ISSN 1440-771X

MONASH
BUSINESS
SCHOOL

Department of Econometrics and Business Statistics

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

Efficient generation of time series with diverse and controllable characteristics

Yanfei Kang, Rob J Hyndman, Feng Li

September 2018

Working Paper 15/18

Efficient generation of time series with diverse and controllable characteristics

Yanfei Kang
School of Economics & Management, Beihang University, Beijing 100191 China.

Rob J Hyndman
Department of Econometrics & Business Statistics, Monash University, Clayton, Victoria, 3800, Australia.

Feng Li
School of Statistics and Mathematics, Central University of Finance and Economics, 100081 Beijing, China.
Email: feng.li@cufe.edu.cn
Corresponding author

1 September 2018

JEL classification: C22, C15

Efficient generation of time series with diverse and controllable characteristics

Abstract

The explosion of time series data in recent years has brought a flourish of new time series analysis methods, for forecasting, clustering, classification and other tasks. The evaluation of these new methods requires a diverse collection of time series data to enable reliable comparisons against alternative approaches. We propose the use of mixture autoregressive (MAR) models to generate collections of time series with diverse features. We simulate sets of time series using MAR models and investigate the diversity and coverage of the simulated time series in a feature space. An efficient method is also proposed for generating new time series with controllable features by tuning the parameters of the MAR models. The simulated data based on our method can be used as evaluation tool for tasks such as time series classification and forecasting.

Keywords: Time series features; Time series generation; Mixture autoregressive models

1 Introduction

With the widespread collection of time series data via scanners, monitors and other automated data collection devices, there has been an explosion of time series analysis methods developed in the past decade or two. Paradoxically, the large datasets are often also relatively homogeneous, which limits their use for evaluation of general time series analysis methods (Keogh & Kasetty 2003; Muñoz et al. 2017; Kang, Hyndman & Smith-Miles 2017). The performance of any time series mining algorithm depends on the diversity of the test data, so that the evaluation of the algorithm can be generalized to a wide range of future data (Smith-Miles & Bowly 2015). As Keogh & Kasetty (2003) argue, after extensive analysis on highly diverse datasets, there is "*a need for more comprehensive time series benchmarks and more careful evaluation in the data mining community*". Although some benchmark data are more popular than others, the time series area still lacks diverse and controllable benchmarks for algorithm evaluation.

Benchmark datasets can be based on real data, simulated data or toy data. However, real data is often expensive, private or simply limited, which makes it difficult to obtain a diverse time

series benchmark. Another drawback is that it is difficult to know if a collection of real data is truly diverse. If such data are used in benchmarking analysis, it may give biased algorithm performances. Toy data is usually artificially generated with a known embedded pattern but not necessarily representative of real data (Olson et al. 2017). Therefore, using realistic simulated data provides a way for benchmark generation.

In recent years, research shows that it is possible to use generated data for algorithm learning under certain application domains, which give great potential for exploring simulated data. One such example is the well-known "Alpha Zero" (Silver et al. 2017), being able to learn from simulated games based on self-play without human input for guidance. Such simulations are usually restricted to a certain rule-based scene, such as the game of Go. However, in this paper, we show that it is possible to simulate diverse time series by exploring the possible time series features and the nature of time dependence.

Some prior approaches have focused on the shapes of one or more given time series, or on some predefined "types" of time series, in order to generate new time series. Vinod, López-de-Lacalle, et al. (2009) use a maximum entropy bootstrap method to generate ensembles for time series data. The generated samples retain the shape, or local peaks and troughs, of the original time series. They are not exactly the same, but 'strongly dependent', and thus can be used for convenient statistical inference. Bagnall et al. (2017) simulate time series data from different shape settings. The simulators are created by placing one or more shapes on a white noise series. Time series classification algorithms are then evaluated on different representations of the data, which helps understand why one algorithm works better than another on a particular representation of the data. The obvious drawback in these generation methods is that it is impossible to create simulated time series that comprehensively cover the possible space of time series.

An alternative approach is to generate new instances with controllable characteristics, a method that has been used in several other areas of analysis including graph coloring (Smith-Miles & Bowly 2015), black-box optimization (Muñoz & Smith-Miles 2016) and machine learning classification (Muñoz et al. 2017). Kang, Hyndman & Smith-Miles (2017) adapt the idea to time series, and show that it is possible to "fill in" the space of a large collection of real time series data by generating artificial time series with desired characteristics. Each time series is represented using a feature vector which is projected on to a two-dimensional "instance space" that can be visually inspected for diversity. Kang, Hyndman & Smith-Miles (2017) use a genetic algorithm to evolve new time series to fill in any gaps in the two-dimensional instance space. In a later related paper, Kegel, Hahmann & Lehner (2017) use STL (an additive decomposition

method) to estimate the trend and seasonal component of a series, which they then modify using multiplicative factors to generate new time series. The evolutionary algorithm approach of Kang, Hyndman & Smith-Miles (2017) is quite general, but computationally slow, while the STL approach of Kegel, Hahmann & Lehner (2017) is much faster but only generates series that are additive in trend and seasonality. In the meta-learning framework of Talagala, Hyndman & Athanasopoulos (2018), a random forest classifier is used to select the best forecasting method based on time series features. The observed time series are augmented by simulating new time series similar to the observed series, which helps to form a larger dataset to train the model-selection classifier. The simulated series rely on the assumed data generating processes (DGPs), which are exponential smoothing models and ARIMA models.

In this paper, we propose a new efficient and general approach to time series generation, using Gaussian mixture autoregressive (MAR) models to simulate a wide range of non-Gaussian and nonlinear time series. Mixture transition distribution models were first developed by Le, Martin & Raftery (1996) to capture many general non-Gaussian and nonlinear features; these were later generalized to MAR models (Wong & Li 2000). We explore simulating data from a random population of MAR models, as well as generating data with specified features. In this way, we provide a solution to the need for a heterogeneous set of time series to use for time series analysis.

Finite mixture models have proven useful in many other contexts as well. Different specifications of finite mixtures have been shown to be able to approximate large nonparametric classes of conditional multivariate densities (Jiang & Tanner 1999; Norets 2010). More general models to flexibly estimate the density of a continuous response variable conditional on a high-dimensional set of covariates have also been proposed (Li, Villani & Kohn 2010; Villani, Kohn & Giordani 2009). Muñoz & Smith-Miles (2016) generate general classification instances with a desired feature vector by fitting Gaussian Mixture Models. Our approach is consistent with this line of literature but we reverse the procedure for generating new time series, in that we use finite mixtures of Gaussian processes to produce time series with specified features.

We first simulate a large time series dataset using MAR models and calculate their features, and the corresponding embedded two-dimensional instance space. The coverage of the simulated data can then be compared with existing benchmarking time series datasets in the two-dimensional space. At the same time, new time series instances with desired features can be generated by tuning a MAR model.

The rest of the paper is organized as follows. Section 2 defines the features we use to characterize a time series. Section 3 generates time series from MAR models and Section 4 investigates the diversity and coverage of the generated data in two-dimensional instance spaces. Section 5 extends our analysis to multi-seasonal time series. To generate time series data with some specific feature targets, Section 6 tunes a MAR model efficiently using a genetic algorithm until the desired time series is evolved. Section 7 concludes the paper.

2 Time series features

A feature F_k can be any kind of function computed from a time series $\{x_1, \dots, x_n\}$. Examples include a simple mean, the parameter of a fitted model, or some statistic intended to highlight an attribute of the data.

A unique "best" feature representation of a time series does not exist (Fulcher 2018). What features are used depends on both the nature of the time series being analyzed, and the purpose of the analysis. For example, consider the mean as a simple time series feature. If some time series contain unit roots, then the mean is not a meaningful feature without some additional constraints on the initial values of the time series. Even if the series are all stationary, if the purpose of our analysis is to identify the best forecasting method, then the mean is probably of no value. On the other hand suppose we are monitoring the CPU usage every minute for numerous servers, and we observe a daily seasonality. Then provided all our time series begin at the same time and are of the same length, the mean provides useful comparative information despite the time series not being stationary. This example shows that it is difficult to formulate general desirable properties of features without knowledge of both the time series properties and the required analysis. We encourage analysts using time series features to consider these things before computing many possibly unhelpful or even misleading features.

Because we are studying collections of time series of different lengths, on different scales, and with different properties, we restrict our features to be ergodic, stationary and independent of scale. Specifically, we consider the set of 26 diverse features shown in Table 1. Some features are from previous studies (Wang, Smith & Hyndman 2006; Fulcher & Jones 2014; Kang, Belušić & Smith-Miles 2014, 2015; Hyndman, Wang & Laptev 2015; Kang, Hyndman & Smith-Miles 2017), and some are new features that we believe provide useful information about our data. Our new features are intended to measure attributes associated with multiple seasonality, non-stationarity and heterogeneity of the time series. The features are defined in the appendix. All features are computed using the `tsfeatures` package (Hyndman et al. 2018b) in R (R Core Team 2018).

Table 1: *The features we use to characterize a time series.*

Feature	Name	Description	Range
F_1	length	Length of the time series	$[1, \infty)$
F_2	nPeriods	Number of seasonal periods	$[1, \infty)$
F_3	Periods	Vector of seasonal periods	$\{1, 2, 3, \dots\}$
F_4	ndiffs	Number of differences for stationarity	$\{0, 1, 2, \dots\}$
F_5	nsdiffs	Number of seasonal differences for stationarity	$\{0, 1, 2, \dots\}$
F_6	(x.acf1, x.acf10, diff1.acf1, diff1.acf10, diff2.acf1, diff2.acf10, seas.acf1)	Vector of autocorrelation coefficients	(-1, 1) or $(0, \infty)$
F_7	(x.pacf5, diff1.pacf5, diff2.pacf5, seas.pacf)	Vector of partial autocorrelation coefficients	(-1, 1) or $(0, \infty)$
F_8	entropy	Spectral entropy	(0, 1)
F_9	nonlinearity	Nonlinearity coefficient	$[0, \infty)$
F_{10}	hurst	Long-memory coefficient	[0.5, 1]
F_{11}	stability	Stability	$(0, \infty)$
F_{12}	lumpiness	Lumpiness	$[0, \infty)$
F_{13}	(unitroot.kpss, unitroot.pp)	Vector of unit root test statistics	$(0, \infty)$ or $(-\infty, \infty)$
F_{14}	(max.level.shift, time.level.shift)	Maximum level shift	$(0, \infty)$
F_{15}	(max.var.shift, time.var.shift)	Maximum variance shift	$(0, \infty)$
F_{16}	(max.kl.shift, time.kl.shift)	Maximum shift in Kulback-Leibler divergence	$(0, \infty)$
F_{17}	trend	Strength of trend	[0, 1)
F_{18}	seasonal.strength	Strength of seasonality	[0, 1)
F_{19}	spike	Spikiness	[0, 1)
F_{20}	linearity	Linearity	$(-\infty, \infty)$
F_{21}	curvature	Curvature	$(-\infty, \infty)$
F_{22}	(e.acf1, e.acf10)	Vector of autocorrelation coefficients of remainder	(-1, 1) or $(0, \infty)$
F_{23}	arch.acf	ARCH ACF statistic	$(0, \infty)$
F_{24}	garch.acf	GARCH ACF statistic	$(0, \infty)$
F_{25}	arch.r2	ARCH R^2 statistic	[0, 1]
F_{26}	garch.r2	GARCH R^2 statistic	[0, 1]

Little previous study has used features for multiple seasonal time series. In multiple seasonal time series, there is more than one seasonal period present in the data; for example, hourly electricity demand data contains a time-of-day pattern (with seasonal period 24), a time-of-week pattern (with seasonal period $7 \times 24 = 168$) and a time-of-year pattern (with seasonal period $365 \times 24 = 8760$). If there are M possible seasonal periods, then $F_2 = M$ and F_3 is an M-vector containing the seasonal periods. For example, with monthly data $F_3 = 12$, and with hourly data $F_3 = (24, 168, 8760)'$. The strength of seasonality (F_{18}) is also an M-vector containing separate measures of the strength of seasonality for each of the seasonal periods.

3 Time series generation from MAR models

3.1 Mixture autoregressive models

Mixture autoregressive models consist of multiple stationary or non-stationary autoregressive components. Being able to capture a great variety of shape-changing distributions, MAR models can handle nonlinearity, non-Gaussianity, cycles and heteroskedasticity in the time series (Wong

& Li 2000). We define a K-component MAR model as:

$$F(x_t|\mathcal{F}_{-t}) = \sum_{k=1}^{K} \alpha_k \Phi\left(\frac{x_t - \phi_{k0} - \phi_{k1}x_{t-1} - \cdots - \phi_{kp_k}x_{t-p_k}}{\sigma_k}\right), \tag{1}$$

where $F(x_t|\mathcal{F}_{-t})$ is the conditional cumulative distribution of x_t given the past information $\mathcal{F}_{-t} \subseteq \{x_{t-1}, \ldots, x_{t-p_k}\}$, $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, $x_t - \phi_{k0} - \phi_{k1}x_{t-1} - \cdots - \phi_{kp_k}x_{t-p_k}$ is the autoregressive term in each mixing component, $\sigma_k > 0$ is the standard error, $\sum_{k=1}^{K} \alpha_k = 1$, and $\alpha_k > 0$ for $k = 1, 2, \ldots, K$. Denoted as MAR$(K; p_1, p_2, \ldots, p_k)$, it is actually a finite mixture of K Gaussian AR models.

The MAR models have appealing properties as they are based on finite mixture models. For example, the conditional expectation, variance and the mth central moment of x_t can be written respectively as:

$$\mathrm{E}(x_t|\mathcal{F}_{-t}) = \sum_{k=1}^{K} \alpha_k(\phi_{k0} + \phi_{k1}x_{t-1} + \cdots + \phi_{kp_k}x_{t-p_k}) = \sum_{k=1}^{K} \alpha_k \mu_{k,t}\,, \tag{2}$$

$$\mathrm{Var}(x_t|\mathcal{F}_{-t}) = \sum_{k=1}^{K} \alpha_k \sigma_k^2 + \sum_{k=1}^{K} \alpha_k \mu_{k,t}^2 - \left(\sum_{k=1}^{K} \alpha_k \mu_{k,t}\right)^2, \tag{3}$$

$$\text{and } \mathrm{E}((x_t - \mathrm{E}(x_t|\mathcal{F}_{-t}))^m) = \sum_{k=1}^{K} \sum_{i=1}^{m} \alpha_k \binom{m}{i} \mathrm{E}((x_t - \mu_{k,t})^i). \tag{4}$$

Thus, the MAR model gives a description of the conditional distribution of the time series, and the shape changing feature of the conditional distributions allow the MAR models to describe processes with heteroskedasticity (Wong & Li 2000). To adapt the MAR model to deal with non-stationarity, one can simply include a unit root in each of the K components in Equation (1). Since a seasonal ARIMA model with seasonal effects and unit roots, denoted as ARIMA$(p, d, 0)(P, D, 0)_{\text{Period}}$, can be simply expanded to AR models, one can flexibly include seasonal effects and non-stationarity in any component in the MAR models.

Furthermore, unlike standard AR models, higher order moments in Equation (4) are also available in MAR densities. The model in Equation (1) is in univariate form, but it is straightforward to extend it to the multivariate case by introducing a multivariate normal CDF and vector autoregressive terms. In principle, one can extend the MAR models to mixtures of both autoregressive and moving average models, but we will keep the MAR form as in Equation (1) and not introduce the unnecessary complexity, because both autoregressive and moving average models can be written in terms of autoregressive models.

In real data, the distribution of the time series can be multi-modal and/or heavy tailed, and so the expectation may not be the best prediction of the future. This is handled nicely with the mixture distribution $F(x_t|\mathcal{F}_{-t})$. From Equation (3), the conditional variance of x_t changes with conditional means of different components. The larger the difference among conditional means $\mu_{k,t}$ $(k = 1, 2, \ldots, K)$, the larger the conditional variance of x_t. The value of $\sum_{k=1}^{K} \alpha_k \mu_{k,t}^2 - (\sum_{k=1}^{K} \alpha_k \mu_{k,t})^2$ is equal to zero only when $\mu_{1,t} = \mu_{2,t} = \cdots = \mu_{k,t}$ which also yields a heavy-tailed distribution; otherwise, it is larger than zero. The baseline conditional variance is $\sum_{k=1}^{K} \alpha_k \sigma_k^2$.

The key merits of MAR models for nonlinear time series modeling are: (1) for a sufficiently diverse parameters space and finite number of components, MAR models are able to capture any time series features in principle (Li, Villani & Kohn 2010); (2) one can simply include seasonal effects and non-stationary in each component (see Section 5); (3) there is no need to treat stationary and non-stationary time series separately as mixtures of stationary and non-stationary components can yield a both stationary and non-stationary process with MAR (Wong & Li 2000); (4) the conditional distributions of the time series given the past information change with time which allows for meaningful time series evolving with historical information; and (5) the MAR models can handle complicated time series features such as multimodality, heavy tails and heteroskedasticity.

3.2 Diverse time series generation

Due to the flexibility of mixture models, they have been successfully applied in many statistical domains; e.g. Bayesian nonparametrics (Escobar & West 1995), forecasting (Li, Villani & Kohn 2010), model selection (Constantinopoulos, Titsias & Likas 2006) and averaging (Villani, Kohn & Giordani 2009), classification methods (Povinelli et al. 2004), and text modeling (Griffiths et al. 2004). Nevertheless, in this extensive literature, little attention has been given to data generation which is crucially important for evaluating the performance of all the tasks mentioned above. Data generating processes do not require sophisticated modeling techniques but they do require a priori knowledge of the target data space. This space is usually huge and extremely difficult to simulate in a non-time-series context. However, generating diverse time series is possible if one can explore a wide range of time dependencies in time series. In this section we demonstrate how to generate a set of diverse time series data based on the nature of time series dependence.

We design a simulation study to provide insights into the time series simulated from mixture autoregressive models. A significant difference in our data generation process compared to typical simulation processes used in the statistical literature (where the data are generated

Table 2: *Parameter settings for simulating time series using mixtures of ARIMA models.*

Parameter	Description	Values
Period	Period of time series	1, 4, 12 or 52
n	Length of time series	Randomly chosen from the lengths of M4 data
K	Number of components	$U\{1,2,3,4,5\}$
α_k	Weights of mixture components	$\alpha_k = \beta_k / \sum_{i=1}^{K} \beta_i$, where $\beta_i \sim U(0,1)$
θ_{ki}	Coefficients of the AR parts	$N(0, 0.5)$
Θ_{kj}	Coefficients of the seasonal AR parts	$N(0, 0.5)$
d_k	Number of differences in each component	Bernoulli(0.9)
D_k	Number of seasonal differences in each component	Bernoulli(0.4)

from models with fixed parameter values), is that we use distributions (see Table 2) instead of fixed values for the parameters in the underlying models. This allows us to generate diverse time series instances. Table 2 shows the parameter settings used in the simulation. These are analogous to non-informative priors (Gelman et al. 2013) in the Bayesian contexts; i.e. the diversity of the generated time series should not rely on the parameter settings.

The periods of the simulated time series are set to be 1, 4, 12 or 52 to match annual, quarterly, monthly and weekly time series. Their lengths are randomly chosen from the lengths of the M4 data (Makridakis, Spiliotis & Assimakopoulos 2018). We randomly draw the number of components, K, from a uniform distribution on $\{1,2,3,4,5\}$. Although, it may be desirable to have a larger K, Villani, Kohn & Giordani (2009) and Li, Villani & Kohn (2010) show that mixture models with comprehensive mean structures are flexible enough with less than five components. The weights of the mixing components, α_k, can be obtained as $\beta_k / \sum_{i=1}^{K} \beta_i$ for $k = 1, 2, \ldots, K$, where the βs follow uniform distributions on (0,1). Assuming that the kth component follows a standard seasonal ARIMA model as ARIMA$(p_k, d_k, 0)(P_k, D_k, 0)_{\text{Period}}$, the coefficients of the AR and seasonal AR parts, $\theta_{ki},\ i = 1, 2, \ldots, p_k$ and $\Theta_{kj},\ j = 1, 2, \ldots, P_k$, follow normal distributions with given mean and variance values. In principle, both the coefficients θ_{ki} and Θ_{kj} could be unbounded, but this limitation is necessary to keep the features of the simulated data as realistic as possible. For the kth mixing component, we perform d_k differences and D_k seasonal differences, where $d_k \sim$ Bernoulli(0.9) and $D_k \sim$ Bernoulli(0.4), respectively.

For the parameter settings given in Table 2, we generate 20,000 yearly, 20,000 quarterly, 40,000 monthly and 10,000 weekly time series based on the MAR models. For each generated time series, we discard the first Period $\times$ 10 samples as burn-in. Figure 1 shows examples of simulated yearly, quarterly, monthly and weekly data. The lengths of the simulated series are set to be

similar to those of the M4 time series data, which is the largest dataset publicly available to be compared in Section 4. Each of the time series can be summarized with a feature vector described in Section 2.

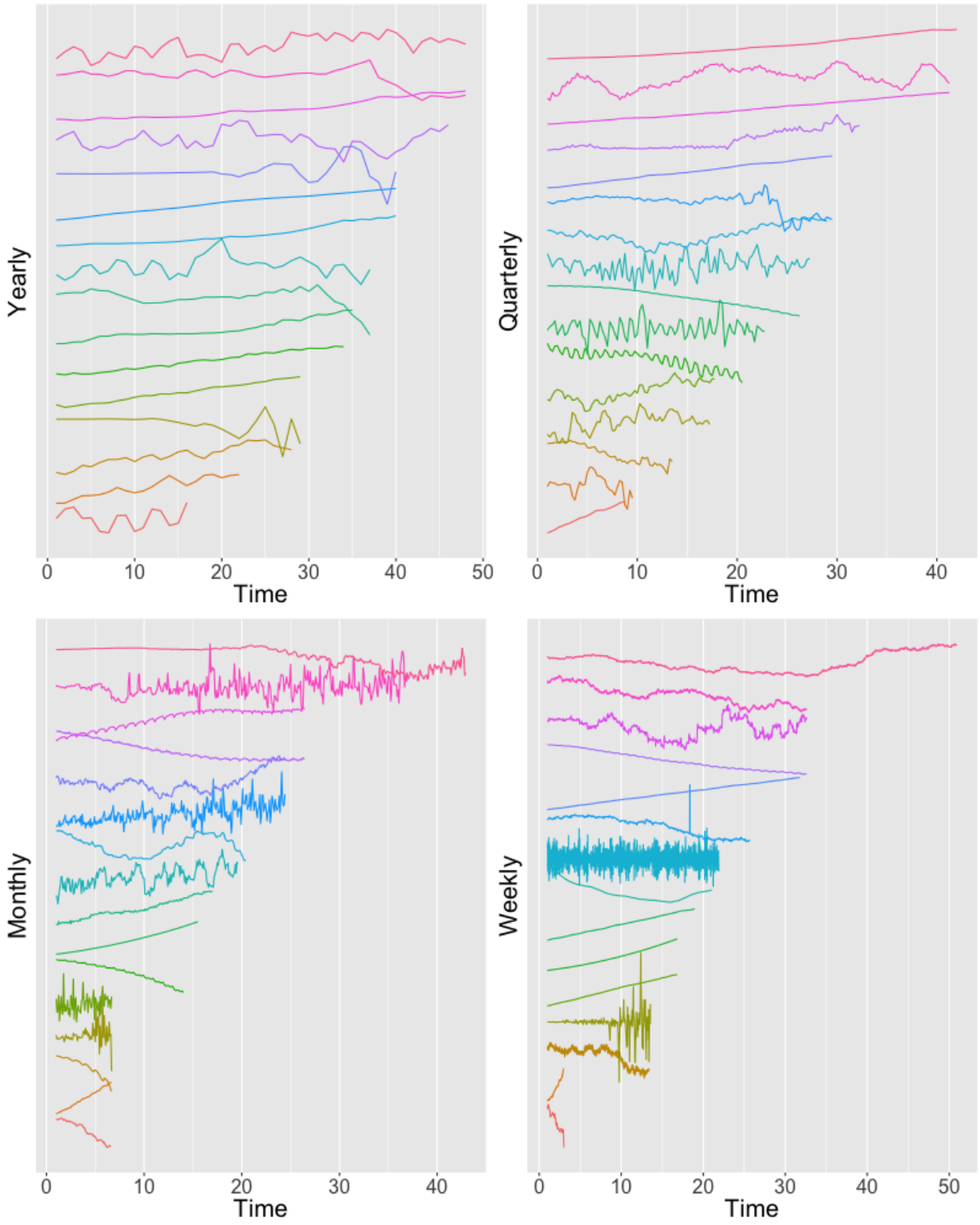

Figure 1: *Examples of simulated yearly, quarterly, monthly and weekly time series of different lengths. The y-axis is omitted to clearly visualize the time series, which are standardized with centering and scaling.*

Table 3: *Computational time for simulation of 1,000 yearly, quarterly, monthly and weekly time series. Different lengths are considered for each seasonal pattern according to the 20%, 50% and 75% quantiles of the time series lengths in M4 data.*

Yearly		Quarterly		Monthly		Weekly	
Length	Time(s)	Length	Time(s)	Length	Time(s)	Length	Time(s)
20	3	60	7	80	13	350	65
30	3	90	10	200	26	900	156
40	4	120	13	300	39	1600	267

Table 3 shows the computational time for simulation of 1,000 yearly, quarterly, monthly and weekly time series. Different lengths are considered for each seasonal pattern according to the 20%, 50% and 75% quantiles of the time series lengths in the M4 data. We have developed an R package `tsgeneration` for the time series generation which is available from https://github.com/ykang/tsgeneration. The code is written in pure R, and we run it on a Laptop with a 2.6 GHz, 8 cores CPU and 16G RAM.

4 Diversity and coverage analysis

We investigate the diversity and coverage of the generated time series data based on MAR models by comparing the feature space of the simulated data with feature spaces of several benchmarking time series datasets, including those from the M1, M3, M4, Tourism, NN5, and NNGC1 forecasting competitions. The R packages they come from, and the numbers of yearly, quarterly, monthly, weekly and daily time series in each dataset are shown in Table 4.

First, we analyze the feature diversity from a marginal perspective. Figure 2 depicts the feature diversity and coverage for simulated yearly, quarterly, monthly and weekly time series compared to the benchmarks for all the possible features we use in the paper. The features in our generated data are diverse in the sense that (1) the shapes of the feature plots for the simulated data widely match to the theoretical ranges of features given in Table 1; and (2) the quantiles of the features for the simulated data cover the shapes of all features in the benchmarking data.

Table 4: *Benchmarking datasets used for comparison with the simulated series from MAR models. The number of series is shown per dataset and seasonal pattern.*

Dataset	R package	Yearly	Quarterly	Monthly	Weekly	Daily
M1	Mcomp	181	203	617	–	–
M3	Mcomp	645	756	1428	–	
M4	M4comp2018	23000	24000	48000	359	4227
Tourism	Tcomp	518	427	366	–	–
NN5	tscompdata	–	–	–	–	111
NNGC1	tscompdata	11	11	11	11	11

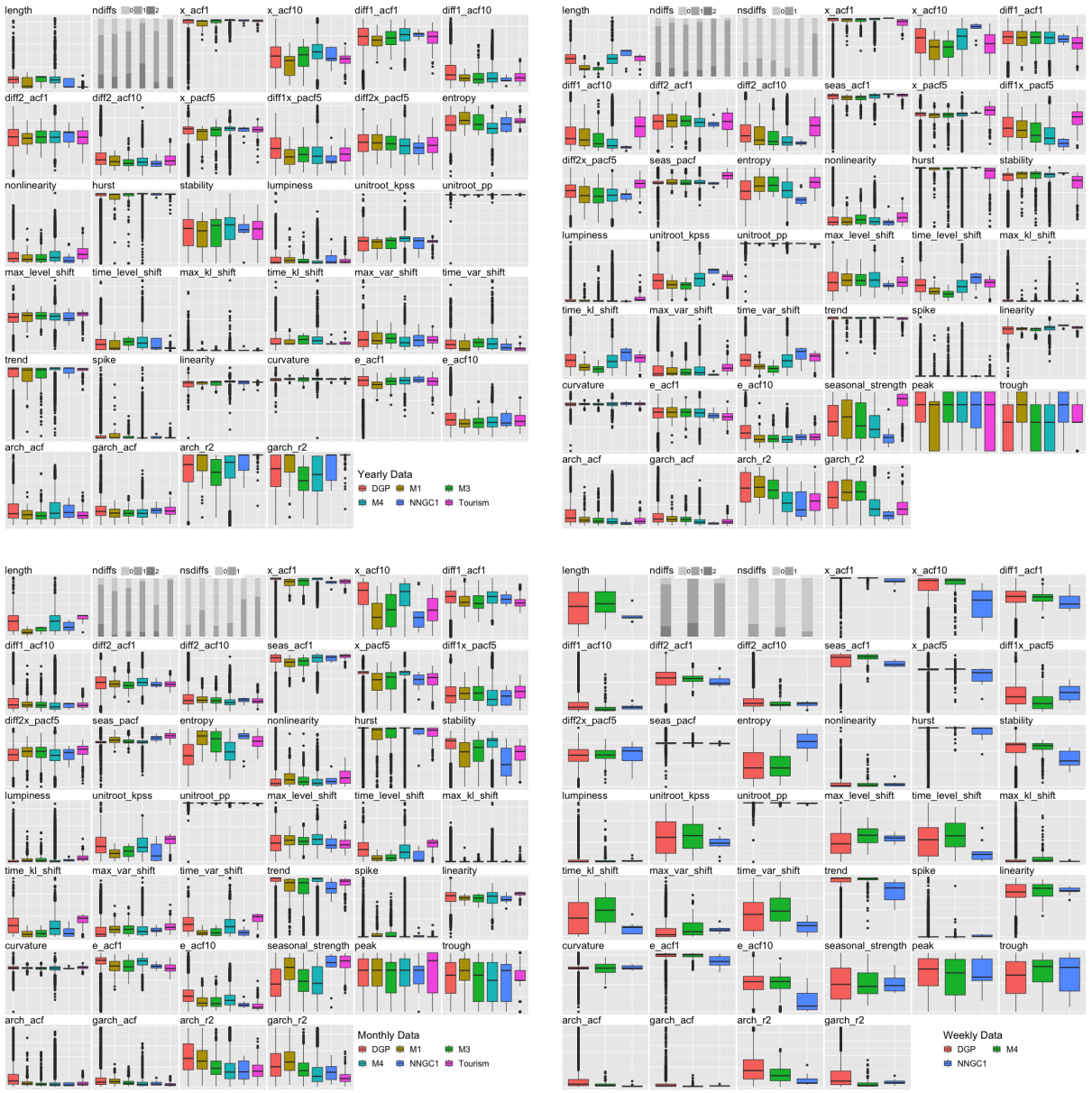

Figure 2: *Plots showing feature diversity and coverage of the simulated yearly, quarterly, monthly, and weekly time series compared with the benchmarking data M1, M3, M4, NNGC1 and Tourism. Boxplots are used for features with continuous values while percentage bar charts for discrete cases. In all the plots, the same order of the datasets is used.*

To better understand the feature space, Kang, Hyndman & Smith-Miles (2017) use principal component analysis (PCA) to project the features to a 2-dimensional "instance space" for visualization. A major limitation of any linear dimension reduction method is that it puts more emphasize on keeping dissimilar data points far apart in the low dimensional space. But in order to represent high dimensional data in a low dimensional, nonlinear manifold, it is also important that similar data points are placed close together. When there are many features and nonlinear correlations are present, using a linear transformation of the features may be misleading. Therefore, we use t-Stochastic Neighbor Embedding (t-SNE), a nonlinear technique

capable of retaining both local and global structure of the data in a single map, to conduct the nonlinear dimension reduction of the high dimensional feature space (Maaten & Hinton 2008).

Figure 3 shows a comparison of PCA and t-SNE for the M3 data. The top row of the plot shows the distribution of the seasonal period period in the t-SNE and PCA spaces, while the bottom row shows that of the spectral entropy feature entropy. It can be seen that t-SNE is better able to capture the nonlinear structure in the data, while the linear transformation of PCA leads to a large proportion of information loss, especially when more features are used. The distribution of other features can be studied similarly.

The simulated yearly, quarterly, monthly and weekly time series are projected into a two-dimensional feature space together with the yearly, quarterly, monthly and weekly time series

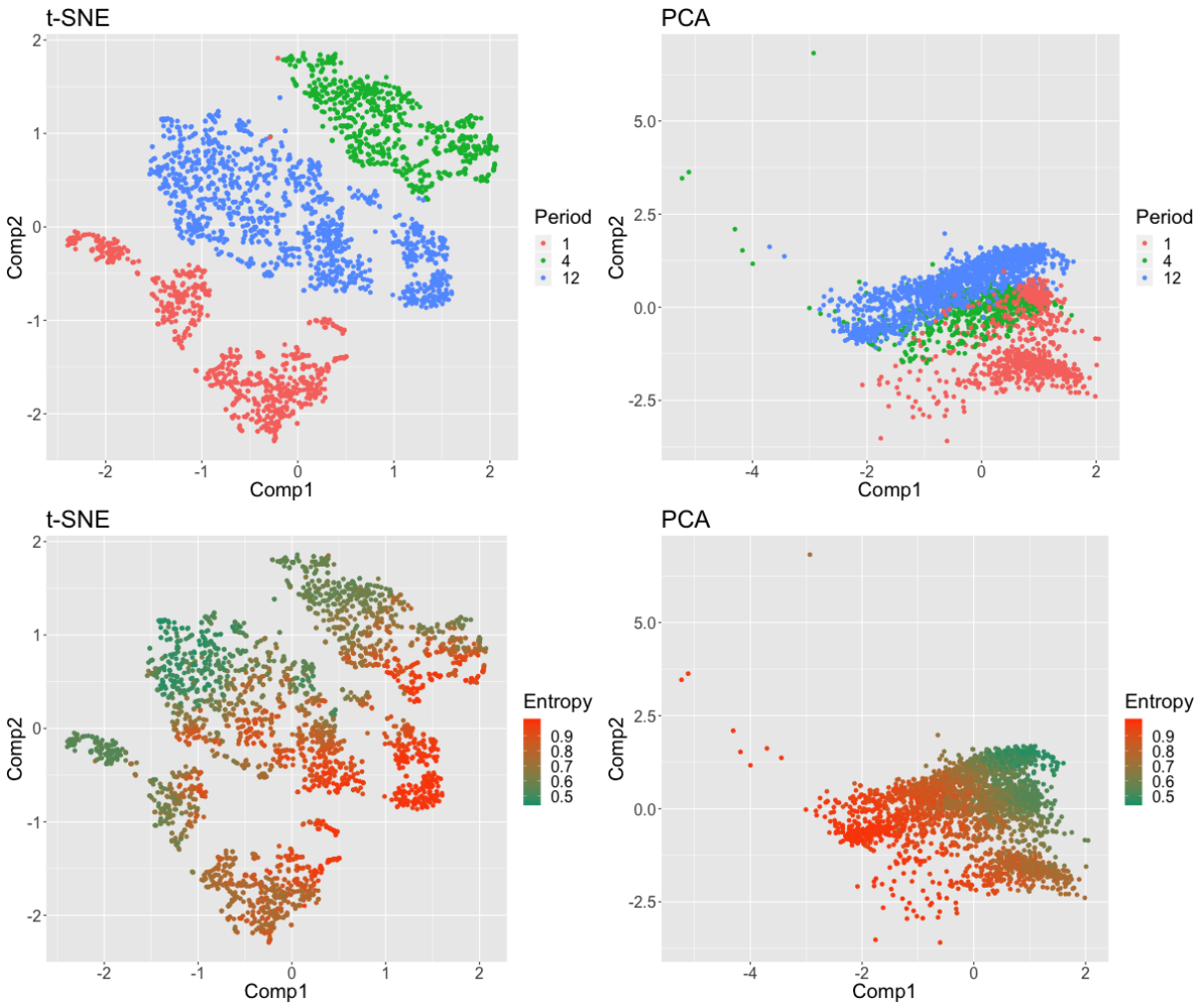

Figure 3: *Two-dimensional spaces of M3 data based on t-SNE (left) and PCA (right). Comp1 and Comp2 are the first two components after dimension reduction using t-SNE and PCA. The top row shows the distribution of* period *in the two spaces, while the bottom row shows that of* entropy*.*

in the benchmarking datasets, as shown in Figure 4. Time series with different seasonal patterns are shown in separate panels of Figure 4 to make the comparisons easier.

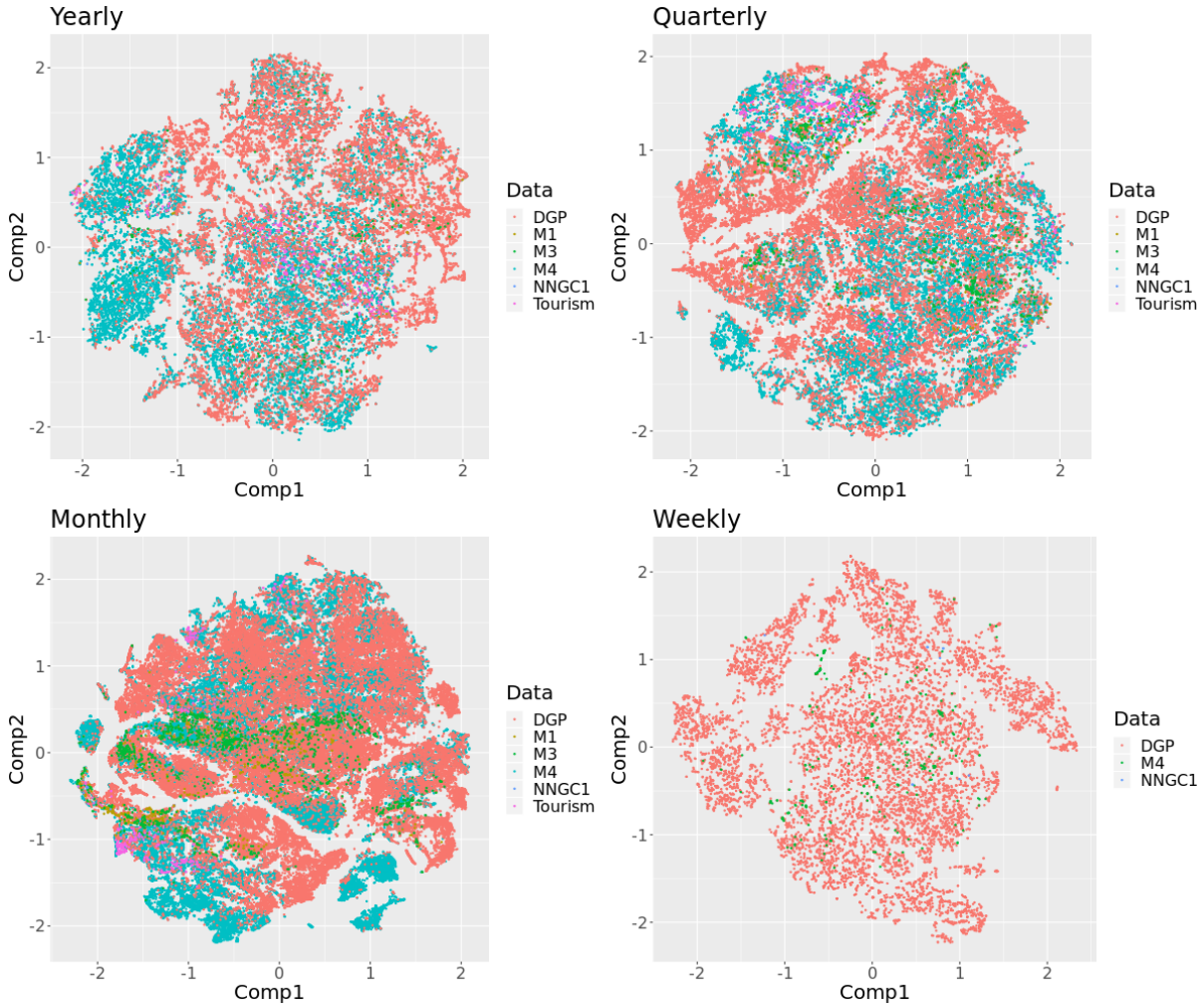

Figure 4: *Two-dimensional t-SNE spaces of the simulated yearly, quarterly, monthly, and weekly time series, together with the time series with the same seasonal patterns from the M1, M3, M4, NNGC1 and Tourism datasets. Comp1 and Comp2 are the first two components after dimension reduction using t-SNE.*

Given the two-dimensional feature spaces of dataset A and dataset B, we quantify the miscoverage of dataset A over dataset B in the following steps:

1. Find the maximum ranges of the x and y axes reached by the combined datasets A and B, and cut the x and y dimensions into $N_b = 30$ bins.
2. In the constructed two-dimensional grid with $N_b^2 = 900$ subgrids, we denote $\mathcal{I}_{i,A} = 0$ if no points in dataset A fall into the ith subgrid, and $\mathcal{I}_{i,A} = 1$ otherwise. An analogous definition of $\mathcal{I}_{i,B}$ applies for dataset B.

Table 5: *Miscoverage of dataset A over dataset B. Take the yearly section for example, the miscoverage of simulated data over M4 is 0.017, while the miscoverage of M4 over simulated data is 0.053.*

Dataset A	Dataset B					
	DGP	M4	M3	M1	Tourism	NNGC1
				Yearly		
DGP	0.000	0.017	0.001	0.001	0.000	0.001
M4	0.053	0.000	0.001	0.001	0.000	0.000
M3	0.609	0.550	0.000	0.041	0.106	0.008
M1	0.680	0.629	0.113	0.000	0.107	0.008
Tourism	0.622	0.568	0.108	0.033	0.000	0.007
NNGC1	0.720	0.669	0.126	0.052	0.124	0.000
				Quarterly		
DGP	0.000	0.013	0.000	0.000	0.000	0.000
M4	0.079	0.000	0.001	0.000	0.001	0.000
M3	0.567	0.536	0.000	0.043	0.096	0.009
M1	0.651	0.616	0.149	0.000	0.110	0.009
Tourism	0.660	0.621	0.200	0.112	0.000	0.010
NNGC1	0.769	0.729	0.243	0.132	0.137	0.000
				Monthly		
DGP	0.000	0.030	0.001	0.000	0.002	0.000
M4	0.061	0.000	0.003	0.001	0.000	0.000
M3	0.488	0.446	0.000	0.020	0.052	0.002
M1	0.566	0.523	0.131	0.000	0.063	0.004
Tourism	0.654	0.602	0.251	0.162	0.000	0.006
NNGC1	0.716	0.669	0.288	0.204	0.084	0.000
				Weekly		
DGP	0.000	0.001	–	–	–	0.000
M4	0.456	0.000	–	–	–	0.007
M3	–	–	–	–	–	–
M1	–	–	–	–	–	–
Tourism	–	–	–	–	–	–
NNGC1	0.576	0.138	–	–	–	0.000

3. The miscoverage of dataset A over dataset B is defined as

$$\text{miscoverage}_{A/B} = N_b^{-2} \sum_{i=1}^{N_b} [(1 - \mathcal{I}_{i,A}) \times \mathcal{I}_{i,B}].$$

Table 5 shows the pairwise miscoverage values of benchmarking dataset A over B. Again, time series with different seasonal patterns are shown separately. The miscoverage values of the simulated dataset from the DGP over others are always smaller than that of others over the DGP. Focusing on the M4 data, the most comprehensive time series competition data to date, the miscoverage values of the DGP over M4 are 0.017, 0.013, 0.030 and 0.001 for yearly, quarterly, monthly and weekly data, respectively. On the other hand, the miscoverage values of M4 over DGP are substantially higher. Therefore, together with Figure 4, we find that the simulated data from MAR models bring more diversity than the existing benchmarking datasets.

5 Multi-seasonal time series generation

So far, we have focused on time series in which there is only one seasonal pattern. However, many time series exhibit multiple seasonal patterns of different lengths, especially those series observed at a high frequency (such as daily or hourly data). For example, Figure 5 shows the half-hourly electricity demand for the state of Victoria, Australia, for 5 weeks in late 2014. There is a clear daily pattern of frequency 48, and a weekly pattern of frequency $48 \times 7 = 336$. With a longer time series, an annual pattern would also become obvious.

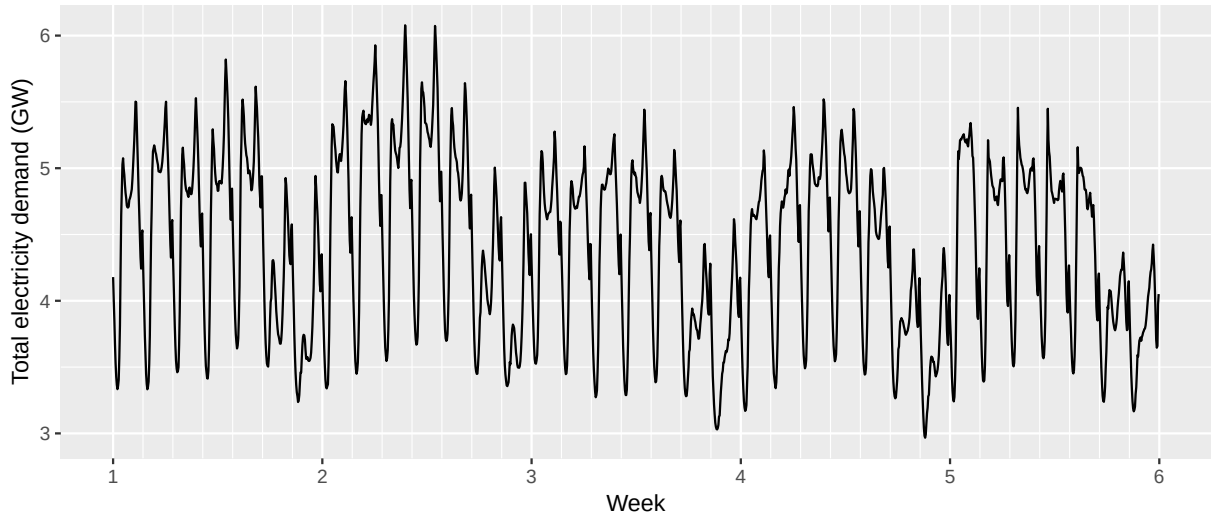

Figure 5: *Half-hourly electricity demand for Victoria, Australia, with a daily pattern of frequency 48 and a weekly pattern of frequency 336.*

Simulation of multi-seasonal time series involves weighted aggregation of simulated time series with the corresponding frequencies. A simulated multi-seasonal time series x_t with M seasonal patterns can be written as

$$x_t = \sum_{m=1}^{M} \omega_m x_{F_m,t},$$

where $m = 1, 2, \ldots, M$, $x_{F_m,t}$ is the mth simulated time series with frequency F_m, and weight ω_m satisfies $\sum_{m=1}^{M} \omega_m = 1$ and $0 < \omega_m < 1$. The weights can be obtained by

$$\omega_m = \frac{\gamma_m}{\sum_{r=1}^{M} \gamma_r}, \text{ where } \gamma_m \sim U(0,1).$$

Figure 6 shows the 2-dimensional t-SNE space of 20000 simulated daily time series (with two seasonal periods, 7 and 365) from the MAR models, compared with the daily series from the NN5 dataset. It can be seen that time series with diverse characteristics are generated and they cover a wider range of characteristics than the NN5 data.

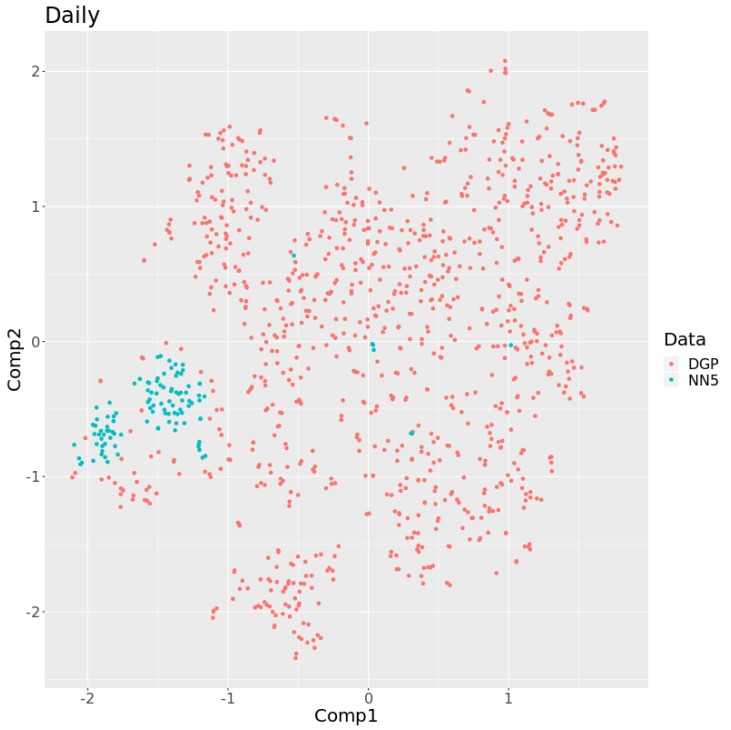

Figure 6: *Two-dimensional t-SNE spaces of the simulated daily time series, together with the daily data in NN5. Comp1 and Comp2 are the first two components after dimension reduction using t-SNE.*

6 Efficient time series generation with target features

In time series analysis, practitioners in certain areas may be only interested in a subset of features, e.g., heteroskedasticity and volatility in financial time series, trend and entropy in time series forecasting, or peaks and spikes in energy time series. Therefore efficient generation of time series with certain features of interest is another important problem to address. For a review of the relevant literature, see Kang, Hyndman & Smith-Miles (2017).

6.1 Tuning a MAR model with target features

Kang, Hyndman & Smith-Miles (2017) use a genetic algorithm (GA) to evolve time series of length n that project to the target feature point $\tilde{\boldsymbol{F}}$ in the two-dimensional instance space as closely as possible. At each iteration of the GA, a combination of selection, crossover and mutation is applied over the corresponding population to optimize the n points of the time series to be evolved. The computational complexity grows linearly as the length of time series increases.

In this paper, instead of evolving time series of length n (i.e., optimization in an n-dimension space), we use a GA to tune the MAR model parameters until the distance between the target feature vector and the feature vector of a sample of time series simulated from the MAR is

close to zero. The parameters for the MAR model, the underlying DGP, can be represented as a vector $\Theta = \{\alpha_k, \phi_i\}$ for $k = 1, \ldots, K$ and $i = k0, \ldots, kp_k$ in Equation (1). A significant improvement compared to Kang, Hyndman & Smith-Miles (2017) is that the length of the vector Θ is much smaller than n, so that the tuning process can be performed efficiently in a much lower-dimensional parameter space. The GA optimization steps are summarized below, for a specified period `Period` and length n for the desired time series.

1. Select a target feature point $\tilde{\boldsymbol{F}}$ in the feature space. Now we aim to find a parameter vector Θ^* that can evolve a time series $X_{\tilde{\boldsymbol{F}}}$ with its feature vector $\boldsymbol{F}$ as close as possible to the target feature point $\tilde{\boldsymbol{F}}$.
2. Generate an initial population of size N_P for the parameter vector Θ, in which each parameter is chosen from a uniform distribution as given in Table 2. That allows the entire range of possible solutions of Θ to be reached randomly.
3. For each iteration, repeat the following steps until some stopping criteria are met.
    1. For each member in the current population, simulate a time series j and calculate its feature vector $\boldsymbol{F}_j$.
    2. Calculate the fitness value for each member:

$$\text{Fitness}(j) = -\frac{1}{c}\|\boldsymbol{F}_j - \tilde{\boldsymbol{F}}\|,$$

    where c is a scaling constant usually defined as $c = \|\tilde{\boldsymbol{F}}\|$.
    3. Produce the new generation based on the crossover, mutation and the survival of the fittest individual to improve the average fitness value of each generation.
4. Keep the time series that is closest to the target feature point (i.e., has the largest fitness value) to be the newly generated time series for the corresponding target.

Table 6 shows the computational time for generating different time series with different sets of features. Lengths are chosen in the same way as for Table 3. The target feature vectors used are median values of the selected features of the simulated data with the same seasonal pattern and similar lengths. The algorithm used in Kang, Hyndman & Smith-Miles (2017) takes about 22,000 seconds to evolve 100 time series of length 100 with 6 features. For the similar task, but with twice as many features, our algorithm is about 40 times faster on average. This speedup allows us to generate time series with controllable features in a reasonable time.

Table 6: *Computational time for generation of 100 yearly, quarterly, monthly and weekly time series. Feature set A consists of* ndiffs*,* x_acf1*,* entropy *and* trend*. Feature set B consists of Feature set A,* diff1_acf1*,* seasonal_strength*,* seas_pacf *and* e_acf1*. Feature set C consists of Feature set B,* e_acf10*,* unitroot_kpss*,* linearity *and* garch_r2*. Median values of the selected features of the simulated data with the same seasonal pattern and similar lengths are used as the targets.*

Yearly		Quarterly		Monthly		Weekly	
Length	Time(s)	Length	Time(s)	Length	Time(s)	Length	Time(s)
Feature set A (four features)							
20	53	60	78	80	62	350	124
30	40	90	71	200	74	900	182
40	40	120	87	300	132	1600	265
Feature set B (eight features)							
20	43	60	130	80	524	350	3001
30	119	90	319	200	480	900	3395
40	101	120	405	300	1340	1600	3674
Feature set C (twelve features)							
20	180	60	650	80	1190	350	1655
30	550	90	530	200	349	900	1360
40	202	120	1160	300	725	1600	1573

6.2 Web application for time series generation

We implement our approach in a shiny app (Chang et al. 2018) as shown in Figure 7. Users can first choose the features that they are interested in. After setting their desired time series seasonal pattern, length, the number of time series, and the feature values required, the generated series are displayed. In Figure 7, we aim to generate ten monthly time series with length 120. The selected features are nsdiffs, x_acf1, entropy, stability, trend, seasonal_strength and garch_r2. The corresponding target vector is $(1, 0.85, 0.55, 0.73, 0.91, 0.95, 0.07)'$. Following the GA process in Section 6.1, the generated time series are shown at the bottom of Figure 7. The simulated data can also be downloaded to a local computer.

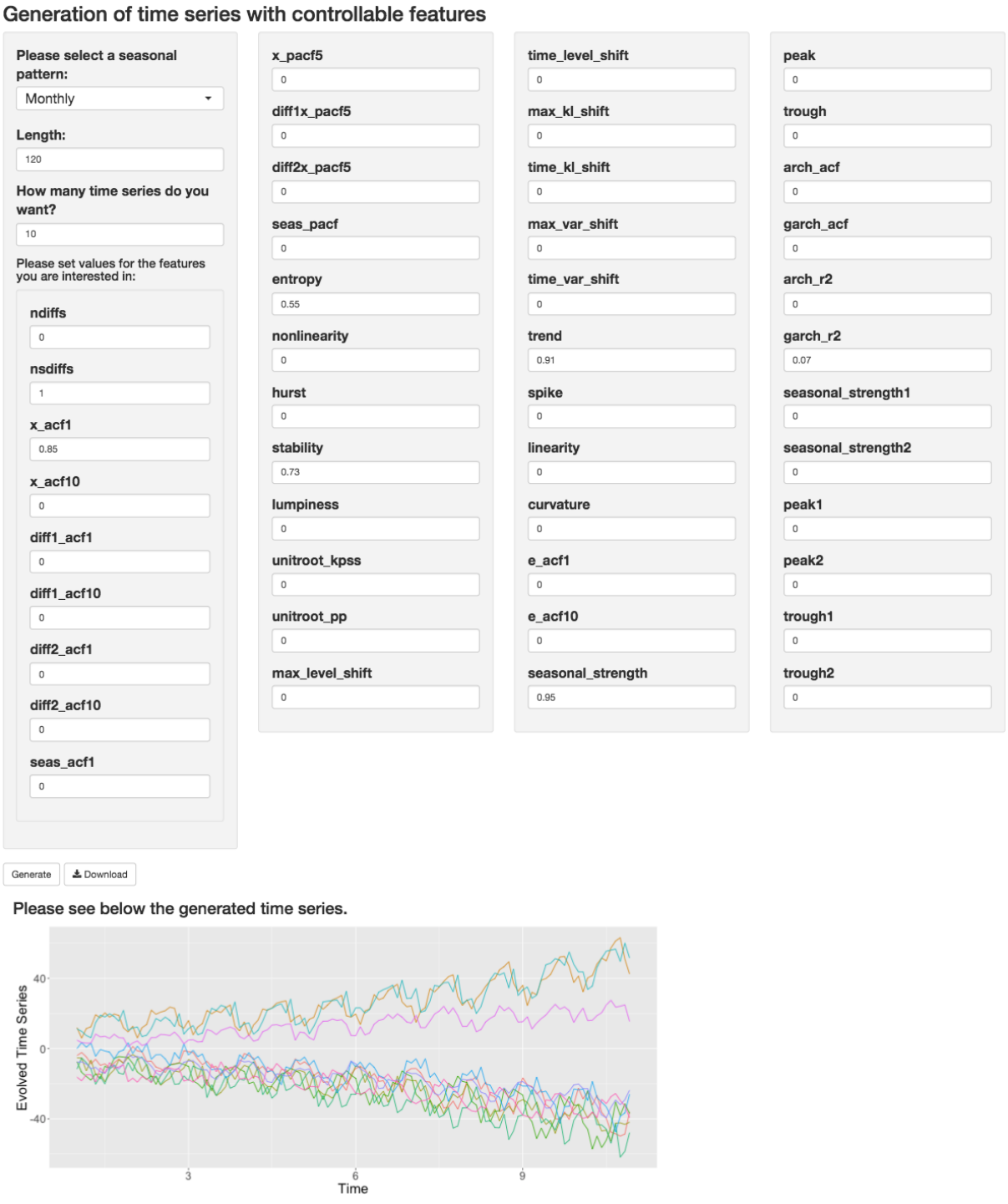

Figure 7: *The shiny application generating time series with controllable features. This interface illustrates the generation of ten monthly time series with length 120. The target feature vector consists of the non-zero values set for the features shown in the interface. The generated ten time series with the target features are shown at the bottom.*

7 Conclusions and future work

We have proposed an efficient simulation method for generating time series with diverse and controllable characteristics. The simulation method is based on the mixture of autoregressive models where the parameters are assigned with statistical distributions. In such way, we provide a general tool serving for advanced time series analysis, e.g. forecasting comparison, model averaging, time series model training with self-generated data, where a large collection of benchmarking data is required. To the best of our knowledge, this is the first paper that thoroughly studies the possibility of generating a rich collection of time series. Our method not only generates realistic time series data but also gives a higher coverage of the feature space than existing time series benchmarking data.

We also have proposed an efficient method for generating new time series with controllable target features by tuning the parameters of MAR models. This is particularly useful in specific areas where only some features are of interest. This procedure is the inverse of feature extraction which usually requires much computational power. Our approach of generating new time series from given features can scale up the computation time by 40 times making feature-driven time series analysis tasks feasible.

A potential application of our work is where it is not possible to share data due to confidentiality or privacy reasons. Instead, one can generate a set of time series data with the same features as the sensitive original time series. The simulated data can easily be shared without compromising commercial, privacy or ethical concerns.

8 Acknowledgments

Yanfei Kang and Feng Li's research were supported by the National Natural Science Foundation of China (No. 11701022 and No. 11501587, respectively).

Appendix: Description of time series features

In this section, we document the feature details in Table 1. We have also developed an R package `tsfeatures` to provide methods for extracting various features from time series data which is available at `https://github.com/robjhyndman/tsfeatures`. Note that all of our features are ergodic for stationary and difference-stationary processes, and not dependent on the scale of the time series. Thus, they are well-suited for applying to a large diverse set of time series.

Each time series, of any length, can be summarized as a feature vector $\boldsymbol{F} = (F_1, F_2, \ldots, F_{26})'$. The length of this vector will be 42 for non-seasonal time series, and for seasonal time series with a single seasonal period. For multiple seasonal time series, $\boldsymbol{F}$ will have a few more elements.

The first five features in Table 1 are all positive integers. F_1 is the time series length. F_2 is the number of seasonal periods in the data (determined by the frequency of observation, not the observations themselves) and set to 1 for non-seasonal data. F_3 is a vector of seasonal periods and set to 1 for non-seasonal data. F_4: the number of first-order differences required before the data pass a KPSS stationarity test (Kwiatkowski et al. 1992) at the 5% level. F_5 is the number of seasonal differences required before the data pass an OCSB test (Osborn et al. 1988) at the 5% level. For multiple seasonal time series, we compute F_5 using the largest seasonal period. For non-seasonal time series (when $F_1 = 1$), we set $F_5 = 0$.

We compute the autocorrelation function of the series, the differenced series, and the twice-differenced series. Then F_6 is a vector comprising the first autocorrelation coefficient in each case, and the sum of squares of the first 10 autocorrelation coefficients in each case. The autocorrelation coefficient of the original series at the first seasonal lag is also computed. For non-seasonal data, this is set to 0.

We compute the partial autocorrelation function of the series, the differenced series, and the second-order differenced series. Then F_7 is a vector comprising the sum of squares of the first 5 partial autocorrelation coefficients in each case. The partial autocorrelation coefficient of the original series at the first seasonal lag is also computed. For non-seasonal data, this is set to 0.

The spectral entropy is the Shannon entropy

$$F_8 = -\int_{-\pi}^{\pi} \hat{f}(\lambda) \log \hat{f}(\lambda) d\lambda,$$

where $\hat{f}(\lambda)$ is an estimate of the spectral density of the data. This measures the "forecastability" of a time series, where low values of F_8 indicate a high signal-to-noise ratio, and large values of F_8 occur when a series is difficult to forecast.

The nonlinearity coefficient (F_9) is computed using a modification of the statistic used in Teräsvirta's nonlinearity test. Teräsvirta's test uses a statistic $X^2 = T \log(\text{SSE1}/\text{SSE0})$ where SSE1 and SSE0 are the sum of squared residuals from a nonlinear and linear autoregression respectively. This is non-ergodic, so is unsuitable for our purposes. Instead, we define $F_9 = X^2/T$ which will converge to a value indicating the extent of nonlinearity as $T \to \infty$.

We use a measure of the long-term memory of a time series (F_{10}), computed as 0.5 plus the maximum likelihood estimate of the fractional differencing order d given by Haslett & Raftery (1989). We add 0.5 to make it consistent with the Hurst coefficient. Note that the fractal dimension can be estimated as $D = 2 - F_9$.

F_{11} and F_{12} are two time series features based on tiled (non-overlapping) windows. Means or variances are produced for all tiled windows. Then stability is the variance of the means, while lumpiness is the variance of the variances.

F_{13} is a vector comprising the statistic for the KPSS unit root test with linear trend and lag one, and the statistic for the "Z-alpha" version of PP unit root test with constant trend and lag one.

The next three features (F_{14}, F_{15}, F_{16}) compute features of a time series based on sliding (overlapping) windows. F_{14} finds the largest mean shift between two consecutive windows. F_{15} finds the largest variance shift between two consecutive windows. F_{16} finds the largest shift in Kulback-Leibler divergence between two consecutive windows.

The following six features (F_{17}– F_{22}) are modifications of features used in Kang, Hyndman & Smith-Miles (2017). We extend the STL decomposition approach (Cleveland et al. 1990) to handle multiple seasonalities. Thus, the decomposition contains a trend, up to M seasonal components, and a remainder component:

$$x_t = f_t + s_{1,t} + \cdots + s_{M,t} + e_t$$

where f_t is the smoothed trend component, $s_{i,t}$ is the ith seasonal component and e_t is a remainder component. The components are estimated iteratively. Let $s_{i,t}^{(k)}$ be the estimate of $s_{i,t}$ at the kth iteration, with initial values given as $s_{i,t}^{(0)} = 0$. Then we apply an STL decomposition to $x_t - \sum_{j\neq i}^{j=1\ M} s_{j,t}^{(k-1)}$ to obtained updated estimates $s_{i,t}^{(k)}$ for $k = 1, 2, \ldots$. In practice, this converges quickly and only two iterations are required. To allow the procedure to be applied automatically, we set the seasonal window span for STL to be 21 in all cases. For a non-seasonal time series (when $F_1 = 1$), we simply estimate $x_t = f_t + e_t$ where f_t is computed using Friedman's "super smoother" (Friedman 1984).

- F_{17} and F_{18} are defined as

$$F_{17} = 1 - \frac{\text{Var}(e_t)}{\text{Var}(f_t + e_t)} \qquad \text{and} \qquad F_{18,i} = 1 - \frac{\text{Var}(e_t)}{\text{Var}(s_{i,t} + e_t)}.$$

If their values are less than 0, they are set to 0, while values greater than 1 are set to 1. For non-seasonal time series $F_{18} = 0$. For seasonal time series, F_{18} is an M-vector, where M is the number of periods. This is analogous to the way the strength of trend and seasonality were defined in Wang, Smith & Hyndman (2006), Hyndman, Wang & Laptev (2015) and Kang, Hyndman & Smith-Miles (2017).

- F_{19} measures the "spikiness" of a time series, and is computed as the variance of the leave-one-out variances of the remainder component e_t.
- F_{20} and F_{21} measures the linearity and curvature of a time series calculated based on the coefficients of an orthogonal quadratic regression.
- We compute the autocorrelation function of e_t, and F_{22} is a 2-vector containing the first autocorrelation coefficient and the sum of the first ten squared autocorrelation coefficients.

The remaining features measure the heterogeneity of the time series. First, we pre-whiten the time series to remove the mean, trend, and autoregressive (AR) information (Barbour & Parker 2014). Then we fit a GARCH(1,1) model to the pre-whitened time series, x_t, to measure for autoregressive conditional heteroskedasticity (ARCH) effects. The residuals from this model, z_t, are also measured for ARCH effects using a second GARCH(1,1) model.

- F_{23} is the sum of squares of the first 12 autocorrelations of $\{x_t^2\}$.
- F_{24} is the sum of squares of the first 12 autocorrelations of $\{z_t^2\}$.
- F_{25} is the R^2 value of an AR model applied to $\{x_t^2\}$.
- F_{26} is the R^2 value of an AR model applied to $\{z_t^2\}$.

The statistics obtained from $\{x_t^2\}$ are the ARCH effects, while those from $\{z_t^2\}$ are the GARCH effects. Note that the two R^2 values are used in the Lagrange-multiplier test of Engle (1982), and the sum of squared autocorrelations are used in the Ljung-Box test proposed by Ljung & Box (1978).

References

Bagnall, A, A Bostrom, J Large & J Lines (2017). Simulated data experiments for time series classification Part 1: accuracy comparison with default settings. *arXiv preprint arXiv:1703.09480*.

Barbour, AJ & RL Parker (2014). psd: Adaptive, sine multitaper power spectral density estimation for R. *Computers & Geosciences* **63**, 1–8.

Chang, W, J Cheng, J Allaire, Y Xie & J McPherson (2018). *shiny: Web Application Framework for R*. R package version 1.1.0. https://CRAN.R-project.org/package=shiny.

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

Constantinopoulos, C, MK Titsias & A Likas (2006). Bayesian feature and model selection for Gaussian mixture models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (6), 1013–1018.

Ellis, P (2016). *Tcomp: Data from the 2010 Tourism Forecasting Competition*. R package version 1.0.0. https://CRAN.R-project.org/package=Tcomp.

Engle, RF (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica* **50**, 987–1007.

Escobar, MD & M West (1995). Bayesian density estimation and inference using mixtures. *Journal of the american statistical association* **90**(430), 577–588.

Friedman, JH (1984). *A variable span scatterplot smoother*. Technical Report 5. Laboratory for Computational Statistics, Stanford University.

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

Fulcher, BD (2018). "Feature-based time-series analysis". In: *Feature engineering for machine learning and data analytics*. CRC Press, pp.87–116.

Gelman, A, JB Carlin, HS Stern, DB Dunson, A Vehtari & DB Rubin (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Griffiths, TL, MI Jordan, JB Tenenbaum & DM Blei (2004). Hierarchical topic models and the nested chinese restaurant process. In: *Advances in neural information processing systems*, pp.17–24.

Haslett, J & AE Raftery (1989). Space-time modelling with long-memory dependence: assessing Ireland's wind power resource. *Journal of the Royal Statistical Society Series C* **38**(1), 1–50.

Hyndman, RJ (2018). *tscompdata: Time series data from various forecasting competitions*. Version 0.0.1. https://github.com/robjhyndman/tscompdata.

Hyndman, RJ, M Akram, C Bergmeir & M O'Hara-Wild (2018a). *Mcomp: Data from the M-Competitions*. Version 2.7. https://CRAN.R-project.org/package=Mcomp.

Hyndman, RJ, E Wang, Y Kang, TS Talagala & SB Taieb (2018b). *tsfeatures: Time Series Feature Extraction*. Version 0.1. https://github.com/robjhyndman/tsfeatures/.

Hyndman, RJ, E Wang & 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.

Jiang, W & MA Tanner (1999). On the approximation rate of hierarchical mixtures-of-experts for generalized linear models. *Neural Computation* **11**(5), 1183–1198.

Kang, Y, D Belušić & 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 & K Smith-Miles (2015). Classes of structures in the stable atmospheric boundary layer. *Quarterly Journal of the Royal Meteorological Society* **141**(691), 2057–2069.

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

Kegel, L, M Hahmann & W Lehner (2017). Generating What-if scenarios for time series data. In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. ACM, pp.3.

Keogh, E & S Kasetty (2003). On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery* **7**(4), 349–371.

Kwiatkowski, D, PCB Phillips, P Schmidt & Y Shin (1992). Testing the null hypothesis of stationarity against the alternative of a unit root How sure are we that economic time series have a unit root? *Journal of Econometrics* **54**, 159–178.

Le, ND, RD Martin & AE Raftery (1996). Modeling flat stretches, bursts outliers in time series using mixture transition distribution models. *Journal of the American Statistical Association* **91**(436), 1504–1515.

Li, F, M Villani & R Kohn (2010). Flexible modeling of conditional distributions using smooth mixtures of asymmetric student-*t densities*. *Journal of Statistical Planning and Inference* **140**(12), 3638–3654.

Ljung, GM & GEP Box (1978). On a measure of lack of fit in time series models. *Biometrika* **65**(2), 297–303.

Maaten, Lvd & G Hinton (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research* **9**(Nov), 2579–2605.

Makridakis, S, E Spiliotis & V Assimakopoulos (2018). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*.

Muñoz, MA & K Smith-Miles (2016). Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary Computation*.

Muñoz, MA, L Villanova, D Baatar & K Smith-Miles (2017). Instance spaces for machine learning classification. *Machine Learning*, 1–39.

Norets, A (2010). Approximation of conditional densities by smooth mixtures of regressions. *Annals of Statistics* **38**(3), 1733–1766.

Olson, RS, W La Cava, P Orzechowski, RJ Urbanowicz & JH Moore (2017). PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining* **10**(1), 36.

Osborn, DR, APL Chui, J Smith & CR Birchenhall (1988). Seasonality and the order of integration for consumption. *Oxford Bulletin of Economics and Statistics* **50**(4), 361–377.

Povinelli, RJ, MT Johnson, AC Lindgren & J Ye (2004). Time series classification using Gaussian mixture models of reconstructed phase spaces. *IEEE Transactions on Knowledge and Data Engineering* **16**(6), 779–783.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Silver, D, J Schrittwieser, K Simonyan, I Antonoglou, A Huang, A Guez, T Hubert, L Baker, M Lai, A Bolton, et al. (2017). Mastering the game of Go without human knowledge. *Nature* **550**(7676), 354.

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

Talagala, TS, RJ Hyndman & G Athanasopoulos (2018). *Meta-learning how to forecast time series*. Working paper 6/18. Monash University, Department of Econometrics and Business Statistics.

Villani, M, R Kohn & P Giordani (2009). Regression density estimation using smooth adaptive Gaussian mixtures. *Journal of Econometrics* **153**(2), 155–173.

Vinod, HD, J López-de-Lacalle, et al. (2009). Maximum entropy bootstrap for time series: the meboot R package. *Journal of Statistical Software* **29**(5), 1–19.

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

Wong, CS & WK Li (2000). On a mixture autoregressive model. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **62**(1), 95–115.