MMR_TP1

Guangyue CHEN

Application:Facebook data set

```r
rm(list=ls())
tab<-read.table("/Users/pingguo/WTF/UE/S3/MMR/facebookdata.txt",sep = ";",header= TRUE)
dim(tab)
```

```
## [1] 14  2
```

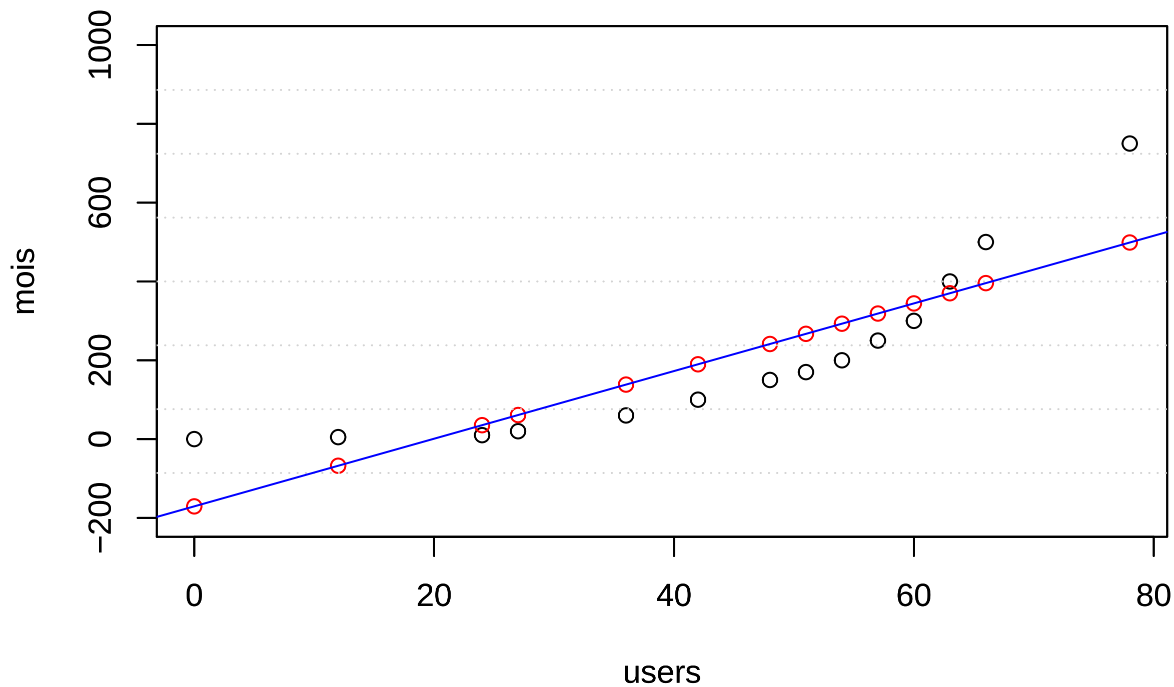
```r
modreg=lm(users~.,tab)
summary(modreg)
```

```
##
## Call:
## lm(formula = users ~ ., data = tab)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -97.07 -86.94 -42.70  62.00 251.17
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -170.695     70.952  -2.406   0.0332 *
## mois           8.584      1.449   5.926 6.97e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 115 on 12 degrees of freedom
## Multiple R-squared:  0.7453, Adjusted R-squared:  0.7241
## F-statistic: 35.11 on 1 and 12 DF,  p-value: 6.97e-05
```

With the function of R, we find that for the numbers of users, mouthes do have the influence. And the coefficients estimated is 8.584 with the intercept -170.695. With the t-test, we find that for each coefficients, we refuse the Hypothesis which is that the coefficients can be 0. And the coefficient of the mouths is not 0 only has the risk 0.001. And then, the value of R-squred is 0.7453 which is high, so the regression equation fits the observations very well. For the F-test, wu refuse both of the coefficients is 0.

```r
Y_esti<-predict(modreg,tab)
Y<-tab$users
plot(x=tab$mois,y=Y,ylim = c(-200,1000),ylab="mois",xlab="users")
par(new=TRUE)
plot(x=tab$mois,y=Y_esti,col="red",ylim = c(-200,1000),ylab="mois",xlab="users")

grid(nx=NA,ny=8,col="lightgray")
abline(coef = coef(modreg),col="blue")
```



And then we plot the predictive values and the real values do have the deviations, with the help of function “summary”, the residual standard error is 115.

The second methode, we compute the results by ourselves. First on compute the coefficients.

```
coefs<-function(table,n,p){
  Y<-table$users
  x<-as.matrix(table[,c(seq(1,p-1))])
  un<-matrix(1,nrow=n,ncol=1)
  X<-cbind(un,x)
  belta<-solve(t(X)%*%X)%*%t(X)%*%Y
  return(belta)
}
d=dim(tab)
coefs(tab,d[1],d[2])
```

```
##              [,1]
## [1,] -170.695067
## [2,]    8.583707
```

they are same as the results of “lm” function. Than we compute the predictive values and the RMSE

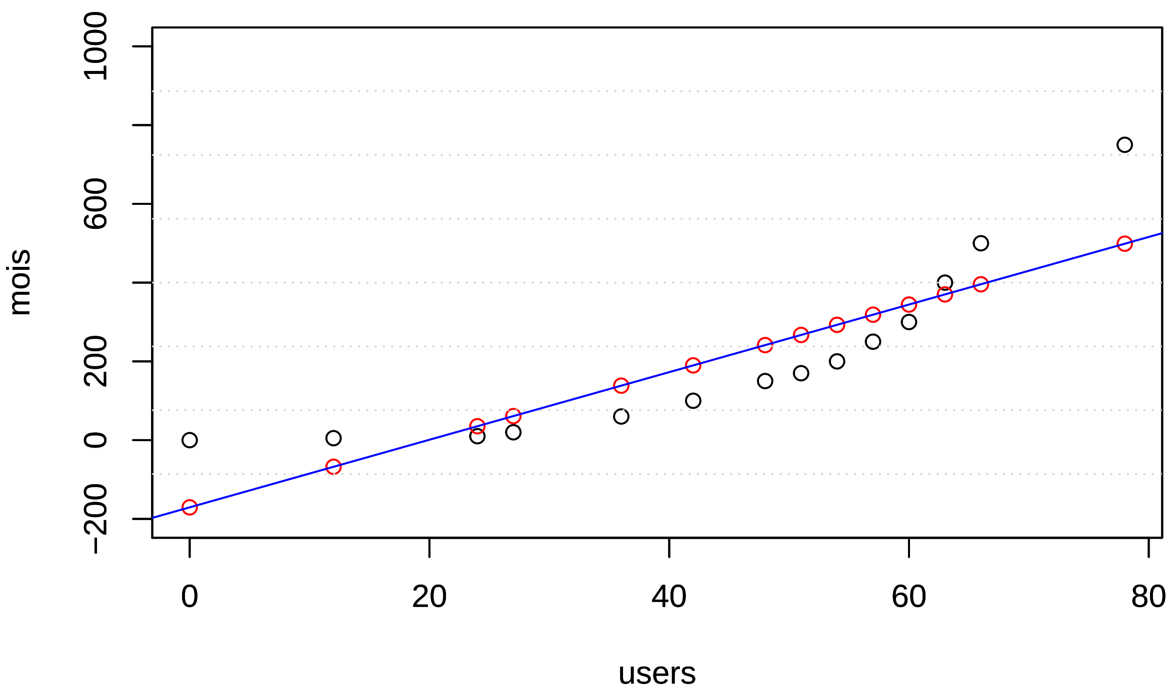
```
Y_estimate<-function(table,n,p){
  Y<-table$users
  x<-as.matrix(table[,c(seq(1,p-1))])
  un<-matrix(1,nrow=n,ncol=1)
  X<-cbind(un,x)
  belta<-solve(t(X)%*%X)%*%t(X)%*%Y
  return(X%*%belta)
}
Y_esti=Y_estimate(tab,d[1],d[2])
print(Y_esti)
```

```
##              [,1]
##  [1,] -170.69507
```

```
## [2,] -67.69058
## [3,]  35.31390
## [4,]  61.06502
## [5,] 138.31839
## [6,] 189.82063
## [7,] 241.32287
## [8,] 267.07399
## [9,] 292.82511
## [10,] 318.57623
## [11,] 344.32735
## [12,] 370.07848
## [13,] 395.82960
## [14,] 498.83408
```

```r
plot(x=tab$mois,y=Y,ylim = c(-200,1000),ylab="mois",xlab="users")
par(new=TRUE)
plot(x=tab$mois,y=Y_esti,col="red",ylim = c(-200,1000),ylab="mois",xlab="users")

grid(nx=NA,ny=8,col="lightgray")
abline(coef = coef(modreg),col="blue")
```




```r
RMSE<-function(Y,Y_esti){
sum=0

err<-vector(length=length(Y))
for(i in seq(1,length(Y))){
  sum<-sum+(Y_esti[i]-Y[i])^2

  err[i]=Y[i]-Y_esti[i]
}
RMSE<- sqrt(sum/length(Y))

return(RMSE)
```

```
}
RMSE(Y,Y_esti)
```

```
## [1] 106.5122
```

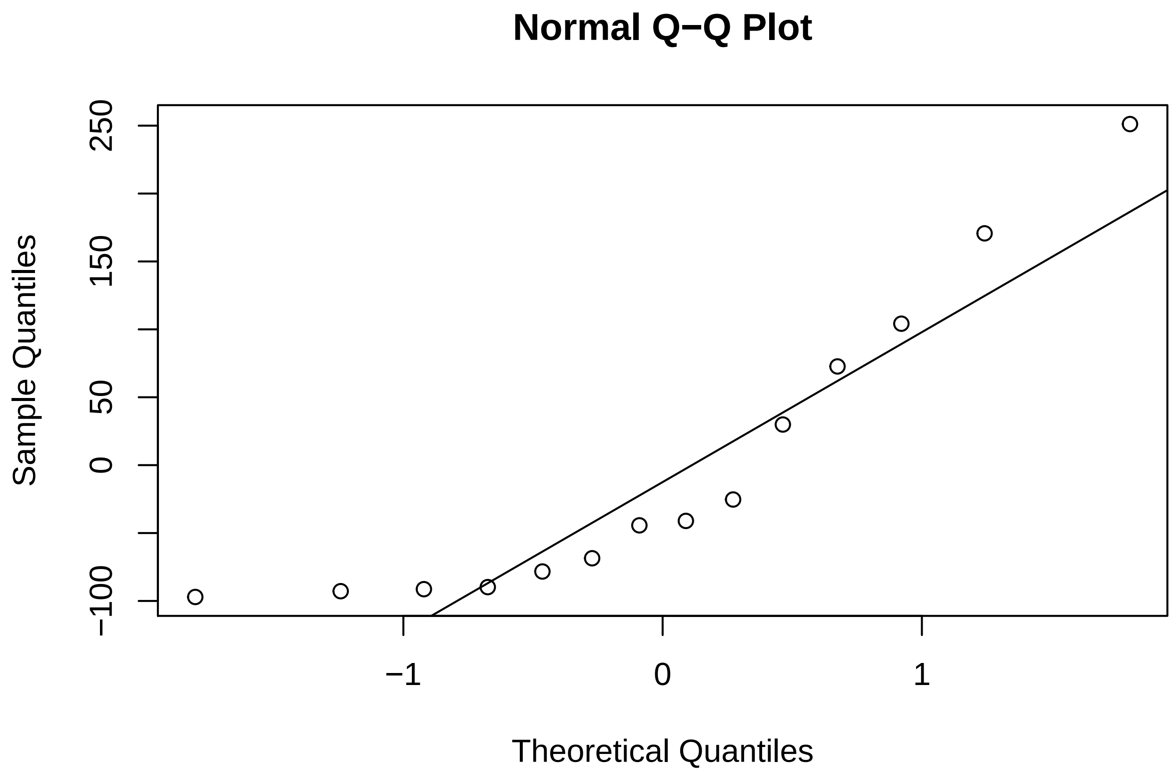
The results are same as the residual standard error that we got before.

```
residual<-function(Y,Y_esti){

err<-vector(length=length(Y))
for(i in seq(1,length(Y))){

  err[i]=Y[i]-Y_esti[i]
}

return(err)
}
err<-residual(Y,Y_esti)
qqnorm(err)
qqline(err)
```




```
shapiro.test(err)
```

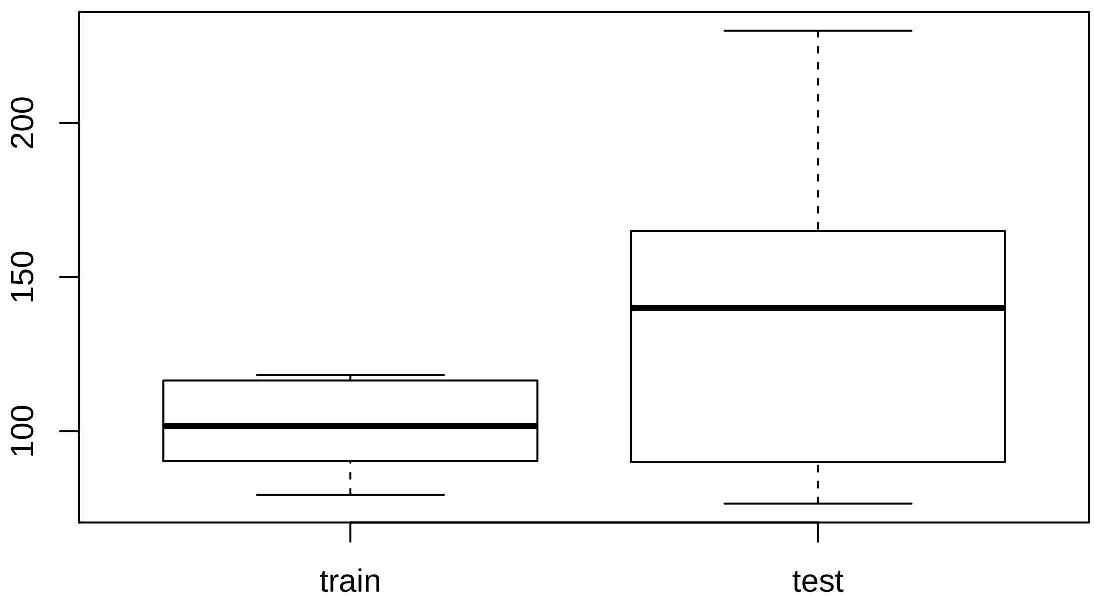
```
##
##  Shapiro-Wilk normality test
##
## data:  err
## W = 0.83562, p-value = 0.01425
```

Although there are some bias, the points are almost a line, so the residual is with the distribution of Normality.

And then we can test the model with the new set.

```
random_partitioning<-function(table){
smp1<-sample(nrow(table), nrow(table)*0.75)
train_data=tab[smp1,]
test_data=tab[-smp1,]
model=lm(users~.,train_data)

Y_esti=predict(model,train_data)
Y=train_data$users
train<-RMSE(Y,Y_esti)
Y_test<-predict(model,newdata=test_data)
Y=test_data$users
test<-RMSE(Y,Y_test)
return(c(train,test))
}
train<-vector(length = 10)
test<-vector(length = 10)
for(i in seq(1,10)){
  res=random_partitioning(tab)
  train[i]=res[1]
  test[i]=res[2]
}
boxplot(data.frame(train,test))
```



With the result we find that the test data set has a higher root mean squred error, but it doesn't have a big difference with the train data set. So we can say that this model fit the new data very well. And the limit is that the data set is a little small so it has high error, with more data we suppose the error will be lower. And with the plot of the real numbers of users, it's not really a stright line, so the Nonlinear regression perhaps will have a better model.

Boston housing data

```
rm(list=ls())
library(mlbench)
data(BostonHousing)
```

so first, we build the trainning data set and the test data set, and get the linear model.

```
smp1<-sample(nrow(BostonHousing), nrow(BostonHousing)*0.75)
train_data=BostonHousing[smp1,]
test_data=BostonHousing[-smp1,]
model=lm(medv~.,train_data)
summary(model)
```

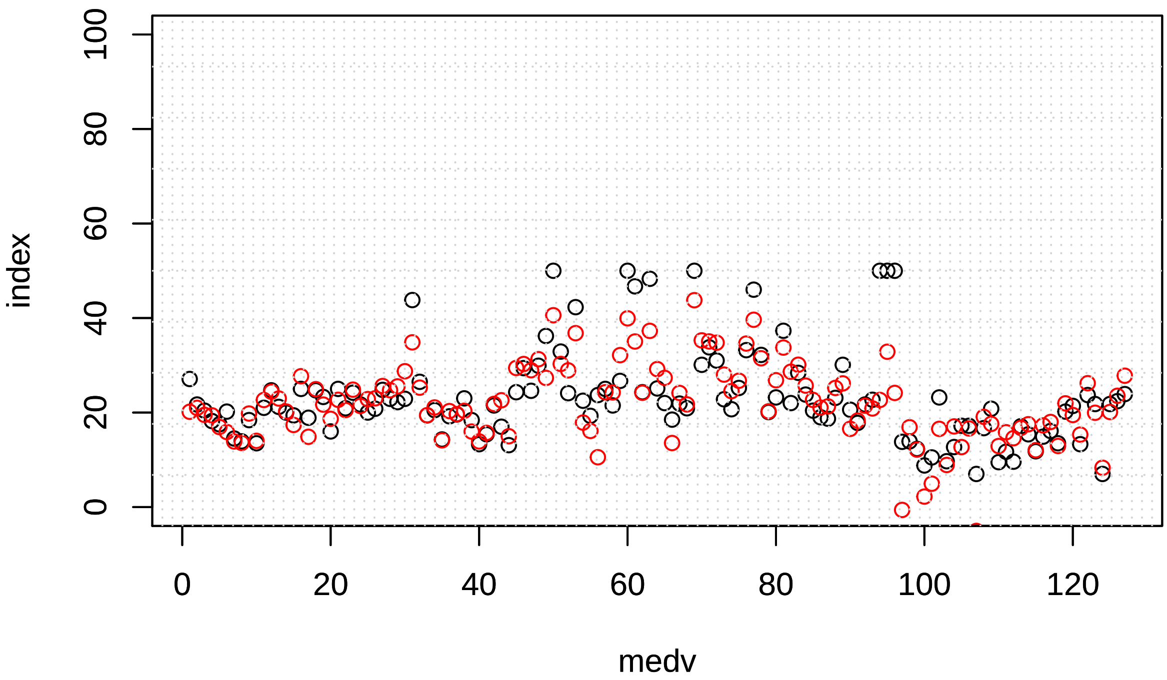
```
##
## Call:
## lm(formula = medv ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.5095  -2.5807  -0.5096   1.9110  26.0307
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  34.549697   5.569166   6.204 1.50e-09 ***
## crim         -0.115161   0.032210  -3.575 0.000397 ***
## zn            0.041516   0.015228   2.726 0.006714 **
## indus         0.047915   0.064860   0.739 0.460537
## chas1         1.817400   1.020272   1.781 0.075697 .
## nox         -15.261304   4.214165  -3.621 0.000334 ***
## rm            3.984129   0.453841   8.779  < 2e-16 ***
## age           0.007844   0.014662   0.535 0.592980
## dis          -1.270058   0.217700  -5.834 1.19e-08 ***
## rad           0.265236   0.071622   3.703 0.000246 ***
## tax          -0.013019   0.003955  -3.292 0.001094 **
## ptratio      -1.001713   0.145054  -6.906 2.23e-11 ***
## b             0.007443   0.002844   2.617 0.009230 **
## lstat        -0.525209   0.055744  -9.422  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.476 on 365 degrees of freedom
## Multiple R-squared:  0.7644, Adjusted R-squared:  0.756
## F-statistic: 91.08 on 13 and 365 DF,  p-value: < 2.2e-16
```

As the result, we find that we can accept that the coefficient of "indus","chas1" and "age" can be 0(we dont refuse the Hypothesis that coefficient is 0). so we can choose the other 10 variables and the Intercept. This model is with a high R-Squared and it pass the F-test, so we can use it to predict our results.

```
Y_esti<-predict(model,newdata=test_data,level = 0.999)
Y_test<-test_data$medv
```

```
plot(Y_test,ylim = c(0,100),ylab="index",xlab="medv")
par(new=TRUE)
plot(Y_esti,col="red",ylim = c(0,100),ylab="index",xlab="medv")
```

```r
grid(nx=100,ny=10,col="lightgray")
```



With the compare bitween predictive value of test data set and the real medv, the model fit the new data set very well.

```r
RMSE<-function(Y,Y_esti){
sum=0

err<-vector(length=length(Y))
for(i in seq(1,length(Y))){
  sum<-sum+(Y_esti[i]-Y[i])^2

  err[i]=Y[i]-Y_esti[i]
}
RMSE<- sqrt(sum/length(Y))

return(RMSE)
}
RMSE(Y_test,Y_esti)
```

```
##        8
## 5.558819
```

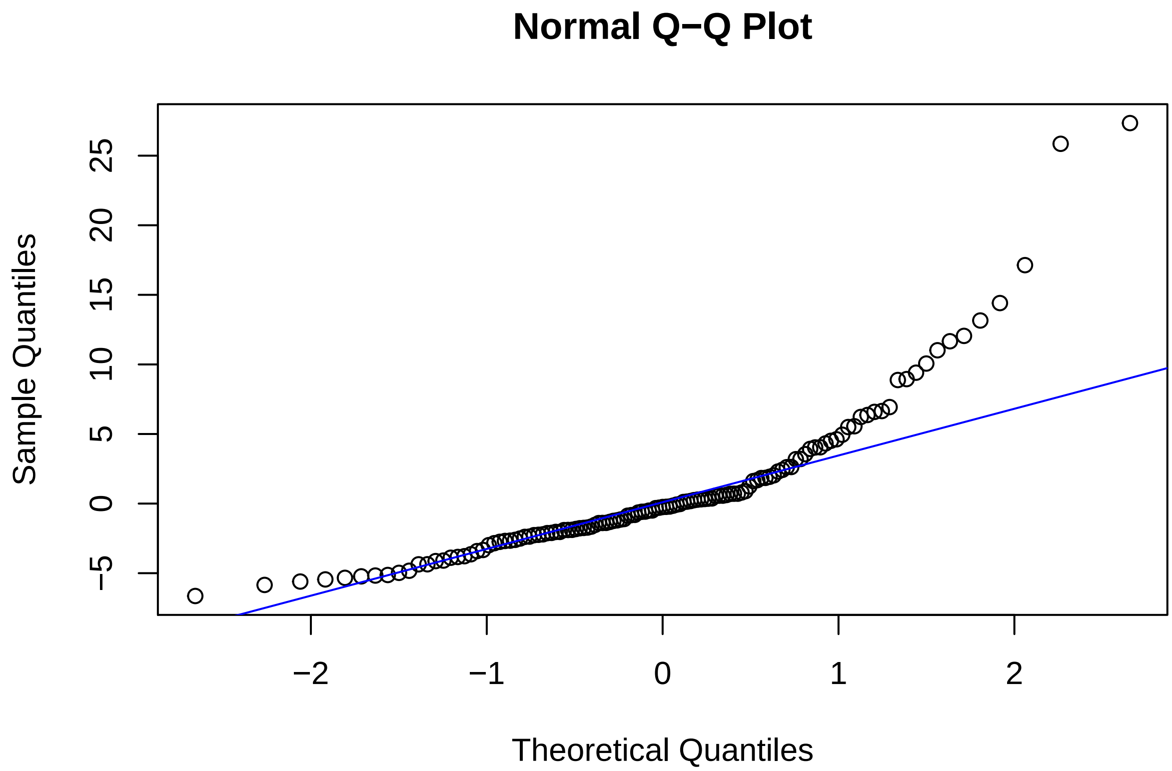
```r
residual<-function(Y,Y_esti){

err<-vector(length=length(Y))
for(i in seq(1,length(Y))){

  err[i]=Y[i]-Y_esti[i]
}

return(err)
}
err<-residual(Y_test,Y_esti)
```

```r
qqnorm(err)
qqline(err,col="blue")
```




```r
shapiro.test(err)
```

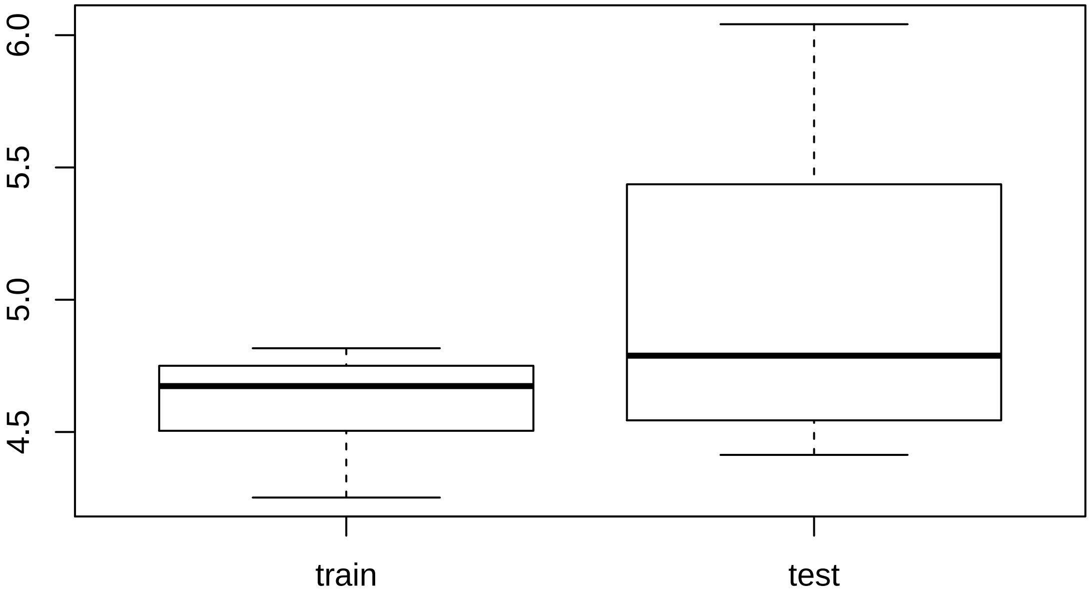
```
##
##  Shapiro-Wilk normality test
##
## data:  err
## W = 0.81322, p-value = 2.057e-11
```

so we see that the Normal Q-Q Plot is almost a stright line , at last where theoretical quantiles > 2 ,there are somme differences, that's the result we most expect to see. The plots with low significance are same as the results we experct. And the plots with high significance is higher than the line blue. So we can summarize that its a good linear model.

```r
random_partitioning<-function(table){
smp1<-sample(nrow(table), nrow(table)*0.75)
train_data=BostonHousing[smp1,]
test_data=BostonHousing[-smp1,]
model=lm(medv~.,train_data)

Y_esti=predict(model,train_data)
Y=train_data$medv
train<-RMSE(Y,Y_esti)
Y_test<-predict(model,newdata=test_data)
Y=test_data$medv
test<-RMSE(Y,Y_test)
return(c(train,test))
}
```

```r
train<-vector(length = 10)
test<-vector(length = 10)
for(i in seq(1,10)){
  res=random_partitioning(BostonHousing)
  train[i]=res[1]
  test[i]=res[2]
}
boxplot(data.frame(train,test))
```



With the result we find that the test data set has a higher root mean squred error, but it doesn't have a big difference with the train data set.