ISSN 1440-771X

Department of Econometrics and Business Statistics

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

Visualising Forecasting Algorithm Performance using Time Series Instance Spaces

Yanfei Kang, Rob J Hyndman and Kate Smith-Miles

June2016

Working Paper 10/16

Visualising Forecasting Algorithm Performance using Time Series Instance Spaces

Yanfei Kang
School of Statistics
Renmin University of China
Beijing, 100872, China.
Email: yanfei.kang@outlook.com

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

Kate Smith-Miles
School of Mathematical Sciences,
Monash University, VIC 3800
Australia.
Email: Kate.Smith-Miles@monash.edu

6 June 2016

JEL classification: C52,C53,C55

Visualising Forecasting Algorithm Performance using Time Series Instance Spaces

Abstract

It is common practice to evaluate the strength of forecasting methods using collections of well-studied time series datasets, such as the M3 data. But how diverse are these time series, how challenging, and do they enable us to study the unique strengths and weaknesses of different forecasting methods? In this paper we propose a visualisation method for a collection of time series that enables a time series to be represented as a point in a 2-dimensional instance space. The effectiveness of different forecasting methods can be visualised easily across this space, and the diversity of the time series in an existing collection can be assessed. Noting that the M3 dataset is not as diverse as we would ideally like, this paper also proposes a method for generating new time series with controllable characteristics to fill in and spread out the instance space, making generalisations of forecasting method performance as robust as possible.

Keywords: M3-Competition; Time series visualisation; Time series generation; Forecasting algorithm comparison.

1 Introduction

The M3 data (Makridakis and Hibon, 2000) are widely used for testing the performance of new forecasting algorithms. These 3003 series have become the de facto standard test base in forecasting research. When any new univariate forecasting method is proposed, if it does not perform well on the M3 data compared to the results on other published algorithms, it is unlikely to receive any further attention or adoption.

We see several problems with this approach. The M3 data were a convenience sample collected from several discipline areas, namely demography, finance, business and economics. All data were positive, with series lengths ranging from 14 to 126, and observed annually, quarterly or monthly (apart from 174 "other" series whose frequency of observation was not provided). So testing algorithms on this data set will tend to favour forecasting methods that work well with data from these domains, of these lengths, and of these frequencies. Further, there is no

guarantee that the data are in any sense "representative" of the types of data found within those domains, as noted in the subsequent discussion of the M3 data (Ord, 2001). Finally, given that 15 years have elapsed since the M3 results were published, it is highly likely that the patterns seen within typical time series have changed over time, even within the collection constraints of the competition.

In the published M3 results, no attempt was made to study *why* some methods did better on some series compared to other methods. Was it just chance, or are there particular features of some time series that make them particularly amenable to being forecast by one method compared to another? In discussing the M3 results, Lawrence, (2001) wrote

> What is needed now is analysis to determine what are the specific time series characteristics for which each technique is generally best and also what are the time series characteristics for which it does not really matter which technique (or set of techniques) is chosen.

The M3 time series might share some specific characteristics, and thus conclusions based on these data might only hold for other series with those particular characteristics (Clements and Hendry, 2001).

Similar comments could apply to other time series collections. How do we know that any time series collection covers the range of possible time series patterns, or is somehow representative of the types of data we are designing forecasting methods to handle?

This paper proposes a new approach to answer some of these questions. The methodology is an adaptation of previous work of the authors applied to objective assessment of combinatorial optimisation algorithms (Smith-Miles, Baatar, et al., 2014), and generation of new test instances (Smith-Miles and Bowly, 2015), extended here for the first time to the time series and forecasting domains.

Our approach involves computing "features" on each time series. For example, we measure the autocorrelation at lag 1, the seasonal period, and the spectral entropy. These, and several other features, are all numerical quantities computed on each time series. Then we study the "feature space" of the collection of time series. By studying the feature space rather than the raw time series, we convert the problem from temporal data to static data. We also convert a large collection of time series of different lengths, to a data set comprising a small number of features for each series. Thus, each time series is represented as a point in a high dimensional feature

space, which can be reduced using dimension reduction techniques to a 2-dimensional instance space.

The idea of characterising a time series as a feature vector is not new, and has been used for classifying time series (Nanopoulos, Alcock, and Manolopoulos, 2001), clustering time series (Wang, Smith, and Hyndman, 2006), and for identifying outlying or anomalous time series (Hyndman, Wang, and Laptev, 2015). In this paper, we generate a 2-dimensional instance space of time series and use it to explore the properties of a given collection of time series: in this case, the M3 dataset. We study the distribution of features across the space to understand the similarity and differences between the time series, and to assess the diversity of the collection. We also study whether the location in the instance space, given by the features, is predictive of forecasting method performance. Finally, we can identify gaps in the instance space, and develop new methods to generate time series with controllable features by evolving time series to lie at target locations.

The nature and number of features to be used depends on the problem context and their discriminatory quality (Nanopoulos, Alcock, and Manolopoulos, 2001). We have suggested a small number of features that we think are useful for studying the M3 data. However, there may be many other features that would also be useful and which would provide different information from those we have chosen. For other collections of time series, other sets of features will need to be used. For example, Hyndman, Wang, and Laptev, (2015) use a set of 18 features designed to identify anomalous time series of web traffic.

In Section 2, we define the features we have chosen for the M3 data, and show how principal components analysis can be used to reduce the dimension of the feature space to enable visualization of the space of the time series via a 2-dimensional instance space.

The scatterplot of the first two principal components suggests that there may be regions of the feature space that are not well-covered by the M3 data. So in Section 3 we use a genetic algorithm to generate new time series designed to fill "gaps" in the feature space of the M3 data. In this sense, we are contributing a broader and more diverse collection of M3-like time series to test the performance of forecasting methods.

2 Time series features

Depending on the research goals and domains, a variety of time series features have been developed to characterising time series in previous studies (e.g., Deng et al., 2013; Fulcher and

Jones, 2014; Kang, Belušić, and Smith-Miles, 2014, 2015; Mörchen, 2003; Nanopoulos, Alcock, and Manolopoulos, 2001; Wang, Smith, and Hyndman, 2006). In this paper, we consider six features, selected because we believe they provide useful information about the M3 data.

The forecasting methods that did best on the M3 data were those that explicitly modelled trend and seasonal components in the data (Makridakis and Hibon, 2000). So we have selected methods that measure those characteristics. In addition, we have included a measure of "forecastability" suggested by Goerg, (2013), and a measure of variance-stability based on a Box-Cox transformation.

In the following descriptions, our time series is denoted by $\{x_1,\dots,x_n\}$, observed at times $1,\dots,n$.

Spectral entropy F_1

Entropy-based measures have been widely used in nonlinear analysis to assess the complexity of signals (e.g., Bandt and Pompe, 2002; Fadlallah et al., 2013; Zaccarelli et al., 2013) and to measure the "forecastability" of a time series (Garland, James, and Bradley, 2014; Goerg, 2013; Maasoumi and Racine, 2002). We use the spectral entropy measure included in the R package **ForeCA** (Goerg, 2013, 2016), which is an estimate of the Shannon entropy of the spectral density $f_x(\lambda)$ of a stationary process x_t:

$$F_1 = -\int_{-\pi}^{\pi} \hat{f}_x(\lambda)\log \hat{f}_x(\lambda)d\lambda,$$

where $\hat{f}_x(\lambda)$ is an estimate of the spectrum of the time series. Because $f_x(\lambda)$ describes the importance of frequency λ within the period domain of x_t, the Shannon entropy represents the relative contribution of different frequencies. A relatively small value of F_1 suggests that $\{x_t\}$ contains more signal and is more forecastable. On the other hand, a relatively large value of F_1 indicates more uncertainty about the future, and hence that the time series is harder to forecast.

Strength of trend F_2

A trend exists when a long-term change occurs in the mean level of a time series (Hyndman and Athanasopoulos, 2014). To measure the strength of trend, we first decompose a time series x_t into trend, season and remainder using an STL decompostion (Cleveland et al., 1990): $x_t = S_t + T_t + R_t$. The strength of trend of the time series is then measured by comparing the variances of the de-trended and de-seasonalized series R_t and the

de-seasonised series $x_t - S_t$ (Wang, Smith, and Hyndman, 2006):

$$F_2 = 1 - \frac{\text{var}(R_t)}{\text{var}(x_t - S_t)}.$$

Strength of seasonality F_3

A seasonal pattern exists when a time series is influenced by seasonal factors, such as the quarter or month of the year. Similar to the measure of trend, the strength of seasonality in x_t can be estimated by comparing the variances of the de-trended and de-seasonalized series R_t and the de-trended series $Y_t - T_t$ (Wang, Smith, and Hyndman, 2006):

$$F_3 = 1 - \frac{\text{var}(R_t)}{\text{var}(x_t - T_t)}.$$

Seasonal period F_4

The seasonal period is an important feature since it explains the length of the periodic patterns in a time series. Fortunately, the period information is known for almost all the M3 data: $F_4 = 4$ for quarterly data, $F_4 = 12$ for monthly data, and $F_4 = 1$ for annual data. For the small number of "other" time series, we also set $F_4 = 1$.

First order autocorrelation F_5

First order autocorrelation measures the linear relationship between a time series x_t and the one-step lagged series x_{t-1}:

$$F_5 = \text{Corr}(x_t, x_{t-1}).$$

A higher absolute value of F_5 means that future values of x_t are more dependent on the past value, which, to some extent, indicates the predictability of a time series.

Optimal Box-Cox transformation parameter F_6

A transformation can be useful when the variance of a series changes with its level. A popular family of transformations are the "Box-Cox transformations" (Box and Cox, 1964), defined as

$$w_t = \begin{cases} \log(x_t), & \text{if } \lambda = 0, \\ (x_t^\lambda - 1)/\lambda, & \text{otherwise.} \end{cases}$$

A good value of λ is one which makes the variation of a series approximately constant across the whole series.

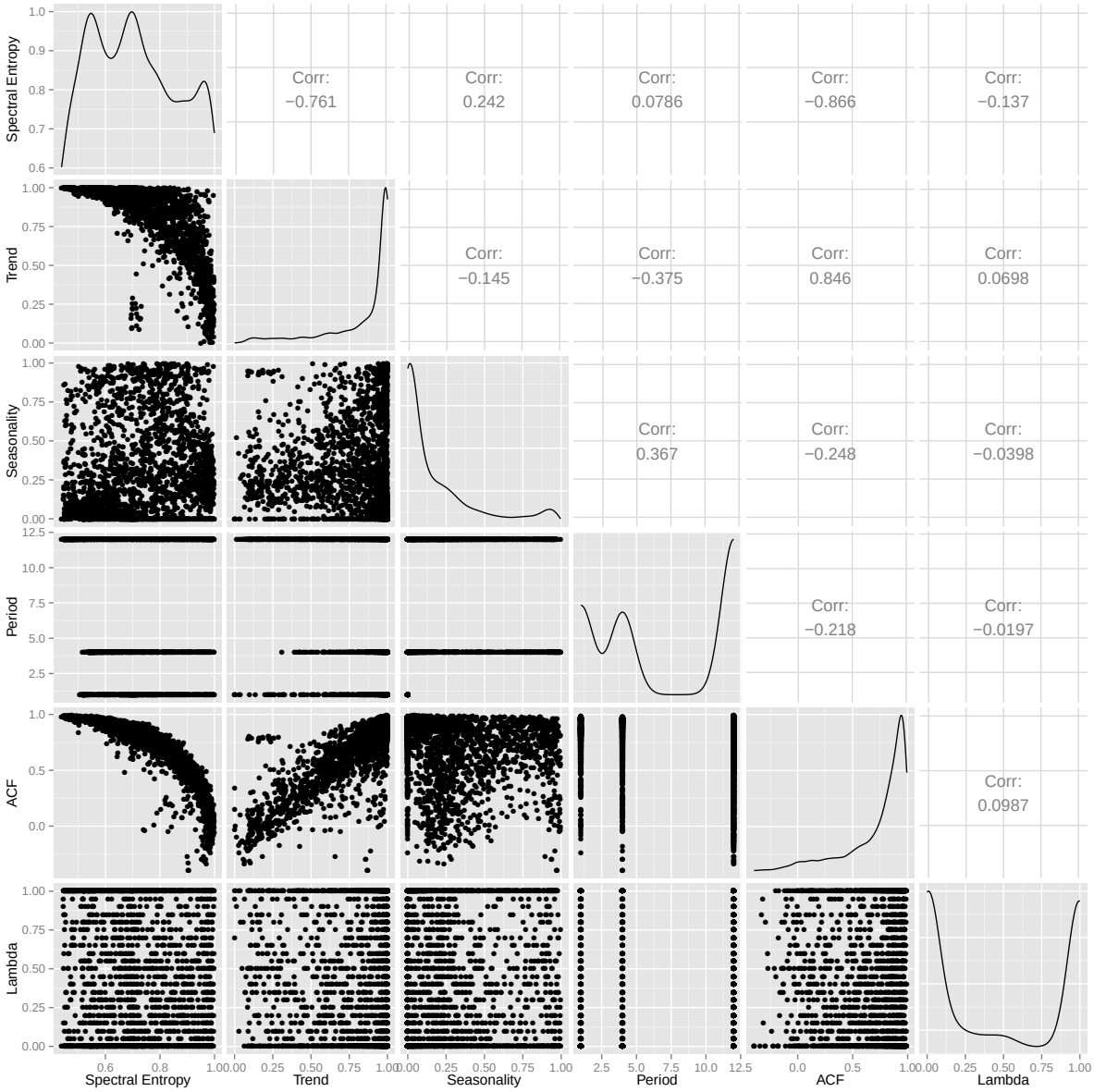

Figure 1: *Pairwise plot of M3 time series features.*

We choose $\lambda \in (0,1)$ to maximise the profile log likelihood of a linear model fitted to X_t. For non-seasonal data, a linear time trend is fitted while for seasonal data, a linear time trend with seasonal dummy variables is used. This value measures the degree of change of variation in the data. The value of λ which maximises the profile log-likelihood is denoted as F_6.

These six features enable any time series, of any length, to be summarised as a feature vector $\boldsymbol{F} = (F_1, F_2, F_3, F_4, F_5, F_6)'$. Figure 1 shows pairwise scatterplots of the six features in the lower

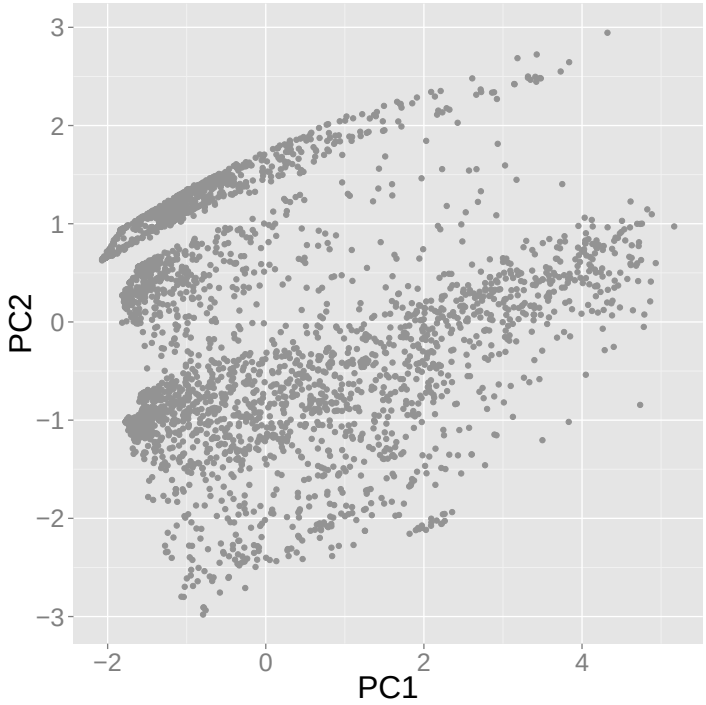

Figure 2: *Instance space of M3 time series.*

half, and the correlations between the features in the upper half. The diagonal shows kernel density estimates for each feature.

Once the set of features are computed for all time series in the collection, we use the first two principal components to project them onto a 2-dimensional space to allow easy visualisation of the data. The two axes both show linear combinations of the six features. For the M3 data, the first two principal components with the largest eigenvalues retain 67.8% of the variation in the data, and are algebraically expressed as:

$$\begin{bmatrix}\text{PC1}\\ \text{PC2}\end{bmatrix} = \begin{bmatrix}0.530 & -0.537 & 0.228 & 0.234 & -0.555 & -0.122\\ 0.252 & -0.160 & -0.656 & -0.655 & -0.180 & 0.136\end{bmatrix}\boldsymbol{F}. \tag{1}$$

The first axis increases with spectral entropy and decreases with trend and first order autocorrelation. The second axis decreases with period and seasonality. We rotate and project all the series into the coordinate system given by these first two axes, generating the time series instance space shown in Figure 2. The projection results in some loss of information, but does not alter the topology of instance similarity (Goodhill and Sejnowski, 1996). The two-dimensional instance space enables easier discovery of interesting time series structures and we consider it to be a

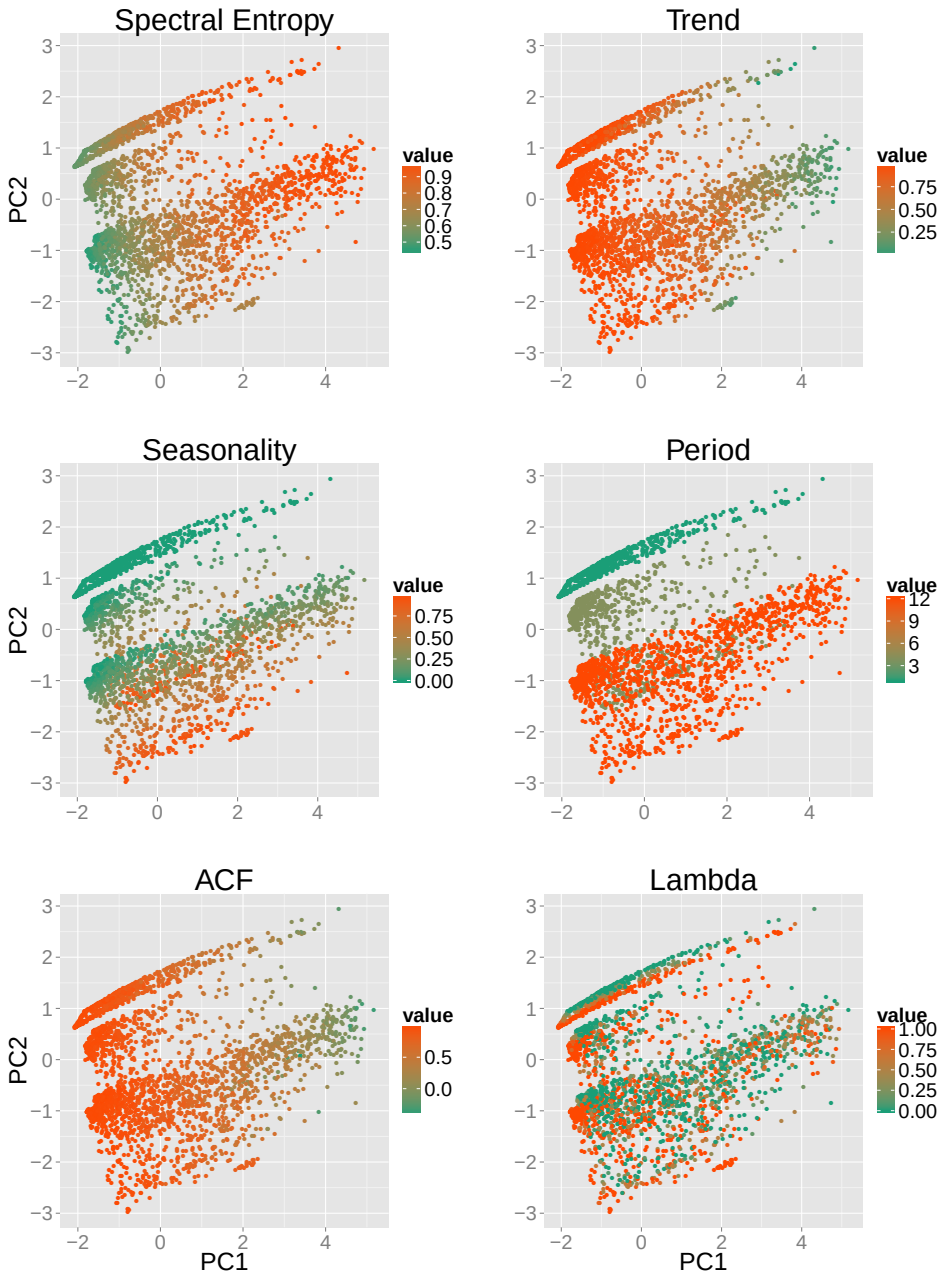

Figure 3: *Distribution of features across the instance space.*

useful representation of the instances. Instances that are close to each other in this instance space have similar values for the six features.

The properties of instances are demonstrated in their feature distributions across the instance space, shown in Figure 3. Towards the left bottom corner, time series become easier to forecast, with lower spectral entropy values, higher trend and higher first order autocorrelation. We also see clearly where yearly, quarterly and monthly data lie in the instance space and how their degree of seasonality varies from the distribution of their seasonal period features.

Interestingly, the current time series in the M3 dataset have failed to fill the entire instance space. There are more instances in the left part of the space, with sparser areas towards the right side. Is it possible to generate more instances to fill and extend the whole space? Can we generate a more diverse and useful set of time series than the current M3-competition series? In the following sections, we present a method to generate new time series instances with controllable features in targeted locations in the instance space to provide some insights to this question.

3 New time series generation in instance space

To enlarge the diversity and the evenness of the instance space, we evolve new instances in the locations where target points are set. Once a target point is set, our goal is to evolve a new time series instance which is as close as possible to the target point when projected to the two-dimensional instance space. The process relies on a genetic algorithm, which starts from randomly selected initial time series and uses a combination of selection, crossover and mutation to evolve time series that project to the target point as closely as possible. The population is composed of a number of individual time series, each of them evaluated by a fitness function. Aiming to reach the location of a target point, we require a fitness function for a time series to increase while its Euclidean distance to the target point decreases, for close analogy with the genetic algorithm's maximisation of a fitness function. Only the fittest individuals reproduce, passing their characteristics to their offspring. Initial populations are then improved through repetitive application of the selection, crossover and mutation operators until the final population is achieved with maximised fitness.

For each given target point T_i, $i = 1, 2, \ldots, N_t$, where N_t is the number of targets to be set, we first generate an initial population of time series. The number of time series in the population is given by the population size N_p. The initial population can be generated randomly, allowing the entire range of possible solutions in the search space. It can also be seeded with time series

in the neighbourhood of the target point to accelerate convergence. Then the following steps enable us to evolve new time series:

1. Calculate the feature vector for each time series $j \in \{1, 2, \ldots, N_p\}$ in the current population. Project the feature vector into two-dimensional instance space using Eq. (1). Denote PC_j as the two-dimensional projection of instance j.

2. Calculate the fitness of each member in the current population:

$$\text{Fitness}(j) = -\sqrt{(|\text{PC}_j - T_i|^2)}.$$

3. Evolve the next generation based on the crossover, mutation and the survival of the fittest individual premise to improve the average fitness of each generation.

These steps are iterated until the whole process meets one of the following convergence criteria:

- The maximum of the fitness function is at least -0.01.
- The number of iterations reaches 3000.
- The number of consecutive generations without any improvement in the best fitness value reaches 200.

From the final population, we select the instance that is closest to the target point (i.e., has the largest fitness value) as the evolved time series for the corresponding target. The new time series instance generation process described above is implemented using the R package **GA** (Scrucca, 2012).

With the instance space in Figure 2, we can observe the ranges of the first two principal component axes, PC1 and PC2. Now we evolve yearly, quarterly and monthly series separately according to the procedure above. Our target points are set to be a 32×32 grid with 1024 points, which are bounded within one unit wider than the upper and lower bounds of PC1 and PC2. This allows us to evolve new series which would lie outside the boundaries of the current space and further find a more general boundary for the instance space. Then we use a genetic algorithm to generate 1024 yearly, quarterly and monthly time series that are previously unknown and evolved by maximising the fitness function so that the evolved series are as close as possible to the target points when projected to the two-dimensional space. Since the evolutionary process only generates time series with a certain length, we evolve yearly, quarterly and monthly time series with lengths 30, 60 and 120, respectively. This means we go through

the above evolution process $3 \times 1024 = 3072$ times to generate 1024 yearly series with length 30, 1024 quarterly series with length 60, and 1024 monthly series with length 120.

For the generation of 1024 yearly series, given a target point T_i, $i = 1, 2, \ldots, 1024$, we set the crossover probability to be 0.8 and mutation probability 0.4. The size of the initial population is set to be $N_p = 20$. These 20 initial series are selected from the M3 data but excluding any series whose distance is less than 0.3 from the target T_i to avoid full replications of M3 data. Specifically, we randomly select 10 yearly time series from those whose distances to T_i are greater than 0.3, as well as the 10 closest time series to T_i other than those closer than 0.3 in distance. For each selected series, if its length is greater than the targeted length (which is 30 for yearly data), it is truncated to 30; otherwise, it is reflected to create a series of the target length.

Initial populations for the evolution of quarterly and monthly series are similar so that they are seeded in areas where optimal solutions are likely to be found. Their lengths are truncated or reflected to 60 for quarterly data and 120 for monthly data. Figure 4 shows the 1024 target points we set and where the evolved yearly, quarterly and monthly time series lie in the two-dimensional space.

The first thing to notice is that there are large parts of the target space where we have not been able to generate series. Although much larger margins are allowed when target points are set, they shrink to smaller well-defined regions for all the evolved yearly, quarterly and monthly series.

This suggests that there are natural boundaries for yearly, quarterly and monthly data within this 2-dimensional instance space, probably due to constraints on combinations of features. For example, Figure 1 suggests that it is impossible to have a series with very low spectral entropy and very low trend, and it is clearly impossible to have a series with both high seasonality and period 1. These kinds of constraints presumably lead to the apparent boundaries seen in Figure 4.

It is also apparent that away from the boundaries, the evolved series are more evenly distributed while the M3 data have higher density on the left side of the space. New series are evolved in the top right of the yearly and quarterly parts of the space, and the bottom right side of the monthly part.

As a validation procedure for the evolution process, we randomly selected three known time series from the yearly, quarterly and monthly series respectively, shown in the top panel of

Figure 5 as points A, B and C. Can we use the location of these known time series as target points, and then evolve new time series that lie near these targets? The bottom panel of Figure 5 shows the three target time series on the left and the corresponding evolved ones on the right. As expected, they are similar-looking in terms of their time series characteristics.

Extending this idea, we attempted to generate new time series that live at empty locations in the instance space. Setting three new target points, shown as D, E and F in the top panel of Figure 6, the bottom panel plot shows the three newly generated time series. They are not from M3 data, and have different characteristics from any of the existing M3 time series.

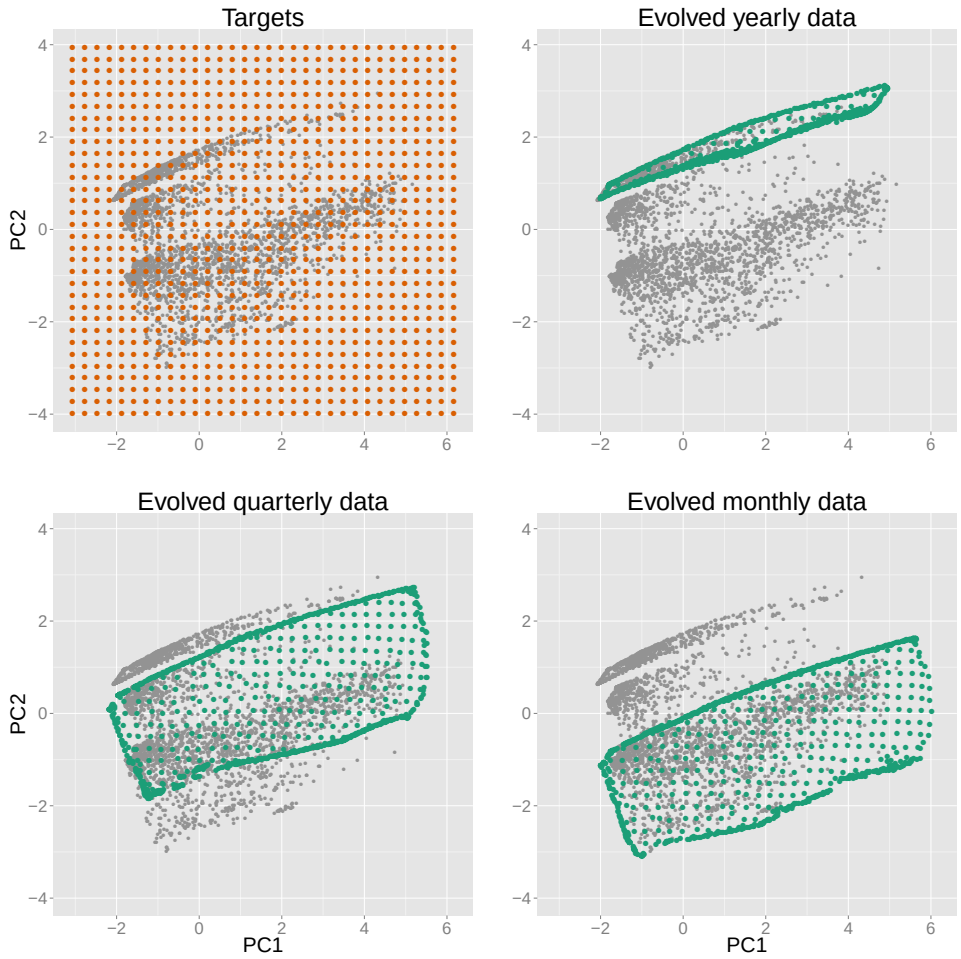

Figure 4: *Evolved new instances using genetic algorithm.*

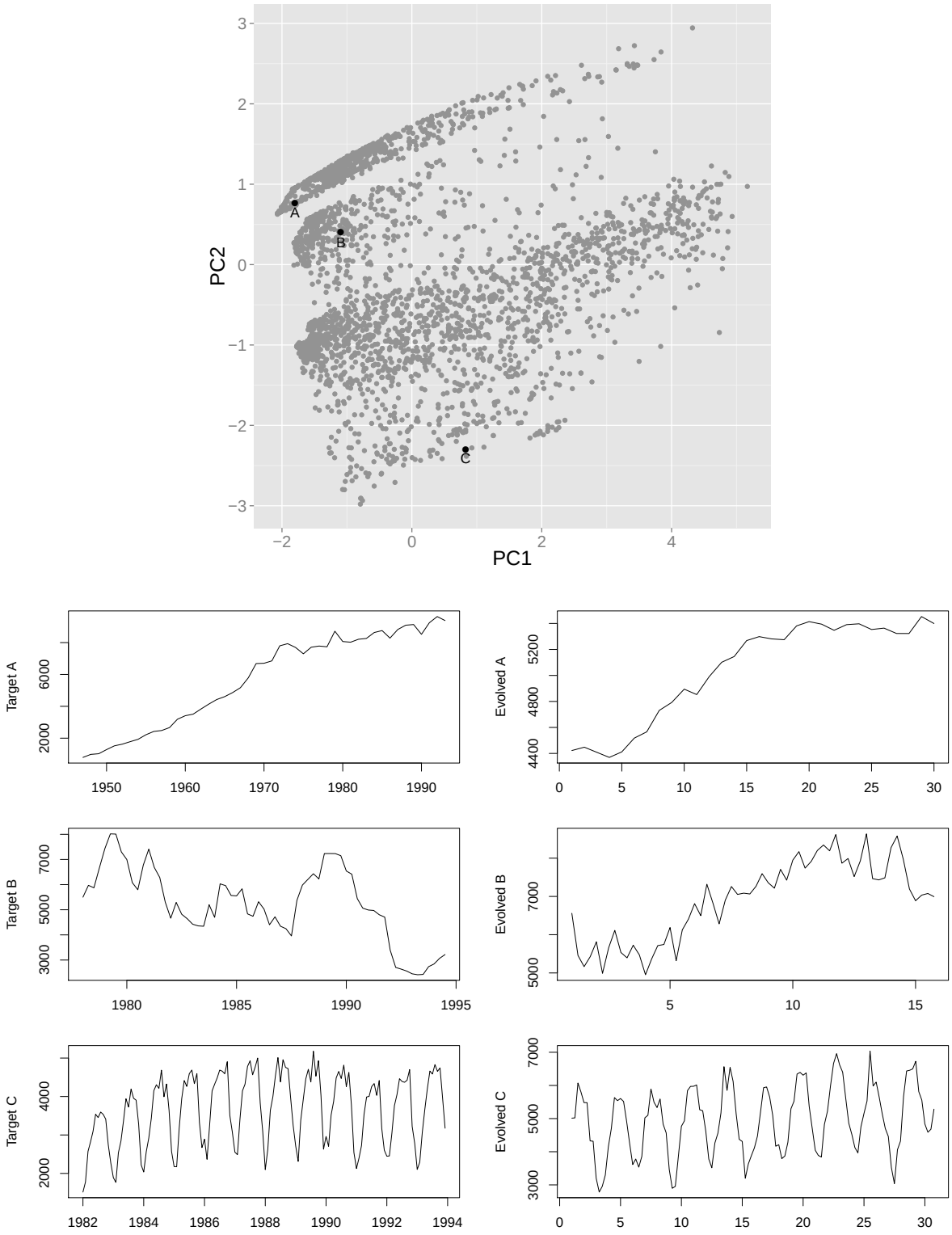

Figure 5: *Top panel: locations of the three targeted time series in the instance space; bottom panel: targeted series and evolved new series using genetic algorithm.*

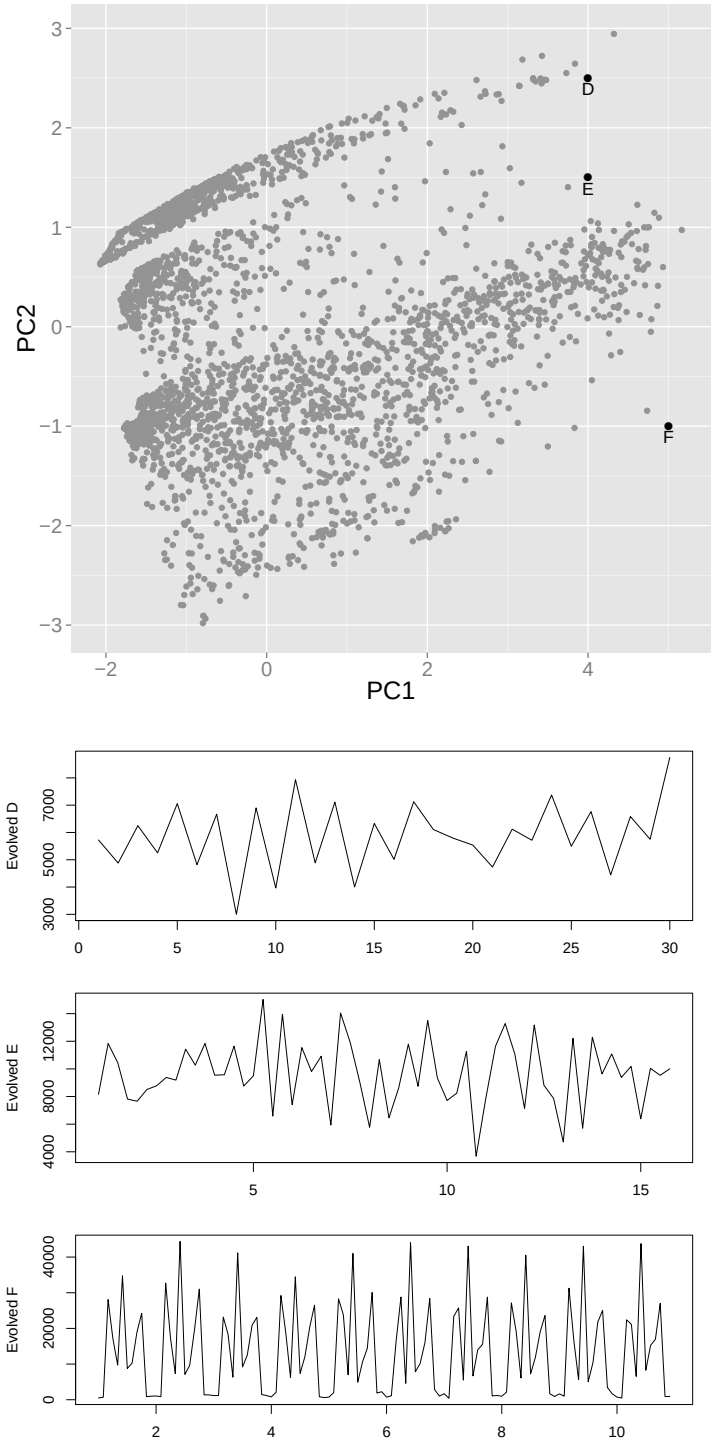

Figure 6: *Top panel: locations of the three new target points in the instance space; bottom panel: evolved new series using genetic algorithm.*

4 Comparison of time series forecasting methods in the instance space

The No-Free-Lunch theorem was proposed in Wolpert, (1996) for supervised machine learning and in Wolpert and Macready, (1997) for search and optimisation. It tells us that there is never likely to be a single method that fits all situations. Smith-Miles, Baatar, et al., (2014) proposed a method to compare and visualise the strengths and weaknesses of different graph colouring algorithms across an instance space. Similarly there is no time series forecasting method that will always perform best. Even for one particular time series, no one technique is consistently superior to others (Lawrence, 2001). Here we consider six general time series forecasting methods to show the potential of the instance space for algorithm performance visualisation. They are

- Naïve: using the most recent observation as the forecast for all future periods.
- Seasonal naïve: forecasts are equal to the most recent observation from the corresponding time of year. For yearly data, this is equivalent to the naïve method.
- The Theta method, which performed particularly well in the M3-Competition (Assimakopoulos and Nikolopoulos, 2000; Makridakis and Hibon, 2000). We apply the method directly to the time series without pre-processing the series using seasonal adjustment. Consequently, this is not expected to perform well for seasonal data.
- ETS: exponential smoothing state space modelling (Hyndman, Koehler, et al., 2002), and is widely used as a general forecasting algorithm for trended and seasonal time series.
- ARIMA: autoregressive integrated moving average models, as implemented in the automated algorithm of Hyndman and Khandakar, (2008).
- STL-AR: an AR model is fitted to the seasonally adjusted series obtained from a STL decomposition (Hyndman and Athanasopoulos, 2014), while the seasonal component is forecast using the seasonal naïve method. The two forecasts are summed to obtain forecasts of the original time series.

First we show how well these methods can forecast the M3 data. Table 1 shows MASE values (Hyndman and Koehler, 2006) for each method and each group of time series. The scaling factor used for the yearly and "other" data was the average in-sample MAE after applying a naïve forecasting method; the scaling factor used for the monthly and quarterly data was the

average in-sample MAE after applying a seasonal naïve forecasting method. All methods were estimated using the training data that were made available to the M3 competitors; the MASE values were computed using the hold-out (test) samples that were unavailable to the original M3 competitors. Both parts of each time series are available in the **Mcomp** package for R (Hyndman, 2013). On average, ETS almost always performed the best for each group of time series except for yearly data, where the Theta method performed slightly better.

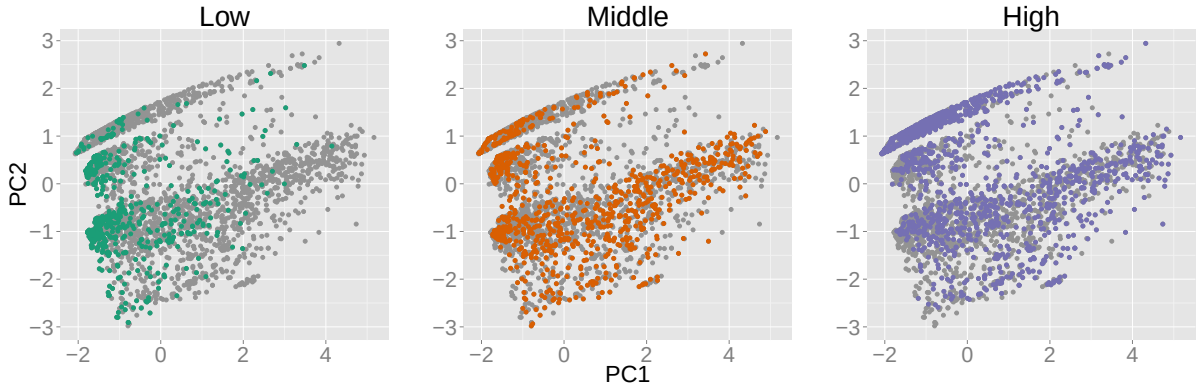

Figure 7: *Minimum MASE obtained from the forecasting algorithms for M3 series.*

For each series, we compute the minimum MASE value achieved from all six methods. Figure 7 highlights the areas where we can get low, middle, and high minimum MASE values (defined by the 20th and 50th percentiles), showing which parts of the instance space are easiest to forecast. As shown in the left panel, instances with low minimum MASE values mainly lie in the bottom and left of the yearly, quarterly and monthly data. From the middle panel, yearly and quarterly instances with middle minimum MASE will not reach into the right hand side of the space. The right panel tells us that quarterly and monthly series with high minimum MASE are mainly around the top right hand sides. These results are consistent with the distribution of spectral entropy in Figure 3, which is an indicator of forecastability.

Forecasting method	Yearly	Quarterly	Monthly	Other	All
Naïve	3.17	1.46	1.17	3.09	1.79
Seasonal naïve	3.17	1.43	1.15	3.09	1.76
Theta	**2.77**	1.30	1.02	2.27	1.54
ETS	2.86	**1.18**	**0.86**	**1.79**	**1.43**
ARIMA	2.96	1.19	0.88	1.83	1.46
STL-AR	2.95	1.91	1.27	1.94	1.83

Table 1: *MASE of the six methods on M3*

Method	Yearly	Quarterly	Monthly	All
Naïve	**1.93**	3.10	2.35	2.46
Seasonal naïve	1.93	1.40	1.21	**1.51**
Theta	2.09	3.02	2.17	2.43
ETS	2.67	**1.07**	**0.93**	1.56
ARIMA	2.46	1.13	0.94	**1.51**
STL-AR	2.44	1.35	1.08	1.62

Table 2: *MASE of the six methods on Evolved series*

Table 2 shows how well the forecasting methods perform on the evolved data. For quarterly and monthly data, ETS still performs the best, as in the two groups of M3 data. But for yearly data, the Naïve method performs the best. The ARIMA and Seasonal Naïve methods do the best overall and ETS comes the second, despite doing the best on average for the M3 data.

This highlights the idiosyncratic nature of the M3 data, and demonstrates that the particular collection of time series will affect any conclusions drawn. Because the evolved series are dense around the perimeter of the instance space, but relatively sparse in the interior of the space, the best performing methods are not the same as on the M3 data. It is worth remembering that the M3 data are not a representative sample of any larger population of time series, but were a convenience sample of the data available to Makridakis and Hibon. Presumably, the results could have been different again if another sample of time series had been selected instead.

We find the instance space a useful visualisation for understanding how the features of time series affect the performance of forecasting algorithms, since the location of instances is determined by their features. Does the location also determine the performance of a forecasting method and the relative performance of different forecasting methods?

Figure 8 shows MASE values for each forecasting method and each time series of the M3 data. The MASE values were capped at 4 to visualise them more properly. The plots show which particular regions of the instance space were best forecast for each method. While there are no regions where one method always has low MASE values, and there are no regions where one method clearly dominates all other methods, the figure does suggest that there are regions of the instance space where some methods are best avoided. For example, the Naïve and Theta methods can not well forecast the bottom right hand sides of the monthly data, which have high seasonality as shown in Figure 3. The Seasonal Naïve and STL-AR perform worst on the left hand sides, which are the "easy" series shown in Figure 7.

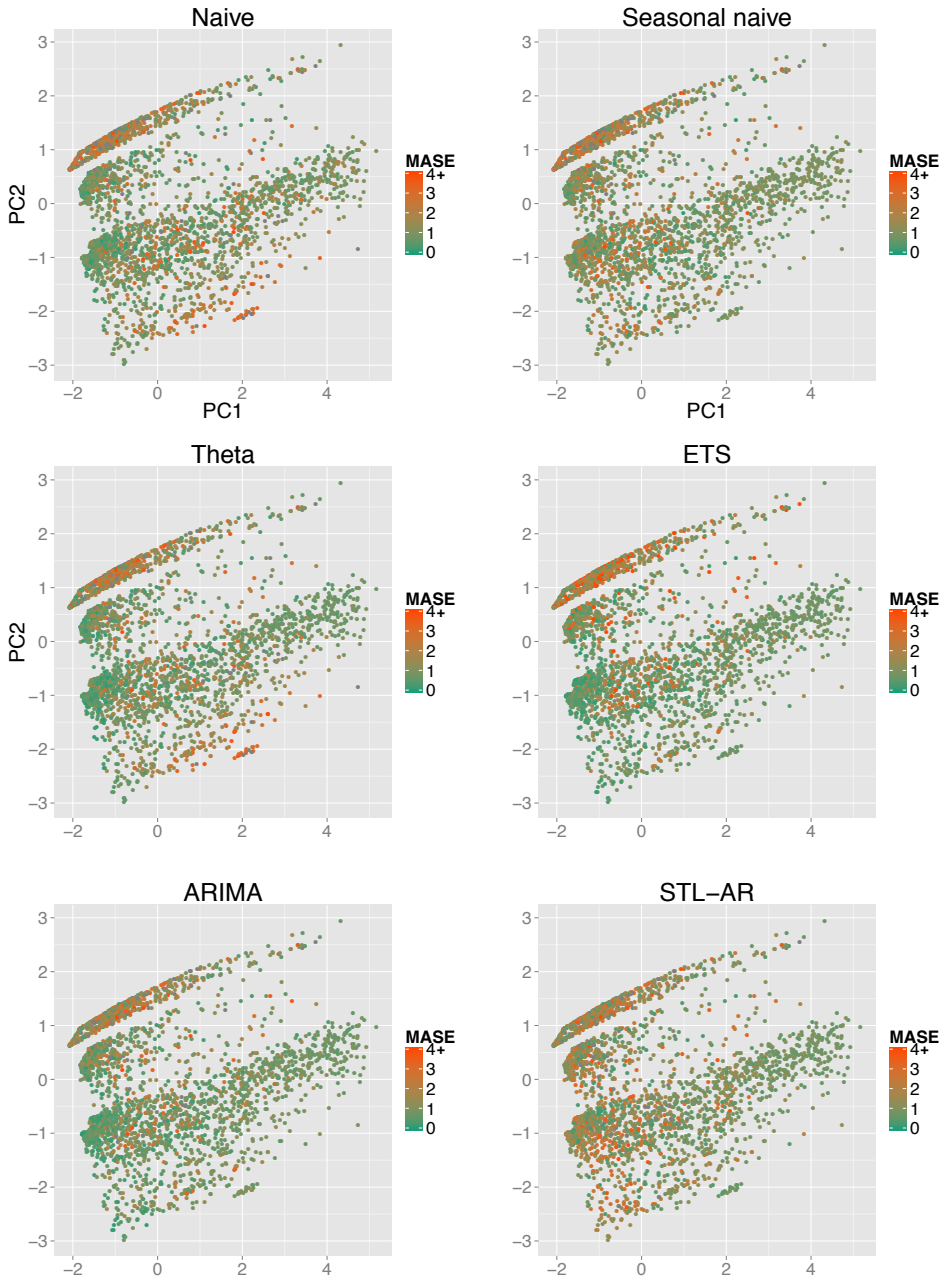

Figure 8: *The distribution of out-of-sample MASE values of the M3 data in the instance space for the six forecasting methods.*

The figure also suggests that the instance space could be used to identify appropriate methods for a given series. This could then lead to a meta-forecasting algorithm where the features of a time series are computed and used to select a suitable forecasting method without the need to test a large collection of methods on the given time series. In a forecasting environment where

a very large number of forecasts have to be computed quickly, this should lead to substantial efficiency improvements. We will leave the specifics of such an algorithm to a later paper.

5 Conclusions

We have represented a collection of time series as points in a feature space. This has allowed us to propose several interesting analysis tools that provide insight into large collections of time series.

First, we can identify unusual time series which have very different combinations of features compared to other time series in the collection. We can also see clustering and other structures within the feature space showing possible sub-groups of time series, and regions where there are many similar time series.

Second, we have proposed an algorithm to generate new time series with controllable characteristics. This can be used to generate new time series with similar characteristics to the collection of existing time series, or to generate new time series which have specific locations in the feature space including regions where we have no existing time series. This allows us to explore the complete feature space of possible time series. It also provides a way of testing new methods without over-fitting models on the existing collection of time series.

Third, we have used this algorithm to show that conclusions based on the M3 competition data will not necessarily hold for other collections of time series with different distributions in the feature space.

Finally, we have shown that different forecasting methods perform better in some regions of the feature space than other methods. This allows the possibility of developing meta-forecasting algorithms which choose a specific forecasting method based on the location of a time series in the instance space.

While we have used the M3 data as a vehicle of illustration, our proposed approach is general and can be applied to any large collection of time series. Further, the specific features we have chosen here are only illustrative, and appropriate features that measure characteristics of interest will depend on the particular nature of the time series collection and the purpose of the subsequent analysis.

References

Assimakopoulos, V and K Nikolopoulos (2000). The theta model: a decomposition approach to forecasting. *International Journal of Forecasting* **16**(4), 521–530.

Bandt, C and B Pompe (2002). Permutation Entropy: A Natural Complexity Measure for Time Series. *Physical Review Letters* **88** (17), 174102.

Box, GEP and DR Cox (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B* **26**(2), 211–252.

Clements, MP and DF Hendry (2001). Explaining the results of the M3 forecasting competition. *International Journal of Forecasting* **17**, 550–554.

Cleveland, RB, WS Cleveland, JE McRae, and I Terpenning (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* **6**(1), 3–73.

Deng, H, G Runger, E Tuv, and M Vladimir (2013). A time series forest for classification and feature extraction. *Information Sciences* **239**, 142–153.

Fadlallah, B, B Chen, A Keil, and J Príncipe (2013). Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information. *Physical Review E* **87** (2), 022911.

Fulcher, B and N Jones (2014). Highly Comparative Feature-Based Time-Series Classification. *IEEE Transactions on Knowledge and Data Engineering* **26**(12), 3026–3037.

Garland, J, R James, and E Bradley (2014). Model-free quantification of time-series predictability. *Physical Review E* **90**(5), 052910.

Goerg, GM (2013). Forecastable Component Analysis. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. Ed. by S Dasgupta and D Mcallester. Vol. 28. JMLR Workshop and Conference Proceedings, pp.64–72.

Goerg, GM (2016). *ForeCA: An R package for Forecastable Component Analysis*. R package version 0.2.4. cran.r-project.org/package=ForeCA.

Goodhill, GJ and TJ Sejnowski (1996). Quantifying neighbourhood preservation in topographic mappings. In: *Proceedings of the 3rd Joint Symposium on Neural Computation*. Vol. 6, pp.61–82.

Hyndman, RJ (2013). *Mcomp: Data from the M-competitions*. R package version 2.05. cran.r-project.org/package=mcomp.

Hyndman, RJ and G Athanasopoulos (2014). *Forecasting: principles and practice*. Melbourne, Australia: OTexts. http://OTexts.org/fpp.

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

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

Hyndman, RJ, AB Koehler, RD Snyder, and 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, and N Laptev (2015). Large-scale unusual time series detection. In: *Proceedings of the IEEE International Conference on Data Mining*. 14–17 November 2015. Atlantic City, NJ, USA. http://robjhyndman.com/working-papers/cikm2015/.

Kang, Y, D Belušić, and K Smith-Miles (2014). Detecting and Classifying Events in Noisy Time Series. *Journal of the Atmospheric Sciences* **71**(3), 1090–1104.

Kang, Y, D Belušić, and K Smith-Miles (2015). Classes of Structures in the Stable Atmospheric Boundary Layer. *Quarterly Journal of the Royal Meteorological Society* **141**(691), 2057–2069.

Lawrence, M (2001). Commentaries on the M3-Competition. Why another study? *International Journal of Forecasting* **17**, 574–575.

Maasoumi, E and J Racine (2002). Entropy and predictability of stock market returns. *Journal of Econometrics* **107**(1–2), 291–312.

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

Mörchen, F (2003). *Time series feature extraction for data mining using DWT and DFT*. Tech. rep. 33. Department of Mathematics and Computer Science, Philips-University Marburg.

Nanopoulos, A, R Alcock, and Y Manolopoulos (2001). Feature-based classification of time-series data. *International Journal of Computer Research* **10**(3).

Ord, K (2001). Commentaries on the M3-Competition. An introduction, some comments and a scorecard. *International Journal of Forecasting* **17**, 537–541.

Scrucca, L (2012). GA: a package for genetic algorithms in R. *Journal of Statistical Software* **53**(4).

Smith-Miles, K, D Baatar, B Wreford, and R Lewis (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research* **45**, 12–24.

Smith-Miles, K and S Bowly (2015). Generating New Test Instances by Evolving in Instance Space. *Computers & Operations Research* **63**, 102–113.

Wang, X, KA Smith, and RJ Hyndman (2006). Characteristic-Based Clustering for Time Series Data. *Data Mining and Knowledge Discovery* **13**(3), 335–364.

Wolpert, DH (1996). The lack of a priori distinctions between learning algorithms. *Neural computation* **8**(7), 1341–1390.

Wolpert, DH and WG Macready (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Zaccarelli, N, BL Li, I Petrosillo, and G Zurlini (2013). Order and disorder in ecological time-series: Introducing normalized spectral entropy. *Ecological Indicators* **28**, 22–30.