Modified repeated median filters

T. BERNHOLT[a], R. FRIED[b,1], U. GATHER[c] and I. WEGENER[a]
[a] *Department of Computer Science, University of Dortmund, 44221 Dortmund, Germany*
Bernholt@ls2.cs.uni-dortmund.de
Wegener@ls2.cs.uni-dortmund.de
[b] *Department of Statistics, University Carlos III de Madrid, 28903 Getafe, Spain*
rfried@est-econ.uc3m.es
[c] *Department of Statistics, University of Dortmund, 44221 Dortmund, Germany*
gather@statistik.uni-dortmund.de

We discuss moving window techniques for fast extraction of a signal comprising monotonic trends and abrupt shifts from a noisy time series with irrelevant spikes. Running medians remove spikes and preserve shifts, but they deteriorate in trend periods. Modified trimmed mean filters use a robust scale estimate such as the median absolute deviation about the median (MAD) to select an adaptive amount of trimming. Application of robust regression, particularly of the repeated median, has been suggested for improving upon the median in trend periods. We combine these ideas and construct modified filters based on the repeated median offering better shift preservation. All these filters are compared w.r.t. fundamental analytical properties and in basic data situations. An algorithm for the update of the MAD running in time $O(\log n)$ for window width n is presented as well.

Keywords: signal extraction, robust filtering, drifts, jumps, outliers, computational geometry, update algorithm

1 Introduction

Signal extraction from high frequency data is a common technological task. To illustrate the arising challenges we consider a time series representing the heart rate of a patient in intensive care, see Fig. 1: We find time periods representing a steady state, monotonic trends, abrupt level shifts as well as many, often one-sided outliers caused e.g. by measurement problems. A filtering procedure for signal extraction from such data should

- track monotonic trends (also called drifts),
- preserve abrupt level shifts (steps or jumps),
- resist outliers (impulses or spikes),
- attenuate 'normal' observational noise,
- require short computation time.

[1]corresponding author

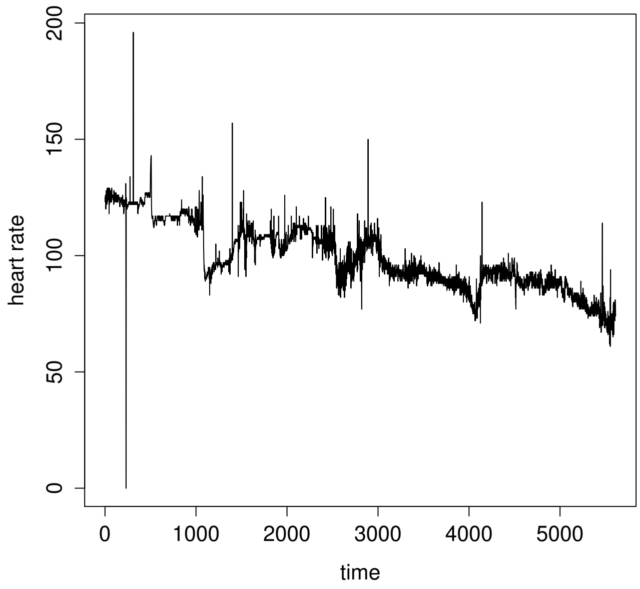

Figure 1: Heart rate of a patient in intensive care, sampling frequency 1 per minute

Moving averages and other linear filters track trends and attenuate Gaussian noise efficiently, but they are highly vulnerable to outliers and blur level shifts. Running medians advocated by Tukey (1977) remove outliers and preserve shifts in a piecewise constant signal, but they have shortcomings in trend periods. Lee and Kassam (1985) suggest modified trimmed means as a compromise between moving averages and running medians. The amount of trimming is chosen adaptively using a robust scale estimator like the median absolute deviation about the median (MAD). These filters are more efficient for Gaussian noise, they also remove spikes and may preserve steps even better than running medians (Himayat and Kassam 1993). Davies, Fried and Gather (2004) propose the repeated median, a robust regression estimator, to overcome the difficulties of the median during trends.

Replacing the median by the repeated median, we construct modified repeated median filters which trim observations with large regression residuals. The filter output is obtained thereafter using either least squares regression to get high efficiency or another repeated median for good robustness. Windows with different widths within these two steps allow to dampen ordinary noise considerably along with the preservation of signal details, particularly of abrupt shifts.

This paper is organized as follows: Section 2 introduces the filters mentioned above. As computation time is often crucial, Section 3 presents fast algorithms for the filters, especially an update algorithm for the MAD. Section 4 investigates analytic properties which are important to meet the demands specified above. Section 5 describes a simulation study which compares the methods in basic data situations. Section 6 presents applications to real and simulated data for further comparison. We finish of with some conclusions.

2 Robust filtering procedures

The task of signal extraction from an observed time series (x_t) can be formalized via a components model for the data

$$x_t = \mu_t + u_t + v_t,\ t \in \mathbb{Z}, \tag{1}$$

where u_t represents observational noise with median zero and variance σ_t^2, such that the signal (μ_t) is the time-varying level of the series, while v_t is impulsive (spiky) noise from an outlier generating mechanism. We assume (μ_t) to be smooth with a few abrupt shifts, and the variance σ_t^2 is allowed to vary smoothly in time as well.

For robust filtering we move a time window $(x_{t-k}, \ldots, x_{t+k})$ of width $n = 2k+1$ through the series and approximate the signal value μ_t in the center of the window. This allows signal extraction with a time delay of k observations. A small choice of k means a short delay and brief computation time, but as we will see it reduces smoothness and robustness.

2.1 Location based filters

The running median simply calculates the median of the observations in the window,

$$MED(x_t) = med(x_{t-k}, \ldots, x_{t+k}),\ t \in \mathbb{Z}.$$

The moving average uses the arithmetic mean instead of the median.

Filters based on trimmed means have been suggested as a compromise between moving averages and running medians (Lee and Kassam 1985, Pitas and Venetsanopoulos 1992). Modified trimmed mean (MTM-) filters choose the amount of trimming depending on the current time window. Observations which are further than a certain distance q_t away from the local median are trimmed and the average of the remaining observations is taken as filter output:

$$\begin{aligned} MTM(x_t) &= \frac{1}{|I_t|} \sum_{i \in I_t} x_{t+i}, \qquad (2) \\ I_t &= \{i = -k, \ldots, k : |x_{t+i} - \tilde{\mu}_t| \le q_t\} \\ \tilde{\mu}_t &= med(x_{t-k}, \ldots, x_{t+k}), \quad t \in \mathbb{Z}\,. \end{aligned}$$

For $q_t = 0$ we get a running median, while for $q_t = \infty$ we get a moving average. An adaptive choice of q_t is possible by using the median absolute deviation about the median (MAD) for calculating a robust estimate of the local scale,

$$\tilde{\sigma}_t^M = c_n \cdot med(|x_{t-k} - \tilde{\mu}_t|, \ldots, |x_{t+k} - \tilde{\mu}_t|).$$

Here, c_n is a correction factor depending on the window width n, which is usually chosen to achieve unbiasedness under Gaussian noise. For very large n we set $c_n = 1.483$, while e.g. for $n = 21$ we have $c_n = 1.625$. A reasonable choice of q_t is e.g. $q_t = 2\tilde{\sigma}_t^M$ (Lee and Kassam 1985, Himayat and Kassam 1993).

Double window modified trimmed mean (DWMTM-) filters are a version of MTM-filters using two windows with different widths. The median and the MAD are calculated from a short signal window with width $m = 2l + 1$, $l < k$, to retain signal details. Then all observations more deviant than q_t from this median are trimmed from the larger window with width $n = 2k + 1$, before the remaining values are averaged for better attenuation of observational noise:

$$\begin{aligned} DWMTM(x_t) &= \frac{1}{|I_t|} \sum_{i \in I_t} x_{t+i} \\ I_t &= \{i = -k, \ldots, k : |x_{t+i} - \tilde{\mu}_t| \leq q_t\} \\ \tilde{\mu}_t &= med(x_{t-l}, \ldots, x_{t+l}), \quad t \in \mathbf{Z}. \end{aligned} \tag{3}$$

2.2 Regression based filters

Application of a running median means to regard the level of the time series as almost constant within each window. This assumption is not appropriate in trend periods. Similarly, MTM-filters rely on a location model for trimming. Instead, Davies *et al.* (2004) suggest robust fitting of a linear trend $\mu_{t+i} = \mu_t + i\beta_t$, $i = -k, \ldots, k$, to the data within each window. Based on a comparison of regression estimators with high breakdown point they recommend Siegel's (1982) repeated median (RM)

$$\begin{aligned} RM(x_t) &= med(x_{t-k} + k\tilde{\beta}_t^{RM}, \ldots, x_{t+k} - k\tilde{\beta}_t^{RM}) \\ \tilde{\beta}_t^{RM} &= med_{i=-k,\ldots,k} \left(med_{j \neq i} \frac{x_{t+i} - x_{t+j}}{i - j} \right). \end{aligned}$$

In analogy to MTM-filters we can trim observations with large residuals in this regression setting and perform a second step. A suitable trimming constant q_t can be obtained by estimating the local variability about the repeated median regression line via the MAD of the regression residuals (Gather and Fried 2003). For the filter output, we can then either apply least squares (LS) regression or another repeated median to the observations with moderately large residuals. We call the resulting procedures TRM- and MRM-filters, respectively:

$$\begin{aligned} TRM(x_t) &= \overline{x}_t - \tilde{\beta}_t^{TRM} \cdot \bar{i}_t \\ \tilde{\beta}_t^{TRM} &= \frac{\displaystyle\sum_{i \in J_t} (i - \bar{i}_t)(x_{t+i} - \overline{x}_t)}{\displaystyle\sum_{i \in J_t} (i - \bar{i}_t)^2} \\ \overline{x}_t &= \frac{1}{|J_t|} \sum_{i \in J_t} x_{t+i}, \qquad \bar{i}_t = \frac{1}{|J_t|} \sum_{i \in J_t} i \\ J_t &= \{i = -k, \ldots, k : |x_{t+i} - \tilde{\mu}_t^{RM} - i\tilde{\beta}_t^{RM}| \leq q_t\} \end{aligned} \tag{4}$$

$$\begin{aligned} MRM(x_t) &= med_{i \in J_t}(x_{t+i} - i\tilde{\beta}_t^{MRM}) \\ \tilde{\beta}_t^{MRM} &= med_{i \in J_t} \left(med_{j \in J_t, j \neq i} \frac{x_{t+i} - x_{t+j}}{i - j} \right) \end{aligned} \tag{5}$$

with $(\tilde{\mu}_t^{RM}, \tilde{\beta}_t^{RM})$ being the repeated median level and slope calculated from $(x_{t-l}, \ldots, x_{t+l})$. If this inner window is shorter, $l < k$, we speak of DWTRM- and DWMRM-filters.

A computationally cheap variant of the double window idea is to estimate only the slope from a shorter window

$$\begin{aligned} DWRM(x_t) &= med(x_{t-k} + k\tilde{\beta}_t, \ldots, x_{t+k} - k\tilde{\beta}_t) \\ \tilde{\beta}_t &= med_{i=-l,\ldots,l}\left(med_{j=-l,\ldots,l,j\neq i}\frac{x_{t+i} - x_{t+j}}{i-j}\right). \end{aligned} \tag{6}$$

We expect the slope estimate to be less affected when a shift intrudes into the outer window then as it is obtained from observations which are not shifted yet. The intention is that the resulting filter preserves shifts about as well as the median, but is independent of a trend.

In the following discussions, we treat the RM, the TRM and the MRM as special cases of the DWRM, the DWTRM and the DWMRM, respectively, with $k = l$.

3 Update algorithms

Computation time is often a limiting constraint for the application of computer intensive methods. Signal extraction based on local estimations within a moving time window can be sped up considerably by applying fast update algorithms which exploit the overlap between subsequent windows instead of calculating the estimates every time from scratch: When moving the window we just need to remove the oldest observation from it and insert the incoming observation. This may be done quickly without loosing the temporal ordering when using suitable data structures. Bernholt and Fried (2003) propose an algorithm for updating the repeated median in linear time using quadratic space, which improves the computational complexity of a straightforward implementation substantially as the latter is of order n^2.

3.1 Updating the MAD

In the following we propose an algorithm for the local MAD

$$\tilde{\sigma}_t^M = \text{med}\left(|x_{t-k} - \tilde{\mu}_t|, \ldots, |x_{t+k} - \tilde{\mu}_t|\right), \quad t \in \mathbb{Z},$$

where $\tilde{\mu}_t = \text{med}(x_{t-k}, \ldots, x_{t+k})$, $t \in \mathbb{Z}$, is the local median of an odd number $n = 2k + 1$ of real values $x_{t-k}, \ldots, x_{t+k}$. To compute the MAD offline the following is known:

Theorem 1 *Given n values $x_{t-k}, \ldots, x_{t+k}$ with $x_i \in \mathbb{R}$, the* MAD *can be computed in time $O(n)$.*

Proof: The median can be computed in $O(n)$ time (see e.g. Cormen *et al.* 2001) and, therefore, the MAD can also be computed with two median operations in time $O(n)$. □

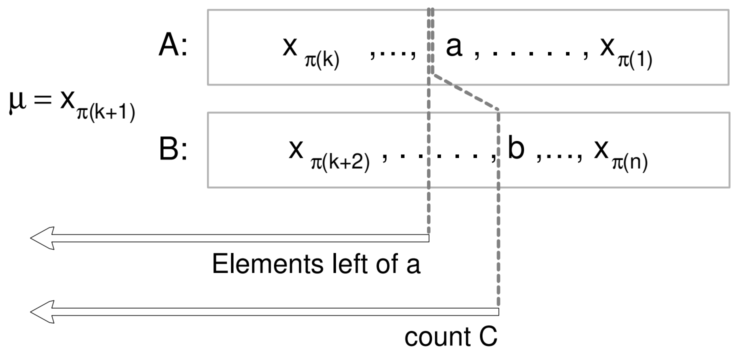

Figure 2: Let π be the permutation, such that the x_i's are sorted and $\mathrm{val}(a) < \mathrm{val}(b)$. The number of elements v with $\mathrm{val}(v) < \mathrm{val}(a)$ is smaller than the count C, which counts the elements with a smaller value than $\mathrm{val}(a)$ in the array A and $\mathrm{val}(b)$ in the array B.

As we assign to each x_{t+i} a value based on its deviation from the median, we will address the x_{t+i} as 'elements' in the following. We mostly drop the index t for ease of notation. To briefly explain the idea of the algorithm for updating the MAD, assume that the median $\tilde{\mu}$ is already computed and store the other elements in two arrays A and B. We will later replace the arrays by other data structures. A value x_i is stored in A if $x_i < \tilde{\mu}$, and in B otherwise. Assume the elements in A and B are sorted, as displayed in Fig. 2. Let $\mathrm{val}(x_i) = (|x_i - \tilde{\mu}|, i)$ and define an ordering relation by $\mathrm{val}(x_i) < \mathrm{val}(x_j)$ if $|x_i - \tilde{\mu}| < |x_j - \tilde{\mu}|$, or in the case $x_i = x_j$ if $i < j$. To compute the MAD, apply the function $\mathrm{val}(x_i)$ to each element and consider the merged sequence M of A, B and $\tilde{\mu}$. The val-function determines the ordering of two elements in M. The unknown element to be found by the algorithm, for which the MAD takes its value, is denoted by MAD henceforth for the reason of memorability. By definition, this MAD-element has the property that there are k elements v with $\mathrm{val}(v) < \mathrm{val}(\mathrm{MAD})$ and k elements v' with $\mathrm{val}(v') > \mathrm{val}(\mathrm{MAD})$.

Instead of merging the two sequences we proceed in as follows: Suppose there are two elements $a \in A$ and $b \in B$ given such that w.l.o.g. $\mathrm{val}(a) < \mathrm{val}(b)$. The choice of a and b will become clear later on. We now count the elements in $A \cup \{\tilde{\mu}\}$ with a smaller value than $\mathrm{val}(a)$ and the elements in B with a smaller value than $\mathrm{val}(b)$. Call this total count C. The first case is $C < k$. Then we know that there are less than k elements left of a in M and, therefore, a and all elements left of a in A cannot be the MAD and we can exclude them from the further search (Lemma 1). The second case is $C \geq k$. Then there are $n - 2 - C < k$ values right of a or b and, therefore, we can exclude b and all elements right of b in B from the further search (Lemma 2). Now we perform a binary search on the remaining parts of A and B until the MAD is found.

This idea describes our method in the static case of a single time point t. We are more interested in the dynamic case. Thus, we need a data structure such that we can perform the following operations efficiently: exclude x_{t-k}, include x_{t+k+1}, increase t by 1, and compute the new MAD.

This affords storing the data in a dynamic data structure, guaranteeing a run time of $O(\log n)$ for the following operations: determine the largest and the smallest element in the tree, insert and delete an element. Therefore, instead of the arrays A and B we use binary search trees like AVL trees or red-black trees for storing the data (Cormen *et al.* 2001). Furthermore, these trees allow to perform the following operations in time $O(1)$:

1. find the root of the tree ("rootof") and
2. find the left/right child of a given element ("leftchildof"/"rightchildof").

If a vertex does not exist the routines return a nil-pointer.

We use two binary search trees $T_<$ and $T_>$. To deal with the degenerated case that two elements x_i and x_j are equal, we store the pair (x_i, i) in the tree. To simplify the description we skip the index in the notation and refer to (x_i, i) by x_i. An element x_i is stored left of an element x_j in the tree $T_>$, if $x_i < x_j$ or in the case $x_i = x_j$ if $i < j$. The elements in the tree $T_<$ are stored in the reverse order. An element x_i is stored in $T_<$ if x_i is smaller than the current median μ, otherwise it is stored in $T_>$. This MAD algorithm works only for an odd number of elements $n = 2k+1$. In each tree k elements are stored, while μ is not contained in any tree.

In the preprocessing phase we just insert elements into the trees. If the number of inserted elements is odd and the numbers of elements in the two trees are no longer equal, w.l.o.g. $T_<$ has less elements than $T_>$, their sizes are balanced by inserting the old $\tilde{\mu}$ into $T_<$ and finding the leftmost element v in $T_>$, deleting it from $T_>$ and using v as the new median $\tilde{\mu}$. In the update step we insert one element and delete one element and also perform the balancing of the sizes of the trees, if necessary.

Consider the sorted order of the elements of one of both trees. Then, $\text{rank}(x_i)$ is the position of the element x_i in this sorted order, the leftmost element has rank 1. For two elements x_i and x_j stored in the same tree, we have that $\text{rank}(x_i) < \text{rank}(x_j)$ iff $\text{val}(x_i) < \text{val}(x_j)$ iff x_i is stored left of x_j. If two elements x_i and x_j are stored in different trees, we also use the terms *left of* and *right of*. The element x_i is left of x_j iff $\text{val}(x_i) < \text{val}(x_j)$.

In each vertex x_i of the trees we additionally store the number $\text{sizeof}(x_i)$, which counts the number of elements stored in the subtree of x_i. During insertion, deletion and rotations this information can be updated with asymptotically negligible costs. To avoid some case distinctions define $\text{sizeof}(\text{nil}) = 0$.

Procedure 1 returns the root v of a tree and computes the rank of v by counting the number of the elements in the left subtree and adding 1 for v itself.

Knowing $\text{rank}(v)$ we design procedures to compute the rank of a child v' of v. For this, we have to subtract or add the number of elements between v and v'. In the case of a branch to the left, these are all elements in the right subtree of v', in the case of a branch to the right, these are all elements in the left subtree of v'. This is described in detail in the Procedures 2 and 3.

An element $a \neq \tilde{\mu}$ is stored either in $T_<$ or $T_>$. Let $T_a \in \{T_<, T_>\}$ be the tree for which $a \in T_a$. Define

$$\text{leftof}(a) = \{v \mid v \in T_a \text{ and } \text{val}(v) < \text{val}(a)\}$$

Procedure 1 GetRootOf

Input:

A tree T.

Output:

The procedure computes the root v of the tree T and the rank of v.

begin

$v \leftarrow$ rootof(T)

rank_v $\leftarrow$ 1+sizeof(leftchildof(v))

return (v, *rank_v*)

end

Procedure 2 BranchLeft

Input:

A vertex v and *rank_v*.

Output:

The procedure computes the left child of v and its rank.

begin

$v' \leftarrow$ leftchildof(v)

if $v' \neq$ nil **then**

rank_v' $\leftarrow$ *rank_v* $-$ 1$-$ sizeof(rightchildof(v'))

return (v', *rank_v'*)

end

Procedure 3 BranchRight

Input:

A vertex v and *rank_v*

Output:

The procedure computes the right child of v and its rank.

begin

$v' \leftarrow$ rightchildof(v)

if $v' \neq$ nil **then**

rank_v' $\leftarrow$ *rank_v* $+$ 1$+$ sizeof(leftchildof(v'))

return (v', *rank_v'*)

end

and

$$\text{rightof}(a) = \{v \mid v \in T_a \text{ and } \text{val}(v) > \text{val}(a)\}.$$

Procedure 4 ComputeMAD

Input:

1: AVL trees $T_<$ and $T_>$ with k vertices each and the median $\tilde{\mu}$.

Output:

2: The MAD-value.

begin

3: $(a, \mathit{rank\_a}) \leftarrow \text{GetRootOf}(T_<)$; $(b, \mathit{rank\_b}) \leftarrow \text{GetRootOf}(T_>)$
4: **while** $a \neq$ nil and $b \neq$ nil **do**
5:     **if** $\text{val}(a) > \text{val}(b)$ **then**
6:         swap $(a, \mathit{rank\_a})$ with $(b, \mathit{rank\_b})$
7:     **if** $\mathit{rank\_a} + \mathit{rank\_b} \leq k$ **then**
8:         $(w, \mathit{rank\_w}) \leftarrow (a, \mathit{rank\_a})$
9:         $(a, \mathit{rank\_a}) \leftarrow \text{BranchRight}(a, \mathit{rank\_a})$
10:     **else**
11:         $(w, \mathit{rank\_w}) \leftarrow (b, \mathit{rank\_b})$
12:         $(b, \mathit{rank\_b}) \leftarrow \text{BranchLeft}(b, \mathit{rank\_b})$
13: **end while**
14: **if** $a =$ nil **then**
15:     $\mathit{rank\_MAD} \leftarrow k - \mathit{rank\_w}$
16:     MAD $\leftarrow \text{SearchRank}(b, \mathit{rank\_b}, \mathit{rank\_MAD})$
17: **else**
18:     $\mathit{rank\_MAD} \leftarrow k + 1 - \mathit{rank\_w}$
19:     MAD $\leftarrow \text{SearchRank}(a, \mathit{rank\_a}, \mathit{rank\_MAD})$
20: return MAD

end

After inserting and deleting some elements and balancing the sizes of the trees, we can execute the Procedure ComputeMAD. The algorithm starts with two pointers a and b at the roots of both trees and in the Lines 5 and 6 it is ensured that $\text{val}(a) < \text{val}(b)$. In Line 7 it is determined whether either $\text{val}(a) < \text{val}(\text{MAD})$ (Lemma 1) or $\text{val}(b) > \text{val}(\text{MAD})$ (Lemma 2). It is essential that this can be checked without knowing the MAD-element. If the algorithm branches to the right at a, all elements in the left subtree of a are excluded from further considerations. Branching at b excludes all elements in the right subtree of b. The algorithm branches in the appropriate direction such that the MAD-element and the element w, which is defined in a moment, are always contained in the subtrees of a or b (Lemma 3).

Assume that MAD $\in T_<$. When the 'while loop' stops, the algorithm has found an element $w \in T_>$ with the property that either w is the rightmost element in $T_>$ with $\text{val}(w) < \text{val}(MAD)$ (Lines 15,16) or w is the leftmost element in $T_>$ with $\text{val}(w) > \text{val}(MAD)$ (Lines 18,19). Using the rank of w the algorithm is able to determine the rank of the MAD-element. With this information it is easy to find the MAD-element using

Procedure 5. These facts and the correctness of the algorithm are proven in Theorem 2.

Procedure 5 SearchRank

Input:

Vertex p, $rank\_p$ and a rank number r.

Output:

The vertex v with rank r if v is contained in the subtree of p.

begin

  while $p \neq$ nil and $rank\_p \neq r$ **do**

    if $rank\_p < r$ **then**

      $(p, rank\_p) \leftarrow$ BranchRight$(p, rank_p)$

    else

      $(p, rank\_p) \leftarrow$ BranchLeft$(p, rank_p)$

  end while

  return p

end

Lemma 1 *If the algorithm branches at vertex a,* $\mathrm{val}(a) < \mathrm{val}(\mathrm{MAD})$ *and the algorithm continues at the right child of a.*

Proof: If the algorithm branches at a we know from Line 9 that the algorithm continues at the right child of a and we know from Line 7 that $\mathrm{rank}(a) + \mathrm{rank}(b) \leq k$. Moreover,

$$
\begin{array}{rlclcll}
 & \mathrm{rank}(a) & + & \mathrm{rank}(b) & & \leq & k \\
\Leftrightarrow & \mathrm{rank}(a) - 1 & + & \mathrm{rank}(b) - 1 & +1 & \leq & k - 1 \\
\Leftrightarrow & |\,\mathrm{leftof}(a)| & + & |\,\mathrm{leftof}(b)| & +|\{\tilde{\mu}\}| & \leq & k - 1.
\end{array}
$$

From the Lines 5 and 6 we know that $\mathrm{val}(a) < \mathrm{val}(b)$ and, therefore, there are less than k elements with a smaller value than $\mathrm{val}(a)$. Therefore, $\mathrm{val}(a) < \mathrm{val}(\mathrm{MAD})$. □

Lemma 2 *If the algorithm branches at the vertex b,* $\mathrm{val}(b) > \mathrm{val}(\mathrm{MAD})$*, and the algorithm continues at the left child of b.*

Proof: If the algorithm branches at b we know from Line 12 that the algorithm continues at the left child of b and we know from Line 7 that $\mathrm{rank}(a) + \mathrm{rank}(b) > k$. Moreover,

$$
\begin{array}{rlclcl}
 & \mathrm{rank}(a) & + & \mathrm{rank}(b) & > & k \\
\Leftrightarrow & k - \mathrm{rank}(a) & + & k - \mathrm{rank}(b) & < & 2k - k \\
\Leftrightarrow & |\,\mathrm{rightof}(a)| & + & |\,\mathrm{rightof}(b)| & < & k.
\end{array}
$$

Note that $|\,\mathrm{rightof}(a)| = k - \mathrm{rank}(a)$, since, by definition, each of the trees $T_<$ and $T_>$ contains exactly k elements. From the Lines 5 and 6 we know that $\mathrm{val}(a) < \mathrm{val}(b)$ and, therefore, there are less than k elements with a greater value than $\mathrm{val}(b)$. Therefore, $\mathrm{val}(b) > \mathrm{val}(\mathrm{MAD})$. □

Lemma 3 *The* MAD*-element is not excluded by the algorithm.*

Proof: There are k elements left of the MAD-element. The algorithm ensures in the Lines 5 and 6 that $\mathrm{val}(a) < \mathrm{val}(b)$.

- **Case 1:** $a = \mathrm{MAD}$.
  From $\mathrm{val}(a) < \mathrm{val}(b)$ it follows that MAD is left of b and, therefore, there are at least $k+1$ elements left of b, namely the elements of the sets leftof(MAD), leftof(b) and $\{\mathrm{MAD}, \tilde{\mu}\}$. Moreover,

$$\begin{array}{cccccccc} & |\,\mathrm{leftof(MAD)}| & + & |\,\mathrm{leftof}(b)| & +|\{\mathrm{MAD}, \tilde{\mu}\}| & \geq & k+1 \\ \Leftrightarrow & \mathrm{rank(MAD)} - 1 & + & \mathrm{rank}(b) - 1 & +2 & \geq & k+1 \\ \Leftrightarrow & \mathrm{rank(MAD)} & + & \mathrm{rank}(b) & & > & k. \end{array}$$

  Therefore, the Lines 8 and 9 are not executed and the MAD-element is not excluded.

- **Case 2:** $b = \mathrm{MAD}$.
  From $\mathrm{val}(a) < \mathrm{val}(b)$ it follows that MAD is right of a and, therefore, there are at least $k+1$ elements right of a. Moreover,

$$\begin{array}{cccccccc} & |\,\mathrm{rightof}(a)| & + & |\,\mathrm{rightof(MAD)}| & +|\{\mathrm{MAD}\}| & \geq & k+1 \\ \Leftrightarrow & k - \mathrm{rank}(a) & + & k - \mathrm{rank(MAD)} & +1 & \geq & k+1 \\ \Leftrightarrow & -\,\mathrm{rank}(a) & - & \mathrm{rank(MAD)} & & \geq & -k \\ \Leftrightarrow & \mathrm{rank}(a) & + & \mathrm{rank(MAD)} & & \leq & k. \end{array}$$

  Therefore, the Lines 11 and 12 are not executed and the MAD-element is not excluded.

- **Case 3:** $a \neq \mathrm{MAD}$ and $b \neq \mathrm{MAD}$.
  If the algorithm branches at element a, then Lemma 1 ensures that the MAD-element is not excluded. Lemma 2 ensures the same for branching at element b.

□

Theorem 2 *For n values $x_{t-k}, \ldots, x_{t+k}$ with $x_i \in \mathbb{R}$ and n odd, the* MAD *can be maintained in time $O(\log n)$ per update with $O(n \log n)$ preprocessing time.*

Proof: To prove the correctness of the update-algorithm, we show that the rank of the MAD is computed correctly. In Line 7, the algorithm branches either at a or b. Let w be the last vertex the algorithm branched at, before the while loop stopped. W.l.o.g. assume $\mathrm{MAD} \in T_<$. From Lemma 3 we know that the MAD is not excluded. Therefore, $w \in T_>$. Let

$$S_< = \{v \mid v \in T_< \text{ and } v \text{ is left of MAD}\} \tag{7}$$
$$S_> = \{v \mid v \in T_> \text{ and } v \text{ is left of MAD}\}. \tag{8}$$

Then the set of the k elements left of the MAD is split into the sets $S_<$, $S_>$ and $\{\tilde{\mu}\}$ implying that

$$|S_<| + |S_>| + |\{\tilde{\mu}\}| = k.$$

- **Case 1:** $w = a$.
  From Lemma 1 we know that val(w) < val(MAD) and that the algorithm continues with the right child of w. Therefore, w is left of the MAD-element and as the while loops stops, w does not have a right child. Hence w is the rightmost element in $T_<$ left of the MAD. Therefore, $S_> = \text{leftof}(w) \cup \{w\}$:

$$\begin{array}{cccccccc} & |S_<| & + & |S_>| & + & |\{\tilde{\mu}\}| & = k \\ \Leftrightarrow & |\,\text{leftof}(\text{MAD})| & + & |\,\text{leftof}(w) \cup \{w\}| & + & |\{\tilde{\mu}\}| & = k \\ \Leftrightarrow & \text{rank}(\text{MAD}) - 1 & + & \text{rank}(w) & + & 1 & = k \\ \Leftrightarrow & & & & & \text{rank}(\text{MAD}) & = k - \text{rank}(w). \end{array}$$

- **Case 2:** $w = b$.
  From Lemma 2 we know that val(w) > val(MAD) and that the algorithm continues with the left child of w. Therefore, w is right of the MAD-element and as the while loops stops, w does not has a left child. Hence w is the leftmost element in $T_<$ right of the MAD. Therefore, $S_> = \text{leftof}(w)$:

$$\begin{array}{cccccccc} & |S_<| & + & |S_>| & + & |\{\tilde{\mu}\}| & = k \\ \Leftrightarrow & |\,\text{leftof}(\text{MAD})| & + & |\,\text{leftof}(w)| & + & |\{\tilde{\mu}\}| & = k \\ \Leftrightarrow & \text{rank}(\text{MAD}) - 1 & + & \text{rank}(w) - 1 & + & 1 & = k \\ \Leftrightarrow & & & & & \text{rank}(\text{MAD}) & = k + 1 - \text{rank}(w). \end{array}$$

The two cases show that the rank is computed correctly in the Lines 15 and 18. Lemma 3 ensures that subtree(a) resp. subtree(b) contains the MAD and, therefore, the MAD is found by Procedure 5.

The algorithm searches along two paths. As AVL or red-black trees guarantee a maximal path length of $O(\log n)$, and each iteration of the while loop takes time $O(1)$, the run time is bounded by $O(\log n)$. The preprocessing needs n insertions into the trees and less than n balancings of the sizes of the trees. Therefore, the preprocessing can be performed in time $O(n \log n)$. $\square$

3.2 Updating the (DW)MTM filter

After using the routines from Section 3.1 to obtain the bound $q_t = 2\tilde{\sigma}_t^M$, we have to compute the sum of the x_i's with val(x_i) < q_t to obtain the estimate of the (DW)MTM-filter. As described before, we store the data in two AVL or red-black trees $T_<$ and $T_>$ and keep the sizes of the trees balanced with respect to the median μ. In each vertex v the sum $S(v)$ of the x_i contained in the subtree of v are stored. This information can be updated during insertion, deletion and rotation with asymptotic no additional costs. In the following, we will only describe the procedure for the tree $T_<$. It is the same for $T_>$. Performing a search for q_t in the tree results in a path P. It is easy to see that an element x_i with val(x_i) < q_t is either contained in a vertex $v \in P$ or is contained in a subtree whose root is a left child of $v' \in P$ with val(v') < q_t. Therefore, we inspect each element x_i on the path. For all x_i with val(x_i) < q_t we compute the sum of x_i and the value S(leftchildof(x_i)) stored in the root of the subtree mentioned above.

Performing this procedure for both trees gives the estimates of the (DW)MTM-filters. Since the path length in the trees is bounded by $O(\log n)$, each step can be computed in time $O(1)$, and the trees need no more than linear space, the bounds mentioned in Table 1 are achieved.

3.3 Updating modified repeated median filters

In the following we describe how to obtain the set J_t from Section 2.2 in linear time. We use the same data structure as Bernholt and Fried (2003) to update the RM- and DWRM-filter in time $O(n)$ and space $O(n^2)$. From the resulting estimates we get the n regression residuals. Then we can compute the MAD of the residuals in linear time (Theorem 1). Additionally, we can obtain the set J_t in the same time. To finally compute the (DW)TRM-estimate we have to evaluate some sums. All steps can be performed in $O(n)$. Therefore, the (DW)TRM-filter can be updated in time $O(n)$ and space $O(n^2)$ as mentioned in Table 1.

For the (DW)MRM it is advisable to apply a fast offline algorithm for the repeated median with an expected run time $O(n \log n)$ as described by Matoušek, Mount and Netanyahu (1998).

Table 1: Run time and space needed to update the filters when one element in the window is replaced by a new one.

	MED	(DW)MTM	(DW)TRM	(DW)RM	(DW)MRM
Time	$O(\log n)$	$O(\log n)$	$O(n)$	$O(n)$	$O(n \log n)$
Space	$O(n)$	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n)$

4 Theoretical analysis

Now we analyse the filters presented in Section 2 within individual time windows. For the ease of notation we drop the time index t and center the window at zero, i.e. we approximate μ_0 using $x_{-k}, \ldots, x_0, \ldots, x_k$.

4.1 Equivariances and invariances

Equivariances and invariances guarantee sensible reactions of an estimate to systematic changes in the data. Location equivariance means that a shift due to a constant added to all observations changes the estimate accordingly. Scale equivariance means that multiplying all observations by a constant changes the extracted signal in the same way. All filters described in Section 2 possess these properties. We note that the scale equivariance of the trimming estimators is due to the scale equivariance of the MAD.

A filter should moreover not be affected by trends. As we approximate the signal in the center of the window, the estimate should not depend on a constant linear trend if the central level is fixed. More precisely, Fried, Bernholt and Gather (2004) call a filter trend invariant if the filter outcomes for $x_{-k} - k\beta, \ldots, x_{-1} - \beta, x_0, x_1 + \beta, \ldots, x_k + k\beta$ and $x_{-k}, \ldots, x_k$ are the same. The regression based filters presented in Section 2.2 (DWRM, DWTRM, DWMRM) are trend invariant, while the location based filters in Section 2.1 are not. The trend invariance of the RM follows directly from the regression equivariance of the repeated median. The trend invariance of the DWTRM and the DWMRM can be derived easily from the proof of Lemma 1 in Fried (2004). The lack of trend invariance of trimmed mean filters with a percentage of trimming larger than zero is easy to see for the running median with $k = 1$: The median of $-1, 0, 3$ is zero, while for $\beta = -2$ we get the data $1, 0, 1$ with median 1. Similar examples can be constructed for every choice of k and for every positive amount of trimming.

4.2 Patterns in small observational noise

The removal of impulsive noise (outliers) and the preservation of shifts are essential properties of robust filters. We inspect the best possible performance of a procedure w.r.t. this when there is no observational noise, i.e. $u_t \equiv 0$. This analysis is analogous to the exact fit investigated in linear regression, see Rousseeuw and Leroy (1987, Section 3.4).

A running median preserves a level shift in an otherwise constant signal exactly and it removes up to k subsequent spikes completely in this idealized situation for $n = 2k + 1$. However, it preserves a shift during a trend only if the shift and the trend go in the same direction. The shift gets blurred otherwise, and a single spike within a trend may cause smearing. Similarly, a DWMTM-filter can remove up to l subsequent large spikes from a constant signal, where $m = 2l + 1$ is the window width used for calculation of the MAD. This explains why m should be chosen depending on the minimal length of relevant signal details. However, this property gets lost in trend periods, and a shift may not be preserved exactly during a trend.

The DWRM, the DWTRM and the DWMRM can remove $l - 1$ spikes within a single window completely as the median slope for each of the $l + 2$ clean data points out of the total $2l + 1$ within the inner window is calculated from $l + 1$ clean and $l - 1$ contaminated pairs of observations. For the DWTRM and the DWMRM we note that the MAD of the residuals is zero then and all spikes are trimmed before the second step. A shift causes smearing at least at three time points. These numbers are slightly worse than those for the location based methods in case of a constant signal, but they do not depend on a trend at all.

The previous results hold for the idealized case with $u_t \equiv 0$. Lipschitz continuity restricts the influence of minor changes in the data due to small observational noise or rounding. The running median and the DWRM are Lipschitz continuous: The median is Lipschitz continuous with constant 1 as changing every observation by less than δ changes any order statistic at most by δ. The repeated median slope changes at most by 2δ then, and we get that the DWRM is Lipschitz continuous with constant $2k + 1$ since none of the trend corrected observations can change more. Trimming filters such as the DWMTM, the DWTRM and the DWMRM are not Lipschitz continuous, i.e. small changes in the data can possibly have large effects.

4.3 Robustness against strong spiky noise

The previous analysis addressed the exact extraction of the signal value under (close to) optimal conditions with no or little observational noise. Alternatively, we can inspect worst-case conditions which can render the extracted value meaningless. As a starting point we assume that there are some data generated from the components model without spiky noise, i.e. $v_t \equiv 0$. Such 'clean' data will typically provide useful information on the underlying signal. How many spikes can now maximally be added to the clean data until the estimate gets arbitrarily far away from the true signal value? To answer this question we measure the minimal percentage of outliers within a window which can cause an arbitrarily large spike in the extracted signal. This corresponds to the breakdown of the estimators applied locally to the time window: The finite sample replacement breakdown point measures the minimal fraction of data being set to arbitrary values which can drive an estimate to infinity. The fractions derived in the following resemble the numbers of spikes obtained in the previous section as there is a relationship between exact fit and breakdown (Rousseeuw and Leroy 1987, pp. 122-124).

The breakdown point is $(k+1)/n$ for the median of $n = 2k+1$ data points, telling us that at least halve of the sample needs to be replaced to completely destroy the local level approximation.

To cause an arbitrary spike in the MTM-output the same number of replacements is needed as for the median since for explosion of the local MAD also at least $k+1$ observations need to be modified: If not more than k observations are replaced, the median is at most $||x||$ in absolute value, with $||x||$ denoting the maximum norm of the window sample vector $x = (x_{-k}, \ldots, x_k)'$. The MAD does not exceed $2c_n||x||$ since $|x_i - \tilde{\mu}| \leq 2||x||$ for each of the at least $k+1$ clean observations in the contaminated sample. Therefore, the trimmed mean is taken using only observations which are less than $|x_i| \leq |\tilde{\mu}| + |x_i - \tilde{\mu}| \leq (4c_n+1)||x||$ in absolute value. Analogously, destroying the DWMTM affords replacement of at least $l+1$ out of the $2l+1$ observations in the inner window, the same number as for the inner median. We note that for the double window filters the worst case positions of outliers are within the shorter window.

For causing a spike of any size in the DWRM at least l observations in the inner window must be modified: Breakdown of the repeated median affords this number when being calculated from $2l+1$ observations, hence the DWRM resists $l-1$ modifications. Setting the observations at times $1, \ldots, l$ in the inner window to $(l+2)M, (l+3)M, \ldots, (2l+2)M$, with $M > 8l \cdot ||x||$ being an arbitrarily large number, the median of the pairwise slopes for the other $l+1$ observations at times $-l, \ldots, 0$ in the calculation of $\tilde{\beta}^{DWRM}$ and hence $\tilde{\beta}^{DWRM}$ itself lies between $M/4$ and $(2l+3)M/(2l) \leq 5M/2$. Thus, the trend corrected observations at times $-k, \ldots, -1$ increase at least by $M/4$ each, while that at time 1 is at least $(l-1/2)M$. The DWRM for the modified sample will be at least $M/8$, which goes to infinity for increasing M.

The DWTRM and the DWMRM also resist arbitrary modification of $l-1$ observations in the inner window: When replacing less than l observations, the initial slope and level estimate are bounded in absolute value by $2||x||$ and $(2l+1)||x||$, respectively, and the MAD is bounded from above by $(4l+2)c_m||x||$. For the observations which are not trimmed we derive the finite bound $|x_i| \leq 2(4l+2)c_m||x|| + (2l+1)||x|| + 2k||x||$ from this, and hence the filter outcome is also bounded as it lies within the convex hull of these observations.

5 Monte Carlo experiments

For a comparison of the procedures we performed a Monte Carlo analysis. We chose basic data situations in accordance to the demands stated in the introduction. We used the components model (1) with an underlying linear trend, which is overlaid by possibly autocorrelated observational noise generated from an autoregressive model of order one, AR(1),

$$\begin{aligned} X_t &= \mu_t + u_t + v_t,\ t = -k, \ldots, k, \qquad (9) \\ \mu_t &= \beta t \\ u_t &= \phi u_{t-1} + \epsilon_t \end{aligned}$$

with the innovations ϵ_t forming Gaussian white noise with zero mean and unit variance. AR(1) models are a convenient choice for autocorrelated data.

The suitable choice of the window width $n = 2k+1$ depends on the application, i.e. on the situations a filtering procedure needs to handle. In intensive care e.g., a medical rule of thumb states that about five subsequent aberrant observations of about the same size are often clinically relevant, while shorter patterns are typically irrelevant (Imhoff *et al.* 2002). In order to preserve a shift we would like a filter to remain stable at the former level until halve of the observations in the window are shifted, if possible. In any case a filter should not be significantly affected by five shifted observations since then we can apply rules for shift detection (Fried 2004). These demands imply that k must be at least five or larger, depending on the robustness of the filter. Upper limits for k are set by the length of time periods in which trends can be assumed to be approximately linear and by the time delay admissible.

We mostly considered windows of width $n = 2k+1 = 21$, setting $m = 11$ for the DWMTM since the inner median resists five subsequent outliers then, while for the DWRM, the DWTRM and the DWMRM we used a larger $m = 15$ as outliers have a stronger impact on the repeated median than on the median in a steady state. This means a small difference between inner and outer window. Furthermore, we mostly set q_t to twice the MAD.

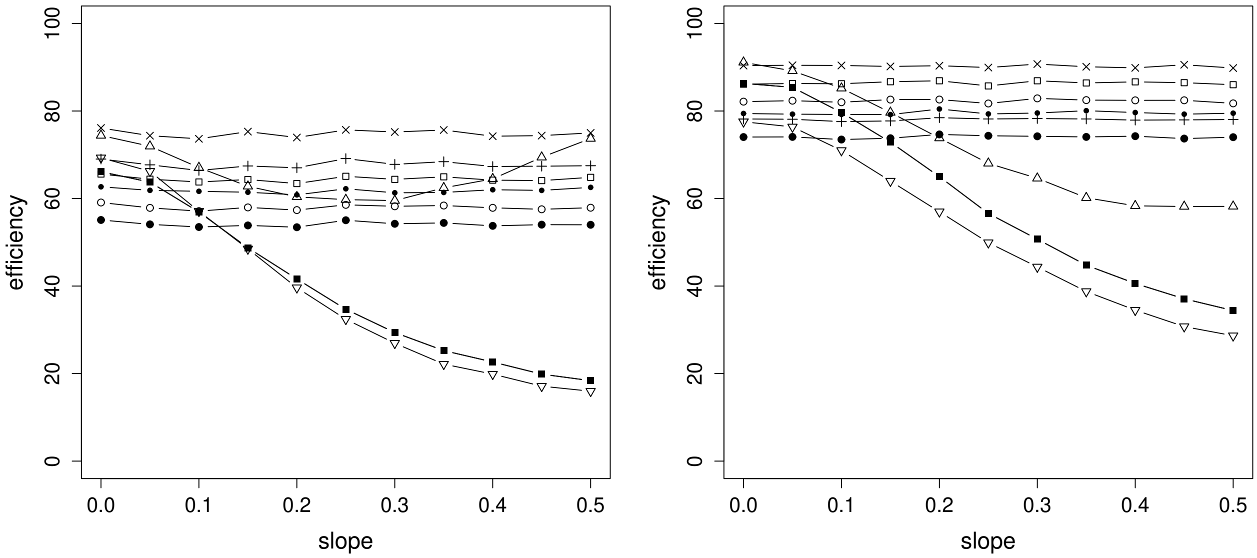

Figure 3: Relative efficiencies for Gaussian noise and autocorrelations $\phi = 0.0$ (left) or $\phi = 0.6$ (right): median (■), RM (□), MTM (△), DWMTM (▽), MRM (o), TRM ($\times$), DWMRM (●), DWTRM (+) and DWRM ($\cdot$)

5.1 Efficiency for Gaussian noise

We firstly compared the attenuation of Gaussian noise. All the methods considered are unbiased then because of symmetry. We simulated 20000 non-overlapping windows for each combination of slope $\beta \in \{0, 0.05, \ldots, 0.5\}$ and autocorrelation $\phi \in \{0, 0.6\}$ and calculated the efficiencies as measured by the percentage MSE relatively to the moving average from the results, see Fig. 3.

The MTM is rather efficient in case of a constant signal and zero correlations, but for positive correlations it looses efficiency when a trend occurs, similarly to the median. Using a shorter inner window generally reduces efficiency. The repeated median is almost as efficient as the median for a constant signal, and the same applies for the TRM and the MTM, as well as for the DWTRM and the DWMTM. However, the regression based filters are not affected by a trend due to their trend invariance. Positive correlations, which occur in many applications, increase the relative efficiencies of the robust regression methods. The second step in the TRM gains some efficiency, while that in the MRM reduces it somewhat. The DWTRM is even more efficient than the RM for zero correlations, but it is less efficient than the DWRM and the MRM in case of positive correlations $\phi = 0.6$.

We note that the efficiency of the TRM (DWTRM) can be further improved by setting q_t to a larger multiple of the MAD. Increasing q_t from two to three times the MAD e.g. yields efficiencies of 92% (86%) for $\phi = 0$ and 97% (86%) for $\phi = 0.6$ instead of the 76% (69%) and 90% (78%) found before. The same modification of the MRM and DWMRM increases the efficiency only by about 4% in each case.

5.2 Preservation of shifts

Level shifts should be localized and tracked as exactly as possible. Outlying observations of similar size at the end of the window, which may be due to a shift, should not influence the output to avoid smearing. In order to investigate the preservation of a shift within a trend we generated signals from model (9) which mimic the intrusion of a shift into the window: We replaced an increasing number $1, 2, \ldots, 10$ of observations at the end of the window by positive additive outliers of size $s \in \{1, 2, \ldots, 10\}$. For each setting we simulated 2000 windows to calculate the bias, the standard deviation and the root of the mean square error RMSE.

Fig. 4 depicts the maximal (w.r.t. the outlier size) RMSE as a function of the number of outliers for a constant signal ($\beta = 0$) and a steep upward or downward trend ($\beta = \pm 0.5$). The closer a RMSE-curve stays to zero, the better shifts are preserved. From $\beta = 0$ the excellent preservation of shifts by the median in case of a constant signal becomes visible. The MTM, and even more the DWMTM resist a shift very well up to eight (about 40%) affected observations. The DWMRM and the DWTRM, which both apply a shorter inner window, also perform well up to six (about 30%) shifted observations. The other regression based filters are slightly worse. Obviously, using a short inner window for an initial fit improves the stability under shift. Application of another repeated median in the second step improves the results only slightly, while application of least squares is not much worse.

The results change substantially in case of a trend, particularly if the shift is in the opposite direction of the trend. The location based filters worsen a lot and smooth a level shift considerably then. The regression based procedures are not affected by a trend because of their invariance, and only the DWMTM performs comparably to them.

We note that the RMSE of most methods is dominated by the bias. The variability contributes substantially only for the median and the DWMTM, while for the MTM it increases when almost half of the observations are shifted. Moreover, if there are less than nine outliers we find the largest bias and RMSE of the trimming filters to occur for outliers of moderate size: large shifts are better preserved than small shifts. This is due to the fact that these filters are not positive, i.e. increasing some observations does not necessarily increase the output. The other methods show the intuitive behavior that larger outliers have stronger effects. Hence, trimming provides its main benefits if shifts are large relative to the observational noise.

Here and in the following, we find the main effect of positive autocorrelations ($\phi = 0.6$) to be a slight increase of variance. The ranking of the methods imposed by the bias is essentially the same as discussed before. Only the bias of the methods with a shorter inner window increases somewhat earlier, and the same is true to some extent for the RM.

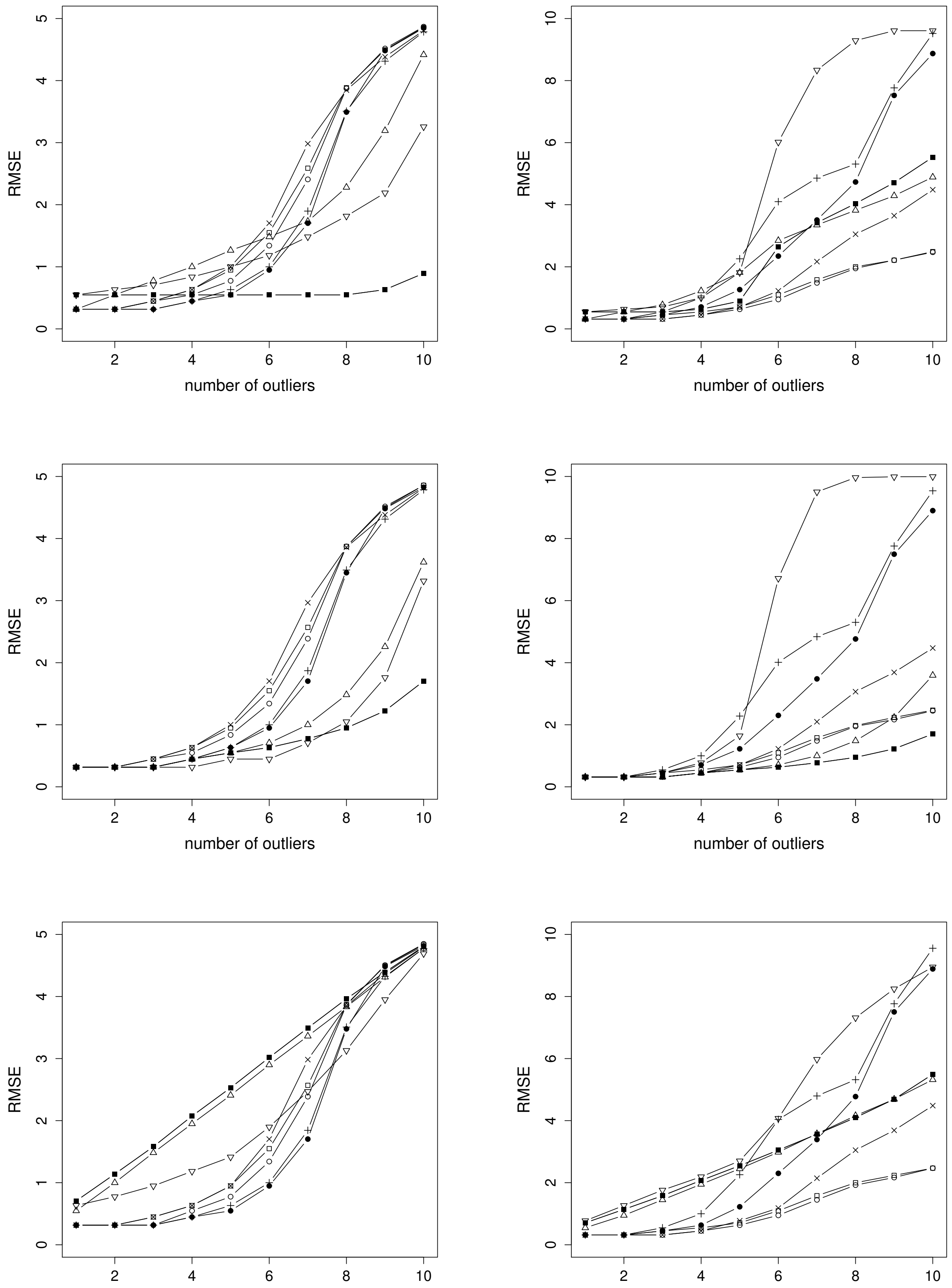

Figure 4: RMSE for the intrusion of a level shift (left) and for an outlier patch in the center (right), slope $\beta = 0.5$ (top), $\beta = 0.0$ (center) and $\beta = -0.5$ (bottom): median (■), RM (□), MTM ($\triangle$), DWMTM (∇), MRM (o), TRM ($\times$), DWMRM ($\bullet$) and DWTRM ($+$)

5.3 Removal of impulsive, patchy noise

In view of the clinical rule of thumb stated above we are interested in removing outlier patches with up to five subsequent outliers, but we consider larger numbers as well since more than one patch can occur within a short period.

For some of the methods the location of the outliers is essential for their effect on the filter output. The double window filters resist outliers best when they are outside the inner window. Since the window is moved through the series an outlier patch will be found at any point in the window at some time. We considered several settings with either one outlier patch in the center of the window or with two separated patches. The situation with one patch at the start or the end of the window has been treated implicitly in the previous subsection. We always replaced an increasing number $1, 2, \ldots, 10$ of observations by additive outliers of size $s \in \{1, 2, \ldots, 10\}$ and calculated the RMSE for all numbers and sizes of outliers from 2000 simulation runs each.

In the first situation there is one patch located in the center of the window, i.e. we replaced the observations at times $5, 4, \ldots, -4$ (in this order) by positive additive outliers, see Fig. 4. In case of a constant signal, all methods resist up to about five (25%) outliers as desired. The double window filters, especially the DWMTM and the DWTRM, become increasingly biased thereafter. The DWMTM is the only filter showing a strongly increasing variance, namely when there are about as many outliers in the inner window as clean observations. In case of a steep trend ($\beta = \pm 0.5$), the location based filters worsen a lot and become strongly biased. The MRM, the RM, and to a smaller extent the TRM are the only filters resisting many outliers then.

Considering next a situation with two outlier patches occurring within half a window width distance we replaced the observations at $t = 0, 10, -1, 9, -2, 8, -3, 7, -4, 6$ (in this order) by positive additive outliers, see Fig. 5. All methods perform well if the signal is constant as they are little affected by two patches with up to four outliers each. The TRM and the MTM-filters are slightly worse than the others in case of at least eight outliers. In case of a steep trend even the median becomes strongly biased and only the regression based procedures resist two patches with between two and four outliers each (20% - 40% altogether), with the DWMRM being slightly superior.

If the outlier patches have different signs (not shown here), application of least squares in the second step after application of the RM gives even slightly better results than a second RM, but the results are generally much better than if the patches have the same sign.

Regarding situations with one patch at the start and another at the end of the window we replaced the observations at $t = -10, 10, -9, 9, \ldots, -6, 6$ (in this order) by positive additive outliers. In case of a constant signal all methods resist up to 6 (30%) outliers then, see Fig. 5. The MTM shows an increasing bias thereafter, while the double window filters perform better than the others. The median deteriorates substantially in case of a trend showing the largest bias and MSE for a small to moderate number of outliers, and the DWMTM also worsens slightly. The regression based procedures, particularly the double window ones, are better then.

We find patches of different signs at the start and the end to cause minor problems only. The MSE is dominated by the variance in this situation since the bias produced by positive and by negative outliers cancels out.

5.4 Long time windows

For further investigation of the merits of a shorter inner window for the initial estimation we now increase n to 61 and set m to 21. Although a delay of $k = 30$ observations may sometimes be too long for online signal extraction, such choices are possible for retrospective analysis if the signal does not fluctuate too rapidly for a local linear approximation. In particular, there should never be more than one shift or signal peak within a window. In view of the previous results we restrict to a comparison of the regression based procedures. Hence, the results do not depend on a linear trend.

We reconsider the occurrence of a shift. We shift an increasing number of $3, 6, \ldots, 30$ observations at the end of the window and calculate the maximal RMSEs for outlier sizes $s \in \{1, \ldots, 10\}$ from 2000 simulation runs as before. Fig. 6 shows the clear benefits obtained from the inner window, while it does not seem to be important whether the repeated median or least squares is applied in the second step. The DWRM, which estimates the slope from the inner and the level from the whole window without trimming, resists the shift also considerably better than the ordinary RM. This is different from the combination $n = 21$ and $m = 15$, where we did not find significant advantages of the DWRM because n and m were close to each other.

Now we treat the occurrence of an outlier patch in the center. Since the RM turns out to be most vulnerable when an outlier patch is at an end of the window we replace an increasing number $1, 2, \ldots, 10$ of observations at times $10, 9, \ldots, 1$ as this should damage the double window filters most. The maximal RMSEs in Fig. 6 show that the double window filters indeed run into problems if at least 6 out of the 21 (about 30%) observations in the center are outlying with the DWMRM being the least and the DWRM the most affected. The other filters are not affected at all by ten central outliers. However, as stated before, patches of five or more subsequent outliers are often relevant in critical care, i.e. protection against up to four subsequent outliers might be sufficient in this application.

6 Application to time series

As a last step we applied the procedures to time series with an underlying complex signal. We chose the width $n = 31$ for all filters, and an inner window of $m = 11$ observations for the DWMTM and $m = 17$ for the RM-based filters, respectively. At the start and the end of the series we extrapolated the first and the last filtered value for the location

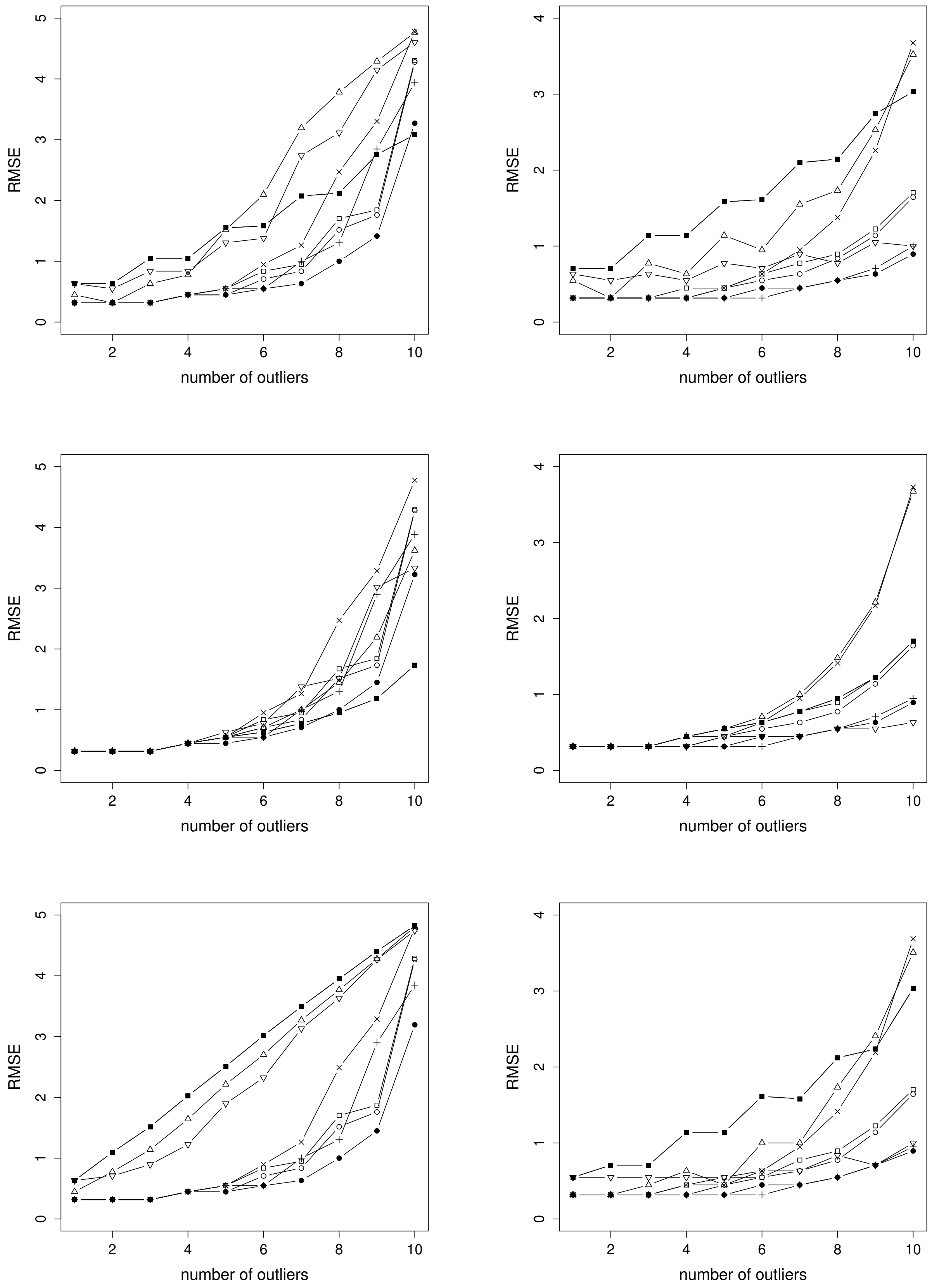

Figure 5: RMSE for outliers lagged by a half (left) and a full window width (right), slope $\beta = 0.5$ (top), $\beta = 0.0$ (center) and $\beta = -0.5$ (bottom): median (■), RM (□), MTM (△), DWMTM (∇), MRM (o), TRM (×), DWMRM (•) and DWTRM (+)

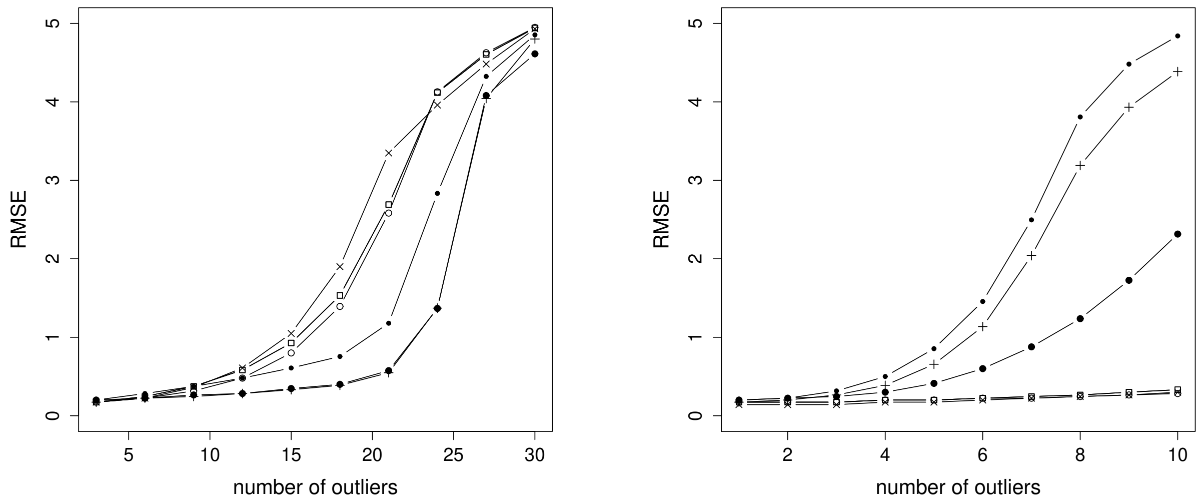

Figure 6: RMSE for the intrusion of a level shift (left) and for an outlier patch in the center (right): RM (□), MRM (o), TRM ($\times$), DWMRM ($\bullet$), DWTRM ($+$) and DWRM ($\cdot$)

based filters, while we used the regression lines fitted in the first and the last window for the RM-based filters.

6.1 Simulated time series

Fig. 7 shows a simulated time series of length 300 and results of signal extraction. The underlying signal contains constant as well as trend periods and two shifts, and it is overlaid by N(0,1) white noise. Twenty observations were replaced by negative additive outliers of size 5 split into four isolated, three pairs, two tripels and one quadrupel of outliers, which were inserted at time points chosen at random.

The RM is smoother than the median in trend periods and resists the outliers there slightly better. Both filters smooth the second level shift considerably. The MTM behaves similarly to the median, while the MRM and the TRM preserve the second shift much better. The DWMTM behaves excellently at the first and similarly to the MRM at the second shift, but the double window regression filters do even better, and they are additionally more robust against outliers. The possibly better preservation of shifts by the DWMTM as compared to the median has been noticed before (Pitas and Venetsanopoulos 1992, Himayat and Kassam 1993).

6.2 Real time series

We finally analysed a real time series representing the pulmonary arterial mean pressure of an intensive care patient, see also Fig. 7. This series includes shifts, trends and outlier patches. Again, the RM is smoother than the median during the trends. Both blur the shift at about $t = 100$ and the local minimum there considerably. The MTM performs

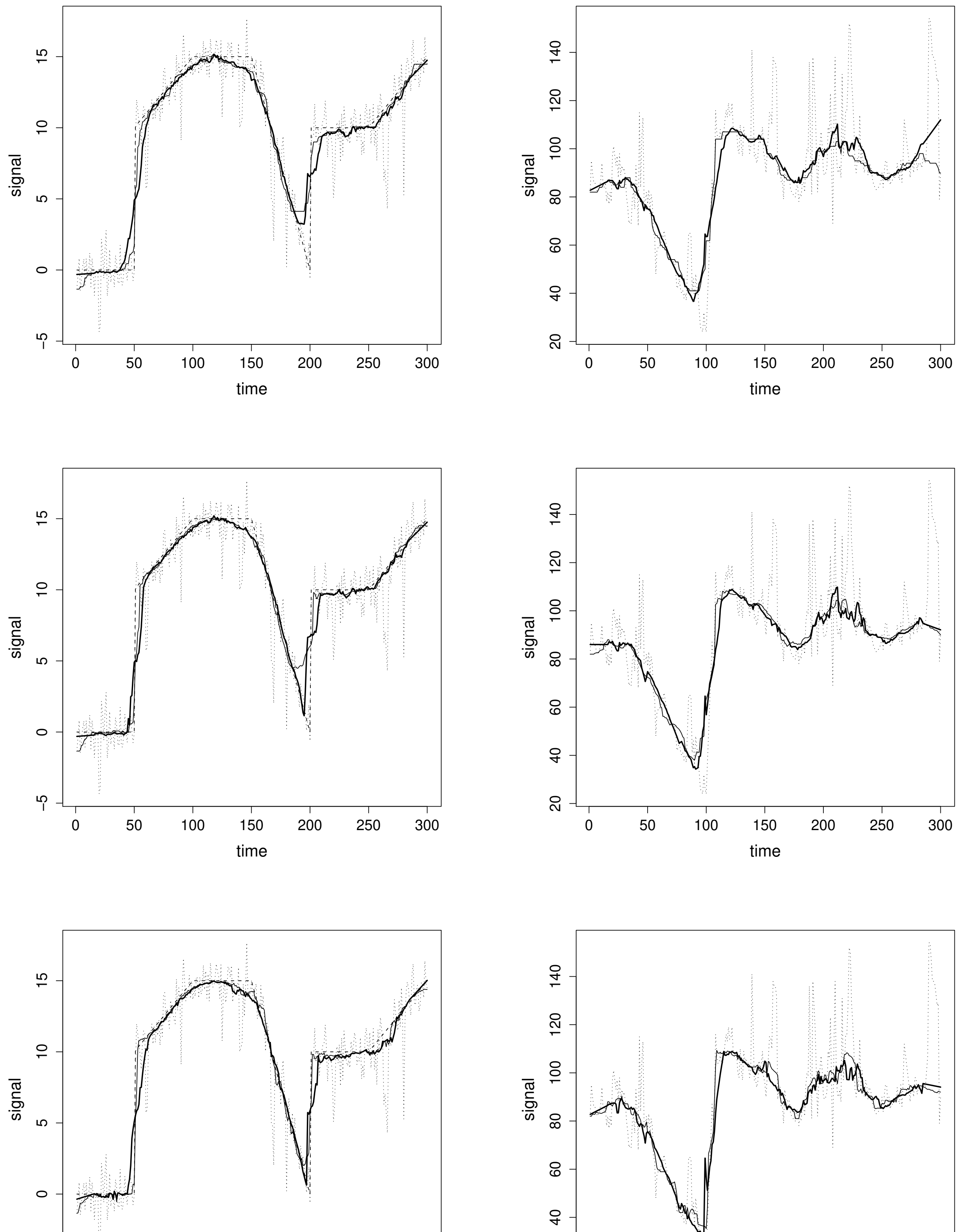

Figure 7: Left: Simulated data (dotted), underlying signal (dashed) and estimates. Right: Pulmonary arterial pressure (dotted) and signal estimates. Top: Median (thin solid) and RM (bold solid). Center: MTM (thin solid) and MRM (bold solid). Bottom: DWMTM (thin solid) and DWTRM (bold solid).

similarly to the median, and the MRM to the RM. The DWMTM and the DWTRM blur the shift only very slightly, with the DWTRM being smoother and somewhat less influenced by long outlier patches.

7 Summary

We have compared robust filters for signal extraction from time series with trends, shifts and outliers. Modified trimmed means perform well in case of constant signals and moderate trends, but they deteriorate during steep trends, similarly to running medians. Application of repeated median regression eliminates the influence of local linear trends and allows to maintain considerable efficiency and a high degree of robustness. The main disadvantage might be the increased smearing of shifts in case of a constant signal.

We have proposed two-stage procedures trimming observations with large regression residuals to improve shift preservation. For this to be effective, we should choose a considerably shorter width for the inner signal window. However, lower limits for the length of the signal window are imposed by the need for robustness: As a rule of thumb, the inner window should be chosen at least three or four times as long as outlier patches to be removed. We have found that the repeated median resists this fraction of contamination well. The trimming constant q_t can be chosen according to the expected height of shifts, similarly as discussed by Lee and Kassam (1985) for modified trimmed mean filters.

Instead of the repeated median, we could have used the least median of squares (LMS) (Rousseeuw 1984) in the first step in combination with least squares regression. The resulting procedure would resemble the popular reweighted least squares (Rousseeuw and Leroy 1987, Gervini and Yohai 2002). However, the instability of the LMS is dangerous in automatic applications (Davies *et al.* 2004), and weighting the observations according to their distance from the LMS fit does not necessarily remove this instability when the observations in the window are located close to two or more straight lines. When using the repeated median for the initial fit as is done here we have not observed such instabilities in spite of the lack of continuity. In view of the computational savings possible by the fast update algorithm described in this paper, we find double window filters combining the repeated median and least squares a promising alternative to established methods like double window modified trimmed means.

Acknowledgements

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of complexity in multivariate data structures") is gratefully acknowledged. The authors thank Eleni Mitropoulou for additional data analysis.

References

Bernholt T. and Fried R. 2003. Computing the update of the repeated median regression line in linear time. Information Processing Letters 88: 111-117.

Cormen T. H., Leiserson C. E., Rivest R. L. and Stein, C. (2001). Introduction to Algorithms. Second edition. MIT Press, Cambridge, Massachusetts, and McGraw-Hill Book Company, New York.

Davies P., Fried R. and Gather U. 2004. Robust signal extraction for on-line monitoring data. Journal of Statistical Planning and Inference 122: 65-78.

Fried R. 2004. Robust filtering of time series with trends. Journal of Nonparametric Statistics 16: 313-328.

Fried R., Bernholt T. and Gather U. 2004. Repeated median and hybrid filters. Computational Statistics & Data Analysis, to appear.

Gather U. and Fried R. 2003. Robust estimation of scale for local linear temporal trends. Tatra Mountains Mathematical Publications 26: 87-101.

Gervini D. and Yohai V.J. (2002). A class of robust and fully efficient regression estimators. Annals of Statistics 30: 583-616.

Himayat N. and Kassam S.A. 1993. Approximate performance analysis of edge preserving filters. IEEE Transactions on Signal Processing 4: 2764-2776.

Imhoff M., Bauer M., Gather U. and Fried R. 2002. Pattern detection in intensive care monitoring time series with autoregressive models: Influence of the model order. Biometrical Journal 44: 746-761.

Lee Y. and Kassam S. 1985. Generalized median filtering and related nonlinear filtering techniques. IEEE Transactions on Acoustics, Speech, and Signal Processing 33: 672-683.

Matoušek J., Mount D.M. and Netanyahu N. 1998. Efficient randomized algorithms for the repeated median line estimator. Algorithmica 20: 136-150.

Pitas I. and Venetsanopoulos A. 1992. Order statistics in digital image processing. Proceedings of the IEEE 80: 1893-1921.

Rousseeuw P.J. (1984). Least median of squares regression. Journal of the American Statistical Association 79: 871-880.

Rousseeuw P.J. and Leroy A.M. 1987. Robust Regression and Outlier Detection. Wiley, New York.

Siegel A.F. 1982. Robust regression using repeated medians. Biometrika 69: 242-244.

Tukey J.W. 1977. Exploratory Data Analysis. Addison-Wesley, Reading, Mass. (Preliminary edition 1971).