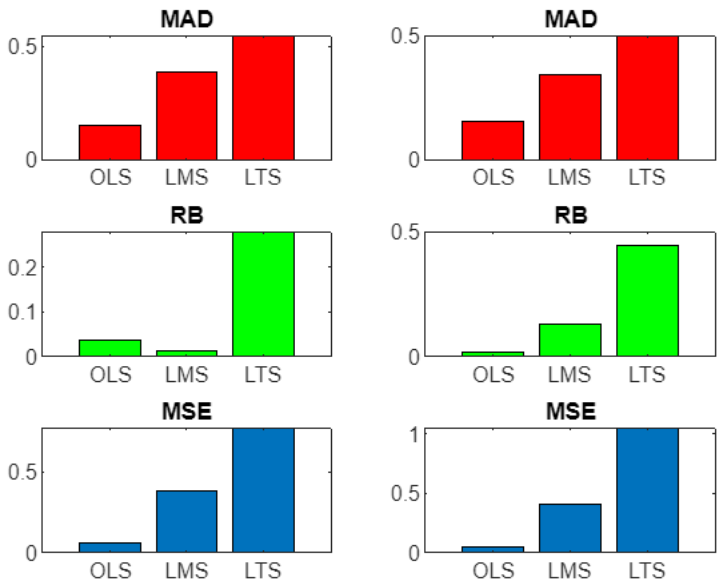
Ordinary Least Squares, Least Median of Squares and Least Trimmed Squares

```
N=20;
R=200;
thetaOLS1=zeros(R,2);
thetaLMS1=zeros(R,2);
thetaLTS1=zeros(R,2);
for i=1:R
    thetaOLS1(i,:)=OLS(N); %the functions (along with DGP) are defined in the
functions below
    thetaLMS1(i,:)=LMS(N);
    thetaLTS1(i,:)=LTS(N);
end
Evaluation_OLS1=zeros(5,2); %bias, variance, mean squared error and robust bias are
stored these matrices
Evaluation_LMS1=zeros(5,2); % for N=20 R=200
Evaluation_LTS1=zeros(5,2);
Evaluation_OLS2=zeros(5,2); % for N=100 R=200
Evaluation_LMS2=zeros(5,2);
Evaluation_LTS2=zeros(5,2);
```

```
%OLS1
%Evaluating the models for N=20 R=200
Evaluation_OLS1(1,:)=(bias(thetaOLS1,[3;5]))';
Evaluation_OLS1(2,:)=variance(thetaOLS1)';
Evaluation_OLS1(3,:)=MAD(thetaOLS1,[3;5])';%Mean absolute deviation
Evaluation_OLS1(4,:)=R_bias(thetaOLS1,[3;5])';%Robust Bias
Evaluation_OLS1(5,:)=(bias(thetaOLS1,[3;5]).^2+variance(thetaOLS1))';%Mean squared
error
%LMS1
Evaluation_LMS1(1,:)=bias(thetaLMS1,[3;5])';
Evaluation_LMS1(2,:)=variance(thetaLMS1)';
Evaluation_LMS1(3,:)=MAD(thetaLMS1,[3;5])';%Mean absolute deviation
Evaluation_LMS1(4,:)=R_bias(thetaLMS1,[3;5])';%Robust Bias
Evaluation_LMS1(5,:)=(bias(thetaLMS1,[3;5]).^2+variance(thetaLMS1))';%Mean squared
error
%LTS1
Evaluation_LTS1(1,:)=bias(thetaLTS1,[3;5])';
Evaluation_LTS1(2,:)=variance(thetaLTS1)';
Evaluation_LTS1(3,:)=MAD(thetaLTS1,[3;5])';%Mean absolute deviation
Evaluation_LTS1(4,:)=R_bias(thetaLTS1,[3;5])';%Robust Bias
Evaluation_LTS1(5,:)=(bias(thetaLTS1,[3;5]).^2+variance(thetaLTS1))';%Mean squared
error
```

```
% plotting the errors
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS1(3,1),Evaluation_LMS1(3,1),Evaluation_LTS1(3,1)]);
subplot(3,2,1)
```

```
bar(x,y1,'r')
title("Median Absolute Deviation")
subplot(3,2,2)
y2=abs([Evaluation_OLS1(3,2),Evaluation_LMS1(3,2),Evaluation_LTS1(3,2)]);
bar(x,y2,'r')
title("Median Absolute Deviation")
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS1(4,1),Evaluation_LMS1(4,1),Evaluation_LTS1(4,1)]);
subplot(3,2,3)
bar(x,y1,'g')
title("Robust Bias")
subplot(3,2,4)
y2=abs([Evaluation_OLS1(4,2),Evaluation_LMS1(4,2),Evaluation_LTS1(4,2)]);
bar(x,y2,'g')
title("Robust Bias")
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS1(5,1),Evaluation_LMS1(5,1),Evaluation_LTS1(5,1)]);
subplot(3,2,5)
bar(x,y1)
title("Mean Square Error")
subplot(3,2,6)
y2=abs([Evaluation_OLS1(5,2),Evaluation_LMS1(5,2),Evaluation_LTS1(5,2)]);
bar(x,y2)
title("Mean Square Error")
```



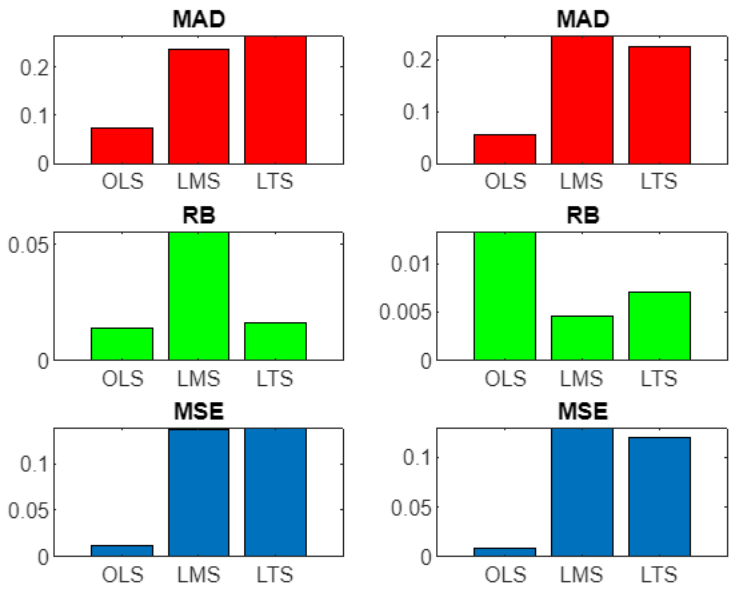
The generated data has white noise alone, hence the possibility of having outliers in the dataset is very less. That is why we can observe the better performance of OLS over LMS,and LTS. Adding outliers (which contribute significantly to the errors) to the dataset will demonstrate the robustness of the LMS and LTS approaches.

```
N=100;
R=200;
thetaOLS2=zeros(R,2);
thetaLMS2=zeros(R,2);
thetaLTS2=zeros(R,2);
for i=1:R
    thetaOLS2(i,:)=OLS(N);
    thetaLMS2(i,:)=LMS(N);
    thetaLTS2(i,:)=LTS(N);
end
```

```
%OLS2
%Evaluation metrics defined as per definition in the function below
%OLS1
Evaluation_OLS2(1,:)=(bias(thetaOLS2,[3;5]))';
Evaluation_OLS2(2,:)=variance(thetaOLS2)';
Evaluation_OLS2(3,:)=MAD(thetaOLS2,[3;5])';%Mean absolute deviation
Evaluation_OLS2(4,:)=R_bias(thetaOLS2,[3;5])';%Robust Bias
Evaluation_OLS2(5,:)=(bias(thetaOLS2,[3;5]).^2+variance(thetaOLS2))';%Mean squared
error
%LMS1
Evaluation_LMS2(1,:)=bias(thetaLMS2,[3;5])';
Evaluation_LMS2(2,:)=variance(thetaLMS2)';
Evaluation_LMS2(3,:)=MAD(thetaLMS2,[3;5])';%Mean absolute deviation
Evaluation_LMS2(4,:)=R_bias(thetaLMS2,[3;5])';%Robust Bias
Evaluation_LMS2(5,:)=(bias(thetaLMS2,[3;5]).^2+variance(thetaLMS2))';%Mean squared
error
%LTS1
Evaluation_LTS2(1,:)=bias(thetaLTS2,[3;5])';
Evaluation_LTS2(2,:)=variance(thetaLTS2)';
Evaluation_LTS2(3,:)=MAD(thetaLTS2,[3;5])';%Mean absolute deviation
Evaluation_LTS2(4,:)=R_bias(thetaLTS2,[3;5])';%Robust Bias
Evaluation_LTS2(5,:)=(bias(thetaLTS2,[3;5]).^2+variance(thetaLTS2))';%Mean squared
error
```

```
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS2(3,1),Evaluation_LMS2(3,1),Evaluation_LTS2(3,1)]);
subplot(3,2,1)
bar(x,y1,'r')
title("Median Absolute Deviation")
subplot(3,2,2)
y2=abs([Evaluation_OLS2(3,2),Evaluation_LMS2(3,2),Evaluation_LTS2(3,2)]);
bar(x,y2,'r')
title("Median Absolute Deviation")
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS2(4,1),Evaluation_LMS2(4,1),Evaluation_LTS2(4,1)]);
subplot(3,2,3)
```

```
bar(x,y1,'g')
title("Robust Bias")
subplot(3,2,4)
y2=abs([Evaluation_OLS2(4,2),Evaluation_LMS2(4,2),Evaluation_LTS2(4,2)]);
bar(x,y2,'g')
title("Robust Bias")
x=["OLS","LMS","LTS"];
y1=abs([Evaluation_OLS2(5,1),Evaluation_LMS2(5,1),Evaluation_LTS2(5,1)]);
subplot(3,2,5)
bar(x,y1)
title("Mean Square Error")
subplot(3,2,6)
y2=abs([Evaluation_OLS2(5,2),Evaluation_LMS2(5,2),Evaluation_LTS2(5,2)]);
bar(x,y2)
title("Mean Square Error")
```




```
function B=bias(thetaModel,theta_truth) %evaluation functions according to the
definitions provided
    %B is a vector with biases of each parameter estimate
    p=size(theta_truth,1);
    B=zeros(p,1);
    for i=1:p
        B(i)=mean(thetaModel(:,i))-theta_truth(i,1);
    end
end

function RB =R_bias(thetaModel,theta_truth)
    %B is a vector with biases of each parameter estimate
    p=size(theta_truth,1);
```

```
    RB=zeros(p,1);
    for i=1:p
        RB(i)=median(thetaModel(:,i))-theta_truth(i,1);
    end
end

function Mdev =MAD(thetaModel,theta_truth)
    %B is a vector with biases of each parameter estimate
    p=size(theta_truth,1);
    Mdev=zeros(p,1);
    for i=1:p
        Mdev(i)=median(abs(thetaModel(:,i)-theta_truth(i,1)));
    end
end


function V=variance(thetaModel)
    %B is a vector with biases of each parameter estimate
    p=size(thetaModel,2);
    V=zeros(p,1);
    for i=1:p
        V(i)=var(thetaModel(:,i));
    end
end
%Ordinary Least Squares
function theta=OLS(N)
    x1=randn(N,1);
    x2=randn(N,1);
    e=randn(N,1);
    X=[x1 x2];
    thetavec_0=[3;5];
    Y=X*thetavec_0+e;

    %gradient descent

    theta=[0;0];
    % Initial guess for theta (adjust as needed)
    theta = [0; 0];

    % Learning rate (experiment with different values)
    alpha = 0.1;

    % Stopping criterion (tolerance for convergence)
    tolerance = 0.001;

    % Maximum number of iterations (to prevent infinite loops in case of poor
convergence)
    max_iterations = 1000;

    % Track cost history (optional)
```

```
    cost_history = zeros(max_iterations, 1);

    % Gradient descent loop
    for i = 2:max_iterations
      % Calculate prediction
      prediction = X * theta;

      % Calculate cost (mean squared error)
      cost_history(i) =sum((prediction - Y) .^ 2);

      % Calculate gradient (vectorized for efficiency)
      gradient = X' * (prediction - Y)./N;

      % Update theta using gradient descent
      theta = theta - alpha * gradient;

      % Check stopping criterion (absolute difference in cost)
      if abs(cost_history(i) - cost_history(i-1)) < tolerance
        break;
      end
    end
end
%Least Median of Squares
```

Here, the LMS function is assumed to be convex. This assumption is needed to make sure we reach a minima upon using the gradient descent algorithm

```
function theta=LMS(N)
    x1=randn(N,1);
    x2=randn(N,1);
    e=randn(N,1);
    X=[x1 x2];
    thetavec_0=[3;5];
    Y=X*thetavec_0+e;
    theta=zeros(2,1);
    cost_history=zeros(N,1);
    max_iterations=1000;
    tolerance=0.001;
    alpha=0.1;
    % Gradient descent loop
    for i = 2:max_iterations

      % Calculate prediction
      prediction = X * theta;

      % Calculate cost (mean squared error)
      cost_history(:,i) =(prediction - Y) .^ 2;
      med(i)=median(cost_history(:,i));
      o=[med(i),theta'];
      [min_diff,idx] = min(abs(cost_history(:,i)-median(cost_history(:,i))));
```

```
      % Calculate gradient
      psi=X(idx,:);
      gradient = 2 * psi' * (Y(idx)-psi*theta);

      % Update theta using gradient descent
      theta = theta + alpha * gradient;

      % Check stopping criterion (absolute difference in cost)
      if sum(abs(cost_history(:,i) - cost_history(:,i-1))) < tolerance
          x=0
        break;
      end
    end
end

%Least Trimmed Squares
function theta=LTS(N)
    x1=randn(N,1);
    x2=randn(N,1);
    e=randn(N,1);
    X=[x1 x2];
    thetavec_0=[3;5];
    Y=X*thetavec_0+e;
    theta=zeros(2,1);
    cost_history=zeros(N,1);
    max_iterations=1000;
    tolerance=0.001;
    alpha=0.1;

    % Gradient descent loop
    for i = 2:max_iterations
      X_new=zeros(N,1);
      % Calculate prediction
      prediction = X * theta;

      % Calculate cost (mean squared error)
      cost_history(:,i) =(prediction - Y) .^ 2;

      [min_diff,idx] = sort(cost_history(:,i));
      sz=size(X);
      columns=sz(1,2);
      for j=1:columns
          temp_col=X(:,j);
          X_new=[X_new,temp_col(idx)];
      end
      X_new=X_new(:,2:columns+1);
      %X_new=[x1(idx), x2(idx)];
      X_new=X_new((N/2)+1,:);
      Y_new=Y(idx);
```

```
      % Calculate gradient (vectorized for efficiency)

      gradient = 2 * X_new' * (Y_new((N/2)+1,:)-X_new*theta);

      % Update theta using gradient descent
      theta = theta + alpha * gradient;

      % Check stopping criterion (absolute difference in cost)
      if sum(abs(cost_history(:,i) - cost_history(:,i-1))) < tolerance
          x=0
        break;
      end
    end
end
```

N=20 R=200

Ordinary Least Squares

	Theta 1	Theta 2
Bias	-0.0242	0.0075
Variance	0.0603	0.0629
Median Absolute Deviation	0.1581	0.1609
Robust Bias	-0.0084	0.0126
Mean Square Error	0.0609	0.0630

Least Median of Square

	Theta 1	Theta 2
Bias	-0.1318	0.2804
Variance	0.4781	0.5743
Median Absolute Deviation	0.4057	0.4193
Robust Bias	-0.0505	-0.2174
Mean Square Error	0.4955	0.6529

Least Trimmed Squares

	Theta 1	Theta 2
Bias	-0.4010	-0.5834
Variance	0.5680	0.6122
Median Absolute Deviation	0.4812	0.5376
Robust Bias	-0.3140	-0.4482
Mean Square Error	0.7288	0.9526

N=100 R=200

Ordinary Least Squares

	Theta 1	Theta 2
Bias	-0.0054	-0.0001
Variance	0.0103	0.0101
Median Absolute Deviation	0.0674	0.0597
Robust Bias	-0.0005	0.0025
Mean Square Error	0.0104	0.0101

Least Median of Squares

	Theta 1	Theta 2
Bias	-0.0142	-0.0186
Variance	0.1264	0.1371
Median Absolute Deviation	0.2034	0.2831
Robust Bias	0.0080	-0.0159
Mean Square Error	0.1266	0.1375

Least Trimmed Squares

	Theta 1	Theta 2
Bias	-0.0008	-0.0184
Variance	0.1269	0.1340
Median Absolute Deviation	0.2504	0.2466
Robust Bias	-0.0139	-0.0263
Mean Square Error	0.1269	0.1344

Using OLS,LMS and LTS on medical_insurance.csv

```
%Q1 part iii
new_sex=zeros(2772,1);
for i=1:2772
    if sex(i,1)=="female"
        new_sex(i,1)=1;
    end
end
new_region=zeros(2772,1);
for i=1:2772
    if region(i,1)=="northeast"
        new_region(i,1)=1;
    elseif region(i,1)=="northwest"
        new_region(i,1)=2;
    elseif region(i,1)=="southeast"
        new_region(i,1)=3;
    else
        new_region(i,1)=4;
    end
end
new_smoker=zeros(2772,1);
for i=1:2772
    if smoker(i,1)=="yes"
        new_smoker(i,1)=1;
    end
end

age=age/mean(age);
new_sex=new_sex/mean(new_sex);
new_smoker=new_smoker/mean(new_smoker);
children=children/mean(children);
new_region=new_region/mean(new_region);
bmi=bmi/mean(bmi);
```

```
X=[age new_sex bmi children new_smoker new_region];
X=X(1:2218,:);
Y=charges/mean(charges);
Y=Y(1:2218,:);
fitlm(X,Y,Intercept=false)
```

```
ans =
Linear regression model:
    y ~ x1 + x2 + x3 + x4 + x5 + x6

Estimated Coefficients:
          Estimate       SE         tStat       pValue
          ________    _________    _______    ___________

    x1     0.61401      0.02636     23.293    1.6277e-107
    x2    -0.02049     0.009874    -2.0751       0.038093
    x3     0.20677     0.034426     6.0061       2.215e-09
```

```
    x4    0.022325    0.0092303     2.4187       0.015657
    x5     0.36057    0.0051396     70.155              0
    x6    -0.16126     0.022683    -7.1094      1.566e-12


Number of observations: 2218, Error degrees of freedom: 2212
Root Mean Squared Error: 0.477
```

```
N=2218;
X=[age new_sex bmi children new_smoker new_region];
X=X(1:2218,:);
thetavec_0=zeros(6,1);
Y=charges/mean(charges);
Y=Y(1:2218,:);

%gradient descent
theta=thetavec_0;
% Learning rate (experiment with different values)
alpha = 0.1;

% Stopping criterion (tolerance for convergence)
tolerance = 0.001;

% Maximum number of iterations (to prevent infinite loops in case of poor
convergence)
max_iterations = 1000;

% Track cost history (optional)
cost_history = zeros(max_iterations, 1);

% Gradient descent loop
for i = 2:max_iterations
  % Calculate prediction
  prediction = X * theta;

  % Calculate cost (mean squared error)
  cost_history(i) =sum((prediction - Y) .^ 2);

  % Calculate gradient (vectorized for efficiency)
  gradient = X' * (prediction - Y)./N;

  % Update theta using gradient descent
  theta = theta - alpha * gradient;

  % Check stopping criterion (absolute difference in cost)
  if abs(cost_history(i) - cost_history(i-1)) < tolerance
    break;
  end
end
thetaOLS=theta
```

```
thetaOLS = 6×1
    0.6023
   -0.0205
    0.2179
    0.0223
    0.3605
   -0.1603
```

```
N=2218;
X=[age new_sex bmi children new_smoker new_region];
X=X(1:2218,:);
thetavec_0=zeros(6,1);
Y=charges/mean(charges);
Y=Y(1:2218,:);
cost_history=zeros(N,1);
max_iterations=1000;
tolerance=0.001;
alpha=0.1;
theta=thetavec_0;
% Gradient descent loop
for i = 2:max_iterations

  % Calculate prediction
  prediction = X * theta;

  % Calculate cost (mean squared error)
  cost_history(:,i) =(prediction - Y) .^ 2;
  med(i)=median(cost_history(:,i));
  o=[med(i),theta'];
  [min_diff,idx] = min(abs(cost_history(:,i)-median(cost_history(:,i))));

  % Calculate gradient
  psi=X(idx,:);
  gradient = 2 * psi' * (Y(idx)-psi*theta);

  % Update theta using gradient descent
  theta = theta + alpha * gradient;

  % Check stopping criterion (absolute difference in cost)
  if sum(abs(cost_history(:,i) - cost_history(:,i-1))) < tolerance
      x=0
    break;
  end
end
thetaLMS=theta
```

```
thetaLMS = 6×1
    0.7800
    0.0420
   -0.1592
    0.0352
```

```
         0
   -0.0098
```

```
N=2218;
X=[age new_sex bmi children new_smoker new_region];
X=X(1:2218,:);
thetavec_0=zeros(6,1);
Y=charges/mean(charges);
Y=Y(1:2218,:);
cost_history=zeros(N,1);
max_iterations=1000;
tolerance=0.001;
alpha=0.1;
theta=thetavec_0;

% Gradient descent loop
for i = 2:max_iterations
  X_new=zeros(N,1);
  % Calculate prediction
  prediction = X * theta;

  % Calculate cost (mean squared error)
  cost_history(:,i) =(prediction - Y) .^ 2;

  [min_diff,idx] = sort(cost_history(:,i));
  sz=size(X);
  columns=sz(1,2);
  for j=1:columns
      temp_col=X(:,j);
      X_new=[X_new,temp_col(idx)];
  end
  X_new=X_new(:,2:columns+1);
  %X_new=[x1(idx), x2(idx)];
  X_new=X_new((N/2)+1,:);
  Y_new=Y(idx);

  % Calculate gradient (vectorized for efficiency)

  gradient = 2 * X_new' * (Y_new((N/2)+1,:)-X_new*theta);

  % Update theta using gradient descent
  theta = theta + alpha * gradient;

  % Check stopping criterion (absolute difference in cost)
  if sum(abs(cost_history(:,i) - cost_history(:,i-1))) < tolerance
      x=0
    break;
  end
end
```

```
thetaLTS=theta
```

```
thetaLTS = 6×1
    0.7404
    0.0348
   -0.1473
    0.1065
         0
   -0.0981
```

```
X=[age new_sex bmi children new_smoker new_region];
X_test=X(2219:2772,:);
Y=charges/mean(charges);
Y_test=Y(2219:2772,:);


MSE_OLS=sum((X_test*thetaOLS-Y_test).^2)./(554)
```

```
MSE_OLS = 0.2355
```

```
MSE_LMS=sum((X_test*thetaLMS-Y_test).^2)./(554)
```

```
MSE_LMS = 0.9220
```

```
MSE_LTS=sum((X_test*thetaLTS-Y_test).^2)./(554)
```

```
MSE_LTS = 0.9567
```

Since, the Mean square error for the given data is least for the Ordinary Least Squares approach it is best to use that. The performance of ordinary least squares can be attributed to the presence of very less contribution from outliers(if any).