UNIVERSITY OF CALIFORNIA

Los Angeles

Improving Automated Time Series Forecasting with the use of Model Ensembles

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Christopher Thomas Cavitt Meade

2019

ABSTRACT OF THE THESIS

Improving Automated Time Series
Forecasting with the use
of Model Ensembles

by

Christopher Thomas Cavitt Meade
Master of Applied Statistics
University of California, Los Angeles, 2019
Professor Frederic Paik Schoenberg, Chair

There currently exist several black box software libraries for the automatic forecasting of time series. Popular among these are the 'forecast' and 'bsts' packages for R, which have functions to automatically fit several common classes of time series models, such as the autoregressive integrated moving average (ARIMA) and the family of exponential smoothing models, among others. It is often the case that what one gains from the ease in fitting these automatic methods comes at the cost of predictive performance. In this paper, we propose several methods to improve the prediction accuracy of automatic time series forecasting, all of which relate to creating ensembles of models automatically fit from these packages. We explore different ways that one can construct these ensembles and evaluate each on a benchmark time series dataset. In addition, we provide the R code used to construct these ensembles.

The thesis of Christopher Thomas Cavitt Meade is approved.

University of California, Los Angeles

2019

For my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Several scenarios exist that necessitate automatic time series forecasting. A manufacturing firm may need to generate monthly production forecasts for each of its products, of which there could be thousands. A hedge fund manager may need to forecast the price of a security every 30 seconds using the latest available stock data. In either case, the high volume of time series data sets or the frequency at which forecasts must be made represent too high a cost for a statistician to manually apply the classical supervised methodology to fit an appropriate, causal, and invertible ARIMA model to each time series. In the first scenario, using such a methodology would consume too many man-hours of work, while in the second, a human could simply not keep up with the data.

In such situations, automatic 'black box' time series forecasting methods, like those implemented in the popular 'forecast' R software library by Rob J. Hyndman and the 'bsts' package by Steven L. Scott, deliver a compelling value proposition – adequate forecasts can be made almost instantly by anyone. Even if these automatic methods do not perform as well as the ideal, manually fitted ARIMA model, the value they create by reducing forecast costs (in hours of work or in time to model) can easily surpass the cost associated with implementing a less-than-ideal forecast. In the first scenario given above, 1,000 adequate production-ready forecasts are better than 100 excellent forecasts and 900 missing values because the statistician ran out of time.

The work presented in this paper deals with how these 'black box' automatic methods can be improved, so that a forecaster need not settle with adequate performance while still enjoying the benefits that black box methods provide. To that end, we explore several ways to create ensembles of four of the popular black box models included in the 'forecast' and ‘bsts’ packages to increase prediction accuracy over any single automatically fit model alone. To evaluate the performance of the ensembles, they will be tested on the M3-Competition data set, which contains 3,003 univariate time series with mostly monthly, quarterly, and yearly periods. Each series in the data set is split into a training and test set, the latter of which will be used to evaluate loss using normalized root mean squared error (nRMSE), mean absolute percent error (MAPE), and normalized mean absolute error (nMAE).

1.1 Related Work

Much work has been done in the space of time series forecasting to increase prediction accuracy with the use of model ensembles. In 2004, Wichard and Ogorzalek proposed combining the forecasts of several classes of time series models to make predictions. They were among the first to recognize that choosing different classes of models increases forecast diversity, thereby reducing forecast bias and variance, leading to an increase in prediction accuracy.

The component models used by Wichard and Ogorzalek include linear and polynomials models, nearest neighbor methods, and neural network methods, among others. Such methods are not as widely used by time series practitioners in comparison to more popular modeling frameworks, such as ARIMA and Exponential Smoothing. In 2018, Rob J. Hyndman, author of the ‘forecast’ package for R, proposed a similar ensemble methodology using more modern methods, taking the average of three different classes of models, ARIMA, Exponential Smoothing, and Theta, all automatically fit from the ‘forecast’ package. He claims such an ensemble should be regarded as a forecasting benchmark against which to test other forecasting methods.

Work proposed in this paper also builds off a time series bagging methodology proposed by Bergmeir et. al. (2016). This paper, which applies bootstrap aggregation to time series forecasting, constructs forecasts by combining Exponential Smoothing predictions for each bootstrapped series. We expand upon this work by combining ensemble forecasts instead of Exponential Smoothing forecasts alone, thereby creating ensembles of ensembles.

The methods proposed in this paper, like the ensembles proposed by Hyndman and

Bergmeir et. al., are evaluated with the M3 Competition Data set. This data set of 3,003 time series is often used to compare the performance of automatic forecasting methods. In fact, the original M3 Competition was first won by the Theta method, one of the four automatically fit models we attempt to improve upon in this paper.

Since its publication in 2000, the M3 Competition Data set has become the defacto forecasting benchmark data set. Rob Hyndman, editor of the International Journal of Forecasting (IJF), claims "The M3 data have continued to be used since 2000 for testing new time series forecasting methods. In fact, unless a proposed forecasting method is competitive against the original M3 participating methods, it is difficult to get published in the IFJ". Keeping with tradition, we evaluate our methods against the same data set.

CHAPTER 2

Automatic Time Series Forecasting

The time series ensembles proposed in this paper will be composed of combinations of four classes of models that can be fit automatically using the 'forecast' and 'bsts' packages in R. These four methods are chosen because they are prominently featured in time series literature, are commonly used by practitioners, and because each can be fit automatically. They are as follows:

2.1 Auto.Arima

Proposed by Hyndman and Khandakar (2008), Auto.Arima was designed to automatically select the best Autoregressive Integrated Moving Average (ARIMA) model for forecasting. Given a nonseasonal $ARIMA(p, d, q)$ process $(1 - B^d)y_t = c + \phi(B)y_t + \theta(B)\epsilon_t$ or seasonal $ARIMA(p, d, q, P, D, Q)_m$ process $(1 - B^m)(1 - B)^d y_t = c + \Phi(B^m)\phi(B)y_t + \Theta(B^m)\theta(B)\epsilon_t$, ϕ and θ are polynomials with orders p and q respectively, Φ and Θ are polynomials with orders P and Q respectively, d and D are the respective numbers of nonseasonal and seasonal differences, B is the backshift operator, ϵ is Gaussian white noise with mean 0 and variance σ^2, and c is a constant.

The goal of ARIMA modeling is to choose the parameters p, d, q, P, D, and Q which best fit the data. Parameter combinations can be evaluated by observing which best minimize an information criterion, such as AIC or BIC.

Auto.Arima provides a framework to automate the ARIMA parameter optimization process. If the data are nonseasonal, the algorithm first chooses d on the basis of KPSS unit-root tests, in which the data are checked for a unit root (Kwiatkowski et al., 1992). If a root is present, it is differenced and checked again. The process repeats until no unit root is detected.

For seasonal data, first D is chosen to equal 0 or 1 on the basis of the Canova-Hansen test (Canova and Hansen, 1995). If $D = 1$, seasonal differencing is applied. Then the KPSS unit-root test algorithm is applied to select d.

Auto.Arima then considers ARIMA models where p and q can take values ranging from 0 to 3 and, if applicable, P and Q can be set equal to 0 or 1. If $d + D \leq 1$, the constant c is fit. Otherwise it is set equal to 0. For the sake of efficiency, the model space is explored using a novel step-wise algorithm. Finally, the model which minimizes AIC is returned at the "best" model.

2.2 ETS

Short for both 'Error, Trend, Seasonality' and 'ExponenTial Smoothing', the 'ets' function from the 'forecast' packages fits 30 different classes of exponential smoothing models to a given time series and chooses best on basis of which minimizes AIC.

As the functions acronym suggests, an exponential smoothing model has three components: error, trend, and seasonality. Each of these three components can be further classified. Hyndman and Khandakar (2008) explain that error can be either "additive" or "multiplicative". Trend can be classified as "none", "additive", "additive damped", "multiplicative", or "multiplicative damped". Finally, seasonality can be classified as "none", "additive", or "multiplicative".

These 30 different combinations or error, trend, and seasonality comprise the 30 exponential smoothing models fit by the ets function. Hyndman and Khandakar provide a taxonomy of each of these 30 methods, including formulas for parameters optimization and point forecasts.

2.3 Theta

Proposed by Assimakopoulos and Nikolopoulos (2000), the Theta method applies a coefficient, θ, to a twice differenced time series in order to change its local curvature. For example, setting $\theta = 0$ reduces the time series to a simple linear regression. At the other extreme, setting $\theta = 2$, Assimakopoulos and Nikolopoulos explain, doubles the the local curvature, thereby magnifying the short term behavior of the series.

The first Theta-line, given by $\theta = 0$, is extrapolated via its simple linear trend. The second line, when $\theta = 2$, is extrapolated by simple exponential smoothing. Forecasts are made by combining these two extrapolations.

Assimakopoulos and Nikolopoulos have developed a six-step algorithm to automate forecasting using their Theta method. First, a time series is tested for seasonality by examining the autocorrelation function at the lag equal to the series periodicity. For example, for monthly data one would check the autocorrelation at lag 12, or lag 4 for quarterly data. Next, the data is deseasonalized if the seasonality was determined to be significant. Then two Theta-lines are generated, corresponding to $\theta = 0$ and $\theta = 2$. Next, these two lines are extrapolated via linear trend and simple exponential smoothing, respectively. Then these two lines are averaged with equal weights at each point in the forecast horizon. Finally, seasonality is reintroduced to the series.

This algorithm is provided in the forecast package as the function named 'thetaf' (Hyndman et. al., 2019).

2.4 BSTS

The BSTS framework, short for Bayesian Structural Time Series, was developed by Scott and Varian (2014) at Google to improve automated time series forecasting. According to Scott and Varian, this approach to time series forecasting combines three methods from Bayesian statistics – Kalman filtering, spike-and-slab regression, and model averaging.

The Bayesian Structural Time Series, according to Scott (2017), is defined in two equations. The first, called the observation equation, “relates the observed data y_t to a vector of latent variables α_t”, called the state. This equation is given by

$$y_t = Z_t^T \alpha_t + \epsilon_t \tag{2.1}$$

Scott continues, describing also a transition equation, which defines how these latent states, or α_t, change over time. The transition equation is given by

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t \tag{2.2}$$

In the two equations above, ϵ_t and η_t are defined as Gaussian white noise, while Z_t, T_t, and R_t are called the structural parameters.

Once the latent state vector is defined, the model is fit using a user defined number of Markov chain Monte Carlo algorithm iterations.

CHAPTER 3

The Case for Ensembles

3.1 Use in Machine Learning

The use of ensembles to improve prediction accuracy is nothing new in the realm of machine learning and statistical modeling. In 1996, Leo Breiman introduced the concept of bootstrap aggregation, shortened to 'bagging' (Breiman, 1996). Given a training set L, Bagging uses the bootstrap to create $B \in \mathbb{N}$ new training sets, L_i, $i = 1, ..., B$.

A given machine learning model is then fit to each of the new training sets. The final decision is made by taking the average of the B models in the case of regression, or by majority vote for classification. The method, one of the first implementations of ensemble learning, was successful, with Breiman concluding "What one loses [...] is a simple and interpretable structure. What one gains is increased accuracy." In the case of bagging, it was found that an ensemble of models performed better than any single learner alone. Dietterich (2002) gives three possible explanations to the improved predictive performance of ensemble learning.

For example, in the event of insufficient training data to establish a "best fit" model, it may be the case that several different learners provide an equally accurate but vastly different fit to the training data. Making a future prediction with only one of these models can be risky, due to the high variance of the predictions. However, an ensemble can reduce this prediction variance and thus reduce risk with a simple majority vote or average over all predictions.

It can also be the case that finding the best model to fit to a training data set can be too computationally expensive. This is especially true with gradient based methods, as a learner may become trapped at a local minimum and fail to reach the global minimum. In such a scenario, an ensemble reduces the risk of choosing the wrong minimum.

Finally, it is possible that no single learner can accurately model a given data. Creating an ensemble allows one to explore more functional relationships between data and model, which comprise what Dietterich calls the representation space. One of these previously untested members of the representation space may perform better than any of its component learners.

In each of the three explanations provided by Dietterich, the advantages attributed to ensemble learning directly relate to the diversity of its member learners (Oliveira, 2014). That is, the predictions made by the member learners are relatively uncorrelated. In bagging, uncorrelated learners are created with bootstrap sampling of the data set. The random forest, in addition to bootstrap sampling, uses a random subset of predictor variables to construct uncorrelated classification and regression trees (Breiman, 2001). The result is an even more diverse set a learners and often an even greater prediction accuracy.

3.2 Application to Time Series Forecasting

How, then, can an ensemble of uncorrelated time series models be constructed? The predictions from two reasonable ARIMA models fit to the same data will likely have a correlation coefficient close to one. An ensemble created from these two models will likely look very similar to both of the original forecasts and would therefore likely not enjoy the benefits of ensembling.

In time series forecasting, a natural way to create diverse forecasts is to combine the forecasts of different classes of time series models, such as the ARIMA, Theta, Exponential Smoothing, and Bayesian Structural Time Series, all of which were discussed in the previous section. Because forecasts created by these different classes of models will naturally vary, an ensemble of these learners should provide the same benefits as those ensemble methods mentioned above, namely a increase in prediction accuracy from a reduction in bias and variance and a robustness to training noise and outliers.

CHAPTER 4

Creating the Ensembles

We propose the following ensemble methods:

4.1 Naive Ensemble

Perhaps the simplest way to create an ensemble of multiple predictors is to take the average of all the predictions for each time point on the forecast horizon. Specifically, let X be a h by p matrix, where h is the prediction horizon (the number of time points into the future to be forecasted) and p is the number of models from which predictions were made. In this design, each column of the forecast matrix X corresponds to a prediction of horizon h made by one of the p models. Let W be a 1 by h weight matrix with every element given by $1/p$. Then a final forecast is given by XW. We call models structured in this fashion Naive Ensembles.

We are specifically interested in Naive Ensembles made with the four classes of automatic time series models we previously discussed. We will refer to this model as the BEAT (BSTS, ETS, Auto.Arima, Theta) Ensemble. Also of interest are the four Naive Ensembles made using three of the four component models, called BEA, EAT, BAT, and BET, following the same naming convention.

4.2 Median Naive Ensembles

The Naive Ensembles proposed in the previous section simply take the average of component forecasts at each time point in the forecast horizon. If one of these component forecasts predicts extreme or unreasonable values, it can have a huge impact on the accuracy of the

Naive Ensemble on the whole.

One solution to combat the influence of an extreme forecast is to take the median, rather than the mean, of the p predictors at each time point in the forecast horizon. If X is the same h by p forecast matrix, let $M = [m_1, ..., m_h]$ be the final forecast, where each m_i is the median of the ith row of $X, i = 1, ..., h$. We will specifically examine the medians of forecasts generated from the BEAT ensemble, and will refer to this method as the Median BEAT Ensemble.

4.3 Bagged Ensemble

Just how Breiman introduced bootstrap aggregation (bagging) to improve the accuracy of classification and regression tree ensembles, so too can bagging be used in time series analysis. Because of the order-dependent nature of time series, the standard methodology of resampling data points with replacement may be ill-suited to create bootstrapped series. Bergmeir et. al. (2016) proposes an adaptation specifically suited for time series analysis in such a way that the trend and seasonal structure of the time series is preserved.

The Box-Cox transformation is applied to the series to ensure that its trend and seasonality components are additive (Petropoulos et. al., 2018). The Box-Cox lambda parameter is chosen on the basis of maximum likelihood estimation (Box and Cox, 1964). The Box-Cox transformed series is then decomposed into seasonal, trend, and error components using LOESS (STL) (Cleveland et al., 1990).

Bootstrapped resampling is then applied to the error components of the series. Bergmeir et. al. utilize moving block bootstrap (Kunsch, 1989), whereby n data points are assigned to $n-b+a$ overlapping blocks, each with length b. Then n/b blocks are drawn with replacement and assigned in the order that they were drawn, creating a bootstrapped set of errors. This method is utilized in the event that there is any remaining autocorrelation in the LOESS residuals after the STL decomposition.

A new bootstrapped time series is then constructed by performing the inverse STL decomposition, whereby the trend, seasonality, and bootstrapped error terms are added together. The inverse Box-Cox transformation is then applied to return the time series to its original scale. This process then repeated a given number of times to create a set of bootstrapped series.

If a time series is not periodic, or has fewer than two periods, the Box-Cox transformation is applied, after which the series is decomposed into trend and error components using LOESS. Seasonality is not calculated. Then the same procedure as above is followed to create a set of bootstrapped time series.

Bergmeir et. al. then fit an ETS model to each of the bootstrapped time series, make a forecast, then take the median of the component forecasts for each point on the forecast horizon to make a final prediction. This method was found to perform better than the ETS model alone the M3 Competition Data set.

We propose the following adaptation to this method: given that we expect the BEAT ensemble to perform better than ETS alone under the ensemble hypothesis, we fit a BEAT ensemble to each bootstrapped time series instead of just an ETS model alone. Final forecasts will then be made by taking an equal-weight average of the BEAT point forecasts for each point on the forecast horizon as specified in the Naive Ensemble description. We also evaluate the median of the BEAT models, as done in the original paper, as described in the Median BEAT Ensemble specification. We refer to these methods as Mean Bagged BEAT and Median Bagged BEAT, respectively.

4.4 Random Error Resampling Ensemble

We propose the Random Error Resampling ensemble model for time series forecasting. This model is functionally very similar to the bagged model proposed in the previous section. Bagging utilizes bootstrap resampling of residuals after performing STL decomposition of a time series. However, it is common in time series literature to assume that these errors are distributed as Gaussian white noise with mean 0 and a finite variance (Schumway and Stoffer, 2011).

Therefore, instead of applying the moving block bootstrap to resample STL residuals to create a new set of bootstrapped time series, we create a new set of residuals by sampling from a Gaussian distribution with mean 0 and variance equal to the sample variance of the original STL residuals. We repeat this sampling a given number of times to create a bootstrapped set of time series.

Except for the previous step, the process is exactly the same as the bagged BEAT model. We evaluate the performance of this ensemble using both the mean and median of the component BEAT forecasts. We refer to these methods as the Mean Perturbed BEAT and Median Perturbed BEAT models, respectively.

The R code to construct each of the proposed ensembles is provided in the Appendix.

Table 4.1: Taxonomy of Methods

Method	Description
Auto.Arima	Function from forecast package to fit ARIMA model
ETS	Function from forecast package to fit exponential smoothing model
Theta	Function from forecast package to fit Theta model
BSTS	Function from bsts package to fit Bayesian structural time series model
BAT	Average of BSTS, Auto.Arima, and Theta forecasts
BEA	Average of BSTS, ETS, and Auto.Arima forecasts
BET	Average of BSTS, ETS, and Theta forecasts
EAT	Average of ETS, Auto.Arima, and Theta forecasts
BEAT	Average of BSTS, ETS, Auto.Arima, and Theta forecasts
medianBEAT	Median of BSTS, ETS, Auto.Arima, and Theta at each point in forecast horizon
meanBaggedBEAT	Mean of 10 bootstrapped BEAT forecasts at each point in forecast horizon
medianBaggedBEAT	Median of 10 bootstrapped BEAT forecasts at each point in forecast horizon
meanPertBEAT	Mean of 10 white noise resampled BEAT forecasts at each point in forecast horizon
medianPertBEAT	Median of 10 white noise resampled BEAT forecasts at each point in forecast horizon

CHAPTER 5

Methodology

Each ensemble method, in addition to the automatic component learners (Auto.Arima, Theta, BSTS, and ETS), will be evaluated on the M3 Competition Data set (Makridakis and Hibon, 2000). This data set contains 3,003 time series, each of which is divided into a training and a test set. Of the 3,003 series, 645 are yearly data, 756 are quarterly, 1428 are monthly, and 174 have frequencies that are not yearly, quarterly, or monthly (referred to as having periodicity 'Other'). In addition to containing series with varying periodicity, the type of series in the M3 data set span different industries and origins. Of the 3,003 series, 828 relate to microeconomic phenomenon, 731 are from macroeconomics, 519 are from industry, 308 are from finance, 413 are from demography, and 204 have other origins.

To ensure that enough data is available to make a reasonable forecast, each yearly series has at least 14 observations, quarterly series have at least 16 observations, monthly data have at least 48 observations, and series with frequencies that are not yearly, quarterly, or monthly (other) have at least 60 observations.

Each method will be fit to the training set of each series, and will create a forecast on the same time interval as the test set. In this way, forecasts can be evaluated against the ground truth for each of the 3,003 series.

The performance of the methods will be evaluated on the basis of normalized RMSE, normalized MAE, and MAPE. Standard RMSE and MAE are insufficient for this problem, as they are scale dependent, meaning that it does not make sense to compare the RMSE and MAE of the same model on multiple time series. We solve this problem by normalization. We first calculate the mean of training set, then divide RMSE and MAE of the model by this mean to get normalized RMSE (nRMSE) and normalized MAE (nMAE). This normalization process allows us to compare the performance of the models on the 3,003 series in the M3 Competition Data set. MAPE, mean absolute percent error, is already scale independent.

5.0.1 nRMSE

$$\frac{\frac{1}{T}\sqrt{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}}{\bar{Y}} \tag{5.1}$$

5.0.2 nMAE

$$\frac{\frac{1}{T}(\sum_{t=1}^{T}|\hat{y}_t - y_t|)}{\bar{Y}} \tag{5.2}$$

5.0.3 MAPE

$$\frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|/y_t \tag{5.3}$$

A smaller number represents a more accurate forecast for each of the three metrics. In evaluating the nRMSE, nMAE, and MAPE for each method on each of the time series in the competition data set, we may compare models and draw conclusions about the "best" methods by determining which best minimizes these loss functions by better predicting the test set.

CHAPTER 6

Experimental Results

A forecast for each of the 3,003 time series in the M3 Competition data set was made with each of the 4 base learners and 10 ensemble methods under consideration. Each forecast was compared with the ground truth values of the time series over the same horizon, allowing the three accuracy metrics, nRMSE, nMAE, and MAPE, to be calculated. The average of each metric for each of the 14 methods is shown below.

Table 6.1: Summary of Methods over M3 Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.209	0.178	18.771
2	BAT	0.194	0.164	17.697
3	BEA	0.199	0.169	18.028
4	BEAT	0.192	0.162	17.450
5	BET	0.191	0.161	17.500
6	BSTS	0.223	0.190	20.908
7	EAT	0.189	0.159	17.108
8	ETS	0.196	0.166	17.694
9	meanBaggedBEAT	0.192	0.163	17.094
10	meanPertBEAT	0.193	0.164	17.104
11	medianBaggedBEAT	0.191	0.162	16.817
12	medianBEAT	0.193	0.163	17.380
13	medianPertBEAT	0.193	0.163	16.835
14	THETA	0.189	0.160	17.092

We also construct the average of each metric grouped by time series periodicity – yearly, quarterly, monthly, and other. These aggregations are shown in Tables 6.2 - 6.5.

Table 6.2: Summary of Methods for Yearly Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.390	0.338	22.071
2	BAT	0.352	0.303	21.013
3	BEA	0.367	0.316	21.314
4	BEAT	0.347	0.299	20.724
5	BET	0.340	0.293	20.794
6	BSTS	0.401	0.347	24.218
7	EAT	0.341	0.294	20.407
8	ETS	0.354	0.305	21.016
9	meanBaggedBEAT	0.351	0.303	21.484
10	meanPertBEAT	0.355	0.307	21.663
11	medianBaggedBEAT	0.350	0.301	21.471
12	medianBEAT	0.354	0.305	20.989
13	medianPertBEAT	0.355	0.308	21.491
14	THETA	0.331	0.285	20.911

In addition to aggregating the performance of methods on the basis of the periodicity of the underlying series, we also aggregate the predictive performance of the 14 methods based on the origin the time series itself. As discussed earlier, these origins include the fields of microeconomics, macroeconomics, industry, finance, demography, and 'other'. These six aggregations are shown in tables 6.6 - 6.11.

We also examine how the nRMSE, nMAE, and MAPE change at each point in the forecast horizon. The nRMSE, nMAE, and MAPE are calculated at each point in the forecast horizon for each of the 3,003 time series and methods, then averaged. The average MAPE over each point in the horizon is provided in Figure 6.1. A detailed view of the average MAPE for horizon points 14 through 18 is given in Figure 6.2.

Table 6.3: Summary of Methods for Quarterly Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.147	0.127	13.229
2	BAT	0.133	0.113	11.993
3	BEA	0.140	0.119	12.376
4	BEAT	0.132	0.113	11.864
5	BET	0.131	0.112	11.704
6	BSTS	0.156	0.133	13.978
7	EAT	0.131	0.112	11.784
8	ETS	0.139	0.119	12.153
9	meanBaggedBEAT	0.135	0.115	12.106
10	meanPertBEAT	0.133	0.113	11.955
11	medianBaggedBEAT	0.134	0.115	12.072
12	medianBEAT	0.134	0.114	11.925
13	medianPertBEAT	0.132	0.113	11.864
14	THETA	0.134	0.115	11.846

Similarly, the average nRMSE and nMAE is shown in Figure 6.3. Because nRMSE and nMAE are calculated for only a single pair of points for each point in the horizon for each of the 3,003 series and 14 methods, nRMSE and nMAE are functionally equal in this plot.

Finally, we include Figures 6.4 - 6.7, which show actual forecasts of the 14 methods on two time series from the M3 Competition Data set. These figures depict some desirable properties of ensembles and highlight the advantages of the methods proposed in this paper.

Table 6.4: Summary of Methods for Monthly Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.179	0.149	21.913
2	BAT	0.173	0.144	20.797
3	BEA	0.174	0.145	21.150
4	BEAT	0.171	0.141	20.482
5	BET	0.173	0.143	20.628
6	BSTS	0.200	0.167	24.997
7	EAT	0.168	0.139	19.944
8	ETS	0.174	0.144	20.698
9	meanBaggedBEAT	0.169	0.140	19.217
10	meanPertBEAT	0.170	0.141	19.245
11	medianBaggedBEAT	0.168	0.139	18.660
12	medianBEAT	0.170	0.141	20.174
13	medianPertBEAT	0.170	0.140	18.817
14	THETA	0.172	0.142	19.562

Table 6.5: Summary of Methods for Other Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.043	0.037	4.821
2	BAT	0.042	0.037	4.743
3	BEA	0.042	0.037	4.787
4	BEAT	0.041	0.037	4.712
5	BET	0.042	0.037	4.795
6	BSTS	0.046	0.041	5.188
7	EAT	0.042	0.037	4.730
8	ETS	0.042	0.037	4.797
9	meanBaggedBEAT	0.044	0.039	5.077
10	meanPertBEAT	0.044	0.039	5.012
11	medianBaggedBEAT	0.044	0.039	5.049
12	medianBEAT	0.042	0.037	4.769
13	medianPertBEAT	0.043	0.038	4.912
14	THETA	0.048	0.042	5.465

Table 6.6: Summary of Methods for Microeconomics Series

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.312	0.263	31.576
2	BAT	0.295	0.246	29.589
3	BEA	0.299	0.250	30.028
4	BEAT	0.290	0.243	29.183
5	BET	0.291	0.243	29.194
6	BSTS	0.339	0.285	34.034
7	EAT	0.287	0.239	28.745
8	ETS	0.299	0.251	29.837
9	meanBaggedBEAT	0.290	0.242	28.923
10	meanPertBEAT	0.290	0.242	29.287
11	medianBaggedBEAT	0.290	0.242	28.662
12	medianBEAT	0.289	0.241	29.003
13	medianPertBEAT	0.291	0.243	28.898
14	THETA	0.288	0.240	27.962

Table 6.7: Summary of Methods for Macroeconomics Series

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.091	0.079	6.895
2	BAT	0.087	0.075	6.154
3	BEA	0.090	0.078	6.365
4	BEAT	0.087	0.075	6.174
5	BET	0.088	0.076	6.170
6	BSTS	0.099	0.085	7.064
7	EAT	0.086	0.075	6.231
8	ETS	0.093	0.080	6.559
9	meanBaggedBEAT	0.086	0.074	6.156
10	meanPertBEAT	0.085	0.074	6.100
11	medianBaggedBEAT	0.086	0.074	6.168
12	medianBEAT	0.088	0.076	6.261
13	medianPertBEAT	0.085	0.074	6.077
14	THETA	0.090	0.079	6.424

Table 6.8: Summary of Methods for Industry Series

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.223	0.186	14.802
2	BAT	0.204	0.171	14.170
3	BEA	0.214	0.179	14.615
4	BEAT	0.205	0.171	14.132
5	BET	0.203	0.170	14.330
6	BSTS	0.225	0.189	16.574
7	EAT	0.203	0.170	13.806
8	ETS	0.215	0.180	14.674
9	meanBaggedBEAT	0.207	0.173	14.419
10	meanPertBEAT	0.207	0.173	14.270
11	medianBaggedBEAT	0.207	0.173	14.336
12	medianBEAT	0.209	0.175	14.148
13	medianPertBEAT	0.207	0.174	14.261
14	THETA	0.200	0.168	13.968

Table 6.9: Summary of Methods for Financial Data

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.334	0.295	34.397
2	BAT	0.302	0.265	35.713
3	BEA	0.317	0.277	36.483
4	BEAT	0.297	0.260	35.059
5	BET	0.290	0.253	35.719
6	BSTS	0.357	0.314	44.263
7	EAT	0.285	0.248	32.824
8	ETS	0.288	0.250	33.834
9	meanBaggedBEAT	0.294	0.256	30.160
10	meanPertBEAT	0.298	0.260	29.621
11	medianBaggedBEAT	0.286	0.249	28.468
12	medianBEAT	0.306	0.268	34.199
13	medianPertBEAT	0.293	0.256	28.143
14	THETA	0.270	0.231	32.241

Table 6.10: Summary of Methods for Demographic Series

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.166	0.142	12.599
2	BAT	0.151	0.128	10.242
3	BEA	0.153	0.130	10.234
4	BEAT	0.146	0.124	9.836
5	BET	0.145	0.123	9.501
6	BSTS	0.184	0.156	13.447
7	EAT	0.144	0.122	10.175
8	ETS	0.146	0.124	9.816
9	meanBaggedBEAT	0.154	0.131	11.012
10	meanPertBEAT	0.158	0.135	11.063
11	medianBaggedBEAT	0.152	0.129	10.861
12	medianBEAT	0.146	0.123	10.094
13	medianPertBEAT	0.157	0.134	11.056
14	THETA	0.152	0.131	11.195

Table 6.11: Summary of Methods for Unlabeled Series

	Method	nRMSE	nMAE	MAPE
1	Auto.Arima	0.070	0.059	8.349
2	BAT	0.064	0.055	7.653
3	BEA	0.065	0.056	7.715
4	BEAT	0.064	0.054	7.513
5	BET	0.063	0.054	7.384
6	BSTS	0.069	0.059	8.107
7	EAT	0.065	0.055	7.554
8	ETS	0.065	0.055	7.570
9	meanBaggedBEAT	0.066	0.056	7.673
10	meanPertBEAT	0.066	0.056	7.632
11	medianBaggedBEAT	0.066	0.056	7.671
12	medianBEAT	0.065	0.055	7.627
13	medianPertBEAT	0.065	0.056	7.601
14	THETA	0.070	0.060	8.220

Figure 6.1: MAPE Over Time

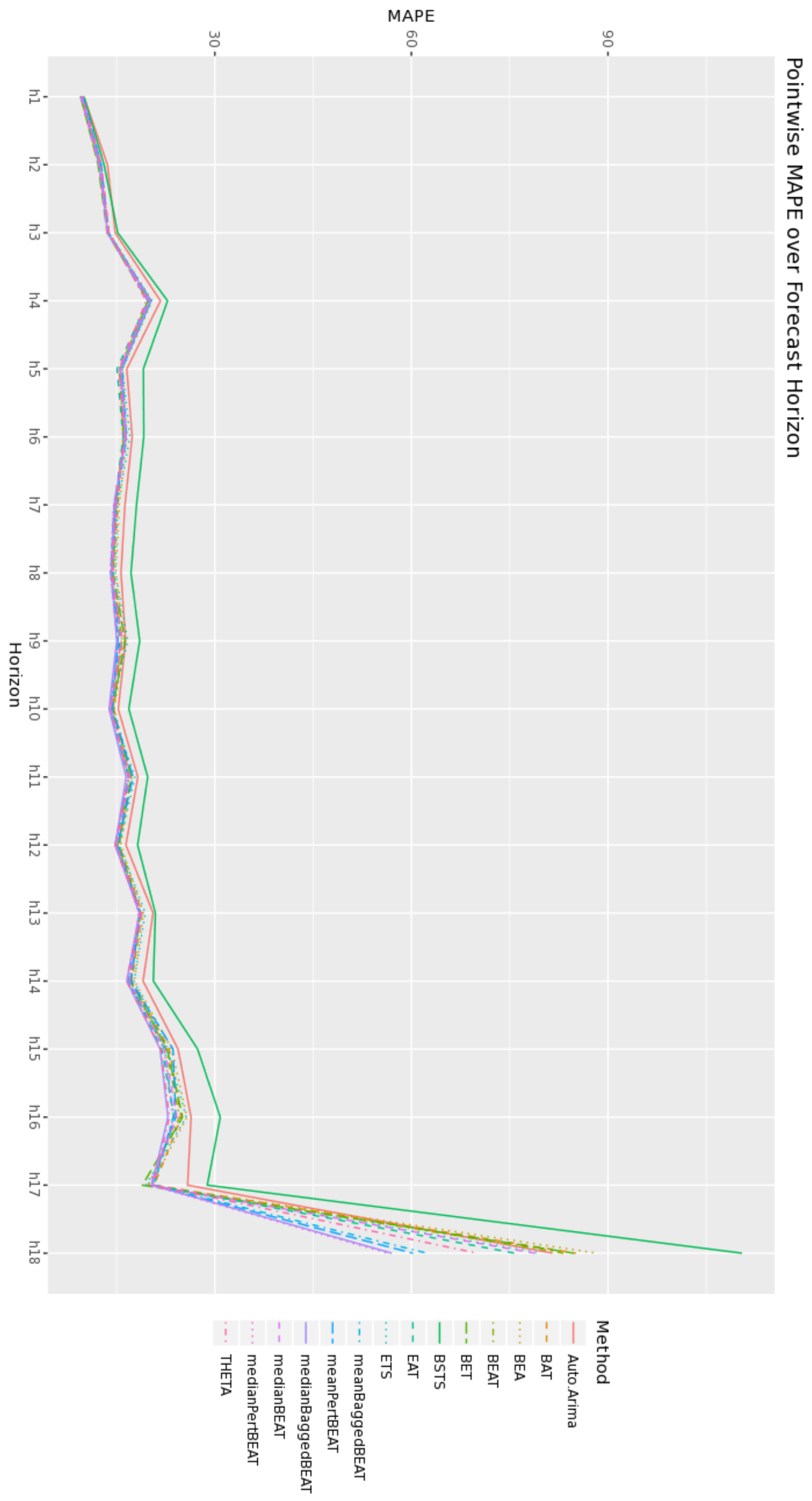


Figure 6.2: MAPE h14 - h18 Detail

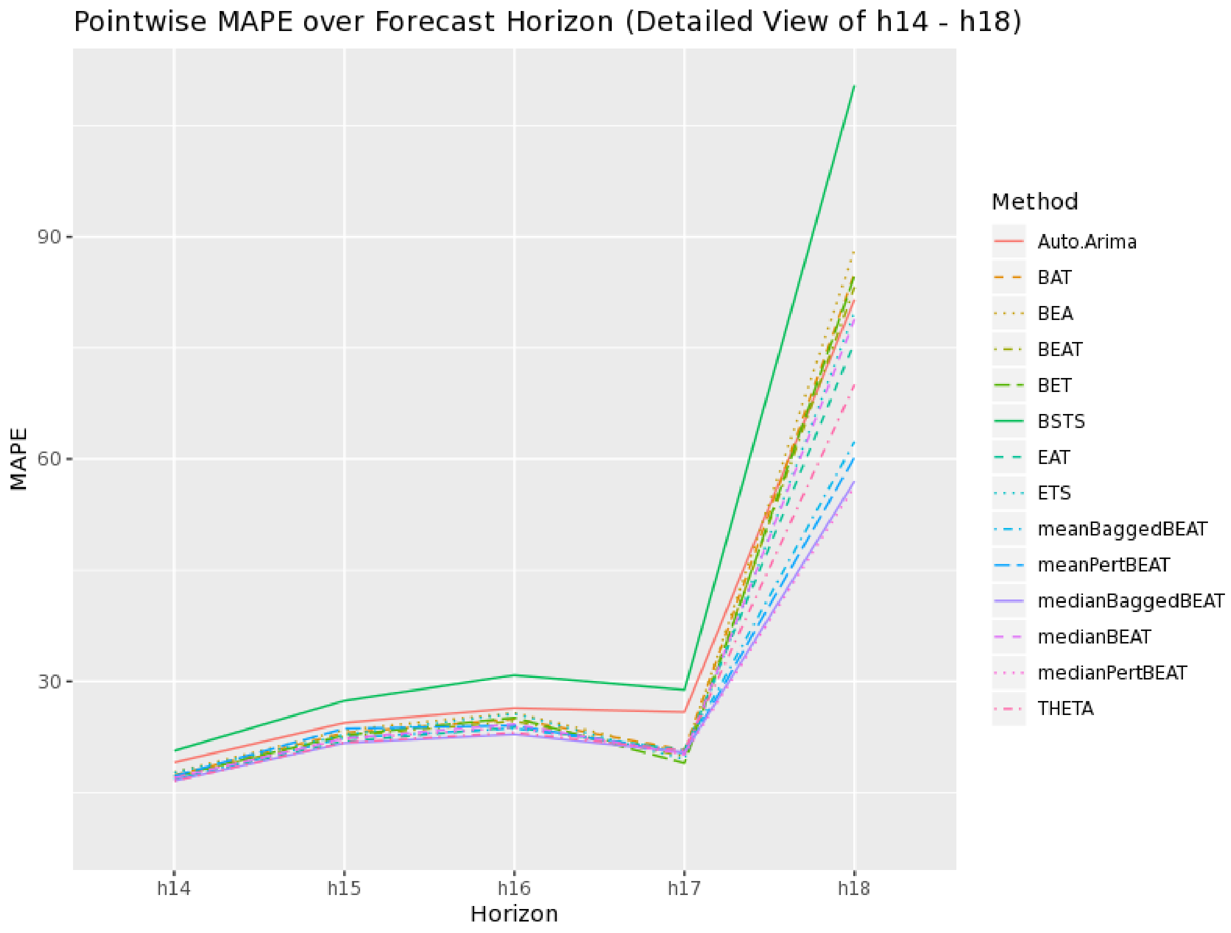


Figure 6.3: nRMSE and nMAE Over Time

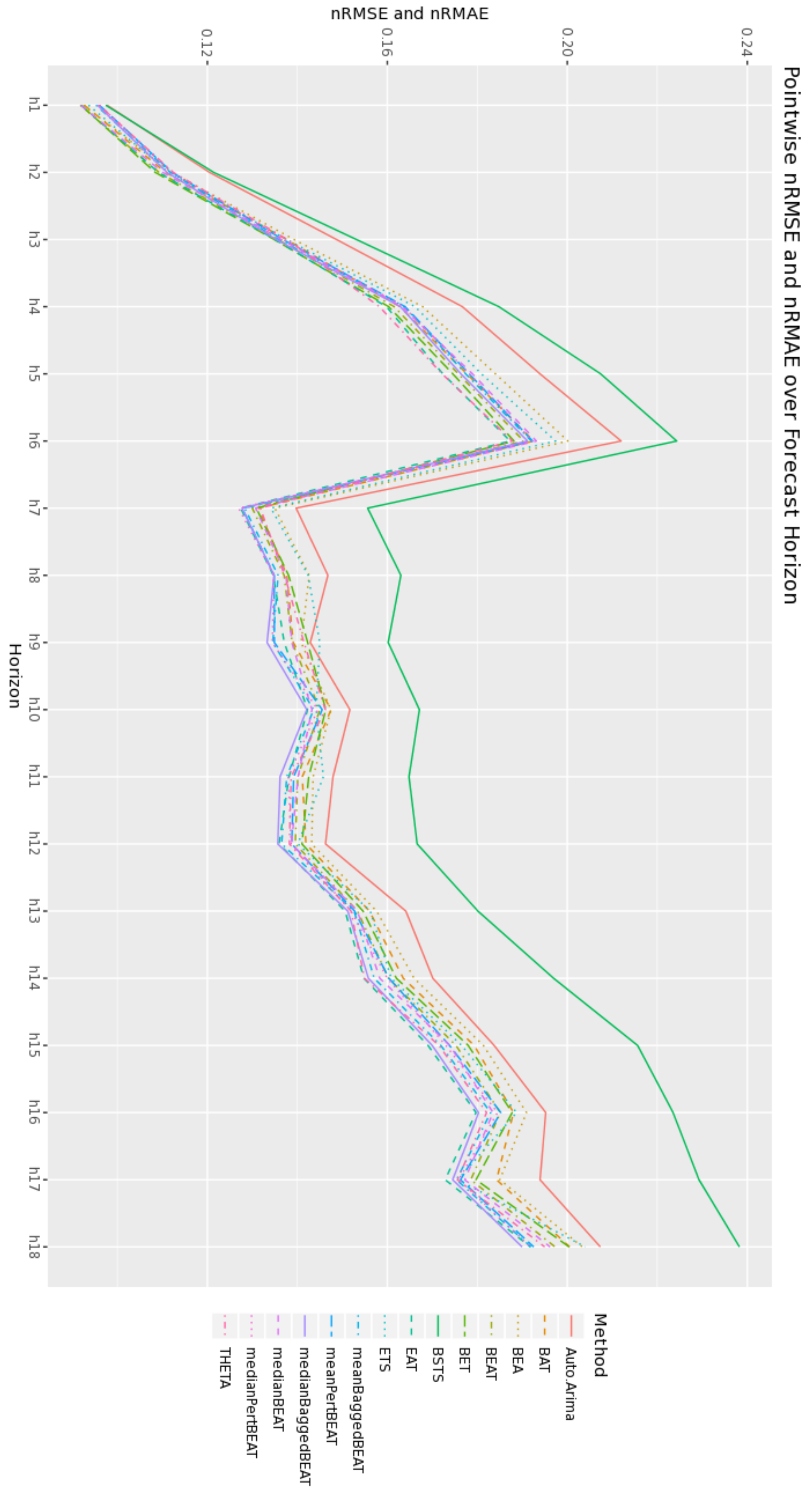


Figure 6.4: Series N2698

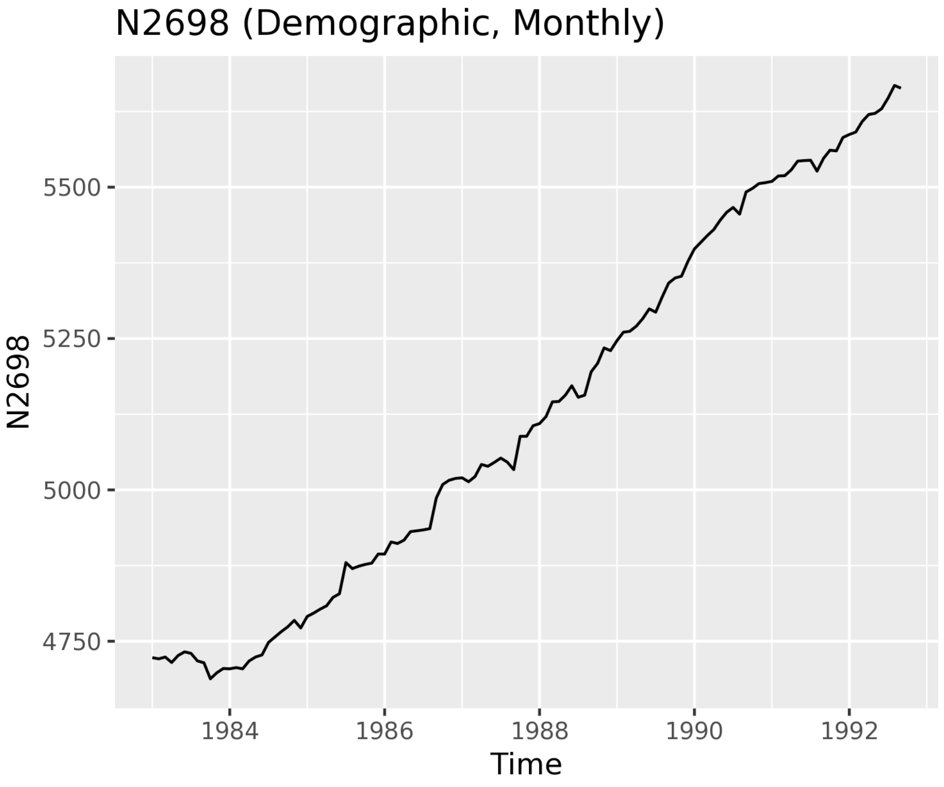


Figure 6.5: Forecast of Series N2698

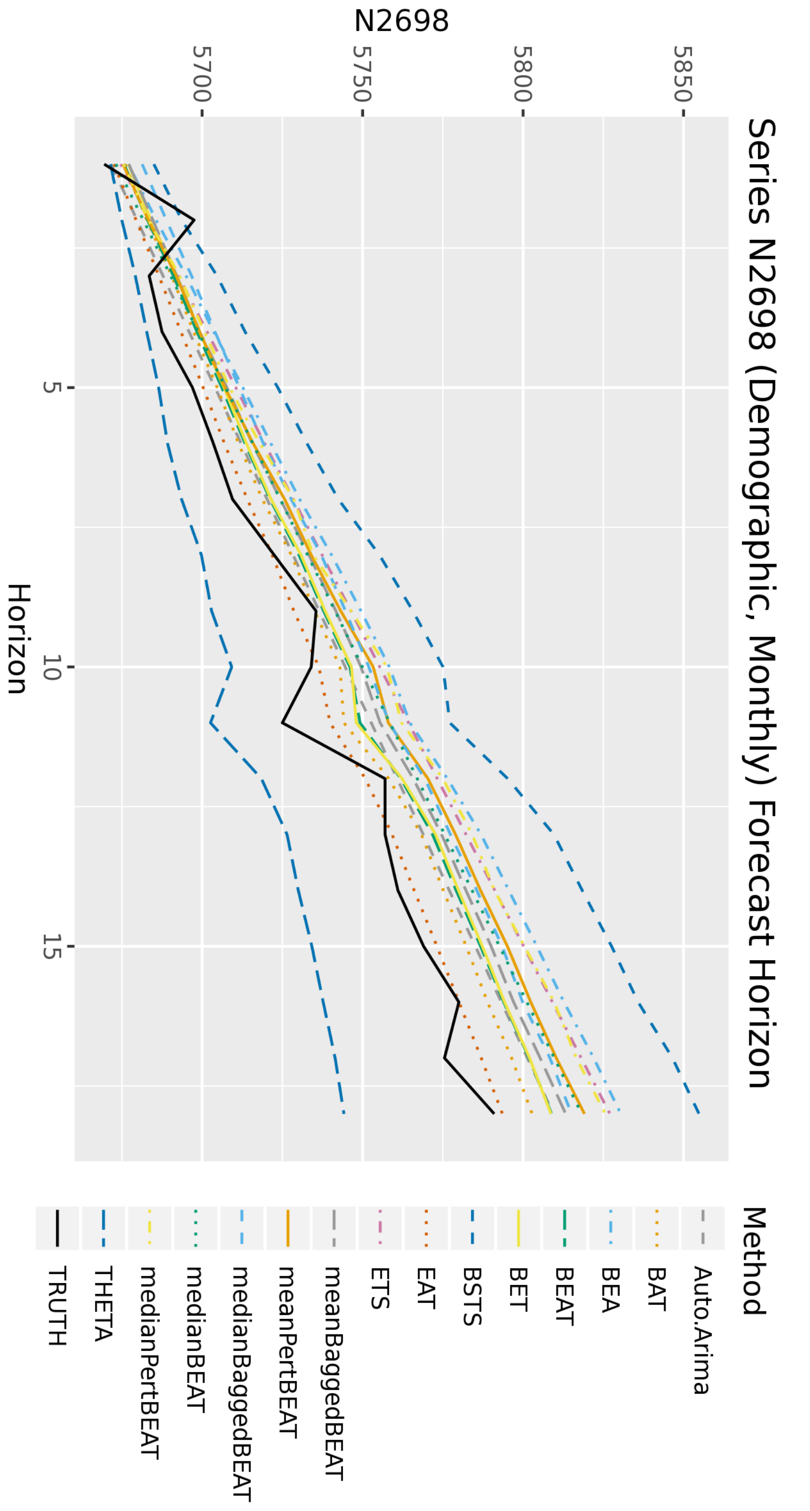


Figure 6.6: Series N2459

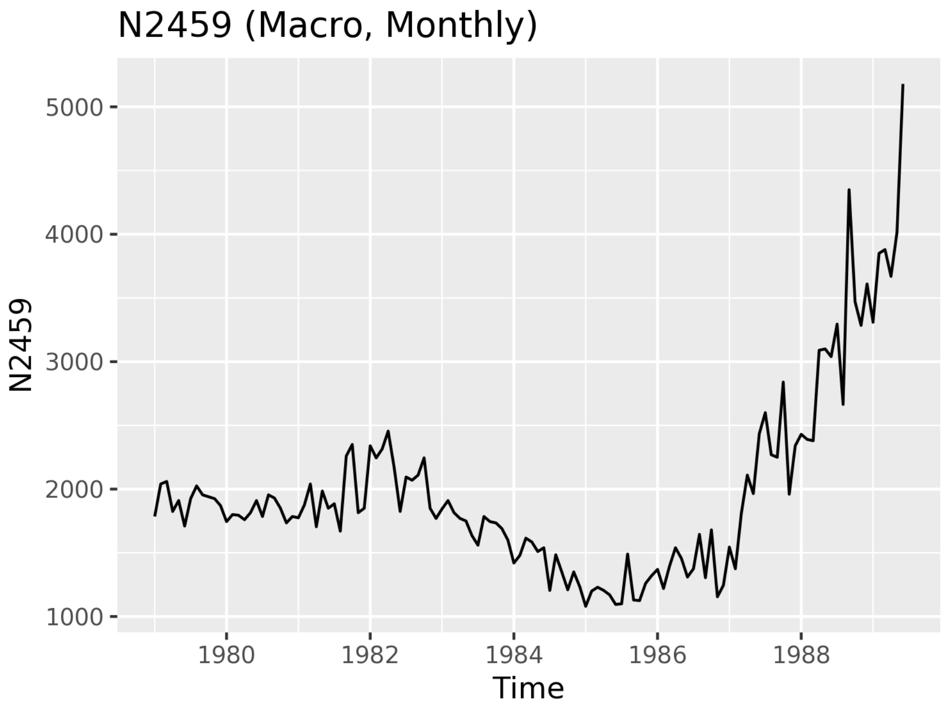
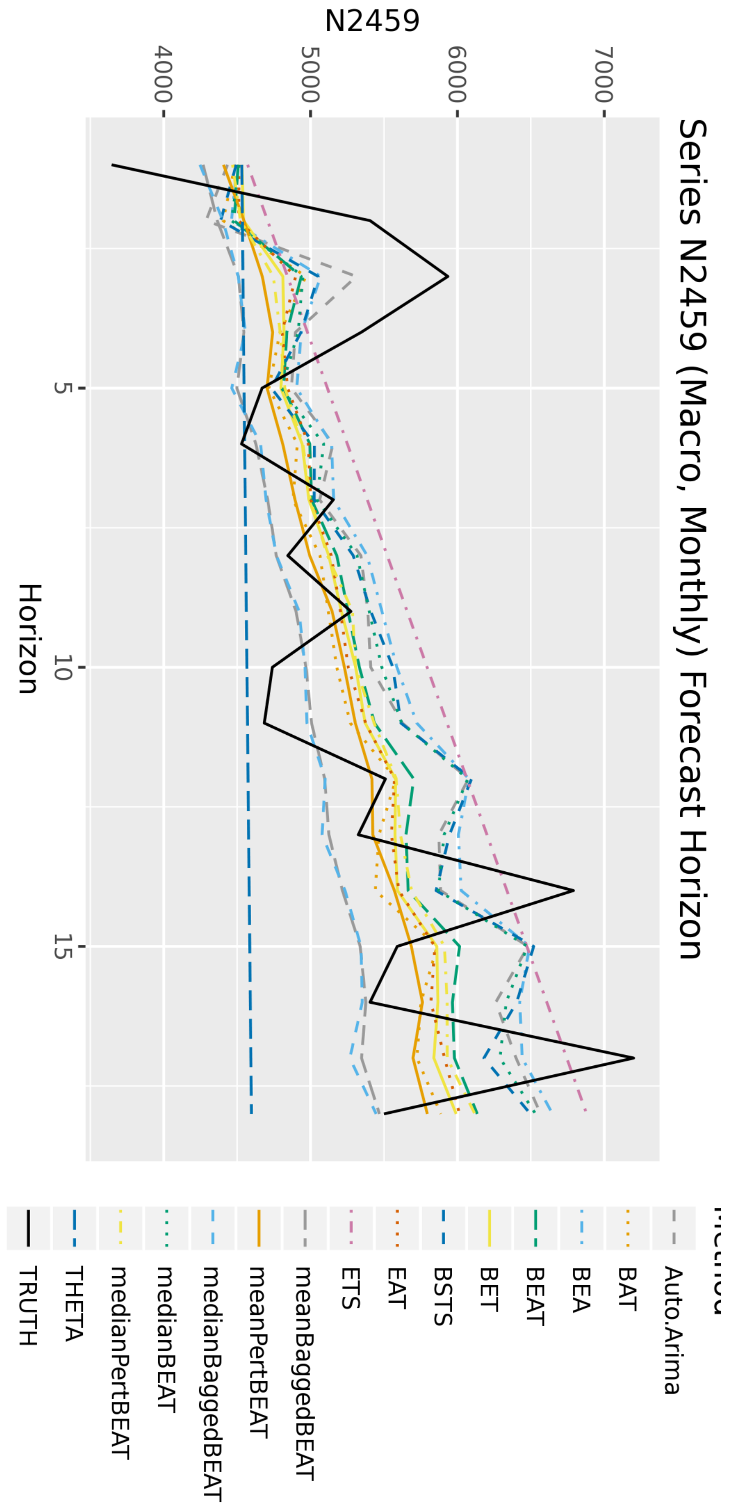


Figure 6.7: Forecast of Series N2459



CHAPTER 7

Discussion

Experimental data indicate that ensembling methods can offer improvements in predictive performance over their component learners. From Table 6.1, which shows the average performance of each method over the entire data set of 3,003 time series, the medianBaggedBEAT model performs the best of 14 methods when evaluated by MAPE, and the EAT Naive Ensemble performs best on the basis of nMAE. The EAT ensemble also minimizes the nRMSE, however it ties with the Theta method.

Table 6.2, which displays the average nRMSE, nMAE, and MAPE of the 14 methods, restricted only to yearly data, indicates that ensemble methods perform especially well in minimizing the MAPE of time series with this periodicity. However, the Theta method minimizes both nRMSE and nMAE. This is in stark contrast to the other component methods, which perform significantly worse than Theta in each of the three evaluation metrics.

Theta does not seem to perform as well when dealing with quarterly time series, however, as shown in Table 6.3. Here, the BET Naive Ensemble achieves the best score for each of nRMSE, nMAE, and MAPE. The EAT ensemble does not perform much worse, tying BET in nRMSE and nMAE, but achieving a slightly worse MAPE. In fact, three of the component models, Auto.Arima, BSTS, and ETS, perform worse than every single one of the ensemble methods when evaluated by MAPE. Auto.Arima and BSTS in particular perform worse than all other methods in each of the three metrics.

When dealing with monthly data, as shown in Table 6.4, the four ensemble methods that utilize resampling – meanBaggedBEAT, meanPertBEAT, medianBaggedBEAT, and medianPertBEAT – perform exceptionally well, achieving lower average nRMSE, nMAE, and MAPE values than all four of component methods, Auto.Arima, BSTS, ETS, and Theta.

For those series with periodicity that is not yearly, quarterly, or monthly, the Theta method is ranked last of the 14 methods in each of the three evaluation metrics, as shown in Table 6.5. This is in stark contrast to the performance of Theta for yearly, quarterly, or monthly series, where it usually performed best of the component learners. In these series with nonstandard periodicity, the family of Naive Ensembles, which includes BAT, BEA, BET, EAT, and BEAT, perform best on average, edging out slight accuracy increases over the family of resampling ensembles. In particular, the BEAT ensemble performs the best of all 14 methods in each metric.

For time series that relate to microeconomics, the EAT Naive Ensemble minimized average nRMSE and nMAE, as seen in Table 6.6. The Theta method achieved the lowest average MAPE, however. As is often the case, Auto.Arima, ETS, and BSTS were the three worst performers in each of average nRMSE, nMAE, while ETS performed better than two ensemble models on the basis of MAPE.

In the case of series relating to macroeconomics, the family of resampling ensembles are again superior. Of the four, medianPertBEAT is able to minimize average MAPE and tie for lowest average nRMSE and nMAE, as shown in Table 6.7. For macroeconomic data, the superiority of the ensemble methods is clear – all four component models are either the worst performers or tie for worst, in terms of average nRMSE, nMAE, and MAPE.

For series relating to industry, results are shown in Table 6.8. The Theta method performs exceptionally well here, minimizing both average nRMSE and average nMAE. As is often the case, the other three component learners are the three worst methods on average for these series. The EAT Naive Ensemble is able to edge out a slight decrease in average MAPE over the Theta method, but besides that all ensemble methods are inferior to Theta in this case.

For series related to demography, results are shown in Table 6.10. The family of Naive Ensembles again offer the best average predictive performance. Average nRMSE and nMAE are minimized by EAT, while average MAPE is minimized by BET. Here again are Auto.Arima and BSTS the worst of the 14 methods for the three evaluation metrics. The Theta method delivers a competitive average nRMSE and the ETS method has the second lowest average

MAPE, but each suffers in the other two respective metrics.

For those series in the M3 data set whose origin is not specified, the Theta method achieves a markedly lower ranking in each of the three evaluation metrics, just as is the case for series where the periodicity was non-standard. The clear best-performing model for this type of data, just as for the data is non-standard periodicity, is the BEAT Naive Ensemble, which ties for or achieves the lowest average nMAE and MAPE, and ranks second for lowest average nRMSE, behind only the BET ensemble.

Figure 6.1, which displays the average MAPE of each of the 14 methods at all points in the forecast horizon, sheds light on how ensemble methods increase forecast accuracy. These increases appear to be primarily driven by the better predictive performance of ensemble methods further down the forecast horizon. A more detailed view of the point-wise average MAPE of tail of the forecast horizon is shown in Figure 6.2. Two of the component learners, Auto.Arima and BSTS, show clear deviation from the ensemble methods the further out that forecasts are made.

The same deviation is clear when evaluating the point-wise nRMSE and nMAE, shown in Figure 6.3. The performance gap between Auto.Arima and BSTS against the ensemble methods becomes increasingly stark the further out a time series is forecast.

Figure 6.4 depicts series N2698 from the M3 Competition Data set. This time series comes from the field of demography and has monthly periodicity. The forecasts made on N2698 are shown in Figure 6.5, along with the ground truth in black. We include this time series because it highlights a few desirable properties of ensembles, namely the increase in accuracy through the reduction of bias and variance. In Figure 6.5, the forecasts from two of the component methods, Theta and BSTS, appear biased. In addition, forecasts from these methods also appear to become more inaccurate over time. However, the ensemble methods correct for both of these issues. They have a clear reduction in bias, and do not appear to become more inaccurate over time.

Figure 6.6 shows N2459, a monthly series from macroeconomics. Forecasts on N2459 are shown in Figure 6.7. Like the previous example, N2459 is included to highlight another advantage of ensemble methods. Here, that advantage is protection against a single poorly performing model. In Figure 6.7, the forecast provided by the Theta method appears flat and does not follow the trend or seasonality of the series, making it ill-suited to model these data. Ensembles containing Theta forecasts, however, still accurately model the series, despite its poor predictive performance.

CHAPTER 8

Conclusion

In most cases, ensembling automatically fit time series models has been shown to offer an increase in forecast accuracy, especially when the alternative is using an automatically fit Auto.Arima, BSTS, or ETS model. These three component learners were inferior to almost all of the proposed ensemble methods for every periodicity and type of time series tested in the M3 data set.

Of the four component learners, the Theta method was often able to deliver average nRMSE, nMAE, and MAPE scores that were competitive with the 10 ensemble methods. However, for time series with non-standard periodicity – that is, time series with periods that are not yearly, quarterly, or monthly – and for time series without origins in microeconomics, macroeconomics, industry, finance, or demography, the Theta method was often the worst of the 14 methods in each evaluation metric.

With regards to the drawbacks of using only one of Auto.Arima, BSTS, ETS, or Theta to make a forecast, and in light of the advantages of the 10 ensembles proposed in this paper, a practitioner is faced with the natural question of 'Which ensemble method should be used?'. This is a difficult question, and an important one.

This question must be answered with regards to the type of time series to be forecast, the periodicity of these time series, and the evaluation metric of interest. With these qualities in mind, a practitioner should design an experiment similar to the one in this paper, using time series that most closely resemble those that are to be automatically forecast. In evaluating which method performs the best in the experiment, the choice in which method to use becomes easy.

If such an experiment in not practical or feasible, the results of this paper indicate that the

BEAT Naive Ensemble and medianBaggedBEAT Ensemble both provide good performance regardless of periodicity or origin of the underlying series. This advantage make both good options for general-use time series forecasting.

APPENDIX A

Code

A.1 Naive Ensembles

```
library(forecast)
library(bsts)

naiveEnsemble <- function(ts, h){
  # ts is a time series object
  # h is the forecast horizon, the number
  # of data points into the future to forecast

  ### Fit the automatic models to the ts object
  # Arima Forecast
  aa <- forecast(auto.arima(ts), h = h)$mean

  # BSTS
  seasonal <- findfrequency(ts)
  ss <- AddLocalLinearTrend(list(), ts)
  if(seasonal>1){ss <- AddSeasonal(ss, ts, nseasons = seasonal)}
  model <- bsts(ts, state.specification = ss,
  niter = 1000, family = "gaussian")
  bsts <- predict(model, horizon = h, burn = 100)$mean
  # Exponential Smoothing
  ets <- forecast(ets(ts), h = h)$mean

  # Theta forecast
  theta <- forecast(thetaf(ts, h = h), h = h)$mean

  # Combine each h by 1 forecast into
  # a h by p matrix, where p is the number
  # of models fit to the data

  # Forecast Matrix
  fcMat <- as.matrix(cbind(as.numeric(aa),
                           as.numeric(bsts),
                           as.numeric(ets),
                           as.numeric(theta)))

  # Return the Forecast Matrix and
  # the average forecast at each point on
  # the horizon as a list object

  out <- list(predictionMatrix = fcMat,
              forecast = rowMeans(fcMat))

  return(out)
}

# Fit the Naive Ensemble Matrix
fcMat <- naiveEnsemble(ts = ts, h = h)
```

```
# Naive Methods are given below
BEA <- rowMeans(fcMat[,c(1,2,3)])
EAT <- rowMeans(fcMat[,c(2,3,4)])
BAT <- rowMeans(fcMat[,c(1,3,4)])
BET <- rowMeans(fcMat[,c(1,2,4)])
BEAT <-rowMeans(fcMat)
medianBEAT <- apply(fcMat, 1, median)
```

A.2 Bagging Methods

```
library(parallel)
library(doMC)
doMC::registerDoMC(cores = detectCores())

baggedBEAT <- function(ts, h){
  bootList <- forecast::bld.mbb.bootstrap(ts, 10)

  outDF <- foreach(i = 1:length(bootList)) %dopar% {
    out <- as.numeric(naiveEnsemble(bootList[[i]],
                                    h = h)[[2]])
  }

  as.data.frame(do.call(rbind, outDF))

}

baggedBEAT <- baggedBEAT(ts = ts, h = h)
meanBaggedBEAT <- as.numeric(colMeans(baggedBEAT))
```

```
medianBaggedBEAT <- apply(baggedBEAT, 2, median)
```

A.3 Error Perturbation Resampling Methods

```
error.resamp <- function(x, num, block_size=NULL) {
  freq <- frequency(x)
  if (is.null(block_size)) {
    block_size <- ifelse(freq > 1,
                         2 * freq,
                         min(8, floor(length(x) / 2)))
  }

  xs <- list()
  xs[[1]] <- x # the first series is the original one

  if (num > 1) {
    # Box-Cox transformation
    if (min(x) > 1e-6) {
      lambda <- BoxCox.lambda(x, lower = 0, upper = 1)
    } else {
      lambda <- 1
    }
    x.bc <- BoxCox(x, lambda)
    lambda <- attr(x.bc, "lambda")

    if (freq > 1) {
      # STL decomposition
      x.stl <- stl(ts(x.bc,
                   frequency = freq), "per")$time.series
      seasonal <- x.stl[, 1]
      trend <- x.stl[, 2]
      remainder <- x.stl[, 3]
    } else {
      # Loess
      trend <- 1:length(x)
      suppressWarnings(
        x.loess <- loess(x.bc ~ trend,
                         span = 6 / length(x), degree = 1)
      )
      seasonal <- rep(0, length(x))
      trend <- x.loess$fitted
      remainder <- x.loess$residuals
    }

    for (i in 2:num) {
      xs[[i]] <- InvBoxCox(trend +
                           seasonal +
                           rnorm(length(remainder),
                           0,
                           sd(remainder)),
                           lambda)
    }
  }

  xs
}

pBEAT <- function(ts, h){
  bootList <- error.resamp(ts, 10)

  outDF <- foreach(i = 1:length(bootList)) %dopar% {
    out <- as.numeric(naiveEnsemble(bootList[[i]],
                                    h = h)[[2]])
  }

  as.data.frame(do.call(rbind, outDF))
}

pBEAT <- pBEAT(ts, h)
meanPertBEAT <- as.numeric(colMeans(pBEAT))
medianPertBEAT <- apply(pBEAT, 2, median)
```

REFERENCES

[1] J. D. Wichard and M. Ogorzalek. (2004). *Time series prediction with ensemble models* 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), Budapest, pp. 1625-1630 vol.2.

[2] Hyndman, Rob. (2018). *A forecast ensemble benchmark* `https://robjhyndman.com/hyndsight/benchmark-combination/`

[3] Hyndman, Rob, and Yeasmin Khandakar. (2008). *Automatic Time Series Forecasting: The forecast Package for R.* Journal of Statistical Software [Online], 27.3.

[4] Kwiatkowski, Denis, Phillips, Peter, Schmidt, Peter and Shin, Yongcheol. (1992). *Testing The Null Hypothesis of Stationarity Against The Alternative of A Unit Root. How Sure Are We That Economic Time Series Have Unit Root?.* Journal of Econometrics. 54. 159-178.

[5] Fabio Canova and Bruce E. Hansen (1995). *Are Seasonal Patterns Constant Over Time? A Test for Seasonal Stability*, Journal of Business and Economic Statistics, 13:3, 237-252.

[6] Assimakopoulos, Vassilis and Nikolopoulos, Konstantinos. (2000). *The theta model: A decomposition approach to forecasting.* International Journal of Forecasting. 16. 521-530.

[7] Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2019). *forecast: Forecasting functions for time series and linear models.* R package version 8.7, `http://pkg.robjhyndman.com/forecast`.

[8] Steven L Scott, Hal R Varian (2014). *Predicting the present with bayesian structural time series.* International Journal of Mathematical Modelling and Numerical Optimisation, vol. 5 (2014), pp. 4-23

[9] Steven L Scott (2017). *Fitting Bayesian structural time series with the bsts R package.* `http://www.unofficialgoogledatascience.com/`.

[10] Breiman, L. (1996) *Bagging Predictors.* Machine Learning 24: 123.

[11] Dietterich, T. G. (2002). *Ensemble Learning.* In The Handbook of Brain Theory and Neural Networks, Second edition, (M.A. Arbib, Ed.), Cambridge, MA: The MIT Press, 2002. 405-408.

[12] Torgo, Luis and Oliveira, Mariana. (2014). *Ensembles for Time Series Forecasting.*

[13] Breiman, L. (2001). *Random Forests* Machine Learning, 2001. 45: 5.

[14] Bergmeir, C., Hyndman, R. J., Benitez, J. M., (2016). *Bagging exponential smoothing methods using STL decomposition and Box-Cox transformation.* International Journal of Forecasting 32, 303312.

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

[16] Box, G.E.P. and Cox, D.R. (1964). *An Analysis of Transformations.* Journal Royal Statistical Society Series B, 26, 211- 252.

[17] B Cleveland, Robert and S Cleveland, William and E McRae, Jean and Terpenning, Irma. (1990). *STL: A Seasonal-Trend Decomposition Procedure Based on Loess.* Journal of Official Statistics. 6. 3-33.

[18] Kunsch, Hans R. (1989). *The Jackknife and the Bootstrap for General Stationary Observations.* The Annals of Statistics. Volume 17, Number 3 (1989), 1217-1241.

[19] Robert H. Shumway and David S. Stoffer. (2011). *Time Series Analysis and its Applications (Springer Texts in Statistics).* Springer-Verlag, Berlin, Heidelberg.

[20] Makridakis, S., and Hibon, M. (2000). *The M3-Competition: results, conclusions and implications.*