MONASH BUSINESS SCHOOL

ISSN 1440-771X

Department of Econometrics and Business Statistics

http://business.monash.edu/econometrics-and-business-statistics/research/publications

Meta-learning how to forecast time series

Thiyanga S Talagala, Rob J Hyndman,
George Athanasopoulos

February 2018

Working Paper ??/18

Meta-learning how to forecast time series

Thiyanga S Talagala
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: thiyanga.talagala@monash.edu
Corresponding author

Rob J Hyndman
Department of Econometrics and Business Statistics,
Monash University, VIC 3800, Australia.
Email: rob.hyndman@monash.edu

George Athanasopoulos
Department of Econometrics and Business Statistics,
Monash University, VIC 3145, Australia.
Email: george.athanasopoulos@monash.edu

7 February 2018

JEL classification: C10,C14,C22

Meta-learning how to forecast time series

Abstract

A crucial task in time series forecasting is the identification of the most suitable forecasting method. We present a general framework for forecast model selection using meta-learning. A Random Forest is used to predict the best forecasting method using only time series features. The proposed framework has been evaluated using time series from the M1 and M3 competitions, and is shown to yield accurate forecasts comparable to several benchmarks and other commonly used automated approaches of time series forecasting. A key advantage of our algorithm is that the time-consuming part of building the random forest can be handled in advance of the forecasting task. So when a time series

Keywords: Time Series, Forecasting, Time Series Features, Random Forest, Meta-learning, Algorithm selection problem

1 Introduction

Forecasting is a key activity for any business to operate efficiently. The rapid advances in computing technologies have enabled businesses to keep track of large number of time series variables. Hence, it is becoming increasingly common to have to regularly forecast many millions of time series. For example, large scale businesses may be interested in forecasting sales, cost, and demand for their thousands of products across different locations, warehouses, etc. Technology companies such as Google collect many millions of daily time series such as web-click logs, web search counts, queries, revenue, number of users for different services, etc., and require fast and accurate automatic forecasts. However, the scale of these tasks have raised some computational challenges that we seek to address by proposing a new fast algorithm for model selection and time series forecasting.

When there are a large number of time series to be forecast, there are at least three possible forecasting strategies: (1) a single method may be used to provide forecasts across all time series; (2) a framework can be developed to select the most appropriate forecasting method for each series; (3) several methods can be applied for each individual series and the resulting forecasts combined. It is very unlikely that a single method will consistently outperform its competitors across all time series, so we reject strategy 1. Because our focus is on fast, scalable forecasting, we also reject the combination approach (despite it often being the most accurate of the three strategies), as the computational requirements are much greater than for strategy 2. We adopt the approach of selecting an individual forecasting method for each time series to be forecast.

However, selecting the most appropriate model for a given time series can also be problematic. Two of the most commonly used automatic algorithms are the automated Exponential Smoothing Algorithm (ETS) of Hyndman et al. (2002) and the automated ARIMA algorithm of Hyndman & Khandakar (2008). Both algorithms are implemented in the forecast package in R (Hyndman et al. 2018). In this paradigm, a class of models is selected in advance, and many models within that class are estimated for each time series. The model with the smallest AICc value is chosen and used to compute forecasts. This approach relies on the expert judgement of the forecaster in first selecting the most appropriate class of models to use, as it is not usually possible to compare AICc values *between* model classes due to differences in the way the likelihood is computed, and the way initial conditions are handled.

An alternative approach, which avoids selecting a class of models *a priori*, is to use a simple "hold-out" test set; but then there is often insufficient data to draw a reliable conclusion. To overcome this problem, time series cross-validation can be used (Hyndman & Athanasopoulos 2018); then models from many different classes may be applied, and the model with the lowest cross-validated MSE selected. However, this increases the computation involved considerably (at least to order n^2 where n is the number of series to be forecast).

Clearly, there is a need for a fast, accurate algorithm to automate forecasting model selection. We propose a general meta-learning framework using features of the time series to select the class of models, or even the specific model, to be used for forecasting. The model selection process is carried out using a classification algorithm — we use the time series features as inputs, and the best forecasting algorithm as the output. The classification algorithm can be built using a large historical collection of time series, in advance of the real forecasting exercise (so it is an "offline" procedure). Then, when we have a new time series to forecast, we can quickly compute its features, use the pre-trained classification algorithm to identify the best forecasting model, and produce the required forecasts. Thus, the "online" part of our algorithm requires only feature computation, and the application of a single forecasting model, with no need to estimate large numbers of models within a class, or to carry out a computationally-intensive cross-validation procedure.

The rest of this paper is organized as follows. We review the related work in Section 2. In Section 3 we explain the detailed components and procedures of our proposed framework for forecast model selection. In Section 4 we present the results, followed by the conclusions and future work in Section 5.

2 Literature Review

2.1 Time series features

Rather than work with the time series directly in the "instance space", we propose analysing time series via an associated "feature space". A time series feature is any measurable characteristic of a time series. For example, Figure 1 shows the instance-based representation of six time series taken from the M3 competition (Makridakis & Hibon 2000) while Figure 2 shows a feature-based representation of same time series. Here only two features are considered: the strength of seasonality and the strength of trend, calculated based on the measures introduced by Wang, Smith-Miles & Hyndman (2009). Time series in the lower left quadrant of Figure 2 are non-seasonal but trended, while there is only one series with both high trend and high seasonality. We also see how the degree of seasonality and trend varies between series. Other examples of time series features include autocorrelation, spectral entropy and measures of self-similarity and non-linearity. Fulcher & Jones (2014) introduced 9000 operations to extract features from time series.

The choice of the most appropriate set of features depends on both the nature of the time series being analysed, and the purpose of the analysis. In Section 4, we study time series that have been used in the M1 and M3 competitions (Makridakis et al. 1982**makridakis2003m3**), and we select time series features for the purpose of forecast model selection. Because the M1 and M3 competitions involved time series of different lengths, on different scales, and with different properties, we restrict our features to be ergodic, stationary and independent of scale. Because we are concerned with forecasting, we select features which have discriminatory power in selecting a good forecasting method.

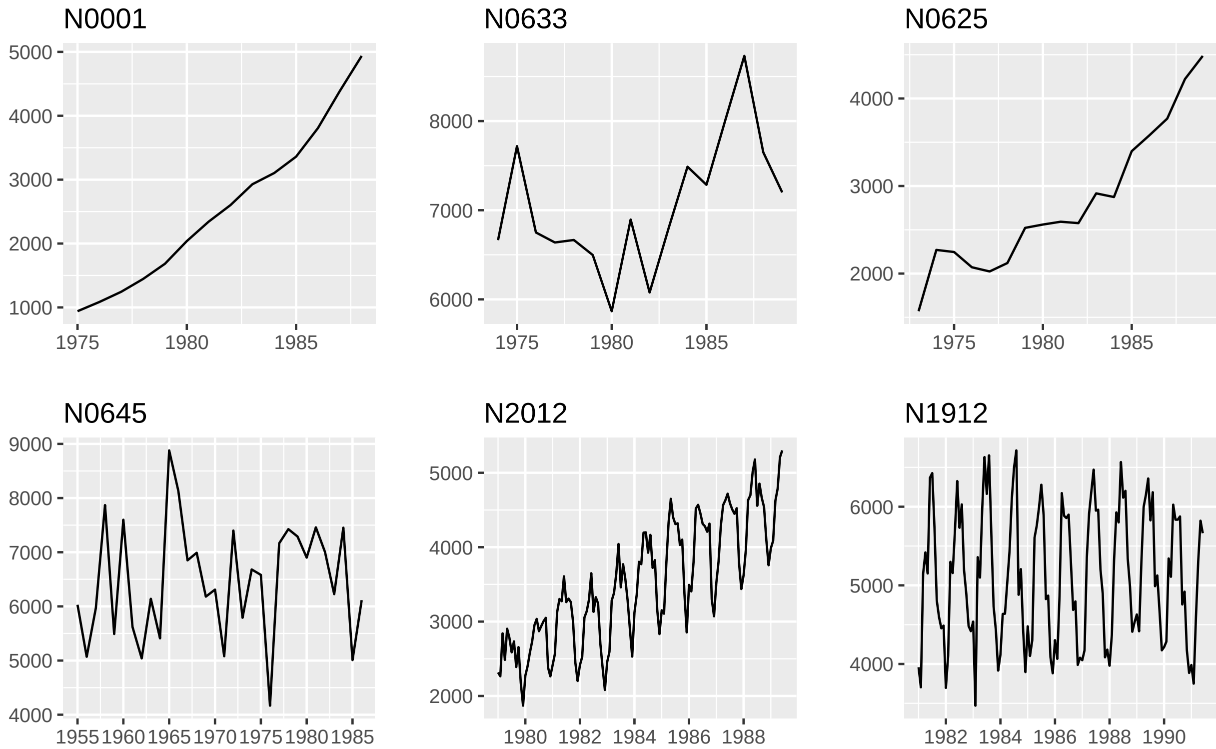

Figure 1: *Instance-based representation of time series*

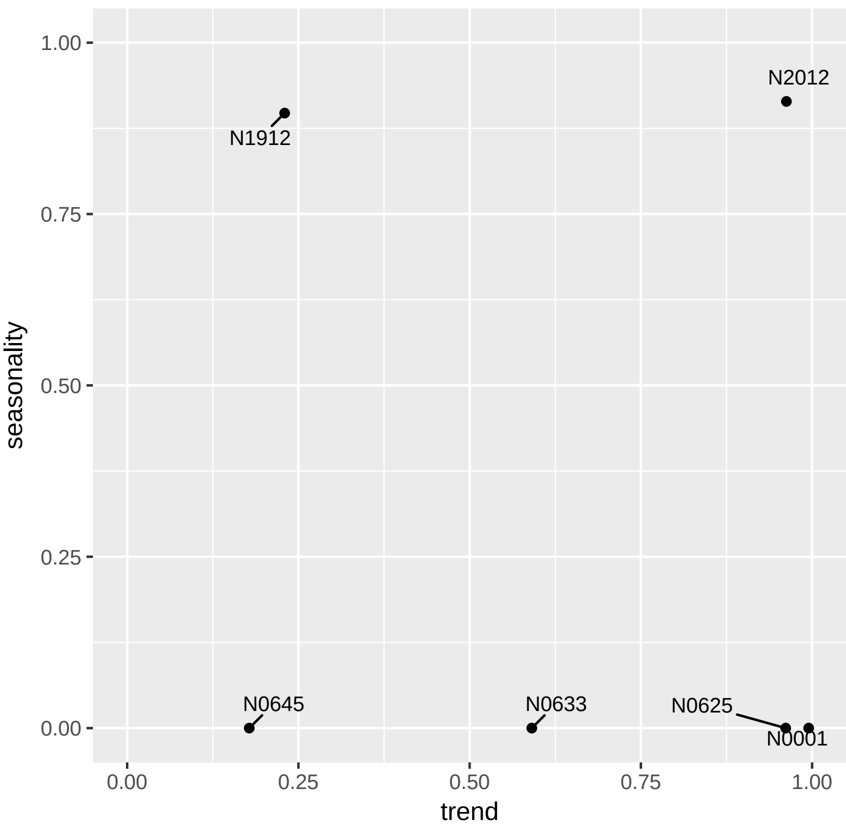

Figure 2: *Feature-based representation of time series*

2.2 What makes features useful for forecasting model identification?

Reid (1972) pointed out that the performance of forecasting methods changes according to the nature of data, and if the reasons for these variations are explored, they may be useful in selecting the most appropriate model. In response to the results of the M3 competition (Makridakis & Hibon 2000), similar ideas have been reported by others, who have argued that the characteristics of various time series may provide useful insights into which forecasting methods are most appropriate to forecast a given time series (Hyndman 2001; Lawrence 2001; Armstrong 2001).

Many time series forecasting techniques have been developed to capture specific characteristcs of time series that are common in a particular discipline. For example, GARCH models were introduced to account for time-varying volatility in financial time series, and ETS models were introduced to handle the trend and seasonal patterns which are typical in quarterly and monthly sales data. An appropriate set of features should reveal the characteristics of the time series that are useful in determining the best forecasting method.

Several researchers have introduced rules for forecasting based on features (Collopy & Armstrong 1992; Adya et al. 2001; Wang, Smith-Miles & Hyndman 2009). Kang, Hyndman & Smith-Miles (2017) applied principal component analysis to project a large collection of time series into a two dimensional feature space in order to visualize what makes a particular forecasting method perform well or not. The features they considered were spectral entropy, first-order auto-correlation coefficient, strength of trend, strength of seasonality, seasonal period and optimal Box-Cox transformation parameter. They also proposed a method for generating new time series based on specified features.

2.3 Meta-learning for algorithm selection

John Rice was an early and strong proponent of the idea of meta-learning, which he called the algorithm selection problem (ASP) (Rice 1976). The term *meta-learning* started to appear with the emergence of the machine-learning literature. Rice's framework for algorithm selection is shown in Figure 3.

There are four main components in Rice's framework. The problem space, P, represents the data sets used in the study. The feature space, F, is the range of measures that characterize the problem space P. The algorithm space, A, is a list of suitable candidate algorithms

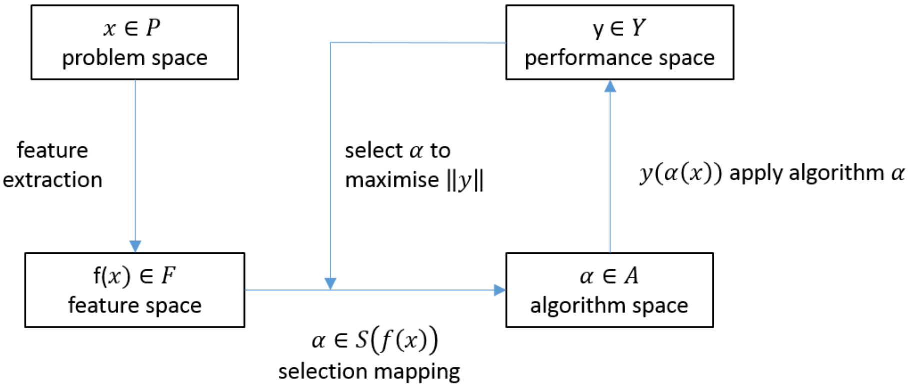

Figure 3: *Rice's framework for the Algorithm Selection Problem (reproduced from Smith-Miles, 2009)*

which we can use to find solutions to the problems in P. The performance metric, Y, is a measure of algorithm performance such as accuracy, speed, etc. Rice's formal definition of the algorithm selection problem is (Smith-Miles 2009) as follows.

Definition 2.1. For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$.

The main challenge in ASP is to identify the selection mapping S from the feature space to the algorithm space. Even though Rice's framework articulates a conceptually rich framework, it does not specify how to obtain S. This gives rise to the meta-learning approach.

The meta-learning framework consists of an offline (training) phase and an online (prediction) phase. In the offline phase, the mapping S is learned based on a collection of training examples. This is performed using a *meta-learner* which can be any supervised learning algorithm designed to estimate S. The inputs for the meta-learner are known as *meta-features*, and instances in the algorithm space are the *output labels*. The database comprising both input-features and output-labels is called *meta-data*. In the online phase of the algorithm, input-features are extracted from new data and passed into the meta-learner that was constructed in the offline phase, in order to predict the output-labels of the new data.

2.4 Forecasting model selection using meta-learning

Forecasting model selection problems can be framed according to Rice's ASP framework.

Definition 2.2. For a given time series $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ minimizes forecasting accuracy error metric $y(\alpha(x)) \in Y$ on the test set of the time series.

Existing methods differ with respect to the way they define the problem space (A), the features (F), the forecasting accuracy measure (Y) and the selection mapping (S).

Collopy & Armstrong (1992) introduced 99 rules based on 18 features of time series, in order to make forecasts for economic and demographic time series. This work was extended by Armstrong (2001) to reduce human intervention. Shah (1997) used the following features to classify time series: the number of observations, the ratio of the number of turning points to the length of the series, the ratio of number of step changes, skewness, kurtosis, the coefficient of variation, autocorrelations at lags 1–4, and partial autocorrelations at lag 2–4. Casting his work in Rice's framework, we can specify: $P = 203$ quarterly series from the M1 competition (Makridakis et al. 1982); $A = 3$ forecasting methods, namely simple exponential smoothing, Holt-Winters exponential smoothing with multiplicative seasonality, and a basic structural time series model; $Y =$ mean squared error for a hold-out sample. In Shah (1997), the mapping S is learnt using discriminant analysis.

Prudêncio & Ludermir (2004) was the first paper to use the term "meta-learning" in the context of time series model selection. They studied the applicability of meta-learning approaches for forecasting model selection based on two case studies. Again using the notation of Definition 2.2, we can describe their first case study with: A contained only two forecasting methods, simple exponential smoothing and a time-delay neural network; $Y =$ mean absolute error; F consisted of 14 features, namely length, autocorrelation coefficients, coefficient of variation, skewness, kurtosis, and a test of turning points to measure the randomness of the time series; S was learnt using the C4.5 decision tree algorithm. For their second study, the algorithm space included a random walk, Holt's linear exponential smoothing and AR models; the problem space P contained the yearly series from the M3 competition (Makridakis & Hibon 2000); F included a subset of features from the first study; and Y was a ranking based on error. Beyond the task of forecasting model selection, they used the NOEMON approach to rank the algorithms (Kalousis & Theoharis 1999).

Lemke & Gabrys (2010) studied the applicability of different meta-learning approaches for forecasting model selection. Their algorithm space A contained ARIMA models, exponential smoothing models and a neural network model. In addition to statistical measures such as the standard deviation of the detrended series, skewness, kurtosis, length, strength of trend, Durbin-Watson statistics of regression residuals, the number of turning points, step changes, a predictability measure, non-linearity, the largest Lyapunov exponent, and auto-correlation and partial-autocorrelation coefficients, he also used frequency domain based features. The feed forward neural network, decision tree and support vector machine approaches were considered to learn the mapping S.

Wang, Smith-Miles & Hyndman (2009) used a meta-learning framework to provide recommendations as to which forecast method to use to generate forecasts. In order to evaluate forecast accuracy, they introduced a new measure Y = *simple percentage better (SPB)*, which calculates the forecasting accuracy of a method against the forecasting accuracy error of random walk model. They used a feature set F comprising nine features: strength of trend, strength of seasonality, serial correlation, non linearity, skewness, kurtosis, self-similarity, chaos and periodicity. The algorithm space A included eight forecasting methods, namely, exponential smoothing, ARIMA, neural networks and random walk model; while the mapping S was learned using the C4.5 algorithm for building decision trees. In addition, they used SOM clustering on the features of the time series in order to understand the nature of time series in a two-dimensional setting.

The set of features introduced by Wang, Smith-Miles & Hyndman (2009) was later used by Widodo & Budi (2013) to develop a meta-learning framework for forecasting model selection. The authors further reduced the dimensionality of time series by performing principal component analysis on the features.

More recently, Kück, Crone & Freitag (2016) proposed a meta-learning framework based on neural networks for forecasting model selection. Here, $P = 78$ time series from the NN3 competition were used to build the meta-learner. They introduced a new set of features based on forecasting errors. The average symmetric mean absolute percentage error was used to identify the best forecasting method for each series. They classify their forecasting models in the algorithm space A, comprising single, seasonal, seasonal-trend and trend exponential smoothing. The mapping S was learned using a feed-forward neural network. Further, they evaluated the performance of different sets of features for forecasting model selection.

3 Methodology

Our proposed framework is presented in Figure 4. The offline and online parts of the framework are shown in blue and red respectively. A classification algorithm (the meta-learner) is trained during the offline phase and then is used to select appropriate forecasting models for new series in the online phase.

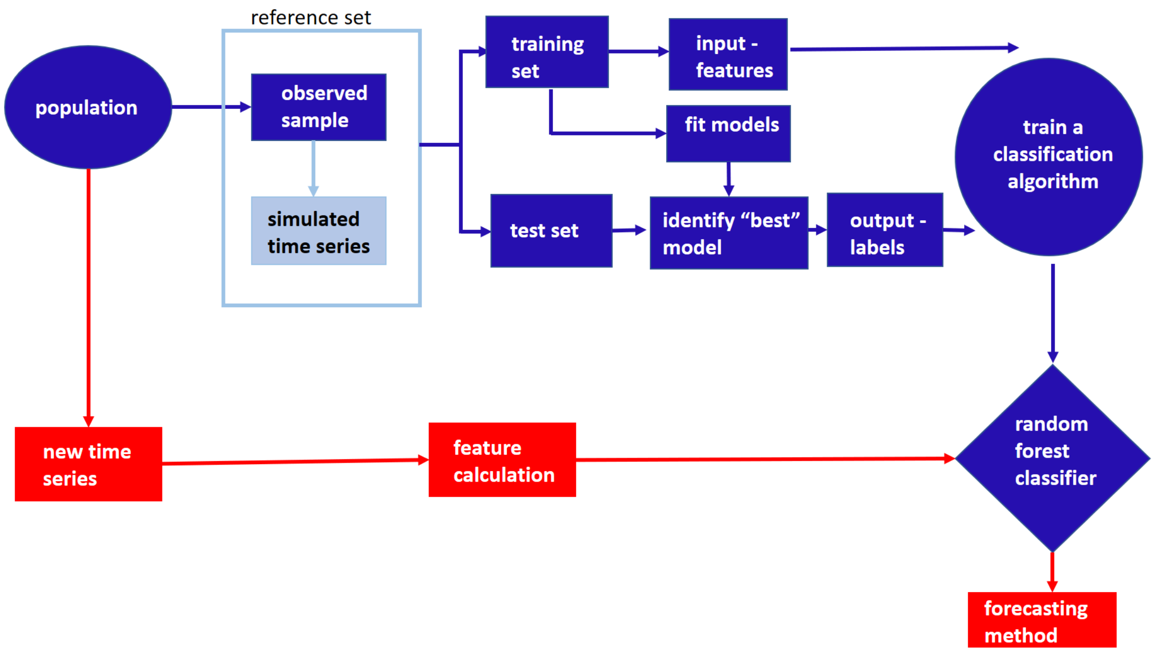

Figure 4: *Proposed framework (blue: offline phase, red: online phase)*

In order to train our classification algorithm, we need a large collection of time series which are similar to those that we will be forecasting. We assume that we have an essentially infinite population of time series, and we take a sample of them in order to train the classification algorithm. The new time series we wish to forecast can be thought of as additional draws from the same population. Hence, any conclusions made from the classification framework refer only to the population from which the sample has been selected. We may call this the "target population" of time series. It is important to have a well-defined target population to avoid misapplying the classification rules. We denote the collection of time series used for training the classifier as the "reference set". We split each time series within the reference set into a training period and a test period. From each training period we compute a range of time series features, and fit a selection of potential models. The calculated features form the input vector to the classification algorithm. Using the fitted models, we generate forecasts and identify the "best" model for each time series based on a forecast error measure (e.g., MASE) calculated over the test period. The models deemed “best” form the output labels for the classification algorithm. The pseudo code for this algorithm is given in Algorithm 1. In the following sections, we briefly discuss aspects of the offline part of the algorithm.

Algorithm 1 Identification of the "best" forecast method for a new time series.

Offline phase

Given:

$O = \{t_1, t_2, \ldots, t_n\}$: the collection of n observed time series.
L : the set of class labels (eg: ARIMA, ETS, SNAIVE).
F : the set of functions to calculate time series features.
$nsim$: number of series to be simulated.
B : number of trees in the random forest.
$mtry$: number of features to be selected at each node.

Output:

a random forest classifier

Prepare the reference set

For $i = 1$ to n:

1: Fit ARIMA and ETS models to t_i.
2: Simulate $nsim$ time series from each model in step 2.
3: The time series in O and simulated time series in step 3 form the reference set $R = \{t_1, t_2, \ldots, t_n, t_{n+1}, \ldots, t_N\}$ where $N = n + nsim$.

Prepare the meta-data

For $j = 1$ to N:

4: Split t_j into a training period and test period.
5: Calculate features F based on the training period.
6: Fit L models to the training period.
7: Calculate forecasts for the test period from each model.
8: Calculate forecast error measure over the test period for all models in L.
9: Select the model with the minimum forecast error.
10: Meta-data: input features (step 7), output labels (step 11).

Train a random forest classifier

11: Train a random forest classifier based on the meta-data.
12: Random forest: the ensemble of trees $\{T_b\}_1^B$.

Online phase

Given:

the random forest classifier from step 14 .

Output:

class labels for newly arrived time series t_{new}.

13: For t_{new} calculate features F.
14: Let $\hat{C}_b(t_{new})$ be the class prediction of the b^{th} random forest tree. Then class label for t_{new} is $\hat{C_{rf}}(t_{new}) = majorityvote\hat{C}_b(t_{new})$.

3.1 Augmenting the observed sample with simulated series

In practice, we may wish to augment our set of training time series by simulating new time series that are similar to those from the population. This process may be useful when our observed sample of time series is too small to build a reliable classifier. Alternatively, we may wish to add more of some types of time series to the reference set in order to get a more balanced sample for the classification. In order to produce simulated series that are similar to our population, we use several standard automatic forecasting algorithms such as ETS or automated ARIMA models, and then simulate multiple time series from the selected model within each model class. Assuming the models produce data that are similar to the observed time series, this ensures that the simulated series are similar to those in the population. Note that this is done in the offline phase of the algorithm, so the computational time in producing these simulated series is of no real consequence.

3.2 Input: features

Our proposed algorithm requires features that enable identification of a suitable forecasting model for a given time series. Therefore, the features used should capture the dynamic structure of the time series, such as autocorrelation, trend, seasonality, nonlinearity, heterogeneity, and so on.

The purpose of this feature-based framework is to reduce the time required for model selection. Therefore, time needed to calculate the input features should be significantly less than the time required to estimate the parameters of all candidate models in a model selection procedure. Furthermore, interpretability, robustness to outliers, scale and length independence need to be considered when selecting features for this classification problem. A comprehensive description of the features used in the experiment is specified in Table 1.

3.3 Output: labels

The task of our classification framework is to identify the best forecasting method for a given time series. We define the best forecast method as the model with the lowest accuracy measure (e.g., MAPE or MASE) in the test period.

It is not possible to train all possible classes of time series models, but at least we should consider enough possibilities so that the algorithm can be used for model selection with high confidence. The models to be considered will depend on the specific population of time series models we need to forecast. For example, if we have only non-seasonal time series, and no chaotic features, we may wish to restrict our models to random walks, white noise, ARIMA processes and ETS processes. Even in this scenario, the number of possible models can be quite large. In order to identify the best forecasting method for each series, all the methods considered are run on all time series in the reference set, and forecasts are generated from each of them. Model estimation is done on the training period of each series and forecasts are compared with the values in the test period.This step is computationally intensive and time-consuming, as all methods have to be tried on each series in the reference set. But since this is done only in the offline phase, the time involved and the computational cost associated with this task is not a problem.

3.4 Random forest algorithm

A random forest (Breiman 2001) is an ensemble learning method that combines a large number of decision trees using a two-step randomization process. Let $Z = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$ be the reference set, where the input $\boldsymbol{x}_i$ is an m-vector of features, the output, y_i, corresponds to the class label of the ith observation, m is the total number of features, and N is the number of training examples in the reference set. Each tree in the forest is grown based on a bootstrap sample of size N from the reference set. At each node of the tree, randomly select $f < m$ features from the full set of features. The best split is selected among those f features. The split which results in the most homogeneous subnodes is considered the best split. Various measures have been introduced to evaluate the homogeneity of subnodes, such as classification error rate, the Gini index and cross entropy (Friedman, Hastie & Tibshirani 2009). In this study, we use the Gini index to evaluate the homogeneity of a particular split. The trees are grown to the largest extent possible without pruning. To determine the class label for a new instance, features are calculated and passed down the trees. Then each tree gives a prediction and the majority vote over all individual trees leads to the final decision. In this work, we have used the randomForest package (Liaw & Wiener 2002) in R (R Core Team 2017) which implements the Fortran code for Random Forest classification by Breiman & Cutler (2004).

4 Application to M competition Data

To test how well our proposed framework can identify suitable forecasting models, we use the time series of the M1 competition (Makridakis et al. 1982) and M3 competition

(Makridakis & Hibon 2000). The R package Mcomp (Hyndman 2013) provides the data from both competitions. The proposed algorithm is applied to yearly, quarterly and monthly series separately. We ran two experiments for each case. In the first experiment, we treat the time series of the M1 competition as the *observed sample* and the time series of the M3 competition as the collection of *new time series*. We reverse the sets in the second experiment, using the M3 data as the *observed sample* and the M1 data as the *new time series*. This allow us to compare our results with published forecast accuracy results for each competition. In both experiments, we fit ARIMA and ETS models to the full length of each series in the corresponding *observed samples* using the `auto.arima` and `ets` functions in the forecast package (Hyndman & Khandakar 2008). From each model fitted to annual and quarterly data, we simulate a further 1000 series. For monthly time series, we simulate a further 100 series (to speed up the offline calculation process). The lengths of the simulated time series are set to be equal to the lengths of the series on which they were based.

As shown in Figure 4, the task of constructing the meta-database contains two main components: (i) identification of an *output-label* and (ii) computation of features.

The output-labels we consider in this experiment are:

(a) White noise process (WN);
(b) AR/ MA/ ARMA;
(c) ARIMA;
(d) Random walk with drift (RWD);
(e) Random walk (RW);
(f) Theta;
(g) STL-AR;
(h) Exponential Smoothing Model (ETS) without trend and seasonal components;
(i) ETS with trend component and without seasonal component;
(j) ETS with damped trend component and without seasonal component;

Most of these are self-explanatory labels based on models implemented in the forecast package (Hyndman & Khandakar 2008). STL-AR refers to a model based on an STL decomposition method applied to the time series, then an AR model is used to forecast the seasonally adjusted data, while the seasonal naive method is used to forecast the seasonal component.

In addition to the above ten output labels, we further include the following five class labels for seasonal data:

(k) ETS with trend and seasonal components

(l) ETS with damped trend and seasonal components

(m) ETS with seasonal components and without trend component

(n) SARIMA

(o) Seasonal naive method.

The models are estimated using the training period for each series, and forecasts are produced for the whole of the test periods. We further use the `auto.arima` and `ets` functions in the forecast package to identify suitable AR/MA/ARMA, ARIMA, SARIMA and ETS models. The model corresponding to the smallest MASE (Hyndman & Koehler 2006) for the test period is selected as the *output-label*.

4.1 Feature computation process

We use a set of 25 features for yearly data and a set of 30 features for seasonal data. Some of these are taken from previous studies (Wang, Smith-Miles & Hyndman 2009; Hyndman, Wang & Laptev 2015; Kang, Hyndman & Smith-Miles 2017), and we have added some new features that we believe provide some useful information. Each feature can be computed rapidly, thus making the online phase of our algorithm very fast. The features are summarized in Table 1, and fully described in the Appendix.

The correlation matrices for all features are presented in Figure 5. The variability in the correlations reflects the diversity of the selected features. In other words, the features we used were able to capture different characteristics of the time series. Further, the structure of the correlation matrices are similar to each other, showing the different collections have similar feature spaces.

This is not reproducible. Replace Figure 5 with R code to generate the required figure.

4.2 Model calibration

The random forest algorithm is highly sensitive to class imbalance (Breiman 2001), and our reference set is unbalanced: some classes contains significantly more cases than other classes. The degree of class imbalance is reduced to some extent by augmenting the observed sample with simulated time series. We consider three approaches to address the

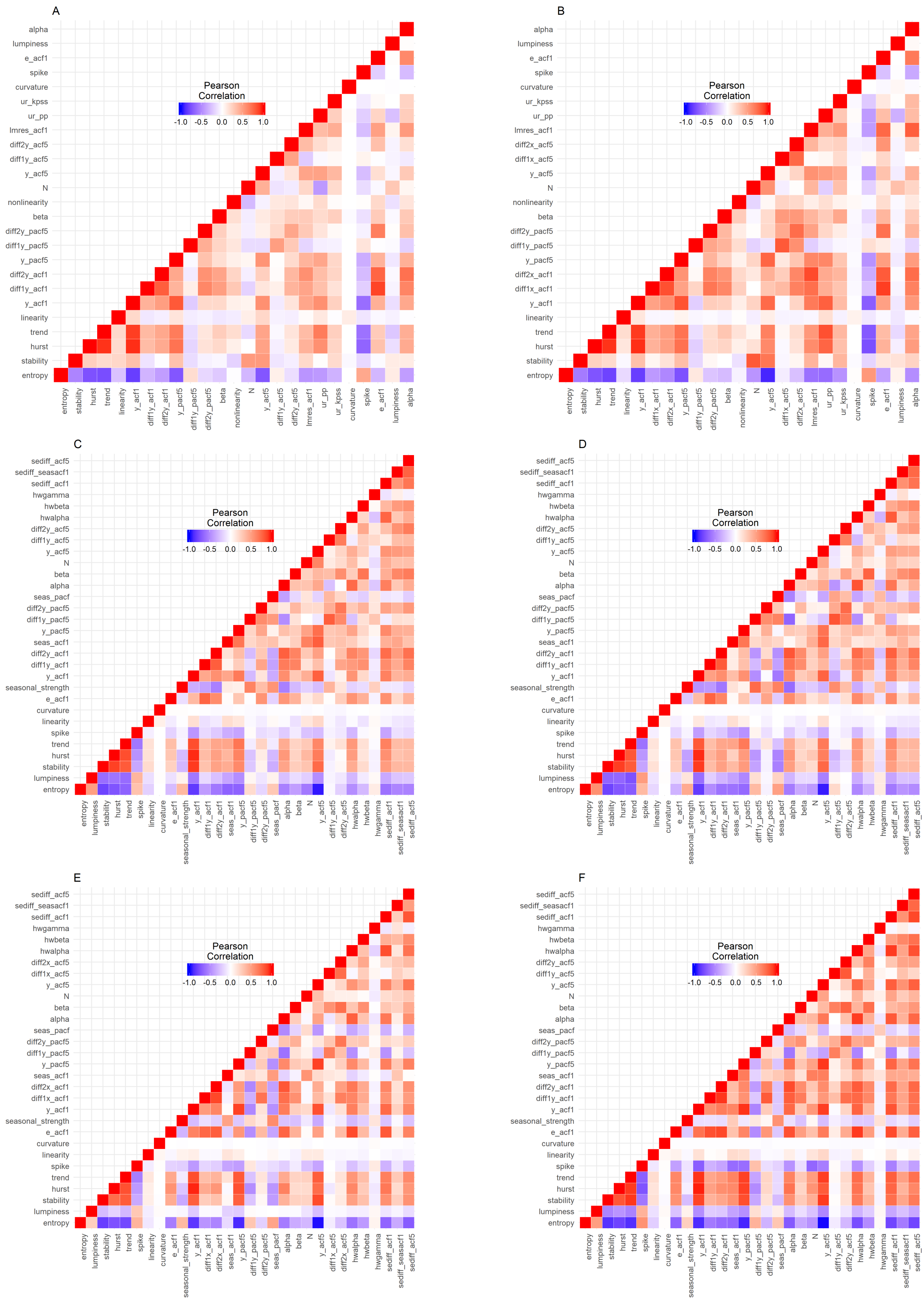

Figure 5: *Correlation matrix plots. A: Experiment 1 yearly, B: Experiment 2 yearly, C: Experiment 1 quarterly, D: Experiment 2 quarterly, E: Experiment 1 monthly, F: Experiment 2 monthly.*

Table 1: *Features used for selecting a forecasting model.*

	Feature	Description	Non-seasonal	Seasonal
1	N	length of time series	✓	✓
2	trend	strength of trend	✓	✓
3	seasonal	strength of seasonality	-	✓
4	linearity	linearity	✓	✓
5	curvature	curvature	✓	✓
6	spikiness	spikiness	✓	✓
7	e_acf1	first ACF value of remainder series	✓	✓
8	stability	stability	✓	✓
9	lumpiness	lumpiness	✓	✓
10	entropy	spectral entropy	✓	✓
11	hurst	Hurst exponent	✓	✓
12	nonlinearity	nonlinearity	✓	✓
13	alpha	ETS(A,A,N) $\hat{\alpha}$	✓	✓
14	beta	ETS(A,A,N) $\hat{\beta}$	✓	✓
15	hwalpha	ETS(A,A,A) $\hat{\alpha}$	-	✓
16	hwbeta	ETS(A,A,A) $\hat{\beta}$	-	✓
17	hwgamma	ETS(A,A,A) $\hat{\gamma}$	-	✓
18	ur_pp	test statistic based on Phillips-Perron test	✓	-
19	ur_kpss	test statistic based on KPSS test	✓	-
20	x_acf1	first ACF value of the original series	✓	✓
21	diff1x_acf1	first ACF value of the differenced series	✓	✓
22	diff2x_acf1	first ACF value of the twice-differenced series	✓	✓
23	x_acf5	sum of squares of first 5 ACF values of original series	✓	✓
24	diff1x_acf5	sum of squares of first 5 ACF values of differenced series	✓	✓
25	diff2x_acf5	sum of squares of first 5 ACF values of twice-differenced series	✓	✓
26	seas_acf1	autocorrelation coefficient at first lag	-	✓
27	sediff_acf1	first ACF value of seasonally-differenced series	-	✓
28	sediff_seacf1	ACF value at the first seasonal lag of seasonally-differenced series	-	✓
29	sediff_acf5	sum of squares of first 5 autocorrelation coefficients of seasonally-differenced series	-	✓
30	lmres_acf1	first ACF value of residual series of linear trend model	✓	-
31	x_pacf5	sum of squares of first 5 PACF values of original series	✓	✓
32	diff1x_pacf5	sum of squares of first 5 PACF values of differenced series	✓	✓
33	diff2x_pacf5	sum of squares of first 5 PACF values of twice-differenced series	✓	✓

class imbalance in the data: 1) we incorporate class priors into the random forest classifier; 2) we use the Balanced Random Forest (BRF) algorithm introduced by Chen, Liaw & Breiman (2004); and 3) we re-balance the reference set with down-sampling. In down-sampling, the size of the reference set is reduced by down-sampling the larger classes so that they match the smallest class in size; this potentially discards some useful information. We compare the results of these three approaches to the classifier built on unbalanced data. The RF algorithms are implemented by the randomForest R package (Liaw & Wiener 2002). The class priors are introduced through the option `classwt`. We use the reciprocal of class size as the class priors. The number of trees `ntree` is set to 1000, and the number of randomly selected features `mtry` is set to be one third of the total number of features available.

Our aim is different from most classification problems in that we are not interested in accurately predicting the class, but in finding a good forecasting model. It is possible that two models produce almost equally good forecasts, and then it does not matter whether the classifier picks one or the other. Consequently, we do not study the classification accuracy of our random forests, but only report the forecast accuracy obtained with using them.

4.3 Summary of the main results

We build separate random forest classifiers for yearly data, quarterly data and monthly data. For the second experiment we take a subset of the simulated time series when training the RF-unbalanced and RF-class priors, to reduce the size of the reference set. The subsets are selected randomly according to the proportions of output-labels in the observed samples. This ensures that our reference set shares similar characteristics to the observed sample.

A principal component analysis is used to visualize the feature-spaces of the different time series collections. We compute the principal components projection using the features in the observed sample, and then project the simulated time series and the new time series using the same projection. The results are shown in Figures 6–8, for yearly, quarterly and monthly data respectively, where the first three principal components are plotted against each other. The points on each graph represent individual time series.

The accuracy of our method is compared against the following commonly-used forecasting methods:

1. automated ARIMA algorithm of Hyndman & Khandakar (2008);
2. automated ETS algorithm of Hyndman & Khandakar (2008);
3. Random walk with drift (RWD);
4. Random walk model (RW);
5. White noise process (WN);
6. Theta method;
7. STL-AR method;
8. seasonal naive (for seasonal data).

The automated ARIMA and ETS algorithms are implemented using the `auto.arima` and `ets` functions available in the forecast package in R (Hyndman & Khandakar 2008). Each method is implemented on the training period and forecasts are computed up to the full length of the test period. Then we compute the MASE for each forecast horizon, by averaging the

MASE across all series in the collection of new time series. To assist in the evaluation of the proposed framework, for each forecast horizon we rank our method compared to the other methods listed above, and an average ranking over all forecast horizons is computed. The results are given in Tables 2–7. The MASE value corresponding to the best performing method in each category is highlighted in bold.

Yearly data

For the yearly data, the first 3 principal components explain 62.5% of the total variance of the features for the M1 competition data, and 62.2% for the M3 competition data.

As seen in Figure 6, simulated time series are able to fill the gaps between the points in the observed sample. By augmenting the reference set with simulated time series, we were able to increase the diversity and evenness of the feature space of the observed time series. Further, in both experiments, all the *observed time series* fall within the space of all simulated data. This guarantees that we have not reduced the feature diversity from the observed sample. The remaining plots in Figures 7–8 can be interpreted similarly.

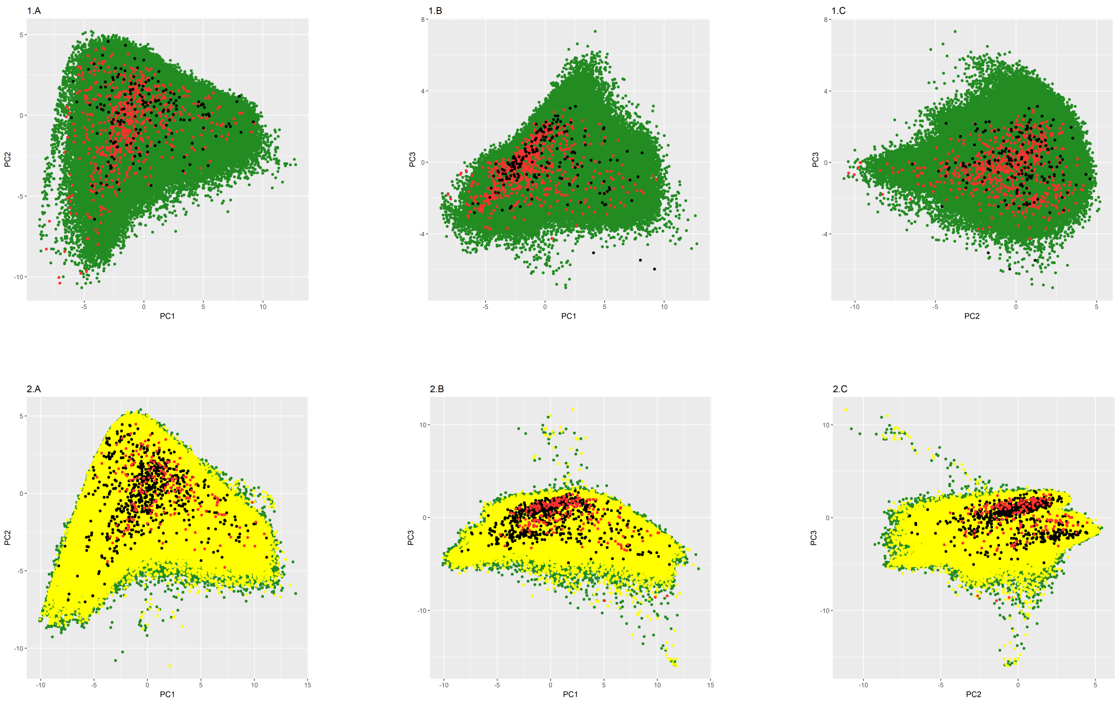

Figure 6: *Distribution of yearly time series in the PCA space. Results of experiment 1 (observed sample M1, new time series M3) are shown in panels 1A– 1C, and results of experiment 2 (observed sample M3, new time series M1) are shown in panels 2A- 2C. On each graph, green indicates simulated time series, yellow shows a subset of simulated time series, black indicates observed time series, while orange denotes new time series.*

Tables 2–3 compare the performance of our proposed framework to the benchmark methods. For each method, we calculate out-of-sample MASE over the forecast horizons, and average over all time series. For yearly series from the M3 competition, our meta-learning method beats all methods other than the random walk with drift model. For the M1 competition data, our meta-learning method using RF-unbalanced and RF-class priors consistently forecast more accurately than all benchmark methods.

Table 2: *Experiment 1 (observed sample M1): Forecast accuracy measures for 645 M3 yearly series.*

	Average of forecasting horizons						
	1	1–2	1–3	1–4	1–5	1–6	Average Rank
RF-unbalanced	1.06	1.42	1.83	2.20	2.54	2.85	3.50
RF-class priors	**1.03**	1.37	1.78	2.14	2.47	2.77	1.83
auto.arima	1.11	1.48	1.90	2.28	2.63	2.96	6.83
ets	1.09	1.44	1.84	2.20	2.54	2.86	4.17
WN	6.54	6.91	7.22	7.48	7.76	8.07	9.00
RW	1.24	1.68	2.11	2.48	2.83	3.17	8.00
RWD	**1.03**	**1.36**	**1.74**	**2.05**	**2.35**	**2.63**	1.17
STL-AR	1.09	1.47	1.89	2.27	2.62	2.95	5.50
Theta	1.12	1.47	1.86	2.18	2.48	2.77	4.17

Table 3: *Experiment 2 (observed sample M3): Forecast accuracy measures for 181 M1 yearly series.*

	Average of forecasting horizons						
	1	1–2	1–3	1–4	1–5	1–6	Average Rank
RF-unbalanced	**0.97**	**1.39**	1.93	2.42	2.90	3.37	1.67
RF-class priors	1.02	1.40	**1.92**	**2.40**	**2.87**	**3.33**	1.33
auto.arima	1.06	1.47	2.01	2.51	3.00	3.47	3.50
ets	1.12	1.59	2.17	2.72	3.26	3.77	6.00
WN	6.38	7.08	7.92	8.59	9.28	10.01	9.00
RW	1.35	2.00	2.80	3.50	4.19	4.89	8.00
RWD	1.03	1.44	2.00	2.51	3.01	3.49	3.33
STL-AR	1.10	1.51	2.07	2.55	3.04	3.52	5.00
Theta	1.15	1.70	2.38	3.00	3.59	4.19	7.00

Quarterly data

For the quarterly data, the first 3 principal components explain 62.40% of the total variance of the features for the M1 competition data, and 64.75% for the M3 competition data.

Tables 4–5 summarize the results for quarterly data. Our method using RF-class priors outperforms all the benchmark methods.

Table 4: *Experiment 1 (observed sample M1): Forecast accuracy measures for 756 M3 quarterly series.*

	Average of forecasting horizons								
	1	1–2	1–3	1–4	1–5	1–6	1–7	1–8	Average rank
RF-unbalanced	0.58	**0.65**	**0.73**	**0.81**	**0.88**	**0.96**	**1.04**	1.12	1.25
RF-class priors	0.58	0.66	0.74	0.81	0.89	0.97	1.05	1.13	2.38
auto.arima	0.58	0.66	0.75	0.85	0.93	1.01	1.10	1.19	4.25
ets	**0.56**	**0.65**	**0.73**	0.82	0.91	0.99	1.08	1.17	2.75
WN	3.25	3.35	3.46	3.59	3.63	3.70	3.78	3.87	10.00
RW	1.14	1.12	1.17	1.16	1.25	1.32	1.41	1.46	7.38
RWD	1.20	1.18	1.23	1.17	1.29	1.36	1.44	1.47	8.38
STL-AR	0.70	0.90	1.08	1.27	1.44	1.60	1.75	1.91	7.88
Theta	0.62	0.68	0.76	0.83	0.90	0.97	1.04	**1.11**	3.25
Snaive	1.11	1.10	1.08	1.09	1.21	1.30	1.36	1.43	6.25

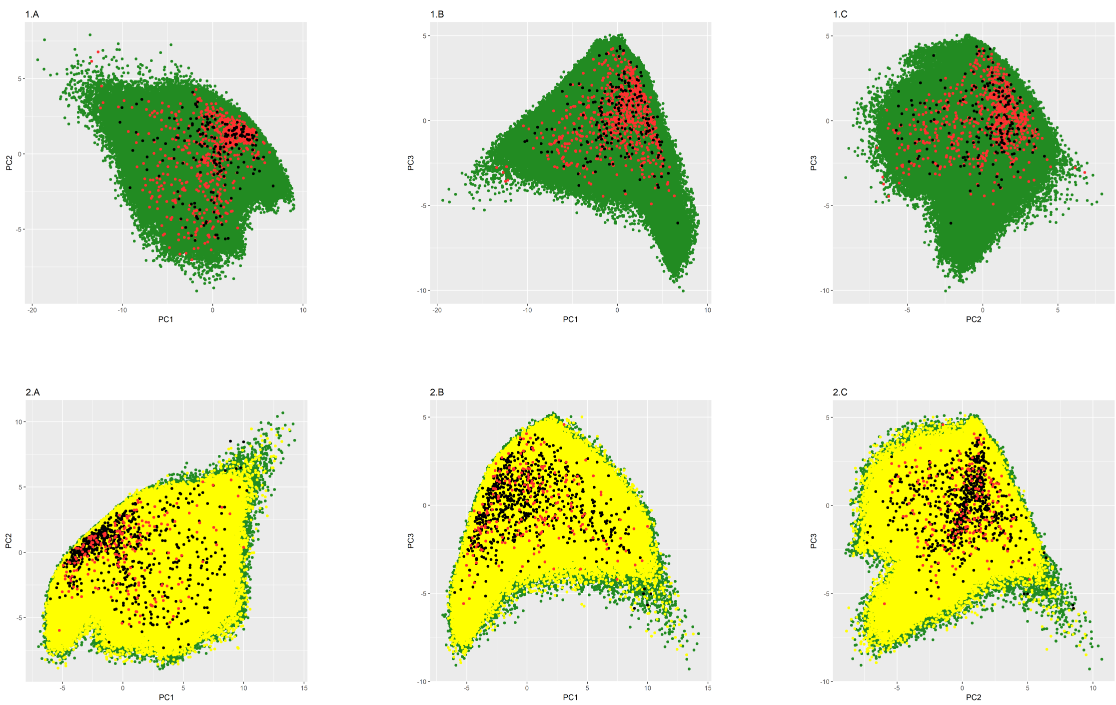

Figure 7: *Distribution of quarterly time series in the PCA space. Results of experiment 1 (observed sample M1, new time series M3) are shown in panels 1A–1C, and results of experiment 2 (observed sample M3, new time series M1) are shown in panels 2A–2C. On each graph, green indicates simulated time series, yellow shows a subset of simulated time series, black indicates observed time series, while orange denotes new time series.*

Table 5: *Experiment 2 (observed sample M3): Forecast accuracy measures for 203 M1 quarterly series.*

	Average of forecasting horizons								
	1	1-2	1–3	1–4	1–5	1–6	1–7	1–8	Average rank
RF-unbalanced	**0.77**	**0.85**	**0.95**	**1.08**	**1.22**	**1.36**	**1.48**	**1.59**	1.00
RF-class priors	0.79	0.88	0.99	1.12	1.28	1.41	1.53	1.65	2.25
auto.arima	0.85	0.94	1.05	1.19	1.37	1.53	1.67	1.80	5.00
ets	0.78	0.89	0.98	1.11	1.28	1.42	1.54	1.66	2.50
WN	3.97	4.14	4.16	4.27	4.35	4.45	4.52	4.64	10.00
RW	0.97	1.10	1.25	1.35	1.52	1.67	1.83	1.95	7.13
RWD	0.95	1.04	1.19	1.26	1.42	1.56	1.71	1.81	6.00
STL-AR	0.96	1.20	1.41	1.63	1.85	2.05	2.23	2.43	8.50
Theta	0.79	0.90	1.00	1.13	1.29	1.42	1.55	1.67	3.75
Snaive	1.52	1.53	1.53	1.56	1.74	1.86	1.98	2.08	8.38

Monthly data

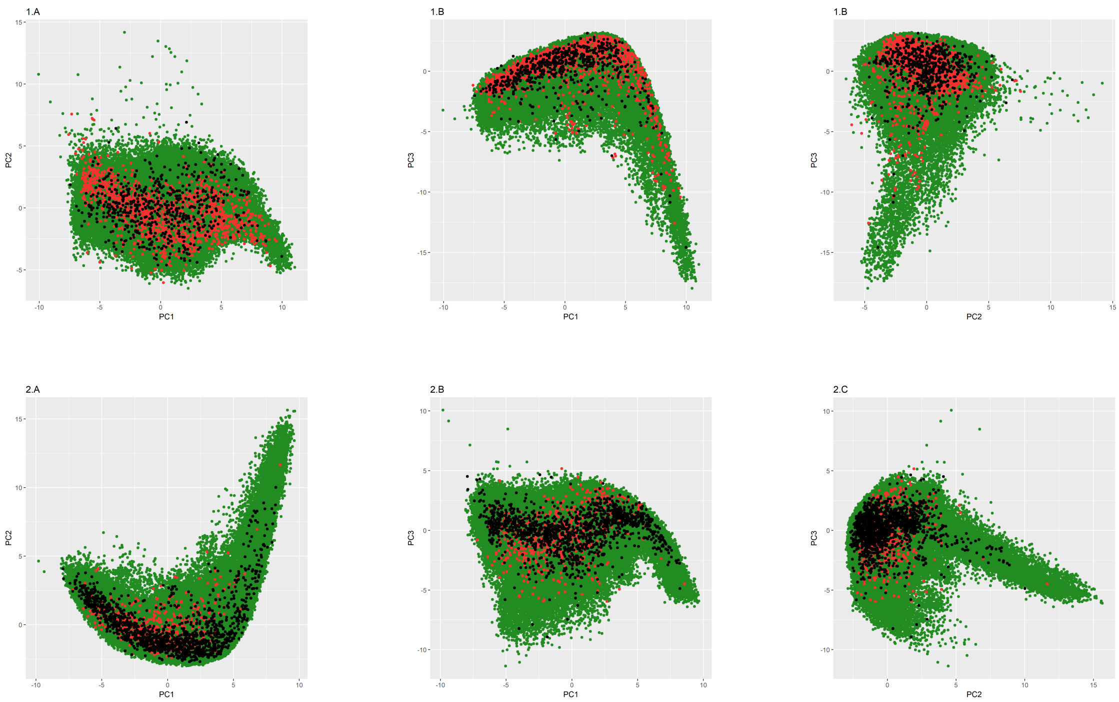

Figure 8: *Distribution of monthly time series in the PCA space; results of experiment 1 (observed sample-M1, new time series - M3) are shown in panels 1.A- 1.C and results of experiment 2 (observed sample-M3, new time series - M1) are shown in panels 2.A- 2.C, on each graph colour scheme is green-simulated time serie, black-observed time series, orange-new time series*

For the monthly data, the first 3 principal components explain 78.07% of the total variance of the features for the M1 competition data, and 65.97% of the variance for the M3 competition data.

Table 6: *Experiment 1 (observed sample M1): Forecast accuracy measures for 1428 M3 monthly series.*

	Average of forecasting horizons						
	1–4	1–6	1–8	1–10	1–12	1–18	Average rank
RF-unbalanced	0.66	0.69	0.72	0.75	0.75	0.78	5.17
RF-class priors	0.65	0.68	0.71	0.74	**0.74**	**0.77**	4.00
auto.arima	0.61	0.65	0.69	0.72	0.75	0.88	2.67
ets	**0.59**	**0.64**	**0.68**	**0.72**	**0.74**	0.86	1.67
WN	2.06	2.08	2.10	2.13	2.15	2.27	12.00
RW	0.91	0.97	1.01	1.04	1.04	1.17	10.33
RWD	0.90	0.96	1.00	1.03	1.02	1.14	9.17
STL-AR	0.73	0.81	0.90	0.98	1.04	1.27	8.83
Theta	0.63	0.67	0.72	0.75	0.77	0.89	5.67
Snaive	0.95	0.97	0.97	0.98	0.98	1.14	9.00

Table 7: *Experiment 2 (observed sample M3): Forecast accuracy measures for 617 M1 monthly series.*

	Average of forecasting horizons						
	1–4	1–6	1–8	1–10	1–12	1–18	Average rank
RF-unbalanced	0.72	0.78	0.83	0.88	**0.89**	**0.97**	2.50
RF-class priors	0.71	0.78	0.83	0.89	0.91	0.99	2.83
auto.arima	0.73	0.81	0.87	0.94	0.99	1.16	6.83
ets	0.68	0.76	0.82	0.88	0.93	1.07	2.50
WN	2.06	2.09	2.12	2.14	2.18	2.28	12.00
RW	1.18	1.24	1.31	1.34	1.33	1.47	10.00
RWD	1.19	1.27	1.37	1.40	1.39	1.55	11.00
STL-AR	0.79	0.91	0.99	1.09	1.17	1.39	8.33
Theta	**0.68**	**0.75**	**0.81**	**0.87**	0.91	1.04	1.67
Snaive	1.09	1.11	1.11	1.13	1.14	1.31	8.67

Tables 6–7 show that our meta-learning method with RF-unbalanced and RF-class priors provides more accurate forecasts than the benchmark methods for the longest forecast horizons (1–18). However, over shorter horizons, ETS does slighly better for the M3 data, and Theta does slightly better for the M1 data.

None of the Figures and Tables in this section are reproducible. Replace them with R code to generate the required figures and tables.

5 Discussion and Conclusions

In this paper we have proposed a novel framework for forecasting model selection using meta-learning based on time series features. Our algorithm uses the knowledge of the past performance of different forecasting methods on a collection of time series in order to identify the best forecasting method for a new series. We have shown that the method almost always performs better than common benchmark methods, and better than the best-performing methods from the M3 competition. Although we have illustrated the method using the M1 and M3 competition data, the framework is general and can be applied to any large collection of time series.

A major advantage of our method is that it is not necessary to estimate several different models for the data and undertake an empirical evaluation of their forecast accuracy on a given time series. Instead, we simply compute a small set of features that can be used to identify the best forecasting method.

It would be possible to obtain more accurate forecasts than those we report by combining the forecasts of many methods. However, this would not satisfy our aim of rapid automatic forecasting, because of the additional optimization that would be required to fit each model.

In addition to our new model selection framework, we have also introduced a simple set of time series features that are useful in identifying the "best" forecast method of a given time series, and can be computed rapidly. We will leave to a later study an analysis of which of these features are the most useful, and how our feature set could reduced further without loss of forecast accuracy.

We have not compared the computation time between our method and the benchmark methods. However, for real-time forecasting, our framework involves only the calculation of features, the selection of a forecast method based on the random forest classifier, and the calculation of the forecasts from the chosen model. None of these steps involve substantial computation and can be easily parallelised when forecasting for a large number of new time series. For future work, we will explore the use of other classification algorithms, and test our approach on several other large collections of time series.

Appendix

Length of time series

The length of time series is the number of observations that constitute it. The appropriate forecasting methods depend largely on how many observations are available. For example, shorter series tend to provide better forecasts with more simple models such as random walk, naive method. On the other hand, for long time series (say up to 200), models with time-varying parameters gives best forecast as it helps to capture the inner structural changes of the model. In this experiment we do not consider the models with time-varying parameter to our algorithm space as we do not have such long time series. However, we include this as a feature as the length of the series vary relatively large.

STL-decomposition based features: strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series

The features strength of trend, strength of seasonality, linearity, curvature, spikiness and first autocorrelation coefficient of the remainder series are calculated based on the STL-decomposition of the time series. In the following description, our notations are as follows: We represent a time series Y of length N as $y_1, y_2, \ldots, y_N$. First, the Box-Cox transformation is applied to the time series. The reasons for applying Box-Cox transformation: i) to stabilize the variance, ii) to make the seasonal effect additive, and iii) to make the data normally distributed. The transformed series is denoted by Y_t^*. The basic decomposition structure of the time series is denoted by: $Y_t^* = T_t + S_t + E_t$, where T_t denotes the trend in time series, S_t denotes the seasonal component, while E_t is the remainder component (Cleveland, Cleveland & Terpenning 1990). Further, the detrended series X_t is $X_t = Y_t^* - T_t$, the deseasonalized series is to be define as $Z_t = Y_t^* - S_t$, and the remainder series, R_t, is defined as $R_t = Y_t^* - T_t - S_t$.

Strength of trend

The long-term increase or decrease in time series data is called the trend (Hyndman & Athanasopoulos 2018). The strength of trend is measured by comparing the variance of de-trended series and the original series as follow (Wang, Smith-Miles & Hyndman 2009):

$$Trend = 1 - \frac{var(R_t)}{var(Z_t)}.$$

The values of this feature range between 0 and 1.

Strength of seasonality

The seasonality pattern occurs when a time series shows a pattern of repetitive behaviour over a year within a fixed period. The strength of seasonality is computed as follows(Wang, Smith-Miles & Hyndman 2009):

$$Seasonality = 1 - \frac{var(R_t)}{var(X_t)}.$$

The values of this feature range between 0 and 1.

Linearity and Curvature

The features linearity and the curvature are computed based on the coefficients of a quadratic regression of the form

$$T_t = \beta_0 + \beta_1 time_t + \beta_2 time_t^2 + \epsilon_t$$

where, $time = 1, 2, \ldots, N$. The estimated value of β_1 is used as a measure of linearity while the estimated value of β_2 is considered as a measure of curvature. The features have been used by Hyndman, Wang & Laptev (2015).

Spikiness

The feature spikiness occurs when the time series is affected by sudden drops or rise. Hyndman, Wang & Laptev (2015) introduced an index to measure spikiness as follow:

$$spikiness = var\left(\frac{var(R_t) \times N - 1 - d}{N - 2}\right);$$

where $d = (R_t - mean(R_t))^2$. Note that R_t is the remainder component calculated based on STL-decomposition.

First autocorrelation coefficient of the remainder series

We compute the first autocorrelation coefficient of the remainder series. The first autocorrelation coefficient calculated based on the remainder series does not influence by seasonality and trend present in the series.

Stability and lumpiness

A time series is stable if it has a constant mean and a constant variance over time. The features “stability” and “lumpiness” are calculated based on tiled windows (windows cannot be overlapped on top of each other). For each window, mean and the variance are calculated. The feature stability is calculated based on the variance of means while the lumpiness is the variance of variances.

Spectral entropy of a time series

Spectral entropy of a time series is an information theory based measure which can be used as an measure of forecastability of a time series. We use the measure introduced by Goerg (2013) to estimate the spectral entropy. It estimates the Shannon entropy of the spectral density of a univariate (or multivariate) normalized spectral density. The spectral density of a univariate time series y_t can be defined as,

$$f_y(\lambda) = \frac{S_y(\lambda)}{\sigma_y^2},$$

where $S_y(\lambda)$ represents spectrum of a univariate stationary process which is the Fourier transformation of the autocovariance function,

$$S_y(\lambda) = \frac{1}{2\pi} \sum_{j=-\infty}^{\infty} \gamma_y(j) e^{ij\lambda},$$

and $\lambda \in [-\pi, \pi]$, and $i = \sqrt{-1}$.

The Shanon entropy of $f_y(\lambda)$ is define as,

$$H_{s,a}(y_t) := -\int_{-\pi}^{\pi} f_y(\lambda) log_a f_y(\lambda) d\lambda,$$

where $a > 0$ is the logarithm base. Since the periodogram is not a consistent estimator for $S_y(\lambda)$, weighted overlapping segment averaging(WOSA) introduced by Nuttall & Carter (1982) was used to estimate $S_y(\lambda)$.

The R package ForeCA (Forecastable Component Analysis) available at CRAN accompanies this work (Goerg 2016). As the name suggests ForeCA introduces a dimension reduction technique for time series analysis using the frequency domain properties of time series to determine the forecastability. This measure is calculated on the original series. Series that are easy to forecast should have a small value for the measure.

The Hurst exponent

The Hurst exponent is used to measure long-term memory of time series. We use the method presented in Wang, Smith-Miles & Hyndman (2009) to estimate the Hurst exponent. The Hurst exponent is estimated using the relation $H = d + 0.5$, where d, is fractal dimension of FARIMA(0, d, 0). Parameters are estimated using the maximum likelihood estimators. The likelihood is approximated using the method illustrated by Haslett & Raftery (1989). To fit FARIMA models we the fradiff package available in CRAN (Fraley 2012) which accompanies the work of Haslett & Raftery (1989).

Nonlinearity

To measure the degree of nonlinearity of the time series, we use Teräsvirts's neural network test for nonlinearity as in Wang, Smith-Miles & Hyndman (2009).

Parameter estimates of Holt's linear trend model

The forecasting equations and two-smoothing equations in Holt's linear trend model can be expressed as follow:

$$\begin{aligned} \text{Forecast equation:} \quad & \hat{y}_{t+h|t} = l_t + hb_t \\ \text{Level equation:} \quad & l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \\ \text{Trend equation:} \quad & b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \end{aligned}$$

where α is the smoothing parameter for the level, and β^* is the smoothing parameter for the trend. These parameters can vary between 0 and 1. The notations are as in Hyndman & Athanasopoulos (2018). We include the parameter estimates of both α and β to our feature set.

Parameter estimates of Holt-Winters additive method

The forecasting equations and three component equations for Holt-Winters additive method is:

$$\begin{aligned} \text{Forecast equation:} \quad & \hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+h_m^+} \\ \text{Level equation:} \quad & l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\ \text{Trend equation:} \quad & b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1} \\ \text{Seasonal equation:} \quad & s_t = \gamma(y_t + l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}. \end{aligned}$$

For mathematical background and notations, we refer the reader to Hyndman & Athanasopoulos (2018). We use the parameter estimates of α, β and γ to our feature set in case of seasonal time series.

Unit root test statistics based on Phillips-Perron test

The test regression for Phillips-Perron test is,

$$y_t = \alpha + (\phi - 1)y_{t-1} + \epsilon_t.$$

The hypotheses of interest are,

$$H_0 : \phi = 1 \text{ vs } H_1 : |\phi| < 1.$$

The test statistic we use as a feature is,

$$Z = T(\hat{\phi} - 1) - \frac{1}{2}\frac{T^2 SE(\hat{\pi})}{\hat{\sigma^2}}(\hat{\lambda}^2 - \hat{\sigma^2}).$$

The terms $\hat{\sigma}^2$ and $\hat{\lambda}^2$ are consistent estimates of the variance parameters,

$$\sigma^2 = \lim_{T\to\infty} T^{-1}\sum_{t=1}^{T} E[\epsilon_t^2],$$

$$\lambda^2 = \lim_{T\to\infty} \sum_{t=1}^{T} E[T^{-1}S_T^2],$$

where $S_T = \sum_{t=1}^{T}\epsilon_t$. The sample variance of the least squares residual $\hat{\epsilon}_t$ is a consistent estimate of σ^2, and the Newey-West long-run variance estimate of ϵ_t using $\hat{\epsilon}_t$ is a consistent estimate of λ^2.

Unit root test statistics based on KPSS test

The test regression is,

$$y_t = c + \delta t + \phi y_{t-1} + \epsilon_t.$$

The hypotheses of interest are,

$$H_0 : \phi = 1 \text{ vs } H_1 : |\phi| < 1.$$

The test statistic we use as a feature is,

$$Z = \left(T^{-2}\sum_{t=1}^{T}\hat{S}_t^2\right)/\hat{\lambda}^2$$

where $\hat{S}_t = \sum_{j=1}^{t}\hat{\epsilon}_j$, $\hat{\epsilon}_t$ is the least squares residuals and $\hat{\lambda}^2$ is a consistent estimate of the long-run variance of ϵ_t using $\hat{\epsilon}_t$.

Unit root tests based features are calculated using the functionality in package `urca`(Pfaff, Zivot & Stigler 2016).

Autocorrelation coefficient based features

The autocorrelation coefficients measure the strength of the linear relationship between lagged values of a time series. We calculate first-order autocorrelation coefficient and sum of squares of first five autocorrelation coefficients of the original series, first-difference series and second-difference series and seasonal differenced series (for seasonal data). These autocorrelation based are useful in identifying, i) stationary vs non-stationary processes, ii) random vs non-random processes, iii) difference stationary processes and seasonality present in the series.

First-order autocorrelation coefficient of the residual of linear trend model

A linear regression model is fitted considering $Y = \{y_1, y_2, \ldots, y_n\}$ as the dependent variable and and time $1, 2, \ldots, n$ as the independent variable. Then the first-order autocorrelation coefficient of the residual series is calculated. The purpose of including this feature is to discriminate between trend stationary and difference-stationary processes. If Y is trend-stationary and if a deterministic trend is fitted then the residuals are white noise. On the other hand if the Y is difference-stationary and a deterministic trend is fitted, residuals follow a random walk model.

Partial-autocorrelation based features

Partial-autocorrelation measures the relationship between y_t and y_{t-k} after removing the effects of other time lags – $1, 2, 3, \ldots, k-1$. We calculate the sum of squares of first five partial autocorrelation coefficients of the original series, first-difference series and second-difference series. Hence, this gives three features to our experiment. Partial-autocorrelation coefficients play an important role in Box-Jenkins(Box et al. 2015) approach to time series modelling as it helps to determine number of AR terms to be included in both AR(P) and ARIMA(P, d, Q).

References

Adya, M, F Collopy, JS Armstrong & M Kennedy (2001). Automatic identification of time series features for rule-based forecasting. *International Journal of Forecasting* **17**(2), 143–157.

Armstrong, JS (2001). Should we redesign forecasting competitions? *International Journal of Forecasting* **17**(1), 542–543.

Box, GEP, GM Jenkins, GC Reinsel & GM Ljung (2015). *Time series analysis: forecasting and control*. 5th ed. Hoboken, NJ: John Wiley & Sons.

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

Breiman, L & A Cutler (2004). *Random Forests*. Version 5.1. http://www.stat.berkeley.edu/users/breiman.

Chen, C, A Liaw & L Breiman (2004). *Using random forest to learn imbalanced data*. Tech. rep. University of California, Berkeley. http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf.

Cleveland, RB, WS Cleveland & I Terpenning (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3.

Collopy, F & JS Armstrong (1992). Rule-based forecasting: development and validation of an expert systems approach to combining time series extrapolations. *Management Science* **38**(10), 1394–1414.

Fraley, C (2012). *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*. R package version 1.4-2. https://CRAN.R-project.org/package=fracdiff.

Friedman, J, T Hastie & R Tibshirani (2009). *The elements of statistical learning*. 2nd ed. New York, USA: Springer.

Fulcher, BD & NS Jones (2014). Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Goerg, GM (2016). *ForeCA: An R package for forecastable component analysis*. R package version 0.2.4. https://CRAN.R-project.org/package=ForeCA.

Goerg, G (2013). Forecastable component analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp.64–72.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics* **38**(1), 1–50.

Hyndman, RJ (2001). It's time to move from what to why. *International Journal of Forecasting* **17**(1), 567–570.

Hyndman, RJ (2013). *Mcomp: Data from the M-Competitions*. R package version 2.05. https://CRAN.R-project.org/package=Mcomp.

Hyndman, RJ & G Athanasopoulos (2018). *Forecasting: principles and practice*. Melbourne, Australia: OTexts. http://OTexts.org/fpp2/.

Hyndman, RJ & Y Khandakar (2008). Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* **26**(3), 1–22.

Hyndman, RJ & AB Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* **22**(4), 679–688.

Hyndman, RJ, AB Koehler, RD Snyder & S Grose (2002). A state space framework for automatic forecasting using exponential smoothing methods. *International Journal of Forecasting* **18**(3), 439–454.

Hyndman, RJ, E Wang & N Laptev (2015). Large-scale unusual time series detection. In: *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*. IEEE, pp.1616–1619.

Hyndman, R, C Bergmeir, G Caceres, M O'Hara-Wild, S Razbash & E Wang (2018). *forecast: Forecasting functions for time series and linear models*. R package version 8.3. http://pkg.robjhyndman.com/forecast.

Kalousis, A & T Theoharis (1999). NOEMON: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis* **3**(5), 319–337.

Kang, Y, RJ Hyndman & K Smith-Miles (2017). Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting* **33**(2), 345–358.

Kück, M, SF Crone & M Freitag (2016). Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, pp.1499–1506.

Lawrence, M (2001). Why another study? *International Journal of Forecasting* **17**(1), 574–575.

Lemke, C & B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Liaw, A & M Wiener (2002). Classification and regression by randomForest. *R News* **2**(3), 18–22.

Makridakis, S, A Andersen, R Carbone, R Fildes, M Hibon, R Lewandowski, J Newton, E Parzen & R Winkler (1982). The accuracy of extrapolation (time series) methods: results of a forecasting competition. *Journal of forecasting* **1**(2), 111–153.

Makridakis, S & M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Nuttall, AH & GC Carter (1982). Spectral estimation using combined time and lag weighting. *Proceedings of the IEEE* **70**(9), 1115–1125.

Pfaff, B, E Zivot & M Stigler (2016). *urca: Unit root and cointegration tests for time series data*. R package version 1.3-0. https://CRAN.R-project.org/package=urca.

Prudêncio, RB & TB Ludermir (2004). Meta-learning approaches to selecting time series models. *Neurocomputing* **61**, 121–137.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Reid, DJ (1972). "A comparison of forecasting techniques on economic time series". In: *Forecasting in Action*. Birmingham, UK: Operational Research Society.

Rice, JR (1976). The algorithm selection problem. *Advances in Computers* **15**, 65–118.

Shah, C (1997). Model selection in univariate time series forecasting using discriminant analysis. *International Journal of Forecasting* **13**(4), 489–500.

Smith-Miles, K (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys (CSUR)* **41**(1), 6.

Wang, X, K Smith-Miles & RJ Hyndman (2009). Rule induction for forecasting method selection: meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10), 2581–2594.

Widodo, A & I Budi (2013). "Model selection using dimensionality reduction of time series characteristics". Paper presented at the International Symposium on Forecasting, Seoul, South Korea. June 2013. https://goo.gl/ig2J57.