Computing LTS Regression for Large Data Sets

Peter J. Rousseeuw and Katrien Van Driessen

Abstract

Least trimmed squares (LTS) regression is based on the subset of h cases (out of n) whose least squares fit possesses the smallest sum of squared residuals. The coverage h may be set between $n/2$ and n. The LTS method was proposed by Rousseeuw (1984, p. 876) as a highly robust regression estimator, with breakdown value $(n-h)/n$. It turned out that the computation time of existing LTS algorithms grew too fast with the size of the data set, precluding their use for data mining. Therefore we develop a new algorithm called FAST-LTS. The basic ideas are an inequality involving order statistics and sums of squared residuals, and techniques which we call 'selective iteration' and 'nested extensions'. We also use an intercept adjustment technique to improve the precision. For small data sets FAST-LTS typically finds the exact LTS, whereas for larger data sets it gives more accurate results than existing algorithms for LTS and is faster by orders of magnitude. Moreover, FAST-LTS runs faster than all programs for least median of squares (LMS). The new algorithm makes the LTS method available as a tool for robust regression in large data sets, e.g. in a data mining context.

KEY WORDS: Breakdown value; Linear model; Outlier detection; Regression; Robust estimation.

Peter J. Rousseeuw is Professor, Department of Mathematics and Computer Science, Universitaire Instelling Antwerpen (UIA), Universiteitsplein 1, B-2610 Wilrijk, Belgium, and Katrien Van Driessen is Assistant, Faculty of Applied Economics, Universitaire Faculteiten Sint Ignatius Antwerpen (UFSIA), Prinsstraat 13, B-2000 Antwerp, Belgium. We wish to thank Doug Hawkins and José Agulló for making their programs available to us, and David Rocke and the group of S.C. Odewahn at Cal Tech for introducing us to the DPOSS data.

1 Introduction

Consider the linear model

$$y_i = x_{i1}\theta_1 + \cdots + x_{ip}\theta_p + e_i \qquad i = 1, \ldots, n$$

where the data points are of the form $(\boldsymbol{x_i}, y_i) = (x_{i1}, \ldots, x_{ip}, y_i)$, with $x_{ip} = 1$ for regression with an intercept term. Many estimators of the parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$ break down in the presence of outliers. There are several kinds of outliers, for which we will follow the terminology of Rousseeuw and van Zomeren (1990). A point $(\boldsymbol{x_i}, y_i)$ which does not follow the linear pattern of the majority of the data but whose $\boldsymbol{x_i}$ is not outlying is called a *vertical outlier.* A point $(\boldsymbol{x_i}, y_i)$ whose $\boldsymbol{x_i}$ is outlying is called a *leverage point.* We say that it is a *good* leverage point when $(\boldsymbol{x_i}, y_i)$ follows the pattern of the majority, and a *bad* leverage point otherwise. Summarizing, a data set can contain four types of points: regular observations, vertical outliers, good leverage points, and bad leverage points. Of course, most data sets do not have all four types.

For simple regression the data are bivariate and can be displayed in a scatterplot, so we can easily detect outlying observations by visual inspection. But for data with several explanatory variables, this is no longer possible. Then the residuals based on a robust regression detect points $(\boldsymbol{x_i}, y_i)$ that deviate from the linear pattern. Moreover, leverage points can be detected by computing the robust distances of $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ as proposed in (Rousseeuw and Van Zomeren 1990), as will be illustrated in Section 6 below.

Several positive-breakdown methods for robust regression have been proposed, such as the least median of squares (LMS) method of Rousseeuw (1984). The LMS is defined by minimizing $(r^2)_{h:n}$ where $(r^2)_{1:n} \leq (r^2)_{2:n} \leq \cdots \leq (r^2)_{n:n}$ are the ordered squared residuals, and where $h = [(n + p + 1)/2]$. Here p is again the number of coefficients, including the intercept term. The LMS attains the highest possible breakdown value, namely $([(n - p)/2] + 1)/n$. This means that the LMS fit stays in a bounded region whenever $[(n - p)/2]$ or fewer observations are replaced by arbitrary points. Positive-breakdown methods such as LMS regression are increasingly being used in practice, e.g. in finance, chemistry, electrical engineering, process control, and computer vision (Meer et al. 1991). For a survey of positive-breakdown methods and some substantive applications, see (Rousseeuw 1997).

The basic resampling algorithm for approximating the LMS, called PROGRESS, was proposed in (Rousseeuw and Leroy 1987) and further developed in (Rousseeuw and Hubert

1997). This algorithm considers a trial subset of p observations and calculates the linear fit passing through them. This procedure is repeated many times, and the fit with the lowest median of squared residuals is retained. For small data sets it is possible to consider all p-subsets, whereas for larger data sets many p-subsets are drawn at random.

Several algorithms have been proposed to compute the LMS regression coefficients exactly (Steele and Steiger 1986, Stromberg 1993). The currently fastest exact algorithm by Agulló (1997a,b) is based on a branch and bound procedure that selects the optimal h-subset without requiring the inspection of all h-subsets. His algorithm is feasible for n up to about 100 and p up to 5. But since for most data sets the exact algorithms would take too long, LMS regression is typically computed by approximate algorithms based on PROGRESS, e.g. in S-Plus (the function 'lmsreg') and in SAS/IML Version 7 (the function 'LMS').

Nowadays we think that the LMS estimator should be replaced by the least trimmed squares (LTS) estimator, which was also proposed in (Rousseeuw 1984, page 876). Its objective is to minimize

$$\Sigma_{i=1}^{h}(r^2)_{i:n} \tag{1.1}$$

where $(r^2)_{1:n} \leq \cdots \leq (r^2)_{n:n}$ are the ordered squared residuals. This is equivalent to finding the h-subset with smallest least squares objective function. The LTS regression estimate is then the least squares fit to these h points. The breakdown value of LTS with $h = [(n+p+1)/2]$ equals that of the LMS. Moreover, LTS regression has several advantages over LMS. Its objective function is more smooth, making LTS less 'jumpy' (i.e., sensitive to local effects) than LMS. Its statistical efficiency is better, because the LTS estimator is asymptotically normal (Hössjer 1994) whereas the LMS estimator has a lower convergence rate (Rousseeuw 1984). This also makes the LTS more suitable than the LMS as a starting point for two-step estimators such as the MM-estimator (Yohai 1987) and generalized M-estimators (Simpson, Ruppert and Carroll 1992, Coakley and Hettmansperger 1993).

In spite of all these advantages, until now the LTS estimator has been applied less often because it was harder to compute than the LMS. However, in this paper we will construct a new LTS algorithm which is actually *faster* than all existing LMS algorithms. The new LTS algorithm can deal with a sample size n in the tens of thousands or more. This should make the LTS the method of choice.

Example 1: DPOSS data. In their report, Odewahn et al. (1998) describe the Digitized

Palomar Sky Survey (DPOSS) data they are working on at the California Institute of Technology. The data consist of many celestial objects or light sources (mainly stars and galaxies) for which nine characteristics in the blue, red, and near-infrared bands are measured. To analyze their data, these astronomers seek collaboration with statisticians. They gave us access to a part of their database, containing the 56,744 stars for which all the characteristics in the blue colour (the F band) are available. The variable on the vertical axis in Figure 1 is MAperF, the aperture magnitude (which is a kind of brightness measure) of the star in the F band. The variable on the horizontal axis is based on spectroscopy and called csfF. On the data set with n =56,744 and $p = 2$ we have applied the classical LS regression as well as the LTS regression method, yielding the two lines in Figure 1. We see that the LS line is affected by a group of stars with outlying x-coordinate, whereas the LTS line fits the (decreasing) trend of the vast majority of the stars.

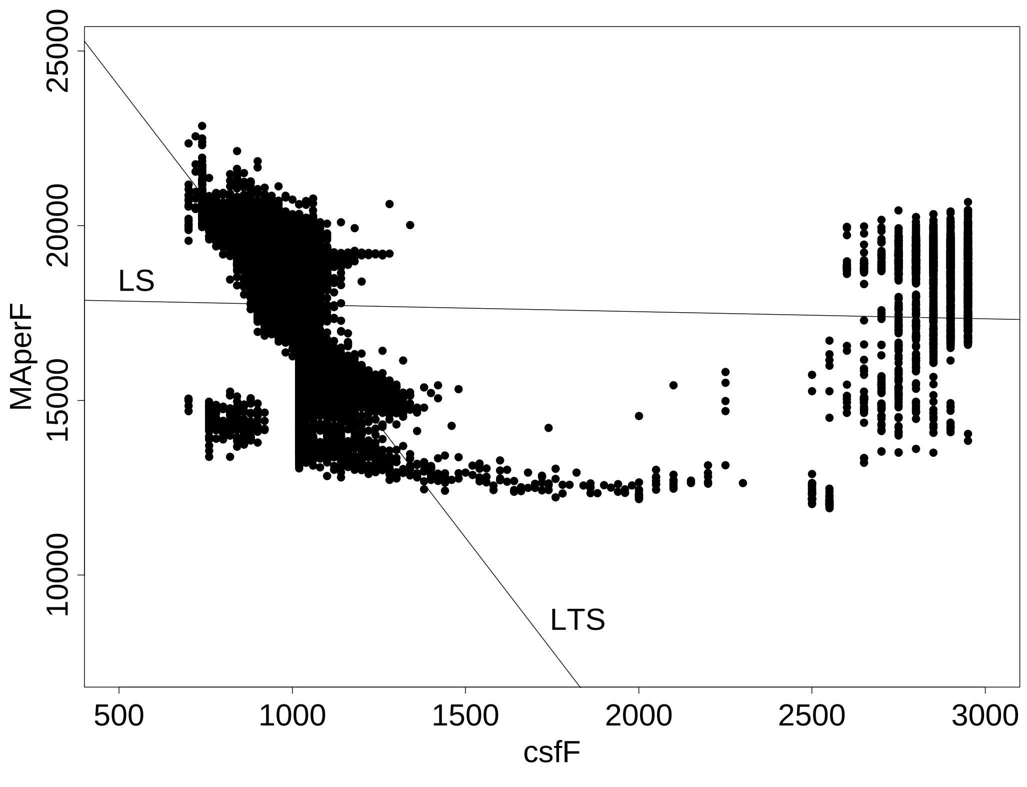


Figure 1. DPOSS data: Plot of aperture magnitude versus the quantity csf which is based on spectroscopy, for 56,744 stars.

Example 1 considered only two variables, but of course we often have data with several explanatory variables. We will show such an example with 8-dimensional $\boldsymbol{x}_i$ in Section 6.

The major disadvantage of existing LTS algorithms is their large computation time when the size of the data set increases. Because the LTS has good statistical properties we would like to develop a faster algorithm for it, which can deal with large data sets. This new algorithm was already used for the n =56,744 stars in Figure 1, which took 3 minutes.

2 Basic idea and the C-step

One of the keys of the new algorithm is the fact that starting from any approximation to the LTS regression coefficients, it is possible to compute another approximation yielding an even lower objective function.

Property. Consider a data set $(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)$ consisting of p-variate $\boldsymbol{x_i}$ (typically with $x_{ip} = 1$) and a response variable y_i. Let $H_1 \subset \{1, \ldots, n\}$ with $|H_1| = h$, and put $Q_1 := \sum\limits_{i \in H_1} (r_1(i))^2$ where $r_1(i) = y_i - (\hat{\theta}_1^1 x_{i1} + \hat{\theta}_2^1 x_{i2} + \cdots + \hat{\theta}_p^1 x_{ip})$ for all $i = 1, \ldots, n$ where $\hat{\boldsymbol{\theta}}_1 = (\hat{\theta}_{11}, \ldots, \hat{\theta}_{p1})$ is any p-dimensional vector. Now take H_2 such that $\{|r_1(i)|; i \in H_2\} := \{|r_1|_{1:n}, \ldots, |r_1|_{h:n}\}$ where $|r_1|_{1:n} \leq |r_1|_{2:n} \leq \cdots \leq |r_1|_{n:n}$ are the ordered absolute values of the residuals, and compute the least squares (LS) fit $\hat{\boldsymbol{\theta}}_2$ of the h observations in H_2. This yields $r_2(i)$ for all $i = 1, \ldots, n$ and $Q_2 := \Sigma_{i \in H_2}((r_2(i))^2$. Then

$$Q_2 \leq Q_1.$$

Proof. Because H_2 corresponds to the h smallest absolute residuals out of n, we have $\Sigma_{i \in H_2}(r_1(i))^2 \leq \Sigma_{i \in H_1}(r_1(i))^2 = Q_1$. Because the LS estimator $\hat{\boldsymbol{\theta}}_2$ of these h observations is such that it minimizes Q_2 we find $Q_2 = \Sigma_{i \in H_2}(r_2(i))^2 \leq \Sigma_{i \in H_2}(r_1(i))^2 \leq Q_1$. $\square$

Applying the above property to H_1 yields H_2 with $Q_2 \leq Q_1$. In our algorithm we will call this a *C-step*, where C stands for 'concentration' since H_2 is more concentrated (has a lower sum of squared residuals) than H_1. In algorithmic terms, the C-step can be described as follows.

Given the h-subset H_{old} then:

- compute $\hat{\boldsymbol{\theta}}_{\text{old}} :=$ LS regression estimator based on H_{old}
- compute the residuals $r_{\text{old}}(i)$ for $i = 1, \ldots, n$

- sort the absolute values of these residuals, which yields a permutation π for which $|r_{\text{old}}(\pi(1))| \leq |r_{\text{old}}(\pi(2))| \leq \cdots \leq |r_{\text{old}}(\pi(n))|$
- put $H_{\text{new}} := \{\pi(1), \pi(2), \ldots, \pi(h)\}$
- compute $\hat{\boldsymbol{\theta}}_{\text{new}} :=$ LS regression estimator based on H_{new}.

Alternatively, any vector $\hat{\boldsymbol{\theta}}_{\text{old}}$ may be given, in which case we do not need any H_{old} and we can skip the first bullet. For a fixed number of dimensions p, the C-step takes only $\mathrm{O}(n)$ time because H_{new} can be determined in $\mathrm{O}(n)$ operations without fully sorting the n absolute residuals $|r_{\text{old}}(i)|$.

Repeating C-steps yields an iteration process. If $Q_2 = Q_1$ we stop; otherwise we apply another C-step yielding Q_3, and so on. The sequence $Q_1 \geq Q_2 \geq Q_3 \geq \ldots$ is nonnegative and hence must converge. In fact, since there are only finitely many h-subsets there must be an index m such that $Q_m = Q_{m-1}$, hence convergence is always reached after a finite number of steps. (In practice, m is often below 10.) This is not sufficient for Q_m to be the global minimum of the LTS objective function, but it is a necessary condition.

This provides a partial idea for an algorithm:

Take many initial choices of H_1 and apply C-steps to each until convergence, and keep the solution with lowest value of (1.1). (2.1)

Of course, several things must be decided to make (2.1) operational: how to generate sets H_1 to begin with, how many H_1 are needed, how to avoid duplication of work since several H_1 may yield the same solution, can't we do with fewer C-steps, what about large sample sizes, and so on. These matters will be discussed in the next sections.

3 Construction of the FAST-LTS algorithm

3.1 Creating initial subsets $\boldsymbol{H_1}$

In order to apply the algorithmic idea (2.1) we first have to decide how to construct the initial subsets H_1. Let us consider the following two possibilities:

(a) Draw a random h-subset H_1.

(b) Draw a random p-subset J, and compute $\hat{\boldsymbol{\theta}}_0 :=$ the coefficients of the hyperplane through J. If J does not define a unique hyperplane (i.e., when the rank of X_J is less than p), extend J by adding random observations until it does. Then compute the residuals $r_0(i) := y_i - \hat{\boldsymbol{\theta}}_0 \boldsymbol{x}_i^t$ for $i = 1, \ldots, n$. Sort them into $|r_0(\pi(1))| \leq \cdots \leq |r_0(\pi(n))|$ and put $H_1 := \{\pi(1), \ldots, \pi(h)\}$.

Option (a) is the simplest, whereas (b) starts like the PROGRESS algorithm (Rousseeuw and Leroy 1987).

When the data set does not contain outliers or deviating groups of points, it makes little difference whether (2.1) is applied with (a) or (b). But to be realistic, let us consider contaminated data sets.

Example 2. To illustrate the difference between the options (a) and (b), we generated a regression data set with $n = 1000$ observations of which 20% are bad leverage points. The first 800 observations (x_i, y_i) were generated by the formula

$$y_i = 1\,x_i + 1 + \epsilon_i \qquad i = 1, \ldots, 800$$

where $x_i \sim \mathrm{N}(0, 100)$ and $\epsilon_i \sim \mathrm{N}(0, 1)$. The other 200 observations (x_i, y_i) were drawn from the bivariate normal distribution with $\boldsymbol{\mu} = (50, 0)$ and $\boldsymbol{\Sigma} = 25\boldsymbol{I}_2$. The entire data set is shown in Figure 2. We wish to compute the LTS with $h = [(n + p + 1)/2] = 501$. We now apply (2.1) with 500 starting sets H_1. Figure 2 shows the results of the algorithm starting with initial subsets of size $h = 501$ (option (a)), which has broken down due to the leverage points. On the other hand, the algorithm starting with p-subsets (option (b)) yields the robust line, which fits the majority of the data.

Similar results are also obtained for a smaller percentage of leverage points, and for more dimensions. Approach (a) fails because each random subset H_1 contains a sizeable number of points from the majority group as well as from the minority group, which follows from the law of large numbers. When starting from a bad subset H_1 the iterations will not converge to the good solution. On the other hand, the probability of a p-subset without outliers is much higher, which explains why (b) yields many good initial subsets H_1 and hence a robust result. In fact, for increasing n the probability of having at least one 'clean' p-subset among m random p-subsets tends to

$$1 - (1 - (1 - \epsilon)^p)^m > 0 \tag{3.1}$$

where ϵ is the percentage of outliers. In contrast, the probability of having at least one clean h-subset among m random h-subsets tends to zero. Therefore, from now on we will always use (b).

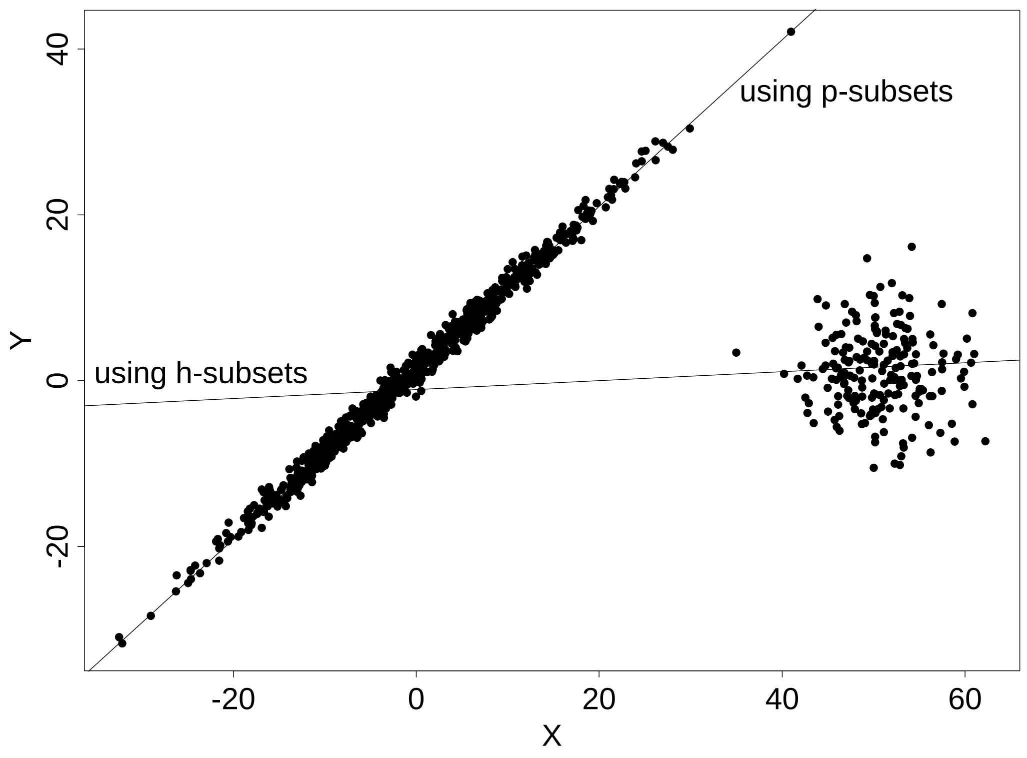


Figure 2. Results of iterating C-steps starting from 500 random h-subsets H_1 of size h=501 (out of $n = 1000$), and starting from 500 random p-subsets of size p=2.

3.2 Selective iteration

Each C-step calculates LS regression coefficients based on h observations, and the corresponding residuals for all n observations. Therefore, reducing the number of C-steps would improve the speed. But is this possible without losing the effectiveness of the algorithm? The following example shows that often the distinction between good (robust) solutions and bad solutions already becomes visible after two or three C-steps.

Example 3. We return to the data set of Example 2. Figure 3 traces the inner workings of the algorithm (2.1). For each starting subsample H_1 it plots the objective value Q_j (i.e. the sum of the squared residuals of the h-subset H_j) versus the step number j. The runs yielding a robust solution are shown as solid lines, whereas the dashed lines correspond to non-robust results. To get a clear picture, Figure 3 only shows 100 starts. After two C-steps (i.e. for $j = 3$), many (in this case all) subsamples H_3 that will lead to the global optimum already correspond to a rather small sum of squared residuals. This global optimum is a

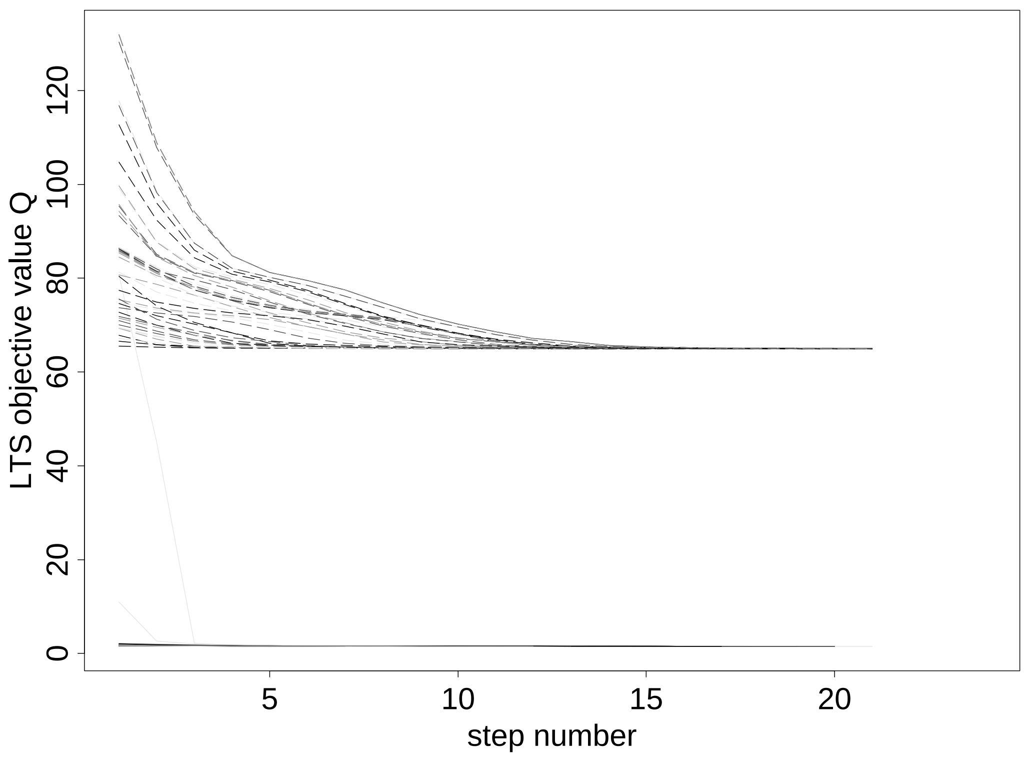


Figure 3. Objective value of subsequent C-steps, for the data in Figure 2.

solution which contains none of the 200 bad points. By contrast, the objective values of the subsets H_3 leading to the wrong fit are considerably larger. For that reason, we can save much computation time and still obtain the same result by taking just two C-steps and retaining only the (say, 10) best subsets H_3 to iterate further.

Other data sets, also in more dimensions and with vertical outliers as well, confirm these conclusions. Therefore, from now on we will take only two C-steps from each initial subsample H_1, select the 10 different subsets H_3 with the lowest sums of squared residuals, and only for these 10 we continue taking C-steps until convergence.

3.3 Nested extensions

For a small to moderate sample size n the above algorithm does not take much time. But when n grows the computation time increases, due to the LS regressions on h-subsets and the n residuals that need to be calculated each time. To avoid doing all the computations in the entire data set, we will consider a special structure. When $n > 1500$, the algorithm generates a nested system of subsets which looks like Figure 4, where the arrows mean ‘is a subset of’. The five subsets of size 300 do not overlap, and together they form the merged set of size

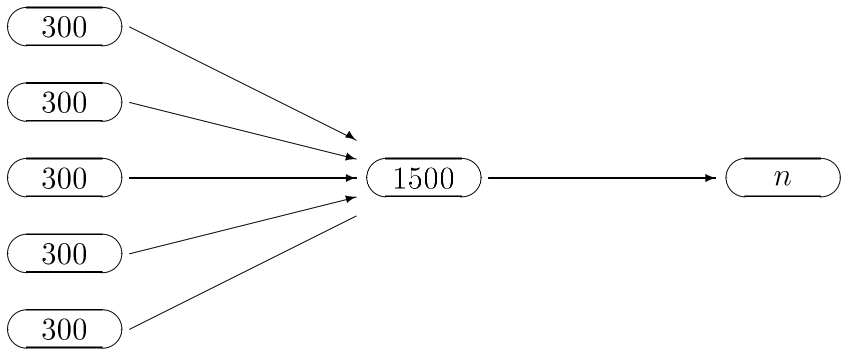


Figure 4. The nested system of subsets constructed by the FAST-LTS algorithm.

1500, which in turn is a proper subset of the data set of size n. (Certain other algorithms, e.g. those of Rousseeuw (1993) and Woodruff and Rocke (1994), partition the entire data set which yields more and/or larger subsets.) To construct Figure 4 the algorithm draws 1500 observations, one by one, without replacement. The first 300 observations it encounters are put in the first subset, and so on. Because of this mechanism each subset of size 300 is roughly representative for the data set, and the merged set with 1500 cases is even more representative.

When $n \leq 600$ we will keep the algorithm as in the previous section, while for $n \geq 1500$ we will use Figure 4. When $600 < n < 1500$ we will partition the data into at most 4 subsets of 300 or more observations, so that each observation belongs to a subset and such that the subsets have roughly the same size. For instance, 601 will be split as 300+301 and 900 as 450+450. For $n = 901$ we use 300+300+301, and we continue until 1499=375+375+375+374. By splitting 601 as 300+301 we don’t mean that the first subset contains the observations with case numbers $1, \ldots, 300$ but that its 300 case numbers were drawn randomly from $1, \ldots, 601$.

Whenever $n > 600$, the FAST-LTS algorithm takes two C-steps from several starting subsamples H_1 within each subset, with a total of 500 starts for all subsets together. For every subset the best 10 solutions are stored. Then the subsets are pooled, yielding a merged set with at most 1500 observations. Each of these (at most 50) available solutions $\hat{\boldsymbol{\theta}}_{\text{sub}}$ is then extended to the merged set. That is, starting from each $\hat{\boldsymbol{\theta}}_{\text{sub}}$ we continue taking C-steps which now use all 1500 observations in the merged set. Only the best 10 solutions $\hat{\boldsymbol{\theta}}_{\text{merged}}$ will be considered further. Finally, each of these 10 solutions is extended to the full data set in the same way, and the best solution $\hat{\boldsymbol{\theta}}_{\text{full}}$ is reported.

Since the final computations are carried out in the entire data set, they take more time when n increases. In the interest of speed we can limit the number of starting points $\hat{\boldsymbol{\theta}}_{\text{merged}}$ and/or the number of C-steps in the full data set as n becomes very large.

The main idea of this subsection was to carry out C-steps in a number of nested random subsets, starting with small subsets of around 300 observations and ending with the entire data set of n observations. Throughout this subsection we have chosen several numbers such as 5 subsets of 300 observations, 500 starts, 10 best solutions, and so on. These choices were based on various empirical trials (not reported here). We implemented our choices as defaults so the user doesn't have to choose anything, but of course the user may change the defaults.

3.4 Intercept adjustment

One other feature of the new algorithm that has not yet been mentioned is the intercept adjustment. This is a technique which decreases the LTS objective value of a any fit. Each initial p-subset and each C-step yields an estimate $\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_1, \ldots, \tilde{\theta}_{p-1}, \tilde{\theta}_p)$ where $\tilde{\theta}_1, \ldots, \tilde{\theta}_{p-1}$ are the slopes and $\tilde{\theta}_p$ is the intercept. The corresponding LTS objective value is given by

$$\Sigma_{i=1}^{h}(r^2(\tilde{\boldsymbol{\theta}}))_{i:n} = \Sigma_{i=1}^{h}((y_i - x_{i1}\tilde{\theta}_1 - \cdots - x_{i,p-1}\tilde{\theta}_{p-1} - \tilde{\theta}_p)^2)_{i:n} \tag{3.2}$$

The adjusted intercept $\tilde{\theta}'_p$ is then calculated as the exact, univariate LTS location estimate of $\{t_i = y_i - x_{i1}\tilde{\theta}_1 - \cdots - x_{i,p-1}\tilde{\theta}_{p-1}; i = 1, \ldots, n\}$, i.e.

$$\tilde{\theta}'_p = \underset{\mu}{\operatorname{argmin}}\, \Sigma_{i=1}^{h}((t_i - \mu)^2)_{i:n}$$

which, by construction, yields the same or a lower objective value (3.2).

For a given univariate data set, the LTS estimator is the mean of the h-subset with the smallest variance. In other words, any contiguous h-subset $t_{j:n}, \ldots, t_{j+h-1:n}$ has a mean $\bar{t}^{(j)}$ and a sum of squares

$$SQ^{(j)} = \Sigma_{i=j}^{j+h-1}(t_{i:n} - \bar{t}^{(j)})^2 \tag{3.3}$$

and we want to minimize (3.3). The complete univariate algorithm can be found in (Rousseeuw and Leroy 1987, pages 171-172).

4 Pseudocode of FAST-LTS

Combining all the components of the preceding sections yields the new algorithm, which we will call FAST-LTS. Its pseudocode looks as follows.

```
1. The default $h$ is $[(n+p+1)/2]$, but the user may choose any integer $h$ with $[(n+p+1)/2] \leq h \leq n$. The program then reports the LTS's breakdown value $(n-h+1)/n$. If you are sure that the data contains less than 25% of contamination, a good compromise between breakdown value and statistical efficiency is obtained by putting $h = [0.75n]$.

2. If $p = 1$ (univariate data) then compute the LTS estimate $\hat{\boldsymbol{\theta}}$ by the exact algorithm of Rousseeuw and Leroy (1987, pages 171–172) and stop.

3. From here on we assume $p \geq 2$. If $n$ is small (say, $n \leq 600$) then
    • repeat (say) 500 times:
        ∗ construct an initial $h$-subset $H_1$ using method (b) in Subsection 3.1, i.e. starting from a random $p$-subset
        ∗ carry out two C-steps (described in Section 2)
    • for the 10 results with lowest $Q_3$:
        ∗ carry out C-steps until convergence
    • report the solution $\hat{\boldsymbol{\theta}}$ with lowest corresponding $Q$

4. If $n$ is larger (say, $n > 600$) then
    • construct up to five disjoint random subsets of size $n_{\text{sub}}$ according to Section 3.3 (say, 5 subsets of size $n_{\text{sub}} = 300$)
    • inside each subset, repeat $500/5 = 100$ times:
        ∗ construct an initial subset $H_1$ of size $h_{\text{sub}} = [n_{\text{sub}}(h/n)]$
        ∗ carry out two C-steps, using $n_{\text{sub}}$ and $h_{\text{sub}}$
        ∗ keep the 10 best results $\hat{\boldsymbol{\theta}}_{\text{sub}}$
    • pool the subsets, yielding the merged set (say, of size $n_{\text{merged}} = 1500$)
    • in the merged set, repeat for each of the 50 solutions $\hat{\boldsymbol{\theta}}_{\text{sub}}$:
```

  - $*$ carry out two C-steps, using n_{merged} and $h_{\text{merged}} = [n_{\text{merged}}(h/n)]$
  - $*$ keep the 10 best results $\hat{\boldsymbol{\theta}}_{\text{merged}}$

- in the full data set, repeat for the m_{full} best results:
  - $*$ take several C-steps, using n and h
  - $*$ keep the best final result $\hat{\boldsymbol{\theta}}_{\text{full}}$

  Here, m_{full} and the number of C-steps (preferably, until convergence) depend on how large the data set is.

Note that the FAST-LTS algorithm is affine equivariant: when the data are translated or subjected to a linear transformation, the resulting $\hat{\boldsymbol{\theta}}_{\text{full}}$ will transform accordingly.

The program FAST-LTS has been thoroughly tested and can be obtained free of charge from the authors.

5 Performance of FAST-LTS

To get an idea of the performance of the overall algorithm, we start by applying FAST-LTS to some small regression data sets taken from (Rousseeuw and Leroy 1987). The first column of Table 1 lists the name of each data set, followed by n and p, where n is the number of observations and p stands for the number of coefficients including the intercept term. We stayed with the default value of $h = [(n+p+1)/2]$. The next column shows the number of starting p-subsets used in FAST-LTS, which is usually 500 except for two data sets where the number of possible p-subsets out of n was fairly small, namely $\binom{12}{3} = 220$ and $\binom{18}{3} = 816$, so we used all of them.

The next entry in Table 1 is the result of FAST-LTS, given here as the final h-subset. By comparing these with the exact LTS algorithm of Agulló (personal communication) it turns out that these h-subsets do yield the exact global minimum of the objective function. The next column shows the running time of FAST-LTS, in seconds on a Sun SparcStation 20/514. We may conclude that for these small data sets FAST-LTS gives very accurate results in little time.

Let us now try the algorithm on larger data sets. Because outliers in x-space are very often the cause for a regression fit to break down, we will generate data with many bad

Table 1. Performance of the FAST-LTS and FSA algorithms on some small data sets.

					time (seconds)	
data set	n	p	starts	best h-subset found	FAST-LTS	FSA
Heart	12	3	220	1 2 4 5 6 7 11 12	1.3	0.7
Phosphor	18	3	816	1 2 3 4 6 7 11 12 14 15 18	3.5	5.7
Coleman	20	6	500	2 5 6 7 8 9 11 13 14 15 16 19 20	4.5	7.8
Wood	20	6	500	2 3 9 10 11 12 13 14 15 16 17 18 20	4.3	8.3
Salinity	28	4	500	2 3 4 6 7 12 14 15 17 18 19 20 21 22 26 27	4.2	13.6
Aircraft	23	5	500	1 5 6 7 8 9 10 11 13 14 15 17 20 23	4.1	9.2
Delivery	25	3	500	2 5 6 7 8 10 12 13 14 15 17 21 22 25	3.1	8.1

leverage points to illustrate the performance of the FAST-LTS algorithm. Each data set is generated according to

$$y_i = x_{i,1} + x_{i,2} + \cdots + x_{i,p-1} + x_{i,p} + e_i$$

in which $e_i \sim \mathrm{N}(0,1)$ is the error term, $x_{i,j} \sim \mathrm{N}(0,100)$ for $j = 1,\ldots,p-1$ are the nontrivial explanatory variables, and $x_{i,p} = 1$ is the intercept term. We have introduced outliers in the x-direction by replacing a percentage of the $x_{i,1}$ by values that are normally distributed with mean 100 and variance 100. For each data set, Table 2 lists n and p followed by the percentage of outliers. The algorithm always used 500 starts.

The results of FAST-LTS are given in the next column, under 'robust'. Here 'yes' means that the correct result is obtained, i.e. corresponding to the first distribution (similar to the correct line in Figure 2), whereas 'no' stands for the nonrobust result, where the estimates describe the entire data set (similar to the incorrect line in Figure 2). For all data sets FAST-LTS yielded the robust result. The computation times were quite low for the given values of n and p. Even for a sample size as high as 50,000 a few minutes suffice, whereas no previous algorithm we know of could handle such large data sets.

The currently most well-known algorithm for approximating the LTS estimator is the Feasible Subset Algorithm (FSA) of Hawkins (1994). Instead of C-steps it uses a differ-

Table 2. Performance of the FAST-LTS and FSA algorithms on larger data sets, with time in seconds.

				FAST-LTS		FSA	
n	p	outliers	starts	robust	time	robust	time
100	2	40%	500	yes	10	no	260
	3	40%	500	yes	12	no	320
	5	40%	500	yes	35	no	440
500	2	40%	500	yes	115	no	29300
	3	40%	500	yes	125	no	34580
	5	40%	500	yes	140	no	48900
1,000	2	35%	500	yes	70	–	–
	5	35%	500	yes	95	–	–
	10	35%	500	yes	160	–	–
10,000	2	40%	500	yes	95	–	–
	5	40%	500	yes	125	–	–
	10	40%	500	yes	200	–	–
50,000	2	40%	500	yes	185	–	–
	5	40%	500	yes	250	–	–

ent kind of steps, which for convenience we will baptise 'I-steps' where the 'I' stands for 'interchanging points'. An I-step proceeds as follows:

Given the h-subset H_{old} with its least squares regression coefficients $\boldsymbol{\theta}_{\text{old}}$:

- repeat for each $i \in H_{\text{old}}$ and each $j \notin H_{\text{old}}$:
  - $*$ put $H_{i,j} = (H_{\text{old}} \setminus \{i\}) \cup \{j\}$
    (i.e., remove point i and add point j)
  - $*$ compute $\triangle_{i,j} = Q_{\text{old}} - Q_{i,j}$
- keep the i' and j' with largest $\triangle_{i',j'}$
- if $\triangle_{i',j'} \leq 0$ put $H_{\text{new}} = H_{\text{old}}$ and stop;
- if $\triangle_{i',j'} > 0$ put $H_{\text{new}} = H_{i',j'}$.

An I-step takes $O(h(n-h)) = O(n^2)$ time because all pairs (i, j) are considered. If we would compute each $Q_{i,j}$ from scratch the complexity would even become $O(n^3)$, but Hawkins (1994, page 189) uses an update formula for $\triangle_{i,j}$.

The I-step can be iterated: if $Q_{\text{new}} < Q_{\text{old}}$ we can take another I-step with H_{new}, otherwise we stop. The resulting sequence $Q_1 \geq Q_2 \geq \ldots$ must converge after a finite number of steps, i.e. $Q_m = 0$ or $Q_m = Q_{m-1}$, so Q_m can no longer be reduced by an I-step. This is again a necessary (but not sufficient) condition for $\boldsymbol{\theta}_m$ to be the global minimum of the LTS objective function. In our terminology, Hawkins' FSA algorithm can be written as:

- repeat many times:
    - ∗ draw an initial h-subset H_1 at random
    - ∗ carry out I-steps until convergence, yielding H_m
- keep the H_m with lowest Q_m
- report this set H_m as well as $\boldsymbol{\theta}_m$.

In Tables 1 and 2 we have applied the FSA algorithm to the same data sets as FAST-LTS, using the same number of starts. For the small data sets in Table 1 the FSA and FAST-LTS yielded identical results, but for the larger data sets in Table 2 the FSA obtains nonrobust solutions. This is because

(1) The FSA starts from randomly drawn h-subsets H_1. Hence for sufficiently large n all the FSA starts are nonrobust, and subsequent iterations do not get away from the corresponding local minimum.

We saw the same effect in Section 3.1, which also explained why it is safer to start from random p-subsets as in PROGRESS and in FAST-LTS.

The tables also indicate that the FSA needs more time than FAST-LTS. In fact, the ratio time(FSA)/time(FAST-LTS) increases from 1 to 3 for n going from 12 to 28. In Table 2 the timing ratio goes from 250 (for $n = 100$) to 350 (for $n = 500$), after which the FSA algorithm took too long to time it. The FSA algorithm is more time-consuming than FAST-LTS because:

(2) An I-step takes $O(n^2)$ time, compared to $O(n)$ for the C-step of FAST-LTS.

(3) Each I-step swaps only 1 point of H_{old} with 1 point outside H_{old}. In contrast, each C-step swaps $h - |H_{\text{old}} \cap H_{\text{new}}|$ points inside H_{old} with the same number outside of H_{old}. Therefore more I-steps are needed, especially for increasing n.

(4) The FSA iterates I-steps until convergence, starting from each H_1. On the other hand, FAST-LTS reduces the number of C-steps by the selective iteration technique of Section 3.2. The latter would not work for I-steps because of (3).

(5) The FSA carries out all its I-steps in the full data set of size n, even for large n. In the same situation FAST-LTS applies the nested extensions method of Section 3.3, so most C-steps are carried out for $n_{\text{sub}} = 300$, some for $n_{\text{merged}} = 1500$, and only a few for the actual n.

In order to do simulations, we linked our FAST-LTS algorithm to S-Plus. The official version of S-Plus computes the LTS by means of a genetic algorithm (Burns 1992). We compared the results of our new algorithm with those obtained by this genetic algorithm with default settings. To our surprise the latter algorithm often broke down. The execution times are not reported here, because the number of iterations of both algorithms could not be compared meaningfully. Generally, when both algorithms result in a robust solution, the objective function for the FAST-LTS algorithm is smaller. Other examples than those listed in Table 2 also indicated that the genetic algorithm easily breaks down for high n and p. Therefore, we personally prefer the FAST-LTS algorithm because it is both robust and fast, even for large data sets.

Remark. Independently of our work, D. Hawkins has recently worked on speeding up his FSA algorithm by including a pretesting step (Hawkins 1997). This does not change the order of computational complexity of the FSA algorithm, but it improves the coefficient of proportionality.

6 Examples

In this section we will illustrate the diagnostic plot introduced by Rousseeuw and Van Zomeren (1990). This descriptive tool plots robust residuals $r_i/\hat{\sigma}$ versus robust distances $\text{RD}(\boldsymbol{x_i})$. The latter will be based on the minimum covariance determinant (MCD) method of (Rousseeuw 1984, 1985) which provides a robust estimate of multivariate location and a robust scatter matrix. We can compute the MCD with the algorithm of (Rousseeuw and Van Driessen 1998), which we apply to the set of $\boldsymbol{x_i}$ (without using the y_i).

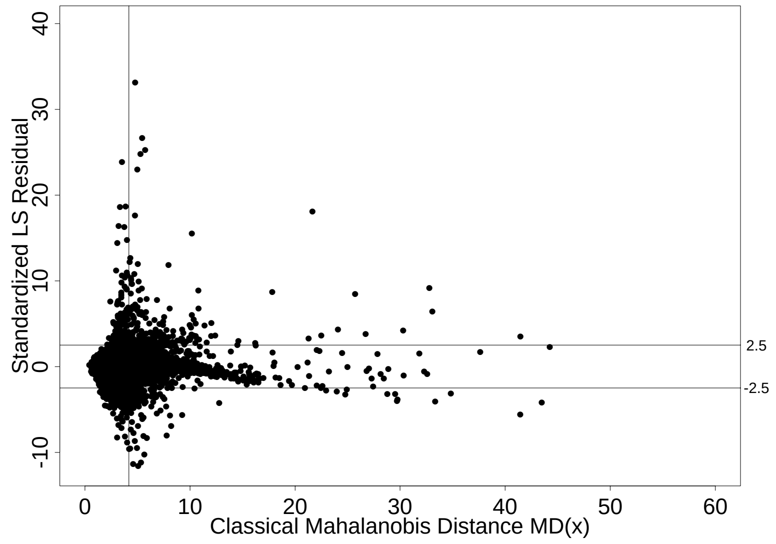
Example 4. Let us return to the DPOSS data of Example 1 and apply the FAST-LTS algorithm to these 56,744 stars. The same explanatory variable MAperF is now regressed against the other 8 characteristics of the colour band F. These characteristics describe the size of a light source and the shape of the spatial brightness distribution in a source. Figure 5a plots the standardized LS residuals versus the classical Mahalanobis distances. Some isolated outliers in the y-direction as well as in x-space fall outside this plot. (We zoomed in to have a better view of the large majority of the data.) According to Figure 5a, most data follow the same linear trend.

However, Figure 5b tells a very different story. It plots the LTS residuals obtained by FAST-LTS versus the robust distances $\mathrm{RD}(\boldsymbol{x_i})$ based on the MCD. This diagnostic plot shows us that there is a rather large group of bad leverage points, i.e. observations with both large robust residuals and large robust distances. These were invisible in Figure 5a because they attract the LS fit as well as the classical Mahalanobis distances. It turns out that this deviating substructure consists of giant stars.

7 Conclusions

The algorithm FAST-LTS proposed here is based on several time-saving techniques: the C-step (Section 2), the procedure for generating initial estimates (Section 3.1), selective iteration (Section 3.2), and nested extensions (Section 3.3). By exploiting the specific structure of the LTS regression problem, the new algorithm is faster and more robust than general-purpose techniques such as reducing an objective function by successively interchanging points, or using an off-the-shelf genetic algorithm. Simulations have shown that FAST-LTS is able to deal with large data sets, while outrunning existing algorithms for LMS and LTS by orders of magnitude. Due to the FAST-LTS algorithm, the LTS estimator becomes available as a tool for analyzing large data sets and to screen them for outliers or deviating substructures, which is one of the main goals of data mining.

(a)



(b)

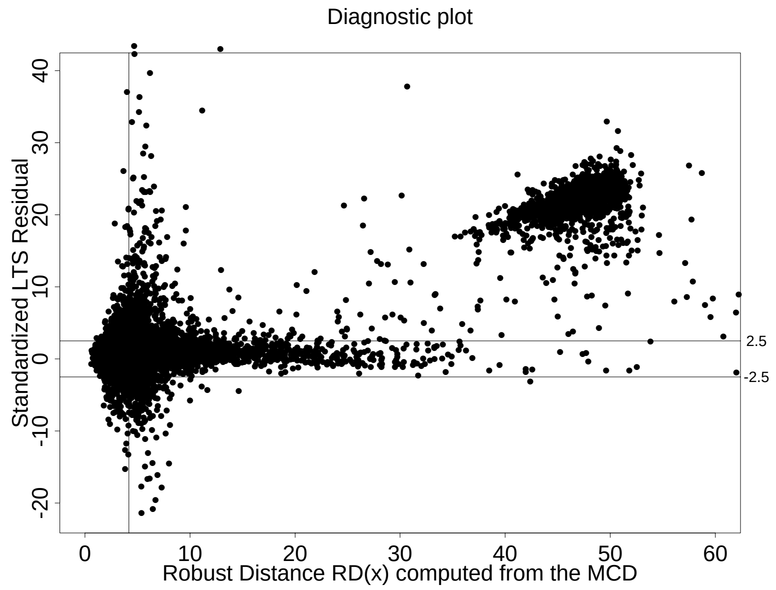


Figure 5. DPOSS data: regression of MAperF on 8-dimensional $\boldsymbol{x_i}$. (a) Plot of LS residual versus Mahalanobis distance MD($\boldsymbol{x_i}$); (b) diagnostic plot of LTS residual versus robust distance RD($\boldsymbol{x_i}$).

References

Agulló, J. (1997a), "Computación de estimadores con alto punto de ruptura," Ph.D. Thesis, University of Alicante, Spain.

Agulló, J. (1997b), "Exact Algorithms to Compute the Least Median of Squares Estimate in Multiple Linear Regression," in L_1-*Statistical Procedures and Related Topics*, ed. Y. Dodge. The IMS Lecture Notes-Monograph Series, Volume 31, 133–146.

Burns, P.J. (1992), "A Genetic Algorithm for Robust Regression Estimation," Statsci Technical Note. Statistical Sciences, Seattle.

Coakley, C.W. and Hettmansperger, T.P. (1993), "A Bounded Influence, High Breakdown, Efficient Regression Estimator," *Journal of the American Statistical Association*, 88, 872–880.

Hawkins, D.M. (1994), "The Feasible Solution Algorithm for Least Trimmed Squares Regression," *Computational Statistics and Data Analysis*, 17, 185–196.

Hawkins, D.M. (1997), "Improved Feasible Solution Algorithms for High Breakdown Estimation," Technical Report, University of Minnesota, September 1997.

Hössjer, O. (1994), "Rand-Based Estimates in the Linear Model with High Breakdown Point," *Journal of the American Statistical Association*, 89, 149–158.

Meer, P., Mintz, D., Rosenfeld, A. and Kim, D. (1991), "Robust Regression Methods in Computer Vision: A Review," *International Journal of Computer Vision*, 6, 59–70.

Odewahn, S.C., Djorgovski, S.G., Brunner, R.J., and Gal, R. (1998), "Data From the Digitized Palomar Sky Survey," Technical Report, California Institute of Technology.

Rousseeuw, P.J. (1984), "Least Median of Squares Regression," *Journal of the American Statistical Association*, 79, 871–880.

Rousseeuw, P.J. (1985), "Multivariate Estimation with High Breakdown Point," in *Mathematical Statistics and Applications, Vol B*, eds. W. Grossmann, G. Pflug, I. Vincze, and W. Wertz. Dordrecht: Reidel, 283–297.

Rousseeuw, P.J. (1993), "A Resampling Design for Computing High-Breakdown Regression," *Statistics and Probability Letters,* 18, 125–128.

Rousseeuw, P.J. (1997), "Introduction to Positive-Breakdown Methods," in *Handbook of Statistics, Vol. 15: Robust Inference,* eds. G.S. Maddala and C.R. Rao. Amsterdam: Elsevier, 101–121.

Rousseeuw, P.J. and Hubert, M. (1997), “Recent Developments in PROGRESS,” in L_1-*Statistical Procedures and Related Topics*, edited by Y. Dodge. The IMS Lecture Notes–Monograph Series, Vol. 31, 201–214.

Rousseeuw, P.J. and Leroy, A.M. (1987), *Robust Regression and Outlier Detection*, New York: John Wiley.

Rousseeuw, P.J. and Van Driessen, K. (1998), “A Fast Algorithm for the Minimum Covariance Determinant Estimator,” Technical Report, University of Antwerp.

Rousseeuw, P.J. and van Zomeren, B.C. (1990), “Unmasking Multivariate Outliers and Leverage Points,” *Journal of the American Statistical Association*, 85, 633–639.

Steele, J.M. and Steiger, W.L. (1986), “Algorithms and Complexity for Least Median of Squares Regression,” *Discrete Applied Mathematics,* 14, 93–100.

Stromberg, A.J. (1993), “Computing the Exact Least Median of Squares Estimate and Stability Diagnostics in Multiple Linear Regression,” *SIAM Journal of Scientific Computing,* 14, 1289–1299.

Simpson, D.G., Ruppert, D., and Carroll, R.J. (1992), “On One-Step GM-estimates and Stability of Inferences in Linear Regression,” *Journal of the American Statistical Association*, 87, 439–450.

Woodruff, D.L. and Rocke, D.M (1994), “Computable Robust Estimation of Multivariate Location and Shape in High Dimension using Compound Estimators,” *Journal of the American Statistical Association*, 89, 888–896.

Yohai, V.J. (1987), “High Breakdown Point and High Efficiency Robust Estimates for Regression,” *Annals of Statistics*, 15, 642–656.