Evolutionary Algorithms for Robust Methods[☆]

Robin Nunkesser[a], Oliver Morell[b]

[a] *Faculty of Computer Science, TU Dortmund University, Germany*
[b] *Faculty of Statistics, TU Dortmund University, Germany*

Abstract

A drawback of robust statistical techniques is the increased computational effort often needed compared to non robust methods. Robust estimators possessing the exact fit property, for example, are NP-hard to compute. This means that—under the widely believed assumption that the computational complexity classes NP and P are not equal—there is no hope to compute exact solutions for large high dimensional data sets. To tackle this problem, search heuristics are used to compute NP-hard estimators in high dimensions. Here, an evolutionary algorithm that is applicable to different robust estimators is presented. Further, variants of this evolutionary algorithm for selected estimators—most prominently least trimmed squares and least median of squares—are introduced and shown to outperform existing popular search heuristics in difficult data situations.

The results increase the applicability of robust methods and underline the usefulness of evolutionary computation for computational statistics.

Key words: Evolutionary algorithms, robust regression, least trimmed squares (LTS), least median of squares (LMS), least quantile of squares (LQS), least quartile difference (LQD)

1. Introduction

Since the works of Box (1953) and Tukey (1960) the need for robust methods is apparent. The strong sensitivity of classical procedures to seemingly negligible deviations from the distributional assumptions calls for robust alternatives. In linear regression, positive-breakdown methods (Rousseeuw, 1997) are among the most important robust techniques. Unfortunately, the computation of these estimators is quite hard. More precisely, Bernholt (2005) shows that estimators with the *exact fit property* (whenever a majority of the observations lies on a hyperplane, an estimator with the exact fit property yields that hyperplane as the solution, see e.g. Rousseeuw, 1984) are NP-hard to compute. When accepting the widely believed assumption that the complexity classes NP and P are not equal (see e.g. Wegener, 2005 for an introduction to complexity theory),

[☆] The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, Reduction of complexity in multivariate data structures) is gratefully acknowledged.

Email addresses: robin.nunkesser@tu-dortmund.de (Robin Nunkesser),
morell@statistik.tu-dortmund.de (Oliver Morell)

we therefore have no hope to compute exact solutions for large high dimensional data sets.

A typical approach to tackle problems we cannot compute exactly is to use heuristics. Evolutionary computation (see e.g. De Jong, 2006) is a well established search heuristic in computer science and steadily gaining importance in computational statistics. Examples for the application of evolutionary computation in computational statistics include evolutionary clustering (Hruschka et al., 2006), association studies (Nunkesser et al., 2007), computation of robust estimators (Meyer, 2003; Morell et al., 2008), time series modeling (Baragona et al., 2004), and many more. Actually, the evolutionary algorithm presented by Morell et al. (2008) is the basis of the algorithm, we present here. We extend this algorithm to a framework that is applicable to different estimators, like the algorithm PROGRESS (Program for Robust Regression) (Rousseeuw and Leroy, 1987; Rousseeuw and Hubert, 1997) which is based on using subsamples of the data. Here, we concentrate on three robust estimators: least median of squares (LMS) (more precisely its generalization least quantile of squares (LQS)) and least trimmed squares (LTS), both proposed by Rousseeuw (1984) and as the third estimator least quartile difference (LQD) by Croux et al. (1994). Section 2 describes these estimators.

Section 3 provides a basic introduction to evolutionary computation, which is the basis of the algorithm presented in Section 4. In the final sections 5 and 6, we compare our algorithm to existing popular search heuristic and draw conclusions.

2. The Estimators LQS, LTS, and LQD

We consider the linear multiple regression model

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i \;\; i = 1, \ldots, n$$

where β_0 is an intercept term and ε_i models statistical errors. The p-dimensional vectors $x_i = (x_{i1}, \ldots, x_{ip})$ contain the explanatory variables and y_i the response. For estimated regression coefficients $\hat{\beta}_1, \ldots, \hat{\beta}_p$ and an estimated intercept $\hat{\beta}_0$, we denote the residuals as

$$r_i\left(\hat{\beta}_0, \ldots, \hat{\beta}_p\right) = y_i - \left(\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \ldots + \hat{\beta}_p x_{ip}\right) \;\; .$$

Further, let $u_{i:n}$ denote the i-th order statistic of n numbers $u_1, \ldots, u_n$. The estimators we consider are defined as follows:

Definition 1 (Rousseeuw, 1984; Croux et al., 1994). Let r_i be the ith residual determined by a linear regression with parameters $\hat{\beta}_0, \ldots, \hat{\beta}_p$ and a given data set Z.

The *least quantile of squares* (LQS), *least trimmed squares* (LTS), and *least quartile difference* (LQD) estimates are given by

$$\begin{aligned}
\mathrm{LQS}\,(Z) &= \underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \{r_1^2, \ldots, r_n^2\}_{h_p:n} \\
\mathrm{LTS}\,(Z) &= \underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \sum_{i=1}^{h_p} \{r_1^2, \ldots, r_n^2\}_{i:n} \\
\mathrm{LQD}\,(Z) &= \underset{\hat{\beta}_0,\ldots,\hat{\beta}_p}{\operatorname{argmin}} \{|r_i - r_j|\,; i < j\}_{\binom{h_p}{2}:\binom{n}{2}}
\end{aligned}$$

where h_p with $1 \leq h_p \leq n$ is a parameter influencing the estimation.

LQS generalizes the least median of squares (LMS), defined as

$$\mathrm{LMS}\,(Z) = \operatorname*{argmin}_{\hat{\beta}_0,\ldots,\hat{\beta}_p} \mathrm{med}\{r_1^2, \ldots, r_n^2\} \ .$$

Note also, that of course the *least trimmed sum of absolute values* (LTA) proposed by Hössjer (1994) and defined as

$$\mathrm{LTA}\,(Z) = \operatorname*{argmin}_{\hat{\beta}_0,\ldots,\hat{\beta}_p} \sum_{i=1}^{h_p} \{|r_1|, \ldots, |r_n|\}_{i:n}$$

may also be computed by our algorithm. All these estimators are high-breakdown methods, possessing the asymptotic breakdown point (Donoho and Huber, 1983) of 50% for the appropriate choice of h_p. This means that almost 50% of the observations may be contaminated without having unbounded effect on the estimate. LQS and LMS have the disadvantage of a low efficiency at Gaussian samples, which is asymptotically 0% (Croux et al., 1994). Arguments for LQS/LMS used to be the easier computation of the objective function compared to LTS and LQD—though the computation of the estimators is still NP-hard—and the intuitive definition. LTS and LQD are alternatives with higher asymptotic Gaussian efficiencies of 7.1% and 67.1%, respectively. An additional advantage of LTS is its smooth objective function leading to a lower sensitivity to local effects (Rousseeuw and Van Driessen, 2006).

Because of the above mentioned advantages and disadvantages, the considered estimators are relevant for different data situations. Therefore, algorithms for them are of high interest. We consider situations where exact algorithms are not feasible and concentrate on heuristics. The heuristics most commonly used are PROGRESS for LQS/LMS and LTS (Rousseeuw and Leroy, 1987; Rousseeuw and Hubert, 1997) and FAST-LTS for LTS (Rousseeuw and Van Driessen, 2006). LQD may be computed with LQS algorithms with a quadratic blow up of computation time (Croux et al., 1994). Aside from the best known heuristics, Hawkins and Olive (1999) propose a so called feasible solution algorithm (FSA) for LQS/LMS and LTS. The FSA is based on sampling data subsets of size h_p and iterative swapping of data points in the sample with data points outside the sample.

3. Evolutionary Computation

The idea of evolutionary computation is to mimic the Darwin–Wallace principle of natural selection in order to obtain an efficient search heuristic. Evolution's main principle is that *populations* of *individuals* evolve through variational inheritance where a concept of *fitness* reflects the ability to survive. Transfered to optimization, the population of individuals is a collection of candidate solutions, where the fitness reflects the goodness of the candidate solution, e.g. the objective value.

Individuals may be characterized by *genotypes* (the genetic makeup) or *phenotypes* (observed qualities) or in algorithmic terms the computer representation of the candidate solution and its (mathematical) interpretation if not apparent. Essential modules of evolutionary algorithms are therefore the fitness function that maps individuals to fitness values, the genotype search space and an optional mapping between genotype and phenotype.

Further, selection schemes determining which individuals are selected for variation, and concepts to modify individuals are needed (typically called *mutation* or *recombination/crossover* depending on whether one or more individuals are involved).

By nature, an evolutionary process is infinite. To obtain an algorithm that terminates, we additionally need a *stopping criterion.*

The basic evolutionary process used by evolutionary algorithms is described in Algorithm 1.

Algorithm 1 (Basic Evolutionary Algorithm).

1. Create an initial random population.
2. Perform the following steps on the current *generation* of individuals:
   (a) Select individuals in the population based on a selection scheme.
   (b) Adapt the selected individuals.
   (c) Evaluate the fitness value of the adapted individuals.
   (d) Select individuals for the next generation according to a selection scheme.
3. If the stopping criterion is fulfilled, then output the final population. Otherwise, set the next generation as current and go to step 2.

Evolutionary computation is a very general and adaptive framework and ideas used in evolutionary algorithms can also be found in existing algorithms for robust regression. For example, the point interchange used by Hawkins (1993) in his feasible set algorithm to compute LMS can be seen as a mutation operator.

4. Outline of the Algorithm

As a first step, we have to appoint the genotype of the individuals, we work on. In order to have a limited number of candidate solutions, we restrict ourselves to candidate solutions uniquely determined by a data subsample of fixed size. In the most common algorithms PROGRESS and FAST-LTS the subsamples are for sound reasons of size p. These reasons include the fact, that p linear independent points uniquely define a hyperplane. Additionally, smaller subsamples decrease the possibility of having outliers in the subsample. Although, strictly speaking, we are computing a different estimator when using subsamples of size p, it remains being a high breakdown estimator (Rousseeuw and Basset Jr., 1991). We will adopt this in letting for explanatory data $Z_e = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$

$$G = \left\{ (g_1, \ldots, g_n) \in \{0,1\}^n : \sum_{i=1}^{n} g_i = p \right\}$$

be the genotype of our individuals that is mapped to its phenotype by the function $m : G \to \{U \subseteq \mathbb{R}^p : |U| = p\}$ with

$$m\left((g_1, \ldots, g_n)\right) = \{x_i \in Z_e; g_i = 1\} \ .$$

Thus, we obtain $\binom{n}{p}$ different possible individuals. The determination of the fitness or goodness of these individuals comprises two steps:

1. Compute a unique candidate solution hyperplane H from the given individual.
2. Compute—depending on the estimator chosen—one of the three objective functions $\{r_1^2, \ldots, r_n^2\}_{h_p:n}$, $\sum_{i=1}^{h_p} \{r_1^2, \ldots, r_n^2\}_{i:n}$, or $\{|r_i - r_j|\,; i < j\}_{\binom{h_p}{2}:\binom{n}{2}}$ (the residuals r_i are determined by H and the given data).

The algorithm we propose is the following:

Algorithm 2 (Evolutionary Algorithm for Robust Regression).

1. Select $(g_1, \ldots, g_n) \in G$ uniformly at random, constituting the initial population.
   (a) Compute a unique hyperplane H from $(g_1, \ldots, g_n)$.
   (b) Determine the objective value f for the chosen estimator.
2. Perform the following steps on the current individual $(g_1, \ldots, g_n)$:
   (a) Conduct one of the following adaptions chosen uniformly at random:
      i. Randomly select $g_i, g_j \in (g_1, \ldots, g_n)$ with $g_i \neq g_j$ and exchange their values to obtain $(g'_1, \ldots, g'_n)$.
      ii. Randomly select $k > p$ and compute the index set I defined by

      $$\big\{i \in \{1, \ldots, n\}; r_i^2 \in \{r_1^2, \ldots, r_n^2\}_{(k-p+1):n} \cup \ldots \cup \{r_1^2, \ldots, r_n^2\}_{k:n}\big\}$$

      where r_i are residuals defined by H. Set $g'_i \in (g'_1, \ldots, g'_n)$ to 1 iff $i \in I$.
      iii. Select $(g'_1, \ldots, g'_n) \in G$ uniformly at random.
   (b) Compute a unique hyperplane H' from $(g'_1, \ldots, g'_n)$.
   (c) Determine the objective value f' for the chosen estimator.
   (d) If $f' < f$ set $(g_1, \ldots, g_n) = (g'_1, \ldots, g'_n)$, $H = H'$, and $f = f'$.
3. If either
   (a) the elapsed time,
   (b) the number of adaptions conducted, or
   (c) the number of adaptions without improved objective value

   exceeds its predetermined maximum, stop and output the final individual. Otherwise go to step 2.

The main difference to PROGRESS and FAST-LTS is that we have a continuous process changing subsamples instead of drawing a fixed number of subsamples. Thus, we are better able to use the information of good candidate solutions. Only using the mutation 2(a)i would lead to staying in local optima. The adaption described in 2(a)ii redeems this disadvantage in potentially moving far away from local optima, but still using information contained in the solution. Note, that the effect is similar to the adaption "move" described by Morell et al. (2008), but the adaption proposed here is simpler to compute. The adaption 2(a)iii introduces resampling to the algorithm.

The question how to compute a unique hyperplane from a subset of size p remains. As a first step, we compute the hyperplane H through the subset of data points. If it does not define a unique hyperplane, we try to add observations in fixed order (e.g. starting with x_1) until it does. The second step depends on the estimator and is described in more detail in the following. A third optional step is to adjust the intercept of the unique hyperplane by doing an LTS/LQS/LQD univariate estimate on the residuals defining the objective value (Rousseeuw and Hubert, 1997). We include this step in our algorithm.

4.1. Computing a Unique Hyperplane for LTS

Rousseeuw and Van Driessen (2006) show that the following procedure guarantees an improvement in the objective value of the LTS estimate$H$:

1. Determine the h_p points with the lowest squared residuals with regard to H.
2. Compute the least squares fit hyperplane H' on these h_p points.

4.2. Computing a Unique Hyperplane for LQS

The situation for LQS is more complex. The following procedure also guarantees an improvement in objective value (Stromberg, 1993):

1. Determine the h_p points with the lowest squared residuals with regard to H.
2. Compute a minimax fit hyperplane H' on these h_p points defined by

   $$\underset{\hat{\beta}_0,\dots,\hat{\beta}_p}{\operatorname{argmin}} \max\{r_1^2, \dots, r_{h_p}^2\}$$

   where $r_1, \dots, r_{h_p}$ are the residuals of the chosen h_p points.

One possibility to compute such a fit is based on computing least squares fits on all $\binom{h_p}{p+1}$ possible subsamples of size $p+1$ (Stromberg, 1993). For large h_p or p this is clearly prohibitive. A second possibility is to do a least squares fit on a subset of the data uniquely defined by H. We propose to use least squares on the h_p points with the smallest squared residuals with regard to H. This often leads to an improved objective value, but different to LTS regression the improvement is not guaranteed. Thus we only choose H' instead of H as the unique hyperplane, if H' leads to a better objective value. A third possibility to improve the objective value is to do weighted least squares on the data points with the lowest squared residuals with regard to H and give higher weight to the data points with higher residuals.

4.3. Computing a Unique Hyperplane for LQD

The similarity of LQD to LQS allows us to use the following procedure to obtain a guaranteed better objective value:

1. Determine the set of points $Z = \{(x_i - x_j, y_i - y_j)\}$ such that the corresponding absolute residual differences $|r_i - r_j|$ with regard to H are among the h_p smallest.
2. Compute a minimax fit hyperplane H' on Z.

It is easy to see, that this procedure guarantees a better objective value, because Croux and Rousseeuw (1992) showed that it is possible to compute the LQD by computing an LQS on the data set of differences $\{(x_i - x_j, y_i - y_j)\,; 1 \leq i < j \leq\}$. This is however, due to a typically larger h_p than in LQS estimation, even more prohibitive than in case of the LQS for large h_p or p. Again, a good alternative is to use least squares fits on subsets of the data (this time on the according data sets of differences).

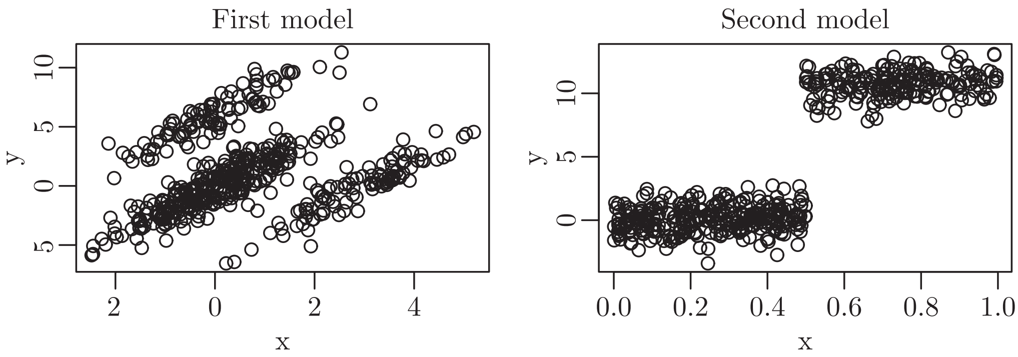

Figure 1: Example of simulated data for $p = 1$.

5. Comparison

We compare our algorithm with results from PROGRESS for LQS and results from FAST-LTS for LTS. We omit the LQD, because—as we have seen—a transformation to LQS exists and to our knowledge, no implementation of LQD algorithms for high dimensional data exist. PROGRESS is implemented in the function `lqs` of the R (R Development Core Team, 2008) package MASS (Venables and Ripley, 2002) and FAST-LTS is implemented in the function `ltsReg` of the R package robustbase (Todorov et al., 2007). Our own algorithms are implemented in the funtion `robreg.evol` of the R package RFreak (Nunkesser, 2008).

To compare the algorithms, we simulate data with $n = 500$ data points for $p = 1, \ldots, 30$ from two different models. In the first model, the independent regressors are normally distributed. In the second model, they stem from a uniformly distributed random design on the interval $(0, 1)$. The first model is given by

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \varepsilon_i \;\; i = 1, \ldots, n$$

where β_0 is an intercept term and $\varepsilon_i \sim \mathcal{N}(0, 1)$ are statistical errors. The parameter β_0 is set to 0, while $\beta_1, \ldots \beta_p$ equal 2. We add 40% outliers to the data in choosing two disjunct samples of size 20%. We add 3 to the explanatory data in the first sample and 6 to the response in the second sample, generating additive outliers in the explanatory and the response variables, respectively.

The second model contains a structural change. The parameter β_1 is 1, while $\beta_2, \ldots \beta_p$ are 0. Thus, the corresponding regressors add only noise to the problem. The structural change is in the intercept, which is

$$\beta_0 = \begin{cases} 0, & \text{if } x_{i1} \leq \{x_{j1}; 1 \leq j \leq 500\}_{300:500} \\ 10, & \text{if } x_{i1} > \{x_{j1}; 1 \leq j \leq 500\}_{300:500} \end{cases} .$$

A good robust estimate of β should give β_0 close to 0 and the slopes as above. Figure 1 shows an example of two simulated data sets based on these models with $p = 1$.

To compare the algorithms fairly, we measure the runtime `lqs` and `ltsReg`, respectively, need for the computation and give our algorithm exactly the same amount of time. Figure 2 shows the results for LTS and Figure 3 shows the results for LMS.

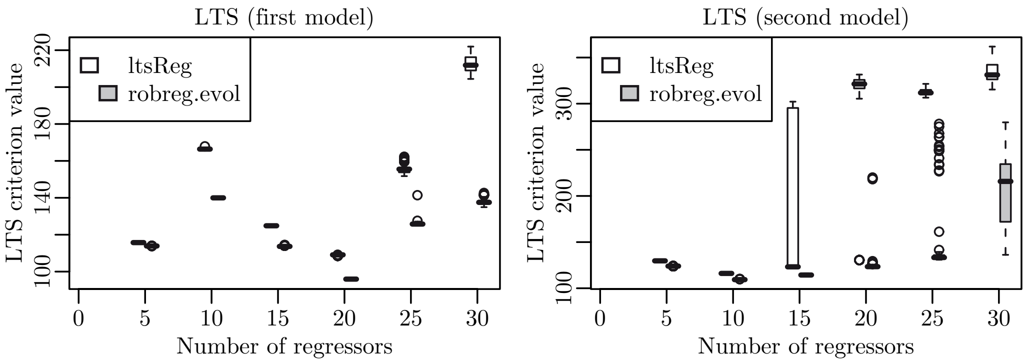

Figure 2: Comparison of `robreg.evol` with `ltsReg`.

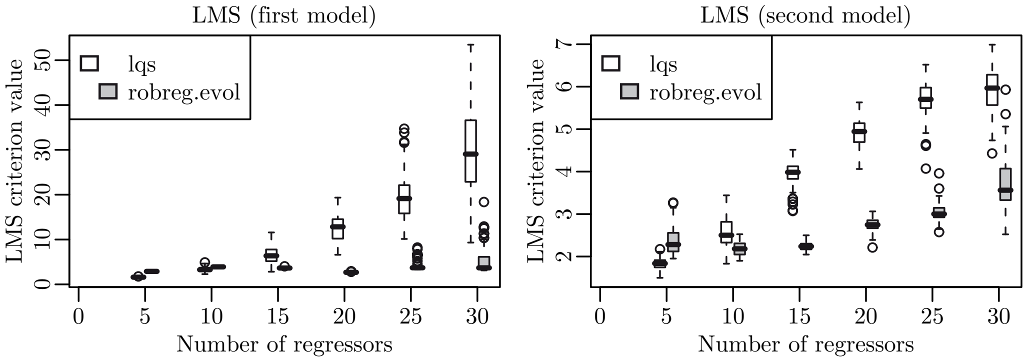

Figure 3: Comparison of `robreg.evol` with `lqs`.

In nearly all conducted runs, our algorithm achieves better results than the established algorithms `lqs` and `ltsReg`. Another observation is, that the structural change in the second data set affects the LTS estimation of `ltsReg` heavily for $p \geq 15$ while the effect on `robreg.evol` is far less. All in all, `robreg.evol` seems to be better suited for the considered data situations.

6. Conclusion

Many high breakdown estimators are in all likelihood not computable exactly in high dimensional regressor spaces. The common heuristics to compute solutions in these cases work with subsample versions of the estimators. In demanding data situations with a high percentage of contamination, the existing algorithms do not work well. The algorithm we propose is able to handle a high percentage of contamination in a high dimensional regressor space. In addition, it also provides superior results on data with less contamination and lower dimension considered in this paper. It therefore is a considerable alternative to the popular algorithms PROGRESS and FAST-LTS.

References

Baragona, R., Battaglia, F., Cucina, D., 2004. Fitting piecewise linear threshold autoregressive models by means of genetic algorithms. Computational Statistics & Data Analysis 47 (2), 277–295.

Bernholt, T., 2005. Robust estimators are hard to compute. Tech. Rep. 52/2005, SFB 475, Universität Dortmund.

Box, G. E. P., 1953. Non-normality and tests on variances. Biometrika 40 (3-4), 318–335.

Croux, C., Rousseeuw, P. J., 1992. Time-efficient algorithms for two highly robust estimators of scale. Computational Statistics 1, 411–428.

Croux, C., Rousseeuw, P. J., Hössjer, O., 1994. Generalized s-estimators. Journal of the American Statistical Association 89, 1271–1281.

De Jong, K. A., 2006. Evolutionary Computation: A Unified Approach. The MIT Press, Cambridge, Mass.

Donoho, D., Huber, P., 1983. The notion of breakdown point. In: Bickel, P., Doksum, K., Hodges, Jr., J. (Eds.), A Festschrift for Erich L. Lehmann. Wadsworth, Belmont, Calif., pp. 157–184.

Hawkins, D. M., 1993. The feasible set algorithm for least median of squares regression. Computational Statistics & Data Analysis 16 (1), 81–101.

Hawkins, D. M., Olive, D. J., 1999. Improved feasible solution algorithms for high breakdown estimation. Computational Statistics & Data Analysis 30 (1), 1–11.

Hössjer, O., 1994. Rank-based estimates in the linear model with high breakdown point. Journal of the American Statistical Association 89 (425), 149–158.

Hruschka, E. R., Campello, R. J., de Castro, L. N., 2006. Evolving clusters in gene-expression data. Information Sciences 176 (13), 1898–1927.

Meyer, M. C., 2003. An evolutionary algorithm with applications to statistics. Journal of Computational & Graphical Statistics 12 (2), 265–281.

Morell, O., Bernholt, T., Fried, R., Kunert, J., Nunkesser, R., 2008. An evolutionary algorithm for lts-regression: A comparative study. In: Brito, P. (Ed.), COMPSTAT 2008: Proceedings in Computational Statistics. Vol. II (Contributed Papers). Physica-Verlag, Heidelberg, pp. 585–593.

Nunkesser, R., 2008. Rfreak–an r package for evolutionary computation. Tech. rep., SFB 475, Technische Universität Dortmund.

Nunkesser, R., Bernholt, T., Schwender, H., Ickstadt, K., Wegener, I., 2007. Detecting high-order interactions of single nucleotide polymorphisms using genetic programming. Bioinformatics 23 (24), 3280–3288.

R Development Core Team, 2008. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Rousseeuw, P., Hubert, M., 1997. Recent developments in PROGRESS. In: Dodge, Y. (Ed.), L_1-Statistical Procedures and Related Topics. Vol. 31 of Lecture Notes-Monograph Series. Institute of Mathematical Statistics, Beachwood, Ohio, pp. 201–214.

Rousseeuw, P. J., 1984. Least median of squares regression. Journal of the American Statistical Association 79, 871–880.

Rousseeuw, P. J., 1997. Introduction to positive-breakdown methods. In: Maddala, G. S., Rao, C. R. (Eds.), Handbook of Statistics. Vol. 15. Elsevier North-Holland, Inc., Amsterdam, The Netherlands, pp. 101–121.

Rousseeuw, P. J., Basset Jr., G. W., 1991. Robustness of the p-subset algorithm for regression with high breakdown point. In: Stahel, W., Weisberg, S. (Eds.), Directions in Robust Statistics and Diagnostics, Part II. Vol. 34 of The IMA Volumes in Mathematics and its Applications. Springer, New-York, pp. 185–194.

Rousseeuw, P. J., Leroy, A. M., 1987. Robust Regression and Outlier Detection. John Wiley & Sons Inc., New York.

Rousseeuw, P. J., Van Driessen, K., 2006. Computing lts regression for large data sets. Data Mining and Knowledge Discovery 12 (1), 29–45.

Stromberg, A. J., 1993. Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression. SIAM Journal on Scientific Computing 14 (6), 1289–1299.

Todorov, V., Ruckstuhl, A., Salibian-Barrera, M., Maechler, M., 2007. robustbase: Basic Robust Statistics. R package version 0.2-8.

Tukey, J. W., 1960. A survey of sampling from contaminated distributions. In: Olkin, I., Ghurye, S. G., Hoeffding, W., Madow, W. G., Mann, H. B. (Eds.), Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling. Stanford: Stanford University Press, pp. 448–485.

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York.

Wegener, I., 2005. Complexity Theory. Springer, Berlin Heidelberg.