard deviation of each estimation methods in case study S2 using the RBR learning method
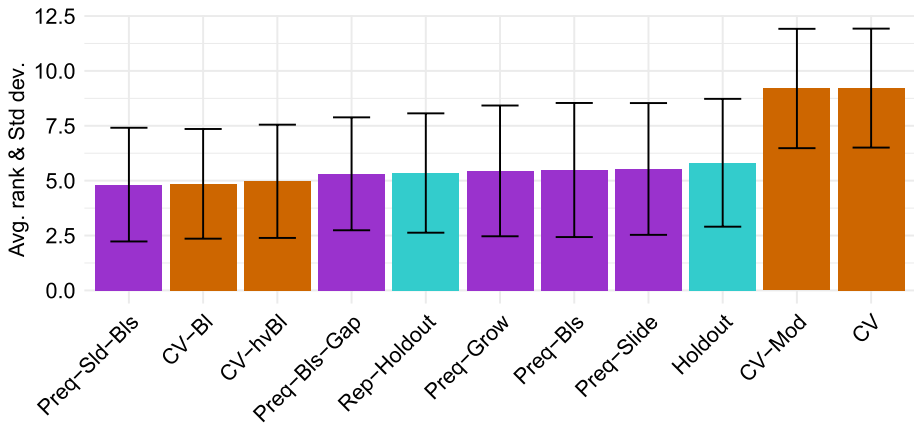

Fig. 9 Average rank and respective standard deviation of each estimation methods in case study S3 using the RBR learning method

Focusing on scenario S1, the estimation method with the best average rank is `Preq-Bls-Gap`, followed by the other two prequential variants (`Preq-Sld-Bls`, and `Preq-Bls`). Although the `Holdout` procedure is clearly a relatively poor estimator, the repeated holdout in multiple testing periods (`Rep-Holdout`) shows a better average rank than the standard cross validation approach (`CV`). Among cross validation procedures, `CV-hvBl` presents the best average rank.

Scenario S2 shows a seemingly different story relative to S1. In this problem, the blocked cross validation procedures show the best estimation ability. Among all, `CV-hvBl` shows the best average rank.

Regarding the scenario S3, the outcome is less clear than the previous two scenarios. The methods show a closer average rank among them, with large standard deviations. `Preq-Sld-Bls` shows the best estimation ability, followed by the two blocked cross-validation approaches, `CV-Bl` and `CV-hvBl`.
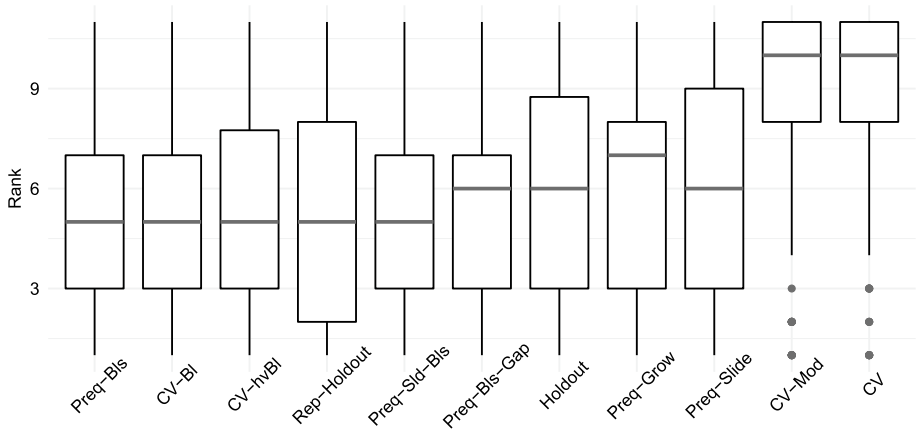
Fig. 10 Rank distribution of each estimation method across the 174 real-world time series

In summary, this first experiment corroborates the experiment carried out by Bergmeir et al. (2018). Notwithstanding, other methods that the authors did not test show an interesting estimation ability in these particular scenarios, namely the prequential variants. In this section, we focused on the RBR learning algorithm. In Sect. 1 of the appendix, we include the results for the GLM and RF learning methods. Overall, the results are similar across the learning algorithms.

The synthetic scenarios comprise time series that are stationary. However, real-world time series often comprise complex dynamics that break stationarity. When choosing a performance estimation method one should take this issue into consideration. To account for time series stationarity, in the next section we analyze the estimation methods using real-world time series.

4.2 Results with real-world data

In this section we analyze the performance estimation ability of each method using a case study which includes 174 real-world time series from different domains.

First, in Fig. 10, we show the rank distribution of each performance estimation method across the 174 time series. This figure shows a large dispersion of rank across the methods. This outcome indicates that there is no particular performance estimation method which is the most appropriate for all time series. Crucially, this result also motivates the need to study which time series characteristics (e.g. stationarity) most influence which method is more adequate for a given task.

4.2.1 Stationary time series

In Fig. 11, we start by analyzing the average rank, and respective standard deviation, of each estimation method using the APAE metric. We focus on the 97 stationary time series in the database.

Similarly to the synthetic case study, the blocked cross-validation approaches CV-Bl and CV-hvBl show a good estimation ability in terms of average rank. Conversely, the
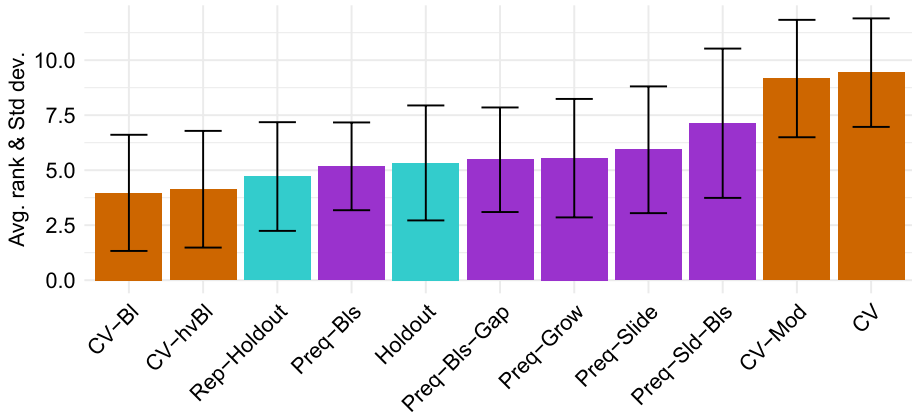
Fig. 11 Average rank and respective standard deviation of each estimation method in stationary real-world time series using the RBR learning method

other cross-validation approaches are the worst estimators. This outcome highlights the importance of blocking when using cross-validation. `Rep-Holdout` is the best estimator among the OOS approaches, while `Preq-Bls` shows the best score among the prequential methods.

We also study the statistical significance of the obtained results in terms of error size (APAE) according to a Bayesian analysis (Benavoli et al. 2017). Particularly, we applied the Bayes signed-rank test to compare pairs of methods across multiple problems. We arbitrarily define the *region of practical equivalence* (Benavoli et al. 2017) (ROPE) to be the interval [-2.5%, 2.5%] in terms of APAE. Essentially, this means that two methods show indistinguishable performance if the difference in performance between them falls within this interval. For a thorough description of the Bayesian analysis for comparing predictive models we refer to the work by Benavoli et al. (2017). In this analysis, it is necessary to use a scale invariant measure of performance. Therefore, we transform the metric APAE into the percentage difference of APAE relative to a baseline. In this experiment we fix the method `Rep-Holdout` as the baseline.

According to the illustration in Fig. 12, the probability of `Rep-Holdout` winning (i.e., showing a significantly better estimation ability) is generally larger than the opposite. The exception is when it is compared with the blocked cross-validation approaches `CV-Bl` and `CV-hvBl`.

For stationary time series, the blocked cross-validation approach `CV-Bl` seems to be the best estimation method among those analysed. The average rank analysis suggests that, on average, the relative position of this method is better than the other approaches. A more rigorous statistical analysis (using the Bayes signed-rank method) suggests that both `CV-Bl` and `CV-hvBl` are significantly better choices relative to `Rep-Holdout`.

4.2.2 Non-stationary time series

In Fig. 13 we present a similar analysis for the 77 non-stationary time series, whose results are considerably different relative to stationary time series. In this scenario, `Rep-Holdout` show the best average rank, followed by the blocked cross-validation approaches. Again, the standard cross-validation approach `CV` shows the worst score.
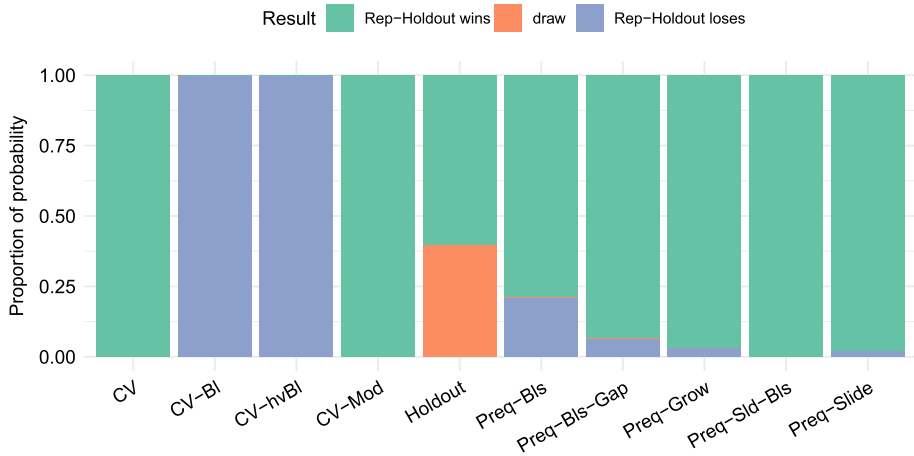
Fig. 12 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with stationary real-world time series. The probabilities are computed using the Bayes signed-rank test and using the RBR learning method
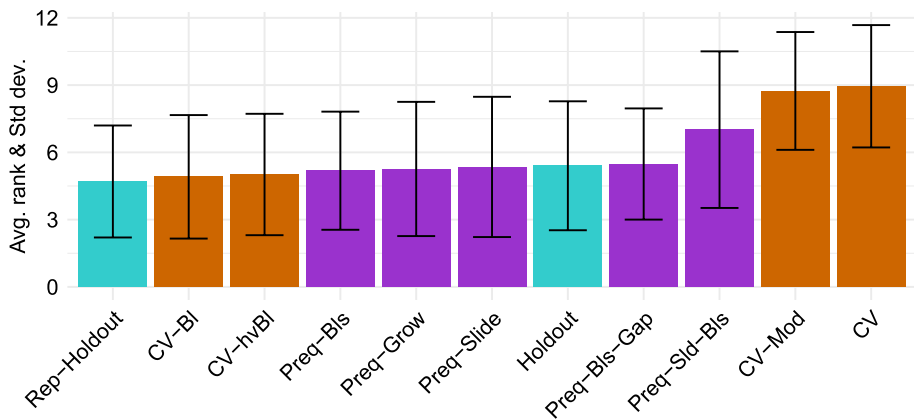

Fig. 13 Average rank and respective standard deviation of each estimation method in non-stationary real-world time series using the RBR learning method

Figure 14 shows the results of the Bayes signed-rank test. This analysis suggests that `Rep-Holdout` is a significantly better estimator relative to the other approaches when we are dealing with non-stationary time series.

To summarise, we compared the performance of estimation methods in a large set of real-world time series and controlled for stationarity. The results suggest that, for stationary time series, the blocked cross-validation approach `CV-Bl` is the best option. However, when the time series are non-stationary, the OOS approach `Rep-Holdout` is significantly better than the others.

On top of this, the results from the experiments also suggest the following outcomes:

– As Tashman pointed out (Tashman 2000), applying the holdout approach in multiple testing periods leads to a better performance relative to a single partition of the data set.

Fig. 14 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with non-stationary real-world time series. The probabilities are computed using the Bayes signed-rank test using the RBR learning method

> Specifically, `Rep-Holdout` shows a better performance relative to `Holdout` in the two scenarios;

- Prequential applied in blocks and in a growing window fashion (`Preq-Bls`) is the best prequential approach. Specifically, its average rank is better than the online approaches `Preq-Slide` and `Preq-Grow`; and the blocked approaches in which the window is sliding (as opposed to growing – `Preq-Sld-Bls`) or when a gap is introduced between the training and testing sets (`Preq-Bls-Gap`).

In the Sect. 1 of appendix, we show the results for the GLM and RF learning methods. The conclusions drawn from these algorithms are similar to what was reported above.

4.2.3 Error Bias

In order to study the direction of the estimation error, in Fig. 15 we present for each method the (log scaled) percentage difference between the estimation error and the true error according to the PAE metric. In this graphic, values below the zero line denote under-estimations of error, while values above the zero line represent over-estimations. In general, the estimation methods tend to over-estimate the error (i.e. are pessimistic estimators). The online prequential approaches `Preq-Slide` and `Preq-Grow`, and the non-blocked versions of cross-validation `CV` and `CV-Mod` tend to under-estimate the error (i.e. are optimistic estimators).

4.2.4 Impact of sample size

Preserving the temporal order of observations, albeit more realistic, comes at a cost since less data is available for estimating predictive performance. As (Bergmeir et al. 2018 argue, this may be important for small data sets, where a more efficient use of the data (e.g. `CV`) may be beneficial.
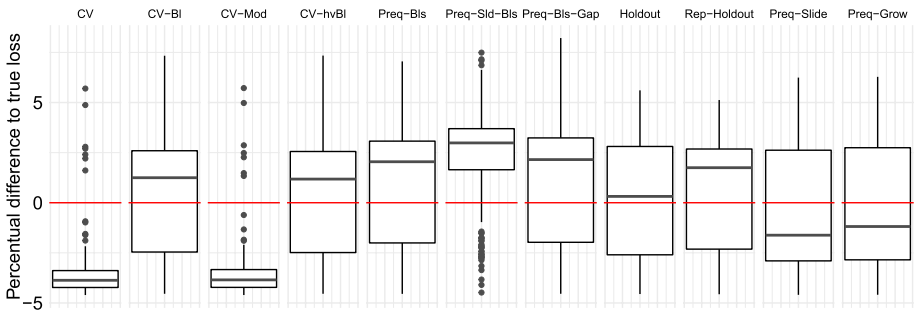
Fig. 15 Log percentage difference of the estimated loss relative to the true loss for each estimation method in the real-world case study, and using the RBR learning method. Values below the zero line represent under-estimations of error. Conversely, values above the zero line represent over-estimations of error

Previous work on time series forecasting has shown that sample size matters when selecting the predictive model (Cerqueira et al. 2019). Learning algorithms with a flexible functional form, e.g. decision trees, tend to work better for larger sample sizes relative to traditional forecasting approaches, for example, exponential smoothing.

We carried out an experiment to test whether the sample size of time series has any effect on the relative performance of the estimation methods. In this specific analysis, we focus on a subset of 90 time series (out of the 174 ones) with at least 1000 observations. Further, we truncated these to 1000 observations so all time series have equal size. Then, we proceed as follows. We repeated the performance estimation methodology described in Sect. 3.3 for increasing sample sizes. In the first iteration, each time series comprises an estimation set of 100 observations, and a validation set of 100 observations. The methodology is applied under these conditions. In the next iterations, the estimation set grows by 100 observations (to 200), and the validation set represents the subsequent 100 points. The process is repeated until the time series is fully processed.

We measure the average rank of each estimation method across the 90 time series after each iteration according to the APAE metric. These scores are illustrated in Fig. 16. The estimation set size in shown in the x-axis, while the average rank score is on the y-axis. Overall, the relative positions of each method remain stable as the sample size grows. This outcome suggests that this characteristic is not crucial when selecting the estimation method.

We remark that this analysis is restricted to the sample sizes tested (up to 1000 observations). For large scale data sets the recommendation by Dietterich (1998), and usually adopted in practice, is to apply a simple out-of-sample estimation procedure (`Holdout`).

4.2.5 Descriptive model

What makes an estimation method appropriate for a given time series is related to the characteristics of the data. For example, in the previous section we analyzed the impact that stationarity has in terms of what is the best estimation method.

The real-world time series case study comprises a set of time series from different domains. In this section we present, as a descriptive analysis, a tree-based model that relates some characteristics of time series according to the most appropriate estimation method for that time series. Basically, we create a predictive task in which the
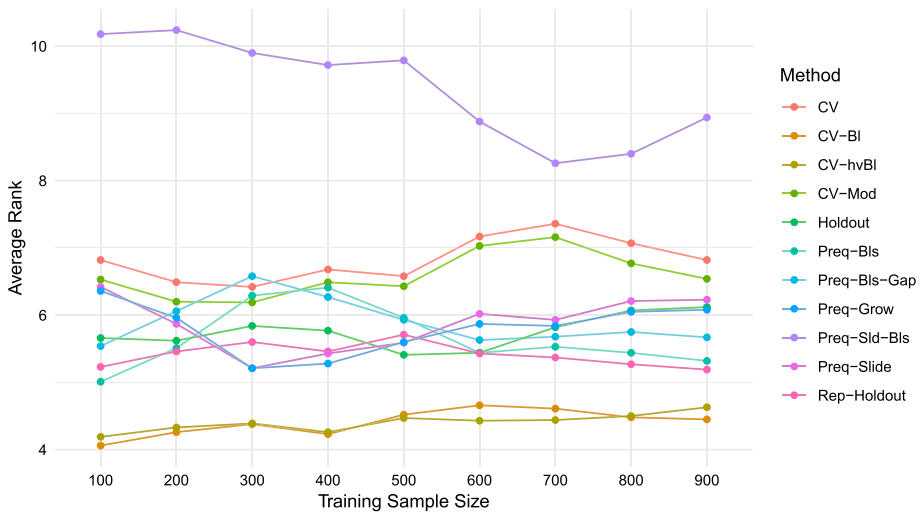
Fig. 16 Average rank of each performance estimation method with an increasing training sample size

attributes are some characteristics of a time series, and the categorical target variable is the estimation method that best approximates the true loss in that time series. We use CART (Breiman 2017) (classification and regression tree) algorithm for obtaining the model for this task. The characteristics used as predictor variables are the following summary statistics:

- *Trend*, estimating according to the ratio between the standard deviation of the time series and the standard deviation of the differenced time series;
- Skewness, for measuring the symmetry of the distribution of the time series;
- *Kurtosis*, as a measure of flatness of the distribution of the time series relative to a normal distribution;
- 5-th and 95-th Percentiles (Perc05, Perc95) of the standardized time series;
- Inter-quartile range (*IQR*), as a measure of the spread of the standardized time series;
- Serial correlation, estimated using a Box-Pierce test statistic;
- Long-range dependence, using a Hurst exponent estimation with wavelet transform;
- Maximum Lyapunov Exponent, as a measure of the level of chaos in the time series;
- a boolean variable, indicating whether or not the respective time series is stationary according to the wavelet spectrum test (Nason 2013).

These characteristics have been shown to be useful in other problems concerning time series forecasting (Wang et al. 2006). The features used in the final decision tree are written in boldface. The decision tree is shown in Fig. 17. The numbers below the name of the method in each node denote the number of times the respective method is best over the number of time series covered in that node.

Some of the estimation methods do not appear in the tree model. The tree leaves, which represent a decision, include the estimation methods `CV-Bl`, `Rep-Holdout`, `Preq-Slide`, and `Preq-Bls-Gap`.

The estimation method in the root node is `CV-Bl`, which is the method which is the best most of the times across the 174 time series. The first split is performed according to the
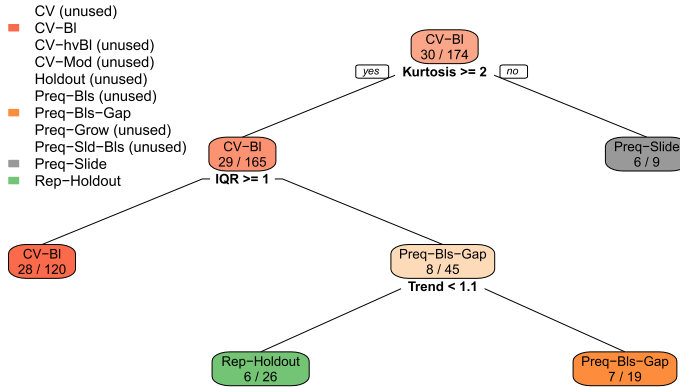
Fig. 17 Decision tree that maps the characteristics of time series to the most appropriate estimation method. Graphic created using the *rpart.plot* framework (Milborrow 2018)

kurtosis of time series. Basically, if the kurtosis is not above 2, the tree leads to a leaf node with `Preq-Slide` as the most appropriate estimation method. Otherwise, the tree continues with more tests in order to find the most suitable estimation method for each particular scenario.

5 Discussion

5.1 Impact of the results

In the experimental evaluation we compare several performance estimation methods in two distinct scenarios: (1) a synthetic case study in which artificial data generating processes are used to create stationary time series; and (2) a real-world case study comprising 174 time series from different domains. The synthetic case study is based on the experimental setup used in previous studies by Bergmeir et al. for the same purpose of evaluating performance estimation methods for time series forecasting tasks (Bergmeir and Benítez 2012; Bergmeir et al. 2014, 2018).

Bergmeir et al. show in previous studies (Bergmeir and Benitez 2011; Bergmeir and Benítez 2012) that the blocked form of cross-validation, denoted here as `CV-Bl`, yields more accurate estimates than a simple out-of-sample evaluation (`Holdout`) for stationary time series forecasting tasks. The method `CV` is also suggested to be "a better choice than OOS[`Holdout`] evaluation" as long as the data are well fitted by the model (Bergmeir et al. 2018). To some extent part of the results from our experiments corroborate these conclusions. Specifically, this is verified by the APAE incurred by the estimation procedures in the synthetic case studies. In our experiments we found out that, in the synthetic case study, prequential variants provide a good estimation ability, which is often better relative to cross validation variants. Furthermore, the results in the synthetic stationary case studies do not reflect those obtained using real-world time series. On one hand, we corroborate the conclusions of previous work (Bergmeir and Benítez 2012) that blocked cross-validation (`CV-Bl`) is applicable to stationary time series. On the other hand, when dealing with non-stationary data sets, holdout applied with multiple randomized testing periods (`Rep-Holdout`) provides the most accurate performance estimates.

In a real-world environment we are prone to deal with time series with complex structures and different sources of non-stationary variations. These comprise nuances of the future that may not have revealed themselves in the past (Tashman 2000). Consequently, we believe that in these scenarios, `Rep-Holdout` is a better option as performance estimation method relative to cross-validation approaches.

5.2 Scope of the real-world case study

In this work we center our study on univariate numeric time series. Nevertheless, we believe that the conclusions of our study are independent of this assumption and should extend for other types of time series. The objective is to predict the next value of the time series, assuming immediate feedback from the environment. Moreover, we focus on time series with a high sampling frequency, for example, hourly or daily data. The main reason for this is because high sampling frequency is typically associated with more data, which is important for fitting the predictive models from a machine learning point of view. Standard forecasting benchmark data are typically more centered around low sampling frequency time series, for example the M competition data (Makridakis et al. 1982).

5.3 Future work

We showed that stationarity is a crucial time series property to take into account when selecting the performance estimation method. On the other hand, data sample size appears not be have a significant effect, though the analysis is restricted to time series up to 1000 data points.

We studied the possibility of there being other time series characteristics which may be relevant for performance estimation. We built a descriptive model, which partitions the best estimation method according to different time series characteristics, such as kurtosis or trend. We believe that this approach may be interesting for further scientific enquiry. For example, one could leverage this type of model to automatically select the most appropriate performance estimation method. Such model could be embedded into an automated machine learning framework.

Our conclusions were drawn from a database of 174 time series from distinct domains of application. In future work, it would be interesting to carry out a similar analysis on time series from specific domains, for example, finance. The stock market contains rich financial data which attracts a lot of attention. Studying the most appropriate approach to evaluate predictive models embedded within trading systems is an interesting research direction.

6 Final remarks

In this paper we analyse the ability of different methods to approximate the loss that a given predictive model will incur on unseen data. We focus on performance estimation for time series forecasting tasks. Since there is currently no settled approach in these problems, our objective is to compare different available methods and test their suitability.

We analyse several methods that can be generally split into out-of-sample approaches and cross-validation methods. These were applied to two case studies: a synthetic environment with stationary time series and a real-world scenario with potential non-stationarities.

In stationary time series, the blocked cross-validation method (`CV-Bl`) is shown to have a competitive estimation ability. However, when non-stationarities are present, the

out-of-sample holdout procedure applied in multiple testing periods (`Rep-Holdout`) is a significantly better choice.

Appendix

Synthetic time series results with *GLM* and *RF*
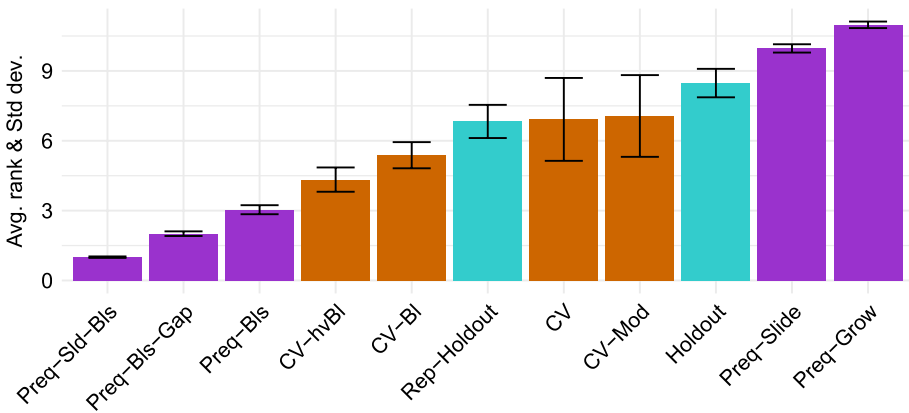
See Figs. 18, 19, 20, 21, 22, 23, 24.

Fig. 18 Average rank and respective standard deviation of each estimation methods in case study S1–using GLM learning method (complement to Fig. 7)
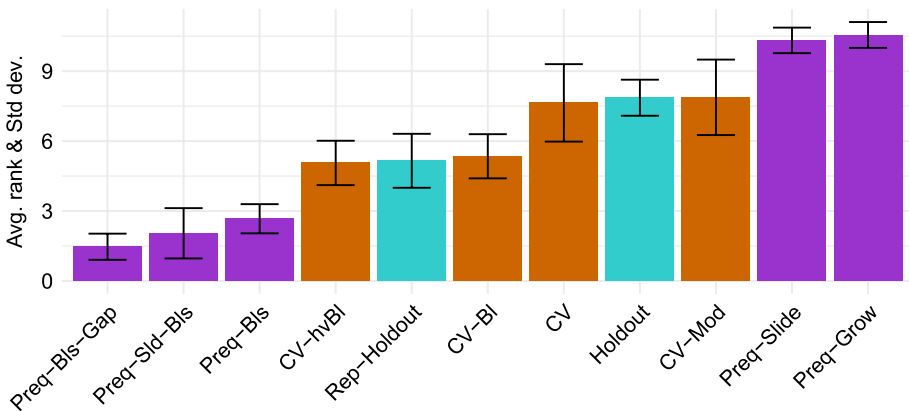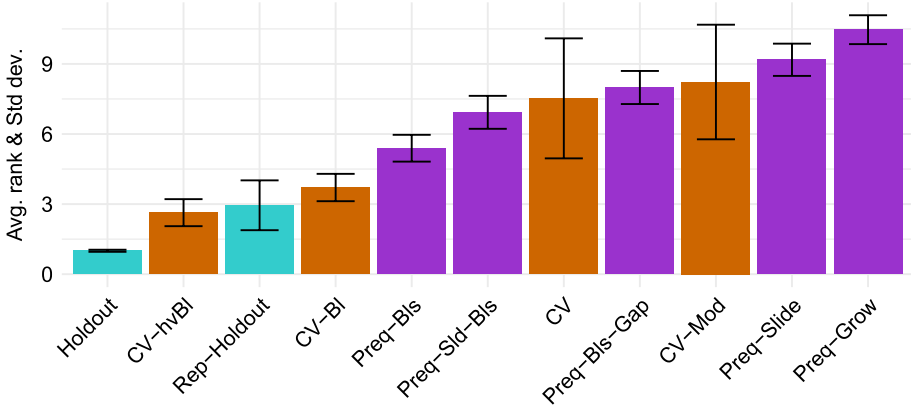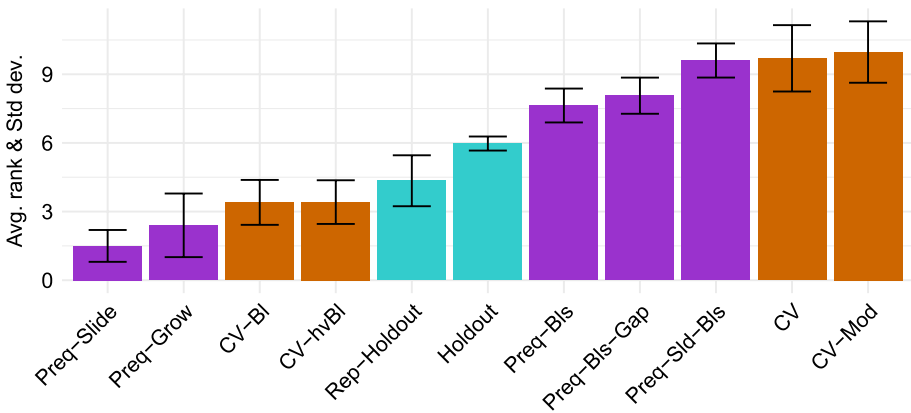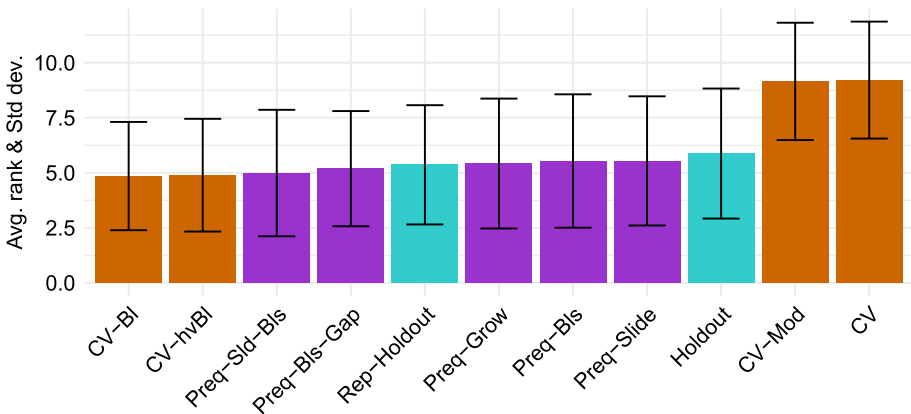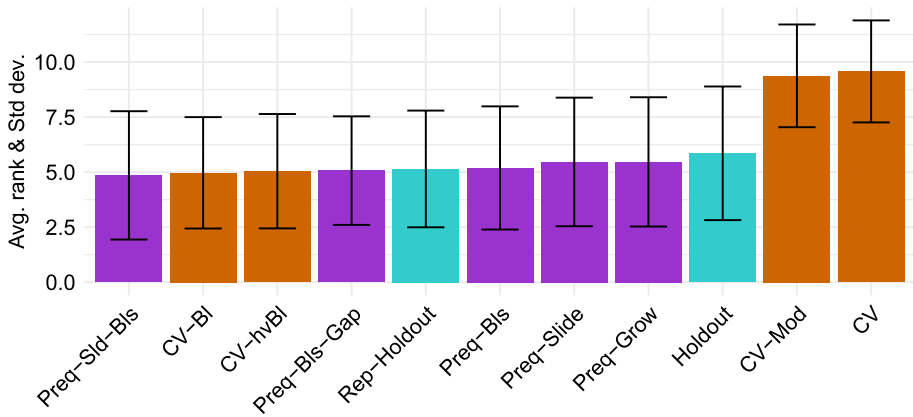

Fig. 19 Average rank and respective standard deviation of each estimation methods in case study S1–using RF learning method (complement to Figure 7)

Fig. 20 Average rank and respective standard deviation of each estimation methods in case study S2 - using GLM learning method (complement to Figure 8)

Fig. 21 Average rank and respective standard deviation of each estimation methods in case study S2–using RF learning method (complement to Fig. 8)

Fig. 22 Average rank and respective standard deviation of each estimation methods in case study S3–using GLM learning method (complement to Fig. 9)

Fig. 23 Average rank and respective standard deviation of each estimation methods in case study S3–using RF learning method (complement to Fig. 9)

Real-world time series results with *GLM* and *RF*

6.0.1 Stationary time series

See Figs. 24, 25, 26, 27.
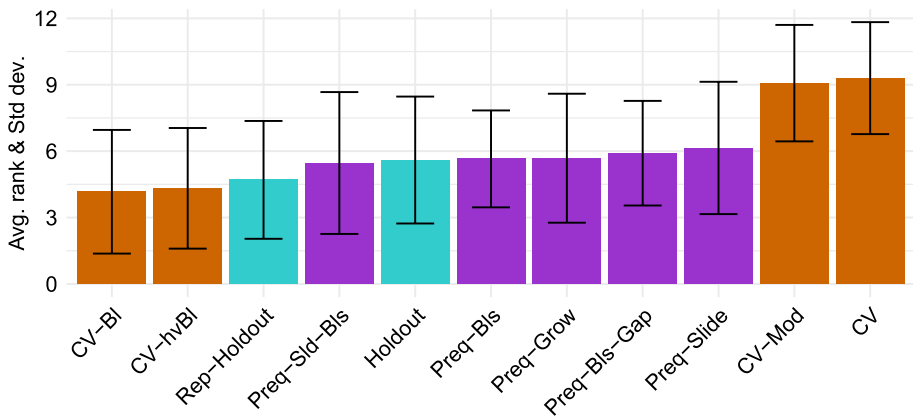
Fig. 24 Average rank and respective standard deviation of each estimation method in stationary real-world time series using the GLM learning method (complement to Fig. 11)
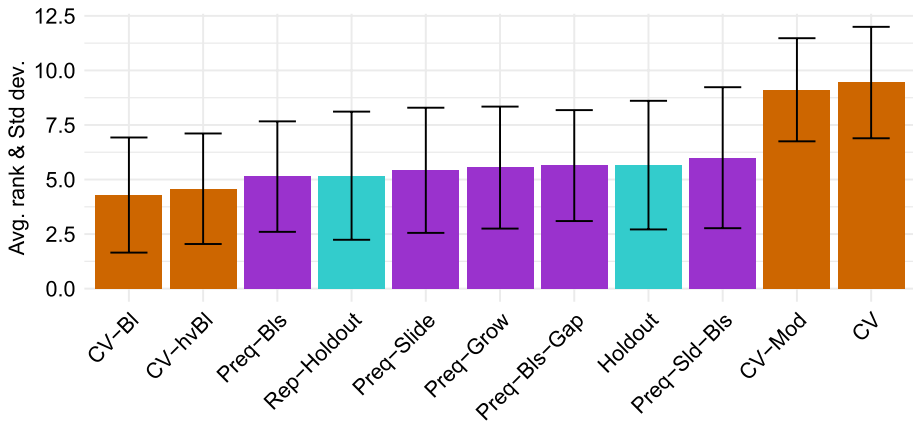
Fig. 25 Average rank and respective standard deviation of each estimation method in stationary real-world time series using the RF learning method (complement to Fig. 11)
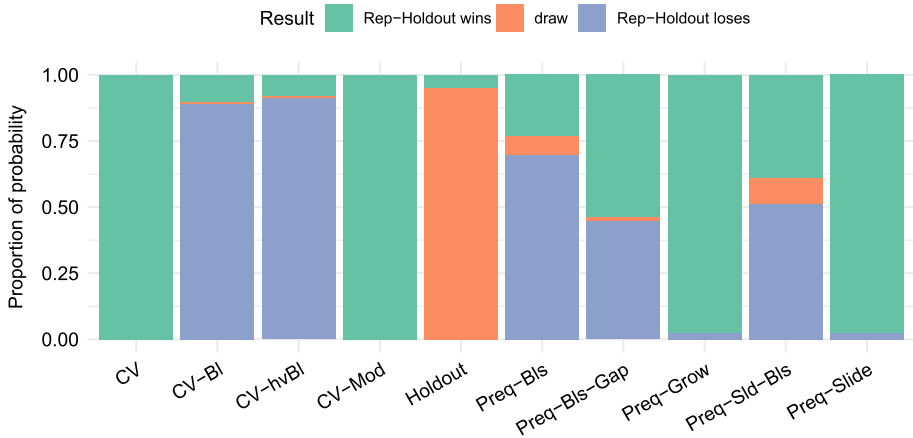

Fig. 26 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with stationary real-world time series. The probabilities are computed using the Bayes signed-rank test and using the GLM learning method (complement to Fig. 12)
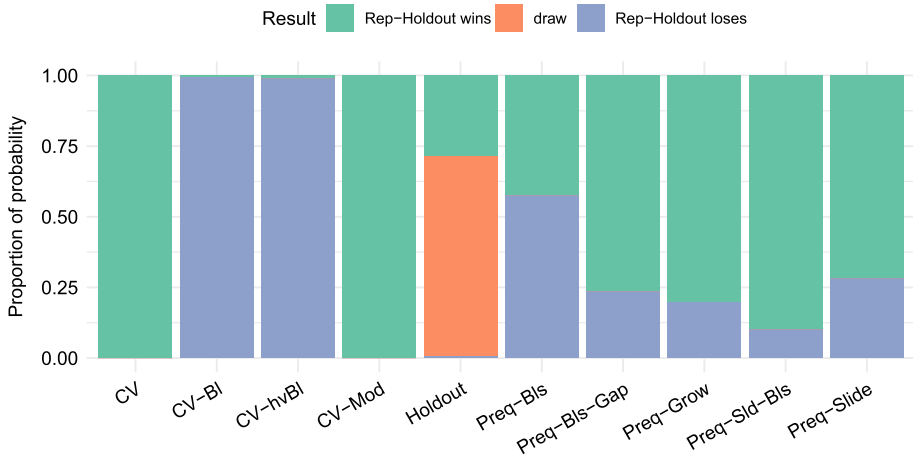
Fig. 27 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with stationary real-world time series. The probabilities are computed using the Bayes signed-rank test and using the RF learning method (complement to Fig. 12)

6.0.2 Non-stationary time series
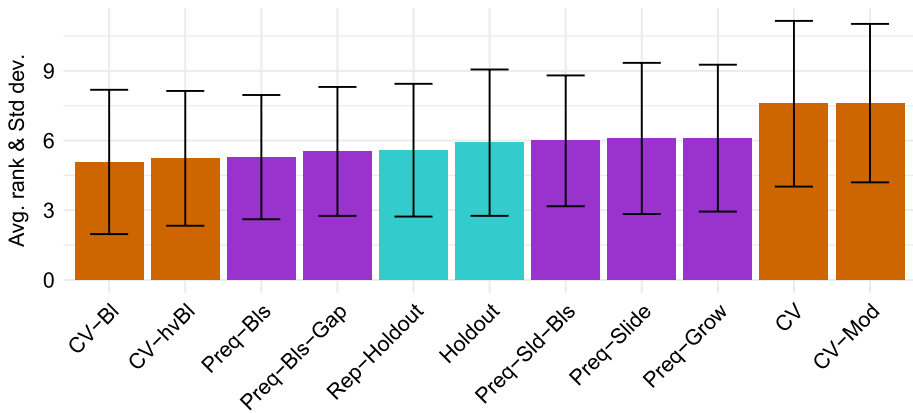
See Figs. 28, 29, 30, 31.

Fig. 28 Average rank and respective standard deviation of each estimation method in non-stationary real-world time series using the GLM learning method (complement to Fig. 13)
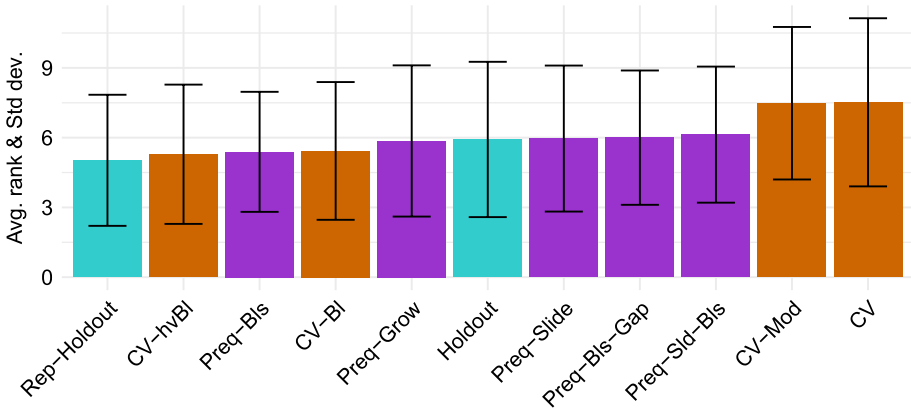
Fig. 29 Average rank and respective standard deviation of each estimation method in non-stationary real-world time series using the RF learning method (complement to Fig. 13)
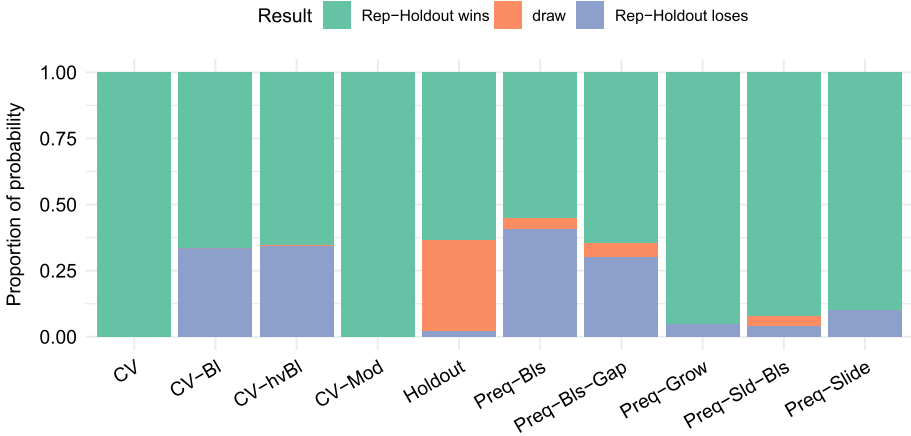

Fig. 30 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with non-stationary real-world time series. The probabilities are computed using the Bayes signed-rank test and using the GLM learning method (complement to Fig. 14)

Fig. 31 Proportion of probability of the outcome when comparing the performance estimation ability of the respective estimation method with the `Rep-Holdout` method with non-stationary real-world time series. The probabilities are computed using the Bayes signed-rank test and using the RF learning method (complement to Fig. 14)
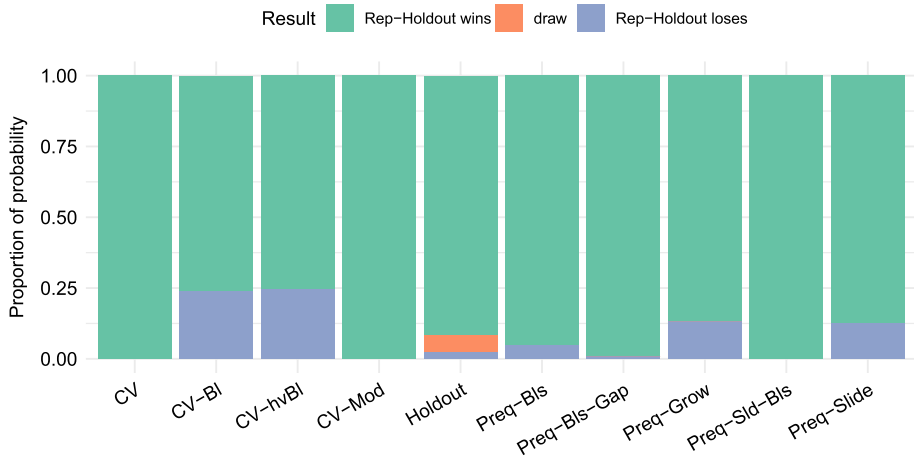
6.0.3 Error bias using the GLM and RF methods

See Figs. 32, 33.

Fig. 32 Log percentage difference of the estimated loss relative to the true loss for each estimation method in the RWTS case study, and using the GLM learning method. Values below the zero line represent underestimations of error. Conversely, values above the zero line represent over-estimations of error (complement to Fig. 15)
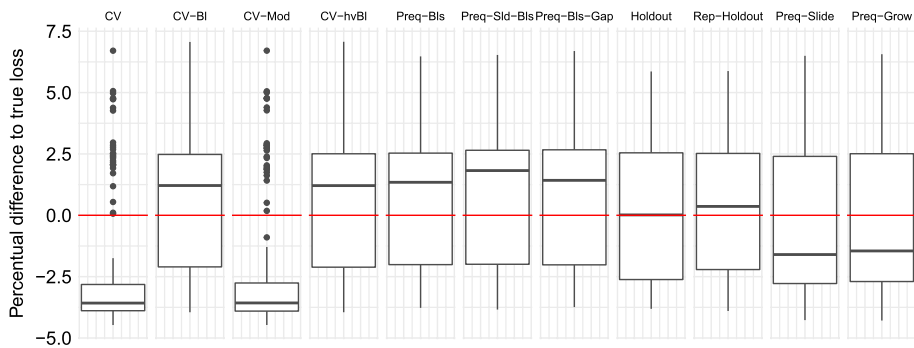
Fig. 33 Log percentage difference of the estimated loss relative to the true loss for each estimation method in the RWTS case study, and using the RF learning method. Values below the zero line represent under-estimations of error. Conversely, values above the zero line represent over-estimations of error (complement to Fig. 15)
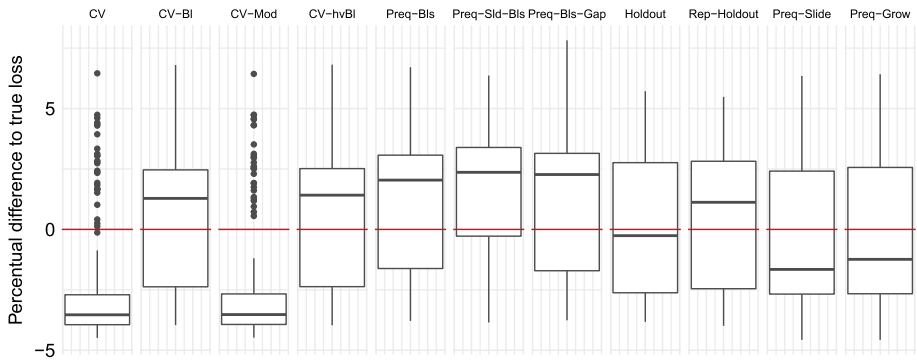
References

Arlot, S., Celisse, A., et al. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, *4*, 40–79.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, *18*(1), 2653–2688.

Bergmeir, C., & Benitez, J.M. (2011) Forecaster performance evaluation with cross-validation and variants. In: 2011 11th international conference on intelligent systems design and applications (ISDA), pp. 849–854. IEEE.

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213.

Bergmeir, C., Costantini, M., & Benítez, J. M. (2014). On the usefulness of cross-validation for directional forecast evaluation. *Computational Statistics & Data Analysis*, *76*, 132–143.

Bergmeir, C., Hyndman, R. J., & Koo, B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, *120*, 70–83.

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

Breiman, L. (2017). *Classification and Regression Trees*. New York: Routledge.

Brockwell, P.J., & Davis, R.A. (2013). Time series: theory and methods. Springer Science & Business Media, Berlin

Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2019). Arbitrage of forecasting experts. *Machine Learning*, *108*(6), 913–944.

Cerqueira, V., Torgo, L., Smailović, J., & Mozetič, I. (2017) A comparative study of performance estimation methods for time series forecasting. In *2017 IEEE international conference on data science and advanced analytics (DSAA)* (pp. 529–538). IEEE.

Cerqueira, V., Torgo, L., & Soares, C. (2019). Machine learning vs statistical methods for time series forecasting: Size matters. arXiv preprint arXiv:1909.13316.

Dawid, A. P. (1984). Present position and potential developments: Some personal views statistical theory the prequential approach. *Journal of the Royal Statistical Society: Series A (General)*, *147*(2), 278–290.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923.

Fildes, R. (1989). Evaluation of aggregate and individual forecast method selection rules. *Management Science*, *35*(9), 1056–1065.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317–346.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, *46*(4), 44.

Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of the American statistical Association*, *70*(350), 320–328.

Hart, J. D., & Wehrly, T. E. (1986). Kernel regression estimation using repeated measurements data. *Journal of the American Statistical Association*, *81*(396), 1080–1088.

Hyndman, R., & Yang, Y. (2019) tsdl: Time series data library. https://github.com/FinYang/tsdl.

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

Kennel, M. B., Brown, R., & Abarbanel, H. D. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, *45*(6), 3403.

Kuhn, M., Weston, S., & Keefer, C. (2014). code for Cubist by Ross Quinlan, N.C.C.: Cubist: Rule- and Instance-Based Regression Modeling. R package version 0.0.18.

Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., et al. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, *1*(2), 111–153.

McCullagh, P. (2019). *Generalized linear models*. New York: Routledge.

McQuarrie, A. D., & Tsai, C. L. (1998). *Regression and time series model selection*. Singapore: World Scientific.

Milborrow, S. (2018). rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'. https://CRAN.R-project.org/package=rpart.plot. R package version 3.0.6.

Modha, D. S., & Masry, E. (1998). Prequential and cross-validated regression estimation. *Machine Learning*, *33*(1), 5–39.

Mozetič, I., Torgo, L., Cerqueira, V., & Smailović, J. (2018). How to evaluate sentiment classifiers for Twitter time-ordered data? *PLoS ONE*, *13*(3), e0194317.

Nason, G. (2013). A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *75*(5), 879–904.

Oliveira, M., Torgo, L., & Costa, V.S. (2018) Evaluation procedures for forecasting with spatio-temporal data. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 703–718). Berlin: Springer.

Quinlan, J.R. (1993). Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning* (pp. 236–243).

Racine, J. (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of Econometrics*, *99*(1), 39–61.

Snijders, T.A. (1988). On cross-validation for predictor evaluation in time series. In *On model uncertainty and its statistical implications* (pp. 56–69). Berlin: Springer.

Stone, M. (1974). Cross-validation and multinomial prediction. Biometrika (pp. 509–515).

Takens, F. (1981). Dynamical systems and turbulence, Warwick 1980: Proceedings of a Symposium Held at the University of Warwick 1979/80, chap. Detecting strange attractors in turbulence, pp. 366–381. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/BFb0091924.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting*, *16*(4), 437–450.

Wald, A. (1973). *Sequential analysis*. Philadelphia: Courier Corporation.

Wang, X., Smith, K., & Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data Mining and Knowledge Discovery*, *13*(3), 335–364.

Wright MN (2015) Ranger: A fast implementation of random forests . R package version 0.3.0.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.