Visual time series analysis

Fischer, Paul; Hilbert, Astrid

Published in:
Proceedings of COMPSTAT 2012

Publication date:
2012

Visual time series analysis

Paul Fischer, *Technical University of Denmark*, paf@imm.dtu.dk
Astrid Hilbert, *Linnaeus University, Sweden*, astrid.hilbert@lnu.se

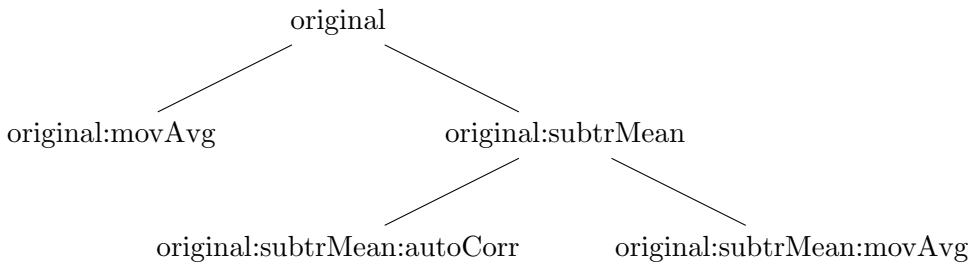
Abstract. We introduce a platform which supplies an easy-to-handle, interactive, extendable, and fast analysis tool for time series analysis. In contrast to other software suits like Maple, Matlab, or R, which use a command-line-like interface and where the user has to memorize/look-up the appropriate commands, our application is select-and-click-driven. It allows to derive many different sequences of deviations for a given time series and to visualize them in different ways in order to judge their expressive power and to reuse the procedure found.

For many transformations or model-fits, the user may choose between manual and automated parameter selection. The user can define new transformations and add them to the system. The application contains efficient implementations of advanced and recent techniques for time series analysis including techniques related to extreme value analysis and filtering theory. It has been successfully applied to time series in economics, e.g. reinsurance, and to vibrational stress data for machinery. The software is web-deployed, but runs on the user's machine, allowing to process sensitive data locally without having to send it away. The software can be accessed under http://www.imm.dtu.dk/~paf/TSA/launch.html.

Keywords. Computational Statistics, times series analysis, efficient algorithms

1 Introduction

When investigating extreme value statistics in econometrics, we often missed a tool that allows fast and easy experiments in order to check conjectures. The aim of the developed platform is to supply an easy-to-handle, interactive, and fast analysis tool for time series analysis. In contrast to other software suits like Maple, Matlab or R, which use a command-line like interface and where the user has to memorize/look-up the appropriate commands, our application is "select-and-click-driven". A time series and a transformation or filter are selected from lists and the result of applying the transformation to the series is shown and can also be selected to apply another transformation to. At any time the parameter settings and values used can be viewed od saved. This results in a rooted-tree data structure of time series with the original series at the top.



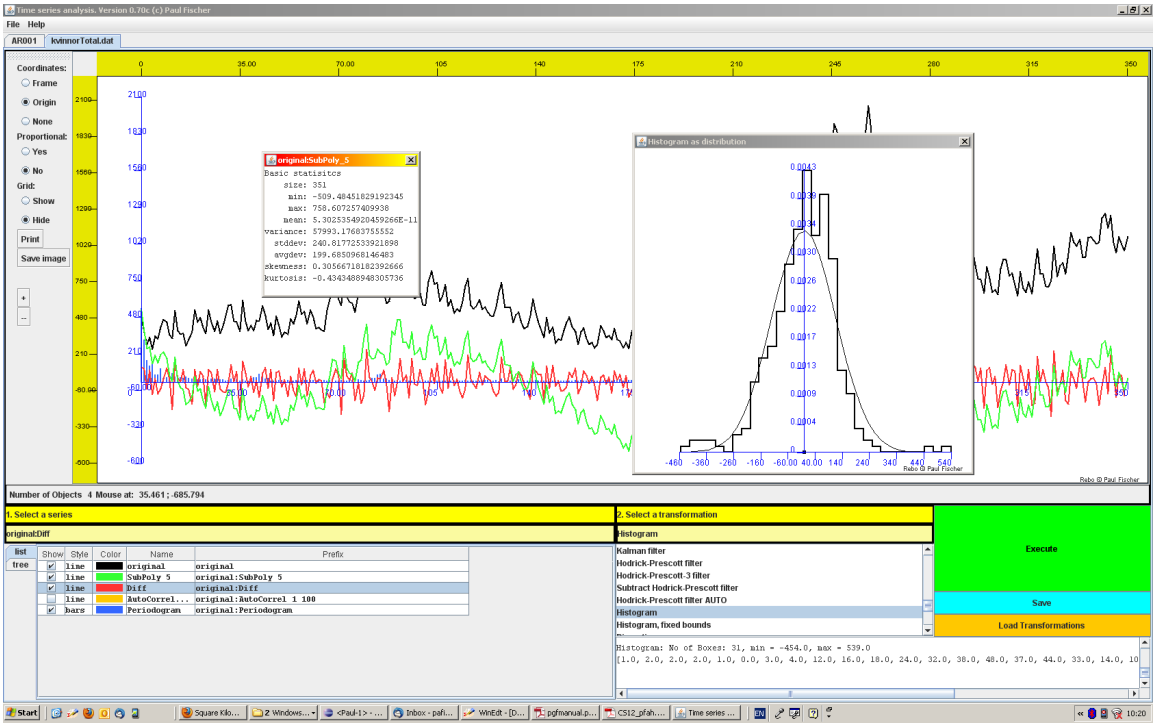


Figure 1. Screen shot of the application showing a time series with a number of derivations and, in separat windows, the histogram and basic statistics for two of the series.

The platform is meant as a tool for experiments for both scientists and students. It includes a number of predefined transformations and allows the user to add his/her own ones. This way user-specific solutions can be rapidly developed. Any derived sequence of transformation can be saved as a single super-transformation for later replay. Some features are not present in the web version because of security restrictions.

The transformations implemented so far include:

Smoothing: Computing or subtracting mean, moving average, polynomial regression, spline regression, trigonometric regression, Kalman filter, Hodrick-Prescot-filter (also in the version described in [3]), sine-cosine-transform,

Transformations: Differences, logarithms, logarithmic or arithmetic returns,

Derivations: Autocorrelation, partial autocorrellation, moving variance, periodogram, extremogram, (see [2]).

Parameters: Hurst index (see [5]), tail index (see [4]).

Models: Fitting AR-, MA-, ARMA-, ARCH-, GARCH-models, box plot.

Other: Random permutation, sorting, histogram, residual analysis.

2 Implementation

The application is programmed in Java, in order to guarantee platform-independence. It consists of three main packages, which have been developed at DTU and LNU. The core package "Lyngstat" contains the statistical routines, package "Rebo" provides generic graphics, and "TSA" (for time series analysis) implements the user interface.

In the implementation we have put focus on flexibility and maintenance. It is easy to add new tools or to add probes to existing tools in order to extract additional information. This is achieved by using an object-oriented implementation of the central concepts like time series, transformation, and also the mechanism for communicating between user interface and the statistical model. For computationally demanding transformations (like Fourier transform) we choose to make a safe reference implementation and a dirty-but-fast implementation. In the test phase both were executed in parallel and the results compared. In a few cases, concurrent/parallel implementations were made. These are however only activated for very large data set, as the overhead causes considerable delay on short series. Our aim to have execution times of under 1 msec for series of length less than 1000.

For the deployment we choose the Java web-start technology. By using a web-based solution instead of distributing an executable, the user will automatically receive the newest updates. Also the application is running stand-alone on the user's computer, not in a browser like an applet. This allows to read in data from the users computer and to save the results there. For security reasons, web-start applications are running in a "sandbox", so the user has to explicitly allow access to local files. This also poses some limitations to the web-start in contrast to the downloadable executable. For example the incorporation of user defined transformations and the memorization of super-transforms is only possible with with an downloaded executable.

3 Application

The software has been used for analyzing a number of different problems. In all cases it proved to be most helpful to perform a fast heuristic analysis of the data and to derive a stable procedure for the analysis of similar problems.

Here, we would like to briefly present the results for Swedish unemployment data and data gathered by measuring vibrations/accelleration of a mechanical device. In both cases we focus (amongst other) on structural breaks.

For the unemployment data for women in Sweden a cubic spline regression with 12 knots (11 intervals) was used to remove the trend. Then a moving variance with window width 40 was

computed and used to identify structural breaks [1]. The visual inspection of the original data suggests three regimes: $[0; 188]$, $[189; 303]$, and $[303; 352]$, see Figure 2.

As reference method we have implemented an automated method for detecting structural breaks in the variance. We have in a straightforward way adapted the method described for structural breaks of the mean as described in [6]. The main change is replacing the t-distribution by a F-distribution. Running this method with a confidence level of $\alpha = 0.05$ in the F-distribution delivered different regimes in the intervals $[0; 190]$, $[191; 304]$, and $[305; 352]$. The results are shown in Figure 2 and match the visual impression.

To investigate the effects of seasonal effects on structural breaks, a sine-cosine transformation was applied to the data after the spline regression. By sorting the absolute values of coefficient of the trigonometric functions decreasingly we concluded that the 20–25 largest were most significant. The trigonometric polynomial with the 22 largest coefficients was then subtracted to remove seasonal effects. Again a moving variance with window width 40 is computed and used to identify structural breaks, see Figure 3. Here the structural breaks were no longer as prominent as in the case without seasonal correction. The automated analysis suggested five regimes $[0; 67]$, $[68; 117]$, $[118; 191]$, $[192; 293]$, and $[294; 352]$. This is in good accordance with the visual impression, and exemplifies the influence the seasonal component on the occurrence of structural breaks. The results are shown in Figure 3.

The fact that the shape of the curves for the moving variances resembles the shape of the original data, but at a different scale, is coincidental.

The sequences of transformation with the relevant parameters have been saved as a new (super-)transformation together with the chosen parameters. This super transformation was then applied to time series generated by similar processes, i.e., unemployment data of men sampled at the same frequency, with similarly informative results.

The latter feature (being able to save an apply sequences of transformations) proved very helpful when analyzing the vibration patterns. This data consists of acceleration measurements sampled at 200Hz for 10 to 30 min. The task is to identify different regimes (machine halted, working normally, working overload) and to detect extremal events that can be hazardous. For the high-frequency data, the method used above, could not be tuned to reveal satisfactory results. It indicated too many structural breaks at unwanted positions. We have implemented two alternative methods. One uses changes in the average variance to identify breaks. The second one uses methods from Computational Geometry and is computationally very efficient.

Structural break identification by variances

An approach based on genetic programming is suggested in [1] for GARCH models. The idea is to fit AR-models to consecutive subsequences of the time series in order to minimize some target function, e.g., the description length. For the long time series under consideration with many 100.000 observations this was not an option because of the running time of many seconds, even minutes on larger sets.

The first approach we used was to divide the series $y_0, y_1, \ldots, y_N$ into k consecutive intervals

$$[t_0, t_1], [t_1 + 1, t_2], \ldots, [t_{k-1} + 1, t_k]$$

[1] Both parameters were chosen on previous experience and not changes afterwards.

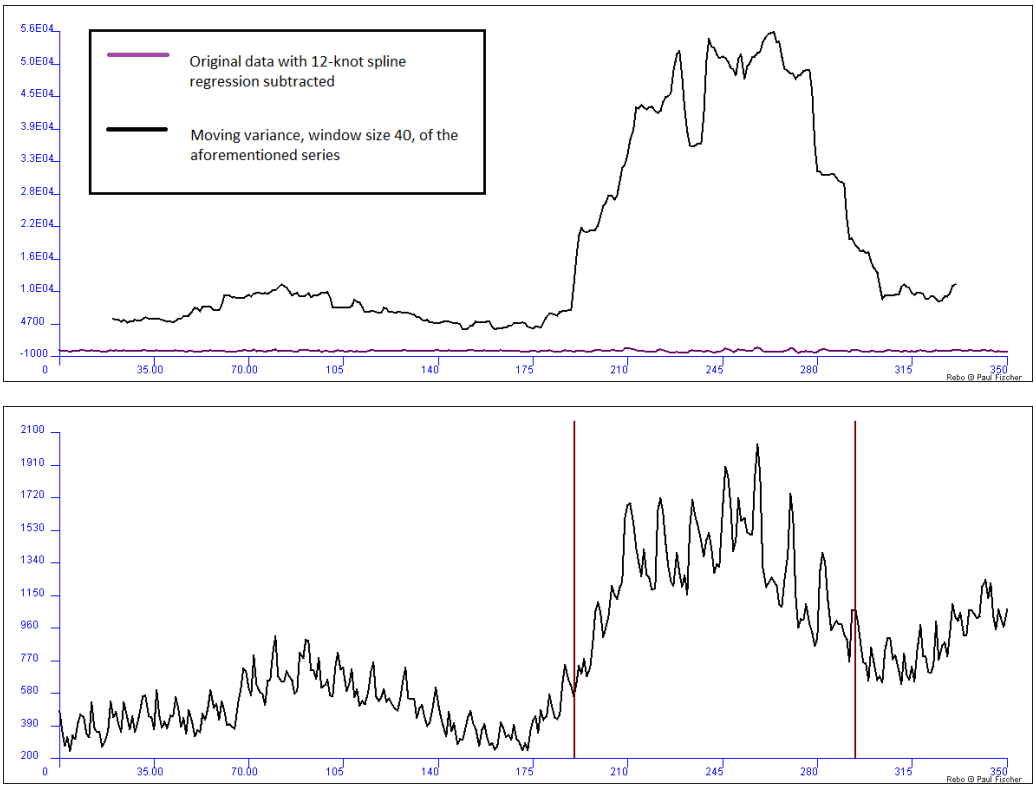


Figure 2. Unemployment amongst women in Sweden on a monthly basis. The figure at the top shows the moving variance of the data after removing the trend by subtracting a cubic spline regression with 12 knots (lower graph in the figure). Then the moving variance with window size 40 is computed (upper graph). The bottom figure shows the original data with the structural breaks found based on the variance as vertical lines.

where $0 = t_0 < t_1 < \cdots < t_k = N$. The t_i, for $i = 1, \ldots, k-1$, were chosen at random, but meeting the condition $a \leq t_i - t_{i-1} \leq b$ to ensure a certain minimum and maximum length of the intervals.

For each interval $[t_i, t_{i+1}]$ the empirical variance s_i^2 was computed. Then the null-hypotheses $s_i^2 = s_{i+1}^2$ was tested against $s_i^2 \neq s_{i+1}^2$ using F-distribution[2] with significance level 0.03. The level 0.03 was chosen empirically to give the best results. Values between 0.01 and 0.04 yielded good results in our simulations. If the null-hypothesis was rejected a structural break is inserted between the intervals at t_{i+1}. This approach produced better results, but introduced still too many structural breaks.

Therefore we tried a second approach where we took into consideration how close one is to the decision boundary. That is, we computed the difference d_i^0 of the ratio of the variances of consecutive intervals and the value of the F-distribution with probability level α.

$$d_i^0 = s_i^2 / s_{i+1}^2 - f_{k-1,\ell-1}(\alpha)$$

[2]Requires a careful implementation of the F-distribution to avoid numerical problems for large degrees of freedom.

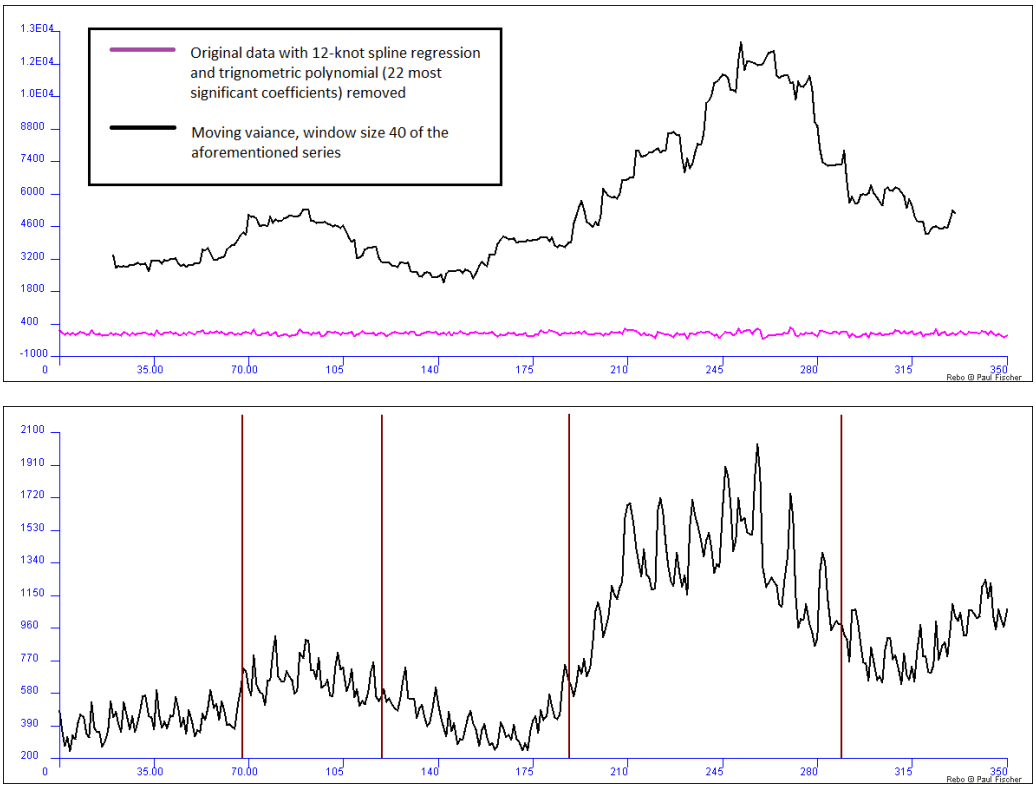


Figure 3. Unemployment amongst women in Sweden on a monthly basis. The figure at top shows moving average of the data after removing spline as in Figure 2 and then removing a trigonometric polynomial with the largest 22 coefficients (lower graph). Then the moving variance with window size 40 is computed (upper graph). The bottom figure shows the original data with the structural breaks found based on the variance as vertical lines.

where k, ℓ are the lengths of the intervals. If $-\beta < d_i^0 < \beta$ for some user chosen $\beta \in [0, \alpha]$ then we begin to move the common boundary t_{i+1} of the two internals back and forth. The boundary is moved deterministically in small steps, maintaining $t_i < t_{i+1} < t_{i+2}$. After moving t_{i+1} the variances s_i^2 and s_{i+1}^2 are updated as are k, ℓ and d_i is recomputed. Let $d_i^1, \ldots, d_i^s$ the values computed during this process.

If for some d_i^j the null-hypotheses is accepted ($s_i^2 = s_{i+1}^2$), we fix the interval boundary at the corresponding value and do not introduce a break.

Otherwise we set a break point at that value of t_{i+1} such that the corresponding d_i^j is maximal. With the appropriate choice of the parameters ($a = 500$, $b = 2000$, $\beta = 0.01$), this heuristic improved the result. Figure 6 shows the result for a series of length 380.000. A visual inspection shows that there still are breaks at of the breaks "inexplicable " locations.

Structural break identification by rectangles

We usually apply the method described below to the sequence of moving variances to identify structural breaks in the original data. However it can be applies to any series (with more or less

meaningful results).

Also in this approach the series $y_0, y_1, \ldots, y_N$ is divided into k consecutive intervals

$$[t_0, t_1], [t_1 + 1, t_2], \ldots, [t_{k-1} + 1, t_k]$$

where $0 = t_0 < t_1 < \cdots < t_k = N$.

For every interval we compute the bounding rectangle. That is the smallest axes-aligned rectangle containing all data points. Formally it is given by

$$[t_i\,;\, t_{i+1} - 1] \times [\min\{y_k \mid t_i \le k < t_{i+1}\}\,;\, \max\{y_k \mid t_i \le k < t_{i+1}\}] \tag{1}$$

The interval boundaries are then varied, maintaining $t_i < t_{i+1} < t_{i+2}$, and $a \le t_i - t_{i-1} \le b$ to ensure a certain minimum and maximum length of the intervals. The aim is to find a rectangle cover of the series with a small volume. To avoid over-fitting[3] we we also want to keep the number of rectangles small.

To this end we use a genetic algorithm which varies the t_i and inserts new ones or deletes existing ones. The fitness function f of the genetic algorithm depends on the current interval boundaries and incorporates the area and the number of rectangles. Formally

$$f(t_0, \ldots, t_k, k) = \left(\sum_{i=0}^{k-1} V_i \right) + \beta k \tag{2}$$

Equation (2) can be evaluated extremely efficiently, see below. The variable β is the penalizing term for the number of intervals. The term $\sum_{i=0}^{k-1} V_i$ depends on the range of the values y_j of the time series and its length. Therefore β has to be scaled accordingly. As of now we use $V_0 \sum_{i=0}^{k-1} V_i$ for the first term in (2) where $V_0 = (n(\max y_i - \min y_i))-1$ is the volume of the bounding rectangle of the whole series. This limits the term to $[0, 1]$.

The genetic algorithm attempts to minimize $f(t_0, \ldots, t_k, k)$ by varying k (the number of intervals) and the location of the interval boundaries t_i. We currently use a cross-over-and-mutation approach similar to that used in [1].

For the unemployment data some results are in Figure 4. If the penalizing term β is large (0.15) the result nicely fits the intuitive impression. In the top row in Figure 4, the solution is $[0; 189]$, $[190; 310]$ and $[311; 352]$ where the visual inspection suggest $[0; 188]$, $[189; 303]$, and $[303; 352]$. Choosing β small (0.02) leads to over-fitting of the rectangles and unmotivated breaks, as can be seen in the lower row in Figure 4

While experimenting with different settings of β we found out that the rectangle method might be useful to also identify the transition phase between different regimes. If these are reflected by an increasing values of the moving variances then a medium setting of the parameter β results in vertically elongated rectangles. This analysis requires a suitably chosen scaling of the ordinate axis. Figure 5 shows an example.

The strong point of this method is that it is computationally extremely efficient. In contrast to other techniques which require to fit a model to (parts of) the time series, e.g [1], we only have to evaluate (2) and perform a crossover-mutation in the genetic algorithm. For evaluating (2), one has to compute the volumes V_i in (1). With a preprocessing of the initial data $y_0, \ldots, y_N$ one can build a data structure (a min-max-tree) which allows to compute $\min\{y_i \mid a \le i \le b\}$

[3] A minimum-area rectangle cover can be achieved by using a 0×0 rectangle at evert data point.

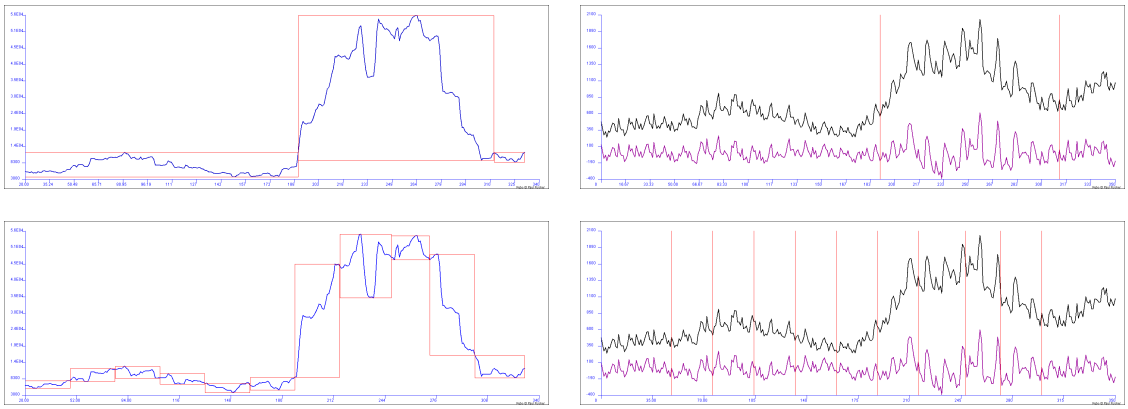


Figure 4. Two examples of the rectangle method for finding structural breaks. In the top row we use $\beta = 0.15$ resulting in a good identification of the breaks. In the bottom row we use $\beta = 0.02$, resulting in over-fitting and unmotivated breaks. At left the series of the moving average with window size 40 is shown. The average is computed on the original data with a 12-knot spline regression subtracted. At right the original series and the one with the spline subtracted are shown, along with the locations of the structural breaks as vertical lines.

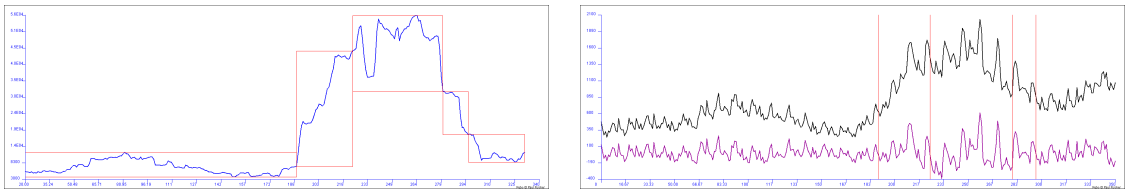


Figure 5. An example of running the rectangle method with $\beta = 0.045$. Two rectangles are found at the parts where the moving average strongly increases and decreases, indicating the areas of transition between regimes.

and $\max\{y_i \mid a \leq i \leq b\}$ for any intreval $[a; b]$ in time $O(\log(N))$. Building the data structure takes time $O(N)$. Thus the expession in Equation (1) can be evaluated in time $O(\log(N)$ and the fitness function (2) in time $O(k \log(N))$, where k is the number of rectangles. A crossover or a mutation of genetic algorithm can be performed in time $O(k)$. The instructive description in [1] suggest that the time for such an operation is $O(N)$, but that is surely not used by authors. Altogether this means that an iteration of the genetic algorithm (evaluating the the fitness of the current solution, generating a new one and selecting the "best" ones can be performed in time $O(k \log(N))$. Usually $k << N$ ($k <<\leq O(\log(N))$ seems reasonable) thus the rectangle method can try out much more possibilities of the choices of the interval boundaries t_i than other approaches.

Together with engineers of the company involved, a few time series were analyzed using the above mentioned methods. The interactive and visual capabilities of the application made it possible to quickly derive a sequence of transformations which revealed the features relevant for the application. This sequence was then applied to the remaining data series which delivered very good results. Clearly the method based on minimum volume rectangles yielded much better results for the practitioners. Figures 6 and 7 show the results.

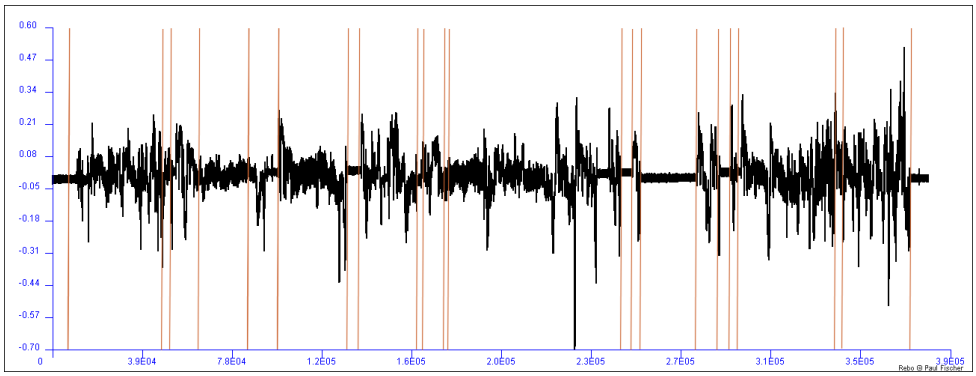


Figure 6. The result of the heuristic structural break detection based on variances. The vertical lines are the breaks found.

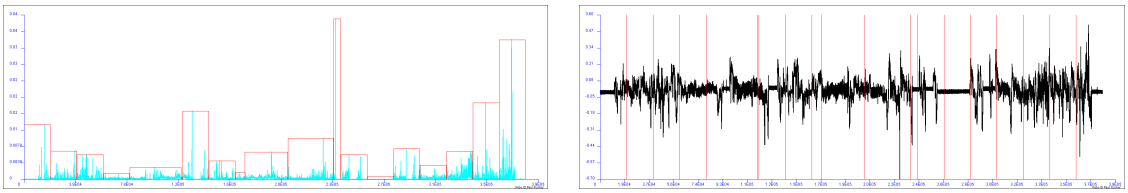


Figure 7. The result of the heuristic structural break detection based on rectangles. At left, the moving variance with window size 40 and rectangles found are shown. At right, the structural breaks are shown on the original data as vertical lines.

Bibliography

[1] R.A. Davis, T. Lee, and G. Rodriguez-Yam. Break detection for a class of nonlinear time series models. Technical report, 2008.

[2] R.A. Davis and Th. Mikosch. The extremogramm: A correlogramm for extreme events. *Bernoulli*, 15(4):977–1009.

[3] A. Dermoune and N. Djehiche, B. andRahmania. Consistant estimator of the smoothing parameter in the hodrick-prescott filter. *J. JapanStatist.Soc.*, 8(2):225–241, 2008.

[4] P. Embrechts, C. Klppelberg, and Th. Mikosch. *Modelling Extremal Events for Insurance and Finance*. Springer Verlag, Berlin, 2005.

[5] B.B. Mandelbrot and J.W. Van Ness. ractional brownian motions, fractional noises and applications. *SIAM Rev.*, 10:422–437, 1968.

[6] S. N. Rodionov. A sequential algorithm for testing climate regime shifts. *Geophysical Research Letters*, 2004.