robslopes: Efficient Computation of the (Repeated) Median Slope

by Jakob Raymaekers

Abstract Modern use of slope estimation often involves the (repeated) estimation of a large number of slopes on a large number of data points. Some of the most popular non-parametric and robust alternatives to the least squares estimator are the Theil-Sen and Siegel's repeated median slope estimators. The **robslopes** package contains fast algorithms for these slope estimators. The implemented randomized algorithms run in $\mathcal{O}(n\log(n))$ and $\mathcal{O}(n\log^2(n))$ expected time respectively and use $\mathcal{O}(n)$ space. They achieve speedups up to a factor 10^3 compared with existing implementations for common sample sizes, as illustrated in a benchmark study, and they allow for the possibility of estimating the slopes on samples of size 10^5 and larger thanks to the limited space usage. Finally, the original algorithms are adjusted in order to properly handle duplicate values in the data set.

1 Introduction

The Theil-Sen estimator (Theil, 1950; Sen, 1968) is arguably the most popular non-parametric and robust alternative to the least squares estimator for estimating the slope in simple linear regression. A variation on this estimator, Siegel's repeated median slope (Siegel, 1982), was the first slope estimator to attain the maximal breakdown value of 50 %, which roughly means that it can withstand up to 50% of outliers in the data. Since their introduction, these estimators have seen widespread use in a variety of applications including signal extraction (Davies et al., 2004; Gather et al., 2006), filtering (Fried et al., 2006; Bernholt et al., 2006; Fried et al., 2007; Gelper et al., 2010), computer vision Meer et al. (1991), climatology (Zhang et al., 2000; Zhai et al., 01 Apr. 2005; Kosaka and Xie, 2013) and very recently differentially private estimation (Alabi et al., 2022; Fu et al., 2019).

Several implementations of these estimators exist, most notably in the R packages **deming** (Therneau, 2018), **zyp** (Bronaugh and for the Pacific Climate Impacts Consortium, 2019), **mblm** (Komsta, 2019), and **RobustLinearReg** (Hurtado, 2020), all publicly available on CRAN. Despite their popularity and importance, all of these publicly available implementations are based on a brute-force computation of the (repeated) median slope. Given a sample of n observations, this approach requires a computational cost of $\mathcal{O}(n^2)$ as well as $\mathcal{O}(n^2)$ space. This makes the currently available implementations unsuitable for modern applications involving larger data sets as they are rather slow and potentially use prohibitively large amounts of storage space.

Nevertheless, there exist several algorithms for the Theil-Sen and repeated median estimators which run in quasilinear time and require only $\mathcal{O}(n)$ space. For the Theil-Sen estimator, Cole et al. (1989), Katz and Sharir (1993), and Brönnimann and Chazelle (1998) proposed algorithms running in $\mathcal{O}(n\log(n))$ deterministic time. A randomized algorithm requiring $\mathcal{O}(n\log(n))$-expected time was proposed in Matoušek (1991), Dillencourt et al. (1992) and Shafer and Steiger (1993). For the repeated median slope, Stein and Werman (1992), Matoušek et al. (1993) and Matoušek et al. (1998) proposed algorithms running in $\mathcal{O}(n\log(n))$ and $\mathcal{O}(n\log(n)^2)$ (-expected) time. A potential explanation for these algorithms not being available in R may be twofold. Firstly, they are all more involved than brute-force computation of the median slopes and sample implementations in other programming languages are scarce, if available at all. Secondly, most of them make efficient use of features not (readily) available in R, such as pointers and in-place assignment, as well as complex data structures.

In this article we discuss the R package **robslopes** (Raymaekers, 2022) which contains implementations of the randomized algorithms by Matoušek (1991) and Matoušek et al. (1998) for the Theil-Sen and repeated median estimators respectively. The original algorithms have been adjusted in order to properly deal with potential duplicates in the data. We start by briefly reviewing the problem setting and the estimators. Next, we present a rough outline of the key ideas underlying the algorithms and a brief description of the usage of the functions in the package. We end with a benchmarking study comparing the implementation with existing implementations in publicly available R packages and concluding remarks.

2 The Theil-Sen and repeated median estimators

The typical problem setting of slope estimation is that of the simple linear model. Suppose we have n observations (x_i, y_i) which follow the model

$$y_i = \alpha + \beta x_i + e_i$$

where α and β are the (unknown) true intercept and slope parameters and e_i represents the noise. The most popular estimator for the regression coefficients is the ordinary least squares (OLS) estimator, i.e. the $(\hat{\alpha}, \hat{\beta})$ minimizing $\sum_{i=1}^{n} (y_i - \hat{\alpha} - \hat{\beta} x_i)^2$. The OLS estimator possesses several attractive properties, including ease of computation and interpretation and optimal performance when the errors are i.i.d. and follow a normal distribution. Despite these properties, it is well known that the OLS estimator is very sensitive to outliers and can have a quickly deteriorating performance as the error term deviates from normality. For these reasons, many alternatives have been suggested, of which the Theil-Sen estimator and Siegel's repeated median slope are two popular and intuitively attractive examples.

The Theil-Sen (TS) estimator (Theil, 1950; Sen, 1968) of β is defined as the median of the slopes of all the lines determined by two different observations (x_i, y_i) and (x_j, y_j):

$$\hat{\beta}_{\text{TS}}(\boldsymbol{x}, \boldsymbol{y}) = \text{med}_{i \neq j} \frac{y_i - y_j}{x_i - x_j},$$

where the median avoids $i = j$ as the slope through one point is undefined and where $\boldsymbol{x} = x_1, \ldots, x_n$ and $\boldsymbol{y} = y_1, \ldots, y_n$. In case there are duplicate values in $x_1, \ldots, x_n$, the proposal of Sen (1968) is to only include the slopes which are constructed using two observations with different x-values and we adopt this approach here.

The TS estimator has been analyzed extensively from a theoretical point of view. Sen (1968) showed its asymptotic normality and equivariance properties. In particular, we have

1. scale equivariance: $\hat{\beta}_{\text{TS}}(\boldsymbol{x}, c\boldsymbol{y}) = c\hat{\beta}_{\text{TS}}(\boldsymbol{x}, \boldsymbol{y})$ for all $c \in \mathbb{R}$
2. regression equivariance $\hat{\beta}_{\text{TS}}(\boldsymbol{x}, \boldsymbol{y} + a\boldsymbol{x}) = \hat{\beta}_{\text{TS}}(\boldsymbol{x}, \boldsymbol{y}) + a$ for all $a \in \mathbb{R}$,

but the estimator is not equivariant under affine transformations of both predictor and response variables. Wang and Yu (2005) give precise conditions for the unbiasedness of the TS estimator and (Wilcox, 1998) studied its behavior in the case of heteroscedastic errors. From a robust statistics perspective, it is known that the TS estimator is much more robust against outliers than the OLS estimator (see Rousseeuw and Leroy (2005)). In particular, it has a bounded influence function (Hampel et al., 1986) and its breakdown value is $1 - \sqrt{\frac{1}{2}} \approx 0.293\%$, where the latter can be roughly interpreted as the maximum percentage of contaminated samples that the estimator can handle (see Donoho and Huber (1983) for an exact definition). Finally, its maximum bias properties under contamination have been studied by Adrover and Zamar (2004).

The search for a slope estimator with a breakdown value higher than 30% prompted Siegel to propose the repeated median (RM) slope (Siegel, 1982). It is the first slope estimator attaining the maximal breakdown value of 50%. The repeated median estimator of β is computed by first calculating the median slope per observation x_i, yielding n values, and then taking the median of these values:

$$\hat{\beta}_{\text{RM}}(\boldsymbol{x}, \boldsymbol{y}) = \text{med}_i \, \text{med}_{j \neq i} \frac{y_j - y_i}{x_j - x_i},$$

where now the inner median avoids $i = j$. We handle duplicate values in $x_1, \ldots, x_n$ by skipping them in the computation of the inner median, in the same spirit as duplicate handling for the TS estimator. The consistency, unbiasedness and efficiency of the RM estimator were discussed by Siegel (1982), whereas the asymptotic normality was analyzed in Hossjer et al. (1994). The estimator was designed to have a 50 % asymptotic breakdown value, and like the TS estimator, its influence function is also bounded (Rousseeuw et al., 1993, 1995; Rousseeuw and Leroy, 2005). Finally, it possesses the same equivariance properties as the TS estimator, i.e. it is both scale and regression equivariant but not affine equivariant, and its maximum bias properties are also discussed in Adrover and Zamar (2004).

It is clear that both $\hat{\beta}_{\text{TS}}$ and $\hat{\beta}_{\text{RM}}$ can be computed by calculating all $\frac{n(n-1)}{2}$ pairwise slopes, and selecting the median or repeated median of these slopes. Clearly, this brute-force approach has a $\mathcal{O}(n^2)$ computational cost, and the available implementations of this approach also require storing the $\mathcal{O}(n^2)$ slopes. In the next section, we describe the more efficient algorithms implemented in the **robslopes** package.

While not the main focus of this article, it is worth mentioning that the intercept can be estimated in several ways. We opt for the most straight-forward approach of taking the median of the residuals

as the estimator for α:

$$\hat{\alpha} = \text{med}_i \left(y_i - \hat{\beta} x_i \right),$$

where $\hat{\beta}$ can be $\hat{\beta}_{\text{TS}}$ or $\hat{\beta}_{\text{RM}}$.

3 Randomized algorithms for slope selection

The R-package **robslopes** contains an implementation of the randomized algorithm by Matoušek (1991) for the Theil-Sen estimator and the algorithm of Matoušek et al. (1993, 1998) for the repeated median estimator. Unlike in their original proposals, we haven taken into account the case of possible duplicate values in the x_i and adjusted the algorithms to work properly in that case as well. We now briefly outline these algorithms without going too much into detail. For specific details and a complete description we refer to the original references and our code.

Suppose we are given n data points (x_i, y_i) with $i = 1, \ldots, n$ and we are interested in the median (or more generally, any order statistic) of the slopes formed by connecting two data points. The main idea behind the algorithms is to consider each observed data point (x_i, y_i) in dual space by associating it with the line

$$v = x_i u - y_i.$$

We will denote the coordinates in dual space with (u, v) in the following. It can be verified that the u-coordinate of the intersection of two lines in dual space (also called "intersection abscissa" (IA)) is equal to the slope of the line passing through the two points in the original space. Therefore, the problem of finding order statistics of slopes can be translated into the problem of finding order statistics of intersection abscissas in dual space. To find these order statistics in dual space, the algorithms use the following idea. Given an interval $(u_{\text{low}}, u_{\text{high}}]$ in dual space, the number of IAs in that interval can be computed by counting the number of inversions in a certain permutation. For a permutation π on $1, \ldots, n$, an inversion is a pair of elements $(\pi(i), \pi(j))$ for which $i < j$ and $\pi(i) > \pi(j)$. Figure 1 shows three lines in dual space and illustrates the connection between IAs and inversions which works as follows. Suppose we have two lines associated with the points (x_i, y_i) and (x_j, y_j). The v-coordinates of the intersection of these lines with u_{low} in dual space are given by $x_i u_{\text{low}} - y_i$ and $x_j u_{\text{low}} - y_j$ respectively. Suppose without loss of generality that $x_i u_{\text{low}} - y_i < x_j u_{\text{low}} - y_j$. Now, if lines i and j intersect in the interval $(u_{\text{low}}, u_{\text{high}}]$, we must have that the v-coordinate of the intersection of these lines with u_{high} have switched order: $x_i u_{\text{high}} - y_i > x_j u_{\text{high}} - y_j$. Therefore, there is a bijection between the IAs in $(u_{\text{low}}, u_{\text{high}}]$ and the permutation obtained by going from the order of the intersections of the lines with $u = u_{\text{low}}$ to the order of the intersections with $u = u_{\text{high}}$. Counting the number of inversions in a permutation can be done in $\mathcal{O}(n \log(n))$ time using an adaptation of merge sort Knuth (1998), and constructing the permutation itself has the same computational complexity.

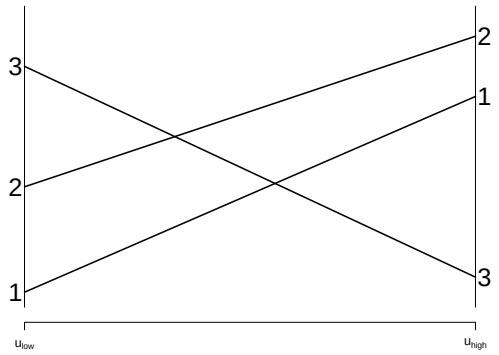

Figure 1: Three lines in dual space, restricted to the interval $(u_{\text{low}}, u_{\text{high}}]$. If two lines intersect in this interval, the ordering of their values of the v-coordinate at u_{low} and u_{high} must have turned around. Therefore, the number of IAs in the interval $(u_{\text{low}}, u_{\text{high}}]$ is given by the number of inversions in the permutation (3 1 2).

For the Theil-Sen estimator, we obtain $n(n-1)$ IAs in dual space, where parallel lines meet at $+\infty$ by convention. Suppose we want to find the k-th smallest IA (e.g., k could be $\lfloor (n(n-1)+1)/2 \rfloor$ for the lower median). The idea is to maintain a half-open interval $(u_{\text{low}}, u_{\text{high}}]$, which is initialized at $(-\infty, \infty]$ and always contains the IA we are looking for. We also keep a count L and C of the total number of IAs to the left and within the interval respectively. If the number of IAs in the interval is of order $\mathcal{O}(n)$, we can enumerate them in $\mathcal{O}(n \log(n))$ time and select the desired element (we call this "brute-force computation"). If there are more IAs left in the interval, we use a contraction

strategy which makes the interval $(u_{\text{low}}, u_{\text{high}}]$ progressively smaller while still containing the target k-th smallest slope, until it is small enough for brute-force computation.

In order to contract the interval, we randomly sample a $\mathcal{O}(n)$ number of IAs. These are then used to estimate a new interval $(u'_{\text{low}}, u'_{\text{high}}] \subset (u_{\text{low}}, u_{\text{high}}]$ which contains the target IA with high probability. To check whether the contracted interval contains the target, we count the number of IAs in the intervals $(u_{\text{low}}, u'_{\text{low}}]$ and $(u'_{\text{low}}, u'_{\text{high}}]$. If the current count L added to the number of IAs in the former interval exceeds k, we know that the target IA is in $(u_{\text{low}}, u'_{\text{low}}]$. If not, we check whether additionally adding the number of IAs in $(u'_{\text{low}}, u'_{\text{high}}]$ gives a total count exceeding k, in which case the target IA is in $(u'_{\text{low}}, u'_{\text{high}}]$. If neither of those cases hold, we know the target IA is in $(u'_{\text{high}}, u_{\text{high}}]$. After the contraction, we update L and C and this process is repeated until C, the number of IAs in the current interval, is of the order $\mathcal{O}(n)$. To execute this strategy, we need a method to randomly sample IAs. It turns out that this can be achieved in $\mathcal{O}(n \log(n))$ time, again using an adaptation of merge sort. For the Theil-Sen estimator there is an expected $\mathcal{O}(1)$ number of iterations required for convergence, leading to a total expected complexity of $\mathcal{O}(n \log(n))$.

For the repeated median estimator, the algorithm is similar to the previously described algorithm in that it also works with an interval-contraction strategy. In contrast with the previous algorithm however, we now keep track of the lines for which the median IA falls within the interval, in addition to the number of abscissas on the left and within the current interval for each individual line. In each contraction step, we first sample a $\mathcal{O}(\sqrt{n})$ number of lines for which we know that their median IA lies within the current interval. For each of these lines, a $\mathcal{O}(\sqrt{n})$ number of IAs are now sampled which allow for the estimation of a new interval $(u'_{\text{low}}, u'_{\text{high}}] \subset (u_{\text{low}}, u_{\text{high}}]$ containing the target slope with high probability. The interval $(u_{\text{low}}, u_{\text{high}}]$ can then be contracted by counting the number of IAs for each line on the left and within $(u'_{\text{low}}, u'_{\text{high}}]$. As before, the algorithm switches to brute-force computation once the number of IAs in $(u_{\text{low}}, u_{\text{high}}]$ is of the order $\mathcal{O}(n)$. For this algorithm, there is an expected $\mathcal{O}(\log(n))$ number of contraction steps required, resulting in a total complexity of $\mathcal{O}(n \log^2(n))$.

The original algorithms were only described for the case where the x_i values are all distinct. In case of duplicate values however, some of the slopes are not defined and the natural way of handling this is by ignoring the undefined slopes and computing the median (or any other order statistic) on the remaining slopes. Fortunately, we can incorporate this into our algorithms by realizing that these duplicate x_i values correspond with parallel lines in dual space. Using the convention that parallel lines in dual space meet at $+\infty$, the (repeated) median slope can be found by appropriately adjusting the value of the order statistic of the slope we want to find. For the Theil-Sen estimator, this means we need to find the order statistic corresponding with $\frac{n(n-1)}{2} - \sum_{i=1}^{n} \frac{d_i - 1}{2}$ values, where d_i denotes the number of times the value x_i occurs in the predictor variable. For the repeated median estimator, we only have to adjust the inner median, but the adjustment is dependent on the individual line i. More specifically, the order statistic for the inner median needs to be computed on $n - d_i$ values. Note that we can count the number of duplicates easily in $\mathcal{O}(n)$ time and so the overall complexity of the algorithm is not affected by this change. There is one additional complication, namely that of duplicate pairs (x_i, y_i). If such pairs are present, one should be careful with computing the permutation of intersections in dual space given an interval $(u_{\text{low}}, u_{\text{high}}]$. Obtaining such a permutation involves sorting (and ranking) a vector of IAs. However, in case of duplicate pairs, this sorting needs to be done using stable sort. If not, duplicate pairs may randomly produce inversions, yielding sampled IAs at ∞ even if the current interval has $u_{\text{high}} < \infty$. In the original algorithm, this scenario was not considered and thus not explicitly accounted for. These changes have been incorporated in the implementation provided in the **robslopes** package.

4 Implementation and usage

We briefly describe the implementation and usage of the functions in the **robslopes** package. The package revolves around 2 main functions, the `TheilSen` function and the `RepeatedMedian` function, both returning a list with the self-explanatory elements `intercept` and `slope`. Both functions start with input checking and duplicate counting, which are done in R. The duplicate counts are then used to set the correct target order statistics for the main part of the algorithm. This information is then passed on to the randomized algorithms which are implemented in C++, making use of the **rcpparmadillo** package (Eddelbuettel and François; Eddelbuettel and Sanderson, 2014).

The `TheilSen` function has the layout `TheilSen(x,y,alpha = NULL,verbose = TRUE)`, where the `x` and `y` arguments are the input vectors for the predictor and response variable respectively. The `verbose` option allows for switching off the printing of the computation progress. Finally, the `alpha` argument is a value between 0 and 1 which determines the order statistic corresponding with the target slope. When `alpha = NULL`, the default, the upper median of the m slopes is computed, which corresponds with the $\lfloor (m+2)/2 \rfloor$-th order statistic. For any other value of $0 \leq$`alpha`≤ 1, the function

computes the $\lfloor$alpha $m\rfloor$-th order statistic of the slopes, where $\lfloor\cdot\rfloor$ is the rounding operator.

The RepeatedMedian function has the layout RepeatedMedian(x,y,alpha = NULL,beta = NULL,verbose = TRUE), where the x, y and verbose arguments play the same role as for the TheilSen function. The arguments alpha and beta determine the order statistics of the inner and outer "median". When NULL, the default, they again correspond to the upper median. If they contain values between zero and one, the order statistics corresponding with the inner and outer "median" are given by $\lfloor$alpha $m\rfloor$ and $\lfloor$beta $m\rfloor$ respectively.

For convenience, the TheilSen and RepeatedMedian functions have been wrapped in the user-friendly functions robslope and robslope.fit. These mimic the structure of common regression functions (e.g., lm and lm.fit). In particular, robslope takes the standard arguments formula, data, subset, weights and na.action, in addition to the type argument selecting which type of slope to compute, as well as the optional alpha, beta and verbose arguments of the TheilSen and RepeatedMedian functions.

As an example we analyze the flights data of the **nycflights13** package (Wickham, 2019). It contains on-time data for all flights that departed in New York City (i.e. JFK, LGA or EWR airports) in 2013. We consider the distance traveled in miles as the predictor variable, and take the air time in minutes as the response. After removing all NA values, we end up with a data set of 327,346 observations. In contrast to the previous example, this one is too large to compute the TS or RM slope with a brute-force algorithm (on most computers). We now calculate the TS and RM slopes three times each, where we change the inner order statistic to the first, second and third quartiles. With the exception of graphical parameters, the code executed is shown below.

```
library("robslopes")
library("nycflights13")
data("flights")

ts.out.25 <- robslope(formula = air_time~distance, data = data, alpha = 0.25)
ts.out.50 <- robslope(formula = air_time~distance, data = data, alpha = 0.50)
ts.out.75 <- robslope(formula = air_time~distance, data = data, alpha = 0.75)
plot(data$distance, data$air_time)
abline(coef(ts.out.25), col = "red", lwd = 3)
abline(coef(ts.out.50), col = "green", lwd = 3)
abline(coef(ts.out.75), col = "blue", lwd = 3)

rm.out.25 <- robslope(formula = air_time~distance, data = data, beta = 0.25,
                      type = "RepeatedMedian")
rm.out.50 <- robslope(formula = air_time~distance, data = data, beta = 0.50,
                      type = "RepeatedMedian")
rm.out.75 <- robslope(formula = air_time~distance, data = data, beta = 0.75,
                      type = "RepeatedMedian")
plot(data$distance, data$air_time)
abline(coef(rm.out.25), col = "red", lwd = 3)
abline(coef(rm.out.50), col = "green", lwd = 3)
abline(coef(rm.out.75), col = "blue", lwd = 3)
```

The resulting figure is shown in Figure 2. In this example, the slopes calculated by the TS and RM estimators are virtually identical due to the very limited number of outliers and the large number of data points. Note that there are many duplicates in this data set, which are handled appropriately by the proposed implementation.

5 Benchmarking study

We now benchmark the implemented algorithm against the existing implementations in the **deming**, **zyp**, **mblm**, and **RobustLinearReg** packages. Note that the **deming** and **zyp** packages only contain the TS estimator, the others contain both the TS and RM estimators. All simulations were done using the R-package **microbenchmark** (Mersmann, 2019) and were run on an Intel® Core™ i7-10750H @2.60GHz processor. The setup is as follows. For each value of $n = \{10, 10^2, 10^3, 10^4\}$, we have generated 100 samples (x_i, y_i) from the bivariate standard normal distribution. Afterward, each of the available implementations was used to estimate the regression parameters and the execution time was measured (in nanoseconds).

The following code snippets show the code used for benchmarking the TS and RM estimator for a sample of size $n = 10^3$, the same code was used for the other sample sizes, with exception of $n = 10^4$,

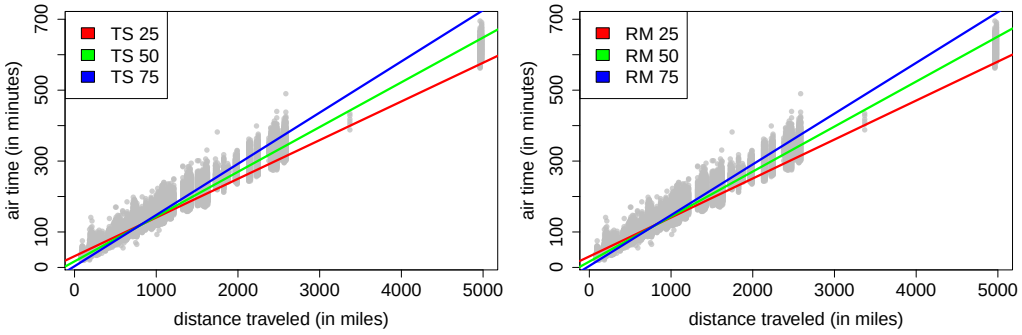

Figure 2: The TS (left) and RM (right) slopes fitted on the flights data. The first, second and third quartiles as the (inner) order statistic are shown in red, green and blue respectively. The slopes are virtually identical for both estimators in this example, as there are not many outliers in the data. The implemented algorithm appropriately deals with the many duplicate x-values in this data set.

in which case the implementation of the **mblm** package was left out due to its computational burden (it takes several hours to compute it once).

```
n   <- 10^3
mbm <- microbenchmark("deming" = deming::theilsen(y~x, data = data),
                      "zyp" = zyp::zyp.sen(y~x, dataframe = data),
                      "mblm" = mblm::mblm(y~x, dataframe = data, repeated = FALSE),
                      "RobustLinearReg" = RobustLinearReg::theil_sen_regression(y~x,
                      data = data),
                      "robslopes" = robslopes::TheilSen(x, y),
                      setup = {x = rnorm(n); y = rnorm(n);
                      data = as.data.frame(cbind(x, y))}, times = 100)
autoplot(mbm)

n   <- 10^3
mbm <- microbenchmark("mblm" = mblm::mblm(y~x, dataframe = data, repeated = TRUE),
                      "RobustLinearReg" = RobustLinearReg::siegel_regression(y~x,
                      data = data),
                      "robslopes" = robslopes::RepeatedMedian(x, y),
                      setup = {x = rnorm(n); y = rnorm(n);
                      data = as.data.frame(cbind(x, y))}, times = 100)
autoplot(mbm)
```

Figure 3 shows the results for the Theil-Sen estimator. We see that for the smallest sample size, $n = 10$, the absolute computation times are extremely low and while there are visible differences, they are probably not of practical relevance. For $n = 100$, the **robslopes** package is roughly five times faster than the fastest competitor, and the **mblm** implementation starts to run away from the other competitors. For $n = 10^3$, the differences become much clearer. The **robslopes** implementation is now between one and two orders of magnitude faster than the best competitor. Finally, for $n = 10^4$ the gap widens further to a difference of over two orders of magnitude. Note that for $n = 10^4$, we have left out the **mblm** package in the comparison as a single run takes several hours and the resulting plot would hinder a clear comparison.

Figure 4 shows the results for the RM estimator. As for the TS estimator, the difference in computation time is already visible for small sample sizes of $n = 10$ and $n = 100$, where the **robslopes** implementation is roughly a factor 20 faster than the competition. However, the absolute computation times for these sample sizes is by no means prohibitive for any of the implementations. Starting at $n = 1000$ however, we start to see a larger difference, especially between the **robslopes** and **mblm** implementations. The former is now roughly 800 times faster than the latter, and 60 times faster than the **RobustLinearReg** implementation. For sample size $n = 10^4$, we did not include the **mblm** implementation as it now takes roughly half an hour to compute, in contrast with 16 seconds for the **RobustLinearReg** implementation and 0.034 seconds (on average) for the **robslopes** implementation. Finally note that interestingly, the repeated median of the **mblm** package is much faster than the Theil-Sen estimator of the same package. This is somewhat counter-intuitive, but turns out to be caused by a much higher number of calls to the `c()` function (concatenate) in the implementation of the TS estimator.

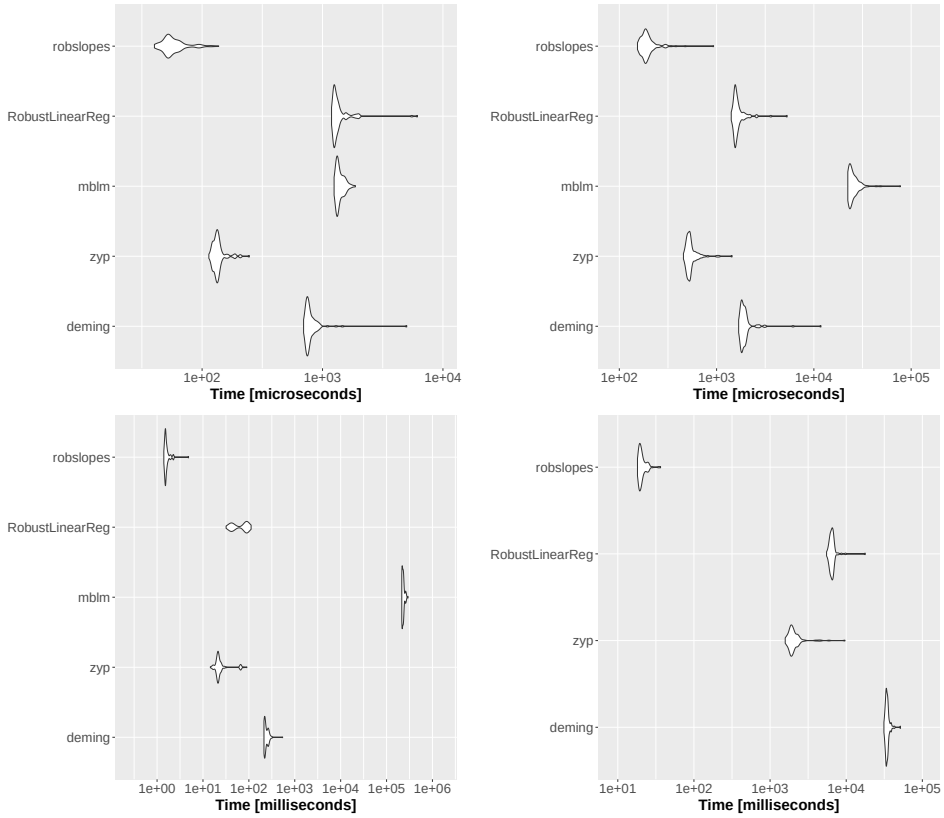

Figure 3: Computation times of the different implementations of the Theil-Sen slope estimator for the sample size n equal to 10, 10^2, 10^3 and 10^4 in the top left, top right, bottom left and bottom right panels respectively. The **robslopes** implementation is consistently faster than the competition by orders of magnitude. The difference scales close to linearly in n, gaining almost one order of magnitude advantage for a similar increase in the sample size, which is what we expect based on the theoretical computational complexities. The **mblm** estimator was left out for sample size $n = 10^4$ due to it requiring several hours to compute once.

As explained before, the brute-force implementations of the (repeated) median slope require $\mathcal{O}(n^2)$ storage as they typically compute all slopes and store them in a $n \times n$ matrix. Therefore, while a sample size of $n = 10^4$ is by no means extremely large, we cannot go much higher than that. A sample size of $n = 10^5$ for example, would require a RAM memory of about 75GB, which is rather uncommon on most computers and laptops. The implementations in the **robslopes** package however, require only $\mathcal{O}(n)$ storage space, and we can thus easily compute the estimators on much larger samples. To illustrate this, we have continued the benchmarking study with only the **robslopes** implementation, but this time for the sample sizes $n = 10^5, 10^6$ and 10^7. This was done using the following code snippet, where the saving of the results in the for loop is omitted:

```
for (n in 10^(5:7)) {
        mbm <- microbenchmark("robslopes" = robslopes::TheilSen(x, y),
                    setup = {x = rnorm(n); y = rnorm(n);
                    data = as.data.frame(cbind(x, y))}, times = 100)
}

for (n in 10^(5:7)) {
        mbm <- microbenchmark("robslopes" = robslopes::RepeatedMedian(x, y),
                    setup = {x = rnorm(n); y = rnorm(n);
                    data = as.data.frame(cbind(x, y))}, times = 100)
}
```

Figure 5 shows the mean computation times of the TS estimator and the RM estimator for sample sizes up to $n = 10^7$. As an example, for a sample size of $n = 10^6$, the TS estimator requires roughly 4 seconds of computation time, whereas the RM estimator requires roughly 6 seconds. The blue shade on the figure indicates the maximum and minimum computation time over the 100 replications, showing that the computation times have a fairly small variance around their mean. The red line

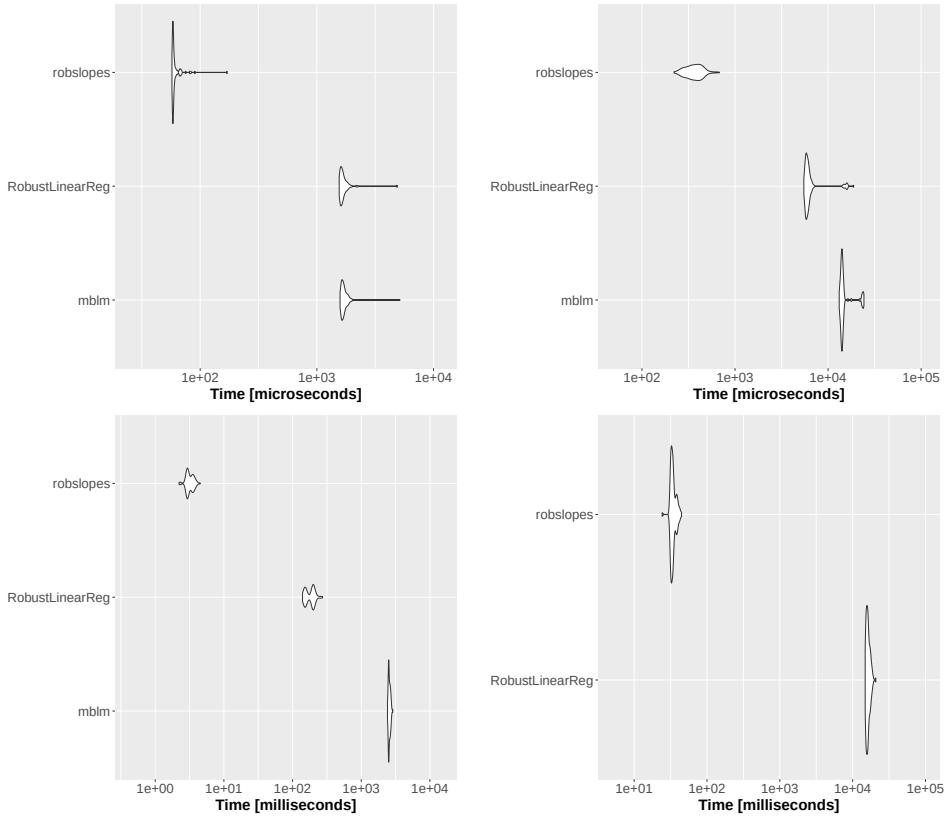

Figure 4: Computation times of the different implementations of the repeated median slope estimator for the sample size n equal to 10, 10^2, 10^3 and 10^4 in the top left, top right, bottom left and bottom right panels respectively. The **robslopes** implementation is consistently faster than the competition by orders of magnitude. The difference scales close to linearly in n, which is what we expect based on the theoretical computational complexities. Note that the **mblm** estimator was left out for sample size $n = 10^4$ due to it requiring more than 10^6 milliseconds (approx. 30 minutes) to compute.

shows a (robustly) estimated fit of the theoretical computation times to the observed ones (i.e. of the functions $f(n) = \beta n \log(n)$ and $f(n) = \beta n \log^2(n)$ for a $\beta \in \mathbb{R}$), indicating that for $n \geq 10^3$, the observed computational cost grows according to the theoretical complexity.

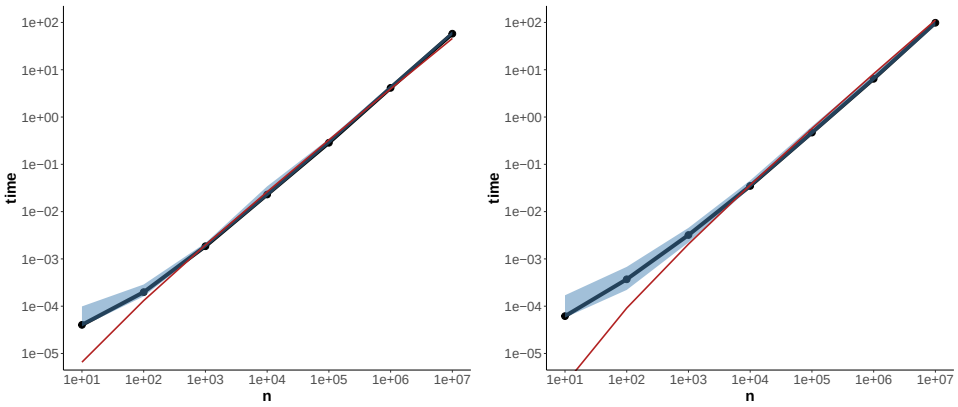

Figure 5: Mean computation times of the Theil-Sen (left) and repeated median (right) estimators as implemented in **robslopes** for increasing sample size n. The red line is an estimate of the $\sim n\log(n)$ (for TS) and $\sim n\log^2(n)$ (for RM) expected computational cost. The blue shade indicates the minimum and maximum computation times. Other than the deviations for very small n which are due to computational overhead, the computational cost scales precisely as the theory predicts. Furthermore, the variance of the computation times around their mean is negligible.

6 Summary

We have introduced the **robslopes** package which contains fast implementations of the popular Theil-Sen and repeated median slope estimators. The implemented algorithms are randomized algorithms running in quasilinear expected time and use linear space. In contrast, the currently available implementations in different R packages on CRAN require quadratic time and space. A benchmark study comparing the common implementations of the slope estimator with the newly introduced one illustrates speedups up to a factor 10^3 compared with the next best alternative implementation for common sample sizes. Additionally, due to the linear space requirements of the algorithms, the slope estimators can be computed on much larger sample sizes than the current maximum. Finally, the original algorithms were adjusted to properly deal with potential duplicates in the predictor variable.

The fast implementation of these popular slope estimators unlocks new possibilities for their use in modern applications where the slope has to be estimated repeatedly and on a large number of data points. Evidently, inferential procedures based on bootstrapping are also highly facilitated by these fast algorithms. Finally, the underlying C++ implementation may serve as a useful reference for implementations of these algorithms in other programming languages, which also seem to be scarce.

Bibliography

J. Adrover and R. H. Zamar. Bias robustness of three median-based regression estimates. *Journal of statistical planning and inference*, 122(1-2):203–227, 2004. URL https://doi.org/10.1016/j.jspi.2003.06.001. [p39]

D. Alabi, A. McMillan, J. Sarathy, A. Smith, and S. Vadhan. Differentially private simple linear regression. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022(2):184–204, 2022. URL https://doi.org/10.2478/popets-2022-0041. [p38]

T. Bernholt, R. Fried, U. Gather, and I. Wegener. Modified repeated median filters. *Statistics and Computing*, 16(2):177–192, 2006. URL https://doi.org/10.1007/s11222-006-8449-1. [p38]

D. Bronaugh and A. W. for the Pacific Climate Impacts Consortium. *zyp: Zhang + Yue-Pilon Trends Package*, 2019. URL https://CRAN.R-project.org/package=zyp. R package version 0.10-1.1. [p38]

H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Computational Geometry*, 10(1):23–29, 1998. URL https://doi.org/10.1016/S0925-7721(97)00025-4. [p38]

R. Cole, J. S. Salowe, W. L. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM Journal on Computing*, 18(4):792–810, 1989. URL https://doi.org/10.1137/0218055. [p38]

P. L. Davies, R. Fried, and U. Gather. Robust signal extraction for on-line monitoring data. *Journal of Statistical Planning and Inference*, 122(1-2):65–78, 2004. URL https://doi.org/10.1016/j.jspi.2003.06.012. [p38]

M. B. Dillencourt, D. M. Mount, and N. S. Netanyahu. A randomized algorithm for slope selection. *International Journal of Computational Geometry & Applications*, 2(01):1–27, 1992. URL https://doi.org/10.1142/S0218195992000020. [p38]

D. Donoho and P. Huber. The notion of breakdown point. In P. Bickel, K. Doksum, and J. Hodges, editors, *A Festschrift for Erich Lehmann*, pages 157–184, Belmont, 1983. Wadsworth. [p39]

D. Eddelbuettel and R. François. Rcpp: Seamless r and c++ integration. URL https://doi.org/10.18637/jss.v040.i08. [p41]

D. Eddelbuettel and C. Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics and Data Analysis*, 71:1054–1063, March 2014. URL http://dx.doi.org/10.1016/j.csda.2013.02.005. [p41]

R. Fried, T. Bernholt, and U. Gather. Repeated median and hybrid filters. *Computational statistics & data analysis*, 50(9):2313–2338, 2006. URL https://doi.org/10.1016/j.csda.2004.12.013. [p38]

R. Fried, J. Einbeck, and U. Gather. Weighted repeated median smoothing and filtering. *Journal of the American Statistical Association*, 102(480):1300–1308, 2007. URL https://doi.org/10.1198/016214507000001166. [p38]

S. Fu, C. Xie, B. Li, and Q. Chen. Attack-resistant federated learning with residual-based reweighting, 2019. URL https://doi.org/10.48550/ARXIV.1912.11464. [p38]

U. Gather, K. Schettlinger, and R. Fried. Online signal extraction by robust linear regression. *Computational Statistics*, 21(1):33–51, 2006. URL https://doi.org/10.1007/s00180-006-0249-8. [p38]

S. Gelper, R. Fried, and C. Croux. Robust forecasting with exponential and holt–winters smoothing. *Journal of forecasting*, 29(3):285–300, 2010. URL https://doi.org/10.1002/for.1125. [p38]

F. Hampel, E. Ronchetti, P. Rousseeuw, and W. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York, 1986. [p39]

O. Hossjer, P. J. Rousseeuw, and C. Croux. Asymptotics of the repeated median slope estimator. *The Annals of Statistics*, pages 1478–1501, 1994. URL https://doi.org/10.1214/aos/1176325638. [p39]

S. I. Hurtado. *RobustLinearReg: Robust Linear Regressions*, 2020. URL https://CRAN.R-project.org/package=RobustLinearReg. R package version 1.2.0. [p38]

M. J. Katz and M. Sharir. Optimal slope selection via expanders. *Information Processing Letters*, 47(3): 115–122, 1993. URL https://doi.org/10.1016/0020-0190(93)90234-Z. [p38]

D. E. Knuth. *The art of computer programming: Volume 3: Sorting and Searching*. Addison-Wesley Professional, 1998. [p40]

L. Komsta. *mblm: Median-Based Linear Models*, 2019. URL https://CRAN.R-project.org/package=mblm. R package version 0.12.1. [p38]

Y. Kosaka and S.-P. Xie. Recent global-warming hiatus tied to equatorial pacific surface cooling. *Nature*, 501(7467):403–407, 2013. URL https://doi.org/10.1038/nature12534. [p38]

J. Matoušek. Randomized optimal algorithm for slope selection. *Information processing letters*, 39(4): 183–187, 1991. URL https://doi.org/10.1016/0020-0190(91)90177-J. [p38, 40]

J. Matoušek, D. M. Mount, and N. S. Netanyahu. Efficient randomized algorithms for the repeated median line estimator. In *Proceedings of the Fourth ACM-SIAM Annual Symposium on Discrete Algorithms*, pages 74–82, 1993. [p38, 40]

J. Matoušek, D. M. Mount, and N. S. Netanyahu. Efficient randomized algorithms for the repeated median line estimator. *Algorithmica*, 20(2):136–150, 1998. URL https://doi.org/10.1007/PL00009190. [p38, 40]

P. Meer, D. Mintz, A. Rosenfeld, and D. Y. Kim. Robust regression methods for computer vision: A review. *International journal of computer vision*, 6(1):59–70, 1991. URL https://doi.org/10.1007/BF00127126. [p38]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2019. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-7. [p42]

J. Raymaekers. *robslopes: Fast Algorithms for Robust Slopes*, 2022. R package version 1.1.2. [p38]

P. Rousseeuw, C. Croux, and O. Hössjer. Sensitivity functions and numerical analysis of the repeated median slope. *Computational Statistics*, 10(1):71–90, 1995. [p39]

P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005. [p39]

P. J. Rousseeuw, N. S. Netanyahu, and D. M. Mount. New statistical and computational results on the repeated median line. In S. Morgenthaler, E. Ronchetti, and W. A. Stahel, editors, *New Directions in Statistical Data Analysis and Robustnes*, pages 177–194. Birkh¨auser-Verlag, Basel, 1993. [p39]

P. K. Sen. Estimates of the regression coefficient based on kendall's tau. *Journal of the American statistical association*, 63(324):1379–1389, 1968. URL https://doi.org/10.1080/01621459.1968.10480934. [p38, 39]

L. Shafer and W. Steiger. Randomizing optimal geometric algorithms. In *CCCG*, 1993. [p38]

A. F. Siegel. Robust regression using repeated medians. *Biometrika*, 69(1):242–244, 1982. URL https://doi.org/10.2307/2335877. [p38, 39]

A. Stein and M. Werman. Finding the repeated median regression line. In *SODA '92*, 1992. [p38]

H. Theil. A rank-invariant method of linear and polynomial regression analysis. i, ii, iii. *Nederl. Akad. Wetensch., Proc.*, 53:386—392, 521–525, 1397–141, 1950. [p38, 39]

T. Therneau. *deming: Deming, Theil-Sen, Passing-Bablock and Total Least Squares Regression*, 2018. URL https://CRAN.R-project.org/package=deming. R package version 1.4. [p38]

X. Wang and Q. Yu. Unbiasedness of the theil–sen estimator. *Journal of Nonparametric Statistics*, 17(6):685–695, 2005. doi: 10.1080/10485250500039452. URL https://doi.org/10.1080/10485250500039452. [p39]

H. Wickham. *nycflights13: Flights that Departed NYC in 2013*, 2019. URL https://CRAN.R-project.org/package=nycflights13. R package version 1.0.1. [p42]

R. Wilcox. A note on the theil-sen regression estimator when the regressor is random and the error term is heteroscedastic. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, 40(3):261–268, 1998. URL https://doi.org/10.1002/(SICI)1521-4036(199807)40:3<261::AID-BIMJ261>3.0.CO;2-V. [p39]

P. Zhai, X. Zhang, H. Wan, and X. Pan. Trends in total precipitation and frequency of daily precipitation extremes over china. *Journal of Climate*, 18(7):1096 – 1108, 01 Apr. 2005. doi: 10.1175/JCLI-3318.1. URL https://doi.org/10.1175/JCLI-3318.1. [p38]

X. Zhang, L. A. Vincent, W. Hogg, and A. Niitsoo. Temperature and precipitation trends in canada during the 20th century. *Atmosphere-ocean*, 38(3):395–429, 2000. URL https://doi.org/10.1080/07055900.2000.9649654. [p38]

Jakob Raymaekers
Maastricht University
Department of Quantitative Economics
6200 MD Maastricht
The Netherlands
j.raymaekers@maastrichtuniversity.nl

KU Leuven
Department of Mathematics
3001 Leuven
Belgium
jakob.raymaekers@kuleuven.be