Piecewise analysis of stock market data

Paolo Antonini Davide Azzalini Fabio Azzalini

Abstract

Time series data is characterised as large in size, high dimensionality and continuous updates. Moreover, it is always considered as a whole, instead of individual numerical fields. As a consequence, in order to analyse and mine time series data, segmentation and dimensionality reduction are essential, as well as trend identification. In particular, in the following pages we are going to collect and summarise some segmentation methods, applied in particular to stock market data. As a matter of fact, stock time series have their own characteristics over other time series.

1 Introduction

The tasks of segmentation and dimensionality reduction, as well as identification of trends, are fundamental to allow a number of time series analysis and mining tasks. As a matter of fact, the fields of application of such procedures are numerous (ECG, exchange rates, sensors detections of any kind, . . .); moreover methods to perform this kind of preprocessing differ greatly, due to the nature and inherent characteristics of data. As a consequence, literature regarding these issues is vast.

In particular, our focus in this report is on stock market data, which is outstandingly large in size, noisy and continuously updated. This paper is structured as follows: in section 2 we are providing a theoretical introduction and overview on the problem. Then in section 3 a number of research papers about dimensionality reduction and piecewise segmentation are listed and summarised. Then, in section 4 we are going to present the implementations of a few methods.

2 Time series dimensionality reduction

In relevant literature, many algorithms have been proposed for representing time series into their segmented forms (i.e., dimensionally reduced), and many methods for creating time series representations can be found.

Generally, basic classification of algorithmic methods for segmentation implies their division into two great categories: offline, and online algorithms. The essence of offline segmentation is contained in the scanning and division of the entire time series $T = \{(y_1, t_1), (y_2, t_2), \ldots, (y_n, t_n)\}$ into a number of segments, while the available number of data (data points) remains unchanged. On the other hand, in the online segmentation the uploading of new data points is performed in parallel with the execution of the algorithm. Data points (y_i, t_i)

are added one by one at every time step t_i, (for $i = 1, 2, \ldots, n$, where n is growing potentially forever); the algorithm must decide whether the new data points belong to the previous segment, or should be assigned to a new segment, which starts at t_i.

Based on the essential context of the offline and online segmentation, numerous algorithms for segmentation of time series are usually classified into one of the following categories of algorithms: *top-down* algorithm; *bottom-up* algorithm; and *sliding window* algorithm. As a result of combining bottom-up and sliding window procedures, a new, precise, streaming *SWAB* algorithm (Sliding Window And Bottom-up algorithm), has been developed.

Top-down The top-down algorithm, often called "divide and conquer" or "binary split", starts with conditional observing of non-segmented time series as one major segment. Based on the consideration of all possible variants for initial division, the best location for placing the breaking point that splits original time series in two segments is identified, in such a way that the difference between those two segments is maximal. Both these segments are then tested with regard to the approximation error. If the approximation error of the observed segment is below the user-defined threshold, the segmentation procedure stops, and the tested segment is accepted. On the other hand, if the approximation error is above the user-defined threshold, further division of the tested segment into two new (sub)segments is performed. For each of the newly formed segments, the process of division into two new segments is repeated in an identical manner, without effect on the location of the breaking point determined in the previous iteration. The algorithm repeats these steps until some of the defined stopping criteria is satisfied (i.e., when further division no longer contributes to the minimisation of the segment or segmentation error):

1. k number of segments, and/or,
2. approximation error less than the user-specified threshold.

Bottom-up The bottom-up algorithm, often called "iterative merge", as a natural complement to the top-down algorithm, begins by dividing the original time series, of length n, into a large number of very small segments with equal lengths. In the next step, based on the comparison of each pair of consecutive segments, the pairs that cause the smallest increase in the error are identified, and consequently merged in a bigger segment. The algorithm repeats these steps until some of the defined stopping criteria is satisfied:

1. k number of segments, and/or,
2. approximation error greater than specified threshold.

Sliding window The process of segmentation by using the sliding window algorithm, often called "brute force" or "one-pass algorithm", begins by determining the left anchor of the first potential segment (usually the first data point of a time series), which is also the starting point for the window which slides to the right along the time series; this way, the segments that satisfy predefined segmentation criterion (typically a user-specified threshold) are identified and selected. While sliding down the sequence, the size of the window gradually increases, since all the visited data points become potential elements of next segment to be defined, until the error of the potential segment exceeds the user-specified threshold. At this point, the right boundary of the moving window ceases to be unknown: the length of the segment is determined, and the stopping point of the newly formed segment becomes the new anchor (i.e., the starting point of the next potential segment).

SWAB SWAB is a more accurate algorithm for segmentation of data streams. Its name points out that it is the approach that is based on a combination of the sliding window and bottom-up algorithms. SWAB segmentation algorithm begins by defining and selecting one initial size of the buffer that is big enough to contain enough data to create 5 or 6 segments. In the next step, the bottom-up procedure is applied to the data points inside the buffer. In that way, the leftmost segment is detected, and the data points that correspond to the detected segment are removed from the buffer. Then, in the buffer, using classical sliding window procedure for defining new entry points, new data points are added. A new iteration of the bottom-up procedure is performed in the buffer. Obviously, this process of incorporating new data points in the buffer can be repeated as long as the data arrive in continuous streams, potentially indefinitely.

3 Methods

What follows is a selection of academical papers focusing specifically on dimensionality reduction methods applied to financial time series.

[1]. "A review on time series data mining"

Authors Fu

Title "A review on time series data mining"

Year 2011

Keywords `overview`

In this paper, a comprehensive revision on the existing time series data mining research is given. Research is generally categorised into representation and indexing, similarity measure, segmentation, visualisation and mining. Moreover, state-of-the-art research issues are also highlighted. The primary objective of this paper is to serve as a glossary for interested researchers to have an overall picture on the current time series data mining development and identify their potential research direction to further investigation.

[2]. "A Pattern Distance-Based Evolutionary Approach to Time Series Segmentation"

Authors Yu, Yin, Zhou, *et al.*

Title "A Pattern Distance-Based Evolutionary Approach to Time Series Segmentation"

Year 2006

Keywords `PIP, pattern matching`

Given a set of pattern templates, evolutionary computation is an appropriate tool to segment time series flexibly and effectively. In this paper, a new distance measure based on pattern distance for fitness evaluation is proposed. Time sequence is represented by a series of perceptually important points and converted into piecewise trend sequence. Pattern distance measures the trend similarity of two sequences.

[3]. "An Evolutionary Approach to Pattern-based Time Series Segmentation"

Authors Chung, Fu, Ng, *et al.*

Title "An Evolutionary Approach to Pattern-based Time Series Segmentation"

Year Oct. 2004

Keywords `pattern matching, PIP`

This paper considers the problem of identifying a suitable set of time points for segmenting the time series in accordance with a given set of pattern templates (e.g., a set of technical patterns for stock analysis). The use of fixed-length segmentation is an oversimplified approach to this problem; hence, a dynamic approach (with high controllability) is preferable so that the time series can be segmented flexibly and effectively according to the needs of the users and the applications. It is proposed an evolutionary time series segmentation algorithm that allows a sizeable set of pattern templates to be generated for mining or query. In addition, defining similarity among time series (or time series segments) is of fundamental importance in fitness computation. By identifying the perceptually important points (PIPs) directly from the time domain, time series segments and templates of different lengths can be compared, and intuitive pattern matching can be carried out in an effective and efficient manner.

[4]. "Stock time series pattern matching: Template-based vs. rule-based approaches"

Authors Fu, Chung, Luk, *et al.*

Title "Stock time series pattern matching: Template-based vs. rule-based approaches"

Year 2007

Keywords `pattern matching, PIP`

When it comes to locating the technical patterns in the stock price movement charts to analyse the market behaviour, there are two main problems: how to define those preferred patterns (technical patterns) for query and how to match the defined pattern templates in different resolutions. Defining the similarity between time series (or time series subsequences) is of fundamental importance. By identifying the perceptually important points (PIPs) directly from the time domain, time series and templates of different lengths can be compared. Three ways of distance measure, including Euclidean distance (PIP-ED), perpendicular distance (PIP-PD) and vertical distance (PIP-VD), for PIP identification are compared in this paper. After the PIP identification process, both template- and rule-based pattern-matching approaches are introduced.

[5]. "Representing financial time series based on data point importance"

Authors Fu, Chung, Luk, *et al.*

Title "Representing financial time series based on data point importance"

Year 2008

Keywords `PIP, binary tree representation`

Stock time series has its own characteristics over other time series. Dimensionality reduction is an essential step before many time series analysis and mining tasks. In this paper, financial time series are represented according to the importance of the data points. With the concept of data point importance, a tree data structure, which supports incremental updating, is proposed to represent the time series and an access method for retrieving the time series data point from the tree, which is according to their order of importance, is introduced. This technique is capable of presenting the time series in different levels of detail and facilitating multi-resolution dimensionality reduction of time series data.

[6]. “Financial Time Series Representation Using Zigzag Based Perceptually Important Points”

Authors Phetking, Sap, and Selamat

Title “Financial Time Series Representation Using Zigzag Based Perceptually Important Points”

Year Jun. 2009

Keywords `PIP, binary tree representation`

Financial time series often exhibit high degrees of fluctuation which are regarded as noise in time series analysis. To remove noise, in financial time series analysis, it is important to retain the important points and remove others. Here is proposed the Zigzag based Perceptually Important Point Identification method to collect those zigzag movement important points. Further, it is also proposed Zigzag based Multiway Search Tree to index these important points in a convenient way.

[7]. “Clustering-Inverse: A Generalized Model for Pattern-Based Time Series Segmentation”

Authors Deng, Chung, and Wang

Title “Clustering-Inverse: A Generalized Model for Pattern-Based Time Series Segmentation”

Year 2011

Keywords `pattern matching, PIP`

In this paper, according to the characteristics of PTSS (Patterned-based time series segmentation), a generalised model is proposed for PTSS. A new interpretation for PTSS is given by comparing this problem with the prototype-based clustering (PC). Then, a novel model, called clustering-inverse model (CI-model), is presented. Finally, two algorithms are presented to implement this model.

[8]. “Financial time series segmentation based on Turning Points”

Authors Yin, Si, and Gong

Title “Financial time series segmentation based on Turning Points”

Year Jun. 2011

Keywords `turning points, PIP`

In this paper, a novel time series segmentation method based on Turning Points is proposed. Turning Points are extracted from the maximum or minimum points of the time series. The proposed segmentation method generates segments at different levels of details and achieves satisfactory results in preserving higher number of trends compared to an existing segmentation approach.

[9]. "A Turning Point Method For Stream Time Series Prediction"

Authors Vo, Jiawei, and Vo

Title "A Turning Point Method For Stream Time Series Prediction"

Year 2013

Keywords `turning points`

In this paper it is proposed an approach established on turning points to reduce the dimensions of stream time series data, and this task supports the prediction process faster in stream data environment. The turning points are extracted from the maximum or minimum points of the time series data proved more efficient and effective in the process of processing data for time series predictive analysis.

[10]. "Stock time series visualization based on data point importance"

Authors Fu, Chung, Kwok, *et al.*

Title "Stock time series visualization based on data point importance"

Year 2008

Keywords `PIP, binary tree representation`

In this paper, a framework that represents and visualises time series data based on data point importance is proposed. Furthermore, discovering frequently appearing and surprising patterns are non-trivial tasks in financial applications. A method for discovering patterns across different resolutions is proposed. The proposed method is based on a modified version of VizTree. By converting the time series to symbol string based on data point importance, the potential patterns with different lengths can be encoded in the VizTree for visual pattern discovery while the important points and the overall shape of the time series patterns can be preserved even under a high compression ratio.

[11]. “Mining of concurrent text and time series”

Authors Lavrenko, Schmill, Lawrie, *et al.*

Title “Mining of concurrent text and time series”

Year 2000

Keywords `trend identification, finance`

This paper describe the design and implementation of Ænalyst, a system for predicting trends in stock prices based on the content of news stories that precede the trends. Trends are identified in time series using piecewise linear fitting and then labels are assigned to trends according to an automated binning procedure. Language models are used to represent patterns of language that are highly associated with particular labelled trends. Ænalyst can then identify news stories that are highly indicative of future trends.

[12]. “A Two-Phase Stock Trading System Using Distributional Differences”

Authors Kim, Lee, Lee, *et al.*

Title “A Two-Phase Stock Trading System Using Distributional Differences”

Year 2002

Keywords `prediction, finance`

This paper presents a two-phase (extraction and filtering) stock trading system that aims at maximising the rates of returns. Extraction of stocks is performed by searching specific time-series patterns described by a combination of values of technical indicators. In the filtering phase, several rules are applied to the extracted sets of stocks to select stocks to be actually traded. The filtering rules are induced from past data using distributional differences. From a large database of daily stock prices, the values of technical indicators are calculated. They are used to make the extraction patterns, and the distributions of the discretisation intervals of the values are calculated for both positive and negative data sets. The values in the intervals of distinctive distribution may contribute to the prediction of future trend of stocks, so the rules for filtering stocks are induced using those intervals.

4 Implementations

Turning points Due to its simplicity, the apparently good results and openness to further improvements, we decided to implement the *turning points* method presented in [8]. So, in the following sections 4.1 and 4.2 we are presenting our implementation both in MATLAB and in Python of the method.

Basically, the algorithm is divided into two phases. First, during a preprocessing phase all points which are neither local maxima nor minima are discarded; then some patterns are simplified, since immaterial. However, we are not going to discuss the theoretical foundations and the steps of the algorithm in detail, as the aforementioned article if sufficient.

Other implementations In order to provide a better view on the implementations, we are presenting other methods in section 4.3.

4.1 Turning points in MATLAB

Implementation The method is developed and tested over `CSV` files downloaded from Yahoo! Finance website[1]. Should another source be used, basic adaptations may be necessary, mostly in the handling of temporal data[2] (i.e., dates), which is performed in `TP_prepareData()` procedure.

After importing the aforementioned `CSV` file (we suggest to use the graphical interface provided by MATLAB itself), the user should issue the following command, in order to compute and plot the results:

```
y = TurningPoints(time, values, n);
```

Here, `time` and `values` represent the time series as imported from the `CSV`; `n` instead lets the user specify the number of times the algorithm shall be performed (preprocessing is excluded from this count). The results are both displayed in a plot and stored in `y` variable.

`TurningPoints()` function is implemented as shown in listing 1, with the support of some side functions.

[1]`http://finance.yahoo.com/market-overview/`

[2]Due to our limited knowledge of Matlab language, we may have dealt with the source data in a naïve way. Nevertheless, the procedure seems to work well.

```
%% Actual TP function
function tp = TurningPoints(time, value, n)

x = TP_prepareData(time, value);
tp = TP_preprocess(x);

while n > 0

    n = n-1;
    y = tp;
    clear tp;

    i = 1;
    while i < (length(y)-3)

        p0 = y(i+0,2);
        p1 = y(i+1,2);
        p2 = y(i+2,2);
        p3 = y(i+3,2);

        condUT = p0 < p1 && p0 < p2 && p1 < p3 && p2 < p3 ...  % uptrend
            && abs(p1 - p2) < abs(p0 - p2) + abs(p1 - p3);

        condDT = p0 > p1 && p0 > p2 && p1 > p3 && p2 > p3 ...  % downtrend
            && abs(p2 - p1) < abs(p0 - p2) + abs(p1 - p3);

        eps = 0.05 * mean([p0 p1 p2 p3]);
        condST = abs(p0 - p2) < eps && abs(p1 - p3) < eps;  % same trend

        if condUT || condDT || condST
            tp(i,:) = y(i,:);
            tp(i+3,:) = y(i+3,:);
            i=i+3;
        else
            tp(i,:) = y(i,:);
            i=i+1;
        end  % end if

    end  % end while i < (length(y)-3)

    tp(length(y),:) = y(length(y),:);
    tp = TP_cleaning(tp);
    TP_output(y,tp)

end  % end while n > 0

plot( datetime ( x(:,1), 'ConvertFrom', 'datenum'), x(:,2), ...
    datetime ( tp(:,1), 'ConvertFrom', 'datenum'), tp(:,2));

end  % end TurningPoints()
```

Listing 1: TurningPoints() function.

The main supporting function is `TP_preprocess()`, which implements the preprocessing phase, and is presented in listing 2. The other supporting functions are of secondary importance, so we are not presenting them here. Basically we have:

- `TP_prepareData()` takes in input raw Yahoo! Finance data and prepares them to the processing;
- `TP_cleaning()` cleans data matrix after processing;
- `TP_output()` shows information about the processing (i.e., the number of deleted elements).

```
%% Data preprocessing
function y = TP_preprocess(x)

% Boundary elements
y(1,:) = x(1,:);
y(length(x),:) = x(length(x),:);

% Core
for i=2:(length(x)-1)

    prec = x(i-1,2);
    curr = x(i,2);
    succ = x(i+1,2);

    condMIN = curr < prec && curr < succ;  % curr: local minimum
    condMAX = curr > prec && curr > succ;  % curr: local maximum
    if condMIN || condMAX
        y(i,:) = x(i,:);
    end  % end if

end  % end for

y = TP_cleaning(y);
TP_output(x,y)

end  % end TP_preprocess()
```

Listing 2: `TP_preprocess()` supporting function.

Test To conclude our discussion, we are presenting some tests we performed. Weekly stock market data from A2A (`A2A.MI`) over the whole 2015 were used as source. In particular, we plotted the weekly `Close` time series. In fig. 1 we show the original data in blue, the preprocessed data in orange and the data after one full run of the algorithm in yellow.

Original data contains 53 samples; preprocessing reduces them to 27, and finally, after the actual processing, only 12 samples are left.

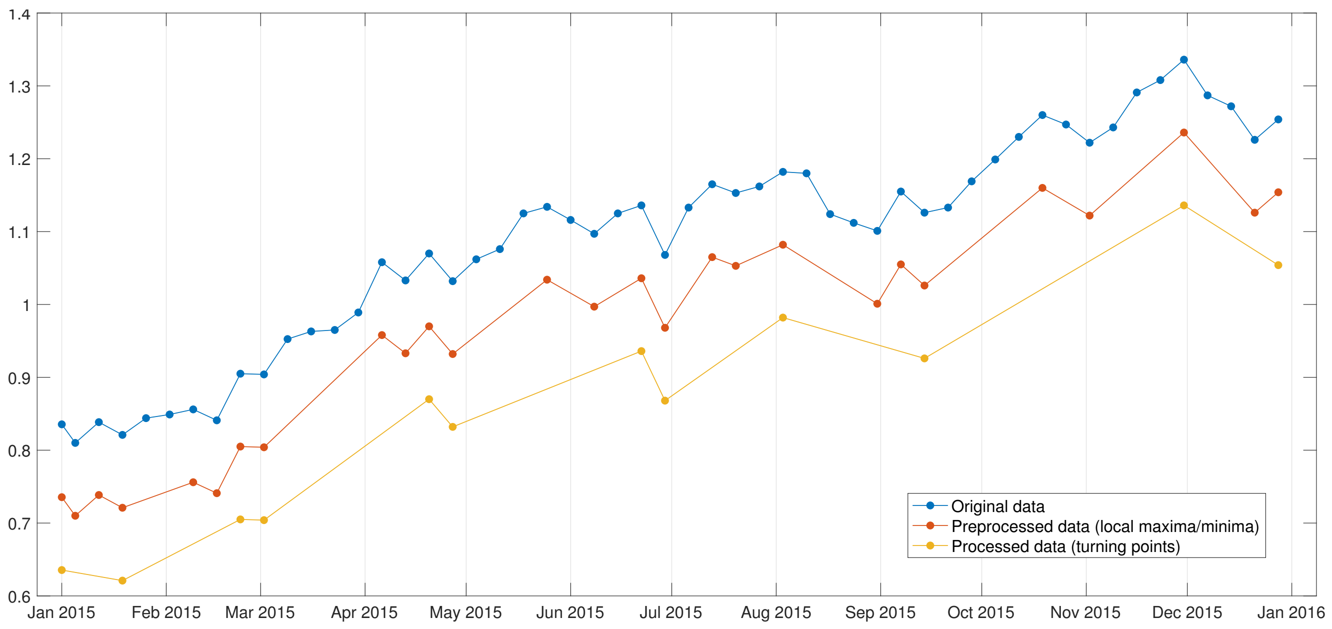

Figure 1: `A2A.MI` weekly 2015. Original data is in the correct position. The other two series are shifted down by 0.1 each.

4.2 Turning points in Python

Python is an excellent language for working with financial data. There is a strong support for downloading, manipulating, and visualising financial market data through popular open source libraries like `Pandas` and `Matplotlib`.

In particular, through `Pandas` we can automatically download data from a number of sources:

- Yahoo! Finance;
- Google Finance;
- St.Louis FED (FRED);
- Kenneth French's data library;
- World Bank;
- Google Analytics.

Focusing on Yahoo! Finance, which is our reference source, the set of available indicators is as follows:

- `Open`;
- `High`;
- `Low`;
- `Close`;

- `Volume`;
- `Adj. Close`.

Let us now go to the code, which is presented in listings 3 and 4 for convenience. The program takes in input the period of time to be considered, the name of the share and the level of deepness. The program is structured in four phases:

1. prepare, cleaning and convert the data in the time series;
2. find the local maximum and minimum points (often called *turning points* since they indicate the change in the trend of the stock during a period);
3. flatten the time serie at different level of deepness;
4. plot the results.

Actually, we are presenting a slightly different algorithm from the one in our reference paper. When looking for local maximum and minimum points, the conditions *greater/less* than were replaced with *greater than or equal to/less than or equal to*, and the points at the same level have been adjusted afterwards. This modification brought to a better alignment between the original data and the trend function. The results of some processing are shown in fig. 2.

```python
import pandas.io.data as web
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime

end = datetime.now()
start = datetime(end.year-1, end.month, end.day)
df = web.DataReader("A2A.MI", 'yahoo', start, end)

#change here the indicator
x = df["Close"]

# levels
N = 4

#value and date of the time series
v = []
d = []

#value and date of the TP of the time series
tpv = []
tpd = []

#value and date with N level of deepness of the time series
tpvl = []
tpdl = []

#from dictionary to list for the values
for key, value in x.iteritems():
    temp = value
    v.append(temp)
#from dictionary to list fot the dates
for key, value in x.iteritems():
    temp = key
    d.append(temp)


#delete a point if has the same value of the previous one
c = []
for i in range (0, len(v)-1):
    if(v[i]==v[i+1]):
        c.append(i+1)

for e in range(0, len(c)):
    del v[c[e]-e]
    del d[c[e]-e]
```

Listing 3: Python implementation (1).

```
#find the TP
tpv.append(v[0])
tpd.append(d[0])
for z in range(0,len(v)):
    if(z==len(v)-1):
        tpv.append(v[z])
        tpd.append(d[z])
    else:
        if (v[z]<=v[z-1] and v[z]<=v[z+1]):
            tpv.append(v[z])
            tpd.append(d[z])
        if (v[z]>=v[z-1] and v[z]>=v[z+1]):
            tpv.append(v[z])
            tpd.append(d[z])

tpvl = list(tpv)
tpdl = list(tpd)

for h in range(0, N):
    a = 0
    b = []
    #finds the points to flatten
    while (a<len(tpvl)-3):
        if (tpvl[a]<tpvl[a+1] and tpvl[a]<tpvl[a+2] and
 ↪  tpvl[a+1]<tpvl[a+3] and tpvl[a+2]<tpvl[a+3] and
 ↪  (abs(tpvl[a+1]-tpvl[a+2])<(abs(tpvl[a]-tpvl[a+2])+abs(tpvl[a+1]-tpvl[a+3])))):
            b.append(a+1)
            b.append(a+2)
            a = a+3
        elif (tpvl[a]>tpvl[a+1] and tpvl[a]>tpvl[a+2] and
 ↪  tpvl[a+1]>tpvl[a+3] and tpvl[a+2]>tpvl[a+3] and
 ↪  (abs(tpvl[a+1]-tpvl[a+2])<(abs(tpvl[a]-tpvl[a+2])+abs(tpvl[a+1]-tpvl[a+3])))):
            b.append(a+1)
            b.append(a+2)
            a = a+3
        elif (tpvl[a+1]==tpvl[a+3] and tpvl[a]==tpvl[a+2]):
            b.append(a+1)
            b.append(a+2)
            a = a+3
        else:
            a = a+1
    #delete the flattened points
    for c in range(0, len(b)):
        del tpvl[b[c]-c]
        del tpdl[b[c]-c]
```

Listing 4: Python implementation (2).

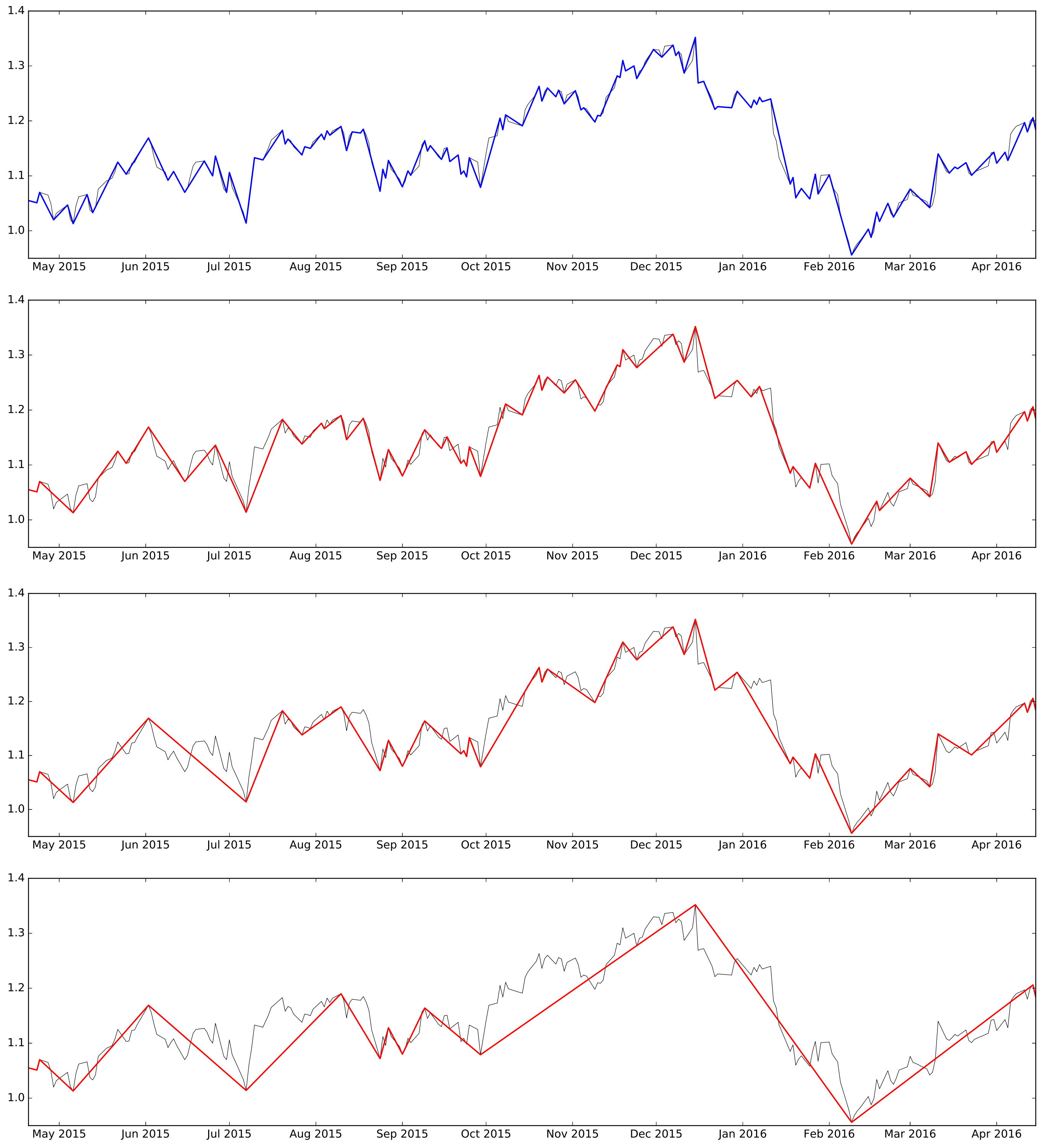

Figure 2: `A2A.MI` daily `Close` quotations from mid-april 2015 to mid-april 2016 after preprocessing (local maxima/minima), then after one, two and five iterations of the algorithm.

4.3 Other implementations

During our research, we also found an algorithm, developed in the 1970s, named after its three creators: Urs Ramer, David Douglas and Thomas Peucker[3]. We are presenting it rather informally, since it was initially developed for cartographic generalisation, and is now used for the processing of vector graphics.

However, an implementation of the algorithm is already available in MATLAB (and others may be easily found in different programming languages), which also gives quite good results (see fig. 3), so we find it appropriate to mention it in this report.

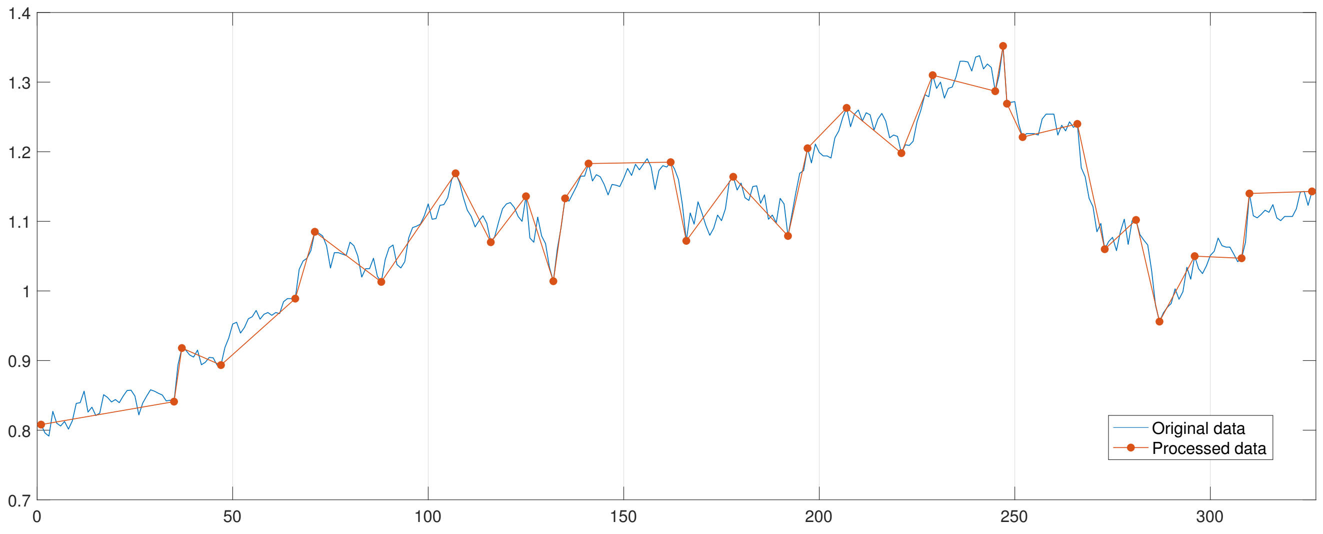

Figure 3: `A2A.MI` daily stock market data from January 2015 to April 2016.

After installing the add-on[4] and importing data, the user may issue the following command:

```
y = DouglasPeucker(M, epsilon, img);
```

The results are stored in `y` matrix. In the command, `M` is the matrix containing the time series data, `epsilon` represents the approximation value (in fig. 3, where points range roughly between 0.8 and 1.4, the value 0.05 was used for `epsilon`), and `img` is a boolean stating whether the user wants a plot as output (actually, the plot is of really low quality, and the user had better to plot `y` matrix with standard MATLAB commands).

[3]`https://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm`

[4]`http://www.mathworks.com/matlabcentral/fileexchange/41986-ramer-douglas-peucker-algorithm-demo`

Bibliography

DOI (Digital Object Identifier), when defined, serves as URL to retrieve the document. Documents may be accessible only after institutional log in (e.g., academic credentials).

[1] T.-c. Fu, "A review on time series data mining", *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 164–181, 2011. DOI: 10.1016/j.engappai.2010.09.007.

[2] J. Yu, J. Yin, D. Zhou, and J. Zhang, "A pattern distance-based evolutionary approach to time series segmentation", in *Intelligent Control and Automation: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006*, 2006, pp. 797–802. DOI: 10.1007/978-3-540-37256-1_99.

[3] F.-L. Chung, T.-C. Fu, V. Ng, and R. W. Luk, "An evolutionary approach to pattern-based time series segmentation", *Trans. Evol. Comp*, vol. 8, no. 5, pp. 471–489, Oct. 2004. DOI: 10.1109/TEVC.2004.832863.

[4] T.-c. Fu, F.-l. Chung, R. Luk, and C.-m. Ng, "Stock time series pattern matching: Template-based vs. rule-based approaches", *Engineering Applications of Artificial Intelligence*, vol. 20, no. 3, pp. 347–364, 2007. DOI: 10.1016/j.engappai.2006.07.003.

[5] ——, "Representing financial time series based on data point importance", *Engineering Applications of Artificial Intelligence*, vol. 21, no. 2, pp. 277–300, 2008. DOI: 10.1016/j.engappai.2007.04.009.

[6] C. Phetking, M. N. M. Sap, and A. Selamat, "Financial time series representation using zigzag based perceptually important points", in *Cognitive Informatics, 2009. ICCI '09. 8th IEEE International Conference on*, Jun. 2009, pp. 295–301. DOI: 10.1109/COGINF.2009.5250725.

[7] Z. Deng, F.-L. Chung, and S. Wang, "Clustering-inverse: A generalized model for pattern-based time series segmentation", *Journal of Intelligent Learning Systems and Applications*, vol. 3, no. 1, pp. 26–36, 2011. DOI: 10.4236/jilsa.2011.31004.

[8] J. Yin, Y.-W. Si, and Z. Gong, "Financial time series segmentation based on turning points", in *System Science and Engineering (ICSSE), 2011 International Conference on*, Jun. 2011, pp. 394–399. DOI: 10.1109/ICSSE.2011.5961935.

[9] V. Vo, L. Jiawei, and B. Vo, "A turning point method for stream time series prediction", in *Advanced Methods for Computational Collective Intelligence*, 2013, pp. 167–176. DOI: 10.1007/978-3-642-34300-1_16.

[10] T.-c. Fu, F.-l. Chung, K.-y. Kwok, and C.-m. Ng, "Stock time series visualization based on data point importance", *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1217–1232, 2008. DOI: 10.1016/j.engappai.2008.01.005.

[11] V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan, "Mining of concurrent text and time series", in *In proceedings of the 6 th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, 2000, pp. 37–44. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.1415.

[12] S.-D. Kim, J. W. Lee, J. Lee, and J. Chae, “A two-phase stock trading system using distributional differences”, in *Database and Expert Systems Applications: 13th International Conference, DEXA 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings*, 2002, pp. 143–152. DOI: 10.1007/3-540-46146-9_15.