Computation of projection regression depth and its induced median

Yijun Zuo[*]

Department of Statistics and Probability, Michigan State University, East Lansing, MI 48824, USA

Abstract

Notions of depth in regression have been introduced and studied in the literature. The most famous example is Regression Depth (RD), which is a direct extension of location depth to regression. The projection regression depth (PRD) is the extension of another prevailing location depth, the projection depth, to regression. The computation issues of the RD have been discussed in the literature. The computation issues of the PRD have never been dealt with before. The computation issues of the PRD and its induced median (maximum depth estimator) in a regression setting are addressed now. For a given $\boldsymbol{\beta} \in \mathbb{R}^p$ exact algorithms for the PRD with cost $O(n^2 \log n)$ $(p = 2)$ and $O(N(n,p)(p^3 + n \log n + np^{1.5} + npN_{Iter}))$ $(p > 2)$ and approximate algorithms for the PRD and its induced median with cost respectively $O(N_{\mathbf{v}} np)$ and $O(RpN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}} N_{Iter}))$ are proposed. Here $N(n,p)$ is a number defined based on the total number of $(p-1)$ dimensional hyperplanes formed by points induced from sample points and the $\boldsymbol{\beta}$; $N_{\mathbf{v}}$ is the total number of unit directions $\mathbf{v}$ utilized; $N_{\boldsymbol{\beta}}$ is the total number of candidate regression parameters $\boldsymbol{\beta}$ employed; N_{Iter} is the total number of iterations carried out in an optimization algorithm; R is the total number of replications. Furthermore, as the second major contribution, three PRD induced estimators, which can be computed up to 30 times faster than that of the PRD induced median while maintaining a similar level of accuracy are introduced. Examples and simulation studies reveal that the depth median induced from the PRD is favorable in terms of robustness and efficiency, compared to the maximum depth estimator induced from the RD, which is the current leading regression median.

Keywords: depth in regression, maximum depth estimator, computation, approximate and exact algorithms.

1. Introduction

Notions of location depth have been introduced and extensively studied in the literature over the last three decades. Depth notions have found applications in diverse fields and disciplines (see Zuo (2018a) for a review). Among others (Simplicial depth (Liu (1990)), Zonoid depth (Koshevoy and Mosler (1997), Mosler (2002, 2012)) and Spatial depth (Vardi and Zhang (2000), etc.), two prevailing location depth notions are the Tukey halfspace depth (HD) (Tukey (1975)) (popularized by Donoho and Gasko (1992)) and the projection depth (PD) (Liu(1992), Zuo and Serfling (2000)) (thoroughly studied in Zuo (2003)), both of which are in the spirit of the projection-pursuit scheme.

[*]Corresponding author
Email address: zuo@msu.edu (Yijun Zuo)

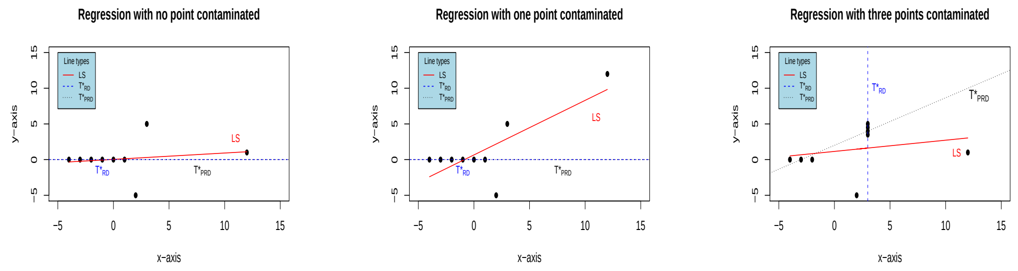
One naturally wonders if the depth notion can be extended to a regression setting. Regression depth (RD) of Rousseeuw and Hubert (1999) (RH99) is the most famous example, which directly extends HD to regression, whereas projection regression depth (PRD), induced from Marrona and Yohai (1993) (MY93) and introduced in Zuo (2018a) (Z18a), is an extension of the PD to regression.

Like their location counterparts, the most remarkable advantage of the notion of depth in regression is the direct introduction of the median-type estimator, otherwise known as the maximum (or deepest) regression depth estimator for regression parameters in a multi-dimensional setting. The maximum (deepest) regression depth estimators serve as *robust* alternatives to the classical least squares or least absolute deviations estimator of unknown parameters in a general linear regression model:

$$y \quad = \quad (1, \mathbf{x}')\boldsymbol{\beta} + e, \tag{1}$$

where $'$ denotes the transpose of a vector, and random vector $\mathbf{x} = (x_1, \cdots, x_{p-1})' \in \mathbb{R}^{p-1}$ and parameter vector $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}_1')' \in \mathbb{R}^p$ $(p \geq 2)$ and random variables y and e are in $\mathbb{R}^1$. Let $\mathbf{w} = (1, \mathbf{x}')'$. Then $y = \mathbf{w}'\boldsymbol{\beta} + e$. We use this model or (1) interchangeably depending on the context.

The maximum depth estimator induced from the RD, $\mathbf{T}^*_{RD}$, can asymptotically resist up to 33% (Van Aelst and Rousseeuw (2000) (VAR00))(whereas the one from the PRD, $\mathbf{T}^*_{PRD}$, can resist up to 50% (Zuo (2019a)(Z19a)) contamination without breakdown, in contrast to the 0% of the classical LS estimator. An illustration of these facts is given in Figure 1, where the data set is given in Table 9 of Chapter 2 from Rousseeuw and Leroy (1987) (RL87). The original data set contains nine bivariate points.



(a) no-contamination (b) 1-point-contaminated (c) 3-points-contamination

Figure 1: Three regression lines for data with or without contamination (solid red for the LS, dashed blue for the $\mathbf{T}^*_{RD}$ and dotted black for the $\mathbf{T}^*_{PRD}$). (a) Original nine-point data set, the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ are identical. (b) Contaminated data set with one original point $(12, 1)'$ moved to $(12, 12)'$, leading to a drastic change in the LS line while both the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ are unchanged and resist the contamination. (c) Contaminated data set with three original points moved to the points with 3 as their x-coordinates, the $\mathbf{T}^*_{RD}$ breaks down while both the $\mathbf{T}^*_{PRD}$ and the LS lines are still informative.

For any $\boldsymbol{\beta} \in \mathbb{R}^p$ and joint distribution P of $(\mathbf{x}', y)$ in $\mathbb{R}^p$, RH99 defined the *regression depth* of $\boldsymbol{\beta}$–denoted hereafter by RD$(\boldsymbol{\beta}; P)$– to be the minimum probability mass that needs to be passed when tilting (the hyperplane induced from) $\boldsymbol{\beta}$ in any way until it is vertical. The maximum regression depth estimating functional $\mathbf{T}^*_{RD}$ (also denoted by $\boldsymbol{\beta}^*_{RD}$) is then defined as

$$\mathbf{T}^*_{RD}(P) = \arg\max_{\boldsymbol{\beta} \in \mathbb{R}^p} RD(\boldsymbol{\beta}; P). \tag{2}$$

Various characterizations of RD$(\boldsymbol{\beta}; P)$ have been given in the literature, e.g. Zuo (2018b).

By modifying the P-estimate of Marrona and Yohai (1993) (MY93) to achieve the scale invariance property, Z18a introduced projection regression depth (PRD), defined based on the so-called "unfitness" (UF) for a given candidate regression parameter $\boldsymbol{\beta} \in \mathbb{R}^p$:

$$\mathrm{UF}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)}) = \sup_{\mathbf{v} \in \mathbb{S}^{p-1}} |R(F_{(\mathbf{w}'\mathbf{v},\ y-\mathbf{w}'\boldsymbol{\beta})})| \big/ S(F_y), \tag{3}$$

$$\mathrm{PRD}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)}) = 1/(1 + \mathrm{UF}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)})), \tag{4}$$

where $F_{\mathbf{Z}'}$ stands for the distribution of the p-dimensional random vector $\mathbf{Z} \in \mathbb{R}^p$, $\mathbf{w} = (1, \mathbf{x}')' \in \mathbb{R}^p$, $\mathbb{S}^{p-1} = \{\mathbf{u} \in \mathbb{R}^p : \ \|\mathbf{u}\| = 1\}$, R will be restricted to the univariate regression functional of the form $R(F_{(\mathbf{w}'\mathbf{v},\ y-\mathbf{w}'\boldsymbol{\beta})}) = T\big(F_{\frac{y-\mathbf{w}'\boldsymbol{\beta}}{\mathbf{w}'\mathbf{v}}}\big)$ and it is regression, scale, and affine equivariant (see page 116 of RL87 for definitions). T could be a univariate location functional that is location, and scale (or called affine) equivariant; S is a scale functional that is translation invariant and scale equivariant (see pages 158-159 of RL87 for definitions), and $S(F_y)$ does not depend on $\mathbf{v}$ and $\boldsymbol{\beta}$, see Z18a.

It is not difficult to see that $\mathrm{UF}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)})$ and $\mathrm{PRD}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)})$ are the regression counterparts of the outlyingness function $O(\boldsymbol{\beta}; F_{\mathbf{x}})$ and the projection depth function $\mathrm{PD}(\boldsymbol{\beta}; F_{\mathbf{x}})$ (Zuo (2003)), respectively.

Examples of T in (3) include mean, quantile, median (Med), and location functionals in Wu and Zuo (2009). Examples of S in (3) include standard deviation, median absolute deviations from the median (MAD), and scale functionals in Wu and Zuo (2008).

For the consideration of robustness, in the sequel, (T, S) is fixed and it is the pair $(\mathrm{Med}, \mathrm{MAD})$, unless otherwise stated. Hereafter, we write $\mathrm{Med}(Z)$ rather than $\mathrm{Med}(F_Z)$. For this special choice of T and S such that

$$\begin{aligned} R(F_{(\mathbf{w}'\mathbf{v},\ y-\mathbf{w}'\boldsymbol{\beta})}) &= \mathrm{Med}_{\mathbf{w}'\mathbf{v}\neq 0}\big(\frac{y-\mathbf{w}'\boldsymbol{\beta}}{\mathbf{w}'\mathbf{v}}\big), \\ S(F_y) &= \mathrm{MAD}(F_y). \end{aligned}$$

We have

$$\mathrm{UF}(\boldsymbol{\beta}; F_{(\mathbf{x}',y)}) = \sup_{\mathbf{v} \in \mathbb{S}^{p-1}} \Big|\mathrm{Med}_{\mathbf{w}'\mathbf{v}\neq 0}\big(\frac{y-\mathbf{w}'\boldsymbol{\beta}}{\mathbf{w}'\mathbf{v}}\big)\Big| \Big/ \mathrm{MAD}(F_y), \tag{5}$$

and

$$\mathrm{PRD}\left(\boldsymbol{\beta}; F_{(\mathbf{x}',y)}\right) = \inf_{\mathbf{v} \in \mathbb{S}^{p-1}, \mathbf{w}'\mathbf{v}\neq 0} \frac{\mathrm{MAD}(F_y)}{\mathrm{MAD}(F_y) + \Big|\mathrm{Med}\big(\frac{y-\mathbf{w}'\boldsymbol{\beta}}{\mathbf{w}'\mathbf{v}}\big)\Big|}. \tag{6}$$

Applying the min-max (or max-min) scheme, we obtain the maximum (deepest) *projection regression depth estimating functional* (also denoted by $\boldsymbol{\beta}^*_{\mathrm{PRD}}$) w.r.t. the pair (T, S)

$$\mathbf{T}^*_{PRD}(F_{(\mathbf{x}',y)}) = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \mathrm{UF}(\boldsymbol{\beta};\ F_{(\mathbf{x}',y)}) = \arg\max_{\boldsymbol{\beta} \in \mathbb{R}^p} \mathrm{PRD}\left(\boldsymbol{\beta};\ F_{(\mathbf{x}',y)}\right). \tag{7}$$

When a sample $\mathbf{Z}^n = \{(\mathbf{x}'_i, y_i)', i = 1, \cdots, n\}$ of $\mathbf{Z} := (\mathbf{x}', y)' \in \mathbb{R}^p$ is given, an empirical distribution $F^n_{\mathbf{Z}}$ based on $\mathbf{Z}^n$ is obtained. Replacing $F_{(\mathbf{x}',y)}$ above by $F^n_{\mathbf{Z}}$ we obtain all empirical versions.

While both the RD and the PRD enjoy desirable properties such as high breakdown robustness, these regression depth functions prove difficult to compute in practice since they involve the projection-pursuit scheme (see Z18a). The computation of the RD has been discussed in RH99, in Rousseeuw and Struyf (1998) (RS98), and in Liu and Zuo (2014) (LZ14). The computation issues of the PRD and the $\mathbf{T}^*_{PRD}$ have never been addressed. Presenting exact and approximate algorithms for the PRD and discussing the algorithms for the computation of the $\mathbf{T}^*_{PRD}$ are the two main goals of this article.

The third goal is to introduce several PRD induced estimators which can be computed much faster than that of the $\mathbf{T}^*_{PRD}$.

The rest of this article is organized as follows. Section 2 presents the computation problem and addresses the exact and approximate computation algorithms for the UF$(\boldsymbol{\beta}, F^n_{\mathbf{Z}})$, and equivalently for the PRD$(\boldsymbol{\beta}, F^n_{\mathbf{Z}})$. Furthermore, theoretical results are established and exact and approximate algorithms are presented along with abounded examples. Section 3 is devoted to (i) the computation of the $\mathbf{T}^*_{PRD}(\boldsymbol{\beta}, F^n_{\mathbf{Z}})$, (ii) examples of the exact computation of the PRD as well as the approximate computation of the $\mathbf{T}^*_{PRD}$, and (iii) comparisons of performance between the $\mathbf{T}^*_{PRD}$ against leading competitors such as LS, $\boldsymbol{\beta}^*_{RD}$, and ltsReg. Section 4 introduces three depth induced regression estimators that can run much faster than $\mathbf{T}^*_{PRD}$ in addition to maintaining small empirical mean squared errors. Section 5 investigates the finite sample relative efficiency of the $\mathbf{T}^*_{PRD}$. Brief concluding comments end Section 6 and the article.

2. Computation of PRD

2.1. The computation problem

To compute the PRD$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$, it suffices to compute the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$. Namely, to compute the following quantity:

$$\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}}) = \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_{\mathbf{w}'_i\mathbf{v}\neq 0}\big\{\frac{y_i - \mathbf{w}'_i\boldsymbol{\beta}}{\mathbf{w}'_i\mathbf{v}}\big\}\Big| \Big/ S_y, \tag{8}$$

where $\mathbf{w}'_i = (1, \mathbf{x}'_i)$ and $S_y = \mathrm{MAD}\{y_i\}$. Hereafter, we assume that **(A1)**: $P(\mathbf{w}'\mathbf{v} = 0) = 0, \forall\, \mathbf{v} \in \mathbb{S}^{p-1}$ and **(A2)** $P(r(\boldsymbol{\beta}) = 0) = 0$, where $r(\boldsymbol{\beta}) = y - \mathbf{w}'\boldsymbol{\beta},\ \ \forall\ \boldsymbol{\beta} \in \mathbb{R}^p$. **(A1)-(A2)** hold automatically if $(\mathbf{x}', y)'$ has a density or if $\mathbf{x}$ does not concentrate on a single $(p-2)$ dimensional hyperplane in $\mathbf{x}$ space and any $(p-1)$ dimensional hyperplane determined by $r(\boldsymbol{\beta}) = 0$ in $(\mathbf{x}', y)'$ space does not contain any probability mass.

For simplicity of description, we write $\mathbf{t}'_i = \mathbf{w}'_i/r_i(\boldsymbol{\beta})$, where $r_i(\boldsymbol{\beta}) = y_i - \mathbf{w}'_i\boldsymbol{\beta}$. Now the computation of UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ in (8) is equivalent to the computation of

$$\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}}) = \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_{\mathbf{t}'_i\mathbf{v}\neq 0}\big\{\frac{1}{\mathbf{t}'_i\mathbf{v}}\big\}\Big| \Big/ S_y. \tag{9}$$

Again for simplicity, we write $k^{\mathbf{v}}_i = 1/\mathbf{t}'_i\mathbf{v}$ and $u^{\mathbf{v}}_i = \mathbf{t}'_i\mathbf{v}$. The latter two are well defined almost surely (a.s.) under **(A1)-(A2)**. Without loss of generality (w.l.o.g.), we assumes that $S_y = 1$ (since it does not depend on $\mathbf{v}$ or $\boldsymbol{\beta}$). The UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ in (9) is then

$$\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}}) = \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_i\big\{k^{\mathbf{v}}_i\big\}\Big| := \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \big|g(\mathbf{v})\big|. \tag{10}$$

The exact computation of (10) above is still very challenging if not impossible. Let $k^{\mathbf{v}}_{(1)} \le k^{\mathbf{v}}_{(2)} \cdots \le k^{\mathbf{v}}_{(n)}$ be ordered values of $k^{\mathbf{v}}_i$. Partition $\mathbb{S}^{p-1}$ into two disjoint parts

$$\mathscr{S}_1 = \{\mathbf{v} \in \mathbb{S}^{p-1} :\ k^{\mathbf{v}}_{(1)} < 0 \text{ and } k^{\mathbf{v}}_{(n)} > 0\};\quad \mathscr{S}_2 = \{\mathbf{v} \in \mathbb{S}^{p-1} :\ k^{\mathbf{v}}_{(1)} > 0 \text{ or } k^{\mathbf{v}}_{(n)} < 0\}. \tag{11}$$

It is readily seen that both $\mathcal{S}_1$ and $\mathcal{S}_2$ are symmetric about the origin. That is, if $\mathbf{v} \in \mathcal{S}_i$ then, $-\mathbf{v} \in \mathcal{S}_i$. Now the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ in (10) can be expressed as follows:

$$\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}}) = \max\big\{\sup_{\mathbf{v}\in\mathcal{S}_1} |g(\mathbf{v})|,\ \ \sup_{\mathbf{v}\in\mathcal{S}_2} |g(\mathbf{v})|\big\}. \tag{12}$$

Exact computation of the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ in (12) is a challenging task, whereas approximate computation is relatively straightforward (but it is still difficult to assess its accuracy without the benchmark of the exact result). We shall address the two approaches separately in the sequel.

2.2. Exact Computation

2.2.1. Theoretical results

For a given sample $\mathbf{Z}^{(n)} := \{(\mathbf{x}_i', y_i)',\ i = 1, \cdots, n\}$, a $\boldsymbol{\beta}$ in $\mathbb{R}^p$ and a $\mathbf{v} \in \mathbb{S}^{p-1}$, since $k^{\mathbf{v}}_{(1)} \le k^{\mathbf{v}}_{(2)} \le \cdots \le k^{\mathbf{v}}_{(n)}$ are ordered values of $k^{\mathbf{v}}_i = 1/\mathbf{t}_i'\mathbf{v}$, then $1/\mathbf{t}_{i_1}'\mathbf{v} \le 1/\mathbf{t}_{i_2}'\mathbf{v} \le \cdots \le 1/\mathbf{t}_{i_n}'\mathbf{v}$ for some $\{i_1, \cdots, i_n\}$, a permutation of $\{1, 2, \cdots, n\}$. Similarly, $u^{\mathbf{v}}_{(1)} \le u^{\mathbf{v}}_{(2)} \le \cdots \le u^{\mathbf{v}}_{(n)}$ corresponds to a permutation $\{j_i, \cdots, j_n\}$ such that $u^{\mathbf{v}}_{j_1} \le u^{\mathbf{v}}_{j_2} \le \cdots, \le u^{\mathbf{v}}_{j_n}$ for $u^{\mathbf{v}}_i = \mathbf{t}_i'\mathbf{v}$.

Proposition 2.1: Assume **(A1)** and **(A2)** hold. Let $N^-_{\mathbf{v}} := \sum_{i=1}^n \mathbf{I}(k^{\mathbf{v}}_i < 0)$. The unfitness of $\boldsymbol{\beta}$ in (8) can be computed equivalently via (12) which can be computed as follows.

Denote $n1 := \lfloor (n+1)/2 \rfloor$ and $n2 := \lfloor (n+2)/2 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function.

(i) For $\mathbf{v} \in \mathcal{S}_2$,

$$\sup_{\mathbf{v}\in\mathcal{S}_2} |g(\mathbf{v})| = \begin{cases} \max_{\mathbf{v}\in\mathcal{S}_2} \frac{(\mathbf{t}_{i_{n1}}' + \mathbf{t}_{i_{n2}}')\mathbf{v}/2}{\mathbf{v}'\mathbf{t}_{i_{n1}}\mathbf{t}_{i_{n2}}'\mathbf{v}} & \text{if } N^-_{\mathbf{v}} = 0, \\ -\min_{\mathbf{v}\in\mathcal{S}_2} \frac{(\mathbf{t}_{i_{n1}}' + \mathbf{t}_{i_{n2}}')\mathbf{v}/2}{\mathbf{v}'\mathbf{t}_{i_{n1}}\mathbf{t}_{i_{n2}}'\mathbf{v}} & \text{if } N^-_{\mathbf{v}} = n. \end{cases}$$

(ii) For $\mathbf{v} \in \mathcal{S}_1$, let m be a non-negative integer.

if $n = 2m + 1$,

$$\sup_{\mathbf{v}\in\mathcal{S}_1} |g(\mathbf{v})| = \begin{cases} -1\Big/\max_{\mathbf{v}\in\mathcal{S}_1} \mathbf{t}_{i_{n1}}'\mathbf{v} & \text{if } k^{\mathbf{v}}_{(n1)} < 0, \\ 1\Big/\min_{\mathbf{v}\in\mathcal{S}_1} \mathbf{t}_{i_{n1}}'\mathbf{v} & \text{if } k^{\mathbf{v}}_{(n1)} > 0, \end{cases}$$

if $n = 2m + 2$,

$$\sup_{\mathbf{v}\in\mathcal{S}_1} |g(\mathbf{v})| = \begin{cases} \left|\max_{\mathbf{v}\in\mathcal{S}_1} \frac{(\mathbf{t}_{i_{n1}}' + \mathbf{t}_{i_{n2}}')\mathbf{v}/2}{\mathbf{v}'\mathbf{t}_{i_{n1}}\mathbf{t}_{i_{n2}}'\mathbf{v}}\right| & \text{if } k^{\mathbf{v}}_{(n1)} < 0 \text{ and } k^{\mathbf{v}}_{(n2)} > 0, \\ \max_{\mathbf{v}\in\mathcal{S}_1} \frac{\left(\mathbf{t}_{i_{n1}}' + \mathbf{t}_{i_{n2}}'\right)\mathbf{v}/2}{\mathbf{v}'\mathbf{t}_{i_{n1}}\mathbf{t}_{i_{n2}}'\mathbf{v}} & \text{if } k^{\mathbf{v}}_{(n1)} > 0, \\ -\min_{\mathbf{v}\in\mathcal{S}_1} \frac{\left(\mathbf{t}_{i_{n1}}' + \mathbf{t}_{i_{n2}}'\right)\mathbf{v}/2}{\mathbf{v}'\mathbf{t}_{i_{n1}}\mathbf{t}_{i_{n2}}'\mathbf{v}} & \text{if } k^{\mathbf{v}}_{(n2)} < 0. \end{cases}$$

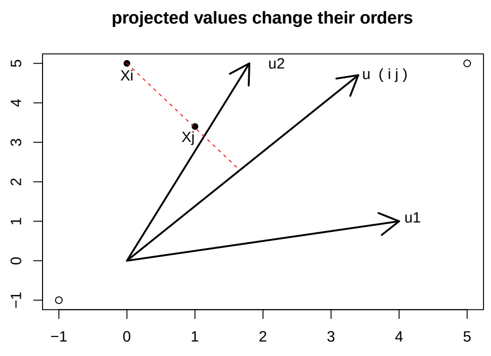
Proof: Note that under **(A1)-(A2)** $\mathcal{S}_i$ $(i = 1, 2)$ are a. s. closed sets. In light of the definitions of the regular univariate sample median and $\mathcal{S}_i$, the proof follows immediately. Details are straightforward to verify and thus are omitted. ■

Remarks 2.1: The proposition gives a clear foundation for the exact computation of the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$, or equivalently the PRD$((\boldsymbol{\beta}; F^n_{\mathbf{Z}}) = \left(1 + \text{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})\right)^{-1}$.

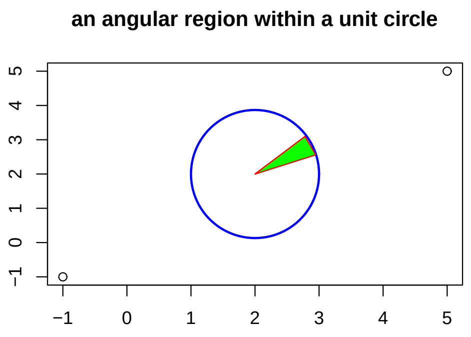
(I) the UF$(\beta; F^n_{\mathbf{Z}})$ can be computed exactly via the optimization over closed sets $\mathcal{S}_i$. There are unified formulas over $\mathcal{S}_i$ for distinct cases of permutations. The two types of optimization problems that exist in the proposition are

(i) **Type I**: min (or max) of $\mathbf{c}'\mathbf{v}$ for $\mathbf{v}$ over a closed subset set of $\mathbb{S}^{p-1}$ and $\mathbf{c} \in \mathbb{R}^p$.

(ii) **Type II**: min (or max) of $\frac{\mathbf{b}'\mathbf{v}}{\mathbf{v}'\mathbf{A}\mathbf{v}}$ for $\mathbf{v}$ over a closed subset set of $\mathbb{S}^{p-1}$ and $\mathbf{b} \in \mathbb{R}^p$, $\mathbf{A} \in \mathbb{R}^{p\times p}$ (A could be treated as symmetric and positive-definite over the set).



(a) circular-sequence demonstration



(b) angular region demonstration

Figure 2: (a) $\mathbf{u}$ (or ij) is perpendicular to the line segment connecting the points X_i, X_j and between $\mathbf{u}_1$ and $\mathbf{u}_2$. When the two points are projected to $\mathbf{u}_1$, $\mathbf{u}_2$ and $\mathbf{u}$, the X_i precedes the X_j on $\mathbf{u}_1$ whereas on $\mathbf{u}_2$ it is reversed. On $\mathbf{u}$ they overlap. (b) a unit circle is cut into pieces (angular regions) by the median sequence. Over each piece, the median of the projected values is the average of the middle two (or one) of the projected values of the same two (or one) fixed points (see Figure 3).

(II) $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{A}$ above are determined by $\{\mathbf{t}_i\}$ and depend on $\mathbf{v}$ only through the permutation $i_1, \cdots, i_n$ which is induced by the projection of $\{\mathbf{t}_i\}$ onto $\mathbf{v}$. That is, for a given sample and $\boldsymbol{\beta} \in \mathbb{R}^p$, and a $\mathbf{v} \in \mathbb{S}^{p-1}$ [or more generally a fixed permutation $i_1, \cdots, i_n$ (of $\{1, 2, \cdots, n\}$) over a set of $\mathbf{v}$]; $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{A}$ are constant vectors and a matrix.

Hence, with the constraints discussed in the sequel, **Type I** optimization can be solved by linear programming and **Type II** optimization can be solved by gradient-type, Newton-type, or interior-point methods (see, e.g. Numerical Recipes (2007) Chapter 10, Freund (2004), and Boyd and Vandenberghe (2004)), among others. ■

To facilitate the explanation of the basic idea to achieve the exact computation via Proposition 2.1, we first invoke the concept of a "circular sequence" (see, e.g. Edelsbrunner (1987)).

Given n general points, $\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_n$ (obtained from $\mathbf{Z}^{(n)}$ and a $\boldsymbol{\beta}$) in $\mathbb{R}^p$, and any unit vector $\mathbf{v}$, assume that $u^{\mathbf{v}}_{i_1} \leq u^{\mathbf{v}}_{i_2} \leq \cdots \leq u^{\mathbf{v}}_{i_n}$ (recall $u^{\mathbf{v}}_j = \mathbf{t}'_j\mathbf{v}$). Then $\{i_1, i_2, \cdots, i_n\}$ forms a permutation of $\{1, 2, \cdots, n\}$ (e.g., see (b) of Fig. 3, where "4321" represents a permutation from the projection of 4 points (labeled as $1, \cdots, 4$) to the direction labeled as "34").

If one rotates $\mathbf{v}$ counter-clockwise (in $\mathbb{R}^2$), then one will get a sequence of permutations. This periodic sequence of permutations is called a *circular sequence* (see the permutations in Fig. 3). In $\mathbb{R}^p$ ($p > 2$), when the unit vector $\mathbf{v}$ rotates on the unit sphere, we again get a sequence of permutations from the subscripts of ordered projected values, or a circular/spherical sequence.

Some observations on circular/spherical sequences

O1 The permutation obtained from the projection of the n points on $\mathbf{v}$ is exactly the reverse of the permutation obtained from the projection of them on $-\mathbf{v}$.

O2 Two successive permutations of a circular/spherical sequence differ only by switching p integers in the sequence (see (a) of Fig. 2).

O3 The permutation changes only whenever the rotation of $\mathbf{v}$ passes through a direction perpendicular to a $(p-1)$-dimensional subspace formed by p data points in a given data set that is in general position (defined later) (see Fig. 3 and (a) of Fig. 2).

Proposition 2.2: Assume **(A1)** and **(A2)** hold. Let $V \subset \mathbb{S}^{p-1}$ be a piece of a unit circle/sphere such that $\forall\ \mathbf{v} \in V$, $u^{\mathbf{v}}_{j_1} \le u^{\mathbf{v}}_{j_2} \le \cdots \le u^{\mathbf{v}}_{j_n}$. That is, over V, $j_1, j_2, \cdots, j_n$ is a fixed permutation of $\{1, 2, \cdots, n\}$. Then (i) $N^-_{\mathbf{v}}$ is a constant over V; (ii) there are no $\mathbf{v}_i \in V$ $(i = 1, 2)$ such that $\mathbf{v}_1 \ne \mathbf{v}_2$ and $\mathbf{v}_i \in S_i$.

Proof:

(i) When $\mathbf{v}$ moves over V, in order for $N^-_{\mathbf{v}}$ to change its value, it is obvious that at least one $k^{\mathbf{v}}_i$ changes from less than zero to greater than or equal to zero. That is, $\mathbf{v}$ must cross a $\mathbf{v_0}$ such that $k^{\mathbf{v_0}}_i = 0$. The latter happens with probability zero under **(A2)**.

(ii) Assume that there is a $\mathbf{v} \in S_2 \cap V$, then $N^-_{\mathbf{v}}$ is either 0 or n. By (i) there exists no $\mathbf{v_1} \in V$ such that $\mathbf{v_1} \in S_1$, since the latter means $0 < N^-_{\mathbf{v_1}} < n$, a contradiction. That is, $V \subset S_2$. Similarly, if there is a $\mathbf{v} \in S_1 \cap V$, one can conclude that $V \subset S_1$. ■

To get the exact value of the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ utilizing Proposition 2.1, it seems that one has to know the set $\mathcal{S}_i$ first, $i = 1, 2$ (or more accurately their boundaries). $\mathcal{S}_2$ can be empty. In fact, when the convex hull formed by all $\mathbf{t}_i$s contains the origin, then $\mathcal{S}_1 = \mathbb{S}^{p-1}$. Fortunately, we do not have to identify $\mathcal{S}_i$, $i = 1, 2$.

Since there is no unique formula over $\mathcal{S}_i$ in the Proposition, the exact computation task requires us to further partition $\mathcal{S}_i$ into disjoint pieces. For example, we could partition $\mathcal{S}_1$ into five pieces and $\mathcal{S}_2$ into two pieces, according to the cases listed in Proposition 2.1. The latter task is not as easy as identifying $\mathcal{S}_i$. For example, identifying all $\mathbf{v} \in \mathcal{S}_1$ such that $k^{\mathbf{v}}_{(n1)} > 0$ for an even n case is not straightforward at all. We seek other approaches below.

For a given sample $\mathbf{Z}^{(n)}$ and a $\boldsymbol{\beta} \in \mathbb{R}^p$ and a $\mathbf{v} \in \mathbb{S}^{p-1}$, there is a unique permutation $i_1, \cdots, i_n$ of $\{1, 2, \cdots, n\}$ induced by $k^{\mathbf{v}}_i = 1/\mathbf{t}'_i\mathbf{v}$. The $k^{\mathbf{v}}_{i_j}$ $(j = 1, \cdots, n)$ is all we need for the calculation in (10) or Proposition 2.1. However, a permutation $i_1, \cdots, i_n$ corresponds to a set of $\mathbf{v} \in \mathbb{S}^{p-1}$, with each member of the set capable of producing the same permutation via $k^{\mathbf{v}}_i$.

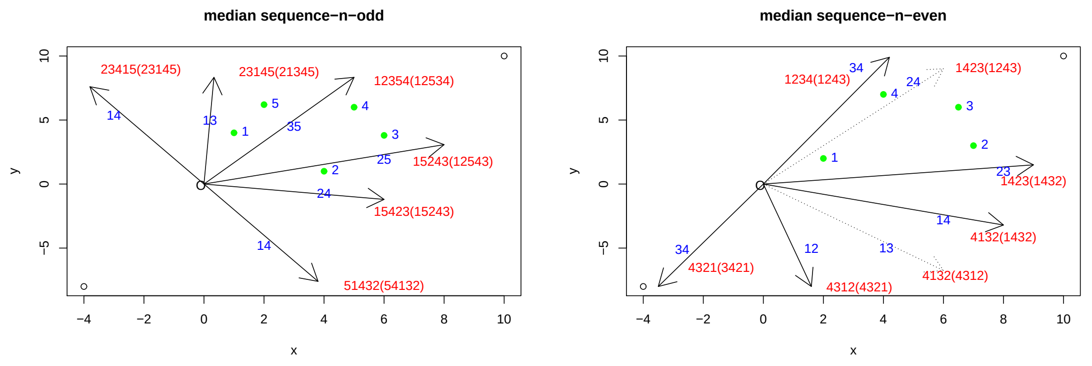
That is, a fixed permutation corresponds to a unique piece of $\mathbb{S}^{p-1}$ (or of the surface of the unit sphere). There are in total at most $n!$ possible permutations hence $n!$ disjoint pieces that partition the $\mathbb{S}^{p-1}$ (or the surface of the unit sphere). By Proposition 2.2, each piece belongs to either $\mathcal{S}_1$ or $\mathcal{S}_2$. Selecting one $\mathbf{v}$ from each piece is sufficient for the exact computation of the UF$(\boldsymbol{\beta}; F^n_Z)$ via Proposition 2.1. The cost is approximately of order $O(n^{n+1/2})$ without counting optimization cost, which is computationally unaffordable. We seek to merge some pieces.

In light of Observation 3 (O3) on $\mathbf{v}$ induced permutations (in regards to the circular or spherical sequence), when $\mathbf{v}$ moves on the surface of the unit sphere, its induced permutation changes only when it crosses a hyperplane (H_0) that goes through the origin and is perpendicular to another hyperplane (H_1) that is formed by sample points from $\{\mathbf{t}_i\}$.

The former hyperplanes (H_0's) (each containing the origin) cut the $\mathbb{S}^{p-1}$ into disjoint $N(n, p)$ pieces P_k $(k = 1, \cdots, N(n, p))$, where $N(n, p) := 2\sum_{i=0}^{p-1}\binom{q-1}{i}$ (see Winder(1966)) and $q := N^p_n(\{\mathbf{t}_i\})$ is the total number of distinct $(p-1)$-dimensional hyperplanes formed by points from $\{\mathbf{t}_i\}$. $q \le \binom{n}{p}$. Assume $q > 1$. When $\{\mathbf{t}_i\}$ are in a general position (IGP) (see Z19a for definition), $q = \binom{n}{p}$. In the latter case, $N(n, p) = O(n^{p(p-1)})$, lower than the cost $O(n^{n+1/2})$ above if $n \ge p(p-1)$.

Each P_k $(k = 1, \cdots, N(n, p))$ corresponds to a unique permutation $\{i_1, \cdots, i_n\}$, that is, $1/\mathbf{t}'_{i_1}\mathbf{v}_0 \le 1/\mathbf{t}'_{i_2}\mathbf{v}_0 \le \cdots \le 1/\mathbf{t}'_{i_n}\mathbf{v}_0$, $\forall\ \mathbf{v}_0 \in P_k$. The latter in turn corresponds to a polyhedral cone which is determined by

$$\mathbf{B}'\mathbf{v} \le \mathbf{0}_{(n-1)\times 1}, \tag{13}$$



(a) odd-n median sequence demonstration (b) even-n median sequence demonstration

Figure 3: Median-sequence demonstration. (a) Five sample points are labeled as "1",...,"5". Line "14" cuts the space into two halfspaces. Focusing on the upper right one suffices. Label "ij" means that the labeled ray is perpendicular to the line segment connecting i and j. When $\mathbf{v}$ rotates within the angular region formed by "ij" and "ik" (or "kj", or "jk"), the median of the projected values is the projected value of the repeated label (point) i (or j, or k). The median sequence is "14", "13", "35", "25", "24" (and "14"). (b) Four sample points are labeled as "1",..., "4". Line "34" cuts the space into two halfspaces, focusing on the lower right one suffices. Along each ray, there are two permutations listed (as in (a)), due to the overlaps of the projected values of some of the two points. The labels of the common middle two points in the permutations help to identify the median sequence "34", "23", "14", "12" (and "34") which form 4 regions corresponding to two middle point pairs "4-2" (formed by "34" (upward), "23" and O), "4-3", "1-3", and "3-2".

where $\mathbf{v} \in \mathbb{S}^{p-1}$ and $B = (B_1, \cdots, B_{n-1})_{p\times(n-1)}$, $B_j := \mathbf{t}_{i_j} - \mathbf{t}_{i_{j+1}}$, $j = 1, \cdots, N^-_{\mathbf{v}_0}$; $B_j := -(\mathbf{t}_{i_j} - \mathbf{t}_{i_{j+1}})$, $j = N^-_{\mathbf{v}_0} + 1, \cdots, (n-1)$, with the vector inequality in the coordinate-wise sense.

By Proposition 2.2, the entire P_k belongs to only one $\mathcal{S}_i$. So as long as we have one $\mathbf{v}_0$ from each P_k, we can easily produce the permutation associated with the P_k and the induced $k_i^{\mathbf{v_0}}$ and determine which $\mathcal{S}_i$ and formulae should be used in Proposition 2.1. Coupled with the constraints $B'\mathbf{v} \leq \mathbf{0}_{(n-1)\times 1}$ above, both Type I and Type II optimization problems in the Proposition could be solved in linear time (note that $\mathbf{b}, \mathbf{c}$ and $\mathbf{A}$ are constants over the entire piece of P_k). The exact computation of the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ can be achieved with the worst-case time complexity of order $TC(n, p, N_{iter}) := O(N(n,p)(p^3 + n\log n + np^{1.5} + npN_{iter})$, where N_{iter} is the number of iterations needed when solving the type II optimization problem.

Theorem 2.1 Under **(A1)-(A2)**, for a given sample $\mathbf{Z}^{(n)}$ and a $\boldsymbol{\beta}$ in $\mathbb{R}^p$, the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ (or the PRD$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$) can be computed exactly with the worst-case complexity of (i) $O(n^2\log(n))$ for $p = 2$; (ii) $TC(n, p, N_{iter})$ for $p > 2$.

Proof: For simplicity of description, we assume that $N^p_n(\{\mathbf{t}_i\}) = \binom{n}{p}$. The general case (with a smaller $N^p_n(\{\mathbf{t}_i\})$) could be treated similarly.

(i) Consider **the case** $p = 2$. That is, the $\mathbf{t}_i$ are bivariate points. We show that we can divide the entire circle $\|\mathbf{v}\| = 1$ into $O(n)$ pieces (arcs) using the so-called median sequence (Zuo and Lai (2011)). These $O(n)$ pieces of arcs further help to divide the entire unit disk into $O(n)$ pieces (each formed by the origin, two radii and a piece of arc) (see (b) of Fig. 2). Over each piece, the middle two numbers (see (b) of Fig. 3) (or one in the odd n case, see (a) of Fig. 3) of the projected values $\mathbf{t}'_i\mathbf{v}$ are the projected values of some two (or one) fixed points (or point) from $\{\mathbf{t}_i\}$.

In (a) of Fig. 3, when $\mathbf{v}$ rotates over the angular region formed by O, if we consider rays labeled as "ij" and "ik" (or "kj", or "jk") then the point labeled as "i" (or "j") (i.e. the common label) is the single point whose projected value will always be the median of the projected values. The median sequence is the rays "14" (up), "13", "35", "25", "24", "14" (down) which form 5 angular regions corresponding to point "1" (formed by "14" (up), "13",and O) "3", "5","2", and "4"; whereas in (b) of Fig. 3, along various rays labeled as "ij", there are permutations listed (also in (a)). Each ray corresponds to two equivalent permutations, because along each direction (ray), the projection of some two points overlap. These permutations help to identify the middle two points and the median sequence. The median sequence is the rays "34" (up), "23", "14", "12", "34" (down) which form 4 regions corresponding to two-point pairs "4-2" (formed by "34" (upward) "23" and O), "4-3", "1-3","3-2". When $\mathbf{v}$ rotates over the angular region formed by O, 23, and 34 (up); the points "4" and "2" are the two points whose projected values are the middle two of all projected values (they appear in the middle of the permutations along the rays "34" (up),"24" "23").

Figure 3 just illustrates a general phenomenon with concrete examples. We have generally

Lemma 2.1: (i) For $p = 2$, there are $O(n)$ rays that divide the unit disk into $O(n)$ pieces (cones, or angular regions) A_j, each with the origin as its vertex. Over A_j, the median of the projected values $\{\mathbf{t}'_i\mathbf{v}\}$ is the projected values of some two (or one in the odd n case) fixed points $\mathbf{t}_{j_1}$ and $\mathbf{t}_{j_2}$. (ii) the UF$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ and the PRD$(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ can be computed exactly in $O(n^2 \log n)$.

Proof:

For simplicity, label sample points $\{\mathbf{t}_i\}$ as $1, 2, \cdots, n$. For each i there are $j_1, \cdots, j_{i_k}$ labels (or points), such that the line connecting i to j_m ($1 \le m \le i_k$), labeled as "ij_m", cuts the plane into two closed halfplanes so that each contains no less than $\lfloor (n+1)/2 \rfloor$ points. In Fig. 3, i_k is 2 (for odd n) and 3 (for even n). But they could be larger in other cases.

Identify the unit vector over the unit circle that is perpendicular to the line ij_m by its polar coordinate angle θ_{ij_m} ($0 \le \theta_{ij_m} \le \pi$) (only halfplane suffices). For each i, keep the two unit vectors that have the minimum and maximum polar angles, respectively. In total, there are $O(n)$ such unit vectors. These $O(n)$ vectors cut the unit disk into $O(n)$ angular regions each formed by the origin and two unit vectors. By the construction (also see Fig. 3), it is readily seen that over each angular region A_j, the middle two (or one in odd n case, skip mentioning this case hereafter) integers of the permutations are the same. When $\mathbf{v}$ rotates over each region A_j the middle two of the projected values $u_i^{\mathbf{v}}$ are the projected values of some two fixed points (say, $\mathbf{t}_{j_1}$, $\mathbf{t}_{j_2}$). This completes the proof of the first part of the Lemma.

Over each piece A_j (in total $O(n)$ pieces), invoking Proposition 2.1 and optimization programming (considering the boundary directions suffices), the job can be done with the cost of $O(n^2)$. However, to find out the boundaries of the $O(n)$ pieces, it costs $O(n^2 \log n)$. Thus, we have the second part of the Lemma. ■

(ii) Consider **the cases** $p > 2$.

Before proving this case, we introduce some basic concepts about a convex body. For more details, refer to Fukuda (2004). A hyperplane H of $\mathbb{R}^p$ is *supporting* P (a p-polyhedron or p-polytope) if one of the two closed halfspaces of H contains P. A subset F of P is called a *face* of P if it is either $\varnothing$, P itself, or the intersection of P with a supporting hyperplane. The faces of dimension $0, 1, \dim(P) - 2$ and $\dim(P) - 1$ are called the *vertices*, *edges*, *ridges* and *facets*, respectively.

$$\begin{aligned}
\text{A hyperplane } H &= \{\mathbf{x} \in \mathbb{R}^p \ | \mathbf{a}'\mathbf{x} = c\}, \mathbf{a} \in \mathbb{R}^p \setminus \{\mathbf{0}\}, c \in \mathbb{R}^1, \\
\text{A closed halfspace } H &= \{\mathbf{x} \in \mathbb{R}^p \ | \mathbf{a}'\mathbf{x} \le c\}, \\
\text{A polyhedron } P &= \{\mathbf{x} \in \mathbb{R}^p \ | A\mathbf{x} \le \mathbf{b}\}, A \in \mathbb{R}^{m \times p}, \mathbf{b} \in \mathbb{R}^m, \\
\text{A Polytope } P &= \{\mathbf{x} \in \mathbb{R}^p \ | A\mathbf{x} \le \mathbf{b}, \mathbf{I} \le \mathbf{x} \le \mathbf{u}\}, \mathbf{I}, \mathbf{u} \in \mathbb{R}^p, \\
\text{A polyhedral cone } P &= \{\mathbf{x} \in \mathbb{R}^p \ | A\mathbf{x} \le \mathbf{0}\},
\end{aligned}$$

Obviously, exact computation is achieved if we can obtain the RHS of display (12). For the latter, we appeal to Proposition 2.1. To implement the proposition, we need to solve the two types of optimization problem (see Section 2.4 for implementation).

By the discussion immediately before the theorem, we know the key for the optimization problems is to identify all pieces P_k $(k = 1, \cdots, N(n,p))$ of $\mathbb{S}^{p-1}$. Equivalently, we need to identify all $N(n,p)$ distinct permutations of $\{1, 2, \cdots, n\}$. The latter is equivalent to finding a unit vector $\mathbf{u} \in P_k$ for each P_k which can produce the unique fixed permutation over P_k.

Each P_k is the intersection of $\mathbb{S}^{p-1}$ and the polyhedron cone formed by the constraint $B'\mathbf{v} \le \mathbf{0}_{(n-1)\times 1}$ in (13). The edge (or ridge) of the cone can be used to find the $\mathbf{u}$ above, which is shared by another adjacent cone. In other words, the edge (or ridge) is the intersection of (at least) two hyperplanes H_0s which go through the origin and are perpendicular to two hyperplanes H_1s, each of which is formed by points from $\{\mathbf{t}_i\}$, respectively.

The direction from the origin to any other point on the intersection hyperline of two hyperplanes H_0s is a solution of the vector sought above. We denote the direction by $\mathbf{u}$ ($\mathbf{u}$ could also be obtained more costly via the origin and any vertex of a polytope through vertex enumeration, see Bremner et al., 1998, Paindaveine and Śiman (2012) and Liu and Zuo (2014)).

Each $\mathbf{u}$ lies on the boundary of P_k. It not only lies in the facet of one cone but also lies in that of an adjacent cone which shares the common intersection hyperline (edge or ridge) with the former cone (cf. Fig. 1 of Mosler et al (2009)). A tiny perturbation of $\mathbf{u}$ in opposite directions will cause $\mathbf{u}$ to enter the interiors of the two adjacent cones. There might be more than two cones that are adjacent. Thus, every $\mathbf{u}$ might yield two or more new permutations (the scheme in the algorithm yields up to $8 \times (p-2)$ distinct ones, $p > 2$).

Update the total number N_{permu} of distinct permutations. With respect to each distinct permutation, or equivalent over each P_k, update $\sup_{v \in \mathbb{S}^{p-1}} |g(\mathbf{v})|$ according to Proposition 2.1 and carry out one of the two types of optimization.

Repeat above steps (find more $\mathbf{u}s$) until $N_{permu} = N(n,p)$ or the UF can not be improved after trying κp more distinct permutations (κ is a positive integer, which could be something like 30).

For a given data set (or $\{\mathbf{t}_i\}$), the total number of H_1s is fixed, but for each H_1, there are infinitely many H_0s ($p > 2$) which go through the origin and are perpendicular to H_1. So by utilizing different H_0s one can always obtain all distinct permutations in theory. If one obtains $N(n,p)$ distinct permutations which means that each piece of P_k has been visited (or all relevant directions $\mathbf{u}$s have been obtained), the resulting UF (or PRD) is exact in theory. In practice, however, not every distinct permutation updates the UF. In the latter case, the stopping rule "until UF can not be improved" becomes handy.

The cost of computation of key elements of the descriptions above is as follows.

(a) calculating all $\{\mathbf{t}_i\}$ (assume they are in a general position) and $N(n,p)$ costs $O(np)$,

(b) calculating normal vectors $\mathbf{v}_i$ of H_1^i, normal vectors $\mathbf{u}_i$ of H_0^i $(i = 1, 2)$, and $\mathbf{u}$ (which is perpendicular to $\mathbf{u}_i$) costs $O(p^3)$,

(c) producing each permutation costs $O(n(p + \log n))$,

(d) updating the total number of distinct permutations cost $O(nN_{permu})$,

(e) updating $\sup_{v \in \mathbb{S}^{p-1}} |g(\mathbf{v})|$ according to Proposition 2.1 costs

  (i) $O(n)$ for obtaining $k_i^{\mathbf{v}}$ $(i = n1, n2)$, $\mathbf{t}_{i_{n1}}$, and $\mathbf{t}_{i_{n2}}$,

(ii) $O(p^{1.5}n + p^{2.5})$ for linear programming (see Yin Tat Lee and Aaron Sidford (2015) which is even further improved by Cohen, Lee, and Song (2019)),

(iii) $O(npN_{iter})$ for the type II non-convex and nonlinear optimization problem. One can use the conjugate gradient method or the even better primal-dual interior-point method (Wright (1997), Morales, et al (2003)) combined with the sequential quadratic programming (SQP) (Nocedal and Wright (2006)), e.g. package LOQO (Vanderbei and Shanno (1999) and Vanderbei (1999)). Where N_{iter} is the number of iterations needed in LOQO.

Keeping only the dominating terms, we have the overall worst-case time complexity $TC(n, p, N_{iter}) = O(N(n,p)(p^3 + n \log n + np^{1.5} + npN_{iter}))$.

This completes the proof of the theorem. ■

Pseudocode (Exact computation of the UF($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$), or equivalently of the PRD($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$))

```
• Input: Given a sample $\mathbf{Z}^{(n)} := \{(\mathbf{x}'_i, y_i)',\ i = 1, \cdots, n\}$ and a $\boldsymbol{\beta}$ in $\mathbb{R}^p$,
• Calculate $\{\mathbf{t}_i\}$ (assume they are in a general position) and $N(n,p)$; set UF= $N_{permu} = 0$.
• While $(N_{permu} < N(n,p))$
    1 Obtain $\mathbf{u}$ and its induced permutations, store distinct permutations and update its total number $N_{permu}$.
    2 Update UF= $\sup_{\mathbf{v} \in \mathbb{S}^{p-1}} |g(\mathbf{v})|$ via Proposition 2.1 and carry out the corresponding optimization for each distinct permutation.
    3 If UF can not be improved after trying $\kappa p$ more distinct permutations ($\kappa = 30$), break the loop.
• Output: UF (or $1/(1+\text{UF})$) of the $\boldsymbol{\beta}$ with respect to $F^n_{\mathbf{Z}}$. ■
```

2.2.2. Exact computation algorithms

(I) Algorithm for the exact computation of the UF($\beta; F^n_{\mathbf{Z}}$) and the PRD($\beta; F^n_{\mathbf{Z}}$) in $\mathbb{R}^2$

Before listing the key steps of the algorithm, we make some comments.

(i) Directions that are perpendicular to the line segment connecting $\mathbf{t}_i$ and $\mathbf{t}_j$ could be the boundary of angular regions, so we will have to include them in our calculation.

(ii) $\sup_{\mathbf{v} \in \mathcal{S}_i} |g(\mathbf{v})|$ $(i = 1, 2)$ can be obtained along the median sequence and the directions given in (i) above.

Exact Algorithm EA-UF2D

Input a $\boldsymbol{\beta}$ and n data points $\mathbf{Z}^{(n)} = \{(\mathbf{x}_i, y_i)'\}$ in $\mathbb{R}^2$; **Output** UF($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$) and PRD($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$).

Initial Step: (i) Obtain $N := N^p_n(\{\mathbf{t}_m\})$ unit vectors $\mathbf{u}_k(i,j)$ that are perpendicular to the all possible N hyperplanes formed by $\mathbf{t}_i$ and $\mathbf{t}_j$ from $\{\mathbf{t}_m\}$, $k = 1, \cdots, N$. (ii) Sort $\mathbf{u}_k(i,j)$ according their polar angles such that $\alpha_{i_1} \le \alpha_{i_2} \le \cdots, \le \alpha_{i_N}$. (iii) Record the pair (i,j) associated with α_k as the pair (i^k, j^k).

Set a seven-component initial matrix I_0 with initial values corresponding to the seven cases in Proposition 2.1. Let $k = 0, M_k = I_0$, $\mathbf{u}_k = (1,0)'$ and $i_1, \cdots i_n$ be a permutation induced by $\mathbf{u}_k$. If $i \ne j$ and $\mathbf{t}'_i\mathbf{u}_k = \mathbf{t}'_j\mathbf{u}_k = \mathbf{t}'_{i_{kk}}\mathbf{u}_k$, set $m^1_k = i^k = i$ and $m^2_k = j^k = j$, else $m^1_k = m^2_k = i^k = j^k = i_{kk}$, where $kk = \lfloor (n+1)/2 \rfloor$.

Loop step: While ($k <= N+1$) { Let $\mathbf{v} = \mathbf{u}_k$. Update M_k according to Corollary 2.1. Let $k = k+1$, $\mathbf{v} = (cos(\alpha_k), sin(\alpha_k))$. If the set $S^m_{k-1} := \{m^1_{k-1}, m^2_{k-1}\}$ intersects with the set $S_k := \{i^k, j^k\}$, then let $m^1_k = i^k, m^2_k = j^k$; otherwise, let $k = k+1$, $m^i_k = m^i_{k-1}$, $i \in \{1,2\}$. }

For (i in 1:n) { get $\mathbf{v}_i$ that is perpendicular to $\mathbf{t}_i$, using $\mathbf{v}_i$ to update M_{N+i-1} according to Proposition 2.1 and to obtain M_{N+i}.}

Final step: Set the maximum no-zero element of $M_{N+n} - I_0$ be the UF($\boldsymbol{\beta}, F^n_{\mathbf{Z}}$).

(II) Algorithm for the exact computation of UF($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$) and PRD($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$) in $\mathbb{R}^p$, $p > 2$

Exact Algorithm (EA-UFHD)

Input a $\boldsymbol{\beta}$ and n data points $\mathbf{Z}^{(n)} = \{(\mathbf{x}'_i, y_i)'\}$ in $\mathbb{R}^p$; **Output** UF($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$) and PRD($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$).

(a) Compute $N(n,p)$ and call it by N, let $k_{pm} = 0$. ($O(p)$, assume that $q = \binom{n}{p}$).

(b) Construct two non-parallel hyperplanes H_i ($i = 1,2$) (each of which is formed by p points from $\{\mathbf{t}_i\}$) with normal vectors $\mathbf{v}_i$, $i = 1,2$, respectively.
($O(pn + p^3)$)
Find two hyperplanes $H^{\perp}_i$ that are through the origin and perpendicular to H_i and with normal vectors $\mathbf{u}_i$, $i = 1,2$, respectively. ($O(p^3)$)
Let $\mathbf{U}$ be the unit vectors matrix each of its $(p-2)$ columns is perpendicular to both of $\mathbf{u}_1$ and $\mathbf{u}_2$. $O(p)$

(c) For each column vector of the $\mathbf{U}$ above, call it $\mathbf{u}$, introduce eight vectors $\pm\mathbf{u} \pm \mathbf{v}_1 \pm \mathbf{v}_2$. For each of eight vectors, obtain its induced permutation, and store it in a matrix if it is a new one. Update the total number of distinct permutations k_{pm}. $O(n(p + \log n + k_{pm}))$
For each distinct new permutation and the associated vector $\mathbf{v}$, carry out the optimization to update $|g(\mathbf{v})|$ via Proposition 2.1. Also use $\mathbf{v}_1$ and $\mathbf{v}_2$ to update $|g(\mathbf{v})|$ via Proposition 2.1. ($O((n + \max\{np^{1.5} + p^{2.5}, npN_{iter}\}))$

(d) while ($k_{pm} < N$) { do (b), (c) above until either $k_{pm} = N$ or UF cannot be improved.} ($O(N(p^3 + (\max\{p^{2.5} + np^{1.5}, npN_{iter}\}) + n(p + \log n)))))$

(e) Output the final UF($\boldsymbol{\beta}; F^n_{\mathbf{Z}}$) via Proposition 2.1 and (12). ($O(1)$)

Overall cost in the worst case is $O(N(n,p)(p^3 + n\log n + np^{1.5} + npN_{iter}))$.

Remarks 2.2

(I) EA-UFHD exactly follows the idea given in the proof of Theorem 2.1. Since there are infinitely many $H^{\perp}_i$s (that go through the origin) for each H_i, in theory one can get all distinct permutations, equivalently all desirable unit directions $\mathbf{u}$, and the final UF (or PRD) obtained is in theory exact.

(II) In the best scenario, $N(n,p)$ can be replaced by $O(n^2)$. Even in this case, the cost of exact computation in the worst case is $O(n^2(p^3 + n\log n + np^{1.5} + npN_{iter})$, which is still unaffordable for large n and/or p. An approximate algorithm, such as **AA-UF-3** (introduced below) with cost of order $O(N(np + p^3))$, where tuning parameter N being the total number of normal directions of the hyperplanes formed by p points from $\{\mathbf{t}_i\}$, is more feasible in practice.

(III) On the other hand, besides theoretical interest itself, without the slow exact algorithm as the benchmark, no one can develop fast practically feasible approximate algorithm with known acceptable accuracy for large n and p. The contribution and importance of the exact algorithm can never be over-emphasized. ■

However, exact computation of the $\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ or the PRD $(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ is not our primary goal. Our ultimate goal is to seek depth induced median or other estimators. Practically, the latter has to be computed approximately. In the following, we discuss some more practically feasible approximate algorithms.

2.3. Approximate computation

Approximate computation of statistical depth functions is common and has been carried out in Rousseeuw and Struyf (1998), Dyckerhoff (2004), Cuesta-Albertos and Nieto-Reyes (2008), Chen et al. (2013), and Zuo (2018) and in the references cited therein.

Here we present three approximate algorithms. The first one is a straightforward naive one. It randomly selects a fixed number N directions from a distribution (e.g. uniform on the hypersphere), and decorrelating the data before calculation the $\mathrm{UF}(\boldsymbol{\beta}; F^n)$ defined in (9) along those directions.

Approximate algorithm AA-UF-1

Input a $\boldsymbol{\beta}$ and n data points $\mathbf{Z}^{(n)} = \{(\mathbf{x}'_i, y_i)'\}$ in $\mathbb{R}^p$; **Output** $\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ and $\mathrm{PRD}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$.

(a) Randomly select N unit directions $\mathbf{v} \in \mathbb{S}^{p-1}$ according to a uniform distribution on the hypersphere, use the formula given in (9) or (8) to calculate/update $\sup_{\mathbf{v}\in\mathbb{S}^{p-1}} |g(\mathbf{v})|$.

(Overall cost is $O(npN)$, the cost to find median can be as low as $O(n)$). ■

The second approximate algorithm below employees the idea in EA-UFHD. It considers the directions that represent the edges of the convex cones, where the cones stem from the origin and partition the entire sphere $\mathbb{S}^{p-1}$ into disjoint (convex) pieces.

When $\mathbf{v}$ moves over each piece, the permutation induced is fixed. By the fundamental theorem of linear programming, the solution of the maxima or minima of a linear function over a convex polygonal region occurs at the region's corners. (Note that we no longer have linear functions in the **Type II** optimization scenario).

Approximate algorithm AA-UF-2

Input a $\boldsymbol{\beta}$ and n data points $\mathbf{Z}^{(n)} = \{(\mathbf{x}'_i, y_i)'\}$ in $\mathbb{R}^p$; **Output** $\mathrm{UF}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$ and $\mathrm{PRD}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$.

(a) Compute the $\mathbf{t}_i$, $i = 1, \cdots, n$. (total cost $O(np)$)

(b) Sample two sets P_i, each with p points, from $\{\mathbf{t}_i\}$. Construct two hyperplanes H_i with normal vectors $\mathbf{v}_i$, uniquely determined by P_i, respectively. Try different P_2 until $\mathbf{v}_2$ is not parallel to $\mathbf{v}_1$. (total cost $(O(p^3))$

(c) Construct two hyperplanes $H_i^{\perp}$ (with normal vectors $\mathbf{u_i}$) that is through the origin and perpendicular to H_i, respectively. (total cost $(O(p^3)))$

(d) Obtain $\mathbf{v} = \mathbf{u}_1 \times \mathbf{u}_2$ and $\mathbf{v}_0 = \mathbf{v}/\|\mathbf{v}\|$; use $\mathbf{v}_0$ and the formula in (9) to update $\sup_{\mathbf{v}\in\mathbb{S}^{p-1}} |g(\mathbf{v})|$. (total cost $O(np)$)

(e) Repeat (b)-(d) N times. (total cost $O(N(np + 2p^3)))$. Overall cost is $O(N(np + 2p^3) + np))$. ■

The one below uses N normal vectors of the hyperplanes determined by p points from $\{\mathbf{t}_i\}$.

Approximate algorithm AA-UF-3

Input a $\boldsymbol{\beta}$ and n data points $\mathbf{Z}^{(n)} = \{(\mathbf{x}'_i, y_i)'\}$ in $\mathbb{R}^p$; **Output** $\mathrm{UF}(\beta; F^n_{\mathbf{Z}})$ and $\mathrm{PRD}(\boldsymbol{\beta}; F^n_{\mathbf{Z}})$.

(a) Compute the $\mathbf{t}_i$, $i = 1, \cdots, n$. (total cost $O(np)$)

(b) Sample p points from $\{\mathbf{t}_i\}$, find the normal vector $\mathbf{v}$ of the hyperplane determined by them. Along $\mathbf{v}$, use the formula (9) to calculate/update $\sup_{\mathbf{v} \in \mathbb{S}^{p-1}} |g(\mathbf{v})|$. (total cost $O(p^3 + np)$)

(c) Repeat (b) N times. (total cost $O(N(p^3 + np))$). Overall cost $O(N(np + p^3) + np)$). ■

2.4. Examples

To better understand the algorithms in the last two subsections, we present some examples. Algorithms for the deepest regression lines discussed in Section 3 are also employed below.

For the exact algorithm, we now explain the implementation of the two types of optimization.

Given a direction $\mathbf{v} \in P_k$, a permutation, say, $i_1, \cdots, i_n$, is obtained. That is, for all the values from $\{k_i^{\mathbf{v}} = 1/\mathbf{t}_i'\mathbf{v}\}$, we have $k_{i_1}^{\mathbf{v}} \le k_{i_2}^{\mathbf{v}} \le \cdots \le k_{i_n}^{\mathbf{v}}$, $\forall\ \mathbf{v} \in P_k$. The **Type I** optimization problem can be described as

minimize: $\mathbf{c}'\mathbf{v}$,

subject to: (i) $\mathbf{B}'\mathbf{v} \le \mathbf{0}_{(n-1)\times 1}$; (ii) $\mathbf{v}'\mathbf{v} = 1$,

where $\mathbf{c}$ is a constant vector, $\mathbf{B}$ is a constant matrix, and minimization could also be exchanged for maximization. That is, we have a linear objective function with a linear inequality constraint and a quadratic equality constraint.

When $p = 2$, each P_k becomes a piece of an arc of the unit circle and the cones formed by the linear constraints are the angular regions with two radii as their boundaries. The optimization problem becomes linear programming over the piece of arc. By the fundamental theory of linear programming, the minimization or maximization occurs only at the boundary. Therefore, only evaluation of $\mathbf{c}'\mathbf{v}$ is needed for $\mathbf{v}$ at the two boundary directions. There are at most $O(n^2)$ pieces of P_k's.

Generally, the **Type I** optimization problem can be solved by an augmented Lagrangian minimization using the R package 'alabama', or by sequential quadratic programming using the R solver 'slsqp'. Alternatively, it can be transformed into semidefinite programming problems and solved using the R solver 'csdp'. Also the R packages 'optisolve' and 'nlopt' are applicable.

Now we turn to the **Type II** optimization problem. It can be described as

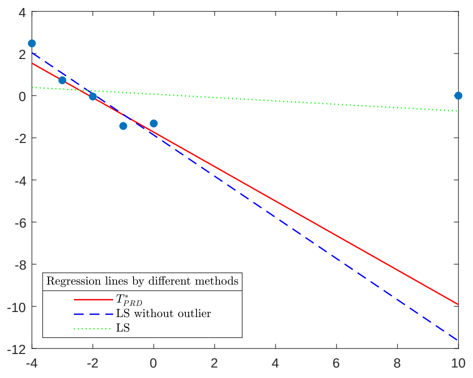
minimize: $\frac{\mathbf{b}'\mathbf{v}}{\mathbf{v}'\mathbf{A}\mathbf{v}}$,

subject to: (i) $\mathbf{B}'\mathbf{v} \le \mathbf{0}_{(n-1)\times 1}$; (ii) $\mathbf{v}'\mathbf{v} = 1$,

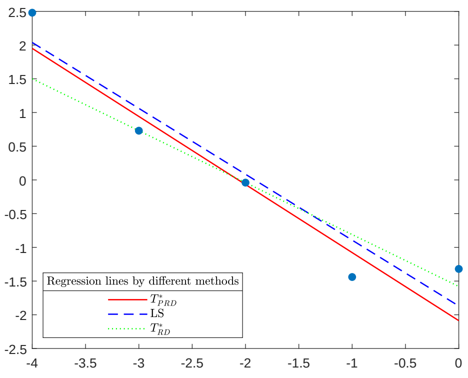
where $\mathbf{b}$ is a constant vector, $\mathbf{A}$ and $\mathbf{B}$ are constant matrices, $\mathbf{A}$ can be treated as a symmetric and positive definite one, and minimization can also be exchanged for maximization.

That is, we have a non-linear, non-convex, but differentiable objective function or a rational objective function with a linear inequality constraint and a quadratic equality constraint. The problem again can be solved by using the R packages 'alabama', 'optisolve', and 'nlopt'.

Example 2.4.1 Performance of exact and approximate algorithm. Here we examine the performance of the exact versus the approximate algorithm (EA-UF2D v.s. AA-UF-1) for computing the UF, w.r.t. their accuracy, speed, and estimated mean squared errors.



(a) three lines and six points



(b) three lines and five points

Figure 4: (a) Solid red: the deepest regression line induced from the PRD. Dashed blue: the least squares line based on five points without the horizontal outlier. Dotted green: the least squares line based on all six points. (b) Solid red: the deepest regression line induced from the PRD w.r.t five points. Dashed blue: the least squares line based on five points without the horizontal outlier. Dotted green: the deepest regression line induced from the RD of RH99 w.r.t. five points.

For illustration purposes, we utilize the data set given in Huber and Ronchetti (2009) Exhibit 7.1. Three regression lines are obtained w.r.t. the data. The first line T^*_{PRD} is the maximum PRD line ($\boldsymbol{\beta_1} = (-1.7317456, -0.8184845)'$ in (intercept, slope)' format); the second one (LS without outlier) is the least squares line *without* the horizontal outlier ($\boldsymbol{\beta_2} = (-1.87, -0.977)'$); the third one (LS) is the least squares line ($\boldsymbol{\beta_3} = (0.07, -0.08)'$) w.r.t. all points. See (a) of Figure 4.

Next, we calculate the unfitness of the three lines ($\boldsymbol{\beta' s}$). First, the unfitness, reported in Table 1, is calculated without the horizontal outlier for the fairness of the comparison between exact and approximate algorithms. That is, they are calculated w.r.t. just five points.

Second, using all six points, the results are very similar to those in Table 1 and details are omitted. As an example, the UF from the EA are $0.8340, 1.2113$, and 2.3367 and the UF from the AA (mean of 1000 replications) are 0.5246, 0.6855 and 2.1846, respectively.

Consistent with expectations, $\boldsymbol{\beta}_1$ (T^*_{PRD}) has the lowest UF, $\boldsymbol{\beta}_2$ has the second lowest UF, and the $\boldsymbol{\beta}_3$ has the highest UF. That is, *in terms of the UF ordering*, T^*_{PRD} is the best choice among the three while $\boldsymbol{\beta}_3$ is the worst, fitting with the intuitive comprehension of (a) of Figure 4.

At the same time, it is not difficult to determine the regression depths (RD) of RH99 of the three lines, they are 2/6, 1/6, 1/6, respectively. (For simple methods of calculation of the RD, see RH99 or RS98). That is, *in terms of the RD ordering*, the least square line $\boldsymbol{\beta}_3$ is as deep (or good) as the line $\boldsymbol{\beta}_2$, while both are less deep than the PRD induced line T^*_{PRD}, which is somewhat inconsistent with the intuitive comprehension of (a) in Figure 4. Of course, the comparison here is not very fair since the different methods (PRD vs RD) are based on different objective criteria and $\boldsymbol{\beta}_2$ and $\boldsymbol{\beta}_3$ use a different number of total points.

In (b) of Figure 4, all three lines are calculated w.r.t. just five points without the outlier. The solid red line is the deepest line induced from the PRD with $\boldsymbol{\beta_1} = (-2.083114, -1.009444)'$, the dashed blue line is the same as in (a), the dotted green line is the deepest line from the RD of RH99 with $\boldsymbol{\beta_3} = (-1.58, -0.77)'$. The RDs of the three lines are 2/5, 1/5, 3/5, respectively. This time, as expected from T^*_{RD}, the line induced from the RD becomes the deepest one.

In Table 1, the calculation of the approximate algorithm (AA) is repeated 1000 times to mitigate the randomness; the mean and the standard deviation of 1000 UFs are calculated. 1000 unit vectors are

Table entries (a,b,c,d) are a:= mean of UF, b:=standard deviation (sd) of UF, c:=time consumed (in seconds), d:=number of unit vectors used.

	$\boldsymbol{\beta}_1$ (line L_1)	$\boldsymbol{\beta}_2$ (line L_2)	$\boldsymbol{\beta}_3$ (line L_3)
EA	(0.59, 0, 1.11e-3, 13)	(0.87, 0, 1.28e-3, 15)	(2.88, 0, 1.30e-3, 15)
AA	(0.58, 4.1e-3, 1.657e-2, e+3)	(0.86, 4.8e-3, 1.662e-2, e+3)	(2.85, 3.4e-2, 1.664e-2, e+3)

Table 1: Performance of exact and approximate algorithms w.r.t. different $\boldsymbol{\beta}'s$ (lines).

used in the calculation per replication, and the time consumed per replication is reported in the table.

The table reveals that the exact algorithm (EA) is much faster than the AA. The former used no more than 15 unit vectors whereas the latter, using 1000 vectors, still returned a smaller (under-approximated) UF than the exact one. This is the beauty of the EA. If the number of unit vectors used is increased to 10^4, then the UF from the AA is still smaller than the one from the EA which just employed 13 unit vectors, in the $\boldsymbol{\beta_1}$ case. Since there is no fluctuation in the EA, all the sd's are zero. The time reported is the average of per replication from 1000 replications. Observations above hold only in this special ($n = 5$ and $p = 2$) example. Note that when n and/or p increase, the EA is no longer feasible in practice. ■

Results above and below in this section are all based on Matlab code running on a desktop: Intel(R)Core(TM) i7-2600 CPU @ 3.40GHz. All Matlab as well as R and C++ code in this article are downloadable via https://github.com/zuo-github/comp-prd-medians.

Example 2.4.2 Performance of exact algorithm versus approximate algorithms

In the last example, the dimension $p = 2$ and $n = 5$ are too limited in practice. In this example, we consider standard normal random points with $p = 3$ and $n = 10$, For this fixed small data set, we employ the EA and the AAs to compare their performance. Here we seek the best output from them, that is the largest UF (note that the UF is defined based on the supremum of univariate unfitness; so generally speaking, the larger the better).

For the EA, the stopping rule of the algorithm is the total number of distinct permutations (N_{perm}) used by it whereas for the AAs, the stopping rule is the total number of unit directions ($N_{\mathbf{v}}$) employed by the AAs. The outputs from different algorithms are listed in the table below. Column one is the number of distinct permutations used by the EA-UFHD, the second column is the number of while loops that need to be executed to get the desired total distinct permutations. The fourth column is the total number of directions used by the AAs.

Inspecting the table immediately reveals that (i) in order to use 120 distinct permutations, one needs to run the while loop 18 times in the EA and can get 4.3505 for the UF. Note that each single while loop yields a p by $(p-2)$ unit vector matrix $\mathbf{u}$ (in this case a single column vector) and each column of $\mathbf{u}$ will induce eight unit vectors (see (c) of the EA-UFHD), these eight vectors will lead to at most eight distinct permutations in each while loop. This explains why only 18 while loops are needed. The number 120 (equal to $\binom{10}{3}$)) in this case is exactly the total possible planes formed by any three sample points from a data set (IGP) of size 10 in a three dimensional space. (ii) if one uses 400 distinct permutations, then 92 while loops are needed and one can get the 8.2669 for the UF, which is the final answer for the exact UF. (iii) for the AAs, the AA-UF-3 yields the same UF for all number of directions considered, as long as the latter is no less than 120. The meaning of the latter number is explained above. This phenomenon is not surprising and it is due to the design of the algorithm which only utilizes the normal vectors of all possible planes formed by three points, the latter is a fixed number 120 in this case. (iv) the AA-UF-1 yields larger UF than that of AA-UF-2 in

EA-UFHD				AAs		
N_{permu} used	$N_{while-loop}$ executed	UF	$N_{\mathbf{v}}$ used	AA-UF-1 UF	AA-UF-2 UF	AA-UF-3 UF
120	18	4.3505	120	4.0414	4.3862	5.4190
200	35	5.1988	1,000	5.8134	5.2446	5.4190
300	61	6.3708	10,000	6.6369	6.4582	5.4190
400	92	8.2669	100,000	6.7699	6.6985	5.4190
500	136	8.2669	1,000,000	6.7924	6.8182	5.4190
600	187	8.2669	10,000,000	6.8930	6.8675	5.4190

Table 2: Performance comparison of exact versus approximate algorithms on a fixed standard normal data set with the size 3 by 10.

most cases. (v) most important, the AAs even after exhausting 10 million directions still can not get a comparable size of UF that the exact algorithm produces using just 400 distinct permutations.

Notice that in this example, the number of total possible distinct permutations is $N(n,p) = 2 * (\binom{119}{0} + \binom{119}{1} + \binom{119}{2}) = 14282$. However, 400 distinct permutations are enough for the exact UF. The superiority of the EA over the AAs is clearly demonstrated. In summary, the AAs can run much faster than the EA, but it is very difficult for them to get the results the EA provides (in fact, AA-UF-1 yields the UF 6.8930 even after depleting 10 million directions).

On the other hand, the EA is not feasible in practice for larger n and p, I would not recommend it for $p > 3$. Recall, the possible distinct permutations in $p = 3$ and $n = 10$ case above is 14282. In table 2, we used 400 permutations to get the exact result of the UF leading immediately to an issue. That is, in practice what should be the cut off number N for the distinct permutations? One suggestion is $N = \min\{\binom{n}{p} + 500, 1000\}$ (obviously, there is an alternative automatic stopping rule based on the UF) and the while loop should be executed at most 1000 times. ■

However, there are at least two drawbacks in the comparison above, (i) the data set is still small, (ii) all results are random (since all use random sampling) (which means perhaps 200 permutations are enough for the exact UF if the EA runs many times. Similarly, the AA using 1000 directions might get much better results if it runs many times). Each result in table 2 comes from a single run. Therefore the results and conclusions above have their limitations.

Example 2.4.3 Performance of exact algorithm versus approximate algorithms

In this example, we will consider the cases $p = 3, n = 50$ and $p = 4, n = 20$. Furthermore, we will run each of the algorithms 100 times (ideally it should be 1000 or even $10,000$ times, but due to the time consumed by the EA-UFHD, they are not affordable options). The q for the two cases are 19600 and 4845 and $N(n,p)$ are 384140402 and 37887089270, respectively.

Furthermore, we try to set up a fair comparison base for the two types of methods. We will use the number of directions employed by them as the criterion for a performance assessment. Note that for the EA, we count the number of directions it utilized as follows. In each while loop, there are 8*(p-2) directions induced by the $\mathbf{u}$ matrix; furthermore, there are $\mathbf{v}_1$ and $\mathbf{v}_2$ vectors, so in total $2+8*(p-2)$ unit vectors are employed by the EA.

dimension	$N_{\mathbf{v}}$ used	methods	UF(mean, min, max)	D_{max}	N_{nega}
$p = 3$	400	EA	(2.0797 1.8820 2.2315)	0	0
$n = 50$		AA1	(2.0242 1.8475 2.1118)	-0.1197	74.0000
		AA2	(1.9094 1.7287 2.0822)	-0.1493	95.0000
		AA3	(2.0800 1.9811 2.1335)	-0.0981	50.0000
	800	EA	(2.2673 2.0406 2.4942)	0	0
		AA1	(2.2217 2.0031 2.3756)	-0.1186	58.0000
		AA2	(2.1308 1.8936 2.3436)	-0.1506	82.0000
		AA3	(2.3037 2.1754 2.3762)	-0.1180	36.0000
$p = 4$	360	EA	(3.3443 2.7657 4.6322)	0	0
$n = 20$		AA1	(2.8832 2.4389 3.4670)	-1.1652	91.0000
		AA2	(2.5424 2.1256 3.1373)	-1.4949	98.0000
		AA3	(3.4406 3.1707 3.5357)	-1.0964	36.0000
	540	EA	(3.8939 2.8989 6.0121)	0	0
		AA1	(2.9247 2.3428 4.4621)	-1.5500	93.0000
		AA2	(2.1350 1.4178 3.2886)	-2.7235	100.0000
		AA3	(4.6775 4.1511 4.7682)	-1.2439	11.0000

Table 3: Performance of the EA and the AAs with respect to different data sets and directions used.

We (i) compute the mean, minimum (min) and maximum (max) of 100 UFs for each of the algorithms, and (ii) compute D_{max} := the difference in max UF (the max of UF of any algorithm subtracted by that of the EA) and (iii) count the times that the UF of any algorithm is less than that of the EA in the 100 trials (denote by N_{nega} := the count of the negative D_{uf}s, D_{uf} is defined as the UF of any algorithm subtracted by that of the EA for each of 100 trials).

Results are listed in table 3. Note that in the $p = 3$ case, 400 and 800 directions used by the AA amounts to 40 and 80 while loops executed in the EA-UFHD (we convert them to the number of while loops in EA to terminate the algorithm). In the $p = 4$ case, 360 and 540 directions for the AA amounts to 20 and 30 while loops executed in the EA.

Examining the table reveals that (i) in all cases considered, the EA produces the largest UF among the 100 trials, this is demonstrated by the max or D_{max} column in the table; (ii) the AA3 is the second best method among all four in the sense that (a) it yields the largest UF among the three AA methods in mean, minimum, and maximum of 100 UFs in all cases, and (b) its mean and minimum of 100 UFs are even larger than those of the EA in all cases considered (except the most important maximum of 100 UFs), and (iii) the last columns in table 3 shows the percentage that the EA yields a larger UF than that of any other AAs. For example, in the $p = 4$, $n = 20$, and $N_{\mathbf{v}} = 540$ case, the UF of the EA is larger than that of AA2 in every trial. ■

The examples above clearly demonstrate the advantage of the EA over the AAs. But as said before, the EA is not very feasible in practice due to its limitation on p and n. For example, it is not capable

Performance of approximate algorithms w.r.t. efficiency and accuracy

p	methods	mean UF	standard deviation	time consumed	number of $\mathbf{v}$ used
2	AA-UF-1	0.3035	0.1053	26.4750	10^3
	AA-UF-2	0.3041	0.1054	116.0503	10^3
	AA-UF-3	0.3042	0.1057	34.7244	10^3
5	AA-UF-1	0.4815	0.0994	31.1470	10^3
	AA-UF-2	0.4447	0.0995	227.3421	10^3
	AA-UF-3	0.5472	0.1095	80.3334	10^3
10	AA-UF-1	0.5198	0.0928	33.4609	10^3
	AA-UF-2	0.4467	0.1049	278.6988	10^3
	AA-UF-3	0.7385	0.1156	100.4025	10^3
20	AA-UF-1	0.5253	0.0877	40.5291	10^3
	AA-UF-2	0.4152	0.1054	570.2156	10^3
	AA-UF-3	1.1656	0.1626	236.5488	10^3

Table 4: Performance comparison of the three approximate algorithms.

of handling the case $p = 20$ and $n = 100$ and running 1000 times with the while loop executed 20 times in each trial. But the AAs are more feasible and can easily handle the situation. Furthermore, the depth induced median has to be computed eventually by approximate methods. The following example is devoted to the comparison of the three AAs.

Example 2.4.4 Performance comparison between three approximate algorithms

Here we generate $m = 1,000$ samples from the model: $y_i = \beta_0 + \beta_1 x_{i_1} + \cdots + \beta_{p-1} x_{i_{p-1}} + e_i, i = 1, 2, \cdots, n$, with sample sizes $n = 100$, where $e_i \sim N(0, 1)$. In light of the regression equivariance of the deepest projection depth estimator and the invariance of the PRD, we can assume without loss of generality (w.l.o.g.) that $\boldsymbol{\beta}_0 = (\beta_0, \beta_1, \cdots, \beta_{p-1})' = (0, 0, \cdots, 0)'$. We generate $\mathbf{Z}_i = (x_{i_1}, \cdots, x_{i_{p-1}}, y_i)$ from a p-dimensional standard normal distribution.

The three AAs compute the unfitness of $\boldsymbol{\beta}_0 = (0, 0, \cdots, 0)'$ with results (means and deviations of 1000 UFs, **total time** consumed (in seconds) for 1000 samples, and unit directions used) listed in Table 4 which features the attributes of the three AAs.

The table reveals that (i) the AA-UF-1 is fastest and the AA-UF-2 is slowest in all cases, confirming the theoretical time complexity results; (ii)the AA-UF-3 is the most accurate in all cases (with the largest mean UF), and the AA-UF-2 is superior over the AA-UF-1 *only for the case* $p = 2$ in terms of accuracy (mean is slightly larger); (iii) the AA-UF-1 is most efficient (smallest s.d. and fastest), and the AA-UF-3 has the largest s.d., but this could be reversed by increasing the number of directions $\mathbf{v}$ used to $\binom{n}{p}$; (vi) p has the least (no greater than linear) effect on the time consumed by AA-UF-1 and it reduces the s.d. of AA-UF-1 when it increases; (v) overall, the AA-UF-3 (or the AA-UF-1) is recommended. ■

3. Computation of the PRD induced median

3.1. Algorithms

Our fundamental goal for introducing the notion of depth in regression is not the depth itself but its induced median (and other estimators). Computation of the latter is our eventual objective. It seems that there is no analytical way to identify the depth induced median. Consequently the computation of the median in high dimensions has to be carried out approximately.

Before addressing the approximate computation of the maximum projection depth estimator (or median), we first show that it indeed deserves to be called a median since it recovers the univariate sample median when $p = 1$. Recall that (assume (w.l.o.g.) again $S_y = 1$)

$$\boldsymbol{\beta}^*_{PRD} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_i\{\frac{y_i - \mathbf{w}_i'\boldsymbol{\beta}}{\mathbf{w}_i'\mathbf{v}}\}\Big|. \tag{14}$$

When $p = 1$, it reduces to the following

$$\beta^*_{PRD} = \arg\min_{\beta\in\mathbb{R}} \sup_{v=\pm 1} \Big|\mathrm{Med}_i\{\frac{y_i - \beta}{v}\}\Big|. \tag{15}$$

We have

Proposition 3.1 When $p = 1$, the β^*_{PRD} recovers the regular sample median of $\{y_i\}$.

Proof: Let $y_{(1)} \le y_{(2)} \le \cdots \le y_{(n)}$ be the ordered values of $\{y_i\}$ and $\mu = (y_{(n1)} + y_{(n2)})/2$, where $n1$ and $n2$ are defined in Proposition 3.1, i.e., μ is the regular sample median of $\{y_i\}$. We show that $\beta^*_{PRD} = \mu$. It is readily seen that

$$\beta^*_{PRD} = \arg\min_{\beta\in\mathbb{R}} \big|\mathrm{Med}_i\{y_i - \beta\}\big| = \arg\min_{\beta\in\mathbb{R}} \big|\mathrm{Med}_i\{y_i\} - \beta\big| = \arg\min_{\beta\in\mathbb{R}} \big|\mu - \beta\big|, \tag{16}$$

where the first equality follows from (15) and the oddness of the median operator; the second one follows from the translation equivalence (see page 249 of RL87 for definition) of the median as a location estimator; the third one follows from the definition of μ.

The RHS of (16) above indicates that μ is the only solution for β^*_{PRD}. ■

Remark 3.1

The proposition holds true for the univariate population median. That is, β^*_{PRD} also recovers the univariate median in the population case. ■

Now we turn to the approximate computation of $\boldsymbol{\beta}^*_{PRD}$ in (14). First, we notice that the $\boldsymbol{\beta}^*_{PRD}$ must be bounded, or equivalently, the search for the optimal $\boldsymbol{\beta}$ in the RHS of (14) can be limited within a bounded set (hypersphere).

Proposition 3.2. For a given $\mathbf{Z}^{(n)} = \{(\mathbf{x}_i', y_i), i = 1, \cdots, n\}$ and a $\boldsymbol{\beta} \in \mathbb{R}^p$, under **(A1)** there is a constant c^* such that

$$\boldsymbol{\beta}^*_{PRD} = \arg\min_{\|\boldsymbol{\beta}\|\le c^*} \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_i\Big\{\frac{y_i - \mathbf{w}_i'\boldsymbol{\beta}}{\mathbf{w}_i'\mathbf{v}}\Big\}\Big|.$$

Proof: To see this, notice that for a given $\boldsymbol{\beta} \neq 0$, let $\mathbf{v}_0 = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$, then

$$\begin{aligned}\mathrm{UF}(\boldsymbol{\beta}; F_Z^n) &= \sup_{\mathbf{v}\in\mathbb{S}^{p-1}} \Big|\mathrm{Med}_i\{(y_i - \mathbf{w}_i'\boldsymbol{\beta})/\mathbf{w}_i'\mathbf{v}\}\Big| \ge \Big|\mathrm{Med}_i\{(y_i - \mathbf{w}_i'\boldsymbol{\beta})/\mathbf{w}_i'\mathbf{v}_0\}\Big| \\ &= \Big|\mathrm{Med}_i\{y_i/\mathbf{w}_i'\mathbf{v}_0\} - \|\boldsymbol{\beta}\|\Big| \longrightarrow \infty,\ a.s., \quad \text{as } \|\boldsymbol{\beta}\| \to \infty,\end{aligned}$$

where the last step follows from the fact that $\text{Med}_i\{y_i/\mathbf{w}_i'\mathbf{v}_0\}$ is bounded a.s.. Let $\delta = \text{UF}(\mathbf{0}, F_Z^n)$ and $c^* = \sup\{\|\boldsymbol{\beta}\|, \text{UF}(\boldsymbol{\beta}, F_Z^n) \le \delta\}$. This completes the proof. ■

The rough idea of the approximate algorithm is as follows. Randomly select $N_{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ over a very *wide* range in the parameter space $\mathbb{R}^p$, calculate all $\text{UF}(\boldsymbol{\beta}, F_{\mathbf{Z}}^n)$ w.r.t. the sample distribution $F_{\mathbf{Z}}^n$ of $F_{\mathbf{Z}}$. Sort the latter and select $p+1$ $\boldsymbol{\beta}$s with the smallest unfitness. Over the simplex formed by these $p+1$ $\boldsymbol{\beta}$ points (in the parameter space), search the point ($\boldsymbol{\beta}$) with the smallest unfitness (equivalent to the deepest regression line or hyperplane). Denote the latter by $\mathbf{T}_n^*$, the sample version of the $\mathbf{T}_{PRD}^*$.

In the above process, we have implicitly taken advantage of the property of the $\text{PRD}(\boldsymbol{\beta}; F_{\mathbf{Z}})$ or the $\text{UF}(\boldsymbol{\beta}; F_{\mathbf{Z}})$. That is, the $\text{PRD}(\boldsymbol{\beta}; F_{\mathbf{Z}})$ satisfies the property (P3) of Z18a (monotonicity relative to the deepest point). Therefore, the depth region of $\boldsymbol{\beta}$ (the set of all $\boldsymbol{\beta}$s with depth no less than a fixed value) is convex and nested. Hence, the deepest point(s) must lie over the closed convex simplex formed by the $p+1$ $\boldsymbol{\beta}$ points. The deepest PRD point is unique (see Zuo (2019b)).

The following is an **approximate algorithm** for the computation of $\mathbf{T}_n^*$.

(A) Randomly select a set of points $\boldsymbol{\beta_j} \in \mathbb{R}^p$, $j = 1, \cdots, N_{\boldsymbol{\beta}}$, where $N_{\boldsymbol{\beta}}$ is a tuning parameter for the total number of random points $\boldsymbol{\beta}$ tried.

(B) For each $\boldsymbol{\beta_j}$, compute, over a set of randomly selected unit directions $\mathbf{v_k} \in \mathbb{S}^{p-1}, k = 1, \cdots, N_{\mathbf{v}}$, an approximate unfitness of $\boldsymbol{\beta_j}$ w.r.t. $\{Z_{ik}^j = (y_i - \mathbf{w_i'}\boldsymbol{\beta_j})/(\mathbf{w_i'}\mathbf{v_k})\}$, $i = 1, \cdots, n$, $k = 1, \cdots, N_{\mathbf{v}}$, where $N_{\mathbf{v}}$ is another tuning parameter.

(C) Select the deepest $p+1$ $\boldsymbol{\beta}_j$s (points with the smallest unfitness). Search over the closed convex hull formed by these $p+1$ points via a common nonlinear optimization algorithm (e.g. the downhill simplex method (Nelder-Mead), or the MCMC technique) to get the final deepest $\boldsymbol{\beta}$ or our approximate $\mathbf{T}_n^*$.

(D) To mitigate the effect of randomness, repeat the steps above (many times) so that the one $\mathbf{T}_n^*$ with the maximum updated regression depth is adopted.

Remarks 3.2:

(I) The candidate (random point) $\boldsymbol{\beta}$ can be produced by randomly selecting p points from $\mathbf{Z}^{(n)} = \{(\mathbf{x_i}, y_i)', i = 1, \cdots, n\}$ which determine a ($\boldsymbol{\beta}$ or) hyperplane $y = \mathbf{w}'\boldsymbol{\beta}$ containing all p points. Let $S_{\boldsymbol{\beta}} := \{\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_{N_{\boldsymbol{\beta}}}\}$ be all $\boldsymbol{\beta}$s.

(II) The random directions could be selected among those which are normal vectors of the hyperplanes formed by p points from $\mathbf{Z}^{(n)}$ above. Furthermore, for each $\boldsymbol{\beta}_j \in S_{\boldsymbol{\beta}}$, one can consider all $\boldsymbol{v}_i^j = (\boldsymbol{\beta_i} - \boldsymbol{\beta_j})/\|\boldsymbol{\beta_i} - \boldsymbol{\beta_j}\|$, $\forall\ \boldsymbol{\beta}_i \ne \boldsymbol{\beta}_j$. Let $S_{\boldsymbol{v}} := \{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_{N_v}\}$ be all $\boldsymbol{v}$s.

(III) For a better approximation of depth (unfitness) of $\boldsymbol{\beta}_j$, tune (increase) $N_{\mathbf{v}}$. For a better approximation of $\mathbf{T}_n^*$, tune $N_{\boldsymbol{\beta}}$. Continue iterating until it satisfies a stopping rule (e.g. the difference between consecutive depths is less than a cutoff value).

(IV) The overall worst case time complexity of the algorithm is: step (A)+(B): $O(pN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}}))$, step (C): $O(pN_{\boldsymbol{\beta}} + N_{\mathbf{v}}N_{Iter}pn)$, where N_{Iter} is the total number of iterations in the optimization algorithm, step(D) $O(RpN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}}N_{Iter}))$, where R is the number of replications. The overall cost of the algorithm is $O(RpN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}}N_{Iter}))$, which could be reduced to $O(pN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}}))$ by skipping steps (C) and (D).

(V) After obtaining the approximate UF of the first $(p+1)$ $\boldsymbol{\beta}_j$s, record UF_{min}, the minimum of all $(p+1)$ UFs. For the calculation of the UF for any future $\boldsymbol{\beta}_k$, if along any direction $\mathbf{v}$, the directional $\text{UF}_{\mathbf{v}}(\boldsymbol{\beta}_k, F^n_{\mathbf{Z}})$ (see (13) of Z18a)$\geq \text{UF}_{min}$, then stop the computation for $\boldsymbol{\beta}_k$ and move to $\boldsymbol{\beta}_{k+1}$. Update UF_{min} if a new UF is obtained. In this way, the overall cost of the algorithm will be drastically reduced.

(VI) **Alternative algorithms**.

(i) After (A), compute the coordinate-wise median of the $\boldsymbol{\beta}$s and use it as an initial point for a nonlinear optimization algorithm (e.g. optimx or DEoptim in R) along with other arguments (e,g. a function compute-UF) to find the $\mathbf{T}^*_n$.

(ii) Increase $N_{\mathbf{v}}$ and $N_{\boldsymbol{\beta}}$ and skip steps (C) and (D). Namely, just employ steps (A)+(B).

(iii) Seek an algorithm similar to MEDSWEEP in Van Aelst et al (2002) for the $\mathbf{T}^*_{RD}$ since when $p = 1$ and regression through the origin, the $\mathbf{T}^*_{PRD}$ recovers the median of $\{y_i/x_i\}$ (see Section 3.3 of RH99 for a related discussion). ■

3.2. Computation times

As requested, we now turn to the comparison of the computation times for the $\mathbf{T}^*_{PRD}$, the $\mathbf{T}^*_{RD}$ and the least trimmed squares (Rousseeuw (1984)) regression (ltsReg) estimator.

As a median in regression, $\mathbf{T}^*_{RD}$ is a promising robust alternative to the classic least squares (LS) regression estimator. In fact, in terms of asymptotic breakdown point (ABP) robustness, the former possesses a 33% ABP (Van Aelst and Rousseeuw (2000) (VAR00)), in contrast to the 0% of the latter.

Zuo (2019b) (Z19b) has investigated the ABP of $\mathbf{T}^*_{PRD}$, it turns out that it possesses the highest possible ABP, 50%. For this advantage over the $\boldsymbol{\beta}^*_{RD}$ (see illustration examples in Z19b), it has to pay a price in the computation. The cost of the computation of the $\boldsymbol{\beta}^*_{RD}$ is generally lower than that of the $\boldsymbol{\beta}^*_{PRD}$.

To see the difference in the computation behavior, we list in table 5 the computation time consumed by both medians for different sample size ns and dimension ps. For the benchmark and comparison purposes, we also list the times consumed by the famous least trimmed squares (Rousseeuw (1984)) regression (ltsReg) estimator. Function rdepth in R package “mtfDepth” was used to calculate the RD of each candidate hyperplane. The performance of three algorithms for $\boldsymbol{\beta}^*_{RD}$, $\boldsymbol{\beta}^*_{PRD}$, and ltsReg, respectively, is demonstrated in the table 5.

We generate 1000 samples $\mathbf{Z}^{(n)} = \{(\mathbf{x}'_i, y_i)', i = 1, \cdots, n, \mathbf{x}_i \in \mathbb{R}^{p-1}\}$ from the Gaussian distribution with zero mean vector and 1 to p as its diagonal entries of the diagonal covariance matrix for various n and p. They are contaminated by 5% i.i.d. normal p-dimensional points with mean vector $(10, \cdots, 10)'$ and diagonal covariance matrix with 0.1 as its diagonal entries. Thus, we no longer have a symmetric errors and homoscedastic variance model (skewness and heteroscedasticity are allowed for the RD of RH99).

For a general estimator $\mathbf{T}$, if it is regression equivariant, then we can assume (w.l.o.g.) that the true parameter $\boldsymbol{\beta}_0 = \mathbf{0} \in \mathbb{R}^p$. We calculate EMSE $:= \frac{1}{R}\sum_{i=1}^{R} \|\mathbf{T}_i - \boldsymbol{\beta}_0\|^2$, the empirical mean squared error (EMSE) for $\mathbf{T}$, where $R = 1000$, $\boldsymbol{\beta}_0 = (0, \cdots, 0)' \in \mathbb{R}^p$, and $\mathbf{T}_i$ is the realization of $\mathbf{T}$ obtained from the ith sample with size n. The EMSE and the average computation time (in seconds) per sample by different estimators are listed in Table 5.

Remarks 3.3 Table 5 reveals that

(I) In terms of the average time consumed per sample, or computation speed, (i) the ltsReg is the fastest in all cases whereas the $\boldsymbol{\beta}^*_{RD}$ is the second fast method when p is 2, or 3 (and $n \leq 60$).

Table entries: (empirical mean squared error, average time per sample (seconds))

n	method	$p = 2$	$p = 3$	$p = 4$	$p = 6$
40	$\boldsymbol{\beta}^*_{PRD}$	(0.232, 0.060)	(0.468, 0.261)	(0.723, 0.304)	(1.429, 0.354)
	$\boldsymbol{\beta}^*_{RD}$	(0.243, 0.038)	(0.492, 0.124)	(2.7e+04, 6.542)	(1.717, 9.619)
	ltsReg	(0.380, 0.007)	(0.579, 0.011)	(0.781, 0.010)	(1.434, 0.018)
60	$\boldsymbol{\beta}^*_{PRD}$	(0.160, 0.080)	(0.323, 0.310)	(0.510, 0.445)	(0.894, 0.532)
	$\boldsymbol{\beta}^*_{RD}$	0.172, 0.043)	(0.366, 0.286)	(2565.1, 23.14)	(1.206, 11.82)
	ltsReg	(0.326, 0.007)	(0.475, 0.013)	(0.599, 0.015)	(0.894, 0.024)
80	$\boldsymbol{\beta}^*_{PRD}$	(0.124, 0.100)	(0.260, 0.436)	(0.413, 0.613)	(0.691, 0.634)
	$\boldsymbol{\beta}^*_{RD}$	(0.130, 0.047)	(0.291, 0.569)	(2012.6, 58.42)	(1.111, 14.08)
	ltsReg	(0.290, 0.009)	(0.416, 0.018)	(0.506, 0.020)	(0.703, 0.029)
100	$\boldsymbol{\beta}^*_{PRD}$	(0.100, 0.123)	(0.221, 0.528)	(0.346, 0.687)	(0.555, 0.763)
	$\boldsymbol{\beta}^*_{RD}$	(0.109, 0.048)	(0.252, 0.950)	(5.5e+06, 101.8)	(0.963, 16.37)
	ltsReg	(0.252, 0.010)	(0.418, 0.021)	(0.455, 0.024)	(0.578, 0.035)

Table 5: Performance of different regression methods for various n and p.

(ii) the $\boldsymbol{\beta}^*_{PRD}$ is the slowest in the $p = 2$ and $p = 3$ and ($n \leq 60$) cases whereas the $\boldsymbol{\beta}^*_{RD}$ unexpectedly becomes the slowest when ($p \geq 4$). (iii) the $\boldsymbol{\beta}^*_{PRD}$ is at least 20 (20 to 148) times faster than the $\boldsymbol{\beta}^*_{RD}$ when $p > 3$.

Note the comparison here is somewhat unfair since the ltsReg uses Fortran and the $\boldsymbol{\beta}^*_{RD}$ or the $\boldsymbol{\beta}^*_{PRD}$ employs Rcpp for the background computation. This example also confirms that old Fortran is still an excellent programming language for scientific computation.

(II) Computation speed is just one important performance criterion. Accuracy or efficiency is another, if not a more important, one. In terms of EMSE, there is an across-the-board winner. That is, the $\boldsymbol{\beta}^*_{PRD}$ has the smallest EMSE in all cases considered (in other words, it has the highest empirical relative efficiency).

(III) The $\boldsymbol{\beta}^*_{PRD}$ runs in less than one second in all cases considered. ■

Overall, Table 5 suggests that the $\boldsymbol{\beta}^*_{PRD}$ is a promising alternative among the competitors in regression.

All R code (downloadable via the link provided at the end of Example 2.4.1) ran on a desktop Intel(R)Core(TM) i7-2600 CPU @ 3.40GHz.

The ltsReg has a fairly good finite sample relative efficiency, but it is also notorious for its inefficiency in the asymptotic sense (with an asymptotic efficiency of just 7% (see Stromberg, et al. (2000)). The ltsReg benefits heavily in terms of speed from Fortran's compilation.

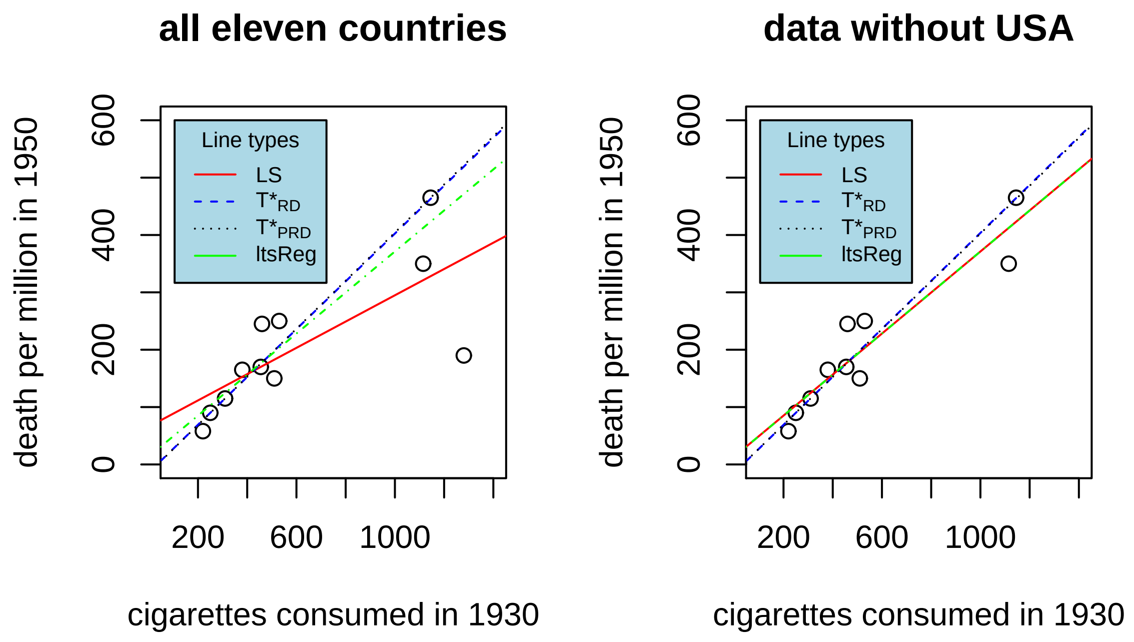


Figure 5: Four regression lines for a dataset with a single outlier (Solid red for the LS, dashed blue for the $\mathbf{T}^*_{RD}$, dotted black for the $\mathbf{T}^*_{PRD}$ and dotdash green for the ltsReg). Left: Original eleven countries, lines from the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg are similar while the LS line is attracted by a single country, the USA. Right: The outlier, USA, is removed from the original data, all four lines are very similar and catch the overall pattern.

3.3. Examples

Example 3.3.1 Deepest regression line $\mathbf{T}^*_{PRD}$ for a real data set

Lung Cancer and Smoking Data set is composed of per capita consumption of cigarettes in eleven countries in 1930 and the death rates (number of deaths per million people) from lung cancer in 1950 (see Table 3-3 of Tufte (1974), source: Doll (1955)).

To find out the relationship between death rate and the cigarettes consumed, we first regress the data with the deepest line $\mathbf{T}^*_{PRD}$, for benchmark and comparison purposes, lines from the $\mathbf{T}^*_{RD}$ (another line induced from depth), the LS (classical one) and the lstReg are also given. In terms of (intercept, slope) form, they are (65.7488570, 0.2291153), (-14.1666667, 0.4166667), (-14.9401198, 0.4191617), and (13.553435, 0.357668) for the LS, the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg, respectively. The two depth lines are almost identical whereas the LS line is attracted by a single country USA downward, see the left panel of Figure 5.

Next we remove the single outlier and repeat the steps above, we have this time (13.553435, 0.357668), (-14.6184633, 0.4182743), (-14.1666667, 0.4166667) and (13.553435 0.357668) for the LS, the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg, respectively. The ltsReg and the LS are the same as the previous lstReg with USA included and the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ are almost the same with the $\mathbf{T}^*_{PRD}$ being exactly the same as the previous $\mathbf{T}^*_{RD}$ with USA included, see the right panel of Figure 5.

Overall, this example indicates that a single outlier can drastically affect the LS line and distinguishes the LS line from other robust lines, but it can not differentiate the other three. ■

Example 3.3.2 Performance of the deepest line of $\mathbf{T}^*_{PRD}$ versus $\mathbf{T}^*_{RD}$

Here we first generate 100 points $\mathbf{Z}_i = (x, y)'$ from the bivariate normal distribution $\mathbf{N}(\boldsymbol{\mu}, \Sigma)$, where

$$\boldsymbol{\mu} = \begin{pmatrix} 8 \\ 0 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 9 & 0.9 \\ 0.9 & 1 \end{pmatrix}.$$

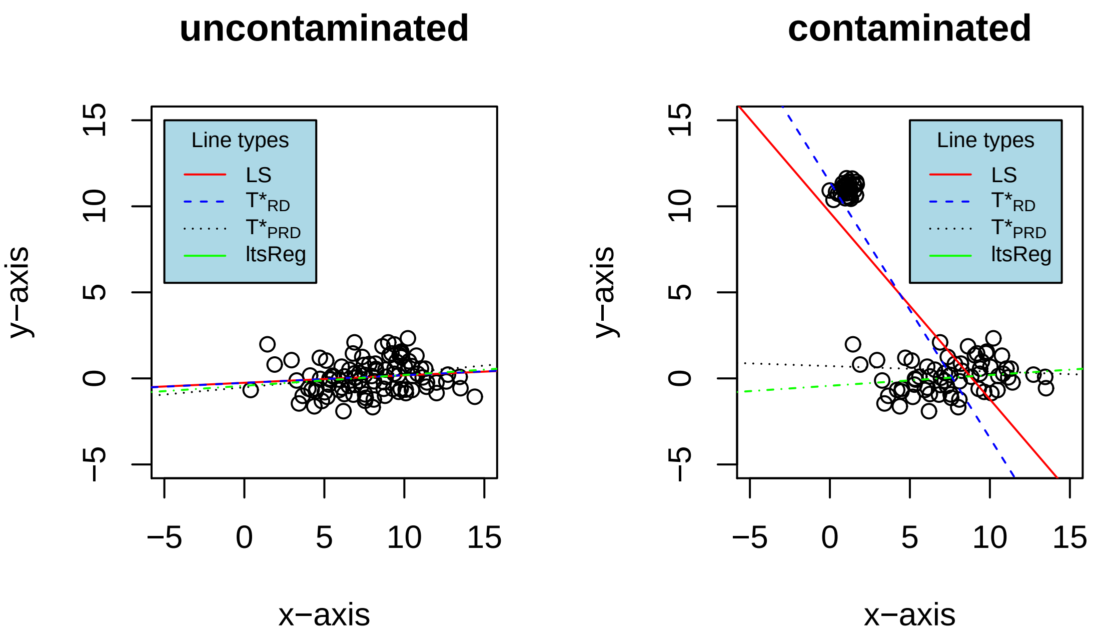


Figure 6: Four regression lines for data with or without contamination (Solid red for the LS, dashed blue for the $\mathbf{T}^*_{RD}$, dotted black for the $\mathbf{T}^*_{PRD}$, and dotdash green for the ltsReg). Left: Original 100 normal points, lines from the LS, the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg are similar and catch the overall linear pattern . Right: 34% contaminated data set, both the LS and the $\mathbf{T}^*_{RD}$ "break down" while the $\mathbf{T}^*_{PRD}$ and the ltsReg resist the contamination and still track the major pattern.

Among the 100 points, we randomly select 34 points and replace them by 34 other points from another bivariate normal distribution $\mathbf{N}(\boldsymbol{\mu}_c, \Sigma_c)$ with

$$\boldsymbol{\mu}_c = \begin{pmatrix} 1 \\ 11 \end{pmatrix}, \quad \Sigma_c = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix},$$

Thus, we have a 34% replacement-contamination data set.

First: w.r.t. the un-contaminated data set, we compute the deepest regression line induced from the PRD and then the competitor line induced from the RD of RH99. Again we also calculated the LS and the ltsReg lines. The four lines in (intercept, slope) form are (-0.2533241, 0.0431503), (-0.25902968, 0.04446287), (-0.51728622, 0.08431267) and (-0.42734074, 0.06248309) for the LS, the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg, respectively. They are almost identical as shown in the left panel of Figure 6. All four seem to be useful, catching the overall linear pattern.

Second: w.r.t. the replacement-contaminated data set, we also compute the four lines. They are (9.643466, -1.085616), (11.407796, -1.487017), (0.71612780, -0.03124584) and (-0.42734074, 0.06248309) for the LS, the $\mathbf{T}^*_{RD}$, the $\mathbf{T}^*_{PRD}$ and the ltsReg, respectively. They differ in obvious manners as shown in the right panel of Figure (6). Both the LS and the $\mathbf{T}^*_{RD}$ lines break down (attracted by the cloud of contamination) whereas the $\mathbf{T}^*_{PRD}$ and the ltsRg can resist the 34% contamination (in fact up to 50%) and catch the major pattern and continue to provide a useful regression line.

The computations in the example above (and below) are carried out with the R programming language for two reasons: (i) available code (package: mrfDepth) for the RD of RH99 is in R and (ii) fair comparisons. R code is available (see the link posted at the end of Example 2.4.1). ■

Remarks 3.2:

(I) Example 3.3.2 confirms the theoretical results in Z18b. That is, the deepest regression line or hyperplane induced from the RPD is a robust alternative to the traditional LS lines or hyperplanes and has a higher asymptotical breakdown point (ABP) (50%) than the leading depth median (33%), the deepest regression estimator induced from the RD of RH99. Note that with an appropriate trimming rate the least trimmed squares line possesses the best possible ABP whereas if the rate tends to 0% it leads to the LS line or hyperplane having 0% ABP since just one outlier can ruin them.

(II) Robustness does not work in tandem with efficiency. So the key question is: Are the deepest projection regression lines or hyperplanes ($\mathbf{T}^*_n$) efficient? In the following, ltsReg is excluded for a pure apples vs apples comparison (depth median vs depth median) and since it is notorious for its inefficiency (with asymptotic efficiency of just 7% (see Stromberg, et al.(2000)). ■

Replication 1000 times, $n = 65$

Performance criteria	$\boldsymbol{\beta}^*_{PRD}$	$\boldsymbol{\beta}^*_{RD}$
	Case I	$p = 3$
EMSE	0.09328212	0.11425414
Time consumed per sample	0.31453851	0.33221344
	Case II	$p = 4$
EMSE	0.1516812	3265582
Time consumed per sample	0.25505812	25.63505434
	Case III	$p = 5$
EMSE	0.2302744	0.2706360
Time consumed per sample	0.21889922	6.42221254

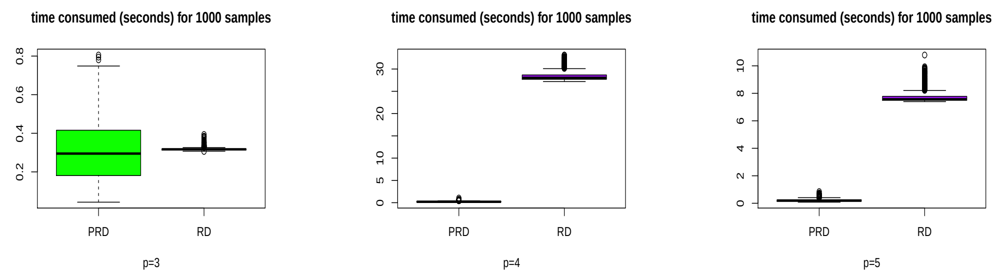
Table 6: Performance of different regression depth medians for the three true $\boldsymbol{\beta}_0$s.

Example 3.3.3 Now we investigate the performance of the two regression depth medians ($\boldsymbol{\beta}^*_{PRD}$, and $\boldsymbol{\beta}^*_{RD}$) in a slightly different setting. We generate 1000 samples $\{(\mathbf{x}'_i, y_i)' \in \mathbb{R}^p\}$ with a fixed sample size 65 from an assumed model: $y = \boldsymbol{\beta_0}'\mathbf{x} + e$, where $\mathbf{x} = (1, x_1, \cdots, x_{p-1})'$ and $\boldsymbol{\beta_0} = (\beta_0, \cdots, \beta_{p-1})'$ are in $\mathbb{R}^p$ and x_i and e are from either a Cauchy or a standard Gaussian distribution.

We list the average time consumed (in seconds) per sample and the EMSE (the same formula as before) for the two methods with respect to different $\boldsymbol{\beta_0}$'s in Table 6. **Case I** $\boldsymbol{\beta_0} = (-2, 0.1, 1)'$, all x_i and e are from $N(0, 1)$ distribution. **Case II** $\boldsymbol{\beta_0} = (-2, 0.1, 1, 5)'$, x_1 is from $N(0, 1)$ and all other x_i and e are from the Cauchy distribution. **Case III** $\boldsymbol{\beta_0} = (50, 0.1, -2, 15, 100)'$, all x_i and e are from $N(0, 1)$.

Inspecting table 6 reveals that (i) the $\boldsymbol{\beta}^*_{PRD}$ is faster than the $\boldsymbol{\beta}^*_{RD}$ in all cases; it is 100.5 and 29.34 times faster in cases $p = 4$ and $p = 5$, respectively. (ii) the $\boldsymbol{\beta}^*_{PRD}$ has a smaller EMSE in all cases. (iii) The sample variance (or more precisely EMSE) of the PRD induced median increases when p increases whereas the time consumed per sample for the fixed sample size by the $\boldsymbol{\beta}^*_{PRD}$ decreases.

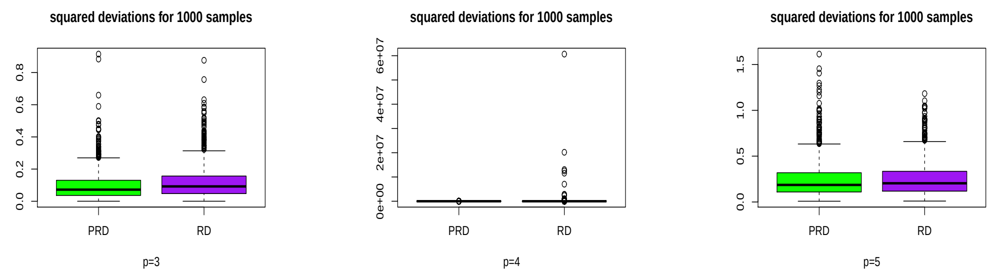
Numerical summary results in table 6 for 1000 samples are also displayed graphically in terms of their distributions in Figures 7 and 8. The times consumed by two methods for each of 1000 samples are displayed in Figure 7 with boxplots. Inspecting the Figure immediately reveals that in terms of time consumed per sample, the $\boldsymbol{\beta}^*_{PRD}$ is faster than the $\boldsymbol{\beta}^*_{RD}$ in all, especially $p = 4$ and $p = 5$ cases.



(a) p=3 (b) p=4 (c) p=5

Figure 7: Time consumed per sample by two depth induced deepest regression estimators $\boldsymbol{\beta}^*_{PRD}$ (green box) and $\boldsymbol{\beta}^*_{RD}$ (purple box) for the three $\boldsymbol{\beta}_0$ cases.

The distributions of the squared deviations $\|\boldsymbol{\beta}^*_i - \beta_0\|^2$ per sample for two methods $\boldsymbol{\beta}^*_{PRD}$ and $\boldsymbol{\beta}^*_{RD}$ are displayed in Figure 8. The figure clearly indicates that in the cases $p = 3$ and $p = 5$, the median of the squared-deviations of the $\boldsymbol{\beta}^*_{PRD}$ is the smaller one whereas it is also true in the case $p = 4$ but less clear. Notice that in the latter case, there is an obvious outlier for the $\boldsymbol{\beta}^*_{RD}$; it is greater than 6.0×10^7, which explains why the mean of the squared-deviations (empirical mean squared error, EMSE) in table 6 for the $\boldsymbol{\beta}^*_{RD}$ is huge. ■



(a) p=3 (b) p=4 (c) p=5

Figure 8: Squared-deviations $\|\boldsymbol{\beta}^*_i - \beta_0\|^2$ per sample by two depth induced deepest regression estimators $\boldsymbol{\beta}^*_{PRD}$ (green box) and $\boldsymbol{\beta}^*_{RD}$ (purple box) for the three $\boldsymbol{\beta}_0$ cases.

All results above and below are obtained on a desktop Intel(R)Core(TM) i7-2600 CPU @ 3.40GHz. R code in this and the next few sections is downloadable via the link listed at the end of Example 2.4.1.

4. Three estimators induced from PRD

From table 5 we see that the $\boldsymbol{\beta}^*_{PRD}$ is slower than $\boldsymbol{\beta}^*_{RD}$ in the case $p = 2$ or $p = 3 (n \leq 60)$. Are there any PRD induced estimators that run even faster than the $\boldsymbol{\beta}^*_{PRD}$? By reviewing the steps in section 3.1 for the computation of $\boldsymbol{\beta}^*_{PRD}$, the answer is yes. There are obviously other projection regression depth (PRD) induced estimators that can be computed even faster.

The first one adds no extra computation cost to the already obtained candidate $\boldsymbol{\beta}$ matrix $S_{\boldsymbol{\beta}}$ and their UFs, it is just the deepest $\boldsymbol{\beta}$ with the minimum UF in the matrix $S_{\boldsymbol{\beta}}$, denote it as $\boldsymbol{\beta}^*_{PRD1}$. The second one is the plain average of the deepest $(p+1)$ $\boldsymbol{\beta}$s from the $S_{\boldsymbol{\beta}}$, denote it as $\boldsymbol{\beta}^*_{PRD2}$. The third one is an UF weighted estimator defined below, denote it as $\boldsymbol{\beta}^*_{PRD3}$,

$$\boldsymbol{\beta}^*_{PRD3} = \frac{\sum_{i=1}^{(p+1)} w(\rho_i)\boldsymbol{\beta}_{(i)}}{\sum_{i=1}^{(p+1)} w(\rho_i)}, \tag{17}$$

Table entries: (empirical mean squared error, average time per sample (seconds))

n	method	$p=2$	$p=3$	$p=4$	$p=6$
40	$\boldsymbol{\beta}^*_{PRD}$	(0.237, 0.062)	(0.448, 0.142)	(0.736, 0.208)	(1.373, 0.343)
	$\boldsymbol{\beta}^*_{PRD1}$	(0.244, 0.023)	(0.481, 0.040)	(0.831, 0.068)	(1.646, 0.142)
	$\boldsymbol{\beta}^*_{PRD2}$	(0.268, 0.023)	(0.489, 0.040)	(0.882, 0.068)	(1.431, 0.142)
	$\boldsymbol{\beta}^*_{PRD3}$	(0.258, 0.023)	(0.476, 0.040)	(0.771, 0.068)	(1.375, 0.142)
	$\boldsymbol{\beta}^*_{RD}$	(0.240, 0.040)	(0.466, 0.124)	(3195.3, 6.507)	(1.678, 9.382)
60	$\boldsymbol{\beta}^*_{PRD}$	(0.157, 0.082)	(0.329, 0.187)	(0.519, 0.268)	(0.923, 0.193)
	$\boldsymbol{\beta}^*_{PRD1}$	(0.167, 0.031)	(0.363, 0.051)	(0.613, 0.090)	(1.139, 0.088)
	$\boldsymbol{\beta}^*_{PRD2}$	(0.188, 0.031)	(0.484, 0.051)	(0.603, 0.090)	(1.131, 0.088)
	$\boldsymbol{\beta}^*_{PRD3}$	(0.175, 0.031)	(0.446, 0.051)	(0.568, 0.090)	(1.075, 0.088)
	$\boldsymbol{\beta}^*_{RD}$	(0.165, 0.043)	(0.350, 0.300)	(4703.0, 21.18)	(1.337, 8.585)
80	$\boldsymbol{\beta}^*_{PRD}$	(0.128, 0.101)	(0.261, 0.441)	(0.412, 0.611)	(0.666, 0.288)
	$\boldsymbol{\beta}^*_{PRD1}$	(0.134, 0.040)	(0.297, 0.095)	(0.492, 0.165)	(0.832, 0.129)
	$\boldsymbol{\beta}^*_{PRD2}$	(0.165, 0.040)	(0.315, 0.095)	(0.509, 0.165)	(0.872, 0.129)
	$\boldsymbol{\beta}^*_{PRD3}$	(0.147, 0.040)	(0.302, 0.095)	(0.481, 0.165)	(0.830, 0.129)
	$\boldsymbol{\beta}^*_{RD}$	(0.132, 0.047)	(0.291, 0.583)	(4446.2, 58.50)	(1.050, 10.64)
100	$\boldsymbol{\beta}^*_{PRD}$	(0.109, 0.121)	(0.218, 0.301)	(0.361, 0.719)	(0.551, 0.338)
	$\boldsymbol{\beta}^*_{PRD1}$	(0.117, 0.048)	(0.247, 0.086)	(0.439, 0.202)	(0.682, 0.148)
	$\boldsymbol{\beta}^*_{PRD2}$	(0.153, 0.048)	(0.275, 0.086)	(0.467, 0.202)	(0.851, 0.148)
	$\boldsymbol{\beta}^*_{PRD3}$	(0.142, 0.048)	(0.263, 0.086)	(0.437, 0.202)	(0.771, 0.148)
	$\boldsymbol{\beta}^*_{RD}$	(0.115, 0.050)	(0.240, 0.960)	(2427164, 113.4)	(0.970, 12.24)

Table 7: Performance of regression depth induced estimators for various n and p.

where $\rho_i = \mathrm{UF}(\boldsymbol{\beta}_{(i)})$ and $\boldsymbol{\beta}_{(1)}, \cdots, \boldsymbol{\beta}_{(p+1)}$ are the first $(p+1)$ deepest $\boldsymbol{\beta}$s (with the least UF) in the $S_{\boldsymbol{\beta}}$ and the weight function w is defined as follows:

$$w(r) = \mathbf{I}(r \le r_0) + \mathbf{I}(r > r_0)\frac{exp\left(k\big(2r_0/r - (r_0/r)^2\big)\right) - 1}{exp\ (k) - 1}, \tag{18}$$

with the tuning parameters $k = 3$ and $r_0 = \rho_{(p-1)}$, the $(p-1)$th smallest UF among the $(p+1)$ minimum UFs. For more discussions on this weight function and the tuning parameters, refer to Zuo (2003) and Z19b.

These estimators obviously can run faster than $\boldsymbol{\beta}^*_{PRD}$ since they skip the time-consuming step of searching over the convex hull. One naturally wonders what are their EMSE's? We investigate the performance of $\boldsymbol{\beta}^*_{PRD}$, $\boldsymbol{\beta}^*_{PRD1}$, $\boldsymbol{\beta}^*_{PRD2}$, and $\boldsymbol{\beta}^*_{PRD3}$ which is reported in table 7. For the benchmark purpose, the depth median: $\boldsymbol{\beta}^*_{RD}$ of RH99 is included in the comparison. 1000 samples are generated with the same scheme as that for table 5.

Inspecting table 7 immediately reveals that (i) the $\boldsymbol{\beta}^*_{PRD}$ has the smallest EMSE in all cases and it can be faster than the $\boldsymbol{\beta}^*_{RD}$ which is the slowest in $p = 3 (n > 40)$, $p = 4$, $p = 6$ cases; (ii) the $\boldsymbol{\beta}^*_{PRD1}$, the $\boldsymbol{\beta}^*_{PRD2}$ and the $\boldsymbol{\beta}^*_{PRD3}$ are the fastest and they are currently regarded as having the same speed (all dependent on the given matrix $S_{\boldsymbol{\beta}}$ of candidate $\boldsymbol{\beta}$s and their unfitness and then on the sorted values of their unfitness); (iii) among the three, the deepest of all $\boldsymbol{\beta}$s in $S_{\boldsymbol{\beta}}$, $\boldsymbol{\beta}^*_{PRD1}$, and the depth weighted deepest $(p+1)$ $\boldsymbol{\beta}$s, $\boldsymbol{\beta}^*_{PRD3}$, seem to perform better than the plain average, $\boldsymbol{\beta}^*_{PRD2}$, which seems to perform the worst in most cases. Furthermore, our empirical evidence indicates that $\boldsymbol{\beta}^*_{PRD3}$ performs even better when p increases (say $p \geq 8$). (vi) Overall, $\boldsymbol{\beta}^*_{PRD}$ should be recommended among the five depth induced regression estimators; it becomes empirically the same as $\boldsymbol{\beta}^*_{PRD1}$ for large p (e.g. $p = 20$, $n = 40, 60, 80$); the second most impressive one is the $\boldsymbol{\beta}^*_{PRD3}$, and $\boldsymbol{\beta}^*_{PRD2}$ seems to be mediocre.

5. Finite sample relative efficiency of deepest projection regression lines/hyperplanes

Example 3.3.2 confirms that the $\mathbf{T}^*_{PRD}$ (or $\mathbf{T}^*_n$ in the empirical case) has a higher breakdown point than that of the leading regression depth induced median, the $\mathbf{T}^*_{RD}$. Robustness, however, is just one performance criterion for an estimator. Efficiency, if not more important, is another major performance measure. One naturally wonders whether the $\mathbf{T}^*_{PRD}$ is inferior to the $\mathbf{T}^*_{RD}$ w.r.t. the efficiency criterion.

The immediate answer is no based on the evidence demonstrated in tables 5, 6 and 7 since the $\mathbf{T}^*_{PRD}$ has a smaller EMSE than that of the $\mathbf{T}^*_{RD}$ uniformly over all cases considered. To confirm this empirical observation in $p = 2$ case, we now carry out a small scale simulation study.

($\epsilon = 0$)
Empirical mean squared error and relative efficiency of the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ w.r.t. the LS estimator

n	measures	$\mathbf{T}^*_{RD}$	$\mathbf{T}^*_{PRD}$	LS
10	EMSE	0.5987264	0.3723071	0.2653862
	RE	44%	71%	100%
20	EMSE	0.2358544	0.1571197	0.1104146
	RE	47%	70%	100%
40	EMSE	0.10163933	0.07492950	0.05287073
	RE	52%	71%	100%
80	EMSE	0.04893200	0.04060671	0.02597673
	RE	53%	64%	100%
100	EMSE	0.03196556	0.02535978	0.01679633
	RE	53%	66%	100%

Table 8: Relative efficiency of the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ for a normal model with 0% contamination.

In the following we investigate via a simulation study the finite-sample relative efficiency of the deepest lines $\mathbf{T}^*_{RD}$ and $\mathbf{T}^*_{PRD}$ w.r.t. the benchmark, the classical least squares line. The latter is optimal with normal models by the Gauss-Markov theorem. We generate $R = 1,000$ samples from the simple linear regression model: $y_i = \beta_0 + \beta_1 x_i + e_i, i = 1, 2, \cdots, n$, with different sample sizes n (see Tables 8 and 9), where $e_i \sim N(0, \sigma^2)$.

($\epsilon = 0.1$)

n	measures	$\mathbf{T}^*_{RD}$	$\mathbf{T}^*_{PRD}$	LS
10	EMSE	0.6612575	0.6181737	0.6373658
	RE	96%	103%	100%
20	EMSE	0.3396453	0.3247345	0.5179225
	RE	152%	159%	100%
40	EMSE	0.1613517	0.1525281	0.4475525
	RE	277%	293%	100%
80	EMSE	0.10348167	0.09775415	0.43277938
	RE	418%	442%	100%
100	EMSE	0.09702797	0.08947668	0.42298543
	RE	436%	473%	100%

Table 9: Relative efficiency of the $\mathbf{T}^*_{RD}$ and the $\mathbf{T}^*_{PRD}$ for a normal model with 10% contamination.

In light of the regression equivariance of the deepest regression estimators (see Z18a), we can assume w.l.o.g. that the true parameter $\boldsymbol{\beta}_0 = (\beta_0, \beta_1)' = (0,0)'$. We generate $(x_i, y_i)'$ from an ϵ% contaminated normal model $(1-\epsilon)N((0,0)', I_{2\times 2}) + \epsilon\delta_{(4,4)'}$ with $\epsilon = 0$ (a pure normal model, no contamination) and $\epsilon = 0.1$ (a 10% contaminated normal model), where $\delta_{\mathbf{Z}}$ is a point mass contaminating distribution at point $\mathbf{Z} \in \mathbb{R}^2$.

For a general estimator $\mathbf{T}$, the relative efficiency (RE) of the $\mathbf{T}$ is obtained by dividing the EMSE of the LS estimator by that of the $\mathbf{T}$. Tables 8 (a pure normal model case) and 9 (a normal model with 10% contamination) demonstrate the results with various ns.

Table 8 reveals that (i) the $\mathbf{T}^*_{PRD}$ is uniformly more efficient than the $\mathbf{T}^*_{RD}$ for all n; (ii) the limited numbers in Table 8 give a false impression that the efficiency of the $\mathbf{T}^*_{RD}$ increases forever as n increases. This is not true since when $n = 200$ the efficiency of the $\mathbf{T}^*_{RD}$ is still just 52%; (iii) as expected, the EMSE of any line decreases when n increases.

On the other hand, with the 10% contaminated normal model, Table 9 shows that (i) when $n = 10$, there is just one point that is contaminated. The classical least squares line as well as the line $\mathbf{T}^*_{RD}$ are drastically affected by just one contaminated point, nevertheless. They are less efficient than the deepest projection regression depth line. It is surprising that the line produced by the $\mathbf{T}^*_{RD}$ is sensitive to just one point contamination and is even less efficient than the LS line (This phenomenon can be explained by the low *finite sample* breakdown point of the $\mathbf{T}^*_{RD}$, which could be much lower than the ABP 1/3, see Zuo (2020)); (ii) when n increases, the efficiency of both deepest depth lines increases and are much higher than that of the LS line; (iii) the $\mathbf{T}^*_{PRD}$ is more efficient than the $\mathbf{T}^*_{RD}$ uniformly for all n; (iv) the EMSE of any line decreases when n increases; (v) the efficiency of the deepest lines increases as n increases, for example, when $n = 200$, the efficiency of the $\mathbf{T}^*_{RD}$ will be 525%.

6. Concluding remarks

The maximum projection regression depth estimator is a robust alternative to the classical least squares estimator. It possesses the best asymptotic breakdown point, a bounded influence function, and a very high finite sample replacement breakdown point (see Zuo (2019a)).

This article addresses the computation issues of the unfitness (UF), or equivalently the projection regression depth (PRD), and the PRD induced regression median, the maximum projection depth estimator. Exact and approximate algorithms are proposed and investigated. Compared with the leading regression depth notion RD and its induced median $\mathbf{T}^*_{RD}$ (RH99), the $\mathbf{T}^*_{PRD}$ is more computationally intensive $O(RpN_{\boldsymbol{\beta}}(p^2 + nN_{\mathbf{v}}N_{Iter}))$ versus $O(p(pn + N_{Iter}n + n\log n))$ of the $\mathbf{T}^*_{RD}$ (Van Alest et al (2002)). The $\mathbf{T}^*_{PRD}$, however, is not only more robust but also more efficient.

The article also introduces three PRD induced estimators that can run very fast. These estimators have a low level of empirical mean squared errors while satisfying regression, scale, and affine equivariance. The three estimators are expected to be highly robust, just like the $\boldsymbol{\beta}^*_{PRD}$ in Z19b with a high finite sample breakdown point.

The advantage (or disadvantage) of the PRD comes from its definition; it is defined based on the magnitude of residuals whereas the RD is defined purely on the sign of residuals. The latter results in a low breakdown point and efficiency, and non-uniqueness of the $\mathbf{T}^*_{RD}(F^n_{\mathbf{Z}})$ (the average might not have the maximum depth, see Van Aelst et al (2002) and Mizera and Volauf (2002)) whereas the $\mathbf{T}^*_{PRD}(F^n_{\mathbf{Z}})$ is unique (see Zuo (2019b)).

However, the RD also gains some unique features. For example, among others, the RD has the unique invariance property under the monotone transformations (see RH99); it is less difficult to compute and its definition does not require symmetry or homoscedasticity (see RH99).

Acknowledgments

The author thanks Hanwen Zuo, Hanshi Zuo, and Professor Emeritus James Stapleton for their careful proofreading. He thanks Yan-Han Chen and Dr. Wei Shao for their proofreading and useful discussions on C++, R, and Rcpp programming. Special thanks go to two anonymous referees for their insightful comments and constructive suggestions, all of which have significantly improved the manuscript.

References

[1] D. Bremner, K. Fukuda, and A. Marzetta, (1998), "Primal-Dual Methods for Vertex and Facet Enumeration", *Discrete and Computational Geometry*, 20, pp. 333–357.

[2] Boyd, S. and Vandenberghe, L. (2004), Convex Optimization. Cambridge University Press.

[3] Cohen, M. B. , Lee, Y. T. , and Song, Z. (2019), "Solving Linear Programs in the Current Matrix Multiplication Time", arXiv:1810.07896v2.

[4] Cuesta-Albertos, J. A. and Nieto-Reyes, A. (2008), "The random Tukey depth", *Computational Statistics and Data Analysis* 52, pp. 4979–4988.

[5] Chen, D., Morin, P. and Wagner. U., (2013), "Absolute approximation of Tukey depth: Theory and experiments", *Computational Geometry* 46, pp. 566-573.

[6] Dyckerhoff, R. (2004). "Data depths satisfying the projection property", *AStA-Advances in Statistical Analysis* 88, 163-190.

[7] Dohono, D. L. and Gasko, M. (1992), "Breakdown properties of location estimates based on halfspace depth and projected outlyingness", *Ann. Statist.*, 20 1803-1827.

[8] Doll, R. (1955), "Etiology of Lung Cancer", *Advances in Cancer Research*, 3, 1-50.

[9] Edelsbrunner, H. (1987), *Algorithms in Combinatorial Geometry.* Springer, Berlin, Heidelberg.

[10] Freund, R. M. (2004), "Issues in Non-Convex Optimization", Lecture Notes, MIT.

[11] Edelsbrunner, H. (1987), *Algorithms in Combinatorial Geometry.* Springer, Berlin, Heidelberg.

[12] Fukuda, K. (2004), "Frequently Asked Questions in Polyhedral Computation", http://www.ifor.math.ethz.ch/staff/fukuda/polyfaq/polyfaq.html

[13] Gonzaga, C. C. (1995), "On the Complexity of Linear Programming", Resenhas IME-USP 1995, Vol. 2, No. 2, 197-207.

[14] Huber, P. J. and Ronchetti, E. M. (2009), "Robust Statistics", 2nd edition, Wiley, New York.

[15] Koshevoy, G. and Mosler, K., (1997), "Zonoid trimming for multivariate distributions", *Ann. Statist.*, 1998-2017.

[16] Lee, Y. T. and Sidford, A. (2015), "Efficient inverse maintenance and faster algorithms for linear programming", In 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS), pages 230–249. IEEE.

[17] Liu, R. Y. (1990), "On a notion of data depth based on random simplices", *Ann. Statist.*, 18, 405-414, 1990.

[18] Liu, R. Y. (1992), "Data depth and multivariate rank tests", *In Y. Dodge (ed.), L1-Statistical Analysis and Related Methods*, North-Holland, Amsterdam, 279-294.

[19] Liu, X. and Zuo, Y. (2014), "Computing halfspace depth and regression depth", *Communications in Statistics - Simulation and Computation*, 43(5), 969-985.

[20] Maronna, R. A., and Yohai, V. J. (1993), "Bias-Robust Estimates of Regression Based on Projections", *Ann. Statist.*, 21(2), 965-990.

[21] Mizera, I. and Volauf, M (2002), "Continuity of halfspace depth contours and maximum depth estimators: diagnostics of depth-related methods", *Journal of Multivariate Analysis* 83, 365–388.

[22] Mosler, K. (2002), "Multivariate Dispersion, Central Regions and Depth: The Lift Zonoid Approach", Springer, New York.

[23] Mosler, K. (2013), "Depth statistics", In: Becker, C., Fried, R, and Kuhnt, S. (eds.), *Robustness and Complex Data Structures: Festschrift in Honour of Ursula Gather*, Springer-Verlag, 17-34.

[24] Mosler, K., Lange, T., and Bazovkin, P. (2009), "Computing zonoid trimmed regions of dimension $d > 2$", *Computational Statistics and Data Analysis* 53, 2500-2510.

[25] Nocedal, J. and Wright, S. J. (2006), Numerical Optimization. Springer.

[26] Paindaveine, D. and Šiman, M. (2012), "Computing multiple-output regression quantile regions", *Computational Statistics and Data Analysis* 56, 840-853.

[27] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007), Numerical Recipes. The Art of Scientific Computing, 3rd Edition, Cambridge University Press, New York.

[28] Rousseeuw, P. J. (1984), "Least Median of Squares Regression", *J. Amer. Statist. Assoc.* 79, 871-880.

[29] Rousseeuw, P. J., and Hubert, M. (1999), "Regression depth" (with discussion), *J. Amer. Statist. Assoc.*, 94: 388–433.

[30] Rousseeuw, P.J., and Leroy, A. (1987), "Robust regression and outlier detection". Wiley New York.

[31] Rousseeuw, P. J., Struyf, A. (1998), "Computing location depth and regression depth in higher dimensions", *Statistics and Computing*, 8:193-203.

[32] Stahel, W. A. (1981), "Robuste Schatzungen: Infinitesimale Optimalitiit und Schiitzungen von Kovarianzmatrizen", Ph.D. dissertation, ETH, Zurich.

[33] Tufte, E. R. (1974), “Data Analysis for Politics and Policy”, Prentice-Hall, Inc., Englewwood Cliffs, New Jersey.

[34] Tukey, J. W. (1975), “Mathematics and the picturing of data”, *In: James, R.D. (ed.), Proceeding of the International Congress of Mathematicians*, Vancouver 1974 (Volume 2), Canadian Mathematical Congress, Montreal, 523-531.

[35] Van Aelst, S., and Rousseeuw, P. J. (2000), “Robustness of Deepest Regression”, *J. Multivariate Anal.*, 73, 82–106.

[36] Van Aelst S., Rousseeuw P.J., Hubert M., Struyf A. (2002). The deepest regression method. *J. Multivariate Anal.*, 81, 138–166.

[37] Vardi, Y. and Zhang, C.-H. (2000). The multivariate l1-median and associated data depth. *Proc. Natl. Acad. Sci. USA*, 97, 1423–1426.

[38] Vanderbei, R.J.(1999), “LOQO: An interior point code for quadratic programming”, *Optimization Methods and Software*, 12:451–484.

[39] Vanderbei, R. J. and Shanno, D.F. (1999), “An Interior-Point Algorithm for Nonconvex Nonlinear Programming”, *Computational Optimization and Applications*, 13:231–252.

[40] Wright, S. J. (1997), Primal-dual interior-point methods, SIAM, Philadelphia.

[41] Yin Tat Lee, Aaron Sidford (2015), “Efficient Inverse Maintenance and Faster Algorithms for Linear Programming”, arXiv:1503.01752v3.

[42] Winder, R., (1966), “Partitions of N-space by hyperplanes”, *SIAM J. Appl. Math.* 14, 811–818.

[43] Wu, M., and Zuo, Y. (2008), “Trimmed and Winsorized Standard Deviations based on a scaled deviation”, *Journal of Nonparametric Statistics*, 20(4):319-335.

[44] Wu, M., and Zuo, Y. (2009), “Trimmed and Winsorized means based on a scaled deviation”, *J. Statist. Plann. Inference*, 139(2) 350-365.

[45] Zuo, Y. (2003), “Projection-based depth functions and associated medians”, *Ann. Statist.*, 31, 1460-1490.

[46] Zuo, Y. (2018), “A new approach for the computation of halfspace depth in high dimensions”. *Communications in Statistics - Simulation and Computation*, 48(3): 900-921.

[47] Zuo, Y. (2018a), “On general notions of depth in regression”, arXiv:1805.02046, *Statistical Science*, 2021, Vol. 36, No. 1, 142–157.

[48] Zuo, Y. (2018b), “Large sample properties of the regression depth induced median”, arXiv:1809.09896, *Statistics and Probability Letters*, 166, November 2020, 108879, https://doi.org/10.1016/j.spl.2020.108879.

[49] Zuo, Y. (2019a), “Robustness of deepest projection regression depth functional”, *Statistical Papers*, https://doi.org/10.1007/s00362-019-01129-4.

[50] Zuo, Y. (2019b), “Depth induced regression medians and uniqueness”, *Stats*, 2020, 3(2), 94-106; https://doi.org/10.3390/stats3020009.

[51] Zuo, Y. (2020), ”Finite sample breakdown point of multivariate regression depth median”, arXiv:2009.00646.

[52] Zuo, Y. and Lao, S. (2011), “Exact computation of the bivariate projection depth and Stahel-Donoho estimator”, *Computational Statistics and Data Analysis* , 53(3), 1173-1179.

[53] Zuo, Y., Serfling, R., 2000, “General notions of statistical depth function”, *Ann. Statist.*, 28, 461-482.