Robust Regression

Dr. Kiah Wah Ong

ILLINOIS TECH

Introduction: Robustness

Suppose we have the following data $x_1, \cdots, x_n$ and consider the average

$$\begin{aligned} \bar{x} &= \frac{1}{n} \sum_{i=1}^{n} x_i \\ &= \frac{1}{n} \left[\sum_{i=1}^{n-1} x_i + x_n \right] \\ &= \frac{n-1}{n} \bar{x}_{n-1} + \frac{1}{n} x_n \end{aligned}$$

Notice that if the value of x_n is large enough, then the average, $\bar{x}$ can be made as large as possible regardless of the other $n-1$ values of x_i.

Because of this, we say that the mean (average) is not a robust measure of central tendency.

Least Squares Method

Recall that in our simple linear regression model

$$y = \beta_0 + \beta_1 x + \epsilon$$

the parameters β_0 and β_1 are unknown and must be estimated using sample data

$$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n).$$

The least squares method is used to estimate β_0 and β_1. To do this, from

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

we write

$$\epsilon_i = y_i - (\beta_0 + \beta_1 x_i)$$

and we want to find $\hat{\beta}_1$ and $\hat{\beta}_2$ that minimize the residual sum of squares.

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

Least Squares Method

Because of squares in

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2$$

outliers are heavily weighted. Bad data points can throw off the regression.

M-Estimation

M-estimation is a generalization of the ordinary least squares. (M for Maximum)

Instead of squaring the residuals, we will define a general function of the residuals, $H(\epsilon_i)$, and then minimize

$$S = \sum H(\epsilon_i).$$

M-Estimation

We want the function H has the following properties:

- H is **non-negative**, i.e. $H(\epsilon_i) \geq 0$
- $H(0) = 0$
- H is **symmetric**, i.e. $H(-\epsilon_i) = H(\epsilon_i)$
- H is **monotonic**, i.e.
  if $|\epsilon_i| > |\epsilon_j|$, then $H(\epsilon_i) > H(\epsilon_j)$
- H has **continuous derivative** with respect to the coefficients.
  This allows us in finding the minimum more effectively.

Notice that $H(\epsilon_i) = \epsilon_i^2$ has all the above properties.

M-Estimation

For the ordinary least squares method, we let $S = \sum_{i=1}^{n} \epsilon_i^2$ and then take the derivative with respect to a parameter and set that equal to zero:

$$\frac{\partial S}{\partial \beta_j} = 0 \quad \text{gives} \quad \sum_{i=1}^{n} \epsilon_i x_{ji} = 0$$

For M-estimator, with $S = \sum H(\epsilon_i)$, we have

$$\frac{\partial S}{\partial \beta_j} = 0 \quad \text{gives} \quad \sum_{i=1}^{n} \frac{\partial H}{\partial \epsilon_i} x_{ji} = 0$$

M-Estimation

Let us define the weight as

$$w_i = \frac{1}{\epsilon_i}\frac{\partial H}{\partial \epsilon_i}$$

Hence, the previous expression

$$\sum_{i=1}^{n} \frac{\partial H}{\partial \epsilon_i} x_{ji} = 0$$

can be written as

$$\sum_{i=1}^{n} \frac{\partial H}{\partial \epsilon_i} x_{ji} = 0 = \sum_{i=1}^{n} w_i \epsilon_i x_{ji}$$

that is, we have the set up of a weighted linear regression. This gives us a scheme that we can use to do M-estimation.

M-Estimation

In order to perform the M-estimation from

$$w_i = \frac{1}{\epsilon_i}\frac{\partial H}{\partial \epsilon_i}, \quad \sum_{i=1}^{n} w_i \epsilon_i x_{ji} = 0$$

we will:

- guess the weight, w_i
- calculate the residuals, ϵ_i
- use the previous residuals to calculate new weight
- we repeat these process until convergence

The process described above are called iteratively reweighted least squares.

M-Estimation

What we want to do now is to find a good function H that has the following properties that can give rise to good result.

- H is **non-negative**, i.e. $H(\epsilon_i) \geq 0$
- $H(0) = 0$
- H is **symmetric**, i.e. $H(-\epsilon_i) = H(\epsilon_i)$
- H is **monotonic**, i.e.
  if $|\epsilon_i| > |\epsilon_j|$, then $H(\epsilon_i) > H(\epsilon_j)$
- H has **continuous derivative** with respect to the coefficients. This allows us in finding the minimum more effectively.

Huber M-Estimator

One of the commonly used M-estimator is the Huber M-estimator given below:

$$H(\epsilon) = \begin{cases} \epsilon^2/2, & \text{for } |\epsilon| \le k \\ k|\epsilon| - k^2/2 & \text{for } |\epsilon| > k \end{cases}$$

Notice that when with in a certain value of k, the function H behave like the ordinary least square estimator, however, when we went beyond the value of k, then we will switch to using the absolute value of the residue (hence avoiding amplifying the effect of outliers.)

The value k for the Huber estimator is called a tuning constant. The tuning constant is picked to give a high efficiency when the data is normal.

Huber choose the value of $k = 1.345\sigma$ that produce a 95% efficiency when the errors are normal, and still offer protection against outliers.

Bisquare M-Estimator

Another commonly used M-estimator is called Bisquare M-Estimator.

$$H(\epsilon) = \begin{cases} k^2/6 \; (1 - [1 - (\epsilon/k)^2]^3) & \text{for } |\epsilon| \leq k \\ k^2/6 & \text{for } |\epsilon| > k \end{cases}$$

The bisquare M-estimator in a sense is more robust than the Huber M-estimator. This is because when $|\epsilon| > k = 4.685\sigma$, the weighting become constant.

M-estimator Weights

Another way of looking at these two M-estimators is by looking at the weight, w_i in:

$$w_i = \frac{1}{\epsilon_i}\frac{\partial H}{\partial \epsilon_i}, \quad \sum_{i=1}^{n} w_i \epsilon_i x_{ji} = 0$$

For Huber M-estimator, we have

$$w(\epsilon) = \begin{cases} 1 & \text{for} \quad |\epsilon| \leq k \\ k/|\epsilon| & \text{for} \quad |\epsilon| > k \end{cases}$$

While, for bisquare M-estimator, we have

$$w(\epsilon) = \begin{cases} \left[1 - \left(\frac{\epsilon}{k}\right)^2\right]^2 & \text{for} \quad |\epsilon| \leq k \\ 0 & \text{for} \quad |\epsilon| > k \end{cases}$$

M-estimator Weights

For the Huber M-estimator,

$$w(\epsilon) = \begin{cases} 1 & \text{for } |\epsilon| \leq k \\ k/|\epsilon| & \text{for } |\epsilon| > k \end{cases}$$

we see that the weight started at one and then weighting less after some point.

For bisquare M-estimator,

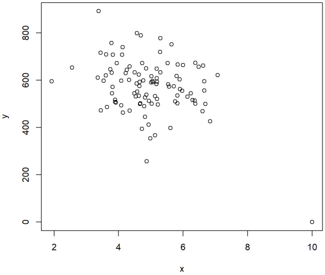
$$w(\epsilon) = \begin{cases} \left[1 - \left(\frac{\epsilon}{k}\right)^2\right]^2 & \text{for } |\epsilon| \leq k \\ 0 & \text{for } |\epsilon| > k \end{cases}$$

we see that the weight immediately started to drop off from one (when $\epsilon = 0$) to zero after $|\epsilon| > k$. Hence extreme outliers will not be weighted at those level.

Robust Regression in *R*

We now look at how to perform robust regression in *R*: Let us import the simulated data from data set $\mathrm{RRobust1.csv}$ and plot the data.

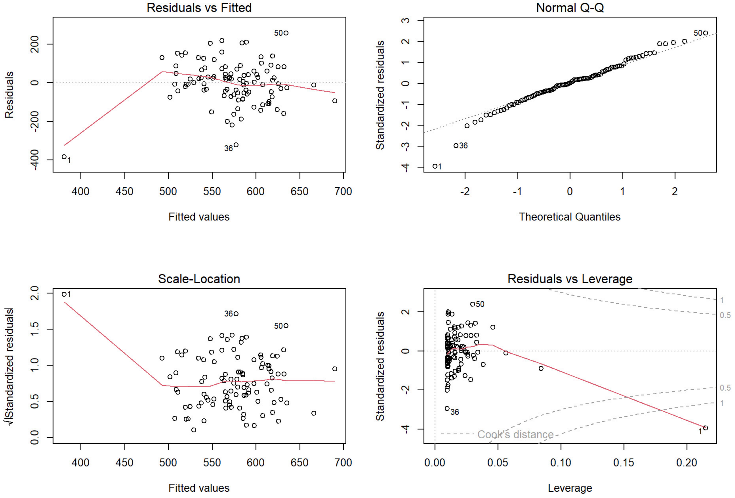
```
Robustregression<-read.csv("RRobust1.CSV", header=TRUE, sep=",")
x<-Robustregression$x
y<-Robustregression$y
plot(x,y)
```



We can see clearly that the data point $(10, -1)$ is an outlier.

Robust Regression in *R*

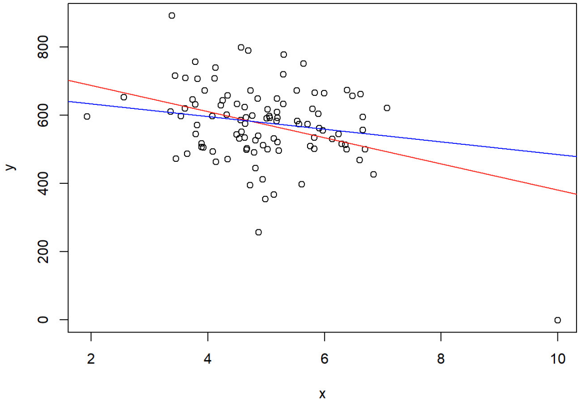
The standard diagnostic plots also make a clear point that $(10, -1)$ is an outlier.



Robust Regression in *R*

We now make two regression models. In model1, an OLS regression is performed, and the regression line is plot in red. In model2, an OLS regression is performed with the outlier removed. This regression line is shown in blue.

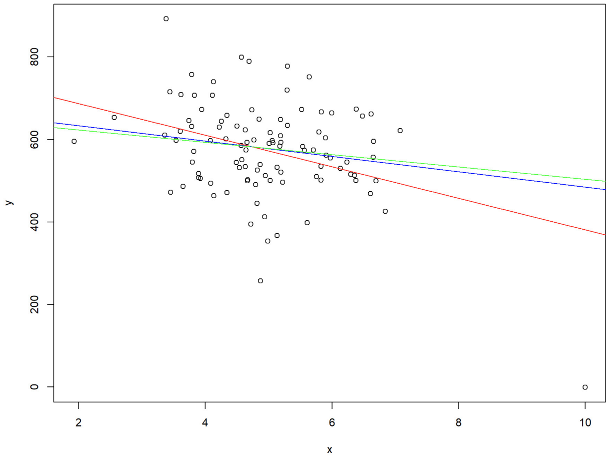
```r
#With outlier
model1=lm(y~x)
abline(model1, col="red")

#Removing outlier
df1<-Robustregression
df2<-df1[-1,]
model2=lm(y~x,data=df2)
abline(model2, col="blue")
```



Robust Regression in R

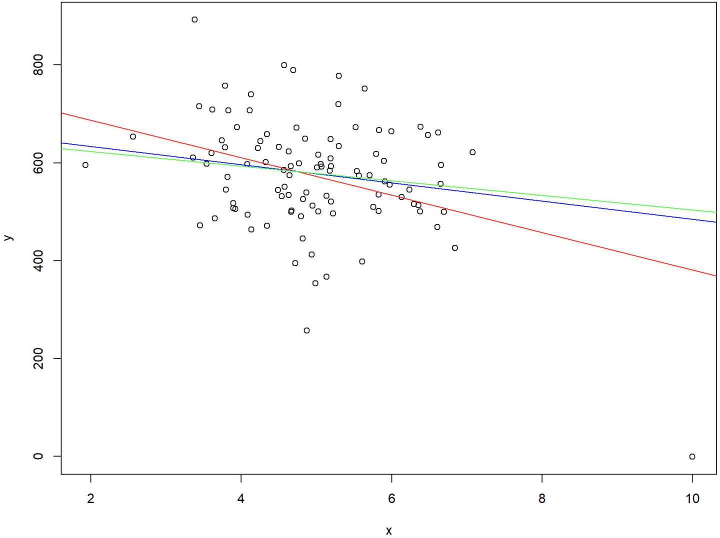
Now, model3 is set up as a robust regression model. The outlier is now weighted down by using the bisquare method.

```
library(MASS)
#Not removing outlier but use robust regression
model3=rlm(y~x, data=df1, psi=psi.bisquare)
abline(model3, col="green")
```



Robust Regression in *R*

Using robust regression technique, we see that there is no need to pretreat the outliers.




```
> coefficients(model1)
(Intercept)           x
  763.77361   -38.31104
> coefficients(model2)
(Intercept)           x
  670.49249   -18.54451
> coefficients(model3)
(Intercept)           x
  653.14169   -14.96592
```