

Name of student: Diganta Bhattacharya (MB2015)

Due on May 27, 2022

Paper: Least Median of Squares Regression by Peter J. Rousseeuw. ^{[Link](#)}A. Problem Setup:

Consider the linear multiple regression model

$$y_i = x_{i1}\theta_1 + x_{i2}\theta_2 + \dots + x_{ip}\theta_p + \sigma e_i = \mathbf{x}_i^t \boldsymbol{\theta} + \sigma e_i$$

for $i = 1, \dots, n$. The p -dimensional vectors \mathbf{x}_i contain the explanatory variables, y_i is the response and σe_i is the error term. The data consisting of n observations is denoted by $Z = (X, \mathbf{y})$. For a given parameter estimate $\hat{\theta}$ we denote the residuals as $r_i(\hat{\theta}) = y_i - \mathbf{x}_i^t \hat{\theta}$. In a regression model with intercept, the observations satisfy $x_{ip} \equiv 1$.

B. Methodology:

A robust regression method tries to estimate the regression parameter vector θ in such a way that it fits the bulk of the data even when there are outliers. We will denote for any numbers u_1, \dots, u_n ; $u_{i:n}$ as the i -th order statistic.

B.(i) Least Median of Squares:

Let $Z = (X, \mathbf{y})$ be a data set of n observations. Then the least median of squares (LMS) estimate for all $p \leq [n/2] + 1$, n odd, $\hat{\theta}_{LMS}(Z)$ is defined as

$$\hat{\theta}_{LMS}(Z) = \underset{\theta}{\operatorname{argmin}} [\underset{n}{\operatorname{median}} (r^2(\theta))] = \underset{\theta}{\operatorname{argmin}} [\underset{n}{\operatorname{median}} |r(\theta)|]$$

B.(ii) Other Similar Variants:

The generalization of (LMS) is the least quantile of squares (LQS) estimate $\hat{\theta}_{LQS}(Z)$ for all $p \leq h \leq n$, is defined as

$$\hat{\theta}_{LQS}(Z) = \underset{\theta}{\operatorname{argmin}} (r^2(\theta))_{h:n} = \underset{\theta}{\operatorname{argmin}} |r(\theta)|_{h:n}$$

For n odd and $h = [n/2] + 1$ the (LQS) becomes the (LMS) which minimizes the median of the squared residuals. Another alternative is the least trimmed squares (LTS) estimate $\hat{\theta}_{LTS}(Z)$, defined by

$$\hat{\theta}_{LTS}(Z) = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^h (r^2(\theta))_{i:n}.$$

With the parameter estimates $\hat{\theta}_{LQS}(Z)$ and $\hat{\theta}_{LTS}(Z)$ we can write down estimators of the error scale σ :

$$s_{LQS}(Z) = s_{LQS}(X, \mathbf{y}) = c_{h,n} \left| r \left(\hat{\theta}_{LQS}(Z) \right) \right|_{h:n}, \quad s_{LTS}(Z) = s_{LTS}(X, \mathbf{y}) = d_{h,n} \sqrt{\frac{1}{h} \sum_{i=1}^h \left(r^2 \left(\hat{\theta}_{LTS}(Z) \right) \right)_{i:n}}$$

The constants $c_{h,n}$ and $d_{h,n}$ are chosen such that the scale estimators consistent with the Gaussian model, which gives

$$c_{h,n} = 1/\Phi^{-1}\left(\frac{h+n}{2n}\right), \quad d_{h,n} = 1/\sqrt{1 - \frac{2n}{hc_{h,n}}\phi(1/c_{h,n})}$$

s_{LQS} is multiplied by a finite-sample correction factor. For $h = [n/2] + 1$, (i.e) s_{LMS} this factor equals $1 + \frac{5}{n-p}$. More efficient scale estimates, based on the preliminary ones, are then given by

$$\hat{\sigma} = \sqrt{\frac{\sum_i w_i r_i^2}{\sum_i w_i - p}}, \quad w_i = \begin{cases} 0 & \text{if } |r_i/s_{L(Q-T)S}| > 2.5 \\ 1 & \text{otherwise.} \end{cases}$$

Here the notation $s_{L(Q-T)S}$ means s_{LQS} or s_{LTS} , whichever is used.

C. Assumptions and Underlying Properties of the Estimator:

We will assume that the original X is in general position. This means that no p of the \mathbf{x}_i lie on a $p-1$ dimensional plane through the origin. For simple regression with intercept ($p=2$) this says that no two x_i coincide. For simple regression without intercept ($p=1$) it says that none of the x_i are zero.

C.(i) Theorem on Existence:

A solution to the following: $\hat{\theta}_{LMS}(Z) = \underset{\theta}{\operatorname{argmin}}[\underset{n}{\operatorname{median}}|r(\theta)|]$ always exists.

C.(ii) Theorem on Breakdown Point:

The finite-sample breakdown value of any regression estimator $T(Z) = T(X, \mathbf{y})$ is given by

$$\varepsilon_n^* = \varepsilon_n^*(T, Z) = \min \left\{ \frac{m}{n}; \sup_{Z'} \|T(Z')\| = \infty \right\}$$

where $Z' = (X', \mathbf{y}')$ ranges over all data sets obtained by replacing any m observations of $Z = (X, \mathbf{y})$ by arbitrary points. It is the smallest amount of contamination that can cause the estimator to take on values arbitrarily far from $T(Z)$. For (LMS) we have the following:

- If $p > 1$ and the observations are in general position, then the breakdown point of the (LMS) method is $([\frac{n}{2}] - p + 2)/n$.
- If $p = 1$ and $x_{ip} = 1$ for all observations, the regression model reduces to the univariate model $y_i = \mu + \sigma e_i$. If

$$m_T^2 : \underset{n}{\operatorname{median}}[r^2] = \underset{n}{\operatorname{median}}[(y_i - T)^2] = \underset{\theta}{\min}[\underset{n}{\operatorname{median}}(y_i - e)^2],$$

then both $T - m_T$ and $T + m_T$ are observations in the sample.

If the \mathbf{x}_i are in general position, then the finite-sample breakdown value of the (LQS) and the (LTS) is

$$\varepsilon_n^* = \begin{cases} (h-p+1)/n & \text{if } p \leq h < [\frac{n+p+1}{2}] \\ (n-h+1)/n & \text{if } [\frac{n+p+1}{2}] \leq h \leq n \end{cases}$$

C.(iii) Other Properties/Corollaries:

If the \mathbf{x}_i are in general position, then the maximal finite-sample breakdown value of the (*LQS*) and the (*LTS*) equals

$$\max_h \varepsilon_n^* = \frac{[(n-p)/2] + 1}{n}$$

and is achieved for

$$[(n+p)/2] \leq h \leq [(n+p+1)/2]$$

When $n+p$ is even we have $[(n+p)/2] = [(n+p+1)/2]$ hence the optimal h is unique. When $n+p$ is odd, it turns out that choosing $h = [(n+p+1)/2]$ gives the better finite-sample efficiency. Therefore, we will always define the optimal h as

$$h_{opt} = [(n+p+1)/2].$$

D. Estimating The Value:

The least-median-of-squares (*LMS*) method estimates the parameters by solving the nonlinear minimization problem:

$$\min \left[\text{median}_i(r_i) \right]$$

- Unlike the M-estimators, however, the (*LMS*) problem cannot be reduced to a weighted least-squares problem.
- It turns out that this method is very robust to false matches as well as outliers due to bad localization.
- It must be solved by a search in the space of possible estimates generated from the data. Since this space is too large, only a randomly chosen subset of data can be analyzed.
- The usual algorithm is based on sampling subsets of points from the data in order to generate candidate LMS estimates. The size of each subset is determined by the number of coefficients to be estimated.

The algorithm which we describe below is for robustly estimating a conic.

D.(i) Conic Fitting Problem:

The problem is to fit a conic section to a set of n points $\{z_i\} = \{(x_i, y_i) : i = 1, \dots, n\}$. A conic can be described by the following equation:

$$Q(x, y) = Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0$$

where A and C are not simultaneously zero. In practice, we encounter ellipses, where we must impose the constraint $B^2 - 4AC < 0$. However, this constraint is usually ignored during the fitting because

- the constraint is usually satisfied if data are not all situated in a flat section.
- the computation will be very expensive if this constraint is considered.

As the data are noisy, it is unlikely to find a set of parameters (A, B, C, D, E, F) (except for the trivial solution $A=B=C=D=E=F=0$) such that $Q(x_i, y_i) = 0, \forall i$. In order to avoid it, we should normalize $Q(x, y)$ which can be done as any of the following ways:

$$1. A + C = 1 \quad 2. F = 1 \quad 3. A^2 + B^2 + C^2 + D^2 + E^2 + F^2 = 1$$

D.(ii) Approach Overview:

- Consider n points, $m_i = [x_i, y_i]^T$. A Monte Carlo type technique is used to draw m random sub-samples of p different points. For the problem at hand, we select five ($p = 5$) points because we need at least five points to define a conic.
- For each sub-sample, indexed by J , we use any of the normalization techniques described above to compute the conic parameters p_J .
- For each p_J , we can determine the median of the squared residuals M_J , with respect to the whole set of points,

$$M_J = \underset{i=1,..,n}{\text{median}}[r_i^2(p_J, m_i)]$$

- Here, we have a number of choices for $r_i(p_J, m_i)$, the residual of the i^{th} point with respect to the conic p_J . Depending on the demanding precision, one can use the algebraic distance, the Euclidean distance, or the gradient weighted distance.
- We retain the estimate p_J for M_J is minimal among all m M_J 's.

D.(iii) How do we determine m ?

A sub-sample is “good” if it consists of p good data points. Assuming that the whole set of points may contain up to a fraction ϵ of outliers, the probability that at least one of the m sub-samples is good is given by

$$P = 1 - [1 - (1 - \epsilon)^p]^m$$

By requiring that P must be near 1, one can determine m for given values of P and ϵ :

$$m = \frac{\log(1 - P)}{\log(1 - (1 - \epsilon)^p)}$$

(LMS) efficiency is poor in the presence of Gaussian noise. The efficiency of a method is defined as the ratio between the lowest achievable variance for the estimated parameters and the actual variance provided by the given method. Based on robust standard deviation estimate given by

$$\hat{\sigma} = 1.4826[1 + 5/(n - p)]\sqrt{M_J},$$

we can assign a weight for each correspondence: $w_i = \begin{cases} 0 & \text{if } \frac{r_i^2}{\hat{\sigma}^2} > 2.5 \\ 1 & \text{otherwise.} \end{cases}$ where r_i is the residual of the i^{th} point with respect to the conic p . The correspondences having $w_i = 0$ are outliers and should not be further taken into account. The conic p is finally estimated by solving the weighted least-squares problem:

$$\min_p \sum_i w_i r_i^2$$

We can thus robustly estimate the conic because outliers have been detected and discarded by the LMS method.

1 The Data Used

The following are the descriptions of the data used for applying LMS method.

1.1 The Conic Data- Implemented in R

Data points from a conic with outliers present are obtained from the following steps.

- In the equation for conic as described before, true parameters are fixed at some values as A, B, C, D, E, F . The true curve is

$$Ax^2 + 2Bxy + Cy^2 + 2Dx + 2Ey + F = 0$$

- $F = 1$ is fixed to have five independent parameters. Otherwise, there would have been many versions of the same equation with scaled parameters.
- Data is generated for the true curve by taking x values arbitrarily and solving the corresponding quadratic equation in y .
- An error distribution is fixed by taking a mean zero Gaussian error contaminated with a degenerate outlier value.
- Adding i.i.d errors from the contaminated distribution, we get data points with outliers present.
- On four different sets of such data points, the LMS method described in the previous section is applied and the fitted values and estimates are shown next.

1.2 The Linear Data- Implemented in Python

The following steps are used for linear data used in the following section.

- The true slope and intercept value is fixed for the line.
 - Random points are chosen on the X axis and corresponding y values stored.
 - Error is added to the y values in the points taken from the line.
 - The resulting points are used for applying the LMS method.
-

2 Results and Plots from R (Conic Data I,II)

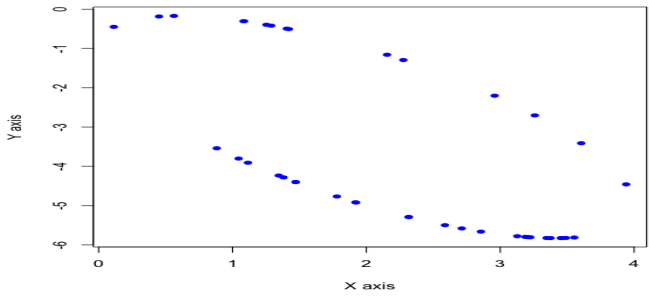
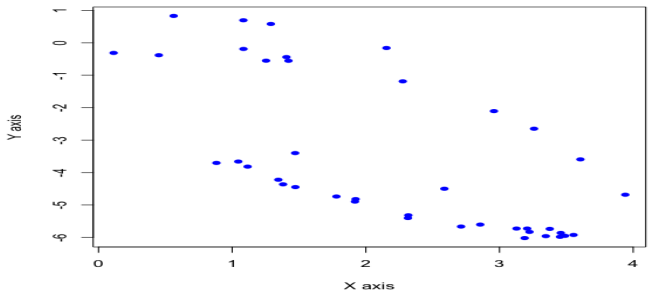
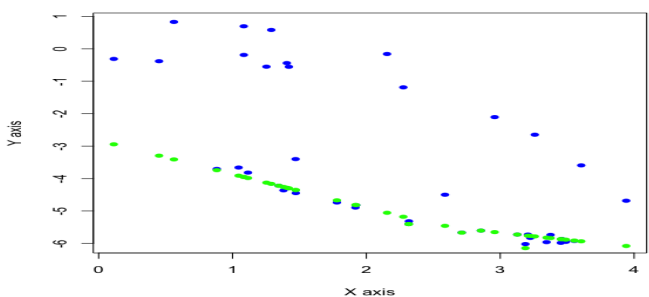
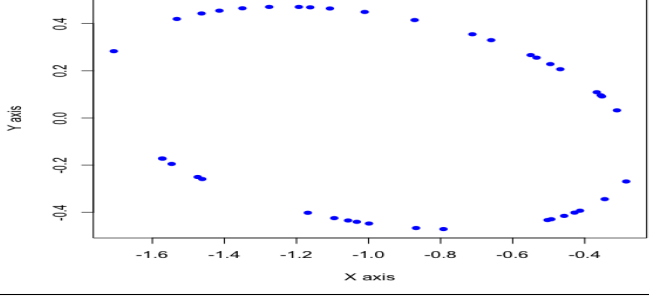
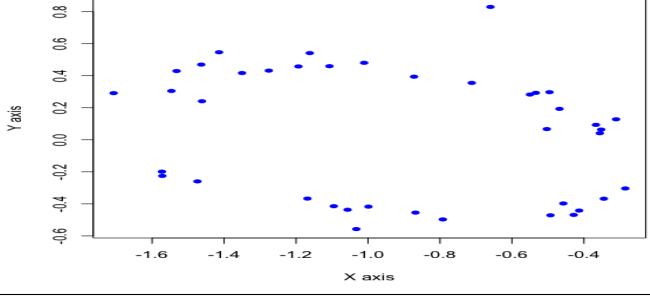
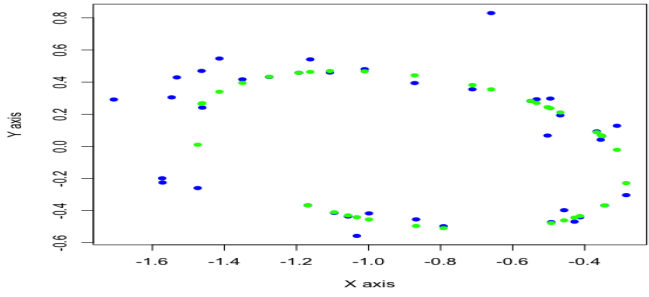
Simulated Data I	Contaminated Data I
	
Fitted Data I	Comments
	<ul style="list-style-type: none"> • The parameters are $A = 2, B = 1, C = 1, D = -1, E = 1, F = 1$. • The estimated parameters are $A = 1.76, B = 0.82, C = 1.05, D = -0.86, E = 1.1$. • The minimized median of squared residual = 2.5×10^{-5}
Simulated Data II	Contaminated Data II
	
Fitted Data II	Comments
	<ul style="list-style-type: none"> • Parameters $A = 2, B = 1, C = 5, D = 2, E = 1, F = 1$ • Estimates for $A = 2.13, B = 0.80, C = 3.22, D = 1.90, E = 0.79$ • The minimized median of squared residual = 0.0030

Table 1: Plots and Results for Simulated Data I,II

3 Results and Plots from R (Conic Data III,IV)

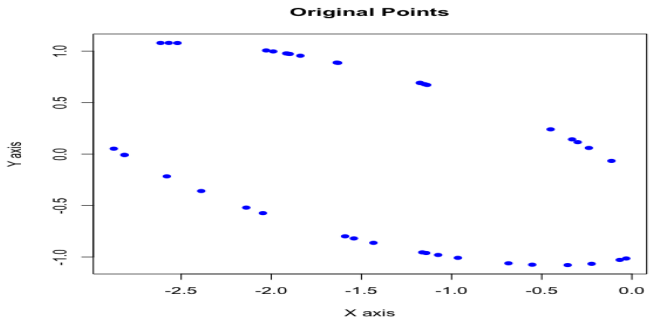
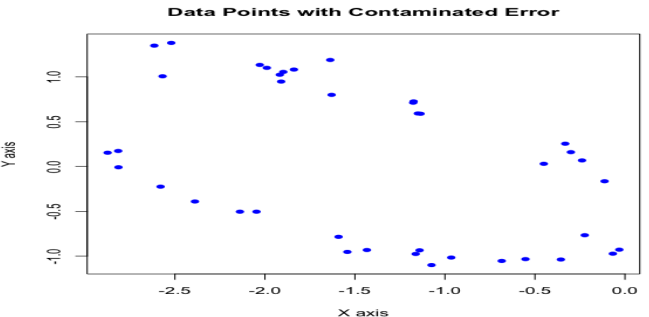
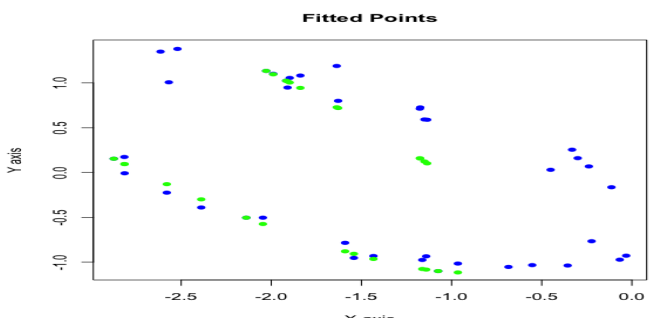
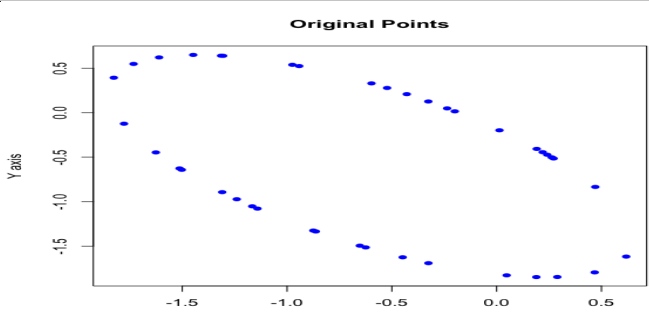
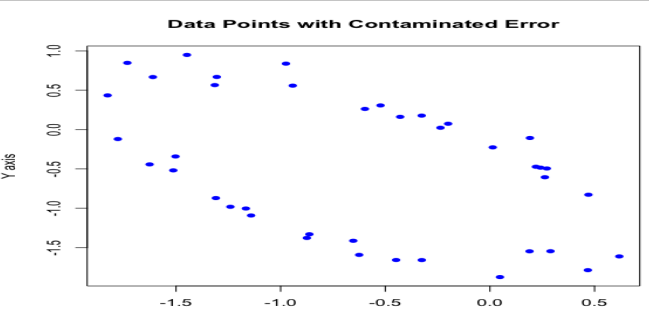
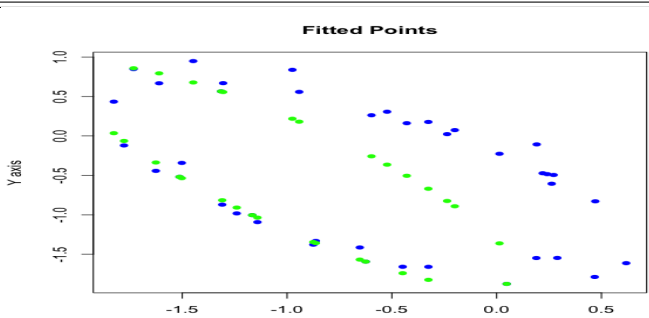
Simulated Data III	Contaminated Data III
	
Fitted Data III	Comments
	<ul style="list-style-type: none"> Parameters : $A = 2, B = 2, C = 5, D = 3, E = 3, F = 1$ Estimates for $A = 1.65, B = 1.5, C = 4.34, D = 3.14, E = 2.29$. Minimized median of squared residuals = 0.0034.
Simulated Data IV	Contaminated Data IV
	
Fitted Data IV	Comments
	<ul style="list-style-type: none"> Parameters: $A = 3, B = 2, C = 3, D = 3, E = 3, F = 1$ Estimates for $A = 2.69, B = 1.44, C = 2.39, D = 2.90, E = 2.63$ Minimized median of squared residuals = 0.0060.

Table 2: Plots and Results for Simulated Data III,IV

4 Results and Plots from Python 1D,2D Case

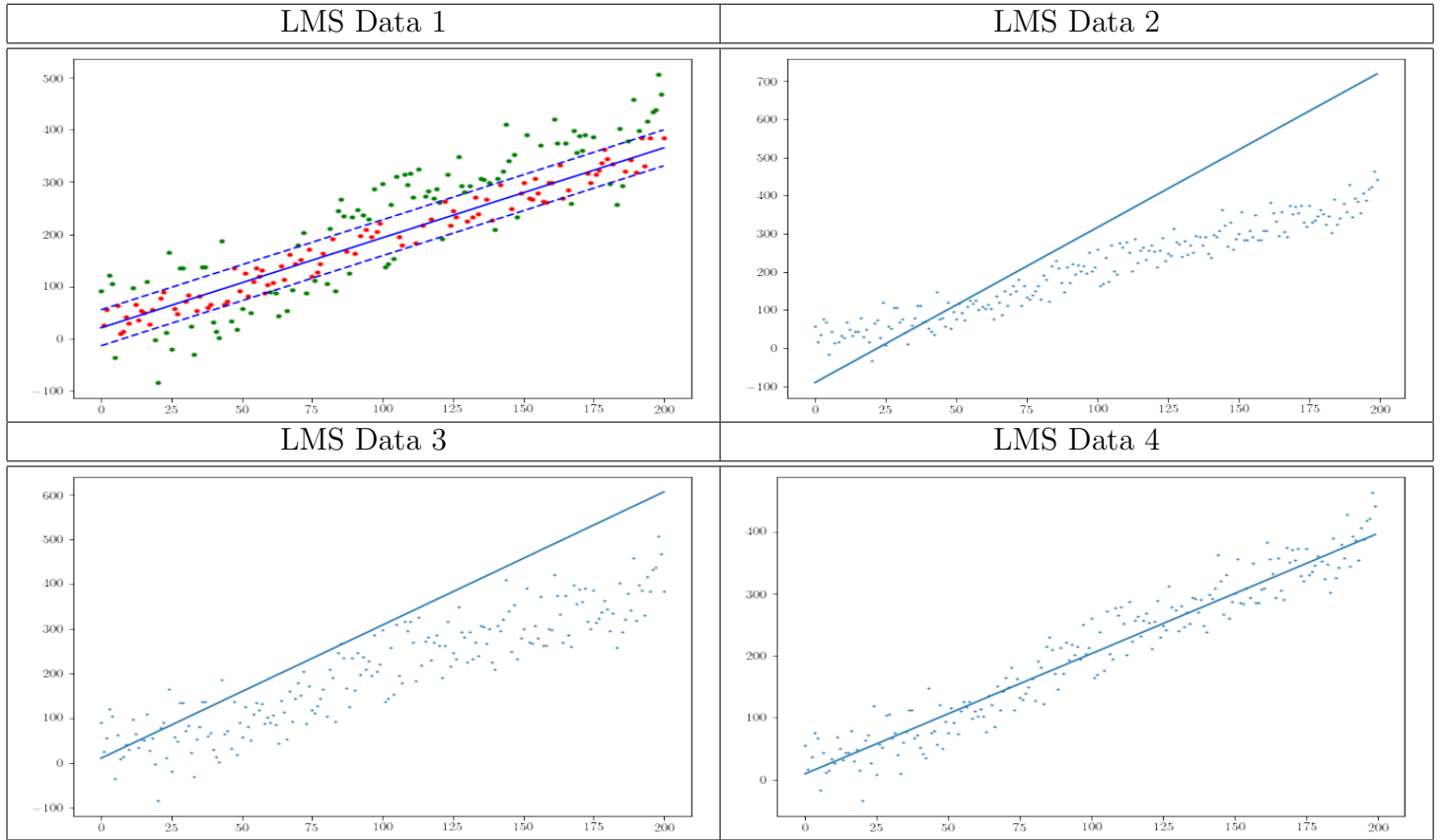


Table 3: Plots for LMS for different Data

```

1 #For the Plots the Main.py code needs this#
2 theta0s = np.linspace(-100, 100, 200)
3 theta1s = np.linspace(-5, 5, 200)
4 costs = [[f([theta0, theta1]) for theta0 in theta0s] for theta1 in theta1s]
5 plt.contour(theta0s, theta1s, costs, 50)
6 plt.xlabel('$\\theta_0$')
7 plt.ylabel('$\\theta_1$')
8 plt.colorbar()

```

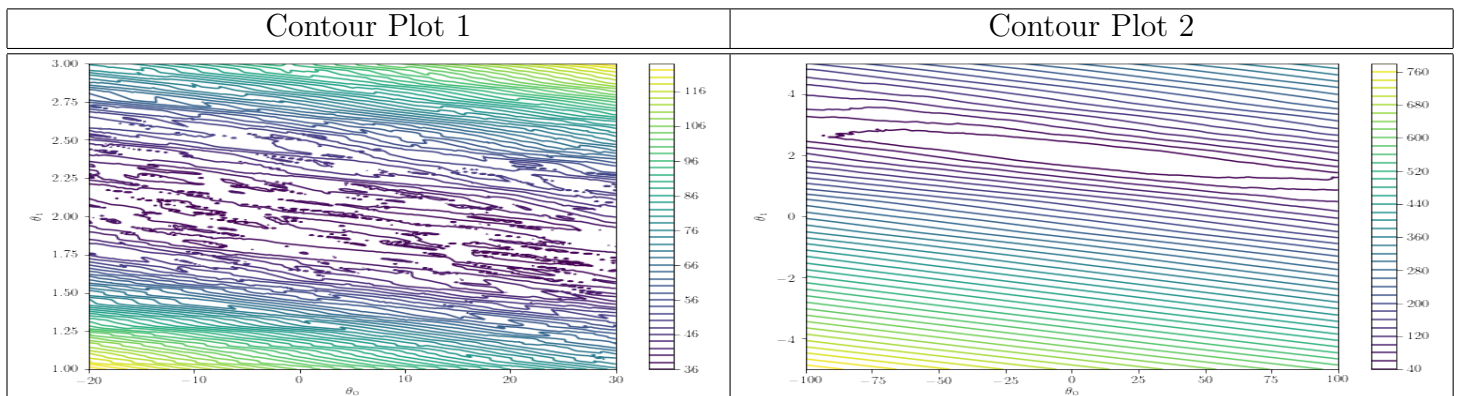


Table 4: Contour Plots To Show Dependence on Initial Guess

5 Comments on The Simulated Data and Plots:

- Conic Plots: The fitted points fairly approximate the original conic points taken even after adding error with a contaminated distribution. This suggests that the LMS provides a robust method for fitting curves of a particular form.
 - Line Fitting Plots: The line plot 1 displays how the LMS can be used to identify outliers for a given set of data points.
 - Contour Plots: For a data set with large number of points, the possibilities for the minimizing line increases. The contour points display that a good initial guess is required for implementing the minimizing algorithm.
-

6 Comparison with Least Squares Estimate

1. Breakdown Point and Properties:

It can be said that LS has a breakdown point of 0%, whereas the breakdown point of the LMS technique is as high as 50%, the best that can be expected. The LMS method has a huge resistance which can be seen from the following corollary.

- If $p > 1$ and there exists some θ such that at least $n - [\frac{n}{2}] + p - 1$ of the observations satisfy $y_i = x_i\theta$ exactly and are in general position, then the LMS solution equals θ whatever the other observations are.

2. Influence Function and Convergence:

Its influence function is not well defined, but its finite-sample breakdown behavior is extremely good. Its convergence follows $n^{-\frac{1}{3}}$. Suppose that the observations y_1, \dots, y_n are i.i.d according to $F(y - \theta)$, where F is a symmetric and strongly unimodal distribution with density f . Then the distribution of the LMS estimator T_n converges weakly as follows:

$$L(n^{1/3}(T_n - \theta)) \rightarrow L(A\tau/f(F^{-1}(.75)))$$

Here $A = (\frac{1}{2}\Lambda^2(F^{-1}(.75)))^{-1/3}$, where $\Lambda = -f'/f$ corresponds to the maximum likelihood scores, and τ is the random time s for which $s^2 + Z(s)$ attains its minimum, where $Z(s)$ is a standard Brownian motion.

3. Interpretation in Lower Dimensions:

In the case of simple regression, there are only a single independent variable and a single dependent variable to be fitted to the model $y_i = ax_i + b + e_i$. One therefore has to find the slope and the intercept of a line determined by n points in the plane. The LMS solution corresponds to finding the narrowest strip covering half of the observations. (To be exact the thickness of the strip is measured in the vertical direction, and we want at least $h = [\frac{n}{2}] + 1$ points on it.)

7 R Code for Simulation and Comparison on Conic Data

```
1 #Data Simulation Part#
2
3 #Program for contaminated Gaussian Error Generation#
4
5 sample_contaminated_dist<-function(y,epsilon,sigma)
6 {
7   e=runif(1,min=0,max=1)
8   if(e<epsilon)
9   {
10     return(y)
11   }else
12   {
13     generated_val=rnorm(1,mean=0,sd=sigma)
14     return(generated_val)
15   }
16 }
17
18 #Assign parameters for the conic with F=1 fixed#
19 F=1
20 A=2
21 B=1
22 C=1
23 D=-1
24 E=1
25
26 #Finding points on the conic#
27 data_matrix=NULL
28 #Fix number of data points#
29 n=40
30 for(i in 1:n)
31 {
32   whether_valid_point=0
33   while (whether_valid_point==0)
34   {
35     x=runif(1,min=-100,max=100)
36     a=C
37     b=(2*B*x)+(2*E)
38     const=(A*x^2)+(2*D*x)+1
39     discriminant=(b^2)-(4*a*const)
40     if(discriminant>=0)
41     {
42       whether_valid_point=1
43     }
44   }
45   e=runif(1,min=0,max=1)
46   if(e<0.5)
47   {
48     y=(-b+sqrt(discriminant))/(2*a)
49   }else
50   {
51     y=(-b-sqrt(discriminant))/(2*a)
52   }
53
54   data_matrix=rbind(data_matrix,c(x,y))
55 }
```

```

55 }
56 u=data_matrix[,1]
57 v=data_matrix[,2]
58 plot(u,v,pch=16,col='blue')
59 #Fix variance and outlier for error
60 s=0.1
61 y=1
62 epsilon=0.1
63 #adding error to the y part
64 for (i in 1:n)
65 {
66     e=sample_contaminated_dist(y,epsilon,s)
67     data_matrix[i,2]=data_matrix[i,2]+e
68 }
69 N=10000
70 estimates=NULL
71 for(i in 1:N)
72 {
73     #Choosing 5 points for solving A,B,C,D,E
74     subsample=sample(1:n,5,replace = FALSE)
75     sample_points=data_matrix[subsample,]
76     #Framing as a linear system with unknowns A,B,C,D,E
77     coef_matrix=sample_points
78     coef_matrix=cbind(coef_matrix,(sample_points[,1])^2)
79     coef_matrix=cbind(coef_matrix,(sample_points[,2])^2)
80     coef_matrix=cbind(coef_matrix,(sample_points[,1]*sample_points[,2]))
81     solution=solve(coef_matrix,rep(-1,times=5))
82     solution[1]=solution[1]/2
83     solution[2]=solution[2]/2
84     solution[5]=solution[5]/2
85     estimates=rbind(estimates,solution)
86 }
87 #Listing median of squared errors for each
88 median_of_squared_residuals=NULL
89 for(i in 1:N)
90 {
91     residual_squares=NULL
92     for(j in 1:n)
93     {
94         residual=(2*estimates[i,1]*data_matrix[j,1])+(2*estimates[i,2]*data_matrix[j,2])+(
95             estimates[i,3]*(data_matrix[j,1]^2))+(estimates[i,4]*(data_matrix[j,2]^2))+(2*
96             estimates[i,5]*data_matrix[j,1]*data_matrix[j,2])+1
97         residual_squares=c(residual_squares,residual^2)
98     }
99     median_of_squared_residuals=c(median_of_squared_residuals,median(residual_squares))
100 }
101 LMS_estimates=estimates[which.min(median_of_squared_residuals),]
102
103 res=NULL
104 for(j in 1:n)
105 {
106     r=(2*D*data_matrix[j,1])+(2*E*data_matrix[j,2])+(A*(data_matrix[j,1]^2))+(C*(data_
107         matrix[j,2]^2))+(2*B*data_matrix[j,1]*data_matrix[j,2])+1
108     res=c(res,r^2)
109 }
110 res
111 median(res^2)

```

8 Python LMS Code for the 1D,2D Regression Case

```
1 #Main.py#
2 np.random.seed(0)
3 n = 200
4 xs = np.arange(n)
5 ys = 2*xs + 3 + np.random.normal(0, 30, n)
6 def f(theta):
7     return np.median(np.abs(theta[1]*xs + theta[0] - ys))
8 initial_theta = [10, 5]
9 res = minimize(f, initial_theta)
10 plt.scatter(xs, ys, s=1)
11 plt.plot(res.x[1]*xs + res.x[0])
12 def least_median_abs_1d(x: np.ndarray):
13     X = np.sort(x) # Precomputing
14     h = len(X)//2
15     diffs = X[h:] - X[:h+1]
16     min_i = np.argmin(diffs)
17     return diffs[min_i]/2 + X[min_i]
18 def best_theta0(theta1):
19     #Using the data points defined above
20     rs = ys - theta1*xs
21     return least_median_abs_1d(rs)
22 def g(theta1):
23     return f([best_theta0(theta1), theta1])
24 thetas = np.linspace(0, 3, 500)
25 plt.plot(thetas, [g(theta1) for theta1 in thetas])
26 thetas = np.linspace(1.5, 2.5, 500)
27 plt.plot(thetas, [g(theta1) for theta1 in thetas])
28 from scipy.optimize import basinhopping
29 res = basinhopping(g, -10)
30 print(res.x)
31 #Final Check
32 def least_median(xs, ys, guess_theta1):
33     def least_median_abs_1d(x: np.ndarray):
34         X = np.sort(x)
35         h = len(X)//2
36         diffs = X[h:] - X[:h+1]
37         min_i = np.argmin(diffs)
38         return diffs[min_i]/2 + X[min_i]
39     def best_median(theta1):
40         rs = ys - theta1*xs
41         theta0 = least_median_abs_1d(rs)
42         return np.median(np.abs(rs - theta0))
43     res = basinhopping(best_median, guess_theta1)
44     theta1 = res.x[0]
45     theta0 = least_median_abs_1d(ys - theta1*xs)
46     return np.array([theta0, theta1]), res.fun
47 theta, med = least_median(xs, ys, 10)
48 active = ((ys < theta[1]*xs + theta[0] + med) & (ys > theta[1]*xs + theta[0] - med))
49 not_active = np.logical_not(active)
50 plt.plot(xs[not_active], ys[not_active], 'g.')
51 plt.plot(xs[active], ys[active], 'r.')
52 plt.plot(xs, theta[1]*xs + theta[0], 'b')
53 plt.plot(xs, theta[1]*xs + theta[0] + med, 'b--')
54 plt.plot(xs, theta[1]*xs + theta[0] - med, 'b--')
```