REGRESSION PROJECT

Rasamanjari Nandan (191111)

(M.Sc. Statistics)

Indian Institute of Technology, Kanpur

The present analysis deals with predicting the median value of a home **(MEDV)** using 10 predictors, *viz.*, **CRIM, INDUS, NOX, RM, AGE, DIS, TAX, PTRATIO, B, LSTAT**, which have been described in the Boston Housing Dataset. For an effective analysis, we have standardized the 10 regressors. Further, we have divided the entire dataset containing 506 observations into a test set of 100 observations selected at random, and a training set of remaining 406 observations.

To the training data, we have fit a linear regression model of the form

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{0.1}$$

through the method of ordinary least squares. Using this regression equation, we have predicted the values of the response MEDV for the test set and tried to determine the extent to which these values agree with the actual values provided to us. The measure used for this purpose is called Root Mean Squared Error, mathematically expressed as

$$RMSE = ||\boldsymbol{y}_{test} - \hat{\boldsymbol{y}}_{test}||/\sqrt{n}. \tag{0.2}$$

For the present pair of training and test data, the value of RMSE has been obtained as **4.6894**.

In the instant problem, we are essentially interested to check whether certain assumptions of OLS indeed hold or not for the model.

1 Dealing with Leverage Points and Outliers

At first, we shall try to identify and subsequently discard the leverage points in the training data. To that effect, we consider the diagonal elements h_{ii}'s of the projection matrix $X(X^TX)^{-1}X^T$. An observation would be declared a leverage point if its corresponding value of h_{ii} exceeds a certain threshold. Theoretically, a cutoff of $2p/n$ serves the purpose, where p is the number of regression coefficients in the linear regression model (in this case, $p = 11$, including the intercept term) and n is the number of observations (to begin with, we have $n = 406$). For the given problem, we obtain a cutoff equal to 0.0542, which suspects 34 leverage points. However, this number seems to be large. In order to avoid too many detections of leverage points, we increase the cutoff by 0.04. In fact, a plot of h_{ii} versus i (see Figure 1.1) suggests that the cutoff 0.0542 is indeed quite conservative, and and enhancement of the same by 0.025 may seem appropriate. Besides, if we consider the RMSE after excluding the 34 observations, we observe that it increases to 4.9206. Accordingly, we deem fit to proceed with a threshold of $(0.0542 + 0.04)$, that is, **0.0942**. Based on this threshold, we would discard 5 observations (which is small enough), marginally reducing the RMSE to **4.6805**. The number of observations consequently becomes 401.

Once we have dropped the leverage points, our interest shifts to detection of influential points. Towards that, we would employ four measures - Cook's Distance, $DFFITS$, COVRATIO and DFBETAS, whose working formulae and theoretical cutoffs are mentioned below.

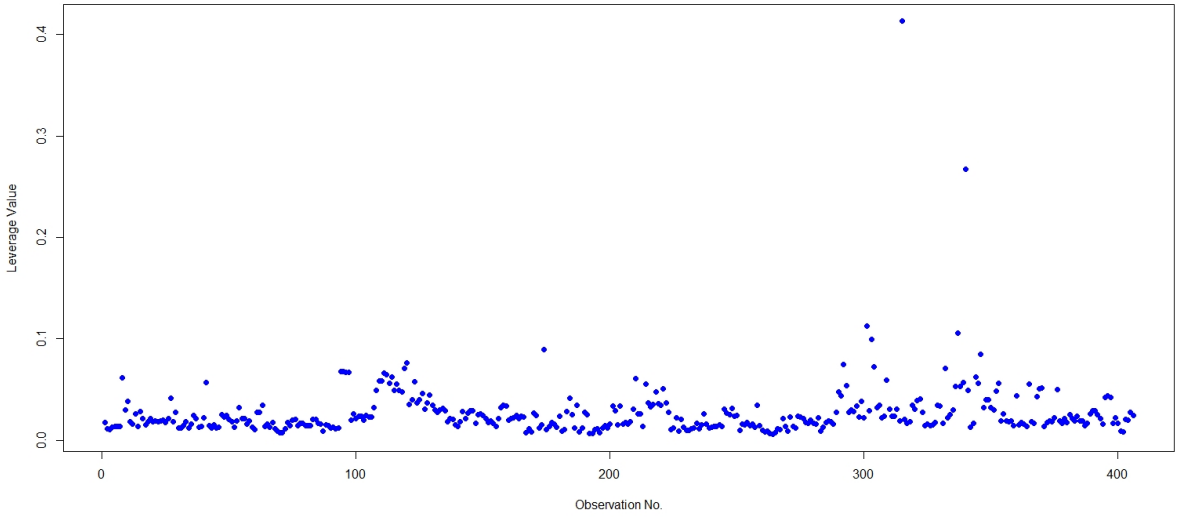


Figure 1.1: Plot of Leverage Values (h_{ii})

- **Cook's Distance:**

$$D_i = \frac{||\hat{\boldsymbol{y}}_{(i)} - \hat{\boldsymbol{y}}_i||^2}{pMS_{Res}} \tag{1.1}$$

Here, $\hat{\boldsymbol{y}_{(i)}}$ denotes the predicted response vector obtained by fitting the regression model excluding the i-th observation. Typically, $D_i > 1$ suspects the i-th observation to be influential.

- **DFFITS:**

$$DFFITS_i = \frac{\hat{y}_{(i)} - \hat{y}_i}{\sqrt{h_{ii}S^2_{(i)}}} \tag{1.2}$$

In this case, $\hat{y}_{(i)}$ is the predicted value of y_i based on the model excluding the i-th observation, while $S^2_{(i)}$ indicates the MSE of that model. Mostly, the i-th observation may be detected as an influential point if the absolute value of $DFFITS_i$ exceeds $2\sqrt{p/n}$. Accordingly, for the given problem, the cutoff becomes 0.3312.

- **COVRATIO:**

$$COVRATIO_i = \left(\frac{S^2_{(i)}}{MS_{Res}}\right)^p \left(\frac{1}{1-h_{ii}}\right) \tag{1.3}$$

In most of the situations, an observation is termed as influential if $|COVRATIO - 1| > 2p/n$. That is, in our case, a value of $COVRATIO$ outside the interval $[0.9177, 1.0823]$ might be enough for a point to be influential.

- **DFBETAS:**

$$DFFITS_{j,i} = \frac{\hat{\beta}_j - \hat{\beta}_{(i)j}}{\sqrt{C_{jj}S^2_{(i)}}}, \tag{1.4}$$

where $C = (X^TX)^{-1}$, and $\hat{\boldsymbol{\beta}}_{(i)}$ is the estimate of the vector of regression coefficients computed by excluding the i-th observation. A cutoff of $2/\sqrt{n}$ for $|DFBETAS_{j,i}|$ generally serves our purpose, and the i-th observation may be highly influential if the value exceeds the cutoff. In view of the value of n in our case, the cutoff turns out to be 0.0999.

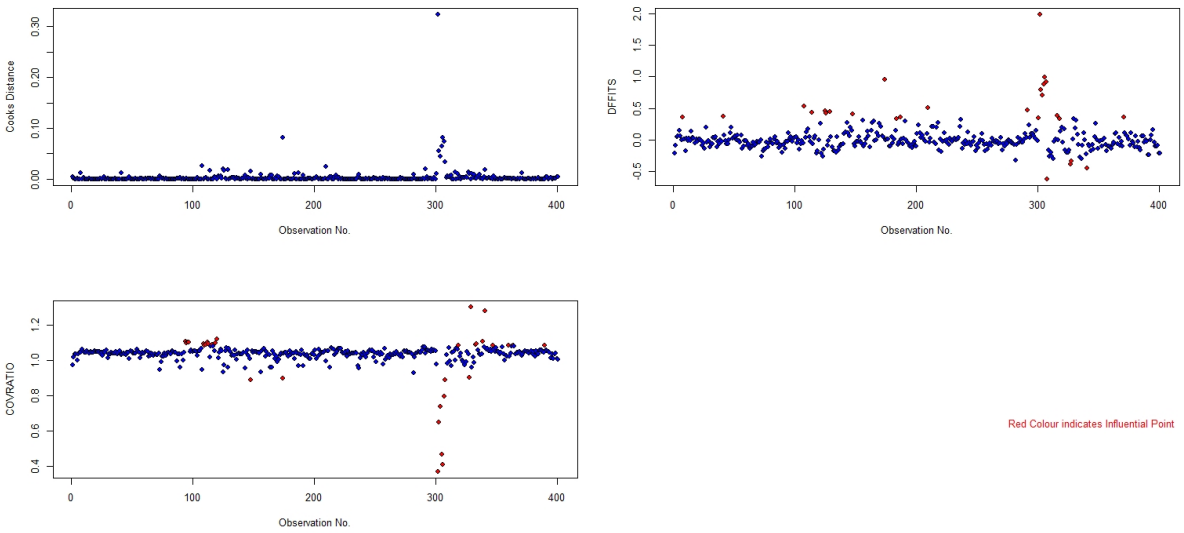


Figure 1.2: Plots of Cook's Distance, DFFITS and COVRATIO, with influential points being suspected using usual cutoffs

In this regard, we would take note of a certain fact about DFBETAS. It can be observed that there are 11 DFBETAS values corresponding to each observation. For our consideration, if at least 60% of those exceed the aforesaid cutoff for DFBETAS, the observation is suspected to be influential. Further, for an overall conclusion, we decide a criterion that a point would be discarded as influential if at least three of the measures suspect it to be so. It must be noted that there may be other plausible criteria. However, we have chosen our benchmark because of its simplicity.

Analyzing the given data yields no suspected influential points using Cook's Distance and 6 using DFBETAS. On the other hand, DFFITS and COVRATIO suspect 27 and 32 points to be influential. Certainly, these figures are quite large. Although only 5 points are declared as influential in overall, the large number of suspicions through DFFITS and COVRATIO warrant to be checked. Besides, through the plots (see Figure 1.2) of Cook's Distance, DFFITS and COVRATIO, we find that that the cutoffs for DFFITS and COVRATIO deserve leniency. In view of this, and also taking help from the plots, we modify the cutoffs as mentioned below, followed by the plots of these measures (Figure 1.3).

- Cook's Distance: 1 (unchanged)
- DFFITS: 0.8312 (increased by 0.5)
- COVRATIO: [0.8177, 1.1823] (higher and lower cutoffs are respectively increased and decreased by 0.1)
- DFBETAS: 0.0999 (unchanged)

Resultantly, the Cook's Distance, DFFITS, COVRATIO and DFBETAS suspect 0, 5, 8 and 6 observations respectively as influential. Observably, these figures are corroborative. In the totality, we would discard 4 observations, reducing the number of observations n to 397. Looking into the RMSE, we observe that it has further decreased to **4.5846**. Accordingly, we can wrap up the analysis of influential points.

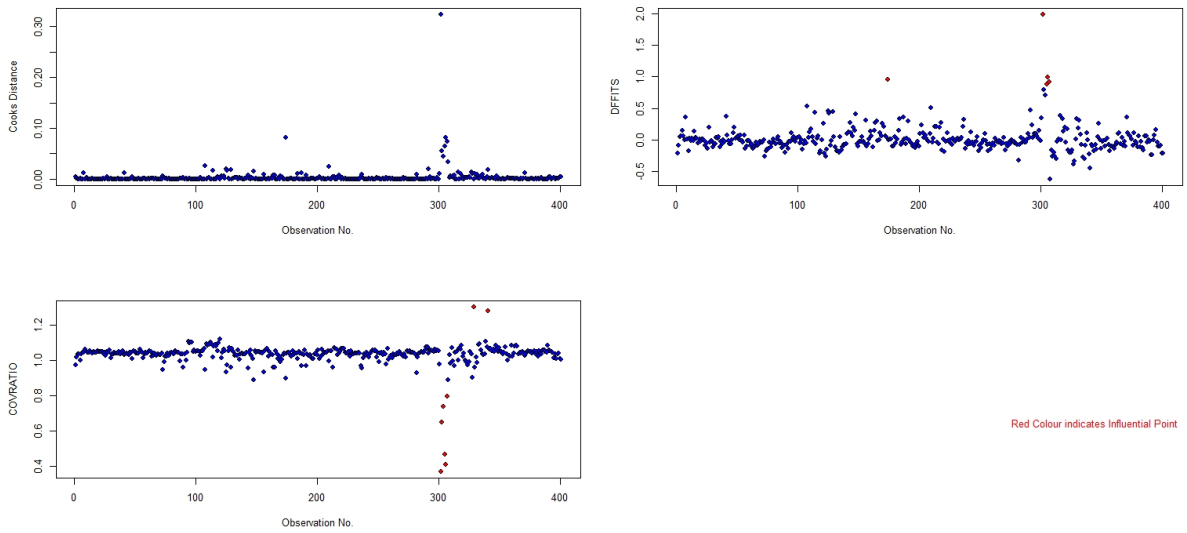


Figure 1.3: Plots of Cook's Distance, DFFITS and COVRATIO classifying the observations as influential or not using the modified cutoffs

2 Dealing with Non-Linearity

After dropping the unusual observations in the previous section, we now proceed to analyze the linearity of relationship between the response and the regressors. In this regard, we first plot the residuals against the individual regressors (Figure 2.1).

It can be observed that there is a U-shaped pattern in the residual plot corresponding to the regressor **RM**, which roughly indicates a possible presence of non-linearity. A slightly decreasing pattern is visible in the residual plot against **DIS** as well. The remaining plots seem to be devoid of any pattern.

To substantiate the presumptions based on residual plots, we take Component plus Residual (CPR) and Augmented Partial Residual (APR) plots into consideration. At this juncture, let us briefly revisit the theoretical aspects of these plots. In CPR plot, we plot $(\boldsymbol{e}_{\boldsymbol{y}|X} + \hat{\beta}_j \boldsymbol{x}_j)$, the portion of response explained linearly by the regressor $\boldsymbol{x}_j$ only, against $\boldsymbol{x}_j$. The APR plot, on the other hand, depicts the portion of response explained by RM only through a quadratic function. To clarify more, the APR diagram plots $(\boldsymbol{e}_{\boldsymbol{y}|X^*} + \hat{\beta}_j \boldsymbol{x}_j + \hat{\gamma}_{jj} \boldsymbol{x}_j^2)$ against RM, where X^* is obtained by augmenting $\boldsymbol{x}_j^2$ as a column to the original design matrix X. In case the response $\boldsymbol{y}$ is linearly related to $\boldsymbol{x}_j$, the CPR and APR plots should not differ significantly.

In the present scenario (see Figure 2.2), there appears to be a marked difference between the two plots for RM and a marginal differnce for LSTAT. For each of the other predectors, including DIS, the APR and CPR plots look fairly similar. As such, we may assume that non-linearity, if any, is caused due to LSTAT and RM only.

To get an overview of the nature of non-linear relationship, we focus on the partial residual plots corresponding to RM and LSTAT in Figure 2.3. In this case, we at first obtain the partial residual vector $(\boldsymbol{e}_{\boldsymbol{y}|X_{-j}})$ by regressing the response on all the predictors except $\boldsymbol{x}_j$. Next, we regress $\boldsymbol{x}_j$ on the remaining predictors to obtain another set of residuals given by $\boldsymbol{e}_{\boldsymbol{x}_j|X_{-j}}$. The partial residual diagram plots $\boldsymbol{e}_{\boldsymbol{y}|X_{-j}}$ against $\boldsymbol{e}_{\boldsymbol{x}_j|X_{-j}}$ to indicate the actual nature of relationship between $\boldsymbol{y}$ and $\boldsymbol{x}_j$. This is so because the two sets of residuals ultimately filter out the effects of other regressors from both $\boldsymbol{y}$ and $\boldsymbol{x}_j$.

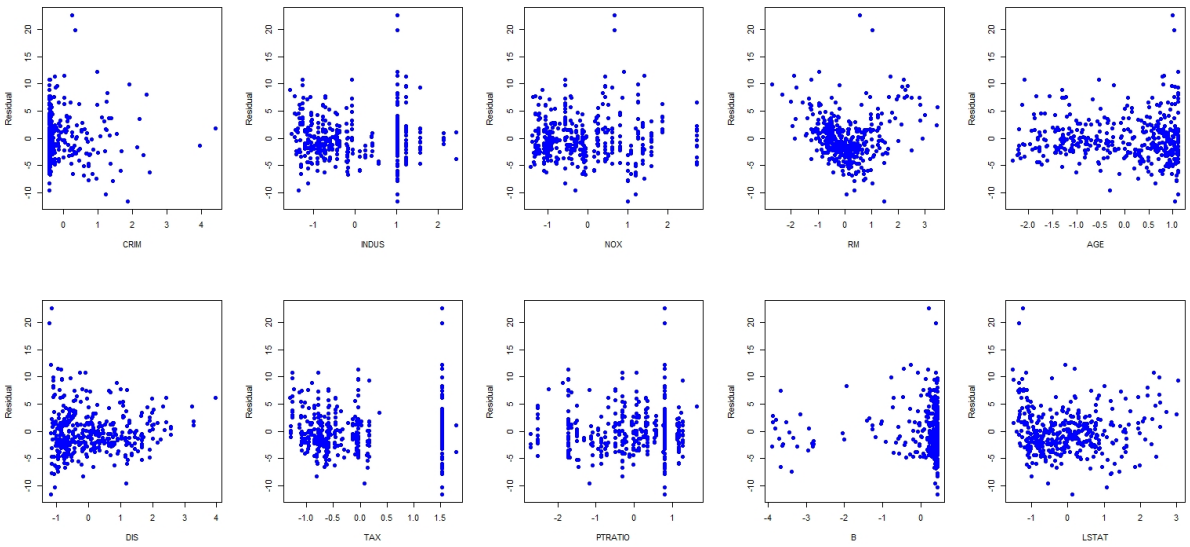


Figure 2.1: Residual Plots (against regressors)

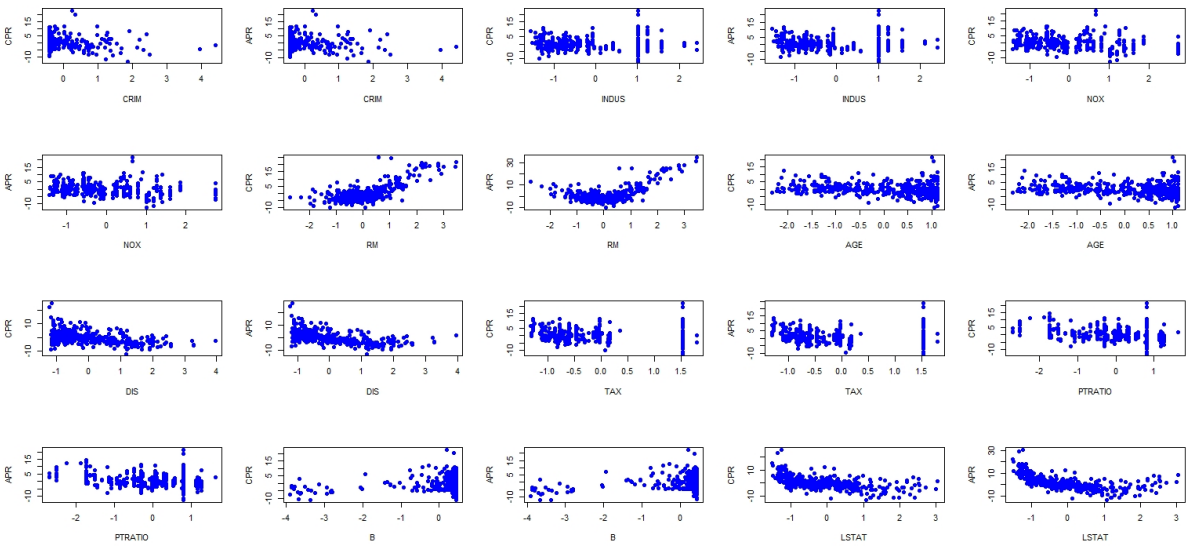
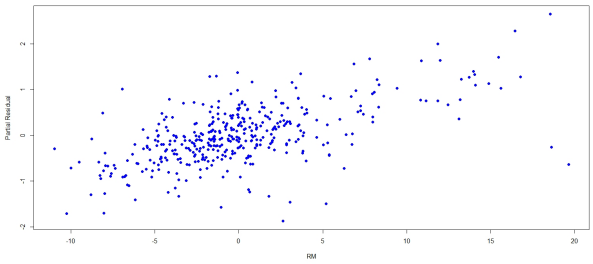
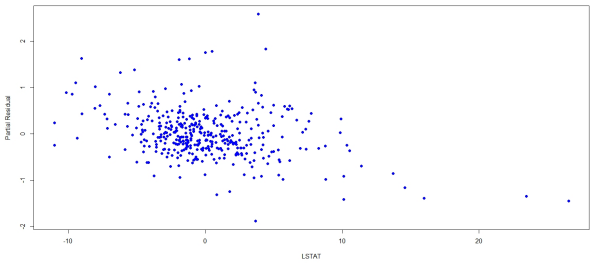


Figure 2.2: APR and CPR Plots for each regressor



(a) For RM



(b) For LSTAT

Figure 2.3: Partial Residual Plots for RM and LSTAT

Once we are done wth the plotting, we observe that the non-linearity is not quite evident between RM and MEDV. Therefore, we do not deem fit to transform RM. The regressor LSTAT, on the other hand, seems to affect MEDV through a decreasing non-linear function which somewhat mimics an exponential function of the type b^x. We accordingly run a model of the nature

$$y_i = \gamma^{x_i}\delta_i,\ i = 1, 2, ..., n,$$

through the data, taking $\boldsymbol{y}$ as MEDV, $\boldsymbol{x}$ as LSTAT and δ_i to be a multiplicative error term. Taking logarithm on both sides, this model can be reconstructed as a linear regression model of log(MEDV) on LSTAT as follows, presuming that the model satisfies the usual assumptions.

$$\boldsymbol{y}^* = \gamma^*\boldsymbol{x} + \boldsymbol{\delta^*}$$

We obtain the OLS estimate of $\hat{\gamma}^*$ as -0.517, and accordingly, $\hat{\gamma} = \exp\hat{\gamma}^* = 0.596$. Resultantly, LSTAT gets transformed into log (LSTAT).

Let us check whether this transformation at all improves the linearity. Before transforming, correlation coefficient between MEDV and LSTAT was -0.756, while upon transformation, value turns out to be 0.816. Thus, there is a slight increase in the magnitude of correlation coefficient, which suggests that the extent of linearity improves marginally (but certainly not significantly) after transformation.

Further, let us compute the RMSE to check whether the transformation results in an overall decrease in RMSE. The value of RMSE is obtained as 13.5845, which is almost thrice the RMSE obtained before the transformation. Hence we refrain from considering the transformed variable in the Model (0.1) and retain the regressor LSTAT in its original form.

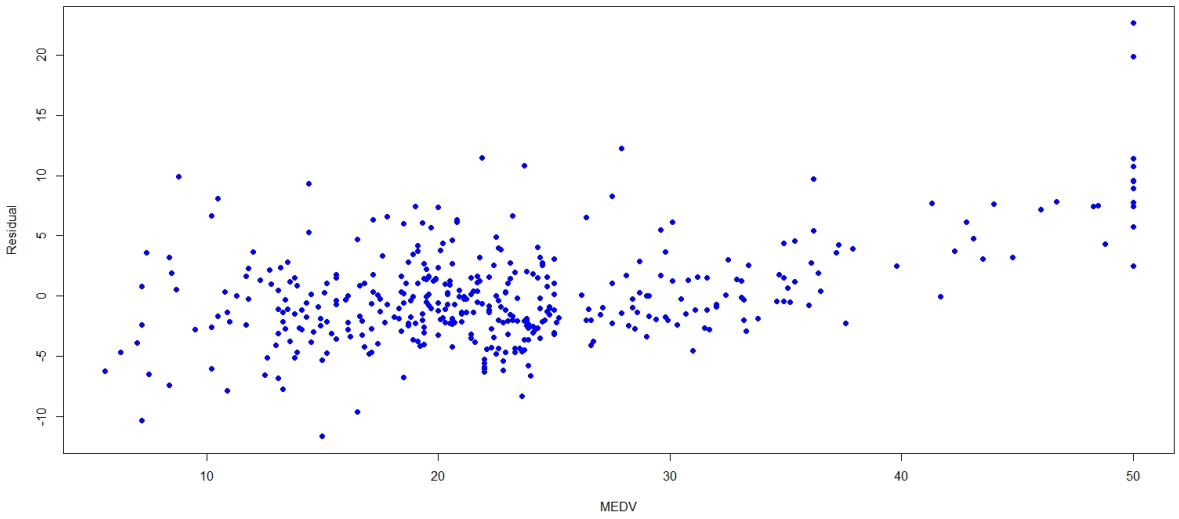


Figure 3.1: Residual Plots (against fitted values)

3 Dealing with Heteroscedasticity

The focus now shifts to the assumption of homoscedasticity of errors. We begin by plotting the residuals against the fitted values of the responses (Figure 3.1), as well as against the invidual regressors (see Figure 2.1 in the previous section).

In the residual plot against fitted values, we find a nearly linear pattern. Besides, in the previous section, we have already mentioned that patterns are visible in residual plots against RM, DIS and LSTAT. Suffice to say, heteroscedasticity may have creeped into the data. To look deeper into this aspect, let us try to identify the regressors with respect to which the the variance (approximated by the square of residual e_i^2) is not constant. On plotting e_i^2 against the regressors (Figure 3.2), we observe that the estimated variances vary with respect to CRIM, AGE, DIS and B. Accordingly, there predictors ar likely to cause heteroscedasticity.

To affirm our contentions, we carry out the Breusch-Pagan test to check the validity of the null hypothesis of homoscedasticity. At first, we compute $d_i^* = \frac{ne_i^2}{\sum e_i^2}$. Next we find the appropriate variables $z_1, z_2, ..., z_k$ (which are nothing but some of the regressors, or their suitable functions) for which the following relation approximately holds,

$$d_i^* = \alpha_0 + \alpha_1 z_{i1} + ... + \alpha_k z_{ik} + \epsilon_i^*, \tag{3.1}$$

with ϵ_i^*'s being i.i.d. standard normal variates. Here, the appropriate z_j's are CRIM, AGE, DIS and B.

The test statistic is accordingly computed using

$$Q = nR_z^2 = \frac{cov^2(d_i^*, \hat{d}_i^*)}{var(d_i^*)var(\hat{d}_i^*)}, \tag{3.2}$$

which is distributed as χ_k^2 under the null hypothesis. The null hypothesis of homoscedasticity is rejected at level α if the observed value of Q exceeds $\chi_{\alpha,k}^2$. In our case, we obtain $Q_{obs} = 17.83$, which becomes large enough to reject the null hypothesis.

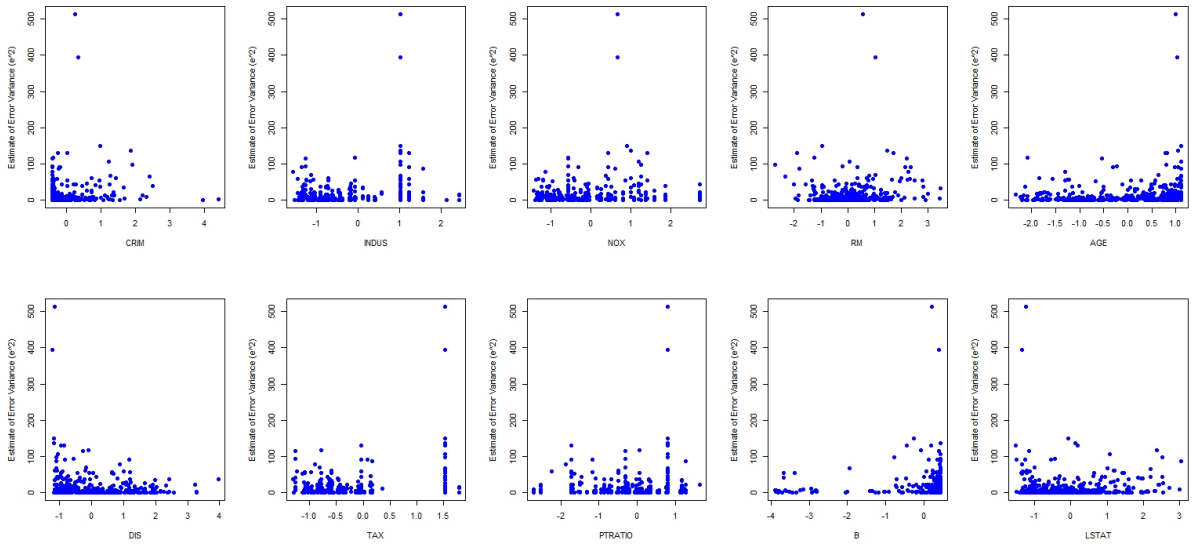


Figure 3.2: Plotting residual squares against regressors to identify the regressors causing heteroscedasticity

Our next task is to estimate the regression coefficient $\boldsymbol{\beta}$ in the presence of heteroscedasticity. Towards that, we follow the algorithm provided to us. We consider $h_i(Z, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to be a linear function of the nature $h_i(Z, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \alpha_0 + \sum_{j=1}^{k} \alpha_i z_{ij}$.

Once the algorithm converges, we compute the RMSE based on the new estimate of $\boldsymbol{\beta}$ and obtain the value to be **4.8866**, which is slightly larger than the one obtained prior to this section (the previous value of 4.5846). Thus the nascent estimate of regression coefficient, although takes into account the heteroscedasticity, marginally compromises with the prediction performance.

4 Dealing with Non-Normality

The final aspect which shall be dealt in the present situation is the validity of the normality assumption of the error (and subsequently the response) in Model (0.1). A graphical idea of such may be fathomed through a QQ-plot (quantile-quantile plot), which plots sample quantiles against population quantiles. If the underlying distribution is symmetric, we can expect the points in the plot to lie on a straight line passing through the origin with unit slope. For our purpose, we would consider the sample quantiles of the R-student residuals with respect to the corresponding population quantiles in the QQ-plot. Mathematically, R-student residuals are defined as

$$r_i^* = \frac{e_i}{\sqrt{S_{(i)}^2(1 - h_{ii})}}, \tag{4.1}$$

which, under usual regression assumptions, follows a t-distribution with $(n - p)$ degrees of freedom. Therefore, in the QQ-plot, we consider the population quantiles of a $t_{(n-p)}$ distribution on the horizontal axis.

The resulting QQ-plot is shown in Figure 4.1. We observe that the points largely deviate from the desired straight line, thereby implying a departure of errors from normality. More precisely, the points form a curve which is concave upward, indicating a postive skew in the underlying distribution of the

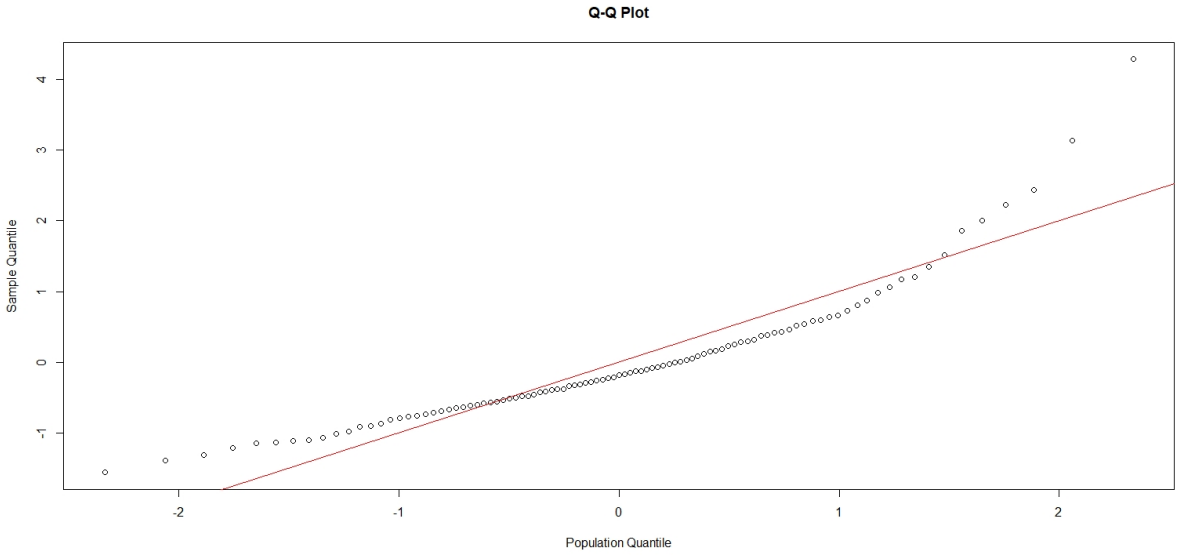


Figure 4.1: QQ-plot based on R-student residuals of original responses

R-student residuals. To normalize the data, we undertake Box-Cox transformation, which is given by

$$u_\lambda = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \log y, & \text{if } \lambda = 0 \end{cases} \tag{4.2}$$

The parameter λ is estimated using the method of profile likelihood. In this method, we first fix a value λ_0 of λ, and estimate the nuissance parameters $\boldsymbol{\beta}$ and σ^2 by maximizing the likelihood function $L(\lambda_0, \boldsymbol{\beta}, \sigma^2)$. Doing so provides the estimates as $\hat{\boldsymbol{\beta}}_{\text{MLE}} = (X^T X)^{-1}) X^T \boldsymbol{u}_\lambda$ and $\hat{\sigma}^2_{\text{MLE}} = \frac{RSS_\lambda}{n}$, where $RSS_\lambda = ||\boldsymbol{u}_\lambda - X\hat{\boldsymbol{\beta}}_{\text{MLE}}||^2$. Subsequently, the estimate of λ is estimated by maximizing the function

$$g(\lambda) = L(\lambda, \hat{\boldsymbol{\beta}}_{\text{MLE}}, \hat{\sigma}^2_{\text{MLE}}) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log\frac{RSS_\lambda}{n} - \frac{n}{2} + n(\lambda - 1)\log G, \tag{4.3}$$

G being the geometric mean of the responses $y_1, y_2, ..., y_n$. Naturally, it is cumbersome to maximize this function analytically, and as such we employ graphical method.

From the graph of $g(\lambda)$ versus λ in Figure 4.2, we observe that the maxima occurs at $\lambda = 0.2$. Therefore, undertake the Box-Cox transformation by putting $\lambda = 0.2$ in (4.2). To check the validity of normality assumption on the transformed response, we draw the QQ-plot of R-Student residuals of transformed variables. From the plot (Figure **??**), we can conclude that the extent of normality in the data has improved. However, normality assumption cannot be said to be perfectly true. It is because the actual distribution seems to have a slightly thicker tail than the desired distribution, as evident from the plot.

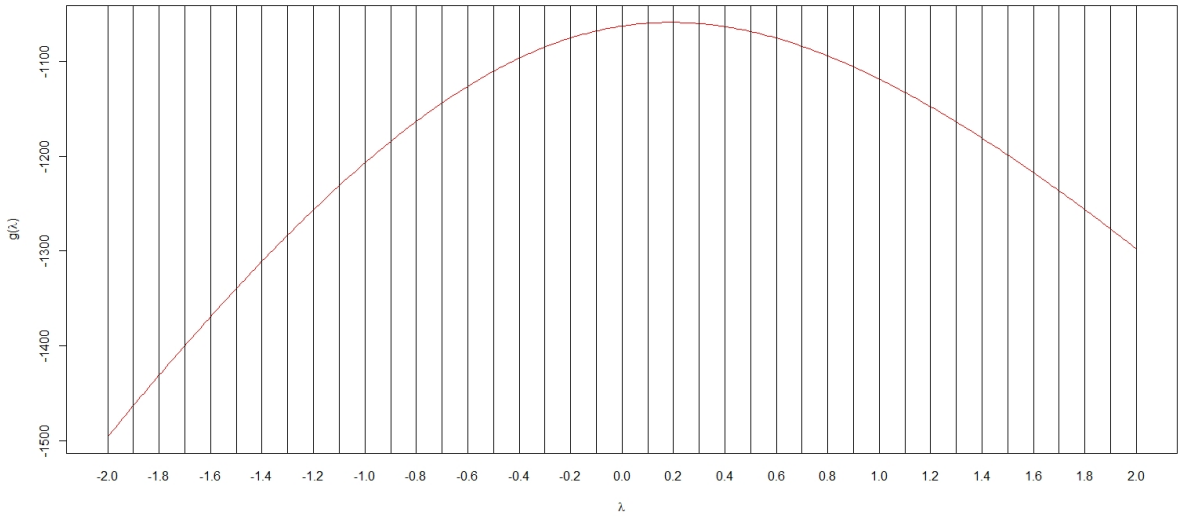


Figure 4.2: Maximizing $g(\lambda)$ graphically w.r.t. λ

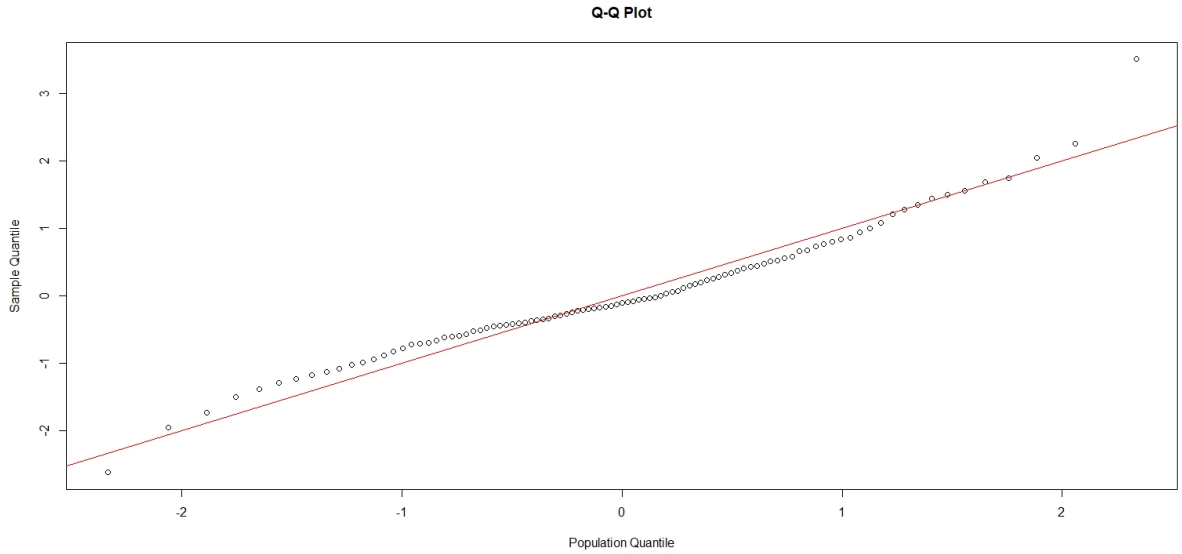


Figure 4.3: QQ-plot based on R-student residuals of transformed responses

```
################################### Loading and Overviewing Data
###################################

data = read.table("Boston_Housing.txt",header=T)
attach(data)
y = MEDV
n = length(y)
X = cbind(CRIM,INDUS,NOX,RM,AGE,DIS,TAX,PTRATIO,B,LSTAT)
X = cbind(rep(1,n),X)
colnames(X)[1] = 'INTERCEPT'
p = ncol(X)
detach(data)

for(i in 2:p)     #Standardizing regressors
      X[,i] = (X[,i]-mean(X[,i])) / sd(X[,i])



################################### Dividing into Training & Test Data
###################################

set.seed(153)
n.test = 100
n.train = n - n.test
ind.test = sort(sample(1:n,n.test,replace=FALSE))
y.test = y[ind.test]
X.test = X[ind.test,]
y.train = y[-ind.test]
X.train = X[-ind.test,]



################################### Implementing OLS
###################################

OLS = function(X,y)
{
      beta.hat = solve(crossprod(X)) %*% crossprod(X,y)
      y.hat = X %*% beta.hat
      err = y - y.hat
      RSS = sum(err^2)
      MSRes = RSS / (nrow(X)-ncol(X))
      r = {}
      r$beta.hat = beta.hat
      r$y.hat = y.hat
      r$err = err
      r$RSS = RSS
      r$MSRes = MSRes
      r
}

################################### Implementing Box-Cox Transformation
###################################

BoxCox = function(x,L)
{
      if(L!=0) u=(x^L-1)/L else u=log(x)
      u
}
```

```
##################################### Computing RMSE
#####################################

OLS.train = OLS(X.train,y.train)
beta.hat = OLS.train$beta.hat
y.test.hat = X.test %*% beta.hat
RMSE = sqrt(sum((y.test-y.test.hat)^2) / n.test)
message("RMSE before dropping observations: ",round(RMSE,4))



##################################### Dealing with Leverage & Influential Points
#####################################

# Leverage Points

#.............................................................#
leverage.cutoff = 2*p / n.train + 2*0.02

# Note: Cutoff is subject to modification as per requirement!
#.............................................................#

H = X.train %*% solve(crossprod(X.train)) %*% t(X.train)
plot(1:n.train,diag(H),pch=16,col='blue', xlab='Observation No.',ylab='Leverage
Value')
leverage.ind = which(diag(H)>leverage.cutoff)
message("No. of Leverage Points (at Cutoff ",round(leverage.cutoff,4),"):
",length(leverage.ind))
X.train = X.train[-leverage.ind,]
y.train = y.train[-leverage.ind]
n.train = nrow(X.train)
OLS.train = OLS(X.train,y.train)
beta.hat = OLS.train$beta.hat
y.test.hat = X.test %*% beta.hat
RMSE = sqrt(sum((y.test-y.test.hat)^2) / n.test)
message("RMSE after dropping leverage points: ",round(RMSE,4))


# Influential Points (after dropping leverage points)

#.............................................................#
CooksD.cutoff = 1
DFFITS.cutoff = 2 * sqrt(p/n.train) + 2*0.25
COVRATIO.cutoff = 3*p/n.train + 1*0.1
DFBETAS.cutoff = 2 / sqrt(n.train)

# Note: Cutoffs are subject to modifications as per requirement!
#.............................................................#

OLS.train = OLS(X.train,y.train)
beta.hat = OLS.train $ beta.hat
MSRes = OLS.train $ MSRes
y.train.hat = OLS.train $ y.hat
C = solve(crossprod(X.train))
H = X.train %*% C %*% t(X.train)

CooksD = 0
```

```
DFFITS = 0
COVRATIO = 0
DFBETAS = matrix(0,n.train,p)
for(i in 1:n.train)
{
      OLS.i = OLS(X.train[-i,],y.train[-i])
      beta.hat.i = OLS.i $ beta.hat
      y.i.hat = X.train %*% beta.hat.i
      Si2 = OLS.i $ MSRes
      CooksD[i] = sum((y.i.hat - y.train.hat)^2) / (p*MSRes)
      DFFITS[i] = (y.train.hat[i] - y.i.hat[i]) / sqrt(H[i,i] * Si2)
      COVRATIO[i] = (Si2/MSRes)^p / (1-H[i,i])
      for(j in 1:p)
            DFBETAS[i,j] = (beta.hat[j]-beta.hat.i[j]) / sqrt(Si2*C[j,j])
}

Inf.Analysis_CooksD = data.frame(i=1:n.train,im=CooksD,d=rep(0,n.train))
Inf.Analysis_DFFITS = data.frame(i=1:n.train,im=DFFITS,d=rep(0,n.train))
Inf.Analysis_COVRATIO = data.frame(i=1:n.train,im=COVRATIO,d=rep(0,n.train))
Inf.Analysis_DFBETAS = cbind(1:n.train, matrix(0,n.train,p+1))
for(i in 1:n.train)
{
      count.1 = 0
      for(j in 1:p)
      {
            if (abs(DFBETAS[i,j]) > DFBETAS.cutoff)
            {
                  Inf.Analysis_DFBETAS[i,j+1] = 1
                  count.1 = count.1 + 1
            }
      }
      if(count.1>0.6*p) Inf.Analysis_DFBETAS[i,p+2] = 1
}

Inf.Analysis_CooksD $ d[which(CooksD>CooksD.cutoff)] = 1
message("No. of Points Suspected to be Influential by Cooks Distance (at Cutoff
",round(CooksD.cutoff,4),"): ",length(which(Inf.Analysis_CooksD$d == 1)))
Inf.Analysis_DFFITS $ d[which(abs(DFFITS)>DFFITS.cutoff)] = 1
message("No. of Points Suspected to be Influential by DFFITS (at Cutoff
",round(DFFITS.cutoff,4),"): ",length(which(Inf.Analysis_DFFITS$d == 1)))
Inf.Analysis_COVRATIO $ d[which(abs(COVRATIO-1)>COVRATIO.cutoff)] = 1
message("No. of Points Suspected to be Influential by COVRATIO (at Cutoffs
",round(1-COVRATIO.cutoff,4)," and ",round(1+COVRATIO.cutoff,4),"):
",length(which(Inf.Analysis_COVRATIO$d == 1)))
message("No. of Points Suspected to be Influential by DFBETAS (at Cutoff
",round(DFBETAS.cutoff,4),"): ",length(which(Inf.Analysis_DFBETAS[,p+2] == 1)))

par(mfrow=c(2,2))
      plot(Inf.Analysis_CooksD$i,Inf.Analysis_CooksD$im, pch=21, bg=c('blue')
[unclass(as.factor(Inf.Analysis_CooksD$d))], xlab='Observation No.',ylab='Cooks
Distance', xlim=c(1,n.train))
      plot(Inf.Analysis_DFFITS$i,Inf.Analysis_DFFITS$im, pch=21, bg=c('blue','red')
[unclass(as.factor(Inf.Analysis_DFFITS$d))], xlab='Observation No.',ylab='DFFITS',
xlim=c(1,n.train))
      plot(Inf.Analysis_COVRATIO$i,Inf.Analysis_COVRATIO$im, pch=21,
bg=c('blue','red')[unclass(as.factor(Inf.Analysis_COVRATIO$d))], xlab='Observation
No.',ylab='COVRATIO', xlim=c(1,n.train))
mtext("Red Colour indicates Influential Point          ",
      side=1,line=-10,adj=1,outer=TRUE,col='red',cex=0.8)
```

```
Inf.Analysis =
cbind(1:n.train,Inf.Analysis_CooksD$d,Inf.Analysis_DFFITS$d,Inf.Analysis_COVRATIO$d
,Inf.Analysis_DFBETAS[,p+2])
count.inf = array(0,n.train)
for(i in 1:n.train)
{
      for(j in 2:5)
      {
            if(Inf.Analysis[i,j]==1)
                  count.inf[i] = count.inf[i] + 1
      }
}
influential.ind = which(count.inf>=3 & Inf.Analysis[,4]==1)
X.train.temp = X.train
y.train.temp = y.train
if(length(influential.ind)!=0)
{
      X.train.temp = X.train[-influential.ind,]
      y.train.temp = y.train[-influential.ind]
      n.train.temp = nrow(X.train.temp)
}
message("No. of Influential Points: ",length(influential.ind))

OLS.temp = OLS(X.train.temp,y.train.temp)
beta.hat = OLS.temp$beta.hat
y.test.hat = X.test %*% beta.hat
RMSE.temp = sqrt(sum((y.test-y.test.hat)^2) / n.test)
message("RMSE after dropping influential observations: ",round(RMSE.temp,4))

X.train = X.train.temp
y.train = y.train.temp
n.train = nrow(X.train)
RMSE = RMSE.temp



##################################### Dealing with Curvature
#####################################

OLS.train = OLS(X.train,y.train)
beta.hat = OLS.train $ beta.hat
err = OLS.train $ err

par(mfrow=c(2,5)) #Plotting Residuals vs Regressors
for(j in 2:p)
      plot(X.train[,j],err,pch=16,col='blue', xlab=colnames(X.train)
[j],ylab='Residual')

par(mfrow=c(4,5)) #APR and CPR Plots  ### TOO MANY PLOTS IN A PAGE
for(j in 2:p)
{
      CPR.j = err + beta.hat[j]*X.train[,j]
      X.star = cbind(X.train,X.train[,j]^2)
      OLS.star = OLS(X.star,y.train)
      beta.star = OLS.star $ beta.hat
      err.star = OLS.star $ err
      APR.j = err + beta.star[j]*X.star[,j] +
beta.star[length(beta.star)]*X.star[,ncol(X.star)]
```

```
      plot(X.train[,j],CPR.j,pch=16,col='blue', xlab=colnames(X.train)
[j],ylab='CPR')
      plot(X.train[,j],APR.j,pch=16,col='blue', xlab=colnames(X.train)
[j],ylab='APR')
}


# Non-linearity suspected for RM and LSTAT

# Removing Non-Linearity

# For RM
reg.RM = which(colnames(X.train)=='RM')  #identifying the regressor which seems to
be non-linearly to E(y)
X_RM = X.train[,-reg.RM]
x.RM = X.train[,reg.RM]
err.y_RM = OLS(X_RM,y.train) $ err
err.RM.X_RM = OLS(X_RM,x.RM) $ err
par(mfrow=c(1,1))
plot(err.y_RM,err.RM.X_RM,pch=16,col='blue', xlab=colnames(X.train)
[reg.RM],ylab='Partial Residual')

# For LSTAT
reg.LSTAT = which(colnames(X.train)=='LSTAT')
X_LSTAT = X.train[,-reg.LSTAT]
x.LSTAT = X.train[,reg.LSTAT]
err.y_LSTAT = OLS(X_LSTAT,y.train) $ err
err.LSTAT.X_LSTAT = OLS(X_LSTAT,x.LSTAT) $ err
par(mfrow=c(1,1))
plot(err.y_LSTAT,err.LSTAT.X_LSTAT,pch=16,col='blue', xlab=colnames(X.train)
[reg.LSTAT],ylab='Partial Residual')

OLS.LSTAT = OLS(x.LSTAT,log(y.train))    #Transformation used: y = b^x
parameter.nonlinear.LSTAT = as.numeric(exp(OLS.LSTAT $ beta.hat))
LSTAT.new = parameter.nonlinear.LSTAT ^ x.LSTAT
cor.old.LSTAT = cor(y.train,x.LSTAT)
cor.new.LSTAT = cor(y.train,LSTAT.new)
message("Correlation:\tAfter transformation: ",cor.new.LSTAT,"\tBefore
transformation: ",cor.old.LSTAT)


X.new = X.train
X.new[,reg.LSTAT] = LSTAT.new
beta.hat.new = OLS(X.new,y.train) $ beta.hat
y.test.hat.new = X.test %*% beta.hat.new
RMSE.new = sqrt(sum((y.test-y.test.hat.new)^2) / n.test)
message("RMSE:\tBefore transformation: ",round(RMSE,4),"\tAfter transformation:
",round(RMSE.new,4))

if(RMSE.new<=RMSE)
{
      X.train = X.new
      RMSE = RMSE.new
}



##################################### Dealing with Heteroscedasticity
#####################################
```

```
OLS.train = OLS(X.train,y.train)
beta.hat = OLS.train $ beta.hat
err = OLS.train $ err
err2 = err * err
plot(y.train,err,pch=16,col='blue', xlab='MEDV',ylab='Residual')
par(mfrow=c(2,5))
for(j in 2:p)
      plot(X.train[,j],err,pch=16,col='blue', xlab=colnames(X.train)
[j],ylab='Residual')
par(mfrow=c(2,5))
for(j in 2:p)
      plot(X.train[,j],err2,pch=16,col='blue', xlab=colnames(X.train)
[j],ylab='Estimate of Error Variance (e^2)')


# Breusch-Pagan Test

di.star = n.train*err2 / sum(err2)
ind = which(colnames(X.train)=='CRIM' | colnames(X.train)=='AGE' |
            colnames(X.train)=='DIS' | colnames(X.train)=='B')
Z = X.train[,c(1,ind)]  #obtained from the plots of ei^2 vs xj
di.star.hat = OLS(Z,di.star) $ y.hat
Q.BP = n.train * cor(di.star,di.star.hat)^2
message("Value of test statistic for Breusch Pagan Test: ",round(Q.BP,4))
if(Q.BP>qchisq(0.95,ncol(Z))) message("Reject Homoscedasticity Assumption") else
message("Accept Homoscedasticity Assumption")


# Estimating Regression Coefficients in the presence of Heteroscedasticity

beta.hat.0 = OLS(X.train,y.train) $ beta.hat
beta.hat.1 = beta.hat.0
alpha.hat.0 = OLS(Z,log(err2)) $ beta.hat
alpha.hat.1 = alpha.hat.0
Sigma.hat = diag(0,n.train)
while(sum(beta.hat.1-beta.hat.0)^2 <= 0.1 & sum(alpha.hat.1-alpha.hat.0)^2 <= 0.1)
## SHOULD BE THE OPPOSITE : WRONG CRITERION
{
      beta.hat.0 = beta.hat.1
      alpha.hat.0 = alpha.hat.1
      err2.I = (y.train - X.train %*% beta.hat.0)^2
      OLS.I = OLS(Z,log(err2.I))
      alpha.hat.1 = OLS.I $ beta.hat
      Sigma.hat = diag(as.vector(OLS.I $ y.hat))
      beta.hat.1 = solve(t(X.train)%*%solve(Sigma.hat)%*%X.train) %*% t(X.train)%*
%solve(Sigma.hat)%*%y.train
}

y.test.hat.new = X.test %*% beta.hat.1
RMSE.new = sqrt(sum((y.test-y.test.hat.new)^2) / n.test)
message("RMSE:\tBased on New Estimate: ",round(RMSE.new,4),"\tBased on Old
Estimate: ",round(RMSE,4))

if(RMSE>RMSE.new)
{
      RMSE = RMSE.new
      X.train = sqrt(solve(Sigma.hat)) %*% X.train
      y.train = sqrt(solve(Sigma.hat)) %*% y.train
```

```
}


#################################### Dealing with Non-Normality
####################################

# Q-Q Plot (before transformation)

H = X.train %*% solve(crossprod(X.train)) %*% t(X.train)
OLS.train = OLS(X.train,y.train)
err = OLS.train $ err
R.student = array(0)
for(i in 1:n.train)
{
      Si2 = OLS(X.train[-i,],y.train[-i]) $ MSRes
      R.student[i] = err[i] / sqrt(Si2*(1-H[i,i]))
}
par(mfrow=c(1,1))
plot(qt(seq(0.01,0.99,0.01),n.train-p),quantile(R.student,seq(0.01,0.99,0.01)),
      xlab='Population Quantile',ylab='Sample Quantile',main='Q-Q Plot')
abline(a=0,b=1,col='red')


# Box-Cox Transformation & Profile Likelihood Method

lambda.val = seq(-2,2,0.01)
g.lambda = array(0)
for(i in 1:length(lambda.val))
{
      y.lambda = BoxCox(y.train,lambda.val[i])
      OLS.lambda = OLS(X.train,y.lambda)
      beta.hat.lambda = OLS.lambda $ beta.hat
      RSS.lambda = OLS.lambda $ RSS
      sigma2.lambda = RSS.lambda / n.train
      g.lambda[i] = -0.5*n.train*(log(2*pi*sigma2.lambda) + 1) +
n.train*(lambda.val[i]-1)*mean(log(y.train))
}
plot(lambda.val,g.lambda,type='l',col='red',
xaxt='n',xlab=expression(lambda),ylab=expression(paste('g(',lambda,')')))
axis(side=1,at=seq(-2,2,0.1),tck=1)

lambda = 0.2      #obtained from the plot  ## YOU CAN EXACTLY FIND THE MAXIMA


# Q-Q Plot (after transformation)

y.BoxCox = BoxCox(y.train,lambda)
OLS.BoxCox = OLS(X.train,y.BoxCox)
err = OLS.BoxCox $ err
R.student = array(0)
for(i in 1:n.train)
{
      Si2 = OLS(X.train[-i,],y.BoxCox[-i]) $ MSRes
      R.student[i] = err[i] / sqrt(Si2*(1-H[i,i]))
}
plot(qt(seq(0.01,0.99,0.01),n.train-p),quantile(R.student,seq(0.01,0.99,0.01)),
      xlab='Population Quantile',ylab='Sample Quantile',main='Q-Q Plot')
abline(a=0,b=1,col='red')
```

##
####################