Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

Nonsingular Subsampling for Regression S Estimators with Categorical Predictors

Manuel Koller · Werner A. Stahel

the date of receipt and acceptance should be inserted later

Abstract Simple random subsampling is an integral part of S estimation algorithms for linear regression. Subsamples are required to be nonsingular. Usually, discarding a singular subsample and drawing a new one leads to a sufficient number of nonsingular subsamples with a reasonable computational effort. However, this procedure can require so many subsamples that it becomes infeasible, especially if levels of categorical variables have low frequency.

A subsampling algorithm called *nonsingular subsampling* is presented, which generates only nonsingular subsamples. When no singular subsamples occur, nonsingular subsampling is as fast as the simple algorithm, and if singular subsamples do occur, it maintains the same computational order. The algorithm works consistently, unless the full design matrix is singular. The method is based on a modified LU decomposition algorithm that combines sample generation with solving the least squares problem. The algorithm may also be useful for ordinary bootstrapping.

Since the method allows for S estimation in designs with factors and interactions between factors and continuous regressors, we study properties of the resulting estimators, both in the sense of their dependence on the randomness of the sampling and of their statistical performance.

Keywords robust regression, MM estimate, S estimate, resampling, collinearity, bootstrap, dummy variables

1 Introduction

S estimators are an important class of robust estimators for linear regression models. Computing them requires solving a non-convex optimization problem, which is possible only for very small number p of coefficients to be estimated. The algorithms used in practice are based on drawing random subsamples. These determine the starting points for local optimization. Then the best local optimum determines the estimate.

The most commonly used version draws subsamples of size p from the n observations. Such a subsample $I = \{i_1, i_2, ..., i_p\}$ is called an "elemental subset" because it allows the coefficients to be determined by solving the respective ordinary linear system of equations $\boldsymbol{X}_I\boldsymbol{\beta} = \boldsymbol{Y}_I$ – if the design matrix $\boldsymbol{X}_I$ is nonsingular.

Since the number of different elemental subsets, $\binom{n}{p}$, increases with an order of n^p, it is feasible to enumerate all of them only for very small datasets. In practice, random subsamples are drawn. If one obtains at least one outlier-free subsample with nonsingular $\boldsymbol{X}_I$, and if one can choose the number s of nonsingular subsamples so that this is achieved with high probability (see Section 2), then the resulting estimator will remain robust in the sense of avoiding breakdown. The computational effort

Manuel Koller
Institute of Social and Preventive Medicine, University of Bern, Bern, Switzerland E-mail: koller@stat.math.ethz.ch

Manuel Koller · Werner A. Stahel
Seminar für Statistik, ETH Zürich, Zürich, Switzerland

Werner A. Stahel
E-mail: stahel@stat.math.ethz.ch

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

needed to arrive at the number s depends critically on the probability with which a subsample produced by the algorithm is nonsingular, that is, has a nonsingular $\boldsymbol{X}_I$.

For continuous regressors, simple random subsamples of the design matrix are almost surely nonsingular under a mild and obvious regularity condition called "general position". In this case, the simple random subsampling algorithm is a good choice. Even if some singular samples occur, e.g., because of rounding effects, it will work by simply discarding any singular subsample and drawing another one in its place. For categorical predictors, however, singular subsamples can have an arbitrarily high probability; thus, the number of trials it takes to obtain a desired number of nonsingular subsamples may tend to infinity.

The problem can be illustrated with a simple example: Imagine a one-way ANOVA with 3 groups and 3 repetitions each. If we use treatment contrasts to encode the grouping structure, we get the following regression model $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ with

$$\boldsymbol{X}^\intercal = \begin{bmatrix} 0\,0\,0\,0\,0\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,0\,0\,0 \\ 1\,1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix}$$

The size of an elemental subset is 3. We can estimate the coefficients $\boldsymbol{\beta}$ only if the subsample contains at least one observation of each group. Of all 84 possible subsamples, only 27 are *good* in this way. Even in this very simple example, there is about a two-thirds probability that collinearities will necessitate discarding a subsample. The probability of discard increases steeply when some levels of factors are relatively infrequent. In this case, the discard rate is excessive and requires an enormous number of primary subsamples before the desired number of nonsingular ones is reached. In this situation, simple random subsampling algorithms are infeasible.

Instead of discarding the whole subsample, we solve this problem by drawing observations sequentially and discarding any observation that causes singularity. We call this refined subsampling strategy *nonsingular subsampling*. The algorithm is much faster than simple random subsampling for solving hard problems, and it is just as fast for solving easy problems.

Maronna and Yohai (2000) proposed another approach to the problem. They used a combination of M and S estimators, called an *M-S estimator*. An M estimator obtains the coefficients for the categorical variables, while an S estimator determines the coefficients for the continuous variables. Their approach works well for models containing only main effects of categorical and continuous variables. For models with interactions of categorical and continuous variables, however, it is not clear how the model matrix should be split. Using the M estimator for the interaction will decrease robustness, but using the S estimator will produce singular subsamples.

For models with categorical predictors, the breakdown points of the estimators are generally lower than for data with only continuous predictors. The maximum possible breakdown point depends on the design matrix. Several authors have considered the problem of breakdown in data with categorical predictors (Davies and Gather, 2005; Hampel, 1975; Mili et al, 1991; Ruckstuhl, 1995; Stahel et al, 1994). Mili and Coakley (1996) showed that S estimators reach the maximum possible breakdown point. Replacing ordinary subsampling with nonsingular subsampling does not change the breakdown point of the S estimator, i.e., for designs where the S estimator can be computed using ordinary subsampling, the S-estimator computed with the new method with the same number of nonsingular subsamples will have exactly the same breakdown point (and other properties).

S estimators may not work well for designs with factors. This is reflected in an instability of the estimates with respect to the randomness of the subsampling, whether classical subsampling could be successfully applied or nonsingular subsampling was used.

In the next section, we introduce the notation, provide essential definitions, and describe the basic algorithm for computing S estimates. In Section 3, the nonsingular subsampling algorithm is introduced. Some definitions and findings about the estimators in which the algorithm is used are recalled in Section 4. We analyze an example in Section 5, and then, in Section 6, we ask if choosing appropriate values for the algorithm's control parameters will allow us to strike a balance between performance and computing. In Section 7, we assess the quality of the resulting estimates. We conclude with Section 8.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

2 S estimators

Consider the standard multiple linear regression model,

$$y_i = \boldsymbol{x}_i^\intercal \boldsymbol{\beta} + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \quad i = 1, \ldots, n,$$

with ε_i i.i.d., independent of $\boldsymbol{x}_i$, and $\boldsymbol{\beta}$ a vector of length p. Throughout this text, we assume that the design matrix $\boldsymbol{X}$ with row vectors $\boldsymbol{x}_i$, is of full rank p. For singular design matrices, one has to select a subset of columns of full rank, and apply the methods to this design matrix. We denote residuals as $r_i(\widehat{\boldsymbol{\beta}}) = y_i - \boldsymbol{x}_i^\intercal \widehat{\boldsymbol{\beta}}$.

Simultaneous M estimates of regression and scale are defined as the solutions $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}$ to,

$$\sum_{i=1}^{n} \psi\left(r_i\left(\widehat{\boldsymbol{\beta}}\right) \Big/ \widehat{\sigma}\right) \boldsymbol{x}_i = \boldsymbol{0}, \tag{1}$$

$$\frac{1}{n} \sum_{i=1}^{n} \chi\left(r_i\left(\widehat{\boldsymbol{\beta}}\right) \Big/ \widehat{\sigma}\right) = \kappa, \tag{2}$$

where ψ is a ψ-function, χ is a ρ-function, and κ is a scalar used to achieve Fisher consistency. A ρ-function, as defined in Maronna et al (2006), is a non-decreasing function of $|r|$, with $\rho(0) = 0$, which strictly increases for $r > 0$ with $\rho(r) < \rho(\infty)$. If ρ is bounded, we also assume it is standardized to $\rho(\infty) = 1$. A ψ-function is proportional to the derivative of a ρ-function (generally different from χ) and is usually standardized to $\psi'(0) = 1$. A ψ-function that is not monotonically non-decreasing is called a *redescending* ψ-function. A solution $\widehat{\sigma}(\boldsymbol{r})$ to (2) for a given vector $\boldsymbol{r}$ is called *M estimate of scale*. The solution vector $\widehat{\boldsymbol{\beta}}$ to (1) is called an *M estimate of regression*.

An *S estimate of regression* is the parameter value $\widehat{\boldsymbol{\beta}}$ that minimizes the M estimate of scale $\widehat{\sigma} = \widehat{\sigma}(\boldsymbol{r}(\widehat{\boldsymbol{\beta}}))$ of the associated residuals,

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, \widehat{\sigma}\left(\boldsymbol{r}\left(\boldsymbol{\beta}\right)\right). \tag{3}$$

The maximal finite sample breakdown point of the S estimate in continuous designs is $(1-p/n)/2$, attained when using $\kappa = (1-p/n)/2$. Note that any solution of (3) is also the solution to the corresponding simultaneous M estimation of regression and scale problem, where $\psi(r) \propto \chi'(r)$. For an introduction to M estimation and details about S estimates, see Maronna et al (2006).

We mentioned in the Introduction that minimization is carried out in practice by choosing subsamples of size p, solving the corresponding linear system, and using the result for a local optimization. We call the last step, the local optimization, *refinement*. The refinement itself consists of an iteration of *refinement steps*. This is expressed in Algorithm 1, for the case in which exhaustive resampling (running over all possible subsamples with a well-defined solution of the linear system $\widehat{\boldsymbol{\beta}}_I$) is feasible.

Data: $n \times p$ matrix $\boldsymbol{X}$, response vector $\boldsymbol{y}$.
Result: $\widehat{\boldsymbol{\beta}}$, $\widehat{\sigma}$.

```
σ̂ ← ∞
for each elemental subset I ⊂ {1, ..., n} of size p do
    if design matrix X_I contains no collinearities then
        β̂_I0 ← X_I^{-1} y_I
        β̂_I, σ̂_I ← Refine β̂_I0 by solving (2) and (1) using an iterative reweighing algorithm.
        if σ̂_I < σ̂ then
            σ̂ ← σ̂_I
            β̂ ← β̂_I
```

Algorithm 1: Basic algorithm for the computation of S estimates.

The number of possible subsamples increases with order n^p. When this number is large, it is neither feasible nor necessary to look at all subsamples. A random set of subsamples can be

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

considered instead. Depending on the expected proportion of outliers, simple combinatorics can be used to determine the number of random subsamples required to select at least one outlier-free subsample with a probability of, say, 0.99 (see Table 5.3 of Maronna et al, 2006). The number required grows exponentially with p. Since this number is often still too high, a number of $s = 1000$ is a pragmatic choice.

Since the refinement step is the most computationally intensive part of Algorithm 1, it is the most attractive part to be optimized for speed. Salibian-Barrera and Yohai (2006) propose calculating only one or a few refinement steps for each subsample in a first phase, and carrying out full refinement iterations for only a few candidates to obtain a number h of "best" values in a second phase. This modified algorithm is therefore characterized by 3 control parameters: the number of subsamples to be generated; the number of refinement steps calculated for each of them; and the number of candidates for which the full refinement iteration is done. We come back to these choices in Section 5.

3 Nonsingular subsampling

Nonsingular subsampling avoids the problem of generating singular subsamples. We start by describing a method from numerical mathematics.

An *LU decomposition* of a square matrix $\boldsymbol{A}$ is defined as the product of a unit lower-triangular matrix $\boldsymbol{L}$ and an upper-triangular matrix $\boldsymbol{U}$. Such a decomposition generally exists only if one allows for suitable re-ordering of rows (by a permutation matrix $\boldsymbol{P}$). This leads to

$$\boldsymbol{A} = \boldsymbol{PLU}.$$

The ordering is best chosen to minimize numerical errors, and the procedure, useful for solving systems of linear equations $\boldsymbol{A\beta} = \boldsymbol{y}$, is called an *LU decomposition with partial pivoting*.

The Gaxpy variant (Golub and Van Loan, 1996) increases speed and computes the LU decomposition observation by observation. To compute the ith column of $\boldsymbol{L}$, only columns 1 to i of $\boldsymbol{U}$ are needed. This is a useful feature if there are singularities. When a singularity is detected, the ith observation can be dropped, and we repeat only the last step with the following observation. Prior results do not need to be recomputed. The computational cost of detecting a singular subsample is one step in the LU decomposition algorithm. With this modification, if the full design matrix has full rank, the algorithm always ends up with a nonsingular subsample, except for ill conditioned matrices where the algorithm might fail because of numerical instabilities (see Remark 1). A detailed description is given in Algorithm 2.

If there is no collinearity, the algorithm reduces to one of the preferred numerical algorithms to solve the system for the subsample. The subsample is identical to what would have been obtained by simple random subsampling. Randomly permuting all observations and then starting with the first observation achieves random selection of the subsample.

Remark 1. The *numerical stability* of the algorithm can be improved by using a matrix preconditioning technique to lower the condition number of the design matrix. We propose to use *equilibration*, described, e.g., in Demmel (1997). It is sufficient to apply this method to the complete design matrix, even if only submatrices are used in the procedure.

Remark 2. Initial nonsingularity of the sample does not guarantee that the fit will stay nonsingular in the refinement step. In this step, observations can be removed from the sample by assigning a zero robustness weight, which may lead to a singular design matrix. Examples of this can be easily constructed. Using one-step updates for both the regression coefficients and the scale, rather than full iterations for scale estimation, can mitigate the problem. A ψ-function that always gives positive weights may be used to avoid that problem altogether.

Remark 3. For data with little contamination, a slightly different sampling strategy will be preferable: Instead of applying the algorithm for each new sample to a new permutation of the data, a single permutation can be used to generate up to n/p samples. Each new sample is started at the first observation that has not yet been used by the previous samples. When a later sample runs out of observations, the first few observations can be recycled, and then a new random permutation must be drawn. If fewer than n/p observations are contaminated, this modification guarantees at least one clean subsample for each run through the permuted dataset.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

4 MM and SMDM estimators

S estimators achieve a high breakdown point at the cost of low asymptotic efficiency. Thus, S estimates are typically not used alone, but as an auxiliary ingredient of MM estimates. These estimators can attain a high breakdown point and high asymptotic efficiency. MM estimates are computed in two steps. In the first step, a high-breakdown regression and scale estimate, typically an S estimate, is computed. In the second step, an M estimator of regression with a redescending ψ-function, tuned for higher asymptotic efficiency, is computed in an iterative procedure that starts from the estimate found in the first step. For designs with categorical variables, an M-S estimator

Data: $n \times p$ matrix $\boldsymbol{X}$, response vector $\boldsymbol{y}$, singularity threshold ε.
Result: Return code (0 for success, otherwise number of failing step), initial estimate $\widehat{\boldsymbol{\beta}}$.
Initialize variables:
pivoting table $\boldsymbol{p}$, selected subsample index vector $\boldsymbol{s}$, counter k
1 $\boldsymbol{U} \leftarrow \boldsymbol{0}$; $\boldsymbol{L} \leftarrow \boldsymbol{I}$; $\boldsymbol{p} \leftarrow 1:p$; $\boldsymbol{s} \leftarrow 1:p$; $k \leftarrow 1$
Permute observations randomly
2 $\boldsymbol{t} \leftarrow \texttt{perm}(1:n)$
3 $\boldsymbol{A} \leftarrow \boldsymbol{X}^{\intercal}_{\boldsymbol{t},1:p}$
4 $\boldsymbol{y} \leftarrow \boldsymbol{y}_{\boldsymbol{t}}$
5 **for** j **in** 1 **to** p **do** ## Find nonsingular subsample and calculate LU decomposition
6 **if** $j == 1$ **then** $\boldsymbol{v}_{1:p} \leftarrow \boldsymbol{A}_{1:p,k}$
7 **else** ## (Forward)solve to get required column of $\boldsymbol{U}$
8 $\boldsymbol{U}_{1:j-1,j} \leftarrow \boldsymbol{L}^{-1}_{1:j-1,1:j-1}\boldsymbol{A}_{1:j-1,k}$
9 $\boldsymbol{v}_{j:p} \leftarrow \boldsymbol{A}_{j:p,k} - \boldsymbol{L}_{j:p,1:j-1}\boldsymbol{U}_{1:j-1,j}$
10 **if** $j < p$ **then** ## Find pivot
11 $\mu \leftarrow \operatorname{argmax}_{l=j}^{p} |\boldsymbol{v}_l|$
12 **if** $|\boldsymbol{v}_\mu| \geq \varepsilon$ **then** ## Subsample is still nonsingular
13 $\boldsymbol{p}_j \leftarrow \mu$
14 $\boldsymbol{s}_j \leftarrow k$
 ## Swap elements of $\boldsymbol{v}$ and rows of $\boldsymbol{A}$
15 $\boldsymbol{v}_j \leftrightarrow \boldsymbol{v}_\mu$
16 $\boldsymbol{A}_{j,k+1:n} \leftrightarrow \boldsymbol{A}_{\mu,k+1:n}$
 ## Update $\boldsymbol{L}$
17 $\boldsymbol{L}_{j+1:p,j} \leftarrow \boldsymbol{v}_{j+1:p}/\boldsymbol{v}_j$
 ## Swap rows of $\boldsymbol{L}$
18 $\boldsymbol{L}_{j,1:j-1} \leftrightarrow \boldsymbol{L}_{\mu,1:j-1}$
19 **if** $|\boldsymbol{v}_j| < \varepsilon$ **then** ## Singularity detected: skip column and try again if possible
20 **if** $k < n$ **then**
21 $k \leftarrow k + 1$
22 **Goto** *6*
23 **else** ## Return with an error
24 **return** j
25 $\boldsymbol{U}_{j,j} \leftarrow \boldsymbol{v}_j$
26 $k \leftarrow k + 1$
 ## Solve $X^{-1}_{\boldsymbol{s},1:p}\boldsymbol{y}_{\boldsymbol{s}}$ and undo pivoting
27 $\widehat{\boldsymbol{\beta}} \leftarrow \boldsymbol{L}^{-\intercal}\boldsymbol{U}^{-\intercal}\boldsymbol{y}_{\boldsymbol{s}}$
28 **for** j **in** $p-1$ **to** 1 **do** $\widehat{\boldsymbol{\beta}}_j \leftrightarrow \widehat{\boldsymbol{\beta}}_{\boldsymbol{p}_j}$
29 **return** 0, $\widehat{\boldsymbol{\beta}}$

Algorithm 2: Constrained subsampling using modified Gaxpy variant of LU decomposition. We use vector index notation to select subvectors and submatrices. The index vector $1:p-1$ in $\boldsymbol{v}_{1:p-1}$ indicates that an operation only acts on the elements 1 to $p-1$ of the vector.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

(Maronna and Yohai (2000)) can be used in the first step instead of an S estimator. For details about MM estimates, see Maronna et al (2006).

MM estimators use the scale parameter of the initial estimator for standardizing residuals before the ψ-function in the M step is applied. Koller and Stahel (2011) showed that a further refinement is useful for improving the properties of the final estimator. We suggested re-estimating the scale based on residuals of the MM estimate, and adding another M step for the regression coefficients. For the scale step, we used the so called Design Adaptive Scale (DAS), and called it a D step. The final result therefore builds on four consecutive steps and is called an *SMDM estimator.* In this notation, MM estimators should be called SM estimators, and, when combined with M-S initialization, may be called (M-S)M, possibly refined to (M-S)MDM.

These estimators are conceived as approximate solutions of implicit equations and should, in principle, be independent of the initial estimate on which they rely–as long as they have a high breakdown point. However, the scale estimate obtained in the initial step (S or M-S) is influenced by the random selection of subsamples. The result of the subsequent step still reflects this random variability – more so for the simple M than a MDM version. In order to minimize this artificial randomness of the estimate for a given dataset, we will iterate the D and M steps until convergence in this study. Calling an M step preceded by a D step M_D, this results in the estimators SM_D and (M-S)M_D.

In the following sections, we assume that S estimates have been computed using nonsingular subsampling.

5 Example

As an example, we analyze NOx air pollution data, a typical medium-sized environmental data set distributed with the R package `robustbase` (Rousseeuw et al, 2015). The dataset consists of hourly measurements of NOx pollution content in the ambient air (`NOx`) near a busy motorway, along with hourly sums of NOx emission on the motorway (`NOxEm`) and wind speed (`WS`). We use a subset of the dataset consisting of 11 days. We choose the 7 days with the most number of outliers, according to the robust fit to the full data set without interactions. We add four other days to complete the example dataset.

To study effects of group frequency, we choose three days with complete hourly observations (24 observations, days 403, 407 and 694), three with 19 (405, 624, 522) and 13 (396, 453, 509) and two with seven observations (461, 606). Within each day, we take a random sample to get the desired number of observations[1].

We fit the model $\texttt{log(NOx)} \sim 0 + \texttt{day} + \texttt{day:log(NOxEm)} + \texttt{day:sqrt(WS)}$, where `day` is the (categorical) measurement day. This results in 33 coefficients; 11 of these belong to dummy variables. To make the results below independent of contrast choice, the model specification does not include an intercept nor the main effects for the two continuous variables.

It is impossible to fit models with that many dummy (or mixed) variables by using simple random subsampling. The probability that a random subsample is nonsingular (contains at least one observation from each day) is simply too low.

For the M-S estimator proposed by Maronna and Yohai (2000), we allocate the interaction terms to the S part (`lmrob` options `init="M-S"` and `split.type="f"`) and use nonsingular subsampling as well (default). Note that allocating the interaction terms to the M part would leave no variables for the S part, leading to an L1 estimator. The control parameters for the S estimator, see next section, are $r = 1000$, $k = 2$, and $h = 20$.

In Figure 1, we compare the fits for the measurements taken on day 693 for the S and the M-S estimator and two different seeds for the random number generator. The 5 observations in the lower right corner are outliers in the predictor space. Both estimators produce a multitude of solutions when varying the seed. For some seeds the outliers in the predictor space are rejected

[1] We used observations 311, 313, 318:319, 323, 326, 332, 502, 505:506, 508:509, 511:514, 516, 520:522, 669:692, 717, 719, 721:725, 727:730, 732, 734:740, 765:788, 1631, 1633:1636, 1639, 1641:1643, 1645:1646, 1648:1649, 1823, 1825, 1828:1829, 1831:1832, 1836, 2920, 2922, 2924, 2927, 2929:2930, 2932, 2934, 2937:2941, 3184:3186, 3188:3189, 3191, 3193:3194, 3196:3199, 3201:3207, 5573:5575, 5577:5581, 5583:5584, 5586, 5588:5591, 5593:5596, 7177:7200 of the `NOxEmissions` dataset from the R package `robustbase` (Rousseeuw et al, 2015).

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

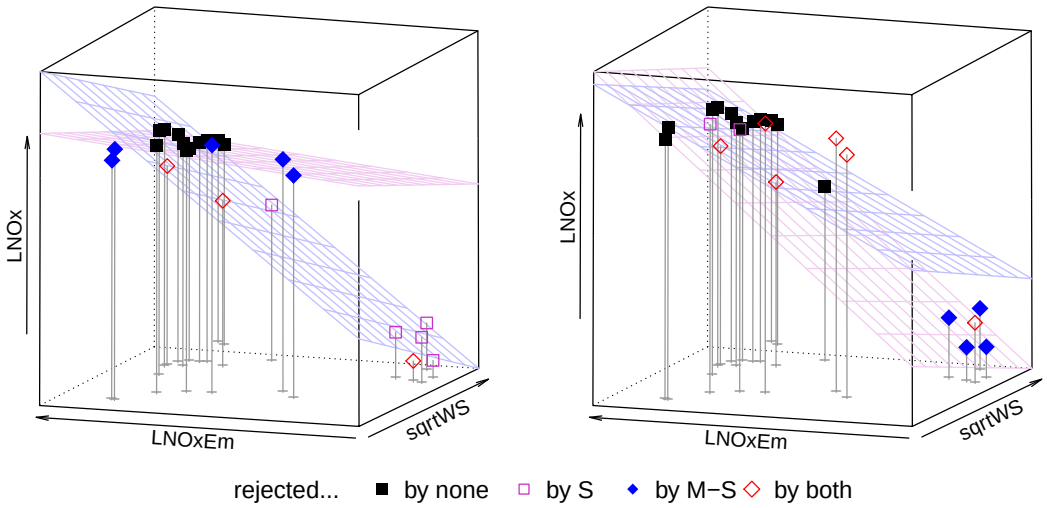

Fig. 1 Comparison of the fits for two different seeds. The 3d-scatterplot shows the original and fitted values of day 693, the planes of the fitted values for the S estimator (magenta) and the M-S estimator (blue). The L1 and Least Squares fits are similar to the the M-S estimator. Observations rejected by each estimator are marked according to the legend.

while for others, they are not. Maronna and Yohai (2000, page 209) already noted "[...] the effect of subsampling to be nontrivial."

The L1 and least squares estimates are similar to the two fits that include the outliers in the predictor space (not shown on the plot).

6 Control parameters of the S step

The established algorithm for calculating an S estimator–a version of Algorithm 1–includes three steps, which give room for three control parameters:

- A number r of nonsingular subsamples of size p is drawn, each of which determines a "candidate" coefficient vector $\boldsymbol{\beta}_{I0}$ through solving the corresponding linear equation (corresponding to step 4 of Algorithm 1). The control parameter r is called `nResample` in the function `lmrob`.
- For each of these subsamples, k steps of iterated Weighted Least Squares are performed to determine an improved candidate $\boldsymbol{\beta}_I$ and a corresponding scale estimate $\widehat{\sigma}_I$ (instead of fully iterating in step 5 of Algorithm 1). k is called `k.fast.s`.
- The h candidates $\boldsymbol{\beta}_I$ with lowest $\widehat{\sigma}_I$ are selected and fully iterated. The candidate with the lowest final $\widehat{\sigma}_I$ is chosen as the resulting S estimate. It determines $\widehat{\boldsymbol{\beta}}$ and $\widehat{\sigma}$. h is called `best.r.s.`.

Since the subsamples are randomly selected, the result is random, too, for any given dataset.

As discussed in Section 4, the S estimate is used as starting point for an M or $\mathrm{M_D}$ estimation algorithm. These algorithms, fully iterated, converge to "fixed points", corresponding to solutions of the defining implicit equations. In the case of multiple fixed points, a criterion is needed for defining the "best" fixed point, see below. The parameter values which, when used as a starting point for the algorithm, lead to a given fixed point, form the "domain of attraction" of this fixed point. Since for the simple M version, the algorithm leaves the scale parameter obtained in the S step untouched, the final fixed points as well as their domains of attraction will still depend on the scale, and therefore on the random subsample. Usually, this randomness will be negligible. The $\mathrm{M_D}$ estimate, on the other hand, gives rise to a well-defined set of fixed points with corresponding domains of attraction.

The goal of selecting algorithmic parameters is thus to obtain a high probability of landing in the domain of attraction of the best fixed point. It is not obvious how to find a suitable definition of "best" in this context. Since we do not want to use any information external to the given sample, we determine the fixed point that is most often selected as the best one.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

For many datasets, there is only one fixed point, or a clear winner among two or three of them. For other datasets, there may be two or more fixed points with comparable frequencies. While the choice of the best may then depend on the exact definition of the quality, it may also be of interest to learn about all of them. As mentioned in the last section, the example leads to two distinct estimates, which differ in their fitted values for day 693, and both of them look plausible.

In order to obtain a few datasets for which the different settings of control parameters lead to different performances, we chose the following pathway: We took the dataset described in Section 5 and applied the SMDM estimator with default settings (`setting = "KS2014", set.seed(1)`). We used the fitted values and the scale estimate ($\sigma = 0.4437$) to generate 100 new datasets as follows.

1. Select three days and 20% of the observations of each of them at random.
2. For the selected days and observations, set both continuous predictors $x_i^{(j)}$ to $\text{mean}(x_\cdot^{(j)}) + 3\,\text{sd}(x_\cdot^{(j)})$.
3. Generate new responses by adding random t_3 distributed errors for the contaminated days and normally distributed errors for the other days. Both distributions were scaled using σ.

We evaluated all the generated datasets for multiple fixed points. We discarded all the datasets that only had one fixed point. This left us with 14 datasets. For only 3 of these, there was a clear winner, whereas for the others, there were two or more solutions with similar quality, and the convergence of the algorithm to one of them did not depend much on the computational effort.

Therefore, the 3 datasets with a clear winner among the possible fixed points were used to examine the dependence of the performance, as measured by the probability to lead to the best fixed point, on the one hand, and the computational effort on the other hand on the control parameters of the S estimating algorithm. We also varied the ψ-function between bisquare and lqq. The tuning constants were chosen to allow for a formal asymptotic breakdown point of 0.5 as usual.

For the combinations of levels of the factors r, k, h, ψ, the SM_D estimator was determined for 100 random seeds each. On the one hand, the computing times for the initial S estimator were averaged over these 100 trials. On the other hand, the fixed points were determined as clusters for which the mean squared differences of the fitted values were all below a given small threshold. (The results were stable for a large range of choices of the threshold in our context.) Then, the number of trials pertaining to the largest cluster was counted to estimate the probability of converging to the desired fixed point.

Table 1 shows, in the first line, the comparison of performance between the lqq and the bisquare ψ-function for a particular setting of control parameters, averaged over the 3 datasets mentioned above. In addition, the performance of the M-S initial estimator is included. The latter was calculated by the respective function in R, which does not allow for manipulating the two algorithm parameters k and h. With the default setting of the number of subsamples, r, the computation time would be much lower than for the S estimator. Since the probability of finding the best solution is also much lower, we increased the number of random subsamples to 40.000 to make the comparison more informative.

In order to gain some more experience, the numbers of observations, n, and of parameters, p, were varied in the following manner:

A. The original datasets had $n = 182$ observations and $p = 33$ coefficients, as described above.
B. For each one of the 14 datasets, we selected half of the observations for each day, resulting in $n = 91$ and the same $p = 33$.
C. We dropped 6 of the 11 days, which resulted in $n = 94$ and $p = 18$.
D. Starting from the last dataset, we doubled the number of observations on each of the remaining 5 days by simulating the additional one as described in Section 7.

The table shows a consistent advantage for the lqq over the bisquare ψ-function. The M-S reaches the best solution with lower probability, even though it was tuned to use a higher computational effort.

Now, we study the tradeoff between achieving a high probability of converging to the desired solution and the computational effort. Figure 2 exhibits the relationship between these two characteristics, indicating the settings of the control parameters by labels, for the three datasets of setup A. Analogous figures for other setups lead to the same conclusions.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

	setup		S				M-S			
			lqq		bisquare		lqq		bisquare	
	n	p	prob	cTime	prob	cTime	prob	cTime	prob	cTime
A	182	33	0.953	4.49	0.877	5.36	0.818	6.62	0.621	6.24
B	91	33	0.847	2.52	0.387	2.31	0.812	4.02	0.494	3.84
C	94	18	1	0.90	1	0.70	0.792	1.91	0.700	1.73
D	188	18	1	1.55	1	1.37	0.369	3.39	0.370	3.00

Table 1 Probabilities of obtaining the best solution and computing times for the lqq and bisquare ψ-function, averaged over the three datasets studied, for four setups described in the text. The computation parameters were $n = 182, p = 33, r = 1000, k = 5, h = 40$ for S and $r = 40.000$ for M-S.

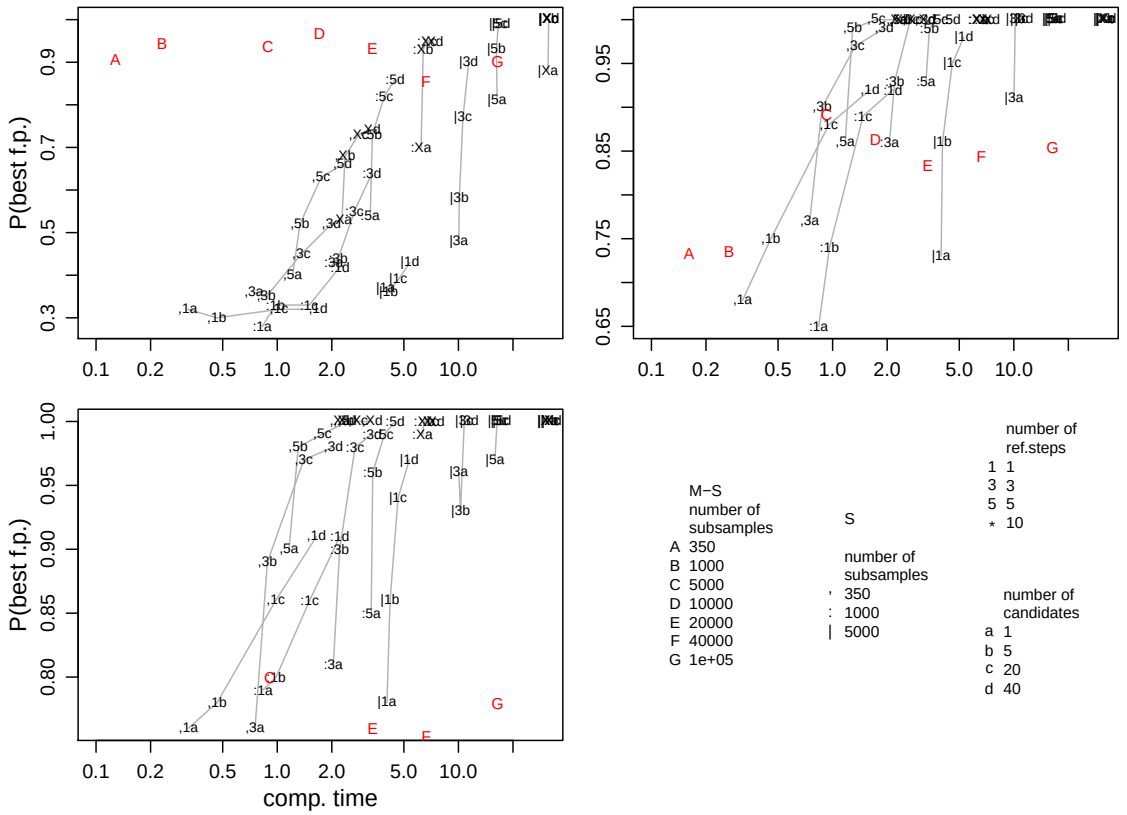

Fig. 2 Probability of converging to the desired solution and computing time for the 3 selected datasets in setup A. The symbols reflect the settings of control parameters as indicated by the lists shown above.

The figure shows that the biggest difference in computational effort comes from the three different numbers of random subsamples, r. The effect of increasing k from 1 to 10 comes next, and the effort of a larger number h of candidates that are fully refined is rather minimal. Of course, the biggest effort, with all 3 control parameters at the highest level, yields the maximum probability of reaching the desired solution. Lowering the tolerated value of this probability, settings with the middle number of resamples, 1000, with high settings of the other parameters beat the strategy of keeping the highest number r and lowering either of the other parameters.

The M-S initial estimator clearly fails in the sense of showing lower probabilities of achieving the desired solution. Increasing the computational effort does not improve the result in these examples. Note that the interactions between factors and continuous regressors overweigh the factor main effects coefficients in our application, and these are attributed to the S part of the M-S scheme.

Based on these results, we recommend to set $r = 1000$, $k = 5$ and $h = 40$ for our setups, and we will use these settings for the simulations in Section 7. For the M-S estimator, the number of random subsamples leading to a similar computational effort is between $r = 20,000$ and $40,000$.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

This study is of course not a sufficient basis to make general recommendations. It would be valuable to collect such results for other setups, but this is beyond the scope of this paper.

For the 11 other datasets, which did not show a clear winner, increasing the control parameters does not improve the stability. As the difference between solutions is very small, the estimators select one of the multiple solutions at random irrespective of computational effort.

7 Efficiency and robustness

So far we have only evaluated the stability of the estimated coefficients but not how well the true values are estimated. We use simulated data to study the quality of the estimates under multiple contamination scenarios.

As the criterion to evaluate the quality of the fits, we use a centered logarithmized Mean Squared Error of the fitted values,

$$U = \log_{10}\left(n^{-1}\sum\nolimits_i (\widehat{y}_i - \mathring{y}_i)^2\right) - \gamma = \log_{10}\left(n^{-1}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\intercal \boldsymbol{X}^\intercal \boldsymbol{X}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})\right) - \gamma\,, \tag{4}$$

where $\mathring{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$ are the model values, $\boldsymbol{\beta}$ is the true parameter vector, and $\gamma = \log_{10}\left(p\sigma^2/n\right)$ is the logarithmized expected value of the Mean Squared Error for the Least Squares fit to the model with normal errors. (To see this, let $\widehat{z} = \boldsymbol{X}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \boldsymbol{H}\boldsymbol{\varepsilon}$, where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal$. Then, $\mathcal{E}(\widehat{z}^\intercal\widehat{z}) = \mathcal{E}(\boldsymbol{\varepsilon}^\intercal\boldsymbol{H}\boldsymbol{\varepsilon}) = \mathcal{E}(\mathrm{trace}(\boldsymbol{H}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\intercal)) = \mathrm{trace}(\boldsymbol{H})\sigma^2 = p\sigma^2$.) Thus, the expected value of the criterion under ideal conditions is zero, and its values are logarithms of inflation factors of mean squared errors for other conditions. Note that this criterion is independent of the parametrization for regression equivariant estimators.

We study three different scenarios with 1000 replicates of each of them:

1. *Efficiency.* Observations are simulated from the estimated model for the example, using normally distributed errors. We used the fitted values from the example and added random errors. For σ, we use the estimated value $\sigma = 0.4437$ of the example, as in the previous section.
2. *Longtailed Errors.* As 1., but errors for days 693, 522, 509 were generated using a t_3 distribution, scaled with σ.
3. *Contaminated covariates.* On each day, 20% of the LNOxEm values were shifted by 5, and independently, 20% of the sqrtWS were increased by 3. The responses were generated using the uncontaminated predictors with errors as in 2.
   The original predictors were also used for evaluating U in form of the second expression in eq. 4. (Note that using the first form naively, one would compare fitted and true values for different regressors, contaminated and uncontaminated, respectively.)

We evaluate the performances of the S and M-S estimators as well as the L_1-estimator, along with their refinements by an M step and an MDM step, and compare them to the OLS estimator. We omit the further refinements by iterating the MD step until convergence, since the changes with respect to the one-step versions shown are minor. For the S initial estimator, the recommended control parameters given in the last section are used. For M-S, we used $r = 40,000$ to make the comparison fair in terms of a similar computational effort.

Figure 3 shows the simulated distributions of the quality criterion U, applied to the whole dataset and to one of the "critical" days, for the three settings. As expected, the refined estimators produce more accurate estimates than the high breakdown initial estimators. The improvement is very remarkable in the cases studied here: The mean squared error criterion 10^U decreases by an order of magnitude. While the S estimator produces better quality than the M-S estimator, the differences disappear for uncontaminated predictors once the first M-step refinement is computed. For contaminated predictors, the estimates based on the S estimator are clearly superior. Further refinement by a DM (or M_D) step is needed after L_1 initialization, but does not produce an improvement any more for S- or M-S-initialization. The DM step after the initial SM or (M-S)M step is nevertheless useful for getting a more appropriate scale estimate, which in turn allows for more precise inference. We have also simulated the M-S estimator and its refinements with the default number of resamples, 1000. It performed equally well for the first two setups, but clearly worse for contaminated covariates.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

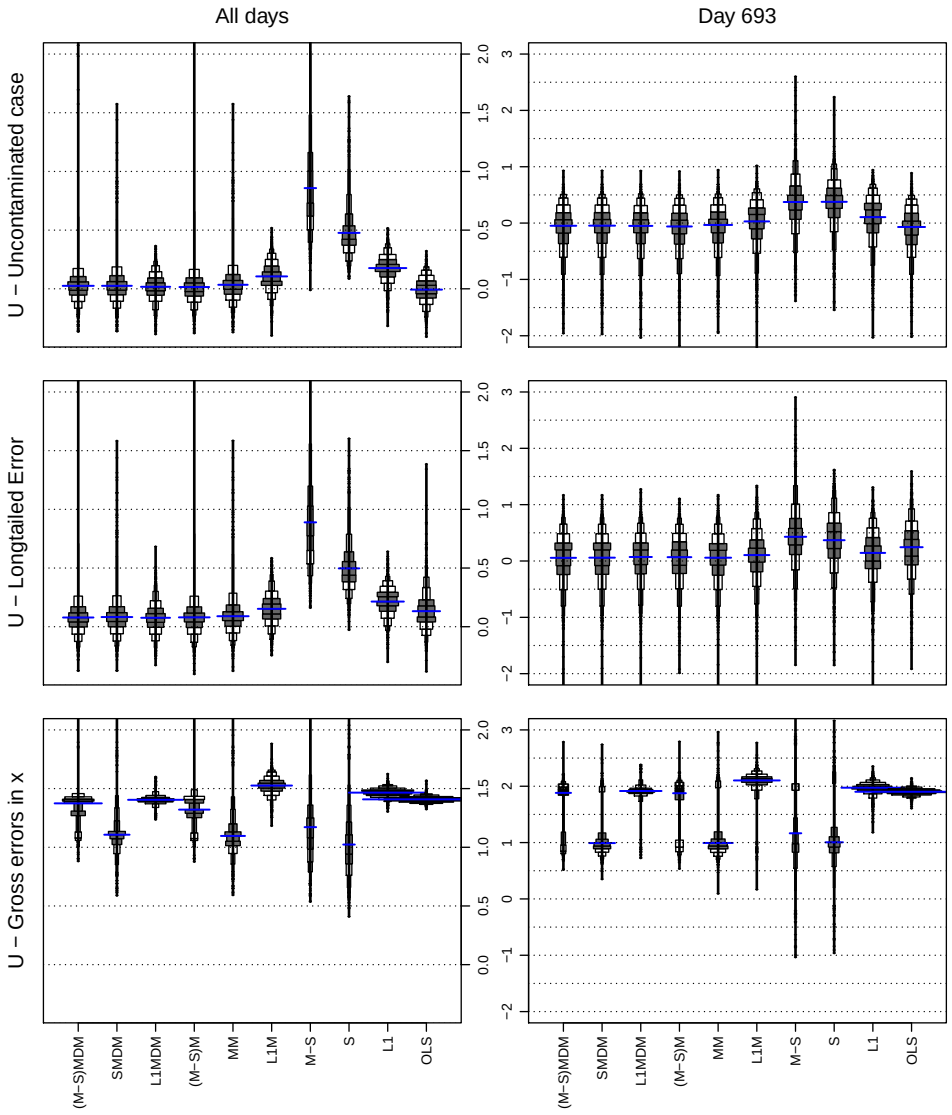

Fig. 3 Multibox-plots of simulated distributions (1000 replicates) of the quality coefficients U for different estimators and the three different scenarios (rows). In the right hand column, U is evaluated only for day 693.

Remark 4. The distributions are displayed by "multibox plots." They combine the idea of the boxplot with a version of a histogram based on variable class widths. The classes are given by quantiles (in this version those related to $p = 0.025, 0.05, 0.125, 0.25, 0.375, 0.5$ and the corresponding $1 - p$ values). The shaded boxes contain the middle half of the data, as does the box in an ordinary box plot.

The Least Squares and $L_!$ estimators show some loss of efficiency for contaminated errors, as expected. They both fail for contaminated regressors in our case, but are in line with the M-S initialized estimators. In fact, high breakdown starting points for refinements are only needed in this last one of our setups.

8 Conclusions

Simple random subsampling fails for models with categorical variables if some of their levels show low frequencies. The established alternative, the M-S estimator, cannot cope well with interactions of continuous and categorical predictors. Nonsingular subsampling is suitable in these situations, and reduces to established simple subsampling in non-problematic cases.

Published in final edited form as: Comput Stat. 2017 June;32(2):631-646. doi:10.1007/s00180-016-0679-x

We have compared the stabilities and the statistical qualities of the resulting predicted values for S and M-S estimators and their refinements to SMDM and (M-S)$\mathrm{M_D}$ estimates. We found a clear advantage for the SMDM over (M-S)$\mathrm{M_D}$ estimate for our case of contaminated covariates, but no difference between these two methods for contamination of the error term.

Remark 5. Our nonsingular subsampling algorithm may also be applied to ordinary bootstrapping (and, even more so, "subsampling" as discussed in Politis et al, 1999). Simple bootstrap sampling may lead to singular bootstrap samples of the design matrix. The problem is much less severe than for computing S estimates, though, since the samples have size n, and, therefore, singular cases are much less frequent. Bootstrap samples can be obtained in two steps. First, generate a bootstrap sample of the indices 1 to n, and summarize it into the frequencies h_i by which each index has been selected. Let $\boldsymbol{S}$ be the indices with non-zero frequency, and N be its length ($N \geq p$). Second, apply nonsingular subsampling of size N to a random permutation of the data and use the resulting observations multiple times, according to $h_{\boldsymbol{S}}$. – Note, however, that it remains doubtful to these authors whether ordinary bootstrapping makes sense for regression problems with categorical variables.

The nonsingular subsampling algorithm is implemented in C in the R package `robustbase` (R Core Team, 2015; Rousseeuw et al, 2015), both as an ordinary function (`Rsubsample` in `system.file("xtraR/subsample-fns.R", package="robustbase")`) and as the default method to get the initial S estimate within the function `lmrob` for robust regression.

Acknowledgments

The authors would like to thank Kali Tal for providing editorial help with the manuscript. A reviewer has provided very helpful suggestions to improve earlier versions of the manuscript.

Compliance with Ethical Standards

Conflict of Interest: The authors declare that they have no conflict of interest.

References

Davies PL, Gather U (2005) Breakdown and groups. The Annals of Statistics 33(3):977–1035

Demmel J (1997) Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics

Golub GH, Van Loan CF (1996) Matrix computations, 3rd edn. The Johns Hopkins University Press, Baltimore

Hampel FR (1975) Beyond location parameters: Robust concepts and methods. Bulletin of the International statistical Institute 46:375–382

Koller M, Stahel WA (2011) Sharpening wald-type inference in robust regression for small samples. Computational Statistics & Data Analysis 55(8):2504–2515, DOI DOI: 10.1016/j.csda.2011.02.014

Maronna RA, Yohai VJ (2000) Robust regression with both continuous and categorical predictors. Journal of Statistical Planning and Inference 89(1–2):197 – 214, DOI 10.1016/S0378-3758(99)00208-6

Maronna RA, Martin RD, Yohai VJ (2006) Robust Statistics, Theory and Methods. Wiley, N.Y.

Mili L, Coakley CW (1996) Robust estimation in structured linear regression. The Annals of Statistics 24(6):2593–2607

Mili L, Phaniraj V, Rousseeuw P (1991) Least median of squares estimation in power systems. Power Systems, IEEE Transactions on 6(2):511–523

Politis DN, Romano JP, Michael W (1999) Subsampling. Springer series in statistics, Springer, NY

R Core Team (2015) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL http://www.R-project.org/, ISBN 3-900051-07-0

Rousseeuw P, Croux C, Todorov V, Ruckstuhl A, Salibian-Barrera M, Verbeke T, Koller M, Maechler M (2015) robustbase: Basic Robust Statistics. URL http://CRAN.R-project.org/package=robustbase, r package version 0.92-5

Ruckstuhl AF (1995) Analysis of the t2 emission spectrum by robust estimation techniques. PhD thesis, Swiss Federal Institute of Technology Zurich

Salibian-Barrera M, Yohai V (2006) A fast algorithm for S-regression estimates. Journal of Computational and Graphical Statistics 15(2):414–427

Stahel WA, Ruckstuhl AF, Senn P, Dressler K (1994) Robust estimation in the analysis of complex molecular spectra. Journal of the American Statistical Association 89(427):788–795