Computation of the Oja median by bounded search

Karl Mosler and Oleksii Pokotylo

Abstract A new algorithm is given for the exact calculation of the Oja median. It modifies the algorithm of Ronkainen, Oja and Orponen (2003) by employing bounded regions which contain the median. The regions are built using the centered rank function. The new algorithm is faster and has less complexity than the previous one. It is also used for an even faster approximative calculation.

1 Introduction

A basic task in multivariate analysis is to describe the general location of data by some point in their middle. Several notions of multivariate medians have been proposed in the literature. They extend different properties and characterizations of the usual univariate median to Euclidean *k*-space. Besides these defining characterizations the multivariate medians may be distinguished by their invariance properties. These include invariances against monotone transformations of the marginals (like the componentwise median), against spherical transformations (like the spatial median), against affine transformations (like the Oja median, proposed in the seminal paper (Oja, 1983)), and combinatorial invariance. The latter means that the data may be varied in their compartments without changing the median. Examples are the Tukey median (Tukey, 1975) and the simplicial median by Liu (1988). These medians are, at least in some sense, more robust against outlying data than the arithmetic mean, which is the center of gravity. Multivariate medians are surveyed by Small (1997) and Oja (2013).

Karl Mosler
University of Cologne, Institute of Econometrics and Statistics, 50923 Köln e-mail: mosler@statistik.uni-koeln.de

Oleksii Pokotylo
University of Cologne, Cologne Graduate School, 50923 Köln e-mail: pokotylo@wiso.uni-koeln.de

Like the univariate median most of the multivariate medians can be regarded as maximizers of goal functions, so called data depths, the Tukey depth, the simplicial depth, the Oja depth, and the spatial depth, among others. See Mosler (2013) for a recent survey.

To be applicable to realistic problems, a median must be computable for dimensions $k > 2$ and at least medium sized data sets. Here we develop an algorithm to calculate the exact value of the Oja median and demonstrate that it is faster, having also less complexity, than the existing ones by Niinimaa, Oja and Nyblom (1992) and Ronkainen, Oja and Orponen (2003), ROO hereafter. The exact algorithm can also serve as a benchmark for faster heuristic procedures. In principle, the computation of the Oja median involves repeated checking of all intersections of hyperplanes generated by the data. Our main idea is to introduce bounding hyperplanes that iteratively restrict the area where the median is searched.

The paper is structured as follows: Section 2 introduces the Oja median and depth and some basic notions and properties connected with them, it also sketches the algorithm of Ronkainen, Oja and Orponen (2003) for exact calculation of the Oja median. In Section 3 the ideas of the new bounding procedure are discussed, followed by a description of the algorithm in Section 4. Finally, in Section 5 numerical experience is reported regarding data in $\mathbb{R}^k$ for k up to dimension seven.

2 Oja median and depth

Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ be a data set of observations in $\mathbb{R}^k$. Each k observations $\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}$ generate an *observation hyperplane* passing through them, which is notated by $p = (i_1, ..., i_k)$, $1 \le i_1 < ... < i_k \le n$. Let P denote the set of all $\binom{n}{k}$ observation hyperplanes.

k observations together with a given point $\mathbf{x} \in \mathbb{R}^k$ span a simplex in k-space. Its k-dimensional volume is found as

$$\begin{aligned} V_p(\mathbf{x}) := V(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}, \mathbf{x}) &= \frac{1}{k!} abs\left(\begin{vmatrix} 1 & ... & 1 & 1 \\ \mathbf{x}_{i_1} & ... & \mathbf{x}_{i_k} & \mathbf{x} \end{vmatrix} \right) \\ &= \frac{1}{k!} abs(d_{0p} + \mathbf{d}_p{}^{\top} \mathbf{x}). \end{aligned}$$

Here d_{0p} is the distance of the hyperplane p from the origin, and $\mathbf{d}_p$ is its normal, given by the vector of cofactors of $\mathbf{x}$ in the determinant. The average of all such volumes is mentioned as the *Oja outlyingness function* of $\mathbf{x}$,

$$\begin{aligned} O(\mathbf{x}|\mathbf{X}) &= ave_{i_1<...<i_k} \left(V(\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_k}, \mathbf{x}) \right) \\ &= ave_{i_1<...<i_k} \left(\frac{1}{k!} abs \left(\begin{vmatrix} 1 & ... & 1 & 1 \\ \mathbf{x}_{i_1} & ... & \mathbf{x}_{i_k} & \mathbf{x} \end{vmatrix} \right) \right) \\ &= \frac{1}{k!} ave_{p \in P} \left(abs(d_{0p} + \mathbf{d}_p^{\top} \mathbf{x}) \right). \end{aligned} \qquad (1)$$

It is clear from (1) that the Oja outlyingness function is piecewise linear and convex on $\mathbf{x}$ as well as continuous on $\mathbf{x}$ and the data in $\mathbf{X}$. The minimizer of the outlyingness function is the *Oja median*, $\mathbf{Med}(\mathbf{X})$. Generally, this median is not unique but forms a convex set. The Oja median is a measure of location and *affine equivariant* regarding $\mathbf{X}$,

$$\mathbf{Med}(\mathbf{Y}) = \mathbf{A}\mathbf{Med}(\mathbf{X}) + \mathbf{b}\,, \tag{2}$$

if $\mathbf{Y} = \{\mathbf{A}\mathbf{x}_1 + \mathbf{b}, \dots, \mathbf{A}\mathbf{x}_n + \mathbf{b}\}$ with some matrix $\mathbf{A}$ of full rank k and $\mathbf{b} \in \mathbb{R}^k$; see Oja (1983). The outlyingness function can be made affine invariant (to simultaneous transformation of $\mathbf{x}$ and $\mathbf{X}$) by multiplying it with a proper scale factor, *viz.* $(\det \mathbf{S}(\mathbf{X}))^{-1/2}$, where $\mathbf{S}(\mathbf{X})$ is a positive definite $k \times k$ matrix depending on $\mathbf{X}$ and measuring the dispersion of the data cloud $\mathbf{X}$ in an affine equivariant way, that is, with $\mathbf{Y}$ as above, satisfying

$$\mathbf{S}(\mathbf{Y}) = \mathbf{A}^\top \mathbf{S}(\mathbf{X})\mathbf{A}. \tag{3}$$

In particular, the usual covariance matrix of $\mathbf{X}$ can serve as $\mathbf{S}(\mathbf{X})$. The *Oja depth function* is defined as (Zuo and Serfling , 2000)

$$depth(\mathbf{x}|\mathbf{X}) = \frac{1}{1 + O(\mathbf{x}|\mathbf{X})(\det \mathbf{S}(\mathbf{X}))^{-1/2}}\,. \tag{4}$$

Observe that the Oja depth function is affine invariant and continuous. It is maximal at the Oja median of $\mathbf{X}$ and vanishes for $||\mathbf{x}|| \to \infty$. Given $\mathbf{X}$, the depth function is a strictly decreasing transformation of the outlyingness function and, thus, the contour lines of the two functions coincide, though at different values. As the function $O(\cdot|\mathbf{X})$ is convex, all its contour lines are convex. Hence the level sets of the Oja depth are convex and compact sets in $\mathbb{R}^k$. Moreover, the Oja depth decreases monotonically on rays from each point in the median set.

In the case of a centrally symmetric distribution the median set includes the center of symmetry. It can be shown that the Oja depth function determines the data cloud $\mathbf{X}$ uniquely (Koshevoy , 2003). The usual breakdown point of the Oja depth is zero, while a slightly different notion of breakdown appears to be positive (Niinimaa, Oja and Tableman , 1990).

Given $\mathbf{X}$, the *centered rank function R* is defined by

$$\mathbf{R}(\mathbf{x}) = \frac{1}{k!} ave_{p \in P}\left(S_p(\mathbf{x})\mathbf{d}_p\right),$$

where

$$S_p(\mathbf{x}) = sign(d_{0p} + \mathbf{d}_p{}^\top \mathbf{x}),$$

indicates on which side of the hyperplane p the point $\mathbf{x}$ is located. Note that $\mathbf{R}(\mathbf{x})$ is the derivative of (1), at all $\mathbf{x}$ at which $O(\cdot|\mathbf{X})$ is smooth. Hence, as $O(\cdot|\mathbf{X})$ is convex, the centered rank function is a subgradient of the outlyingness function, at all $\mathbf{x} \in \mathbb{R}^k$. Below, $-\mathbf{R}(\mathbf{x})$ will be used as a direction of descent at point $\mathbf{x}$. It is easily seen from (1) that the outlyingness function is also represented as

$$\begin{aligned} O(\mathbf{x}) &= \frac{1}{k!}\Big(ave_{p\in P}(S_p(\mathbf{x})d_{0p}) + ave_{p\in P}(S_p(\mathbf{x})\mathbf{d}_p{}^{\top}\mathbf{x})\Big) \\ &= \frac{1}{k!}\frac{1}{\binom{n}{k}}\Big(D_0(\mathbf{x}) + \mathbf{D}(\mathbf{x})^{\top}\mathbf{x}\Big)\,, \end{aligned} \tag{5}$$

where the sums,

$$D_0(\mathbf{x}) = \sum_{p\in P} S_p(\mathbf{x})d_{0p}\,, \quad \mathbf{D}(\mathbf{x}) = \sum_{p\in P} S_p(\mathbf{x})\mathbf{d}_p\,, \tag{6}$$

are piecewise constant. They change by $2d_{0p}$ and $2\mathbf{d}_p$, respectively, when a hyperplane p is crossed.

2.1 Calculating the median according to ROO

In what follows we assume that the data are in general position. Hettmansperger, Möttönen and Oja (1999) have shown that a version of the Oja median is always found among the intersection points of observation hyperplanes. The exact algorithm of ROO iteratively optimizes the outlyingness function along the intersection lines of $k-1$ observation hyperplanes, called *observation lines*. At first a searching line is randomly selected among the observation lines and the outlyingness function is optimized along the line. When the point of the minimum is found, the next searching line through this point is chosen. The possible choices of lines depend on the type of the point: the smallest number of lines is obtained if the point is an intersection of hyperplanes that have no common observation points, the largest number is obtained if the point coincides with one of the observation points; see also the discussion before subsection 4.1.

Minimizing the outlyingness function along the searching line is the most time consuming task. The chosen line L is intersected with all hyperplanes and the outlyingness function (1) is calculated at each intersection point. At the first intersection point the constant terms d_{0p} and $\mathbf{d}_p$ are summed up along with the signs $S_p(\mathbf{x}_m)$, yielding the sums D_0, and $\mathbf{D}$ according to (6). Then the other intersections are considered step by step. The outlyingness function is calculated as in (5). In each new point one of the hyperplanes changes its sign and the sums D_p and $\mathbf{D}$ are updated. Note, that there are $\binom{n}{k}$ intersections, almost all of which have to be considered, which causes the great complexity of the algorithm.

3 A bounding approach

The centered rank function is a subgradient of the outlyingness function. Note that no unique gradient exists at intersections of the observation hyperplanes, hence the centered rank function will in general not vanish at the Oja median. The negative rank function (= negative subgradient) $-\mathbf{R}(\mathbf{x})$ is a vector that points in a direction of descent of the outlyingness function, hence ascent of the depth function. It defines a hyperplane through $\mathbf{x}$, which separates the space into two halfspaces. The positive side of the hyperplane is indicated by the negative subgradient, which equals the negative rank function. Therefore, the Oja median is always found on the positive side of these hyperplanes.

Regarding the Oja depth function, observe that its subgradients have the same direction as the negative subgradients of the Oja outlyingness function,

$$\mathbf{grad}\, depth(\mathbf{x}) = -\mathbf{R}(\mathbf{x})\,(\det \mathbf{S}(\mathbf{X}))^{-1/2}\,(depth(\mathbf{x}))^2 .$$

Their contour lines coincide since the depth function is a strictly decreasing transform of the outlyingness function.

An example of Oja depth contours and subgradients of the depth function is shown in Fig. 1. As expected, all negative subgradients point to the halfspace containing the median, and the gradients are perpendicular to the depth contours.

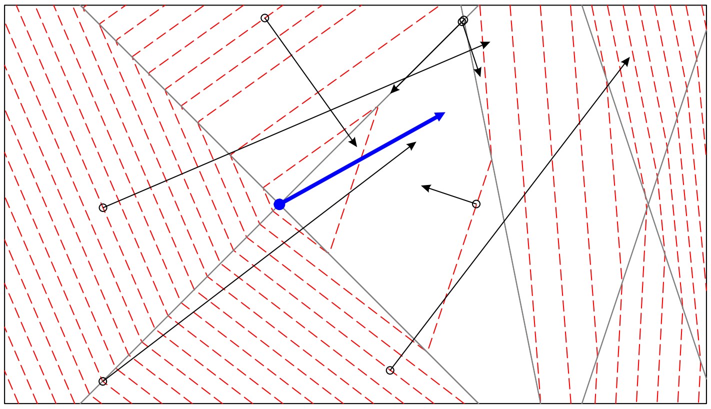

Fig. 1 An example of Oja depth contours with values of the negative rank function. The median (unique) is shown at the intersection of the observation lines as a bold point, together with its subgradient.

The halfspaces defined by the negative rank function can be used to build a bounded region that contains the median. In our algorithm we select those halfspaces in an iterative way and restrict the further search to their intersection. The hyperplanes bordering such a search region will be called *bounding hyperplanes* or simply *bounds*. The bounded regions reduce the complexity of the searching procedure by reducing the number of hyperplanes that cross the searching lines as well as the number of their intersections actually considered in the minimization procedure.

The obtained hyperplanes form a bounded region, which is the intersection of the positive sides of the hyperplanes. Actually, such a bounded region is determined by part of these hyperplanes only, as bounds lying outside the region provide no additional information. In our algorithm, we adjust the bounded regions step by step. We begin with a rectangular region limited by hyperplanes that are perpendicular to the coordinate axes and go through the maximal and minimal coordinates of the data points on these axes. Then we add hyperplanes as new bounds. For each added new bound it is checked whether it is *efficient*, that is, actually crosses the bounded region, and thus reduces it. Then the intersection of the new hyperplane with the bounded region is determined, and all bounds that are made inefficient by the new one are removed. To check whether a hyperplane crosses the bounded region, it suffices to check if there exist any two bounds' intersections lying on different sides of the hyperplane. As the calculation of the Oja rank function is itself a rather expensive operation, we will try to obtain the smallest possible central region by performing as few calculation as possible.

We have developed several approaches of the iterative bounds search. The divisive approach (A) is the simplest solution.

Approach A:

The bounded region is iteratively reduced by a divisive approach (A) *viz.* by iteratively adding hyperplanes that go through a properly chosen central point of the region and have their normal vectors equal to the corresponding negative rank function. The central point should be selected to cut a large amount of volume from the bounded region, and shall ideally be the center of the volume, so that any hyperplane through this point will approximately cut off half of the bounded region's volume. Here, we select the mean value of the bounds' intersection points as a central point. As the region is reduced by a hyperplane through the central point, it is expected that its volume shall become (on an average) twice smaller at each step. Ideally, after nine such steps, in any dimension k, a subspace volume of approximately 0.1% of the initial one should be obtained. The experiments in section 5 show that the volumes decrease slower in concrete calculations.

The divisive approach (A) considers only the directions of the subgradients, although their lengths also give the information about the location of the median. Another solution (approach B) consists in moving along the subgradients as it is shown in Fig. 2.a. The length of $\mathbf{R}(\mathbf{x})$ decreases as $\mathbf{x}$ moves towards the median.

Approach B:

1) Start with $i = 0$. Select an initial point $\mathbf{x}_0$. Specifically, we choose the componentwise median of all observations.

2) Determine the subgradient $-\mathbf{R}(\mathbf{x}_i)$.

3) Add the subgradient vector, $\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{R}(\mathbf{x}_i)$, and continue.

We continue building such gradients, each time getting closer to the median, until they become either zero or increase in length. The zero case means that the point $\mathbf{x}_i$ lies in the median set, where the Oja depth assumes its minimal value. As it is seen in Fig. 1, the subgradient's length depends not only on the distance from the median, but also on the subspace, formed by hyperplanes, that contains $\mathbf{x}_i$. Thus if the gradients become longer, their lengths may be restricted to the length of the shortest one, and this bound will consequently decrease.

Several of the gradients found may be used to build the bounded searching region, containing the median. The points having shortest gradients are closest to the median. An example of a bounded region built on such gradients is shown in Fig. 2.b.

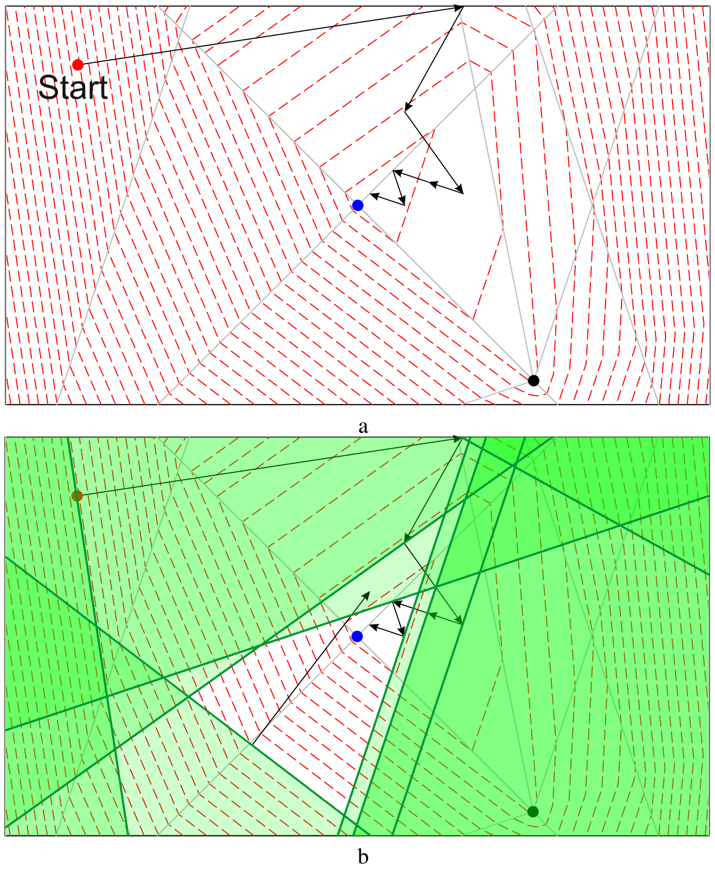

Fig. 2 A gradient path (a) and a bounded region (b), built using the gradients. The subspaces, cut off by each of the bounds are shaded.

The divisive approach needs an almost constant number of calculations to reach the intended volume. However, the efficiency of moving along the subgradients (approach A) strongly depends on the form of the data. In most cases, the subsequent gradients extend in rather different directions, and the volume of the bounded region decreases fast. But in certain cases, especially with asymmetric datasets, this is not true. The subgradients in the sequence may approach the median in a more common direction and thus leave too much space inside the bounded region. The gradients may also end outside the bounded region or jump between two subsets formed by the observation hyperplanes, providing not much information on each step.

It is therefore reasonable to start with moving along the subgradients, and then, as soon as this procedure slows down, shift to the divisive procedure, until the needed volume is reached:

Approach C:

This yields the following hybrid approach, where the next cutting point may be defined as the end of the subgradient, $\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{R}(\mathbf{x}_i)$, as long as it lies inside of the bounded region, or as the center of the bounded region otherwise.

Approach D:

Also the direction of the subgradients can be used to define the next cutting point as a central point of the segment between the subgradient's origin and its intersection with the bound.

Further, the calculation can be accelerated by using rougher bounds, *viz.* enlarging the given bounded region by a circumscribed k-variate box. Then *a fortiori* a point lies outside the bounded region if it lies outside the circumscribed box.

Once a bounded region is defined, the observation hyperplanes lying outside of it are excluded from the searching process, which decreases the number of intersections when minimizing on a line. A problem may occur if the bounded region contains no path through the intersections of the observation lines from the initial searching line to the line containing the median. Such a path connecting any two observation lines may be provided by including the bounds themselves into the searching process as ordinary observation hyperplanes. Fig. 3 shows an example of a path from the initial line through the observation lines and bounds to the median.

The bounding method may also be used to find the median in an approximative way with some given precision. The space may be cut until the bounded region has the proper size and its center may be taken as an approximation of the median. It is clear, that the median cannot lie outside the bounded region, so its center can be assumed to be the median with precision equal to half of the region's size. As the method considers all observation hyperplanes, it cannot be more efficient than existing approximative methods that consider subsamples of the data.

4 The algorithm

To start with, the first bounded region is created as described in the previous section. The desired size of the bounded region is selected as a part of the original volume. Here the volume is calculated as the volume of a minimal multivariate circumscribed rectangle with edges parallel to the coordinate axes. In subsequent iterations the first bounded region is reduced until the desired volume is reached. Here, the divisive approach (A) is considered, as it shows the best results in experiments (see section 5).

Next the initial line is determined. In a two-dimensional space any of the observation lines crossing the bounded region may be selected. In higher dimensions

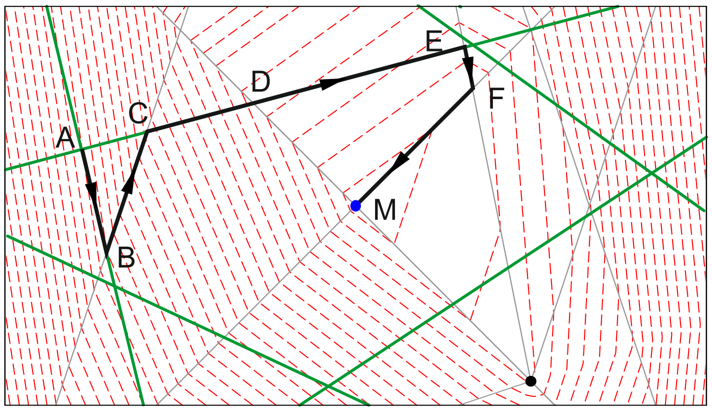

Fig. 3 A path through the observation lines (thin) and the bounds (bold). We start from taking one of the bounds AB as the initial line, and find a minimum point B. Then the outlyingness function is minimized along the next line through this point. As it is seen from the line CE, the point of minimum E is not necessarily the closest one (D) to the median, and the selected path may be not the shortest one. The paths BC and EFM are isolated, as there are no observation lines inside the bounded region to connect them, but they are connected with the bound CE.

the search of the initial line is more complicated. All intersections of $(k-1)$ hyperplanes are inspected until a first intersection line that crosses the bounded region is found. For this, we start with the lines that border the initial bounded region, which makes the search for a fitting line much easier.

It is clear that all points inside the bounded region lie on the same side of any hyperplane which does not cross this region. Therefore, the respective parts of the sums in (6) can be calculated beforehand, which significantly decreases the number of calculations on each step. Thus, on every searching line we may restrict ourselves to iterating the remaining hyperplanes.

The bounded region reduces the procedure of minimization along a line to its part lying inside the region. The searching line is usually intersected by most of the bounds. Therefore the two bounds that cut the bounding region at the intersection line are of primary interest. In order to find these bounds, all bounds are sorted according to their intersections with the searching line. Then the intersection point of the first bound with the searching line is taken as a reference point. The first bound which has the reference point on its positive side is selected as well as the previous one. If the searching line goes through the bounded region, all other bounds must have the reference point on the positive side. This property is used to determine whether a searching line hits the bounded region in dimensions higher than two, as there exist hyperplanes that are crossing the bounded region, but whose intersection line lies outside of it, as it is shown in Fig. 4 for a two-dimensional example and in Fig. 5 for a higher dimensional one.

The searching line is intersected with the included hyperplanes, and the outlyingness function (1) is calculated at every intersection point that lies between the two bounds, found on the previous step. At first, the hyperplanes that intersect the

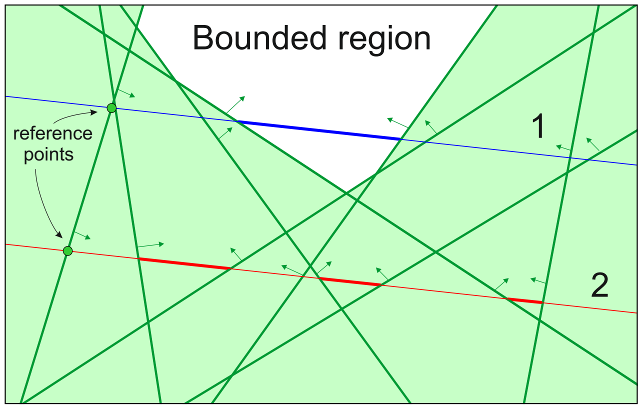

Fig. 4 Two lines: crossing the bounded region (1) and lying outside of it (2). The arrows show the positive sides of the hyperplanes. The segments between the bounds, one having the reference point on the negative and another one on the positive side, are shown in bold. A line that hits the bounded region has only one such segment.

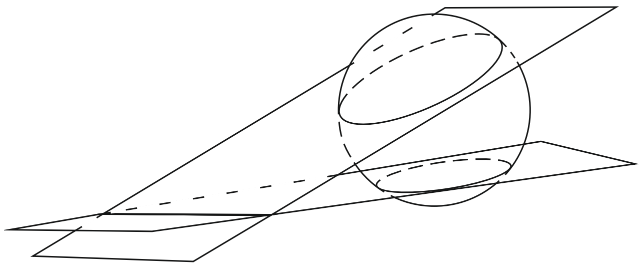

Fig. 5 An example of an observation line lying outside of the bounded region (shown as a sphere) formed by two hyperplanes crossing the bounded region in a higher dimensional space.

line outside the rougher, i.e. more liberal, bound are filtered out and added to (6). Then the first bound's intersection is taken as a median candidate, and as a starting point for the minimization procedure. The left hyperplanes are added to (6) with the sign they have in the first bound's intersection. The intersection points are iterated, the corresponding hyperplanes change the sign in the sum (6) and the outlyingness function is calculated as in (5). The outlyingness function is also calculated at the intersections with the bounds. When the second bound's intersection is reached, the procedure is terminated. The outlyingness function has convex contours and therefore is unimodal on any line. However, in practice the outlyingness function may slightly fluctuate when it is optimized along a line. In this case, as soon as the outlyingness value begins to increase by a certain threshold amount, the minimization along the line is terminated.

When the minimum is found on the searching line, the next observation line is chosen among the lines that contain the minimum. In the simplest case, k hyperplanes, each defined by k unique observation points, define a point at their intersection and produce k observation lines through this point. More complex cases occur when some of the hyperplanes have observation points in common, and their intersection point lies in an affine subspace of dimension $d < k$, generated by these common points. Such a point may then be described by all possible observation hyperplanes that have the same common points, and thus the number of observation lines increases. If the number of observation lines exceeds the predefined maximum number max_{n_L}, ROO propose either to stop, or to take a random subset of these lines. Fischer et al. (2010) in their R-package *OjaNP* used to choose a new initial line in such cases.

If the minimum is defined with one or more bounds, we treat them like ordinary hyperplanes. In order to explicitly determine the bounded region's boarding lines and corners, the bounds are identified by k unique points that are found as intersections with the coordinate axes. If a bound is parallel to some of the axes, the diagonal axes in the space are taken. Thus the bounds do not have identifying points in common, and each intersection of the bounds and observation hyperplanes produces a minimum possible number of observation lines.

4.1 Formal description of the algorithm

The formal description has modifies the one of Ronkainen, Oja and Orponen (2003, *A.1, A.2*) and includes parts of it to make the comparison easier. In particular, Procedure 1 extends A.1 with the bounded region search (steps 2–14), and Procedure 3 modifies the minimization algorithm A.2 to be used in a bounded region (added steps 1-6, modified steps 18-31). Procedure 2 describes the bounded region construction as in the divisive approach A.

Procedure 1. Compute the exact Oja median.

```
Input:  Data set $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ in $\mathbb{R}^k$.
        The desired size $s$ of the bounded box, $s = \frac{bounded\ box\ volume}{original\ volume}$.
        Max number of observation lines to scan $max_{n_L}$.
Output: Exact Oja median $\mathbf{T} = \mathbf{Med}(\mathbf{X})$.
1: Precalculate all observation hyperplanes $p = (i_1, ..., i_k)$, $1 \le i_1 < ... < i_k \le n$.
2: Build the bounded region $\mathbf{B}$, that is, the set of bounds defining it, using procedure 2.
        Chose the initial line $L$:
3: for all subsets $\mathbf{B}_s \subset \mathbf{B}$ with $|\mathbf{B}_s| = k-1$ do        ▷ find lines
4:     Set $L \leftarrow \bigcap \mathbf{B}_s$.
5:     Sort the bounds $\mathbf{b} \in \mathbf{B}$ according to their intersection points with $L$ as ROO
       do in A.2, i.e. if $L = \{\mathbf{L}_0 + \beta \mathbf{u}_L : \beta \in \mathbb{R}\}$ and we have $\mathbf{b}_i \cap L = \{\mathbf{L}_0 + \beta_i \mathbf{u}_L\}$ and
       $\mathbf{b}_j \cap L = \{\mathbf{L}_0 + \beta_j \mathbf{u}_L\}$ for some $\mathbf{b}_i, \mathbf{b}_j \in \mathbf{B}$, then $i < j \iff \beta_i < \beta_j$. Denote the
       order $\mathbf{b}_{(1)}, \mathbf{b}_{(2)}, ..., \mathbf{b}_{(nb)}$, where $nb = |\mathbf{B}|$.
6:     Set $\mathbf{y}_1 \leftarrow L \cap \mathbf{b}_{(1)}$.
7:     $i \leftarrow$ smallest $i$ at which $S_{\mathbf{b}_{(i)}}(\mathbf{y}_1) = 1$.
```

```
 8:     if $\exists j : j > i, S_{\mathbf{b}_{(j)}}(\mathbf{y}_1) \neq 1$ then
 9:         Continue                                    ▷ the line is out of bounds
10:     else
11:         Break                                       ▷ the line is found
12:     end if
13: end for

14: Precalculate $\frac{1}{k!}\times$ the common part of (6), for given $\mathbf{t}$ in the bounded region:
        $\mathbf{H} \leftarrow \sum_{p \notin \mathbf{B}} \frac{1}{k!} S_p(\mathbf{t})\mathbf{d}_p$,
        $H_0 \leftarrow \sum_{p \notin \mathbf{B}} \frac{1}{k!} S_p(\mathbf{t}) d_{0p}$.
15: Compute $\hat{\mathbf{T}} \leftarrow \arg\min_{\mathbf{t}\in L} O(\mathbf{t})$ using procedure 3.
16: Set the median candidate $\mathbf{T} \leftarrow \hat{\mathbf{T}}$.
17: Initialize the collection of investigated lines $\mathscr{L} \leftarrow \{L\}$.
18: Let $n_L$ be the number of the observation lines containing $\hat{\mathbf{T}}$.
19: if $n_L > max_{n_L}$ then
20:     There are too many possibilities. Goto 3.
21: end if
22: Construct the observation lines $\mathscr{L}' \leftarrow L_1, \dots L_{n_L}$.
23: Set $\mathscr{L}' \leftarrow \mathscr{L}' \setminus \mathscr{L}$.
24: while $\mathscr{L}' \neq \emptyset$ do
25:     Find the line $L \in \mathscr{L}'$ of deepest descent.
26:     Compute $\hat{\mathbf{T}} \leftarrow \arg\min_{\mathbf{t}\in L} O(\mathbf{t})$ using procedure 3.
27:     Update $\mathscr{L} \leftarrow \mathscr{L} \cup \{L\}$ and $\mathscr{L}' \leftarrow \mathscr{L}' \setminus \{L\}$
28:     if $O(\hat{\mathbf{T}}) < O(\mathbf{T})$ then
29:         $\mathbf{T} \leftarrow \hat{\mathbf{T}}$
30:         Goto 16.
31:     end if
32: end while
33: return $\mathbf{T}$
```

Procedure 2. Build the bounded region as in the divisive approach A, sec 3.

```
Input:  Data set $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ in $\mathbb{R}^k$.
        Precalculated observation hyperplanes $P$.
        The desired size $s$ of the bounded box, $s = \frac{bounded\ box\ volume}{original\ volume}$.
Output: The bounded region $\mathbf{B}$.
        Enclosing box $E$.
 1: Define $\mathbf{B} \leftarrow \emptyset$.                        ▷ the set of bounds
 2: Define $\mathbf{C} \leftarrow \emptyset$.                        ▷ the set of bounds' intersections
        Build the Initial Box:
 3: for $d = 1, ..., k$ do   ▷ index $(-d)$ means all coordinates from 1 to $k$ except of $d$
 4:     Set the seed of a bound $\mathbf{o}_d \leftarrow \max\{x_{1d}, ..., x_{nd}\}$, $\mathbf{o}_{-d} \leftarrow 0$.
 5:     Set the normal vector $\mathbf{n}_d \leftarrow -1$, $\mathbf{n}_{-d} \leftarrow 0$.
 6:     Define bound $\mathbf{b}$ with $\mathbf{o}$ and $\mathbf{n}$.
 7:     ADDBOUND($\mathbf{b}$)
 8:     Set the seed of a bound $\mathbf{o}_d \leftarrow \min\{x_{1d},...,x_{nd}\}$, $\mathbf{o}_{-d} \leftarrow 0$.
 9:     Set the normal vector $\mathbf{n}_d \leftarrow +1$, $\mathbf{n}_{-d} \leftarrow 0$.
10:     Define bound $\mathbf{b}$ with $\mathbf{o}$ and $\mathbf{n}$.
11:     ADDBOUND($\mathbf{b}$)
12: end for                                   ▷ the Initial Box is now built
        Proceed with the following divisions:
13: Calculate the original volume of the space as
        $OriginalVolume = \prod_{d=1}^{k} (\max\{x_{1d},...,x_{nd}\} - \min\{x_{1d},...,x_{nd}\})$
14: while $NewVolume/OriginalVolume > s$ do
15:     Define the center of $\mathbf{B}$ as $\bar{\mathbf{C}}$.
16:     Calculate the negative rank function $\mathbf{g} = -\mathbf{R}(\bar{\mathbf{C}})$.
17:     Define bound $\mathbf{b}$ with $\bar{\mathbf{C}}$ and $\mathbf{g}$.
18:     ADDBOUND($\mathbf{b}$)
19:     Calculate $NewVolume = \prod_{d=1}^{k} (\max\{\mathbf{C}_{1d},...,\mathbf{C}_{|\mathbf{C}|d}\} - \min\{\mathbf{C}_{1d},...,\mathbf{C}_{|\mathbf{C}|d}\})$
    using the updated intersection points.
20: end while

21: function ADDBOUND(new bound $\mathbf{b}$)
            ▷ here $(\mathbf{b} \cdot \mathbf{x})$ is the dot product of a point $\mathbf{x}$ and the normal vector of $\mathbf{b}$
22:     if (Initial Box is built) and
        $(sign(\mathbf{b} \cdot \mathbf{c}_1) = sign(\mathbf{b} \cdot \mathbf{c}_2) \forall \mathbf{c}_1, \mathbf{c}_2 \in \mathbf{C})$ then
23:         exit without changes                  ▷ $\mathbf{b}$ lies outside of $\mathbf{B}$
24:     end if
25:     for all subsets $\mathbf{B}_s \subset \mathbf{B}$ with $|\mathbf{B}_s| = k-1$ do      ▷ find new intersections
26:         Set $\mathbf{c} \leftarrow \bigcap(\mathbf{B}_s \cup \mathbf{b})$
27:         if $\forall \mathbf{b} \in \mathbf{B}$ $sign(\mathbf{b} \cdot \mathbf{c})! = -1$ then
28:             Add the new crossing point $\mathbf{C} \leftarrow \mathbf{C} \cup \mathbf{c}$.
29:         end if
30:     end for
31:     Add the new bound $\mathbf{B} \leftarrow \mathbf{B} \cup \mathbf{b}$.
32:     $\mathbf{C} \leftarrow \mathbf{C} \setminus \{\mathbf{c} : \mathbf{c} \in \mathbf{C}, sign(\mathbf{b} \cdot \mathbf{c}) = -1\}$.   ▷ Remove the cut off intersections
33:     $\mathbf{B} \leftarrow \mathbf{B} \setminus \{\mathbf{b} \in \mathbf{B} : sign(\mathbf{b} \cdot \mathbf{c}_1) = sign(\mathbf{b} \cdot \mathbf{c}_2) \forall \mathbf{c}_1, \mathbf{c}_2 \in \mathbf{C}\}$.
34: end function
```

Procedure 3. Minimize the outlyingness function O on the chosen line.

Input: Precalculated observation hyperplanes P.
Searching line L.
The bounded region $\mathbf{B}$.
Enclosing box E.

Output: The minimum $\hat{\mathbf{T}} \leftarrow \arg\min_{\mathbf{t} \in L} O(\mathbf{t})$ or an empty point if $L \cap \mathbf{B} = \emptyset$.

```
1: Sort bounds $\mathbf{b} \in \mathbf{B}$ according to their intersection points with $L$ as in procedure
   1.5, $\mathbf{b}_{(1)}, \mathbf{b}_{(2)}, ..., \mathbf{b}_{(nb)}$, where $nb = |\mathbf{B}|$.
2: Set $\mathbf{y}_1 \leftarrow L \cap \mathbf{b}_{(1)}$.
```

```
3: Set $\mathbf{y}_{b1} \leftarrow L \cap \mathbf{b}_{(i-1)}$ and $\mathbf{y}_{b2} \leftarrow L \cap \mathbf{b}_{(i)}$ where $i = \arg\min_i (S_{\mathbf{b}_{(i)}}(\mathbf{y}_1) = 1)$
4: if $\exists j : j > i, S_{\mathbf{b}_{(j)}}(\mathbf{y}_1) \neq 1$ then
5:     return empty point.                         ▷ the line is out of bounds
6: end if
7: Chose any point $\mathbf{t}_0 \in \mathbf{B} \cap L$ (e.g. $\mathbf{t}_0 = \mathbf{y}_{b1}$).
8: Initialize $\mathbf{D} \leftarrow \mathbf{H}$, $D_0 \leftarrow H_0$, $\mathscr{H} \leftarrow \emptyset$.
9: for all $p \in \mathbf{B}$ do  ▷ Compute the sum for hyperplanes, crossing $L$ outside of $E$.
10:     if $p \cap L \subset E$ then
11:         $\mathscr{H} \leftarrow \mathscr{H} \cup p$
12:     else
13:         $\mathbf{D} \leftarrow \mathbf{D} + \frac{1}{k!} S_p(\mathbf{t}_0)\mathbf{d}_p$
14:         $D_0 \leftarrow D_0 + \frac{1}{k!} S_p(\mathbf{t}_0) d_{0p}$.
15:     end if
16: end for
17: Sort hyperplane indexes $p \in \mathscr{H}$ according to their intersection points with $L$
    as ROO do in A.2, $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(np)}$, where $np = |\mathscr{H}|$ and $<$ resp. $\leq$
    denote the order of intersection points.
18: Define $\mathscr{H}_1 \leftarrow \{p : p \in \mathscr{H}, p \cap L < \mathbf{y}_{b1}\}$,
           $\mathscr{H}_2 \leftarrow \mathscr{H} \setminus \mathscr{H}_1$,
           $\mathscr{H}_3 \leftarrow \{p : p \in \mathscr{H}_2, p \cap L \leq \mathbf{y}_{b2}\}$.
19: Set $\mathbf{y}_1 \leftarrow L \cap p_{(1)}$ and $\mathbf{y}_{np} \leftarrow L \cap p_{(np)}$.
20: Compute $\mathbf{D} \leftarrow \mathbf{D} + \sum_{p \in \mathscr{H}_1} \frac{1}{k!} S_p(\mathbf{y}_{np})\mathbf{d}_p + \sum_{p \in \mathscr{H}_2} \frac{1}{k!} S_p(\mathbf{y}_1)\mathbf{d}_p$ and
           $D_0 \leftarrow D_0 + \sum_{p \in \mathscr{H}_1} \frac{1}{k!} S_p(\mathbf{y}_{np}) d_{0p} + \sum_{p \in \mathscr{H}_2} \frac{1}{k!} S_p(\mathbf{y}_1) d_{0p}$.
21: Set potential minimum $\hat{\mathbf{T}} \leftarrow \mathbf{y}_{b1}$.
22: Evaluate $O(\hat{\mathbf{T}}) = \mathbf{D}^\top \hat{\mathbf{T}} + D_0$.
23: for all $\{i : p_{(i)} \in \mathscr{H}_3\}$ do
24:     Set $\mathbf{D} \leftarrow \mathbf{D} - \frac{1}{k!} S_{p_{(i-1)}}(\mathbf{y}_1)\mathbf{d}_{p_{(i-1)}} + \frac{1}{k!} S_{p_{(i-1)}}(\mathbf{y}_{np})\mathbf{d}_{p_{(i-1)}}$,
25:     Set $D_0 \leftarrow D_0 - \frac{1}{k!} S_{p_{(i-1)}}(\mathbf{y}_1) d_{0p_{(i-1)}} + \frac{1}{k!} S_{p_{(i-1)}}(\mathbf{y}_{np}) d_{0p_{(i-1)}}$.
26:     Set $\mathbf{t} \leftarrow L \cap p_{(i)}$.
27:     Evaluate $O(\mathbf{t}) = \mathbf{D}^\top \mathbf{t} + D_0$
28:     if $O(\mathbf{t}) < O(\hat{\mathbf{T}})$ then
29:         Set $\hat{\mathbf{T}} \leftarrow \mathbf{t}$ and $O(\hat{\mathbf{T}}) \leftarrow O(\mathbf{t})$.
30:     end if
31: end for
32: return $\hat{\mathbf{T}}$.
```

5 Numerical experience and conclusions

The new algorithm was implemented along the lines of the R-package *OjaNP* of Fischer et al. (2010). A function `ojaMedianExB` was implemented to be used in place of the previous `ojaMedianEx` by ROO. A parameter `alg="exact_bounded"` was added to the function `ojaMedian`, and the corresponding C++ routines were modified. The codes may be found on the second author's page www.cgs.uni-koeln.de/pokotylo.html. The benchmark values were measured inside the C++ routines, using the file logging. This allows to easily compare the efficiency of the original and modified algorithms, excluding the data transformation and hyperplanes generation time.

The desired volume of the bounded region was set, and the calculation time was determined for the new exact algorithm as well as for the ROO procedure. Best results were received at around 10^{-8} of the original volume in most of tried datasets. The new algorithm showed to be three to six times faster than the one by ROO.

The new algorithm is able to calculate data sets of the same size and dimension as the ROO algorithm. It is mainly restricted by the amount of RAM, as it needs to store all $\binom{n}{k}$ hyperplanes. E.g. the calculation of the median in a data set of size 5×100 needs 12 GB RAM. A PC with a 3.4 GHz processor and 32 GB RAM was employed in the experimental studies. Only one processor core was used. The algorithm was able to find the median in data sets of sizes 3×750, 4×150, 5×75, 6×50 in less than half an hour, and of sizes 4×200, 5×100 in less than an hour.

In constructing the bounded regions we have tried the different variants proposed in section 3. As it was observed, all proposed approaches (B, C, D) that use the subgradient's ending point or direction to define the next cutting point converge extremely slow, compared to the simple divisive approach (A). Although the subgradients may sometimes produce really good cutting points, which strongly reduce the bounded region, they often stick at the angles of the bounded region, so that the next steps reduce the bounded region by a narrow slice only, which is close to an existing bound. Particularly in higher dimensions, the subgradients also appear to be too short, so that the amount of the volume cut in each step becomes unsatisfying. Therefore in out search we desist from the lengths of the subgradients and use only their directions. All the numerical results provided in this paper were received using the divisive approach starting with the initial rectangular bounded region. The number of cuts needed to obtain the desired volume appears to depend only moderately on the size and dimensionality of the data.

For both algorithms, the ROO and the new one, the performance of the searching procedure strictly depends on the selected initial line. As ROO select this line at random, their calculation times differ significantly between different launches. Our bounding algorithm selects the firstly found border line of the bounded region as the initial line, which makes the searching path completely deterministic, although in general not the fastest possible.

Employing bounds considerably decreases the complexity of the algorithm. The minimization along a line produces most of the complexity of the ROO algorithm. The line is intersected with $H = \binom{n}{k}$ hyperplanes, the intersections are sorted and all of them iterated, which has a complexity of $O(H^2 \log H)$. The bounding algorithm leaves a smaller amount $h < H$ of hyperplanes. Only b hyperplanes, $b < h$, that have intersections between the bounds remain to be considered. The rougher bound also strongly decreases the number of hyperplanes which need to be sorted to $s : b < s < h$. This provides a complexity of $O(b \times h \log s)$ only.

Tables 1, 2 and 3 exhibit a few exemplary results. The experimental data is an even mixture of two multidimensional normal distributions $N(\mathbf{0}_k, diag(\mathbf{1}_k))$ and $N([15,0,...,0]_k, diag([1,25,1,...,1]_k))$ although the conclusions are the same for the data having other form and for the real data sets. They show how the performance parameters listed below depend on the data dimension and size, given the intended volume equal to 10^{-8} (for Tables 1 and 2), where *#Cuts* is the number of cuts needed to reach the intended volume using the divisive approach, *H. planes (%)* is the percent of the hyperplanes intersecting the bounded region, *#Steps* is the number of minimization steps needed to find the median, and time periods needed to: determine the bounded region T_{bounds}, calculate the median (after the bounded region is determined) T_{count}, perform the whole procedure T_{total}, and to find the median using the algorithm by ROO $T_{original}$. The given times do not include the generation of all observation hyperplanes, which is the same for both algorithms.

Table 1 The performance parameters for $n \in \{50, 75\}$ and intended volume 10^{-8}.

k	n	#Cuts	H. planes (%)	#Steps	$\mathrm{T_{bounds}}$	$\mathrm{T_{count}}$	$\mathrm{T_{total}}$	$\mathrm{T_{original}}$
2	50	29	0.16	3	0.009	0.001	0.010	0.018
3	50	39	0.64	9	0.216	0.053	0.269	0.557
4	50	42	3.33	34	3.015	2.433	5.448	25.529
5	50	42	9.65	45	31.359	42.876	74.235	476.600
6	50	45	17.65	77	345.128	774.382	1119.510	3149.010
2	75	32	0.11	2	0.023	0.002	0.025	0.038
3	75	36	0.34	14	0.658	0.243	0.901	3.033
4	75	42	2.11	39	15.353	13.720	29.073	110.291
5	75	45	7.17	70	281.888	474.461	756.349	2667.890

Table 2 The performance parameters for $k \in \{4, 5\}$ and intended volume 10^{-8}.

k	n	#Cuts	H. planes (%)	#Steps	$\mathrm{T_{bounds}}$	$\mathrm{T_{count}}$	$\mathrm{T_{total}}$	$\mathrm{T_{original}}$
4	25	38	3.26	25	0.159	0.095	0.254	0.707
4	50	42	3.33	34	3.015	2.433	5.448	25.529
4	75	42	2.11	39	15.353	13.720	29.073	110.291
4	100	43	2.60	35	49.360	41.691	91.051	338.950
5	25	44	11.77	39	0.930	0.932	1.862	6.171
5	50	42	9.65	45	31.359	42.876	74.235	476.600
5	75	45	7.17	70	281.888	474.461	756.349	2667.890
5	100	43	8.53	71	1166.930	2220.330	3387.260	9803.730

The part of the hyperplanes crossing the bounded region of the given volume grows quickly with dimension, as it is seen in Tables 1 and 3. On the other hand, the part of these hyperplanes that take part in the minimization process decreases, since many of their intersections with a searching line lie outside the bounded region. Note that the bounded region, being located in the middle of the data cloud, is

Table 3 The performance parameters for data sets 4×100 and 6×50, with different intended volumes. Volume equal to one corresponds to the ROO algorithm.

k	n	Volume	#Cuts	H. planes (%)	#Steps	T_{bounds}	T_{count}	T_{total}
4	100	1			36			338.950
4	100	10^{-02}	14	67.45	50	17.648	283.936	301.584
4	100	10^{-03}	18	46.62	49	23.216	201.245	224.461
4	100	10^{-04}	22	28.47	39	28.304	112.429	140.733
4	100	10^{-05}	27	14.23	48	33.403	97.364	130.767
4	100	10^{-06}	31	9.76	29	37.707	49.499	87.206
4	100	10^{-07}	37	5.47	38	44.391	55.032	99.423
4	100	10^{-08}	43	2.60	35	49.360	41.691	91.051
4	100	10^{-09}	47	1.64	33	57.209	37.402	94.611
4	100	10^{-10}	52	0.83	33	59.709	35.347	95.056
4	100	10^{-20}	97	<0.01	22	108.205	21.089	129.294
4	100	10^{-30}	100	<0.01	13	109.004	12.778	121.782
6	50	1			73			3149.010
6	50	10^{-05}	31	53.86	91	215.747	1842.733	2058.480
6	50	10^{-06}	36	35.61	71	270.619	1272.041	1542.660
6	50	10^{-07}	40	26.68	75	301.413	885.707	1187.120
6	50	10^{-08}	45	17.65	77	345.128	774.382	1119.510
6	50	10^{-09}	49	12.97	70	373.170	545.055	918.225
6	50	10^{-10}	53	9.14	94	378.424	705.386	1083.810

intersected by most of the hyperplanes, so that the part of the included hyperplanes is much larger than the part of the final volume, compared to the initial one. Our calculations demonstrate that the part of included hyperplanes strongly depends on the dimensionality and the number of observations, which is also shown in Fig.6. However, the number of observations has less influence than the dimension.

These three tables also show that the new exact bounding algorithm finds the median much faster than the one of ROO. We observe that for each given data set the number of necessary minimization steps is almost the same in both algorithms. As the intended volume is reduced, the time needed to build the bounded region increases, while the minimization time decreases along with the number of hyperplanes and their intersections involved, and the total time also decreases (table 3, Fig. 7). However, beyond some point, usually at around 10^{-08} of the volume, this procedure becomes less efficient, and the total time increases. A smaller volume may also contain a higher amount of isolated routes through the observation lines, which involves travelling along the bounds and additionally slows the procedure down.

If the volume is small enough, any point of it (e.g. the average of the bounds' intersections) may be taken as an approximate value of a median. For example, for a four-dimensional dataset bounded by a cube of side length 10, 10^{-8} of the volume was reached in 43 cuts, and the center of the final bounded region equalled the median ± 0.05 by each coordinate, which is quite precise. The precision of this approximative method depends on the volume of the bounded region and is controlled

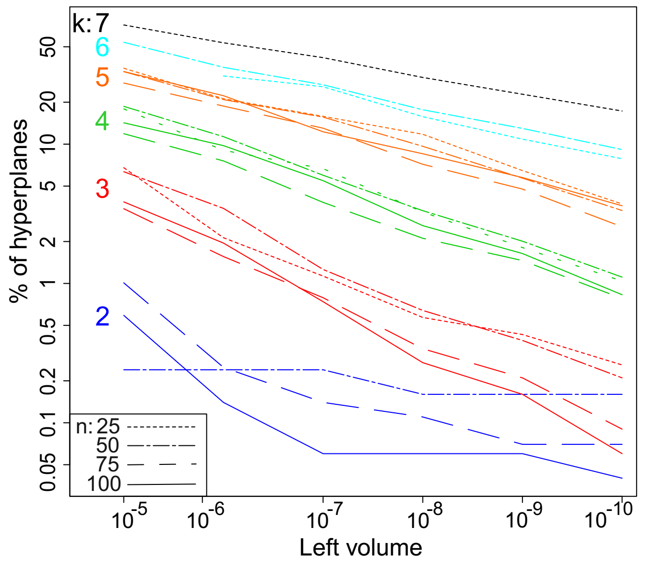

Fig. 6 The dependence of the part of hyperplanes crossing the bounded region (log scale) on the size of the region for $k \in [2..7]$ and $n \in \{25, 50, 75, 100\}$.

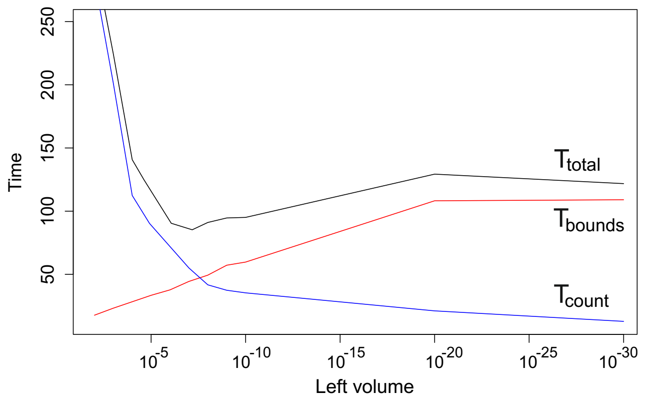

Fig. 7 The dependence of calculation time on the size of the bounded region. Note that the ROO algorithm has total time of ca. 340 seconds.

by it. The method yields as precise results as the approximative methods provided in the *OjaNP* R-package. In general, the approximate value of the median is found much faster than the exact one. We searched an approximative median with precision equal to half of the bounded region's volume, the computation times of which are given in the column T_{bounds} of Table 3. However, this approximation method is not really useful, as it considers all observation hyperplanes and is therefore largely outperformed by the approximative methods of ROO.

References

Fischer, D., Möttönen, J., Nordhausen, K., Vogel, D.: OjaNP: Multivariate Methods Based on the Oja Median and Related Concepts. R package version 0.9-8 (2014), http://cran.r-project.org/web/packages/OjaNP/ .

Hettmansperger, T. P., Möttönen, J., Oja, H.: Affine-invariant multivariate one-sample signed-rank tests. Journal of the American Statistical Association, 92, pp. 1591–1600 (1997)

Hettmansperger, T. P., Möttönen, J., Oja, H.: The geometry of the affine invariant multivariate sign and rank methods. Journal of Nonparametric Statistics, 11, pp. 271–285 (1999)

Koshevoy, G.: Lift-zonoid and multivariate depths. In: R.Dutter (ed.) Developments in Robust Statistics, pp.194–202. Springer (2003)

Liu, R.Y.: On a notion of simplicial depth. Proceedings of the National Academy of Sciences 85, pp. 1732–1734 (1988)

Mosler, K.: Depth statistics. In: C. Becker, R. Fried, S. Kuhnt , eds., Robustness and Complex Data Structures, Festschrift in Honour of Ursula Gather, pp. 17–34. Berlin (Springer) 2013

Niinimaa, A., Oja, H. and Nyblom, J.: Algorithm AS 277. The Oja bivariate median. Applied Statistics 41, pp. 611–617 (1992)

Niinimaa, A., Oja, H. and Tableman, M.: The finite-sample breakdown point of the Oja bivariate median and of the corresponding half-samples version. Statistics & Probability Letters, 10, pp. 325–328 (1990)

Oja, H.: Descriptive statistics for multivariate distributions. Statistics & Probability Letters 1, pp. 327–332 (1983)

Oja, H.: Affine invariant multivariate sign and rank tests and corresponding estimates: a review. Scandinavian Journal of Statistics, 26:3, pp. 319–343. Wiley Online Library (1999)

Oja, H.: Multivariate Median. In: C. Becker, R. Fried, S. Kuhnt , eds., Robustness and Complex Data Structures, Festschrift in Honour of Ursula Gather, pp. 3–15. Springer (2013)

Ronkainen, T., Oja, H., Orponen, P.: Computation of the multivariate Oja median. In: R.Dutter (ed.) Developments in robust statistics, pp. 344–359. Springer (2003)

Small, C.G.: Multidimensional medians arising from goedesics on graphs. Annals of Statistics 25, pp. 478–494 (1997)

Tukey, J.W.: Mathematics and picturing data. In: R.D. James (ed.) Proceedings of the 1974 International Congress of Mathematicians, Vancouver, pp. 523–531 (1975)

Zuo, Y., Serfling, R.: General notions of statistical depth function. Annals of Statistics 28, pp. 461–482 (2000)