International Journal of Forecasting 33 (2017) 345–358

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Visualising forecasting algorithm performance using time series instance spaces

Yanfei Kang [a,*], Rob J. Hyndman [b], Kate Smith-Miles [c]

[a] School of Economics and Management, Beihang University, Beijing, 100191, China
[b] Department of Econometrics and Business Statistics, Monash University, Clayton VIC 3800, Australia
[c] School of Mathematical Sciences, Monash University, Clayton VIC 3800, Australia

ARTICLE INFO

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

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. The question is, though, how diverse and challenging are these time series, and do they enable us to study the unique strengths and weaknesses of different forecasting methods? This paper proposes a visualisation method for collections of time series that enables a time series to be represented as a point in a two-dimensional instance space. The effectiveness of different forecasting methods across this space is easy to visualise, and the diversity of the time series in an existing collection can be assessed. Noting that the diversity of the M3 dataset has been questioned, this paper also proposes a method for generating new time series with controllable characteristics in order to fill in and spread out the instance space, making our generalisations of forecasting method performances as robust as possible.

1. Introduction

The M3 data (Makridakis & Hibon, 2000) are used widely for testing the performances of new forecasting algorithms. These 3003 series have become the de facto standard test base in forecasting research. When a new univariate forecasting method is proposed, it is unlikely to receive any further attention or be adopted unless it performs better on the M3 data than other published algorithms.

We see several problems with this approach. The M3 dataset was a convenience sample that was collected from several disciplines, namely demography, finance, business and economics. All of the data were positive, with series lengths ranging from 14 to 126, and were observed annually, quarterly or monthly (apart from 174 "other" series, whose frequencies of observation were not provided). Methods that work well on this data set may overfit data with similar data structures. Thus, testing algorithms on this data set will tend to favour forecasting methods that work well with data from these domains, and of these lengths and frequencies. Furthermore, there is no guarantee that the series will be in any way "representative" of the types of data that are found within those domains, as is 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 will have changed over time, even within the collection constraints of the competition.

There has been no attempt in the published M3 results to study *why* some methods perform better on certain series than other methods. Is it just chance, or do some time

* Corresponding author.
E-mail addresses: yanfei.kang@outlook.com (Y. Kang), Rob.Hyndman@monash.edu (R.J. Hyndman), Kate.Smith-Miles@monash.edu (K. Smith-Miles).

http://dx.doi.org/10.1016/j.ijforecast.2016.09.004
0169-2070/

series have particular features that make them particularly amenable to being forecast by one method rather than 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.

Given that the M3 time series might share some specific characteristics, conclusions based on these data might only hold for other series with these particular characteristics (Clements & Hendry, 2001).

Similar comments apply to other collections of time series. 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 that we are designing forecasting methods to handle?

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

Our approach involves computing the "features" of 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 that are computed on each time series, and we then study the "feature space" of the collection of time series. By studying the feature space rather than the raw time series, we convert the data from temporal to static. We also convert a large collection of time series of different lengths to a data set that comprises 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 to a two-dimensional instance space using dimension-reduction techniques.

The idea of characterising a time series as a feature vector is not new, and has been used for classifying time series (e.g., Fulcher & Jones, 2014; Fulcher, Little, & Jones, 2013; Nanopoulos, Alcock, & Manolopoulos, 2001), clustering time series (e.g., Fulcher et al., 2013; Wang, Smith, & Hyndman, 2006), and identifying outlying or anomalous time series (e.g., Hyndman, Wang, & Laptev, 2015). This paper generates a two-dimensional instance space of time series and uses 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 in order to obtain an understanding of the similarities and differences between the time series, and to assess the diversity of the collection. We also investigate whether the location in the instance space, given by the features, can predict forecasting method performances. Finally, we identify gaps in the instance space and develop new methods for generating time series with controllable features by evolving time series to lie at given target locations.

The nature and number of the features to be used depends on the problem context and their discriminatory quality (Nanopoulos et al., 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 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 et al. (2015) use a set of 18 features that are designed for the identification of anomalous time series of web traffic. Fulcher et al. (2013) and Fulcher and Jones (2014) use thousands of features for reduced representations of time series data and their analysis methods, thus extending our ability to capture nuanced characteristics of time series.

Section 2 defines the features that we have chosen for the M3 data and shows how principal components analysis can be used to reduce the dimensions of the feature space in order to enable a visualization of the space of the time series via a two-dimensional instance space.

The scatterplot of the first two principal components suggests that there may be regions of the feature space that are not covered well by the M3 data. Thus, Section 3 uses a genetic algorithm to generate new time series that are designed to fill the "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 for testing the performances of forecasting methods.

2. Time series features

Depending on the research goals and domains, previous studies have developed a variety of time series features for the characterisation of time series (e.g., Deng, Runger, Tuv, & Vladimir, 2013; Fulcher & Jones, 2014; Kang, Belušić, & Smith-Miles, 2014, 2015; Mörchen, 2003; Nanopoulos et al., 2001; Wang et al., 2006). This paper considers six features, which are selected because we believe that they provide useful information about the M3 data.

The forecasting methods that performed best on the M3 data were those that modelled the trend and seasonal components of the data explicitly (Makridakis & Hibon, 2000), so we have selected methods that measure those characteristics. In addition, we have also included a measure of "forecastability", as 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 $\{x_1, \ldots, x_n\}$, observed at times $1, \ldots, n$.

Spectral entropy F_1. Entropy-based measures have been used widely in non-linear analysis for assessing the complexity of signals (e.g., Bandt & Pompe, 2002; Fadlallah, Chen, Keil, & Príncipe, 2013; Zaccarelli, Li, Petrosillo, & Zurlini, 2013) and measuring the "forecastability" of a time series (Garland, James, & Bradley, 2014; Goerg, 2013; Maasoumi & Racine, 2002). We use the spectral entropy measure that is included in the R package **ForeCA** (Goerg, 2013, 2014), 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 contributions 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, which suggests that the time series is harder to forecast.

Strength of trend F_2. A trend exists when there is a long-term change in the mean level of a time series (Hyndman & Athanasopoulos, 2014). We measure the strength of the trend by first decomposing a time series x_t into the trend, season and remainder using an STL decomposition (Cleveland, Cleveland, McRae, & Terpenning, 1990): $x_t = S_t + T_t + R_t$. The strength of the trend of the time series is then measured by comparing the variances of the de-trended and de-seasonalised series R_t and the de-seasonalised series $x_t - S_t$ (Wang et al., 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. Similarly to the measure of the trend, the strength of the 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 et al., 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. When the period is unknown, it can be estimated from the data using, for example, the `findfrequency()` function from the forecast package in R (Hyndman, 2016), which removes any trend and finds the maximum of the spectral density from the best-fitting autoregressive model based on the AIC. The location of the maximum is then rounded to the nearest integer. However, this is not necessary for most time series, as the period is usually known. For the M3 data, the period is given by $F_4 = 4$ 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. The first order autocorrelation measures the linear relationship between a time series and the one-step lagged series. It is affected strongly by the trend and seasonality, so we compute the autocorrelations in the de-trended and de-seasonalized series $\{R_t\}$:

$$F_5 = \text{Corr}(R_t, R_{t-1}).$$

Table 1
Summary of the six features that are used to characterize a time series.

Feature	Description
F_1	Spectral entropy
F_2	Strength of trend
F_3	Strength of seasonality
F_4	Seasonal period
F_5	First order autocorrelation
F_6	Optimal Box–Cox transformation parameter

A higher absolute value of F_5 means that future values of R_t are more dependent on past values, which, to some extent, indicates the predictability of a time series after adjusting for the trend and seasonality.

Optimal Box–Cox transformation parameter F_6 A transformation can be useful when the variance of a series changes with its level. One popular family of transformations is the "Box–Cox transformations" (Box & Cox, 1964), which are 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 in a series approximately constant across the whole series. We choose $\lambda \in (0, 1)$ in order to maximise the profile log likelihood of a linear model fitted to x_t. A linear time trend is fitted for non-seasonal data, while a linear time trend with seasonal dummy variables is used for seasonal data. This value measures the degree of change of variation in the data. The value of λ that maximises the profile log-likelihood is denoted by F_6.

These six features enable any time series, of any length, to be summarised as a feature vector $\boldsymbol{F} = (F_1, F_2, F_3, F_4, F_5, F_6)'$. Their descriptions are summarised in Table 1. Fig. 1 shows pairwise scatterplots or conditional histograms of the six features in the lower half, and their correlations or conditional boxplots in the upper half. The diagonal shows either a bar plot or a kernel density estimate for each feature.

Once the set of features has been computed for each time series in the collection, we use a dimension reduction method to project them all onto a two-dimensional space in order to allow a easy visualisation of the data. For the sake of simplicity, this paper uses principal components analysis, although we note that other methods may also be appropriate, especially given the non-linearity between the trend and entropy that is seen in Fig. 1.

The two principal component axes both show linear combinations of the six raw features. For the M3 data, the first two principal components with the largest eigenvalues explain 60.20% of the variation in the data, and are described algebraically as:

$$\begin{bmatrix} \text{PC1} \\ \text{PC2} \end{bmatrix} = \begin{bmatrix} 0.614 & -0.588 & 0.321 & 0.258 & -0.292 & -0.150 \\ 0.210 & 0.000 & -0.307 & -0.687 & -0.608 & -0.114 \end{bmatrix} \boldsymbol{F}. \quad (1)$$

The horizontal axis (PC1) increases with the spectral entropy and decreases with the trend. The vertical axis

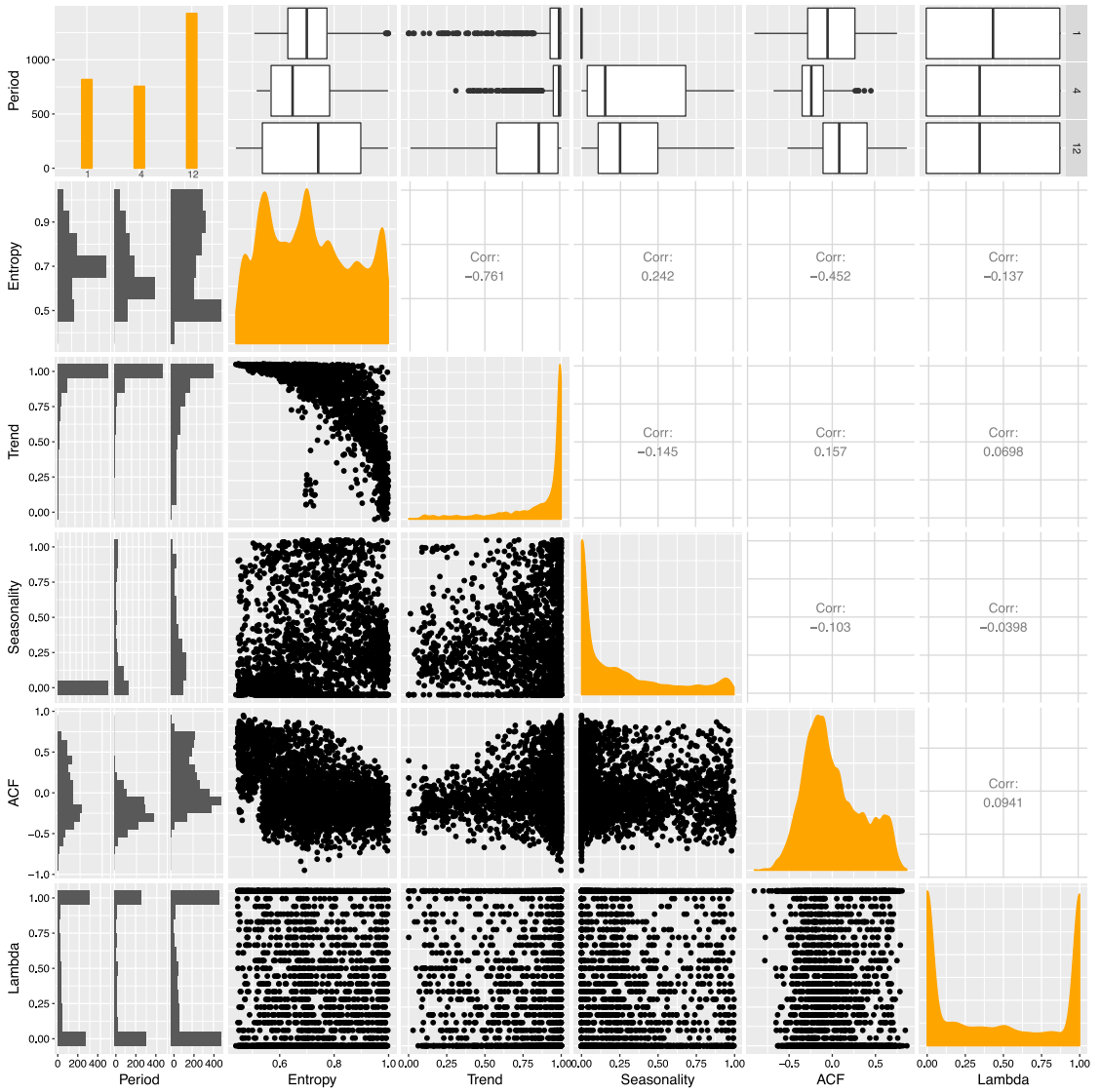

Fig. 1. Pairwise scatterplot of the six features calculated on the M3 time series. The period of a time series x_t in M3 may be 1, 4, or 12, depending on its periodic pattern. The entropy feature measures the forecastability of x_t. Trend and seasonality are the strength of the trend and seasonality components in x_t. ACF is the first order autocorrelation, calculated on the de-trended and de-seasonalized series of x_t. Lambda is the optimal Box–Cox transformation parameter, which is a measure of the variance-stability of x_t.

(PC2) is related negatively to period and ACF, while the seasonality increases with PC1 and decreases with PC2. The Box–Cox transformation parameter is not related strongly to either of the first two principal components.

We project each series onto the coordinate system given by these first two axes in order to generate the time series instance space shown in the top panel of Fig. 2, where we highlight 12 points, labelled a–l, that populate distinct parts of the instance space; the actual time series are shown in the bottom panel. It is shown that time series with visually similar properties do indeed populate distinct parts of the space. Instances that are close to each other in this instance space have similar values for the six features. The Pearson correlation coefficient between instance distances in six-dimensional space versus distances in two-dimensional space is 0.84. The projection causes some loss of information, but does not alter much of the topology of instance similarity. The two-dimensional instance space enables an easier discovery of interesting time series structures, and we consider it to be a useful representation of the instances. Note here that Fig. 2 demonstrates the existence of two 'clusters', since the discrete period feature leads to a separation of the dataset into monthly data in one cluster and the remaining data in the other cluster, making the space more explicit. Note also that the distributions of the trend and the seasonality are both highly skewed from the kernel density estimates in Fig. 1, and therefore their correlations with other features may be distorted when PCA is performed. The inclusion of more data or more features can change the appearance of the space.

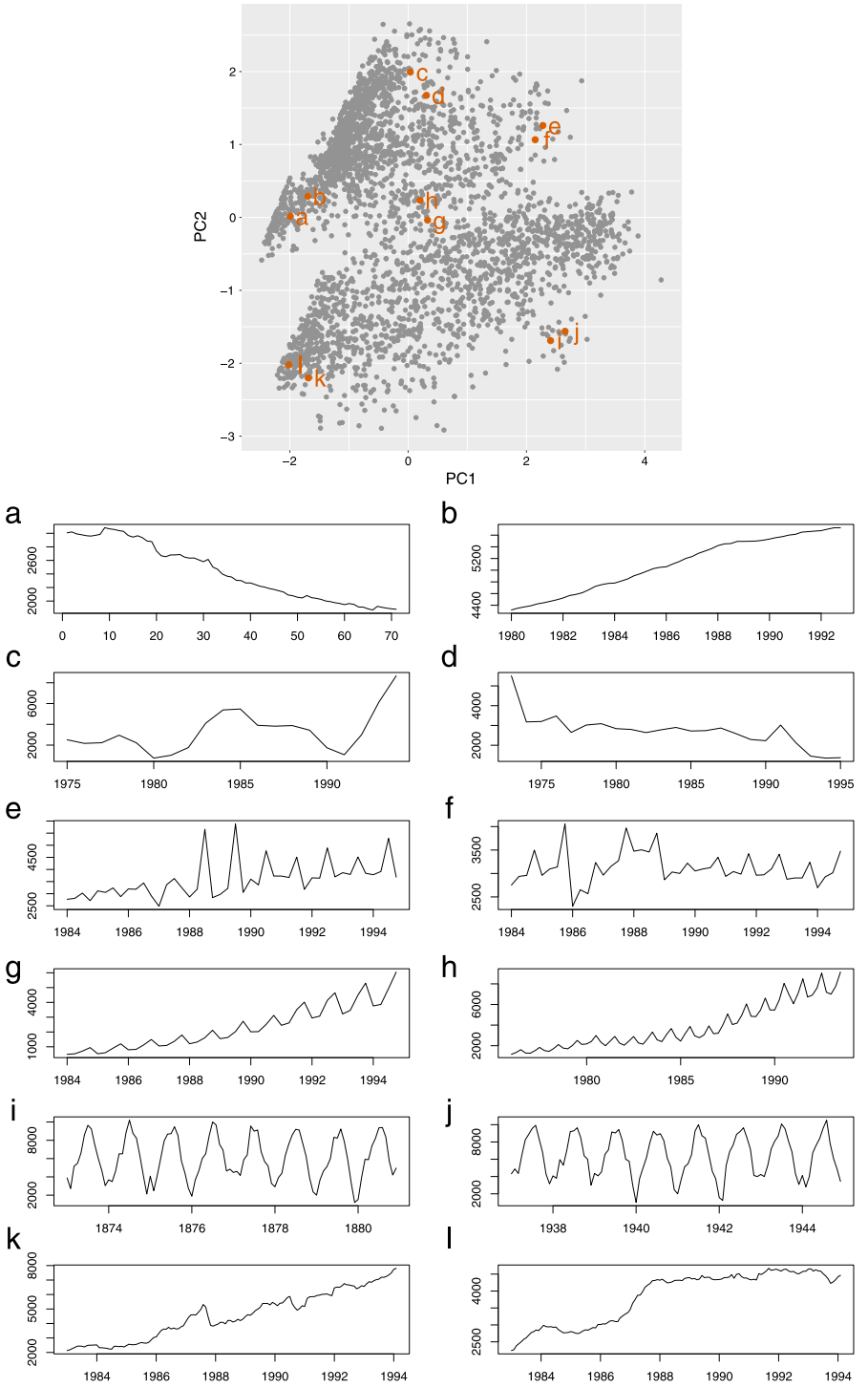

Fig. 2. Top panel: Instance space of M3 time series. PC1 and PC2 are the first two principal components, projected from the six-dimensional feature space. Points a–l, are also shown, and represent 12 examples of actual series in M3. Bottom panel: Actual M3 time series with locations a–l shown in the top panel.

Interesting properties of instances are demonstrated in their feature distributions across the instance space in Fig. 3. Time series in the lower left quadrant are easier to forecast, with lower spectral entropy values and higher trends. Most of the time series in M3 are trended and have low seasonality values, which is consistent with the kernel density estimate shown in Fig. 1. Outlying series can be observed, such as the monthly series on the far right of the plot that have low trend but high seasonality and high entropy. We also see clearly where yearly, quarterly

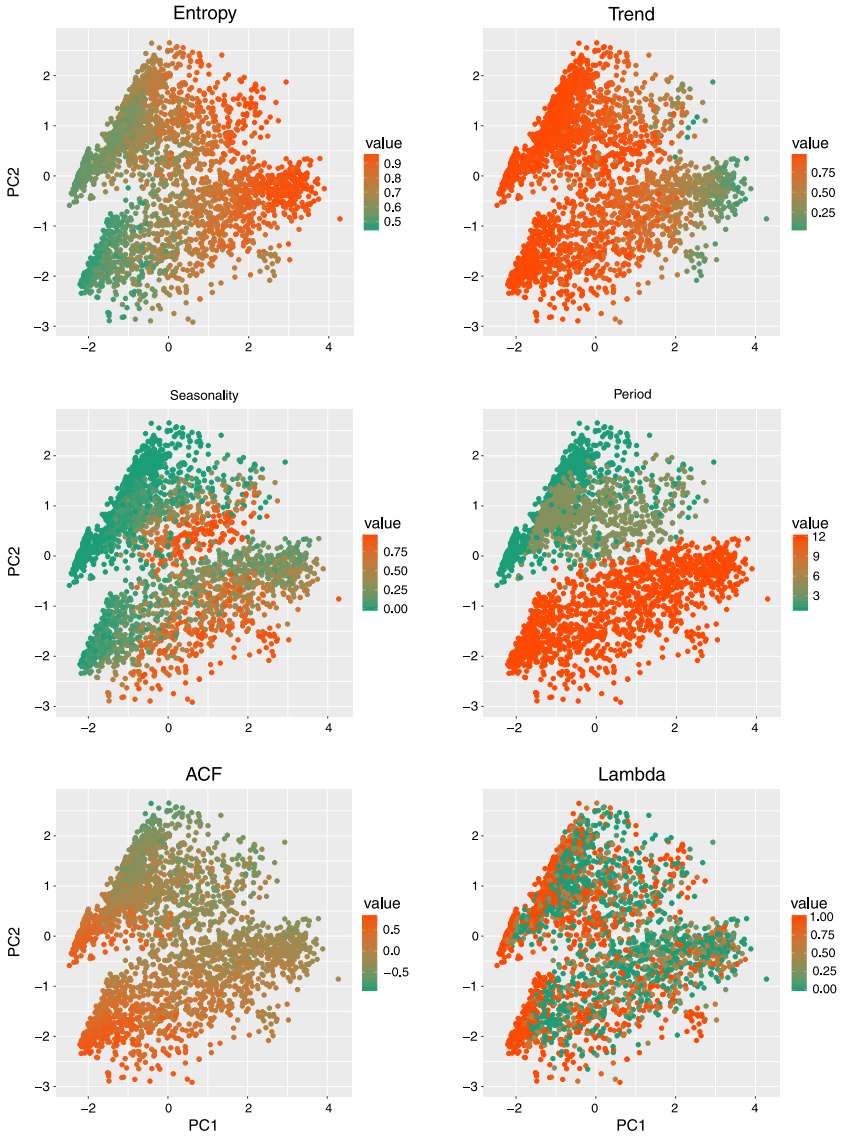

Fig. 3. Distribution of the features' values across the instance space. Given a time series x_t in M3, the entropy feature measures its forecastability. The trend and seasonality are the strength of the trend and seasonality components in x_t. The period of x_t is 1, 4, or 12, depending on its periodic pattern. ACF is the first order autocorrelation, calculated on the de-trended and de-seasonalized series of x_t. Lambda is the optimal Box–Cox transformation parameter, which is a measure of the variance-stability of x_t.

and monthly series lie in the instance space and how their degree of seasonality varies by looking at the distribution of their seasonal period features. Time series in the lower left quadrant have higher first order autocorrelation values after being de-trended and de-seasonalized. The Box–Cox transformation parameter is not as representative as other features from its distribution, as can also be seen from Eq. (1).

More 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, but sparser areas towards the right side. Is it possible to generate more instances in order 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? The following sections provide some insights regarding this question by presenting a method for the generation of new time series instances with controllable features in targeted locations in the instance space.

3. New time series generation in instance space

We increase the diversity and evenness of the instance space by evolving new instances in locations in which target points are set. Once a target point has been 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 as close as possible to the target point. The population is composed of a number of individual time series, each of which is evaluated by a fitness function. Aiming to reach the location of a target point, we require the fitness function of a time series to increase while its Euclidean distance to the target point decreases, for a close analogy with the genetic algorithm's maximisation of a fitness function. Only the fittest individuals reproduce, passing their characteristics to their offspring. The initial populations are then improved through a repeated application of the selection, crossover and mutation operators until the final population is achieved with a 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, thus 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 in order 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 onto the two-dimensional instance space using Eq. (1). Denote the two-dimensional projection of instance j by PC_j.
2. Calculate the fitness of each member in the current population:

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

3. Evolve the next generation based on crossover, mutation and the survival of the fittest individual premise in order to improve the average fitness of each generation. In the crossover process, the offspring time series are formed with the crossover probability by combining different parts of the samples from their parental time series, which are partitioned by a random integer in the range of the time series length. The mutation process alters the values of some samples in the parental time series randomly using the mutation probability.

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 then 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).

The instance space in Fig. 2 allows us to observe the ranges of the first two principal component axes, PC1 and PC2, and we now 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 that lie outside the boundaries of the current space and to find a more general boundary for the instance space. We then use a genetic algorithm to generate 1024 previously unknown yearly, quarterly and monthly time series that are 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 certain lengths, we evolve yearly, quarterly and monthly time series, with lengths of 30, 60 and 120, respectively. This means that we go through the above evolution process $3 \times 1024 = 3072$ times and generate 1024 yearly series with length 30, 1024 quarterly series with length 60, and 1024 monthly series with length 120. These lengths were chosen as round numbers that are close to the upper quartile of lengths in the corresponding M3 series.

Given a target point T_i, $i = 1, 2, \ldots, 1024$, we generate 1024 yearly series by setting the crossover probability to be 0.8 and the mutation probability to be 0.4. The size of the initial population is set to $N_p = 20$. These 20 initial series are selected from the M3 data but exclude any series with distance from the target T_i that is less than 0.3, to avoid a full replication of the 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 that are closer in distance than 0.3. For each selected series, if its length is greater than the targeted length (which is 30 for yearly data), it is truncated to the target length; otherwise, it is reflected to create a series of the target length.

The initial populations for the evolution of quarterly and monthly series are similar, meaning that they are seeded in areas in which optimal solutions are likely to be found. Their lengths are truncated or reflected to 60 for quarterly data and 120 for monthly data. Fig. 4 shows the 1024 target points that we set and indicates 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 still shrink to smaller, well-defined regions for all of the evolved yearly, quarterly and monthly series.

This suggests that yearly, quarterly and monthly data have natural boundaries within this two-dimensional instance space, probably due to constraints on combinations of features. For example, Fig. 1 suggests that it is impossible to have a series with a very low spectral entropy and very low trend, and it is clearly impossible to have a series with both a high seasonality and a period of 1. Also note that there is a small area in the M3 instance space to the left of the evolved yearly data boundary that is not reached. That group of data is actually from the "OTHER" category in M3, and the series are much longer than those for the evolved yearly data. We find that shorter series cannot get very low entropy feature values for the M3 data (figures not shown here). Being much shorter, the evolved yearly

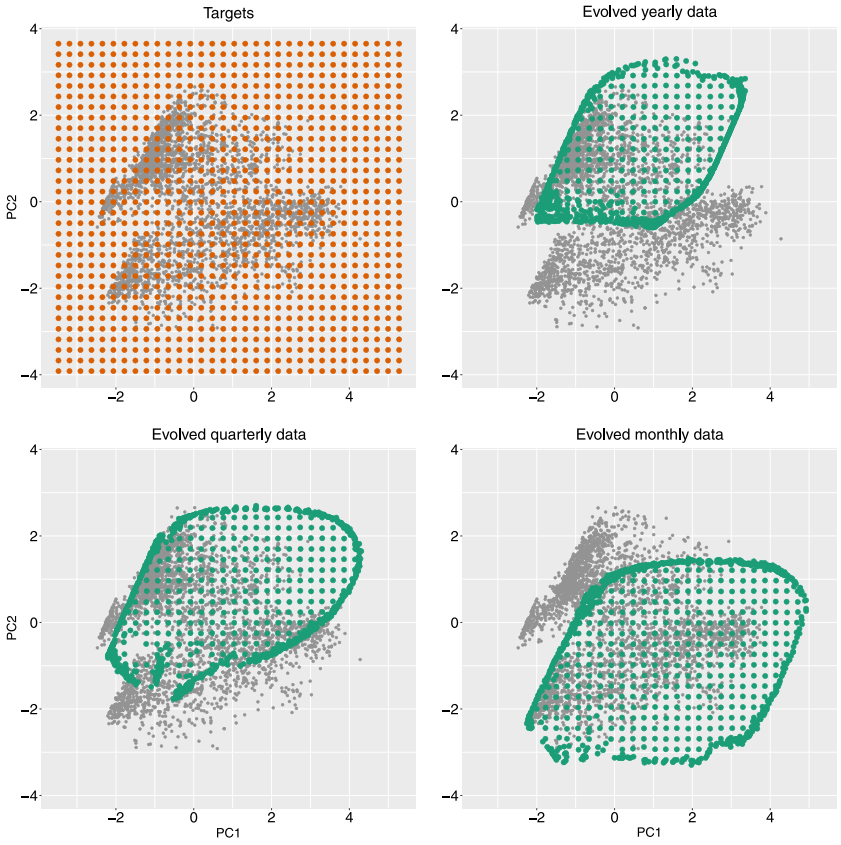

Fig. 4. Evolved new instances using the genetic algorithm. The red points in the top-left panel are the 1024 target points we set; the green points in the top-right panel represent the 1024 yearly time series that are evolved newly from the target points using the generic algorithm. Similarly, the two bottom panels show the 1024 evolved quarterly and monthly time series, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

series cannot reach the very far left area, since the horizontal axis (PC1) is positively related to the entropy feature. These kinds of constraints are presumably what lead to the apparent boundaries seen in Fig. 4.

It is also apparent that the evolved series are more evenly distributed away from the boundaries, while the M3 data have a 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 selected six known time series at random, two from each of the yearly, quarterly and monthly series respectively, which are shown in the top panel of Fig. 5 as points A–F. Can we use the locations of these known time series as target points, then evolve new time series that lie near these targets? The bottom panel of Fig. 5 shows the six target time series on the left and the corresponding evolved ones on the right. As expected, they look similar in terms of their time series characteristics.

Extending this idea, we next attempted to generate new time series that live at empty locations in the instance space. Setting six new target points, shown as points G–L in the top panel of Fig. 6, the bottom panel plots show the six M3 time series that are closest to the target points on the left and the six newly generated time series on the right. They have different characteristics from any of the existing M3 time series.

4. Comparison of time series forecasting methods in the instance space

The No-Free-Lunch theorem was proposed for supervised machine learning by Wolpert (1996) and for search and optimisation by Wolpert and Macready (1997). It tells us that there is never likely to be a single method that fits all situations. Similarly, there is no one time series forecasting method that will always perform best. Even for one particular time series, no one technique is consistently superior to all others (Lawrence, 2001). Petropoulos, Makridakis, Assimakopoulos, and Nikolopoulos (2014) wrote, "as there are 'horses for courses', there must also be forecasting methods that are more tailored to some types of data", and measured the extent of the effects of seven time series features on the forecasting accuracy. Smith-Miles et al. (2014) proposed a method for comparing and visualising the strengths and weaknesses of different graph colouring algorithms across an instance space. Here, we consider six general time series forecasting methods in order to demonstrate the potential of the instance space for algorithm performance visualisation. These are:

Table 2
MASE values of the six methods on the M3 data.

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**	**1.11**	0.89	2.27	**1.43**
ETS	2.88	1.19	**0.86**	**1.82**	**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

- 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 & Nikolopoulos, 2000; Makridakis & Hibon, 2000). We apply the method as implemented in the `stheta` function from the *forecTheta* package (Fiorucci, Louzada, & Yiqi, 2016).
- ETS: exponential smoothing state space modelling (Hyndman, Koehler, Snyder, & Grose, 2002), which is used widely as a general forecasting algorithm for trended and seasonal time series.
- ARIMA: autoregressive integrated moving average models, as implemented rin 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 & 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.

Unless noted above, all methods are implemented in the *forecast* package (Hyndman, 2016).

We begin by showing how well these methods work for forecasting the M3 data. Table 2 shows MASE values (Hyndman & 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; that used for the monthly and quarterly data was the average in-sample MAE after applying a seasonal naïve forecasting method. All of the methods were estimated using the training data that were made available to the M3 competitors, while the MASE values were computed using the hold-out (test) samples that were not available to the original M3 competitors. Both parts of each time series are available in the **Mcomp** package for R (Hyndman, 2013). On average, the Theta method performed the best for the yearly and quarterly data, while ETS performed the best for the monthly and other groups of time series.

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

Table 3
MASE values of the six methods on the evolved series.

Method	Yearly	Quarterly	Monthly	All
Naïve	1.51	2.32	1.64	1.82
Seasonal naïve	1.51	1.19	1.33	1.34
Theta	1.44	1.35	1.07	1.28
ETS	1.69	1.30	1.04	1.34
ARIMA	1.44	1.29	**0.93**	**1.22**
STL-AR	**1.41**	**1.07**	1.34	1.27

Table 3 shows how well the forecasting methods perform on the evolved data. ETS still performs the best for monthly data, as with the M3 data, but the STL-AR method performs the best for yearly and quarterly data, unlike with the M3 data. The ARIMA method does the best overall, and ETS comes fourth, 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 that are 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 for 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 merely a convenience sample of the data available to Makridakis and Hibon. Presumably, the results could have been different again if a different sample of time series had been selected instead.

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

Fig. 8 shows MASE values for each forecasting method and time series of the M3 data. The MASE values were capped at four to make them easier to visualise properly. The plots show which particular regions of the instance space were forecast best by each method. While there are no regions where one method always has low MASE values, or 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 STL-AR performs worse on the quarterly

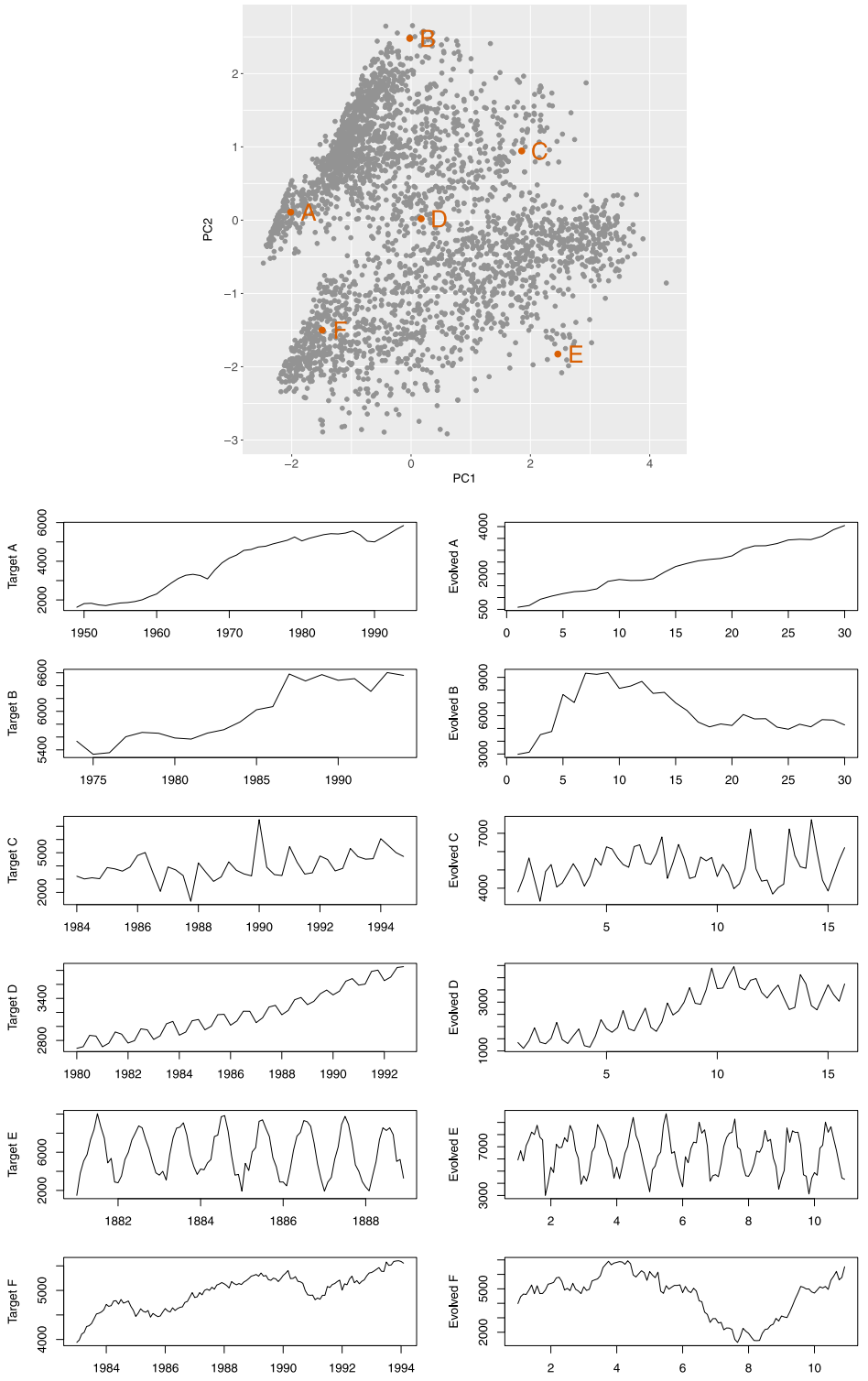

Fig. 5. Top panel: locations of the six target points that correspond to six known time series in the M3 instance space; bottom panel: the six known time series in M3 (on the left), and their corresponding evolved new series using the genetic algorithm (on the right).

series than using ARIMA directly, and the Naïve method cannot forecast the bottom right hand sides of the monthly data well, since they have a high seasonality, as was shown by Fig. 3.

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

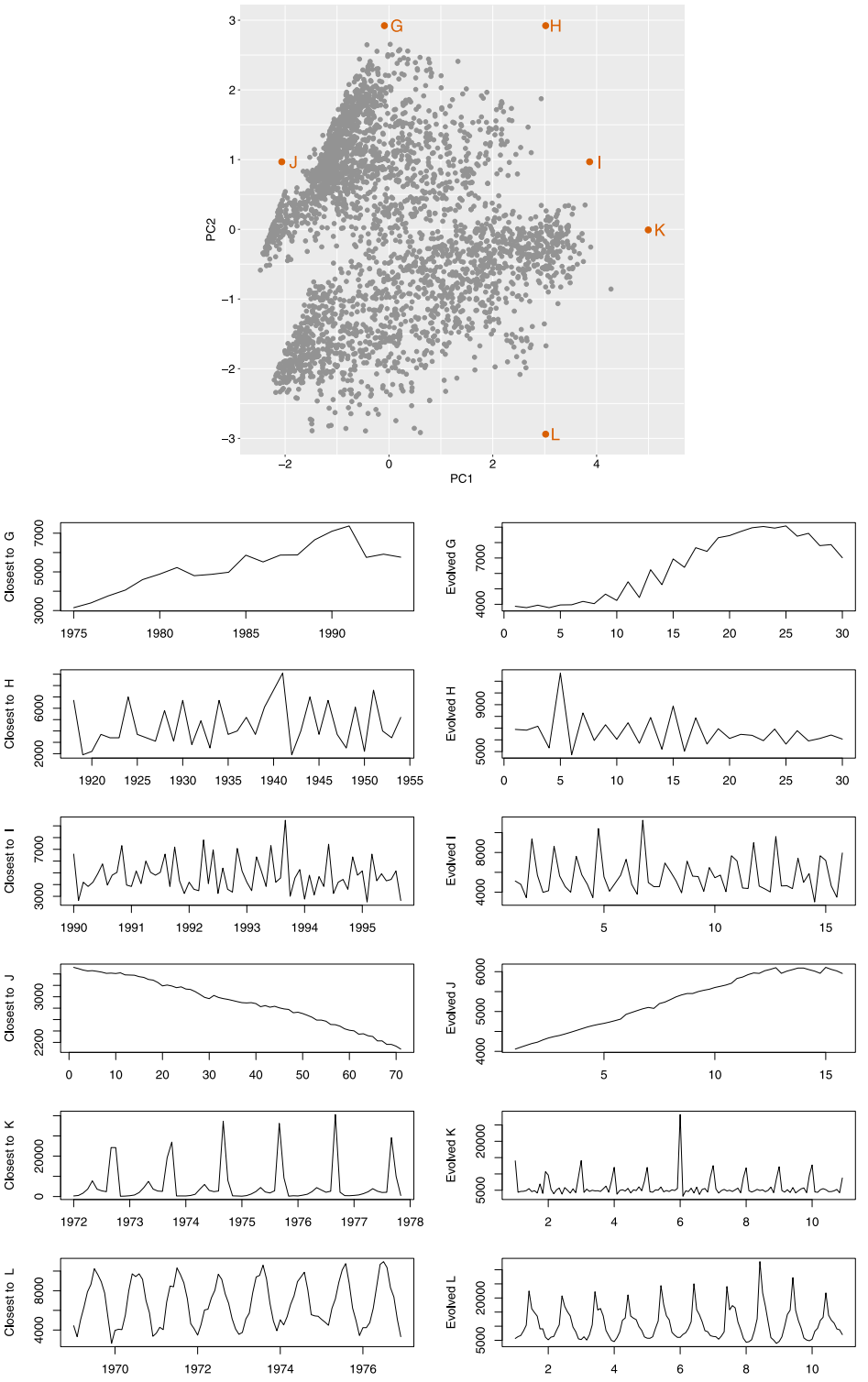

Fig. 6. Top panel: locations of the six new target points that live at empty locations in the M3 instance space; bottom panel: the six time series in M3 which are closest to the six target points (on the left), and the newly evolved series using the genetic algorithm (on the right).

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 very large numbers of forecasts have to be computed quickly, this should lead to substantial efficiency improvements, in which case forecasting performance metrics that balance accuracy and computation times can be considered. We leave the specifics of such an algorithm to a later paper.

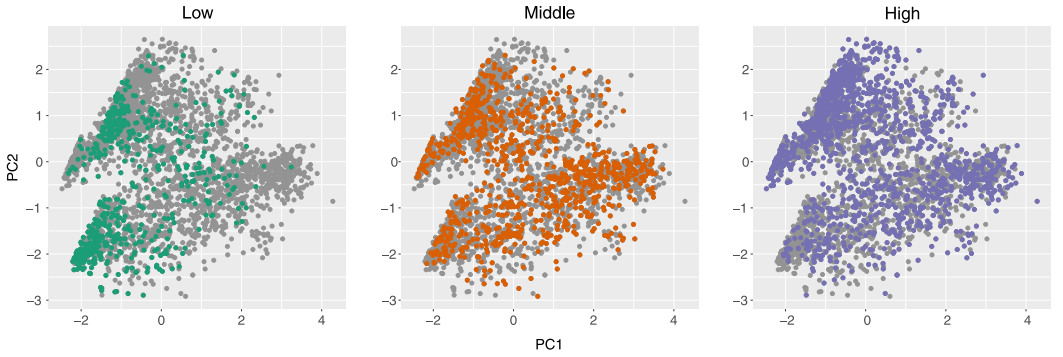

Fig. 7. Locations of M3 series which achieved low, middle and high minimum MASE values from all six forecasting algorithms. These low, middle, and high minimum MASE values are defined based on the 20th and 50th percentiles of the minimum MASE values.

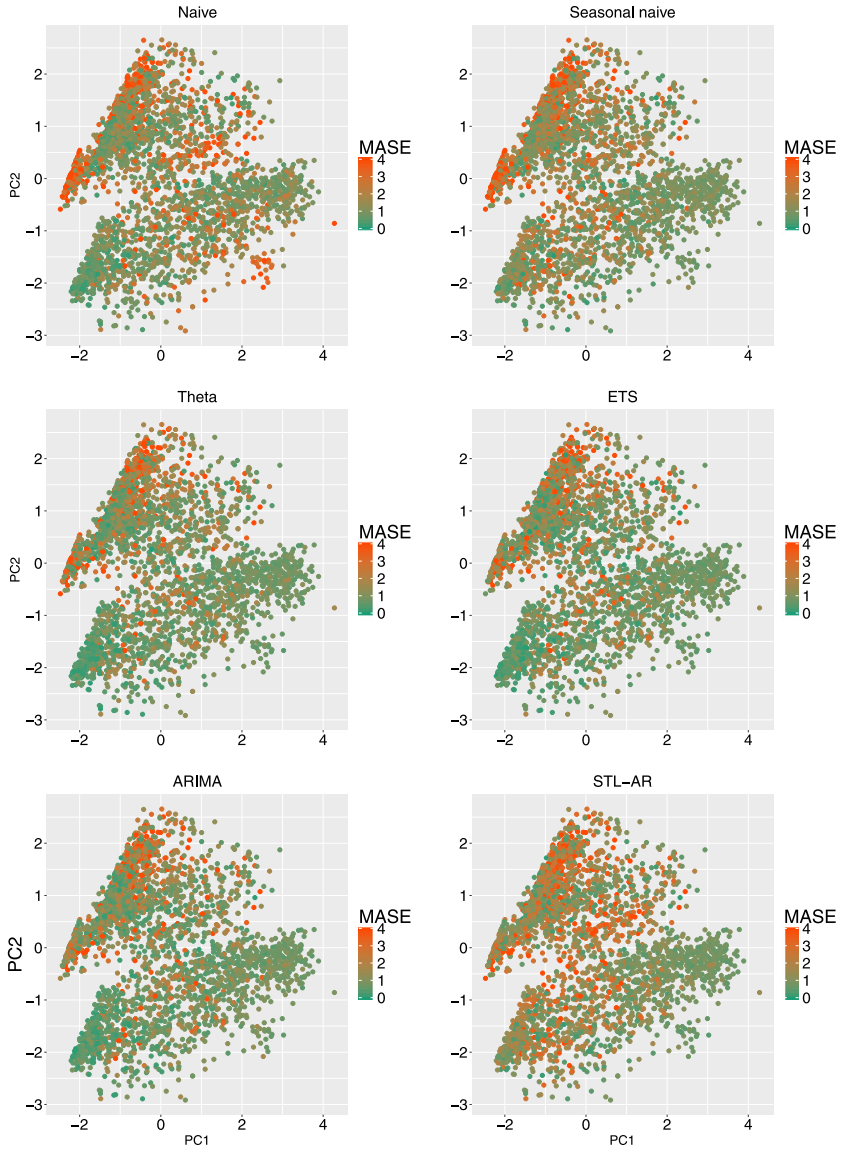

Fig. 8. The distribution of out-of-sample MASE values of the M3 data in the instance space for the six forecasting methods. The MASE values were capped at four to make them easier to visualise properly.

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 insights into large collections of time series.

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

Second, we have proposed an algorithm for generating new time series with controllable characteristics. This can be used to generate either new time series with characteristics that are similar to those of the collection of existing time series, or new time series which have specific locations in the feature space, including regions with no existing time series. This allows us to explore the complete feature space of possible time series, and 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 some forecasting methods perform better in some regions of the feature space than other methods. This introduces the possibility of developing meta-forecasting algorithms that 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 here, our proposed approach is general and can be applied to any large collection of time series. Further, the specific features that 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.

Acknowledgment

This research was supported by the Australian Research Council under grants DP120103678 and FL140100012. The authors are grateful to the handling editor Professor Dick Van Dijk and the two reviewers who made useful comments that improved the clarity of the paper.

References

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

Bandt, C., & Pompe, B. (2002). Permutation entropy: A natural complexity measure for time series. *Physical Review Letters*, *88*, 174102.

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B*, *26*(2), 211–252.

Clements, M. P., & Hendry, D. F. (2001). Explaining the results of the M3 forecasting competition. *International Journal of Forecasting*, *17*, 550–554.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, *6*(1), 3–73.

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

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

Fiorucci, J. A., Louzada, F., & Yiqi, B. (2016). forecTheta: Forecasting Time Series by Theta Models. R package version 2.2. URL: https://CRAN.R-project.org/package=forecTheta.

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

Fulcher, B. D., Little, M. A., & Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, *10*(83), 20130048.

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

Goerg, G. M. (2013). Forecastable component analysis. In *S. Dasgupta D. Mcallester, Eds., Proceedings of the 30th International Conference on Machine Learning, ICML-13, Vol. 28, JMLR Workshop and Conference Proceedings* (pp. 64–72).

Goerg, G. M. (2014). ForeCA: An R package for Forecastable Component Analysis. R package version 0.1. URL: http://cran.r-project.org/package=ForeCA.

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

Hyndman, R. J. (2016). forecast: Forecasting functions for time series and linear models. R package version 7.1. URL: http://cran.r-project.org/package=forecast.

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

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

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

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

Hyndman, R. J., Wang, E., & Laptev, N. (2015). Large-scale unusual time series detection. In *Proceedings of the IEEE international conference on data mining, Atlantic City, NJ, USA. 14–17 November 2015* URL: http://robjhyndman.com/working-papers/cikm2015/.

Kang, Y., Belušić, D., & Smith-Miles, K. (2014). Detecting and classifying events in noisy time series. *Journal of the Atmospheric Sciences*, *71*(3), 1090–1104.

Kang, Y., Belušić, D., & Smith-Miles, K. (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., & Racine, J. (2002). Entropy and predictability of stock market returns. *Journal of Econometrics*, *107*(1–2), 291–312.

Makridakis, S., & Hibon, M. (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, Technical Report 33*. Department of Mathematics and Computer Science, Philips-University Marburg.

Nanopoulos, A., Alcock, R., & Manolopoulos, Y. (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.

Petropoulos, F., Makridakis, S., Assimakopoulos, V., & Nikolopoulos, K. (2014). 'Horses for courses' in demand forecasting. *European Journal of Operational Research*, *237*(1), 152–163.

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

Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, *45*(0), 12–24.

Smith-Miles, K., & Bowly, S. (2015). Generating new test instances by evolving in instance space. *Computers & Operations Research*, *63*, 102–113.

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

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

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.
Zaccarelli, N., Li, B.-L., Petrosillo, I., & Zurlini, G. (2013). Order and disorder in ecological time-series: Introducing normalized spectral entropy. *Ecological Indicators*, *28*, 22–30.

Yanfei Kang is Associate Professor of Statistics in the School of Economics and Management at Beihang University, China. Prior to that, she was Senior R&D Engineer in the Big Data Group of Baidu Inc. Yanfei obtained her Ph.D. degree in Applied and Computational Mathematics at Monash University in 2014. She then worked as a postdoctoral research fellow in time series forecasting algorithm performance visualization at Monash University in 2014 and 2015. Her research interests include time series forecasting, time series visualization, text mining and statistical computing.

Rob J. Hyndman is Professor of Statistics in the Department of Econometrics and Business Statistics at Monash University, and Editor-in-Chief of the *International Journal of Forecasting*. Rob has written more than 100 research papers and five books. He also maintains an active consulting practice, and has provided advice to hundreds of clients around the world. He has won awards for his research, teaching, consulting and graduate supervision.

Kate Smith-Miles holds a Professorial position in the School of Mathematical Sciences at Monash University in Australia, and a five-year Laureate Fellowship from the Australian Research Council. She is also the inaugural Director of MAXIMA (the Monash Academy for Cross & Interdisciplinary Mathematical Applications). Kate has published two books on neural networks and data mining, and over 230 refereed journal and international conference papers in the areas of neural networks, optimization, data mining, and various applied mathematics topics. Kate was elected Fellow of the Institute of Engineers Australia (FIEAust) in 2006, and Fellow of the Australian Mathematical Society (FAustMS) in 2008. Her awards include the Australian Mathematical Society Medal in 2010 for distinguished research, and the Vice-Chancellor's Award for Excellence in Postgraduate Supervision in 2012. She regularly acts as a consultant to industry in related areas and currently serves as Area Editor for the journal *Computers and Operations Research*.