CHARLES UNIVERSITY
FACULTY OF SOCIAL SCIENCES
Institute of Economic Studies

Consistency of S-weighted estimators in panel data models

Master's thesis

Author: Bc. Jan Provazník
Study program: Economics and Finance
Supervisor: RNDr. Michal Červinka, Ph.D.
Year of defense: 2024

Declaration of Authorship

The author hereby declares that he or she compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, July 30, 2024

Jan Provaznik

Abstract

The thesis focuses on the S-weighted estimator and its performance on contaminated data. The first part summarizes the historical background, providing a basic orientation in the field of robust statistics and reviewing the existing literature on S-weighted estimator. In a simulation study performed in Matlab, the estimator's performance is compared with that of LWS and S-estimator. The results show that S-weighted estimator achieves the same efficiency as LWS in lower contamination levels. Contamination exceeding 10% causes significantly higher mean squared error of the S-weighted estimates. The last part of the thesis focuses on developing a simple implementation of the estimator in Python.

JEL Classification	F12, F21, F23, H25, H71, H87
Keywords	consistency, S-weighted estimators, panel data
Title	Consistency of S-weighted estimators in panel data models

Abstrakt

Tato práce se zaměřuje na S-vážený odhad a jeho výkon na kontaminovaných datech. První část práce shnuje historický kontext, poskytuje základní orientaci v oblasti robustní statistiky a sumarizuje existující literaturu o S-váženém odhadu. V simulační studii provedené v jazyce Matlab porovnáváme kvalitu odhadů pořízených pomocí S-váženého odhadu s metodou LWS a S-odhadem. Výsledky ukazují, že v nízkých úrovních kontaminace je účinnost S-váženého odhadu stejná s metodou LWS. Kontaminace vyšší než 10 % způsobuje významně vyšší střední kvadratickou odchylku u S-váženého odhadu. Závěrečná část práce se věnuje jednoduché implementaci S-váženého odhadu v jazyce Python.

Klasifikace JEL	F12, F21, F23, H25, H71, H87
Klíčová slova	consistency, S-weighted estimators, panel data
Název práce	Konzistence S-vážených estimátorů v modelech panelových dat

Acknowledgments

The author is grateful especially to RNDr. Michal Červinka, Ph.D. for his helpful advice and for his infinite patience. The author would also like to express gratitude to prof. Víšek for introduction to the topic and for providing invaluable Matlab codes.

Typeset in LaTeXusing the IES Thesis Template.

Bibliographic Record

Provaznik, Jan: *Consistency of S-weighted estimators in panel data models.* Master's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2024, pages 82. Advisor: RNDr. Michal Červinka, Ph.D.

Contents

List of Tables

List of Figures

Acronyms

OLS Ordinary Least Squares

LTS Least Trimmed Squares

LMT Least Median of Squares

LWS Least Weighted Squares

WLS Weighted Least Squares

EDF Empirical Distribution Function

MSE Mean Squared Error

Master's Thesis Proposal

Author	Bc. Jan Provazník
Supervisor	RNDr. Michal Červinka, Ph.D.
Proposed topic	Consistency of S-weighted estimators in panel data models

Motivation The standard method of estimation of regression coefficients, the ordinary least squares estimation (OLS), is based on minimizing the sum of squared residuals. This means that the farther a given observation is from the regression plane, the more influential it is in determining the result. Thus we say that the OLS method is susceptible to outliers. An outlier may not only be an untypical observation, it can also be a contaminated or otherwise unfit one. The branch of robust statistics has been developed to address these issues. The S-weighted estimator modifies the OLS method in two ways. Firstly, it minimizes the sum of a function of residuals, a function which does not grow to infinity as fast as the quadratic function used in OLS. Thus the outliers do not increase the sum as much as in OLS. Secondly, it uses a weight function designed so as to further diminish the impact of outliers (the largest residuals are assigned the least weight in the sum and vice versa.) The first alteration - different objective function - is inherited from a robust method called S-estimator. The second - the employment of weight function - is inherited from the method of least weighed squares (LWS). Hence the name S-weighted estimator. The literature on S-weighted estimator is limited. Professor Víšek (2019) proved its consistency under heteroscedasticity. But the application of this method to typical problems encountered in econometrics has, to my knowledge, not yet been studied. One of the tasks of the thesis shall therefore be to develop the S-weighted estimator for the study of panel data. The analogue has been done in Víšek (2014) for LWS. Since the fixed and random effects models belong to the standard methods employed in econometric studies, it is useful to develop their robustified versions which can provide reliable estimates even in cases of contaminated data.

Hypotheses

Hypothesis #1: The S-weighted estimator is a weakly consistent estimator of regression coefficients in the fixed effects panel-data framework.

Hypothesis #2: The S-weighted estimator is a weakly consistent estimator of regression coefficients in the random effects panel-data framework.

Hypothesis #3: In a study on simulated, partially contaminated data, the S-weighted estimator gives estimates which are closer to the true parameters than the estimates generated by other robust and non-robust methods.

Methodology The first two hypotheses must be proved mathematically. Similarly as in Víšek (2014), this might be attainable by generalization to the panel-data case of the analogous result proved for the cross-sectional case in Víšek (2019). The third hypothesis will be tested by estimating a panel-data model on simulated data with various levels of contamination. The simulation will follow the Monte Carlo method and will be performed in MATLAB.

Expected Contribution Apart from developing the method and studying the properties of its estimates, which was discussed above in the section on motivation, the thesis shall also contain a chapter summarizing the history and theory of robust statistical methods written in such a way as to be understandable to a student of IES master's program. In the planned simulation study, the S-weighted estimator will be employed together with OLS, LWS and possibly other robust methods, which should shed light on the strengths and weaknesses of each.

Outline

1. Introduction
2. History and theory of robust statistics
3. S-Weighted estimator
4. Consistency of S-Weighted estimator in panel-data models
5. Simulation study
6. Conclusion

Core bibliography

Bramati, M.C. & Croux, C. (2007): "Robust estimators for the fixed effects panel data model." The Econometrics Journal 10: pp. 521-540.

Rousseeuw, P.J. & Leroy, A.M. (1987): "Robust regression and outlier detection." New York: Wiley.

Víšek, J.Á. (2014): "Estimating the Model with Fixed and Random Effects by a Robust Method." Methodology and Computing in Applied Probability. 17. DOI: 10.1007/s11009-014-9432-5.

Víšek, J.Á. (2015): "Representation of the least weighted squares." Advances and Applications in Statistics 47: pp. 91-144.

Víšek J.Á. (2019): "Asymptotics of S-Weighted Estimators." In: Crocetta C. (eds) Theoretical and Applied Statistics. SIS 2015. Springer Proceedings in Mathematics & Statistics, vol 274. Springer, Cham. DOI: 10.1007/978-3-030-05420-5_4

Chapter 1

Introduction

Regression analysis has become the primary tool employed by econometricians, as it allows to quantify the effect of one or more variables on another. The standard method used for the estimation of regression models, the ordinary least squares, has been shown to suffer greatly from the effect of outlying observations. Whether these are natural occurrences of unlikely cases, or the product of contamination in the data, it might be beneficial to use a method which, sacrificing some efficiency, produces results that are unaffected by solitary influential observations.

For this purpose, statisticians have devised various estimators, but most of them suffered from significant problems. Various robust estimation methods have been devised by statisticians. The M-estimators proposed by Huber (1964) have not achieved great popularity due to their lack of regression equivariance which necessitated standardization of residuals by a robust estimator of spread. On top of that, they also turned out to be immune only up to $100/p\%$ of contamination, p being the dimension of the model. Siegel (1982) discovered the first method robust to 50% contamination, the repeated median of squares, but its calculation was not feasible for any reasonable sample sizes. The least median of squares and the least trimmed squares methods proposed by Rousseeuw (1984) achieved robustness to up to 50% contamination, but their estimates turned out to be unstable with respect to very small shifts of individual observations. These problems have largely been overcome by the least weighted squares estimator developed by Víšek (2000) and by the S-estimator developed by Rousseeuw (1984).

This thesis focuses on a recently developed method, the S-weighted estimator, which was proposed by Víšek (2015) as a combination and generalization

of least weighted squares and S-estimator. We summarize the very limited existing literature on this topic, and study the behavior of S-weighted estimator on simulated data in a range of situations, comparing its results with older estimators. Specifically, we evaluate the performance of S-weighted estimator and compare it to its two predecessors, the method of least weighted squares and the S-estimator. We find that the S-weighted estimator inherits from least weighted squares the ability to utilize good leverage points and acquire better estimates than S-estimator, however, in higher contamination levels, the S-weighted estimator's performance does not equal that of least weighted squares. Since the literature contains very little information on how the estimator is to be calculated, we discuss in some detail the algorithm by Boček & Lachout (1995), modified by Víšek (2016b). As there is presently no publicly available implementation of this particular algorithm, we offer our implementation in Python. Note that this content differs from the original intentions described in the proposal. The complexity of the proof of S-weighted estimator's consistency in panel data models turned out to be an insurmountable obstacle while promising very little use, as the consistency in standard cross-sectional case has already been proved.

The rest of the thesis is organized as follows. In Chapter 2 we study the sources and types of contamination in data and provide some basic terminology. Also in the same chapter, we summarize the development of robust statistical methods of estimating regression models, provide basic orientation in the field and explain some mathematical concepts that robust statistics utilizes and that will also be utilized in our study of S-weighted estimator. Chapter 3 reviews the existing literature on S-weighted estimator and explains in some detail the logic of its definition and functioning. The algorithm for the estimator's calculation is described. Chapter 4 presents the methodology and results of our simulation study. In Chapter 5 we present our implementation of the S-weighted estimator in Python, and Chapter 6 concludes.

Chapter 2

Overview of robust statistics

2.1 Contamination

Before we dive into the discussion of various methods of robust statistics, we need to spend some time on the topic of contamination. We will give a specific meaning to the term and briefly review the causes of contamination that have been identified by statisticians. Then we will define those types of contamination which are the most destructive in terms of their effects on regression, and look at some simple cases that illustrate those effects. That will provide a motivation to the subject of the thesis.

In the broadest sense, then, we could perhaps define contamination as any deviation of the collected data from the true quantities that have been measured. From this, we can immediately see that any dataset is necessarily littered with contamination. No quantity is ever measured with infinite exactitude and no data gathering process is impervious to error. Not every minor shift of value, however, has the potential to affect the results of whatever analysis we may choose to perform on the data. In what follows, we shall reference the discussion of Hampel *et al.* (1986), which classifies contamination into four main types and assesses the harm they may potentially cause. These types are: the occurrence of gross errors, rounding and grouping, approximate model, and approximately fulfilled independence assumption.

Gross errors are the cause of outliers and leverage points and thus constitute the most dangerous kind of contamination. They are consequences of wrong copying, typos, computation errors and many other possible blunders. Another source of outliers is the situation, which can also possibly occur, that part (presumably a small part) of the data follows a different distribution. The

presence of gross errors, especially in data collected manually, is not uncommon. The observation contaminated by gross error can have a completely random value and thus can possibly influence the results significantly, it can even spoil the analysis entirely. A typical example of a magnitude-changing type of gross error is a wrongly placed decimal point. Hampel *et al.* (1986) provide an extended passage on the causes and frequency of gross errors. They distinguish between high-quality data with almost no gross errors, or with no gross errors at all (which is very rare, but instances are known,) and routine data which typically contains 1-10% of gross errors, but can be much worse. Let us note for completeness that outliers do not of course have to be an instance of gross error. They can occur naturally by virtue of the data having a long-tail distribution. That is a case of approximate model and is treated below. As will be seen, robust statistical methods can deal with outliers quite well.

Rounding is considered a mild deviation from the model. It is however present in all data. Since the precision in any dataset is always finite, we are in fact always dealing with discrete data. Usually the data are rounded, sometimes grouped or otherwise distorted. Sometimes the data are infected by a small but systematic inaccuracies in measurement. Although the effects of these factors are generally considered harmless, Hampel *et al.* (1986) state that the mild deviations from normality caused by rounding and other systematic inaccuracies can cause up to 30% loss of efficiency. As we will see in Section 2.3, small shifts of data can cause surprisingly large shifts of estimates by some robust methods, and thus are not as harmless as might seem. The effect of contamination of this class can be studied by means of local-shift sensitivity, a property of influence function which will also be discussed later.

Sometimes the model is conceived as approximate, for instance when the statistician ignores the normality assumption, relying on central limit theorem. Even high-quality data often follow distribution which is longer-tailed than the normal distribution. It can also happen that the distribution is shorter-tailed than normal, although it is much less frequent. This may be the case when the data have been artificially cleaned of all outliers prior to the statistical analysis. With the normality assumption being broken or only approximately fulfilled, the statistical inference can be invalid. Standard errors of the estimates can turn out to be much larger than they previously seemed to be.

This last applies also to the fourth kind of contamination, namely the breach of independence assumption or unexpected serial correlation. As the title suggests, this refers to the situation where the sample cannot be trusted to contain

independent and identically distributed observations, or, in case of time series data, when a variable is correlated with its delayed version. Since this thesis is focused on the robust estimation of regression parameters, not on the robustification of test statistics, we will refer the reader to said publication for further details.

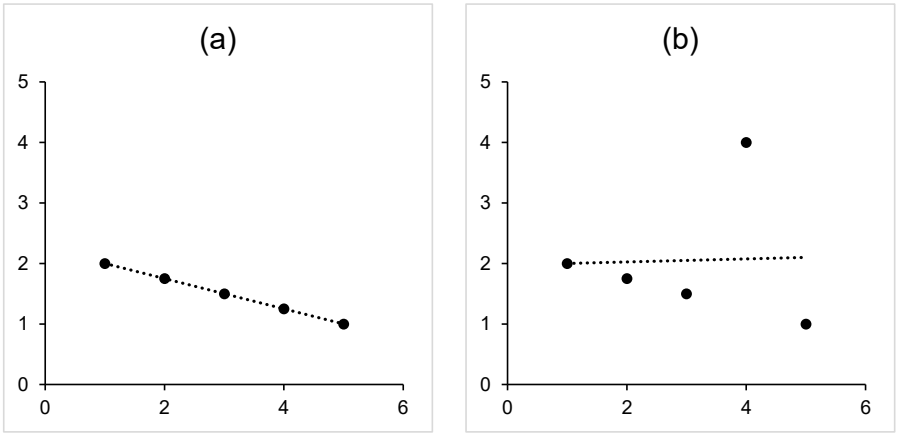
Having appreciated the problems that contamination can cause, we will now turn our attention to the types of contamination that will concern us in the present thesis, namely outliers and leverage points. We will not use the term gross errors, since the origin of outliers and leverage points in the data is not our concern and we do not necessarily want to label them as false. Perhaps they are just unlikely cases, unique cases, or maybe the dataset contains two different populations. All we want to achieve in robust statistics is to define methods whose results are based on the main part of the data and not on some one influential observation.

The term outlier is given a rather specific meaning in the context of regression. By an outlier, we will understand an observation whose value of the explained variable is far from the rest of the data (in other words, in the tail), while the values of the explanatory variables are close to the bulk of the data. On the other hand, by leverage point, we will understand that observation whose explanatory variable has an unusual value, regardless of the value of the explained variable. Another terminology, which will not be used here, but can be preferred by some, is "outlier in the y-direction" and "outlier in the x-direction." Importantly, two types of leverage points are distinguished according to the value of the explained variable. If the explained variable follows the same model as the main bulk of the data, i.e. if the observation is close to the regression line obtained from the uncontaminated part of the data, then we are dealing with a good leverage point. If the observation is far from the regression line, then such a leverage point is called a bad leverage point. For our purposes, this informal definition of outliers and leverage points will suffice, as the simulation studies in previous literature never required a more rigorous definition either. In our simulation, we will study various types of contamination, i.e. the specific combination of outliers and leverage points, and each type of contamination will be studied on multiple levels of contamination, which we will define as the number of bad leverage points and outliers in the data divided by the sample size. Note that the good leverage points are not included, as they are not considered cases of contamination. On the contrary, it will be to the method's credit if it is able to disregard the outliers and bad

leverage points while at the same time utilizing the information contained in the good leverage points. We will see that this is indeed possible.

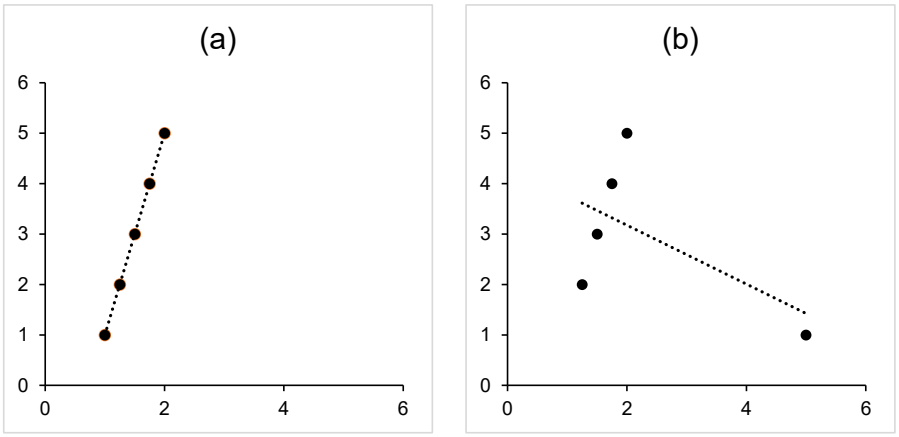
Let us now take a closer look at the effects of outliers and leverage points on simple regression in very primitive, theoretical examples based on Víšek (2000). Figure 2.1 depicts the effect of an outlier. In part (a) we see the default situation which is altered in part (b) by moving one observation along the y-axis. We can see that the regression line has moved significantly, so much in fact that the slope estimate now seems to be positive rather than negative.

Figure 2.1: The effect of an outlier



Source: author's computations (based on Víšek (2000))

Figure 2.2: The effect of a bad leverage point



Source: author's computations (based on Víšek (2000))

In Figure 2.2 we see similar effect of a bad leverage point. One observation has been moved alongside the direction of the x-axis, which has changed the

slope of the regression line in such a way that it is almost perpendicular to the original one. In simple regression it is possible to discover influential observations graphically as we just have. In multiple regression, however, this becomes much more difficult and robust statistical methods have to be employed.

2.2 Early robust methods

Attempts to deal with contamination in data are probably as old as statistics itself. One informal method we have already mentioned is the exclusion of suspicious outlying values. Using median instead of arithmetic average as an estimator of expected value is also a simple robust method because median is robust to outliers. For discrete or grouped data, mode can also be a robust statistic, see Hampel *et al.* (1986). According to Bernoulli (1777) the rejection of outliers was already a common action among astronomers of his time. In the nineteenth century, this practice started to be formalized by the first objective rules of rejection, see for instance Peirce (1852) or Chauvenet (1863). On the other hand, Student (1927) proposed the opposite, namely addition of new observations obtained as arithmetic averages of two observations which are, according to certain rules, considered too far apart. Used together with rejection of outliers, this method is claimed to have had practical results, although it seems to not have received further attention in literature.

An early paper written by Fisher (1920) compares the efficiency of two different estimators of variance of normally distributed data: the standard deviation and the mean deviation. He concludes that standard deviation is in large samples more efficient than mean deviation. Later, Fisher (1922) addressed the issue of efficiency again, this time in the presence of contamination in data. He tested the efficiency of the method of moments in the system of Pearsonian curves and concluded that high efficiency is achieved only on a small neighborhood of the normal distribution with great losses of efficiency caused by deviation from this point.

Pursuing further the line of thought initiated by Fisher, the first great pioneer of robust statistics was John Wilder Tukey, who is credited by Hampel *et al.* (1986) to have made robust statistics a matter of general interest of statisticians rather than a collection of isolated attempts. His work demonstrates that even small deviations from standard assumptions can have surprisingly sizable effects. Similarly to Fisher, Tukey (1960) considered a small and highly specific deviation from a perfectly normally distributed model. Let us have a

small $\varepsilon \in (0, 0.5)$ which will denote the fraction of data that is contaminated. Tukeys data then have the following distribution function

$$F(x) = \varepsilon\Phi\left(\frac{x}{3}\right) + (1-\varepsilon)\Phi(x),$$

where Φ is the cumulative distribution function of the standard normal distribution, i.e. the distribution function is now a linear combination of the standard normal distribution function Φ (for the "good" observations) and another normal distribution function with its standard deviation diminished. Tukey showed that in this case of contamination, values of ε as small as $\varepsilon = 0.0018$ (sic!) lead to the standard deviation being an asymptotically less efficient estimator than the mean deviation, the situation being the converse in the normal case, where standard deviation is the optimum, as we mentioned above with Fisher (1920).

Before we move now to the introduction of the robust methods of estimation, let us, as a starting point, recall the classical method of estimating regression coefficients, namely OLS. This will be useful later as the rubustifying alterations will be more apparent against this background. We will be considering the standard linear regression model

$$y = X\beta + u,$$

$y \in R^n$ being the explained variable, $X \in R^{n\times p}$ the matrix of explanatory variables, $\beta \in R^p$ the vector of unknown population parameters, and $u \in R^n$ the vector of unobserved factors. In this regression setup, we define the OLS estimator in the following way.

Definition 2.1 (The OLS estimator). The OLS estimator is defined as

$$\hat{\beta}_{OLS} = \underset{\beta}{\operatorname{argmin}}(y - X\beta)'(y - X\beta).$$

Because we will be working extensively with residuals later in the thesis, it will be useful to rewrite the above definition in terms of them. By the i-th residual we will understand $r_i(\beta) = y_i - X_i\beta$. The OLS estimator would then be defined as

$$\hat{\beta}_{OLS} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} r_i^2(\beta).$$

Now we may continue in our historical survey. Under the influence of Tukey,

researchers started to develop new robust methods and robust statistics became a general research area. Three approaches have been particularly significant. Two of them developed by Huber, of which we will briefly summarize one here, and the third, developed by Hampel, which has the most relevance to our subject, and thus will be treated in greater detail in a separate section.

The basis for the first comprehensive robust theory was laid down by Huber (1964) in the form of what is called Huber's minimax approach. Huber generalized Tukey's idea of a mixture of two normal distributions. In his "gross-error model," he allows the contamination to have arbitrary distribution, not only normal. Thus, the overall distribution of the data is

$$F(x) = \epsilon H(x) + (1 - \epsilon)G(x),$$

where G is the known distribution function of the healthy data and H is the unknown distribution of the contaminated data. The essence of Huber's minimax approach lies in utilizing game theory framework to optimize the worst possible outcome. He uses the form of a two-person zero-sum game: reality chooses the distribution function F and the researcher chooses the penalizing function γ. The prize for nature is the asymptotic variance $V(\gamma, F)$ which the researcher is intent upon minimizing. Under mild assumptions, Huber shows the existence of an equilibrium in this game which is composed of a distribution called Huber's least favorable distribution and an estimator called Huber-estimator. A combination of two functions, Huber's least favorable distribution is exponential on the tails and normal in the middle.

Huber (1964) also introduced the so-called M-estimator which is a generalization of the maximum likelihood estimator, a standard non-robust estimation method. Let us define each in turn, so that the differences are apparent.

Definition 2.2 (The Maximum Likelihood Estimator). The Maximum Likelihood Estimator is defined as

$$\hat{\beta}_{ML} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} -\log f\left(r_i(\beta)\right).$$

Definition 2.3 (The M-Estimator). The M-estimator is defined as

$$\hat{\beta}_{M} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \rho\left(r_i(\beta)\right).$$

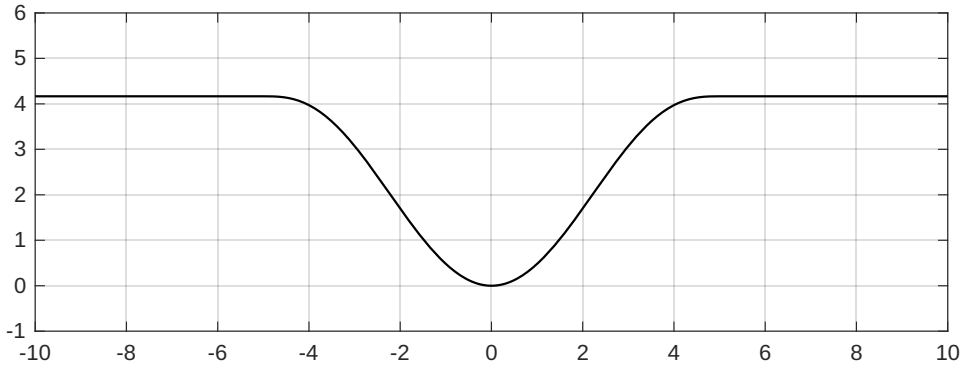
Instead of a logarithm of the density function, the M-estimator uses a conve-

nient non-constant function ρ which allows to be selected from various options. Thus the M-estimator is actually a class of estimators which follow the same principle but vary according to the selection of the ρ function. Notice that if we put $\rho(x) = x^2$, we obtain the ordinary least squares estimator. The M-estimators can therefore also be understood as a generalization thereof. Huber derived the properties of the M-estimators such as consistency and asymptotic normality, but unfortunately these estimators do not have the property of scale and regression equivariance. Let us illustrate this problem on a specifically selected function ρ, one which we will work with later. Suppose ρ was defined as

$$\rho(x) = \begin{cases} \frac{x^2}{2} - \frac{x^4}{2c^2} + \frac{x^6}{6c^4} & \text{for } |x| \leq c \\ \frac{c^2}{6} & \text{for } |x| \geq c. \end{cases}$$

In Figure 2.3 we can see the plot of such a function for $c = 5$.

Figure 2.3: Example of ρ function



Source: author's computations.

Now suppose we were to multiply every value in every observation both in the explanatory and in the explained variable by 10. The proportions between the variables would remain the same, so we would expect the estimates of the regression coefficients to remain the same. But since the residuals would also be proportionately increased, they would be "weighted" differently by the ρ function, as more of them would now find themselves under the constant part of it. This means that some residuals would be given less importance relative to others then they were given in the original situation, and thus the results of the estimation would now probably be different. This is of course not intended as a proof, but merely as an intuitive illustration of what scale equivariance means.

The lack of scale and regression equivariance can be solved by standardizing the residuals. Specifically,

$$\hat{\beta}_M = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \rho\left(\frac{r_i(\beta)}{S_n}\right),$$

where S_n is some estimator of scale, such as standard deviation or, preferably, some robust equivalent of it. This, however, is a tedious process and became one of the reasons why M-estimators are not commonly used. We will see that more recent robust methods, including the S-weighted estimator, do this implicitly and therefore are scale and regression equivariant.

The second approach of Huber (1973) is based on robustified likelihood ratio tests. In this approach, instead of probabilities, Huber uses more general set functions called Choquet capacities. This can be used for the computation of robust confidence intervals and robust point estimates of location. By this approach Huber derived solutions which are exact, not asymptotic, for every finite sample size. According to Hampel *et al.* (1986), Huber's second approach uses an elegant mathematical theory and gained respect of mathematicians for the field of robust statistics, but in practice it has not been used very often due to its limited applicability and certain complications it poses in estimation, such as the fact that it turned from simple to composite parametric hypotheses. Therefore, and also because of its difficult mathematical theory, we will not treat Huber's second approach in any greater detail here.

2.3 Standard robust methods

A new approach was introduced by Hampel (1968), and later developed and generalized by Ronchetti (1982), Rousseeuw (1984) and others and turned out to be the future mainstream of robust statistics. It is based on three important concepts: qualitative robustness, breakdown point, and influence function. The first two are important but not unique to Hampel's approach, therefore we will focus in our presentation on the third, the influence function, which is a central concept of Hampel's approach.

Definition 2.4 (Influence function). Let Δ_x and F be distribution functions defined on probability space $(\Omega, \mathcal{A}, P)$. Then the influence function IF of the

functional T at function F is defined as

$$IF(x, T, F) = \lim_{t \searrow 0} \frac{T\left((1-t)F + t\Delta_x\right) - T(F)}{t}$$

in those $x \in \Omega$ where the limit exists.

In this definition, function F represents the distribution of the uncontaminated data, functional T is some estimator whose robustness we investigate and Δ_x is a probability measure which puts mass 1 at point x. One can understand Δ_x as distribution of a random variable whose value is equal to x with probability 1. The first term in the numerator, then, stands for the estimate we obtain when fraction t of data is contaminated by values equaling x, and the numerator as a whole is the difference in the estimate such contamination causes. Now we can notice that the whole definition resembles the definition of a derivative of real function. The influence function, therefore, measures the sensitivity of the estimator T to an infinitesimal amount of contamination by values equal to x, i.e. the asymptotic bias caused by data contamination. Using the influence function, we can define three other important terms that serve as measures of robustness of estimators.

Definition 2.5 (Gross-error sensitivity). The gross error sensitivity of the functional T at function F is defined as

$$y^* = \sup_x |IF(x, T, F)|$$

where the supremum is taken over the set of all $x \in \Omega$ where the influence function exists.

The gross-error sensitivity is a measure of the worst influence that a fixed amount of contamination can have on the estimator. Thus it can be understood as an upper bound on the asymptotic bias that the estimator has. If the gross-error sensitivity of an estimator T is finite, we say that T is B-robust, B standing for bias. Typically there is a trade-off between B-robustness and efficiency and therefore the goal is to find the optimal B-robust estimator. We have mentioned earlier that the effects of the second type of contamination, rounding and grouping, can be gauged by means of local-shift sensitivity.

Definition 2.6 (Local-shift sensitivity). The local-shift sensitivity of the functional

T at function F is defined as

$$\lambda^* = \sup_{x \neq y} \frac{|IF(y, T, F) - IF(x, T, F)|}{|y - x|}.$$

Local-shift sensitivity, as seen above, is defined as the supremum of the slope of influence function, and thus it measures the worst (i.e. the largest) effect a small change of an observation from x to y can have. Thus it measures the effect of small fluctuations in the data which can be caused by rounding or grouping and other small inaccuracies. As we mentioned in the previous section, no data actually comes from a continuous distribution and therefore some rounding always takes place.

Definition 2.7 (Rejection point). Let F be the distribution function of a distribution symmetric around 0. Then the rejection point is defined as

$$\rho^* = \inf\{r > 0;\ IF(x, T, F) = 0\, when\, |x| > r\}.$$

We mentioned at the beginning of this chapter the old robust method of rejecting outliers at the outset. The rejection point is intended as a mathematically exact and objective substitute for this method. The idea is that robust estimators should be constructed in such a way that beyond certain point r, the influence function is equal to zero. Observations which are beyond this boundary are rejected entirely and have no influence on the estimates. Therefore it is desirable for the estimator T to have a finite rejection point. With non-robust estimators, the exact opposite is the case. The arithmetic average, for instance, is affected the more, the further the contamination occurs from the bulk of the data, and therefore its influence function does not converge to zero, but rather grows beyond any bounds.

Central to the field of robust statistics has become the concept of breakdown point. This term was coined by Hodges (1967) and later developed by Hampel. Here we will use its definition as provided by Hampel *et al.* (1986).

Definition 2.8 (Breakdown point). The finite-sample breakdown point ε^* of the estimator T_n at the sample $(x_1, ..., x_n)$ is defined as

$$\varepsilon_n^*(T_n; x_1, ..., x_n) = \frac{1}{n} \max\{m; \max_{i_1,...,i_m} \sup_{y_1,...,y_m} |T_n(z_1, ..., z_n)| < \infty\}$$

where sample $(z_1, ..., z_n)$ is made from sample $(x_1, ..., x_n)$ by replacing m observations by arbitrary data points $y_1, ..., y_m$.

The value m in the definition above can be understood as the number of contaminated data points. The asymptotic version of breakdown point is defined generally in Hampel *et al.* (1986). Here we will treat it only as the limit of the finite-sample breakdown point, since in many situations it can be obtained that way.

$$\varepsilon^* = \lim_{n\to\infty} \varepsilon_n^*.$$

The breakdown point is the smallest fraction of observations which need to be contaminated in order for the estimator to become unbounded, which means completely unreliable. In the case of OLS, one observation is sufficient for this, therefore the finite-sample breakdown point of OLS is $1/n$ and the asymptotic breakdown point is zero (as the limit of $1/n$ for $n \to \infty$ is 0.) The finite-sample breakdown point of median (as an estimator of expected value) is 1/2. Generally, the breakdown point takes on values between 0 and 1. The maximal possible breakdown point, however, is 1/2, as for levels of contamination higher than 50% there is no justification for calling the minority of the data correct and the majority contaminated,. For further discussion, see Rousseeuw & Leroy (1987). As a measure of the global reliability of an estimator, the breakdown point is one of the most important robustness measures and from now on, as we inspect various robust estimators, we will always report their breakdown points.

The first method with 50% breakdown point was introduced by Siegel (1982), in the form of the repeated median of squares estimator. It has desirable properties and it can be computed, but as it requires the computation of a parameter vector for every subset of p observations, it becomes unfeasible for larger datasets. Thus this method, although important theoretically, because it showed that an estimator with 50% breakdown point is possible, could not be employed in practice.

The later methods became standard equipment of robust statisticians. They are the least median of squares estimator and the least trimmed squares estimator. Both were first introduced by Rousseeuw (1984), the latter was only mentioned and formally it was developed later by Rousseeuw & Leroy (1987). We will now briefly discuss each.

The least median of squares was originally defined literally as the name suggests:

$$\hat{\beta}_{LMS} = \underset{\beta}{\operatorname{argmin}} \operatorname{med}(r_i^2(\beta)).$$

Later, however, it was redefined in a slightly generalizing way, which we will follow here, as it has become the standard. Let us denote by $r_{(i)}^2(\beta)$ the order statistics of the i-th squared residual, so that $r_{(1)}^2(\beta) \leq r_{(2)}^2(\beta) \leq \cdots \leq r_{(n)}^2(\beta)$. Now the least median of squares estimator is defined in the following way.

Definition 2.9 (The least median of squares). The least median of squares estimator (LMS) is defined as

$$\hat{\beta}_{LMS,n,h} = \underset{\beta}{\operatorname{argmin}} r_{(h)}^2(\beta),$$

where $h \in N$, $\frac{n}{2} \leq h \leq n$.

This definition therefore does not use the median of squares, but some other squared residual larger than, but close enough to the median one, as the value which is to be minimized. The asymptotic breakdown point of LMS is 50% and it is scale and regression equivariant without the need of standardizing the residuals. The optimal value of h for which the breakdown point reaches its maximum for finite sample can be calculated as $h = n/2 + (p+1)/2$ where p is the dimension of the model.

The main problem with this estimator is its slow rate of convergence with respect to the sample size. Since it is only $\sqrt[3]{n}$-consistent, it is less efficient than $\sqrt{n}$-consistent estimators and thus requires comparatively larger samples in order to provide strong estimates. The other method, first mentioned together with LMS by Rousseeuw (1984), but mathematically developed only later by Rousseeuw & Leroy (1987), is based on rejecting those observations which yield the largest residuals.

Definition 2.10 (The least trimmed squares). The least trimmed squares estimator (LTS) is defined as

$$\hat{\beta}_{LTS,n,h} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{h} r_{(i)}^2(\beta),$$

where $h \in N$, $\frac{n}{2} \leq h \leq n$.

It is easy to see that the least trimmed squares estimator is equivalent to OLS computed on those h observations that have the smallest residuals. Just as the LMS, LTS estimator is scale and regression equivariant. The main advantage over LMS consists in its faster rate of convergence and therefore higher efficiency than the LMS has. Rousseeuw & Leroy (1987) proved that the optimal value of h is, as with LMS, $h = n/2 + [(p+1)/2]$. Under this selection, the estimator attains the same finite-sample breakdown point as LMS, namely $(((n-p)/2)+1)/n$, which for large sample sizes converges to 1/2. Therefore the asymptotic breakdown point of LTS is 50%.

We shall now pay some attention to the way LTS estimates can practically be obtained, since the same means will be employed for obtaining the results of simulation study in this thesis. Rousseeuw & Leroy (1987) used an algorithm implemented as a program called PROGRESS. This algorithm randomly selects p observations (p being the number of explanatory variables, which means that p observations in general position define a regression plane exactly), fits a model through them and calculates the sum $S = \sum_{i=1}^{h} r^2_{(i)}(\beta)$ obtained from that model. This is repeated until there is sufficient probability that at least one uncontaminated p-tuple of observations has been selected. Then the model resulting in the smallest S is returned as the solution.

Without going into technicalities, it is obvious that this algorithm can only be used for small sample sizes. Larger n would result in too many possible subsamples of size p and it would take the algorithm too long to investigate sufficiently many of them. An improvement was presented by Boček & Lachout (1995). Their algorithm, with straightforward modifications, can also be used for calculation of LWS and S-weighted estimator, therefore we will now present it in some detail. We will not use the original notation of Boček & Lachout (1995) as it introduces too many terms that would have to be explained despite serving no purpose here. Rather, we will follow the algorithm as presented by Víšek (2000), whose formulation is much more intuitive.

1. *Select p observations at random and fit a regression plane through them.*
2. *Calculate the residuals from this regression for all observations.*
3. *Select h points with the smallest squared residuals and save the sum of these residuals.*
4. *If the sum is smaller than the sum obtained previously, go to step 5. Otherwise go to step 6.*

5. *Estimate a new regression plane by OLS on the h observations and go to step 2.*

6. *If the same model has been found for q times or the algorithm has already gone through r repetitions, end the algorithm. Otherwise go to step 1.*

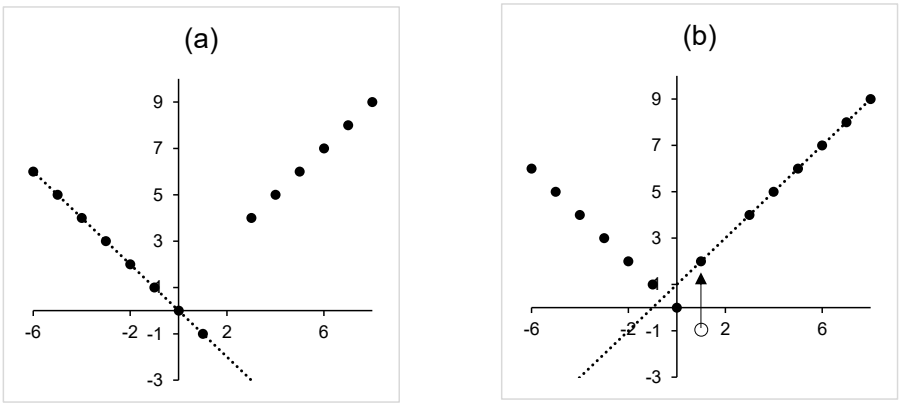
This is obviously not a rigorous description of an algorithm, but it is sufficient to allow one to implement the procedure in a programming language. We shall make a few clarifying comments on some of the steps. Beginning with step 1, p is the dimension of the model. If a model with intercept is considered, then it is included in p and there are $p-1$ explanatory variables. That means that p observations determine a regression plane exactly (with no residuals). Looking at step 3, the idea of the algorithm is, then, to select h observations more or less at random, much like Rousseeuw & Leroy (1987) did in PROGRESS, and then to iterate a few times to slightly better selections of h observations, until the iterations stop bringing improvement. Once the algorithm reaches the same best model for q different random selections according to step 1, the program stops. Values around $q = 100$ seem to have been found sufficient. If there is no model that the algorithm repeatedly finds itself converging to, the procedure is stopped after r repetitions of the outer cycle. The same algorithm, with straightforward changes, was used by Víšek (2012) for the calculation of LWS, and later also for the S-weighted estimator. We shall discuss this in more detail in later sections.

The main problem with LMS and LTS is that their estimates can be very different. This is especially concerning bearing in mind that these methods are supposed to be robust. The cause of their variation is their high local-shift sensitivity. As illustrated in Figure 2.4 which is based on Víšek (2000), they may take into consideration only very narrow majority of observations, and then the shift in one point (depicted in part (b) of Figure 2.4) can cause the regression line to move dramatically, so much as to be orthogonal to the original. Therefore the robust statisticians continued to search for new methods that would not suffer from this issue while preserving the efficiency and high breakdown point of LTS.

2.4 Least Weighted Squares

The least weighted squares estimator, designed to overcome the issue of high local-shift sensitivity of LMS and LTS, was introduced by Víšek (2000). We

Figure 2.4: Local-shift sensitivity of high breakdown-point estimators



Source: author's computations.

shall begin with the definition and then we will follow Víšek (2012) in an overview of the LWS estimator's properties.

Definition 2.11 (The least weighted squares). The least weighted squares estimator (LWS) is defined as

$$\hat{\beta}_{LWS,n,w} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} w\left(\frac{i-1}{n}\right) r^2_{(i)}(\beta),$$

where $w(i)$ is a weight function.

Definition 2.12 (Weight function). The weight $w : [0,1] \to [0,1]$ is a continuous, non-increasing function such that $w(0) = 1$. Moreover, w is Lipschitz in absolute value, i.e. there is $L \in R$ such that for any $u_1, u_2 \in [0,1]$ we have $|w(u_1) - w(u_2)| \leq L|u_1 - u_2|$.

The last condition in 2.12 requires that the slope of w be bounded, that is, that w is sufficiently smooth in this sense. It is easy to see that LTS is a special case of LWS. We just select $h \in (0.5, 1)$ and put

$$w(x) = \begin{cases} 1 & \text{for } x \in [0,h] \\ 0 & \text{for } x \in (h,1]. \end{cases}$$

Similarly, we can obtain OLS by putting $w(x) = 1, x \in [0,1]$ and LMS by $w(x) = 1$ for $x \in (h - \epsilon, h + \epsilon)$ for some $\epsilon < 1/n$, and 0 otherwise. Of course then w would not be Lipschitz, so the theorems proven about LWS cannot automatically be extended to OLS, LMS or LTS.

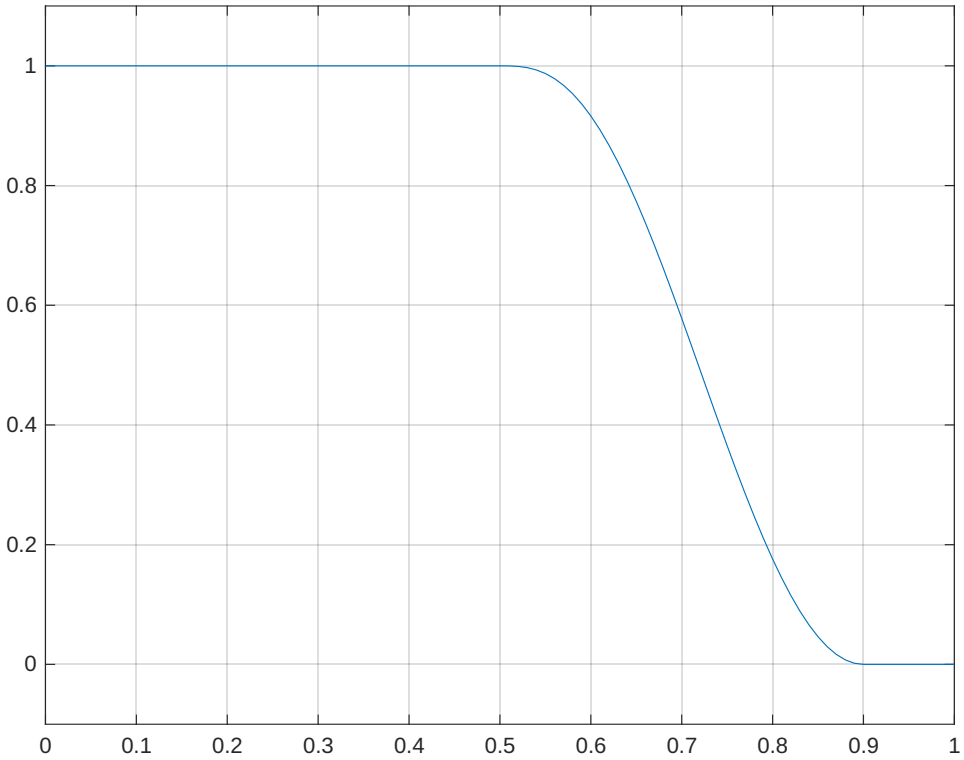
The LWS estimator is, in essence, the LTS with the weight function gradually decreasing instead of being zero-one. Thus it inherits several properties from LTS. It is $\sqrt{n}$-consistent and scale and affine equivariant. The breakdown point depends on the selection of weight function. Naturally one would like to put $w(i/n) = 1$ for those observations that we are certain to be uncontaminated and $w(i/n) = 0$ for those that are clearly instances of contamination, such as bad leverage points. It is convenient that there can be an interval on which the weights diminish smoothly, because in real life situations the level of contamination is unknown. It has been this very property that has led to the choice of weights based on Tukey's ρ function. The idea behind this particular weight function is that we subdivide the interval $[0, 1]$ into three parts with the use of constants $h < g \in (0, 1)$. On $(0, h)$ and $(g, 1)$, the function is constant on values 1 and 0 respectively. On (h, g) it takes the shape of Tukey's ρ function, which will be treated in more detail in the section on S-estimator. In summary, we obtain the following function

$$w(x) = \begin{cases} 1 & \text{for } x \in [0, h], \\ 3\frac{(g-x)^2}{(g-h)^2} - 3\frac{(g-x)^4}{(g-h)^4} + \frac{(g-x)^6}{(g-h)^6} & \text{for } x \in (h, g], \\ 0 & \text{for } x \in (g, 1]. \end{cases} \tag{2.1}$$

Figure 2.5 depicts such a function for $h = 0.5$, $g = 0.9$. The constants were selected in such a way as to make all three sections of w function visible. But what should their values be in practical application to data? Víšek (2017) performed a simulation in order to find the values of h and g that minimize the mean squared error (MSE) of estimates for various levels of contamination by bad leverage points. Simplifying his results a little, we can say that for contamination level c, the optimal value of g seems to be $g = 1 - c$. That means that all bad leverage points are assigned weight zero. The variations in h appeared not to have strong effect on the MSE of estimates, but steep decline of w, i.e. h close to g seemed to be somewhat better. In our simulation study we will follow the rule on g and devise a simplified rule for h. It is clear that by putting h and g very close to each other, we obtain in effect the LTS estimator, and therefore we are able to achieve its asymptotic breakdown point of 50% as well.

Let us turn our attention to the benefits of LWS as compared to previous methods, namely LTS. To begin with, the smoothly decreasing weight function seems to improve the method's sensitivity to inliers (that is, decreases it),

Figure 2.5: Weight function based on Tukey's ρ function



Source: author's computations.

because while LTS either included an observation by assigning weight 1, or disregarded it altogether, LWS can assign any weight from $[0, 1]$, so it does not switch between two very different models, but rather glides smoothly between them. Further, diagnostic tools for robust methods are seldom developed, as the robust statisticians focus primarily on estimation. However, Víšek (2017) shows the distribution of the t-statistics for LWS, which allows to test significance of obtained estimates. The estimator has also been developed to accommodate various special situations, such as panel data, instrumental variables estimation, or estimation in presence of heteroscedastic error term.

Before we discuss the algorithm by means of which the estimator is calculated, let us recall a non-robust method called the weighted least squares (WLS), firstly because it is utilized in the algorithm, and secondly to illustrate how the reversed order of words in the name corresponds to the difference between the two estimators.

Definition 2.13 (The weighted least squares). The weighted least squares estima-

tor (WLS) is defined as

$$\hat{\beta}_{WLS,n,w} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} w_i r_i^2(\beta),$$

where $w_i \in (0, 1)$ are weights assigned to residuals.

The difference between the methods is that WLS requires an extrinsic rule according to which the weights are assigned to the residuals. LWS, on the other hand, assign the weights implicitly. The weighting takes place, as it were, inside the procedure, so the word "weighted" is placed in the middle of the name. Now we can describe the algorithm. It uses the same logic and follows the same steps as Boček & Lachout (1995)'s algorithm for LTS, with rather logical alterations. Again, further details can be found in Víšek (2012).

1. *Select p observations at random and fit a regression plane through them.*
2. *Calculate the residuals from this regression for all observations.*
3. *Reorder the observations according to the smallest squared residual and save the sum of the residuals weighted by function w.*
4. *If the sum is smaller than the sum obtained previously, go to step 5. Otherwise go to step 6.*
5. *Estimate a new regression plane by WLS using weights w and go to step 2.*
6. *If the same model has been found for q times or the algorithm has already gone through r repetitions, end the algorithm. Otherwise go to step 1.*

The alterations occur in steps 3 and 5. Instead of considering h observations with the smallest squared residuals, we sort the observations from smallest squared residual to the largest, multiply each by the appropriate weight, and save the resulting sum. If the sum is smaller than the one obtained in previous iteration, we estimate a new model not by OLS on the h observations with smallest squared residuals, as we would in LTS, but by WLS on all observations, using the same weighting function w.

2.5 S-estimator

The S-estimator was first proposed by Rousseeuw & Yohai (1984), that is, in the same year as LMS and LTS. It follows the logic of M-estimators, which we discussed at the beginning of this chapter, and supplements it so as to remedy their weaknesses. Let us recall that these were two, in principle. Firstly, the M-estimators are not scale and regression equivariant and require studentization of residuals be means of a robust estimator of scale. Secondly, it was discovered that in multiple regression their breakdown point is only $1/p$, p being the dimension of the regression model. Therefore Rousseeuw & Yohai (1984) designed the S-estimator in such a way that instead of minimizing some objective function as was the case with the other estimators we have covered so far, it minimizes the scale of residuals under a certain constraint. Let us follow the steps in which the estimator was defined originally.

Assumption 2.1 (The objective function). Function $\rho : R \to [0, \infty]$ is symmetric, continuously differentiable and $\rho(0) = 0$. Moreover, there is $c > 0$ such that ρ is strictly increasing on $[0, c]$ and constant on $[c, \infty)$.

For a sample $(x_1, \ldots, x_n)$ the scale estimator $s(x_1, \ldots, x_n)$ is defined as the solution of

$$\frac{1}{n} \sum_{i=1}^{n} \rho \left(\frac{x_i}{s} \right) = K$$

where $K = E_\Phi[\rho]$, where Φ is the standard normal distribution. This last is to be understood as follows. Let us have a random variable X with distribution $N(0, 1)$. Then $K = E_\Phi[\rho] = E(\rho(X))$. Then the S-estimator is defined as

$$\hat{\beta}_{S,n,\rho} = \underset{\beta}{\operatorname{argmin}}\, s(r_1(\beta), \ldots, r_n(\beta)).$$

For our purposes, we will reformulate the definition in a way that is equivalent, but more convenient. This definition can be found for instance in Víšek (2015) and is analogical to the way the S-weighted estimator is defined, so we will be able to see very clearly the differences.

Definition 2.14 (The S-estimator). The S estimator is defined as

$$\hat{\beta}_{S,n,\rho} = \underset{\beta}{\operatorname{argmin}} \left\{ \sigma \in (0, \infty) : \sum_{i=1}^{n} \rho \left(\frac{r_i(\beta)}{\sigma} \right) = K \right\},$$

where $K = E_\Phi[\rho]$, where Φ is the standard normal distribution.

This definition is somewhat more complex than the definitions of previous estimators, let us therefore shortly contemplate its logic. Suppose we calculate the value of K. That can be done analytically. For an explicit formula, see Campbell *et al.* (1998). Then we can select some β, i.e. some regression plane, at random and calculate the corresponding residuals. After that, we need to find s such that the equality in the definition (within the set) holds. We save the value of s and proceed to another β. If we did this for all possible values of β, we would select the one that resulted in the smallest value of s. In other words, we choose the value of β for which the residuals were the least spread. Notice that while in other definitions we were minimizing a specified value, in this definition, in the curly bracket, we specify a set and request that value of β which results in the minimum of the set. We pay special attention to the definition, because the definition of S-weighted estimator is analogical.

The estimator's definition specifies properties required of the objective function ρ, but allows for various functions to be selected, much like M-estimators. Rousseeuw & Yohai (1984) proposed to use Tukey's ρ function, which we have already mentioned in passing. Let us take a look at it, since it will be used in our simulation study. Tukey's ρ function is defined as follows.

Definition 2.15 (Tukey's ρ function). Let $c \in (0, \infty)$. Then Tukey's ρ function is defined as

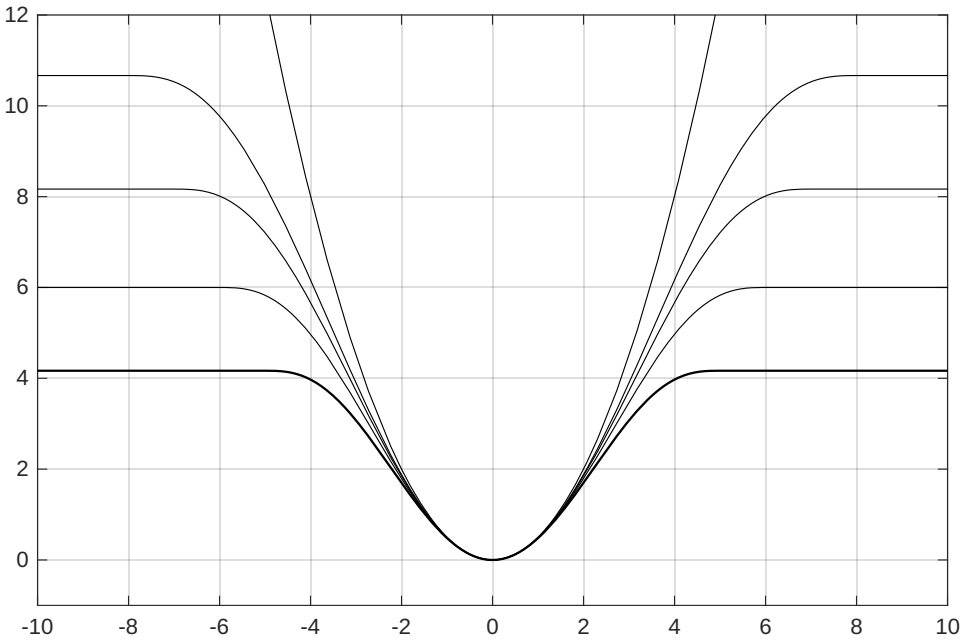
$$\rho(x) = \begin{cases} \frac{x^2}{2} - \frac{x^4}{2c^2} + \frac{x^6}{6c^4} & \text{for } |x| \leq c \\ \frac{c^2}{6} & \text{for } |x| \geq c. \end{cases}$$

It is easy to see that for any fixed point $x \in R$ and increasing c, Tukey's ρ converges to $\frac{1}{2}x^2$. For large c, therefore, the S-estimator uses in effect the same objective function as OLS. This point is illustrated in Figure 2.6 where we can see Tukey's ρ function for for $c = 5, 6, 7, 8$ and the uppermost curve is $\frac{1}{2}x^2$.

Consider what the objective function ρ does. We can imagine the residuals as points on the x-axis. If they are close to zero, they will be evaluated similarly as if we were using OLS. If they get farther, however, their influence stops growing, as ρ is already constant. That means that outlying observations do have an influence, but only to a limited extent. Once they are evaluated by the estimator as outliers, the regression plane can get farther away from them without penalization.

Rousseeuw & Yohai (1984) proved that the asymptotic breakdown point of

Figure 2.6: Tukey's ρ function



Source: author's computations.

S-estimator is 50%. Moreover, for any objective function ρ satisfying 2.1, the asymptotic breakdown point is λ, if

$$\frac{E_\Phi[\rho]}{\rho(c)} = \lambda$$

for $\lambda \in (0, \frac{1}{2})$. Being mostly interested in the maximum breakdown point available, that is 50%, and considering Tukey's ρ function, Rousseeuw & Yohai (1984) report the value $c = 1.547$ that yields this result. However, in practice as well as in simulation, we might be interested in S-estimator with lower breakdown point, if we are expecting (or simulating) lower degree of contamination. The idea is that healthy observations provide useful information and therefore should not be suppressed. For this purpose, we shall now embark on a side-quest of sorts, to estimate all 50 values of c that result in corresponding integer value of breakdown point of S-estimator. It will be useful to us later in the simulation study. Let us describe our method.

Our objective is for every $\lambda \in \{0.01, 0.02, \ldots, 0.50\}$ to find $c_\lambda > 0$ so that

$$\frac{E_\Phi[\rho]}{\rho(c_\lambda)} = \lambda.$$

For finite sample sizes, $E_\Phi[\rho]$ can be calculated exactly, see for instance

Campbell *et al.* (1998), but as we are interested in asymptotic breakdown point, and we do not need extremely precise values of c, we can generate a random sample $(x_1, \ldots, x_n)$ from standard normal distribution and simply approximate as

$$E_\Phi[\rho] \approx \frac{1}{n} \sum_{i=1}^{n} \rho(x_i).$$

Of course we know that $\rho(c) = c^2/6$, so we obtain the formula

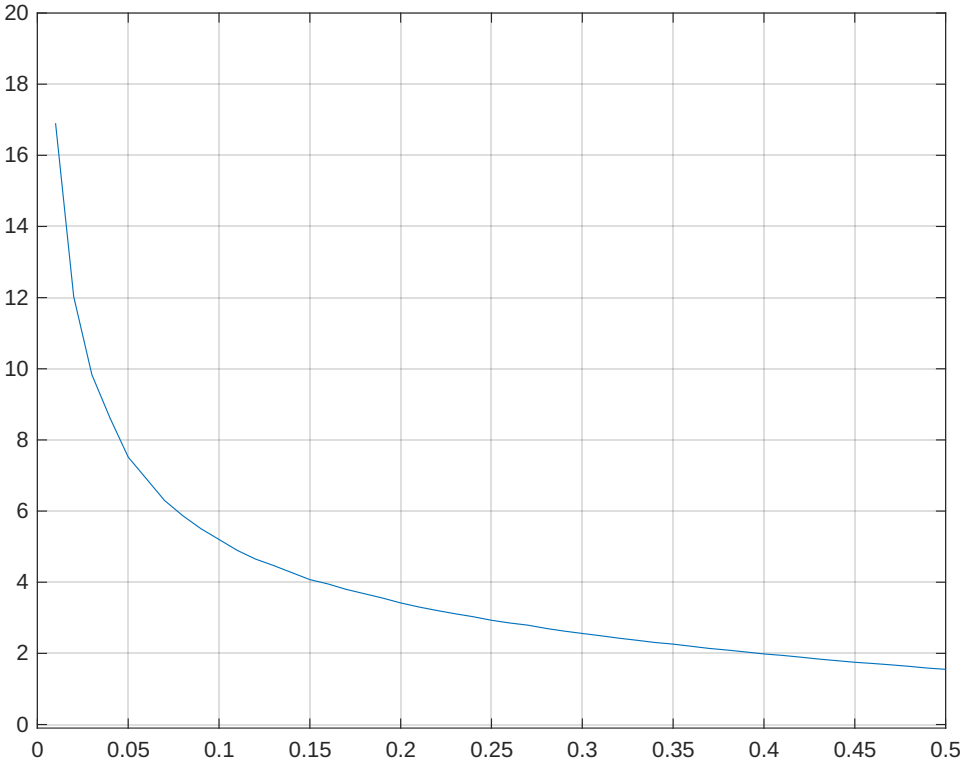
$$\frac{6 \sum_{i=1}^{n} \rho(x_i)}{nc^2} = \lambda.$$

We cannot isolate c from this formula, so we try different values of c until the left-hand side is close enough to the right-hand side. An algorithm was implemented in Matlab and we provide the code in Appendix A. In Table 2.1 we report all the calculated values and the same is depicted in Figure 2.7 where we have the desired breakdown point on the x-axis and the corresponding c value on the y-axis. We will make use of these results in our simulation study.

Table 2.1: Approximate value of parameter c in Tukey's ρ function

λ	c	λ	c	λ	c	λ	c	λ	c
0.01	16.90	0.11	4.90	0.21	3.30	0.31	2.50	0.41	1.94
0.02	12.03	0.12	4.65	0.22	3.20	0.32	2.43	0.42	1.89
0.03	9.83	0.13	4.47	0.23	3.11	0.33	2.37	0.43	1.84
0.04	8.61	0.14	4.27	0.24	3.03	0.34	2.31	0.44	1.80
0.05	7.52	0.15	4.07	0.25	2.93	0.35	2.26	0.45	1.75
0.06	6.91	0.16	3.95	0.26	2.85	0.36	2.20	0.46	1.72
0.07	6.30	0.17	3.80	0.27	2.79	0.37	2.14	0.47	1.68
0.08	5.87	0.18	3.68	0.28	2.70	0.38	2.09	0.48	1.64
0.09	5.51	0.19	3.56	0.29	2.63	0.39	2.04	0.49	1.59
0.10	5.20	0.20	3.42	0.30	2.56	0.40	1.98	0.50	1.54

Figure 2.7: Selection of parameter c in Tukey's ρ function



Source: author's computations.

Chapter 3

S-Weighted Estimator

The S-weighted estimator was first introduced by professor Jan Ámos Víšek (2015) and to this day, literature on the subject is very thin. To the best of the present author's knowledge, only five articles have been published concerning the estimator, all of them authored by professor Víšek. They include the just referenced article *S-Weighted Estimators*, then *Representation of SW-estimators* and *Coping with Level and Different Type of Contamination by SW-Estimator* published the following year, and two more papers from 2019, *Asymptotics of S-Weighted Estimators* and *S-Weighted Instrumental Variables*. To make things even worse, some of these articles are rather difficult to obtain.

Our general discussion of the S-weighted estimator will be somewhat shorter, since we have established all the preliminaries in the previous chapter, and the estimator's definition consists essentially in putting together all the estimators discussed so far. Let us therefore proceed with the definition.

Definition 3.1 (The S-weighted estimator). The S-weighted estimator is defined as

$$\hat{\beta}_{SW,n,\rho} = \underset{\beta}{\operatorname{argmin}} \left\{ \sigma \in (0,\infty) : \sum_{i=1}^{n} w\left(\frac{i-1}{n}\right) \rho\left(\frac{r_{(i)}(\beta)}{\sigma}\right) = K \right\},$$

where $K = E_\Phi[\rho]$, where Φ is the standard normal distribution and w is a weight function.

We can see that this is a modification of the definition of S-estimator. What is new is the inclusion of the weight function in the sum. We can therefore see the S-weighted estimator as the combination of LWS and S-estimator, which was exactly the idea that led to its formulation. Víšek (2015) showed how LWS and the S-estimator are all covered by S-weighted estimator as its special cases. Since LMS and LTS can be obtained from LWS by employing appropriate

weight functions, LMS and LTS are also subsumed under S-weighted estimator. Finally, to turn S-weighted estimator into an S-estimator, one needs merely to put $w(x) = 1, x \in [0, 1]$. It is in this sense that Víšek (2015) characterizes the S-weighted estimator as a "roof" of all the estimators that precede.

Now to the estimator's properties. Its consistency was proved together with its proposal for a slightly narrower case where σ is restricted in the definition to a finite interval $[a, b], 0 < a < b < \infty$. This however is a technicality only, because there seem to be no limitations on b, except that it be finite. In addition to that, in *Representation of SW-estimators*, Víšek (2016b) proved the $\sqrt{n}$-consistency, so that we know that the rate of convergence of the estimator is the standard one in this field (recall LMS which suffered from slow rate of convergence, being only $\sqrt[3]{n}$-consistent. In the same paper, the asymptotic representation of the estimator is derived, facilitating future development of diagnostic tools.

Some properties of S-weighted estimator follow from its definition and the fact that the two main components that constitute it, LWS and S-estimator, also possess them. S-weighted estimator is clearly scale and regression equivariant for the same reasons that the S-estimator is – the residuals are implicitly being standardized inside the objective function. The breakdown point seems to require more attention in future research, since it appears to be barely mentioned in the existing literature. This is probably also due to the estimator's derivation from LWS and S-estimator, whose breakdown points are known. It is however an entirely new feature that the S-weighted estimator allows us to decide what part of the robustness is achieved in S-fashion, that is, by the objective function ρ, and what part is achieved in the LWS-fashion, that is, by the weight function w. We shall pay some attention to this in the simulation study.

Now a crucial problem with this new estimator seems to be its implementation. In this regard, the present author should like again to express his gratitude to professor Víšek for very generously providing codes in Matlab, without which this thesis would not have been possible. The published literature, however, contains only passing references to the effect that the S-weighted estimator can be calculated by a straightforward modification of the algorithm used for LWS and LTS. The only explicit description of thus modified algorithm that we could discover was in lecture slides from ROBUST 2016 international statistical conference (Víšek 2016a). We have included a link in the biography and the slides are also available from the author of this thesis upon request.

Let us see what the modified algorithm presented therein consists in.

1. *Select p observations at random and fit a regression plane through them.*
2. *Calculate* $S(\hat{\beta}_{present}) = \sum_{i=1}^{n} w\left(\frac{i-1}{n}\right) \rho\left(\frac{r_{(i)}(\beta)}{\sigma}\right)$.
3. *If* $S(\hat{\beta}_{present}) < S(\hat{\beta}_{past})$*, go to step 4. Otherwise go to step 5.*
4. *Reorder the observations by the order of the smallest squared residuals, estimate new* $\hat{\beta}_{present}$ *by weighted* M_ρ*-estimator using weights* w *and go to step 2.*
5. *If the same model has been found for* q *times or the algorithm has already gone through* r *repetitions, end the algorithm, and return* $\hat{\beta}$ *for which* $S(\hat{\beta})$ *was the smallest. Otherwise go to step 1.*

Observe that step 3 is the crucial one here. The algorithm would still work if we only selected random p-tuple of observations, evaluated the sum from step 2 and saved it if it was the smallest one found so far. That is in essence what PROGRESS does (see Section 2.3) and the only problem is that it is too slow. That is why Boček & Lachout (1995) added the inner cycle that iteratively improves on the randomly selected model from first step. It in fact makes no difference what estimator is used in step 4, as long as the result is checked in step 3 as to whether it is closer to the solution of the extremal problem or not. Of course some estimators can make the algorithm faster, as they converge faster to the solution, some estimators might not converge to it at all. But as long as the algorithm returns some result at the end (and provided that the r in step 5 is sufficiently high), it will return a model that is close to the sought out minimum (the closest one it could find at any rate).

We shall now turn our attention to simulation studies in which the estimator has so far been employed. The first article, *S-Weighted Estimators* presents the new estimator and offers the proof of its consistency, but contains yet no numerical study. The four remaining papers all include at least a minor simulation study.

In *Coping with Level and Different Type of Contamination by SW-Estimator*, Víšek (2016b) performs the relatively most systematic simulation that has been performed so far. He considers the standard linear regression model with constant and four explanatory variables and contaminates the data in three different ways. First by bad leverage points only, then by bad and good leverage

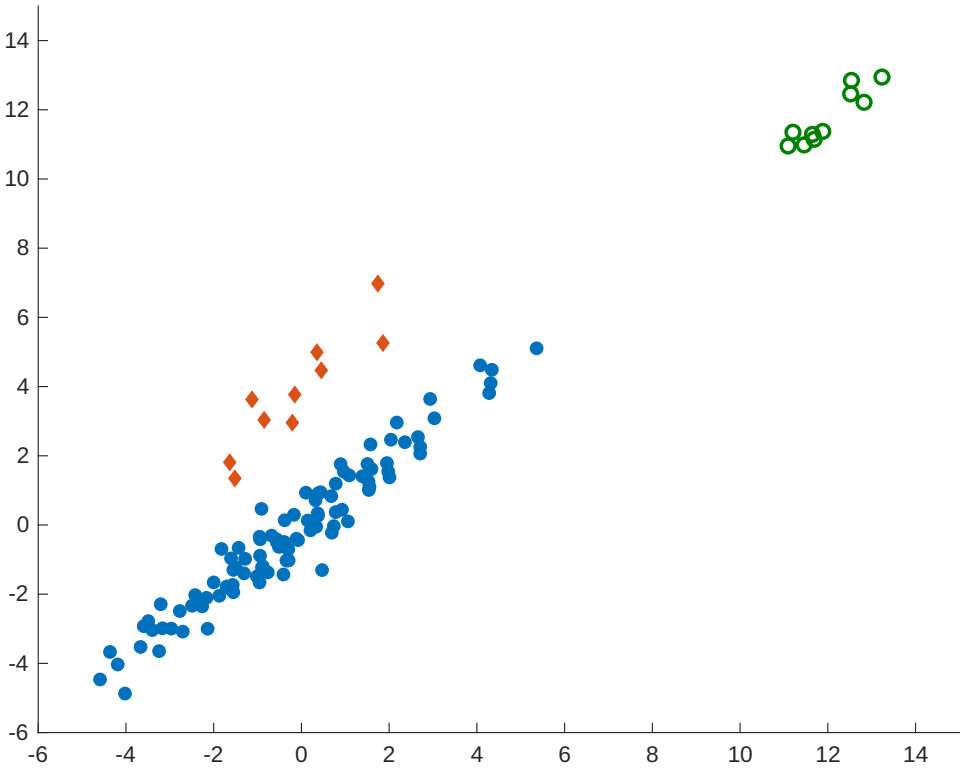
points, and finally by outliers and good leverage points. Each type of contamination is done on five different levels of intensity, namely on 1, 2, 3, 5, and 10% contamination levels. For each level and type of contamination, the model is estimated by OLS, S-estimator, W-estimator, and S-weighted estimator. We do not discuss W-estimator in this thesis, since it is not one of the standard tools of robust statistics, so let us only repeat after Víšek (2016b) that it is a modified S-estimator that is estimated only after deleting those observations, whose Mahalanobis distance from the center of gravity of the whole dataset exceeds given threshold.

The results show that the S-weighted estimator is able to obtain estimates closer to the real regression parameters of the uncontaminated population than the other estimators (most notably S-estimator) whenever good leverage points were present. This is illustrated by means of a scatter plot which we replicate in Figure 3.1. The blue filled circles represent the uncontaminated data, the orange diamond-shaped points are the outliers, and the green empty circles depict good leverage points. All robust methods were able to distinguish outliers and suppress their influence. But S- and W-estimators, according to Víšek (2016b), in effect look for the smallest-volume ellipsoid containing a priori given portion of the data, and thus are led to rejecting the good leverage points, while the S-weighted estimator includes them. Thus in cases where good leverage points were present, the S-weighted estimator performed significantly better than S- and W-estimators. This is the main conclusion which was also reaffirmed in the simulation studies that followed.

We can present the rest of the simulations in more brevity, since they are less extensive and bring no conclusions comparable in significance to the one we just described. *Representation of SW-estimators* (Víšek 2016c) includes minor simulation, which considers only contamination by outliers on 1% and 5% levels, with the presence of good leverage points. The standard linear model is estimated by OLS, S- and S-weighted estimators, the last one surpassing S-estimator in exactness of estimation. It is worthwhile to mention that even for 1% of contamination the OLS estimates were already completely off.

Asymptotics of S-Weighted Estimators (Víšek 2019a) contains pretty much the same simulation setup, only investigating more different levels of contamination. Again, due to the presence of outliers and good leverage pooints, the S-weighted estimator performs better than S-estimator, but strangely this time the OLS estimates are basically correct and even vary less than those obtained by S-estimator. The paper includes no detailed discussion of the results of the

Figure 3.1: Structure of contaminated data



Source: author's computations.

simulation, but after experiences from our simulation study, we can attribute this effect also to the good leverage points. Since the OLS method does not suppress any observations, the good leverage points exert their strong influence and force OLS to estimate the correct values.

Finally, *S-Weighted Instrumental Variables* (Víšek 2019b), as the title suggests, investigates a modified model, which is developed for those situations where the orthogonality condition is broken. The usual set of estimators is put to test, namely OLS, S-, W- and S-weighted estimator. Leading to the same conclusions as previous studies, this simulation is noteworthy (apart from estimating a specialized model, different from the basic linear regression in the preceding articles) for varying the generated sample sizes from 100 to 500 in 100-observations steps.

To summarize, the limited existing literature on S-weighted estimator contains its definition and proof of $\sqrt{n}$-consistency, derives the asymptotic representation and tests the estimator's performance in mostly standard situations, comparing it primarily to OLS and S-estimator. The main limitation is the algorithm for the estimator's calculation as well as its implementation, which

seem to have never been properly published. Further, the existing simulation studies focus on comparisons of the estimator with S-estimator. No numerical study has been found comparing the estimator's performance to that of LWS. Yet, such a comparison is important, firstly since LWS, not S-estimator, appears to be the state-of-the-art method of robust statistics, and secondly because the S-weighted estimator is a combination of the two, so a side-by-side employment of all three methods would be logical. These two tasks, therefore shall be performed in the following chapters of this thesis.

Chapter 4

Simulation

4.1 Simulation setup

In the previous chapter we have summarized the simulation studies involving the S-weighted estimator that have been performed so far. We came to the conclusion that for a full appraisal of the new estimator's merits, a study should be conducted to compare its performance not only with the S-estimator but also with the LWS estimator, as these two are the S-weighted estimator's "parents." To such a simulation is devoted this section.

Our simulation will follow generally the same patters as that of Víšek (2016b), the main difference being that we shall include a LWS estimate. Let us describe the simulated data first. We will be considering a standard linear model with a constant and four i.i.d. explanatory variables $X_k \sim N(0,1)$, $k = 1, \ldots, 4$. The vector of true population parameters will be $\beta^{true} = (1, 2, -3, 4, -5)^T$. The selection of these parameters is more or less arbitrary, since all of the estimators we employ are scale and affine equivariant. We therefore selected integer values that are easy to remember and include both positive and negative values. The vector of disturbances e is independent of the explanatory variables and also normally distributed, $e \sim N(0,1)$. Putting this all together, we obtain the model

$$y_i = 1 + 2X_{i1} - 3X_{i2} + 4X_{i3} - 5X_{i4} + e_i, \quad i = 1, \ldots, n.$$

In selecting the sample size n we kept in mind the computational requirements of the whole simulation, and the need to have large n relative to the dimension of the model p. Parenthetically, let us say explicitly, to prevent any misunderstanding, that in the model dimension we include the constant, there-

fore the dimension is $p = 5$. Taking all this into consideration, the sample size shall be $n = 500$.

We consider two types of contamination: outliers and leverage points. Moreover, in some scenarios, we also include good leverage points, but those are not considered contamination, since they strengthen the results of the regression (if the estimator is able to utilize the information they contain.)

For an outlier, the X values were left the same, but the response variable y was replaced by

$$\tilde{y}_i = -5\left(1 + 2X_{i1} - 3X_{i2} + 4X_{i3} - 5X_{i4}\right) + e_i.$$

This definition ensures that the values of the X variables are within the main bulk (except for rare naturally occurring outliers) while the y value is unusual. We also considered simply adding a constant to shift the outliers alongside the y-axis like so: $\tilde{y} = y + c$, but it seems that this would contaminate only the estimate of intercept. Therefore we decided for this kind of definition of outliers, which in effect results in part of the data following a different model with all the parameters being different from the model of the uncontaminated data.

The leverage points were generated as follows. First we multiplied the explanatory variables (but not the constant) as $\tilde{X}_{ik} = 20X_{ik}$, $k = 1, 2, 3, 4$. Then the y variable was calculated as

$$\tilde{y}_i = \begin{cases} 1 + 2\tilde{X}_{i1} - 3\tilde{X}_{i2} + 4\tilde{X}_{i3} - 5\tilde{X}_{i4} + e_i & \text{for a good leverage point,} \\ -1\left(1 + 2\tilde{X}_{i1} - 3\tilde{X}_{i2} + 4\tilde{X}_{i3} - 5\tilde{X}_{i4}\right) + e_i & \text{for a bad leverage point.} \end{cases}$$

In other words, the good leverage points follow the same model, only their values of explanatory variables are extreme, while the bad leverage points follow a model with the signs switched.

We considered five different combinations of outliers and leverage points, which we call scenarios. In the first scenario, we contaminate the data by outliers only. Then, in the second scenario, we include only bad leverage points. This allows us to observe the effects of these two different kinds of contamination separately. Scenario three includes outliers and good leverage points. This is the case where the S-weighted estimator was observed in the past simulations to outperform the S-estimator, as the latter disregarded the information provided by good leverage points. The fourth scenario contains both good and

bad leverage points but no outliers. This is to see whether the estimators can utilize the former and avoid confusion by the latter. Finally, in the fifth and last scenario we include all of the above, namely outliers, bad leverage points, and good leverage points. For better orientation, we summarize this in Table 4.1.

Table 4.1: Scenarios

Scenario 1	Only outliers
Scenario 2	Only bad leverage points
Scenario 3	Outliers and good leverage points
Scenario 4	Good and bad leverage points
Scenario 5	Outliers and good and bad leverage points

In each scenario we considered six different levels of contamination, namely 1%, 2%, 3%, 5%, 10%, and 25%. As we have already mentioned, good leverage points are not considered as contamination, and their number was always the same as the number of contaminated observations. For instance in scenario 3, we had 50 outliers and 50 good leverage points for contamination of 10%. In scenario 5, contamination 2%, we had 5 outliers, 5 bad leverage points (that is, 10 observations out of 500 were contaminated) and 10 good leverage points.

For each sample, we estimated the model by OLS, LWS, S-estimator and S-weighted estimator. While the first requires no further comments, the remaining three allow selection from various weight and objective functions, and the algorithms by which they are calculated have flexible rules regarding the repetitions of inner and outer cycles. Therefore we now need to discuss our choices of these parameters.

Recalling the definition of the LWS (least weighted squares) estimator

$$\hat{\beta}_{LWS,n,w} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} w\left(\frac{i-1}{n}\right) r^2_{(i)}(\beta),$$

we can see that the weight function w needs to be specified. Throughout the simulation, we are going to be using the weight function described in Equation 2.1. It decreases smoothly on $[h, g]$, $0 < h < g < 1$. The choice of constants h and g is described in Section 2.4 and we will follow it here. We device the following rule. Let κ be the level of contamination, not as percentage now but as a fraction, so that $\kappa \in [0, 0.5)$. We calculate the constants h and g as

$$g = 1 - \kappa - 0.005$$

$$h = -0.3 + 1.3g.$$

This rule was obtained by collecting the values of h and g from previous simulations and regressing g on κ, then h on g (each time with constant) and then rounding a little. Generally, g is set in such a way that all the contamination has a chance to receive weights 0 (function w equals 0 on $[g, 1]$). As we concluded in Section 2.4, the value of h seems not to have a strong influence or the results, so we follow the rule obtained from the regression. The slope coefficient 1.3 means that the interval $[h, g]$ should be a little wider for higher contamination levels.

In case of the S-estimator

$$\hat{\beta}_{S,n,\rho} = \underset{\beta}{\operatorname{argmin}} \left\{ \sigma \in (0, \infty) : \sum_{i=1}^{n} \rho \left(\frac{r_i(\beta)}{\sigma} \right) = K \right\},$$

we are to select the ρ function. We continue with the traditional choice of Tukey's ρ function

$$\rho(x) = \begin{cases} \frac{x^2}{2} - \frac{x^4}{2c^2} + \frac{x^6}{6c^4} & \text{for } |x| \leq c \\ \frac{c^2}{6} & \text{for } |x| \geq c. \end{cases}$$

As we discussed in Section 2.5, the constant c determines the breakdown point (which we treat as the measure of robustness against outliers and leverage points) of the estimator, and we have calculated the approximate value of c for every integer value of breakdown point that we might desire (Table 2.1). However, it turned out in trial simulations that it is better to set c a little lower, so as to obtain a higher breakdown point than the barest minimum required. For instance, if the contamination is 10%, then we need the breakdown point to be more than 10%. Furthermore, looking at simulations such as was done by Víšek (2016b), where some optimal values of c were estimated by a forward search, we can confirm that the optimal breakdown point is higher than the level of contamination. Therefore we decided to set c in such a way that the resulting breakdown point equals twice the contamination level.

Finally, as the S-weighted estimator

$$\hat{\beta}_{SW,n,\rho} = \underset{\beta}{\operatorname{argmin}} \left\{ \sigma \in (0, \infty) : \sum_{i=1}^{n} w \left(\frac{i-1}{n} \right) \rho \left(\frac{r_{(i)}(\beta)}{\sigma} \right) = K \right\},$$

uses both a weight function and an objective function, we have to figure

out both. Strangely, nothing explicit can be found in the existing simulations with S-weighted estimator. Constants c, h, and g are selected for S-estimator and LWS, respectively, and then the same values are used for the S-weighted estimator. Let us consider it. The residuals first go to the objective function ρ, where the largest ones are suppressed but not quite obliterated, thus achieving some degree of robustness. But then they are weighted by function w in such a way that the largest ones are removed completely. Does this not add more robustness on top of that achieved by the objective function ρ? In other words, we conjecture that by using the same values of constants c, h, and g, the S-weighted estimator is calibrated in such a way that it becomes more robust and potentially less efficient than LWS and S-estimator. To make our results comparable with those obtained by Víšek (2016b), we keep constants c, h, and g the same, not deriving separate values for the S-weighted estimator. We try to answer this issue in a separate part of the simulation.

For each scenario and for each level of contamination, we create $m = 100$ datasets and estimate the model on each of them by OLS, LWS, S-estimator, and S-weighted estimator. For each estimator we summarize the results by arithmetic average and mean squared error (MSE) separately for each estimated coefficient

$$\hat{\beta}_k^{estimator} = \frac{1}{100}\sum_{j=1}^{100}\hat{\beta}_{k,j}^{estimator},\ k = 1,\ldots,5,$$

$$\hat{MSE}\left(\hat{\beta}_k^{estimator}\right) = \frac{1}{100}\sum_{j=1}^{100}\left(\hat{\beta}_{k,j}^{estimator} - \beta_k^{true}\right)^2,\ k = 1,\ldots,5.$$

Note that we are using mean squared error and not sample variance or some other estimate of scale. That is because we are interested in how far on average the estimates are from the ones we were hoping to obtain. Observe that mean squared error does not measure the spread of the values. Even if the estimator reported the same estimates every time we repeated the simulation, MSE would still be positive if they were systematically different from β^{true}.

4.2 Results

4.2.1 Only outliers

Contamination level = 1%, $c = 12.15$, $h = 0.980$, $g = 0.985$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.926_{(0.072)}$	$1.846_{(0.110)}$	$-2.783_{(0.118)}$	$3.704_{(0.178)}$	$-4.656_{(0.191)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.990_{(0.020)}$	$2.013_{(0.014)}$	$-2.980_{(0.018)}$	$3.998_{(0.022)}$	$-4.975_{(0.016)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.986_{(0.020)}$	$2.009_{(0.015)}$	$-2.970_{(0.019)}$	$3.987_{(0.020)}$	$-4.969_{(0.016)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.990_{(0.021)}$	$2.014_{(0.015)}$	$-2.978_{(0.018)}$	$3.999_{(0.022)}$	$-4.977_{(0.016)}$
Contamination level = 2%, $c = 9.85$, $h = 0.968$, $g = 0.975$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.897_{(0.100)}$	$1.773_{(0.156)}$	$-2.603_{(0.249)}$	$3.507_{(0.368)}$	$-4.452_{(0.424)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.003_{(0.022)}$	$2.003_{(0.020)}$	$-3.000_{(0.021)}$	$3.995_{(0.019)}$	$-4.989_{(0.015)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.004_{(0.023)}$	$2.006_{(0.017)}$	$-2.987_{(0.021)}$	$3.984_{(0.018)}$	$-4.974_{(0.017)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.003_{(0.022)}$	$2.000_{(0.020)}$	$-3.001_{(0.022)}$	$3.995_{(0.018)}$	$-4.985_{(0.016)}$
Contamination level = 3%, $c = 7.55$, $h = 0.954$, $g = 0.965$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.807_{(0.168)}$	$1.659_{(0.265)}$	$-2.480_{(0.383)}$	$3.280_{(0.708)}$	$-4.103_{(1.013)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.004_{(0.020)}$	$1.997_{(0.020)}$	$-2.996_{(0.018)}$	$4.016_{(0.020)}$	$-4.995_{(0.023)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.999_{(0.018)}$	$1.984_{(0.020)}$	$-2.996_{(0.018)}$	$4.011_{(0.018)}$	$-4.982_{(0.022)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.010_{(0.020)}$	$1.995_{(0.022)}$	$-3.002_{(0.019)}$	$4.010_{(0.022)}$	$-4.992_{(0.024)}$
Contamination level = 5%, $c = 5.15$, $h = 0.928$, $g = 0.945$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.662_{(0.308)}$	$1.359_{(0.628)}$	$-2.103_{(1.023)}$	$2.807_{(1.651)}$	$-3.526_{(2.457)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.003_{(0.023)}$	$1.982_{(0.025)}$	$-3.016_{(0.022)}$	$3.994_{(0.022)}$	$-4.982_{(0.020)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.007_{(0.023)}$	$1.966_{(0.024)}$	$-3.003_{(0.023)}$	$3.984_{(0.020)}$	$-4.981_{(0.021)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.013_{(0.026)}$	$1.976_{(0.032)}$	$-3.002_{(0.027)}$	$3.991_{(0.027)}$	$-4.983_{(0.022)}$
Contamination level = 10%, $c = 2.92$, $h = 0.863$, $g = 0.895$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.443_{(0.693)}$	$0.809_{(1.862)}$	$-1.164_{(3.826)}$	$1.499_{(6.808)}$	$-2.074_{(9.065)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.016_{(0.024)}$	$1.998_{(0.030)}$	$-2.994_{(0.030)}$	$4.005_{(0.026)}$	$-5.019_{(0.025)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.035_{(0.046)}$	$1.977_{(0.075)}$	$-3.023_{(0.075)}$	$4.025_{(0.069)}$	$-5.053_{(0.063)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.016_{(0.041)}$	$2.018_{(0.051)}$	$-3.009_{(0.034)}$	$4.016_{(0.033)}$	$-5.027_{(0.043)}$
Contamination level = 25%, $c = 1.55$, $h = 0.669$, $g = 0.745$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$-0.642_{(3.245)}$	$-1.065_{(10.237)}$	$1.299_{(19.347)}$	$-1.883_{(35.453)}$	$2.461_{(56.666)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.990_{(0.047)}$	$1.990_{(0.033)}$	$-2.993_{(0.030)}$	$3.979_{(0.034)}$	$-5.009_{(0.035)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.812_{(4.824)}$	$2.273_{(17.168)}$	$-1.902_{(36.794)}$	$2.937_{(32.411)}$	$-3.361_{(45.118)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.053_{(0.252)}$	$1.923_{(0.375)}$	$-3.044_{(0.382)}$	$3.889_{(0.251)}$	$-5.020_{(0.367)}$

In this section we present the main results of the simulation. Let us briefly inspect the structure of the table, since it will stay the same in all the scenarios. In each heading we can see the contamination level, beginning with the lowest, 1%, and increasing up to 25%, the highest level of contamination we were considering. For each level we report the values of the tuning constants, c for objective function ρ in S- and S-weighted estimators, and constants h and g which determine the shape of weight function w utilized by LWS and S-weighted estimator. Below the heading, we have the results, always beginning with OLS as the baseline against which we compare the robust methods, followed by LWS, S- and S-weighted estimates. Each cell contains the average estimate and small, in the parenthesis, the mean squared error. The leftmost column of the results contains the estimate of intercept and the remaining columns are the four slope estimates from β_1 to β_4.

Looking at the results for scenario 1, where we contaminated the data only by outliers, we can first notice that all three robust estimators performed better than OLS. Even 1% contamination already shifts the OLS estimates' average values and causes MSE to be higher than that of robust methods by an order of magnitude. Apart from the highest contamination level, all three robust methods return estimates close to the true values of the parameters and do so with relatively small MSE. At contamination levels up to 5%, the results of LWS, S-, and S-weighted estimators are basically indistinguishable. At level 10%, the S-estimator begins to suffer from slightly larger MSE as compared to other two robust estimators. At 25% contamination, the S-estimators MSE skyrockets into double digits and even its average estimates are somewhat off. We are at a loss to find an explanation for this failure. The c constant that we used for this level of contamination, $c = 1.55$ in fact results in the theoretic breakdown point of S-estimator being close to 50%. Perhaps we have made the estimator too robust, sacrificing too much of its efficiency. But the same value of c was used for the S-weighted estimator which seems to be doing much better at this contamination level.

Looking now at the S-weighted estimator's performance, we can conclude that up to 10% of contamination the results are indistinguishable from those of LWS. At 25% contamination, the S-weighted estimator is still robust, its estimates are close to the real values, but the MSE here is ten times that of LWS. We perform two-sided Kolmogorov-Smirnov test to confirm this finding. For details on this test, see for example Bartoszynski & Niewiadomska-Bugaj (2008). In Table 4.2 we can see the p-values from all the individual test, which

were performed separately for each parameter and each contamination level. The values are generally very high, except for 25% contamination where we can reject the null hypothesis of the S-weighted estimates and LWS estimates being from the same distribution. Therefore at the highest contamination level, we conclude that the S-weighted estimator has performed slightly worse than the LWS.

Table 4.2: Kolmogorov-Smirnov test, scenario 1

	β_1	β_2	β_3	β_4	β_5
1%	0.9921	1.0000	0.9995	0.9921	0.9995
2%	0.9995	0.9921	1.0000	0.9610	0.9610
3%	0.9995	0.9610	0.9921	0.5560	0.8938
5%	0.6766	0.6766	0.5560	0.9610	0.8938
10%	0.5560	0.5560	0.8938	0.4431	0.5560
25%	0.0205	0.0205	0.0003	0.0010	0.0006

To add one final, minor observation, we can notice that more often than not, all three robust estimates seem to be deviated in the same direction. All three of them are either a little higher than the true value, or all three of them are a little lower. We conjecture that this is on account of the structure of the data caused by the artificial contamination, but to give a deeper analysis is beyond this thesis' scope.

4.2.2 Only bad leverage points

Contamination level = 1%, $c = 12.15$, $h = 0.980$, $g = 0.985$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.077_{(0.307)}$	$-0.751_{(10.680)}$	$1.096_{(20.695)}$	$-1.139_{(30.202)}$	$0.992_{(40.235)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.990_{(0.016)}$	$2.002_{(0.022)}$	$-3.005_{(0.023)}$	$4.000_{(0.019)}$	$-5.007_{(0.023)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.992_{(0.016)}$	$1.995_{(0.021)}$	$-2.995_{(0.020)}$	$4.002_{(0.020)}$	$-5.004_{(0.020)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.991_{(0.017)}$	$2.007_{(0.026)}$	$-3.005_{(0.023)}$	$4.000_{(0.020)}$	$-5.004_{(0.024)}$
Contamination level = 2%, $c = 9.85$, $h = 0.968$, $g = 0.975$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.989_{(0.330)}$	$-1.238_{(11.315)}$	$1.905_{(25.041)}$	$-2.587_{(44.426)}$	$3.299_{(70.096)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.023_{(0.022)}$	$1.990_{(0.023)}$	$-3.018_{(0.022)}$	$3.984_{(0.026)}$	$-4.996_{(0.026)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.025_{(0.020)}$	$1.989_{(0.021)}$	$-3.019_{(0.019)}$	$3.982_{(0.023)}$	$-5.007_{(0.018)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.021_{(0.023)}$	$1.995_{(0.023)}$	$-3.011_{(0.025)}$	$3.984_{(0.028)}$	$-4.994_{(0.025)}$
Contamination level = 3%, $c = 7.55$, $h = 0.954$, $g = 0.965$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.921_{(0.362)}$	$-1.634_{(13.343)}$	$2.325_{(28.549)}$	$-3.192_{(51.901)}$	$3.924_{(79.919)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.018_{(0.020)}$	$2.008_{(0.027)}$	$-2.995_{(0.021)}$	$3.995_{(0.020)}$	$-5.003_{(0.026)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.015_{(0.019)}$	$2.005_{(0.020)}$	$-3.002_{(0.014)}$	$3.996_{(0.013)}$	$-4.998_{(0.023)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.021_{(0.022)}$	$2.008_{(0.030)}$	$-2.997_{(0.025)}$	$3.990_{(0.021)}$	$-5.009_{(0.029)}$
Contamination level = 5%, $c = 5.15$, $h = 0.928$, $g = 0.945$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.003_{(0.457)}$	$-1.775_{(14.277)}$	$2.700_{(32.513)}$	$-3.569_{(57.327)}$	$4.514_{(90.544)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.989_{(0.025)}$	$2.038_{(0.027)}$	$-3.032_{(0.029)}$	$4.027_{(0.024)}$	$-4.992_{(0.019)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.992_{(0.022)}$	$2.023_{(0.018)}$	$-3.032_{(0.023)}$	$4.024_{(0.023)}$	$-5.007_{(0.021)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.996_{(0.030)}$	$2.028_{(0.031)}$	$-3.026_{(0.031)}$	$4.018_{(0.029)}$	$-5.005_{(0.023)}$
Contamination level = 10%, $c = 2.92$, $h = 0.863$, $g = 0.895$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.069_{(0.438)}$	$-1.906_{(15.262)}$	$2.858_{(34.325)}$	$-3.802_{(60.876)}$	$4.746_{(94.982)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.021_{(0.023)}$	$1.968_{(0.043)}$	$-2.982_{(0.038)}$	$4.009_{(0.044)}$	$-4.971_{(0.039)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.014_{(0.041)}$	$2.024_{(0.053)}$	$-2.977_{(0.070)}$	$4.002_{(0.056)}$	$-4.935_{(0.071)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.012_{(0.043)}$	$1.977_{(0.045)}$	$-3.012_{(0.061)}$	$4.009_{(0.064)}$	$-4.944_{(0.056)}$
Contamination level = 25%, $c = 1.55$, $h = 0.669$, $g = 0.745$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.960_{(0.416)}$	$-1.971_{(15.768)}$	$2.947_{(35.372)}$	$-3.935_{(62.972)}$	$4.924_{(98.488)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.995_{(0.040)}$	$1.962_{(0.115)}$	$-3.011_{(0.116)}$	$4.010_{(0.061)}$	$-4.968_{(0.066)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.053_{(1.574)}$	$2.076_{(4.129)}$	$-3.543_{(17.310)}$	$3.986_{(1.554)}$	$-4.877_{(3.398)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.061_{(0.409)}$	$1.824_{(3.119)}$	$-2.901_{(1.686)}$	$3.969_{(1.557)}$	$-4.894_{(0.981)}$

Having contaminated the data with bad leverage points exclusively, we can see that the OLS method is completely lost even at 1% contamination level. At 25%, its estimates have the correct absolute value, but the signs are switched, which reflects the way we have defined the bad leverage points. Notice however, that at the highest contamination level, OLS still suffers from high MSE, which suggests that the bad leverage points are not strong enough to completely overpower the healthy observations. It seems that the leverage points here have convinced OLS that they represent the true model, but the uncontaminated data still function as strong noise.

The robust methods have successfully identified the bad leverage points as contamination and suppressed their influence. At contamination levels up to 5% there is no discernible difference between them. At 10%, LWS seems to perform ever so slightly better than the other two estimators, which is reflected in the somewhat smaller p-values in Kolmogorov-Smirnov tests. At level 25% we can reject the null hypothesis that LWS and S-weighted estimator give estimates from the same population, with LWS being the more successful one. Recalling that the S-weighted estimator was developed as a combination of LWS and S-estimator, we can conjecture that perhaps its S-element is at fault here, since the results of S-estimator are worse still. Nevertheless, we must acknowledge that all three robust methods have provided reliable estimates at all levels of contamination.

Table 4.3: Kolmogorov-Smirnov test, scenario 2

	β_1	β_2	β_3	β_4	β_5
1%	0.9995	0.9995	0.9995	0.9921	0.9995
2%	0.9995	0.9921	0.9921	0.9995	0.9921
3%	0.9921	0.9995	0.8938	0.9921	0.9921
5%	0.9610	0.5560	0.7942	0.7942	0.9610
10%	0.5560	0.9610	0.6766	0.4431	0.4431
25%	0.0003	0.0000	0.0006	0.0001	0.0030

4.2.3 Outliers and good leverage points

Contamination level = 1%, $c = 12.15$, $h = 0.980$, $g = 0.985$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.963_{(0.055)}$	$1.963_{(0.021)}$	$-2.902_{(0.030)}$	$3.915_{(0.026)}$	$-4.893_{(0.027)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.002_{(0.015)}$	$1.999_{(0.008)}$	$-2.987_{(0.006)}$	$4.005_{(0.007)}$	$-4.991_{(0.008)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.997_{(0.015)}$	$1.986_{(0.024)}$	$-2.983_{(0.014)}$	$3.997_{(0.020)}$	$-4.977_{(0.024)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.002_{(0.016)}$	$1.996_{(0.009)}$	$-2.986_{(0.006)}$	$4.005_{(0.007)}$	$-4.990_{(0.008)}$
Contamination level = 2%, $c = 9.85$, $h = 0.968$, $g = 0.975$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.892_{(0.101)}$	$1.965_{(0.012)}$	$-2.946_{(0.012)}$	$3.920_{(0.015)}$	$-4.908_{(0.017)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.996_{(0.018)}$	$2.009_{(0.005)}$	$-2.994_{(0.004)}$	$3.990_{(0.004)}$	$-4.997_{(0.004)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.991_{(0.018)}$	$1.997_{(0.023)}$	$-2.985_{(0.017)}$	$3.983_{(0.019)}$	$-4.970_{(0.022)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.999_{(0.020)}$	$2.011_{(0.005)}$	$-2.994_{(0.004)}$	$3.989_{(0.004)}$	$-4.996_{(0.004)}$
Contamination level = 3%, $c = 7.55$, $h = 0.954$, $g = 0.965$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.798_{(0.184)}$	$1.955_{(0.007)}$	$-2.942_{(0.008)}$	$3.938_{(0.009)}$	$-4.896_{(0.018)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.008_{(0.020)}$	$2.000_{(0.002)}$	$-2.991_{(0.002)}$	$4.005_{(0.002)}$	$-5.000_{(0.002)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.998_{(0.020)}$	$2.023_{(0.017)}$	$-2.978_{(0.017)}$	$4.010_{(0.018)}$	$-4.994_{(0.017)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.003_{(0.021)}$	$1.999_{(0.002)}$	$-2.991_{(0.002)}$	$4.004_{(0.002)}$	$-5.001_{(0.002)}$
Contamination level = 5%, $c = 5.15$, $h = 0.928$, $g = 0.945$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.688_{(0.260)}$	$1.956_{(0.006)}$	$-2.947_{(0.005)}$	$3.932_{(0.008)}$	$-4.916_{(0.009)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.021_{(0.017)}$	$1.999_{(0.001)}$	$-2.997_{(0.001)}$	$4.002_{(0.001)}$	$-4.997_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.017_{(0.018)}$	$2.018_{(0.017)}$	$-2.979_{(0.020)}$	$3.982_{(0.017)}$	$-4.975_{(0.019)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.029_{(0.023)}$	$2.000_{(0.001)}$	$-2.999_{(0.001)}$	$4.002_{(0.001)}$	$-4.998_{(0.001)}$
Contamination level = 10%, $c = 2.92$, $h = 0.863$, $g = 0.895$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.470_{(0.731)}$	$1.969_{(0.002)}$	$-2.950_{(0.004)}$	$3.936_{(0.005)}$	$-4.921_{(0.008)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.012_{(0.025)}$	$1.996_{(0.001)}$	$-2.998_{(0.001)}$	$4.001_{(0.001)}$	$-5.003_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.006_{(0.043)}$	$1.973_{(0.057)}$	$-2.945_{(0.072)}$	$3.987_{(0.059)}$	$-4.982_{(0.054)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.005_{(0.042)}$	$1.999_{(0.001)}$	$-2.998_{(0.001)}$	$3.998_{(0.001)}$	$-5.002_{(0.001)}$
Contamination level = 25%, $c = 1.55$, $h = 0.669$, $g = 0.745$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$-0.438_{(3.081)}$	$1.969_{(0.001)}$	$-2.956_{(0.003)}$	$3.936_{(0.005)}$	$-4.922_{(0.007)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.992_{(0.039)}$	$1.999_{(0.000)}$	$-3.003_{(0.000)}$	$3.998_{(0.000)}$	$-4.999_{(0.000)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.962_{(0.207)}$	$1.840_{(2.895)}$	$-3.037_{(0.987)}$	$4.034_{(0.590)}$	$-5.036_{(1.953)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.085_{(0.473)}$	$1.964_{(0.137)}$	$-2.974_{(0.223)}$	$4.024_{(0.066)}$	$-5.045_{(0.116)}$

In the third scenario, we have contaminated the data by outliers, and for every outlier we have included one good leverage point. This is the type of data on which the S-weighted estimator outperformed the S-estimator in past simulations. Let us see whether our results confirm this. Beginning with a general overview, we can say that all methods, even OLS, have returned correct estimates, except, in case of OLS, the intercept. For some reason the good leverage points attract the regression plane strongly enough to enforce the correct slope estimates but not the correct intercept. The robust methods do not suffer from this nearly as much, but we can notice in case of LWS and S-weighted estimator that the MSE on the intercept is higher than on the other coefficients. This is not the case with S-estimator, which seems to be having no more problem with the intercept than with the slope. In all contamination levels, the S-estimator has a significantly higher MSE than the other two robust methods, which is one of the results we were curious to see confirmed. LWS and S-weighted estimator confirm their ability to utilize the information in good leverage points, while the S-estimator fails to do so. This is also the reason why S-estimator's MSE is of the same magnitude for all five coefficients: the leverage points helped OLS, LWS and S-weighted estimator in estimating the right values of slope estimates, but not with the intercept, where their MSE is mostly comparable to that of S-estimator. As to the comparison between LWS and S-weighted estimator, the situation is similar to the previous scenarios: at contamination levels up to 5%, they are basically the same, at 10% LWS becomes slightly better, and it is significantly better at 25%. Again, note the p-values of Kolmogorov-Smirnov test in Table 4.4.

Table 4.4: Kolmogorov-Smirnov test, scenario 3

	β_1	β_2	β_3	β_4	β_5
1%	1.0000	0.9995	1.0000	0.9921	1.0000
2%	1.0000	1.0000	0.9921	1.0000	1.0000
3%	0.7942	0.9610	0.9995	0.9610	0.9921
5%	0.6766	0.7942	0.9921	0.8938	0.6766
10%	0.5560	0.1400	0.5560	0.3439	0.6766
25%	0.0470	0.0000	0.0002	0.0002	0.0000

4.2.4 Good and bad leverage points

Contamination level = 1%, $c = 12.15$, $h = 0.980$, $g = 0.985$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.835_{(0.500)}$	$0.203_{(7.456)}$	$-0.420_{(12.459)}$	$0.673_{(17.128)}$	$-0.854_{(25.364)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.997_{(0.020)}$	$1.981_{(0.009)}$	$-2.991_{(0.008)}$	$3.997_{(0.007)}$	$-5.001_{(0.007)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.999_{(0.019)}$	$1.978_{(0.019)}$	$-2.974_{(0.021)}$	$3.991_{(0.020)}$	$-4.991_{(0.015)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.997_{(0.021)}$	$1.980_{(0.009)}$	$-2.992_{(0.008)}$	$3.998_{(0.007)}$	$-5.001_{(0.007)}$
Contamination level = 2%, $c = 9.85$, $h = 0.968$, $g = 0.975$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.876_{(1.230)}$	$-0.000_{(6.805)}$	$-0.322_{(10.351)}$	$0.561_{(15.078)}$	$-0.128_{(27.265)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.986_{(0.024)}$	$2.003_{(0.003)}$	$-2.995_{(0.003)}$	$4.000_{(0.003)}$	$-5.002_{(0.004)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.987_{(0.024)}$	$1.976_{(0.018)}$	$-2.986_{(0.020)}$	$3.978_{(0.015)}$	$-4.993_{(0.022)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.990_{(0.023)}$	$2.003_{(0.004)}$	$-2.995_{(0.003)}$	$3.998_{(0.004)}$	$-5.004_{(0.004)}$
Contamination level = 3%, $c = 7.55$, $h = 0.954$, $g = 0.965$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.155_{(2.084)}$	$-0.100_{(6.777)}$	$0.061_{(11.783)}$	$0.074_{(17.769)}$	$-0.416_{(23.776)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.031_{(0.018)}$	$2.004_{(0.002)}$	$-2.999_{(0.002)}$	$4.002_{(0.002)}$	$-5.003_{(0.002)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.028_{(0.018)}$	$2.024_{(0.018)}$	$-3.006_{(0.019)}$	$3.992_{(0.023)}$	$-4.998_{(0.018)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.030_{(0.020)}$	$2.003_{(0.002)}$	$-2.998_{(0.002)}$	$4.003_{(0.002)}$	$-5.003_{(0.002)}$
Contamination level = 5%, $c = 5.15$, $h = 0.928$, $g = 0.945$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.631_{(3.371)}$	$0.011_{(5.218)}$	$-0.226_{(8.841)}$	$0.034_{(17.065)}$	$-0.195_{(24.632)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.991_{(0.021)}$	$1.996_{(0.002)}$	$-3.002_{(0.001)}$	$4.000_{(0.001)}$	$-4.997_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.996_{(0.021)}$	$2.003_{(0.024)}$	$-3.006_{(0.022)}$	$3.988_{(0.024)}$	$-5.021_{(0.026)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.989_{(0.028)}$	$1.996_{(0.002)}$	$-3.000_{(0.001)}$	$4.001_{(0.001)}$	$-4.995_{(0.001)}$
Contamination level = 10%, $c = 2.92$, $h = 0.863$, $g = 0.895$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.988_{(7.269)}$	$0.020_{(4.661)}$	$0.106_{(10.336)}$	$0.053_{(16.291)}$	$-0.179_{(23.906)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.986_{(0.023)}$	$2.002_{(0.000)}$	$-3.001_{(0.001)}$	$3.999_{(0.001)}$	$-4.998_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.994_{(0.031)}$	$2.014_{(0.040)}$	$-3.024_{(0.054)}$	$4.016_{(0.043)}$	$-5.012_{(0.050)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.000_{(0.039)}$	$1.999_{(0.001)}$	$-2.996_{(0.002)}$	$4.000_{(0.001)}$	$-4.996_{(0.001)}$
Contamination level = 25%, $c = 1.55$, $h = 0.669$, $g = 0.745$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.581_{(21.526)}$	$0.052_{(4.084)}$	$0.012_{(9.348)}$	$-0.008_{(16.295)}$	$-0.000_{(25.408)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.959_{(0.041)}$	$1.999_{(0.000)}$	$-2.999_{(0.000)}$	$4.000_{(0.000)}$	$-5.000_{(0.000)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.947_{(0.198)}$	$2.084_{(0.351)}$	$-3.012_{(0.479)}$	$4.055_{(0.376)}$	$-4.878_{(0.393)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.715_{(4.275)}$	$1.767_{(0.878)}$	$-2.754_{(1.731)}$	$3.682_{(2.910)}$	$-4.560_{(5.156)}$

Having contaminated the data with bad leverage points, and at the same time countering their effect with good leverage points, we are interested to see how well the estimators can ignore the former and pay heed to the latter. Unsurprisingly, we observe that the OLS estimator can not do the first. As the contamination level rises, the effects of the good and bad leverage points cancel out and the slope estimates of OLS tend to zero. The intercept seems unaffected by leverage points much like in the previous scenarios, and therefore here the OLS estimates remain more or less correct. LWS and S-weighted estimator, on the other hand, have successfully distinguished the good from the bad and report correct values of all estimates with very small MSE. The exception again being the 25% level of contamination, where S-weighted estimator performs worse than LWS both in higher MSE and in having the average estimate farther away from the correct value. Kolmogorov-Smirnov test confirms this description. The S-estimator returns the correct values of estimates for all contamination levels, although its MSE is generally higher. Consistent with Víšek (2016b), we attribute this tho the fact that S-estimator disregards any leverage points, good or bad, and therefore obtains good results but with higher MSE.

Table 4.5: Kolmogorov-Smirnov test, scenario 4

	β_1	β_2	β_3	β_4	β_5
1%	0.9921	1.0000	0.9921	1.0000	0.9995
2%	0.9921	0.9995	1.0000	0.9995	1.0000
3%	0.9610	0.9921	0.9921	0.9610	0.9921
5%	0.9610	0.9921	0.8938	0.9610	0.7942
10%	0.1400	0.4431	0.1930	0.3439	0.1400
25%	0.0002	0.0000	0.0000	0.0000	0.0000

4.2.5 Outliers and good and bad leverage points

Contamination level = 1%, $c = 12.15$, $h = 0.980$, $g = 0.985$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.039_{(0.304)}$	$1.418_{(6.570)}$	$-1.880_{(6.181)}$	$2.299_{(8.454)}$	$-2.414_{(13.433)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.994_{(0.019)}$	$1.999_{(0.008)}$	$-2.989_{(0.008)}$	$4.005_{(0.007)}$	$-5.011_{(0.008)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.992_{(0.018)}$	$1.976_{(0.023)}$	$-2.997_{(0.015)}$	$3.988_{(0.017)}$	$-5.000_{(0.018)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.995_{(0.019)}$	$2.005_{(0.009)}$	$-2.988_{(0.009)}$	$4.009_{(0.006)}$	$-5.008_{(0.007)}$
Contamination level = 2%, $c = 9.85$, $h = 0.968$, $g = 0.975$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$1.160_{(1.184)}$	$0.908_{(4.806)}$	$-1.239_{(6.966)}$	$1.255_{(12.333)}$	$-1.714_{(15.416)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.994_{(0.026)}$	$1.991_{(0.003)}$	$-3.004_{(0.003)}$	$3.991_{(0.003)}$	$-5.001_{(0.002)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.996_{(0.024)}$	$1.984_{(0.025)}$	$-2.967_{(0.020)}$	$3.979_{(0.018)}$	$-5.003_{(0.023)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.993_{(0.027)}$	$1.990_{(0.003)}$	$-3.004_{(0.003)}$	$3.991_{(0.003)}$	$-5.000_{(0.002)}$
Contamination level = 3%, $c = 7.55$, $h = 0.954$, $g = 0.965$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.901_{(1.317)}$	$0.905_{(3.822)}$	$-1.007_{(6.600)}$	$1.584_{(8.132)}$	$-2.063_{(11.525)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$1.009_{(0.022)}$	$2.001_{(0.003)}$	$-3.002_{(0.002)}$	$4.001_{(0.002)}$	$-4.999_{(0.003)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.009_{(0.022)}$	$2.016_{(0.023)}$	$-3.003_{(0.022)}$	$3.993_{(0.022)}$	$-4.996_{(0.023)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.012_{(0.022)}$	$2.001_{(0.003)}$	$-3.003_{(0.002)}$	$4.001_{(0.002)}$	$-5.000_{(0.003)}$
Contamination level = 5%, $c = 5.15$, $h = 0.928$, $g = 0.945$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.875_{(2.885)}$	$0.778_{(2.979)}$	$-1.120_{(5.100)}$	$1.280_{(9.350)}$	$-1.629_{(13.080)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.990_{(0.025)}$	$2.000_{(0.001)}$	$-2.996_{(0.002)}$	$4.000_{(0.001)}$	$-4.997_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.994_{(0.020)}$	$1.985_{(0.018)}$	$-3.010_{(0.025)}$	$4.020_{(0.023)}$	$-4.971_{(0.024)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.990_{(0.027)}$	$2.003_{(0.001)}$	$-2.995_{(0.002)}$	$3.998_{(0.001)}$	$-4.997_{(0.001)}$
Contamination level = 10%, $c = 2.92$, $h = 0.863$, $g = 0.895$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$0.553_{(5.715)}$	$0.619_{(2.665)}$	$-0.911_{(5.131)}$	$1.320_{(8.031)}$	$-1.728_{(11.548)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.994_{(0.019)}$	$1.996_{(0.001)}$	$-2.998_{(0.001)}$	$3.998_{(0.001)}$	$-5.005_{(0.001)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$0.990_{(0.036)}$	$2.000_{(0.042)}$	$-3.022_{(0.045)}$	$3.990_{(0.042)}$	$-5.048_{(0.039)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$0.981_{(0.043)}$	$1.997_{(0.001)}$	$-3.000_{(0.001)}$	$3.998_{(0.001)}$	$-5.004_{(0.001)}$
Contamination level = 25%, $c = 1.55$, $h = 0.669$, $g = 0.745$					
$\hat{\beta}^{(OLS)}_{(MSE)}$	$-0.482_{(16.238)}$	$0.729_{(1.924)}$	$-0.994_{(4.257)}$	$1.387_{(7.131)}$	$-1.657_{(11.518)}$
$\hat{\beta}^{(LWS)}_{(MSE)}$	$0.988_{(0.036)}$	$2.002_{(0.000)}$	$-2.997_{(0.000)}$	$3.999_{(0.000)}$	$-5.000_{(0.000)}$
$\hat{\beta}^{(S)}_{(MSE)}$	$1.110_{(0.463)}$	$2.039_{(0.359)}$	$-2.983_{(0.384)}$	$3.949_{(0.352)}$	$-4.870_{(0.386)}$
$\hat{\beta}^{(SW)}_{(MSE)}$	$1.035_{(0.420)}$	$1.918_{(0.410)}$	$-2.988_{(0.232)}$	$3.903_{(0.515)}$	$-5.127_{(0.720)}$

In the final contamination scenario, we include everything. Outliers, bad leverage points, and good leverage points. Again starting with OLS, we can see that its estimates have decreased in absolute value, much like they did in the previous scenario, but they retain the correct sign. This may be explained by there being as many good leverage points as bad leverage points and outliers *combined*. Thus the effect of the good leverage points has been stronger. Otherwise this scenario brings nothing new compared to the previous ones. LWS and S-weighted estimator bring excellent results for up to 10% contamination, and in case of 25% the LWS performs better. The S-estimator gives good results throughout the different contamination levels, but has systematically higher MSE than LWS, and, except for the 25% case, also than S-weighted estimator.

Table 4.6: Kolmogorov-Smirnov test, scenario 5

	β_1	β_2	β_3	β_4	β_5
1%	1.0000	0.9921	0.9921	0.9995	0.9921
2%	0.9610	0.9921	0.9995	0.9921	0.9610
3%	0.9610	0.9995	0.9921	0.9995	0.9610
5%	0.8938	0.6766	0.9921	0.9610	0.8938
10%	0.2606	0.1930	0.3439	0.6766	0.0994
25%	0.0030	0.0000	0.0000	0.0000	0.0000

4.2.6 Recapitulation

Let us briefly summarize our methodology and our findings in this simulation study. We have generated $m = 100$ samples of data for each type and level of contamination. Considered were five types and six levels of contamination, that is altogether thirty cases, each one simulated one hundred times, which gives us 3000 datasets. On each dataset, we estimated the standard linear regression model with intercept and obtained estimates by means of OLS, LWS S-estimator and S-weighted estimator. The estimates belonging to the same type and level of contamination were then summarized by means of their arithmetic average and mean squared error. Since we were most interested in the difference between LWS and S-weighted estimates, we performed the Kolmogorov-Smirnov test on the null hypothesis that these two estimates are from the same distribution.

There has been a number of observations we have made on the results. Beginning with the OLS method, we saw that it indeed is not robust to any kind of contamination, least of all bad leverage points. Good leverage points,

on the other hand, were able to force OLS to estimate the correct values, simply by virtue of not being able to resist any influential observations. The robust methods did generally perform well in contamination level up to 10%, returning approximately the correct values of estimators and maintaining small MSE. The S-estimator and S-weighted estimator usually had worse results in the 25% contamination case. This might be because of too high breakdown point which we have set to be twice the contamination, meaning in this case 50%, so the two estimators have perhaps sacrificed too much efficiency. In none of the scenarios have we found a case of the S-weighted estimator outperforming the LWS. In most scenarios they reported essentially the same results up to 5% contamination, on 25% the S-weighted estimator performed worse than LWS. As to the issue of good leverage points, LWS and S-weighted estimator did prove to be better suited to utilize their information and obtain stronger results than S-estimator. This effect only took place in the case of slope estimates, the good leverage points seemed to provide no help in estimating the intercept.

Generally speaking, we have observed in the behavior of the S-weighted estimator similarities with both LWS (mainly for smaller contamination levels and in their ability to use good leverage points to their advantage) and with S-estimator (mainly in the higher levels of contamination, where both estimators seemed to be having more trouble than LWS.) This is consistent with the S-weighted estimator being defined as a combination of the two, therefore inheriting properties from each. Overall however, we must say that LWS appeared to be the most reliable and efficient of the three estimators.

Chapter 5

Implementation of S-weighted estimator in Python

We have remarked earlier that the main practical problem with S-weighted estimator is lack of available implementation, or even a description of an algorithm by which the estimator is to be calculated. That is why we decided to include in this thesis our own implementation of the estimator. We have selected Python to be the programming language in which our implementation is to be taking place. The code can be found in Appendix B and the author encourages anybody to use it.

To see in detail what the program does, let us first overview the code from first line to the last. After that, it will be easy to explain the program as a whole. We will be citing parts from it here and interjecting our comments. For the whole code, refer to the above mentioned Appendix B. We begin by importing the necessary packages.

```
import numpy as np
import math
from random import seed
from random import gauss
from random import randint
from random import sample
from random import shuffle
```

The program uses `numpy` extensively, as most of the commands are operations with arrays. After that, the next five lines are not absolutely necessary for the function, they are included in order for the accompanying example code

to be functional. The only function imported here that appears in the main part of the program is `shuffle` from the `random` package. After importing the packages, we start defining the main function, `SW`, which contains the whole rest of the program.

```
def SW(X,Y,w,c,max_inner,max_outer,max_match):
```

The entire estimator is implemented as a function which returns nothing but the estimated regression coefficients. Let us take a look at the arguments. The function takes the matrix of explanatory variables `X`, the vector with the explained variable `Y`, and the vector of weights `w`, all as `numpy` arrays. This allows the user to define the weight function according to their preferences. Our sample example code provides a function that calculates vector `w` based od Tukey's ρ function, as we did in the simulation. It takes as arguments only the constants h and g and the sample size n.

We do not, however allow the user to choose the objective function, as we have coded Tukey's ρ function into the workings of the program. The user only calibrates it by selecting the constant c. In this, the previous chapters of this thesis can be their guide. The remaining three arguments are the constants that determine the maximum number of repetitions of the algorithm's cycles. The inner cycle is stopped after `max_inner` repetitions, or after it stops finding better models, whichever comes first. The outer cycle is stopped after `max_outer` repetitions or when the same best model has been found `max_match` times. Needless to say that these last three arguments should be positive integers. Experience suggests that setting all three of them to 20 brings reliable results.

Inside the main function, we begin by defining a number of inner functions that make the the workings of the algorithm easier to understand and the whole code more legible. We will briefly comment on each one in turn. They themselves follow no particular order and could be ordered differently in the code.

```
    def select_p(X,Y,p,n):
        sez = [i for i in range(n)]
        shuffle(sez)
        SI  = [i < p for i in sez]
        X_p = X[SI]
        Y_p = Y[SI]
        return [X_p, Y_p]
```

We already know from previous chapters that each repetition of the outer cycle begins with selecting p observations randomly. Function `select_p()` does just that. It takes as arguments the dataset `X`, `Y`, and its dimensions `p` and `n`, that the outer function determines automatically. The function's outputs are arrays `X` and `Y` with all but p rows omitted.

```
def OLS(X,Y):
    # b_OLS = (X'X)^(-1)X'Y
    Xt = X.transpose()
    B  = np.matmul(Xt,X)
    B  = np.linalg.inv(B)
    B  = np.matmul(B,Xt)
    B  = np.matmul(B,Y)
    return B
```

Function `OLS()` requires hardly any explanation. Although we had to use `np.matmul()` successively, so the computation process is not as obvious as it would be for example in Matlab, it is still clear that what we are doing here is to use the well known formula $(X'X)^{-1}X'y$ by which the OLS estimate is obtained. Therefore the function returns a single array of dimension p, which is the OLS estimate of the regression coefficients.

```
    def objective(r,c):
    m = r.shape[0]
    C = [c for i in range(m)]
    d = np.array([min(r[i], C[i]) for i in range(m)])
    rho = (d**2)/2 - (d**4)/(2*(c**2)) + (d**6)/(6*(c**4))
    return rho
```

Function `objective()` takes a vector of points `r` and calculates the values of Tukey's ρ function with parameter `c` at those points. Normally we would put just a one-dimensional arguent in Tukey's ρ, but here it is convenient to call the function just once for the whole vector of residuals and receive all the values in one vector.

```
 def K_estim(c):
    n = 1000
    K = np.random.normal(loc=0, scale=1, size=n)
    K = objective(K,c)
```

```
    K = sum(K)/n
    return K
```

This function, `K_estim`, estimates the right-hand side of the equality in 3.1. We have said that it can be calculated analytically, the formula can be found in Campbell *et al.* (1998), but that is not necessary for this purpose. Here we estimate it on a random sample of size 1000 from standard normal distribution, which gives sufficiently close results. This is a potential opportunity for further development, since the direct calculation might make the algorithm faster. The output of this function is then used by function `S_estim` which we shall describe now.

```
def S_estim(n,w,c,r_s,K):
    S_small = 0
    S_big   = 10000
    r_ss    = r_s**(0.5)
    summa   = 0
    while abs(summa - K) > 0.01:
        S = (S_small + S_big)/2
        r = r_ss/S
        rho_r = objective(r,c)
        summa = np.dot(w,rho_r)
        if summa > K:
            S_small = S
        if summa < K:
            S_big = S
    return S
```

In this function, `S_estim`, we calculate the σ from 3.1 which is to be minimized. Therefore, the output of this function is that quantity by which the algorithm assesses the just estimated model's worth. Let us inspect the logic of this function in somewhat more detail. Recall that in 3.1, the nature of function ρ prevents the equality from being solved for constant σ and then simply calculated. Instead, we have to try different σ's until we find one for which the left-hand side of the equality is sufficiently close to the right-hand side. Recalling also the shape of function ρ, and the fact that σ is in the denominator, we see that in normal circumstances, the left-hand side is decreasing in σ. From now on, we shall be referring to the left-hand side as `summa`, since that is the name given to it in the code.

At the beginning of our approximation, we set two values of potential σ, `S_small` and `S_big`. The algorithm needs `S_big` to be higher than the σ we are looking for, so for some purposes our value of 10000 might not be sufficient, but we could not use infinity. In the next step, we take the average between `S_big` and `S_small` and calculate `summa`. If `summa` is larger than the right-hand side value `K`, then we need a larger σ. That means that the σ we just used is too small, and so we can use its value to be the new `S_small`. Then we try the arithmetic average between `S_big` and the new `S_small` again, and so we iterate until we find the value of σ which is close enough to the solution of the equality in 3.1. It sounds complicated, but since each step divides the distance by 2, the algorithm can find an appropriate σ rather quickly.

```
def WLS(X,Y,w):
    # b_WLS = (X'WX)^(-1)X'WY
    W  = np.diag(w)
    Xt = X.transpose()
    B  = np.matmul(Xt,W)
    B  = np.matmul(B,X)
    B  = np.linalg.inv(B)
    B  = np.matmul(B,Xt)
    B  = np.matmul(B,W)
    B  = np.matmul(B,Y)
    return B
```

Function `WLS()` takes the dataset `X` and `Y`, and a vector of weights and calculates the weighted least squares estimate. The output value is the array of estimates of dimension p.

```
def ordered_r(X,Y,b):
    r = (Y - np.matmul(X,b))**2
    r_s = np.sort(r)
    order = np.argsort(r)
    return [r_s, order]
```

Function `ordered_r()` takes the data and a vector of coefficients obtained from previous regression and calculates the squared residuals from that regression. Then it reorders them from smallest to largest and returns a list of two arrays. First is the vector of ordered squared residuals, and second, the way in

which they were reordered. The last is necessary later when we need to reorder the whole dataset in the same way the residuals were reordered.

```
def inner(X,Y,w,p,n,K,c,max_inner):
    S_past = float('inf')
    X_p, Y_p = select_p(X,Y,p,n)
    b = OLS(X_p, Y_p)
    r_s, order = ordered_r(X,Y,b)
    X_s, Y_s = X, Y
    S_present = S_estim(n,w,c,r_s,K)
    while S_present < S_past and max_inner > 0:
        S_past = S_present
        X_s, Y_s = X_s[order], Y_s[order]
        b = WLS(X_s, Y_s, w)
        r_s, order = ordered_r(X_s,Y_s,b)
        S_present = S_estim(n,w,c,r_s,K)
        max_inner = max_inner - 1
    return [b, S_present]
```

Finally we get to the last inner function. It is the one which calls all the other functions that we have so far discussed, and performs the inner cycle of the algorithm. As arguments, it is given the data `X`, `Y`, array of weights `w`, dimensions of the data `p`, `n`, the right-hand side `K` of the equality in 3.1, constant `c` for Tukey's ρ function, and the stopping parameter `max_inner`.

First we assign infinity as the value of `S_past`, since later the cycle will iterate as long as it keeps finding models with `S_present` $<$ `S_past`. Then we select p observations at random and fit a regression plane exactly through them by `OLS()`. We use the estimate from this regression to calculate the residuals, reorder them and save the information on how we reordered them. Then, by `S_estim()`, we calculate the σ from this regression and we enter the loop. Inside the loop, we first save the latest value of σ since that is what we will be comparing the next value of σ with. We reorder the observations in the same way we reordered the residuals and we estimate a new regression model on the ordered data by means of `WLS()`. Here we can see the significance of reordering the data, because WLS requires an extrinsic rule for weighting the observations. Reordering ensures that the smallest weights are assigned to the largest squared residuals. As we have discussed in Chapter 3, a different

choice of estimator in this particular step cannot spoil the results of the algorithm as long as `max_outer` is sufficiently high, but it can make the algorithm slower. Employing a weighted M_ρ-estimator here might therefore improve the algorithm's performance. After obtaining the new estimates, we calculate the new ordered squared residuals, estimate the value of σ this model results in, decrease the number of remaining repetitions of the loop by 1 and proceed to repeat the loop. When the repetitions stop bringing models better in terms of σ or when the maximum number of repetitions of the inner cycle is reached, the function returns the best estimate found and the corresponding value of σ.

```
n = X.shape[0]
p = X.shape[1]
K = K_estim(c)
S_best = float('inf')
match = max_match
while match > 0 and max_outer > 0:
    max_outer = max_outer - 1
    [b, S_new] = inner(X,Y,w,p,n,K,c,max_inner)
    if S_new < S_best:
        S_best = S_new
        b_best = b
        match = max_match
    if S_new == S_best and sum(b - b_best) == 0:
        match = match - 1
return b_best
```

It remains now to discuss the outer function, which contains the outer cycle of the algorithm and calls function `inner()`. First the function establishes the dimensions of the data and estimates constant `K`. We preallocate infinity as the value of `S_best`, so that the first model we obtain from the inner cycle is accepted as an improvement. Then we step inside the outer cycle. This cycle is repeated maximum `max_outer` times or until we have found the same best model `max_match` times, whichever comes first.

In the outer cycle, we decrease `max_outer` by 1 and call function `inner()` to provide us with some model and its σ value. If the value of σ is worse than the best one yet, nothing happens and we try again. If it is the same as the best one yet and the model is the same as the best one obtained, we decrease `match`

by 1, `match` being the number of times the best model has yet to reappear in order for us to decide that it is the really best model. If the value of σ is the smallest one yet, it means that we have found a new best model, so we save the estimate and the σ value and reset `match` to `max_match`, meaning that we have to find the same model another `match` times in order to believe that it is the best model there is. When the algorithm performs all the repetitions of the outer cycle, it stops and returns the estimated values of the regression coefficients.

We have seen that there are opportunities for improvement of this implementation. Yet we must stress that, to the best of the present authors knowledge, no other implementations are publicly available and by publishing this one, imperfect as it is, we are attempting to make the use of the S-weighted estimator one step easier.

Chapter 6

Conclusion

In this thesis we focused on the newly developed S-weighted estimator in three crucial ways: first summarizing the existing literature on the subject and providing historical context, then putting the estimator to use and comparing its performance with that of other estimators, and finally implementing the algorithm by which the estimator can be calculated.

In the first section, we briefly surveyed the history of robust estimation methods and explained the terminology that the field uses. We summarized the early attempts at robustifying regression estimation and mentioned two early approaches, Huber's minimax method and the robustified likelihood ratio approach by the same author. Then we studied Hampel's approach based on influence functions and paid some attention to the terms defined therein, most notably the notion of breakdown point which has become the center of focus of robust statisticians.

In particular detail were studied two relatively more successful methods, the least weighted squares developed by Víšek (2000) and the S-estimator proposed by Rousseeuw & Yohai (1984). The logic of their definitions were at the center of our attention, because it is inherited by the S-weighted estimator which is defined as their combination and intended as a unification that subsumes the former two, and by extension even some other, older methods, as its special cases.

In Chapter 3, we summarized the available literature on S-weighted estimator, focusing primarily on two subjects. Fist, the connection between the estimator's definition and the algorithm by which it is to be calculated. Here we had to rely to some extent on unofficial sources, since the explicit description of the algorithm could not be found in published literature. Second, we

summarized the up to now performed simulation studies and found a lack of a study employing side-by-side the S-weighted estimator and its two immediate predecessors, the LWS and the S-estimator.

In Chapter 4 we performed a simulation study wherein four estimators, namely OLS, LWS, S- and S-weighted estimator, were employed to estimate the standard linear regression model with contaminated data, and the quality of their results was compared. Each on 1%, 2%, 3%, 5%, 10%, and 25% levels of intensity, the following types of contamination were considered:

Table 6.1: Types of contamination

Scenario 1	Only outliers
Scenario 2	Only bad leverage points
Scenario 3	Outliers and good leverage points
Scenario 4	Good and bad leverage points
Scenario 5	Outliers and good and bad leverage points

By comparing the average value of the obtained estimates and their mean squared error, and by testing whether the estimates of LWS and S-weighted estimator were from the same distribution by means of Kolmogorov-Smirnov test, we came to the following conclusions.

1. The OLS estimator is not robust against any kind of contamination and suffers the most from contamination by bad leverage points. It performs somewhat adequately in presence of good leverage points because they force the method to return the proper results. In combination of good and bad leverage points, the method gives hardly any results, because the effects of good and bad leverage points cancel each other out.

2. All robust methods performed adequately in all scenarios up to 10% contamination. S- and S-weighted estimators experienced higher mean squared error in case of 25% contamination, which we attributed to calibration for overly high breakdown point.

3. The estimates of LWS and S-weighted estimator were indistinguishable for contamination up to 5%. For higher levels, especially 25%, LWS attained significantly smaller mean squared error and more exact average of estimates.

4. The S-weighted estimator behaved similarly to LWS in small contamination and to S-estimator in high contamination, confirming that both S- and LWS-elements of are present and active in its functioning.

5. Overall, the LWS method appeared to be the most reliable and efficient of the three.

We hope to have shed some light on the working and merits of these particular robust methods and provided a simple guide in how they are to be employed. This last was also our goal in providing an implementation of the S-weighted estimator in Python, which was the content of Chapter 5. Therein we explain the implementation step-by-step, commenting on every part of the code. The resulting function is reliable although not very fast. Parts that provide opportunity for improvement are indicated. Apart from facilitating the use of S-weighted estimator in practice, the chapter might also be useful in supplementing deeper understanding of the underlying algorithm that is being implemented. Future research might focus on the behavior of S-weighted estimator in higher levels of contamination and trying to find a better calibration of the weighting function and the objective function which might be able to improve the estimator's performance. The implementation in Python is of course very basic and constitutes only the first step in developing a versatile tool for the estimator's employment. Most notably, more research is needed for developing diagnostic tools that would allow not only estimation but also testing of the estimates' qualities.

Bibliography

Bartoszynski, R. & M. Niewiadomska-Bugaj (2008): *Probability and Statistical Inference*. Hoboken, N.J: Wiley-Interscience.

Bernoulli, D. (1777): "Dijudicatio maxime probabilis plurium observationum discrepantium atque verisimillima inductio inde formanda." *Actu Acud. Sci. Petropolit.* **1**: pp. 3–33.

Boček, P. & P. Lachout (1995): "Linear Programming Approach to LMS-Estimation." *Computational Statistics & Data Analysis* **19**: pp. 129–134.

Campbell, N. A., H. Lopuhaä, & P. Rousseeuw (1998): "Onthe calculation of a robust s-estimator of a covariance matrix." *Statistics in medicine* **17**: pp. 2685–2695.

Chauvenet, W. (1863): "Method of Least Squares." *Appendix to Manual of Spherical and Practicul Astronomy* **2**: pp. 469–566.

Fisher, R. A. (1920): "A mathematical examination of the methods of determining the accuracy of an observation by the mean error and by the mean square error." *Monthly Notices of the Royal Astronomical Society* **80**: pp. 469–566. Reprinted in Collected Papers of R. A. Fisher, ed. J. H. Bennett, Vol.1, 188-201, University of Adelaide.

Fisher, R. A. (1922): "On the mathematical foundations of theoretical statistics." *Philosophical Transactions of the Royal Society of London* **222**: pp. 309–368.

Hampel, F. R. (1968): "Contributions to the theory of robust estimation." *Ph.D. thesis. University of California, Berkeley* .

Hampel, F. R., E. M. Ronchetti, P. J. Rousseeuw, & W. A. Stahel (1986): *Robust statistics: the approach based on influence functions*. New York: Wiley.

Hodges, Jr., J. L. (1967): "Efficiency in normal samples and tolerance of extreme values for some estimates of location." In "Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability," 1, pp. 163–168.

Huber, P. J. (1964): "Robust estimation of a location parameter." *The Annals of Mathematical Statistics* **35**: pp. 73–101.

Huber, P. J. Strassen, V. (1973): "Minimax tests and the Neyman-Pearson lemma for capacities." *Annual Statistics* **1**: pp. 251–263. 2, 223-224.

Peirce, B. (1852): "Criterion for the rejection of doubtful observations." *Astronomical Journal* **2**: pp. 161–163.

Ronchetti, E. (1982): "Robust testing in linear models: The infinitesimal approach." *Ph.D. thesis. ETH, Zurich.* .

Rousseeuw, P. & V. Yohai (1984): "Robust Regression by Means of S-Estimators." In J. Franke, W. Härdle, & D. Martin (editors), "Robust and Nonlinear Time Series Analysis," pp. 256–272. New York, NY: Springer US.

Rousseeuw, P. J. (1984): "Least median of squares regression." *Journal of the American Statistical Association* **79**: pp. 871–880.

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

Siegel, A. F. (1982): "Robust regression using repeated medians." *Biometrika* **69**: pp. 242–244.

Student (1927): "Errors of routine analysis." *Biometrika* **19**: pp. 151–164.

Tukey, J. (1960): "A survey of sampling from contaminated distributions." In "Contributions to Probability and Statistics, I. Olkin (ed.)," pp. 448–485. Stanford University Press, Stanforn, Calif.

Víšek, J. A. (2000): "Regression with high breakdown point." *Proceedings of ROBUST 2000* pp. 324–356.

Víšek, J. A. (2012): *Advantages and disadvantages, challenges and threads of robust methods*. Faculty of Social Sciences, Charles University, Prague.

Víšek, J. A. (2015): "S-Weighted Estimators." *Proceedings of the 16th Conference on the Applied Stochastic Models, Data Analysis and Demographics 2015* pp. 1031–1042.

Víšek, J. A. (2016a): "Are the Bad Leverage Points the most Difficult Problem for Estimating the Underlying Regression Model?" `https://www.karlin.mff.cuni.cz/~antoch/robust16/prednasky/Nedele/visek.pdf`. Lecture notes from Robust 2016 (Jeseníky) international statistical conference. Online; accessed 19-July-2024.

Víšek, J. A. (2016b): "Coping with Level and Different Type of Contamination by SW-Estimator." In A. Colubi, A. Blanco, & C. Gatu (editors), "Proceedings of COMPSTAT 2016, 22nd International Conference on Computational Statistics," pp. 59–72.

Víšek, J. A. (2016c): "Representation of SW-estimators." *Proceedings of the 4th Stochastic Modeling Techniques and Data Analysis International Conference with Demographics Workshop, SMTDA 2016* **53**: pp. 425–438.

Víšek, J. A. (2017): "Instrumental Weighted Variables under Heteroscedasticity. Part II — Numerical study." *Kybernetika* **53**: pp. 26–58.

Víšek, J. A. (2019a): "Asymptotics of S-Weighted Estimators." *Springer Proceedings in Mathematics & Statistics* **274**: pp. 31–42.

Víšek, J. A. (2019b): "S-Weighted Instrumental Variables." *Data Analysis and Applications* **1**: pp. 53–71.

Appendix A

Algorithm for finding optimal c

```
function [c] = c_finder(breakdown)
% c_finder takes the required breakdown point of S-estimator and returns
% the value of c in Tukey's rho function that results in it

c_low  = 1.3;
c_high = 1000;

n = 10^5; %sample size. The bigger the n, the more exact the calculation

breakdown_point = 0;

function [output] = rho(x, c)
% rho - calculate value of Tukey's rho function with parameter c at point x
    x = abs(x);
    x = min([x,c]);
    output = (x^2)/2 - (x^4)/(2*(c^2)) + (x^6)/(6*(c^4));
end

while abs(breakdown - breakdown_point) > 0.001

    c = (c_low + c_high)/2;
    e = normrnd(0,1,[1,n]);

    for i = 1:n
        e(i) = rho(e(i),c);
    end

    for i = 1:n
        breakdown_point = breakdown_point + e(i);
    end

    breakdown_point = breakdown_point/n;
    breakdown_point = (6*breakdown_point)/(c*c);

    if breakdown_point < breakdown; c_high = c; end
    if breakdown_point > breakdown; c_low  = c; end

end
end
```

Appendix B

Implementation of S-weighted estimator in Python

```
import numpy as np
import math
from random import seed
from random import gauss
from random import randint
from random import sample
from random import shuffle

def SW(X,Y,w,c,max_inner,max_outer,max_match):

    def select_p(X,Y,p,n):
        sez = [i for i in range(n)]
        shuffle(sez)
        SI  = [i < p for i in sez]
        X_p = X[SI]
        Y_p = Y[SI]
        return [X_p, Y_p]

    def OLS(X,Y):
        # b_OLS = (X'X)^(-1)X'Y
        Xt = X.transpose()
        B  = np.matmul(Xt,X)
        B  = np.linalg.inv(B)
```

```
    B  = np.matmul(B,Xt)
    B  = np.matmul(B,Y)
    return B

def objective(r,c):
    C = [c for i in range(r.shape[0])]
    d = np.array([min(r[i], C[i]) for i in range(n)])
    rho = (d**2)/2 - (d**4)/(2*(c**2)) + (d**6)/(6*(c**4))
    return rho

def K_estim(c):
    n = 1000
    K = np.random.normal(loc=0, scale=1, size=n)
    K = objective(K,c)
    K = sum(K)/n
    return K

def S_estim(n,w,c,r_s,K):
    S_small = 0
    S_big   = 10000
    r_ss    = r_s**(0.5)
    summa   = 0
    while abs(summa - K) > 0.01:
        S = (S_small + S_big)/2
        r = r_ss/S
        rho_r = objective(r,c)
        summa = np.dot(w,rho_r)
        if summa > K:
            S_small = S
        if summa < K:
            S_big = S
    return S

def WLS(X,Y,w):
    # b_WLS = (X'WX)^(-1)X'WY
    W  = np.diag(w)
    Xt = X.transpose()
```

```
    B  = np.matmul(Xt,W)
    B  = np.matmul(B,X)
    B  = np.linalg.inv(B)
    B  = np.matmul(B,Xt)
    B  = np.matmul(B,W)
    B  = np.matmul(B,Y)
    return B

def ordered_r(X,Y,b):
    r = (Y - np.matmul(X,b))**2
    r_s = np.sort(r)
    order = np.argsort(r)
    return [r_s, order]

def inner(X,Y,w,p,n,K,c,max_inner):
    S_past = float('inf')
    X_p, Y_p = select_p(X,Y,p,n)
    b = OLS(X_p, Y_p)
    r_s, order = ordered_r(X,Y,b)
    X_s, Y_s = X, Y
    S_present = S_estim(n,w,c,r_s,K)
    while S_present < S_past and max_inner > 0:
        S_past = S_present
        X_s, Y_s = X_s[order], Y_s[order]
        b = WLS(X_s, Y_s, w)
        r_s, order = ordered_r(X_s,Y_s,b)
        S_present = S_estim(n,w,c,r_s,K)
        max_inner = max_inner - 1
    return [b, S_present]

n = X.shape[0]
p = X.shape[1]
K = K_estim(c)
S_best = float('inf')
match = max_match
while match > 0 and max_outer > 0:
    max_outer = max_outer - 1
```

```
        [b, S_new] = inner(X,Y,w,p,n,K,c,max_inner)
        if S_new < S_best:
            S_best = S_new
            b_best = b
            match = max_match
        if S_new == S_best and sum(b - b_best) == 0:
            match = match - 1
    return b_best
```

Example code that can be used to run the program:

```
# creating uncontaminated data
N = 500
p = 5
X = []
e = []
S = []
sigmae = 1
sigmax = 1
for i in range(N):
    e.append(gauss(0, sigmae))
beta = [randint(-5,5) for i in range(p)]
ones = [1 for i in range(N)]
X.append(ones)
for i in range(p-1):
    newcol = []
    for j in range(N):
        newcol.append(gauss(0, sigmax))
    X.append(newcol)
X = np.array(X)
X = np.transpose(X)
Y = np.matmul(X, beta) + e

def TukeyWeights(n,h,g):
    g = min(1,g)
    h = min(0.999*g, h)
    c = g-h
```

```
    cc = c*c
    cccc = cc*cc
    w = [0 for i in range(n)]
    for i in range(n):
        x = (i-1)/n
        if x < h:
            w[i] = 1
        elif x <= g:
            y = g-x
            w[i] = 6*((y**2)/2-(y**4)/(2*cc)+(y**6)/cccc)/cc
        else:
            w[i] = 0
    return np.array(w)

c = 5
h = 0.8
g = 0.9
w = TukeyWeights(N,h,g)
max_inner = 20
max_outer = 20
max_match = 20
betaSW = SW(X,Y,w,c,max_inner,max_outer,max_match)
print(beta)
print(betaSW)
```