Time Series Prediction with Automated Periodicity Detection

Michael Schaidnagel
School of Computing
University of the West of Scotland
Paisley PA1 2BE, UK
Email: Michael.Schaidnagel@web.de

Fritz Laux
Faculty of Computer Science
Reutlingen University
D-72762 Reutlingen, Germany
Email: fritz.laux@reutlingen-university.de

***Abstract*—When forecasting sales figures, not only the sales history but also the future price of a product will influence the sales quantity. At first sight, multivariate time series seem to be the appropriate model for this task. Nonetheless, in real life history is not always repeatable, i.e., in the case of sales history there is only one price for a product at a given time. This complicates the design of a multivariate time series. However, for some seasonal or perishable products the price is rather a function of the expiration date than of the sales history. This additional information can help to design a more accurate and causal time series model. The proposed solution uses an univariate time series model but takes the price of a product as a parameter that influences systematically the prediction based on a calculated periodicity. The price influence is computed based on historical sales data using correlation analysis and adjustable price ranges to identify products with comparable history. The periodicity is calculated based on a novel approach that is based on data folding and Pearson Correlation. Compared to other techniques this approach is easy to compute and allows to preset the price parameter for predictions and simulations. Tests with data from the Data Mining Cup 2012 as well as artificial data demonstrate better results than established sophisticated time series methods.**

***Index Terms*—sales prediction, multivariate time series, periodicity mining**

I. Introduction

Time series capture the development of given values over a uniform time interval. There are many areas in which this kind of data can appear: power consumption data of different housing areas per month, heartbeat rate of a patient per minute, hourly weather data or also product sales per day. In this article we will focus mainly on prediction of sales time series in order to keep the main thread examples consistent. However, this is not a limitation since the underlying patterns, which are used in our algorithms, can occur in all of above's application areas.

This work is based on our findings from [1], in which we firstly applied an proposed algorithm named F_r on sales data. The algorithm is characterized by its ability to use the price as a input variable as well as the adaption of a hidden periodicity. Sales prediction is an important goal for any time series based analysis [2], [3]. The task consists of forecasting sales quantities given the sales history. This can be achieved by extending the time series into the future.

The extrapolation of the time series into the future is determined by the underpinning time series model [4]. If this model is not well supported by the empirical data it is likely that the accuracy of the forecast is low. So the challenge is to find data from "similar" situations (e.g., in terms of time and price). If a major sales factor like the product price changes, a model solely based on previous sales will lead to wrong forecasts. Therefore, it is important to include the price as parameter into the model in addition to the sales history.

Standard solutions for this problem need to be provided with a long history of sales with sufficient data to validate the model and to correlate the sales data with the variable product price. The mathematical tools of choice for analyzing multiple time series simultaneously are multivariate statistical techniques like Vector AutoRegressive (VAR) models [5], [6] or such as the Vector ARIMA (AutoRegressive Integrated Moving Average) [7]. The model parameters are estimated with least square or Yule-Walker functions [5]. The accuracy of the estimator depends on the number of observations and its degree of correlation.

To illustrate the process consider an excerpt from the Data Mining Cup 2012 dataset (Table I, http://www.data-mining-cup.de/en/review/dmc-2012):

TABLE I
Sample Data, Data Mining Cup 2012

day	Prod#	price	quantity
1	1	4.73	6
1	2	7.23	0
1	3	10.23	1
1	4	17.90	0
...	...	...	...
1	570	7.91	0
2	1	4.73	12
2	2	7.23	1
...	...	...	...
42	569	9.83	2
42	570	7.84	0
43	1	5.35	?
43	2	7.47	?
...	...	...	...
43	570	7.84	?
...	...	...	...
56	570	8.12	?

The information provided comprises a collection of 570 products whose history of sales and prices are given over

a period of 42 days. The task was to predict the sales quantities for the next 14 days where the daily sales price was preset. The majority of products produced only low quantity sales. Comparisons with other sales data showed a similar distribution [8], [9] which indicates that the sample is typical for larger collections. When we tried to predict the future sales with commercial ARIMA products we experienced a low prediction quality with a relative accuracy of only 47%.

The disappointing results from professional tools implementing ARIMA encouraged us to look for a simpler and better prediction model. Thereby we assumed that the future price is causally influenced and should not be treated as stochastic variable. Second, we assumed that it is helpful to filter out cyclic behavior from the "white noise".

In the next section follows a discussion of related work and we contrast it with our contribution. The rest of the paper is structured as follows: The research problem will be described formally in Section III, which is followed by a description of data profiles under investigation (Section IV). In Section V, we present our parametrized time series algorithm that predicts sales volumes with variable product prices and low data support. This algorithm can benefit from a inherent (i.e., hidden) periodicity within the given data. The periodicity calculation method we used is further described in Section VI. The following Section VII gives a description of the technical framework for the implementation of the prototype. The results are discussed in Section VIII and compared with standard methods found in commercial products like ARIMA. Based on these experiences we draw conclusion in the last section.

II. Related Work

Adaptive correlation methods for prognostic purposes have been proposed early in the 1970^{th} by Griese [10] and more specifically as AutoRegressive Moving Average (ARMA) method by Box and Jenkins [11]. As ARMA is constrained to a stationary stochastic process the ARIMA is of more practical use as it can handle time series with a linear trend and is therefore widely implemented.

The idea behind ARMA and ARIMA is that the model adapts automatically to a given history of data. A natural extension is to include other influential factors beside the prognostic value itself. This leads to multivariate models, namely Vector Autoregressive (VAR) models [7]. The development of the model was influenced and motivated by critiques of Sims [12] and Lucas [13]. In essence, their statement is: every available data is potentially correlated.

If the model is extended to cover the influence from correlated data this leads to a vectorial stochastic model $(\mathbf{X}_t(\pi, \pi_r))$[1] that allows not only the serial time dependence t of each component but also the interdependence of products π and product prices π_r. To estimate the parameters of such a multivariate ARMA process the following Equation has to be solved [14], [5]:

$$\Phi(L)X_t(\pi, \pi_r) \quad = \quad \Theta(L)Z_t \tag{1}$$

where L denotes the backshift (lag) operator and

$$\Phi(x) \quad := \quad I - \Phi_1 x - \Phi_2 x^2 - \ldots - \Phi_p x^p \tag{2}$$

$$\Theta(x) \quad := \quad I + \Theta_1 x + \Theta_2 x^2 + \ldots + \Theta_q x^q \tag{3}$$

are matrix-valued polynomials with dimensions of p (order of regression) and q (order of moving average). Z_t denotes a multivariate "white noise" process.

There is one major drawback to this approach in our problem setting. The model treats all historical input values as stochastic variables. However, the product price does not vary *stochastically*, its value is preset by the vendor. Economic models assume a causal dependency between the price of a product and its sales quantities (see Arnold [15], chap. 17). Variations in consumer demand are caused by various factors like price, promotions, etc, Vorst [16] . This causal dependency is not modeled by VAR methods. This is an issue for the multivariate model.

Another complication that can arise in time series prediction is a noisy periodicity resulting, e.g., from low volume sales. The low sales quantity introduces a kind of random pattern that makes it hard to find even a known periodicity. In the sample data we used for our work, the overall sales history shows a clear 7-day periodicity (see Fig. 1) but not for most of the individual products. Therefore, we were looking for a method to calculate the eventually existing periodicity on product level. This kind of calculation is called periodicity mining and has received some attention from the research community lately.

Elfeky, Aref and Elmagarmid [17] introduce a periodicity mining algorithm that is based on symbols and a Fourier transformation inspired convolution. They defined two types of periodicity (segment and symbol periodicity) and described an convolution based algorithm for both of them. Our suggested algorithm uses the 'shifting' mechanic similar to Elfekys idea, but it differentiates on how similarity of a sequence of symbols is defined. The same authors also describe a method called "WARP" (WArping foR Periodicity) [18] in order to deal with noisy data. Thereby, their algorithm extends or shrinks the time axis at several locations of the time series to remove noise. Rasheed and Alhajj [19] recently used suffix trees (build by Ukonen's linear algorithm) as an underlying data structure in order to detect periodicity in time series. Their iterative approach decorates their suffix tree in a way that highlights repeated occurrences of a sequence of symbols.

Another approach for periodicity mining worth-mentioning was brought up by Berberidis, Aref, Atallah, Vlahavas and Elmagarmid [20]. They create a set of candidate periods out of a given time series and then use the autocorrelation function as well as Fast Fourier Transformation (FFT) for calculating a confidence value for each candidate period. We are also using candidate periods in our approach, but then we use the well known Pearson Correlation Coefficient in order to assess, which of our candidate periods is most suitable.

[1] We use parentheses () for a stochastic process instead of braces {} because it is rather a sequence of stochastic variables than a set

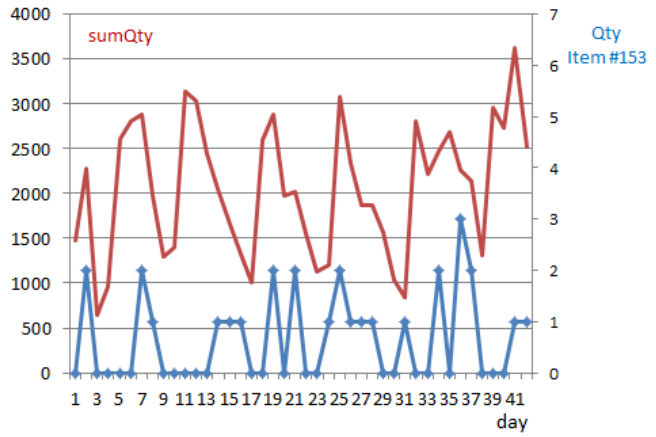

Fig. 1. Seven day periodicity for the overall sales data (DMC2012) and a typical low selling product (item # 153).

In the field of time series analysis, it is quite common to incorporate natural seasons or cycles into the prediction. Cyclic sales quantities are a typical behavior for short shelf-life products and are important for building a causal sales model. Doganis, Alexandridis, Patrinos and Sarimveis [21] investigated the sales quantity of fresh milk (a short shelf-life product) in Greece. They used a genetic algorithm applied to the sales quantities of the same weekday of last year. Our approach is only similar in that we take corresponding weekdays but it differs in how we analyze the weekly periodicity and correlate it with the sales prices.

To recapitulate, there are two general arguments against the multivariate VAR approach sketched above: Granger and Newbold [22] showed that simpler models often outperform forecasts based on complex multivariate models. And Lucas [13] criticized that the economic models are too static and that "any change in policy will systematically alter the structure of the econometric model". Applied to the sales forecast situation the variation of the price does not play a stochastic, but a *systematic*, i.e., a functional role.

Our idea is to filter the seasonality by a period-based "folding" of the sales quantity, i.e., the aggregation of sales quantities for the same weekdays. This cancels the stochastic variation and accumulates the seasonal effect. Applying such a model improves the prediction coverage and accuracy for low volume data with a cyclic behavior.

III. Problem Description and Contribution

In Section I, we pointed out that the nature of the data and its sales profile play an important role for the time series analysis. In particular, the influence of price and periodicity are dominant factors as we will see in the following.

A. Formal Problem Description

The problem in terms of predicting time series consists of developing a parametrized time series model that is able to forecast future sales quantities depending on the given sales history and a price parameter. The solution of the stochastic Equation (1) is a multidimensional mapping

$$F: \quad (\mathbf{\Pi}, \mathbf{T}) \longrightarrow (\mathbb{R}^+, \mathbb{N}_0) \qquad (\pi, t) \longmapsto (\hat{\pi}_r, \hat{x}_t) \tag{4}$$

where $\mathbf{\Pi}$ is the set of products and $\mathbf{T}$ are consecutive time intervals. A product $\pi \in \mathbf{\Pi}$ is described by its identification number π_i and its price π_r. The mapping F computes sales quantity $\hat{x}_t$ and price $\hat{\pi}_r$ for every product π and time interval t.

The bi-variate time series $(\hat{\pi}_r, \hat{x}_t)$ is a concrete realization of the stochastic process (X_t) of Equation (1). The mapping F has to be adjusted so that the process (X_t) explains best a given realization. This can be done by various estimator functions: least square error, Yule-Walker, maximum-likelihood, or Durbin-Levison algorithms. This is where our approach differs from the traditional because in real-life business the price is not a stochastic variable but is preset by the vendor. Instead of predicting the future price $\hat{\pi}_r$ we use the price as input parameter.

Having fixed the model in this way it is possible to transform the mapping F to the following form:

$$F_r: \quad (\mathbb{N}, \mathbb{R}^+, \mathbf{T}) \longrightarrow \mathbb{N}_0 \qquad (\pi_i, \pi_r, t) \longmapsto \hat{x}_t \tag{5}$$

With this predictor $F_r(\pi_i, \pi_r, t)$ it is possible to forecast the sales quantities for future time periods $t > T$ (T is the present time) of a product $\pi \in \Pi$ using the future price π_r as input.

B. Contribution

By restricting our approach to model a linear trend, seasonality, and using historic and future prices as causal parameter leads to a predictor function that is easy to compute and explain. It yields higher accuracy for data with hidden periodicity and variable prices than the ARIMA model. The novelty of our contribution comprises:

- a model that has a causal explanation
- where the future price is a major input factor
- the overall periodicity is respected by individual items
- an algorithm for fast and automated periodicity mining

The prediction function can also be used for simulation to see how the price will influence the sales quantity. In addition to that, we introduce a new approach for periodicity mining, which is able to identify complex seasonal components within a time series. It is based on a simple form of data folding and comparisons of Pearson correlation coefficients.

IV. Data Profile

We used two types of data sets in this article. The first one was obtained from the DataMiningCup (DMC) in 2012. This real life data set was used to assess the prediction performance of our time series algorithm. It will be more closely described in the following subsection. During the development of our periodicity mining method, we also created an artificial data set in order to vary different aspects of a time series, such as

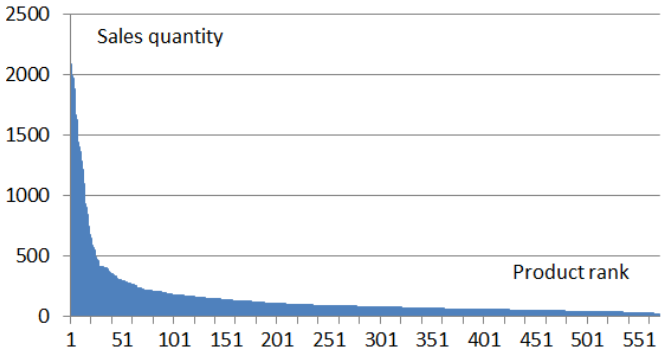

Fig. 2. Sales quantity ranking of sample data (DMC2012).

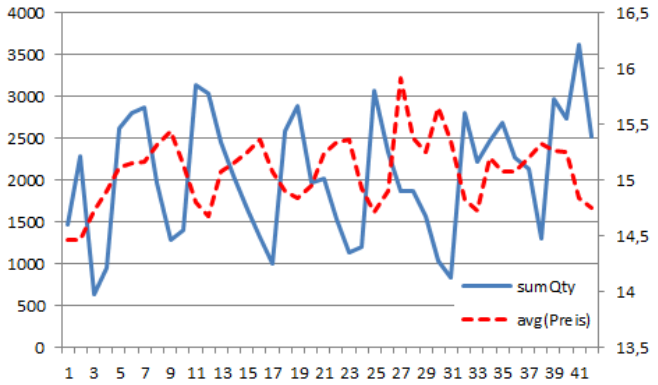

Fig. 3. Sales quantity and average price time series of sample data (DMC2012).

noise, time series length or season length (length of period). The method to create these data is detailed in subsection IV-2.

1) DMC data: The 570 products of the sample data realized a total of 86641 units sold. The average price for all products of a day ranged between 14.46 and 15.92 over the given time series, which included a total of 42 days. The maximum price variability of a single product is $\pm 48\%$, but on average the price varies only by $\pm 9\%$. However, for high selling products (> 500 units) the variability stands at $\pm 15\%$.

The total sales quantity per product ranged from 17 to 2083 over the 6 weeks. Broken down to the day level the product price ranged from 0.24 (cheapest product) to 152.92 (most expensive product) and the sales quantities between 0 and 193. The sample had average sales per product of 152 units with a standard deviation of 257, which indicates a high sales variability of the items.

This conjecture is confirmed by the product sales ranking that roughly follows a shifted hyperbolic distribution (see Fig. 2), which supports the assumption that low volume sales contribute significantly to the overall sales and may not be neglected. From the total of 570 products, the majority (506 products) sell less than 250 units in total, but contribute with approximately the same quantity sold (43991 units) as the 64 high selling products.

The low volume sales (sum of sales < 250) showed a strong positive trend ($\approx 40\%$ increase over 42 days) whereas the high volume sales (sum of sales ≥ 250) had a more stationary behavior. In the sample data are more than 100 products that sell less than six units a day. Nearly all of them sell none at half of the time.

The above properties require an adequate forecasting algorithm, which is able to handle low volume sales with high variability and which is also able to adapt to some price variability.

2) Artificial data set: The creation of artificial time series enabled us to accurately modify different aspects of a time series. This allowed us to investigate on how our periodicity mining algorithm reacts on changes of different parameters. Our goal was to create time series that show rather complex, periodic behavior with some added noise of various intensity.

The time series x_t of length n consist of a constant value c, which is modified by a repeating seasonal components S, which are (again) modified by a randomized noise parameter r as well as an noise intensifying factor f. Seasonal components consist of values $s \in S$ and have the length of m. The length n of a time series is the length m of one season multiplied by the number of seasons. The seasonal component repeats l times [see Equation (8)] along the time series to be created.

$$T = \{t_i | \forall i = 1, \ldots, n\} \quad (6)$$

$$S = \{s_j | \forall j = 1, \ldots, m\} \quad (7)$$

$$n = l * m \quad (8)$$

$$i \bmod m = j \quad (9)$$

A value of the constructed time series is calculated as the sum of constant c and the value of the current season value multiplied by the noise factor r and intensifying factor f:

$$x_{ti} = c + (s_j * r * f) \quad (10)$$

The subsequent value in that time series x_{ti+1} is calculated by shifting the seasonal component one position:

$$x_{ti+1} = c + (s_{j+1} * r * f) \quad (11)$$

This allowed us to create constant time series with an additive period that is disguised by a strong multiplicative interference factor. We created a total of 12 time series for this work with the following specifications:

- season length m: varied between 7 and 33 days
- time series length: 384 days
- basic constant c: was set to 100
- randomized noise parameter r: ranged between $0 < r < 1$
- noise intensifying factor f: varied between 1, 5, and 10

An excerpt of one of the created time series can be seen in Fig. 4. The name of the shown time series is 29_days_1_n, this indicates that a 29 days long periodic component was used and also a noise intensifying factor of 1. The corresponding periodic component, which was used in this example, can be seen in Fig. 5.

Our intention was to create rather difficult seasonal components. We used several 'hand made' components. In addition

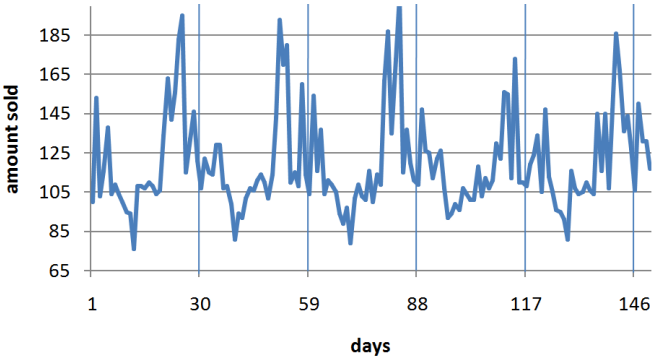

Fig. 4. This example time series was created by using a noise factor of 1 and the seasonal component from Fig. 5 with a length of 29. The vertical blue lines indicate the end of one seasonal component.

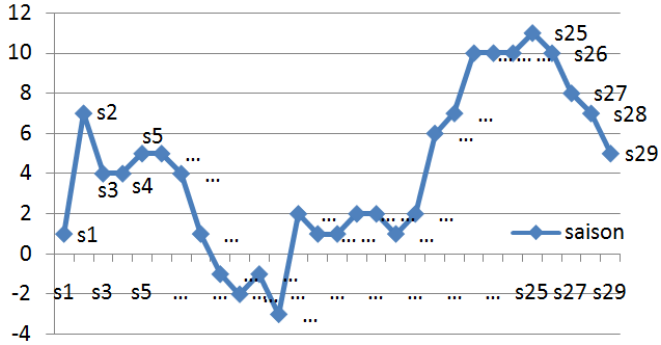

Fig. 5. This example seasonal profile has a length of 29 and was created using a noise factor of 1.

to that we used the following sinus Formula for creating some of the seasonal profiles:

$$\sin(\frac{2\pi}{m} * t) + \frac{1}{3} * \sin(\frac{2\pi * 3}{m} * t) \quad (12)$$

The parameter in above Equation is the base frequency, which determines the period length after the pattern is repeated. Variable t represents the time. This Formula creates time series with a base frequency as well as its third harmonic. An example of a sinus based time series is shown in Fig 6.

V. The Parametrized Time Series Model

This section describes the process from our initial analysis of the data for the DMC 2012, to the justification of our suggested time series model. For the causal predictor function F_r we needed to identify and quantify all influencing factors. Therefore, we firstly analyzed correlations of the attributes quantity, price and time of all products given in the DMC 2012 task. We used the standard Pearson Correlation [23] as a measure to determine the linear dependence between two time series. It is widely used and can range between -1 and $+1$. The following subsections will present the relations that have been analyzed.

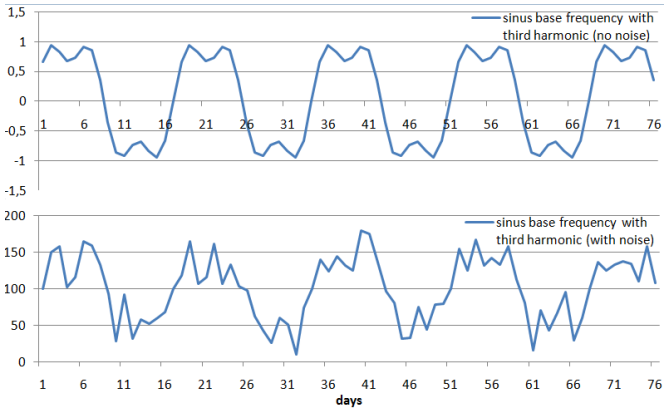

Fig. 6. Upper part: Example of a Sinus based time series with a period length of 17 days and no noise added. Lower part: shows the same time series, which was altered by a noise intensifying factor of 10.

A. Price-Sales Correlation

The main conjecture was that the price has a causal influence on the quantity. This is justified by the price elasticity of demand theory by Alfred Marshall [24]. As the correlation coefficients of all 570 products ranged from -0.6515 to $+0.3471$, we expected that the products with strong correlation exhibit a better prediction accuracy. Surprisingly, this seemed not to be the case.

A systematic analysis with three synthetic time series lead to an explanation. The first series had a growing price trend, the second and third had a cyclic price development where one product responded immediately and the other responded with a delay. ARIMA did recognize the price trend but forecasted a constant quantity instead of a decreasing one. This was the result of the low integer sales numbers that produced a monotone decreasing step function. Our approach managed to forecast the right quantities as long as a matching price was found in the history.

Surprisingly ARIMA could not deal well with the systematic cyclic price development and a detailed analysis showed that the step function of the price (which was kept constant for two days) was the reason. Fig. 7 shows the result of the ARIMA compared to our F_r algorithm (see Equation (5)). The reduced extrema produced by our algorithm results from the delay in the response to the price change. Without lag, no damping of extrema occurs in F_r.

B. Price Similarity

We also analyzed correlations between the price development of different products. The assumption was to find product bundles that are linked together via their price development. For the analysis the prices were normalized first in order to be able to easily compare the different price levels. Several bunches of products were linked together via their prices. But the corresponding sales figures of these products were not related. This is why we ignored the possible cross price influence from other products for the forecast.

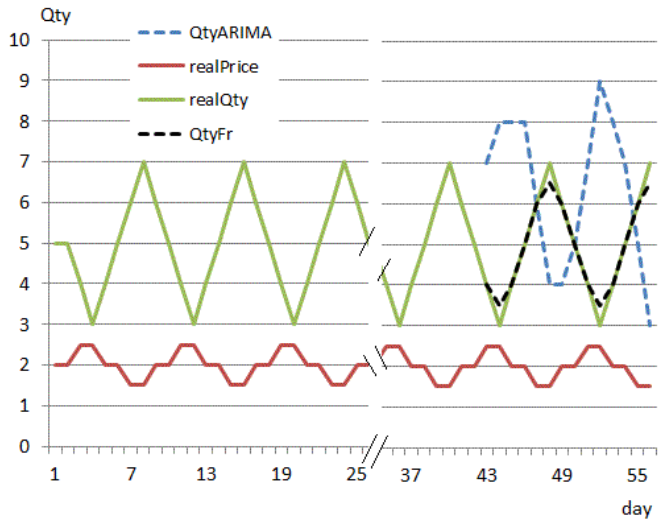

Fig. 7. Forecast of synthetic time series with delayed price-sales dependency.

C. Sales Periodicity

One of the most interesting properties of the DMC 2012 data was the periodicity of the total sales curve. It showed a clear 7 day period (Fig. 1). This period was not directly observable in most sales time series of individual products. Also the Pearson Correlation between the sum curve and the single products was too low to draw any conclusions. Nevertheless, since the total sales curve consists of all products, there must be a hidden periodicity within the individual products.

A systematic spectral analysis discovered not only the dominant weekly patterns but weaker 4, 5 and 14 days patterns. Since these periods differ on item level, there is the need for an automated periodicity mining method. For our work we developed JaPerCalc (Java Periodicity Calculator), which is detailed in Section VI.

D. The Parametrized Predictor Function

Putting all correlation observations together the result is a function F_r whose pseudo code is shown in Fig. 9. As input, it takes the price π_r of a product π at the prediction day t, the periodicity m and a price range δ. The upper and lower price limits are set to $\pm\delta$ percent. Using the periodicity from the previous analysis the algorithm looks for prices $\pi_r(w)$ that occur on days $w = t$ modulo $period$. For example, if the prediction day is 43 and JaPerCalc returns a suggested periodicity of 7 days, only the information from the days 36, 29, 22, 15, 8 and 1 will be considered (see Fig. 8). If the price on such a day is outside of the upper or the lower limit (day 1 in our example), the sales quantity is ignored. If the price is within the bounds, the corresponding quantity is selected. After all matching quantities have been selected, the forecast quantity is computed as linear trend of these quantities.

If all prices are outside of the upper and lower limit, no forecast is produced. The procedure may be repeated with enlarged upper and lower limits if needed. The F_r algorithm defines a simple forecasting model that takes into account the sales trend, the periodicity, and the price influence to predict sales quantities.

VI. Periodicity Calculation

In the normal case, the periodicity of a time series is unknown. However, it is beneficial for most time series algorithms to use this kind of information. The normally suggested standard algorithms [17] - [20] used to calculate this information are based on Fourier Transformation (FT), which is dependent on long time series histories. In order to automate and reduce complexity of the periodicity mining process, we developed our own approach, which will be detailed in the following subsections.

A. General Idea

JaPerCalc is an iterative approach in which a range of candidate periods (e.g., from 5 days period as minimum to 40 days period maximum) are tested. These upper and lower boundaries are the only input that needs to be defined prior by the user. For our hypothetical example, lets assume that we start with candidate period of 5 days. The first step is to fold the data accordingly to the given candidate period (in our case 5 days). This means that we sum up the sold quantity on every 5^{th} day (e.g., day 1, 6, 11, 16 etc.). This is repeated for each consecutive day until our candidate array is of length 5.

The next step is to take this folded candidate array and evaluate it against the given time series using the Pearson Correlation. In terms of our hypothetical example, we compare the folded 5 day candidate array with the days 1-5, 6-11, 11-16 etc. Each comparison results in a Pearson Correlation value r. Thereby, r is given as the mean of the products of the standard scores as it can be seen in Equation (13).

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (13)$$

The average from all of these Person Correlation values is then associated with the candidate period of 5 days.

The process is repeated for all periods within the given boundaries. The finally suggested period is the one with the highest average Pearson Correlation Coefficient.

B. Formula Description

As mentioned before JaPerCalc takes a lower and upper period boundary (l, u) as well as a time series under examination as user input. All possible periods between the mentioned boundaries are referred as candidate periods. The algorithm starts by selecting the first suggested l period and use it to fold (i.e., sum up) every modulo c_j days of the given time series x_t (j is the length of candidate period c_j, see also Fig. 11). The resulting candidate period is held against all parts b of time series x_t of corresponding length. This process is also visualized in Fig. 10.

The algorithm calculates the Pearson correlation coefficient r for each pair of c_j and b. The average of these correlations is associated with the length j of the current candidate period

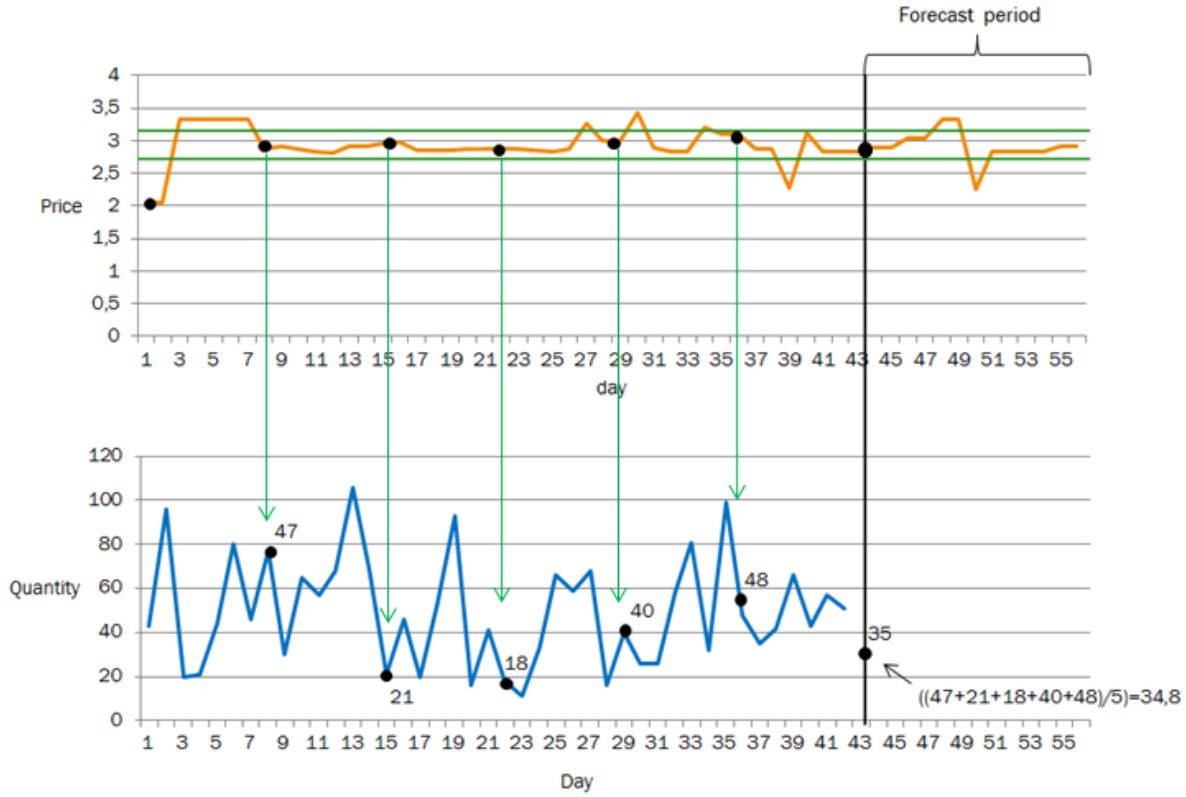

Fig. 8. Illustration of prediction concept using trend, a 7 days periodicity, and a parametrized price.

Input: $t > n$ // prediction day
π_r (input) // price at day t
δ (input) // price range (e.g. $\pm$ 10%)
$period$ (input) // period suggested by JaPerCalc
Def: u, l // upper & lower price limit
QtyList // list of sales quantities
w // = t - n * period ($n \geq 0$)
$\hat{x_t}$ // predicted quantity at day t
for each $\pi \in \Pi$ {
$u := \pi_r(1 + \delta/100)$; $l := \pi_r(1 - \delta/100)$
$w := t - period$
while $(w \geq 1)$ {
if $(\pi_r(w) < u)\,\&\,(\pi_r(w) > l)$
QtyList.add(x_w)
$w := w - period$
}
if (QtyList $\neq \emptyset$)
$\hat{x_t} := trend$(QtyList)
return $\hat{x_t}$
else
return nil
}

Fig. 9. Parametrized Sales Prediction Algorithm F_r in conjunction with JaPerCalc.

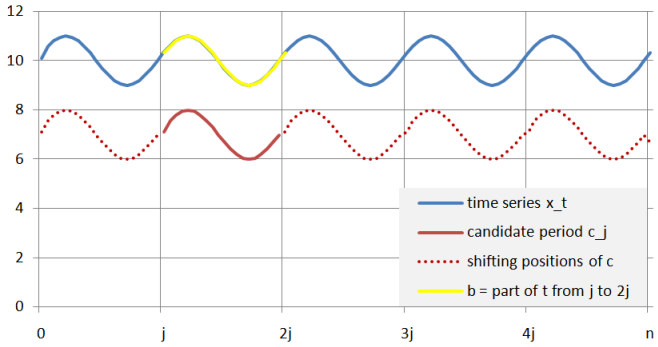

Fig. 10. Illustrative example of JaPerCalc comparing a candidate period c_j with all b parts of a time series x_t of length n.

and put into $PearsonResult$ array for later use. This process is repeated for each candidate period between the lower boundary l and the upper boundary u. The length of c_j is thereby incremented by one for each round. After retrieving the average Pearson correlation coefficient for each candidate period, the algorithm simply returns the j for the candidate component with the highest Pearson correlation coefficient. The complete pseudo code can also be seen in Fig. 11.

VII. Technical Framework and Infrastructure

This section covers some technical details about execution and implementation of the algorithms mentioned in this article.

Input: $(x_{ti}), i = 1, 2, ..., n$ // time series length n
u (input) // upper bound candidate period
l (input) // lower bound candidate period
Def: PearsonList // Set of Pearson corr coefficients
PearsonResult // stores average Pearson results
r // Pearson correlation coefficient
c_j // candidate period array of length j
b // part of time series x_t, from day e to day r
y // integer to iterate through c_j
x // integer to shift b through x_t
$iter := 0$
while $l + iter \leq u\{$
$j := l + iter$
$y = 0$
for $(y < j)$ {
//create candidate period c_j of length j
$c_y = sum_{i+(y \equiv mod(j))}(x_{ti})$
y++
}
repeat $(\lfloor n/j \rfloor)$ **times** {
//compare component c_j with all parts b
//of time series (x_t)
$x = 0$
$b = (t_i | \forall i : x * j \leq i < x * j + j)$
$r := PearsonCorr(c_j, b)$
PearsonList.add(r, j)
x++
}
PearsonResult.add(PearsonList.getAvg(r), j))
$iter$++
}
if (PearsonList $\neq \emptyset$)
return j **from** $PearsonResult$
where $r = PearsonResult.getMax(r)$
else
return nil

Fig. 11. Pseudo-code for JaPerCalc. Returns a suggested period for a given timeline.

A. ARIMA Model Execution

The Microsoft Visual Studio 2008 and Microsoft SQL Server 2008 were used to apply the ARIMA model on the two given data sets (DMC and artificial data). In order to run the ARIMA mining models for both data sets, a OLAP cube was build. It consists of the dimensions *price*, *product* and *time*. In the corresponding time series mining model we used *itemId* and *day* as key attributes and the *price* attribute as input. The *quantity* was set as predictable attribute. For the DMC data we used mostly the default parameters, apart from the the minimum series value and the periodicity hint. The following table shows all model parameters used:

AUTO_DETECT_PERIODICITY	0.6
FORECAST_METHOD	ARIMA
HISTORIC_MODEL_COUNT	1
HISTORIC_MODEL_GAP	10
INSTABILITY_SENSITIVITY	1.0
MAXIMUM_SERIES_VALUE	$+1E308$
MINIMUM_SERIES_VALUE	0
MISSING_VALUE_SUBSTITUTION	None
PERIODICITY_HINT	7
PREDICTION_SMOOTHING	0.5

The specification for the artificial data is equal to the one shown above, except for the PERIODICTIY_HINT that was left blank.

B. Implementation of Time Series Model F_r

Our suggested approach was implemented in Java. We used Eclipse (Version: Indigo Service Release 1) with Java Platform Standard Edition 6.0 (JRE6). The data was stored in a MySQL database on an Apache web server (2.2.21). During execution time the data is queried from the database, the model parameters computed and the forecast results are instantly stored in the corresponding result table in the database. The model was developed using the standard java.sql.* package, which was used to interface with the database and for SQLException handling.

C. Implementation of Periodicity Calculation JaPerCalc

As the name indicates, JaPerCalc was also implemented in Java. We used the two freeware libraries java.util.* and org.apache.commons.math3.stat.* in order to compute the Pearson correlation. We implemented two version of it. The first one outputs the suggested period as well as statistics about the average Pearson correlations of the compared components. It was used for pitching JaPerCalc against Fourier Transformation, see also Section VIII-B. The second version simply returns the suggested period and was used within the F_r algorithm for periodicity detection.

VIII. Experiments and Results

There have been three different sets of experiments, that we carried out. The first set used data from the DataMiningCup 2012. We applied our suggested F_r algorithm in order to predict sales quantities for a given price. We compared its results with the predictions from an ARIMA implementation described in Section VII-A. A discussion about the results can be found in the following Section VIII-A. During the time of the DataMiningCup 2012 we assumed a hidden periodicity within the data on a item level [1]. However, there were some exceptions from our assumption, which raised the need for an automated periodicity detection.

In order to meet this need, we were looking for periodicity detection methods. Our experiments with Fourier transformation led to rather unsatisfactory results and this was the starting point for the development of JaPerCalc. We compare both approaches in Section VIII-B.

The third set of experiments includes both, the time series prediction as well as the periodicity detection problem. We

TABLE II
COMPARISON OF ARIMA PREDICTION ERROR WITH F_r ALGORITHM ON DMC DATA

Class	ARIMA	F_r	improvement
All products	30512	22093	26.7%
quantity < 500	24338	17248	29.1%
quantity ≥ 500	6174	4845	21.5%
(quantity = 0) in $< 1/3$ time	19178	15942	16.9%
(quantity = 0) in $\geq 1/3$ time	11334	6151	45.7%
avg(π_r) < 20	26756	18711	30.1%
avg(π_r) ≥ 20	3756	3382	10.0%
top 100 items	15805	6131	61.2%
least 470 items	16167	15962	1.2%

used artificial data for these experiments and compared to the results of ARIMA. We will discuss the results in Section VIII-C.

A. Comparison of Results with ARIMA

The absolute prediction error was measured as $|realQty - predictQty|$. The F_r algorithms benefited from two input parameters: the hidden periodicity that was calculated in a previous step and the predefined future price (see also [1] for more details). The hidden sales periodicity contributed for an improvement of about 20%. The overall forecast was improved by 26.7%. The price influence was less dominant than expected, but was determinant for a cluster of 26 products. Cluster characteristics:

- correlation < -0.25
- relative standard error < 0.25
- sales quantity > 160
- price variation $(max(\pi_r) - min(\pi_r)) > 4$

In total, F_r could forecast this cluster 36.4% better than ARIMA. Table II shows the prediction error points of both ARIMA and F_r on certain product clusters. These were grouped based on attributes which we guessed to have an impact on the prediction performance (e.g., the top 100 vs the least 470 items). The total error of all 570 products was 30152 for the ARIMA and 22093 for F_r. This is an improvement of 26.7% compared to ARIMA. For further analysis we clustered the products into disjoint sets according to different criteria. This allowed us to find the strengths and weaknesses of F_r in terms of total sales quantity, sales sparsity, and price.

B. Periodicity Calculation

This subsection will show some results from our experiments in which we compare Fourier Transformation with our suggested methods for periodicity mining on our artificial data set. Table III shows all results.

The FFT detected 8 out of 12 periods correctly. It seems like FFT performs better on high frequency periods with a shorter period length (e.g., 7, 16, 17). However, for longer periods (e.g., 33, 29), FFT slightly underestimated the periodicity length.

F_r was able to detect 9 out of 12 periods correctly. Please note that in all of the three wrong suggested cases, a multiple

TABLE III
COMPARISON OF PERIODICITY SUGGESTED BY FOURIER TRANSFORMATION AND F_r ON ARTIFICIAL DATA

time series name	FFT	F_r	real
16_days_1	16	16	16
17_days_10_n_sin	28	17	17
17_days_1_n_sin	17	17	17
17_days_5_n_sin	17	17	17
20_days_1_n	20	20	20
29_days_1_n	28	29	29
33_days_10_n	32	33	33
33_days_1_n	32	33	33
33_days_5_n	32	33	33
7_days_10_n	7	35	7
7_days_1_n	7	35	7
7_days_5_n	7	42	7

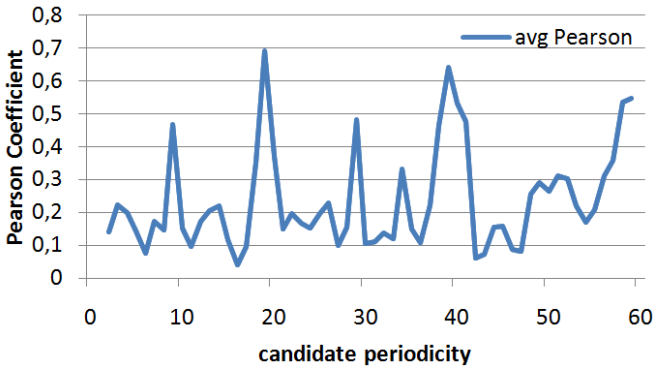

Fig. 12. Average Pearson Correlations for several candidate periods in time series 20_days_1_n. Please note that also multiple and partial candidate components are highlighted (i.e., the correct periodicity of 20 is highlighted, but also partial components such as 10, 30 or multiple components such as 40 or 60.)

of the correct period length (harmonic) was suggested. Reason for this lies within the intrinsic mechanic of JaPerCalc. If a periodicity of a certain length n is repeating within a time series, there is also a multiple of the inherent periodicity with length $2n$ ($4n$, $6n$ and so forth) visible. Same goes for partial periodicity of length $1.5n$ or $2.5n$. JaPerCalc is able to highlight all of these periodicity lengths. Fig. 12 shows this for the example of time series 20_days_1_n. In case of the wrong suggested periodicity from time series 7_days_*_n it is very likely that noise caused a multiple component of the inherent periodicity length to have a slightly higher average Pearson correlation coefficient than the inherent (given) periodicity. This can occur in following scenario: if a inherent period has a significant peak on, lets say the 7^{th} day, this periodicity is disguised if noise randomly causes every second of these peaks to diminish. Result is a higher average Pearson correlation coefficient for a period of length 14 (rather than the correct 7 days). How this affected the predictions made by F_r can be seen in the following section.

TABLE IV
COMPARISON OF ARIMA PREDICTION ERROR WITH F_r ALGORITHM ON ARTIFICIAL DATA

time series name	ARIMA	F_r + JaPerCalc	improvement
16_days_1	797	240	69.89 %
17_days_10_n_sin	10443	3400	67.44 %
17_days_1_n_sin	1048	304	70.99 %
17_days_5_n_sin	5165	1627	68.50 %
20_days_1_n	3002	1422	52.63 %
29_days_1_n	5006	2139	57.27 %
33_days_10_n	7883	3234	58.98 %
33_days_1_n	862	330	61.72 %
33_days_5_n	3776	1471	61.04 %
7_days_10_n	11307	9610	15.01 %
7_days_1_n	1154	937	18.80 %
7_days_5_n	5942	4576	22.99 %

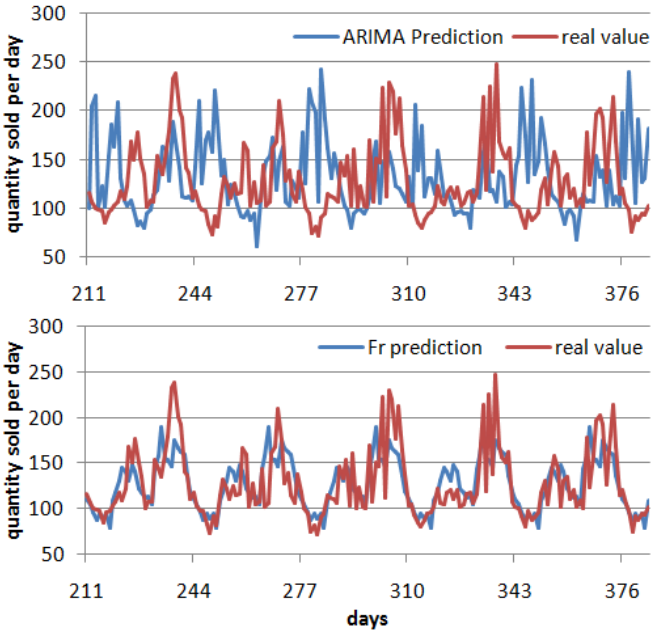

Fig. 13. prediction results for ARIMA (upper part) and prediction results for F_r (lower part).

C. Time Series Prediction with automated Periodicity Detection

As mentioned before we used artificial data described in IV-2. We once again calculated the absolute prediction error as $|realQty - predictQty|$. Table IV shows the results for all generated time series.

JaPerCalc was able to find the correct period for 9 out of the given 12 time series. The predictions for both compared methods for 33_days_10_n are visualized in Fig. 13. The period length prediction for 7_days_*_n was incorrectly predicted by JaPerCalc (as described in previous section). However, since a multiple of correct period was suggested, F_r is still able to outperform ARIMA by 15.01% to 22.99%. If the suggestions from JaPerCalc is correct, the prediction improvement from F_r compared to ARIMA rises to averaged 63.13%.

As it can be seen ARIMA is not able to compensate the noise within the time series and gets biased. F_r on the other, hand is able to average the given noise by following the candidate period, which was detected by JaPerCalc. The interaction of two rather simple algorithms is able to deliver impressive predictions results. Similar behavior can be seen for the time series created by the sinus function described in Equation (12). An example can be seen in Fig. 14.

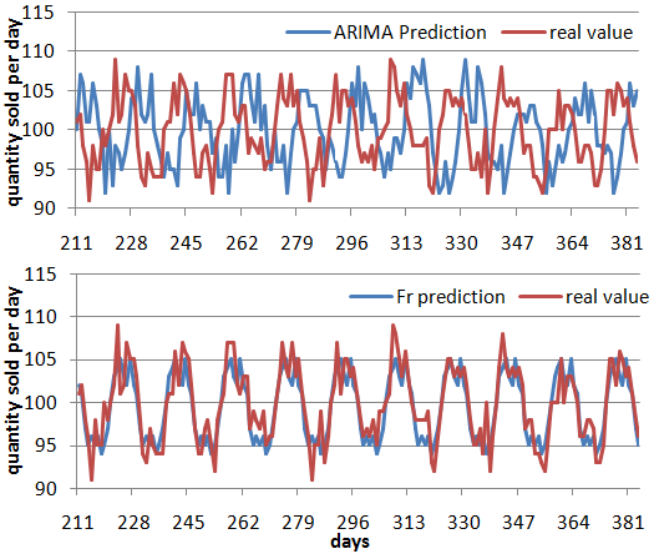

Fig. 14. prediction results for ARIMA (upper part) and prediction results for F_r (lower part). We used a sinus function to create this time series.

As the name of the time series 17_days_1_noise indicates, there was only a noise factor of 1 used. Although a rather simple period component and low noise was used, ARIMA gets distracted and produced poor prediction results.

IX. CONCLUSION AND FUTURE WORK

This article covered a long development. We started with the DMC 2012 and its challenging data set. The broad range of products with its hidden periodicity made the analysis difficult. The low volume sales further complicated the analysis of the influence of the price on the sales quantities. The conclusions drawn from the above results can be summarized in the following three statements:

1) Data profiling and periodicity mining is crucial for choosing the best time series model
2) Low sales volume can hide a cyclic sales behavior and the price should be treated as input parameter
3) Simple models for sales forecasting based on causal parameters can outperform some sophisticated stochastic models.

This lead to the development of the rather simple F_r algorithm. If it is applied on other data, in which the periodicity is not known, it is likely to produce disappointing results. Reason for this is its dependency on the input period (which was set to 7 in case of the DMC 2012 data).

JaPerCalc was created in order to overcome that weakness. It is also a rather simple and fast algorithm that is able to detect

periodicity in a given time series. The Pearson correlation coefficient allows to detect periodicity of any form or shape.

The combination of both simple algorithms is able to significantly outperform standard methods such as the ARIMA, which was shown in Section VIII. JaPerCalc as well as F_r algorithm can be used with incomplete time series. This is particularly useful for real-time analysis used in recommendation systems. The simplicity and low computational effort for both algorithms makes them ideal for people who want to delve into the field of time series prediction.

In terms of future work for F_r, a spectral analysis on an individual product level could further improve the prediction accuracy. For products with a strong monotone price development, our approach to look for similar prices is not well suited. The price trend should be computed instead. There is also the option to adjust the price ranges by the variance of the so far known time series. Products with a high variance could be allowed to have a broader prince range then product with a lower variance. JaPerCalc could also be adapted in a way to automatically recognize and adjust eventually existing trends. This would help to improve periodicity recognition for monotonous increasing time series. Another direction of development could be to use confidence values to help JaPerCalc to find the base frequency of a time series (i.e., shortest suitable period length). This could help to prevent incorrect suggestion of multiples of the correct period length as described in Section VIII-B.

REFERENCES

[1] M. Schaidnagel, "Sales Prediction with Parametrized Time Series Analysis", DBKDA 2013, The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications, Seville, Spain - January 27^{th} - February 1^{st}, 2013

[2] A.-W. Scheer, *Sales Forecast*, Springer Verlag, Berlin, 1983

[3] M. Hüttner, *Market and Sales Forecast*, Kohlhammer, Stuttgart, 1982

[4] Y. Lan and D. Deagu, *A New Approach and Its Applications for Time Series Analysis and Prediction Based on Moving Average of n^{th}-Order Difference*, in: D. E. Holmes and L. C. Jain (Eds.), *Data Mining: Foundations and Intelligent Paradigms*, Vol 2: Statistical, Bayesian, Time Series and other Theoretical Aspects, pp. 157 - 182, Springer Berlin Heidelberg, 2012

[5] K. Neusser, *Time Series Analysis for Economic Science*, B. G. Teubner Verlag, Wiesbaden, 2006

[6] A. J. Izenman, *Modern Multivariate Statistical Techniques - Regression, Classification, and Manifold Learning*, Springer Science + Business Media, New York, 2008

[7] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*, corr. repr., Springer Verlag, Berlin Heidelberg, 2007

[8] S. Oches, "Top 50 Sorted by Total Units", Special Report of QSR Magazine, Journalistic Inc., August 2011, [Online] http://www.qsrmagazine.com/reports/top-50-sorted-total-units, last access: 14.12.2013

[9] Economist Intelligence Unit, "Denmark: Market Indicators and Forecasts", [Online] http://datamarket.com/data/set/1wmo/ (indicators: Private consumption, Consumer goods), last access: 30.08.2012

[10] J. Griese, *Adaptive Verfahren im betrieblichen Entscheidungsprozess*, Physica Verlag, Würzburg - Wien, 1972

[11] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 1^{st} rev. ed., Holden Day,Oakland, San Francisco, 1976

[12] C. A. Sims, "Macroeconomics and Reality", in: Econometrica, Vol. 48, No. 1, pp. 1 - 48, 1980

[13] R. E. Lucas, "Econometric policy evaluation: A critique", In: K. Brunner and A. H. Meltzer (Eds), *The Phillips Curve and Labor Markets*, Vol. 1, Carnegie-Rochester Conference Series on Public Policy, pp. 19 - 46, Amsterdam, North-Holland, 1976

[14] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, 2^{nd} Edition, Springer Science + Business Media, New York, 2006

[15] R. A. Arnold, *Economics*, 9^{th} ed., South-Western College Publ., 2008

[16] J. G. A. J. van der Vorst, A. J. M. Beulens, W. de Wit, P. van Beek, *Supply chain management in food chains: Improving performance by reducing uncertainty*, in: International Transactions in Operational Research, Vol. 5(6), pp 487 - 499, 1998

[17] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid *Periodicity Detection in Time Series Databases*, in: IEEE Transactions on Knowledge and Data Engineering, pp. 875 - 887 Vol. 17, No. 7, July 2005

[18] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid *WARP: Time Warping for Periodicity Detection*, in: Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM05)

[19] F. Rasheed, R. Alhajj *Using Suffix Trees for Periodicity Detection in Time Series Databases*, in 4th International IEEE Conference "Intelligent Systems", 2008

[20] C. Berberidis, W. G. Aref, M. Atallah, I. Vlahavas, A. K. Elmagarmid *Multiple and Partial Periodicity Mining in Time Series Databases*, in F. van Harmelen (ed.): ECAI2002, Proceedings of the 15th European Conference on Artificial Intelligence, IOS Press, Amsterdam, 2002, pp.30-37

[21] P. Doganis, A. Alexandridis, R. Patrinos, and H. Sarimveis, *Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing*, Journal of Food Engineering 75, pp. 196 - 204, Elsevier Ltd., 2006, [Online] http://www.elsevier.com/locate/jfoodeng, last access: 30.08.2012

[22] C. W. J. Granger and P. Newbold, *Economic Forecasting: The Atheist's Viewpoint*, in: G.A. Renton (ed.), *Modelling the Economy*, pp. 131 - 148, Heinemann, London, 1975

[23] H. Rinne and K. Specht, *Time Series - statistical modelling, Estimation and Prediction*, Verlag Vahlen, Munich, 2002

[24] A. Marshall, *Principles of Economics*, 8^{th} ed., Cosimo Classics, 2009, first publ. 1890