Introductory Computer Programming

Assignment 1

December 13, 2021

Name : Debarshi Chakraborty
Stream : M.Stat (NB)
Roll No :MD2105
Semester : 1
Instructor : Soumya Chakraborty

Question 1

By simulation, present a comparative study of sample mean and sample median as estimates of Normal mean.

Solution:

We first specify our simulation setup.We fix our population to be $N(\mu = 5, \sigma^2 = 16)$.
Number of replications (r)=100.Sample size for each replication (n)=1000.

```
n=1000
r=100
mu=5
sigma=4
```

We are going to use Bias and Mean Squared Error(MSE) as our measure of accuracy and precision respectively.
If $\hat{\mu}$ is an estimator of μ then
$Bias(\hat{\mu}) = E(\hat{\mu} - \mu)$
$MSE(\hat{\mu}) = E(\hat{\mu} - \mu)^2$
We have two estimates of μ namely Sample Mean = $\bar{x}$ and Sample Median = $\tilde{x}$
For each estimate,we execute the following steps to get the emperical Bias and Emperical MSE:-

- Draw a sample of size n.
- Compute the estimate (i.e sample mean/median).
- Replicate the above steps r times.
- We get "r" estimates $\hat{\mu_1}, \hat{\mu_2}, ...\hat{\mu_r}$.
- Our final estimate is $\hat{\mu} = \frac{\sum_{i=1}^{r} \hat{\mu_i}}{r}$.
- Compute the Bias as $\hat{\mu} - \mu$.
- Similarly we compute $(\hat{\mu_1} - \mu)^2, (\hat{\mu_2} - \mu)^2, ...(\hat{\mu_r} - \mu)^2$
- Get the final MSE as $\frac{\sum_{i=1}^{r}(\hat{\mu_i} - \mu)^2}{r}$

```
mu_hat_1=array(0)
mu_hat_2=array(0)
for (i in 1:r)
{
  x<-rnorm(n,mean=mu,sd=sigma)
  mu_hat_1[i]=mean(x)
  mu_hat_2[i]=median(x)
}
mu_hat_1

  [1] 4.884752 5.072714 4.852403 5.161065 4.974853 4.921165 4.943473 4.634220
  [9] 5.030704 4.937323 5.011442 4.934836 5.185067 4.986646 5.047887 5.097221
 [17] 4.904775 5.026096 5.010759 4.955002 5.138168 4.993764 4.924253 5.096551
 [25] 4.689547 5.103361 4.755528 5.088291 5.228953 4.956258 4.780584 5.110299
 [33] 5.318055 4.940074 5.009173 4.928208 4.827066 5.116825 5.006454 5.035994
 [41] 5.128177 4.997469 5.158832 4.873176 4.842275 4.783666 5.049671 4.947619
 [49] 4.713192 5.027945 4.967569 5.111101 5.026215 5.201520 4.908490 4.901617
 [57] 5.054886 5.088260 5.030599 4.879383 5.005358 5.217686 4.885574 5.141142
 [65] 5.164946 5.084177 4.927075 4.960892 4.981785 5.214239 4.947130 4.857131
 [73] 4.957284 4.862949 5.127278 5.262322 4.928339 5.024812 4.985626 5.081237
 [81] 4.675007 5.151727 5.115834 5.187205 5.080091 4.892539 5.029137 4.725621
 [89] 4.919487 4.953322 4.909449 4.829725 5.109937 5.017197 4.905780 4.905275
 [97] 4.864462 4.871863 5.171016 5.050560
```

```
mu_hat_2

  [1] 5.049621 4.868553 4.941065 5.278026 4.963777 4.928347 4.864216 4.680066
  [9] 5.146510 5.040193 4.968575 5.067326 5.173007 5.017980 5.110909 5.067610
 [17] 4.886751 4.945214 4.949269 5.025291 5.228414 5.010616 4.840179 5.128273
 [25] 4.665134 4.952877 4.662693 5.033378 5.172646 4.848568 4.554421 5.067525
 [33] 5.426185 4.957989 5.057406 4.909269 4.866201 5.348540 4.982697 4.902661
 [41] 5.128313 4.892621 5.139244 4.838777 4.887230 4.771839 5.098381 4.973285
 [49] 4.501529 4.974481 5.016843 5.187295 4.944542 5.141684 4.901983 4.987791
 [57] 5.185675 4.919201 4.967337 4.691446 4.952025 5.223812 4.692426 5.155345
 [65] 5.179922 5.319942 4.836180 4.871509 5.021184 5.228034 5.015704 4.783669
 [73] 4.892925 4.866201 5.331744 5.341069 4.905266 5.064033 4.911402 5.035829
 [81] 4.565987 4.961559 5.165332 5.021002 4.899322 4.791627 5.133897 4.758256
 [89] 4.843587 4.941922 4.794894 4.757590 5.135526 4.831706 5.012166 4.927620
 [97] 4.663882 4.938902 5.109755 5.021537

(mse1=(mu_hat_1-mu)^2)

  [1] 1.328208e-02 5.287313e-03 2.178477e-02 2.594178e-02 6.323826e-04
  [6] 6.215002e-03 3.195319e-03 1.337952e-01 9.427132e-04 3.928453e-03
 [11] 1.309263e-04 4.246305e-03 3.424968e-02 1.783309e-04 2.293185e-03
 [16] 9.451830e-03 9.067731e-03 6.809950e-04 1.157587e-04 2.024844e-03
 [21] 1.909028e-02 3.889097e-05 5.737581e-03 9.322103e-03 9.638118e-02
 [26] 1.068354e-02 5.976644e-02 7.795389e-03 5.241946e-02 1.913370e-03
 [31] 4.814344e-02 1.216577e-02 1.011589e-01 3.591147e-03 8.414050e-05
 [36] 5.154120e-03 2.990624e-02 1.364815e-02 4.165468e-05 1.295585e-03
 [41] 1.642926e-02 6.407004e-06 2.522746e-02 1.608422e-02 2.487702e-02
 [46] 4.680056e-02 2.467183e-03 2.743812e-03 8.225866e-02 7.809365e-04
 [51] 1.051770e-03 1.234344e-02 6.872458e-04 4.061041e-02 8.374022e-03
 [56] 9.679289e-03 3.012444e-03 7.789824e-03 9.362877e-04 1.454837e-02
 [61] 2.871117e-05 4.738718e-02 1.309340e-02 1.992109e-02 2.720726e-02
 [66] 7.085691e-03 5.318115e-03 1.529434e-03 3.317805e-04 4.589821e-02
 [71] 2.795281e-03 2.041153e-02 1.824642e-03 1.878298e-02 1.619977e-02
 [76] 6.881288e-02 5.135292e-03 6.156494e-04 2.066198e-04 6.599508e-03
 [81] 1.056204e-01 2.302118e-02 1.341760e-02 3.504553e-02 6.414515e-03
 [86] 1.154795e-02 8.489697e-04 7.528399e-02 6.482325e-03 2.178812e-03
 [91] 8.199395e-03 2.899374e-02 1.208625e-02 2.957239e-04 8.877461e-03
 [96] 8.972798e-03 1.837052e-02 1.641918e-02 2.924648e-02 2.556314e-03

(mse2=(mu_hat_2-mu)^2)

  [1] 0.0024622532 0.0172783475 0.0034733582 0.0772986241 0.0013121191
  [6] 0.0051341415 0.0184373309 0.1023580179 0.0214652194 0.0016155103
 [11] 0.0009875142 0.0045327342 0.0299314750 0.0003232900 0.0123007332
 [16] 0.0045711115 0.0128253038 0.0030014548 0.0025735905 0.0006396279
 [21] 0.0521730997 0.0001126891 0.0255427503 0.0164539410 0.1121351651
 [26] 0.0022205526 0.1137756911 0.0011141240 0.0298068102 0.0229315106
 [31] 0.1985410323 0.0045595955 0.1816336330 0.0017649266 0.0032954891
 [36] 0.0082321020 0.0179022748 0.1214798576 0.0002993808 0.0094748576
 [41] 0.0164642433 0.0115302559 0.0193888475 0.0259929967 0.0127169952
 [46] 0.0520574850 0.0096788808 0.0007136862 0.2484731389 0.0006512264
 [51] 0.0002836794 0.0350794405 0.0030755348 0.0200743685 0.0096073098
 [56] 0.0001490646 0.0344751428 0.0065284632 0.0010669014 0.0952055412
 [61] 0.0023015815 0.0500916455 0.0946014632 0.0241321018 0.0323717994
```

```
 [66] 0.1023628239 0.0268371006 0.0165098458 0.0004487749 0.0519993255
 [71] 0.0002466268 0.0467989591 0.0114651087 0.0179021203 0.1100543101
 [76] 0.1163280205 0.0089745223 0.0041002420 0.0078496143 0.0012836939
 [81] 0.1883669348 0.0014777004 0.0273346373 0.0004410991 0.0101361131
 [86] 0.0434193135 0.0179283840 0.0584399377 0.0244649385 0.0033730967
 [91] 0.0420685102 0.0587623934 0.0183672105 0.0283228256 0.0001480132
 [96] 0.0052388875 0.1129750035 0.0037329264 0.0120461952 0.0004638270
```

Since we have got the r values,now we average them to get our final measures.

```
emp_mu_1=mean(mu_hat_1)
emp_mu_2=mean(mu_hat_2)
```

emp_bias_1 denotes emperical bias for sample mean.
emp_bias_2 denotes emperical bias for sample median.
Similar notations for MSE.

```
(emp_bias_1=emp_mu_1-mu)

[1] -0.007063162

(emp_bias_2=emp_mu_2-mu)

[1] -0.02358231

(emp_mse1=mean(mse1))

[1] 0.01795555

(emp_mse2=mean(mse2))

[1] 0.03233352
```

Question 5

Describe a suitable procedure to simulate random numbers from a probability distribution with PDF:
$f(x) = \frac{2x}{3} + \frac{4x^3}{6} + \frac{7x^6}{2}; 0 \le x \le 1$

Solution:

We use simulation for mixture models here.Here, X is a random variable from a mixture distribution with probability density function-$f(x) = \sum_{i=1}^{3} \pi_i f_i(x)$
where

$$\pi_1 = \frac{1}{3}, \pi_2 = \frac{1}{6}, \pi_3 = \frac{1}{2}, f_1(x) = 2x, f_2(x) = 4x^3, f_3(x) = 7x^6$$

Define $P_i = \sum_{j=1}^{i} \pi_j$ and $P_0 = 0$

Now we use a 2-step method:
1)Simulate U from Uniform(0,1).
2)If $P_{i-1} \le U < P_i$,generate an RV Z_i with pdf f_i.
3)Repeat the above 2 steps n times where n is the sample size.

Here,a random observation x from thr pdf $f_i()$ is generated by the inverting the CDF i.e. since $x \sim f_i(x)$ and we know the CDF $F_i(x) \sim$ Uniform(0,1), we generate a random observation from Uniform(0,1) and let it be u.
Then $x = F_i^{-1}(u)$ is a random observation from $f_i(x)$.
Here in our problem, $F_1(x) = x^2, F_2(x) = x^4, F_3(x) = x^7$
and hence $F_1^{-1}(u) = u^{\frac{1}{2}}$, $F_2^{-1}(u) = u^{\frac{1}{4}}$, $F_3^{-1}(u) = u^{\frac{1}{7}}$
First we declare the constants we will need.

```
(pi_1=1/3)

[1] 0.3333333

(pi_2=1/6)

[1] 0.1666667

(pi_3=1/2)

[1] 0.5

(p0=0)

[1] 0

(p1=pi_1)

[1] 0.3333333

(p2=pi_1+pi_2)

[1] 0.5

(p3=pi_1+pi_2+pi_3)

[1] 1
```

Now we generate the random sample.

```
fx=array(0)
for (i in 1:100)
{
  u=runif(1)
  if(u<p1){
    fx[i]=(runif(1))^(1/2)
  }else if(u>=p1 & u<p2){
    fx[i]=(runif(1))^(1/4)
  }else{
    fx[i]=(runif(1))^(1/7)
  }

}
fx
```

```
  [1] 0.8267660 0.8971360 0.8614325 0.9942462 0.9418763 0.8155280 0.6985764
  [8] 0.7660713 0.9171617 0.9159722 0.6367489 0.6941620 0.9865934 0.9546133
 [15] 0.8046515 0.7200881 0.8270046 0.8886673 0.9660960 0.6928639 0.9614958
 [22] 0.9421676 0.8357245 0.9294743 0.9401348 0.5712650 0.9784915 0.6367823
 [29] 0.7493284 0.8906208 0.6776265 0.8354276 0.8761195 0.6193968 0.5096240
 [36] 0.9010195 0.3577858 0.4153605 0.9258898 0.8742823 0.8167424 0.7989708
 [43] 0.8916835 0.6662597 0.7560196 0.9683291 0.9165722 0.9634918 0.8829364
 [50] 0.9010984 0.9400168 0.1517456 0.9717820 0.6698198 0.3517360 0.6721394
 [57] 0.6717405 0.7548552 0.8354413 0.8368929 0.7368732 0.9802527 0.8147945
 [64] 0.5990395 0.9958223 0.5834213 0.6566041 0.9032039 0.8020182 0.9860127
 [71] 0.7764741 0.9758253 0.8909905 0.7647986 0.9731436 0.9350176 0.9377554
 [78] 0.9161804 0.7506023 0.9951817 0.9737134 0.9274299 0.7951452 0.9816239
 [85] 0.9154323 0.8069289 0.6779200 0.9293684 0.7654218 0.9082161 0.9604335
 [92] 0.7024260 0.5617550 0.9084437 0.5124229 0.7978311 0.8692053 0.7301322
 [99] 0.7562683 0.9803623
```

Question 3

Simulate 10000 observations from the standard normal distribution in R. Also,simulate another set of 10000 observations using Box-Muller transformation. Compare these two sets of random numbers to assess whether you have simulated from the same distribution twice. Which statistical measure is appropriate to do that?

Solution

The Box Muller Transformation is a method to generate independent standard normal random variables from independent uniform random variables.
Let U_1 and U_2 be two independent Uniform(0,1) random variables.
Then, $Z_0 = \sqrt{-2log(U_1)}cos(2\pi U_2)$ and $Z_1 = \sqrt{-2log(U_1)}sin(2\pi U_2)$
are independent standard normal random variables.
We generate random sample of size 10000 from standard normal directly at first.

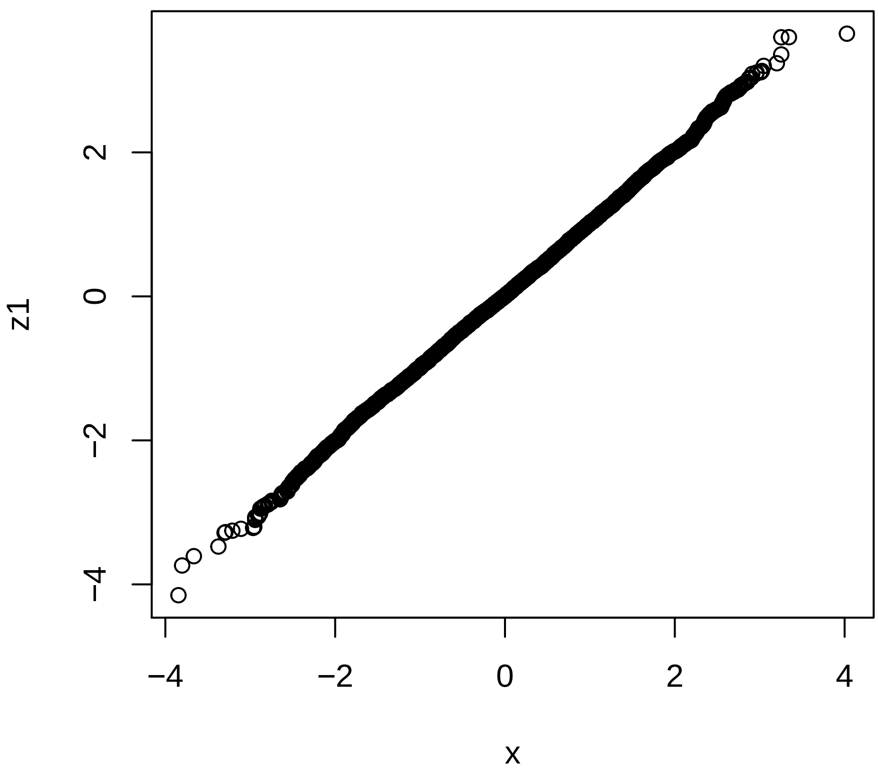
```
n=10000
x=rnorm(n)
```

Now we use the Box Muller Transformation to generate the same.

```
u1=runif(n)
u2=runif(n)
z0=sqrt(-2*log(u1))*cos(2*pi*u2)
z1=sqrt(-2*log(u1))*sin(2*pi*u2)
```

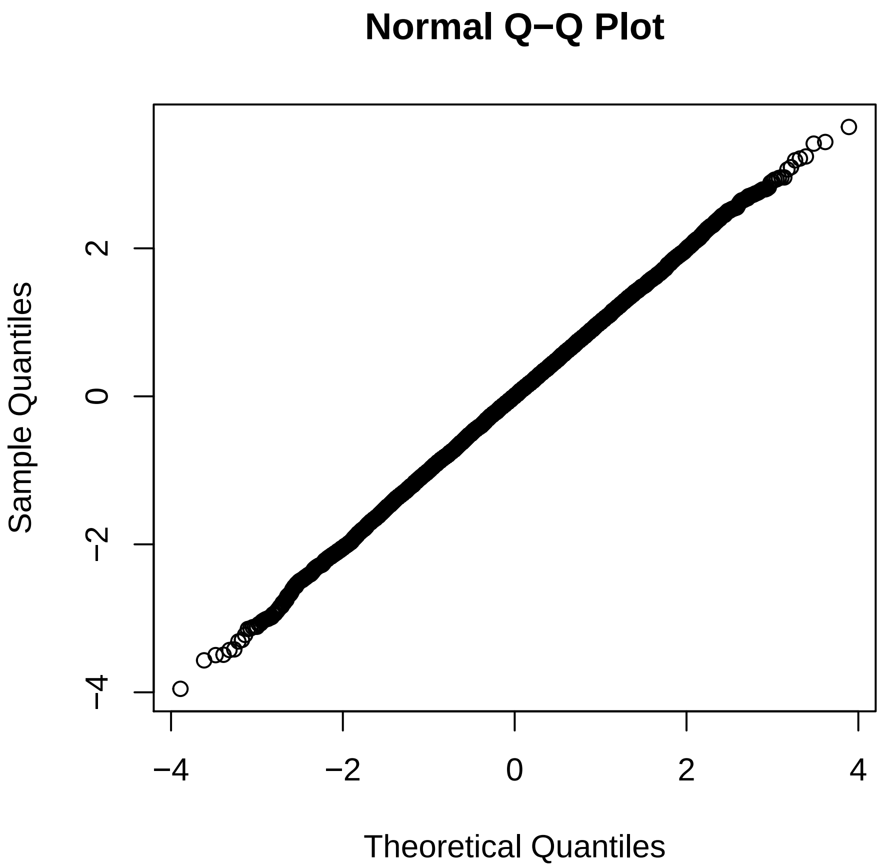
Now we compare both the samples by Quantile-Quantile plot.Here we compare sample x with sample z1.

```
qqplot(x,z1,plot.it = TRUE)
```



We can do the same comparison using the function qqnorm(),here we use the sample z0.

```
qqnorm(z0,main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       plot.it = TRUE)
```



Question 2
Find a dataset (real data) which needs regression techniques to be analyzed. Fit an appropriate regression model and check the assumptions. If there exists an interesting prediction problem (scientific, social or financial), describe it and comment on your findings.

Solution

Here I have a dataset named "concrete strength".
The dataset can be found at https://www.kaggle.com/prathamtripathi/regression-with-neural-networking .
The feature columns i,e explanatory variables are -

1. Cement
2. Blast Furnace Slag
3. Fly Ash
4. Water
5. Super-plasticizer
6. Coarse Aggregate
7. Fine Aggregate
8. Age

The target/response variable is "Stength of the Cement".
We want to fit a linear regression model in order to predict Strenth of Cement based on the features.
Let us load the dataset and check the no of observations and if there are any problematic elements.

```
data<-read.csv(file.choose(),header=TRUE)
head(data)

  Cement Blast.Furnace.Slag Fly.Ash Water Superplasticizer Coarse.Aggregate
1  540.0                0.0       0   162              2.5           1040.0
2  540.0                0.0       0   162              2.5           1055.0
3  332.5              142.5       0   228              0.0            932.0
4  332.5              142.5       0   228              0.0            932.0
5  198.6              132.4       0   192              0.0            978.4
6  266.0              114.0       0   228              0.0            932.0
  Fine.Aggregate Age Strength
1          676.0  28    79.99
2          676.0  28    61.89
3          594.0 270    40.27
4          594.0 365    41.05
5          825.5 360    44.30
6          670.0  90    47.03

dim(data)

[1] 1030    9

check<-apply(data, 2, function(x) any(is.na(x) | is.infinite(x) | is.null(x) | is.nan(x)))
check_counts<-length(check[as.vector(check)==TRUE])
check_counts

[1] 0
```

Since there are no problematic elements,we proceed to fit the linear model.

```
model<-lm(data$Strength~.,data=data)
summary(model)


Call:
lm(formula = data$Strength ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-28.654  -6.302   0.703   6.569  34.450 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -23.331214  26.585504  -0.878 0.380372    
Cement               0.119804   0.008489  14.113  < 2e-16 ***
Blast.Furnace.Slag   0.103866   0.010136  10.247  < 2e-16 ***
Fly.Ash              0.087934   0.012583   6.988 5.02e-12 ***
Water               -0.149918   0.040177  -3.731 0.000201 ***
Superplasticizer     0.292225   0.093424   3.128 0.001810 ** 
Coarse.Aggregate     0.018086   0.009392   1.926 0.054425 .  
Fine.Aggregate       0.020190   0.010702   1.887 0.059491 .  
Age                  0.114222   0.005427  21.046  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.4 on 1021 degrees of freedom
Multiple R-squared:  0.6155,Adjusted R-squared:  0.6125 
F-statistic: 204.3 on 8 and 1021 DF,  p-value: < 2.2e-16
```

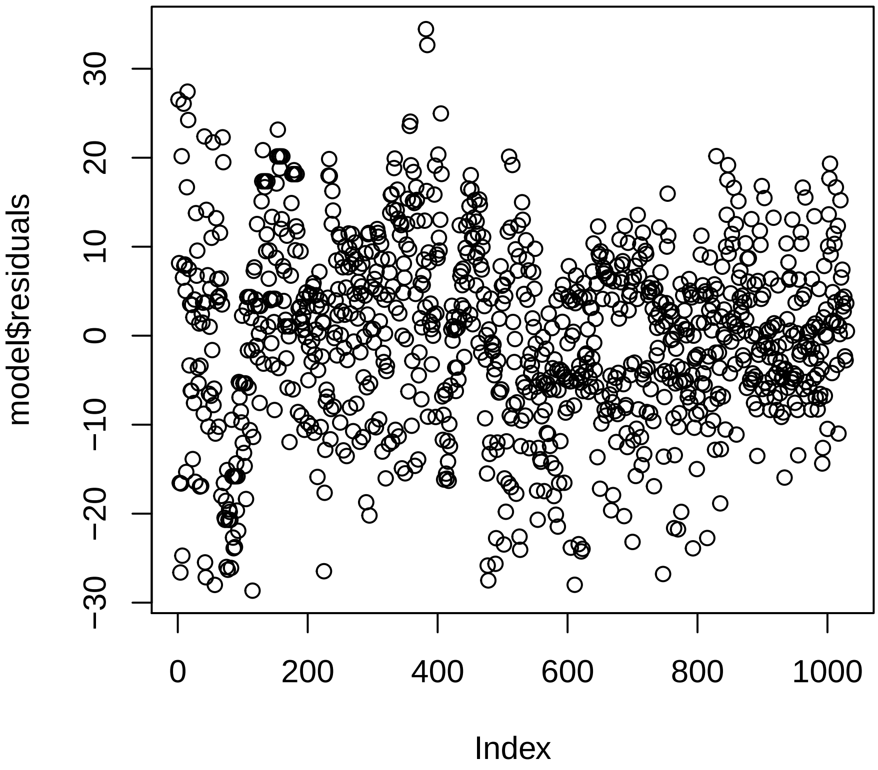
We check different assumptions.
Correlation between fitted values and residuals should be close to 0,so should be sum of residuals.

```
cor(model$fitted.values,model$residuals)

[1] 1.08368e-16

sum(model$residuals)

[1] 5.5006e-13
```

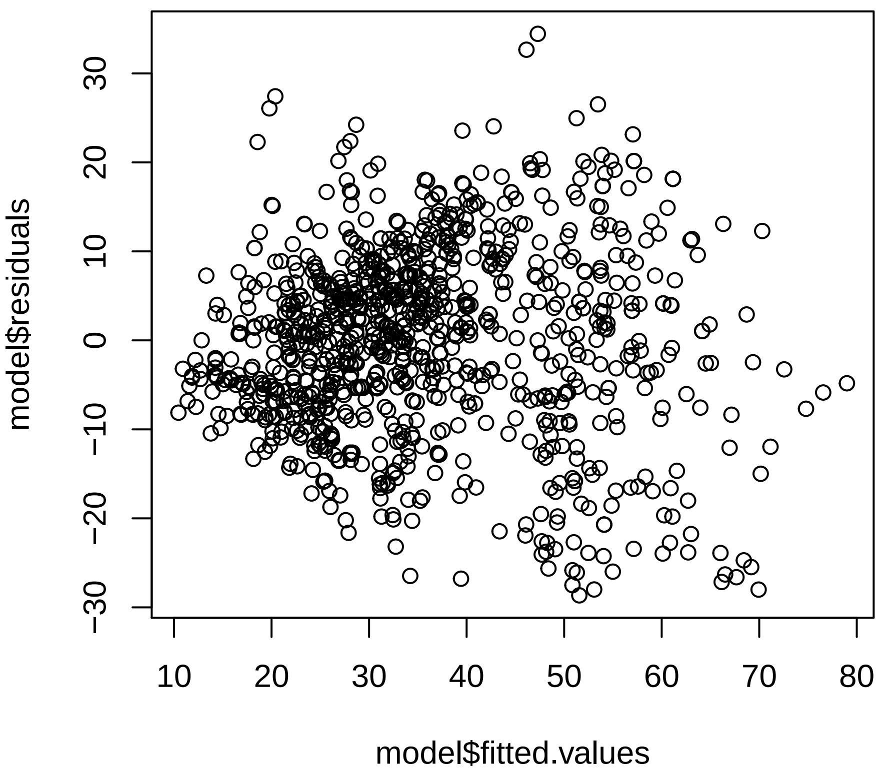
Residuals should not exhibit any pattern.

```
par(mfrow=c(2,1))
plot(model$residuals,main="RESIDUAL PLOT 1")
plot(model$fitted.values,model$residuals,main="RESIDUAL PLOT 2")
```

RESIDUAL PLOT 1

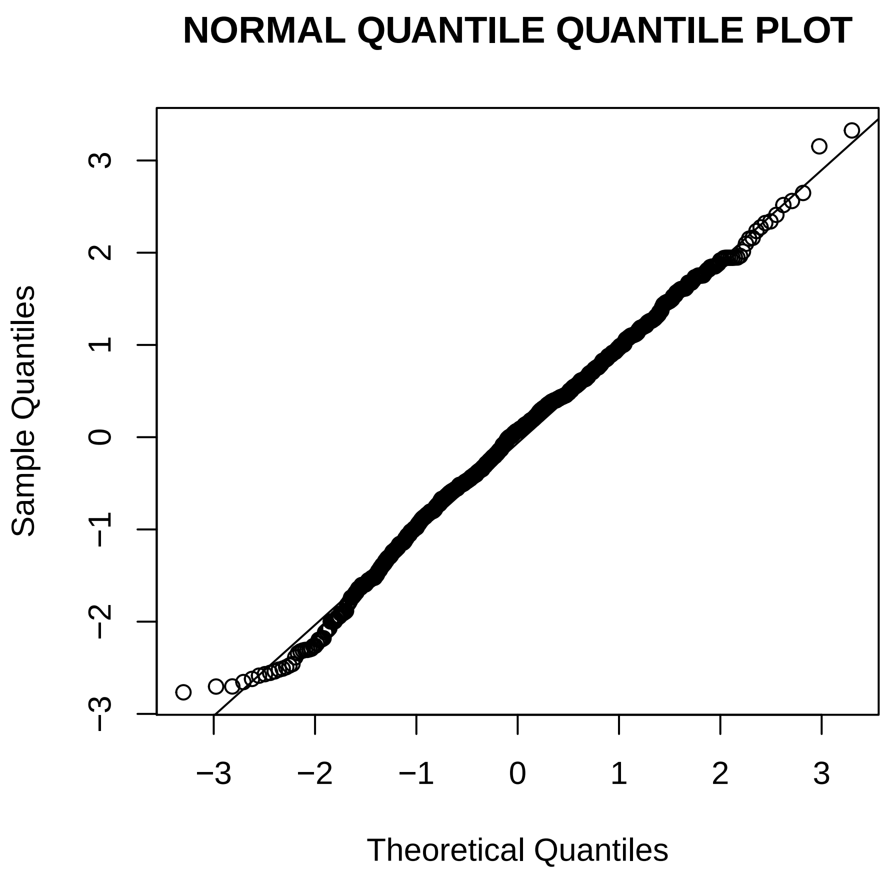


RESIDUAL PLOT 2



Errors should be normally distributed.

```
e=(model$residuals-mean(model$residuals))/sd(model$residuals)
shapiro.test(e)


Shapiro-Wilk normality test

data:  e
W = 0.99532, p-value = 0.002993

qqnorm(e,main="NORMAL QUANTILE QUANTILE PLOT")
qqline(rnorm(1000))
```



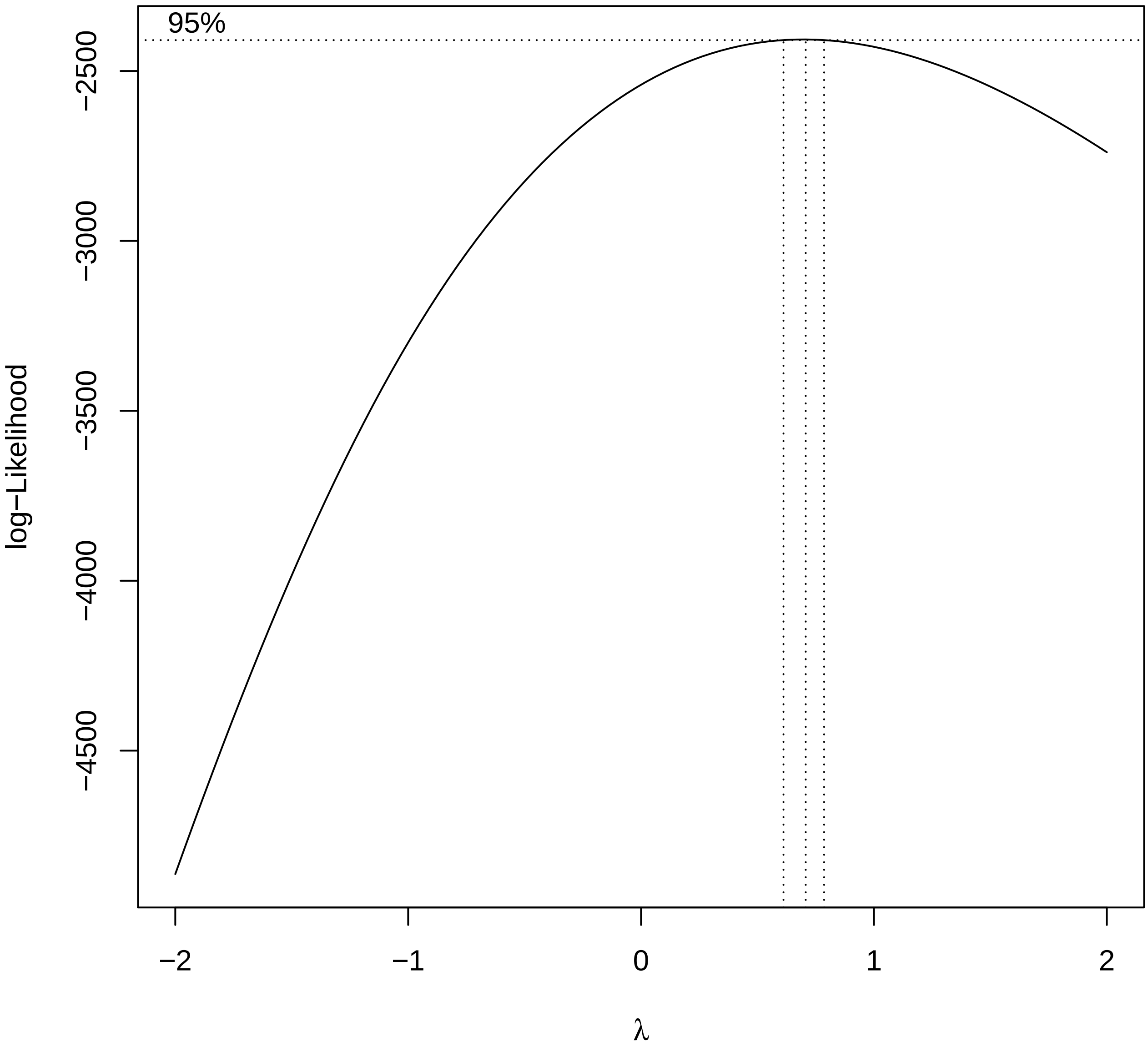
Let's see whether the design matrix is full rank or not.

```
X<-as.matrix(data[-9])
det(t(X)%*%X)

[1] 1.253427e+53

library(olsrr)

Warning:  package 'olsrr' was built under R version 4.0.5

Attaching package:  'olsrr'
The following object is masked from 'package:datasets':
      rivers
```

```
ols_coll_diag(model)[1]

$vif_t
           Variables Tolerance      VIF
1             Cement 0.1335302 7.488944
2 Blast.Furnace.Slag 0.1374200 7.276963
3            Fly.Ash 0.1620579 6.170634
4              Water 0.1427764 7.003957
5   Superplasticizer 0.3374074 2.963776
6   Coarse.Aggregate 0.1970592 5.074617
7     Fine.Aggregate 0.1427535 7.005081
8                Age 0.8941612 1.118367
```

Hence,intuitively,problem of collinearity should not be serious.
Apart from that,the other assumptions do not hold.
Let us to do box cox transformation.

```
library(MASS)


Attaching package:  'MASS'
The following object is masked from 'package:olsrr':
      cement

bc <- boxcox(model)
```




```
lambda <- bc$x[which.max(bc$y)]
y_new=((data$Strength)^(lambda)-1)/lambda
df<-data[-9]
df[9]<-y_new
c1=colnames(data)[-9]
colnames(df)=c(c1,"y")
head(df)
```

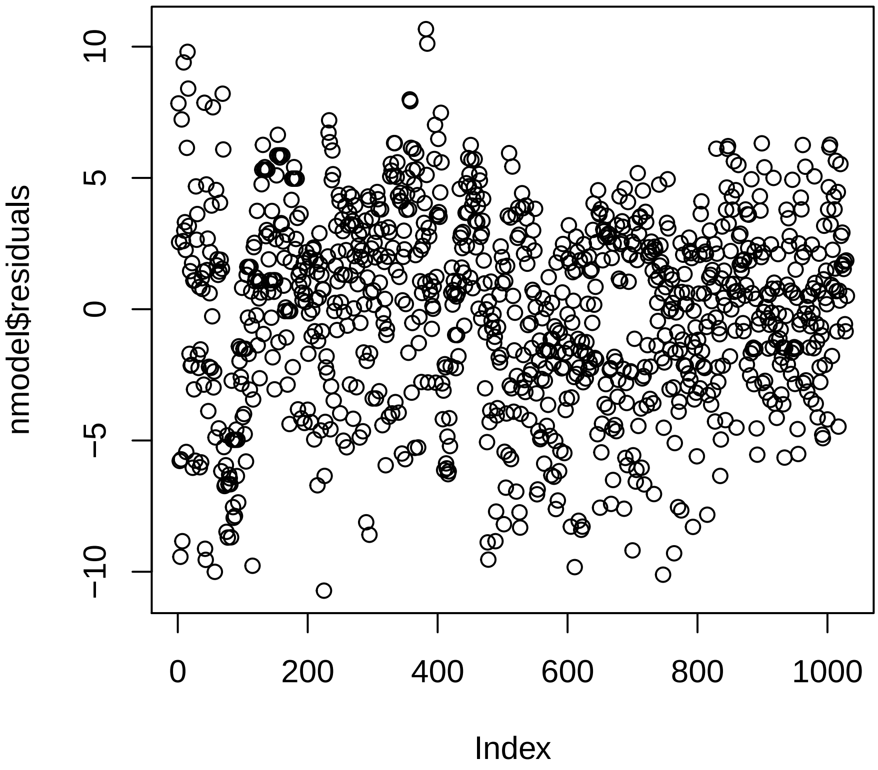
```
  Cement Blast.Furnace.Slag Fly.Ash Water Superplasticizer Coarse.Aggregate
1  540.0                0.0       0   162              2.5           1040.0
2  540.0                0.0       0   162              2.5           1055.0
3  332.5              142.5       0   228              0.0            932.0
4  332.5              142.5       0   228              0.0            932.0
5  198.6              132.4       0   192              0.0            978.4
6  266.0              114.0       0   228              0.0            932.0
  Fine.Aggregate Age        y
```

```
1           676.0  28 29.92708
2           676.0  28 24.72776
3           594.0 270 17.87751
4           594.0 365 18.14098
5           825.5 360 19.22340
6           670.0  90 20.11475
```

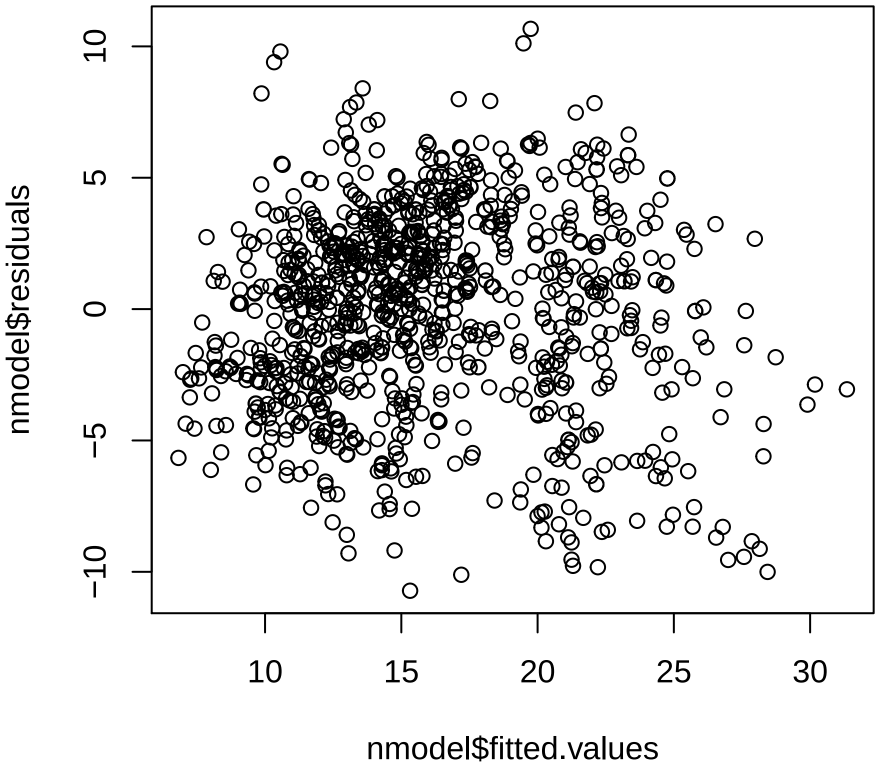
Fit the model and repeat the same tasks.

```
nmodel<-lm(df$y~.,data=df)
summary(nmodel)


Call:
lm(formula = df$y ~ ., data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.7200  -2.4227   0.5112   2.5693  10.6675 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)        -4.380403   9.481842  -0.462 0.644195    
Cement              0.042167   0.003028  13.927  < 2e-16 ***
Blast.Furnace.Slag  0.036076   0.003615   9.980  < 2e-16 ***
Fly.Ash             0.032187   0.004488   7.172 1.42e-12 ***
Water              -0.051005   0.014329  -3.559 0.000389 ***
Superplasticizer    0.104688   0.033320   3.142 0.001727 ** 
Coarse.Aggregate    0.006011   0.003350   1.794 0.073062 .  
Fine.Aggregate      0.006353   0.003817   1.665 0.096314 .  
Age                 0.041216   0.001936  21.294  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.709 on 1021 degrees of freedom
Multiple R-squared:  0.6114,Adjusted R-squared:  0.6083 
F-statistic: 200.8 on 8 and 1021 DF,  p-value: < 2.2e-16

enew=(nmodel$residuals-mean(nmodel$residuals))/sd(nmodel$residuals)
par(mfrow=c(2,1))
plot(nmodel$residuals,main="NEW RESIDUAL PLOT 1")
plot(nmodel$fitted.values,nmodel$residuals,main="NEW RESIDUAL PLOT 2")
```

NEW RESIDUAL PLOT 1



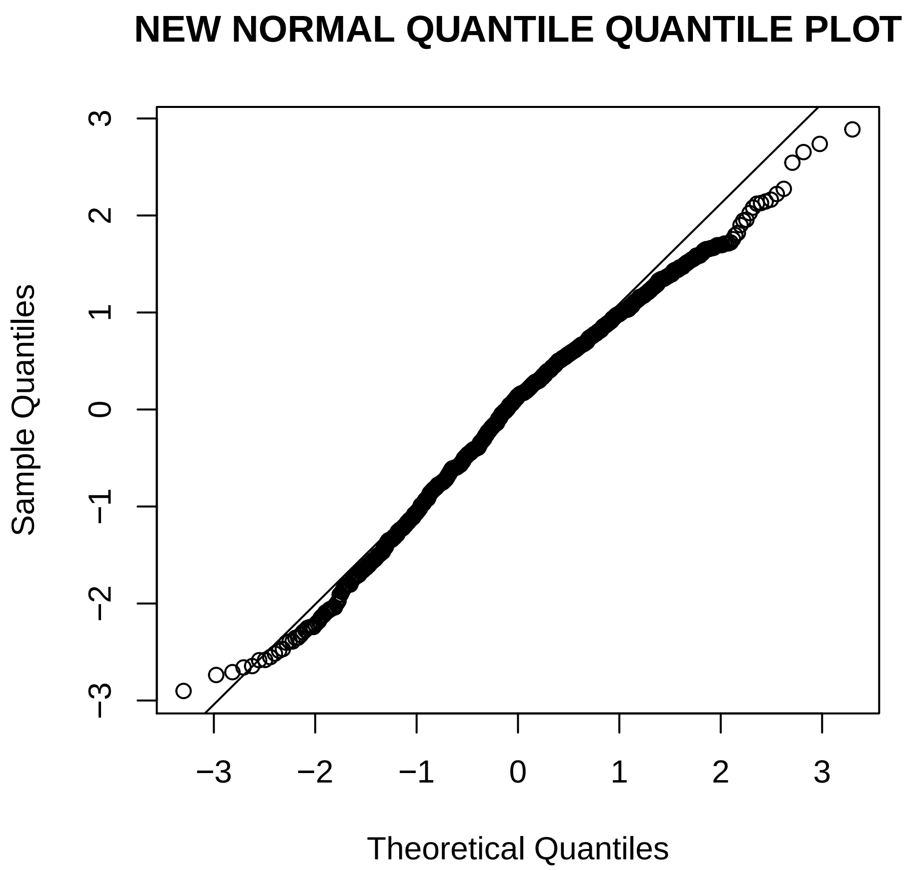
NEW RESIDUAL PLOT 2



```r
shapiro.test(enew)
```

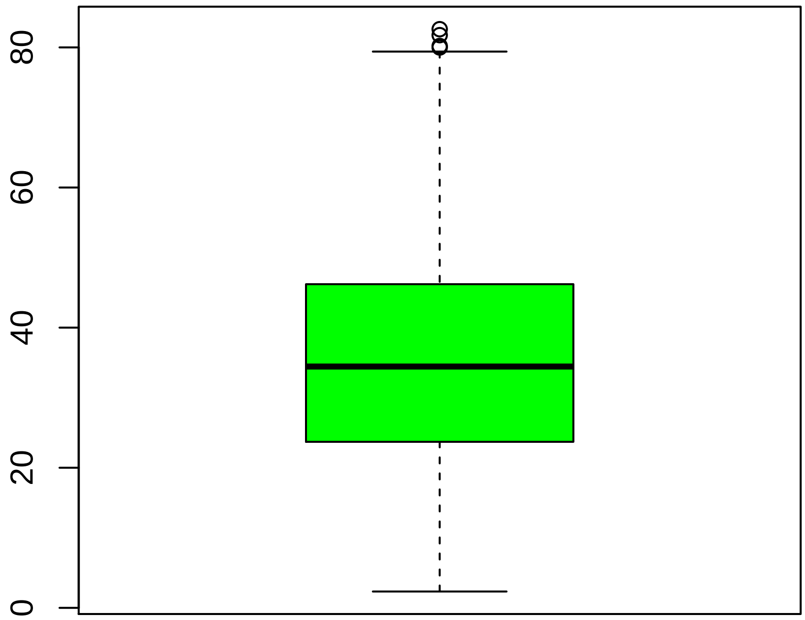
```
Shapiro-Wilk normality test

data:  enew
W = 0.98941, p-value = 8.959e-07
```

```r
qqnorm(enew,main="NEW NORMAL QUANTILE QUANTILE PLOT")
qqline(rnorm(1000))
```



The results are worse. Maybe Box Cox performs well for skewed data,for heavy tailed data we may use some other transformation

```r
boxplot(data$Strength,col="green")
```



This may also be the reason, it doesn't perform well is presence of outliers.

Question 4

Suppose $(Y_i, x_i); i = 1, 2, \ldots n$ be a bivariate dataset.We want to treat x as the explanatoryV and Y as the response variable and fit a linear regression model. But instead of Least Squared Estimation, we want to minimize the following objective function in order to estimate the unknown regression parameters.

$\sum_{i=1}^{n} |Y_i - \alpha - \beta x_i|$

Describe a suitable iterative procedure (as closed form solution does not exist in this case) and provide an R code.

Solution

The procedure we use here is "Iteratively Reweighted Least Squares".

In general , if we have a function like $\sum_{i=1}^{n} f(X_i - \mu)$ and we want to optimize that w.r.to μ, the steps are as follows:-

1)Write $A = \sum_{i=1}^{n} w_i (X_i - \mu)^2$, where $w_i = \frac{f(x_i - \mu)}{(X_i - \mu)^2}$

2)Start with an initial value of the vector $(w_1, w_2, \ldots, w_n)$.

3)Obtain an estimate of μ say $\hat{\mu}$ by optimising A.

4)With the help of this $\hat{\mu}$, obtain w_i again.

5)Repeat the same procedure till the difference between 2 consecutive estimates becomes very small.

Here in our problem,$f(\alpha, \beta) = \sum_{i=1}^{n} |Y_i - \alpha - \beta x_i|$. Hence, our $w_i = \frac{\sum_{i=1}^{n} |Y_i - \alpha - \beta x_i|}{\sum_{i=1}^{n} (Y_i - \alpha - \beta x_i)^2}$.

Let the design matrix of regression be X and X_i denote the i^{th} row of X. Initially we take $w_i = 1 \forall i = 1(|)n$.

At the $(t+1)^{th}$ iterate, $(\alpha, \beta)^{(t+1)} = (X^t W^{(t)} X)^{-1} X^t W^{(t)} \mathbf{y}$,where $\mathbf{y}$ is the vector of dependent variables.

where $w_i^{(t)} = \frac{1}{|y_i - X_i \beta^{(t)}|}$ and $W^{(t)}$=diag$(w_1, \ldots, w_n)$.

Here I generated a self-made data and ran a simulation to verify how well the algorithm works.

```
n<-1000
b0=2
b1=3
x1<-runif(n)
e<-rnorm(n)
y=b0+b1*x1+e
X<-matrix(c(rep(1,n),x1),ncol=2)
w=rep(1,n)
eps1=1
eps2=2
b_old=(solve(t(X)%*%X))%*%t(X)%*%y
while(eps1>0.001 | eps2>0.001){
  for(i in 1:n){
    w[i]=max((abs(y[i]-X[i,]%*%b_old)),0.001)
  }
  W=diag(w)
  b_new=(solve(t(X)%*%solve(W)%*%X))%*%t(X)%*%solve(W)%*%y
  eps1=abs(b_new[1]-b_old[1])
  eps2=abs(b_new[2]-b_old[2])
  b_old=b_new
}
b_new

         [,1]
[1,] 1.986071
[2,] 2.984830
```