Regression Analysis Course Project

张若曦 20307100127

2023-12-25

Regression Analysis Course Project Code

1. Data Preprocessing

1.1 load the packages

```
library(mlbench)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(MASS)
library(car)
```

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

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

1.2 load the data

```r
data("BostonHousing") # load "BostonHousing" data
? BostonHousing # check the documentation of BonstonHousing data
```

1.3 delete "chas" variable

```r
index_chas <- which(names(BostonHousing) == 'chas') # find the index of "chas"
df <- BostonHousing[,-index_chas] # delete "chas"
```

1.4 split train-test set

```r
set.seed(123) # Setting a seed for reproducibility
n_rows <- nrow(df) # Get the number of rows in df
train_indices <- sample(n_rows, round(0.80 * n_rows)) # Generate random indices for training d
df_train <- df[train_indices, ] # Subset df to create the training dataset
df_test <- df[-train_indices, ] # Subset df to create the testing dataset
```

No NA data in this dataset. Thus no further treatment needed.

1.5 Centralization and Standardization

In the following analysis, when fitting the linear model, all the data used are already centralize

```r
df_scale <- data.frame(scale(df, center = TRUE, scale = TRUE))

train_mean <- apply(df_train, 2, mean)
train_sd <- apply(df_train, 2, sd)

df_train <- sweep(df_train, 2, train_mean, "-")
df_train <- sweep(df_train, 2, train_sd, "/")
df_test <- sweep(df_test, 2, train_mean, "-")
df_test <- sweep(df_test, 2, train_sd, "/")
```

2. Descriptive Analysis and Dignoses of Multicollinearity

In this part, we first show the descriptive statistics of the dataset and then test the multicollinearity of the covariates, to decide whether resolutions such as ridge regression or PCA would be required later.

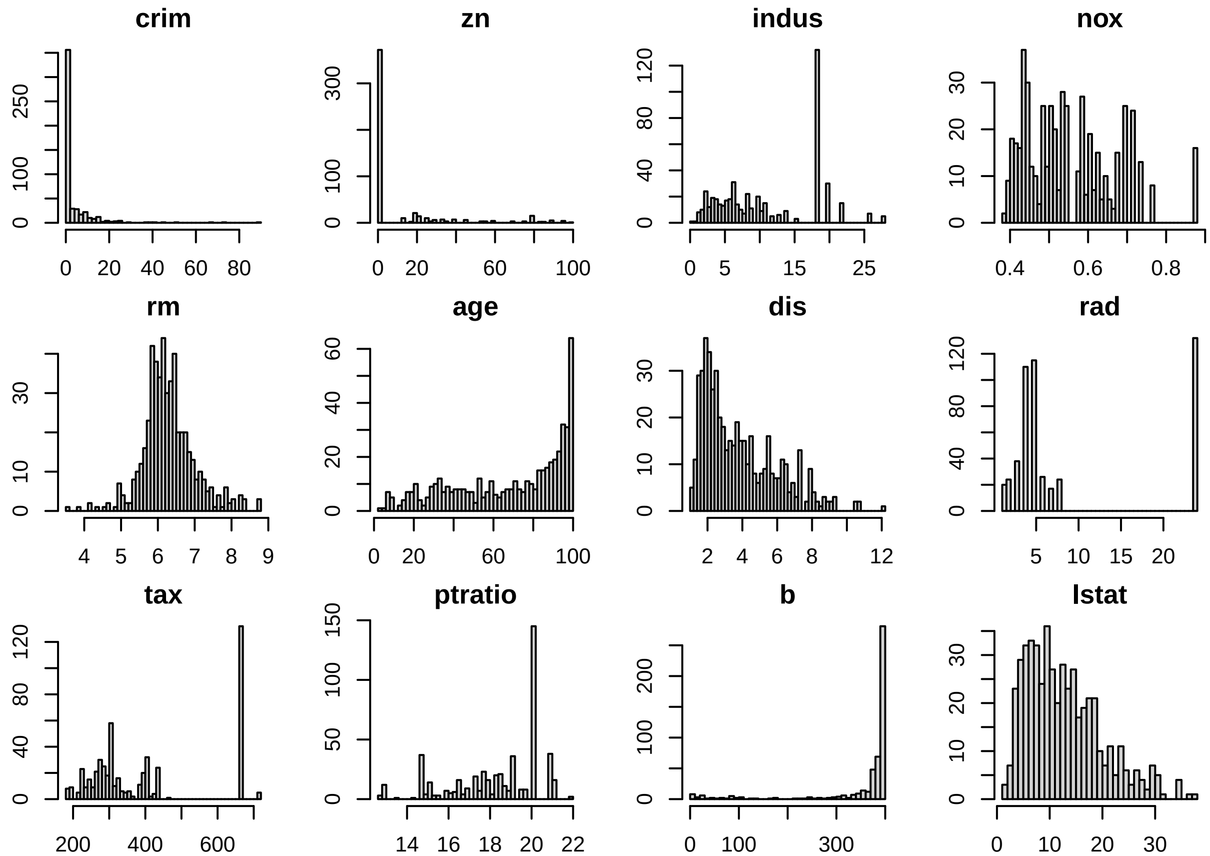
2.1 Descriptive Analysis

```r
summary(df)
```

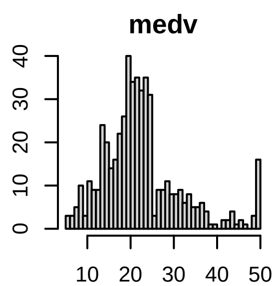
```
##       crim                zn             indus            nox        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.3850  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.4490  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.5380  
##  Mean   : 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
```

```r
  par(mar=c(2, 2, 2, 2), mfrow=c(3, 4))

  # Loop through each column in the dataframe and create a histogram
  for(i in 1:ncol(df)) {
```

```r
  hist(df[,i], breaks = 50, main = paste(names(df)[i]), xlab = names(df)[i])
}
```



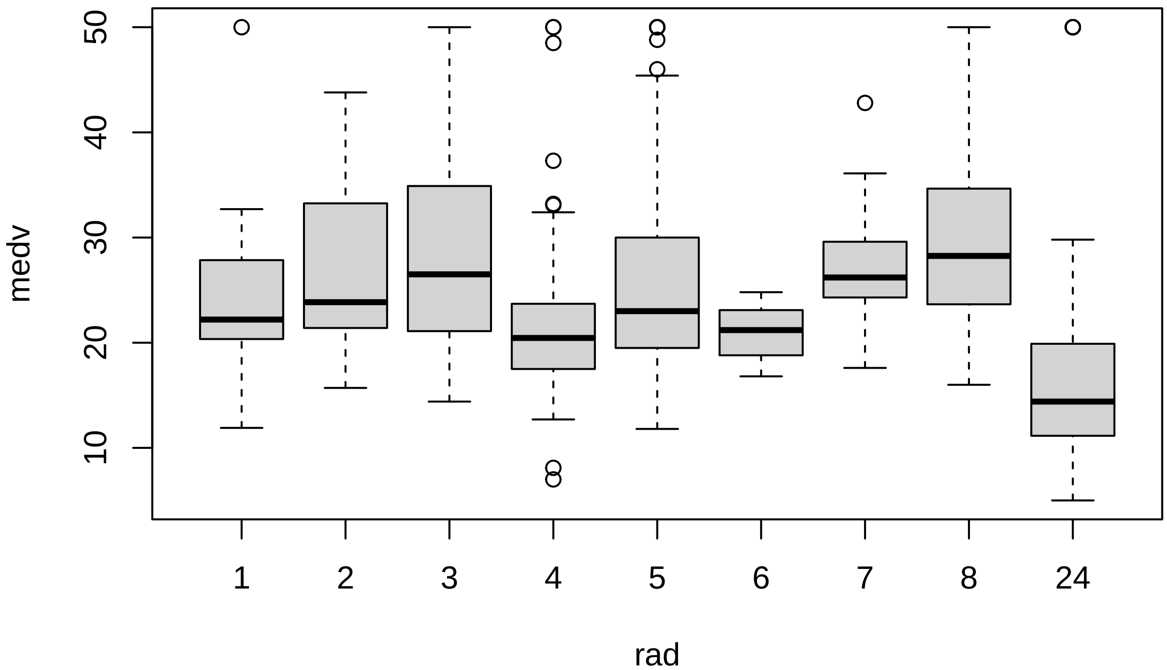

```r
par(mfrow = c(1, 1)) # Reset to default
```



Check the correlation between the response variable and covariates

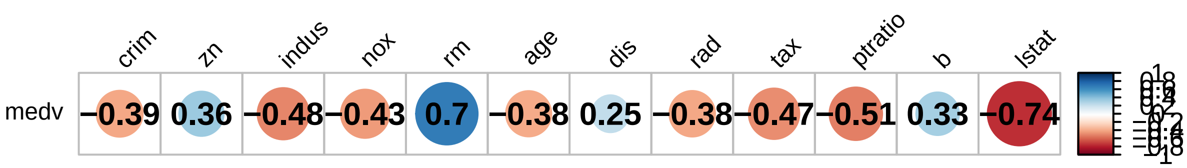
response "medv" vs qualitative variale "rad"

```r
# Create the boxplot
bp = boxplot(medv ~ rad, data = df)
```

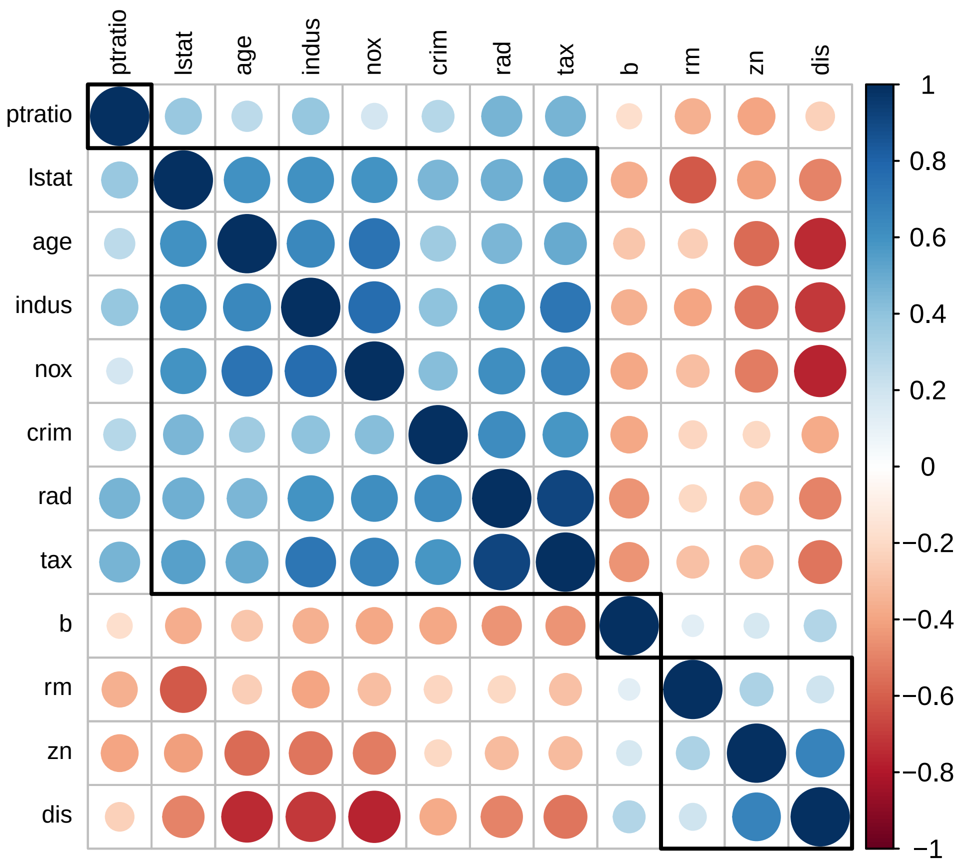


Show the correlation plot

```
corr <- cor(df)

# Extract correlation of 'medv' with other variables
corr_medv <- corr["medv", -which(names(df) == "medv")]

# Convert to a matrix and set column names
corr_medv_matrix <- matrix(corr_medv, nrow = 1)
rownames(corr_medv_matrix) <- "medv"
colnames(corr_medv_matrix) <- names(corr_medv)

# Plot1: Correlation of 'medv' with other variables
corrplot(corr_medv_matrix, is.corr = TRUE, method = "circle",
         type = "upper", tl.col = 'black', tl.cex = 0.75,
         tl.srt = 45, addCoef.col = "black")
```




```
# Plot2: Correlation among covariates
corr_covariates <- corr[-which(names(df) == 'medv'), -which(names(df) == 'medv')]
corrplot(corr_covariates, order = "hclust", tl.col = 'black',
         tl.cex = 0.75, addrect = 4)
```



By first sight, some covariates show quite strong correlation between each other, we might worry i

2.2 Diagnoses of Multicollinearity

Using the standardized data for calculating the VIF and Condition Number is equivalent to using th

VIF

```
lm_full = lm(data = df_scale, formula = medv~.)
cat("VIF of full model:\n")
```

```
## VIF of full model:
```

```
vif(lm_full) # Calculate the VIF for regression result
```

```
##     crim       zn    indus      nox       rm      age      dis      rad
## 1.787705 2.298257 3.949246 4.388775 1.931865 3.092832 3.954961 7.397844
##      tax  ptratio        b    lstat
## 8.876233 1.783302 1.344971 2.931101
```

Condition Number

```
XX = cor(dplyr::select(df_scale, -c('medv'))) # Calculate the correlation of the design matrix
cat("Condition Number of full model:\n")
```

```
## Condition Number of full model:
```

```r
kappa(XX, exact = T) # Calculate the condition number
```

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

```
From the results of VIF(<10) and Condition Number(<100), we could conclude that there is no strong
```

3. OLSE and Model Selection

1. OLS

```r
n = nrow(df_scale) # sample size
p = ncol(df_scale) - 1 # covariates numbers

# Recall that in 2.2, to test the VIF, we have already fitted a linear regression model:
# lm_full = lm(data = df_scale, formula = medv~.)
sum(resid(lm_full)^2)
```

```
## [1] 133.5642
```

```r
print(summary(lm_full))
```

```
##
## Call:
## lm(formula = medv ~ ., data = df_scale)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45663 -0.30556 -0.07019  0.20812  2.86780
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.986e-15  2.314e-02   0.000 1.000000
## crim        -1.058e-01  3.097e-02  -3.417 0.000686 ***
## zn           1.193e-01  3.511e-02   3.398 0.000734 ***
## indus        3.007e-02  4.603e-02   0.653 0.513889
## nox         -2.188e-01  4.852e-02  -4.509 8.13e-06 ***
## rm           2.942e-01  3.219e-02   9.137  < 2e-16 ***
## age          8.520e-03  4.073e-02   0.209 0.834407
## dis         -3.401e-01  4.606e-02  -7.383 6.64e-13 ***
## rad          3.108e-01  6.300e-02   4.934 1.10e-06 ***
## tax         -2.521e-01  6.901e-02  -3.653 0.000287 ***
## ptratio     -2.333e-01  3.093e-02  -7.542 2.25e-13 ***
```

```
## b            9.670e-02  2.686e-02   3.600 0.000351 ***
## lstat       -4.147e-01  3.965e-02 -10.459  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5205 on 493 degrees of freedom
## Multiple R-squared:  0.7355, Adjusted R-squared:  0.7291
## F-statistic: 114.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

```r
print(Anova(lm_full, type = 'III' )) # conduct F test on all the coefficients and output the r
```

```
## Anova Table (Type III tests)
##
## Response: medv
##              Sum Sq  Df  F value    Pr(>F)
## (Intercept)   0.000   1   0.0000 1.0000000
## crim          3.163   1  11.6742 0.0006861 ***
## zn            3.128   1  11.5468 0.0007336 ***
## indus         0.116   1   0.4268 0.5138885
## nox           5.509   1  20.3354 8.130e-06 ***
## rm           22.619   1  83.4909 < 2.2e-16 ***
## age           0.012   1   0.0437 0.8344066
## dis          14.768   1  54.5096 6.635e-13 ***
## rad           6.595   1  24.3430 1.104e-06 ***
## tax           3.615   1  13.3439 0.0002870 ***
## ptratio      15.409   1  56.8756 2.250e-13 ***
## b             3.511   1  12.9591 0.0003506 ***
## lstat        29.636   1 109.3905 < 2.2e-16 ***
## Residuals   133.564 493
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. Model Selection with Stepwise Regression

The default setting is with AIC as selection criterion

```r
# stepwise regression
step_for = step(lm_full, direction = 'forward')
step_bac = step(lm_full, direction = 'backward')
step_step = step(lm_full, direction = 'both')
```

```r
# output the model summary of stepwise model selection
summary(step_for)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + indus + nox + rm + age + dis +
##     rad + tax + ptratio + b + lstat, data = df_scale)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45663 -0.30556 -0.07019  0.20812  2.86780
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.986e-15  2.314e-02   0.000 1.000000
## crim        -1.058e-01  3.097e-02  -3.417 0.000686 ***
## zn           1.193e-01  3.511e-02   3.398 0.000734 ***
## indus        3.007e-02  4.603e-02   0.653 0.513889
## nox         -2.188e-01  4.852e-02  -4.509 8.13e-06 ***
## rm           2.942e-01  3.219e-02   9.137  < 2e-16 ***
## age          8.520e-03  4.073e-02   0.209 0.834407
## dis         -3.401e-01  4.606e-02  -7.383 6.64e-13 ***
## rad          3.108e-01  6.300e-02   4.934 1.10e-06 ***
## tax         -2.521e-01  6.901e-02  -3.653 0.000287 ***
## ptratio     -2.333e-01  3.093e-02  -7.542 2.25e-13 ***
## b            9.670e-02  2.686e-02   3.600 0.000351 ***
## lstat       -4.147e-01  3.965e-02 -10.459  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5205 on 493 degrees of freedom
## Multiple R-squared:  0.7355, Adjusted R-squared:  0.7291
## F-statistic: 114.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

```r
summary(step_bac)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + nox + rm + dis + rad + tax +
##     ptratio + b + lstat, data = df_scale)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45389 -0.30382 -0.05989  0.20596  2.87027
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.003e-15  2.310e-02   0.000 1.000000
## crim        -1.067e-01  3.089e-02  -3.453 0.000602 ***
## zn           1.160e-01  3.461e-02   3.352 0.000864 ***
## nox         -2.075e-01  4.480e-02  -4.631 4.65e-06 ***
## rm           2.937e-01  3.131e-02   9.381  < 2e-16 ***
## dis         -3.494e-01  4.285e-02  -8.155 2.89e-15 ***
## rad          2.987e-01  6.039e-02   4.947 1.04e-06 ***
## tax         -2.323e-01  6.215e-02  -3.737 0.000208 ***
## ptratio     -2.303e-01  3.057e-02  -7.535 2.34e-13 ***
## b            9.658e-02  2.675e-02   3.611 0.000337 ***
## lstat       -4.100e-01  3.714e-02 -11.042  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5197 on 495 degrees of freedom
## Multiple R-squared:  0.7353, Adjusted R-squared:  0.7299
## F-statistic: 137.5 on 10 and 495 DF,  p-value: < 2.2e-16
```

```r
summary(step_step)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + nox + rm + dis + rad + tax +
##     ptratio + b + lstat, data = df_scale)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.45389 -0.30382 -0.05989  0.20596  2.87027
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.003e-15  2.310e-02   0.000 1.000000
## crim        -1.067e-01  3.089e-02  -3.453 0.000602 ***
## zn           1.160e-01  3.461e-02   3.352 0.000864 ***
## nox         -2.075e-01  4.480e-02  -4.631 4.65e-06 ***
## rm           2.937e-01  3.131e-02   9.381  < 2e-16 ***
## dis         -3.494e-01  4.285e-02  -8.155 2.89e-15 ***
## rad          2.987e-01  6.039e-02   4.947 1.04e-06 ***
```

```
## tax         -2.323e-01  6.215e-02  -3.737 0.000208 ***
## ptratio     -2.303e-01  3.057e-02  -7.535 2.34e-13 ***
## b            9.658e-02  2.675e-02   3.611 0.000337 ***
## lstat       -4.100e-01  3.714e-02 -11.042  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5197 on 495 degrees of freedom
## Multiple R-squared:  0.7353, Adjusted R-squared:  0.7299
## F-statistic: 137.5 on 10 and 495 DF,  p-value: < 2.2e-16
```

As we could see, the backward approach and stepwise approach yield the same ending model, which is

Thus we keep this variable selection result in the following analysis.

```r
lm_reduced = step_step # save the reduced model result
df_reduced = as.data.frame(select(df_scale, -c('age', 'indus'))) # drop the reduced covariates
```

4. Regression Diagnostics

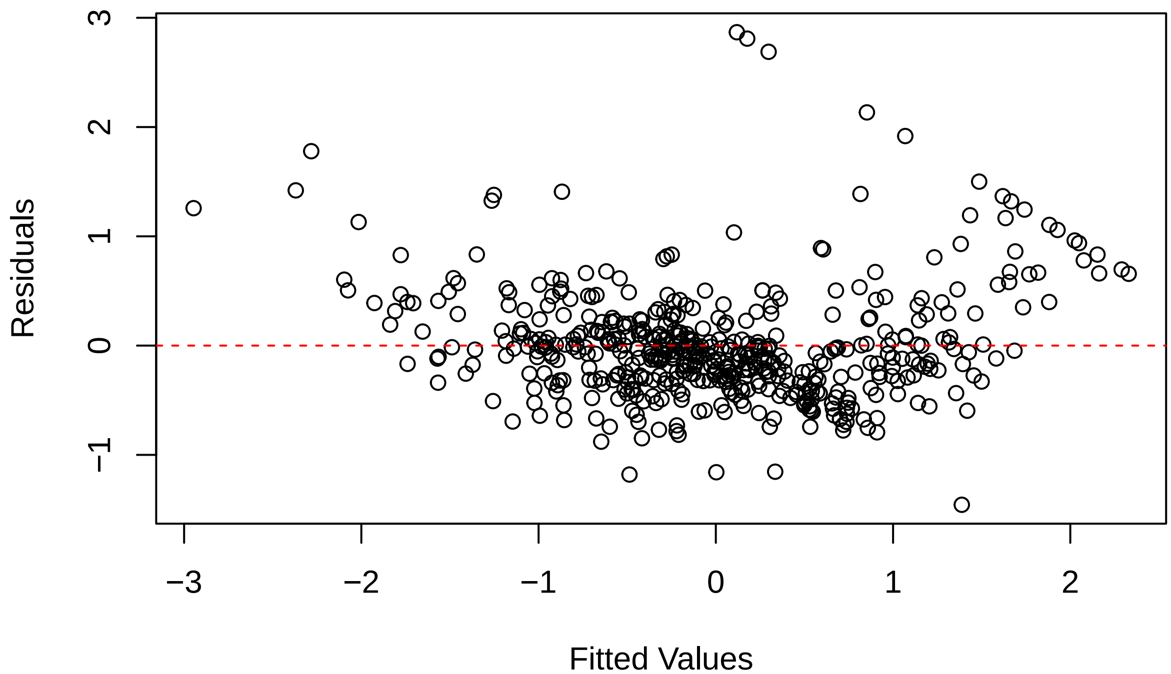
We first perform the Diagnoses on the full model, then repeat the steps for the reduced model.

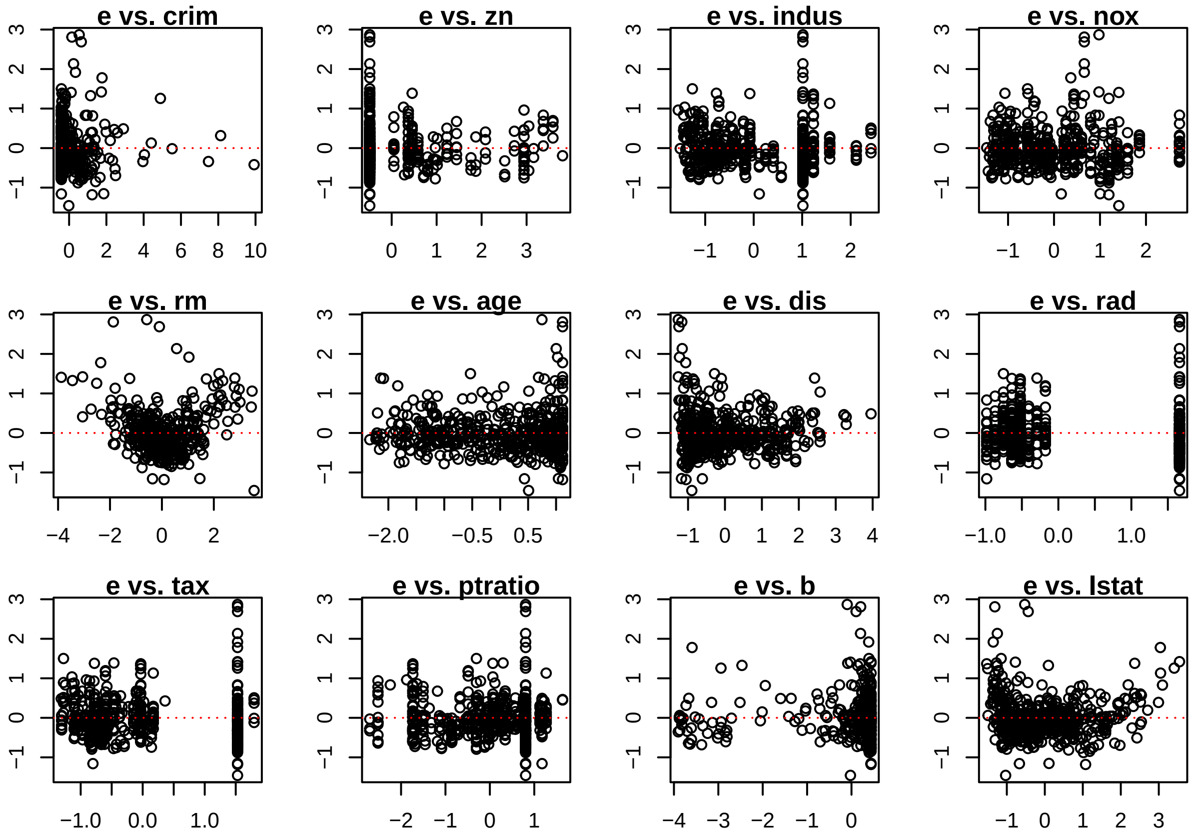
1. Diagnosis for Heteroscedasticity

   a. Residual plot analysis

```r
# Define the function for future repetitive use
residual_plot <- function(df, lm){

  X = as.matrix(select(df, -c(medv)))
  residuals =  resid(lm)

  # Plot the fitted values y_hat with the residuals
  plot(lm$fitted.values, residuals, xlab = "Fitted Values", ylab = "Residuals")
  abline(h = 0, lty = 2, col='red')

  # Plot the covariates with the residuals
  # Set up a multi-panel plotting layout
  par(mfrow = c(3, 4), mar = c(3, 3, 1, 1))  # 3 rows and 4 columns

  for (i in 1:ncol(X)) {
    # Create a data frame for plotting
    plot(X[, i], residuals,
```

```
          xlab = colnames(X)[i],
          ylab = "residuals",
          main = paste("e vs.", colnames(X)[i]))
    abline(h = 0, lty = 9, col = 'red')
  }
}

residual_plot(df_scale, lm_full)
```





As we could see, for some covariates, the fluctuation of residuals shows relationship with the covariate value, i.e. when "tax" increases, "age" increases, and "dis" decreases, the fluctuation of residuals increases. We then proceed to conduct test to obtain more grounded diagnoses.

b. Rank correlation coefficient test

```
# Define the Spearman correlation test function to be used by all the covariates
heteroscedasticity_test <- function(df, lm){
  X = as.matrix(select(df, -c(medv)))
  abse = abs(resid(lm))

  spearman_results <- data.frame(
    Variable = character(0),
    Spearman_Correlation = numeric(0),
    Spearman_P_Value = numeric(0)
  )
  for(i in 1:ncol(X)){
    spearman_cor <- suppressWarnings(cor.test(X[, i], abse, method = "spearman"))
    spearman_results <- rbind(spearman_results, data.frame(
        Variable = colnames(X)[i],
        Spearman_Correlation = spearman_cor$estimate,
```

```r
        Spearman_P_Value = spearman_cor$p.value
      ))
  }
  cat("\nTable1: Spearman Correlation Results\n")
  print(spearman_results)
  cat("\nTable2: Spearman Correlations Under Threshold 0.05\n")
  print(spearman_results[spearman_results$Spearman_P_Value < 0.05, ])
}

# Perform the Spearman's Rank correlation test on the full model
heteroscedasticity_test(df_scale, lm_full)
```

```
##
## Table1: Spearman Correlation Results
##        Variable Spearman_Correlation Spearman_P_Value
## rho        crim           0.04565010     0.3054252835
## rho1         zn           0.09147063     0.0397049085
## rho2      indus          -0.03157379     0.4785386194
## rho3        nox           0.05245796     0.2388342911
## rho4         rm           0.15833075     0.0003497853
## rho5        age           0.12307461     0.0055679808
## rho6        dis          -0.15510651     0.0004624398
## rho7        rad           0.11346964     0.0106377560
## rho8        tax           0.07621891     0.0867578522
## rho9    ptratio          -0.11498021     0.0096364316
## rho10         b          -0.02559566     0.5656766008
## rho11     lstat          -0.09895071     0.0260274953
##
## Table2: Spearman Correlations Under Threshold 0.05
##        Variable Spearman_Correlation Spearman_P_Value
## rho1         zn           0.09147063     0.0397049085
## rho4         rm           0.15833075     0.0003497853
## rho5        age           0.12307461     0.0055679808
## rho6        dis          -0.15510651     0.0004624398
## rho7        rad           0.11346964     0.0106377560
## rho9    ptratio          -0.11498021     0.0096364316
## rho11     lstat          -0.09895071     0.0260274953
```

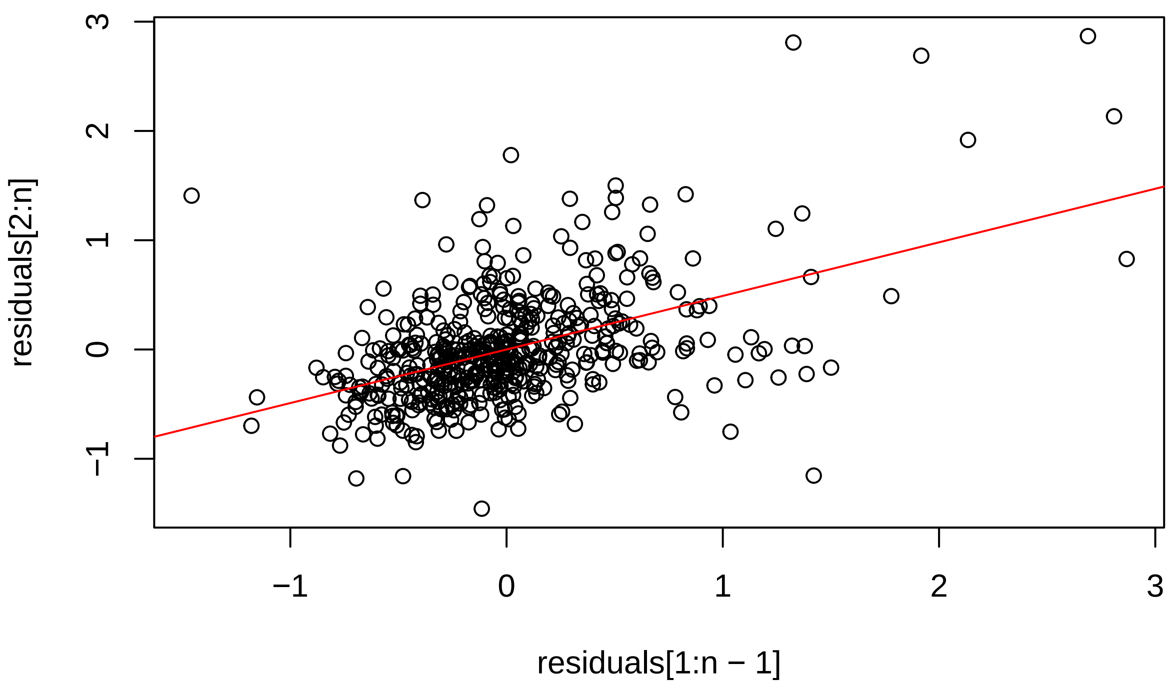
We could see that quite a few covariates have significant correlation with the residuals, indicating heteroscedasticity.

2. Diagnoses of Autocorrelation

   a. Plot Analysis

   Plot the scatter plot of e_t and e_t-1 (t = 2, 3, ...)

```
n = nrow(df_scale)
residuals = resid(lm_full)

model = lm(residuals[2:n]~residuals[1:n-1])
plot(residuals[1:n-1], residuals[2:n])
abline(model,col = 'red', lwd = 1) # Add the line fitted by univariate linear regression
```



From the plot, we could see some correlation between e_t and e_t-1.

   b. DW Test

```
dwtest(lm_full) # conduct the dwtest on the result
```

```
##
##  Durbin-Watson test
##
## data:  lm_full
## DW = 1.0159, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

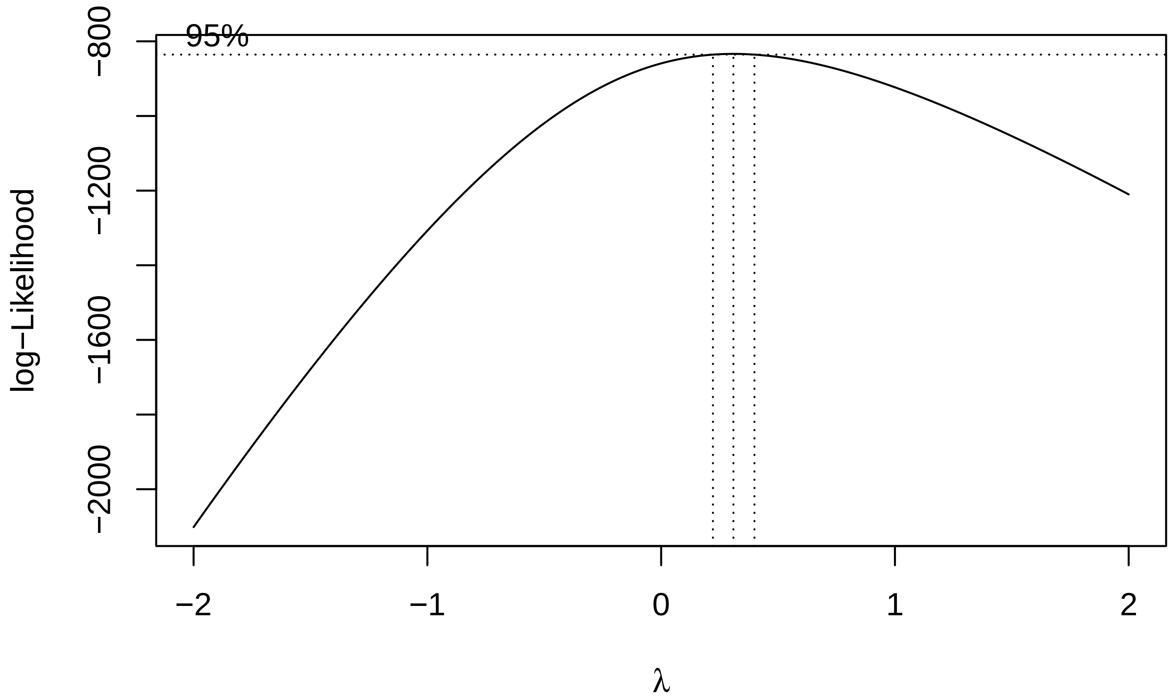
The results from DW test also supports that autocorrelation between observations exits in the model.

3. Resolution of Heteroscedasticity and Autocorrelation

   a. See if Box-Cox transformation could solve these two problems altogether at once

```
boxcox.fitting <- function(df){

  # The function would return: a new df_bc after box-cox transformation,
  # the lambda used in the transformation, and the transformed model
  # summary of the model and the likelihood plot would also be printed out

  # Adjusting 'medv' in the standardized dataset as it contains negative values
  if (any(df$medv < 0)) {
    # Apply the transformation when negative values exist
    df_bc <- df
    df_bc$medv <- df_bc$medv - 1.1 * min(df_bc$medv)
    cat("Note: conducted y = y - 1.1 min(y) since there are negative values in the 'medv' vari
  } else{
    df_bc <- df
  }

  # Conducting Box-Cox transformation on 'medv'
  bc <- boxcox(medv ~ ., data = df_bc, lambda = seq(-2, 2, 0.001))
  # Determining the lambda value that maximizes the log-likelihood
  lambda <- bc$x[which.max(bc$y)]

  # Applying the Box-Cox transformation to 'medv'
  df_bc$medv <- (df_bc$medv ^ lambda - 1) / lambda

  # Fitting a linear regression model with the transformed 'medv'
  lm_bc <- lm(formula = medv ~ .-medv, data = df_bc)

  # Summarizing the results of the linear regression
  print(summary(lm_bc))

return(list(lambda=lambda, df_bc=df_bc, lm_bc=lm_bc))
}

result = boxcox.fitting(df_scale)
```

```
## Note: conducted y = y - 1.1 min(y) since there are negative values in the 'medv' variable.
```

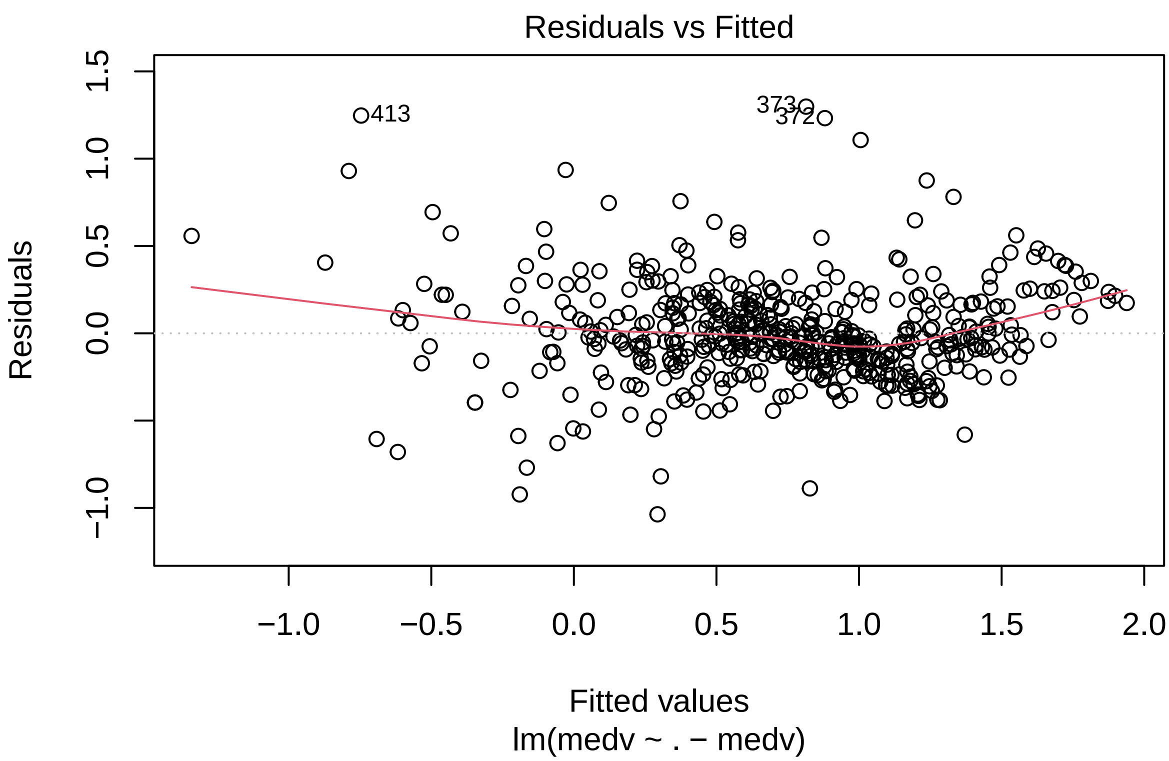



```
##
## Call:
## lm(formula = medv ~ . - medv, data = df_bc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03654 -0.14804 -0.02736  0.14553  1.29803
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.73439    0.01261  58.252  < 2e-16 ***
## crim        -0.12419    0.01687  -7.360 7.74e-13 ***
## zn           0.04653    0.01913   2.432 0.015357 *
## indus        0.02999    0.02508   1.196 0.232354
## nox         -0.13014    0.02644  -4.923 1.17e-06 ***
## rm           0.10910    0.01754   6.220 1.06e-09 ***
## age          0.01159    0.02219   0.522 0.601644
## dis         -0.16193    0.02510  -6.452 2.64e-10 ***
## rad          0.19354    0.03432   5.639 2.89e-08 ***
## tax         -0.16589    0.03760  -4.412 1.26e-05 ***
## ptratio     -0.12863    0.01685  -7.633 1.20e-13 ***
## b            0.05636    0.01464   3.851 0.000133 ***
## lstat       -0.29950    0.02161 -13.862  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2836 on 493 degrees of freedom
## Multiple R-squared:  0.7812, Adjusted R-squared:  0.7758
## F-statistic: 146.6 on 12 and 493 DF,  p-value: < 2.2e-16
```

```r
lambda = result$lambda
df_bc = result$df_bc
lm_bc = result$lm_bc
```

```r
# Plot for Residuals vs Fitted
plot(lm_bc, which = 1)
```




```r
dwtest(lm_bc)
```

```
##
##  Durbin-Watson test
##
## data:  lm_bc
## DW = 1.0357, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
# Heteroscedasticity Test
heteroscedasticity_test(df_bc, lm_bc)
```

```
##
## Table1: Spearman Correlation Results
##         Variable Spearman_Correlation Spearman_P_Value
## rho         crim           0.17001592     1.215552e-04
## rho1          zn          -0.01467808     7.418711e-01
## rho2       indus           0.09432510     3.389871e-02
## rho3         nox           0.15734494     3.811723e-04
## rho4          rm          -0.01124769     8.007374e-01
## rho5         age           0.20294133     4.189903e-06
## rho6         dis          -0.24144300     3.813913e-08
## rho7         rad           0.19134183     1.466994e-05
## rho8         tax           0.19073461     1.563293e-05
## rho9     ptratio           0.01182515     7.907387e-01
## rho10          b          -0.04907681     2.705098e-01
## rho11      lstat           0.05917585     1.838478e-01
##
## Table2: Spearman Correlations Under Threshold 0.05
##        Variable Spearman_Correlation Spearman_P_Value
## rho        crim            0.1700159     1.215552e-04
## rho2      indus            0.0943251     3.389871e-02
## rho3        nox            0.1573449     3.811723e-04
## rho5        age            0.2029413     4.189903e-06
## rho6        dis           -0.2414430     3.813913e-08
## rho7        rad            0.1913418     1.466994e-05
## rho8        tax            0.1907346     1.563293e-05
```

We found that conducting the Box-cox transformation directly does not provide a good solution to either the heteroscedasticity(7 covariates still have significant correlation with abse) or autocorrelation (dwtest result still significant) problem.

b. Iterative Method

```r
forward <- function(dataframe) { # Define the function for iteration
  res <- resid(lm(data = dataframe, formula = medv ~ .))
  rou_hat <- (sum(res[2:n] * res[1:n-1])) / (sqrt(sum(res[2:n] ^ 2) * sum(res[1:n-1] ^ 2))) #
  new_dataframe <- dataframe
  for (t in 2:n) {
    new_dataframe[t, ] <- dataframe[t, ] - rou_hat * dataframe[t-1, ] # Original data is trans
```

```r
  }
  return(list(new_dataframe, rou_hat))
}


iterative_method <- function(df){

  # This function would return a new df after iterative correction of autocorrelation,
  # and the rou used for the transformation,
  # The model after transformation and the dw_test result would also be printed out

  rou_set <- c() # Create a collection to record the correlation coefficients at each iteratio
  df_original <- as.matrix(df) # the initial dataset
  df_original <- as.data.frame(df_original)
  rou_present <- 1 # initialze rou
  df_present <- df_original # set the iteration start to be the initial dataset
  dw_p_value <- dwtest(lm(data = df_present, formula = medv ~ .))$p.value

  while (dw_p_value <= 0.05) { # Iterate until Durbin-Watson p-value is larger than 0.05
    output <- forward(df_present)
    df_present <- output[[1]]
    rou_present <- output[[2]]
    rou_set <- c(rou_set, output[[2]])
    # Perform the Durbin-Watson test on the residuals
    dw_test <- dwtest(lm(data = df_present, formula = medv ~ .))
    dw_p_value <- dw_test$p.value
  }

df_it = df_present
lm_it = lm(data = df_it, formula = medv ~ .)
print(summary(lm_it))
print(dwtest(lm_it))

  return(list(rou_set=rou_set, df_it=df_it, lm_it=lm_it))
}

result = iterative_method(df_scale)
```

```
##
## Call:
```

```
## lm(formula = medv ~ ., data = df_it)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77197 -0.24790 -0.06291  0.16178  2.56696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.001091   0.019768  -0.055 0.956015
## crim        -0.064970   0.026858  -2.419 0.015921 *
## zn           0.082745   0.038270   2.162 0.031087 *
## indus        0.013217   0.059351   0.223 0.823871
## nox         -0.193279   0.062998  -3.068 0.002273 **
## rm           0.355589   0.028140  12.637  < 2e-16 ***
## age         -0.085449   0.040258  -2.123 0.034291 *
## dis         -0.325140   0.059538  -5.461 7.52e-08 ***
## rad          0.304280   0.078069   3.898 0.000111 ***
## tax         -0.322934   0.081831  -3.946 9.09e-05 ***
## ptratio     -0.177572   0.039861  -4.455 1.04e-05 ***
## b            0.100848   0.030054   3.356 0.000853 ***
## lstat       -0.279247   0.038133  -7.323 9.95e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4447 on 493 degrees of freedom
## Multiple R-squared:  0.6289, Adjusted R-squared:  0.6199
## F-statistic: 69.63 on 12 and 493 DF,  p-value: < 2.2e-16
##
##
##  Durbin-Watson test
##
## data:  lm_it
## DW = 2.0082, p-value = 0.4695
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
rou = result$rou_set
df_it = result$df_it
lm_it = result$lm_it
```

```r
# Conduct Heteroscedasticity test on the data after autocorrelation correction
heteroscedasticity_test(df_it, lm_it)
```

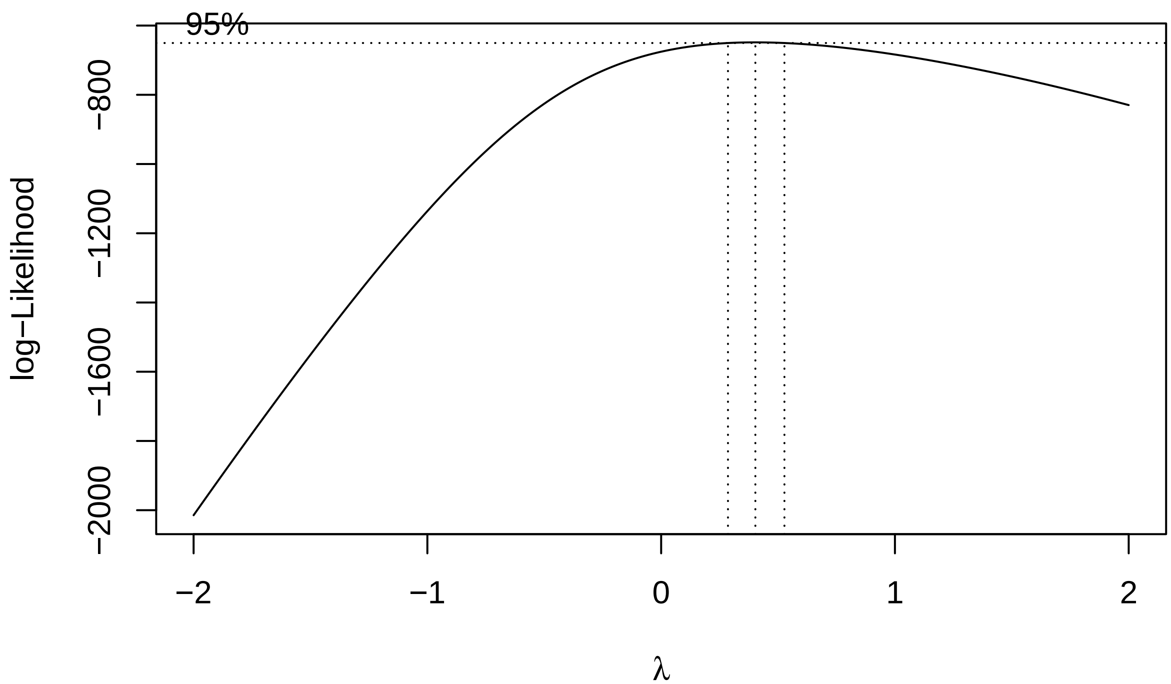
```
##
## Table1: Spearman Correlation Results
##       Variable Spearman_Correlation Spearman_P_Value
## rho       crim           0.03039600      0.494978181
## rho1        zn           0.07995918      0.072324943
## rho2     indus          -0.01200690      0.787598658
## rho3       nox           0.04382456      0.325192293
## rho4        rm           0.11697636      0.008472971
## rho5       age           0.08372550      0.059835181
## rho6       dis          -0.16839168      0.000141393
## rho7       rad           0.09104109      0.040647610
## rho8       tax           0.09619618      0.030499083
## rho9   ptratio          -0.10686928      0.016176427
## rho10        b           0.02775898      0.533285394
## rho11    lstat          -0.10806568      0.015051351
##
## Table2: Spearman Correlations Under Threshold 0.05
##       Variable Spearman_Correlation Spearman_P_Value
## rho4        rm           0.11697636      0.008472971
## rho6       dis          -0.16839168      0.000141393
## rho7       rad           0.09104109      0.040647610
## rho8       tax           0.09619618      0.030499083
## rho9   ptratio          -0.10686928      0.016176427
## rho11    lstat          -0.10806568      0.015051351
```

We found that the autocorrelation has been removed by the DW-test standards, while the problem of heteroscedasticity remains. We thus apply Box-Cox transformation to handle that.

c. Apply Box-Cox transformation to the data after iteration method

```r
result = boxcox.fitting(df_it)
```

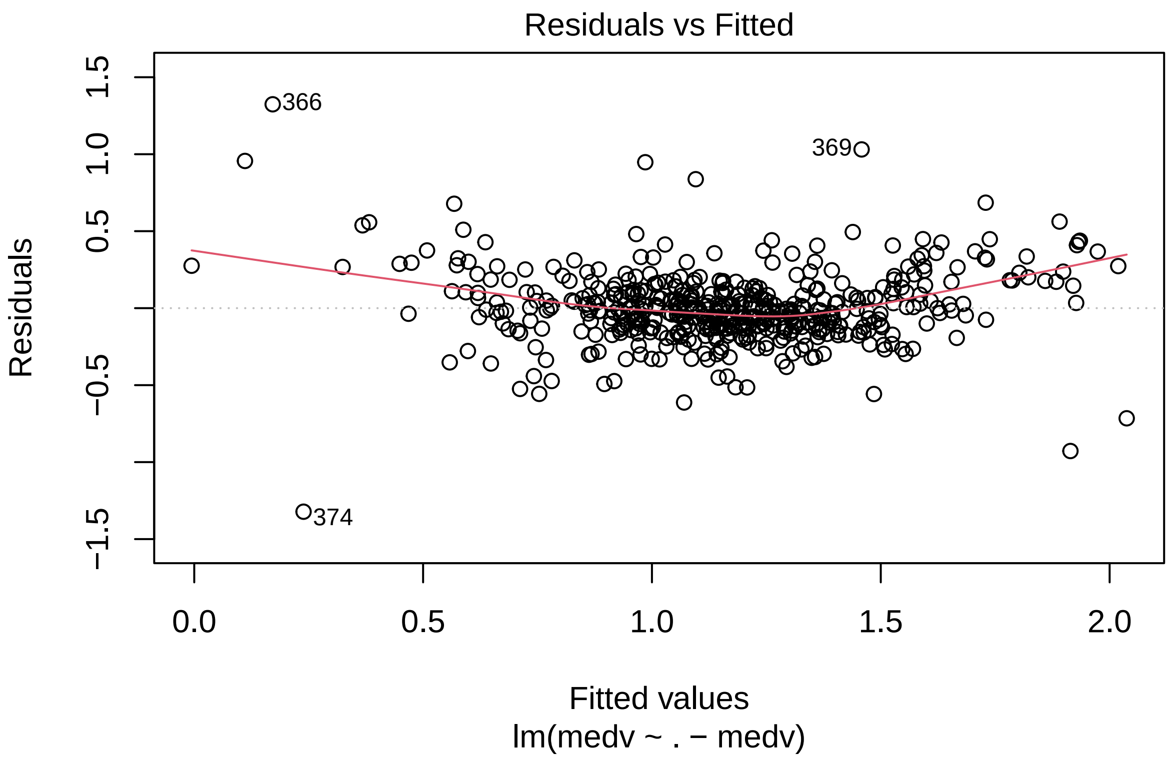
```
## Note: conducted y = y - 1.1 min(y) since there are negative values in the 'medv' variable.
```




```
##
## Call:
## lm(formula = medv ~ . - medv, data = df_bc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.32203 -0.12291 -0.01708  0.10368  1.32413
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.162057   0.010529 110.368  < 2e-16 ***
## crim        -0.044856   0.014305  -3.136 0.001817 **
## zn           0.038034   0.020383   1.866 0.062644 .
## indus        0.007624   0.031612   0.241 0.809525
## nox         -0.087791   0.033554  -2.616 0.009159 **
## rm           0.177734   0.014988  11.859  < 2e-16 ***
## age         -0.046244   0.021442  -2.157 0.031514 *
## dis         -0.155893   0.031711  -4.916 1.20e-06 ***
## rad          0.160964   0.041581   3.871 0.000123 ***
## tax         -0.182466   0.043585  -4.186 3.36e-05 ***
## ptratio     -0.081648   0.021231  -3.846 0.000136 ***
## b            0.049894   0.016007   3.117 0.001934 **
## lstat       -0.177611   0.020310  -8.745  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2368 on 493 degrees of freedom
## Multiple R-squared:  0.6397, Adjusted R-squared:  0.6309
## F-statistic: 72.94 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
lambda_it_bc = result$lambda
df_it_bc = result$df_bc
lm_it_bc = result$lm_bc

# Creating a plot for Residuals vs Fitted
plot(lm_it_bc, which = 1)
```




```
# Performing Durbin-Watson test for autocorrelation
dwtest(lm_it_bc)
```

```
##
##  Durbin-Watson test
##
## data:  lm_it_bc
## DW = 2.1791, p-value = 0.9681
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
# Executing heteroscedasticity test
heteroscedasticity_test(df_it_bc, lm_it_bc)
```

```
##
## Table1: Spearman Correlation Results
##         Variable Spearman_Correlation Spearman_P_Value
## rho         crim          0.066657466     1.342600e-01
## rho1          zn          0.051632840     2.463123e-01
## rho2       indus          0.011882363     7.897498e-01
## rho3         nox          0.076179763     8.692034e-02
## rho4          rm          0.041870835     3.471255e-01
## rho5         age          0.096735396     2.957530e-02
## rho6         dis         -0.188613692     1.948976e-05
## rho7         rad          0.100683711     2.351385e-02
## rho8         tax          0.131972845     2.936261e-03
## rho9     ptratio         -0.073367479     9.924985e-02
## rho10          b          0.008136189     8.551359e-01
## rho11      lstat         -0.038981933     3.814257e-01
##
## Table2: Spearman Correlations Under Threshold 0.05
##        Variable Spearman_Correlation Spearman_P_Value
## rho5        age            0.0967354     2.957530e-02
## rho6        dis           -0.1886137     1.948976e-05
## rho7        rad            0.1006837     2.351385e-02
## rho8        tax            0.1319728     2.936261e-03
```

As we can see, even after performing iteration method and box-cox transformation, some variables still show quite high correlation with the abse of the model.

4. Apply Iterative Method and Box-Cox transformation to the dataset after variable selection

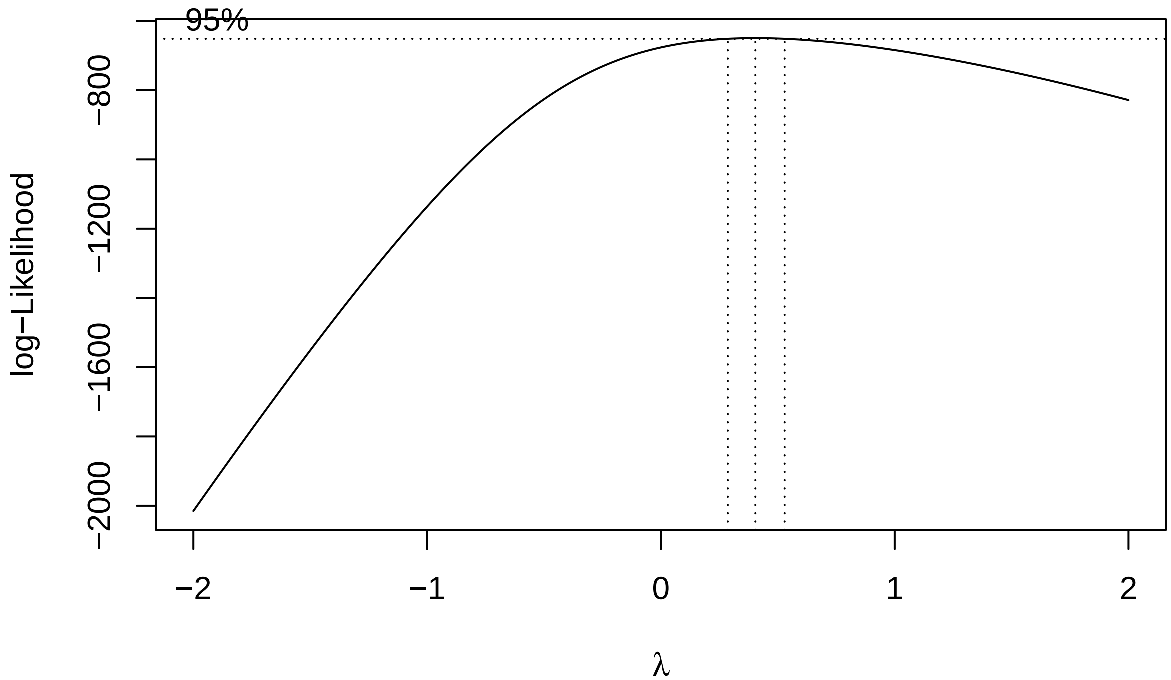
```r
it_result = iterative_method(df_reduced)
```

```
##
## Call:
## lm(formula = medv ~ ., data = df_it)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.78967 -0.24590 -0.05054  0.16713  2.50206
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.001114   0.019815  -0.056 0.955178
## crim        -0.067895   0.026868  -2.527 0.011815 *
## zn           0.087214   0.038062   2.291 0.022364 *
## nox         -0.215184   0.059980  -3.588 0.000367 ***
## rm           0.345006   0.027722  12.445  < 2e-16 ***
## dis         -0.297648   0.056615  -5.257 2.18e-07 ***
## rad          0.314323   0.076866   4.089 5.05e-05 ***
## tax         -0.324810   0.076400  -4.251 2.54e-05 ***
## ptratio     -0.180967   0.039584  -4.572 6.12e-06 ***
## b            0.095579   0.030045   3.181 0.001559 **
## lstat       -0.302119   0.036407  -8.298 1.01e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4457 on 495 degrees of freedom
## Multiple R-squared:  0.6248, Adjusted R-squared:  0.6173
## F-statistic: 82.44 on 10 and 495 DF,  p-value: < 2.2e-16
##
##
##  Durbin-Watson test
##
## data:  lm_it
## DW = 2.0327, p-value = 0.5886
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
rou_reduced = it_result$rou_set

bc_result = boxcox.fitting(it_result$df_it)
```

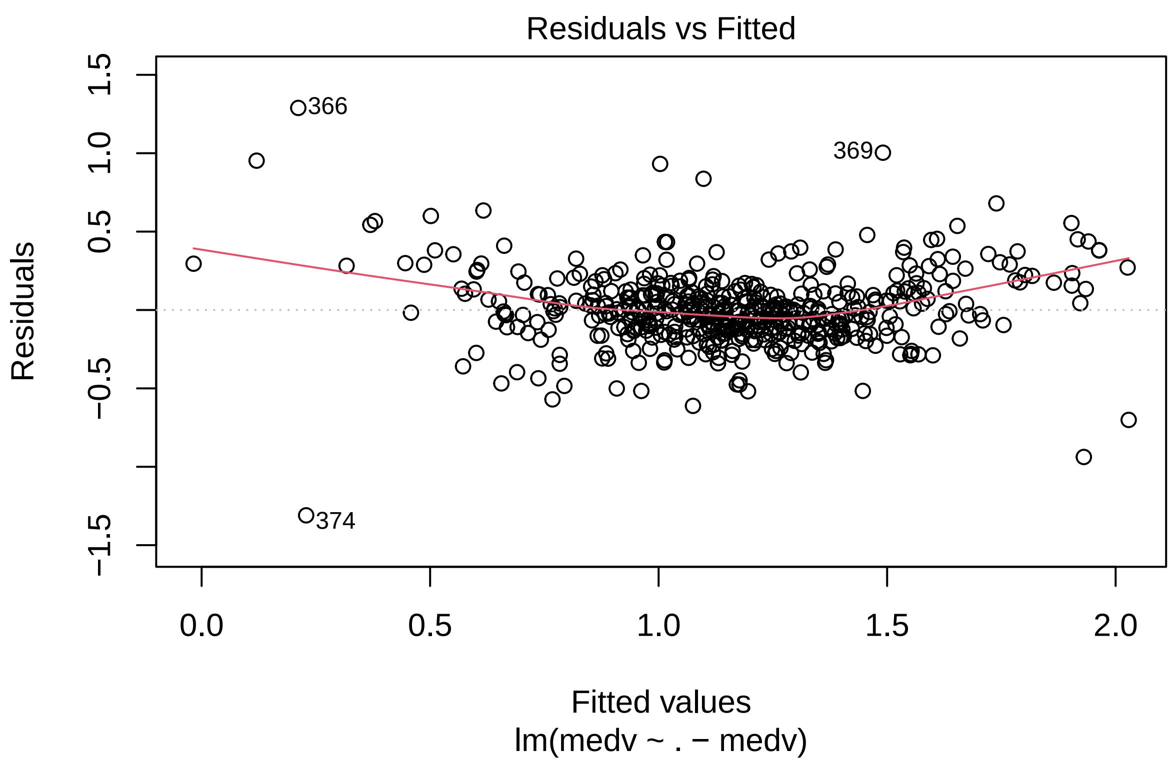
```
## Note: conducted y = y - 1.1 min(y) since there are negative values in the 'medv' variable.
```




```
##
## Call:
## lm(formula = medv ~ . - medv, data = df_bc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30968 -0.12389 -0.01372  0.10437  1.28881
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.16714    0.01056 110.565  < 2e-16 ***
## crim        -0.04632    0.01431  -3.236  0.00129 **
## zn           0.04049    0.02028   1.997  0.04640 *
## nox         -0.09939    0.03195  -3.110  0.00198 **
## rm           0.17206    0.01477  11.651  < 2e-16 ***
## dis         -0.14112    0.03016  -4.679 3.73e-06 ***
## rad          0.16623    0.04095   4.059 5.72e-05 ***
## tax         -0.18314    0.04070  -4.500 8.49e-06 ***
## ptratio     -0.08339    0.02109  -3.954 8.79e-05 ***
## b            0.04699    0.01601   2.936  0.00348 **
## lstat       -0.18976    0.01939  -9.784  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.2375 on 495 degrees of freedom
## Multiple R-squared:  0.6353, Adjusted R-squared:  0.6279
## F-statistic: 86.22 on 10 and 495 DF,  p-value: < 2.2e-16
```

```r
lambda_reduced = bc_result$lambda
df_reduced_bc = bc_result$df_bc
lm_reduced_bc = bc_result$lm_bc

# plot residuals vs fitted values
plot(lm_reduced_bc, which = 1)
```




```r
print(dwtest(lm_reduced_bc))
```

```
##
##  Durbin-Watson test
##
## data:  lm_reduced_bc
## DW = 2.2, p-value = 0.9827
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
heteroscedasticity_test(df_reduced_bc, lm_reduced_bc)
```

```
##
```

```
## Table1: Spearman Correlation Results
##      Variable Spearman_Correlation Spearman_P_Value
## rho      crim          0.068438935     0.1241452688
## rho1       zn          0.053392713     0.2305573752
## rho2      nox          0.076745282     0.0845964945
## rho3       rm          0.054072045     0.2245876234
## rho4      dis         -0.175953557     0.0000691255
## rho5      rad          0.105975073     0.0170942981
## rho6      tax          0.133985670     0.0025268503
## rho7  ptratio         -0.064841584     0.1452541342
## rho8        b         -0.004553671     0.9186141992
## rho9    lstat         -0.033863623     0.4470779572
##
## Table2: Spearman Correlations Under Threshold 0.05
##      Variable Spearman_Correlation Spearman_P_Value
## rho4      dis           -0.1759536      6.91255e-05
## rho5      rad            0.1059751      1.70943e-02
## rho6      tax            0.1339857      2.52685e-03
```

Based on the significance of the coefficients, we can tentatively conclude that the variable selection performed earlier is reasonable. It is unlikely that previously non-significant variables have become significant after model transformation, and previously significant variables have become non-significant after model transformation.

In summary, the changes we have made to the dataset and the model till now include:

i. standardize the dataset
ii. delete age、indus
iii. df_scale[t,] := df_scale[t,] - 0.4909 * df_scale[t-1,]
iv. df_scale$medv := df_scale$medv - 1.1 * (-2.415)
v. df_scale$medv := ((df_scale$medv) ^0.404 - 1) / 0.404

5. Outliers and Influential Points

a. Identify influential points

```r
cook_distance <- cooks.distance(lm_it_bc) # Calculate Cook Distance of all observations
index_influential <- which(cook_distance > 4/(nrow(df_it_bc) - (ncol(df_it_bc) - 1) - 1)) # I
cat(index_influential)
```

```
## 8 55 65 66 142 148 158 162 165 167 168 188 196 206 215 229 254 255 270 343 365 366 368 369
```

b. Identify outliers

```r
r <- rstandard(lm_it_bc) # Calculate the SRE for all observations
p <- ncol(df_it_bc) - 1 + 1 # number of covariates +1
n <- nrow(df_it_bc) # sample size
F <- (n - p - 1) * r ^ 2 / (n - p -r ^ 2) # calulate the F-statistic for outliers
p.value <- 1 - pf(F, 1, n - p - 1) # calculate the p-value for the F-test of the outliers
index_outlier <- which(p.value < 0.05)
cat(index_outlier) # print out the outliers
```

```
## 8 66 142 162 165 167 206 215 255 270 343 365 366 368 369 371 372 373 374 375 376 413 416 42
```

c. Delete observations that fall into the intersection of outliers and influential points

```r
df_full_final <- df_it_bc
bad_index <- intersect(index_influential, index_outlier)
length(bad_index)
```

```
## [1] 23
```

```r
cat(bad_index) # output the index of the observations that are both outliers and influential p
```

```
## 8 66 142 162 165 167 206 215 255 270 343 365 366 368 369 372 373 374 375 376 413 416 420
```

```r
df_full_final <- df_full_final[-bad_index,]
lm_full_final <- lm(medv~., data = df_full_final) # estimate the model after deleting the the
summary(lm_full_final)
```

```
##
## Call:
## lm(formula = medv ~ ., data = df_full_final)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55246 -0.09777 -0.01359  0.08772  0.64674
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.156256   0.007676 150.632  < 2e-16 ***
## crim        -0.047538   0.010512  -4.522 7.76e-06 ***
## zn           0.027879   0.015328   1.819  0.06957 .
## indus       -0.001613   0.022942  -0.070  0.94397
## nox         -0.057214   0.024329  -2.352  0.01910 *
## rm           0.262525   0.012870  20.398  < 2e-16 ***
## age         -0.084622   0.016088  -5.260 2.19e-07 ***
```

```
## dis         -0.129992   0.023210  -5.601 3.64e-08 ***
## rad          0.097617   0.030460   3.205  0.00144 **
## tax         -0.142868   0.031836  -4.488 9.07e-06 ***
## ptratio     -0.063153   0.015481  -4.079 5.31e-05 ***
## b            0.057248   0.011848   4.832 1.83e-06 ***
## lstat       -0.108952   0.016561  -6.579 1.27e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1684 on 470 degrees of freedom
## Multiple R-squared:  0.781,  Adjusted R-squared:  0.7754
## F-statistic: 139.7 on 12 and 470 DF,  p-value: < 2.2e-16
```

After deleting the "bad points", the significance of coefficients improved.

d. Then apply the same steps to the reduced model

```r
cook_distance <- cooks.distance(lm_reduced_bc) # Calculate Cook Distance of all observations
index_influential <- which(cook_distance > 4/(nrow(df_reduced_bc) - (ncol(df_reduced_bc) - 1)
cat(index_influential)
```

```
## 55 65 66 142 148 158 162 165 167 168 188 196 206 215 254 255 270 343 365 366 368 369 372 37
```

```r
r <- rstandard(lm_reduced_bc) # Calculate the SRE for all observations
p <- ncol(df_reduced_bc) - 1 + 1 # number of covariates +1
n <- nrow(df_reduced_bc) # sample size
F <- (n - p - 1) * r ^ 2 / (n - p -r ^ 2) # calulate the F-statistic for outliers
p.value <- 1 - pf(F, 1, n - p - 1) # calculate the p-value for the F-test of the outliers
index_outlier <- which(p.value < 0.05)
cat(index_outlier) # print out the outliers
```

```
## 66 142 159 162 165 167 206 215 229 255 270 343 365 366 368 369 371 372 373 374 375 376 413
```

```r
df_reduced_final <- df_reduced_bc
bad_index <- intersect(index_influential, index_outlier)
length(bad_index)
```

```
## [1] 23
```

```r
cat(bad_index) # output the index of the observations that are both outliers and influential p
```

```
## 66 142 162 165 167 206 215 255 270 343 365 366 368 369 372 373 374 375 376 413 416 420 506
```

```r
df_reduced_final <- df_reduced_final[-bad_index,]
lm_reduced_final <- lm(medv~., data = df_reduced_final) # estimate the model after deleting th
summary(lm_reduced_final)
```

```
##
## Call:
## lm(formula = medv ~ ., data = df_reduced_final)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.44584 -0.10984 -0.00794  0.09559  0.60587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.16329    0.00783 148.568  < 2e-16 ***
## crim        -0.04934    0.01071  -4.607 5.27e-06 ***
## zn           0.03422    0.01549   2.210 0.027584 *
## nox         -0.07889    0.02358  -3.345 0.000889 ***
## rm           0.24733    0.01281  19.304  < 2e-16 ***
## dis         -0.09860    0.02245  -4.391 1.39e-05 ***
## rad          0.10908    0.03051   3.575 0.000386 ***
## tax         -0.15265    0.03020  -5.054 6.18e-07 ***
## ptratio     -0.06791    0.01564  -4.342 1.73e-05 ***
## b            0.05233    0.01204   4.346 1.70e-05 ***
## lstat       -0.13774    0.01573  -8.758  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1718 on 472 degrees of freedom
## Multiple R-squared:  0.7687, Adjusted R-squared:  0.7638
## F-statistic: 156.8 on 10 and 472 DF,  p-value: < 2.2e-16
```

5. Compare the MSE between final models

Consider the reduced model after model selection: `medv \~ crim + zn + nox + rm + dis + rad + tax +`

```r
target_sd = train_sd['medv']
model = lm(formula=medv~crim + zn + nox + rm + dis + rad + tax + ptratio + b + lstat, data=df_
model_pre <- predict(model, newdata = df_test)
mse <- mean((df_test$medv - model_pre)^2) * (target_sd^2)
cat("Prediction mse for reduced model:", mse)
```

```
## Prediction mse for reduced model: 23.20066
```

Consider the full model medv \~.

```r
target_sd = train_sd['medv']
model = lm(formula=medv~., data=df_train)
model_pre <- predict(model, newdata = df_test)
mse <- mean((df_test$medv - model_pre)^2) * (target_sd^2)

adj_mse <- mean((df_test$medv - model_pre)^2) * (target_sd^2)
cat("Prediction mse for full model:", adj_mse)
```

```
## Prediction mse for full model: 23.44789
```