Bazovkin, Pavel; Mosler, Karl

Working Paper
An exact algorithm for weighted-mean trimmed regions in any dimension

Discussion Papers in Statistics and Econometrics, No. 6/10

Provided in Cooperation with:
University of Cologne, Institute of Econometrics and Statistics

Suggested Citation: Bazovkin, Pavel; Mosler, Karl (2010) : An exact algorithm for weighted-mean trimmed regions in any dimension, Discussion Papers in Statistics and Econometrics, No. 6/10, University of Cologne, Seminar of Economic and Social Statistics, Cologne

This Version is available at:
http://hdl.handle.net/10419/45356

Terms of use:
Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.
An Exact Algorithm for Weighted-Mean Trimmed Regions in Any Dimension

by

Pavel Bazovkin
Karl Mosler

1st version
October 15, 2010
An Exact Algorithm for Weighted-Mean Trimmed Regions in Any Dimension

by

Pavel Bazovkin
Karl Mosler

Abstract

Trimmed regions are a powerful tool of multivariate data analysis. They describe a probability distribution in Euclidean $d$-space regarding location, dispersion, and shape, and they order multivariate data with respect to their centrality. Dyckerhoff and Mosler (201x) have introduced the class of weighted-mean trimmed regions, which possess attractive properties regarding continuity, subadditivity, and monotonicity.

We present an exact algorithm to compute the weighted-mean trimmed regions of a given data cloud in arbitrary dimension $d$. These trimmed regions are convex polytopes in $\mathbb{R}^d$. To calculate them, the algorithm builds on methods from computational geometry. A characterization of a region’s facets is used, and information about the adjacency of the facets is extracted from the data. A key problem consists in ordering the facets. It is solved by the introduction of a tree-based order. The algorithm has been programmed in C++ and is available as an R package.

Keywords: Central regions, data depth, multivariate data analysis, convex polytope, computational geometry, algorithm, C++, R.
An Exact Algorithm for Weighted-Mean Trimmed Regions in Any Dimension

Pavel Bazovkin
Universität zu Köln

Karl Mosler
Universität zu Köln

October 15, 2010

Abstract

Trimmed regions are a powerful tool of multivariate data analysis. They describe a probability distribution in Euclidean \( d \)-space regarding location, dispersion, and shape, and they order multivariate data with respect to their centrality. Dyckerhoff and Mosler (201x) have introduced the class of weighted-mean trimmed regions, which possess attractive properties regarding continuity, subadditivity, and monotonicity.

We present an exact algorithm to compute the weighted-mean trimmed regions of a given data cloud in arbitrary dimension \( d \). These trimmed regions are convex polytopes in \( \mathbb{R}^d \). To calculate them, the algorithm builds on methods from computational geometry. A characterization of a region’s facets is used, and information about the adjacency of the facets is extracted from the data. A key problem consists in ordering the facets. It is solved by the introduction of a tree-based order. The algorithm has been programmed in C++ and is available as an \( \text{R} \) package.

Keywords: central regions, data depth, multivariate data analysis, convex polytope, computational geometry, algorithm, C++, \( \text{R} \).

1. Introduction

Trimmed regions are a powerful tool of multivariate data analysis. They describe a probability distribution in Euclidean \( d \)-space regarding location, dispersion, and shape, and they order multivariate data with respect to their centrality. Given \( d \)-variate data \( x_1, x_2, \ldots, x_n \), an \( \alpha \)-trimmed region \( D_\alpha(x_1, x_2, \ldots, x_n) \) is a convex compact set in \( \mathbb{R}^d \) that depends on the data in an affine equivariant way, i.e., for every matrix \( A \in \mathbb{R}^{m \times d} \) and every \( b \in \mathbb{R}^m \) it holds

\[
D_\alpha(Ax_1 + b, \ldots, Ax_n + b) = AD_\alpha(x_1, \ldots, x_n) + b.
\]

The parameter \( \alpha \) varies in an interval such that the family \( (D_\alpha(x_1, x_2, \ldots, x_n))_\alpha \) is nested decreasing in \( \alpha \), i.e., \( \alpha < \beta \) implies \( D_\beta(x_1, \ldots, x_n) \subset D_\alpha(x_1, \ldots, x_n) \). The smallest region is regarded as a particular median of the data.

Several special notions of trimmed regions have been introduced in the literature, among them the Mahalanobis regions, the halfspace regions, and the zonoid regions; for recent surveys, see Serfling (2006), Cascos (2009). Applications include multivariate data analysis (Liu, Parelius, and Singh 1999), classification (Mosler and Hoberg 2006), tests for multivariate location and
scale (Dyckerhoff 2002), risk measurement (Cascos and Molchanov 2007), and many others. The various notions of trimmed regions differ in properties like continuity, robustness, and sensitivity regarding the data. Depending on the type of application different properties are relevant. E.g., Mahalanobis regions are ellipses around the mean of the data and based on their covariance matrix; by this they can neither reflect a possible asymmetry of the distribution nor characterize it in a unique way. Both halfspace regions and zonoid regions reflect asymmetries and characterize the distribution. Halfspace regions are more robust against outliers than zonoid regions; if robustness is an issue, the latter need some preprocessing of the data.

Dyckerhoff and Mosler (201x) have introduced the class of weighted-mean trimmed regions, which possess additional attractive properties and include the zonoid regions as a special case. Weighted-mean trimmed regions are continuous in the data as well as in the parameter, which means that both the function \((x_1, \ldots, x_n) \mapsto D_\alpha(x_1, \ldots, x_n)\) and the function \(\alpha \mapsto D_\alpha(x_1, \ldots, x_n)\) are continuous in terms of Hausdorff convergence of sets. Moreover, weighted-mean trimmed regions are subadditive and monotone, which properties have a substantial interpretation in terms of \(d\)-variate risk and allow the construction of set-valued risk measures that are coherent (Cascos and Molchanov 2007).

To be useful in data applications, a notion of trimmed regions must be computable. Bivariate trimmed regions of any type can be calculated by constructing a circular sequence, like in Dyckerhoff (2000) and Cascos (2007), but only a few procedures are known in dimension \(d > 2\). Mahalanobis regions are easily determined in any dimension \(d\), as they only employ the mean and the dispersion matrix of the data. Mosler, Lange, and Bazovkin (2009) develop an efficient geometric algorithm for zonoid regions in any dimension, and Hallin, Paindaveine, and Šiman (2010) provide a parametric linear program for calculating halfspace regions.

In this paper we present an exact algorithm to compute the weighted-mean trimmed regions of a given data cloud in arbitrary dimension \(d\). These trimmed regions are convex polytopes in \(\mathbb{R}^d\). To calculate them, the algorithm builds on methods from computational geometry. A region’s facet is characterized by \(d - 1\) pairs of data points. Based on them the normal (support vector) of the facet is determined and by properly rotating the support vector an adjacent facet is found. A key problem consists in ordering the facets. It is solved by the introduction of a tree-based order.

Overview of the paper: Section 2 provides a brief introduction into the notion of weighted-mean trimmed (WMT) regions. The main results of the paper are contained in Section 3, which presents the basic geometrical ideas of the algorithm, in particular the construction of a facet on the basis of \(d - 1\) data point differences and the transition to a neighboring facet by rotating the support vector and exchanging the basis. Section 4 provides a formal description of the algorithm. Section 5 delineates the R package and the program realization in C++, while Section 6 concludes with a discussion of complexity issues.

2. Weighted-mean trimming

This section reviews the general notion of weighted-mean trimmed regions and two of its special cases, the zonoid regions and a modified version of the expected convex hull (ECH) regions - the ECH\(^*\) regions.
2.1. Definition and principal properties

Weighted-mean trimmed regions are convex bodies in $\mathbb{R}^d$. Recall that a convex body $K \subset \mathbb{R}^d$ is uniquely determined by its support function $h_K$ (see, e.g., Rockafellar (1970)),

$$h_K(p) = \max \{ p'x \mid x \in K \}, \quad p \in S^{d-1},$$

where $S^{d-1}$ denotes the unit sphere in $\mathbb{R}^d$.

To define the weighted-mean $\alpha$-trimmed region of a given data cloud $x_1, x_2, \ldots, x_n$, we construct its support function as follows: For $p \in S^{d-1}$, consider the subspace $\{ \lambda p \mid \lambda \in \mathbb{R} \}$. By projecting the data on this subspace a linear ordering is obtained,

$$p'x_{\pi_p(1)} \leq p'x_{\pi_p(2)} \leq \cdots \leq p'x_{\pi_p(n)},$$

and, by this, a permutation $\pi_p$ of the indices $1, 2, \ldots, n$. Note that, if no equalities arise in (1), the permutation $\pi_p$ is unique, otherwise a class $\Pi_p$ of several permutations is generated.

The set of directions $p$ at which $\pi_p$ is not unique will be denoted $H(x_1, \ldots, x_n)$,

$$H(x_1, \ldots, x_n) = \left\{ p \in S^{d-1} \mid \text{there are } i \neq j \text{ such that } p'x_i = p'x_j \right\}.$$

Consider weights $w_{j,\alpha}$ for $j \in \{1, 2, \ldots, n\}$ and $\alpha \in [0, 1]$ that satisfy the following restrictions (i) to (iii):

(i) $\sum_{j=1}^n w_{j,\alpha} = 1$, $w_{j,\alpha} \geq 0$ for all $j$ and $\alpha$,

(ii) $w_{j,\alpha}$ increases in $j$ for all $\alpha$,

(iii) if $\alpha < \beta$ then

$$\sum_{j=1}^k w_{j,\alpha} \leq \sum_{j=1}^k w_{j,\beta}, \quad k = 1, \ldots, n. \quad (2)$$

Then, as it has been shown in Dyckerhoff and Mosler (201x), the function $h_{D_\alpha(x_1, \ldots, x_n)}$,

$$h_{D_\alpha(x_1, \ldots, x_n)}(p) = \sum_{j=1}^n w_{j,\alpha} p'x_{\pi_p(j)}, \quad p \in S^{d-1} \quad (3)$$

is the support function of a convex body $D_\alpha = D_\alpha(x_1, \ldots, x_n)$, and $D_\alpha \subset D_\beta$ holds whenever $\alpha > \beta$.

Now we are ready to give the general definition of a family of weighted-mean trimmed regions.

**Definition 1.** (Dyckerhoff and Mosler) Given weights $w_{1,\alpha}, \ldots, w_{n,\alpha}$ that satisfy the restrictions (i) to (iii), the convex body $D_\alpha = D_\alpha(x_1, \ldots, x_n)$ having support function (3) is named the weighted-mean $\alpha$-trimmed region of $x_1, \ldots, x_n$, $\alpha \in [0, 1]$.

The next proposition explains the name by stating that a weighted-mean trimmed region is the convex hull of weighted means of the data. Further it describes the region’s extreme points.
An Exact Algorithm for WMT Regions

**Proposition 1.** It holds

\[ D_\alpha(x_1, \ldots, x_n) = \text{conv}\left\{ \sum_{j=1}^{n} w_{j,\alpha} x_{\pi(j)} \mid \pi \text{ permutation of } \{1, \ldots, n\} \right\}, \]

and the set of extreme points of \( D_\alpha \) is given by

\[ \text{Ext}(D_\alpha(x_1, \ldots, x_n)) = \left\{ \sum_{j=1}^{n} w_{j,\alpha} x_{\pi_p(j)} \mid p \in S^{d-1} \setminus H(x_1, \ldots, x_n) \right\}. \]

Due to their attractive analytical properties, WMT regions are useful statistical tools. Besides being *continuous* in the data and in \( \alpha \), they are *subadditive*, that is,

\[ D_\alpha(x_1 + y_1, \ldots, x_n + y_n) \subset D_\alpha(x_1, \ldots, x_n) \oplus D_\alpha(y_1, \ldots, y_n), \]

and *monotone*: If \( x_i \leq y_i \) holds for all \( i \) (in the componentwise ordering of \( \mathbb{R}^d \)), then

\[ D_\alpha(y_1, \ldots, y_n) \subset D_\alpha(x_1, \ldots, x_n) \oplus \mathbb{R}^d_+, \quad \text{and} \]

\[ D_\alpha(x_1, \ldots, x_n) \subset D_\alpha(y_1, \ldots, y_n) \oplus \mathbb{R}^d_-, \]

where \( \oplus \) signifies the Minkowski sum of sets. For proofs and more results, like projection properties, the reader is again referred to Dyckerhoff and Mosler (201x).

### 2.2. Special notions of weighted-mean trimming

The general notion of WMT regions provides a flexible approach to the trimming of multivariate data. Depending on the choice of the weights \( w_{j,\alpha} \) different families of trimmed regions are obtained. They include the zonoid regions (Koshevoy and Mosler 1997), the ECH and ECH* regions (Cascos 2007), the geometrically trimmed regions, and many others. For an illustration in dimension \( d = 3 \) see Figure 1. Here the left panel shows zonoid regions for different parameters \( \alpha \), while the right one consists of ECH* regions for the same data and \( \alpha \). Note from Figure 1 that the surface of a zonoid region appears to have less facets than an ECH* region.

Historically, the first family of WMT regions was that of *zonoid trimmed regions* \( ZD_\alpha(x_1, \ldots, x_n) \) for \( 0 < \alpha \leq 1 \) proposed by Koshevoy and Mosler (1997),

\[ ZD_\alpha(x_1, \ldots, x_n) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \mid 0 \leq \lambda_i \leq \frac{1}{n\alpha}, \sum_{i=1}^{n} \lambda_i = 1 \right\}. \]

The corresponding support function is

\[ h_{ZD_\alpha}(p) = \sum_{j=1}^{n} w_{j,\alpha} p' x_{\pi_p(j)}, \]

with weights

\[ w_{j,\alpha} = \begin{cases} 0 & \text{if } j < n - \lfloor n\alpha \rfloor, \\ \frac{n\alpha - \lfloor n\alpha \rfloor}{n\alpha} & \text{if } j = n - \lfloor n\alpha \rfloor, \\ \frac{1}{n\alpha} & \text{if } j > n - \lfloor n\alpha \rfloor. \end{cases} \]
Many properties of the zonoid regions are developed in Mosler (2002); particularly important is that they contain full information about the data.

Another important notion of WMT regions is that of ECH* regions (Cascos 2007). Their support function

$$h_{ECH^*}(p) = \sum_{j=1}^{n} w_{j,\alpha} p' x_{\pi_j(j)}$$

employs the weights

$$w_{j,\alpha} = \frac{j^{1/\alpha} - (j - 1)^{1/\alpha}}{n^{1/\alpha}}.$$  \(7\)

For a detailed discussion of these and other special weighted-mean trimmed regions, like ECH and geometrically trimmed regions, the reader is referred to Dyckerhoff and Mosler (201x).

3. Geometry of the algorithm

In this section we present the basic ideas of the algorithm. Specifically, it relies on notions from convex geometry.

3.1. Trimmed region as a convex polytope

Consider a data cloud, which is a finite set of data points, \(\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d\), and assume that the points are all different and in general position (i.e., no more than \(d - 1\) of them lie on the same hyperplane).

For given \(\alpha\), the \(\alpha\)-trimmed region \(D_\alpha = D_\alpha(x_1, x_2, \ldots, x_n)\) is a convex polytope in \(\mathbb{R}^d\) that is bounded and closed. Such a polytope is the nonempty and bounded intersection of finitely many closed halfspaces. Thus the polytope can be completely described by its bounding hyperplanes. The intersection of a bounding hyperplane with the polytope is named a facet if it has the affine dimension \(d - 1\). Similarly, it is named a ridge if it has the dimension \(d - 2\). In dimension \(d = 3\) a facet is a face, and a ridge is an edge.

In the sequel, we calculate the weighted-mean trimmed regions by their facets. Two computational tasks will have to be repeatedly performed:

1. Calculate a facet,
2. find an adjacent facet.

A ridge is the intersection of two facets. Therefore, investigating the ridges is a way to extract information about the adjacency of facets. Each ridge of a given facet provides an indicator whether another facet is adjacent or not. A bounding hyperplane is fully described by its (outwards pointing) normal and one additional point, in particular one of its vertices. Hence, for every facet we determine its normal and a vertex as well as the adjacency indicator of each of its ridges. By doing this successively for all facets, a complete representation of the trimmed region is obtained.

Mosler, Lange, and Bazovkin (2009) develop an exact algorithm for calculating zonoid trimmed regions. They demonstrate that, in the case of zonoid regions, the normal of a facet is characterized by \(d\) points of the data cloud.
An Exact Algorithm for WMT Regions

Figure 1: Examples of WMT regions in $\mathbb{R}^3$. Representation of the zonoid (left) and ECH* (right) regions for the same data and depths.

Regarding a general WMT region, we will firstly characterize its facets. Let $F$ be a given
Figure 2: Characterizing the normal $p$ of a facet (zonoid region, $d = 3, n = 10, \alpha = 0.25$): Data points and their projections; $p$-ordered indices; weights; active pairs of indices.

The facet of $D_\alpha(x_1, \ldots, x_n)$ and $p$ denote its normal. Then $F$ has at least $d$ vertices, which all are supported by $p$. Due to (3) and (4) each vertex $v$ has the form

$$v = \sum_{j=1}^{n} w_{j,\alpha} x_{\pi_p(j)} \text{ with some } \pi_p \in \Pi_p.$$  \hfill (8)

Consequently, not all $p'x_i$ can be different: It holds $p \in H(x_1, \ldots, x_n)$, and $\Pi_p$ has at least $d$ elements. Now let us consider the $p$-ordered series of indices

$$\pi_p(1), \pi_p(2), \ldots, \pi_p(n).$$

In the sequel we will mention those pairs of indices $(\pi_p(j), \pi_p(k))$ as active that satisfy the equation $p'x_{\pi_p(k)} = p'x_{\pi_p(j)}$ plus a restriction on their weights $w_j$ and $w_k$, which will be specified below. The equation means that the difference $x_{\pi_p(k)} - x_{\pi_p(j)}$ is orthogonal to $p$,

$$x_{\pi_p(k)} - x_{\pi_p(j)} \perp p.$$  \hfill (9)

At a given $p$, all indices that belong to an active pair will be mentioned as active indices, all others as passive indices.

From now on, we will distinguish data points and data vectors. By a data vector we mean the difference of two data points. To determine $p$, $d - 1$ data vectors are needed. Each of them is based on an active pair of indices and thus satisfies the orthogonality relation (9). As, by assumption, the data are in general position, any such $d - 1$ data vectors are linearly independent. They will be mentioned as a basis of $F$ and denoted by $V_F$. Note that the basis of a facet is not unique: To form a basis, out of all active pairs of indices any $d - 1$ pairs that yield linearly independent data vectors may be chosen. To summarize:
The normal of a facet $F$ is orthogonal to exactly $d - 1$ linearly independent data vectors, which form a basis of $F$. The facet is characterized by a basis and one of its vertices.

Next we develop the two essential steps of calculating a facet and finding an adjacent facet in detail.

**Task 1: Calculating a facet**

In our algorithm we have to construct a basis for each facet of the polytope. Let $p$ be the normal of a given facet $F$, choose some $\pi_p \in \Pi_p$, and consider the series of $p$-ordered indices $\pi_p(1), \pi_p(2), \ldots, \pi_p(n)$. This series contains $d - 1$ active pairs of indices, $\pi_p(j), \pi_p(k)$, that define a basis $V_F$.

The special case of zonoid regions (having weights (6)) appears to be particularly simple: A facet is identified by exactly $d$ data points (carrying serially $p$-ordered indices), which yield $d - 1$ linearly independent difference vectors that are orthogonal to $p$ (Mosler et al. 2009). As an example, Figure 2 exhibits ten points in $\mathbb{R}^3$ and their projections to the line generated by $p$. The lower panel contains the $p$-ordered series of indices and the weights (6) for $\alpha = 0.25$. Here, three indices (9, 8, and 4) are active, as well as three index pairs $((9, 8), (9, 4)$, and $(8, 4))$. A basis of the facet is given, e.g., by the data vectors $x_8 - x_9$ and $x_4 - x_8$. Note that for these weights (at $\alpha = 0.25$) and any direction $p$ the indices $\pi_p(7), \pi_p(8)$, and $\pi_p(9)$ become the active ones.

Other types of weighted-mean trimmed regions employ less simple weights. With them the number of active indices involved in the identification of a facet $F$ may be larger than $d$. E.g., Figure 3 illustrates the characterization of a facet of an ECH$^*$ region, with weights (7) and $\alpha = 0.25$. It shows another example of ten points in $\mathbb{R}^3$ and their projections, given some $p$. In this example, four indices (7, 6, 4, and 2), and two index pairs ((7, 6) and (4, 2)) are active, and a basis consists of $x_6 - x_9$ and $x_2 - x_4$, being unique up to sign.

In general, we consider the following disjoint blocks $\mathcal{A}_l$ of indices, $l = 1, \ldots, L$,

$$\mathcal{A}_l = \{\pi_p(i) | i \in \{a_l, a_l + 1, \ldots, a_l + n_l - 1\}, \ p'x_{\pi_p(i)} = p'x_{\pi_p(i-1)} \text{ for } i > a_l\},$$

where $a_{l-1} < a_l$ holds ($a_0 = 0$), and define: A pair of indices is called **active** if a block $\mathcal{A}_l$ exists that contains both of them. In particular, each block contains at least two elements, $n_l \geq 2$, and it holds $w_{a_l,a} < w_{a_l+n_l-1,a}$, which is the restriction on weights announced above. Moreover, $\mathcal{A}_l \cap \mathcal{A}_m = \emptyset$ if $l \neq m$, and

$$\mathcal{V}_F = \bigcup_{l=1}^{L} \{x_{\pi_p(i)} - x_{\pi_p(i+1)} | \pi_p(i), \pi_p(i + 1) \in \mathcal{A}_l\}.$$

Note that in the case of zonoid regions only one block of active indices arises; it holds $L = 1$. The remaining indices, which are not in $\bigcup_{l=1}^{L} \mathcal{A}_l$, are the **passive** ones. Among them we distinguish disjoint blocks that have equal weights,

$$\mathcal{B}_k = \{\pi_p(i) | i \in \{b_k, b_k + 1, \ldots, b_k + m_k - 1\}, \ w_{i-1,a} = w_{i,a} \text{ for } i > b_k\},$$

$k = 1, 2, \ldots, K$, while $m_k \geq 1$, $b_{k-1} < b_k$ with $b_0 = 0$, and $w_{b_{k-1},a} < w_{b_k,a}$.
Figure 3: Characterizing the normal $p$ of a facet ($ECH^*$ region, $d = 3, n = 10, \alpha = 0.25$): Data points and their projections; $p$-ordered indices; weights; active pairs of indices.

Thus $\pi_p(1), \pi_p(2), \ldots, \pi_p(n)$ divides into a series $S_F$ of blocks

Figure 4: Series of blocks of active and passive indices; weights as indicated.
Task 2: Finding an adjacent facet

To identify adjacent facets we start from a given facet $F$, which has support vector $p$ and which from now on will be called the current facet. Each ridge of $F$ offers a way of "jumping" to a neighboring facet. Therefore we investigate the ridges of the current facet $F$ and, consequently, its adjacent facets. Each element of the basis $\mathcal{V}_F$ may be regarded as a reduction of one degree of freedom of the support vector $p$. To determine $p$ as the normal of the current facet $F$, we have to reduce $d - 1$ degrees of freedom and calculate the uniquely determined support vector $p$ that is orthogonal to $d - 1$ linearly independent data vectors (differences of vectors from the original data cloud). A ridge of the current facet is supported by vectors that result from adding one degree of freedom to the given support vector $p$. The degree of freedom is added by leaving out one of the $d - 1$ data vectors from the basis $\mathcal{V}_F$, or, more generally, by replacing some $k$ data vectors in $\mathcal{V}_F$ with some $k - 1$ ones, while keeping linear independence within the basis.

Removing one element from the basis $\mathcal{V}_F$ corresponds to splitting one of the active blocks in $\mathcal{S}_F$, say $A_l$, into $A_1^l$ and $A_2^l$. By this, a modified series of blocks, $\mathcal{S}_{F_*}$, is obtained. Observe that, if $A_1^l$ (or $A_2^l$) is a singleton, its element becomes a passive index in $\mathcal{S}_{F_*}$.

Now, the removed element of the basis has to be substituted by another data vector. For this, any pair $(i^*, j^*)$ of indices that belong to two neighboring blocks of $\mathcal{S}_{F_*}$ can be chosen and the corresponding data vector $x_{i^*} - x_{j^*}$ be added to the basis. (However, no pair from $A_1^l \times A_2^l$ must be selected.) Then the new basis defines a facet that is adjacent to the current facet $F$.

This step may be visualized as follows (see Figure 5 for $d = 3$): Starting at $p$, the support vector is rotated in a plane (of dimension two in $\mathbb{R}^d$) until another data vector enters the basis $\mathcal{V}_F$, i.e., until another data vector becomes orthogonal to $p$. Let $\mathcal{E}_p$ denote the set of vertices of the polytope corresponding to the support vector $p$. We turn $p$ until it stops at the position $\tilde{p}$ where $\tilde{p} x_{i^*} = \tilde{p} x_{j^*}$ for some $i^*$ and $j^*$, i.e., $(x_{i^*} - x_{j^*}) \perp \tilde{p}$. Then, if $\mathcal{E}_{\tilde{p}} \supset \mathcal{E}_p$, this means that $\tilde{p}$ is a normal to some facet $\tilde{F}$ which is a neighbor to the current facet. Otherwise, $p$ is turned further until the condition is met. Obviously, to meet the condition, the indices $i^*$ and $j^*$ must be in different blocks of $\mathcal{S}_{F_*}$. On the other hand, indices can continuously interchange places only with their neighbors, that is, $x_{i^*}$ and $x_{j^*}$ must be in blocks that neighbor each other.

So far we have exchanged a single basis vector against another one. However, the elements within each active block at $p$ can be arbitrarily rearranged, and each active index used in the exchange step just represents a class of equivalent active indices. Therefore more than one, say $k$, active pairs living on $A_1^l \times A_2^l$ may be exchanged simultaneously.

As a result of the basis exchange we have found a single adjacent facet. Our next task is to identify all facets that are adjacent to the current facet. For this, it is not necessary to enumerate all pairs of indices from neighboring blocks of $\mathcal{S}_{F_*}$. Note that the elements of each active block $A_l$ are equivalent in the $p$-order, i.e., $p^* x_{i^*} = p^* x_{j^*}$ for all $i^*, j^* \in A_l$.

Hence, we may permute indices within the active blocks in an arbitrary way, which means employing some other permutation from $\Pi_p$ in place of the given permutation $\pi_p$. Therefore, in generating all possible basis exchanges, we need not consider all active indices for pairing, but may restrict to a representative index of each active block, say $r_l \in A_l$, $l = 1, \ldots, L$. However, in the passive blocks, all indices have to be taken into account.

A pair $(i^*, j^*)$ from two neighboring blocks in $\mathcal{S}_{F_*}$ is called a critical pair if it consists of
indices that are either passive or representative active indices. More formally, we may write the series $S_{F*}$ of active and passive blocks as

$$S_{F*} = (C_1, C_2, \ldots, C_{L+K})$$

and define

$$\tilde{C}_m = \begin{cases} \{r_l\} & \text{if } C_m = A_l \text{ for some } l, \\ B_k & \text{if } C_m = B_k \text{ for some } k. \end{cases}$$

Then the set of critical pairs (that have to be checked for finding all adjacent facets) is given by

$$\bigcup_{m=1}^{L+M-1} \tilde{C}_m \times \tilde{C}_{m+1}. \quad (10)$$

The two computational tasks, calculating a facet and finding a neighboring facet, are performed until all facets of the polytope have been visited and computed. As a result of the algorithm, the WMT region is completely described by its facets. Alternatively and in addition, we may be interested in calculating vertices of the polytope. These are easily determined by the following procedure.

**Proposition 2. (Calculating vertices)** Consider a facet $F$ having normal $p$. Each vertex of $F$ exactly corresponds to a permutation of $(\pi_p(1), \ldots, \pi_p(n))$ that is restricted to permutations within the $A_l$.

**Corollary 1.** The minimum possible number of vertices of a facet is $d$ (e.g., for zonoid regions). The maximum possible number of vertices of a facet is $d!$.

E.g., in the case of ECH*-regions, the number of vertices of a facet varies from $d$ to $d!$. 

---

*Figure 5: Rotating $p$ in a plane of dimension two in $\mathbb{R}^d$.***
3.2. Spanning tree order

Based on the adjacency information obtained by the above procedure we are able to calculate the facets in a sequential order. For this sequence, we use the spanning tree order (STO) discussed in Mosler et al. (2009). The STO provides a complete ordering of all facets according to which they are generated in the algorithm. The general idea is:

1. Represent all facets adjacency information by a tree,
2. organize an efficient search procedure to traverse the tree.

In the algorithm we organize a breadth-first search as, e.g., described in Knuth (1997). Using the STO we generate each facet only once, which is an efficient procedure.

Moreover, as the STO is based on the neighboring relation among facets, we can restrict the calculation of facets to some connected part of the trimmed region’s surface, e.g. the part having support vector \( p \geq 0 \). This proves to be useful in certain applications like multivariate risk measurement.

Note that we calculate the trimmed region by sequentially generating its facets, but not its vertices. In dimension \( d = 2 \) it is also possible to determine the region by enumerating its vertices; this is done by means of a so called circular sequence (Edelsbrunner 1987).

It is easy to see that the proposed procedure applies to any choice of a weighting function satisfying the above WMT restrictions (i) to (iii). Thus the algorithm is able to calculate any weighted-mean trimmed region.

4. The algorithm

Input
Figure 7: The sample scheme of the procedure.

\(d\) (dimension of the data space, \(d \geq 2\));

\(n\) (number of data points, \(n > d\));

*cloud* (data \(x_1, \ldots, x_n \in \mathbb{R}^d\));

\(\alpha\) (depth parameter);

\(w_\alpha\) (weight vector; alternatively: name of special type of WMT regions).

**Output**

- *trimmed_region* (all facets of the trimmed region, with coordinates of their vertices);
- *Visualization*.

**Steps of the Algorithm**

A. Initialization:

a. Read the input.

b. Calculate \(\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i\).

c. Substitute \(x_i - \bar{x}\) for \(x_i\), \(i = 1, \ldots, n\).

B. Determine a first facet:
a. From *cloud*, form a set vec\_defining\_set of $d - 1$ linearly independent data vectors (= basis).

b. Calculate, to the hyperplane through vec\_defining\_set, a normal vector $r$.

c. Substitute $r$ for $p$ and choose a permutation $\pi_p \in \Pi_p$. Determine the series of active blocks $\{A_l\}_{l=1 \ldots L}$ in this permutation.

d. $\{A_l\}_{l=1 \ldots L}$ defines vec\_defining\_set and, hence, the first facet $ffacet$.

e. Place $ffacet \hookrightarrow$ the head of queue.

C. Determine all facets:

a. Take curr\_facet $\hookleftarrow$ front of the queue.

b. Create neighboring facets of curr\_facet.

I. Create all ridges by adding a degree of freedom to $p$ (reducing cardinality of the basis vec\_defining\_set by one).

i. Take the next $A_i$ and create all possible splittings of it into two subsets: $\langle A^1_i, A^2_i \rangle$. Replace $\{A_i\}$ by $A^1_i$ and $A^2_i$. If either $A^1_i$ or $A^2_i$ is a singleton, remove it from the active blocks. A set partial\_facets($l$) is obtained.

ii. Drop off all elements of partial\_facets($l$) that are no active blocks. A set ridges($l$) is obtained.

iii. Add partial\_facets($l$) to the set ridges. If an unprocessed $A_i$ is left, go to C.(b.)I.i.

II. For the next ridge in ridges do the following:

i. Build the maximum number of linearly independent data vectors that are based on active pairs. Put the vectors as rows into a matrix $A$. There will be $d - 2$ rows.

ii. Given a normal vector $r$ to curr\_facet, put it as an additional row into $A$. Put any non-zero vector that is linearly independent of the $d - 1$ previously chosen rows as a last row into $A$. Let $b$ be a vector that consists of $d - 1$ zeros and a last non-zero entry.

iii. Solve the linear equation $Az = b$. Its solution $z$ and $r$ span a plane $B_2$ that is orthogonal to the ridge.

iv. If curr\_facet is not marked, calculate a hash code of the ridge and place it $\hookrightarrow$ hash\_table. Then go to C.(b.)II.

v. Check in hash\_table, whether the ridge is blocked. If not, go to C.(b.)II.

vi. Calculate critical pairs according to (10).

vii. Rotate $p$ in the plane $B_2$. In doing so, start at $p = r$ and move $p$ in a way that the new ordering of points in the permutation corresponds to the previous splitting of an active block.

viii. Stop if $p$ becomes orthogonal to some vector built on a critical pair of indices. Take this vector as new\_vector.

ix. Add new\_vector to vec\_defining\_set. new\_facet is obtained. The current position of $p$ is a normal $r$ to new\_facet.

If new\_vector is built on indices from an active block $A_j$ and a neighboring passive block, then augment $A_j$ with the passive index.
If new_vector is built on indices from two active blocks, $A_j$ and $A_{j+1}$, then merge these two blocks.

If new_vector is built on two passive indices, then a new block $A'_j$ is created having them as its two elements.

x. Place new_facet $\rightarrow$ the head of queue.

III. If curr_facet is not marked, mark it and place $\rightarrow$ queue. Then, go to C.a.

IV. For curr_facet, calculate the vertices and its absolute distance from the origin by (8).

V. Shift curr_facet by $\pi$ and transfer it to trimmed_region.

c. If queue is not empty, go to C.a. Otherwise, stop: Then, trimmed_region contains all facets of the trimmed region.

4.1. Some details of the algorithm

1. Using a double hash: The ridges are hashed using a "double-hash" table. That is, a ridge is blocked if it has been marked twice.

2. Calculating the hash code: The hash code is calculated by creating a bit row from integer numbers describing $\{A_l\}_{l=1...L}$ and one number describing the absolute position of a ridge (to distinguish parallel ridges).

3. Determining all adjacent facets: Given a basis $V_F$ of a facet $F$, let $A$ be a nonsingular $d \times d$ matrix that contains the basis vectors as its first $d - 1$ rows and an arbitrary last row that is linearly independent from the other rows. Consider the linear equation

$$Ar = e_d := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$ \hspace{1cm} (11)

The unique solution $r$ to this equation is a scalar multiple of the normal vector $p_F$ of $F$.

In search for a neighboring facet, the support vector has to be rotated in a plane of dimension two in $\mathbb{R}^d$ (Step C.(b.)II.iii.). To reduce the algorithmic complexity of this step we compute the rotation plane in the following efficient way. By this, the complexity of the algorithm is improved at its bottleneck.

The transition from $F$ to a neighboring facet, say via a ridge $m$, is done by a basis exchange. This means replacing some $k$ rows of the matrix $A$ (having indices $i \in I$) by $k - 1$ other data vectors and, as its last row, some vector that is linear independent from all previous rows and non-orthogonal to $p$, for example $p$ itself. Let $S_m$ denote the $d \times d$ matrix obtained from $A$ by this exchange, and $V_m$ the $d \times k$ matrix built from the $k$ new vectors as columns. Thus, the solution $z = z_m$ of the linear equation

$$S_mz = e_d,$$ \hspace{1cm} (12)

spans, together with $p_F$, a plane in which the support vector $p$ may be rotated.

Note that (12) can be solved directly by calculating $S_m^{-1}$, which is the straightforward computation and has complexity $O(d^3)$. Instead, in our algorithm we decompose $S_m$ in order to
reduce the complexity of this step. It is easy to see that

\[ S_m = K_m \cdot A, \]

where \( K_m \) is an identity matrix with substituted rows of indices \( i \in I \). Let these rows form a matrix \( C_m \) whose \( i \)-th row corresponds to the \( j_i \)-th row of \( K_m \). Then it holds \( A' C_m = V_m \) and, consequently,

\[ C_m = (A^{-1})' V_m. \]

Note that \( A^{-1} \) has to be computed only once at each facet. Given \( A^{-1} \), calculating \( C_m \) has complexity \( O(d^2) \).

Henceforth we denote the elements of \( A \) and \( C \) by \( a_{ij} \) and \( c_{ij} \) respectively. Consider rewriting (12) as

\[ K_m A z = e_d, \]

where

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & \ddots & 0 \\
\cdots & \cdots & \cdots & \cdots \\
c_{j_1} & c_{j_2} & \cdots & c_{j_d} \\
\cdots & \cdots & \cdots & \cdots \\
0 & 0 & \cdots & 1
\end{pmatrix}
\]

and

\[
K_m^{-1} e_d = \begin{pmatrix}
0 \\
\vdots \\
\frac{c_{j_d}}{c_{j_i}} \\
\vdots \\
0 \\
1
\end{pmatrix}. \]

Then it holds

\[
z_m = A^{-1} K_m^{-1} e_d = A^{-1} \begin{pmatrix}
0 \\
\vdots \\
\frac{c_{j_d}}{c_{j_i}} \\
\vdots \\
0 \\
1
\end{pmatrix}.
\]

For a single facet we have to compute (in \( O(d^3) \))

\[
A^{-1} = \begin{pmatrix}
\alpha_1 & \cdots & \alpha_i & \cdots & \alpha_d
\end{pmatrix}.
\]

Thus, besides computing \( C_m \), only the following computation is done in finding a basis for the \( m \)-th "jump":

\[
z_m = \alpha_d - \sum_{i \in I} \frac{c_{j_i}}{c_{j_i}} \alpha_i, \quad O(d).
\]

Recall that a basis for an \( m \)-th jump is given by \( \{ z_m, \bar{p} = \bar{r} \} \). Let us denote the number of ridges for a facet \( F \) by \( R(F) \). The complexity of finding bases for all jumps from the facet is

\[
O(d^3 + (d^2 + d) \cdot R(F)) \approx O(d^2 \cdot R(F)).
\]

It can be easily checked that, if we do not exploit the common information on \( A^{-1} \), the complexity amounts to \( O(d^3 \cdot R(F)) \).
5. The R package WMTRegions

The algorithm has been programmed as an R package and named WMTRegions. It is available for downloading from Comprehensive R Archive Network at http://CRAN.R-project.org/package=WMTregions.

5.1. Realization in C++

The algorithm has been realized in a C++ program. It is compiled as a library and is attached to the R package, which uses its functionality through an external function ComputeWMTR(char* Filename, char* Location).

The paper is also supplied with an autonomous C++ program that can be compiled under Windows operating system. This program provides a 3d-visualization as it is shown, for instance, in Figure 8. The interface is the same as for the R package (see Subsection 5.3). The visualization is designed by means of a cross-platform graphical specification OpenGL. As a future work, we plan to integrate this visualization into the R environment using a cross-platform realization of the extension OpenGL - GtkGLExt (http://library.gnome.org/devel/gtkglext/) to a graphical toolkit GTK+ (http://www.gtk.org). Note that WMTRegions also uses GTK+ through its R proxy package RGtk2 (Lawrence and Lang 2010).

5.2. R functions

The package contains functions for calculating and representing WMT regions:

- Function WMTR(fname) calculates the trimmed region. It is the main function, which reads input data from an input file (the name is specified by fname) and writes the result into an output file (by default, "Cloud.dat"). The format of the files is described in Subsection 5.3.

- Function showWMTR() offers a 3d-demonstration of the results in the R environment (in case d = 3). It visualizes a calculated WMT region (in dimension d = 3) as a convex polytope by representing its vertices in the rggobi (Lang, Swayne, Wickham, and Lawrence 2010) interactive graphics framework.

- Function loadWMTR(dim). This function loads the calculated WMT region of d = dim into a matrix object in order to work with it in the R environment.

- Function pointinTR(dpoint, tregion) checks whether a point is in a specified trimmed region.

- Function generTRsample(fname, dim, num, alpha, trtype) This function is an auxiliary function. It generates a random uniformly distributed data cloud of any size num and any dimension dim with a format, which is suitable for applying WMTR(fname). trtype defines a type of a WMT region, e.g., "zonoid". More information about the functions is found in the documentation of the R package.
5.3. Input and output

A data cloud is read from a text file of the following format:

- Type of the trimmed region (zonoid, ECH, ECH*, geometrically trimmed; given weight vector)
- Depth parameter
- Dimension
- Number of points of the data cloud
- Coordinates of each point

A full description of the format is given in the package documentation for the function WMTR. You can get it in a standard way:

```
R> library(WMTregions)
R> help(WMTR)
```

An output file "TRegion.dat" consists of lines, each representing a facet of the trimmed region. This representation is given by $d+1$ numbers, the first $d$ of which are coordinates of a normal to the facet while the last one defines an absolute distance of a hyperplane containing the facet to the origin.

5.4. Illustrative example

As an illustration of the program work we present a comparative example of four different types of weighted-mean trimmed regions for the same data and depth parameter ($\alpha = 0.221$). The calculation of a, e.g., zonoid region can be conducted as follows:

```
R> library("WMTregions")
Loading required package: rggobi
Loading required package: RGtk2

R> generTRsample("Cloud.dat", 3, 10, 0.221, "zonoid")
R> WMTR("Cloud.dat")
[1] "The trimmed region was successfully calculated!"

R> showWMTR()
R> tregion <- loadWMTR(3)
R> tregion
```
By changing the \texttt{trtype} parameter in the file "Cloud.dat" manually, one can calculate different types of WMT regions for the same data. Their visualization is exhibited in Figure 8.

6. Complexity of the algorithm

An exact algorithm has been constructed to compute the WMT regions of an empirical distribution in \( d \)-space for an arbitrarily given weight vector. It calculates all facets, edges, and vertices of a region at any given depth \( \alpha \in [0, 1] \). (Recall that \( \alpha = 0 \) and \( \alpha = 1 \) are trivial cases.)

A hash table plays a significant role, as it stores information about traversed ridges in a special structure, thus guaranteeing (due to the "double-hash") that each facet is generated only once. Hence the algorithm has as many loops as the WMT region has facets. Obviously, this is the minimum number of facet generating loops in this sort of algorithm.

At each facet \( F \) we have to calculate the normal of the facet and its distance from the origin. Further we have to determine all neighboring facets. This is done by only solving linear equations and calculating inner products. We have shown above that the complexity of this algorithmic loop amounts to \( O(d^2 \cdot R(F)) \), where \( R(F) \) is the number of ridges of the facet. \( R(F) \) can vary between \( d \) and \( 2d - 1 \), depending on the type of the WMT region and on \( \alpha \).

To obtain a rough conservative estimate of \( R(F) \), we may proceed as follows: First, note that \( R(F) \) is bounded by \( \bar{R} = \prod_{l=1}^{L} 2^{q_l - 1} \). Then suppose that the active blocks \( \mathcal{A}_l \) have about equal size and that their number \( L \), as a first-order approximation, is proportional to the dimension \( d \), say \( L \approx d/c \). Under these assumptions \( \bar{R} \approx L \cdot 2^{d/L - 1} \approx d/c \cdot 2^{c - 1} \), that is, \( R(F) \) is approximately bounded by the dimension \( d \) multiplied with a constant \( K = 2^{c - 1}/c \) that...
An Exact Algorithm for WMT Regions

does not depend on the dimension.

Searching for all neighbors of a facet, we have to calculate $n$ inner products, which gives complexity $O(nd)$. Hence, the complexity of one facet generating loop is described by $O((d^2 + nd) \cdot R) \approx O(d^2 n \cdot R)$, since $n > d$. If the average number of facets is denoted by $N(n, d)$, the average computational complexity of the algorithm amounts to $O((d^2 + nd) \cdot R \cdot N(n, d)) \approx O(d^2 n \cdot K \cdot N(n, d))$.

Regarding the hash table of created facets, each facet occupies $O(d \cdot \log_2 n)$ storage size, while the hash table, in almost any case, has a constant size $C$, not depending on $n$ and $d$. Therefore, the use of general memory is of the order $O(N(n, d) \cdot d \cdot \log_2 n + C)$. Facets, once they have been created, are put into a secondary store, thus considerably lowering the storage cost.

Table 1 exhibits the results of a first small simulation study. It gives an idea how the time for computing a single facet depends on $d$ and $n$ and how it varies with several types of WMT regions: zonoid, ECH*, ECH, and geometrically trimmed regions (for the latter two, see Dyckerhoff and Mosler (201x)). The data is distributed uniformly on a $d$-dimensional cube.

We focus on the time per facet (TpF) because it characterizes the efficiency of the algorithm in a most obvious way. The total computational time amounts to the latter multiplied by the number of facets, which is a parameter depending only on the data. We observe that the TpF shows the following growth behavior: Approximately linear on $n$ and slightly convex on $d$, which may be seen as some low order polynomial dependency on dimension.

Acknowledgments

We are thankful to Rainer Dyckerhoff for fruitful discussions and valuable advices.

References


<table>
<thead>
<tr>
<th>WMTD type</th>
<th>$d$</th>
<th>$n$</th>
<th>Time per facet</th>
<th>Total time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zonoid 3</td>
<td>10</td>
<td></td>
<td>0.009700</td>
<td>0.445</td>
</tr>
<tr>
<td>Zonoid 3</td>
<td>15</td>
<td></td>
<td>0.013840</td>
<td>1.531</td>
</tr>
<tr>
<td>Zonoid 4</td>
<td>10</td>
<td></td>
<td>0.012474</td>
<td>1.609</td>
</tr>
<tr>
<td>Zonoid 4</td>
<td>15</td>
<td></td>
<td>0.015862</td>
<td>14.140</td>
</tr>
<tr>
<td>Zonoid 5</td>
<td>10</td>
<td></td>
<td>0.017370</td>
<td>2.398</td>
</tr>
<tr>
<td>Zonoid 5</td>
<td>15</td>
<td></td>
<td>0.022335</td>
<td>40.953</td>
</tr>
<tr>
<td>ECH 3</td>
<td>10</td>
<td></td>
<td>0.009111</td>
<td>0.843</td>
</tr>
<tr>
<td>ECH 3</td>
<td>15</td>
<td></td>
<td>0.012212</td>
<td>2.375</td>
</tr>
<tr>
<td>ECH 4</td>
<td>10</td>
<td></td>
<td>0.015255</td>
<td>21.891</td>
</tr>
<tr>
<td>ECH 4</td>
<td>15</td>
<td></td>
<td>0.019632</td>
<td>97.765</td>
</tr>
<tr>
<td>ECH 5</td>
<td>10</td>
<td></td>
<td>0.023519</td>
<td>117.625</td>
</tr>
<tr>
<td>ECH 5</td>
<td>15</td>
<td></td>
<td>0.029733</td>
<td>1032.75</td>
</tr>
<tr>
<td>ECH* 3</td>
<td>10</td>
<td></td>
<td>0.009610</td>
<td>0.750</td>
</tr>
<tr>
<td>ECH* 3</td>
<td>15</td>
<td></td>
<td>0.012218</td>
<td>1.617</td>
</tr>
<tr>
<td>ECH* 4</td>
<td>10</td>
<td></td>
<td>0.015286</td>
<td>22.922</td>
</tr>
<tr>
<td>ECH* 4</td>
<td>15</td>
<td></td>
<td>0.020011</td>
<td>94.070</td>
</tr>
<tr>
<td>ECH* 5</td>
<td>10</td>
<td></td>
<td>0.022970</td>
<td>139.070</td>
</tr>
<tr>
<td>ECH* 5</td>
<td>15</td>
<td></td>
<td>0.029660</td>
<td>890.68</td>
</tr>
<tr>
<td>Geometrical 3</td>
<td>10</td>
<td></td>
<td>0.009355</td>
<td>0.930</td>
</tr>
<tr>
<td>Geometrical 3</td>
<td>15</td>
<td></td>
<td>0.013056</td>
<td>1.101</td>
</tr>
<tr>
<td>Geometrical 4</td>
<td>10</td>
<td></td>
<td>0.015356</td>
<td>23.805</td>
</tr>
<tr>
<td>Geometrical 4</td>
<td>15</td>
<td></td>
<td>0.020157</td>
<td>93.406</td>
</tr>
<tr>
<td>Geometrical 5</td>
<td>10</td>
<td></td>
<td>0.023036</td>
<td>137.312</td>
</tr>
<tr>
<td>Geometrical 5</td>
<td>15</td>
<td></td>
<td>0.029794</td>
<td>1028.51</td>
</tr>
</tbody>
</table>

Table 1: Sample computational results.


Affiliation:
Pavel Bazovkin
Graduate School of Risk Management
University of Cologne
Meister-Ekkehart-Str. 11
50923 Cologne, Germany
Telephone: +49/221/470-7705
E-mail: bazovkin@wiso.uni-koeln.de

Univ.-Prof. Dr. Karl Mosler
Department of Economic and Social Statistics
University of Cologne
Albertus-Magnus-Platz
50923 Cologne, Germany
Telephone: +49/221/470-4266
E-mail: mosler@statistik.uni-koeln.de