CHAPTER 5

# Algorithms

## 1. STRUCTURE OF THE ALGORITHM USED IN PROGRESS

The computation of the least median of squares (LMS) regression coefficients is not obvious at all. It is probably impossible to write down a straightforward formula for the LMS estimator. In fact, it appears that this computational complexity is inherent to all (known) affine equivariant high-breakdown regression estimators, because they are closely related to projection pursuit methods (see Section 5 of Chapter 3). The algorithm implemented in PROGRESS (Leroy and Rousseeuw 1984) is similar in spirit to the bootstrap (Diaconis and Efron 1983). Some other possible algorithms will be described in Section 2.

The algorithm in PROGRESS proceeds by repeatedly drawing subsamples of $p$ different observations. (We will use the same notation as in Section 6 of Chapter 3.) For such a subsample, indexed by $J = \{i_1, \ldots, i_p\}$, one can determine the regression surface through the $p$ points and denote the corresponding vector of coefficients by $\boldsymbol{\theta}_J$. (This step amounts to the solution of a system of $p$ linear equations in $p$ unknowns.) We will call such a solution $\boldsymbol{\theta}_J$ a trial estimate. For each $\boldsymbol{\theta}_J$, one also determines the corresponding LMS objective function with respect to the whole data set. This means that the value

$$\underset{i=1,\ldots,n}{\operatorname{med}} (y_i - \mathbf{x}_i \boldsymbol{\theta}_J)^2 \tag{1.1}$$

is calculated. Finally, one will retain the trial estimate for which this value is minimal. (Note that this algorithm is affine equivariant, as it should be.)

But how many subsamples $J_1, J_2, \ldots, J_m$ should we consider? In principle, one could repeat the above procedure for all possible sub-

samples of size $p$, of which there are $C_n^p$. Unfortunately, $C_n^p$ increases very fast with $n$ and $p$. In many applications, this would become infeasible. In such cases, one performs a certain number of random selections, such that the probability that at least one of the $m$ subsamples is "good" is almost 1. A subsample is "good" if it consists of $p$ good observations of the sample, which may contain up to a fraction $\varepsilon$ of bad observations. The expression for this probability, assuming that $n/p$ is large, is

$$1-(1-(1-\varepsilon)^p)^m . \tag{1.2}$$

By requiring that this probability must be near 1 (say, at least 0.95 or 0.99), one can determine $m$ for given values of $p$ and $\varepsilon$. This has been done in Table 1 for $p \leq 10$ and for a percentage of contamination varying between 5% and 50%. For fixed $p$, the number of subsamples increases with $\varepsilon$ in order to maintain the same probability (1.2). This permits us to reduce the computation time of PROGRESS in situations where there are reasons to believe that there are at most 25% of bad data. This choice can be made by answering the question:

```
WHICH VERSION OF THE ALGORITHM WOULD YOU LIKE TO USE?
----------------------------------------
Q=QUICK VERSION
E=EXTENSIVE SEARCH
ENTER YOUR CHOICE PLEASE (Q OR E):
```

**Table 1. Number $m$ of Random Subsamples, Determined in Function of $p$ and $\varepsilon$ by Requiring That the Probability of at Least One Good Subsample Is 95% or More**

| Dimension | Fraction $\varepsilon$ of Contaminated Data | | | | | | |
|---|---|---|---|---|---|---|---|
| $p$ | 5% | 10% | 20% | 25% | 30% | 40% | 50% |
| 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 |
| 2 | 2 | 2 | 3 | 4 | 5 | 7 | 11 |
| 3 | 2 | 3 | 5 | 6 | 8 | 13 | 23 |
| 4 | 2 | 3 | 6 | 8 | 11 | 22 | 47 |
| 5 | 3 | 4 | 8 | 12 | 17 | 38 | 95 |
| 6 | 3 | 4 | 10 | 16 | 24 | 63 | 191 |
| 7 | 3 | 5 | 13 | 21 | 35 | 106 | 382 |
| 8 | 3 | 6 | 17 | 29 | 51 | 177 | 766 |
| 9 | 4 | 7 | 21 | 36 | 73 | 296 | 1533 |
| 10 | 4 | 7 | 27 | 52 | 105 | 494 | 3067 |

The number of subsamples computed from (1.2) becomes tremendous for large $p$, at least in the most extreme case of about 50% of bad observations. At a certain stage, one can no longer increase the number $m$ of replications, because the computation time must remain feasible. One then incurs a larger risk (the probability of which is known) of not encountering any "good" subsample during the $m$ replications. Table 2 lists the values of $m$ that are used in PROGRESS. For small $p$ we have systematically taken $m$ larger than the value needed in formula (1.2), in order to increase the expected number of good subsamples, so the objective function (1.1) may become even smaller.

Several elements of this algorithm can also be found in other statistical techniques. For instance, Oja and Niinimaa (1984) and Hawkins et al. (1985) have used trial estimates such as $\boldsymbol{\theta}_J$ for different purposes. The idea of drawing many (but not necessarily all) subsamples of the data at hand is related to the bootstrap (Efron 1979), which is, however, mainly concerned with the estimation of bias and variability of existing (point) estimators at the actual data. Formula (1.2) was first used by Stahel

**Table 2. Number $m$ of Subsamples That Are Actually Used in the Program PROGRESS, According to the Options "Extensive" or "Quick" [a]**

| | Option "Extensive" | Option "Quick" |
|---|---|---|
| $p = 1$ | $m = C_n^p$ if $n \le 500$<br>$m = 500$ if $n > 500$ | $m = C_n^p$ if $n \le 150$<br>$m = 150$ if $n > 150$ |
| $p = 2$ | $m = C_n^p$ if $n \le 50$<br>$m = 1000$ if $n > 50$ | $m = C_n^p$ if $n \le 25$<br>$m = 300$ if $n > 25$ |
| $p = 3$ | $m = C_n^p$ if $n \le 22$<br>$m = 1500$ if $n > 22$ | $m = C_n^p$ if $n \le 15$<br>$m = 400$ if $n > 15$ |
| $p = 4$ | $m = C_n^p$ if $n \le 17$<br>$m = 2000$ if $n > 17$ | $m = C_n^p$ if $n \le 12$<br>$m = 500$ if $n > 12$ |
| $p = 5$ | $m = C_n^p$ if $n \le 15$<br>$m = 2500$ if $n > 15$ | $m = C_n^p$ if $n \le 11$<br>$m = 600$ if $n > 11$ |
| $p = 6$ | $m = C_n^p$ if $n \le 14$<br>$m = 3000$ if $n > 14$ | $m = 700$ |
| $p = 7$ | $m = 3000$ | $m = 850$ |
| $p = 8$ | $m = 3000$ | $m = 1250$ |
| $p \ge 9$ | $m = 3000$ | $m = 1500$ |

[a] The notation $m = C_n^p$ means that *all* subsamples with $p$ elements are being considered.

(1981) in the context of robust multivariate location and covariance estimators.

Let us now illustrate the basic idea of the algorithm used in PROGRESS by means of the artificial two-dimensional example in Figure 1. For this data set, $n$ equals 9 and $p$ equals 2. Because $n$ is very small, PROGRESS considers *all* pairs of points. We will restrict the explanation to only three such combinations, namely $(f, g)$, $(f, h)$, and $(g, h)$. Let us start with the points $f$ and $g$. The regression surface (which is a line here) passing through $f$ and $g$ is found by solving the system of equations

$$y_f = \theta_1^0 x_f + \theta_2^0$$

$$y_g = \theta_1^0 x_g + \theta_2^0 ,$$

where $(x_f, y_f)$ and $(x_g, y_g)$ are the coordinates of the points $f$ and $g$. The resulting trial estimate is $(\theta_1^0, \theta_2^0)'$. Next, the residuals $y_i - \theta_1^0 x_i - \theta_2^0$ from this line are determined for all points $i$ in the sample. The median of the squared residuals is then calculated, and compared with the smallest value found for previous pairs of points. Because we want to minimize this quantity, the trial estimate corresponding to $f$ and $g$ will be retained only when it leads to a strictly lower value. Examining the scatterplot in Figure 1, we see that $(f, g)$ is better than either $(f, h)$ or $(g, h)$. Indeed, the majority of the observations have small residuals with respect to the

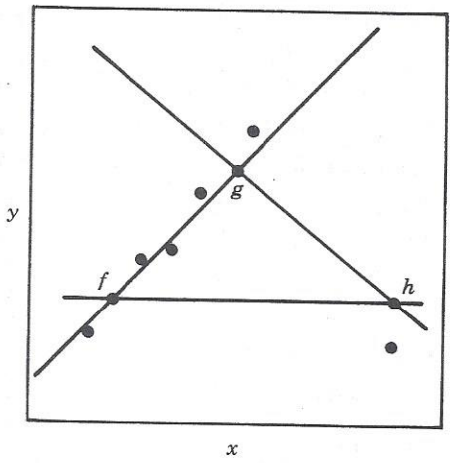


**Figure 1.** Illustration of the algorithm used in PROGRESS.

line passing through $f$ and $g$. Repeating this procedure for each pair of points will finally yield the lowest objective function.

The entry $p = 1$ in Table 2 refers to simple regression through the origin (see Section 6 of Chapter 2) in which there is no intercept term. Note that in one-dimensional location we also have $p = 1$, but then we do not apply the resampling technique because an exact algorithm is available for that situation (see Section 2 of Chapter 4). Moreover, this exact algorithm is very useful in any regression model with intercept, because we can adjust our constant term with it. Indeed, suppose that the resampling algorithm has provided regression coefficients $\hat{\theta}_1, \ldots, \hat{\theta}_p$. Then we can replace $\hat{\theta}_p$ by the LMS location estimate of the $n$ numbers

$$y_i - x_{i,1}\hat{\theta}_1 - \cdots - x_{i,p-1}\hat{\theta}_{p-1}, \qquad i = 1, \ldots, n$$

because we are certain that the resulting objective function (1.1) will become smaller, since the one-dimensional algorithm yields the optimal intercept term (conditional on the values of $\hat{\theta}_1, \ldots, \hat{\theta}_{p-1}$). Therefore, this last step has been incorporated in PROGRESS.

This argument also shows us how we could improve the result even further. Instead of adjusting the intercept term once (at the end), we could do the same thing at each replication, so that each trial estimate would contain an optimal intercept term. It follows that this slight modification of the algorithm could only lower the resulting objective function. However, we have decided not to include this version because it would consume more computation time. At present, we evaluate the median of the squared residuals (for each trial estimate) in $O(n)$ operations, whereas we would spend at least $O(n \log n)$ if we wanted to apply the one-dimensional algorithm each time. On the other hand, the present evolution of computers might soon allow this refinement.

REMARK. Note that the algorithm can be speeded up enormously by means of *parallel computing*, because it is very easy to split it up into parts that are carried out simultaneously. Indeed, each processing element can compute trial estimates $\boldsymbol{\theta}_j$ and the associated median squared residual (1.1). To try this out, a parallelized version of PROGRESS was implemented on the ICAP 1 system at the IBM Research Center in Kingston, NY. (This happened in March 1987 in collaboration with Ph. Hopke and L. Kaufman.) This system contains an IBM 3081 host computer and 10 loosely coupled FPS-164 processors. PROGRESS was adapted by adding special statement lines for the precompiler used on the ICAP 1. Two versions of the program were constructed, one with and one without shared memory. In both cases, an important reduction of the

computation time was achieved. This speed will, of course, still increase when a larger number of processors is used. Already systems with hundreds of processing elements are being developed (such as the DPP87 system in Delft), so eventually *all* trial values may be computed at the same time, making the LMS as fast as least squares.

Apart from the regression coefficients, also the scale parameter $\sigma$ (the dispersion of the errors $e_i$) has to be estimated in a robust way. We begin by computing an initial scale estimate $s^0$. This $s^0$ is based on the minimal median and multiplied by a finite-sample correction factor (which depends on $n$ and on $p$) for the case of normal errors:

$$s^0 = 1.4826(1 + 5/(n-p))\sqrt{\underset{i}{\text{med}}\, r_i^2(\hat{\theta})}\,. \tag{1.3}$$

The factor $1.4826 = 1/\Phi^{-1}(0.75)$ was introduced because $\text{med}_i\,|z_i|/\Phi^{-1}(0.75)$ is a consistent estimator of $\sigma$ when the $z_i$ are distributed like $N(0, \sigma^2)$. From an empirical study, it appeared that this factor 1.4826 alone was not enough, because the scale estimate became too small in regressions with normal errors, especially for small samples. Indeed, if $n$ is only slightly larger than $2p$, then it easily happens that half of the data (which may mean only $p+1$ or $p+2$ points!) almost lie on a linear structure. As a consequence, also some good points may obtain relatively large standardized residuals. It was not obvious at all to find an appropriate factor to compensate for this effect. On the one hand, such a factor should not be too small because good points should not possess large standardized residuals, but on the other hand, a too-large factor would result in neglecting real outliers in contaminated samples. We have studied the behavior of the original scale estimate through simulation, both in normal error situations and in situations where there was contamination in the response or in the explanatory variables. It turned out that multiplication with the factor $1 + 5/(n-p)$ gave a satisfactory solution. (Of course, for large $n$ this factor tends to 1.)

This preliminary scale estimate $s^0$ is then used to determine a weight $w_i$ for the $i$th observation, namely

$$w_i = \begin{cases} 1 & \text{if } |r_i/s^0| \leq 2.5 \\ 0 & \text{otherwise} \end{cases} \tag{1.4}$$

By means of these weights, the final scale estimate $\sigma^*$ for LMS regression is calculated:

$$\sigma^* = \sqrt{\left(\sum_{i=1}^{n} w_i r_i^2\right) \Big/ \left(\sum_{i=1}^{n} w_i - p\right)}\,. \tag{1.5}$$

The advantage of this formula for $\sigma^*$ is that outliers do not influence the scale estimate anymore. Moreover, at the classical model, $\sigma^*$ would be a consistent estimator of $\sigma$ if the weights $w_i$ were independent of the data $(\mathbf{x}_i, y_i)$.

Like most regression programs, PROGRESS performs its actual computations on standardized data. The main motivation for this is to avoid numerical inaccuracies caused by different units of measurement. For instance, one of the explanatory variables might take values around $10^8$, whereas another might be of the order of $10^{-11}$. Therefore, the program first standardizes the data to make the variables dimensionless and of the same order of magnitude. The standardization used in PROGRESS is described in Section 1 of Chapter 4. The regression algorithm is then applied to these standardized data, but afterwards the results have to be transformed back in terms of the original variables. The regression coefficients of LS, LMS, and RLS are transformed in exactly the same way (by means of the subroutine RTRAN) because all three regression methods satisfy the equivariance properties listed in Section 4 of Chapter 3. For LS and RLS, the variance-covariance matrix between the coefficients is also transformed.

The running time of PROGRESS depends not only on the sample size $n$, the dimension $p$, and the required probability in (1.2), but, of course, also on the computer processing speed. To give an illustration, Table 3

**Table 3. Computation Times on an Olivetti M24 Microcomputer Without 8087 Mathematical Coprocessor, for Some Data Sets Used in Chapters 2 and 3**

| Data Set | Computation Time (minutes) | |
|---|---|---|
| | Option "Extensive" | Option "Quick" |
| Pilot-Plant data | 1.37 | 1.37 |
| Hyperinflation in China | 0.90 | 0.90 |
| Fires data | 0.83 | 0.83 |
| Mickey data | 1.32 | 1.32 |
| Hertzsprung–Russell | 3.88 | 1.93 |
| Telephone data | 1.45 | 1.45 |
| Lactic acid concentration | 1.00 | 1.00 |
| Kootenay River | 0.83 | 0.83 |
| Stackloss data | 7.50 | 2.58 |
| Coleman data | 19.13 | 5.23 |
| Salinity data | 9.72 | 3.05 |
| Air quality data | 8.93 | 2.18 |
| Hawkins–Bradu–Kass data | 16.23 | 4.77 |
| Education expenditure | 12.52 | 3.87 |

lists some typical running times on an IBM-PC compatible microcomputer. These times can be reduced substantially by means of an 8087 Mathematical Coprocessor.

## *2. SPECIAL ALGORITHMS FOR SIMPLE REGRESSION

In simple regression, one only has to find the slope $\hat{\theta}_1$ and the intercept $\hat{\theta}_2$ of a line determined by $n$ points in the plane. One can, of course, apply the above resampling algorithm, as it is implemented in PROGRESS. However, in this particular situation some other approaches are also possible.

Our oldest algorithm for the LMS and the LTS was of the "scanning" type (Rousseeuw et al. 1984b). As in Section 1, one starts by standardizing the observations. The idea of the algorithm can be understood most easily when writing the definition of the LMS estimator in the following way:

$$\min_{\theta_1} \{\min_{\theta_2} \operatorname*{med}_i ((y_i - \theta_1 x_i) - \theta_2)^2\} . \tag{2.1}$$

Indeed, one can treat the parts in (2.1) separately. The second portion of the minimization is quite easy, because for any given $\theta_1$ it becomes essentially a one-dimensional problem, which can be solved explicitly as we saw in Chapter 4. (This part even cancels when there is no constant term in the model.) Then one has to find $\hat{\theta}_1$ for which

$$m^2(\theta_1) = \min_{\theta_2} \operatorname*{med}_i ((y_i - \theta_1 x_i) - \theta_2)^2$$

is minimal. This is just the minimization of a one-dimensional function $m^2(\theta_1)$, which is continuous but not everywhere differentiable. In order to find this minimum, one goes through all angles $\alpha$ from $-1.55$ rad to 1.55 rad with a stepsize of 0.02 rad and uses the slope $\theta_1 = \tan \alpha$ to compute the corresponding value of $m^2(\theta_1)$. Then one scans with a precision of 0.001 rad in the two most promising areas (as determined by the local minima of the first step). In this way, the objective function does not have to be calculated too often. In Figure 2, a part of the function $m^2(\theta_1)$ is plotted for the telephone data in Table 3 of Chapter 2. Such a plot may also be useful in keeping track of local minima. Indeed, a prominent secondary minimum indicates a possible ambiguity in that there may be two lines, each fitting the data reasonably well.

The generalization of this scanning-type algorithm to multivariate models is not feasible. In such situations, the scanning would have to be

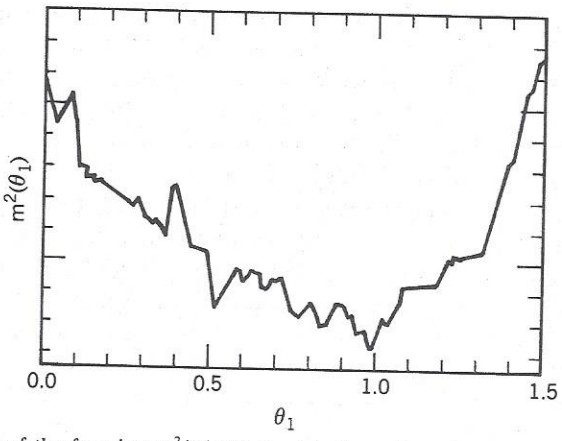


**Figure 2.** A plot of the function $m^2(\theta_1)$ for the telephone data, for $\theta_1$ between 0.0 rad and 1.5 rad.

performed on higher-dimensional grids, which leads to a tremendous amount of computation.

Yohai (1986, personal communication) is presently experimenting with a multivariate algorithm based on repeated application of the estimate $\hat{\theta} = \text{med}_i\, y_i / \text{med}_i\, x_i$ for simple regression through the origin. At each step, the variable is selected which minimizes $\text{med}_i\, r_i^2$, and then its effect is swept from the response variable and the remaining carriers. The objective function $\text{med}_i\, r_i^2$ is also used to decide how many variables are used within a cycle and as a stopping rule for the iteration of cycles. For such an algorithm it is difficult to prove anything in the way of equivariance or overall breakdown point, but some preliminary experience indicates that the computation time is not too high, and the results have been satisfactory.

Recently, Steele and Steiger (1986) studied the numerical aspects of the LMS estimator in the framework of simple regression with intercept. Their first result states that not only are there two points that possess an LMS residual of $\pm m_T$ (as we have proved in Theorem 1 of Chapter 4) but that there are even *three* such observations. This property is one out of three (necessary and sufficient) conditions under which a pair $(\theta_1, \theta_2)$ is a local optimizer of the LMS objective function. The two other conditions say that the three residuals of magnitude $\pm m_T$ alternate in sign, and secondly that the number of points with $|r_i| > m_T$ is at least one more than the number of points with $|r_i| < m_T$. As a consequence, this characterization of local minimizers reduces the continuous minimization

to a discrete one, in the sense that only lines satisfying the above conditions have to be considered and that the one with the smallest objective function value has to be retained. Indeed, it implies that there exists (at least) one pair of data points such that the line joining them has the same slope as the LMS line. This provides a posteriori justification of the way we work in PROGRESS because it means that our algorithm, at least in the simple regression case and for small $n$, leads to an exact solution. (For large $n$, we only use random pairs of points to reduce the computation time.) Furthermore, Steele and Steiger also showed that the LMS objective function can have at most $O(n^2)$ local minima, and they provide some ideas on decreasing the computation time. Finally, they stated that the necessary and sufficient conditions for local minimizers can be generalized to $p$ dimensions ($p > 2$). This corresponds with our own algorithm for multiple regression, where we consider hyperplanes passing through $p$ points, which are again vertically adjusted by estimating the intercept term in a one-dimensional way.

Souvaine and Steele (1986) provided even more refined algorithms for LMS simple regression, making use of modern tools of numerical analysis and computer science such as heaps and hammocks. Their best computational complexity is again $O(n^2)$, and they also stated that everything can be generalized to $p$ dimensions, yielding $O(n^p)$. It seems reasonable to conjecture that one can do no better than $O(n^p)$ for computing LMS or any other affine equivariant high-breakdown regression estimator.

## *3. OTHER HIGH-BREAKDOWN ESTIMATORS

In addition to the LMS estimator, we have also discussed the LTS estimator in Section 2 of Chapter 1 and in Section 4 of Chapter 3. In order to compute the LTS regression coefficients, one only has to change the objective function in either the resampling algorithm of Section 1 or the scanning algorithm described in Section 2. That is, we now have to minimize

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

instead of $\text{med}_i\, r_i^2$. Note, however, that the calculation of the LTS objective requires sorting of the squared residuals, which takes more computation time than the LMS goal function. In a model with a constant term, one also needs the LTS location estimator in order to adjust the intercept. For this purpose, Section 2 of Chapter 4 contains an exact

algorithm, which again consumes more computation time than its LMS counterpart. Therefore, the overall LTS regression algorithm is somewhat more expensive than the LMS one. This effect is shown in Figure 3, where the computation times of LMS and LTS are plotted for different sample sizes. (Note that, according to Table 2, the number of trial estimates $m$ increases until $n = 50$ and then remains equal to 1000.) Also, analogous changes in the LMS algorithm are sufficient for computing the least winsorized squares (LWS) estimator.

The calculation of $S$-estimators (Section 4 of Chapter 3) is somewhat more complicated. The objective function now must be replaced by the solution $s$ of the equation

$$\frac{1}{n}\sum_{i=1}^{n}\rho\left(\frac{r_i}{s}\right) = K\,, \tag{3.1}$$

which may be computed in an iterative way. [A possible choice for $\rho$ is given by (4.31) in Chapter 3.] The residuals $r_i$ in (3.1) correspond to a trial estimate $\boldsymbol{\theta}_j$ in the resampling algorithm or to a certain slope and

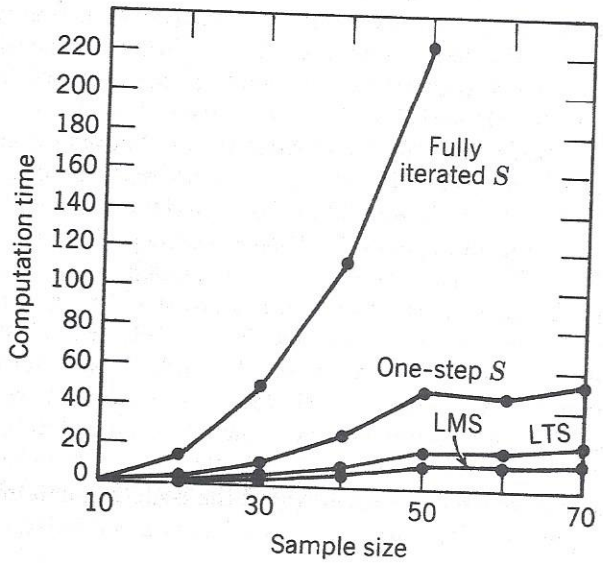


**Figure 3.** Computation times (in CP seconds on a VAX 11/750) of the LMS, the LTS, one-step $S$-, and fully iterated $S$-estimators, for $p = 2$ and sample sizes $n = 10, \ldots, 70$. The resampling algorithm was used, and the running times have been averaged over several simulated data sets. (Data from van Zomeren 1986.)

intercept in the scanning algorithm. It is possible to restrict our attention to the first step of the iteration [see (2.12) in Chapter 4] in order to reduce the computation time. The vector $\hat{\boldsymbol{\theta}}$ minimizing $s$ is called an $S$-estimate of the regression coefficients. The final scale estimate is then given by

$$\hat{\sigma} = s(r_1(\hat{\boldsymbol{\theta}}), \ldots, r_n(\hat{\boldsymbol{\theta}})) .$$

From a computational point of view, Figure 3 shows that even the one-step version of the $S$-estimator is very time-consuming, not to mention the fully iterated version. Moreover, some preliminary experience (both for simple and multiple regression) indicates that $S$-estimators do not really perform better than the LMS, at least from a practical point of view, which is why we have focused our attention to the LMS as the most easily computable member of our high-breakdown family.

## *4. SOME SIMULATION RESULTS

In Chapters 2 and 3 we compared the LS and LMS regressions by means of some real data sets. A shortcoming of this approach is that it may lead to disputes, because it is impossible to "prove" which analysis is best. A popular alternative is to carry out a simulation, because then one knows the true parameter values of the generated data.

In the well-known Princeton robustness study (Andrews et al. 1972), a large number of location estimators have been tried out in a variety of situations. This large-scale simulation has provided many new insights, resulting in a significant impact on robustness theory. We hope that, some day, funds may be raised to carry out a similar project devoted to regression estimators. In the meantime there have been some smaller comparative studies, such as Kühlmeyer's (1983) investigation of 24 regression estimators of the types $M$, $L$, and $R$, and the article by Johnstone and Velleman (1985b). In the present section we report on some of our own simulation results, which focus on the LMS and its variants.

As a preliminary step, we investigated the scale estimate $\hat{\sigma}$ associated with LMS regression. This was necessary because the naive estimate

$$1.4826\sqrt{\underset{i}{\text{med}}\, r_i^2(\hat{\boldsymbol{\theta}})} \tag{4.1}$$

underestimates $\sigma$ in small-sample situations, as was already explained in

Section 1 above. To determine a suitable correction factor, we generated many samples according to the standard linear model. For each choice of $n$ and $p$ we considered 200 samples and compared the resulting values of (4.1) with the true $\sigma$. [It turned out that (4.1) behaved somewhat differently depending on whether $n$ and $p$ were even or odd. A first attempt to understand this phenomenon was restricted to $p = 1$, so only the effect of $n$ had to be studied (Rousseeuw and Leroy 1987). The effect of parity—or of mod $(n, 4)$—persisted even in artificial samples of the type $\{\Phi^{-1}(i/(n+1)), i = 1, \ldots, n\}$ and could be traced back by expanding $\Phi^{-1}$ by means of Taylor series in the endpoints of the shortest halves.] In order to determine a reasonable overall adjustment factor, we have plotted the average values of (4.1) versus $1/(n-p)$, yielding the factor $1 + 5/(n-p)$ in (1.3) above.

After this preliminary step, we set up the actual simulation experiment in order to investigate the LS and the LMS, as well as the one-step reweighted least squares (RLS) based on the LMS, and the one-step $M$-estimator (OSM) based on the LMS. For this purpose, we have resorted to three types of configurations. The first one is the normal situation,

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

in which $e_i \sim N(0, 1)$ and the explanatory variables are generated as $x_{i,j} \sim N(0, 100)$ for $j = 1, \ldots, p$ (if there is no intercept term) or for $j = 1, \ldots, p-1$ (if there is an intercept term, and then $x_{i,p} = 1$).

In the second situation we construct outliers in the $y$-direction. For this purpose, we generate samples where 80% of the cases are as in the first situation and 20% are contaminated by using an error term $e_i \sim N(10, 1)$.

Finally, in the third situation we introduce outliers in the $x$-direction. Eighty percent of the cases are again as in the first situation. In the remaining 20% the $y_i$ are generated as before, but afterwards the $x_{i,1}$ are replaced by values that are now normally distributed with mean 100 and variance 100.

Figure 4 shows an example of each of these three configurations in the framework of simple regression through the origin, with $n = 50$.

The purpose of our simulation is to measure to what extent the estimates differ from the true values $\theta_1 = \cdots = \theta_p = \sigma = 1$. Some summary values over the $m = 200$ runs are computed, such as the *mean estimated value*

$$\bar{\theta}_j = \frac{1}{m} \sum_{k=1}^{m} \hat{\theta}_j^{(k)} , \qquad (4.2)$$

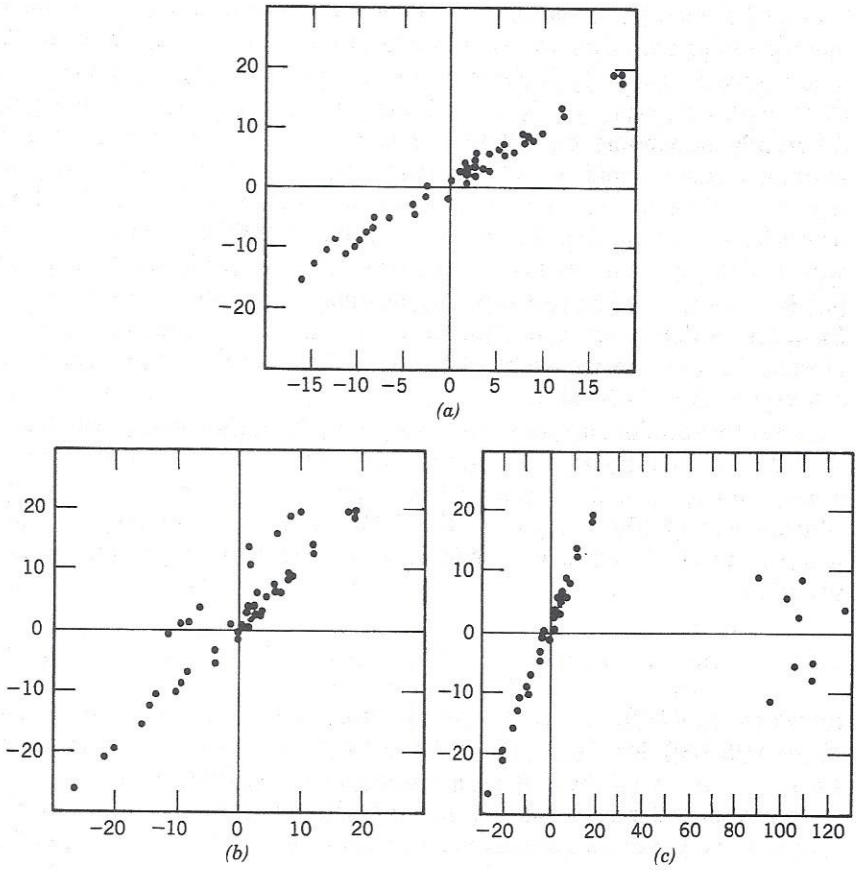


**Figure 4.** Examples of simulated data with 50 points: (*a*) normal situation, (*b*) outliers in $y$, and (*c*) outliers in $x$.

which yields the bias $\bar{\theta}_j - \theta_j$. Also, the *mean squared error*

$$\text{MSE}(\theta_j) = \frac{1}{m} \sum_{k=1}^{m} (\hat{\theta}_j^{(k)} - \theta_j)^2 \tag{4.3}$$

is an important quantity, which can be decomposed into a sum of the squared bias and the variance:

$$\text{MSE}(\theta_j) = (\bar{\theta}_j - \theta_j)^2 + \frac{1}{m} \sum_{k=1}^{m} (\hat{\theta}_j^{(k)} - \bar{\theta}_j)^2 . \tag{4.4}$$

The same summary statistics are also computed for the estimates of $\sigma$.

In our simulation we performed 200 replications for each value of $p$ between 1 and 10; for $n$ equal to 10, 20, 30, 40 and 50; and for all three sampling situations; this was repeated for models with and without intercept term. (This being only a modest study, we did not yet investigate the suitability of variance reduction techniques as described by Johnstone and Velleman 1985a.) Table 4 lists the results for regression with intercept term, for $n = 40$ and $p = 4$. (The tables for other values of $n$ and $p$ are all extremely similar, so we only display one of them to save space.)

In Table 4, we see that in the normal error situation the mean estimated values produced by the robust techniques are almost as good as the LS ones, whereas their MSE and variance do increase, but not dramatically. (The constant term produces the largest MSE and variance for all four techniques.) For all estimates, one also sees that the variance makes up most of the MSE, which means that the bias is negligible in this situation. To conclude, the LS provides the best results when the errors are normally distributed, but then the robust techniques also behave quite well.

However, the LS becomes very bad when there are outliers in $y$. In this situation the mean estimated intercept and scale differ considerably from the true value of 1, and they have a high MSE. This large MSE is mostly due to the bias, since the variances are rather small. This also means that the LS estimates behave in the same way in all 200 runs. In other words, the LS breaks down systematically at these contaminated samples. On the other hand, the robust regression estimates are scarcely altered when compared to the normal error situation. Only the scale estimate of the OSM is too large, which is because the outliers are not really rejected since the OSM scale is not defined by means of a redescending function. (Note that the LMS scale does contain a rejection step, given by (1.4) above.)

In the situation of outliers in $x$, we only have put contamination on $x_{i,1}$. The LS regressions are pulled in the direction of these leverage points, thereby affecting the mean estimated value of $\theta_1$ and causing a large MSE. The LS estimates of intercept and scale also break down in a spectacular way. On the other hand, the robust regressions still yield good results for all parameters. (Note that the OSM again performs somewhat less than the RLS, especially for the scale parameter. This is one of the reasons why we have preferred to use the RLS in the previous chapters, some of the other reasons being intuitive appeal and ease of interpretation.)

It may be noted that the LMS columns in Table 4 are *exactly* equal in the situations of outliers in $y$ and outliers in $x$, and the same holds for the RLS. This is not surprising since the samples in both situations share the

**Table 4. Simulation Results of Regression with Intercept, for $n = 40$ and $p = 4$ (Including the Constant Term)[a]**

| | Normal | | | | Outliers in $y$ | | | | Outliers in $x$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Parameter | LS | LMS | RLS | OSM | LS | LMS | RLS | OSM | LS | LMS | RLS | OSM |
| $\theta_1$ | 1.000365 | 1.001555 | 1.000796 | 1.000829 | 0.999226 | 0.999641 | 0.999238 | 0.999367 | 0.042766 | 0.999641 | 0.999238 | 0.999365 |
| | 0.000279 | 0.001350 | 0.000479 | 0.000619 | 0.004935 | 0.001262 | 0.000430 | 0.000556 | 0.918092 | 0.001262 | 0.000430 | 0.001224 |
| | 0.000279 | 0.001347 | 0.000479 | 0.000618 | 0.004934 | 0.001262 | 0.000429 | 0.000556 | 0.001794 | 0.001262 | 0.000429 | 0.001223 |
| $\theta_2$ | 1.000884 | 1.002744 | 1.001689 | 1.002117 | 1.004542 | 1.003353 | 1.001238 | 1.002902 | 1.006389 | 1.003353 | 1.001238 | 1.002395 |
| | 0.000268 | 0.001320 | 0.000481 | 0.000570 | 0.004563 | 0.001646 | 0.000533 | 0.000726 | 0.022725 | 0.001646 | 0.000533 | 0.000730 |
| | 0.000267 | 0.001313 | 0.000478 | 0.000565 | 0.004542 | 0.001635 | 0.000532 | 0.000717 | 0.022684 | 0.001635 | 0.000532 | 0.000724 |
| $\theta_3$ | 0.999125 | 0.996997 | 0.998000 | 0.997707 | 0.997524 | 0.997381 | 0.997542 | 0.997274 | 0.987313 | 0.997381 | 0.997542 | 0.996846 |
| | 0.000301 | 0.001598 | 0.000625 | 0.000785 | 0.005007 | 0.001500 | 0.000492 | 0.000694 | 0.031526 | 0.001500 | 0.000492 | 0.000731 |
| | 0.000300 | 0.001589 | 0.000621 | 0.000780 | 0.005001 | 0.001493 | 0.000486 | 0.000687 | 0.031365 | 0.001493 | 0.000486 | 0.000721 |
| Constant | 0.999892 | 1.007599 | 0.989971 | 0.990862 | 2.994885 | 0.981571 | 1.004995 | 0.996266 | 0.292098 | 0.981571 | 1.004995 | 0.997807 |
| | 0.023451 | 0.127050 | 0.034034 | 0.040130 | 4.041006 | 0.109773 | 0.033947 | 0.045180 | 3.089755 | 0.109773 | 0.033947 | 0.040871 |
| | 0.023451 | 0.126992 | 0.033934 | 0.040046 | 0.061441 | 0.109434 | 0.033922 | 0.045166 | 2.588630 | 0.109434 | 0.033922 | 0.040867 |
| $\sigma$ | 0.993785 | 0.937518 | 0.861388 | 1.029884 | 4.156255 | 1.094557 | 0.937264 | 1.442410 | 9.680913 | 1.094557 | 0.937264 | 1.461946 |
| | 0.013573 | 0.043511 | 0.047492 | 0.038412 | 10.005019 | 0.047954 | 0.026198 | 0.258837 | 76.745529 | 0.047954 | 0.026198 | 0.280485 |
| | 0.013535 | 0.039607 | 0.028279 | 0.037519 | 0.043075 | 0.039012 | 0.022262 | 0.063110 | 1.387274 | 0.039012 | 0.022262 | 0.067090 |

[a]The entries in each cell are the mean estimated value, the mean squared error, and the variance over the 200 runs.

**Table 5. Simulation Results of Regression Without Intercept, for $n = 50$ and $p = 8$[a]**

| Parameter | Normal | | | | Outliers in $y$ | | | | Outliers in $x$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | LS | LMS | RLS | OSM | LS | LMS | RLS | OSM | LS | LMS | RLS | OSM |
| $\theta_1$ | 1.001943 | 1.002612 | 1.001572 | 1.002003 | 0.999899 | 1.000960 | 1.001128 | 1.001258 | 0.034819 | 1.000960 | 1.001182 | 1.000959 |
| | 0.000218 | 0.001252 | 0.000405 | 0.000531 | 0.004761 | 0.001377 | 0.000326 | 0.000519 | 0.932610 | 0.001377 | 0.000325 | 0.001322 |
| | 0.000214 | 0.001245 | 0.000403 | 0.000527 | 0.004761 | 0.001376 | 0.000325 | 0.000517 | 0.001035 | 0.001376 | 0.000324 | 0.001321 |
| $\theta_2$ | 0.999810 | 0.999127 | 0.999727 | 1.000177 | 1.002180 | 1.000647 | 1.000132 | 1.000959 | 1.005823 | 1.000647 | 1.000297 | 1.000575 |
| | 0.000287 | 0.001180 | 0.000445 | 0.000537 | 0.004794 | 0.001459 | 0.000457 | 0.000618 | 0.020716 | 0.001459 | 0.000448 | 0.000620 |
| | 0.000287 | 0.001179 | 0.000445 | 0.000537 | 0.004789 | 0.001458 | 0.000457 | 0.000617 | 0.020682 | 0.001458 | 0.000448 | 0.000619 |
| $\theta_3$ | 1.001059 | 1.000904 | 1.000546 | 1.000880 | 0.996228 | 1.004423 | 1.002371 | 1.001842 | 0.991331 | 1.004423 | 1.002275 | 1.003173 |
| | 0.000233 | 0.001217 | 0.000470 | 0.000608 | 0.004548 | 0.001153 | 0.000327 | 0.000449 | 0.026097 | 0.001153 | 0.000327 | 0.000454 |
| | 0.000232 | 0.001216 | 0.000469 | 0.000607 | 0.004534 | 0.001134 | 0.000321 | 0.000445 | 0.026022 | 0.001134 | 0.000322 | 0.000444 |
| $\theta_4$ | 1.000511 | 1.000979 | 1.000768 | 1.000216 | 1.001978 | 1.003547 | 1.001338 | 1.002251 | 1.002652 | 1.003547 | 1.001330 | 1.002426 |
| | 0.000254 | 0.001116 | 0.000419 | 0.000498 | 0.004536 | 0.001325 | 0.000334 | 0.000507 | 0.025608 | 0.001325 | 0.000334 | 0.000526 |
| | 0.000290 | 0.001115 | 0.000419 | 0.000498 | 0.004532 | 0.001312 | 0.000333 | 0.000502 | 0.025601 | 0.001312 | 0.000333 | 0.000520 |
| $\theta_5$ | 1.000265 | 1.001546 | 0.998959 | 0.998897 | 0.999070 | 1.001347 | 0.999485 | 1.000078 | 0.999605 | 1.001347 | 0.999607 | 0.999709 |
| | 0.000290 | 0.001185 | 0.000467 | 0.000640 | 0.006565 | 0.001339 | 0.000426 | 0.000645 | 0.024870 | 0.001339 | 0.000419 | 0.000641 |
| | 0.000290 | 0.001183 | 0.000466 | 0.000639 | 0.006564 | 0.001337 | 0.000426 | 0.000645 | 0.024870 | 0.001337 | 0.000419 | 0.000641 |
| $\theta_6$ | 0.999495 | 0.997944 | 0.998165 | 0.997982 | 1.001850 | 1.002854 | 0.999834 | 1.000624 | 0.990852 | 1.002854 | 0.999726 | 1.000547 |
| | 0.000257 | 0.001186 | 0.000456 | 0.000574 | 0.004388 | 0.001341 | 0.000406 | 0.000536 | 0.020759 | 0.001341 | 0.000401 | 0.000559 |
| | 0.000257 | 0.001182 | 0.000452 | 0.000570 | 0.004384 | 0.001333 | 0.000406 | 0.000536 | 0.020675 | 0.001333 | 0.000400 | 0.000558 |
| $\theta_7$ | 1.000390 | 1.000006 | 1.000231 | 1.000143 | 1.002043 | 0.998583 | 1.000299 | 0.999266 | 1.001794 | 0.998583 | 1.000298 | 0.999228 |
| | 0.000267 | 0.001237 | 0.000469 | 0.000603 | 0.006173 | 0.001446 | 0.000349 | 0.000557 | 0.022490 | 0.001446 | 0.000349 | 0.000585 |
| | 0.000267 | 0.001237 | 0.000469 | 0.000603 | 0.006168 | 0.001444 | 0.000349 | 0.000556 | 0.022487 | 0.001444 | 0.000349 | 0.000584 |
| $\theta_8$ | 1.001183 | 1.002922 | 1.002269 | 1.002989 | 1.007825 | 0.999898 | 1.001737 | 1.000080 | 0.989495 | 0.999898 | 1.001769 | 1.000586 |
| | 0.000285 | 0.001232 | 0.000474 | 0.000590 | 0.005454 | 0.001277 | 0.000423 | 0.000550 | 0.023626 | 0.001277 | 0.000424 | 0.000566 |
| | 0.000284 | 0.001223 | 0.000469 | 0.000581 | 0.005393 | 0.001277 | 0.000420 | 0.000550 | 0.023515 | 0.001277 | 0.000421 | 0.000566 |
| $\sigma$ | 0.999105 | 1.030184 | 0.865524 | 0.920439 | 4.592490 | 1.270508 | 0.957984 | 1.302358 | 9.701845 | 1.268939 | 0.956662 | 1.324905 |
| | 0.013545 | 0.034845 | 0.039224 | 0.032424 | 12.959248 | 0.117720 | 0.022465 | 0.132850 | 76.718758 | 0.115622 | 0.021174 | 0.149173 |
| | 0.013544 | 0.033934 | 0.021140 | 0.026095 | 0.053265 | 0.044545 | 0.020700 | 0.041430 | 0.996646 | 0.043294 | 0.019296 | 0.043610 |

[a]The entries in each cell are the mean estimated value, the mean squared error, and the variance over the 200 runs.

80% of "good" observations, whereas the 20% of outliers are properly rejected by the LMS and the RLS. This is no longer true for the OSM, because there the outliers still have an (albeit very small) influence.

Table 5 contains the summary statistics for regression *without* intercept, with $n = 50$ and $p = 8$. (The results for other choices of $n$ and $p$ are very similar and hence are not listed here.) For the normal situation we arrive at the same conclusions as before, the robust techniques behaving almost as well as LS.

In the presence of outliers in the response variable, we find that the LS coefficients are still quite good. This is because the model now forces the fit to go through the origin, so the contamination (as in Figure 4*b*) cannot move up the fit as in a model with intercept. (Moreover, the contamination happens to be balanced about the origin, so the solution does not even tilt in this experiment.) However, the LS scale estimate clearly breaks down because all points are taken into account, including the bad ones. On the other hand, the robust estimators provide decent estimates of both the $\theta_j$ and $\sigma$. The simulation results also show that the RLS is really an improvement on the LMS as the mean squared errors are decreased even more, particularly that of the scale estimate. This is not true for the OSM, which again overestimates $\sigma$ because it does not really reject the outliers.

The last columns of Table 5 deal with the outliers in $x_{i,1}$, which cause the LS estimate of $\theta_1$ to break down. As was to be expected, also the corresponding scale estimate explodes. Fortunately, the robust techniques yield satisfactory estimates for all quantities. Moreover, the RLS again constitutes a slight improvement on the LMS results.

The overall conclusion of these simulations is that the LMS indeed achieves the goals for which it was constructed, because it gives reasonable results in the normal situation and is also able to withstand substantial amounts of outliers in $x$ and in $y$.

## EXERCISES AND PROBLEMS

### Section 1

1. Illustrate that the LMS algorithm in PROGRESS is regression, scale, and affine equivariant by running one or more example(s) in which you transform the data as in (4.2), (4.3), and (4.4) of Section 4 in Chapter 3.
2. Derive the probability (1.2).

3. Write down formulas expressing the regression coefficients $\hat{\theta}_j$ in terms of the regression results on the standardized data. Why are these formulas the same for LS, LMS, and RLS regression?

**Section 2**

4. Why is it useful to look at more than one local minimum of the objective function (2.2)? Consider both computational and data-analytic aspects.
5. (Research problem) Is it true that one can do no better than $O(n^p)$ algorithms for computing affine equivariant high-breakdown regression estimators?

**Section 3**

6. In Section 3 the computation times of the LMS, the LTS, and $S$-estimators are discussed. What do you think would be the total cost of a combination of LMS with a subsequent one-step improvement (either a one-step RLS or a one-step $M$-estimator)? Indicate the computation time of such an estimator in Figure 3.

**Section 4**

7. (Research problem) Why does the behavior of (4.1) depend on $n$ and $p$ being even or odd?