Simple Linear Regression On Boston Housing

2022/02/04

Package invocation and data import

```
library(mlbench)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(car)
```

```
## Loading required package: carData
```

```
library(MASS)
library(lars)
```

```
## Loaded lars 1.3
```

```
data("BostonHousing")
```

Introduction

In this project, we use the data set of "BostonHousing," which is a Boston census tracts from the 1970 census composed by Harrison and Rubinfeld. There are 506 observations and 14 variables. Our goal is to find the best model that predict the median value of owner-occupied homes (medv).

Simple Linear Regression

$y = \beta_0 + \beta_1 X + \epsilon$
Simple linear regression is performed to determine the association between two quantitative variables. It can be used to determine:
1. The degree to which two variables are significantly correlated.
2. The dependent variable's value at a certain level of the independent variable.

Variables

crim per capita crime rate by town
zn proportion of residential land zoned for lots over $25,000$ sq.ft
indus proportion of non-retail business acres per town
chas Charles River dummy variable ($= 1$ if tract bounds river; 0 otherwise)
nox nitric oxides concentration (parts per 10 million)
rm average number of rooms per dwelling
age proportion of owner-occupied units built prior to 1940
dis weighted distances to five Boston employment centres
rad index of accessibility to radial highways
tax full-value property-tax rate per USD $10,000$
ptratio pupil-teacher ratio by town
b $1000(B - 0.63)^2$ where B is the proportion of blacks by town
lstat percentage of lower status of the population
medv median value of owner-occupied homes in USD $1000's$

Descriptive Statistics

Mean, Median, Extremum and Quartile

```
data = BostonHousing[,-c(4)] # delete dummy variable "chas", ignore the influence on housing prices of
summary(data)
```

```
##       crim                zn             indus            nox        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.3850  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.4490  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.5380  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.5547  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.6240  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :0.8710  
##        rm             age              dis              rad        
##  Min.   :3.561   Min.   :  2.90   Min.   : 1.130   Min.   : 1.000  
##  1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000  
##  Median :6.208   Median : 77.50   Median : 3.207   Median : 5.000  
##  Mean   :6.285   Mean   : 68.57   Mean   : 3.795   Mean   : 9.549  
##  3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000  
##  Max.   :8.780   Max.   :100.00   Max.   :12.127   Max.   :24.000  
##       tax           ptratio            b               lstat      
##  Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73  
##  1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95  
##  Median :330.0   Median :19.05   Median :391.44   Median :11.36  
##  Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65  
##  3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95  
##  Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97  
##       medv      
##  Min.   : 5.00  
##  1st Qu.:17.02  
##  Median :21.20  
##  Mean   :22.53  
##  3rd Qu.:25.00
```

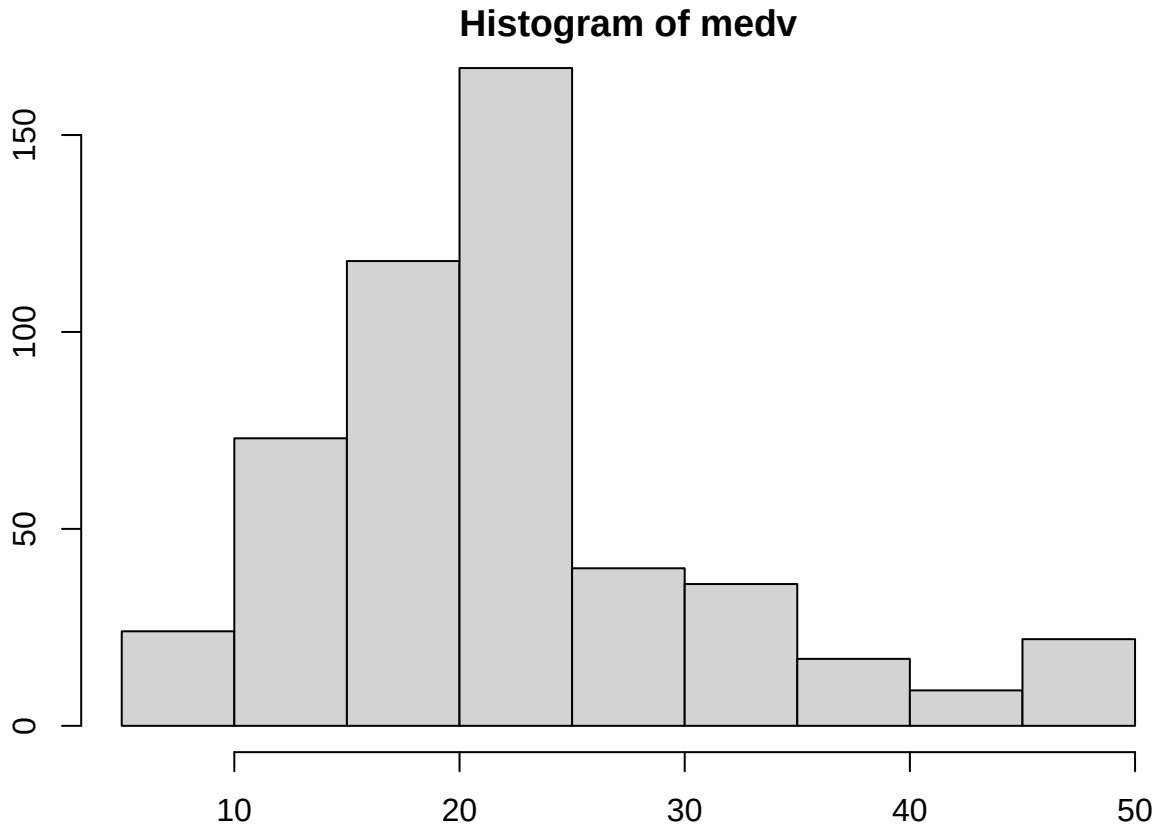
```
##  Max.   :50.00
```

Plot Histograms

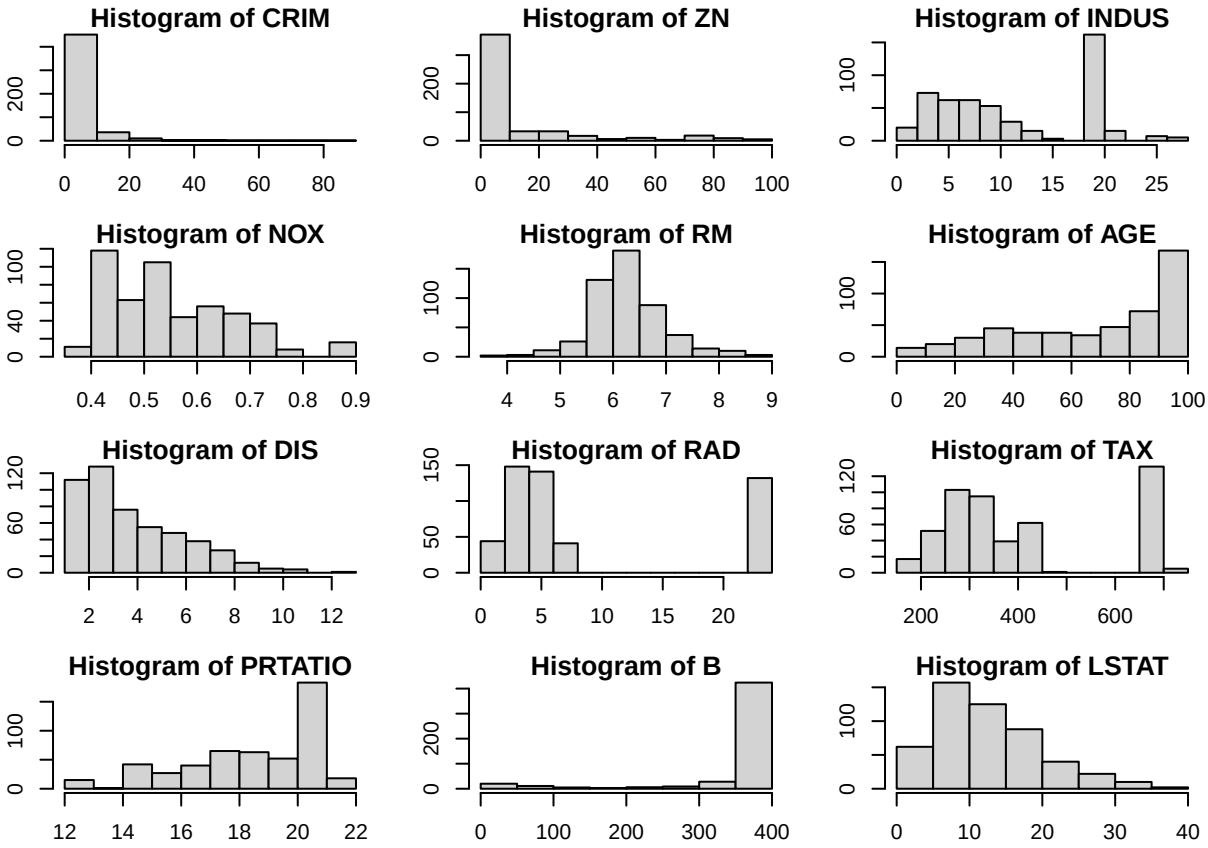
```
x = data[,c(1:12)]
y = data[,c(13)]

title = c("CRIM","ZN","INDUS","NOX","RM","AGE","DIS",
          "RAD","TAX","PRTATIO","B","LSTAT")

par(mfrow=c(1,1))
par(mar = c(3,3,1,1))
hist(y,main='Histogram of medv')
```

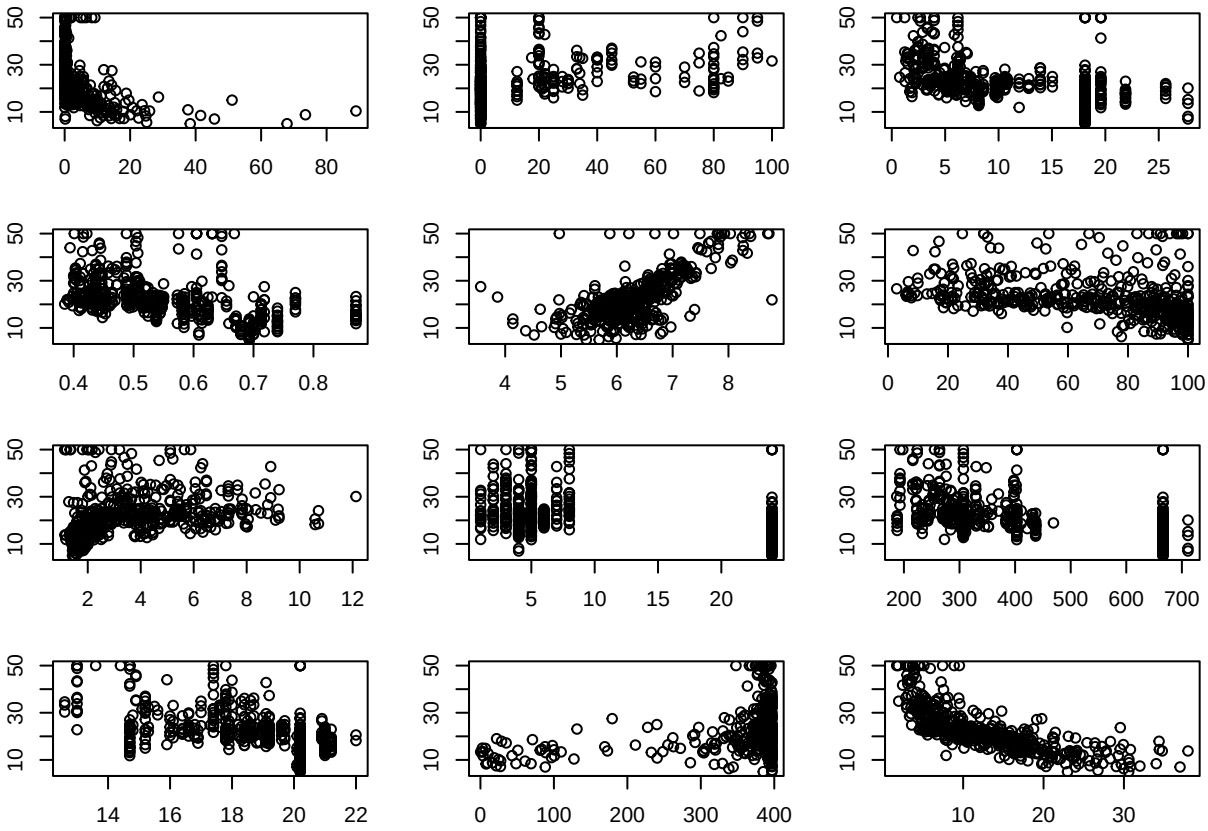



```
par(mfrow=c(4,3))
for(i in 1:12){
  hist(x[,i], xlab = paste0(title[i]),main=paste('Histogram of',title[i]))
}
```



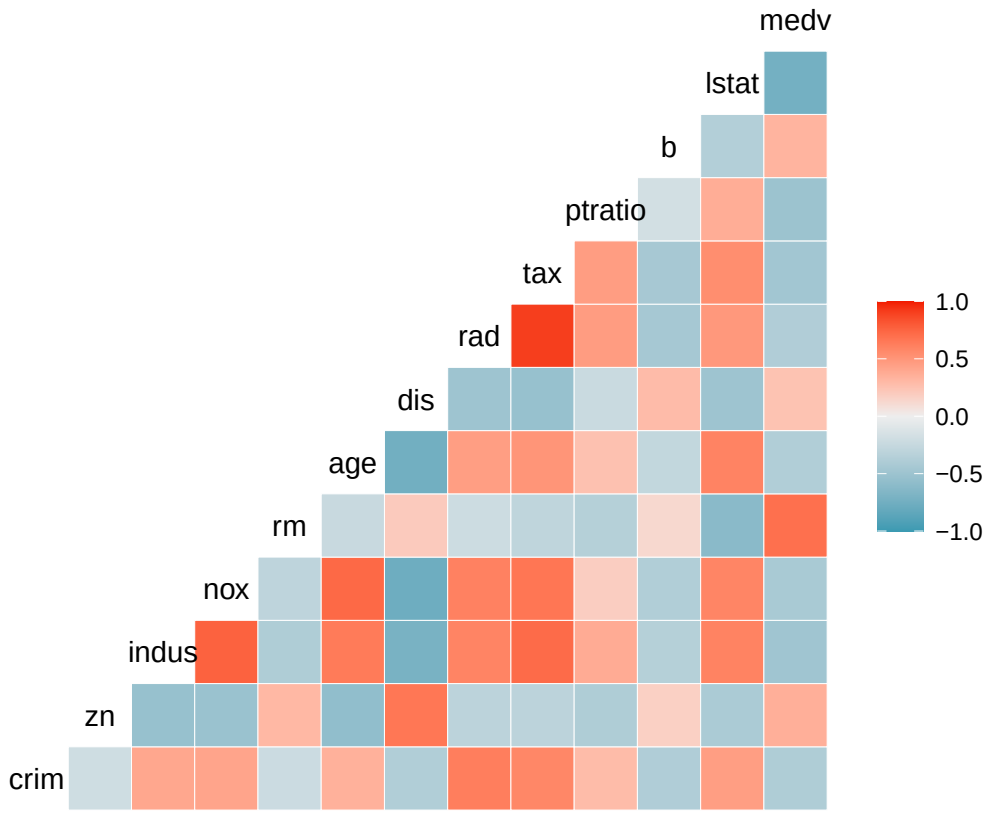
Scatter Plots

```
par(mfrow=c(4,3))
par(mar = c(3,3,1,1))
for(i in 1:12){
  plot(x[,i], y, xlab = paste0(title[i]),ylab = "medv")
}
```



Correlation Heat Map

```
ggcorr(data, method = c("everything", "pearson"))
```



Multivariable linear regression

```
data.1 = data.frame(scale(data))
fit.1 = lm(medv~.-1,data.1)
summary(fit.1)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = data.1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45663 -0.30556 -0.07019  0.20812  2.86780
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## crim    -0.10581    0.03094  -3.420 0.000678 ***
## zn       0.11932    0.03508   3.401 0.000725 ***
## indus    0.03007    0.04598   0.654 0.513462
## nox     -0.21881    0.04847  -4.514 7.96e-06 ***
## rm       0.29416    0.03216   9.147  < 2e-16 ***
## age      0.00852    0.04069   0.209 0.834241
## dis     -0.34008    0.04602  -7.391 6.29e-13 ***
```

```
## rad      0.31083    0.06293   4.939 1.08e-06 ***
## tax     -0.25208    0.06894  -3.657 0.000283 ***
## ptratio -0.23327    0.03090  -7.549 2.13e-13 ***
## b        0.09670    0.02683   3.604 0.000346 ***
## lstat   -0.41474    0.03961 -10.470  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.52 on 494 degrees of freedom
## Multiple R-squared:  0.7355, Adjusted R-squared:  0.7291
## F-statistic: 114.5 on 12 and 494 DF,  p-value: < 2.2e-16
```

Many coefficients of the variables are not significant, so variable selection is needed.

Variable Selection

```
XX<-cor(data.1[,1:12])
kappa(XX, exact=TRUE)
```

```
## [1] 94.77388
```

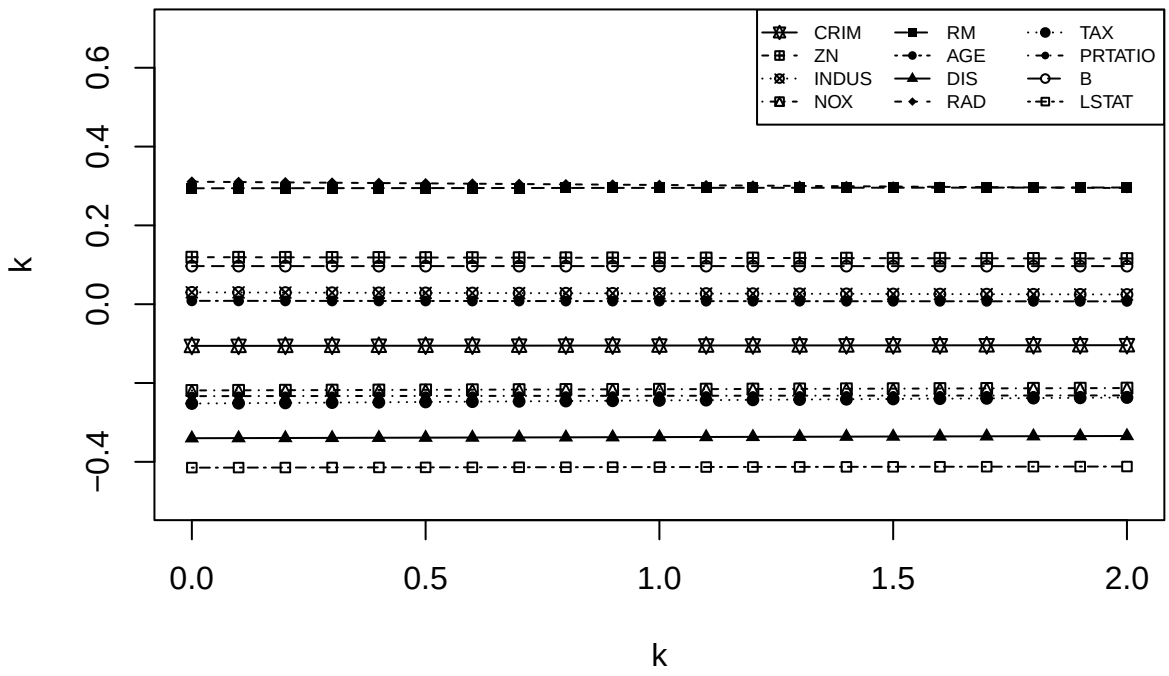
Condition number is smaller than but close to 100, so there is multicollinearity to some extent.

Ridge Regression

Any data that exhibits multicollinearity can be analyzed using the model tuning technique known as ridge regression. This technique carries out $L2$ regularization. Predicted values differ much from real values when the problem of multicollinearity arises, least-squares are unbiased, and variances are significant.

```
fit.ridge = lm.ridge(medv~.-1,data=data.1,lambda=seq(0,2,0.1))
beta = coef(fit.ridge)

k = fit.ridge$lambda
plot(k,k,type='n',ylim=c(-0.5,0.7))
linetype = c(1:12)
char = c(11:22)
for(i in 1:12){
  lines(k,beta[,i],type='o',lty=linetype[i],pch=char[i],cex=0.75)
}
legend('topright',legend=title,cex=0.6,lty=linetype,pch=char,ncol=3)
```



As we can see from the graph, the lines are all constant. The data we generated show that each variable's ridge coefficient is stable, so this is a bad model, and we can't use ridge regression to select variables.

Stepwise regression

Stepwise regression is the iterative process of building a regression model step by step while choosing independent variables to be included in the final model. After each cycle, the possible explanatory factors are successively added or removed, and the statistical significance is tested. Furthermore, there are three approaches to stepwise regression: forward selection, backward elimination, and bidirectional elimination.

```
fit.2 = step(fit.1,direction="both")
```

```
## Start:  AIC=-649.97
## medv ~ (crim + zn + indus + nox + rm + age + dis + rad + tax +
##     ptratio + b + lstat) - 1
##
##           Df Sum of Sq    RSS     AIC
## - age      1    0.0119 133.58 -651.92
## - indus    1    0.1156 133.68 -651.53
## <none>                 133.56 -649.97
## - zn       1    3.1283 136.69 -640.25
## - crim     1    3.1628 136.73 -640.13
## - b        1    3.5109 137.07 -638.84
## - tax      1    3.6151 137.18 -638.46
## - nox      1    5.5093 139.07 -631.52
```

```
## - rad      1    6.5950 140.16 -627.58
## - dis      1   14.7678 148.33 -598.90
## - ptratio  1   15.4088 148.97 -596.72
## - rm       1   22.6195 156.18 -572.81
## - lstat    1   29.6362 163.20 -550.57
##
## Step:  AIC=-651.92
## medv ~ crim + zn + indus + nox + rm + dis + rad + tax + ptratio +
##     b + lstat - 1
##
##           Df Sum of Sq    RSS     AIC
## - indus    1     0.116 133.69 -653.48
## <none>                 133.58 -651.92
## + age      1     0.012 133.56 -649.97
## - zn       1     3.127 136.70 -642.21
## - crim     1     3.162 136.74 -642.08
## - b        1     3.554 137.13 -640.64
## - tax      1     3.607 137.18 -640.44
## - nox      1     5.787 139.36 -632.46
## - rad      1     6.589 140.16 -629.56
## - ptratio  1    15.430 149.01 -598.61
## - dis      1    16.418 149.99 -595.26
## - rm       1    23.876 157.45 -570.71
## - lstat    1    33.035 166.61 -542.10
##
## Step:  AIC=-653.48
## medv ~ crim + zn + nox + rm + dis + rad + tax + ptratio + b +
##     lstat - 1
##
##           Df Sum of Sq    RSS     AIC
## <none>                 133.69 -653.48
## + indus    1     0.116 133.58 -651.92
## + age      1     0.012 133.68 -651.53
## - zn       1     3.034 136.73 -644.13
## - crim     1     3.220 136.91 -643.44
## - b        1     3.521 137.21 -642.33
## - tax      1     3.772 137.46 -641.41
## - nox      1     5.793 139.49 -634.02
## - rad      1     6.609 140.30 -631.07
## - ptratio  1    15.334 149.03 -600.54
## - dis      1    17.962 151.65 -591.70
## - rm       1    23.770 157.46 -572.68
## - lstat    1    32.930 166.62 -544.07
```

In this project, we used bidirectional elimination. The goal is to have the combination of variables that has the lowest AIC or lowest residual sum of squares (RSS). If we focus on those columns, we can see that "indus" and "age" have the smallest AIC and RSS. Thus, we should consider deleting those two variables.

Lasso Regression

Lasso is also known as least absolute shrinkage and selection operator. It is a regression analysis technique that does both variable selection and regularization in order to improve the accuracy of the resulting statistical model's predictions and its readability.

```
fit.lar = lars(as.matrix(x),as.matrix(y),type="lasso")
summary(fit.lar)
```

```
## LARS/LASSO
## Call: lars(x = as.matrix(x), y = as.matrix(y), type = "lasso")
##    Df   Rss       Cp
## 0   1 42716 1360.011
## 1   2 36326 1083.143
## 2   3 21335  431.009
## 3   4 14960  154.804
## 4   5 13867  109.115
## 5   6 13720  104.718
## 6   7 13262   86.725
## 7   8 12658   62.354
## 8   9 12186   43.749
## 9  10 12091   41.625
## 10 11 11328   10.335
## 11 12 11310   11.514
## 12 13 11298   13.000
```

```
fit.lar
```

```
##
## Call:
## lars(x = as.matrix(x), y = as.matrix(y), type = "lasso")
## R-squared: 0.736
## Sequence of LASSO moves:
##      lstat rm ptratio  b crim dis nox zn rad tax indus age
## Var     12  5      10 11    1   7   4  2   8   9     3   6
## Step     1  2       3  4    5   6   7  8   9  10    11  12
```

From the data generated by Lasso, we find that the smallest Cp value of 10.335 exists at the $10th$ step. Then, it is suggesting us to delete the variables in the $11th$ and $12th$ step, which correspond to "indus" and "age".

New model after deleting "indus" and "age"

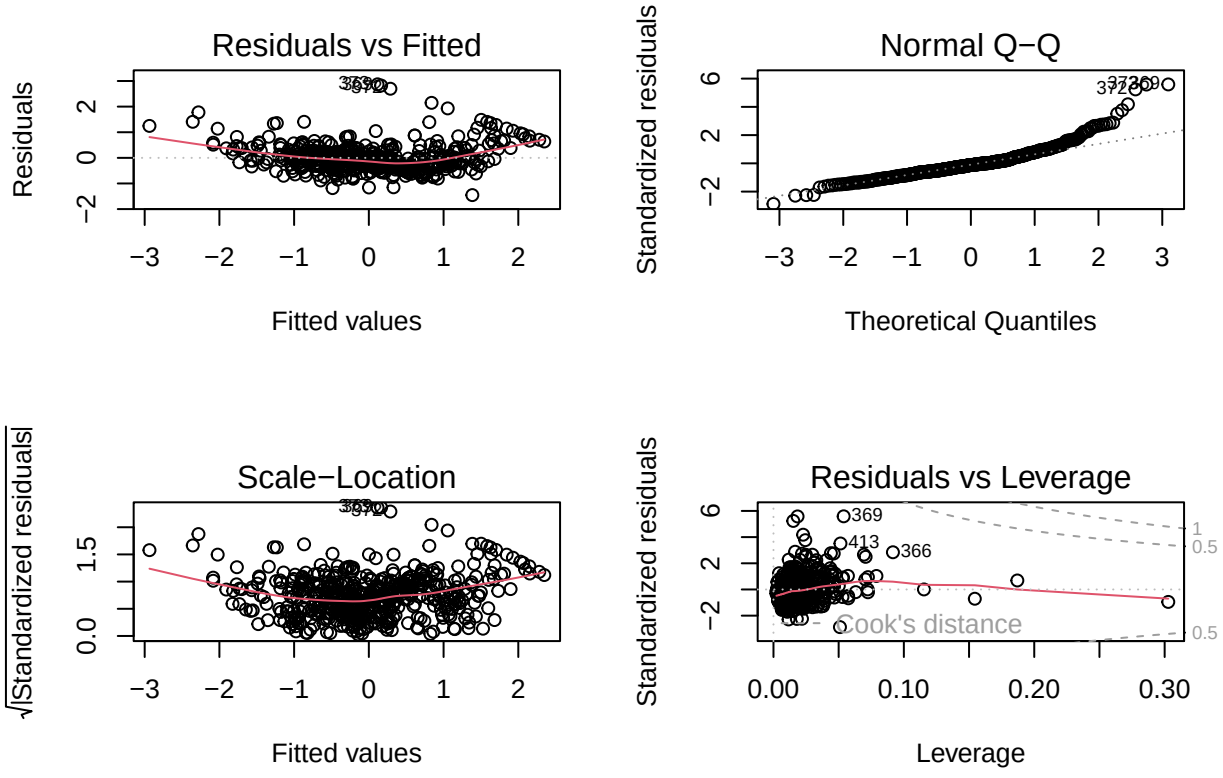
Since the results from both Stepwise regression and Lasso regression suggest to delete "indus" and "age," we should generate a new model by removing those two variables and check if there is any improvement.

```
data.2 = data.1[,-c(3,6)]
fit.2 = lm(medv~.-1,data=data.2)
summary(fit.2)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = data.2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45389 -0.30382 -0.05989  0.20596  2.87027
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## crim     -0.10667    0.03086  -3.456 0.000595 ***
## zn        0.11600    0.03457   3.355 0.000854 ***
## nox      -0.20750    0.04476  -4.636 4.55e-06 ***
## rm        0.29371    0.03128   9.391  < 2e-16 ***
## dis      -0.34941    0.04280  -8.163 2.71e-15 ***
## rad       0.29873    0.06033   4.952 1.01e-06 ***
## tax      -0.23226    0.06208  -3.741 0.000205 ***
## ptratio -0.23032    0.03054  -7.542 2.22e-13 ***
## b         0.09658    0.02672   3.614 0.000332 ***
## lstat    -0.41004    0.03710 -11.053  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5192 on 496 degrees of freedom
## Multiple R-squared:  0.7353, Adjusted R-squared:  0.7299
## F-statistic: 137.8 on 10 and 496 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(fit.2)
```



If we focus on the result, deletion of two variables lead to significant increase in F-statistics. However, R-Squared and Adjusted R-squared show little improvement. Next, as shown in the graph, Normal Q-Q plot indicates residuals of the model are nearly normal but still slightly off. Later, we need to further deal with

outliers and influential points.

Outliers and influential points

```
abnormal=data.frame(rstudent(fit.2),cooks.distance(fit.2))
head(abnormal[order(-abs(abnormal$rstudent.fit.2.)),])
```

```
##     rstudent.fit.2. cooks.distance.fit.2.
## 369        5.767510            0.17770857
## 373        5.758382            0.05859518
## 372        5.382262            0.04231751
## 370        4.254720            0.04073018
## 371        3.816851            0.03548701
## 413        3.554349            0.06686426
```

```
head(abnormal[order(-abnormal$cooks.distance.fit.2.),])
```

```
##     rstudent.fit.2. cooks.distance.fit.2.
## 369        5.767510            0.17770857
## 366        2.863291            0.08150281
## 413        3.554349            0.06686426
## 373        5.758382            0.05859518
## 368        2.678373            0.05282269
## 415        2.505076            0.04720772
```

```
delete_1 = c(369,373,372,370,371,413) # delete if cook distance is larger than 1 or standard residual i
data.3.1=data.2[-delete_1,]
#repeat the process
fit.3.1 = lm(medv~.-1,data=data.3.1)
summary(fit.3.1)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = data.3.1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.42082 -0.28477 -0.09735  0.15682  1.89798
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## crim    -0.09034    0.02617  -3.452 0.000605 ***
## zn       0.09529    0.02933   3.249 0.001236 **
## nox     -0.17295    0.03820  -4.528 7.49e-06 ***
## rm       0.35772    0.02708  13.212  < 2e-16 ***
## dis     -0.27299    0.03667  -7.444 4.42e-13 ***
## rad      0.22330    0.05139   4.345 1.69e-05 ***
## tax     -0.24620    0.05263  -4.678 3.76e-06 ***
## ptratio -0.23076    0.02590  -8.908  < 2e-16 ***
```

```
## b        0.09488    0.02288   4.146 3.99e-05 ***
## lstat   -0.30966    0.03267  -9.479  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4398 on 490 degrees of freedom
## Multiple R-squared:  0.794,  Adjusted R-squared:  0.7898
## F-statistic: 188.9 on 10 and 490 DF,  p-value: < 2.2e-16
```

```
abnormal=data.frame(rstudent(fit.3.1),cooks.distance(fit.3.1))
head(abnormal[order(-abs(abnormal$rstudent.fit.3.1)),])
```

```
##     rstudent.fit.3.1. cooks.distance.fit.3.1.
## 366          4.641660              0.22817651
## 368          4.125720              0.13185856
## 162          3.374991              0.02547836
## 365         -3.353378              0.06106596
## 375          3.334171              0.05111255
## 187          3.293585              0.01825271
```

```
head(abnormal[order(-abnormal$cooks.distance.fit.3.1),])
```

```
##     rstudent.fit.3.1. cooks.distance.fit.3.1.
## 366          4.641660              0.22817651
## 368          4.125720              0.13185856
## 381         -1.393922              0.08445953
## 365         -3.353378              0.06106596
## 415          2.744070              0.05875491
## 375          3.334171              0.05111255
```

```
delete_2 = c(366,368,162,365,375,187)
data.3.2=data.3.1[-delete_2,]
fit.3.2 = lm(medv~.-1,data=data.3.2)
summary(fit.3.2)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = data.3.2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.19716 -0.29289 -0.08712  0.14470  1.43534
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## crim    -0.08759    0.02426  -3.610 0.000338 ***
## zn       0.08628    0.02725   3.167 0.001638 **
## nox     -0.15990    0.03550  -4.504 8.35e-06 ***
## rm       0.43552    0.02754  15.813  < 2e-16 ***
## dis     -0.23585    0.03422  -6.893 1.71e-11 ***
## rad      0.19338    0.04795   4.033 6.40e-05 ***
## tax     -0.23213    0.04890  -4.747 2.72e-06 ***
```

```
## ptratio -0.21597    0.02408  -8.970  < 2e-16 ***
## b        0.11292    0.02139   5.279 1.96e-07 ***
## lstat   -0.23797    0.03172  -7.503 3.02e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4067 on 484 degrees of freedom
## Multiple R-squared:  0.8181, Adjusted R-squared:  0.8144
## F-statistic: 217.7 on 10 and 484 DF,  p-value: < 2.2e-16
```

```
abnormal=data.frame(rstudent(fit.3.2),cooks.distance(fit.3.2))
head(abnormal[order(-abs(abnormal$rstudent.fit.3.2)),])
```

```
##     rstudent.fit.3.2. cooks.distance.fit.3.2.
## 375          3.659628              0.06345520
## 167          3.199999              0.02803721
## 163          3.177871              0.02682262
## 415          3.039070              0.07355861
## 408          3.038872              0.01773286
## 402         -2.985563              0.01082574
```

```
head(abnormal[order(-abnormal$cooks.distance.fit.3.2),])
```

```
##     rstudent.fit.3.2. cooks.distance.fit.3.2.
## 381         -1.856430              0.15158542
## 415          3.039070              0.07355861
## 375          3.659628              0.06345520
## 254          2.461476              0.03385412
## 167          3.199999              0.02803721
## 163          3.177871              0.02682262
```

```
delete_3= c(375,167,163,415,408,402)
data.3.3=data.3.2[-delete_3,]
fit.3.3 = lm(medv~.-1,data=data.3.3)
summary(fit.3.3)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = data.3.3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.19465 -0.28626 -0.08509  0.13752  1.42596
##
## Coefficients:
##         Estimate Std. Error t value Pr(>|t|)
## crim    -0.09475    0.02416  -3.921 0.000101 ***
## zn       0.09091    0.02708   3.357 0.000850 ***
## nox     -0.15753    0.03541  -4.449 1.07e-05 ***
## rm       0.43489    0.02787  15.607  < 2e-16 ***
## dis     -0.23345    0.03404  -6.859 2.16e-11 ***
## rad      0.20424    0.04775   4.277 2.29e-05 ***
```

```
## tax     -0.24374    0.04866  -5.009 7.72e-07 ***
## ptratio -0.21005    0.02401  -8.748  < 2e-16 ***
## b        0.10613    0.02183   4.861 1.59e-06 ***
## lstat   -0.23211    0.03171  -7.319 1.06e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4029 on 478 degrees of freedom
## Multiple R-squared:  0.8169, Adjusted R-squared:  0.8131
## F-statistic: 213.3 on 10 and 478 DF,  p-value: < 2.2e-16
```

```
abnormal=data.frame(rstudent(fit.3.3),cooks.distance(fit.3.3))
head(abnormal[order(-abs(abnormal$rstudent.fit.3.3)),])
```

```
##     rstudent.fit.3.3. cooks.distance.fit.3.3.
## 375          3.674181              0.06638704
## 167          3.304046              0.03127726
## 163          3.288479              0.02996975
## 408          3.076550              0.01862419
## 415          3.047486              0.07670372
## 402         -3.008557              0.01127973
```

```
head(abnormal[order(-abnormal$cooks.distance.fit.3.3),])
```

```
##     rstudent.fit.3.3. cooks.distance.fit.3.3.
## 381         -1.672751              0.12543268
## 415          3.047486              0.07670372
## 375          3.674181              0.06638704
## 254          2.487764              0.03570633
## 167          3.304046              0.03127726
## 163          3.288479              0.02996975
```

Compare fit.3.3 and fit.2, F-statistic and R-square increase significantly, so the model improves. But outliers still exist, so OLS doesn't fit the data set.

Heteroscedasticity

After building a linear regression model, it is common to check for heteroscedasticity in the residuals, which means that the variance of the residuals is unequal for certain values within a range. If heteroscedasticity exists, the analysis results may be invalid due to the use of a population with unequal variance. There are two methods for testing heteroscedasticity, one is the residual plot analysis method and the other is the rank correlation coefficient test method. In this project, we use the rank correlation coefficient test method to calculate the Spearman rank correlation coefficient in regression analysis to test whether the heteroscedasticity hypothesis is correct. The Spearman correlation coefficient is more robust to extreme values in the data and can reflect the nonlinear relationship between variables, so it is more suitable for measuring the complex relationship between the absolute values of the independent variable and the residual.

```
e = resid(fit.3.3)
abse = abs(e)
spearman_result = list()
cor.spearman = vector()
```

```r
for(i in 1:10){
  spearman_result[[i]] = cor.test(data.3.3[,i], abse, exact = FALSE, method = "spearman")
  cor.spearman[i] = cor.test(data.3.3[,i], abse, exact = FALSE, method = "spearman")$p.value
}
spearman_result
```

```
## [[1]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 18229252, p-value = 0.1944
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.05884218
##
##
## [[2]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 17767511, p-value = 0.06801
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.08268141
##
##
## [[3]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 17783532, p-value = 0.07082
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.08185424
##
##
## [[4]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 15851100, p-value = 5.448e-05
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.1816238
##
```

```
##
## [[5]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 22068247, p-value = 0.00203
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## -0.1393613
##
##
## [[6]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 17088385, p-value = 0.009229
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##      rho
## 0.117744
##
##
## [[7]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 17763441, p-value = 0.06731
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho
## 0.08289153
##
##
## [[8]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 20898986, p-value = 0.08129
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##         rho
## -0.07899348
##
##
## [[9]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
```

```
## S = 20130165, p-value = 0.3863
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##         rho
## -0.03930001
##
##
## [[10]]
##
##  Spearman's rank correlation rho
##
## data:  data.3.3[, i] and abse
## S = 19515066, p-value = 0.868
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##         rho
## -0.00754311
```

```
cor.spearman
```

```
##  [1] 1.944051e-01 6.801105e-02 7.082236e-02 5.448079e-05 2.030279e-03
##  [6] 9.229305e-03 6.731165e-02 8.128597e-02 3.863399e-01 8.679934e-01
```

```
names(data.3.3[,-(11)])[cor.spearman<0.05]
```

```
## [1] "rm"  "dis" "rad"
```

In this model, we first calculate the residuals of the new regression, then store the test results by taking the absolute value of the residuals and creating a list to store the p-value for each test in a new vector, The output is the p-value for each independent variable tested by spearman test. After the comparison, there are three variables with p-value less than 0.05 which is the average number of rooms per dwelling (rm), weighted distances to five Boston employment centers (dis), and the index of accessibility to radial highways (rad).

Autocorrelation

The graph shows the change in the correlation coefficient of time series data as the lag number changes. When the lag number is 0, the correlation coefficient is 1, indicating that the two variables are completely positively correlated; as the lag number increases, the correlation coefficient gradually decreases and becomes stable, indicating that the correlation between the two variables gradually decreases and finally becomes uncorrelated.

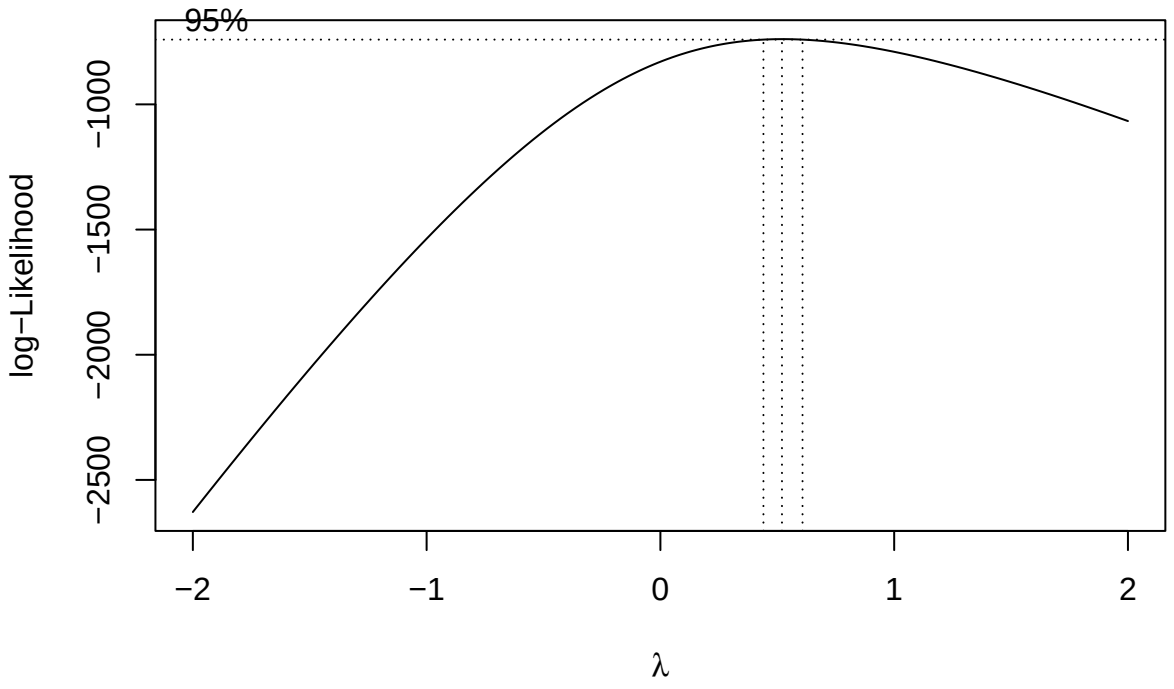
```
dev.new(width=4.75,height=3,pointsize=10)
acf(residuals(fit.3.3))
```

Box-Cox: deal with autocorrelation and heteroscedasticity

Regression diagnosis determines the validity of the model hypothesis through residual analysis and identifies high-impact data points through residual analysis and influence analysis. We use the Spearman rank correlation coefficient test. After finding problems, the main tool for resolving them is the Box-Cox transformation.

Residual vs fitted: the residuals are well scattered above and below zero, with vertical spread (indicating variance) which does not depend much on the fitted value. Normal Q-Q: The figure shows that the errors basically obey the normal distribution. Scale-location : describe homoscedasticity. Constant variance is satisfied, and the points next to the horizontal line are randomly distributed. Residuals vs Leverage: it involves the calculation of standardized residuals, leverage values, and Cook's distance, which can identify outliers and high leverage points (leverage points) and thus identify strong influence points. If there are leverage points, it is necessary to determine which are the bad leverage points, specifically in the plot, this means points with Cook's distance greater than $0.5 - 1$, that is, these points need to be evaluated for their impact on the fitting model (further evaluation of the impact is not involved).

```
x=rep(2,488)
medv = data.3.3[c("medv")] +x
data.4 = cbind(data.3.3[,c(1:10)],medv)
bc = boxcox(medv~., data=data.4, lambda=seq(-2, 2, 0.01))
```




```
lambda = bc$x[which.max(bc$y)]
lambda
```

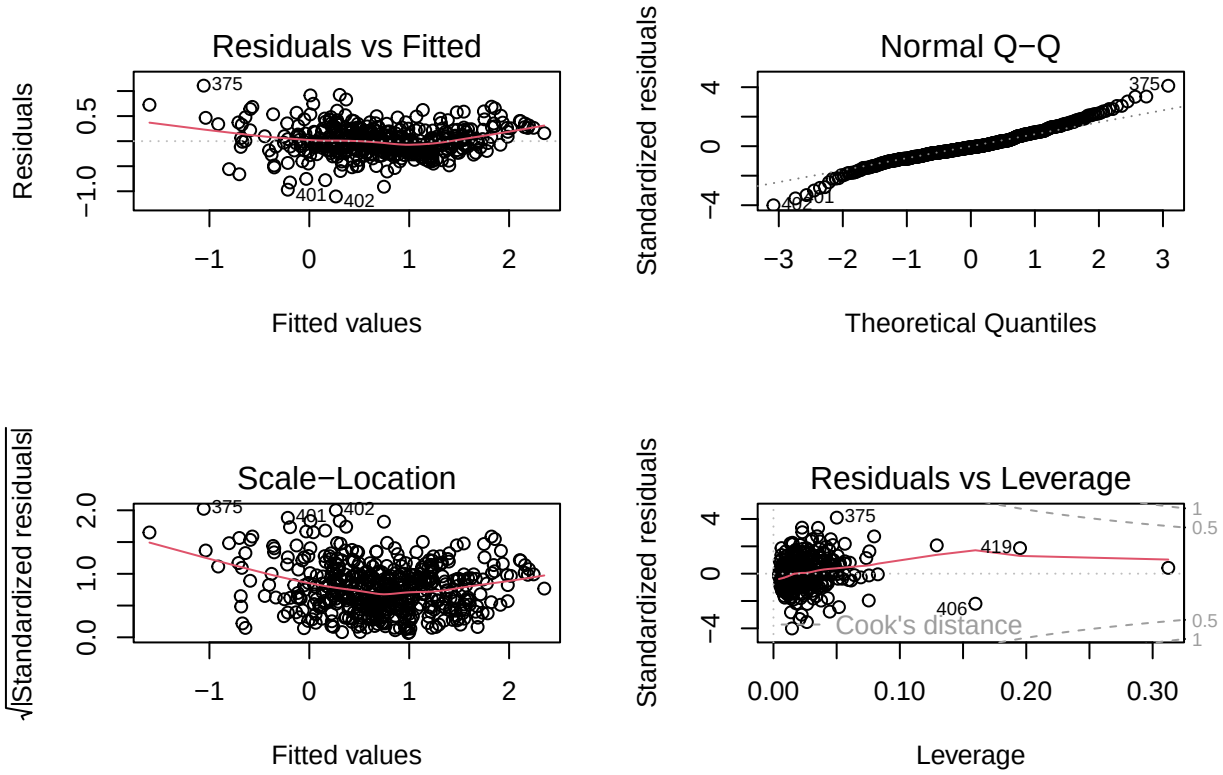
```
## [1] 0.52
```

```
medv_bc = (data.4$medv ^ lambda - 1) / lambda
fit.4 = lm(medv_bc~.-medv,data=data.4)
summary(fit.4)
```

```
##
```

```
## Call:
## lm(formula = medv_bc ~ . - medv, data = data.4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10338 -0.14968 -0.02337  0.14951  1.10706
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.72469    0.01259  57.567  < 2e-16 ***
## crim        -0.12156    0.01663  -7.308 1.15e-12 ***
## zn           0.04399    0.01864   2.360   0.0187 *
## nox         -0.10894    0.02438  -4.469 9.83e-06 ***
## rm           0.24044    0.01919  12.528  < 2e-16 ***
## dis         -0.14616    0.02344  -6.235 9.98e-10 ***
## rad          0.15773    0.03288   4.798 2.15e-06 ***
## tax         -0.17683    0.03350  -5.279 1.98e-07 ***
## ptratio     -0.13990    0.01653  -8.463 3.22e-16 ***
## b            0.07696    0.01503   5.119 4.45e-07 ***
## lstat       -0.22734    0.02185 -10.404  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2774 on 477 degrees of freedom
## Multiple R-squared:  0.8326, Adjusted R-squared:  0.8291
## F-statistic: 237.2 on 10 and 477 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2,2))
plot(fit.4)
```



Compared with fit.3.3, R-square and F-statistic increase, the model is improved, so we choose fit.4 as the result.

Conlusion

As can be seen from the historical fluctuations of Boston housing prices, Boston housing prices continuously and rapidly rose in the third quarter of 1983 − 1988, with a magnitude and speed far higher than the average of the US, and then showed a downward trend in 1989 − 1993, with a large downward amplitude and a long-lasting stagnant market. In addition to being related to the prosperity of the local economy and consumer expectations, the significant increase in population and housing demand caused by a large influx of immigrants is particularly noticeable. Therefore, from a historical perspective, how to regulate the stability of Boston housing prices has a extremely important impact on the local political, economic, and social development.

According to data from the U.S. Bureau of Census Statistics, during the period from 1980 to 1990, Boston had a total of 187, 000 new immigrants, accounting for 4.7% of its population in 1980, showing the significant impact of immigration on its population. Due to the prosperity of the local real estate industry, a large influx of funds into the industry led to a significant increase in housing supply. However, due to the lag of the real estate industry, housing prices also fell several years later and it was difficult to reverse them in a short time. Excessive prosperity or decline of the real estate industry is not conducive to the economic development and social stability of the city, therefore the Boston government should control the fluctuation range of housing prices to prevent excessive fluctuations.

Combining the signs of the regression coefficients of each explanatory variable in the regression results with the median of own homes, the government's housing price regulation policies can be divided into the following two situations. When the real estate market is overheated, first, the housing price can be suppressed in a

short time by adjusting the property tax rate (tax coefficient is negative), although this method is flexible and efficient, but due to the lack of policy consistency over time, it may cause panic among investors in the real estate industry and is not conducive to the long-term development of the industry; Second, administrative means can be used to limit environmental indicators such as nitrogen oxide content (Nox coefficient is negative, through traffic restriction, reducing nitrogen oxide emission from vehicle use); Third, by promoting low-rent and affordable housing (rm coefficient is negative), increasing the supply of single-room housing, and ensuring the housing needs of the floating population to stabilize housing prices.

When housing prices go down (which is very likely due to social stability and financial economic issues), the first step the Boston government can take is to increase police force and security spending (as crim is negatively correlated), which can help stabilize society by reducing crime rates and maintain housing stability. Second, the government can also improve infrastructure, increase the number of regional schools, attract more high-level teachers (as ptratio is negatively correlated), further plan for urban road reconstruction (as rad is positively correlated), and enhance the accessibility and convenience of city transportation, thus improving public services. Third, the government can also build a social welfare system and improve the basic system of protection for the floating population, and increase the income of the low-income group (as lstat is negatively correlated), thereby expanding the proportion of the middle-income group. Fourth, the government can also promote population mobility, attract high-quality labor force into urban areas (as b is positively correlated), enhance the development of the tertiary industry service, increase residents' consumption and income levels, and promote the stable development of the real estate industry.