Lars Kegel, Martin Hahmann, Wolfgang Lehner

Feature-based comparison and generation of time series

Erstveröffentlichung in / First published in:

SSDBM '18: Proceedings of the 30th International Conference on Scientific and Statistical Database Management, Bozen 09.-11.07.2018. ACM Digital Library, Art. Nr. 20. ISBN 978-1-4503-6505-5

DOI: https://doi.org/10.1145/3221269.3221293

Feature-based Comparison and Generation of Time Series

Lars Kegel[*]
Technische Universität Dresden
Dresden, Germany
lars.kegel@tu-dresden.de

Martin Hahmann
Technische Universität Dresden
Dresden, Germany
martin.hahmann@tu-dresden.de

Wolfgang Lehner
Technische Universität Dresden
Dresden, Germany
wolfgang.lehner@tu-dresden.de

ABSTRACT

For more than three decades, researchers have been developping generation methods for the weather, energy, and economic domain. These methods provide generated datasets for reasons like system evaluation and data availability. However, despite the variety of approaches, there is no comparative and cross-domain assessment of generation methods and their expressiveness. We present a similarity measure that analyzes generation methods regarding general time series features. By this means, users can compare generation methods and validate whether a generated dataset is considered similar to a given dataset. Moreover, we propose a feature-based generation method that evolves cross-domain time series datasets. This method outperforms other generation methods regarding the feature-based similarity.

CCS CONCEPTS

• **Mathematics of computing** → **Time series analysis**; • **Computing methodologies** → **Modeling and simulation**; • **Information systems** → *Similarity measures*;

KEYWORDS

time series generation, time series features, similarity measure

ACM Reference Format:
Lars Kegel, Martin Hahmann, and Wolfgang Lehner. 2018. Feature-based Comparison and Generation of Time Series. In *SSDBM '18: 30th International Conference on Scientific and Statistical Database Management, July 9–11, 2018, Bozen-Bolzano, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3221269.3221293

1 INTRODUCTION

Exhaustive data gathering and analytics are not novel trends anymore but can be considered standard practice in many domains. A major part of this practice deals with time series data from sensor measurements or other types of monitoring that form the prime data sources for automation and the "internet of things" vision. Thus, substantial amounts of research are spent on issues like time series storage, processing, analysis, forecasting, and many more. However, past decades have also produced research concerning time series generation. While this looks like a paradox considering the abundance of data that is collected, it is still a very important task for two major reasons: *system evaluation* and *data availability*.

System evaluation is a crucial phase in many industrial and organizational processes. It is done in order to test and verify component performance, to assess system robustness, and to estimate the correct sizing of necessary resources. Generated datasets are key to this phase by providing comparability and a variety of possible inputs for the systematic and thorough assessment of a system [6, 9, 19]. In addition, generated datasets can include user-given hypotheses. Thus, they allow users to study system behavior and assess risks in possible future scenarios [13, 23].

Data availability covers the many side conditions that accompany data gathering in general. Often, data might not be available in the required amount or quality. This can be due to very complex and expensive measurement procedures or because of sensitive and error-prone sensors. In addition to these technical issues, data availability is often restricted by legal regulations concerning privacy and intellectual property. With generated time series, the impact of these issues can be lessened, e.g. by compensating missing or erroneous data and by anonymizing confidential data [3, 8, 14, 16].

Time series generation has been applied in a multitude of domains. It is used for simulating weather parameters such as wind speed and direction, solar radiation, humidity, and rainfall precipitation [3, 9, 14, 16]. The energy domain utilizes it to assess renewable energy power plants [8, 9], while industry and economy apply generated time series for various evaluation purposes [6, 11, 15, 19].

While these approaches are valuable, they represent isolated solutions tailored to specific domains and applications. Thus, each approach employs an individual generation method based on individual time series characteristics arising from domain specifics. Concerning the importance of time series generation, this poses a challenge as it makes the topic difficult to access for new users that want to utilize it in their domain. In the worst case, an interested user would have to survey existing work before either adapting a method to his/her respective domain, or starting from scratch.

In this paper we address this issue by proposing a concept for cross-domain time series generation. We analyze existing generation methods in Section 2 regarding their properties and their expressiveness. In Section 3 we propose a feature set that represents a component-based time series model. Moreover, we describe a similarity measure based on this feature set. In Section 4, we introduce our feature-based time series generation approach. Our dataset-oriented approach not only allows the controlled generation of time series that abide by the key characteristics of their originating domain, but also provides a way to evolve new characteristics for hypothetical "what-if" scenarios. We compare the performance of our approach with existing generation methods in Section 5 before we conclude the paper in Section 6.

[*] The author is currently visiting Inria Saclay - Île-de-France, France.

DOI: https://doi.org/10.1145/3221269.3221293

Final edited form was published in "SSDBM '18: 30th International Conference on Scientific and Statistical Database Management. Bozen 2018", Art. Nr. 20, ISBN 978-1-4503-6505-5
https://doi.org/10.1145/3221269.3221293

2 STATE OF THE ART

We review cross-domain generation methods for time series from the literature and compare them regarding their *class*, their *properties*, and their *expressiveness*. Moreover, we summarize standard similarity measures that assess the accuracy of generated time series. Domain-dependent methods are out of scope because they take physical information of a given time series into account. Thus, they cannot be applied to others domains.

We distinguish between two classes: *generation methods with model* and *without model*. The former generates a time series from a model that captures information from a given dataset. The latter do not need this intermediate step and take the given data as input.

Beyond this classification, we characterize generation methods regarding four principal properties that express what they generate and how. It is desired that a generation method fulfills all of these properties in order to be comprehensive and to be widely applicable.

Finally, generation methods are reviewed regarding their expressiveness, i.e., the time series characteristics they are able to capture and to reproduce. We aim to derive characteristics that are relevant across domains and to include them in the feature-based similarity.

2.1 Properties of Generation Methods

We characterize generation methods with four properties that we define as follows.

Dataset-oriented. There are two different input scopes for a generation method: either it focuses on one time series at a time or on the whole dataset at a time. By processing each time series individually, the method reproduces the characteristics of only one sequence. Thus, it cannot take the full feature space of a dataset into account [8, 11]. Dataset-oriented methods take advantage of the feature space and they are able to take relationships of time series into account.

Deterministic. A deterministic generation method takes time series values as is. It reproduces deterministic characteristics of a time series which are the long-term trend and cyclical seasons. Random characteristics are shuffled or recombined [8, 19].

Stochastic. A stochastic generation method models the random characteristics of a time series. This model is then used to simulate new random values instead of taking the randomness as is [10, 18].

Innovative. There are methods that not only reflect given characteristics but also incorporate new characteristics in a generated dataset. Such innovative methods are important when it comes to inflate a dataset or to provide "what-if" scenarios [7, 12, 13].

Generation methods that fulfill all these properties are considered comprehensive since they are able to evolve time series for a variety of domains. Subsequently, we present generation methods with model and without model regarding these properties. They are summarized in Table 1.

2.2 Generation Methods with Model

Three generation methods with model are reviewed: *statistical models*, *Markov chains*, and *artificial neural networks*. They capture time series characteristics as scalar values such as, e.g., a series' variance or as a matrix such as, e.g., a transition probability matrix.

Statistical Models. Statistical models are generation methods that reproduce statistical and stochastic characteristics of a given time series. One example is a *random variable* that generates a sequence of independent and identically distributed values with a characteristic value distribution. Most often, such models generate normal, Weibull or Reighley distributions [10].

Besides the characteristic value distribution, the correlation between values of a sequence is important. This cannot be achieved by the sample of a probability distribution. *Autoregressive models*, $AR(p)$, are employed for this characteristic. They are of the form:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + w_t \qquad (1)$$

where y_t is an autoregressive process, p is the order, and $\phi_1, \ldots, \phi_p$ are the constants of the autoregressive model ($\phi_1 \neq 0$). The error component w_t is assumed to be normally distributed with mean 0 and variance σ_w^2. The correlation can be calculated from the constants such that a simulation of an autoregressive model exhibits an expected correlation within the generated sequence. Specifically for an $AR(1)$ model, the autocorrelation of lag 1 equals ϕ_1 [20]. Simulations from autoregressive models rely on a normally distributed error component w_t which is why the value distribution of the given data may be poorly represented. In summary, these models are stochastic but they are not deterministic, dataset-oriented, or innovative.

Markov Chains. With the aim to represent both, correlation and distribution characteristics, Markov chains are applied in numerous works. A Markov chain is a stochastic process with a fixed number of states. From one discrete time instance to another, the state may change whereas the probability of the new state only depends on one or more of the past states.

The Markov chain needs to capture the state transitions of a given time series. Therefore, the continuous time series values are first transformed to discrete states. Then, the transition probability for each combination of states is calculated by counting the occurrences of corresponding state transitions in the transformed time series which results in a transition probability matrix. The generation is carried out by simulating the state transitions with a sample of uniformly distributed values between 0 and 1. Finally, the resulting sequence of states is transformed back to real values.

This generation method is able to reproduce the distribution and correlation characteristics of a time series pretty well [4]. It has been used since the 1980s [9]. Since then, it was extended to better capture the autocorrelation of longer lags which is why Markov chains of second order were applied [10]. Recent research focuses on finding the best distance for the second lag in order to optimally fit a Markov chain to a given scenario [18]. The problem of Markov chains of orders higher than 2 is their increased complexity, therefore existing research is generally limited to this order. In summary, the Markov chain method is not dataset-oriented because it trains one transition probability matrix per time series. It is stochastic but not deterministic or innovative.

Artificial Neural Networks. Artificial neural networks (ANNs) are a family of models inspired by biological neural networks. They are universal approximators of functions with unknown type. Artificial

Final edited form was published in "SSDBM '18: 30th International Conference on Scientific and Statistical Database Management. Bozen 2018",
Art. Nr. 20, ISBN 978-1-4503-6505-5
https://doi.org/10.1145/3221269.3221293

neurons are inter-connected in order to transmit signals. The transmission activity depends on connection weights that are trained using given data.

While ANNs have been employed for time series generation, existing literature is scarce. Almonacid et al. [1] use them to learn characteristics from a dataset and to generate time series for new combinations of these characteristics. Regarding this reference, an ANN is dataset-oriented, deterministic and innovative. While ANNs in general are capable of evolving stochastic characteristics, the present method is not. Therefore, we classify it as non-stochastic.

2.3 Methods without Model

Generation methods without a model do not need an intermediate model to generate a dataset. They are based on *bootstrapping*, *modification*, *averaging*, *recombination*, or on selection and mutation with a *genetic algorithm.*

Bootstrapping. Bootstrapping splits a time series into intervals of the same length and permutes the values within these intervals [19]. The method can generate several new time series from one given time series. Obtained results stay similar to the original data as long as the chosen interval length is reasonably small. We conclude that bootstrapping is a deterministic but not stochastic method. It only focuses on one time series at a time and it is not innovative.

Modification. In a previous paper, we applied modification for generating a new dataset [12]. This method extracts features from a time series and applies a factor such that each feature is shifted to an expected target value. It is deterministic and innovative in that it systematically generates time series with new feature combinations. However, it is neither dataset-oriented nor stochastic.

Averaging. Averaging evolves new time series by averaging given time series from a dataset. It can be combined with varying weights in order to increase variety and in order to follow the characteristics of the given data [7]. This method is dataset-oriented, deterministic and innovative. However, since the random characteristics from the given time series are taken as is we consider this method non-stochastic.

Recombination. The recombination method first decomposes the values of time series into a long-term trend, cyclical seasons and residuals. Then it shuffles these components and finally recombines them in a new order. Time series generated in that way keep the characteristics of the original dataset but in new combinations.

Iftikhar et al. [8] brings time series recombination together with statistical models. The authors cluster components and shuffles them within their clusters. New residuals are simulated using an autoregressive model.

Overall, recombination is a dataset-oriented, deterministic generation method. It is innovative in that it creates new combinations of characteristics. However, it is not considered stochastic because the randomness in [8] stems from a statistical model.

Genetic Algorithm. The genetic algorithm generates time series by combining randomly selected given time series and ensures that the result is as close as possible to a given set of characteristics. Combination is carried out by selection, crossover, and mutation processes that occur with a specified probability. This method which was presented by Kang et al. [11] is considered a dataset-oriented and deterministic generation method. It is innovative in that it is able to evolve specific target characteristics. Mutation evolves new random values even though they are not modeled as in the statistical model.

Table 1: Properties of Generation Methods

	SM	MC	ANN	BT	MD	AV	RE	GA
Dataset-oriented	-	-	✓	-	-	✓	✓	✓
Deterministic	-	-	✓	✓	✓	✓	✓	✓
Stochastic	✓	✓	-	-	-	-	(✓)	(✓)
Innovative	-	-	✓	-	✓	✓	✓	✓

Statistical Model (SM), Markov Chain (MC), Artificial Neural Network (ANN), Bootstrapping (BT), Modification (MD), Averaging (AV), Recombination (RE), Genetic Algorithm (GA)

Table 1 summarizes the properties of the presented methods. It can be concluded that none of them fulfills all criteria that we judge important. Subsequently, we establish our generation method that focuses on all these criteria. Moreover, we compare it with recently used approaches, a Markov chain method [18], a recombination method [8], and a method based on the genetic algorithm [11].

2.4 Assessing Similarity

Generated datasets need to be assessed regarding their similarity to given datasets. The literature presents visual and numerical similarity measures in order to assess the expressiveness of generation methods.

Most often, the comparison of raw values is applied. It is carried out visually by providing line plots or scatter plots [1, 3, 4, 6, 8, 11, 14, 18]. In some cases, it is also assessed numerically [1, 3, 4, 19], most importantly with the root mean squared error measure. The RMSE is an absolute error measure which is defined as:

$$rmse(x_t, \tilde{x}_t) = \sqrt{\frac{1}{T} \cdot \sum\nolimits_{t=1}^{T} (x_t - \tilde{x}_t)^2} \tag{2}$$

It compares two discrete time series of length T where x_t and $\tilde{x}_t$ are the given and the generated value at time instance t. RMSE represents the mean distance between the values of the given and the generated series, whereas higher distances are more influential due to squaring. It is not tolerant to time series invariances: It cannot cope with scaling and translation, unless a time series is z-normalized. Invariance regarding shifts is not considered in this work which is why more sophisticated error measures such as dynamic time warping are out of scope.

A second measure is the value distribution which compares the frequency of occurrences of each value and thus, rather focuses on the value domain than on the time domain. Histograms are an appropriate visualization for this similarity, they plot the estimated probability distribution of given and generated time series values. Thus, they show the frequency of occurrences bucket-wise [1, 3, 9, 10, 18]. However, the literature on generation methods that numerically represent this similarity is scarce. For Markov chains, it can be shown that the generated time series nearly follow the probability distribution of the given time series [4]. However, the comparison of the distribution with a histogram distance or the comparison of statistical moments [17, 24] is not carried out.

Final edited form was published in "SSDBM '18: 30th International Conference on Scientific and Statistical Database Management. Bozen 2018",
Art. Nr. 20, ISBN 978-1-4503-6505-5
https://doi.org/10.1145/3221269.3221293

Finally, the autocorrelation is a property that has often been assessed [3, 4, 9, 10, 14, 18]. It represents the linear dependence of a time series on itself, shifted by a sequence of lags. For example, the autocorrelation of lag 1 is defined by:

$$acf_1(x_t) = \frac{\sum_{t=1}^{T-1}(x_{t+1} - \bar{x})(x_t - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2} \tag{3}$$

where $\bar{x}$ is the mean value of time series x_t. The autocorrelation can be visualized by a line plot whose x-axis is the sequence of lags and whose y-axis represents the corresponding autocorrelation. Thus, the autocorrelation of a generated time series is similar to a given time series if their lines are narrow to each other. These lines can be further assessed numerically by calculating their RMSE [4].

In our work we apply these three measures together (1) to assess whether a generated time series is similar to a given one and (2) to assess the expressiveness of generation methods. We combine them to the feature-based similarity that gathers the deterministic characteristics of a time series and the distribution and correlation characteristics of its residuals.

There are a number of other characteristics that may be captured such as, e.g., similarity in the frequency domain. We focus on the characteristics of the three presented measures since they are the most general and allow for comparison of generation methods from different domains.

3 FEATURE-BASED REPRESENTATION

In this section, we introduce the feature set for a component-based time series model along with the notion of feature-based similarity. We start by giving the definition of a time series dataset that we adopt in this work. Subsequently, we derive key features for the cross-domain representation of time series characteristics. Based on these features, we define the feature-based distance that describes the similarity of two time series in a dataset.

3.1 Prerequisites

Our goal is to provide a feature set for a component-based time series model. We therefore define a *time series* as follows:

Definition 1 (Time Series). A time series x_t is a sequence of values x that are measured at discrete time instances t:

$$[x_1, x_2, \ldots, x_t, \ldots, x_T] \text{ where } x_t \in \mathbb{R}, t \in \mathbb{N}_{>0}, t \leq T \tag{4}$$

The distance between two time instances is called *granularity*. We assume that a time series (1) is finite with a fixed length T, (2) is complete, i.e., there are no null values, and (3) is equidistant, i.e., the distance between two time instances is constant.

These three constraints keep the focus on time series similarity and generation methods. They assume that time series have been cleaned beforehand. In this work, we do not focus on this step.

Typically, time series data occurs in the form of datasets containing numerous individual time series that share a set of characteristics and can have mutual dependencies. Therefore, we define a *time series dataset* as follows:

Definition 2 (Time Series Dataset). A time series dataset X is a set of I time series:

$$X = \{x_{1,t}, x_{2,t}, \ldots, x_{i,t}, \ldots, x_{I,t}\} \tag{5}$$

where i is the identifier of a time series. All the time series have been measured at the same time instances. Thus, they have the same start and end time instance and they have the same granularity. For readability, we omit the indexes i and t when they are not needed.

Throughout the paper, we select three datasets for our demonstrations: an electricity consumption dataset, a wind speed dataset, and a dataset of macro- and micro-economic time series. They are called *Metering*, *Wind*, and *Economy*, respectively. Their characteristics are described in more detail in Section 5.

3.2 Time Series Components

Most time series from the aforementioned domains exhibit deterministic patterns. The wind speed is often stronger in winter than in other seasons, the solar irradiation has a strong daily season. Consequently, this season behavior arises in energy production series of renewables. In long-term studies trends can also be observed. If human behavior comes into play, time series exhibit other seasonal cycles. For example, weekly patterns can be observed in energy consumption due to a different behavior of consumer during weekdays and weekends. Economic time series may exhibit long term changes due to, for example, an increase in sales of a product. As a consequence, we argue that extracting these *components* from time series is important for their characterization.

A time series consists of *base*, *trend*, *season*, and *residual* components. The base is the long-term mean of the time series while the trend represents the long-term change of the mean. A season is behavior that is cyclically repeated. A time series can have several seasonal components with different *season lengths.* For the remainder of this paper, we refer to the base, trend, and season components as *deterministic components.* Residuals form the *stochastic component* of a time series. They are unstructured information that is usually assumed to be random. Together, these components describe the *time series model* which is adopted in this work.

Definition 3 (Time Series Model). A time series is a combination of components:

$$x_t = ba_t + tr_t + \sum\nolimits_{s=1}^{S} seas_{s,t} + res_t \tag{6}$$

where ba_t, tr_t, $seas_{s,t}$, and res_t are the base, trend, season, and residual component, respectively. The season length L_s, $1 \leq s \leq S$ as well as the number of seasons S is fixed for every time series dataset. We adopt an additive combination of components which is a common assumption in the aforementioned domains.

A decomposition technique extracts these components from a time series. Knowing the season length, it makes a non-unique split into trend, season, and residuals which can be further used for component analysis. Seasonal and Trend decomposition using Loess (*STL*) is a widely applied decomposition technique which is based on Loess smoothing, a locally weighted regression approach [5]. It is a versatile and robust decomposition technique which can handle every type of season length. Therefore, we adopt it for our multi-seasonal decomposition.

Multi-seasonal decomposition de-seasonalizes a time series from the shortest up to the longest season. As input, a time series, a list of season lengths, and a list of *season granularities* are provided. A season granularity A_s determines the aggregation of time instances

Final edited form was published in "SSDBM '18: 30th International Conference on Scientific and Statistical Database Management. Bozen 2018", Art. Nr. 20, ISBN 978-1-4503-6505-5
https://doi.org/10.1145/3221269.3221293

before the time series is de-seasonalized. For instance, a yearly season is best extracted with monthly values, so the season length is 12 and the season granularity is "month". The trend component is extracted normally.

We shortly explain our algorithm. First, the seasons are extracted from the time series one by one: (1) the time series is aggregated to the expected season granularity, (2) STL is used to decompose the object into trend, season, and residuals, (3) the season is extracted and (4) stored in the original granularity. From the remainder, this process is repeated with the longer seasons.

Second, the remaining deterministic components, base and trend, are extracted from the trend fit. STL yields a trend fit with local trend changes. We split this trend fit into a base component and a linear increasing trend component. Local trend changes are not part of our time series model. The residuals result from the subtraction of the extracted components and the given time series.

Figure 1 illustrates the multi-seasonal decomposition for a time series from the Metering dataset. Figure 1a shows the first 20 days of the time series in a half-hourly granularity. We can observe daily peaks in consumption, with less intensive consumption on the week-ends. We assume a half-hour season granularity for the extraction of the daily season, which results in the season component (Figure 1b). A clear daily pattern with higher consumption during the day and a small peak at noon can be identified. We further aggregate the remainder to a daily granularity and extract the weekly season (Figure 1c). The values show that consumption is higher during the weekdays than the week-end. Finally, we aggregate the time series to a monthly granularity and extract the yearly season (Figure 1d). Due to the short time interval of two years, the monthly values are rather fluctuating. Nevertheless, they show an increased energy consumption during the winter months. In addition, the decomposition also yields a base, a trend component, and a residual component (not shown).

3.3 Time Series Features

Decomposition separates characteristics of a time series and represents them as components. However, components still have the same length as the time series itself and they are not easier to handle. Therefore, they are reduced to features that are defined as follows:

Definition 4 (Feature, Feature Set). A feature is a mapping $f_k : \mathbb{R}^T \to \mathbb{R}$ that transforms a time series of length T into a scalar. A feature set $\{f_k(x_t), 1 \le k \le K\}$ is a short representation of a time series capturing its most important characteristics.

We represent the deterministic components with the *deterministic features*: *base value*, *trend slope*, and a *season mask* for every available season. Most of the meaningful information is extracted with the deterministic components. However, there is information remaining that cannot be described with a long-term or a cyclical expression. Thus, we consider this information as stochastic and provide features that capture their characteristics. The stochastic component is represented by *stochastic features* which are three *moments*: *standard deviation*, *skewness*, and *kurtosis* as well as the *autocorrelation of lag 1*. The *mean* of the residuals m is assumed to be 0 since it is extracted with the base component.

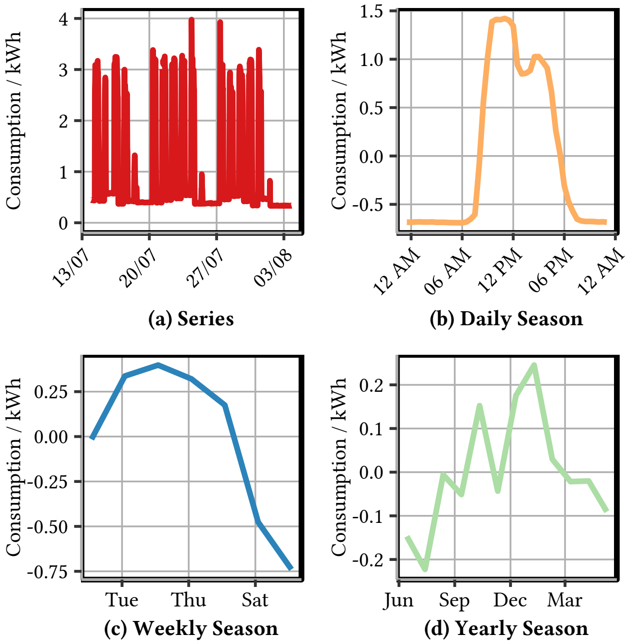


Figure 1: Multi-seasonal Decomposition

Base Value. The base value is the overall mean of the time series. We call this feature θ_1 such that:

$$\theta_1(x_t) = ba_1 \tag{7}$$

Trend Slope. The trend slope is the overall increase or decrease of a time series. In an attempt to represent the trend slope as a feature we assume a linear increase such that:

$$\theta_2(x_t) = tr_2 - tr_1 \tag{8}$$

Season Mask. The season mask represents the cyclical repeated behavior of the time series. It is a set of L_s features $\sigma_{s,l}(1 \le l \le L_s)$ such that:

$$\sigma_{s,l}(x_t) = seas_{s,l} \tag{9}$$

for all $1 \le l \le L_s$. The season mask is in line with the season granularity A_s. Since we assume a stable season, the season mask applies to the full time series. Figures 1b - 1d show the season masks for the daily, weekly, and yearly season.

Standard Deviation. The sample standard deviation is one of the most general statistical features. It is a measure of dispersion within the residuals and defined by:

$$sd(res_t) = \sqrt{\frac{1}{T-1} \cdot \sum\nolimits_{t=1}^{T} (res_t - m)^2} \tag{10}$$

A low standard deviation means that the values are very narrow to the mean, whereas a high standard deviation represents a wide spread of the data.

Skewness. The skewness is the third standardized moment and represents the degree of asymmetry of the residuals around the mean. Is is defined by:

$$skew(res_t) = \frac{T^{-1}\sum_{t=1}^{T}(res_t - m)^3}{(T^{-1}\sum_{t=1}^{T}(res_t - m)^2)^{3/2}} \tag{11}$$

The skewness ranges around 0: $skew = 0$ means that the residuals are symmetric. If $skew < 0$, then the left tail of the distribution is longer, i.e., the residuals are skewed to the left. If $skew > 0$, then the right tail is longer, i.e., the residuals are skewed to the right.

Kurtosis. The kurtosis is the fourth standardized moment. It represents the peakness or flatness relative to the probability density function of the normal distribution. The estimator of Pearson's kurtosis is defined as:

$$kurt(res_t) = \frac{T \cdot \sum_{t=1}^{T}(res_t - m)^4}{(\sum_{t=1}^{T}(res_t - m)^2)^2} \tag{12}$$

Kurtosis values range around 3: $kurt = 3$ means that the distribution of the residuals is as flat as the normal distribution, i.e., the tails of the distribution are as thin as the tails of the normal distribution. If $kurt < 3$, the distribution has a stronger peak and thinner tails. If $kurt > 3$, the distribution is more flat and has thicker tails.

Autocorrelation of Lag 1. The residual autocorrelation of lag 1 represents the linear relationship between the residual component with itself lagged by one time instance. It measures the linear predictability of the residual value res_{t+1} using only the value res_t. It is defined by:

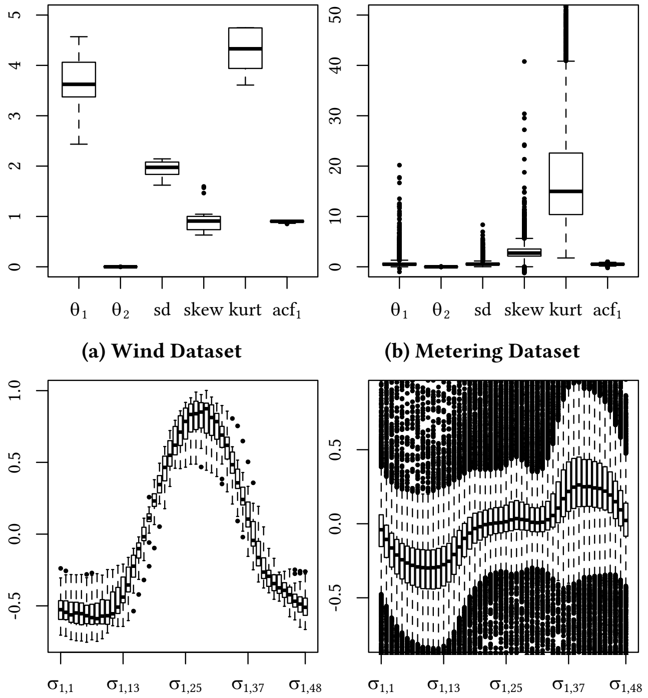
$$acf_1(res_t) = \frac{\sum_{t=1}^{T-1}(res_{t+1} - m)(res_t - m)}{\sum_{t=1}^{T}(res_t - m)^2} \tag{13}$$

Autocorrelation values ranges between -1 and +1: $acf_1 = 0$ means that there is no linear dependency between a value res_t and its successor res_{t+1}. If $acf_1 > 0$, then there is a positive linear dependency between the values. The maximum correlation, $acf_1 = 1$ means that res_t perfectly predicts res_{t+1}. If $acf_1 < 0$, there is a negative linear dependency between successive values: if res_t increases, res_{t+1} decreases. This leads also to perfect predictability, if $acf_1 = -1$.

The stochastic component has an important share in the overall signal. The feature set would lose expressiveness if it were assumed to be random and simulated with a normal distribution. This finding is also confirmed in the literature. Theodosiou [22] reported that autocorrelation remains in the residuals that should be modeled for better results. Modelers for wind speed generation report that Weibull distribution yields more realistic results than normal distribution because the given residuals are skewed and not symmetric [18]. The value distribution has been reduced to moment features by Nanopoulos in [17]. The autocorrelation of lag 1 has been used as a feature for time series generation in [11].

The presented features are calculated for each time series of the Wind and the Metering dataset. They are represented in Figure 2 as boxplots which show the range of every feature. The features of the Wind dataset have a small spread which confirms that the dataset is very homogeneous. The features of the Metering dataset (Figure 2b) is less homogeneous, the share of extreme outliers (black circles) ranges between 1 and 8%.

The season masks confirm this observation. While the daily season of the Wind dataset has a very clear shape (Figure 2c), the daily season of the Metering dataset is very fluctuating (Figure 2d) with a share of extreme outliers ranging between 4 and 11%. This fluctuation is mainly due to different electricity consumption behavior resulting in differently shaped season masks. The other season masks (weekly, yearly - not shown) also confirm this observation.



(a) Wind Dataset **(b) Metering Dataset**

(c) Daily Season Mask (Wind) **(d) Daily Season Mask (Metering)**

Figure 2: Unscaled Features

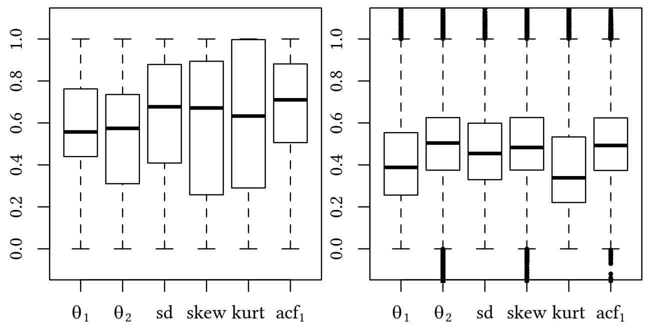
Not only are features used for representing a dataset. They also enable us to assess the expressiveness of a generation method. In the following subsection we define the feature-based distance that makes a given and a generated dataset comparable.

3.4 Feature-based Similarity

In order to compare relevant characteristics of a generated dataset and a given dataset, we propose a similarity measure based on features. It incorporates standard similarity measures (Section 2) to some extent: the deterministic features cover the raw-value shape of the time series while the stochastic features represent the histogram as well as the autocorrelation. Moreover, it is able to express an error threshold that defines the expectation of similarity.

To provide a way of characterizing an error threshold, features have to be scaled to a common range. This range has to (1) standardize the value range across different features and (2) diminish the influence of outliers compared to the most frequent feature values. The boxplot provides an intuitive notion of common values (represented as box), near outliers (whiskers) and extreme outliers (points below and above whiskers). We adopt this for our scaling.

Definition 5 (Scaled Feature). Let $F = \{f(x_i), 1 \leq i \leq I\}$ be a set of one feature value for a time series dataset. Let $Q_1(F)$ be the value of the lower quartile and $Q_3(F)$ the value of the upper quartile,



(a) Wind Speed Germany **(b) Smart Metering Project**

Figure 3: Scaled Features

respectively. Let $IQR(F) = Q_3(F) - Q_1(F)$ be the interquartile range. Then we define a scaled feature $f^s(x_i)$ based on:

$$\begin{aligned} lower(F) &= \underset{f(x_i)\in F}{\arg\max}\{f(x_i) \geq median(F) - 1.5 \cdot IQR(F)\} \\ upper(F) &= \underset{f(x_i)\in F}{\arg\min}\{f(x_i) \leq median(F) + 1.5 \cdot IQR(F)\} \end{aligned} \qquad (14)$$

as follows:

$$f^s(x_i) = (f(x_i) - lower(F))/(upper(F) - lower(F)) \qquad (15)$$

where f^s is the scaled feature that maps a time series to a feature value that mostly ranges between 0 and 1.

Figure 3 illustrates how features are shifted to the same range. The values we are focusing on, are between 0 and 1, the extreme outliers are below or above this range. Their share has not changed compared to the unscaled representation.

Based on the scaled features, we now define the *feature-based distance* that measures the similarity in the feature space:

Definition 6 (Feature-based Distance). We define the distance d_k of one feature f_k of two time series x_i and x_j as follows:

$$d_k(x_i, x_j) = \mid f_k^s(x_i) - f_k^s(x_j) \mid \qquad (16)$$

Thus, the feature-based distance of two time series x_i and x_j is the set of distances of every feature $f_k (1 \leq k \leq K)$.

If a generation method expresses a feature f well, the feature-based distance of a generated and a given time series will be close to 0, thus expressing a high similarity. If it does not express a feature well, this distance increases. It is either between 0 and 1, i.e., the feature value ranges in the usual feature scale (0 to 1) or it is close to it. If it is greater than 1, i.e., the feature value is not covered by the usual features of the given dataset.

Based on the feature-based distance, a user defines a *lower* and an *upper error threshold* p and *q*. The time series $x_i, x_j \in X$ are similar if every feature-based distance is within these error thresholds:

$$p \leq d_k(x_i, x_j) \leq q \text{ for all } k \; (1 \leq k \leq K) \qquad (17)$$

With this notion, the user configures the maximum distance that is considered as similar. This is an important notion for time series generation. While generation methods from the literature do not define an upper error threshold on standard distance measures, they can now be assessed by the feature-based distance.

Likewise, the user defines the lower error threshold p that two time series shall have. By default, this values is 0, thus ensuring perfect similarity. When this value is increased, a minimum deviation between two time series is expected which ensures that a generated time series is not fully equivalent to a given time series.

4 FEATURE-BASED GENERATION

The feature-based generation method described here is based on recombination and statistical modeling. Regarding these approaches, it is similar to Iftikhar et al. [8] but it extends some concepts in order to be more accurate with respect to feature-based similarity.

The method is cross-domain and addresses all required properties described in Section 2: (1) by recombining time series, it is dataset-oriented, (2) by reusing base, trend, and season components, it is deterministic, (3) by simulating residuals, it is stochastic, and (4) by relying on features that are modifiable, it is innovative.

The method generates time series that adhere to the features presented in Section 3. By taking thresholds into account, it allows the generation of time series with an expected similarity to the original data.

4.1 Recombination of Deterministic Components

The deterministic part of a time series det_t can be reconstructed by its features as follows:

$$det_t = \theta_1(x_t) + (t-1) \cdot \theta_2(x_t) + \sum\nolimits_{s=1}^{S} \sigma_{s,(t-1)\%L_s+1}(x_t) \qquad (18)$$

where % is the modulo operator. We used this relationship in order to draw *recombination candidates*. These candidates are similar to a given time series regarding their deterministic features. Thus, their recombination is also similar to the given time series.

The recombination comprised (1) the construction of a similarity matrix for each component, (2) a nearest neighbor search to identify recombination candidates, and (3) the recombination of components.

For every deterministic feature f_k, we calculated the feature-based distance $d_k(x_i, x_j)$ between every pair of given time series. These distances were then stored in a similarity matrix to identify neighboring components.

Subsequently, a nearest neighbor search was carried out on the similarity matrix to identify neighboring features that fulfill the error thresholds p and q. For every deterministic feature f_k, each given time series (identified by i) was annotated with a set of candidates (identified by j) that fulfill the condition $p \leq d_k(x_i, x_j) \leq q$.

Finally, we carried out the recombination of components as follows. From the recombination candidates, we randomly selected one candidate per feature, i.e., one candidate j_{ϕ_1} for the base value, one candidate j_{ϕ_2} for the trend slope and $\sum_{s=1}^{S} L_s$ candidates $j_{\sigma_{s,l}}$ for the season masks. Their recombination yielded the deterministic part $\tilde{det}_i$ of the generated time series $\tilde{x}_i$:

$$\begin{aligned} \tilde{det}_{i,t} = \; & \phi_1(x_{j_{\phi_1}}) + (t-1) \cdot \phi_2(x_{j_{\phi_2}}) \\ & + \sum\nolimits_{s=1}^{S} \sigma_{s,(t-1)\%L_s+1}(x_{j_{\sigma_{s,l}}}) \end{aligned} \qquad (19)$$

Figure 4 illustrates the feature space of the Economy dataset for two scaled features, base value and trend slope. We assume a lower error threshold $p = 0.10$ and an upper error threshold $q = 0.25$.

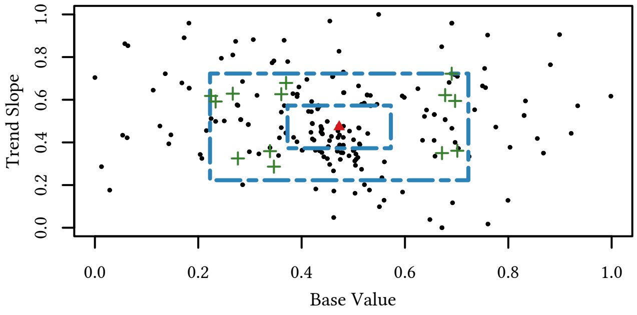


Figure 4: Feature Space of Economy Dataset

For an example time series (red triangle in the center), possible recombination candidates lie within the error thresholds (black points between the inner and the outer blue dashed rectangle). They are recombined and result in new combinations that still fulfill the feature-based similarity (green crosses). These recombinations are the deterministic part of generated time series.

If two recombinations contained the same components, they were considered as duplicates. If a recombination matched the components of a given time series, it was considered as original time series. Both anomalies were analyzed in order to make sure that they did not influence the results of the generation.

4.2 Simulation of Stochastic Component

The stochastic component does not consist of simple shapes such as trend and season. It includes the remaining random information that cannot be modeled directly. In order to describe this component, its value distribution and the remaining autocorrelation can be used. Our feature-based approach captures these characteristics with the features: standard deviation, skewness, kurtosis, and autocorrelation of lag 1.

To generate a component that agrees with these features, we applied a composite statistical model. It combines a distribution function that generates random values with a simulation of an autoregression model. Moreover, we took care that the generated residuals did not exceed the value limits from the given dataset by imposing constraints to the generated residuals. These three elements are explained subsequently.

Distribution Function. The goal is to generate residuals that agree with the expected moments. A distribution that fits this purpose is the *Pearson Distribution System*. It is a set of eight different distribution functions that cover a large space of distribution moments.

For each moment combination $\{sd, skew, kurt\}$, a distribution function has been selected based on its ability to provide a sample for these moments. The mean was assumed to be 0 since it was captured by the base value. For a given time series x_i and error thresholds p and q, a time series has been generated whose moments are expected to be in the interval below the given features, $[f_k^s(x_i) - q, f_k^s(x_i) - p]$, or the interval above the given features, $[f_k^s(x_i) + p, f_k^s(x_i) + q]$, where f_k^s is the scaled feature of standard deviation, skewness, and kurtosis.

Autoregressive Model. In the presence of significant autocorrelation, an autoregressive model was simulated. It weaved autocorrelation into the generated residuals so that the residuals followed both, the moments and the autocorrelation. As presented in Subsection 2.2, an $AR(1)$ model is of the form:

$$y_t = \phi_1 y_{t-1} + w_t \tag{20}$$

The autocorrelation of lag 1 equals ϕ_1. Thus, we generated residuals res_t for the expected feature $acf_1(res_t) = \phi_1$.

Since the error component w_t is normally distributed, the simulation is normally distributed too. Therefore, we transformed the simulation to the expected distribution that we defined above. The simulation was first normalized with a mean of 0 and a standard deviation of 1. Second, it was transformed to a uniform process using $z = N(y)$, where N is the cumulative distribution function (cdf) of the standard normal distribution. Third, by using the inverse of the distribution function F from the Pearson Distribution System, $\tilde{res} = F^{-1}(z)$ yielded a series that is similar to the moments of the cdf F and to the autocorrelation from the $AR(1)$ model.

Value Constraints. The simulation of an autoregressive model provides residuals with stochastic features. However, once the residuals are combined with the deterministic components they may lead to unexpectedly low or high time series values. For example, such combinations may lead to a negative or extremly high wind speed value. Since these results are unrealistic, they should be avoided.

Therefore, we defined two dataset wide *constraints* min_X and max_X that express the lowest possible and the highest possible value in a time series, respectively. The generation of residuals was aware of these constraints and of the deterministic part of the time series they were combined with. It ensured that evolved residuals never exceed these constraints as follows.

An error component w was generated which was a set of samples from a normal distribution. For a time instance t, a new value y_t from the autoregressive process was simulated using w_t. If the resulting residual value $\tilde{res}_t$ did not fulfill the conditions:

$$min_X - \tilde{det}_t \leq \tilde{res}_t \wedge \tilde{res}_t \leq max_X - \tilde{det}_t \tag{21}$$

it was rejected and another sample for w_t was drawn. The originally selected value w_t was used at another future time instance. Thus, it ensured the distribution properties of the sample.

5 EVALUATION AND DISCUSSION

We evaluate that (1) the feature-based generation method evolves a dataset that is highly similar to a given dataset regarding the feature-based distance, and (2) the feature-based distance is able to compare generation methods regarding their expressiveness.

5.1 Example Datasets

We evaluated our approach with three time series datasets from the energy, weather, and economic domain:

Example 1 (Metering). The Irish Commission for Energy Regulation initiated the Smart Metering Project in order to assess the performance of smart meters in Ireland [21]. The dataset contains the electricity consumption of households and small or medium businesses between July 2009 and December 2010. The consumption has been measured in kilowatt hour at a half-hour granularity.

Table 2: Example Datasets

Dataset	Dataset Size I	Time Instances	Length T
Metering	6089	2 years, half-hourly	35040
Wind	16	7 years, half-hourly	122736
Economy	217	11 years, monthly	132

The data is available in an anonymized form, indicating smart meter ID, timestamp, and consumption. All recorded time series show seasonal components (daily, weekly, yearly) but lack a strong trend. In order to extract the yearly season (which needs at least two cycles), six months (January 2010 until June 2010) have been appended to the end of the dataset.

Example 2 (Wind). To represent the weather domain, we selected a dataset that contains time series of wind speed from 16 German airports between 2010 and 2016. The wind speed has been measured in meters per second at a half-hour granularity. In order to fulfill Definition 2, gaps in the time series had to be interpolated. There are at most 2 % interpolated values per time series.

Example 3 (Economy). The M3-Competition is the third of four M-Competitions [15]. Its goal is the systematic evaluation of forecast method accuracy on a defined dataset. The dataset contains about 3000 time series from different domains (industry, finance, demographic, macro-/microeconomic, other). The values of each time series have a defined interval (year, quarter, month, other) and exhibit a trend and a seasonal component. In compliance with the time series dataset (Definition 2), we only selected time series with the same time interval and granularity. Thus, we focused on ca. 200 macro- and microeconomic time series from January 1983 to December 1993 with monthly granularity.

The Metering and Economy datasets have already been used for assessing generation methods by [8, 11]. The Wind dataset is similar to the dataset used in [18]. Table 2 recaps the dataset dimensions.

5.2 Experimental Setting

We compared the feature-based generation method (FBG) to three generation methods that have been recently cited in the literature (Table 3). For each example dataset X, each method generated one dataset $\tilde{X}$. For each time series $x_i \in X$, we generated one time series $\tilde{x}_i \in \tilde{X}$. The size of the generated dataset was equal to the size of the given dataset with one exception: only 100 time series were generated for the Metering dataset which is due to time limitations of the genetic algorithm. The comparison was carried out regarding similarity, the comparison on computational performance is deferred to future work. In order to reproduce the results, we shortly summarize the configuration of the selected methods.

Table 3: Experimental Setting

Generation Method	Dataset	Relevant Features
Recombination	Metering	All without θ_2
Markov chain	Wind Speed	Stochastic
Genetic algorithm	Economy	All
FBG	All	All

The generation method from Iftikhar et al. [8] is based on *recombination.* It splits a time series into a season mask, a base component, and a remainder. The season masks are then clustered, shuffled, and assigned to another base component and remainder from the same cluster. We carried out a k-means clustering of the daily season masks with the Euclidean distance measure. The number of clusters was 20 as proposed by the Iftikhar et al. We applied this setting to the Metering and the Economy dataset. We created only 5 clusters for the Wind dataset because it contains less time series. The residuals were generated with an AR(3) model as suggested by Iftikhar et al. Since the trend component was not considered by this method, we excluded the trend slope from the feature-based comparison and we added the original trend for comparing the raw-value distance.

A second comparison was carried out for *Markov chains*. The generation method of Pesch et al. [18] was re-implemented with some slight modifications. Each time series was decomposed into deterministic and stochastic components. A Markov chain of second order with 65 states (Metering, Wind) and a second order lag of 6 captured the state transitions of a given time series. These parameters correspond to the suggestion of Pesch et al. We only set 10 states for the Economy dataset because its time series are too short to fill a transition probability matrix for 65 states. In the rare case of anomalies in the transition probability matrix a healing mechanism was applied as suggested by Pesch et al. In contrast to [18], we adopted the multi-seasonal decomposition instead of the trend and season elimination that the authors applied. Moreover, we did not apply a running average filter for smoothing the time series because this characteristic is not relevant for this work. Generation has been carried out for the stochastic component only since it is not applicable to deterministic components.

Finally, we re-implemented the *genetic algorithm* based on Kang et al. [11]. As suggested by the authors, the initial population consisted of 20 time series (Metering, Economy) that were randomly selected. Its size was set to 10 for the Wind dataset. The time series x_i which sets the feature target for the generation of $\tilde{x}_i$ was not among the initial population. A crossover of time series occurred with a probability of 80%, a mutation occurred with a probability of 40%. Time series were selected by their fitness. We defined the fitness function as the RMSE of all K scaled features:

$$fitness(x_i, x_j) = \sqrt{\frac{1}{K} \cdot \sum_{k=1}^{K} (f_k^s(x_i) - f_k^s(x_j))^2} \tag{22}$$

The iteration stopped if (1) the fitness was at least -0.01, (2) if the maximum number of iterations, 3000, has been reached, or (3) if there was no improvement in a sequence of 200 iterations.

FBG was set to a lower error threshold $p = 0.01$ and an upper threshold $q = 0.05$ for all features and all experiments.

5.3 Feature-based Distance

The selected generation methods have been compared regarding the feature-based distance. First, they have been applied to the domain they were originally designed for, second they have been applied to all domains.

Recombination on Metering. Figure 5 compares FBG with the recombination method. The x-axis shows the feature f_k, the y-axis shows the feature-based distance $\{d_k(x_i, \tilde{x}_i), 1 \leq i \leq I\}$ as boxplot.

FBG generated time series whose deterministic components respected the error threshold q (Figure 5a). The base component also respected the error threshold p. However, there were rare cases where the season component of a given time series were not recombined which led to a feature-based distance below p. The recombination method generated time series with a higher feature-based distance to its given counterpart. While the base components (which were not recombined) yielded good results, the clustered and recombined daily season masks were more distant.

FBG also generated stochastic components whose features were similar to their given counterparts (Figure 5b). Although the moment features from FBG are remarkably good, they are not below the expected error threshold q. The simulation process had to select a trade-off between the expected features and the constraints. The recombination method used an AR simulation that generated values with a fixed standard deviation, skewness, and kurtosis. Thus, these features did not fit well the given moments of the original dataset. The autocorrelation could be well reproduced.

Markov Chain on Wind. The Markov chain method evolved only stochastic components. The generated dataset reproduced well the given features for the Wind dataset (Figure 6). FBG method yielded better results but the skewness is higher than for the Markov chain due to the trade-off with the constraints.

The feature-based distances seem high because the Wind dataset is very homogeneous. However, the unscaled distances are not.

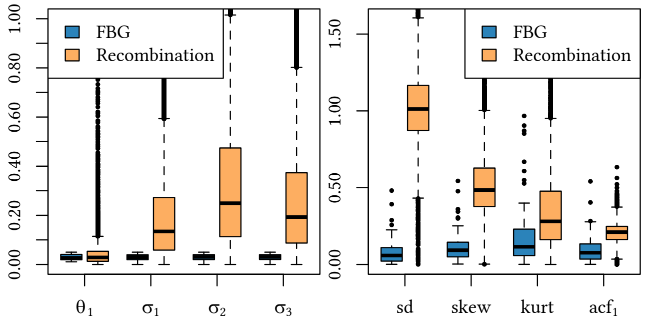
Genetic Algorithm on Economy. Figure 7 compares FBG with the genetic algorithm on the Economy dataset. For the deterministic components, FBG did not exceed the error threshold q (Figure 7a). In less than 8%, no recombination candidates were found which is why the component was identical to the given one. Thus, the lower whiskers reached 0 instead of stopping at p. The genetic algorithm generated highly similar components. The base value and trend slope had a median feature distance of 10% while the season masks differed from given season masks by only 2%.

Regarding the stochastic component, FBG provided more similar residuals in terms of standard deviation and autocorrelation. However, the genetic algorithm outperforms it regarding skewness and kurtosis (Figure 7b) for the same reason as in the experiments above.

Summary. Table 4a lists the median feature-based distance per dataset and method. The best distances are printed in boldface. FBG generally outperformed the other methods on the deterministic components due to the error thresholds. It partially yielded the best results on the stochastic components. However, Markov chains were sometimes better because they use a more comprehensive (but also bigger) model for generating residuals. The good results of the genetic algorithm on the Economy dataset were due to the 3000 iterations which led to best results on some of these features.

5.4 Standard Distance

The feature-based distance compares characteristics that generation methods are able to express. In this subsection, we show that this distance measure is related to the standard distance measures that we presented in Section 2: the *raw-value distance*, the *distribution distance*, and the *autocorrelation distance*.



(a) Deterministic Features **(b) Stochastic Features**

Figure 5: Feature-based Distance of Recombination

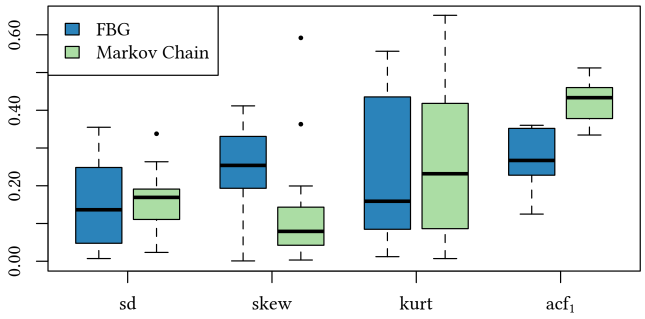
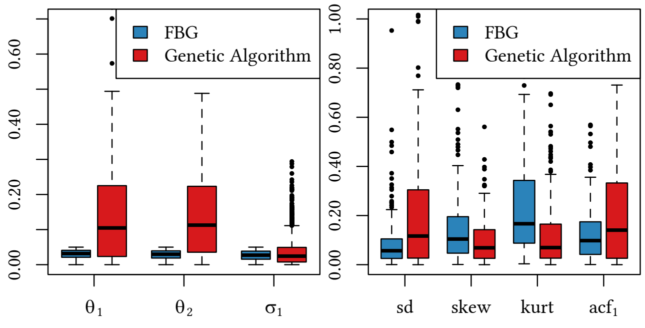


Figure 6: Feature-based Distance of Markov Chain



(a) Deterministic Features **(b) Stochastic Features**

Figure 7: Feature-based Distance of Genetic Algorithm

Raw-value Distance. Visually, the raw-value distance was assessed with a lineplot. Figure 8 shows the first four years of an exemplary time series from the Economy dataset (black solid line) together with the result from FBG (blue dashed line) and from the genetic algorithm (red dotted line). Due to the error thresholds and the strong deterministic components, the FBG result was very close to the original time series. The genetic algorithm had some peaks and lows due to mutation.

Final edited form was published in "SSDBM '18: 30th International Conference on Scientific and Statistical Database Management. Bozen 2018", Art. Nr. 20, ISBN 978-1-4503-6505-5
https://doi.org/10.1145/3221269.3221293

Table 4: Median Distances on All Datasets

(a) Feature-based Distances

Metering	θ_1	θ_2	σ_1	σ_2	σ_3	sd	$skew$	$kurt$	acf_1
RE	0.0285	-	0.1344	0.2492	0.1930	1.0120	0.4846	0.2805	0.2105
MC	-	-	-	-	-	**0.0388**	**0.0494**	**0.0691**	**0.0367**
GA	0.1145	-	0.0977	0.0880	0.1244	0.1025	0.2063	0.3159	0.1970
FBG	**0.0272**	-	**0.0296**	**0.0306**	**0.0306**	0.0582	0.0926	0.1151	0.0764
Wind	θ_1	θ_2	σ_1	σ_2	σ_3	sd	$skew$	$kurt$	acf_1
RE	0.0494	-	0.2703	0.5889	-	1.3999	2.2606	0.9540	1.9901
MC	-	-	-	-	-	0.1692	**0.0792**	0.2318	0.4334
GA	0.2078	0.2520	0.0749	0.1881	-	0.2469	0.1850	0.3778	0.6589
FBG	**0.0192**	**0.0190**	**0.0180**	**0.0184**	-	**0.1364**	0.2539	**0.1590**	**0.2670**
Economy	θ_1	θ_2	σ_1	σ_2	σ_3	sd	$skew$	$kurt$	acf_1
RE	-	-	0.2193	-	-	0.3517	0.1429	0.3325	0.2440
MC	-	-	-	-	-	0.0575	0.1245	0.1896	0.2663
GA	0.1047	0.1127	**0.0244**	-	-	0.1166	**0.0687**	**0.0696**	0.1406
FBG	**0.0299**	**0.0294**	0.0275	-	-	**0.0349**	0.1199	0.1548	**0.0886**

(b) Standard Distances

	Raw-value	Distribution	ACF
RE	1.7829	0.4471	0.2320
MC	0.7934	0.1478	**0.0183**
GA	0.8214	**0.0858**	0.1562
FBG	**0.7167**	0.1305	0.0492
Wind			
RE	3.4251	0.0676	0.1142
MC	2.7296	0.0242	0.0205
GA	**2.0618**	**0.0090**	0.0336
FBG	2.7559	0.0149	**0.0166**
Economy			
RE	**277.4685**	0.0790	0.0268
MC	335.3713	**0.0767**	0.0146
GA	690.3626	0.2520	0.0200
FBG	387.8705	0.0999	**0.0145**

Numerically, the raw-value distance was calculated as RMSE (Equation 2) since we took into account differences in scale, transformation and shift. Table 4b lists the median RMSE of a given and the corresponding generated time series per dataset and method.

On the Metering dataset, FBG was the most accurate method regarding raw-value distance. The Markov chain method ranked second. Although this method copied the deterministic components as is, it had a higher RMSE. One reason was that it did not take value constraints into account.

On the Wind dataset, the genetic algorithm yielded the highest accuracy. Due to the homogeneous data, every time series in the initial population was already highly similar to the time series that set the target features. The methods Markov chain and FBG returned a similar distance. While the Markov chain method copied the deterministic components, FBG recombined components with a lower error threshold that led to a slightly higher RMSE.

On the Economy dataset, the recombination method showed the lowest distance. Since this method did not exhibit a high similarity of monthly season masks (σ_1), the trend component that has been copied as is had a major influence on this accuracy.

Distribution Distance. The distribution distance focuses on the overall occurrence of values rather than the time domain. Visually, Figure 9 illustrates the histogram of an original Wind time series and the corresponding generated ones. The genetic algorithm, FBG, and the Markov chain generated values that were highly similar to the original ones. The recombination method was less similar because it did not take into account the higher order moments.

Numerically, the distribution distance was assessed with the *Bhattacharyya distance* [2]:

$$D_B(x_t, \tilde{x_t}) = -\log \sum\nolimits_{b=1}^{B} \sqrt{h_b(x_t) \cdot h_b(\tilde{x_t})} \qquad (23)$$

where b is one of B equidistant buckets and h_b is the relative frequency of values within the bucket. Equal distributions have a distance 0, the more dissimilar they are the higher is the distance. Table 4b lists the median Bhattacharyya distance for every dataset and generation method. We split the value range into $B = 100$ (Metering), $B = 30$ (Wind), and $B = 10$ (Economy), respectively.

On the Metering and Wind datasets, the genetic algorithm ranked first and FBG second. The Metering dataset contains several deterministic effects that are not captured by the component-based time series model. Thus, they are considered as residuals and are more difficult to reproduce.

The Wind dataset is very homogeneous. New combinations and mutations of its time series reproduced the value distribution very well. The Markov chain and recombination method yielded less accurate distribution distances which was mainly due to negative values that arose when generated residuals were stronger than the deterministic component.

On the Economy dataset however, the Markov chain and recombination yielded better results than the competitors. Since these methods took the strong trend component as is, they could better reproduce the original distribution than FBG and the genetic algorithm. Moreover, these time series were very short and thus, FBG was unable to accurately reproduce the value distribution.

Autocorrelation Distance. We assessed the unscaled autocorrelation of lag 1 of original and generated time series (Equation 3). Table 4b lists the median distance per dataset and method.

These results correspond to the scaled feature acf_1. FBG had the best results on the Wind and Economy dataset. The Markov chain had the smallest distances on the Metering dataset. On the Economy dataset, its results were only slightly worse than those of FBG.

Summary. By comparing the results from standard distance measures, we showed that FBG competed with state-of-the-art generation methods. It ranked first, second or third place on all the example datasets. The feature-based distance corresponded only

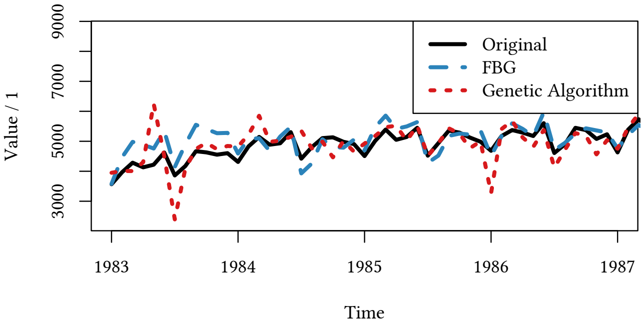


Figure 8: Lineplot of Genetic Algorithm

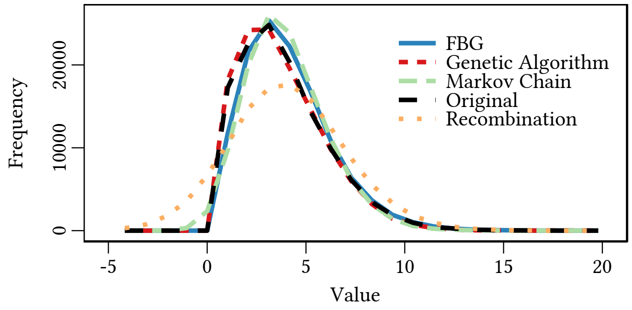


Figure 9: Histogram of Residuals on Wind Dataset

partially to standard distance measures from the literature. While the feature-based distance captures one specific characteristic of a time series, the standard approaches capture an overall distance. For example, the feature moments did not represent the full spectrum of a histogram which is why these features only correspond to the histogram in special cases.

6 CONCLUSION AND FUTURE WORK

In this work, we introduced a feature set which is an efficient representation for a component-based time series model. Subsequently, we showed that the feature-based distance compares generation methods from different domains regarding their expressiveness.

The feature-based generation FBG method takes these features into account and evolves highly similar time series. Moreover, it respects user-given similarity expectations which are expressed by the feature-based similarity and limiting error thresholds. By this means, it outperforms currently available generation methods.

Overall, this work is the first to present a cross-domain comparison of generation methods. This assessment is important in order to generalize generation methods from different domains that were considered as isolated applications but that could evolve datasets independent of their original application.

Feature-based similarity is a tool for the validation of generated datasets. Users can select time series by selecting those who fulfill their similarity expectations. In a future work, we will also validate generated datasets on extrinsic data-analysis tools such as classification in order to assess their applicability.

Other components can be added to the proposed time series model. Further research should also exploit the correlation between components which could improve the recombination.

In this work, we focused on statistical characteristics that have been presented in the literature of generation methods. Considering the automation in many domains, interesting features should also be derived automatically. In a future work, we will focus on learning features with, e.g., ANNs and compare them with our feature set.

ACKNOWLEDGMENTS

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731232.

REFERENCES

[1] F. Almonacid, P. Pérez-Higueras, P. Rodrigo, and L. Hontoria. 2013. Generation of ambient temperature hourly time series for some Spanish locations by artificial neural networks. *Renew. Energy* 51 (2013), 285 – 291.

[2] A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.* 35 (1943), 99 – 109.

[3] J. Bilbao, A. H. de Miguel, and H. D. Kambezidis. 2002. Air Temperature Model Evaluation in the North Mediterranean Belt Area. *J. Appl. Meteorol.* 41, 8 (2002), 872 – 884.

[4] K. Brokish and J. Kirtley. 2009. Pitfalls of Modeling Wind Power Using Markov Chains. In *IEEE PES*. 1 – 6.

[5] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *J. Off. Stat.* 6, 1 (1990), 3–73.

[6] M. A. Cuddihy, J. B. Drummond Jr., and D. J. Bourquin. 1994. Vehicle Crash Data Generator. (1994), 335 – 338 pages.

[7] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh. 2017. Generating synthetic time series to augment sparse datasets. In *Proc. of ICDM*.

[8] N. Iftikhar, X. Liu, S. Danalachi, F. E. Nordbjerg, and J. H. Vollesen. 2017. A Scalable Smart Meter Data Generator Using Spark. In *OTM*. 21 – 36.

[9] D. I. Jones and M. H. Lorenz. 1986. An Application of a Markov Chain Noise Model to Wind Generator Simulation. *Math. Comput. Simul.* 28, 5 (1986), 391 – 402.

[10] F. C. Kaminsky, R. H. Kirchhoff, C. Y. Syu, and J. F. Manwell. 1991. A Comparison of Alternative Approaches for the Synthetic Generation of a Wind Speed Time Series. *J. Sol. Energy Eng.* 113, 4 (1991), 280 – 289.

[11] Y. Kang, R. J. Hyndman, and K. Smith-Miles. 2017. Visualising forecasting algorithm performance using time series instance spaces. *Int. J. Forecast.* 33, 2 (2017), 345 – 358.

[12] L. Kegel, M. Hahmann, and W. Lehner. 2017. Feature-driven Time Series Generation. In *Proc. of GvDB*.

[13] L. Kegel, M. Hahmann, and W. Lehner. 2017. Generating What-If Scenarios for Time Series Data. In *Proc. of SSDBM*.

[14] K. M. Knight, S. A. Klein, and J. A. Duffie. 1991. A Methodology for the Synthesis of Hourly Weather Data. *Sol. Energy* 46, 2 (1991), 109 – 120.

[15] S. Makridakis and M. Hibon. 2000. The M3-Competition: results, conclusions and implications. *Int. J. Forecast.* 16, 4 (2000), 451 – 476.

[16] H. Müller and U. Haberlandt. 2015. Temporal Rainfall Disaggregation with a Cascade Model: From Single-Station Disaggregation to Spatial Rainfall. *J. Hydrol. Eng.* 20, 11 (2015).

[17] A. Nanopoulos, R. Alcock, and Y. Manolopoulos. 2001. Feature-based Classification of Time-series Data. *Int. J. Comput. Res.* 10 (2001), 49 – 61.

[18] T. Pesch, S. Schröders, H. J. Allelein, and J. F. Hake. 2015. A new Markov-chain-related statistical approach for modelling synthetic wind power time series. *New J. Phys.* 17, 5 (2015).

[19] J. Schaffner and T. Januschowski. 2013. Realistic tenant traces for enterprise DBaaS. In *Workshops Proc. of ICDE*. 29 – 35.

[20] R. H. Shumway and D. S. Stoffer. 2011. *Time Series Analysis and Its Applications*. Springer.

[21] The Commission for Energy Regulation. 2015. CER Smart Metering Project. (2015). www.ucd.ie/issda

[22] M. Theodosiou. 2011. Forecasting monthly and quarterly time series using STL decomposition. *Int. J. Forecast.* 27, 4 (2011), 1178 – 1195.

[23] A. H. C. van Paassen and Q. X. Luo. 2002. Weather data generator to study climate change on buildings. *Build. Serv. Eng. Res. Technol.* 23, 4 (2002), 251 – 258.

[24] X. Wang, K. A. Smith, and R. J. Hyndman. 2006. Characteristic-Based Clustering for Time Series Data. *Data Min. Knowl. Discov.* 13, 3 (2006), 335–364.