Computational Statistics & Data Analysis 19 (1995) 265–282
North-Holland

Exact computation of the least trimmed squares estimate in simple linear regression

Ola Hössjer *
Lund Institute of Technology, Lund, Sweden

Received November 1992
Revised December 1993

Abstract: The least trimmed squares estimate is defined by minimizing the sum of the h smallest squared residuals. We describe a simple algorithm for computing the least trimmed squares estimate exactly in simple linear regression, which requires $O(n^3 \log n)$ computations and $O(n^2)$ storage. The idea is to compute the least squares (LS) estimate for $O(n^2)$ subsets of size h (and not all $\binom{n}{h}$ possible subsets!) and then to choose the LS-estimate with the smallest sum of trimmed squares. A faster but more complicated version of the algorithm is obtained by updating the subsets and LS-estimates recursively. This refined algorithm has computation time $O(n^2 \log n)$ and storage $O(n)$. Some numerical examples are presented to compare the exact algorithm with approximative ones. The exact algorithm can easily be extended to nonlinear regression. Other possible extensions are exact computation of the LMS-estimate and the R-estimate defined in Hössjer (1994).

Keywords: Breakdown point; Dual plot; Least trimmed squares estimator; Simple linear regression; Recursive; Robustness

1. Introduction

In the simple linear regression model, the observations y_i are generated according to

$$y_i = \alpha + \beta x_i + e_i, \quad i = 1, \ldots, n, \tag{1.1}$$

where α is the intercept parameter, β the slope parameter, x_i the explanatory

* This paper was written under support by the Swedish Board for Technical Development, contract 712-89-1073 and the Swedish Natural Science Research Council, contract F-DP 6689-300.
Correspondence to: O. Hössjer, Department of Mathematical Statistics, Lund Institute of Technology, Box 118, S-22100 Lund, Sweden.

SSDI 0167-9473(94)0005-4

variables and finally e_i the error terms. If $\alpha = 0$ we obtain the simple linear regression model without intercept,

$$y_i = \beta x_i + e_i, \quad i = 1, \ldots, n. \tag{1.2}$$

Let $\boldsymbol{\theta}$ denote the parameter vector ($\boldsymbol{\theta} = (\alpha, \beta)$ in (1.1) and $\boldsymbol{\theta} = \beta$ in (1.2)). A quite general class of estimators may be constructed by minimizing a dispersion measure D of the residual vector $\boldsymbol{r}(\boldsymbol{\theta}) = (r_1(\boldsymbol{\theta})), \ldots, r_n(\boldsymbol{\theta}))$ w.r.t. $\boldsymbol{\theta}$, where $r_i(\boldsymbol{\theta}) = r_i(\alpha, \beta) = y_i - \alpha - \beta x_i$ in (1.1) and $r_i(\boldsymbol{\theta}) = r_i(\beta) = y_i - \beta x_i$ in (1.2). In other words,

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} D(\boldsymbol{r}(\boldsymbol{\theta})). \tag{1.3}$$

Two well-known special cases of (1.3) are the least squares (LS) and L_1-estimators (with D the sum of squared residuals or the sum of absolute residuals respectively). In this paper, we consider the least trimmed squares estimator (LTS) defined by Rousseeuw (1985), where

$$D(\boldsymbol{r}) = \sum_{i=1}^{h} r_{\sigma_i}^2, \tag{1.4}$$

where $|r_{\sigma_1}| \leqslant \cdots \leqslant |r_{\sigma_n}|$ are the ordered absolute values of the residuals, h is the trimming constant and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is the vector of absolute antiranks, i.e., the inverse permutation of the ranks of the absolute values of the residuals. The trimming constant h is chosen to achieve the desired degree of robustness of the estimator. A frequently used measure of robustness is the finite sample breakdown point ϵ_n^* (Donoho and Huber, 1983), defined as the smallest proportion of observations that after being replaced can change the estimate arbitrarily much. If $h/n \to \epsilon$ as $n \to \infty$, it follows that $\epsilon_n^* \to \epsilon^* = \min(\epsilon, 1 - \epsilon)$ (cf. Rousseeuw, 1984, Remark 1 and Hössjer, 1994, Remark 2.2). In particular, if $h/n \to 0.5$ we obtain the maximal possible value ($= 0.5$) for the asymptotic breakdown point ϵ^*.

Other special cases of (1.4) are the least median of squares (LMS, Rousseeuw, 1984), S-estimators (Rousseeuw and Yohai, 1984) and R-estimators (Hössjer, 1994). For these estimators, D is defined as the median of squared residuals, an M-estimator of scale and an L-estimator of scale respectively. A problem with the LTS-estimator is that no reasonably fast algorithm for computing it exactly has been known. In the location model, an exact algorithm which requires $O(n \log n)$ operations is described in Rousseeuw and Leroy (1987, Chapter 4). In the general (multiple regression) case we could in principle compute the least squares estimate of all $\binom{n}{h}$ subsamples of size h and then choose as our LTS-estimate the LS-estimate with minimal trimmed sum of squares. However, the required number of computations grows so fast with n that this algorithm becomes unpractical even for moderately large values of n.

Instead, approximative algorithms have been proposed. In the PROGRESS algorithm described in Rousseeuw and Leroy (1987, Chapter 5), exact fits are computed for elemental subsets of size q, where q is the number of unknown

parameters ($q = 2$ in (1.1) and $q = 1$ in (1.2)). The exact fit with minimal dispersion D is then selected and finally (for the model (1.1)), the intercept of this fit is adjusted by means of the exact one-dimensional LTS-algorithm mentioned above. A refined version is obtained by adjusting for the intercept at each selected elemental subset (described on page 201 of Rousseeuw and Leroy, 1987). Either all possible ($\leqslant \binom{n}{p}$) elemental subsets are used, or alternatively, a smaller number of (randomly chosen) subsets. If all elemental subsets are used, the total number of operations becomes $O(n^3 \log n)$ in (1.1) and $O(n^2 \log n)$ in (1.2). This is because each evaluation of the objective function D requires $O(n \log n)$ operations for sorting the absolute values of the residuals. Ruppert (1992) defines the RANDDIR algorithm for approximate computation of the LTS-estimate. In each step of this algorithm, a convex linear combination of the current best parameter and the exact fit of the last elemental subset is evaluated, so that the search for the LTS-estimate is concentrated at the current best value. We remark that both of these algorithms are applicable for any estimator of the kind (1.4). Only the computation of the objective function D has to be changed. Atkinson and Weisberg (1991) define an LTS-algorithm based on simulated annealing and Hawkins (1993) introduces the "Feasible Set Algorithm" for approximative computation of the LTS-estimate. Both of these methods are based on first selecting a subset of h data-points at random and then successively swapping one data-point from the selected subsample with one of the trimmed data points. Both of these algorithms are probabilistic. They yield the exact estimate with a certain probability (that can be chosen arbitrarily close to one by running the algorithm for a sufficiently long time).

In this paper, we present a simple algorithm for computing the LTS-estimate which requires $O(n^3 \log n)$ computations and $O(n^2)$ storage for both of the models (1.1) and (1.2). The basic idea is that the LS-estimate of only $O(n^2)$ out of all $\binom{n}{h}$ subsamples of size h need to be computed. The (absolute) dual plot of the data (Section 2) determines which subsamples we need to consider. In Section 3 we describe the algorithm for simple linear regression without intercept, and in Section 4 when the intercept is included. An improved, but more complicated, version of the algorithm is outlined in Section 5. It only requires $O(n^2 \log n)$ computations and $O(n)$ storage. We also discuss how to extend the algorithm to nonlinear regression in Section 6. Finally, we discuss some numerical aspects and extensions to multivariate regression and to the computation of other estimators in Section 7.

2. Dual plots and absolute dual plots

2.1. Dual plots

The *dual plot* of the regression data $(x_1, y_1), \ldots, (x_n, y_n)$ interchanges the role of points and lines. Each point (x_i, y_i) is represented by the line $\beta \to r_i(\beta) = y_i - \beta x_i$ with intercept y_i and slope $-x_i$. This line describes how the residual of

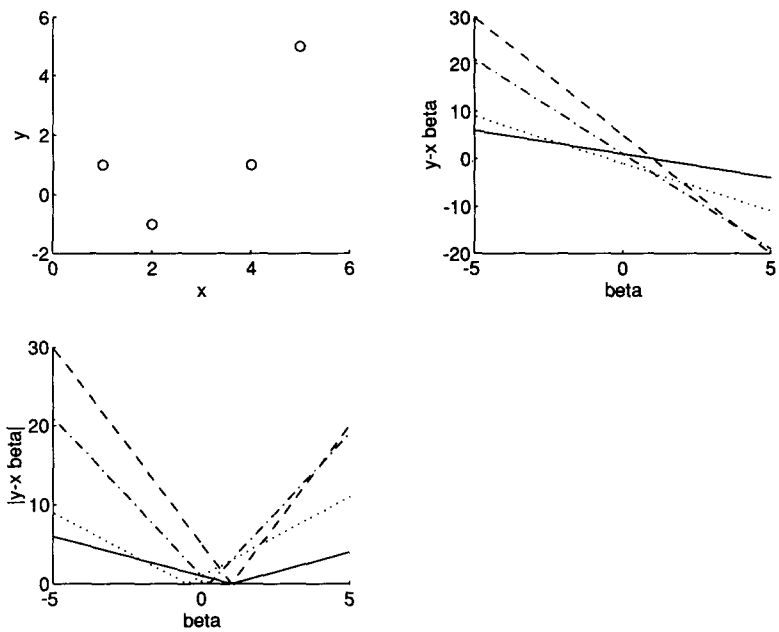

Fig. 1. Upper left: Artificial data set consisting of the four points (1, 1), (5, 5), (2, −1) and (4, 1). Upper right: Dual plot for the data, with $N = 6$, $\beta_1 = -2$, $\beta_2 = 0$, $\beta_3 = 1$, $\beta_4 = 2$ and $\beta_5 = 4$. We see that $M_k = 1$ and $N_k = 2$ when $k \neq 3$, $M_3 = 2$, $N_3 = 4$ and $N_{31} = N_{32} = 2$. Lower left: Absolute dual plot for the data, with $N^+ = 9$, $\beta_1^+ = -2$, $\beta_2^+ = 0$, $\beta_3^+ = 0.4$, $\beta_4^+ = 4/7$, $\beta_5^+ = \frac{2}{3}$, $\beta_6^+ = 1$, $\beta_7^+ = 2$ and $\beta_8^+ = 4$. We see that $N_3^+ = 3$, $M_3^+ = 1$, $N_6^+ = 4$, $M_6^+ = 2$. For the remaining values of k we have $N_k^+ = 2$ and $M_k^+ = 1$.

the point (x_i, y_i) varies in a model (1.2) without intercept. Figure 1b depicts the dual plot of the data set from Figure 1a. For a discussion of dual plots, cf. Johnstone and Velleman (1985, Sect. 2.1), and the references in this article. Two other papers that make use of dual plots are Mount and Netanyahu (1992) and Rosenqvist (1992).

Define $\tau(\beta) = (\tau_1(\beta), \ldots, \tau_n(\beta))$ as the vector of antiranks, i.e., the inverse permutation of the ranks of the residuals $r_1(\beta), \ldots, r_n(\beta)$. In Section 4, we will make constant use of how $\tau(\beta)$ varies with β. It is clear from Figure 1b that the antiranks $\tau_i(\beta)$ only change at the intersection of two lines $\beta \to r_i(\beta)$ and $\beta \to r_j(\beta)$. This occurres when

$$\beta = \bar{\beta}_{ij} = \frac{y_i - y_j}{x_i - x_j}, \quad x_i \neq x_j, \tag{2.1}$$

assuming that we only define these quantities when the denominators are nonzero. For simplicity, we assume that all (x_i, y_i) are different, so that the lines in the dual plot intersect at isolated values of β. Let $\beta_1 < \cdots < \beta_{N-1}$, be the ordered set of β-values that is formed when (i, j) ranges over all possible pairs in (2.1). This defines a division of the real line into N intervals $I_k = (\beta_{k-1}, \beta_k)$, $k = 1, \ldots, N$ (where $\beta_0 = -\infty$ and $\beta_N = \infty$). If at least two x-coordinates of the data points are distinct we have $2 \leqslant N \leqslant \binom{n}{2} + 1$.

The vector of antiranks $\tau(\beta)$ is constant on the interval I_k, $k = 1, \ldots, N$. We denote this constant value by $\tau^k = (\tau_1^k, \ldots, \tau_n^k)$. The vector τ^{k+1} differs from τ^k in that only a few ranks are switched, because some lines in the dual plot intersect at β_k. Typically, only two lines intersect, but in principle the number N_k of intersecting lines could be as high as n (which occurs when $r_1(\beta_k) = \cdots = r_n(\beta_k)$). Furthermore, the lines that intersect at β_k can be organized into M_k subsets depending on the value of $r_i(\beta_k)$. Only the antiranks within the same subset are switched when β passes β_k. Suppose that the ranks on I_k of the l-th subset are i_j, $i_l + 1, \ldots, j_l$, with $i_1 < j_1 < i_2 \cdots < j_{M_k}$, and let $N_{kl} = j_l - i_l + 1$ be the number indices in the l-th group. (We omit the dependence of i_l and j_l on k in the notation.) We see that in the regression plot, there are N_{kl} points that are collinear with slope β_k.

2.2. *Absolute dual plots*

Alternatively, we may express how the absolute values of the residuals $|r_i(\beta)|$ vary with β. We call this an *absolute dual plot*. An absolute dual plot of a regression data set is shown in Figure 1c. We will make use of this plot in Section 3.

Typically, two curves $\beta \to |r_i(\beta)|$ and $\beta \to |r_j(\beta)|$ intersect at two points, either when $\beta = \bar{\beta}_{ij}$, or when

$$\beta = \tilde{\beta}_{ij} = \frac{y_i + y_j}{x_i + x_j}, \quad i \neq j, \quad x_i \neq -x_j, \tag{2.2}$$

which happens when $r_i(\beta) = -r_j(\beta)$. Let $\beta_1^+ < \cdots < \beta_{N^+-1}^+$ be the ordered sequence of all intersection points. This defines N^+ intervals $I_k^+ = (\beta_{k-1}^+, \beta_k^+)$.

Let $\sigma(\beta)$ denote the vector of absolute antiranks computed from $|r_1(\beta)|, \ldots, |r_n(\beta)|$. We see that $\sigma(\beta)$ is constant on each of the intervals I_k^+. Denote this constant value by $\sigma^k = (\sigma_1^k, \ldots, \sigma_n^k)$. In analogy with the notation for dual plots, let N_k^+ be the number of curves intersecting at β_k^+, divided into M_k^+ groups. The l-th group has N_{kl}^+ elements with indices $i_l, \ldots, j_l$.

3. Simple linear regression without intercept

3.1. *Minimizing h-samples*

To simplify notation, we introduce (valid also in the general multiple linear regression setup):

Definition. A subsample of size h of $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, for which the LS-estimate equals the LTS-estimate, is called a minimizing h-sample.

The following proposition gives a clue to finding a minimizing h-sample:

Proposition 1. *Let $\hat{\beta}$ be an exact LTS-estimate (not necessarily unique) of the slope parameter in* (1.2). *Then $(\sigma_1(\hat{\beta}), \ldots, \sigma_h(\hat{\beta}))$ define the indices of a minimizing h-sample.*

Proof. For any β,

$$\sum_{i=1}^{h} r_{\sigma_i(\hat{\beta})}(\beta)^2 \geqslant \sum_{i=1}^{h} r_{\sigma_i(\beta)}(\beta)^2 \geqslant \sum_{i=1}^{h} r_{\sigma_i(\hat{\beta})}(\hat{\beta})^2, \tag{3.1}$$

where the first inequality follows by the definition of the absolute antiranks, and the second one from the definition of the LTS-estimate. But (3.1) is equivalent to saying that the h-sample with indices $\sigma_1(\hat{\beta}), \ldots, \sigma_h(\hat{\beta})$ yields an LS-estimate $\hat{\beta}$. □

3.2. The simple algorithm

According to Proposition 1, we find the indices of a minimizing h-sample by varying $\{\sigma_i(\beta)\}_{i=1}^{h}$ as a function of β. But since the absolute antiranks are piecewise constant as described above, we only have to compute them for N^+ values of β, one from each interval I_k.

Given a permutation $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ of $(1, \ldots, n)$, let $\hat{\beta}_{LS}^{hi}(\boldsymbol{\sigma})$ be the LS-estimate computed from $\{(x_{\sigma_j}, y_{\sigma_j})\}_{j=i}^{i+h-1}$. The algorithm for computing the LTS-estimate $\hat{\beta}$ may now be described as follows:

Exact LTS algorithm for calculating $\hat{\beta}$:

1. Initialize: Compute $\beta_1^+ < \cdots < \beta_{N^+-1}^+$, $\boldsymbol{\sigma}^1$, $\hat{\beta}_1 := \hat{\beta}_{LS}^{h1}(\boldsymbol{\sigma}^1)$, $D_1 := \sum_{i=1}^{h} r_{\sigma_i^1}(\hat{\beta}_1)^2$ and put $\hat{\beta} = \hat{\beta}_1$, $D_{\min} = D_1$

For $k = 1$ **to** $N^+ - 1$ **do**

2. Compute $\boldsymbol{\sigma}^{k+1}$
3. **If** $(\sigma_1^{k+1}, \ldots, \sigma_h^{k+1}) \neq (\sigma_1^k, \ldots, \sigma_h^k)$ **then**
   Compute $\hat{\beta}_{k+1} := \hat{\beta}_{LS}^{h}(\boldsymbol{\sigma}^{k+1})$
   Compute $D_{k+1} := \sum_{i=1}^{h} r_{\sigma_i^{k+1}}(\hat{\beta}_{k+1})^2$
   If $D_{k+1} < D_{\min}$ **then**
   $D_{\min} \to D_{k+1}$ and $\hat{\beta} \to \hat{\beta}_{k+1}$

Some remarks on the steps in the algorithm:

1. Computation of all β_i^+ requires sorting of $\{\bar{\beta}_{ij}, \tilde{\beta}_{ij};\ 1 \leqslant i < j \leqslant n\}$, which may be done in $O(n^2 \log n)$ steps and requires $O(n^2)$ storage. Then σ^1 requires sorting of $\{|r_i(\beta)|\}_{i=1}^{n}$ for some $\beta \in I_1$. This requires $O(n \log n)$ computations. The computation of $\hat{\beta}_1$ and D_1 on the other hand requires only $O(n)$ operations.

2. For each k, $\boldsymbol{\sigma}^{k+1}$ may be computed in $O(n \log n)$ time by sorting

$\{|r_i(\beta)|, 1 \leqslant i \leqslant n\}$ for any $\beta \in I_{k+1}^+ = (\beta_k^+, \beta_{k+1}^+)$. The total number of operations becomes $O(n^3 \log n)$.

3. This step is actually skipped for most values of k. We only have to go through it when $h \in \{i_l, \ldots, j_l\}$, for some $l \in \{1, 2, \ldots, M_k^+\}$. (For instance, when no three points are collinear and if $N_k^+ = 2$ for all k, the condition for entering step 3 becomes $i_1 = h$, and the fraction of intersection points satisfying this should be approximately $1/n$, although we have no strict proof of this.) Each LS-estimate $\hat{\beta}_{k+1}$ may be found in $O(n)$ steps. An upper bound for the total number of operations in Step 3 is therefore $O(n^3)$.

Remark. The $O(n^2)$ storage is not necessary if we let β range over the values $\{\bar{\beta}_{12}, \bar{\beta}_{13}, \ldots, \bar{\beta}_{n-1,n}, \tilde{\beta}_{12}, \tilde{\beta}_{12}, \ldots, \tilde{\beta}_{n-1,n}\}$ instead of $\{\beta_1^+, \ldots, \beta_{N^+-1}^+\}$. This requires som changes in Step 2, since we don't know the intervals $I_1^+, \ldots, I_{N^+}^+$.

3.3. Numerical examples

Figure 2 displays an artificial data set consisting of six points, with six LTS fits, one using the exact algorithm described here, one using the PROGRESS algorithm and three using the RANDDIR algorithm with 20 randomly chosen elemental subsets. The truncation point h is 5 in all cases. In this case, the PROGRESS algorithm gives a poor approximation to the exact estimate, whereas the RANDDIR estimates are fairly close.

Our next example is the lactic data described in Afifi and Azen (1979, p. 125). This data set concerns the measured value of lactic acid concentration in blood

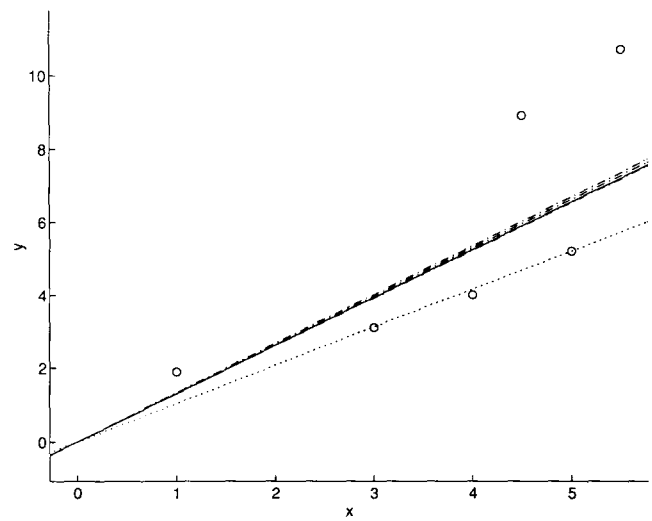

Fig. 2. Simple linear regression through the origin. LTS estimates for an artificial data set with $n = 6$ and $h = 5$, using an exact algorithm (solid line, $\hat{\beta} = 1.309$, $D_{\min} = 13.427$), the PROGRESS algorithm (dotted line, $\hat{\beta} = 1.040$, $D_{\min} = 18.574$) and the RANDDIR algorithm with 20 elemental subsets (dashdotted lines, $\hat{\beta} = 1.336, 1.305, 1.322$ and $D_{\min} = 13.481, 13.428, 13.440$).

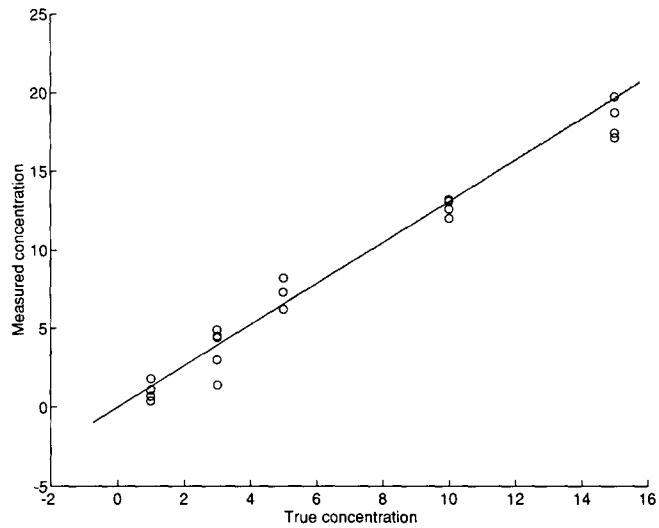

Fig. 3. Simple linear regression through the origin. LTS estimates for the lactic data with $n = 20$ and $h = 10$, using an exact algorithm (solid line, $D_{\min} = 1.5785$, $\hat{\beta} = 1.3061$) and the PROGRESS algorithm (dotted line, $D_{\min} = 1.5871$, $\hat{\beta} = 1.3100$). Four RANDDIR fits with 50 elemental subsets had $\hat{\beta}$ ranging between 1.3057 and 1.3074 and three RANDDIR estimates with 100 elemental subsets varied between 1.3061 and 1.3062.

(y_i) as a function of the true value (x_i). The exact LTS fit and PROGRESS LTS fit are displayed in Figure 3. In this case, the two lines are almost indistinguishable, as were all RANDDIR fits (with 50 and 100 random elemental subsets) that we tried.

4. Simple linear regression with intercept

4.1. Minimizing h-samples

The following proposition characterizes the minimizing h-samples when an intercept is present.

Proposition 2. *Let $\hat{\boldsymbol{\theta}} = (\hat{\alpha}, \hat{\beta})$ be an exact LTS-estimate in the model* (1.1) *(not necessarily unique). Then for some i, $1 \leqslant i \leqslant n - h + 1$, the vector $(\tau_i(\hat{\beta}), \ldots, \tau_{i+h-1}(\hat{\beta}))$ gives the indices of a minimizing h-sample.*

Proof. We may rewrite (1.3) as

$$(\hat{\alpha}, \hat{\beta}) = \arg \min_{\beta} \min_{\alpha} D(\boldsymbol{r}(\alpha, \beta)). \tag{4.1}$$

The inner minimization in (4.1), when β is kept fixed, corresponds to finding the LTS location estimate from the "sample" $\{y_i - \beta x_i\}_{i=1}^n$. By the one-dimensional

algorithm described in Rousseeuw and Leroy (1987, Chapter 4), there exists an $i(\beta)$, $1 \leqslant i(\beta) \leqslant n-h+1$, such that

$$\min_{\alpha} D(\boldsymbol{r}(\alpha, \beta)) = \sum_{j=i(\beta)}^{i(\beta)+h-1} r_{\tau_j(\beta)}(\alpha(\beta), \beta)^2, \tag{4.2}$$

with

$$\alpha(\beta) = \frac{1}{h} \sum_{j=i(\beta)}^{i(\beta)+h-1} r_j(\beta).$$

In particular, $\hat{\theta} = (\hat{\alpha}, \hat{\beta}) = (\alpha(\hat{\beta}), \hat{\beta})$. We then obtain for any (α, β) that

$$\begin{aligned} \sum_{j=i(\hat{\beta})}^{i(\hat{\beta})+h-1} r_{\tau_j(\hat{\beta})}(\alpha, \beta)^2 &\geqslant \sum_{j=i(\beta)}^{i(\beta)+h-1} r_{\tau_j(\beta)}(\alpha(\beta), \beta)^2 = \min_{\alpha} D(\alpha, \beta) \\ &\geqslant \min_{\alpha} D(\alpha, \hat{\beta}) = \sum_{j=i(\hat{\beta})}^{i(\hat{\beta})+h-1} r_{\tau_j(\hat{\beta})}\big(\alpha(\hat{\beta}), \hat{\beta}\big)^2 \\ &= \sum_{j=i(\hat{\beta})}^{i(\hat{\beta})+h-1} r_{\tau_j(\hat{\beta})}(\hat{\boldsymbol{\theta}})^2, \end{aligned} \tag{4.3}$$

where the first inequality in (4.3) follows from (4.2). But (4.3) implies that the LS-estimate computed from the h-sample $\tau_{i(\hat{\beta})}(\hat{\beta}), \ldots, \tau_{i(\hat{\beta})+h-1}(\hat{\beta})$ is $\hat{\theta}$. □

4.2. *The simple algorithm*

We know from Proposition 2 that the indices of a minimizing h-sample are of the form $\{\tau_i(\beta), \ldots, \tau_{i+h-1}(\beta)\}$ for some $\beta \in \mathbb{R}$ and $i \in \{1, \ldots, n-h+1\}$. The antiranks of the residuals are piecewise constant, and they only change at $\beta_1, \ldots, \beta_{N-1}$. The algorithm is based on the following: At each β_k we compute the LS-estimates of those h-samples $(\tau_i(\beta), \ldots, \tau_{i+h-1}(\beta))$ that are changed at $\beta = \beta_k$.

Given a permutation $\boldsymbol{\tau}$, let $\hat{\boldsymbol{\theta}}^{hi}_{\mathrm{LS}}(\boldsymbol{\tau}) = (\hat{\alpha}^{hi}_{\mathrm{LS}}(\boldsymbol{\tau}), \hat{\beta}^{hi}_{\mathrm{LS}}(\boldsymbol{\tau}))$ be the LS-estimate computed from $\{(x_{\tau_j}, y_{\tau_j})\}_{j=i}^{i+h-1}$. The algorithm may be formulated as follows:

Exact LTS algorithm for calculating $\hat{\boldsymbol{\theta}} = (\hat{\alpha}, \hat{\beta})$:

1. Initialize: Compute $\beta_1 < \cdots < \beta_{N-1}$, $\boldsymbol{\tau}^1$, $\hat{\boldsymbol{\theta}}_{1i} := \hat{\boldsymbol{\theta}}^{hi}_{\mathrm{LS}}(\boldsymbol{\tau}^1)$, $D_{1i} := \sum_{j=i}^{i+h-1} r_{\tau_j}(\hat{\boldsymbol{\theta}}_{1i})^2$, $i = 1, \ldots, n-h+1$, and put $D_{\min} = \min_i D_{1i}$, $\hat{\boldsymbol{\theta}} = \arg\min_{\hat{\boldsymbol{\theta}}_{1i}} D_{1i}$.

For $k = 1$ **to** $N-1$ **do**

2. **For** all i such that $(\tau_i^{k+1}, \ldots, \tau_{i+h-1}^{k+1}) \neq (\tau_i^k, \ldots, \tau_{i+h-1}^k)$ **do**
   Compute $\hat{\boldsymbol{\theta}}_{k+1,i} := \hat{\boldsymbol{\theta}}^{hi}_{\mathrm{LS}}(\boldsymbol{\tau}^{k+1})$
   Compute $D_{k+1,i} := \sum_{j=i}^{i+h-1} r_{\tau_j^{k+1}}(\hat{\boldsymbol{\theta}}_{k+1,i})^2$
   If $D_{k+1,i} < D_{\min}$ **then**
   $D_{\min} \to D_{k+1,i}$ and $\hat{\boldsymbol{\theta}} \to \hat{\boldsymbol{\theta}}_{k+1,i}$

Some remarks on the algorithm:

1. As in Section 3, the computation of all β_i requires $O(n^2 \log n)$ operations and $O(n^2)$ storage. The vector τ^1 is obtained by ordering of $\{r_i(\beta)\}$ for some $\beta \in I_1$. This part requires $O(n \log n)$ operations. The quantities $\hat{\theta}_{1i}$ and D_{1i} are computed in $O(n)$ time for each i giving a total of $O(n^2)$ operations.

2. This step is analogous to Step 2 in Section 3. We compute τ^{k+1} by sorting of $\{r_i(\beta)\}$ for some $\beta \in I_{k+1}$. This gives a total of $O(n^3 \log n)$ operations.

3. In this step we have to determine for which pairs $(k+1, i)$ $\hat{\theta}_{k+1,i}$ and $D_{k+1,i}$ have to be computed, and then compute them. The condition is simply that $(\tau_i^{k+1}, \ldots, \tau_{i+h-1}^{k+1}) \neq (\tau_i^k, \ldots, \tau_{i+h-1}^k)$. A pair $(k+1, i)$ satisfies this iff either $i_l \leqslant i \leqslant j_l$ or $i_l \leqslant i-h+1 \leqslant j_l$ for some l. Observe that the indices $i_1 < j_1 < i_2 < \cdots < j_{M_k}$ may be obtained in $O(n)$ time when comparing τ^k with τ^{k+1} (which has been found in Step 2). It thus takes $O(n)$ operations for each k and all in all $O(n^3)$ steps to find out for which pairs $(k+1, i)$ further computation is needed. For each fixed k there are at most $2N_k$ such pairs, which gives $2\Sigma N_k = O(n^2)$ pairs when ranging over k. Since each $\hat{\theta}_{k+1,i}$ and $D_{k+1,i}$ may be computed in $O(n)$ time, the total number of operations becomes $O(n^3)$.

4.3. Numerical examples

Figure 4 displays an artificial data set consisting of 13 points together with nine fitted regression lines, one using the exact algorithm described here, three using

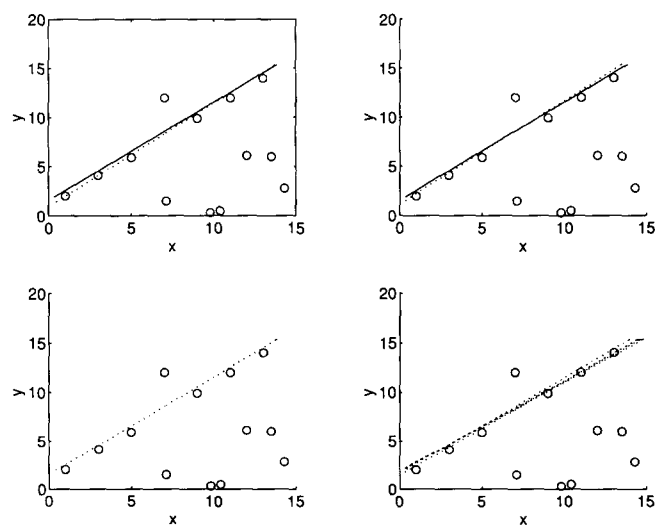

Fig. 4. Simple linear regression. LTS estimates for an artificial data set with $n = 13$ and $h = 7$, using an exact algorithm (solid line, $D_{\min} = 13.8557$), the PROGRESS algorithm without intercept adjustment (dotted line, upper left, $D_{\min} = 14.64$), with one intercept adjustment (dotted line, upper right, $D_{\min} = 14.17$) and with several intercept adjustments (dotted line, lower left, $D_{\min} = 13.8571$) and the RANDDIR algorithm with 100 elemental subsets (dotted lines, lower right, $D_{\min} = 14.827, 15.262, 14.520, 14.832, 13.884$).

different versions of the PROGRESS algorithm (without, with one and with several intercept adjustments respectively, as described in Section 1), and finally five RANDDIR fits with 100 randomly chosen elemental subsets. The truncation point h is 7 in all cases. The PROGRESS lines with zero and one intercept adjustments differ noticeably form the exact LTS fit, whereas the PROGRESS line with several intercept adjustments is practically indistinguishable from it. The RANDDIR fits approximate the true regression line better than the two simpler PROGRESS algorithms, but not as good as the PROGRESS algorithm with several intercept adjustments.

The next example, the stars data set (cf. Rousseeuw and Leroy, 1987, p. 27), shows for 47 stars the light intensity against the effective temperature at the surface on a log–log scale. The data points are shown in Figure 5 together with four regression lines: Exact, PROGRESS without intercept adjustments and two RANDDIR fits with 500 randomly chosen elemental subsets. The truncation point h was 24 in all cases. The PROGRESS fit with one intercept adjustment almost coincides with the simple PROGRESS fit, whereas the PROGRESS fit with several adjustments was almost identical to the exact fit. Of the two RANDDIR fits, one was even closer to the exact one than the most accurate PROGRESS line based on several intercept adjustments, whereas the other was further away.

In order to obtain a more quantitative comparison between the various algorithms, we undertook a small simulation study. The main bulk of $[n(1-\epsilon^*)]$ data points (x_i, y_i) were generated from a bivariate normal distribution with mean (0, 0) and diagonal covariance matrix diag(1, σ^2). The remaining outlying

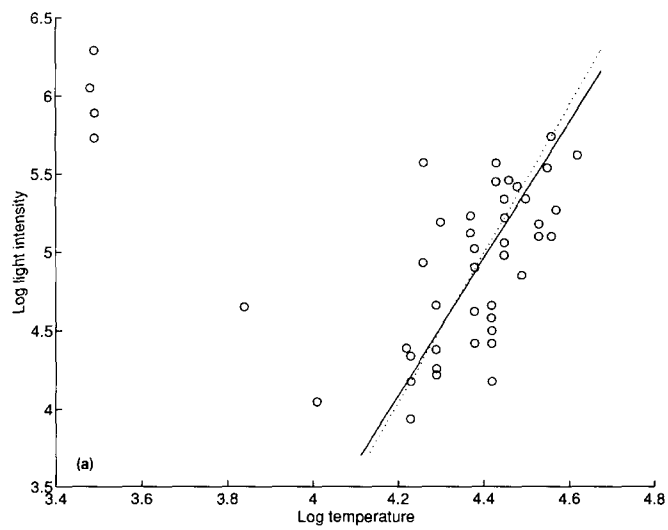

Fig. 5a. Simple linear regression. LTS estimates for the stars data with $n = 47$ and $h = 24$, using an exact algorithm (solid line, $D_{\min} = 0.7324$), and the PROGRESS algorithm (dotted line, $D_{\min} = 0.7431$).

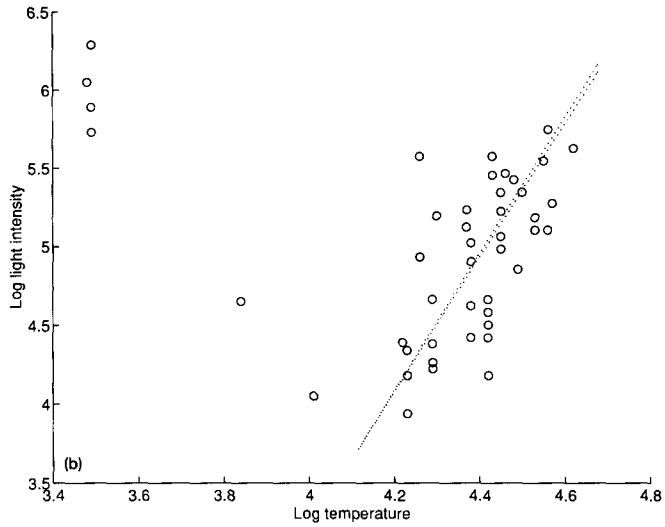

Fig. 5b. The stars data ($n = 47$, $h = 24$), using two RANDDIR fits with 500 elemental subsets ($D_{\min} = 0.7324, 0.7389$).

observations were generated form another normal distribution with mean (μ, μ) and covariance matrix diag$(\frac{1}{9}, \frac{1}{9})$. We computed exact LTS estimates $\{(\hat{\alpha}_i, \hat{\beta}_i)\}_{i=1}^{N_{MC}}$ for $N_{MC}(= 200)$ Monte Carlo replicates of the data, and corresponding estimates $\{(\bar{\alpha}_i, \bar{\beta}_i)\}_{i=1}^{N_{MC}}$ for each of the studied approximative LTS algorithms. Table 1 shows values of

$$\Delta\alpha = \sqrt{\frac{n}{N_{MC}} \sum_{i=1}^{N_{MC}} (\bar{\alpha}_i - \hat{\alpha}_i)^2}$$

Table 1
Comparison between different approximative LTS algorithms; PROGRESS without intercept adjustment (P), with one intercept adjustment (P1) and several intercept adjustments (P2), RANDDIR with $N_{rand} = \binom{n}{2}$ (R1) and RANDDIR with $N_{rand} = 5\binom{n}{2}$ (R2), where N_{rand} is the number of randomly chosen elemental subsets

	n	ϵ^*	σ	μ	P	P1	P2	R1	R2
$\Delta\alpha$	20	0.0	0.5	2	0.208	0.158	0.025	0.299	0.125
$\Delta\beta$	20	0.0	0.5	2	0.299	0.299	0.045	0.411	0.321
$\Delta\alpha$	20	0.2	0.5	2	0.508	0.477	0.081	0.610	0.446
$\Delta\beta$	20	0.2	0.5	2	0.995	0.995	0.051	1.082	0.945
$\Delta\alpha$	20	0.4	0.5	2	0.482	0.386	0.037	0.544	0.108
$\Delta\beta$	20	0.4	0.5	2	0.238	0.238	0.045	0.302	0.049
$\Delta\alpha$	50	0.2	0.5	2	0.346	0.290	0.014	0.266	0.215
$\Delta\beta$	50	0.2	0.5	2	0.537	0.537	0.015	0.488	0.487
$\Delta\alpha$	20	0.2	0.5	5	0.335	0.267	0.014	0.203	0.087
$\Delta\beta$	20	0.2	0.5	5	0.402	0.402	0.030	0.657	0.242
$\Delta\alpha$	20	0.2	2.0	2	1.014	0.871	0.170	1.016	0.172
$\Delta\beta$	20	0.2	2.0	2	0.613	0.613	0.074	0.514	0.047

and

$$\Delta\beta = \sqrt{\frac{n}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} \left(\bar{\beta}_i - \hat{\beta}_i\right)^2}$$

for each algorithm and different combinations of n, ϵ^*, σ and μ. The truncation point h was chosen to $[0.6n]$.

Generally, the PROGRESS algorithm with several intercept adjustments (P2) gave the best results, followed by RANDDIR. (Of course, the RANDDIR algorithm can be made arbitrarily good by increasing the number of randomly chosen elemental subsets.) There is a theoretical reason why our exact fit is very close to the P2 fit. The latter algorithm computes intercept adjustments over the grid $\beta_1, \ldots, \beta_{N-1}$, whereas the exact algorithm can be viewed as if intercept adjustments are made for all $\beta \in \mathbb{R}$. For large n, the grid is so fine that the difference is negligible. However, for small n, the difference can be quite noticeable. We conjecture that the RANDDIR algorithm would have (at least) similar performance as P2 if intercept adjustment is performed for each elemental subset. Of course, an alternative to this computationally more intensive refinement of RANDDIR is simply to take more random elemental subsets.

5. The refined algorithm

Considerable savings in computation time and storage are possible in Sections 3 and 4. The price for this is a more complicated algorithm. We confine ourselves to describing how to modify the algorithm for regression with intercept. The modification of the algorithm in Section 3 may be done similarly.

The basic idea is to compute $\boldsymbol{\tau}^{k+1}$, $\hat{\boldsymbol{\theta}}_{k+1,i}$ and $D_{k+1,i}$ recursively from $\boldsymbol{\tau}^k$, $\hat{\boldsymbol{\theta}}_{ki}$ and D_{ki} respectively. This reduces the computation time from $O(n^3 \log n)$ to $O(n^2 \log n)$. The amount of storage may also be decreased from $O(n^2)$ to $O(n)$ by computing β_{k+1} recursively from $\boldsymbol{\tau}^k$ and β_k for each k in Step 2. This implies that $\beta_1 < \cdots < \beta_{N-1}$ don't have to be computed and stored at once in Step 1. More precisely, the steps of the algorithm are modified as follows:

1. The computation of $\beta_1 < \cdots < \beta_{N-1}$ is postponed until Step 2. This means that we cannot find $\boldsymbol{\tau}^1$ by sorting $\{r_i(\beta)\}$ for some $\beta \in I_1 = (-\infty, \beta_1)$, since we don't know β_1 yet. However, as $\beta \to -\infty$, sorting $\{r_i(\beta)\}$ becomes equivalent to ordering $\{x_i\}$, where observations with equal x-coordinates are ordered with respect to their y-coordinates.

2. We start with a schematic description on how to compute $\boldsymbol{\tau}^{k+1}$ recursively from $\boldsymbol{\tau}^k$:

$\boldsymbol{\tau}^{k+1} = \boldsymbol{\tau}^k$
For $l = 1$ **to** M_k **do**
  For $m = i_l$ **to** j_l **do**
    $\tau_m^{k+1} = \tau_{j_l + i_l - m}^k$

It remains to compute M_k, i_1, $j_1, \ldots, j_{M_k}$. Given β_{k-1} (with $\beta_0 = -\infty$) and $\boldsymbol{\tau}^k$ we only have to compute and store the $n-1$ intersection points $\bar{\beta}_{\tau_i^k \tau_{i+1}^k}$, $i = 1, \ldots, n-1$. The reason is that β_k may be found within this "sublist" of intersection points as the smallest value greater than β_{k-1}, or more precisely; $\bar{\beta}_{\tau_i^k \tau_{i+1}^k} = \beta_k$ exactly when $i \in \{i_l, \ldots, j_l - 1\}$ for some $l = 1, \ldots, M_k$. So the union of all pairs $(i, i+1)$ for which $\bar{\beta}_{\tau_i^k \tau_{i+1}^k} = \beta_k$ is $\cup_{1 \leqslant l \leqslant M_k} \{i_l, \ldots, j_l\}$. At each iteration of Step 2 we keep an ordered list of the $n-1$ quantities $(\bar{\beta}_{\tau_i^k \tau_{i+1}^k}, i)$, $i = 1, \ldots, n-1$. The ordering is made w.r.t. the first component, and if two vectors have equal first coordinate they are ordered w.r.t. the second one. It requires $O(N_k)$ operations to extract β_k, M_k and i_1, j_1, $i_2, \ldots, j_{M_k}$ from this list. We also have to update this ordered list at each iteration k. There are at most $N_k + M_k$ $\bar{\beta}$-values that have to be replaced from the old list, namely all $(\bar{\beta}_{\tau_i^k \tau_{i+1}^k}, i)$ such that $i_l - 1 \leqslant i \leqslant j_l$ for some $l = 1, \ldots, M_k$. Since each new value may be inserted into the ordered sequence of intersection points in $O(\log n)$ time using binary search, the whole list can be updated after $O(N_k \log n)$ operations. In conclusion, the total number of operations in Step 2 becomes $O(\sum_k N_k \log n) = O(n^2 \log n)$.

3. In this step we now compute $\hat{\boldsymbol{\theta}}_{k+1,i}$ and $D_{k+1,i}$ recursively from $\hat{\boldsymbol{\theta}}_{ki}$ and D_{ki} respectively, for all $1 \leqslant i \leqslant n-h+1$ such that $(\tau_i^{k+1}, \ldots, \tau_{i+h-1}^{k+1}) \neq (\tau_i^k, \ldots, \tau_{i+h-1}^k)$. We have already computed M_k, $i_1, \ldots, j_{M_k}$ in Step 2, and this information may be used in order to derive for which values of i $\hat{\boldsymbol{\theta}}_{ki}$ and D_{ki} have to be changed: iff either $i \in \{i_l + 1, \ldots, j_l\}$ or $i + h - 1 \in \{i_l, \ldots, j_l - 1\}$ for some $1 \leqslant l \leqslant M_k$. We will use the fact that both $\hat{\boldsymbol{\theta}}_{ki}$ and D_{ki} may be computed from the five quantities $S_x^{hi}(\boldsymbol{\tau}^k) = \sum_{j=i}^{i+h-1} x_{\tau_j^k}$, $S_y^{hi}(\boldsymbol{\tau}^k) = \sum_{j=i}^{i+h-1} y_{\tau_j^k}$, $S_{xx}^{hi}(\boldsymbol{\tau}^k) = \sum_{j=i}^{i+h-1} x_{\tau_j^k}^2$, $S_{xy}^{hi}(\boldsymbol{\tau}^k) = \sum_{j=i}^{i+h-1} x_{\tau_j^k} y_{\tau_j^k}$ and $S_{yy}^{hi}(\boldsymbol{\tau}^k) = \sum_{j=i}^{i+h-1} y_{\tau_j^k}^2$. We will now show how to update $S_{xx}^{hi}(\boldsymbol{\tau}^{k+1})$ from $S_{xx}^{hi}(\boldsymbol{\tau}^k)$:

Recursive computation of S_{xx}-quantities in Step 3.
$S_{xx}^{hi}(\boldsymbol{\tau}^k) \to S_{xx}^{hi}(\boldsymbol{\tau}^{k+1})$, $i = 1, \ldots, n-h+1$
For $l = 1$ **to** M_k **do**
  $m_0 = \max(h, i_l)$
  For $m = m_0$ **to** $j_l - 1$ **do**
    $S_{xx}^{h,m-h+1}(\boldsymbol{\tau}^{k+1}) \to S_{xx}^{h,m-h+1}(\boldsymbol{\tau}^{k+1}) + \sum_{j=i_l}^{m} x_{\tau_j^{k+1}}^2 - \sum_{j=i_l+j_l-m}^{j_l} x_{\tau_j^{k+1}}^2$
  $m_0 = \min(n-h+1, j_l)$
  For $m = i_l + 1$ **to** m_0 **do**
    $S_{xx}^{hm}(\boldsymbol{\tau}^{k+1}) \to S_{xx}^{hm}(\boldsymbol{\tau}^{k+1}) + \sum_{j=m}^{j_l} x_{\tau_j^{k+1}}^2 - \sum_{j=i_l}^{i_l+j_l-m} x_{\tau_j^{k+1}}^2$

The first line represents no computation, it is just for convenience that we change names on the S_{xx}-quantities. We see that the number of computations in Step 3 for updating S_{xx} is $O(\sum_{k,l} N_{k,l}^2) = O(n^2)$. The last estimate holds since $\sum_{k,l} \binom{N_{k,l}}{2} \leqslant \binom{n}{2}$, which follows form the fact that each pair of lines in the dual plot intersect at no more than one value of β. Hence, the total amount of computation becomes $O(n^2)$ in this step (since the updating of S_{xy}, S_{yy}, S_x and S_y require the same order number of computations). The amount of storage is $O(n)$, since we need to store $\{S_{xx}^{hi}, S_{xy}^{hi}, S_{yy}^{hi}, S_x^{hi}$ and $S_y^{hi}; i = 1, \ldots, n-h+1\}$.

6. Nonlinear regression

Our algorithm may also be extended to the nonlinear regression model

$$y_i = g(\alpha, \beta, x_i) + e_i, \quad i = 1, \ldots, n, \tag{6.1}$$

where g is some known nonlinear function that is strictly monotone in its first argument. We confine ourselves to the case when α is unknown, extending the algorithm of Section 4. When $\alpha = 0$, the algorithm in Section 3 may be extended similarly. Stromberg (1993b) describes a method for computing the LMS-estimate exactly, in the special case when $g(\alpha, \beta, x) = \tilde{g}(\alpha + \beta x)$, with $\tilde{g}$ strictly monotone.

The dual plot now consists of the curves $\beta \to r_i(\beta) = y_i - g(0, \beta, x_i)$. Consider the equation

$$r_i(\beta) = r_j(\beta) \quad \Leftrightarrow \quad g(0, \beta, x_i) - g(0, \beta, x_j) = y_i - y_j, \qquad i \neq j, \tag{6.2}$$

which is nonlinear and may have an arbitrary number of solutions (including 0). Assuming that (6.2) has a finite number of solutions for each pair (i, j) we may order the solutions (not including values occuring several times) as $\beta_1 < \cdots < \beta_{N-1}$. The vector of antiranks $\tau(\beta)$ (still computed from $\{r_i(\beta)\}_{i=1}^n$) is then piecewise constant on the intervals $I_k = (\beta_{k-1}, \beta_k)$, $k = 1, \ldots, N$. Denote this constant value by $\tau^k = (\tau_1^k, \ldots, \tau_n^k)$. (We use the same notation as in Section 4 and hope that this causes no confusion.)

The LTS-estimate, if it exists, is still given by (1.3)–(1.4), but with $r_i(\theta) = y_i - g(\alpha, \beta, x_i)$. Moreover, Proposition 2 carries over (even though the proof is changed a little), and this means that we only have to compute the nonlinear LS-estimate $\hat{\theta}_{\mathrm{NLS}}^{hi}(\tau^k)$ of $\theta = (\alpha, \beta)$ from the h-sample $(x_{\tau_i^k}, y_{\tau_i^k}), \ldots, (x_{\tau_{i+h-1}^k}, y_{\tau_{i+h-1}^k})$, for all $1 \leqslant k \leqslant N$, $1 \leqslant i \leqslant n - h + 1$. (And as in Section 4, given k this only has to be done for some values of i.) The algorithm in Section 4 carries over with only the change that $\hat{\theta}_{ki} = \hat{\theta}_{\mathrm{NLS}}^{hi}(\tau^k)$, and $\hat{\theta}_{ki}$ may be found by Newton–Raphson's algorithm in $O(n)$ steps, if the number of iterations is no larger than a fixed predefined number. On the other hand D_{ki} is derived as before from $\{r_j(\hat{\theta}_{ki})\}_{j=i}^{i+h-1}$ in $O(n)$ steps. If also the number of solutions of (6.2) is upper bounded by some constant C, we obtain $N \leqslant C\binom{n}{2} + 1 = O(n^2)$. Let N_k, M_k and $N_{k,l}$ have the same meaning as in Section 4. The number of pairs (k, i) for which $\hat{\theta}_{ki}$ and D_{ki} have to be computed is then $O(\sum_{k=1}^{N-1} N_k) = O(n^2)$. Hence, the total number of operations required for computing all $\hat{\theta}_{ki}$ and D_{ki} is $O(n^3)$. Since the computation time of Step 2 is $O(n^3 \log n)$ as in Section 4, the overall computation time is $O(n^3 \log n)$.

The refined algorithm for linear regression in Section 5 may also be extended to the nonlinear case, with the following modifications:

- The calculation of τ^1 depends on how $g(0, \beta, x)$ varies with x when $\beta \to -\infty$.

- $\hat{\boldsymbol{\theta}}_{k+1,i}$ cannot be found recursively from $\hat{\boldsymbol{\theta}}_{ki}$, but has to computed "from scratch".
- We also need to compute $D_{k+1,i}$ from scratch.

This means that the computation time is still $O(n^3)$ in Step 3. On the other hand, the recursive computation of $\boldsymbol{\tau}^{k+1}$ from $\boldsymbol{\tau}^k$ may be carried over to the nonlinear case. Since this requires only $O(n^2 \log n)$ operations, the overall computation time becomes $O(n^3)$.

7. Concluding remarks

In principle, it is possible to use dual plots in multiple linear regression for computing the LTS-estimate. This would subdivide the parameter space into polygon-shaped regions (generalizing $I_1^+, \ldots, I_{N^+}^+$ and $I_1, \ldots, I_N$) for each of which LS-estimates are computed. For the model including an intercept, these regions are formed by the $O(n^2)$ $(p-1)$-dimensional hyperplanes in $\mathbb{R}^p$ defined by $r_i(\boldsymbol{\theta}) = r_j(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is now the p-dimensional vector of slope parameters. For models without intercept, $\boldsymbol{\theta}$ is the vector of regression parameters, and we also have to include the hyperplanes $r_i(\boldsymbol{\theta}) = -r_j(\boldsymbol{\theta})$. It can be shown by combinatorial geometry (cf. e.g. Edelsbrunner, 1987), that the number of regions formed by the hyperplanes is $O(n^{2p})$, which grows quickly with n for $p > 1$. It is also difficult to define the partition properly, since we can no longer order the regions as in the one-dimensional case.

We formulated the improved algorithm in Section 5 recursively, in order to reduce the order number of computations. (Rousseeuw and Leroy, 1987, Chapter 4, also formulated their exact LTS location algorithm recursively.) Apart from the increased complexity of the algorithm, there is also a problem with unstability. First, roundoff errors propagate in the recursive computations of $\hat{\boldsymbol{\theta}}_{ki}$ (or $\hat{\beta}_k$) and D_{ki} (or D_k) in Step 3, so it is important to have high accuracy in the computations. An even more crucial point is the recursive computation of the antirank vectors $\boldsymbol{\tau}^k$ (or $\boldsymbol{\sigma}^k$) in Step 2. If a mistake is done in this step for some k, the whole algorithm sidetracks. The important thing is to order all the intersection points $\bar{\beta}_{ij}$ (or $\tilde{\beta}_{ij}$) correctly, in order to get the true values of N_k, M_k, $i_1, \ldots, j_{M_k}$. The risk of a numerical mistake is greatest when several intersection points are very close or equal ($N_k \geqslant 3$ for some k). Since the number of intersection points is of the order $O(n^2)$, the situation quickly becomes worse when the sample size increases. A way out of this dilemma is to represent all the data points (x_i, y_i) and intersection points $\bar{\beta}_{ij}$ by rational numbers instead of floating point numbers, so that all calculations and comparisons can be made exactly. If a floating point mode is used, it is still possible to correct all possible mistakes made in the calculation of $\boldsymbol{\tau}^k$ ($\boldsymbol{\sigma}^k$). We simply check that $\{r_{\tau_i^k}(\beta)\}_{i=1}^n$ is an ordered sequence for some $\beta \in I_k$. However, this requires $O(n)$ operations for each k, and the total number of calculations becomes $O(n^3)$.

Another issue is the generalization of the exact LTS algorithm to computation of other estimates. In principle, any estimator of the form (1.3) can be computed in the same way, if the objective function has the form

$$D(\boldsymbol{r}) = \tilde{D}(|r|_{\sigma_1}, \ldots, |r|_{\sigma_h}), \tag{7.1}$$

a function of the h smallest absolute values that is increasing in each argument. In the rest of this section we confine ourselves to discussing how the algorithm in Section 4 for the model (1.1) with intercept may be extended. We only have to modify the computation of $\hat{\boldsymbol{\theta}}_{ki}$ in the LTS algorithm, so that

$$\hat{\boldsymbol{\theta}}_{ki} := \arg\min_{\boldsymbol{\theta}} \tilde{D}(|r_{\tau_i^k}(\boldsymbol{\theta})|, \ldots, |r_{\tau_{i+h-1}^k}(\boldsymbol{\theta})|). \tag{7.2}$$

First, we see that the LTS-estimator corresponds to

$$\tilde{D}(|r_1|, \ldots, |r_h|) = \sum_{i=1}^{h} r_i^2,$$

so that $\hat{\boldsymbol{\theta}}_{ki}$ in (7.2) is the LS-estimator of $\boldsymbol{\theta}$ computed from $\{x_{\tau_j^k}, y_{\tau_j^k}\}_{j=i}^{i+h-1}$. The main reason why we have concentrated on the LTS-estimator in this paper is that the LS-estimator is easy to compute, both from scratch and recursively. Another example is the R-estimator considered by Hössjer (1994), where

$$D(\boldsymbol{r}) = \sum_{i=1}^{h} a_n(i)|r_{\sigma_i}|,$$

and $a_n(1), \ldots, a_n(h)$ are nonnegative scores (with $a_n(h) > 0$). In this case

$$\tilde{D}(|r_1|, \ldots, |r_h|) = \sum_{i=1}^{h} a_n(R_i)|r_i|, \tag{7.3}$$

where R_i is the rank of $|r_i|$ among $|r_1|, \ldots, |r_h|$. The function $\tilde{D}(|r_1(\boldsymbol{\theta})|, \ldots, |r_h(\boldsymbol{\theta})|)$ is convex in $\boldsymbol{\theta}$ if $0 \leqslant a_1(1) \leqslant \cdots \leqslant a_n(h)$ (McKean and Schrader, 1980, Theorem 1), and $\hat{\boldsymbol{\theta}}_{ki}$ may be computed by iterating the reweighted least-squares algorithm described by Cheng and Hettmansperger (1983). Observe also that the gradient of $\tilde{D}$ is a signed rank statistic (cf. e.g. Hettmansperger and McKean, 1983).

The LMS-estimator is (essentially) a special case of the least quantile of squares (LQS) estimator, which has an objective function $D(\boldsymbol{r}) = |r_{\sigma_h}|^2$. The square is unimportant for the minimization, so we may as well put $D(\boldsymbol{r}) = |r_{\sigma_h}|$, and

$$\tilde{D}(r_1, \ldots, r_h) = \max_{1 \leqslant i \leqslant h} |r_i|. \tag{7.4}$$

In this case $\hat{\boldsymbol{\theta}}_{ki}$ becomes a Chebyshev (or minimax) estimator. Hence, the exact LQS-estimate may be found by means of the algorithm in Section 4. Note that (7.4) is a special case of (7.3), with $a_n(1) = \cdots = a_n(h-1) = 0$ and $a_n(h) = 1$. Exact algorithms for the LQS-estimate have been found by Steele and Steiger (1986), Souvaine and Steele (1986) and Stromberg (1993a).

Acknowledgement

The author wishes to thank Peter Rousseeuw, Arnold Stromberg and two referees for valuable comments. All programs were written in MATLAB code.

References

Afifi, A.A. and S.P. Azen, *Statistical analysis, a computer oriented approach*, 2nd ed. (Academic Press, New York, 1979).

Atkinson, A.C. and S. Weisberg, Simulated annealing for the detection of multiple outliers using least squares and least median of squares fitting, in: W. Stahel and S. Weisberg (Eds.), *Directions in robust statistics and diagnostics* (New York, Springer-Verlag, 1991).

Cheng, K-S. and T.P. Hettmansperger, Weighted least squares estimates, *Commun. Statist.-Theor. Meth.* **12**(9) (1983) 1069–1086.

Donoho, D.L. and P.J. Huber, The notion of breakdown point, in: P. Bickel, K. Doksum and J.L. Hodges, Jr. (Eds.), *A festschrift for Erich Lehmann* (Wadsworth, Belmont, CA, 1983).

Hettmansperger, T.P. and J.W. McKean, A geometric interpretation of inferences based on ranks in the linear model, *J. Amer. Statist. Assoc.* **78** (1983) 885–893.

Hawkins, D.M., The feasible set algorithm for least trimmed squares regression, to appear in *Comput. Statist. Data Anal.* (1993).

Hössjer, O., Rank-based estimates in the linear model with high breakdown point, to appear in *J. Amer. Statist. Assoc.* (1994).

Johnstone, I.M. and P.F. Velleman, The resistant line and related regression methods, *J. Amer. Stat. Assoc.* **80** (1985) 1041–1054.

McKean, J.W. and R.M. Schrader, The geometry of robust procedures in linear models, *J. Roy. Statist. Soc. B* **42** (1990) 366–371.

Mount, D.M. and N.S. Netanyahu, Computationally efficient algorithms for a highly robust line estimator, Technical Report (Center for Automation Research, University of Maryland, 1992).

Rosenqvist, A., Robust simple regression using the Hough transform, Technical Report (Dept. of Mathematical Statistics, Lunds University, Sweden, 1992).

Rousseeuw, P.J., Least median of squares regression, *J. Amer. Stat. Assoc.* **79** (1984) 871–880.

Rousseeuw, P.J., Multivariate estimation with high breakdown point, in: W. Grossmann, G. Pflug, I. Vincze and W. Wertz (Eds.), *Mathematical statistics and applications, Vol. B*, (Reidel, Dordrecht, The Netherlands, 1985) 283–297.

Rousseeuw, P.J. and A. Leroy, *Robust regression and outlier detection* (Wiley, New York, 1987).

Rousseeuw, P.J. and V.Y. Yohai, Robust regression by means of *S*-estimators, in: J. Franke, W. Härdle and R.D. Martin (Eds.), *Robust and Nonlinear Time Series Analysis*, Lecture Notes in Statistics No. 26 (Springer Verlag, New York, 1984) pp. 256–272.

Ruppert, D., Computing *S*-estimators for regression and multivariate location/dispersion, *J. Comput. Graph. Statist.* **1**(3) (1992) 253–270.

Souvaine, D.L. and J.M. Steele, Time and space efficient algorithms for least median of squares regression (Technical Report, Princeton University, 1986).

Steele, J.M. and W.L. Steiger, Algorithms and complexity for least median of squares regression, *Discrete Appl. Math.* **14** (1986) 93–100.

Stromberg, A.J., Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression, *SIAM J. Scient. Statist. Comput.* **14** (1993a) 1289–1299.

Stromberg, A.J., High breakdown estimators in nonlinear regression, in: Y. Dodge. (Ed.), *L_1-statistical analysis and related methods* (Elsevier Science Publishers, 1993b) pp. 103–112.