Preprint: Fitting Regression and General Hyperplanes in the Presence of Confounding Outliers

J. Paul Brooks, Dept. of Information Systems, Virginia Commonwealth University, Richmond, Virginia, USA
John W. Chinneck, Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada

July 3, 2023

***Abstract*—Confounding outliers (especially when clustered) can severely skew the fit of a hyperplane to data, so robust fitting methods try to identify and remove them before fitting. We present three new methods for this problem: (1) New heuristic methods for identifying and removing outliers and then fitting hyperplanes for regression (to fit the dependent variable values) and for general datasets (to fit the data points). In experiments with up to thousands of points and variables, we observed that these new methods are faster and more accurate than existing state-of-the-art methods. (2) New heuristics for generating quality feasible solutions for finding a hyperplane minimizing the quantile error. (3) A faster and more efficient mixed-integer optimization (MIO) formulation for exactly minimizing the quantile error. We demonstrate the performance of the new MIO on benchmark datasets from the literature and new synthetic outlier-contaminated datasets. In time-limited experiments, the new MIO has better results than competing methods in 23 of 28 experiments (and the same results for 2). We establish that when the fraction of outliers is unknown (as usual in practice) minimizing the quantile error can produce hyperplanes that are a poor fit according to the trimmed sum of errors criterion.**

I. Introduction

Fitting a hyperplane to a set of data points has many applications, including multivariate statistics [1], location theory [2], data compression [3], and pattern recognition [4]. We consider two versions of this problem:

- *Fitting a regression hyperplane*. Find a hyperplane that best predicts the output variable value given the values of a set of input variables. The usual error measures are the difference between the predicted and actual values of the output variable (the residual), or its square.
- *Fitting a general hyperplane*. Find a hyperplane that best approximates the given data points. Common error measures are the Euclidean distance of a data point from the fitted hyperplane, or its square.

Intuitively, the best fitting hyperplane is "close" to the largest number of data points in the set (general case), or to the largest number of output variable values (regression). These are *cardinality*-based (or L_0) measures. Because optimizing the L_0 measure is difficult, many methods instead minimize an L_1 [5]–[7] or L_2-based [6], [8] measure of error, which allows outliers to have an excessive effect on the hyperplane placement. While an L_1 criterion can impart resistance to the influence of outliers, minimizing the L_1 or L_2 error degrades the L_0 measure when outlier contamination is large. The problem is especially acute when there are clusters of outliers [9] and/or outliers are far from the inlier data, perhaps generated by systematic errors in measurement or data entry.

Any fitting method gives better results when the data is not contaminated by outliers. Outliers cause difficulties such as *swamping* [10], [11], when outliers skew the hyperplane so that the inliers are incorrectly identified as outliers, and *masking*, when outliers are fit as well as inliers [10], [11]. Outlier detection and hyperplane fitting must be addressed simultaneously. Robust techniques such as *LTS* take this relationship into account [12], [13].

The first part of this work focuses on developing fast methods for fitting hyperplanes to datasets that are robust to outliers, especially clustered outliers. The methods try to identify inlier points and provide good fits to them. We use synthetic datasets where the number of inliers (m_{in}) is known in advance and evaluate methods on their ability to minimize the sum of the m_{in} smallest squared errors.

The second part of this work focuses on developing improved algorithms for fitting hyperplanes that minimize the qth quantile error (i.e. the qth smallest error). Methods in this class try to minimize the largest error in a specified fraction of the input data (often 50%). Rousseeuw and Van Driessen [14] argue that it is preferable to minimize the sum of the q smallest errors because of statistical efficiency considerations. Minimizing the quantile error remains an attractive option because it can be solved exactly (but slowly) via mixed-integer optimization (MIO) [15] whereas minimizing the sum of the q smallest errors cannot [16]. We develop a more efficient MIO formulation and new heuristics for minimizing the quantile error. We examine the effectiveness of the heuristics in providing initial solutions for the MIO formulation, and look at whether an early exit from optimization can provide a useful result. We evaluate whether minimizing the quantile error can produce a hyperplane that also has a small sum of the m_{in} smallest squared errors.

Our focus is practical and computational. We compare to existing techniques and consider issues such as scalability and runtime.

The main contributions of this paper are:

- A fast method for identifying a superset of the outliers in a dataset. It is a modification of earlier robust measures, is applied across both the original and the principal component analysis (*PCA*) axes, and does not use a fixed cutoff value for identifying outliers.

- An L_0 measure for choosing between two competing hyperplanes. The Absolute Better Metric (ABM) counts the number of points whose error is within some specified tolerance. The novelty is in how the error tolerance is chosen.
- A fast and highly scalable heuristic approach for fitting both regression and general hyperplanes (*CB*). It includes a remediation step that reinstates inlier points that may have been identified as outliers earlier. The method provides better fits than competing methods in a fraction of the solution time.
- A fast and effective heuristic for minimizing the quantile error (*CBq*), with variants for both regression and general fits. Its runtime is little affected by the number of data points, so it is a particularly good choice for very large datasets.
- An improved MIO formulation (*MIO-CB*) for minimizing the quantile error that is faster and more compact than the existing MIO formulation. *MIO-CB* is especially good when coupled with an initialization routine. We explore the impact of stopping this algorithm early.
- We provide examples to establish that when outlier contamination is at most 35%, minimizing the median error can be detrimental to the objective of minimizing the sum of the m_{in} smallest squared errors.

A. Nomenclature

A regression data set consists of an $m \times n$ matrix X of input variable values having m data points in n variables, and output variable values y_i, $i = 1, \ldots, m$. A general data set consists of only the matrix X. The rows of X are points in $\mathbb{R}^n$ denoted x_i, $i = 1, \ldots, m$. We assume that the points in X are in *general position*, meaning that every subset of m points is affinely independent.

For regression fits, the goal is to fit a hyperplane $\{(x, y) \in \mathbb{R}^{n+1} : \beta_0 + x\beta = y\}$. Note that x is a row vector. Fitting methods find values of β_0 and β so that an error measure (e.g the sum of the squared residuals) is minimized. Because the intercept β_0 is found by fitting methods, its value is not prespecified.

For general fits, the goal is to fit a hyperplane $\{x \in \mathbb{R}^n : \beta_0 + x\beta = 0\}$ given by a normal vector $\beta = (\beta_1, \beta_2, \ldots, \beta_n)^T$ and constant β_0. The intercept β_0 is prespecified to avoid null fits. Fitting methods find values of β so that an error measure (e.g. the sum of the squared Euclidean errors) is minimized.

We distinguish between *outliers* and *inliers*, or non-outliers. At various places we will distinguish between the numbers of points in these subsets (m_{out} and m_{in}).

We use italics to denote methods. *LTS* refers to a method for minimizing the least trimmed sum of squares while LTS refers to the least trim sum of squares metric.

B. Best Fit Hyperplanes: Relevant Work

Data analysis in the presence of confounding outliers is an active field of study within statistics and computer science.

L_1 regression, also known as *least absolute deviation* regression, is an outlier-resistant method that replaces the sum-of-squared error criterion in *OLS* with the sum of absolute deviations [7]. An L_1 regression hyperplane can be fit by solving a linear program. Brooks and Dula [5] proved that a general hyperplane that minimizes the sum of L_1 distances can be found by fitting an L_1 regression hyperplane for each variable independently and then selecting the best one. Because these methods are based on minimizing the sum of distances to all points, a sufficiently corrupted single point can adversely affect the result [7], [13]. The problem is exacerbated when there are clusters of outliers [17]. These L_1-norm methods require the solution of dense linear programs, which scales poorly.

Several robust statistical approaches have been proposed for outlier analysis including robust regression. The typical approach is to identify a subset of inliers and then apply a traditional method (e.g., *OLS*) to the subset to fit a model. Many robust methods are heuristics for optimizing desired criteria that are difficult to compute [13]. These methods include M estimators [18], S estimators, *LQS* [19], [20] for least quantile of squares, *LTS* [12] for least trimmed sum of squares, and *LTA* [21] for least trimmed sum of absolution deviations. Many of these have attractive theoretical and asymptotic properties that address some of the shortcomings of outlier-resistant methods such as L_1 regression [20].

A common approach for generating least quantile of squares and least trimmed sum of squares estimates is to generate fits to samples of data points. Each fit to a sample is an *attractor*. A subsequent *concentration* step can be applied by fitting the points with the smallest residuals [13]. *OLS* is commonly used to fit the models. For least quantile of squares, the objective is to minimize the q^{th} smallest squared error while for least trimmed sum of squares, the objective is to minimize the sum of the q smallest squared errors [12]. Rousseeuw and Van Driessen [14] propose the *Fast-LTS* method and argue that minimizing the least trimmed sum of squares is preferred to minimizing the least quantile of squares. The desirable theoretical properties no longer apply to such heuristics unless an exponential number of subsets is used [13]. We agree that the trimmed sum of errors (TSE), of which least trimmed sum of squares (LTS) and least trimmed sum of absolute deviations (LTA) are special cases, is a more natural metric for assessing hyperplane fits as it reflects how well the model fits the inlier data. We will demonstrate that our proposed framework outperforms current implementations of *LQS* and *LTS* using the least trimmed sum of squares metric.

An exact approach for least quantile of squares based on MIO was proposed by Bertsimas and Mazumder [15]. Blanco [16] provides a generic framework for fitting hyperplanes including the least trimmed sum of squares criterion using mixed-integer nonlinear programming and applies them to datasets with 2 or 4 variables. In this work, we provide an improved MIO formulation for least quantile of squares that scales better than the formulation of [15], show that optimizing the least quantile of squares can be detrimental to the least trimmed sum of squares objective, and propose faster methods that generate better hyperplane fits according to the least trimmed sum of squares objective.

Olive and Hawkins proposed *hbreg*, a consistent "practical" high breakdown estimator [13]. In the context of regression,

breakdown is the proportion of data points that can be replaced by arbitrarily corrupt data to make the norm of the vector of coefficient estimates arbitrarily large. A *high breakdown estimator* is one where the breakdown is 1/2 in the limit as more data points are added. In a *consistent* estimator the coefficients converge in probability to their true values [13]. The *hbreg* procedure is comprised of four steps, each with a number of options. We describe here the set of choices that we use for comparisons in our numerical experiments.

1) Apply median ball algorithm regression (*mbareg*, an outlier-resistant estimator: fit 50 *OLS* models with different specified subsets of points and with different variables serving as responses [22]).
2) Fit an *OLS* model with all the data. If the TSE for the *OLS* model is more than 140% of the TSE for the *mbareg* model, then use the *mbareg* estimator.
3) Apply a practical high breakdown estimator where for each of 10 concentration steps, an *OLS* model is fit to the 50% of the points whose response value is nearest to the median; call it *bb*.
4) If the TSE for the *bb* estimator is less than the TSE for the *mbareg* estimator and is less than 140% of TSE for the *OLS* model, then use the *bb* estimator. If not, use the *OLS* model.

We compare our proposed method to *mbareg*, *bb*, and *hbreg* and show that our method produces hyperplanes with smaller TSE values.

Robust statistical methods attempt to fit all or most of the data and then identify outliers as points that deviate significantly from the fit. The "robust PCA" *ROBPCA* method [23] [10] is in this class, though its primary goal is dimension reduction. It uses a robust form of principal component analysis based on projection pursuit for fitting a subspace. Outliers are identified based on two metrics relative to that subspace: (i) the *robust score* distance, and (ii) the orthogonal distance of a point from the *PCA* subspace. The robust score in *ROBPCA* is the Mahalanobois distance. Based on cutoff values for each measure, points can be divided into 4 categories: (i) inlier *regular observations* that have small values of both measures, (ii) *orthogonal outliers* that have a large orthogonal distance from the *PCA* subspace but a small robust scale score, (iii) *bad leverage points* that have a large orthogonal distance and a large robust scale score, and (iv) *good leverage points* that have a large robust scale score, but a small orthogonal distance. See Fig. 6 in [10] for an illustration. While good leverage points are far from the mass of points, they are well fit by the subspace. An ideal hyperplane-fitting method should remove the bad leverage points and the orthogonal outliers and fit a hyperplane to the regular observations and the good leverage points.

Despite the long history of robust regression and robust hyperplane fitting, there are few large benchmark datasets. Exceptions include datasets with 50,000 points used by Rousseeuw and Van Driessen [14] and datasets with 10,001 points used by Bertsimas and Mazumder [15]; however, these datasets all had at most 20 variables and outlier contamination is limited to one or two variables. We evaluate our proposed method on similar datasets. In addition, we experiment with datasets having clustered and unclustered outliers with up to 4,000 data points and up to 1,000 variables. Our experiments with these ranges of size allow a clear picture of the scalability of methods.

Most robust methods are designed for datasets with five or ten times as many data points as variables [13]. Cases where there are more variables than data points are called *high dimension low sample size (HDLSS)* [11], [24], [25]. In our computational experiments, we experiment only with datasets where $m \geq n$. We observe an improvement in the performance of our proposed approaches when $m \geq 2n$.

In the computer science literature, regression is only a small part of outlier analysis, assumptions regarding data representations are rare, and scalability is emphasized [26]. We developed our proposed methods in response to a need for scalable methods for hyperplane fitting in the presence of outliers. Our assumption of an underlying hyperplane is reasonable as hyperplane fitting is the foundation of many data science methods including regression. Our approach, and many other heuristics implemented for robust procedures such as *LQS* and *LTS*, is a *sequential ensemble* method where algorithms are applied sequentially to potentially changing subsets of data points [26]. Our first proposed method includes three successive fits of hyperplanes, two outlier removal steps, and a novel metric for comparing hyperplanes.

C. Least Quantile Error: Relevant Work

For some purposes it is desirable to minimize the error of a chosen quantile q of the points relative to the hyperplane fit. For example, you may wish to minimize the maximum error seen by half of the points. This provides a different hyperplane than minimizing TSE or other metrics.

Bertsimas and Mazumder [15] used MIO to find a hyperplane providing the globally minimum solution for the least quantile regression problem (Eqn. 1). The formulation makes extensive use of Special Ordered Sets of type 1 (SOS-1) that allow at most one nonzero value among a set of variables; these are well handled by modern MIO solvers.

The formulation finds a hyperplane $\beta_0 + x\beta = y$ that minimizes γ, the qth absolute residual, instead of its square. The ith residual is $r_i = r_i^+ - r_i^-$, where $r_i^+, r_i^- \geq 0$. y_i is the value of the dependent variable for the ith point. The full formulation is in (Eqn. 1) (with a correction to the original

paper in the last set of SOS restrictions).

$$
\begin{aligned}
&\textit{(MIO-BM):}\\
&minimize\,\gamma\\
&subject\,to: r_i^+ + r_i^- - \gamma = \overline{\mu}_i - \mu_i, i = 1\ldots m\\
&\qquad r_i^+ - r_i^- = y_i - \beta_0 - x_i\beta, i = 1\ldots m\\
&\qquad \sum_{i=1}^{m} z_i = q\\
&\qquad \gamma \geq \mu_i,\ i = 1\ldots m\\
&\qquad \mu_i, \overline{\mu}_i, r_i^+, r_i^- \geq 0,\ i = 1\ldots m\\
&\qquad (\overline{\mu}_i, \mu_i) : \text{SOS-1},\ i = 1\ldots m\\
&\qquad (r_i^+, r_i^-) : \text{SOS-1},\ i = 1\ldots m\\
&\qquad (z_i, \overline{\mu}_i) : \text{SOS-1},\ i = 1\ldots m\\
&\qquad z_i \in \{0, 1\},\ i = 1\ldots m
\end{aligned}
\tag{1}
$$

The μ_i and $\overline{\mu}_i$ variables act as nonnegative elastic variables equal to the difference between γ and the ith residual. The z_i binary variables select the q residuals that are less than or equal to γ.

II. Novel Fast Heuristics for Fitting Hyperplanes

A fast and accurate new heuristic for fitting regression (*CBreg*) and general (*CBgen*) hyperplanes is developed in this section. The method is comprised of: (i) a new robust method for identifying outliers, (ii) an L_0-based way to compare hyperplanes (ABM), and (iii) a predictor-corrector approach.

Our predictor-corrector approach differs from attractor-concentrator methods in one important way. The latter require that the size of the retained subset q used for the next fit be specified in advance. Our predictor-corrector methods do not: the number of points retained varies depending on which points are identified as outliers (or reinstated if no longer seen as outliers) at each step.

A. A New Way to Identify Outliers

Given the data matrix X, a common way to identify outliers is to apply standard univariate measures to each variable in turn. A point is identified as an outlier if the value of one or more of its variables is deemed to be an outlier. There are a variety of single-variable outlier detection measures, including values beyond 3 standard deviations from the mean, more than 3 scaled median absolute differences from the median, more than 1.5 interquartile ranges above the upper quartile or below the lower quartile, the Grubbs test [27], and the generalized extreme studentized deviate test [28].

No matter which method is applied, the weakness of the univariate approach is that the variable axes may not align with any of the important directions for identifying outliers in a multi-dimensional dataset. This causes failures such as identifying all points as outliers, or not identifying any outliers at all.

We develop here a robust way to identify outliers that is quick and effective in our tests. It makes no assumptions about the number of outliers (beyond assuming that outliers constitute no more than half of all points).

The new method analyzes the dataset along $2n$ single-dimension axes: the n original variable columns, and the n principal component axes given by a singular value decomposition (SVD)-based *PCA* analysis of the entire dataset. For each axis we calculate a robust scaled measure of the distance of each point from the median vs. the median distance of all points from the median in that axis. Where $A = \{a_1, a_2, a_3, \ldots, a_m\}$ is the scalar set of values for the m points along some axis a, the *median absolute difference from the median (MAD)* for axis a is:

$$MAD_a = median(|A - median(A)|) \tag{2}$$

(same as Eqn. 3 in [10] but without the correction factor to make the MAD consistent with Gaussian distributions). Our scaled robust measure for element i in a univariate list is:

$$d_i = |a_i - median(A)|/(MAD_a + \epsilon) \tag{3}$$

This is similar to Eqn. 6 in [10], but includes the small positive offset ϵ, set to 1 in our later experiments. The denominator scales the distance from the median by MAD, with ϵ added to handle cases where the MAD is zero or close to it. For example, when the number of points is only slightly larger than the number of variables, the fit is perfect for many of the points, giving a median of zero. This is not infrequent in our experiments as suspected outliers are removed, hence a hedge is needed.

This gives a scaled measure of how atypical a point is, allowing axes on different scales to be compared fairly. The largest d_i found for each point i over the $2n$ axes is denoted D_i. The set $D = \{D_1, D_2, \ldots, D_m\}$ is analyzed to identify outliers. There are two ways we do this. In both cases D is first sorted in ascending order.

The first method simply assumes that the $m/2$ points associated with the $m/2$ largest values in D are outliers. This very likely identifies too many points as outliers, but the later reinstatement of points in Algs. 1 and 2 corrects this. This works very well when $m/n \geq 2$, but if $m/n < 2$, the method assumes that the $m-n$ points associated with the $m-n$ largest values in D are outliers. This leaves an $n \times n$ submatrix, whose rows define a unique hyperplane.

The second method tries to make a more accurate distinction between outlier and inlier points. It finds the first abrupt change in value in the sorted set D past some designated number of points (see [29] for background on identifying changepoints in datasets). For $m/n \geq 2$ it takes the first abrupt change (at some point k) past point $\lceil m/2 \rceil$ to set a cutoff value $c = D_k$. This typically corresponds to the change from inlier to outlier points. All points having $D_i \geq c$ are identified as outliers. When $m/n < 2$ it looks for the first abrupt change past point n in sorted D to set the cutoff value. If there is no abrupt change in the search area, then it defaults to the first method. While just as accurate in identifying outliers, this method mislabels fewer inlier points as outliers.

Computational Complexity: The main steps are an $O(mn^2)$ singular valued decomposition, $O(mn\log m)$ for the $2n$ MAD calculations, $O(m\log m)$ sorting for the abrupt cutoff calculations, and $O(m)$ for the abrupt cutoff identification. This

results in an overall computational complexity of $O(mn^2 + mn\log m + m)$.

B. Absolute Better Metric: L_0 Comparison of Hyperplanes

An L_0 metric counts the number of points that are within some maximum residual (regression fit) or maximum distance (general fit) from the fitted hyperplane. The crucial decision is selecting the maximum residual (*maxRes*) or maximum distance (*maxDist*). The ABM metric does this by making an initial approximate fit to the entire dataset, and taking *maxRes* or *maxDist* as a specified percentile of the residuals or distances.

For regression fits we calculate the absolute residual for each point. For general fits we calculate the Euclidean distance of each point from the initial approximate fit, assuming that the data is appropriately scaled. We use the 16^{th} percentile value of the residuals for *maxRes* or the distances for *maxDist* since this is approximately one standard deviation below the mean in a normal distribution.

The initial approximate fit is via *OLS* for a regression fit, or via *PCA* for a general fit, applied to the entire dataset. Even if the initial fit hyperplane is skewed by outliers, the values of *maxRes* or *maxDist* are not typically greatly affected, and are still useful.

A significant increase in the ABM count between the initial fit and the final fit increases confidence that a much better fit has been found.

C. Chinneck-Brooks (CB) Algorithms for Hyperplane Fitting

The *CB* algorithms for fitting hyperplanes leverage the novel outlier identification method and ABM. Our method for identifying outliers may initially erroneously label some inliers as outliers, and remove them. To compensate, the algorithm includes a remediation step in which inliers may be reinstated before a revised final hyperplane is fit. This approach relies on a reasonable set of assumptions:

1) No more than half of the data points are outliers.
2) A superset of the outliers can be found. All (or nearly all) outliers are included in this set along with some inliers.
3) There are sufficient data points that when the outlier superset is removed the remaining points constitute a representative subset of the original dataset.

Assumption 2 is satisfied by the outlier identification method described in Sec. II-A, which makes use of Assumption 1 to cap the maximum number of points identified as outliers relative to the initial hyperplane. Assumption 3 is key to reinstating any inlier points that may have been erroneously removed: the *predictor* hyperplane fit to the reduced set of points after outlier removal is a better approximation of the best-fit hyperplane. We examine the errors of all points relative to the predictor hyperplane and omit from the original dataset any points that are outliers in the error measure (this reinstates any good leverage points while leaving out bad leverage points and orthogonal outliers). Finally the *corrector* hyperplane is fit to the reduced dataset.

An overview of the *CBreg* algorithm for fitting regression hyperplanes is given in Alg. 1. HP$^{(k)}$ indicates the *k*th fitted hyperplane, ABM$^{(k)}$ indicates the total number of points within $maxRes$ of hyperplane HP$^{(k)}$, and resid$^{(k)}$ indicates the set of residual errors for all m points relative to hyperplane HP$^{(k)}$. The outlier identification method is applied to the matrix $[X|y]$ in Step 3. Step 5 identifies outliers as points more than three scaled MAD from the median absolute residual (includes both orthogonal outliers and bad leverage points).

Algorithm 1 *CBreg* for Fitting Regression Hyperplanes

1: INPUTS: $m \times n$ matrix $X^{(1)} = X$ of predictor variables, $m \times 1$ vector $y^{(1)} = y$ of outcome values.
2: Regress over $y^{(1)}$ and $X^{(1)}$ to find HP$^{(1)}$. Use HP$^{(1)}$ to find $maxRes$, resid$^{(1)}$, and ABM$^{(1)}$.
3: Identify and remove outliers as per Sec. II-A to create $X^{(2)}$ and $y^{(2)}$.
4: *Predictor*: regress over $y^{(2)}$ and $X^{(2)}$ to find HP$^{(2)}$. Use HP$^{(2)}$ to find resid$^{(2)}$ and ABM$^{(2)}$ over all m points in $X^{(1)}$ and $y^{(1)}$.
5: Identify outliers in resid$^{(2)}$, and remove the corresponding points from $X^{(1)}$ to create $X^{(3)}$ and $y^{(3)}$.
6: *Corrector*: regress over $y^{(3)}$ and $X^{(3)}$ to find HP$^{(3)}$. Use HP$^{(3)}$ to find resid$^{(3)}$ and ABM$^{(3)}$ over all m points in $X^{(1)}$ and $y^{(1)}$.
7: OUTPUT: Set HP^{out} to the hyperplane associated with the largest ABM count. Break ties in favor of the later hyperplane.

The algorithm for fitting general hyperplanes is similar with the main differences being the substitution of a *PCA* algorithm in place of *OLS*, and the substitution of the Euclidean distance from a hyperplane as the error measure in place of the residuals. An overview of the *CBgen* algorithm for fitting general hyperplanes is shown in Alg. 2. The maximum Euclidean distance error $maxDist$ takes the place of $maxRes$, and the Euclidean distance errors edist$^{(k)}$ for HP$^{(k)}$ replace resid$^{(k)}$. ABM$^{(k)}$ for HP$^{(k)}$ is calculated using the Euclidean distance errors relative to $maxDist$. The outlier identification in Step 3 is applied to X.

Algorithm 2 *CBgen* for Fitting General Hyperplanes

1: INPUT: $m \times n$ data matrix $X^{(1)} = X$.
2: Apply *PCA* to $X^{(1)}$ to find HP$^{(1)}$. Use HP$^{(1)}$ to find $maxDist$, edist$^{(1)}$, and ABM$^{(1)}$.
3: Identify and remove outliers as per Sec. II-A to create matrix $X^{(2)}$.
4: *Predictor*: Apply *PCA* to $X^{(2)}$ to find HP$^{(2)}$. Use HP$^{(2)}$ to find edist$^{(2)}$ and ABM$^{(2)}$ over all m points in $X^{(1)}$.
5: Identify outliers in edist$^{(2)}$ and remove them from $X^{(1)}$ to create $X^{(3)}$.
6: *Corrector*: Apply *PCA* to $X^{(3)}$ to find HP$^{(3)}$. Use HP$^{(3)}$ to find edist$^{(3)}$ and ABM$^{(3)}$ over all m points in $X^{(1)}$.
7: OUTPUT: Set HP^{out} to the hyperplane associated with the largest ABM count. Break ties in favor of the later hyperplane.

While inspired by the *ROBPCA* outlier removal technique, this fitting method differs from it in several key ways. *CBgen* uses a standard SVD-derived *PCA* to find the initial hyperplane, but then follows it with the predictor and corrector hyperplanes, both seeking better fits to the inliers by more accurate elimination of any outliers. It uses a different univariate robust score measure, applied across both the original and initial *PCA* axes instead of just the *PCA* axes, and uses the largest of the $2n$ scores per point. The outlier identification does not rely on arbitrary cutoff values of the robust scores, instead using a dynamic "abrupt change" criterion. Finally, the *CB* methods use the ABM to decide on the final hyperplane fit. Advantages include:

1) The robust score calculated over $2n$ axes gives a better chance of identifying leverage outliers.
2) The robust score is impervious to cases where MAD is zero or very small (causing score blowup) due to the offset in the denominator.
3) The abrupt change analysis sets the outlier cutoff dynamically in many cases, eliminating the need to use an arbitrary fixed value. In general the method removes either the exact set of leverage outliers, or a superset of them. In the latter case, any inliers are retrieved during the subsequent correction step.
4) The orthogonal outliers are determined relative to the much better predictor hyperplane found after removing the leverage outliers, instead of relative to the initial hyperplane.

Our experiments show this to be a very effective method.

Computational Complexity: For a worst-case analysis, we assume that no points are removed at any of the outlier detection steps, so that the algorithm is dealing with all m points at each step. For *CBreg* there are 3 runs of *OLS* of $O(mn^2+n^3)$, the outlier detection in Sec. II-A at $O(mn^2+mn\log m+m)$, and a one-dimensional outlier detection step at $O(m)$ for an overall worst-case computational complexity of $O(mn^2+n^3+mn\log m+m)$. The steps are identical for *CBgen* except for the substitution of 3 runs of *PCA* of $O(mn\min(m,n)+n^3)$ in place of *OLS*, for an overall worst-case complexity of $O(mn\min(m,n)+n^3+mn^2+mn\log m+m)$.

III. Novel Algorithms for Quantile Error Minimization

Least quantile regression algorithms are designed to minimize the error at some specified quantile, typically the median. Thus the algorithms seek to minimize the maximum error seen in a subset of q of the m data points in the set. In this section, we describe a new heuristic and an improved MIO formulation for fitting hyperplanes according to a least quantile measure of error. There are versions of the algorithm for fitting both regression and general hyperplanes.

A. CBq: New Heuristic for Least Quantile Hyperplane Fitting

CBq is a fast heuristic for quantile error minimization. Given X and y for regression fits (or just X for general fits) and q, it returns a hyperplane that heuristically minimizes the q^{th} smallest error relative to the hyperplane (absolute residual error for regression, Euclidean distance error for general fits). It uses an attractor-concentrator technique (similar to *C-steps* [14]), always retaining the q data points having the smallest errors as successive hyperplanes are found. Results are not always repeatable: when the points are sorted by error, there are always ties near the q^{th} value. These ties are broken differently due to minor numerical differences, which can lead to dissimilar results in different runs.

CBq differs from *LQS* by starting with the hyperplane returned by the *CBreg* or *CBgen* heuristic. *CBq* also uses a linear program to minimize the maximum error in a given subset Q of q points (Eqn. 4), instead of applying ordinary least squares as in *LQS*. Alg. 3 summarizes the steps in *CBqreg* for minimizing the quantile error γ in a regression hyperplane fit. X^Q is the subset of the data matrix corresponding to some set of data points Q. "Error" is the absolute residual.

$$\begin{aligned} &minimize\, e_{max} \\ &\quad subject\, to: \beta_0 + x_i\beta + e_i^+ - e_i^- = y_i,\ i \in Q \\ &\qquad\qquad e_i^+ + e_i^- - e_{max} \le 0,\ i \in Q \\ &\qquad\qquad e_i^+, e_i^-, e_{max} \ge 0,\ i \in Q \end{aligned} \tag{4}$$

Algorithm 3 *CBqreg* for Quantile Error Minimization in a Regression Hyperplane Fit

1: INPUTS: $m \times n$ data matrix X and $m \times 1$ output values vector y. Number of data points q over which to minimize the maximum absolute error.
2: $\gamma^{min} \leftarrow \infty$. $k \leftarrow 0$.
3: Run *CBreg* to obtain regression hyperplane fit $HP^{(0)}$ to X and y.
4: Q ={q points with smallest errors relative to $HP^{(0)}$}. $\gamma^{(0)}$ = maximum error among points in Q.
5: **if** $\gamma^{(k)} < \gamma^{min}$ **then**
6: $\quad\gamma^{min} \leftarrow \gamma^{(k)}$.
7: $\quad k \leftarrow k+1$.
8: $\quad$Solve LP (Eqn. 4) over X^Q to find $HP^{(k)}$ to minimize maximum error.
9: $\quad Q$ ={q points in X with smallest errors relative to $HP^{(k)}$}. $\gamma^{(k)}$ = maximum error among points in Q.
10: $\quad$Go to Step 5.
11: **end if**
12: OUTPUT: γ^{min} and hyperplane $HP^{(k-1)}$.

Note that the LP is solved over the subset of points Q, but the estimate of the minimum quantile error γ^k is updated by applying hyperplane $HP^{(k)}$ to all points in X.

The algorithm is the same for general fits (*CBqgen*) with the exception that *CBgen* is used in Step 3, and the first constraint in Eqn. 4 becomes $\beta_0 + x_i\beta + e_i^+ - e_i^- = 0,\ i \in Q$, where β_0 is preset at n for numerical reasons.

B. Improved Mixed-Integer Formulation

The Bertsimas and Mazumder MIO formulation for regression [15] (Eqn. 1) has $3m+1$ linear constraints and $3m$ SOS-1 conditions in $4m+n+2$ real and m binary variables. The

SOS-1 conditions slow the solution by generating branches in the search tree.

We introduce here an equivalent but more compact formulation for regression that has far fewer SOS-1 constraints and fewer variables (Eqn. 5). e_i^+ and e_i^- are nonnegative elastic variables whose difference gives the residual (only one of these two will be nonzero). q of the residuals must be less than or equal to γ. This is arranged by setting q of the binary z_i to 1, meaning the SOS-1 associated r_i must be zero, so the residual value will be less than or equal to γ.

(MIO-CB-regression):

$$
\begin{aligned}
& minimize\, \gamma \\
& subject\, to: \beta_0 + x_i\beta + e_i^+ - e_i^- = y_i, i = 1 \dots m \\
& \qquad e_i^+ + e_i^- - r_i - \gamma \leq 0,\ i = 1 \dots m \\
& \qquad \sum_{i=1}^{m} z_i = q \\
& \qquad (r_i, z_i) : \text{SOS-1},\ i = 1 \dots m \\
& \qquad e_i^+, e_i^-, r_i \geq 0,\ i = 1 \dots m \\
& \qquad \gamma \geq 0 \\
& \qquad z_i \in \{0, 1\},\ i = 1 \dots m
\end{aligned}
\tag{5}
$$

This formulation has $2m+1$ linear constraints and m SOS-1 conditions in $3m + n + 2$ real and m binary variables. This is a much smaller model than *MIO-BM*: it has m fewer linear constraints and $2m$ fewer SOS-1 conditions in m fewer real variables. These reductions reduce solution times.

The MIO formulation for regression in Eqn. 5 is easily adapted for general hyperplanes: substitute $\beta \mathbf{x}_i^T + e_i^+ - e_i^- = \beta_0, i = 1 \dots m$ in place of the first constraint, with β_0 set to n for numerical stability.

1) Heuristic Initialization: Speed can be improved by initializing an MIO with a high quality solution provided by a quick heuristic, which reduces the size of the branch and bound search tree. Bertsimas and Mazumder [15] tested a number of initialization heuristics, the best of which was their Algorithm 3 (*Alg3-BM*). For general hyperplane fitting, we substitute a *PCA* hyperplane (which tested better than an elastic solution) in place of an L1 regression hyperplane in our implementation of their Algorithm 3. Our new *CBq* algorithms can also be used for MIO initialization and are evaluated for this purpose.

2) Early Termination: The MIO solution process can be halted early, in which case it returns the best integer-feasible solution seen as yet in the run. We observe that this incumbent value of γ often declines steeply early in the run, but then levels off, with only minor improvements following later at increasingly lengthy intervals. The latter behavior indicates minor shuffling between points included in the set of q points having residuals less than γ, and relatively similar points in the complementary set. Terminating the MIO early may have only a minor impact on solution quality. We evaluate this possibility in our experiments.

IV. Experimental Setup

A. Hardware

Computational experiments were conducted on a machine with an Intel Core i7-2600 CPU @ 3.4 GHz processor and 16GB RAM.

B. Software

CBreg and *CBgen* are written in Matlab and use several built-in Matlab routines:

- *PCA* is done by the *pca* function with default settings.
- *OLS* is done by the *regress* function with default settings.
- Orthogonal outliers are identified using the *isoutlier* function with default settings (more than 3 MAD from the median) applied to the absolute Euclidean distances of points from the predictor fit.
- Abrupt changes are identified by the *ischange* function with the 'linear' change detection method.

Instances are solved by the Gurobi [30] solver version 9.1.2. All code, datasets, and results developed for this paper are available online at https://github.com/JChinneck/HPFit.

C. Metrics

Results for methods that use some form of randomization (*mbareg, bb, hbreg, Alg3-BM, LTS,* and *LQS*) may produce different results for different random seeds, but their summary statistics over large dataset collections are stable. Results for the *CB* methods are deterministic and repeatable.

The following metrics are used to evaluate hyperplane fits.

1) LTS:* The m_{in} inlier points are known in our synthetic datasets. Hence our main metric for measuring the goodness of a hyperplane fit is a form of least trimmed squares: the sum of the m_{in} smallest squared residual errors for regression fits, or the sum of the m_{in} smallest squared Euclidean distances to the fitted hyperplane for general fits. We designate this as LTS* to distinguish it from ordinary LTS where the number of inliers is not known in advance. Note that the m_{in} points used in the calculation are not necessarily the original m_{in} inliers if a better solution exists.

2) bnd2: We know the base hyperplane used to generate the synthetic datasets, and so can readily calculate two upper bounds on LTS*:

- *bnd1*. Regression fit: sum of squared residual errors for the inliers relative to the base hyperplane prediction. General fit: sum of the squared Euclidean distances from the inliers to the base hyperplane.
- *bnd2*. Regression fit: use *OLS* to fit a hyperplane to the input inliers, and then sum the squared residual errors for the inliers. General fit: use *PCA* to fit a hyperplane to the input inliers (using the average of the inliers to center the data), and then sum the squared Euclidean distances from the inliers to this fit.

Because of the random noise in the inliers relative to the base hyperplane, bnd2 is tighter than bnd1, and is used as the comparator in the subsequent experiments. Bnd2 is a good bound, but is not tight: in a few datasets there is a solution having a lower value of LTS* that uses a subset of m_{in} points

that is different from the generated set of inliers, or there is a better fit to the inliers than provided by *OLS* or *PCA*.

3) R and geo(R): To compare results over a collection of many datasets, we compute the ratio R = LTS*/bnd2 for each dataset, and then calculate geo(R), the geometric mean of this ratio over the whole collection. The geometric mean mutes the effect of a small number of very large R ratios in the collection, giving a more meaningful metric.

If bnd2 = 0, which happens in a few cases, we set R = 1 if LTS* is also zero, otherwise we omit the case from the calculation of geo(R).

4) Successes: Some methods fail to produce a fitted hyperplane on some datasets. The main reasons are exceeding a time limit, or violating a required minimum on m/n for some methods. For each data collection we record the number of successful completions.

5) Solution Time: This is the time, in seconds, for an algorithm to run, excluding the initial data input time. *hbreg, mbareg*, and *bb* are run together: the time reported for *hbreg* is the overall solution time. The much smaller times to simply retrieve the *mbareg* and *bb* solutions are not reported.

D. Compared Methods

The *mbareg*, *bb*, and *hbreg* algorithms are described in the book by Olive [13]. Their software implementations are written in R and are available online at http://parker.ad.siu.edu/Olive/mpack.txt. We use these methods with default settings.

We use the implementations of *LTS* and *LQS* in the R library *MASS* [31] with default settings.

For fitting general hyperplanes, regression algorithms *mbareg*, *bb*, *hbreg*, *LQS* and *LTS* are each run n times, using each variable as the output variable in turn, and the result having the lowest LTS* is taken as the general fit. This gives these methods the maximum advantage since the set of inliers is not normally known in advance. *LQS* and *LTS* are not run for general hyperplanes with $n > 150$ due to excessive runtimes. For the same reason, *hbreg*, *mbareg*, and *bb* are not run for general hyperplanes with $n > 350$.

Alg3-BM is an implementation in Matlab of Algorithm 3 from [15] for both regression and general hyperplanes. For regression hyperplanes, L_1 regression is used and for general hyperplanes, PCA is used to obtain an initial estimate given by $\beta_0^{(0)}$ and $\beta^{(0)}$. Then 100 random initializations around the hyperplane given by $[\beta_j^{(0)} - 2|\beta_j^{(0)}|, \beta_j^{(0)} + 2|\beta_j^{(0)}|]$ for $j = 0, \ldots, n$ are used as input to Algorithm 2 from [15] for 500 iterations. The hyperplane with the smallest error is given by $\beta_0^{(1)}$ and $\beta^{(1)}$ which is used as input to Algorithm 1 from [15] with a tolerance parameter of 0.0001. The resulting $\beta_0^{(2)}$ and $\beta^{(2)}$ are used as the estimates for *Alg3-BM*.

We ran preliminary experiments using the Matlab *ROBPCA* routine in the LIBRA library (https://wis.kuleuven.be/stat/robust/LIBRAfiles/LIBRA-home-orig) on a collection of 1,000 clustered outliers datasets. The primary goal of *ROBPCA* is dimension reduction where smaller-dimensional subspaces are of primary interest, so good results were not expected in our best-fit hyperplane application. This was confirmed by the experiments in which *ROBPCA* recorded a geo(R) two orders of magnitude worse than *CBgen*. For this reason *ROBPCA* was omitted from further testing.

TABLE I
BM-LIKE SUMMARY STATISTICS (70 DATASETS).

	m	n	m/n	outFrac
avg:	2,629.57	11.00	184.80	0.20
min:	201.00	6.00	18.27	0.20
max:	10,001.00	21.00	476.24	0.20

TABLE II
RESULTS FOR 70 BM-LIKE DATASETS.

Alg.	avg. time	sd (s)	geo(R)	gsd
Alg3-BM	52.94	79.36	5.64	3.25
mbareg	0.24	0.18	11.22	7.72
bb	0.24	0.18	56.24	1.18
hbreg	0.24	0.18	9.48	6.49
OLS	**0.01**	0.01	99.2	1.72
LTS	1.44	2.22	1.75	1.53
LQS	1.43	2.34	1.76	1.55
CBreg	0.97	0.53	**1.00**	1.00

V. EXPERIMENTS: HEURISTICS FOR FITTING REGRESSION HYPERPLANES

We use synthetic and real world datasets based on those previously reported in the literature to compare *CBreg* to other regression methods. All datasets have $m/n > 2$.

A. BM-like Datasets

These 70 regression datasets are similar to those used by Bertsimas and Mazumder [15]: a noisy base regression hyperplane is constructed and then the results are corrupted in one of two ways: (i) by perturbing one of the input variables, or (ii) by perturbing the output variable. Summary statistics are in Table I. The column *outFrac* indicates the fraction of points that are corrupted; outFrac is 20% in all data sets.

Results are summarized in Table II. As in all subsequent tables, the best results are bolded. All methods succeed on all datasets. *CBreg* provides the smallest geo(R), with R=1 in 42 of 70 cases, and a maximum R of 1.01. When comparing the methods to each other rather than to bnd2, *CBreg* provides the smallest LTS* in every case. The two closest geo(R) competitors (*LQS* at 1.75 and *LTS* at 1.76) are both slower. Some other methods (*mbareg, bb, hbreg, OLS*) are faster than *CBreg* but provide much worse R values.

Geo(R) is 99.20 for *OLS* (the initial hyperplane in *CBreg*; same as *OLS*). The subsequent steps in *CBreg* reduce R to 1.0 or close in all cases. Whether or not ABM is used to select the final hyperplane has little impact on the results.

B. NOx Datasets

There are 10 large real-world datasets relating to air pollution in the NOx collection available online at https://rdrr.io/rforge/robustbase/man/NOxEmissions.html [32]. These are modified as in [15] to contaminate the data in the same manner as for the BM-like datasets. Basic statistics are identical for all 10 datasets: $m = 8088$, $n = 4$, $m/n = 2022$, and *outFrac* = $\lfloor 0.01n \rfloor = 0.00989$, or about 1%.

TABLE III
RESULTS FOR 10 NOX DATASETS.

Alg.	avg. time	sd (s)	geo(R)	gsd
Alg3-BM	88.44	17.10	2.50	1.43
mbareg	0.25	0.05	4.12	2.99
bb	0.25	0.05	1.81	1.00
hbreg	0.25	0.05	1.67	1.16
OLS	**0.01**	0.00	86.7	1.02
LTS	2.32	0.37	1.07	1.02
LQS	2.26	0.39	1.07	1.02
CBreg	0.72	0.13	**1.00**	1.00

TABLE IV
RVD-LIKE SUMMARY STATISTICS (90 DATASETS).

	m	n	m/n	outFrac
avg:	8,133.33	5.67	1,513.33	0.20
min:	100.00	3.00	20.00	0.20
max:	50,000.00	10.00	10,000.00	0.20

Results are summarized in Table III. As for the BM-like datasets, *CBreg* has the smallest geo(R), and is faster than the closest competitors (*LTS* and *LQS*). The fastest method *OLS* provides worse values of geo(R). Comparing the methods to each other rather than bnd2, *CBreg* always finds the lowest LTS*.

C. RVD-like Datasets

The 90 datasets in this collection are constructed as done by Rousseeuw and Van Driessen [14]: the output variable generated from a base regression hyperplane is perturbed by a small amount of noise; the first input variable values are then contaminated by outliers. Summary statistics are shown in Table IV. The outlier fraction is 20% for all datasets.

The results summarized in Table V are similar to those for the BM-like and NOx collections. *CBreg* provides the best geo(R) at 1.00. The closest competitor is *mbareg* at geo(R) = 1.03, but faster at an average of 0.28s. The next two closest competitors (*LTS, LQS*) are significantly slower on average. Comparing the methods to each other rather than bnd2 shows that *CBreg* finds the lowest LTS* in 79 of 90 cases, and is never more than 1% worse than the best result.

D. Olive Datasets

These six small datasets are used in the book by Olive [13] to develop and evaluate robust methods for regression. Basic statistics are summarized in Table VI.

TABLE V
RESULTS FOR 90 RVD-LIKE DATASETS.

Alg.	avg. time	sd (s)	geo(R)	gsd
Alg3-BM	104.69	172.01	2.59	5.09
mbareg	0.28	0.21	1.03	1.05
bb	0.28	0.21	47.61	1.22
hbreg	0.28	0.21	1.03	1.05
OLS	**0.04**	0.07	43.53	1.13
LTS	4.20	8.61	1.23	1.21
LQS	4.06	8.38	1.22	1.20
CBreg	0.88	0.34	**1.00**	1.01

TABLE VI
STATISTICS FOR THE OLIVE DATASETS.

	m	n	m/n	outFrac
buxton	87	5	17.40	0.057
glado	267	12	22.25	0.026
hbk	75	4	18.75	0.187
major	112	6	18.67	0.018
nasty	32	5	6.40	0.250
wood	20	6	3.33	0.200

TABLE VII
R RESULTS FOR THE OLIVE DATASETS.

Alg.	buxton	glado	hbk	major	nasty	wood	geo(R)	gsd
Alg3-BM	1.86	4.18	0.98	4.77	7.31	95.73	5.42	4.24
mbareg	0.93	**0.80**	1.10	1.11	2.60	1.30	1.21	1.45
bb	1.14	1.11	0.90	1.19	5.95	5.04	1.86	2.16
hbreg	**0.80**	**0.80**	0.90	1.11	2.69	1.30	**1.15**	1.52
OLS	**0.80**	**0.80**	2.35	1.11	2.69	5.75	1.72	2.05
LTS	1.24	1.60	0.92	1.23	**1.77**	1.87	1.40	1.28
LQS	0.94	1.40	**0.89**	1.25	3.17	293.99	3.33	7.74
CBreg	1.00	1.17	1.00	**1.01**	2.69	**1.00**	1.21	1.44

Results are summarized in Table VII. The best R values in each column are bolded. *CBreg* does not always have the smallest R, but it's values are never large, and its geo(R) over all six datasets is close to the smallest. Average runtimes are very small for most methods, except for *Alg3-BM* at 5.07s and *CBreg* at 0.50s. Note that bnd2 is very small for the *nasty* (0.002185) and *wood* (0.000555) datasets, so small differences in LTS* can generate much larger differences in R.

E. Outlier Identification Accuracy for Regression Datasets

Table VIII summarizes the accuracy of the outlier finding routine when used on the four regression dataset collections. The "inliers" column shows the average fraction of inliers that are identified as outliers (incorrectly); the "outliers" column shows the average fraction of outliers that are identified as outliers (correctly). Our outlier-identification algorithm finds all or the vast majority of the outliers, on average, and misidentifies relatively smaller fractions of the inliers as outliers. The later steps in *CBreg* correct for the misidentifications.

VI. EXPERIMENTS: HEURISTICS FOR FITTING GENERAL HYPERPLANES

A. Synthetic Datasets

We experiment using collections of synthetic datasets constructed so that we have a tight bound on the best possible solution. This supports better evaluation of the competing methods. Datasets are constructed as follows:

TABLE VIII
PERCENTAGE OF INLIERS AND OUTLIERS IDENTIFIED AS OUTLIERS BY CBREG.

coll.	datasets	inliers	outliers
BM-like	70	18.5%	97.3%
NOx	10	41.3%	100.0%
RVD-like	90	21.2%	100.0%
Olive	6	27.4%	93.8%

1) Construct a random *base hyperplane* equation: Set $\beta_0 = 0$ and sample the n coefficients of β from a Uniform(-1,1) distribution. Then normalize β by dividing by the Euclidean norm.
2) Translate the base hyperplane away from the origin: Sample each of the n coordinates for the intercepts of the hyperplane δ from a Uniform(-100,100) distribution. The internal representation of the base hyperplane is $\{x : x = Va + \delta\}$ where $a \in \mathbb{R}^{n-1}$ is a set of multipliers for the $n-1$ basis vectors in the columns of V and δ is the vector of intercepts. Recall that $\beta = (\beta_1, \ldots, \beta_n)$ is the normal to the hyperplane and let β_0 be the offset. The external representation of the hyperplane is $\{x : \beta_0 + x\beta = 0\}$ where the offset β_0 is given by $\beta_0 = -x_i\beta$ for any point x_i on the hyperplane. See Figure 1.
3) Generate inliers: Create random points on the base hyperplane and perturb them with noise as follows. Sample multipliers a_i from a Uniform(-100,100) distribution for $i = 1, \ldots, m_{in}$ and sample noise ϵ_i, $i = 1, \ldots, m_{in}$ from a standard normal distribution. Set $x_i = Va_i + \delta + \beta\epsilon_i$ for $i = 1, \ldots, m_{in}$.
4) Generate random outlier clusters: Steps are (i) generate a random point on the base hyperplane, (ii) generate an outlier cluster center at a random distance perpendicular to the point on the base hyperplane, above or below it, (iii) create outliers by adding noise from a normal distribution with mean 0 to the outlier centers. Multipliers $\alpha \in \mathbb{R}^{n-1}$ for the point on the base hyperplane are each sampled from a Uniform(-100, 100) distribution. The perpendicular distance d from the hyperplane of the outlier center is sampled from a Uniform(1000,10000) distribution. Which side of the base hyperplane an outlier center falls on is chosen at random. A vector of noise for an outlier point ξ_i is sampled from a multivariate normal distribution with means 0 and covariance matrix σI where I is the $n \times n$ identity matrix. The outlier points are given by $V\alpha + \delta \pm \beta d + \xi_i$ for each point i in an outlier cluster where the sign on βd is given by the side of the hyperplane. An outlier center is depicted in Figure 2.

Unclustered outliers are generated using the same procedure but omitting step 4(iii).

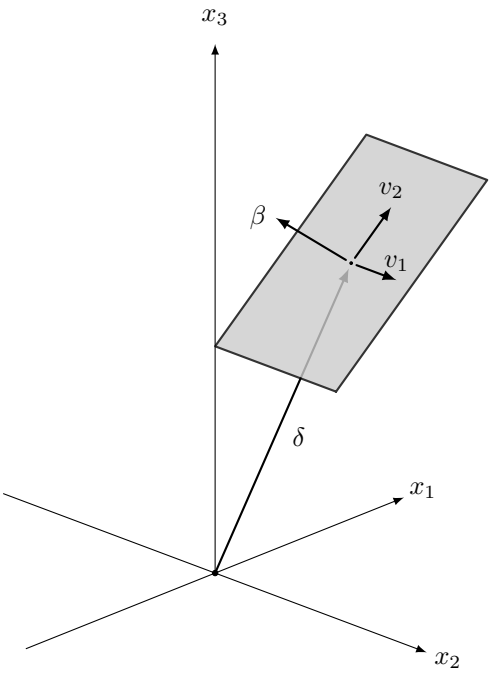

Fig. 1. The base hyperplane is specified by vectors V whose columns are v_1 and v_2 and offset δ. The normal to the hyperplane is β. Points on the base hyperplane are $V\alpha + \delta$ for multipliers α sampled within a bounded region as depicted by the shaded region.

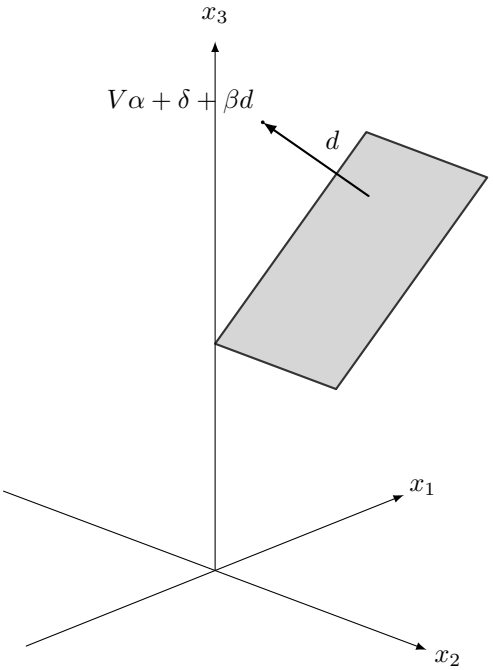

Fig. 2. The center of the outlier cluster is $V\alpha + \delta + \beta d$ where the point $V\alpha + \delta$ is on the base hyperplane. The center of the outlier cluster is a distance d from the hyperplane along the normal vector β.

B. Experiment: Unclustered Outliers

This experiment uses a collection of 1,000 datasets in which outliers are individual points rather than clusters. The number of outliers is sampled from a discrete uniform(1,$m_{in} - 1$) distribution and σ is sampled from a Uniform(0,3) distribution for each dataset. There are 8 configurations for m_{in} and n used to create 125 datasets each as given in Table IX. Basic statistics about the collection are given in Table X. The quantity *outFrac* is the fraction of data points that are outliers.

Experimental results are summarized in Table XI. The bnd2 quantity is zero for 4 datasets, all having $m/n < 2$. There are 807 datasets having $m/n \geq 2$. The best results in each column are in bold. Statistics are calculated over the subset of datasets

TABLE IX
UNCLUSTERED AND CLUSTERED OUTLIERS DISTRIBUTIONS FOR NUMBERS OF DIMENSIONS AND POINTS.

n	**m_{in}**
DiscreteUniform(10,150)	DiscreteUniform(n, $2n$)
DiscreteUniform(10,150)	DiscreteUniform($2n$, 2000)
DiscreteUniform(151,350)	DiscreteUniform(n, $2n$)
DiscreteUniform(151,350)	DiscreteUniform($2n$, 2000)
DiscreteUniform(351,500)	DiscreteUniform(n, $2n$)
DiscreteUniform(351,500)	DiscreteUniform($2n$, 2000)
DiscreteUniform(501,1000)	DiscreteUniform(n, $2n$)
DiscreteUniform(501,1000)	DiscreteUniform($2n$, 2000)

TABLE X
UNCLUSTERED OUTLIERS SUMMARY STATISTICS (1,000 DATASETS).

	m	**n**	**m/n**	**outFrac**
avg:	1,456.7	378.0	7.0	0.307
min:	14.0	10.0	1.1	0.001
max:	3,811.0	1,000.0	149.8	0.499

for which each algorithm succeeds (e.g. over just 189 datasets for *LTS*). "gsd" is the geometric standard deviation, shown with geo(R).

Some observations:

- Only 3 methods succeed on all 1,000 datasets: *Alg3-BM*, *OLS* and *CBgen*. Some methods (*mbareg, bb, hbreg*) are totally unsuccessful for datasets having $m/n < 2$, as well as some datasets having $m/n \geq 2$.
- *CBgen* has the smallest geo(R) over the methods that succeed on all datasets. It is even better for datasets having $m/n \geq 2$ where it has the lowest geo(R) of 1.10. *CBgen* has the lowest geo(R) of 1.07 for the subset of 397 datasets for which the closest competitors *mbareg, bb*, and *hbreg* all succeed.
- *PCA* is applied in Step 2 of *CBgen*, so we can also compare to it. It succeeds in all 1,000 datasets with geo(R) = 425.00. The subsequent *CBgen* steps improve this considerably.
- Comparing the methods to each other rather than to bnd2, *CBgen* has the lowest LTS* in 967 of 1,000 cases.
- *CBgen* is two orders of magnitude faster than the other methods on average.

CBgen dominates in this experiment: it succeeds on all datasets and has the best geo(R), despite the other methods having prior knowledge of how many points are inliers. It is the fastest method, but this is unsurprising since the other methods each run n times.

The outlier detection method in *CBgen* Step 2 is quite accurate. It identifies 100% of the outliers in 998 of the 1,000 datasets. In the remaining 2 datasets, it identifies 99% and 67% of the outliers; both are cases in which $bnd2 = 0$. On average it falsely identifies 10.5% of the inliers as outliers, but these are reinstated in Step 5.

Applying the ABM metric has a minor effect: the output geo(R) is 1.409 vs. 1.410 for Step 6 of *CBgen*.

C. Experiment: Clustered Outliers

This experiment uses a collection of 1,000 datasets in which the outliers are in 1-10 clusters. Basic statistics about the collection are given in Table XII. For each dataset, *pts/clust* is the average number of data points in an outlier cluster.

The number of attributes and points are sampled as for the unclustered outliers (see Table IX) and the total number of outliers m_{out} is sampled from a discrete uniform(1, $m_{in} - 1$) distribution. The number of outlier clusters is sampled from a discrete uniform(1, $\min\{10, m_{out}\}$) distribution. The remaining parameters are the same as for the unclustered outlier datasets.

Whether there are more outliers on one side of the base hyperplane or not can affect the accuracy of a hyperplane placement method. The sizes of the imbalances in the number of outlier clusters on each side of the base hyperplane are summarized in Table XIII. The numbers are balanced in just 162 datasets.

Experimental results are summarized in Table XIV. Bnd2 is zero for two datasets (in this case R = 1 if a method finds a solution having LTS* = 0; R = ∞ is ignored in compiling statistics). There are 815 datasets having $m/n \geq 2$ (none of these have bnd2 = 0). The best results in each column are in bold. Geometric means and averages are calculated over the subset of datasets for which each algorithm succeeds. The geo(R)|gsd column shows the geo(R) and geometric standard deviation. Observations are similar to those for the unclustered datasets:

- Only *Alg3-BM*, *OLS* and *CBgen* are successful for all datasets. *mbareg, bb,* and *hbreg* are again unsuccessful for any dataset having $m/n < 2$, as well as many of the other datasets.
- *CBgen* has the smallest geo(R) over all datasets, and over all datasets having $m/n \geq 2$ (where the geo(R) is significantly improved). *CBgen* has geo(R) = 1.11 for the subset of 397 datasets for which *mbareg, bb,* and *hbreg* succeed.
- *PCA* succeeds for all 1,000 datasets and has geo(R) = 44.30. The subsequent *CBgen* steps improve this significantly.
- Comparing the methods to each other rather than to bnd2, *CBgen* has the lowest LTS* in 879 of 1,000 cases.
- *CBgen* is two orders of magnitude faster than the other methods on average, which is unsurprising since those methods must run n analyses for each fit. A scatterplot of *CBgen* runtime vs. the number of data matrix elements ($m \times n$) is shown in Fig. 3 with a linear trendline in red. Solution time grows slowly with the number of matrix elements.

CBgen again dominates in this experiment, completing all datasets, with the smallest geo(R), and in the least amount of time.

The outlier identification algorithm identifies 100% of the outliers in 980 of the 1,000 datasets, and on average identifies 98.7% of the outliers. It misidentifies 13.0% of inliers as outliers on average, but these are usually reinstated in a later step of the *CBgen* algorithm.

There are 9 datasets for which the outlier identification routine finds none of the nominal outliers. In all 9 cases the output solution has R $<<$ 1, indicating a much better fit than the base hyperplane. All 9 cases have a single cluster

TABLE XI
RESULTS FOR UNCLUSTERED OUTLIERS.

	All (1000 data sets)			m/n $\geq$ 2 (807 data sets)	
Alg.	**avg. time|sd (s)**	**succ.**	**geo(R)|gsd**	**succ.**	**geo(R)|gsd**
Alg3-BM	115.21|137.12	**1000**	1699.76|3.05	**807**	1927.32|2.49
mbareg	591.67|956.17	397	**1.31**|1.74	397	1.31|1.74
bb	591.67|956.17	397	7.56|12.03	397	7.56|12.03
hbreg	591.67|956.17	397	1.43|1.94	397	1.43|1.94
OLS	726.71|1417.66	**1000**	803.46|1.72	**807**	767.86|1.48
LTS	531.59|688.66	189	188.93|7.24	189	188.93|7.24
LQS	527.16|699.96	189	200.64|7.19	189	200.64|7.19
CBreg	**3.62**|4.39	**1000**	1.41|2.41	**807**	**1.10**|1.29

TABLE XII
CLUSTERED OUTLIERS SUMMARY STATISTICS (1,000 DATASETS).

	m	**n**	**m/n**	**outFrac**	**clust**	**pts/clust**
avg:	1,436.62	370.45	6.37	0.304	5.53	424.08
min:	19.00	10.00	1.09	0.003	1.00	4.00
max:	3,969.00	1,000.00	151.94	0.500	10.00	3,969.00

TABLE XIII
OUTLIER CLUSTER IMBALANCES.

imbalance	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
datasets	162	349	221	118	77	38	25	5	5

of outliers, and a very high fraction of outliers (average 47%, smallest 43%). The better fitting hyperplane passes through the more densely packed outlier cluster, which constitutes nearly half of the points. The hyperplane also passes near some of the nominal inliers, and can do so at very little cost if the angle to the base hyperplane is small. We do not see any similar behavior in the unclustered outlier datasets.

The elements of the *CBgen* algorithm are quite effective, as shown on these difficult clustered outliers datasets. Let LTS_k^* be the value of LTS* for the kth hyperplane of the 3 generated by the steps in the algorithm. The geometric mean of LTS_2^*/LTS_1^* over the 1,000 datasets is 0.039, indicating a massive drop in LTS*. The geometric mean of LTS_3^*/LTS_2^* is 0.849, a further improvement. The number of points that are close to the hyperplane (i.e. within *maxDist*) also increases across the 3 hyperplanes. Let NC_k be the number of points that are close to hyperplane k. The geomean of NC_2/NC_1 over the 1,000 datasets is 2.88, and NC_3/NC_2 is 1.02.

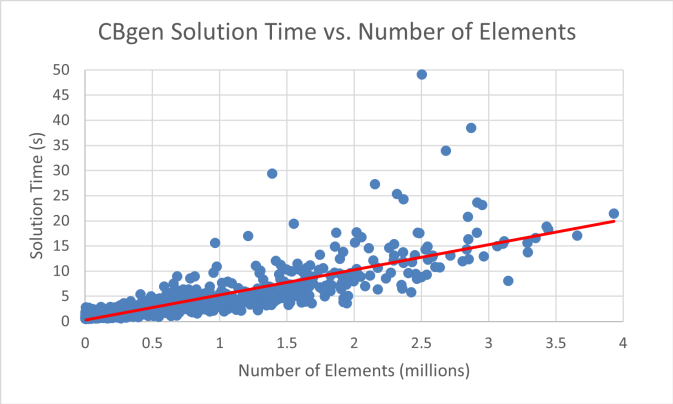

Fig. 3. CBgen Solution Time vs. Number of Elements on the Clustered Outliers Datasets.

Applying the ABM metric in *CBgen* has a minor impact on average. Geo(R) for the result from hyperplane 3 is 1.46; for the hyperplane chosen by the ABM metric it is 1.48. Closer analysis shows that using the ABM-selected hyperplane trades off slightly worse results in 205 cases versus very big improvements in R in 27 cases. It is interesting to note that there are 4 cases in which *CBgen* finds all of the outliers and yet still returns a solution with R < 1; in these cases the ABM criterion correctly selects the initial hyperplane as best.

VII. EXPERIMENTS: HEURISTICS FOR MINIMIZING THE QUANTILE ERROR

We compare the heuristic methods introduced in Section III to existing methods for minimizing the quantile error using real world and synthetic data.

A. Metrics

The synthetic datasets are not constructed so that the minimum quantile error is known in advance, so we compare the results of the various methods to each other. R_k^{best} is the ratio of γ_k, the quantile error found by some method k, to the smallest γ found by any of the compared methods. We calculate the geomean of these ratios, geo(R^{best}), for each method over each collection of datasets.

Solution speed is an important metric for these heuristics.

Throughout these experiments we minimize the median error by setting $q = \lfloor 0.5m \rfloor$.

B. Datasets

We use three data collections described previously (Olive, BM-like, RVD-like), and construct 4 additional synthetic data collections:

- *BM* regression collection: similar in size and construction to the datasets used by Bertsimas and Mazumder [15] to ease comparison. Many of the basic statistics are similar to the BM-like collection used in Section V, but the outlier fractions are much higher (35-40%) to match those in [15].
- *BM-small* regression collection: structurally similar to the *BM* collection but having fewer points and variables so that computation time is reduced for MIO instances. The goal is to have some instances achieve optimality within the time limits.

TABLE XIV
RESULTS FOR CLUSTERED OUTLIERS.

	All (1000 data sets)			$m/n \geq 2$ (815 data sets)	
Alg.	**avg. time|sd (s)**	**succ.**	**geo(R)|gsd**	**succ.**	**geo(R)|gsd**
Alg3-BM	118.2|140.57	**1000**	334.34|12.05	**815**	396.39|9.97
mbareg	554.45|960.26	397	2.05|2.90	397	2.05|2.90
bb	554.45|960.26	397	1.75|2.05	397	1.75|2.05
hbreg	554.45|960.26	397	2.97|3.61	397	2.97|3.61
OLS	682.18|1384.83	**1000**	77.0|7.19	**815**	69.14|7.97
LTS	552.86|664.31	202	15.53|18.77	202	15.53|18.77
LQS	533.89|642.83	202	15.33|17.78	202	15.33|17.78
CBreg	**3.59**|4.43	**1000**	**1.48**|2.48	**815**	**1.14**|1.60

TABLE XV
BASIC STATISTICS FOR MINIMIZING QUANTILE ERROR EXPERIMENTS.

		BM	**BM-small**	**RVD**	**COS**
	No. Datasets:	70	100	60	100
m	avg	2,629.57	42.59	533.33	63.16
	min	201.00	25.00	100.00	6.00
	max	10,001.00	60.00	1,000.00	185.00
n	avg	11.00	7.79	5.17	14.15
	min	6.00	4.00	3.00	3.00
	max	21.00	11.00	10.00	25.00
m/n	avg	184.80	6.13	103.33	6.23
	min	18.27	2.27	20.00	1.25
	max	476.24	14.25	200.00	46.25
outFrac	avg	0.41	0.39	0.38	0.30
	min	0.40	0.37	0.35	0.02
	max	0.50	0.40	0.40	0.50

- *RVD* regression collection: similar in size and construction to the datasets used by Rousseeuw and Van Driessen [14] to ease comparison, though we omitted the very small examples having just 2 variables. These datasets have larger proportions of outliers (35-40%) than the RVD-like datasets described in Section V.
- *Clustered-outliers-small (COS)* general fit collection: constructed as for the clustered outlier datasets in Section VI. The number of attributes n is sampled from a discrete uniform(3,25) distribution. There are 50 datasets with m_{in} sampled from a discrete uniform(n, $2n$) distribution and 50 datasets with m_{in} sampled from a discrete uniform($2n$,100) distribution. 30 of the 100 datasets are found to have a median error of 0 by one or more of the tested methods. In these cases, we assign $\mathrm{R}_k^{best} = 1$ if method k also finds a median error of 0, otherwise we omit the dataset from the geo(R^{best}) for method k. We omit 30 datasets for *MIO-BM-1st* and *MIO-CB-1st*, 9 for *Alg3-BM*, 10 for *LQS*, and none for *CBq*.

The basic statistics for the Olive collection are in Table VI, for BM-like in Table I, for RVD-like in Table IV (we use 80 of the 90 data sets; 10 are too large for solution by MIO), and for the new data collections in Table XV. The number of datasets in each collection appears below the collection name.

C. Compared Methods

CBq is compared with 4 other heuristics for minimizing the quantile error:

- *MIO-BM-1st*: the first solution returned by the Bertsimas and Mazumder MIO [15].
- *MIO-CB-1st*: the first solution returned by our *MIO-CB* formulation (Eqn. 5).
- *Alg3-BM*: our implementation of the Bertsimas and Mazumder Algorithm 3 heuristic [15] for minimizing the quantile error.
- *LQS*: the *LQS* implementation in the R library *MASS* [31].

D. Results

Results are summarized in Table XVI with best results in bold. "gsd" is the geometric standard deviation. In the COS data collection, the lowest γ is zero in 30 cases: all methods find this value and are assigned $R^{best} = 1$, except *Alg3-BM* which is unable to do so in 22 cases, which are ignored in calculating statistics. *LQS* has the lowest geo(R^{best}) in 4 of the 7 data sets, *CBq* in 2, and *Alg3-BM* in 1. Taking the first solution returned by either of the MIO formulations is uniformly much worse, though these solutions are often found very quickly. *Alg3-BM* is always much slower than all other methods while *LQS* and *CBq* are relatively quick.

LQS has the best overall results in terms of both geo(R^{best}) and solution speed, however the randomization in the algorithm means that the results are not repeatable.

The data collections are sorted from highest average outlier fraction (BM at 0.41) to lowest (Olive at 0.13) to see the effect of that factor. *LQS* and *CBq* generally provide better results at lower outlier fractions. This is expected behavior for *CBq* because the *CBreg* step is much better at identifying outliers below about 35% outliers.

VIII. EXPERIMENTS: MIXED-INTEGER OPTIMIZATION FOR MINIMIZING THE QUANTILE ERROR

The datasets are the same as those in Sec. VII. MIO solutions are limited to 3600 seconds total (including time for heuristic initialization). Metrics include R^{best}, solution time, and number of instances with solutions proved to be optimal within the time limit.

A. Compared Methods

We compare the two MIO formulations *MIO-BM* (Eqn. 1) and the new *MIO-CB* (Eqn. 5), both stand-alone and in conjunction with advanced starts provided by the heuristics *Alg3-BM, LQS* and *CBq*.

TABLE XVI
RESULTS FOR HEURISTIC MINIMIZATION OF QUANTILE ERROR.

	BM	BM-small	RVD	COS	BM-like	RVD-like	Olive
avg. outFrac	0.41	0.39	0.38	0.30	0.20	0.20	0.13
	geo(R^{best})|gsd						
MIO-BM-1st	13.95|1.80	23.32|2.33	9.79|1.36	7.98|6.63	20.47|1.63	18.78|1.38	3.07|1.93
MIO-CB-1st	7.87|1.94	18.82|2.42	6.58|1.24	5.54|6.37	12.89|1.67	12.23|1.44	3.04|1.95
Alg3-BM	**1.62**|2.04	1.35|3.31	1.43|1.88	7.24|9.17	2.40|1.86	1.72|2.41	2.94|3.22
LQS	1.99|2.49	**1.32**|1.40	**1.10**|1.16	**1.16**|1.34	1.31|1.27	1.06|1.09	**1.02**|1.03
CBq	2.94|2.26	4.60|3.79	1.52|2.01	1.68|3.58	**1.00**|1.01	**1.03**|1.05	1.35|1.38
	avg. time|sd (s)						
MIO-BM-1st	**1.32**|2.65	**0.02**|0.01	**0.08**|0.06	**0.03**|0.01	**0.85**|1.68	0.57|0.85	**0.02**|0.02
MIO-CB-1st	1.6|2.23	**0.02**|0.01	0.13|0.11	**0.03**|0.01	1.31|1.99	**0.28**|0.28	**0.02**|0.01
Alg3-BM	40.62|70.12	2.37|0.33	4.37|3.02	2.56|0.36	33.58|59.95	20.71|22.27	2.51|0.83
LQS	1.69|2.82	**0.02**|0.01	0.29|0.66	0.06|0.04	1.34|2.36	2.49|3.31	0.09|0.12
CBq	2.71|2.93	1.0|0.18	0.75|0.25	0.99|0.15	1.81|2.16	1.6|1.19	1.13|0.47

B. Results

Results are summarized in Table XVII, which groups the methods in *MIO-BM* vs. *MIO-CB* pairs that differ only by the initialization heuristic, providing 28 comparisons of the two over the 7 data collections. The best result in each column is in bold and the best result in each pair is in italics. In the COS collection, the lowest γ is zero in 30 cases: all methods return a non-zero value in one case which is ignored in calculating statistics, except *CBq/MIO-BM* which correctly returns $\gamma = 0$ in all 30 cases. Some observations:

- The *MIO-CB* formulation provides a better geo(R^{best}) than *MIO-BM* in 23 of the 28 comparisons, the same value in 2 comparisons, and worse values in 3 comparisons. Two of the worse results are for the MIOs without heuristic initialization.
- The best geo(R^{best}) for each data set collection is always provided by some combination of an initialization heuristic and *MIO-CB*.
- Restricting attention to the 21 comparisons in which initialization heuristics are used, *MIO-CB* provides a better geo(R^{best}) than *MIO-BM* in 18 of them and the same value in 2.
- The worst geo(R^{best}) values for *CBq* coupled with either MIO formulation are for the BM data collection. As shown in Table XVI, *CBq* does relatively poorly on this dataset as a stand-alone heuristic due to the high outlier fraction.
- Obscured in the overall geometric mean is the fact that *CBq* coupled with either MIO formulation provides the best R^{best} results for the 10 largest BM datasets (m=10,001, n=21, outlier fraction 40%). This is likely because the *CBq* runtime averages 6.90s over those 10 datasets vs. 210.50s for *Alg3-BM* (which provides the best geo(R^{best}) among the heuristics on the BM collection). The short *CBq* runtime leaves more time for the MIO branch and bound operations to improve the result, vs. initialization via *Alg3-BM*.

As shown in the bottom third of Table XVII, all methods time out on the BM and BM-like collections. In the remaining 20 comparisons, *MIO-CB* reaches the optimum solution more often than *MIO-BM* 18 times, less often once, and the same number of times twice, whether initialized by a heuristic method or not. For those same 20 comparisons, the middle third of the table shows that *MIO-CB* is always faster than *MIO-BM*, with an average speedup of about 10%.

Overall, the best results are returned by some combination of an initialization heuristic and *MIO-CB*, with no dominant advantage seen for any of the 3 initialization heuristics tested. *CBq/MIO-CB* may be the best choice for large datasets because the short *CBq* runtime leaves more time for MIO operations.

C. Scalability and Early Termination

The bottom third of Table XVII shows that the MIO methods reach optimality for very few of the data sets within the 60 minute time limit. Every method times out on every data set in the BM and BM-like collections. The best results are for *Alg3/MIO-CB*, *LQS/MIO-CB*, and *CBq/MIO-BM* on the BM-small collection: they reach optimality for 49 of the 100 data sets. This lack of scalability prompts interest in exiting the MIO solution early.

For general MIOs, it is often the case that the optimum solution is found early in the branch and bound search, with the rest of the time spent on eliminating possibly better solutions. For this reason the solver can sometimes be stopped early, before proving that it has found the optimum solution, without much impact on the outcome. We tested this by comparing the MIO results in Table XVII with those obtained when a 1 minute time limit is imposed.

Results are summarized in Table XVIII which shows the geomeans of the ratios of the γ values obtained with a 1 minute vs. 60 minute time limit. We expect the γ found at 1 minute to be greater than the γ found at 60 minutes, so these ratios should be ≥ 1. When $\gamma = 0$ under the 60 min time limit, we set the ratio at 1.0 if $\gamma = 0$ also under the 60 second time limit. For the MIOs that initialize via the *Alg3-BM* heuristic, 10 data sets are omitted from the BM collection, and 5 from the RVD-like collection, because the heuristic takes more than 60s to return a solution.

The table shows that the 1 minute γ geomeans are always larger than the 60 minute γ geomeans (ratios > 1), but they are similar (within 1%) in many cases. There is a wide range: only 4% of the 1 minute γ values are within 1% of the 60 minute γ values for *MIO-BM* on the BM data collection, but

TABLE XVII
RESULTS FOR MIO MINIMIZATION OF QUANTILE ERROR (60 MIN TIME LIMIT).

	BM	BM-small	RVD	COS	BM-like	RVD-like	Olive
	geo(R^{best}) |gsd						
MIO-BM	*3.53*|*2.65*	1.01|1.02	2.43|2.43	1.23|1.42	*2.83*|*3.25*	2.48|2.84	1.04|1.06
MIO-CB	5.69|3.02	***1.00***|*1.00*	*1.48*|*2.08*	*1.17*|*1.33*	3.58|4.38	*2.04*|*2.68*	***1.00***|*1.00*
Alg3-BM/MIO-BM	1.28|1.48	***1.00***|*1.02*	1.05|1.30	1.27|1.49	1.43|1.62	1.22|1.62	1.05|1.08
Alg3-BM/MIO-CB	***1.07***|*1.30*	***1.00***|*1.00*	***1.01***|*1.01*	***1.12***|*1.16*	*1.26*|*1.59*	*1.09*|*1.32*	*1.01*|*1.02*
LQS/MIO-BM	1.48|2.08	1.01|1.04	1.03|1.05	1.18|1.37	1.23|1.24	1.06|1.08	1.03|1.05
LQS/MIO-CB	*1.45*|*2.40*	***1.00***|*1.02*	***1.01***|*1.02*	***1.12***|*1.24*	*1.15*|*1.19*	*1.04*|*1.07*	*1.01*|*1.02*
CBq/MIO-BM	*2.18*|*2.44*	***1.00***|*1.02*	1.12|1.49	1.23|1.44	1.02|1.03	1.01|1.01	1.04|1.06
CBq/MIO-CB	2.70|2.49	***1.00***|*1.00*	*1.03*|*1.22*	*1.14*|*1.37*	***1.01***|*1.01*	***1.00***|*1.01*	*1.03*|*1.04*
	avg. time |sd (hours)						
MIO-BM	**1.00**|0.00	0.62|0.44	1.00|0.01	0.57|0.48	**1.00**|0.00	1.00|0.00	0.67|0.51
MIO-CB	**1.00**|0.00	**0.58**|0.46	**0.84**|0.36	**0.55**|0.49	**1.00**|0.00	**0.93**|0.19	0.55|0.50
Alg3-BM/MIO-BM	**1.00**|0.00	0.62|0.44	0.99|0.06	0.58|0.48	**1.00**|0.00	1.00|0.00	0.67|0.51
Alg3-BM/MIO-CB	**1.00**|0.00	**0.58**|0.46	**0.84**|0.35	**0.55**|0.49	**1.00**|0.00	**0.93**|0.19	0.56|0.50
LQS/MIO-BM	**1.00**|0.00	0.62|0.44	0.98|0.07	0.57|0.49	**1.00**|0.00	1.00|0.00	0.67|0.51
LQS/MIO-CB	**1.00**|0.00	0.59|0.46	**0.84**|0.36	**0.55**|0.49	**1.00**|0.00	**0.93**|0.20	**0.54**|0.51
CBq/MIO-BM	**1.00**|0.00	0.62|0.44	0.99|0.05	0.57|0.49	**1.00**|0.00	1.00|0.00	0.67|0.51
CBq/MIO-CB	**1.00**|0.00	0.59|0.46	**0.84**|0.36	**0.55**|0.49	**1.00**|0.00	0.94|0.18	0.57|0.49
	No. Optimum Solutions						
data sets	*70*	*100*	*60*	*100*	*70*	*80*	*6*
MIO-BM	0	48	2	45	0	0	2
MIO-CB	0	48	**10**	**47**	0	**10**	**3**
Alg3-BM/MIO-BM	0	47	2	44	0	0	2
Alg3-BM/MIO-CB	0	**49**	**10**	**47**	0	**10**	**3**
LQS/MIO-BM	0	47	4	44	0	0	2
LQS/MIO-CB	0	**49**	**10**	**47**	0	9	**3**
CBq/MIO-BM	0	**49**	4	44	0	0	2
CBq/MIO-CB	0	48	**10**	46	0	**10**	**3**

TABLE XVIII
IMPACT OF MIO TIME LIMIT ON MINIMIZING QUANTILE ERROR.

	BM	BM-small	RVD	COS	BM-like	RVD-like	Olive
	geomean gamma ratio (1 min: 60 min time limits)						
MIO-BM	2.47	1.13	1.56	1.40	2.61	1.75	1.10
MIO-CB	*1.71*	*1.08*	*1.35*	*1.28*	*2.32*	*1.16*	*1.04*
Alg3-BM/MIO-BM	1.19	1.12	1.07	1.28	*1.16*	*1.20*	1.06
Alg3-BM/MIO-CB	*1.13*	*1.09*	*1.02*	*1.22*	*1.16*	1.22	*1.04*
LQS/MIO-BM	1.21	*1.07*	1.06	1.27	*1.04*	*1.01*	1.03
LQS/MIO-CB	*1.11*	*1.07*	*1.02*	*1.19*	1.08	1.02	*1.02*
CBq/MIO-BM	1.45	*1.15*	1.12	1.34	1.02	*1.01*	1.04
CBq/MIO-CB	*1.15*	*1.15*	*1.04*	*1.21*	*1.01*	*1.01*	*1.03*
	no. gamma ratios $\leq$ 1.01						
data sets	*70*	*100*	*60*	*100*	*70*	*80*	*6*
MIO-BM	3	66	10	46	3	25	2
MIO-CB	*8*	*87*	*27*	*57*	*12*	*44*	*4*
Alg3-BM/MIO-BM	9	68	30	49	2	*42*	*4*
Alg3-BM/MIO-CB	*17*	*91*	*40*	*60*	*18*	41	2
LQS/MIO-BM	17	78	27	50	*29*	54	*4*
LQS/MIO-CB	*28*	*89*	*38*	*61*	22	*56*	*4*
CBq/MIO-BM	25	71	26	47	38	46	*3*
CBq/MIO-CB	*38*	*89*	*33*	*59*	*45*	*63*	*3*

91% of the 1 minute γ values are within 1% of the 60 minute γ values for *Alg3-BM/MIO-CB* on the BM-small data collection. *MIO-CB* generally benefits less from additional runtime than *MIO-BM*, indicating that it does a better job of finding good solutions early.

IX. EXPERIMENT: DOES MINIMIZING THE QUANTILE ERROR ALSO MINIMIZE LTS*?

It is often assumed that minimizing the quantile error will produce a good fit to the inliers in a data set, as measured here by LTS*. This experiment tests that assumption. We compare the LTS* associated with the smallest γ solution produced by any of the 8 γ-minimizing MIOs (see e.g. Table XVII) with the smallest LTS* produced by any of the 8 LTS*-minimizing heuristics (see e.g. Table II). We analyze the 486 data sets in the Olive, BM, RVD, COS, BM-small, BM-like, and RVD-like data sets.

For each data set we calculate the ratio of the smallest LTS* found by any of the LTS*-minimizing heuristics to the smallest LTS* produced by any MIO finding the smallest γ (there are frequently ties for the smallest γ). This ratio is labelled "Heuristic LTS* / LTS* from Minimum Gamma MIO" in Fig. 4 which summarizes the results. The vertical axis has been truncated for display purposes. This omits 24 data points, all

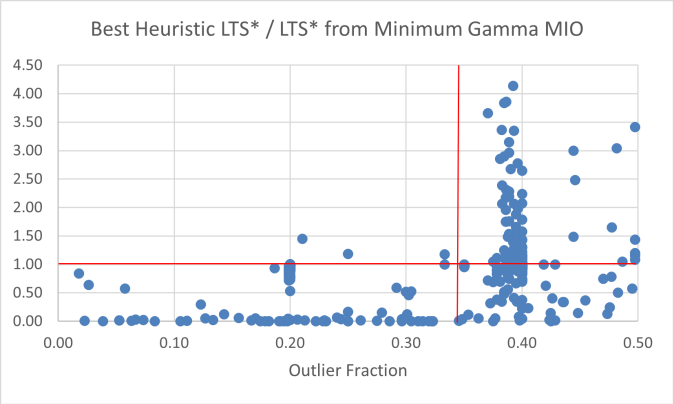

Fig. 4. Ratio of smallest LTS* from LTS*-minimizing Heuristic to LTS* from Smallest γ MIO.

of which are at an outlier fraction of 0.375 or greater.

Ratios less than 1.0 (below the horizontal red line) mean that the LTS*-minimizing heuristics produce a better value of LTS* than the γ-minimizing MIO, and the reverse is true for ratios greater than 1.0. The MIO LTS* results are worse than those produced by the LTS*-minimizing heuristics up to an outlier fraction of about 0.35 (the vertical red line), and beyond that outlier fraction the results are more mixed, with the MIO results often better.

This happens because γ-minimization is a different problem than LTS* minimization. The MIO minimizes gamma for 50% of the data points, so when the outlier fraction is near 0.5 the best subset of q points for γ-minimization consists mostly or entirely of the inliers (outlier points increase γ). Hence for outlier fractions near 0.5, the γ-minimizing fit will also be a good LTS*-minimizing fit, and in fact will often be better because the (much slower) MIO identifies the inliers correctly. However when the outlier fraction is much lower than 0.5 the two problems diverge: the number of inliers is greater than q, so the MIO finds a solution over only a subset of them, and this gives it the freedom to find a γ-minimizing solution that differs considerably from the minimum LTS* solution.

We conclude that γ minimization should not be used to fit a hyperplane to the inliers (i.e. to minimize LTS*) unless the number of outliers is known accurately in advance (so that q can be set appropriately). This is unlikely to be the case in practice.

X. Conclusions

This paper describes several new algorithms for fitting hyperplanes to large datasets that are contaminated by outliers, including clustered outliers:

- The *CBgen* algorithm for fitting hyperplanes to general datasets provides better fits and is faster than competing alternatives, even though the competing methods are given advance knowledge of the number of inliers where *CBgen* is not. It also scales well.
- Conclusions are similar for the *CBreg* algorithm for regression, though at least one other method (*mbareg*) provides similar results faster on at least one data collection (RVD-like). The *CBreg* and *CBgen* heuristics are deterministic so results are repeatable.
- *CBq* is a promising heuristic for minimizing the quantile error. It provided the best results on 2 of the 7 data collections in our experiments, with better results when the outlier fraction is less than about 0.35. Solution times are relatively short.
- *MIO-CB* improves on the state of the art for exact minimization of the quantile error. It is a smaller and more efficient MIO formulation. It provides better solutions in less time, and reaches an optimum solution more frequently than the existing *MIO-BM* formulation. Results are improved with heuristic initialization.

Our computational results further demonstrate:

- *CBq* is a better initialization heuristic when running MIO on large datasets under a time limit because its shorter runtime leaves more time for the MIO to run.
- MIO solutions for minimizing the quantile error do not scale well. Early termination can often provide a γ that is a relatively small multiple of solutions found much later in the branch and bound process.
- Minimizing the quantile error is not a good way to generate a hyperplane that has a small LTS* when the outlier fraction is dissimilar to q/m. Since the outlier fraction is not normally known in advance, we do not recommend quantile minimization for fitting hyperplanes when the goal is to minimize LTS*.

There are a variety of other robust scoring systems. It is worth investigating whether substituting a different one in place of Eqn. 3 improves results for *CBreg* and *CBgen*.

The framework used by *CBreg* and *CBgen* includes identifying potential outliers, reinstating inliers, and producing a robust final fit. This same framework could be used with any fitting method for the intermediate and final steps.

References

[1] A. Rencher and W. Christensen, *Methods of Multivariate Analysis*, 3rd ed. New Jersey: Wiley, 2012.
[2] A. Schöbel, *Locating Lines and Hyperplanes: Theory and Algorithms*, 1st ed. New York: Springer, 1999.
[3] J. Fang and H. Li, "Hyperplane-Based Vector Quantization for Distributed Estimation in Wireless Sensor Networks," *IEEE Transactions on Information Theory*, vol. 55, no. 12, pp. 5682–5699, 2009.
[4] H. Cevikalp, "Best Fitting Hyperplanes for Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1076–1088, 2016.
[5] J. Brooks and J. Dulá, "The L1-Norm Best-Fit Hyperplane Problem," *Applied Mathematics Letters*, vol. 26, pp. 51–55, 2013.
[6] H. Martini and A. Schöbel, "Median Hyperplanes in Normed Spaces - A Survey," *Discrete Applied Mathematics*, vol. 89, no. 1-3, pp. 181–195, 1998.
[7] P. Bloomfield and W. Steiger, *Least Absolute Deviations*. Boston, MA: Birkhäuser Boston, 1984. [Online]. Available: http://link.springer.com/10.1007/978-1-4684-8574-5
[8] R. Reris and J. Brooks, *Principal Component Analysis and Optimization: A Tutorial*. INFORMS, 2015, pp. 212–225.
[9] D. Rocke and D. Woodruff, "Identification of Outliers in Multivariate Data," *Journal of the American Statistical Association*, vol. 91, no. 435, pp. 1047–1061, 1996.
[10] P. Rousseeuw and M. Hubert, "Anomaly Detection by Robust Statistics," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, p. e1236, 2018.

[11] A. Zimek and P. Filzmoser, "There and Back Again: Outlier Detection between Statistical Reasoning and Data Mining Algorithms," *WIREs Data Mining and Knowledge Discovery*, vol. 8, p. e1280, 2018.

[12] P. Rousseeuw, "Least Median of Squares Regression," *Journal of the American Statistical Association*, vol. 79, pp. 871–880, 1984.

[13] D. Olive, *Robust Multivariate Analysis*. Springer International Publishing AG, 2017.

[14] P. Rousseeuw and K. Van Driessen, "Computing LTS Regression for Large Data Sets," *Data Mining and Knowledge Discovery*, vol. 12, pp. 29–45, 2006.

[15] D. Bertsimas and R. Mazumder, "Least Quantile Regression via Modern Optimization," *The Annals of Statistics*, vol. 42, no. 6, pp. 2494–2525, 2014.

[16] V. Blanco, J. Puerto, and R. Salmerón, "Locating Hyperplanes to Fitting Set of Points: A General Framework," *Computers and Operations Research*, vol. 95, pp. 172–193, 2018.

[17] J. P. Brooks, J. H. Dulá, and E. L. Boone, "A Pure L1-Norm Principal Component Analysis," *Computational Statistics and Data Analysis*, vol. 61, pp. 83–98, 2013.

[18] P. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, pp. 73–101, 1964.

[19] F. Hampel, "Beyond Location Parameters: Robust Concepts and Methods," *Bulletin of the International Statistics Institute*, vol. 46, pp. 375–382, 1975.

[20] P. Rousseeuw and A. Leroy, *Robust Regression and Outlier Detection*. New York: Wiley, 1987.

[21] O. Hössjer, "The Change-of-Variance Function for Dependent Data," *Probability Theory and Related Fields*, vol. 90, no. 4, pp. 447–467, 1991.

[22] D. Olive, "Two Simple Resistant Regression Estimators," *Computational Statistics and Data Analysis*, vol. 49, pp. 809–819, 2005.

[23] M. Hubert, P. Rousseeuw, and K. Vanden Branden, "ROBPCA: A New Approach to Robust Principal Component Analysis," *Technometrics*, vol. 47, no. 1, pp. 64–79, 2005.

[24] J. Ahn, M. Lee, and J. Lee, "Distance-Based Outlier Detection for High Dimension, Low Sample Size Data," *Journal of Applied Statistics*, vol. 46, no. 1, pp. 13–29, 2019.

[25] H. Chung and J. Ahn, "Subspace Rotations for High-Dimensional Outlier Detection," *Journal of Multivariate Analysis*, vol. 183, p. 104713, 2021.

[26] C. Aggarwal, *Outlier Analysis*, 2nd ed. New York: Springer, 2017.

[27] F. Grubbs, "Sample Criteria for Testing Outlying Observations," *Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 27–58, 1950.

[28] B. Rosner, "Percentage Points for a Generalized ESD Many-Outlier Procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, 1983.

[29] R. Killick, P. Fearnhead, and I. Eckley, "Optimal Detection of Changepoints with a Linear Computational Cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012.

[30] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022. [Online]. Available: https://www.gurobi.com

[31] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0. [Online]. Available: https://www.stats.ox.ac.uk/pub/MASS4/

[32] M. Maechler, P. Rousseeuw, C. Croux, V. Todorov, A. Ruckstuhl, M. Salibian-Barrera, T. Verbeke, M. Koller, E. Conceicao, and M. Anna di Palma, *robustbase: Basic Robust Statistics*, 2022, r package version 0.95-0. [Online]. Available: http://robustbase.r-forge.r-project.org/