Available online at www.sciencedirect.com

ScienceDirect

Computational Statistics and Data Analysis 53 (2009) 2354–2362

COMPUTATIONAL STATISTICS & DATA ANALYSIS

www.elsevier.com/locate/csda

Online analysis of time series by the Q_n estimator

Robin Nunkesser[a,*], Roland Fried[b], Karen Schettlinger[b], Ursula Gather[b]

[a] *Fakultät für Informatik, TU Dortmund, 44221 Dortmund, Germany*
[b] *Fakultät Statistik, TU Dortmund, 44221 Dortmund, Germany*

Available online 4 March 2008

Abstract

A fast update algorithm for online calculation of the Q_n scale estimator is presented. This algorithm allows robust analysis of high-frequency time series in real time. It provides reliable estimates of a time-varying volatility even if many large outliers are present and it offers good efficiency in the case of clean Gaussian data.
© 2008 Elsevier B.V. All rights reserved.

1. Introduction

The increasing availability of high-frequency data in financial markets and many other fields requires fast, automatic, and reliable methods which can extract the relevant information from the data in real time. Measurement artifacts can influence the output of such analyses severely. High-frequency data are especially susceptible to errors: as stated by Brownlees and Gallo (2006), "the higher the velocity in trading, the higher the probability that some error will be committed in reporting trading information".

Many preprocessing procedures for automatic data cleaning and outlier detection have been suggested. As it is well known that non-robust estimators like empirical means and standard deviations can be strongly mislead by outliers, we should not rely on such methods in automated applications, not even for data cleaning. Robust estimators which are able to resist isolated outliers and patches of outlying values should be preferred. Robust methods even allow us to work with the raw data. Nevertheless, the use of robust methods in time series analysis is not widely established yet, especially in the online context. The computational demands of naive algorithms are typically much higher than those of non-robust methods, causing computation times which are unacceptably large, especially in applications to ultra-high-frequency data.

In this paper we discuss robust scale estimators in time series analysis, which allow us to extract possibly time-varying volatilities in the presence of outliers, see Gather and Fried (2003) and Gelper et al. (2007). These scale estimators can also be applied to estimate the autocorrelations within the process (Ma and Genton, 2000). Moreover, they can be used to standardize test statistics, e.g. for robust level shift detection (Fried, 2007; Nunkesser et al., in press).

* Corresponding author. Tel.: +49 231 755 5132; fax: +49 231 755 2047.
E-mail addresses: robin.nunkesser@tu-dortmund.de (R. Nunkesser), fried@statistik.tu-dortmund.de (R. Fried), schettlinger@statistik.tu-dortmund.de (K. Schettlinger), gather@statistik.tu-dortmund.de (U. Gather).

0167-9473/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2008.02.027

We focus on the robust Q_n estimator of scale (Rousseeuw and Croux, 1993) applied to time series. This estimator is defined as a multiple of an order statistic of all pairwise absolute differences between data points $x_1, \dots, x_n \in \mathbb{R}$:

$$Q_n(x_1, \dots, x_n) = c_n \cdot \{|x_i - x_j| : 1 \le i < j \le n\}_{(\ell)}, \qquad \ell = \binom{\lfloor n/2 \rfloor + 1}{2}, \tag{1}$$

where c_n denotes a finite-sample correction factor achieving unbiasedness at Gaussian samples. For data in general position the breakdown point of Q_n is about 50%, i.e. the estimate is bounded and stays away from zero even if almost 50% of the data are contaminated arbitrarily. Another, more widely-known scale estimator with this property is the *median absolute deviation about the median* (MAD):

$$\text{MAD} = d_n \cdot \text{med}\,\{|x_i - \text{med}\{x_1, \dots, x_n\}| : i = 1, \dots, n\}, \tag{2}$$

where again d_n yields unbiasedness at Gaussian samples. For independent Gaussian data, Q_n is less variable than other high-breakdown point scale estimators. Its asymptotic efficiency of 82% (relative to the empirical standard deviation) is much larger than the asymptotic efficiency of the MAD, which is only 36%. A drawback of Q_n has been its computational complexity. Calculation of the MAD from n data points needs $O(n)$ computation time, and its value can be updated in $\mathcal{O}(\log n)$ time when applying it to moving time windows of n subsequent observations when analyzing locally stationary data (Bernholt et al., 2006). On the other hand, a straightforward implementation of Q_n would result in a computation time of $\mathcal{O}(n^2)$. Croux and Rousseeuw (1992) provide an offline algorithm for Q_n which needs $\mathcal{O}(n \log n)$ time.

Another class of robust estimators of scale which combine high breakdown point and good efficiency are τ estimators (Maronna and Zamar, 2002). They are defined by

$$\tau(x_1, \dots, x_n) = \frac{\hat{\sigma}^2}{n} \sum_{i=1}^{n} \rho\left(\frac{x_i - \hat{\mu}}{\hat{\sigma}}\right), \tag{3}$$

where $\hat{\sigma}$ is a highly robust initial estimate of scale, $\hat{\mu}$ is a robust location estimate and ρ is a weight function. The τ estimator implemented in the R package `robustbase` uses the MAD as initial scale estimate, a weighted mean with weights based on Tukey's biweight applied to robustly scaled distances from the sample median, and $\rho_c(u) = \min\{c^2, u^2\}$ with the default value $c = 3$.

Section 2 describes a new update algorithm for the Q_n estimator. This algorithm is easy to implement and allows online application since it is substantially faster in practice than the offline algorithm. It allows us to incorporate incoming new observations and to remove old data quickly when using a moving time window. It can also be used for online computation of the Hodges–Lehmann location estimator and the *medcouple* estimator (Brys et al., 2004). Section 3 compares the performance of the Q_n estimator with the MAD, a trimmed standard deviation, and a τ estimator of scale for online extraction of time-varying volatilities. Computation times are analyzed for different window widths to show the practical relevance of the new update algorithm. Finally, Section 4 gives concluding remarks.

2. An update algorithm for the Q_n estimator

To analyze the scale of a time series $x_1, \dots, x_N$ online, we apply the Q_n estimator (1) at each time t to a time window of length $n \le N$, which contains the observations $x_{t-n+1}, \dots, x_t$. Instead of calculating Q_n for each window from scratch, we use an *update* algorithm. This means that for each move of the window from t to $t+1$ all stored information concerning the oldest observation x_{t-n+1} is deleted and new information concerning the incoming observation x_{t+1} is inserted. Note that this algorithm is not restricted to moving time windows; it can also handle arbitrary sequences of deletions and insertions of data points.

For offline computation of Q_n, Croux and Rousseeuw (1992) suggest the algorithm of Johnson and Mizoguchi (1978) with an optimal running time of $\mathcal{O}(n \log n)$ for n observations. Therefore, an optimal update algorithm for the Q_n estimator needs at least $\mathcal{O}(\log n)$ time for insertion or deletion.

In the following, we construct an online version of the algorithm of Johnson and Mizoguchi (1978). It computes an arbitrary, say kth order statistic in a multiset of form $\mathcal{X} + \mathcal{Y}$, where $\mathcal{X} + \mathcal{Y}$ is the multiset $\{x_i + y_i \mid x_i \in \mathcal{X} \text{ and } y_i \in \mathcal{Y}\}$ for $\mathcal{X} = (x_1, \dots, x_n)$ and $\mathcal{Y} = (y_1, \dots, y_n)$ n-tuples of real numbers. This algorithm can be used to compute Q_n, the Hodges–Lehmann location estimator (HL), and the MC_n estimator (see Brys et al. (2004) or Nunkesser et al. (in press)).

The optimal $\mathcal{O}(\log n)$ time bound for online computation for $k = 1$ was achieved by Bespamyatnikh (1998). Smid (1991) suggests to use a buffer of possible solutions to get an online algorithm for $k = 1$. Computation of the Q_n estimator requires larger k. Hence, we generalize the idea of using a buffer to arbitrary values of k, because it is easy to implement and achieves a good running time in practice. Theoretically, the worst case amortized time per update is possibly not better than that for the offline algorithm, i.e. $\mathcal{O}(n \log n)$. However, no better lower bound than $\mathcal{O}(n \log n)$ is known for $k = \mathcal{O}(n^2)$ so far and we show that our algorithm runs substantially faster for many data sets.

It is convenient to visualize the algorithm of Johnson and Mizoguchi working on a partially sorted matrix $B = (b_{ij})$ with $b_{ij} = x_{(i)} + y_{(j)}$, although B is never constructed. Here, $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ and $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(n)}$ are the elements of $\mathcal{X}$ and $\mathcal{Y}$, increasingly ordered. The algorithm utilizes the monotonicity of the matrix in the rows and columns, i.e. $x_{(i)} + y_{(j)} \leq x_{(i)} + y_{(\ell)}$ and $x_{(j)} + y_{(i)} \leq x_{(\ell)} + y_{(i)}$ for $j \leq \ell$. In consecutive steps, a matrix element is selected, regions in the matrix are determined with values definitely smaller or certainly larger than this element, and parts of the matrix are excluded from further consideration according to a case differentiation. As soon as less than n elements remain for consideration, they are sorted and the element sought for is returned. The algorithm may easily be extended to compute a *buffer* $\mathcal{B}$ of size s of matrix elements $b_{(k-\lfloor (s-1)/2 \rfloor):n^2}, \ldots, b_{(k+\lfloor s/2 \rfloor):n^2}$.

Firstly, we describe the framework of the online algorithm and later give the details on how to insert and delete elements.

Algorithm 1. ONLINEJM(set $\mathcal{X}$, set $\mathcal{Y}$, k, window width n, buffer size s)

```
Set $\mathcal{X}_w := \{x_1, \ldots, x_n\}$ and $\mathcal{Y}_w := \{y_1, \ldots, y_n\}$
Compute the $k$th order statistic of $\mathcal{X}_w + \mathcal{Y}_w$ and a buffer $\mathcal{B}$ of size $s$ offline
do for $n \leq t < |\mathcal{X}|$
  Call INSERT($x_{t+1}$,$\mathcal{X}_w$,$\mathcal{B}$) and DELETE($x_{t-n+1}$,$\mathcal{X}_w$,$\mathcal{B}$)
  Call INSERT($y_{t+1}$,$\mathcal{Y}_w$,$\mathcal{B}$) and DELETE($y_{t-n+1}$,$\mathcal{Y}_w$,$\mathcal{B}$)
  if DELETE($y_{t-n+1}$,$\mathcal{Y}_w$,$\mathcal{B}$) returned that the $k$th order statistic of $\{x_{t-n+1}, \ldots, x_{t+1}\} + \{y_{t-n+1}, \ldots, y_{t+1}\}$ is in $\mathcal{B}$
    Determine the $k$th order statistic directly
  else
    Recalculate the $k$th order statistic and the buffer $\mathcal{B}$ offline
  endif
enddo
return the determined order statistics
```

To speed up online computation, we ensure fast insertion and deletion and few recalculations. To achieve this, we use balanced trees, more precisely indexed AVL-trees, as the main data structure. Inserting, deleting, finding and determining the rank of an element takes $\mathcal{O}(\log n)$ time in this data structure. Moreover, every element in the balanced tree has two pointers allowing access to the element pointed at in time $\mathcal{O}(1)$. In detail, we store $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{B}$ in separate balanced trees and let the pointers of an element $b_{ij} = x_{(i)} + y_{(j)} \in \mathcal{B}$ point to $x_{(i)} \in \mathcal{X}$ and $y_{(j)} \in \mathcal{Y}$, respectively. The first and second pointers of an element $x_{(i)} \in \mathcal{X}$ point to the smallest and largest element such that $b_{ij} \in \mathcal{B}$ for $1 \leq j \leq n$. The pointers for an element $y_{(j)} \in \mathcal{Y}$ are defined analogously.

The following algorithms handle insertions into and deletions from $\mathcal{X}$. Insertions and deletions for $\mathcal{Y}$ work analogously.

Algorithm 2. DELETE(x_{del},set $\mathcal{X}$, buffer $\mathcal{B}$)

```
Search in $\mathcal{X}$ for $x_{\text{del}}$
Determine the rank $i$ of $x_{\text{del}}$ and the elements $b_s$ and $b_g$ pointed at
Determine $y_{(j)}$ and $y_{(\ell)}$ with the help of the pointers of $b_s$ and $b_g$ such that $b_s = x_{(i)} + y_{(j)}$ and $b_g = x_{(i)} + y_{(\ell)}$
Find $B_m := \{x_{(i)} + y_{(m)} \in \mathcal{B} \mid j \leq m \leq \ell\}$
Delete the elements in $B_m$ from $\mathcal{B}$
Delete $x_{\text{del}}$ from $\mathcal{X}$
Update the affected pointers accordingly
Compute the new position of the $k$th element in $\mathcal{B}$
return $\mathcal{X}$, $\mathcal{B}$ and whether the $k$th element is still in $\mathcal{B}$
```

Algorithm 3. INSERT(x_{ins},set $\mathcal{X}$, buffer $\mathcal{B}$)

```
Determine the smallest element $b_s$ and the greatest element $b_g$ in $\mathcal{B}$
Determine with a binary search the smallest $j$ such that $x_{\text{ins}} + y_{(j)} \geq b_s$ and the greatest $\ell$ such that $x_{\text{ins}} + y_{(\ell)} \leq b_g$
Compute all elements $B_m := \{x_{\text{ins}} + y_{(m)} \mid j \leq m \leq \ell\}$
Insert the elements in $B_m$ into $\mathcal{B}$
Insert $x_{\text{ins}}$ into $\mathcal{X}$
Update pointers to and from the inserted elements accordingly
Compute the new position of the $k$th element in $\mathcal{B}$
return $\mathcal{X}$, $\mathcal{B}$ and whether the $k$th element is still in $\mathcal{B}$
```

It is easy to see that we need a maximum of $\mathcal{O}(|\text{deleted elements}| \cdot \log n)$ and $\mathcal{O}(|\text{inserted elements}| \cdot \log n)$ time for deletion and insertion, respectively.

To achieve few recomputations of the buffer $\mathcal{B}$ and a space efficient algorithm, the buffer size should be chosen as $\mathcal{O}(n)$, with n being our standard choice. Note that the size of $\mathcal{B}$ is not bounded, i.e. it may increase during the algorithm run. It is possible to introduce e.g. linear bounds on the size of $\mathcal{B}$ and to recompute $\mathcal{B}$ if these bounds are violated to achieve that the algorithm only needs $\mathcal{O}(n)$ space. For arbitrary sequences of insertions and deletions, the size of $\mathcal{B}$ can vary more and we may have to recompute the buffer more often than for moving time windows. However, since size bounds increase the amortized computation time and the algorithm needs $\mathcal{O}(n^2)$ space in the worst case, size bounds should only be used for very large n, say, if n is in the range of tens of thousands.

To determine the running time we have to consider first the number of elements in the buffer that depend on the inserted or deleted observation.

Theorem 1. *For a time series with constant mean and identically distributed noise terms the expected time needed for insertion or deletion of data points is* $\mathcal{O}(\log n)$.

Proof. For a time series, consisting of a constant mean with added identically distributed error terms, data points are *exchangeable* in the sense that each rank of a data point in the set of all data points occurs with equal probability. Assume w.l.o.g. that we only insert into and delete from $\mathcal{X}$. Consider for each rank i of an element in $\mathcal{X}$ the number $\left|\{j \mid b_{ij} \in \mathcal{B}\}\right|$ of buffer elements depending on it. With $\mathcal{O}(n)$ elements in $\mathcal{B}$ and equiprobable ranks of the observations inserted into or deleted from $\mathcal{X}$, the expected number of buffer elements depending on an observation is $\mathcal{O}(1)$. Hence, the expected number of buffer elements to delete or insert during an update step is also $\mathcal{O}(1)$ and the expected time we spend for the update is $\mathcal{O}(\log n)$. □

At times, we have to recompute the buffer which needs $\mathcal{O}(n \log n)$ time and increases the amortized time needed per update. The frequency of recomputations depends on the amount the kth element may move in the buffer. In the worst case, this may happen very often, leading to the same running time as the offline algorithm. However, this is not very likely for real data sets, as our simulations in Section 3 show. With equiprobable ranks as e.g. in Theorem 1, the expected position of the kth element in the buffer after a deletion and a subsequent insertion is the same as before. Thus, we expect to recompute the buffer very rarely.

3. Online estimation of time-varying volatilities

In this section we illustrate the use of the Q_n estimator for online extraction of time-varying volatilities using a moving time window. The advantages of Q_n for robust scale estimation from time series have already been pointed out in Gather and Fried (2003) in a setting with some time delay. Here, a moving MAD estimator, a moving 10%-trimmed standard deviation, and a τ estimator of scale are considered for comparison with Q_n. The MAD is included as well-known robust scale estimator, while the 10%-trimmed standard deviation has been suggested by Brownlees and Gallo (2006) for data cleaning. The τ estimator of scale is included as a further robust and efficient competitor. Furthermore, we demonstrate the gain in computation time when using the new update algorithm as compared to the offline algorithm on the four data sets.

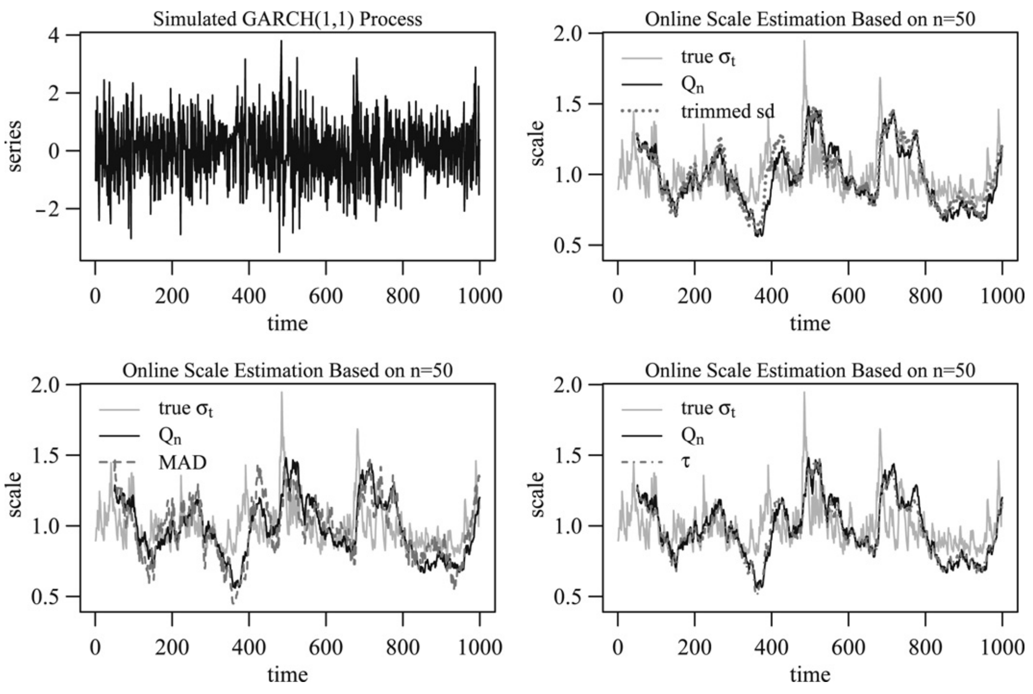

Fig. 1. Simulated GARCH(1,1) time series (top left) and online estimated volatilities by the 10%-trimmed standard deviation, the MAD and a τ estimate of scale in comparison to Q_n and the true scale.

3.1. Simulated time series

A benchmark model which is commonly used to estimate and predict volatility processes is given by the GARCH(1,1) model proposed by Bollerslev (1986):

$$X_t = \sigma_t \varepsilon_t, \quad t \in \mathbb{Z},$$

where ε_t is Gaussian white noise with zero mean and unit variance and σ_t is a time-varying volatility coefficient; more precisely, the conditional variance $\sigma_t^2 = Var(X_t | X_{t-1}, X_{t-2}, \ldots)$ is given by

$$\sigma_t^2 = \alpha_0 + \alpha_1 X_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

with parameters $\alpha_0 > 0$ and $\alpha_1, \beta_1 \geq 0$. Note that GARCH(1,1) processes with the restriction $\alpha_1 + \beta_1 < 1$ are stationary and thus fulfill the assumptions of Theorem 1.

Fig. 1 depicts a time series of length $N = 1000$ generated from a GARCH(1,1) model with coefficients $\alpha_0 = 0.1$, $\alpha_1 = 0.1$, and $\beta_1 = 0.8$. Additionally, the volatilities estimated by Q_n, the 10%-trimmed standard deviation, the MAD, and the τ estimator, all based on a window width of $n = 50$, are displayed in comparison with the true time-varying conditional volatility σ_t.

All methods track the volatilities rather well. However, since for online analyses only past observations are taken into account, a sudden increase or decrease in volatility is traced by all methods with some time delay, see e.g. before time 400. The MAD estimations show the largest variability, see e.g. around time 200 or 700–800 in the bottom left panel of Fig. 1, while the 10%-trimmed standard deviation is slightly smoother but not quite as much as the Q_n. For this simulation setting, the difference between τ and Q_n is negligible. All these findings could be expected because of the known Gaussian efficiencies and robustness properties of the different estimators. Accordingly, we expect these results to apply also for other time series with structures comparable to those considered here.

As a second example we examine a time series of length $N = 1000$ which possesses a piecewise constant volatility σ_t of 1, 5, 1, 3, and 5, in time periods of length 200, 250, 150, 250, and 150, respectively. Such models with piecewise constant volatility are suggested by Mercurio and Spokoiny (2004) to approximate financial time series.

The simulated time series shown in Fig. 2 (top left panel) consists of independent normal errors with standard deviations given above, plus 10% positive additive outliers of size $6\sigma_t$ at random time points. Fig. 2 further shows

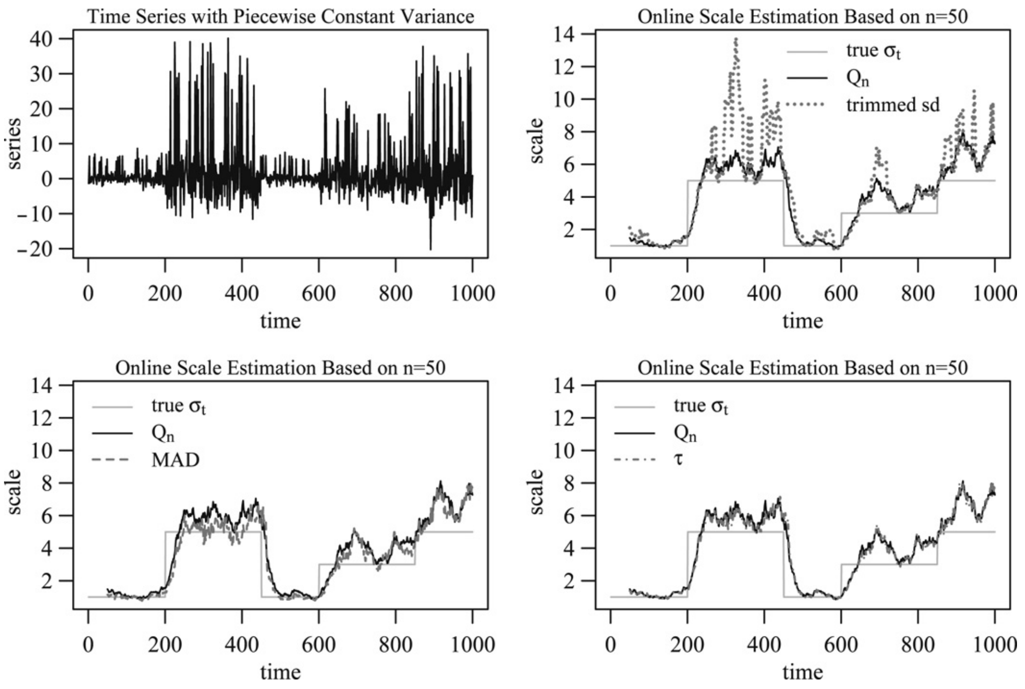

Fig. 2. Time series with piecewise constant variance (top left) and online estimated volatilities by the 10%-trimmed standard deviation, the MAD and a τ estimate of scale in comparison to Q_n and the true scale.

the volatilities estimated by Q_n, the 10%-trimmed standard deviation, the MAD, and the estimator τ, each using a window width of $n = 50$. Obviously, the 10%-trimmed standard deviation is not robust enough to resist the positive outliers which is clearly apparent e.g. in the time period between 200 and 450 in the top right panel of Fig. 2. The other estimators are robust enough to cope with the outliers. Again, the MAD shows more variation than Q_n, see e.g. around times 250–400, while again τ and Q_n do not yield obviously different results in this setting.

The third example consists of a time series of length $N = 1000$ with positive level shifts of size 1, 3 and 5 at times 201, 401 and 601, respectively, and a negative level shift of size -9 at time 801. Thus, the observations vary around the values 0, 1, 4, 9 and 0. The errors are generated from an AR(1) model which is defined by

$$X_t = \varphi X_{t-1} + \varepsilon_t, \quad t \in \mathbb{Z},$$

where ε_t denotes Gaussian white noise with zero mean and unit variance. The variance in this model is $\sigma_t^2 = 1/(1-\varphi^2)$, independent of t, see e.g. Brockwell and Davis (1996).

Fig. 3 shows a time series simulated according to the settings described above where, in the AR(1) model, the parameter $\varphi = 0.4$ was chosen for moderate correlation between successive observations resulting in a marginal standard deviation of $\sigma_t = 1.091$ for all t. We observe that all estimations lie close to the true value of σ_t, but that shifts cause strong biases. Larger shifts have larger impacts on the online scale estimation, see e.g. after times 600 and 800. Online scale estimation by Q_n is the least affected one by a shift. This can be explained by the fact that it does not need a level estimate for centering. Again, the MAD shows more variation than the other methods.

3.2. *Time series of stock returns*

Fig. 4 depicts a real data set of the first differences of the logarithms of the 11928 daily closing prices of Bayer AG stocks between January 4, 1960 and July 17, 2007. The variability of these returns is important for assessing derivate finance products like options. Because of unexpected events like regulatory changes, technological advances, natural catastrophes or accidents, financial time series can contain arbitrarily large outliers. Therefore, we should apply a robust scale estimator to evaluate the (local) variability. In this particular case, this is especially reasonable since the underlying prices are not adjusted for dividends and splits.

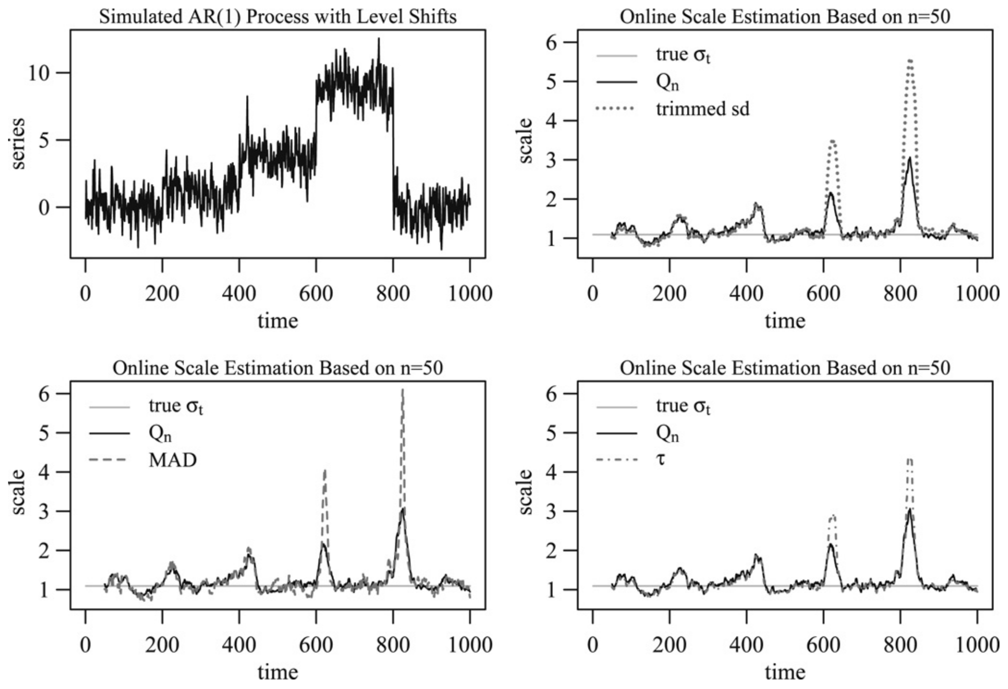

Fig. 3. Time series with piecewise constant level plus additive noise generated from an AR(1) model (top left) and online estimated volatilities by the 10%-trimmed standard deviation, the MAD and a τ estimate of scale in comparison to Q_n and the true scale.

We apply the moving MAD, trimmed standard deviation, τ estimator and Q_n to a time window of width $n = 250$, see Fig. 4. As a reference, a GARCH(1,1) model is fitted to the data using the R package `fGarch`. For a robust GARCH fit we first replace the five obvious outliers which can easily be spotted in Fig. 4 by the unconditional mean zero before estimating the model parameters and the time-varying conditional volatilities by maximum likelihood assuming t_ν-distributed errors with $\nu = 5$ degrees of freedom to optimize robustness within this approach (this number of degrees of freedom is the smallest one for which fourth moments exist, including the variance of the variance). Note that these estimates are not online since we use the whole data set for parameter estimation, but GARCH methods are a common standard for such data. The online estimates based on robust scale estimators look similar here, and they are much smoother than the conditional volatilities estimated by GARCH. Actually, the online estimates lead to similar results as a lowess smoother with bandwidth 1% (of the data points) applied to the conditional volatilities obtained using GARCH. However, the former work fully online.

3.3. Running time

We analyze the average time needed for an update of Q_n when using windows of width $50 \leq n \leq 5000$. For windows with width $n < 50$, the difference in running times is too small for accurate measurement, i.e. smaller than 0.1 ms, and there is little difference in using the offline or the online algorithm. In addition, we included the τ estimator of scale which is the best choice next to the Q_n estimator according to the simulations. Note that the running time for the τ estimator is $\mathcal{O}(n)$ for offline and online computations.

In order to get similar situations for all window widths, we generate new data sets for each n. For the GARCH model we simulate data sets of size $N = n + 2500$. For the time series with a piecewise constant σ_t and the AR(1) process with level shifts we set the length of each part to $n + 500$.

Fig. 5 illustrates a very good online running time for the GARCH(1,1) series, as could be expected from Theorem 1. In the case of the real data example and the data set with piecewise constant variances, we also observe a considerable improvement in running time. The offline computation time is nearly the same for all four data sets. Therefore we include only the overall average of the offline computation times for each window width for comparison.

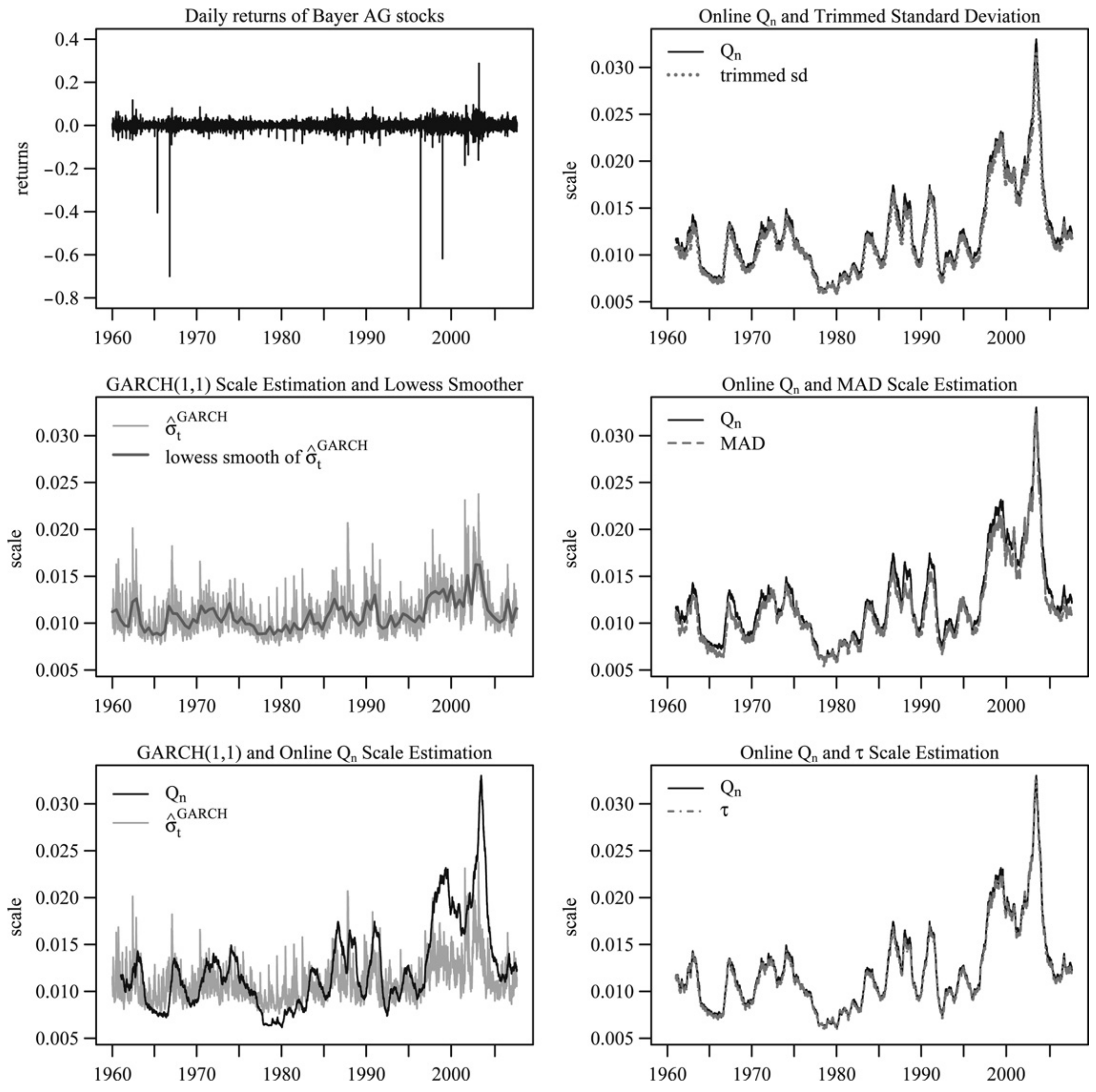

Fig. 4. Daily returns of Bayer AG stocks between January 4, 1960 and July 17, 2007 (top left) and estimated volatilities. The left-hand side panels show scale estimations based on a GARCH(1,1) model, the right-hand side panels show online scale estimations based on robust scale estimates applied to a moving time window of size $n = 250$.

4. Conclusions

The proposed new update algorithm for calculating the Q_n scale estimator, the medcouple or the Hodges–Lehmann location estimator in a moving time window shows good running time behavior in different data situations. The gain in computation time permits *real time application* of all these estimators which are robust and also quite efficient for Gaussian data. In particular, this offers the possibility of online scale estimation from stationary time series by a running Q_n estimator. This is especially useful for high-frequency data which are contaminated by artifacts. Irrespective of the existence of autocorrelations, the proposed Q_n update algorithm can directly be used in a nonparametric manner to estimate the marginal local volatility of a time series. As compared to simpler methods like (trimmed) standard deviations, Q_n leads to similarly good results in the case of Gaussian data, but it provides a much higher resistance against large number of outliers. Q_n yields less variable estimations in comparison to other highly robust scale estimators which are able to deal with a substantial amount of outliers without becoming strongly biased. Moreover, due to its definition via pairwise differences, which does not need an estimate of the local mean, Q_n is less biased than other scale estimators in the presence of outlier patches or shifts in the mean of the time series.

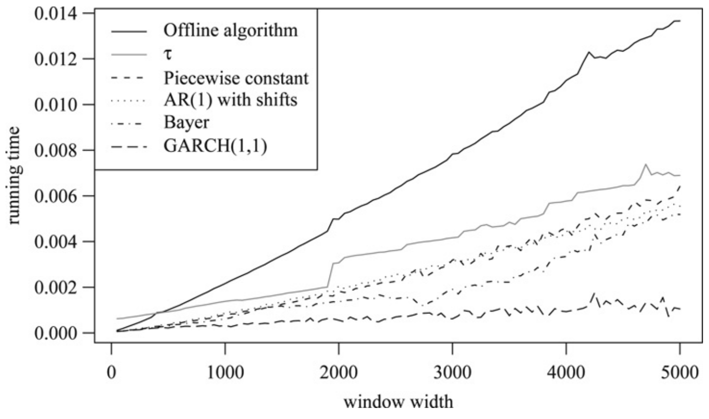

Fig. 5. Running time needed for the online analyses of the four considered data sets with Q_n in comparison to the Q_n offline algorithm and the τ estimator for window widths n between 50 and 5000.

Fried (2007) and Nunkesser et al. (in press) show that standardization by Q_n leads to more powerful and quite robust tests for the detection of abrupt level shifts than standardizing by other highly robust and less efficient scale estimators like the MAD. Besides, it is also possible to use Q_n for robust estimation of the autocorrelations and partial autocorrelations of a time series (Ma and Genton, 2000).

Acknowledgements

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, Reduction of complexity in multivariate data structures) and the helpful comments of two referees and an associate editor are gratefully acknowledged.

References

Bernholt, T., Fried, R., Gather, U., Wegener, I., 2006. Modified repeated median filters. Statistics and Computing 16 (2), 177–192.
Bespamyatnikh, S.N., 1998. An optimal algorithm for closest-pair maintenance. Discrete and Computational Geometry 19 (2), 175–195.
Bollerslev, T., 1986. Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics 31 (3), 307–327.
Brockwell, P.J., Davis, R.A., 1996. Introduction to Time Series and Forcasting. Springer, New York.
Brownlees, C.T., Gallo, G.M., 2006. Financial econometric analysis at ultra-high frequency: Data handling concerns. Computational Statistics & Data Analysis 51 (4), 2232–2245.
Brys, G., Hubert, M., Struyf, A., 2004. A robust measure of skewness. Journal of Computational & Graphical Statistics 13 (22), 996–1017.
Croux, C., Rousseeuw, P.J., 1992. Time-efficient algorithms for two highly robust estimators of scale. Computational Statistics 1, 411–428.
Fried, R., 2007. On robust shift detection in time series. Computational Statistics & Data Analysis 52, 1063–1074.
Gather, U., Fried, R., 2003. Robust scale estimation for local linear temporal trends. Tatra Mt. Math. Publ. 26, 87–101.
Gelper, S., Schettlinger, K., Croux, C., Gather, U., 2007, Robust online scale estimation in time series: A regression-free approach, Tech. Rep. 17/2007, SFB 475, University of Dortmund.
Johnson, D.B., Mizoguchi, T., 1978. Selecting the kth element in $x + y$ and $x_1 + x_2 + \cdots + x_m$. SIAM Journal on Computing 7 (2), 147–153.
Ma, Y., Genton, M.G., 2000. Highly robust estimation of the autocovariance function. Journal of Time Series Analysis 21 (6), 663–684.
Maronna, R.A., Zamar, R.H., 2002. Robust estimates of location and dispersion for high-dimensional datasets. Technometrics 44 (4), 307–317.
Mercurio, D., Spokoiny, V., 2004. Statistical inference for time-inhomogeneous volatility models. The Annals of Statistics 32 (2), 577–602.
Nunkesser, R., Schettlinger, K., Fried, R., 2007, Applying the Q_n-estimator online. In: Proceedings of the 31st Annual Symposium of the German Classification Society. (in press).
Rousseeuw, P.J., Croux, C., 1993. Alternatives to the median absolute deviation. Journal of the American Statistical Association 88 (424), 1273–1283.
Smid, M., 1991. Maintaining the minimal distance of a point set in less than linear time. Algorithms Review 2, 33–44.