Martin Hahmann, Claudio Hartmann, Lars Kegel, Wolfgang Lehner

Large-Scale Time Series Analytics

Erstveröffentlichung in / First published in:

Datenbank-Spektrum. 2019. 19(1), S. 17–29. Springer. ISSN 1610-1995.

DOI: https://doi.org/10.1007/s13222-018-00304-5

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

Large-Scale Time Series Analytics

Novel Approaches for Generation and Prediction

Martin Hahmann[1] · Claudio Hartmann[1] · Lars Kegel[1] · Wolfgang Lehner[1]

Received: 4 October 2018 / Accepted: 27 December 2018 / Published online: 21 January 2019

Abstract
More and more data is gathered every day and time series are a major part of it. Due to the usefulness of this type of data, it is analyzed in many application domains. While there already exists a broad variety of methods for this task, there is still a lack of approaches that address new requirements brought up by large-scale time series data like cross-domain usage or compensation of missing data. In this paper, we address these issues, by presenting novel approaches for generating and forecasting large-scale time series data.

Keywords Data analytics · Time series generation · Time series forecasting · Big Data

1 Introduction

Big Data and the trend of extensive data gathering have been around for years, so we assume these subjects to be known and jump right to the core of this paper. Looking closer at what is actually collected under the label of Big Data, we find time series to be one of the most prominent data types. There are several reasons for this. First and foremost, it simply is pretty useful to monitor measurements over time. Furthermore, the general drive for digitalization places sensors in more and more areas of life, that all contribute to the growth of time series data. In this paper, we showcase two solutions that were developed in the Competence Center for Scalable Data Services and Solutions (ScaDS) [25]. Both originate from two different projects and contradict some of the most common Big Data assumptions: "We are drowning in data" and "More data is always better". We propose a feature-based approach for time series generation in Sect. 2. Although vast amounts of time series data do exist, they might not be available to everyone or lack some characteristics a user requires [18]. Our cross-domain approach allows the generation of time series datasets of any desired size and specific characteristics based on a real world dataset. Two of the main use cases for generated data are scaling experiments and the evaluation of so called what-if scenarios. In Sect. 3, we describe a cross-sectional forecasting approach which is specifically designed for the prediction of datasets that consists of a multitude of time series [8]. Our forecast technique represents many time series with one model and can predict their future behavior in short time. Furthermore, the specifically selected training data of many time series contributes to the model training which enables the compensation of noisy behavior and missing values which is crucial when forecasting Big Data time series.

Martin Hahmann
martin.hahmann@tu-dresden.de

✉ Claudio Hartmann
claudio.hartmann@tu-dresden.de

Lars Kegel
lars.kegel@tu-dresden.de

Wolfgang Lehner
wolfgang.lehner@tu-dresden.de

[1] Database Systems Group, Technische Universität Dresden, Dresden, Germany

2 Feature-based Comparison and Generation of Time Series

Considering the abundance of collected data, the idea of time series generation at first looks pretty paradoxical if not superfluous. However, there are two major reasons that make this idea very important: *system evaluation* and *data availability*.

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

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, 13, 26]. In addition, generated datasets can include user-given hypotheses. Thus, they allow users to study system behavior and assess risks in possible future scenarios [17, 22].

Data availability covers the many side conditions that accompany data gathering in general. While data is often available, it 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 [2, 11, 20].

Time series generation has been applied in a multitude of domains. It is used for simulating weather parameters such as wind speed and solar radiation, for the assessment of renewable energy power plants and further various evaluation purposes [2, 11, 13, 15, 19, 20].

Existing approaches typically 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 and costly to adapt to new domains.

We address this issue by proposing a feature-based concept for time series generation. For this, we first analyze existing methods for time series generation and derive a set of universal core properties from each generation method.

2.1 State-of-the-Art Time Series generation

We distinguish between two classes of generation approaches: *generation methods with model* and *without model*. The former capture information from a given dataset as a model, before generating time series from it. The latter only takes given time series as input. In general, generation methods share four principal properties that express what they generate and how. These are defined 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 only reproduces the characteristics of one sequence. Dataset-oriented methods utilize the full feature space and are able to take relationships of time series into account [11, 15].

Deterministic A deterministic generation method takes time series values as is. It reproduces deterministic characteristics of a time series such as long-term trend and cyclical seasons. Random characteristics are shuffled or recombined [11, 26].

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 [14, 23].

Innovative There are methods that do not only reflect given characteristics but also incorporate new characteristics in a generated dataset. Such innovative methods are important when it comes to inflating a dataset or providing "what-if" scenarios [7, 16, 17].

We reviewed three common generation methods with model. All of them capture time series characteristics as scalar values or small matrices. *Statistical Models (SM)* typically reproduce the value distribution of a given time series using normal, Weibull, or Reighley distributions [14]. In addition, correlations can be modeled with autoregressive models. *Markov chains (MC)* aim to represent correlation and distribution characteristics by capturing state transition probabilities from one discrete time instance to another. Generation is done by simulating these transitions with a set of probabilities, generated from a sample. *Artificial neural networks (ANN)* are universal function approximators and inter-connect artificial neurons in order to transmit signals. The transmission activity depends on connection weights that are trained using given data. With regard to our principle properties, ANNs are considered as dataset oriented, deterministic, and innovative. In contrast, SM and MC only fulfill the stochastic property.

As methods without model, we reviewed four approaches. *Bootstrapping (BT)* splits a time series into intervals of the same length and permutes the values within these intervals [26]. Obtained results stay similar to the original data as long as the chosen interval length is rea-

Table 1 Properties of generation methods

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

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

sonably small. The method only fulfills the deterministic property. *Modification (MD)* extracts features from a time series and applies a factor such that each feature is shifted to an expected target value [16]. Thus, the technique is deterministic and innovative. *Averaging (AV)* evolves new time series by averaging given time series from a dataset. It can be combined with varying weights in order to increase variety [7]. This method is dataset-oriented, deterministic and innovative. *Recombination (RE)* first decomposes the values of time series into a long-term trend, cyclical seasons and residuals. Then it shuffles these components and recombines them in a new order. Time series generated in that way keep the characteristics of the original dataset but in new combinations. Overall, recombination is a dataset-oriented and deterministic method that is able to innovate new combinations of characteristics. *Genetic Algorithms (GA)* generate time series by combining randomly selected given time series and ensure that the result is as close as possible to a given set of characteristics [15]. This method is considered as dataset-oriented, deterministic and innovative.

Table 1 summarizes the properties of the described generation methods. We can state that none of them fulfills all the criteria we defined as important.

2.2 Feature-based Representation

We introduce our feature set for a general time series model along with the notion of feature-based similarity. Our goal is to provide features for a unified time series representation that include time and measurement information as well as an identifier for each time series.

We define a *time series* as a sequence of measurements $x_t = [x_1, ..., x_T]$ ordered by their timestamp t. Consequently, a *time series dataset* X is a set of N time series $X = x_t^1, ..., x_t^N$. Typically, time series from a dataset share a set of characteristics.

We illustrate our approach with three time series datasets: a Smart Metering dataset [27], a wind speed dataset, and a dataset of macro- and micro-economic time series from the M3-Competition [19].

Time Series Components Most time series from the aforementioned domains exhibit deterministic patterns. Wind speed is often stronger in winter than in other seasons while solar irradiation has a strong daily season. Consequently, this seasonal behavior also arises in times series representing renewable energy production. In long-term studies, also trends can be observed. If human behavior is involved, time series can exhibit other seasonal cycles, e.g. weekly patterns in energy consumption due to a different behavior of consumers during weekdays and weekends. Economic time series may exhibit long term changes due to, e.g. 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* (*ba*), *trend* (*tr*), *season* (*seas*), and *residual* (*res*) 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 a cyclically repeated behavior. Individual time series can have several seasonal components with different cycle lengths L. For the remainder of this paper, we refer to the base, trend, and season components as *deterministic* components.

Residuals are *stochastic* components of time series, which represent unstructured information that is usually assumed to be random. These components describe the *time series model* which is adopted in this work.

Time Series Model A time series x_t is a combination of components:

$$x_t = ba_t + tr_t + \sum_{s=1}^{S} seas_{s,t} + res_t \quad (1)$$

We adopt an additive combination of components which is a common assumption in many domains. The *season length* $L_s, 1 \leq s \leq S$ as well as the number of seasons S is fixed for every dataset.

In order to extract these components from a time series, a decomposition technique is used. Knowing the season length, it performs non-unique splits into trend, season, and residuals which can be further used for component analysis. A widely applied decomposition technique proposed by Cleveland [5] is based on *Loess smoothing*, a locally weighted regression approach. It is a versatile and robust decomposition technique which can handle every type of season length. Therefore, we adopt it for multi-seasonal decomposition.

Fig. 1 illustrates the multi-seasonal decomposition for a time series from the Smart Metering Project dataset. Fig. 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 weekends. We assume a half-hour season granularity for the extraction of the daily season, which results in the season component shown in Fig. 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 which is shown in Fig. 1c. The values show that consumption is higher during the weekdays than the weekend. Finally, we aggregate the time series to a monthly granularity and extract the yearly season shown in Fig. 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

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

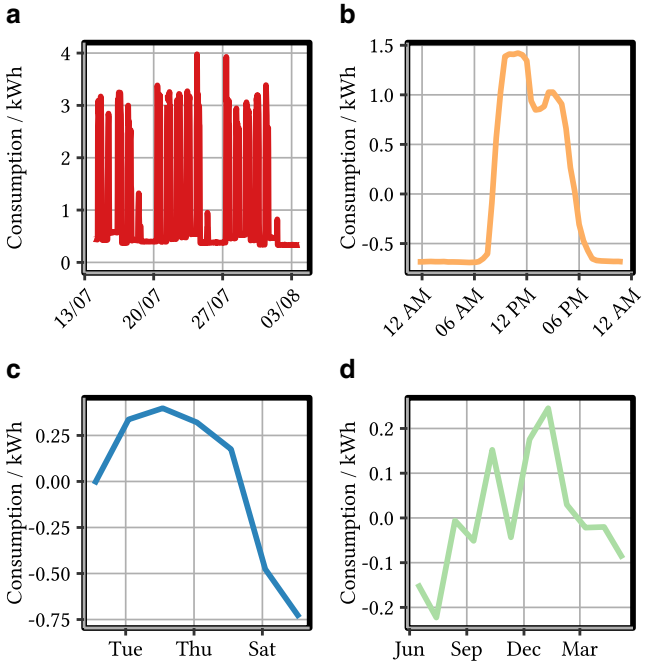

Fig. 1 Multi-seasonal decomposition. . **(a)** Series, **(b)** daily season, **(c)** weekly season, **(d)** yearly season

decomposition also yields a base, a trend component, and a residual component (not shown).

Time Series Features Time series components are further reduced to features $f_k (1 \leq k \leq K)$ that form a short representation of a time series capturing its most important characteristics. We represent the deterministic components with *base value* (θ_1), *trend slope* (θ_2), and a *season mask* ($\sigma_{s,l}$) for every season s. The stochastic component is represented by three *moments*: *standard deviation* (*sd*), *skewness* (*skew*), and *kurtosis* (*kurt*) as well as the *autocorrelation of lag 1* (acf_1).

The resulting vector of features is a representation of a time series that can be used in the assessment of similarity as well as the generation of time series. In the following, we define a feature-based distance measure.

Feature-based Similarity We propose a similarity measure that incorporates our features in order to assess the similarity of generated time series. To a certain degree, it combines traditional similarity measures like raw-value, histogram, and autocorrelation similarity. The deterministic features capture the raw-value shape of the time series, while the stochastic features represent the value distribution histogram as well as the autocorrelation. Using the feature similarity, the user can express an error threshold that defines his/her expectation of similarity during time series generation.

In order to define the *feature-based distance*, features have to be scaled to a common range first. This range has to (1) standardize the value range across different features and (2) diminish the influence of outliers compared to the most common feature values. The lowest datum still within 1.5-fold interquartile range (IQR) of the lower quartile is scaled to 0, while the highest datum still within 1.5-fold IQR of the upper quartile is scaled to 1. Based on the scaled features, the feature-based distance of two time series x^i and x^j is the vector of distances of every scaled feature:

$$f_k^s: \quad d_k\left(x^i, x^j\right) = \left| f_k^s\left(x^i\right) - f_k^s\left(x^j\right) \right|. \tag{2}$$

If a generation method expresses a feature well, the feature-based distance of a generated and a given time series will be near 0, thus expressing a high similarity. If it does not well express a feature, this distance increases. We can now describe an error threshold for time series generation that contains an upper and a lower threshold. Two time series are considered similar if every feature distance is within the upper bound q. With the lower threshold p, the user defines the minimum distance that two time series should have. A lower bound of zero allows perfect similarity. A higher value, ensures a minimum deviation between two time series which can be desirable during time series generation.

2.3 Feature-based Generation

Our proposed feature-based generation method is based on recombination and statistical modeling, similar to [11]. It extends some concepts in order to be more accurate with respect to feature-based similarity.

Our approach addresses all of the defined required properties: (1) by utilizing recombination, it focuses on a dataset rather than a single time series, (2) by relying only on statistical characteristics, it is cross-domain, (3) by simulating residuals, it generates a new stochastic component, and (4) by relying on modifiable features, it can evolve time series with new characteristics.

By taking user-given error thresholds into account, our approach allows the generation of time series with an expected similarity to the original data. For each time series $x^i \in X$, we generated one time series $\widetilde{x}^i \in \widetilde{X}$ with the expectation that their feature-based distance fulfills $p \leq d_k(x^i, \widetilde{x}^i) \leq q$ for every feature. Subsequently, we describe how the deterministic components and the stochastic component were processed.

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

$$det_t = \theta_1(x_t) + (t-1) \cdot \theta_2(x_t) + \sum_{s=1}^{S} \sigma_{s,(t-1)\%L_s+1}(x_t) \tag{3}$$

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

where % is the modulo operator. We used this relationship in order to draw *recombination candidates* that have similar features as a given time series and recombine them. For every deterministic feature, we calculate the feature-based distance for every pair of given time series. These distances are 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. For every deterministic feature, each given time series is annotated with a set of candidates. From the recombination candidates, one candidate per feature is randomly selected, i.e., one candidate for the base value, one candidate for the trend slope and $\sum_{s=1}^{S} L_s$ candidates for the season masks. Their recombination creates the deterministic part of the newly generated time series $\widetilde{x}_t$. If two recombinations contain the same components, they are considered as duplicates. If a recombination matches the components of a given time series, it is considered as original time series. Both anomalies must be monitored and should only occur in negligible amounts in order to make sure that they do not influence the results of the generation.

Simulation of Stochastic Component The stochastic component includes the remaining random information that cannot be modeled. 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.

To generate a component that agrees with these features, we use a composite statistical model. It combines a distribution function that generates random values with a simulation of an autoregression model. In the absence of significant autocorrelation the generation was carried out with the distribution function only. For the *distribution function* we utilize the Pearson Distribution System which contains a set of eight different distribution functions that cover a large space of distribution moments. For each moment combination, a distribution function is selected based on its ability to provide a sample for these moments. For a given time series x^i, a time series $\widetilde{x}^i$ is generated, whose moments are expected to be in the scaled feature range of standard deviation, skewness, and kurtosis. In the presence of significant autocorrelation, an autoregressive model is simulated to weave autocorrelation into the generated residuals. For this, an $AR(1)$ model is used. The simulation is transformed to the expected distribution that we defined above.

2.4 Evaluation

We evaluated our approach on the three already introduced time series datasets from the energy, weather, and economic domain. Our feature-based generation method (FBG) was compared to three generation methods that have been recently cited in the literature. Each method was used to generate a dataset for the example datasets. The size of the generated dataset was equal to the size of the original dataset. There was one exception for the Smart Metering data, where only 100 time series were generated due to the runtime costs of the genetic algorithm. To get a better understanding of the results we give a short description of the compared generation approaches.

The method from Iftikhar et al. [11] is based on recombination. It splits a time series into season mask, base component, and a remainder. The season masks are clustered, shuffled, and assigned to another base component and remainder from the same cluster. A trend component is not considered from this method which is why we exclude the trend feature from the further comparison.

A second comparison was carried out with an implementation of the Markov chains used by Pesch et al. [23]. Each time series was decomposed into determinstic and stochastic components. In contrast to [23], we adopted the multi-seasonal decomposition instead of the trend and season elimination that the authors applied. The evaluation only considers the stochastic component since the generation of deterministic components is not part of the original method.

We also re-implemented the genetic algorithm from Kang et al. [15]. The intial population consisted of 20 time series (Smart Metering, M3-Competition) and 10 (Wind Speed) that were randomly selected. It never contained the original time series which sets the feature target for the generated time series.

The feature-based generation method was set to a lower error threshold $p = 0.01$ and an upper threshold $q = 0.05$ for all features and all experiments. First, each generation methods was applied to the dataset from the referred domain, second it was applied to all domains.

Recombination on Smart Metering Dataset Fig. 2 compares FBG with the recombination method. The x-axis shows the feature f_k, the y-axis shows the feature-based

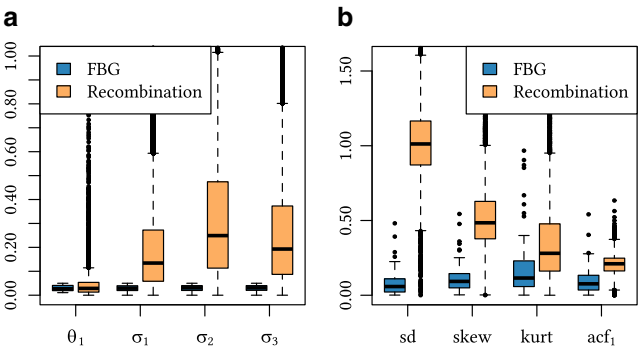

Fig. 2 Feature-based distance of recombination. **(a)** Deterministic features, **(b)** stochastic features

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

distance of the generated dataset as boxplot. For each generated time series the feature-based distance to its original time series was calculated. FBG only evolves time series whose deterministic components respect the error threshold q (Fig. 2a). The base component also respects the threshold p. However, there are rare cases where the season component of a given time series is not recombined, leading to an identical season. The recombination method generates series whose feature distances are higher to the given series.

The feature-based method also generates stochastic components whose features are similar to their given counterparts (Fig. 2b). Although the moment features from FBG are remarkably good, they are not below the expected error threshold q.

Markov Chain on Wind Speed Dataset The Markov chain only generates a stochastic component. As the feature distance shows, it reproduces the given features for the Wind dataset well (Fig. 3). The feature-based generation method yields better results but the skewness is higher than for the Markov chain.

Genetic Algorithm on M3-Competition Data Fig. 4 compares FBG with the genetic algorithm on the M3-Competition dataset. For the deterministic components, FBG does not exceed the user-given error threshold q (Fig. 4a). In less than 8%, there was no recombination candidate found which is why the component was identical with the given one. The genetic algorithm generated components that are highly similar to the given ones. The base value and trend slope have a median feature distance of 10 % while the season masks differ from given season masks by only 2 %.

Regarding the stochastic component, FBG provides more similar residuals in terms of standard deviation and autocorrelation. However, the genetic algorithm outperforms regarding skewness and kurtosis (Fig. 4b) for the same reason as in the experiments above. Moreover, due to the short series length, the generator for the random component deviates a lot from the expected moments.

Summary In general, FBG outperforms the other methods on the deterministic components due to the error thresholds. The stochastic features are sometimes better represented by Markov chains since they have a more comprehensive model for generating residuals. The good results of the genetic algorithm on the M3-Competition dataset are due to the 3000 iterations leading to the best results on some of these features. Most importantly, this evaluation shows that generation methods from different domains and applications are comparable regarding their expressiveness. Regarding the universality of our feature-based distance, we can state that it corresponds to the raw-value-based similarity measures from the literature. The four moments of a time series do not represent the full spectrum of a histogram which is why these features only correspond to the histogram in special cases.

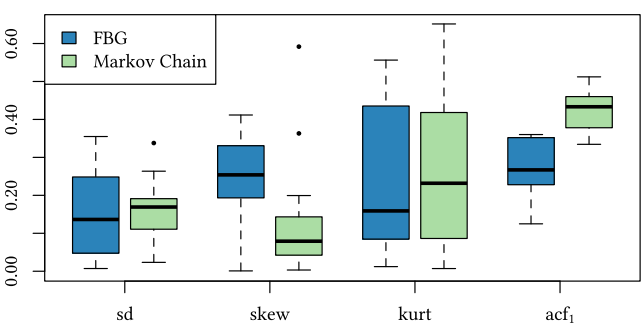

Fig. 3 Feature-based distance of Markov chain

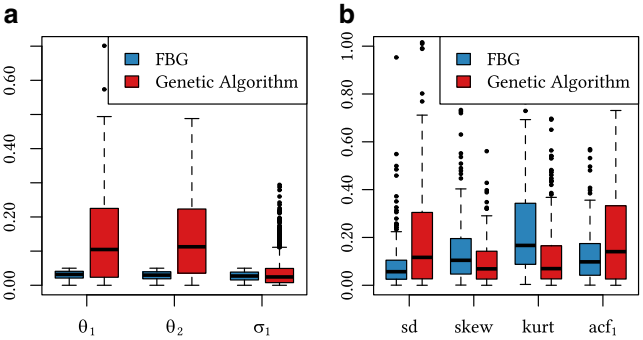

Fig. 4 Feature-based distance of genetic algorithm. **(a)** Deterministic features, **(b)** stochastic features

3 CSAR: The Cross-sectional Autoregression Model

In many application scenarios a lot of data is collected from multiple sources. This creates a large number of time series originating from the same domain which have to be analyzed, modeled, and forecast.

Usually, the task of forecasting focuses on only one time series x. For the forecasting, a model is optimized on the values of the time series to represent them as good as possible and then this model is used to calculate the requested forecast values [4]. Fig. 5 shows an example of a time series represented by the connected black crosses $\times$. The x-axis of the diagram denotes the time and the y-axis denotes the corresponding measure values. The red crosses mark the forecast values $\widehat{x}_{T+1}, ..., \widehat{x}_{T+h}$. The exact number of requested forecast values is called forecast horizon h. In the example three values are predicted $h = 3$. The prediction of long forecast horizons entails additional problems beyond the focus of this work. Therefore, we limit ourselves to one-step ahead forecasts with $h = 1$.

In order to produce accurate forecasts, a model has to address two very important characteristics of time series. First, the trend characteristic which sums up all long term

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

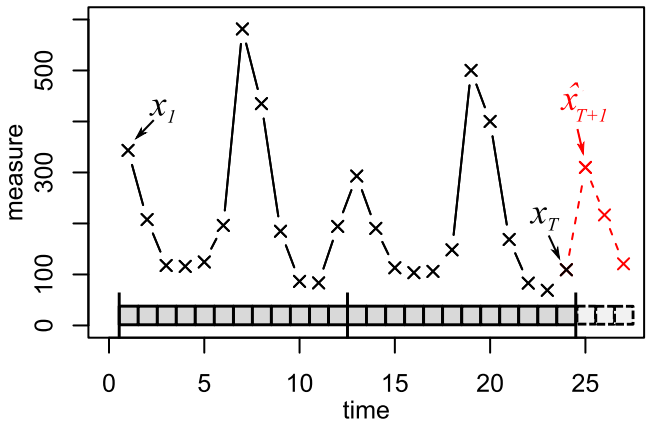

Fig. 5 Example time series and forecast with a forecast horizon of $h = 3$

changes without reoccurring patterns. Second, the seasonality which describes regularly reoccurring patterns within fixed intervals. The example time series in Fig. 5 has a seasonality with a season length of $s = 12$, which is recognizable at the reoccurring peaks, but lacks a clearly visible trend, e.g., a continuous rise or decline in the measured values. Between the time series and the x-axis of Fig. 5 there is a secondary representation of the same time series. Each square represents one time series value. Historical values are marked by gray squares with a solid contour, forecast values are marked by light gray squares with a dashed contour. In the remainder of this paper, we will use this representation to visualize how time series values are used to calculate forecasts.

Data collection has shifted towards more and more time series being recorded on increasingly fine structural and temporal granularities. This leads to new requirements for the forecasting process [9] which we illustrate in the following using the already introduced Smart Metering dataset as an example.

R1 – Numerous Series The high number of time series, makes the application of models that only predict one series at a time very difficult, as it requires the creation of a large number of models. The example dataset consists of more than 6000 individual time series which demand an equal number of forecast models. Training such an amount of models is a very time consuming task. Suitable prediction of large-scale time series datasets necessitates a modeling technique that provides forecast values for a large number of time series in reasonable time.

R2 – Incomplete Data Time series originate from a multitude of different data sources, e.g., Smart Meters of individual households and enterprises. Malfunctions during recording or data collection can lead to missing values and incomplete time series which many forecasting techniques are not able to work with. While only 5% of the overall data is missing in the example dataset, 29% of all time series have an incomplete history. For only a few time series, data can often be completed by applying imputation methods or searching for compensation values from similar time series[28]. However, these approaches become costly and often infeasible if datasets with thousands of time series are considered. Hence, large-scale forecasting techniques must provide accurate predictions despite incomplete data.

R3 – Increasingly Fine Granularity The increasingly fine structural and temporal granularity at which time series are monitored makes them prone to noise originating from external influences, e.g., the operation times of domestic appliances which are unique per household and will vary on a daily basis [21]. This makes modeling hard as time series behavior does not seem to be deterministic and describable. Therefore, a large-scale modeling technique has to compensate noisy behavior of time series.

While different existing approaches address one or two of these requirements, there is no approach that fulfills all of them. Therefore, we introduce our Cross-Sectional Autoregression model (CSAR) for large-scale time series forecasting.

3.1 CSAR-Modelling

In this section, we describe in detail how our CSAR model is working. It combines the qualities of cross-sectional forecasting [9] and ARIMA [3], thus CSAR meets all the requirements R1 to R3 and is adaptable to different datasets with their unique characteristics.

Cross-sectional forecasting is an approach that does not model individual time series in their entire temporal extent. Instead, so-called cross-sections are chosen that contain the measure values of all time series of a data set at a certain point in time [9]. In Fig. 6, the vertical rectangles colored in blue and red represent such cross-sections. A cross-sectional forecasting model represents future cross-sections based on a weighted sum of historical cross-sections. In doing so, one model represents an entire dataset and a fast forecast calculation for all time series is ensured. Furthermore, by using the data of many time series, noisy behavior and missing values can be compensated. Noisy data of individual series is covered by the data of others, such that there is no negative influence on the model training and the overall behavior of the source domain of the data is still properly represented. Incomplete series that cannot be modeled at all by other forecast techniques can still be forecast with the cross-sectional model that is trained on the other series of the dataset. Altogether, this leads to a significant improvement of the forecast accuracy compared to traditional forecast techniques with less execution time.

For the combination with cross-sectional forecasting the ARIMA model is a logical choice. It is a mature forecast technique and suits the paradigm of cross-sectional fore-

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

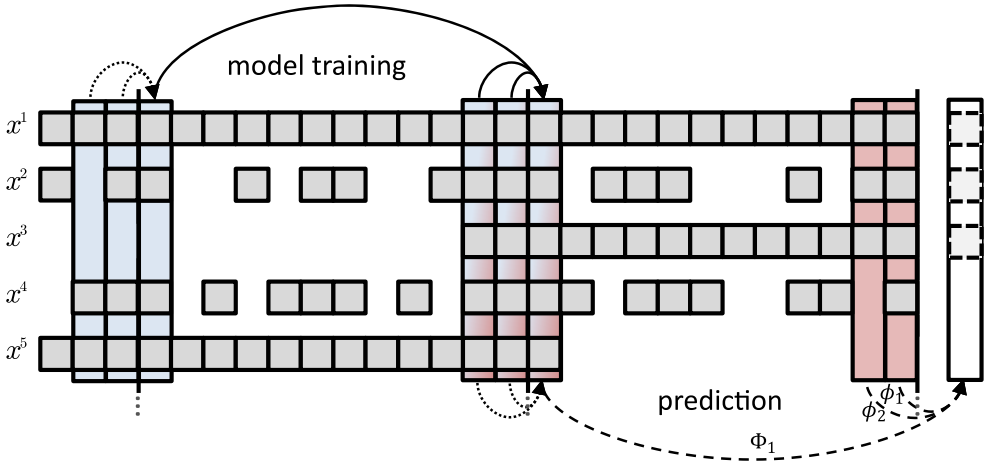

Fig. 6 CSAR model with one seasonal and two non-seasonal AR components

casting. We follow the structure of ARIMA [3] and highlight the adaptations we introduced to combine every individual component with the paradigm of cross-sectional forecasting. We begin with the integration component. It is a preparation step and removes static seasonal and trend components from the analyzed time series. Afterwards, we focus on the actual modeling components. The autoregression component calculates forecast values based on historical time series values. The moving average component is an error correction mechanism. Forecasts for historical time series values and the corresponding forecast errors are calculated. These error values are used to calculate the actual forecasts for the analyzed time series. In the CSAR model, these components serve a similar purpose.

Integration The integration component removes trend and seasonal characteristics from the time series and makes them stationary. Every time series is differentiated individually, such that the properties of each time series are preserved and not changed by the influence of other series. When the integration is used, every time series is differentiated first, then the dataset is forecast, and finally the series are integrated to obtain the forecast values for the original time series. As in the ARIMA model there is a distinction between a non-seasonal and a seasonal case.

The non-seasonal differentiation is used to eliminate trend characteristics. The first degree of numeric differentiation is shown in Eq. (4). The value x'_t of the differentiated time series is calculated as the difference of the original time series value x_t and an earlier value x_{t-d} divided by their distance d. In the usual case, when there are no missing values we set $d = 1$ and x'_t is calculated directly from the corresponding value x_t and its predecessor x_{t-1}. If there are one or more missing values directly before x_t then d is increased, such that the next available value x_{t-d} is used for the differentiation. The division by their distance is necessary to represent the trend change from one period to the next one.

$$x'_t = \frac{x_t - x_{t-d}}{d} \tag{4}$$

This kind of numeric differentiation is called backward differentiation since we are looking backwards from the current point in time t. Alternatives like forward differentiation $x'_t = (x_{t+d} - x_t)/d$ or a symmetrical approach $x'_t = (x_{t+d} - x_{t-d})/2d$ are not suited for the task of forecasting since the differentiation of the last value of the time series is not possible; this value is crucial for the forecast calculation in most if not all forecast methods. The absence of the first value as it is the case for the backward differentiation is not a problem if the time series is long enough to train a model without depending on this first value.

The seasonal differentiation is used to eliminate reoccurring seasonal patterns. Eq. (5) shows how x'_t is calculated by the difference of x_t and its corresponding value in a previous season $x_{t-D\cdot s}$. s is the seasonality of the dataset which is either known from the contextual information about the dataset or can be determined, e.g., using the auto-correlation function. When the value in the direct preseason is available we set $D = 1$, otherwise D is increased such that the next available corresponding seasonal value of x_t is used to calculate the differentiated value.

$$x'_t = x_t - x_{t-D\cdot s} \tag{5}$$

The seasonal differentiation does not need a division by the size of the gap that is bridged, since it is assumed that the seasonal behavior is stable over time and should be removed entirely.

Autoregression The autoregressive part of the CSAR model is a combination of the cross-sectional forecasting approach and the autoregressive part of ARIMA. It predicts all time series of a dataset based on their most recent historical observations. The autoregressive part of CSAR

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

consists of non-seasonal and seasonal components and the optimized weights are applied to cross-sections which span over all time series in the dataset. Hence, every time series is forecast based on a weighted sum of its own historical values while the model parameters (the weights) are optimized on all time series of the dataset which have the necessary historical data.

The Eqs. (6) and (7) show how the predictions are calculated in the non-seasonal and seasonal case. In the non-seasonal case (Eq. 6), the forecast values $\widehat{\vec{x}}_{t+1}$ are calculated as the weighted sum of their direct predecessor values $\vec{x}_t$ to $\vec{x}_{t-(p-1)}$ weighted with the model parameters ϕ_1 to ϕ_p. p denotes the number of non-seasonal autoregressive model components. $\vec{x}_t$ refers to the cross-section at time t which contains the historical values x_t^n of every individual time series x^n. Additionally, there is a constant part c which is also optimized during the model training and used for every time series. c can be excluded in order to fit the optimal model to a dataset.

In the seasonal case (Eq. 7), $\widehat{\vec{x}}_{t+1}$ is calculated by the corresponding seasonal historical values $\vec{x}_{t-s+1}$ to $\vec{x}_{t-P\cdot s+1}$ with a time distance of s periods. P is the number of seasonal autoregressive model components. The seasonal weights are represented by Φ_1 to Φ_P.

$$\widehat{\vec{x}}_{t+1} = c + \phi_1 \cdot \vec{x}_t + ... + \phi_p \cdot \vec{x}_{t-(p-1)} \tag{6}$$

$$\widehat{\vec{x}}_{t+1} = c + \Phi_1 \cdot \vec{x}_{t-s+1} + ... + \Phi_P \cdot \vec{x}_{t-P\cdot s+1} \tag{7}$$

Fig. 6 shows an example of five time series x^1 to x^5 with a season length of $s = 12$ which are predicted using the same CSAR model. The model shown in this example is comparable to an autoregressive (AR) model with two non-seasonal and one seasonal component. The difference is, that this model is not applied to only one individual time series but, following the idea of the cross-sectional forecasting approach, to cross-sections which are highlighted by vertical boxes that stretch over all time series. The solid arrows show how the involved cross-sections (highlighted in blue) contribute to the model training. The two short arrows represent the non-seasonal components of the model and the long arrow reaching back exactly one season represents the seasonal component. Every time series which has values in all involved cross-sections contributes to the model creation. Thus, the model represents how in average the target cross-section of the training data can be composed from the historical values for *all involved time series*. In the example, these are the time series x^1, x^4 and x^5. Series x^3 for example does not contribute to the model creation since it has no value for the seasonal component and the correction terms (leftmost three cross-sections colored in blue). The dotted arrows represent correction terms which are necessary due to the combination of seasonal and non-seasonal components. They subtract the direct predecessor values from $\vec{x}_{t-s+1}$ in the same way as the direct predecessors of $\widehat{\vec{x}}_{t+1}$ would add up in the forecast value. In doing so, the seasonal component only represents the actual seasonal change. There are always as many correction terms to every seasonal component as the model contains non-seasonal components and they are also mandatory for a time series in order to contribute to the model creation. The data for the model creation is still, as in the cross-sectional forecasting model, situated exactly one season before the model application which is represented by the dashed arrows in Fig. 6. The optimized model is now used to calculate the forecast values for the target period $t+1$. A forecast value can be calculated for every time series which has historical values in all involved cross-sections (highlighted in red). In the example, these are the time series x^1 to x^3. Series x^4 for example cannot be forecast because it has no values for the second non-seasonal component (leftmost cross-section entirely colored in red).

Eq. (8) shows the corresponding formula to the model of Fig. 6. Next to the constant c there are two non-seasonal components with the respective weights ϕ_1 and ϕ_2 and one seasonal component with its weight Φ_1 followed by the two corresponding correction terms.

$$\begin{aligned}\widehat{\vec{x}}_{t+1} = c &+ \phi_1 \cdot \vec{x}_t + \phi_2 \cdot \vec{x}_{t-1} + \Phi_1 \cdot \vec{x}_{t-s+1} \\ &+ (-\Phi_1\phi_1) \cdot \vec{x}_{t-s} + (-\Phi_1\phi_2) \cdot \vec{x}_{t-s-1}\end{aligned} \tag{8}$$

Considering this example, it becomes clear how a CSAR model is created on a multitude of time series like a cross-sectional forecast model but offers higher flexibility in the selection of the underlying data. The model keeps the positive properties of the cross-sectional forecasting model. This means, it is still possible to compensate for high levels of noise and to handle time series with missing values since the creation of a model does not solely depend on the historical data of only one time series. Please note, albeit CSAR can compensate for missing values a higher model complexity (more non-seasonal and/or seasonal model components) increases the risk of cases where missing values forbid the forecast calculation of individual time series when they occur in the base for the forecast calculation. Hence, for very sparse datasets a less complex model may lead to better forecasting results. Although it might not represent the dataset as good as possible, it can predict more of the incomplete series.

Error Terms Moving average components from the ARIMA model are not applicable in combination with a cross-sectional model where the core idea is that the same model parameters are used for all time series of a dataset. In contrast to the autoregressive components, the moving average does not rely on the most recent historical values but applies a smoothing process to the full history of a time series x. This is done by using error terms as shown

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

in Eq. (9) where the error e at time t is the difference of the original time series value x and the corresponding forecast $\widehat{x}$. Eq. (10) shows the calculation of a forecast value using moving average components. The forecast $\widehat{x}_{t+1}$ results from subtracting the error terms e_t to e_{t-q+1} from the constant part c, respectively from a forecast calculated by autoregressive components. Every error component is weighted with a corresponding parameter θ, q denotes the number of error terms which influence the current forecast.

$$e_t = x_t - \widehat{x}_t \tag{9}$$

$$\widehat{x}_{t+1} = c - \theta_1 \cdot e_t - ... - \theta_q \cdot e_{t-q} \tag{10}$$

Considering this calculation, it becomes clear that missing values make this approach impossible since the calculation of a forecast value depends on the *error terms of all historical values* of the time series.

There are already concepts available to apply smoothing techniques such as exponential smoothing to incomplete time series [1]. In the proposed solution, the next available historical predecessor value is used instead of the direct predecessor and its weight is lowered depending on how many missing values are bridged. Although, this solution could be transferred to the moving average part of the ARIMA model, the missing values of different time series are not evenly distributed in the dataset (see the example in Fig. 6) and, thus, an individual adaptation of the model parameters for every time series with missing values is required. This contrasts the core idea of the cross-sectional forecasting approach to train a single model with one single set of parameters for a multitude of time series.

For the CSAR model we introduce an alternative way to incorporate the error terms into the forecast calculation. Instead of applying the moving average function of Eq. (10) we use the average of the error terms of each individual time series of the dataset and include these into the forecast calculation. Eq. (11) shows how the average error $\overline{e^n_{t+1}}$ for time series n at time t is calculated. The first sum collects the non-seasonal forecast errors for the periods directly prior to period t. The second sum collects all seasonal forecast errors which are situated exactly one or more full seasons prior to t. The error terms of the individual time series are summed up and divided by the number of non-seasonal f and seasonal error terms F. If a time series misses values to calculate either forecast or error these specific values are neglected during the error calculation and f or F are lowered accordingly for this time series. Finally, Eq. (12) shows how the error is incorporated into the forecast calculation by subtracting it from the constant part c or the corresponding forecast calculated by a CSAR model without error terms.

$$\overline{e^n_{t+1}} = \frac{1}{f+F}\left(\sum_{i=1}^{f} e^n_{t-i} + \sum_{j=1}^{F} e^n_{t-j\cdot s}\right) \tag{11}$$

$$\widehat{x}_{n,t+1} = c - \overline{e^n_{t+1}} \tag{12}$$

In this way, it is possible to compensate the forecast errors for time series which are systematically mispredicted. Actually, we have assumed, that a high number of error terms would be necessary to obtain a reliable error component. As the evaluation in the next section will show, this is not the case and a few error terms already lead to improvements of the forecast accuracy.

3.2 Evaluation

We conduct an experimental study to evaluate the accuracy and execution time of our CSAR model. CSAR is implemented in the statistical computing environment R [24]. The experiments are executed on a six-core AMD Opteron(tm) server with 32 GB of RAM. For the evaluation we use two real world datasets:

Smart Metering The first dataset is the already mentioned Irish Smart Meter dataset. We use the last complete week of data as evaluation part. For this dataset we executed our experiments on different time granularities from 30 min to daily energy consumption. Due to space limitations, we only present the 6 hour granularity in this paper since this shows the effects we want to emphasize during the evaluation best.

Payment The second dataset is taken from the IJCAI-2017 Data Mining Contest [12]. It consists of payment transactions of 2000 distinct shops in daily granularity monitored over 494 days. We use the number of payments per shop and day of the last 14 days as evaluation period. None of the time series has a complete history which means that every time series is either not monitored right from the beginning or has missing values in its history. Compared to a complete dataset, 40% of data is missing.

Forecast Accuracy In the first experiment, we evaluate the accuracy of our forecasting approach on two different aggregation levels per dataset. We begin with the top aggregation level where all base time series are aggregated only grouped by the time. This represents for example the overall energy consumption of all households and enterprises in the Energy dataset. Afterwards, we analyze the base aggregation level where each base time series is evaluated individually. As comparison methods we use the ARIMA model as implemented in the auto.arima-function of the forecast package of R [10] and the cross-sectional forecasting model (CS) as presented in [9]. To

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

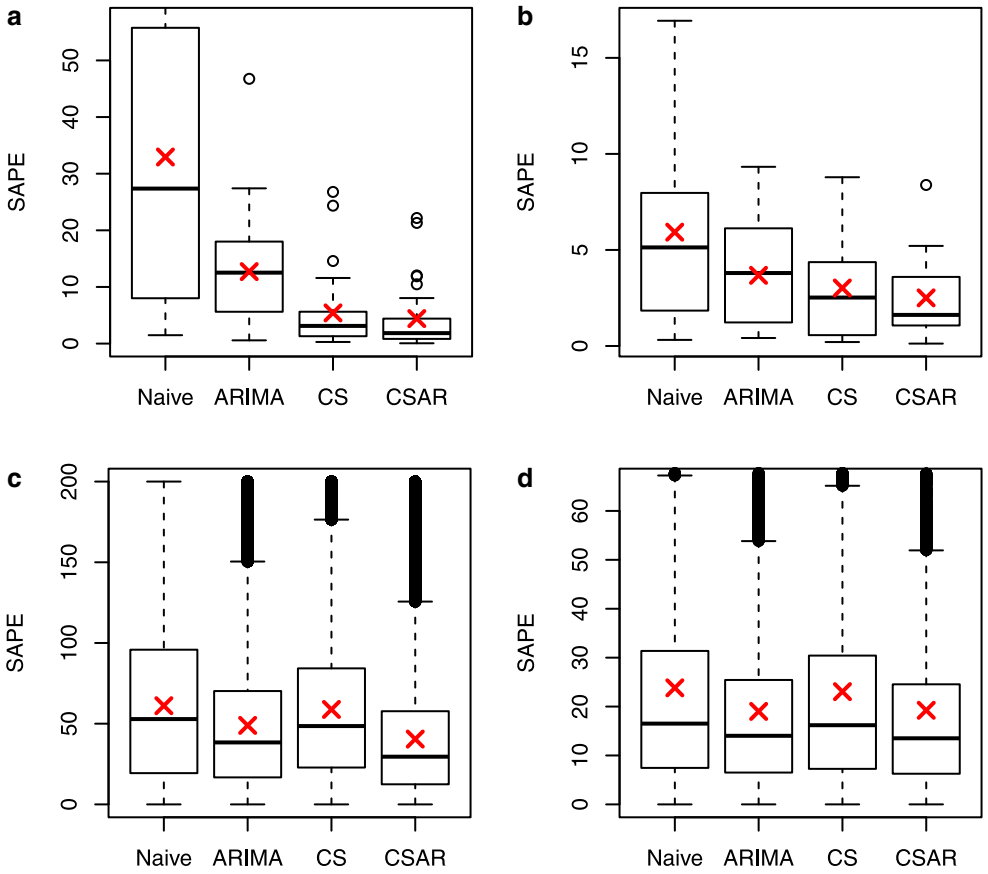

Fig. 7 Forecast error on top and base aggregation level. **(a)** Energy—top level, **(b)** payment—top level, **(c)** energy—base level, **(d)** payment—base level

enable the application of ARIMA, we filled missing values with zero values. CS is represented by a CSAR model with only one non-seasonal autoregressive component and the constant part. Additionally, we include the native forecast where every predicted period shows the same value as its predecessor $\widehat{x}^n_{t+1} = x^n_t$. This states our baseline. Any forecasting technique performing worse, is not suited for the specific dataset. For a comparison with other forecasting techniques, i.e., Triple Exponential Smoothing, Vector Autoregression, Croston's Method, and Hierarchical Forecasting, please refer to the evaluation of [9].

The ARIMA model is always applied at the evaluation aggregation level, which means the data is aggregated first and then forecast. CS and CSAR are applied on the base level and the forecasts are aggregated afterwards to obtain the top level. The optimal metaparameters for CSAR, i.e., number of seasonal and non-seasonal autoregressive components and error terms as well as the degree of integration, were optimized manually. The model parameters, i.e., the weights of the autoregressive model components and the constant c, were optimized using the optim-function of R.

All datasets are divided into a training and an evaluation part as mentioned in the dataset description. All data preceding the evaluation part may be used for the model training. We apply a rolling forecast and create a new model for every period t in the evaluation part of each dataset to calculate the forecast values. Then, we compare the forecasts to the corresponding time series values and calculate the forecast error with the SAPE measure (Symmetric Absolute Percentage Error):

$$\text{SAPE} = \frac{|x - \widehat{x}|}{(|x| + |\widehat{x}|)/2} \cdot 100, \tag{13}$$

x denotes the real time series value and $\widehat{x}$ is the corresponding forecast of one of the evaluated techniques. If the time series value and the corresponding forecast both equal zero we assume a forecast error of zero. Values in the evaluation part that a method was not able to predict are filled with a zero forecast and, therefore, punished with a maximum error.

The results of this experiment are shown in Fig. 7. Each diagram presents the forecast errors for one dataset and aggregation level as a Box-Whisker-Plot. The y-axis denotes the SAPE forecast error. Each box represents the forecast errors of one forecast technique. The red cross in each box denotes the corresponding average error.

The first two diagrams (Fig. 7a–b) show the results of the top aggregation level. Our new CSAR model (rightmost box) performs best on both datasets. For the Payment dataset all approaches perform well since the number of

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

Table 2 Comparison of execution times

CSAR					ARIMA
0,1	1,0	0,2	1,1	2,2	
0.4 s	0.4 s	0.8 s	0.9 s	5.9 s	42 min 38 s

overall payments does not fluctuate very strong on a daily basis. For the energy dataset the native forecast performs significantly worse since there is a strong seasonality which this approach cannot model. The cross-sectional forecasting model performs better than the ARIMA model because it incorporates the knowledge of all base time series which leads to a better representation of the overall datasets. CSAR performs even better and profits from the higher adaptability.

The results for the base aggregation level are shown in the Fig. 7c–d. As each time series is evaluated individually, it is much harder to achieve a high accuracy. The diagrams show again that the adaptability of the CSAR model leads to the most accurate forecasting results. The forecast errors of CSAR are the overall best for the energy data and on par with the best comparison method for the Payment data. For Energy and Payment the time series on the base level still have some predictable properties. This leads to acceptable results for ARIMA which is only outperformed by CSAR.

In summary, our CSAR model performs best for both datasets without requiring a manual preparation of the data or any missing value treatment. Thus, of all compared techniques CSAR is suited best to derive accurate forecasts for noisy and incomplete datasets and satisfies the requirements R2 and R3.

Execution Time In the second experiment we evaluate the execution time for different complexities of CSAR and compare them with those of ARIMA. Using the set-up of the previous experiments, we calculate forecasts for the base level of the Energy dataset and monitor the execution time for the prediction of one one-step ahead forecast for all time series of the dataset. We execute the experiment ten times and use the average time of all ten passes for the comparison.

The results of the experiment are presented in Table 2. The first five columns show different complexities of our CSAR model using the notation: number of *seasonal, non-seasonal* parameters. First of all, there is no difference between the execution times for seasonal and non-seasonal models. They access the same amount of data and have to optimize the same number of parameters. Hence, I/O cost and computation times are quite similar. The addition of more model parameters increases the execution time by making model training and parameter optimization more costly due to an increased amount of data that has to be accessed. Combining seasonal and non-seasonal components further increases the execution time since the correction terms have to be taken into account (ref. Sect. 3.1). A further increase of the number of model components leads to a higher execution time with super linear growth.

The sixth column shows the execution time of ARIMA. Note, the execution time of the ARIMA model was measured on ten individual time series and the overall execution time was extrapolated using the number of time series in the dataset. Moreover, we did not use the auto.arima function which includes the search for the optimal metaparameters and would have lead to a much higher execution time. Compared to the ARIMA model CSAR has a significantly lower execution time, as only one model is created for an entire dataset. For the Energy dataset on its original 30min granularity ARIMA is not even able to provide forecasts in time since the execution time exceeds the monitoring granularity by more than 40%.

Therefore, we can show that CSAR meets the requirement R1 and in combination with the results from the first experiment satisfies all the requirements (R1 to R3) on the prediction of large-scale time series datasets.

4 Conclusion

In this paper, we presented two unconventional approaches for the analysis of large-scale time series data. Both highlight novel ways to address and overcome some of the challenges that the Big Data trend poses on time series analytics. Our feature-based generation method enables more exhaustive and more meaningful evaluations for systems that process time series data in a large scale by providing data sets which are tailor-made for the specific use case. In addition, it offers an approach for characterization and comparison of time series datasets allowing more insights and cross-domain application. The CSAR model for forecasting provides highly accurate results in a short amount of time while overcoming noise and missing data, two of the major challenges in large-scale time series analytics.

Funding This work was funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions Phase 1—ScaDS Dresden/ Leipzig (BMBF 01IS14014A).

References

1. Aldrin M, Damsleth E (1989) Forecasting non-seasonal time series with missing observations. J Forecast 8(2):97–116
2. Bilbao J, de Miguel AH, Kambezidis HD (2002) Air temperature model evaluation in the north mediterranean belt area. J Appl Meteorol 41(8):872–884
3. Box GEP, Jenkins GM, Reinsel GC (2008) Time series analysis: forecasting and control. Wiley, Oxford
4. Chatfield C (2000) Time-series forecasting. Chapman & Hall, CRC, New York. ISBN: 9781584880639

Final edited form was published in "Datenbank Spektrum". 19 (1), S. 17-29. ISSN: 1610-1995.
https://doi.org/10.1016/j.intacc.2008.04.005

5. Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) STL: a seasonal-trend decomposition procedure based on loess. J Off Stat 6(1):3–73
6. Cuddihy MA, Drummond JB Jr., Bourquin DJ (1994) Vehicle crash data generator. US patent/grant: US5608629A
7. Forestier G, Petitjean F, Dau HA, Webb GI, Keogh E (2017) Generating synthetic time series to augment sparse datasets. Proc ICDM. https://doi.org/10.1109/ICDM.2017.106
8. Hartmann C, Hahmann M, Habich D, Lehner W (2017) CSAR: the cross-sectional Autoregression model. Proc DSAA. https://doi.org/10.1109/DSAA.2017.27
9. Hartmann C, Hahmann M, Rosenthal F, Lehner W (2015) Exploiting big data in time series forecasting: a cross-sectional approach. Proc DSAA. https://doi.org/10.1109/DSAA.2015.7344786
10. Hyndman RJ, Khandakar Y (2008) Automatic time series for forecasting: the forecast package for R. J Stat Softw. https://doi.org/10.18637/jss.v027.i03
11. Iftikhar N, Liu X, Danalachi S, Nordbjerg FE, Vollesen JH (2017) A Scalable Smart Meter Data Generator Using Spark. OTM. https://doi.org/10.1007/978-3-319-69462-7_2
12. IJCAI (2017) IJCAI 2017 – Data Mining Contest. http://tb.am/s0a3o. Accessed 08.02.
13. Jones DI, Lorenz MH (1986) An application of a Markov chain noise model to wind generator simulation. Math Comput Simul 28(5):391–402
14. Kaminsky FC, Kirchhoff RH, Syu CY, Manwell JF (1991) A comparison of alternative approaches for the synthetic generation of a wind speed time series. J Sol Energy Eng 113(4):280–289
15. Kang Y, Hyndman RJ, Smith-Miles K (2017) Visualising forecasting algorithm performance using time series instance spaces. Int J Forecast 33(2):345–358
16. Kegel L, Hahmann M, Lehner W (2017) Feature-driven time series generation. Proc 29th GvDB Workshop, p. 54–59
17. Kegel L, Hahmann M, Lehner W (2017) Generating what-if scenarios for time series data. Proc SSDB. https://doi.org/10.1145/3085504.3085507
18. Kegel L, Hahmann M, Lehner W (2018) Feature-based comparison and generation of time series. Prof SSDBM. https://doi.org/10.1145/3221269.3221293
19. Makridakis S, Hibon M (2000) The M3-Competition: results, conclusions and implications. Int J Forecast 16(4):451–476
20. Müller H, Haberlandt U (2015) Temporal rainfall disaggregation with a cascade model: from single-station disaggregation to spatial rainfall. J Hydrol Eng. https://doi.org/10.1061/(ASCE)HE.1943-5584.0001195
21. Neupane B, Pedersen TB, Thiesson B (2014) Towards flexibility detection in device-level energy consumption. Workshop Proc ECML PKDD. https://doi.org/10.1007/978-3-319-13290-7_1
22. van Paassen AHC, Luo QX (2002) Weather data generator to study climate change on buildings. Build Serv Eng Res Technol 23(4):251–258
23. Pesch T, Schröders S, Allelein HJ, Hake JF (2015) A new Markov-chain-related statistical approach for modelling synthetic wind power time series. New J Phys. https://doi.org/10.1088/1367-2630/17/5/055001
24. R Core Team (2014) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria
25. Rahm E, Nagel WE, Peukert E, Jaekel R, Gaertner F, Stadler PF, Wiegreffe D, Zeckzer D, Lehner W (2019) Competence center ScaDS Dresden/leipzig: overview and selected research activities. Datenbank Spektrum 19(1)
26. Schaffner J, Januschowski T (2013) Realistic tenant traces for enterprise DBaaS. Work Proc ICDE. https://doi.org/10.1109/ICDEW.2013.6547423
27. The Commission for Energy Regulation (2015) CER smart metering project. http://www.ucd.ie/issda/data/commissionforenergyregulationcer/. Accessed 18.12.2017
28. VDE e.V. (2011) Messwesen Strom (Metering Code); VDE-AR-N 4400