Group 2 INF2190 Final Project

Junwei Shen

2023-11-20

```r
# Reading necessary libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(dplyr)
library(ggplot2)
library(RColorBrewer)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(tree)
library(rpart.plot)
```

```
## Loading required package: rpart
```

```r
library(ISLR2)
```

1. EDA and Data Visualization

```r
# Glimpse of the dataset
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

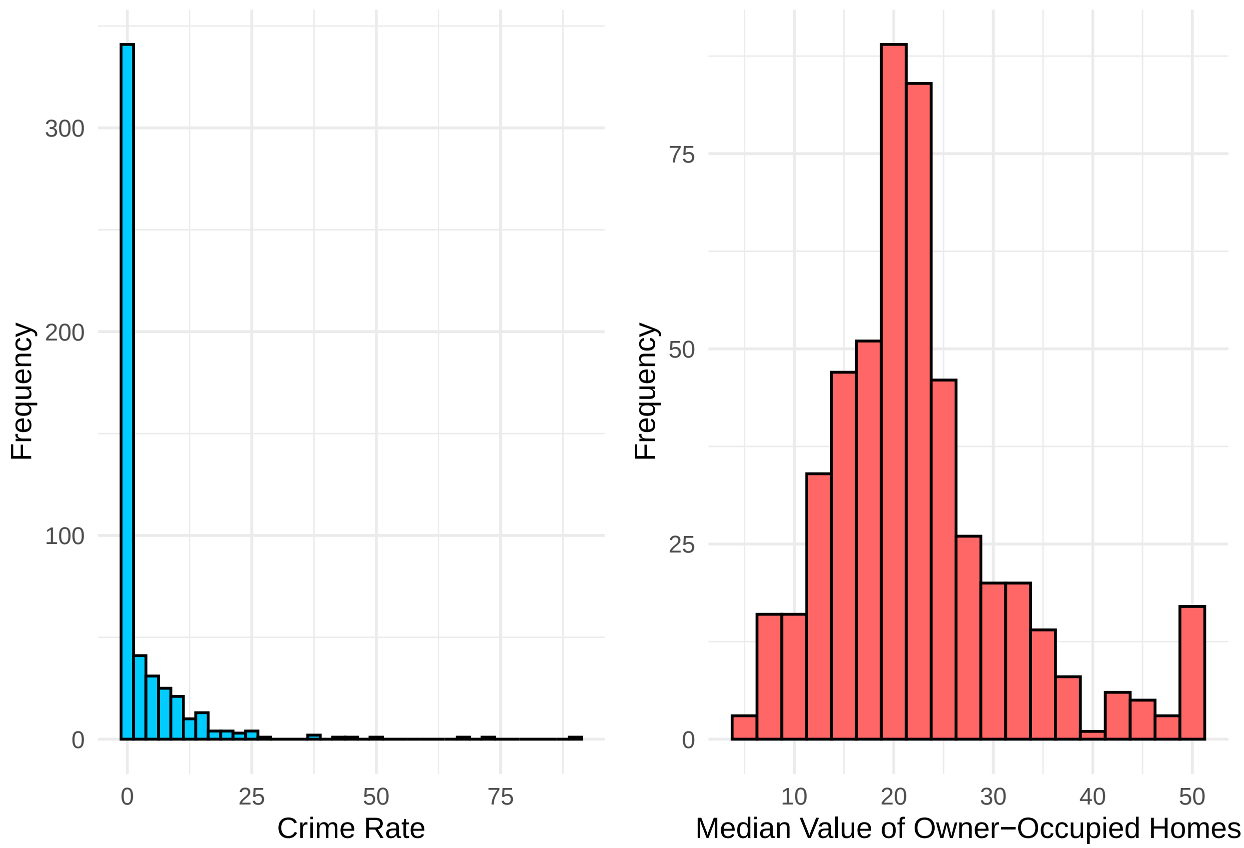
```r
# Investigate if there are any missing/null values
sum(is.na(Boston))
```

```
## [1] 0
```

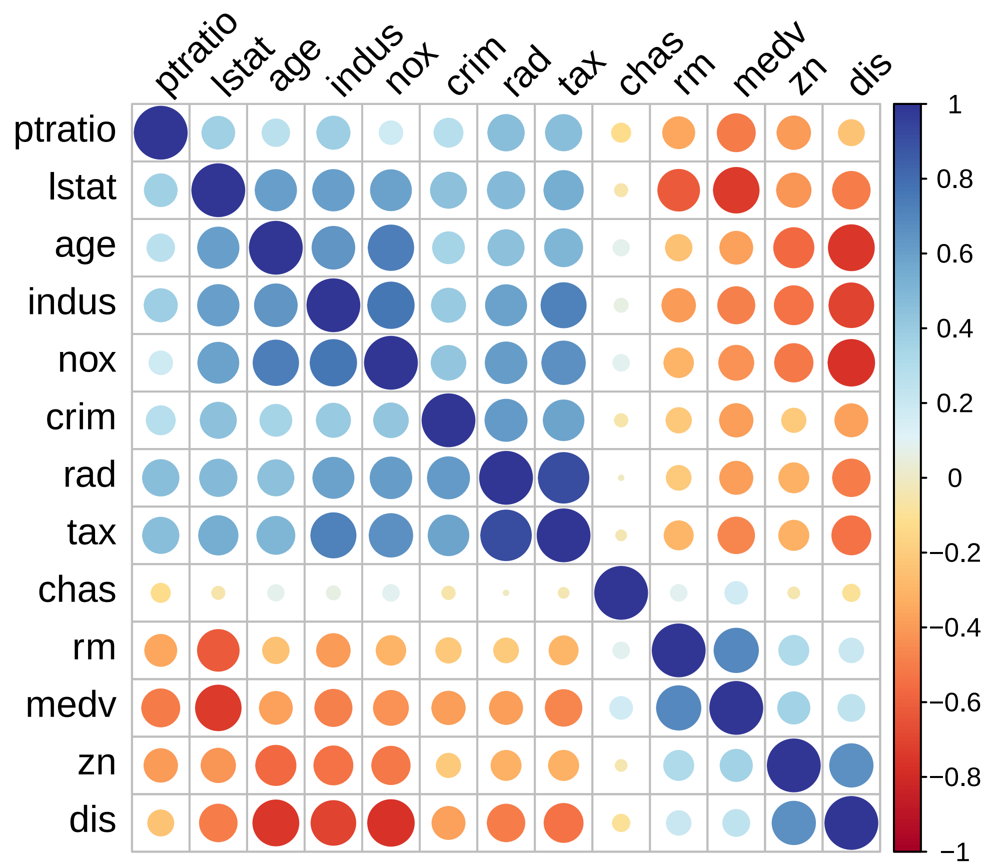
```r
# Variable Summary Stats
summary(Boston)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          lstat      
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95  
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97  
##       medv      
##  Min.   : 5.00  
##  1st Qu.:17.02  
##  Median :21.20  
##  Mean   :22.53  
##  3rd Qu.:25.00  
##  Max.   :50.00
```

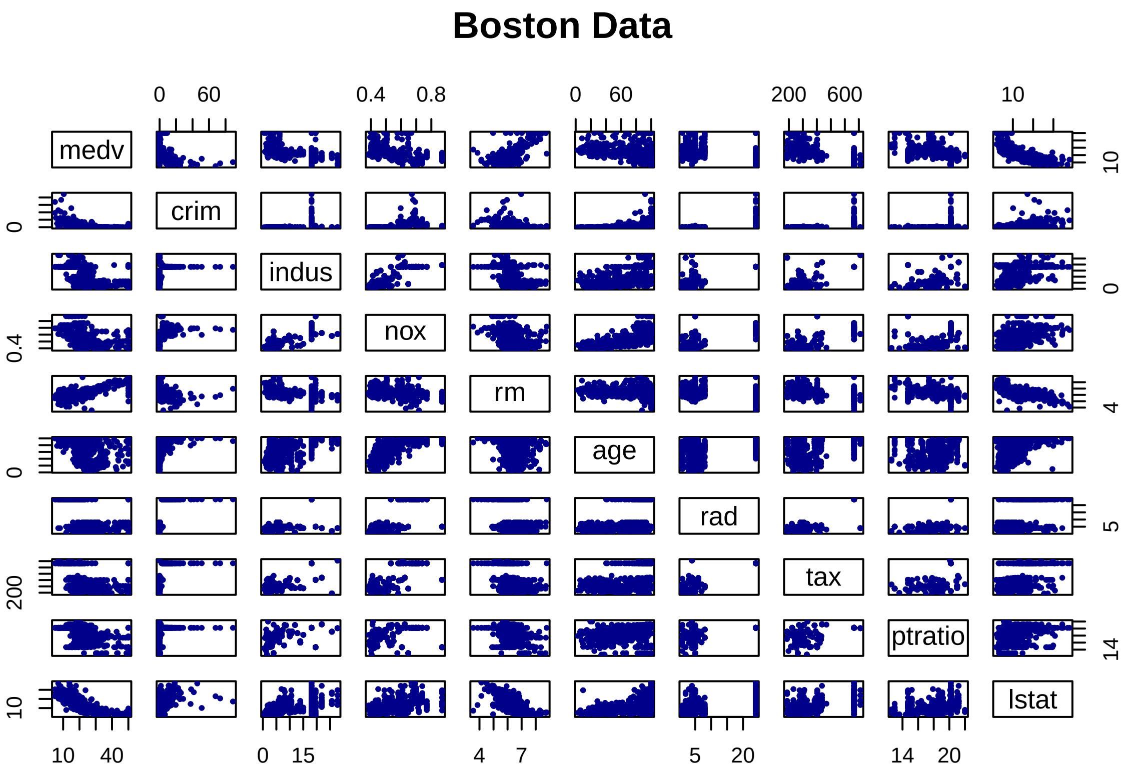
```r
# Histogram of Crime Rate
h1 <- ggplot(Boston, aes(x = crim)) +
    geom_histogram(binwidth = 2.5, fill = "#00CCFF", color = "black") +
    labs(x = "Crime Rate",
         y = "Frequency") +
    theme_minimal()
# Histogram of Median Value of Owner-Occupied Homes
h2 <- ggplot(Boston, aes(x = medv)) +
    geom_histogram(binwidth = 2.5, fill = "#FF6666", color = "black") +
    labs(x = "Median Value of Owner-Occupied Homes",
         y = "Frequency") +
    theme_minimal()

grid.arrange(h1, h2, nrow = 1, ncol = 2)
```



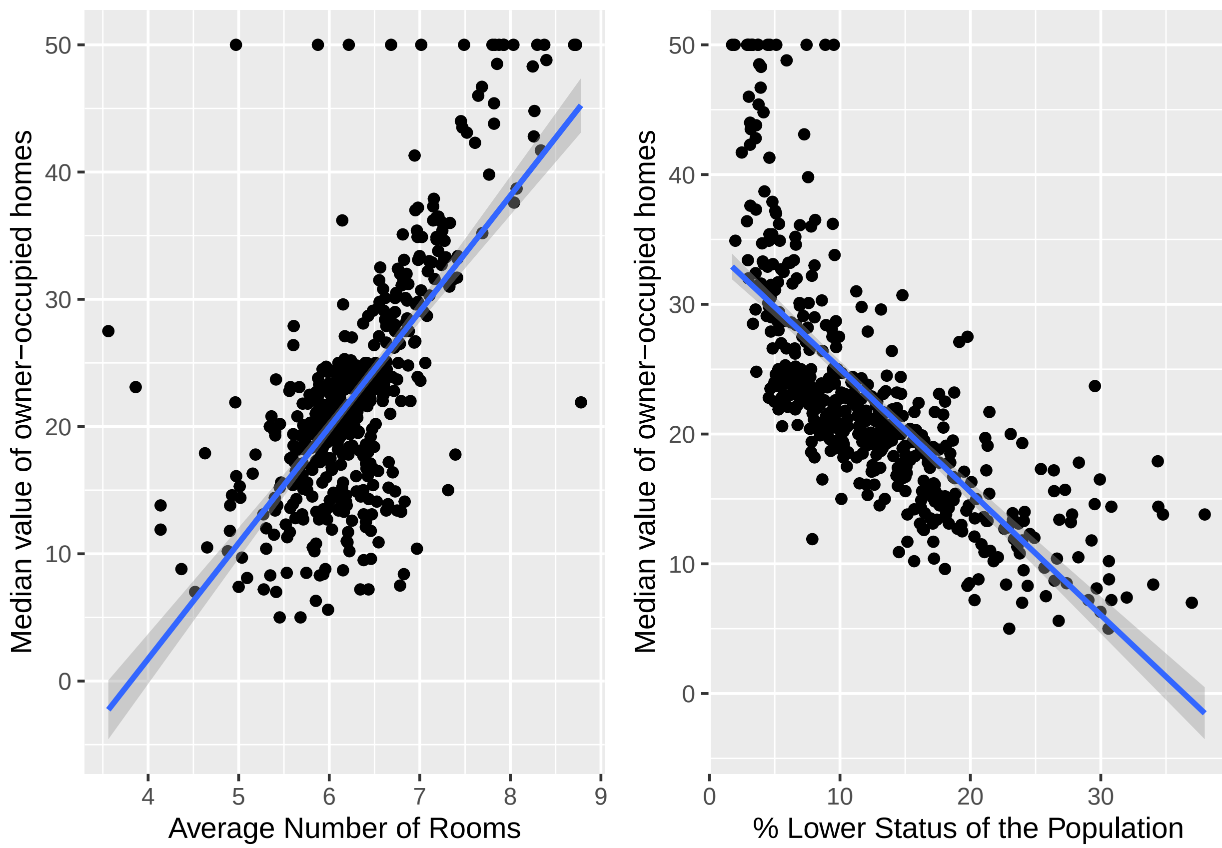

```r
# Heatmap
corr_matrix <- cor(Boston)
corrplot(corr_matrix, method = "circle",
         tl.col = "black", # Change text (label) color
         tl.srt = 45,     # Rotate text labels
         tl.cex = 1.2,    # Change text size
         col = colorRampPalette(brewer.pal(10, "RdYlBu"))(200), # Change color palette
         order = "hclust", # Order variables using hierarchical clustering
)
```



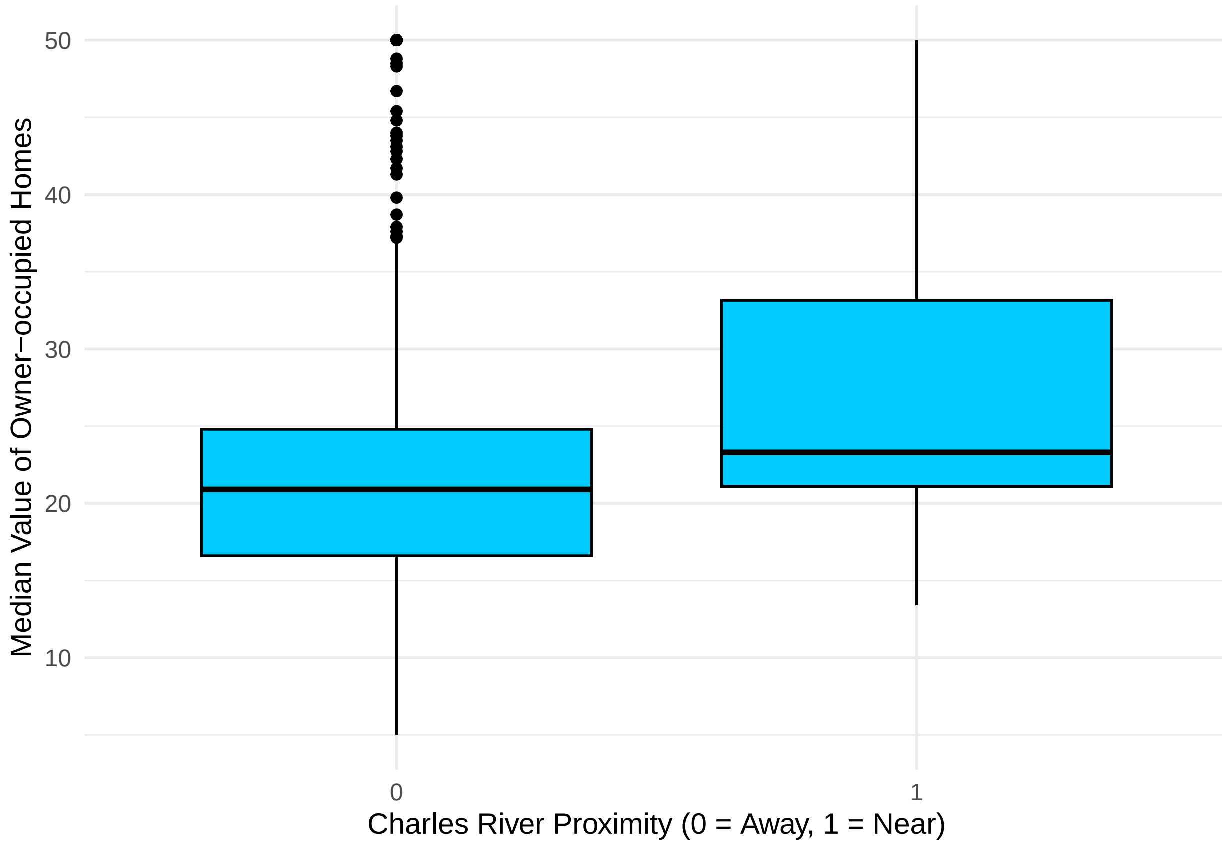

```r
# correlation plot
pairs(~ medv + crim + indus + nox + rm + age + rad + tax + ptratio + lstat,
      data = Boston,
      main = "Boston Data",
      pch = 19,          # Use solid dots
      cex = 0.4,         # Smaller dot size
      col = "darkblue"
     )
```




```r
# Scatterplot of Rooms vs Median value of owner-occupied homes
s1 <- ggplot(Boston, aes(x=rm, y=medv)) + geom_point() + geom_smooth(method="lm") + labs(x="Average Numb

# Scatterplot of % Lower Status of the Population vs Median value of owner-occupied homes
s2 <- ggplot(Boston, aes(x=lstat, y=medv)) + geom_point() + geom_smooth(method="lm") + labs(x="% Lower S

grid.arrange(s1, s2, nrow = 1, ncol = 2)
```




```r
# Convert "chas" integer variable into categorical variable, since its is a dummy variable (1 if tract
Boston_copy <- Boston
Boston_copy$chas <- factor(Boston$chas, levels = c(0, 1), labels = c("0", "1"))
# Boxplot of Median Value of Owner-occupied Homes by Charles River Proximity
ggplot(Boston_copy, aes(x = factor(chas), y = medv)) +
    geom_boxplot(fill = "#00CCFF", color = "black") +
    labs(x = "Charles River Proximity (0 = Away, 1 = Near)",
         y = "Median Value of Owner-occupied Homes") +
    theme_minimal()
```




```r
# Unpaired t-test, used for comparing two different, independent groups. Two-sample t-test, used when c
t.test(medv ~ chas, data = Boston, paired = FALSE, var.equal = FALSE, conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  medv by chas
## t = -3.1133, df = 36.876, p-value = 0.003567
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -10.476831  -2.215483
## sample estimates:
## mean in group 0 mean in group 1
##        22.09384        28.44000
```

```r
# The resulted p-value of 0.003567 is less than an alpha = 0.05, meaning that we have evidence against
```

2. Linear Regression

```r
# Building a simple linear regression to investigate the outcome medv
# Selecting because rm and lstat have the highest correlation values (positive and negative)
simple_model_1 <- lm(medv ~ rm, data=Boston)
summary(simple_model_1)
```

```
##
## Call:
## lm(formula = medv ~ rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.346  -2.547   0.090   2.986  39.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -34.671      2.650  -13.08   <2e-16 ***
## rm             9.102      0.419   21.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.616 on 504 degrees of freedom
## Multiple R-squared:  0.4835, Adjusted R-squared:  0.4825
## F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
simple_model_2 <- lm(medv ~ lstat, data=Boston)
summary(simple_model_2)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
# Building a Generalized Linear Model, assuming a Gaussian, building an "overfitting" model and remove
glm_model_full <- glm(medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + ls
summary(glm_model_full)
```

```
##
## Call:
## glm(formula = medv ~ crim + zn + indus + chas + nox + rm + age +
##     dis + rad + tax + ptratio + lstat, family = gaussian, data = Boston_copy)
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.617270   4.936039   8.431 3.79e-16 ***
```

```
## crim          -0.121389   0.033000  -3.678 0.000261 ***
## zn             0.046963   0.013879   3.384 0.000772 ***
## indus          0.013468   0.062145   0.217 0.828520
## chas1          2.839993   0.870007   3.264 0.001173 **
## nox          -18.758022   3.851355  -4.870 1.50e-06 ***
## rm             3.658119   0.420246   8.705  < 2e-16 ***
## age            0.003611   0.013329   0.271 0.786595
## dis           -1.490754   0.201623  -7.394 6.17e-13 ***
## rad            0.289405   0.066908   4.325 1.84e-05 ***
## tax           -0.012682   0.003801  -3.337 0.000912 ***
## ptratio       -0.937533   0.132206  -7.091 4.63e-12 ***
## lstat         -0.552019   0.050659 -10.897  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 23.02113)
##
##     Null deviance: 42716  on 505  degrees of freedom
## Residual deviance: 11349  on 493  degrees of freedom
## AIC: 3037.8
##
## Number of Fisher Scoring iterations: 2
```

```r
# indus, age shows non-significant p-value, can be removed
```

```r
# Backward Elimination Approach starting from full model
# Building new glm model without indus, age attributes, finding out that zn attribute is not significan
# I have also considered the correlation between rm and lstat attributes, and fit the correlation effec
glm_model_1 <- glm(medv ~ crim + chas + nox + rm + dis + rad + tax + ptratio + lstat + rm:lstat, family=
summary(glm_model_1)
```

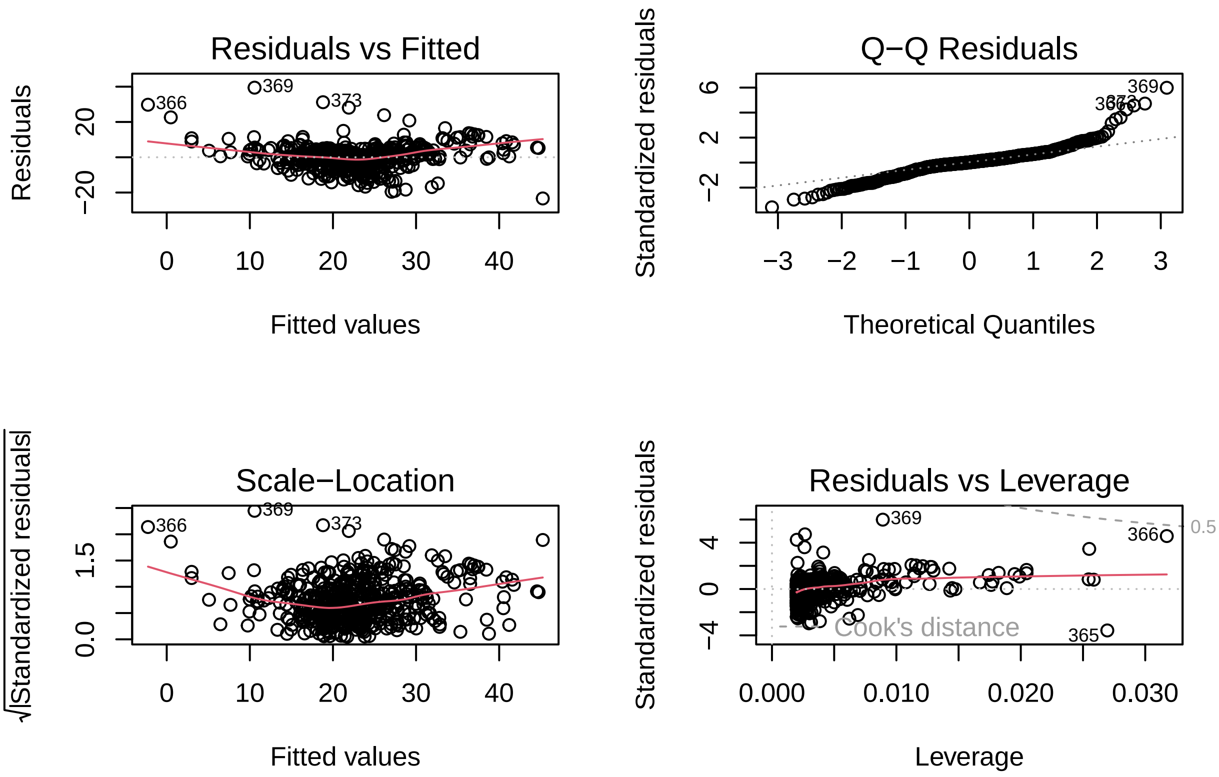
```
##
## Call:
## glm(formula = medv ~ crim + chas + nox + rm + dis + rad + tax +
##     ptratio + lstat + rm:lstat, family = gaussian, data = Boston_copy)
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.852248   4.987027   1.374  0.17006
## crim          -0.160071   0.028600  -5.597 3.62e-08 ***
## chas1          2.214348   0.748139   2.960  0.00323 **
## nox          -14.039663   3.095514  -4.535 7.22e-06 ***
## rm             8.208553   0.475853  17.250  < 2e-16 ***
## dis           -1.053185   0.141087  -7.465 3.79e-13 ***
## rad            0.284982   0.055216   5.161 3.56e-07 ***
## tax           -0.009472   0.002892  -3.275  0.00113 **
## ptratio       -0.758667   0.108730  -6.978 9.68e-12 ***
## lstat          1.938032   0.189128  10.247  < 2e-16 ***
## rm:lstat      -0.432623   0.032153 -13.455  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 17.17935)
```

```
##
##     Null deviance: 42716.3  on 505  degrees of freedom
## Residual deviance:  8503.8  on 495  degrees of freedom
## AIC: 2887.8
##
## Number of Fisher Scoring iterations: 2
```

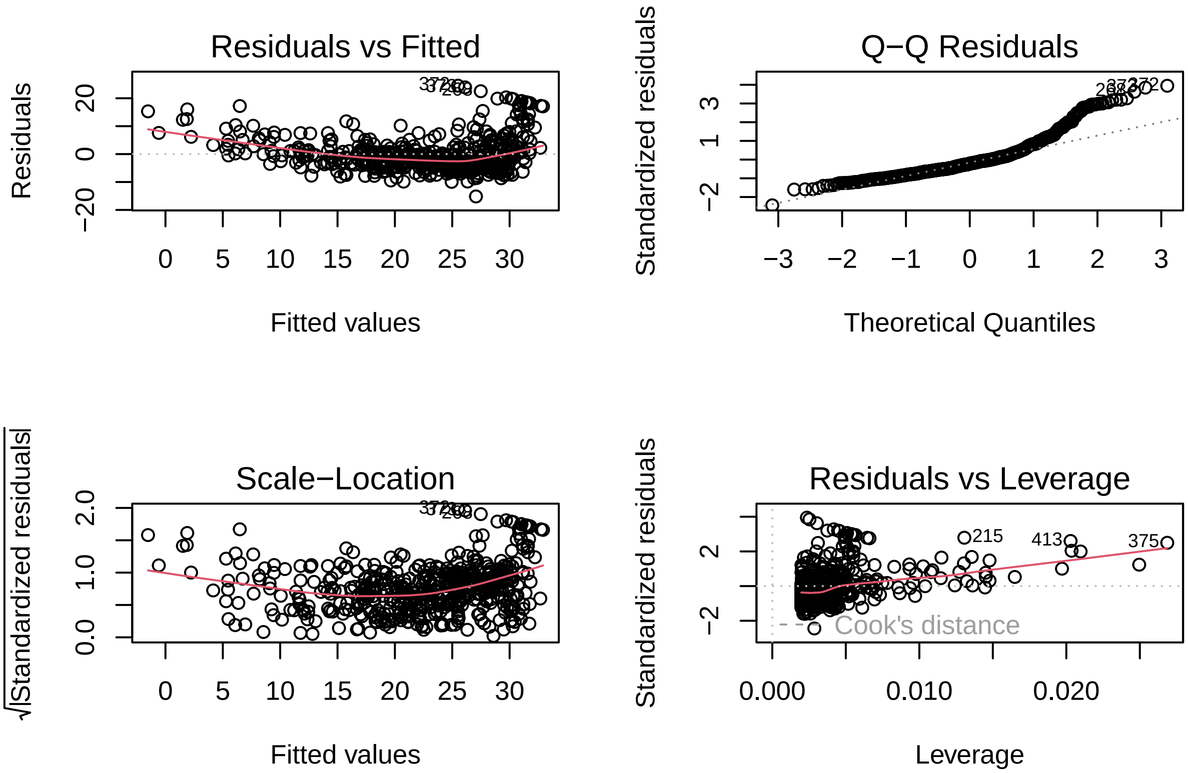
```r
# Forward Addition Approach starting from two most significant attributes, adding more attributes from
glm_model_2 <- glm(medv ~  rm + lstat + rm:lstat + crim + dis + ptratio, family=gaussian, data=Boston_c
summary(glm_model_2)
```

```
##
## Call:
## glm(formula = medv ~ rm + lstat + rm:lstat + crim + dis + ptratio,
##     family = gaussian, data = Boston_copy)
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.54955    4.24616  -2.484   0.0133 *
## rm            8.88259    0.48909  18.161  < 2e-16 ***
## lstat         2.01865    0.19677  10.259  < 2e-16 ***
## crim         -0.12371    0.02622  -4.719 3.09e-06 ***
## dis          -0.67833    0.10922  -6.211 1.11e-09 ***
## ptratio      -0.56850    0.10255  -5.544 4.80e-08 ***
## rm:lstat     -0.45483    0.03311 -13.737  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 18.89673)
##
##     Null deviance: 42716.3  on 505  degrees of freedom
## Residual deviance:  9429.5  on 499  degrees of freedom
## AIC: 2932
##
## Number of Fisher Scoring iterations: 2
```

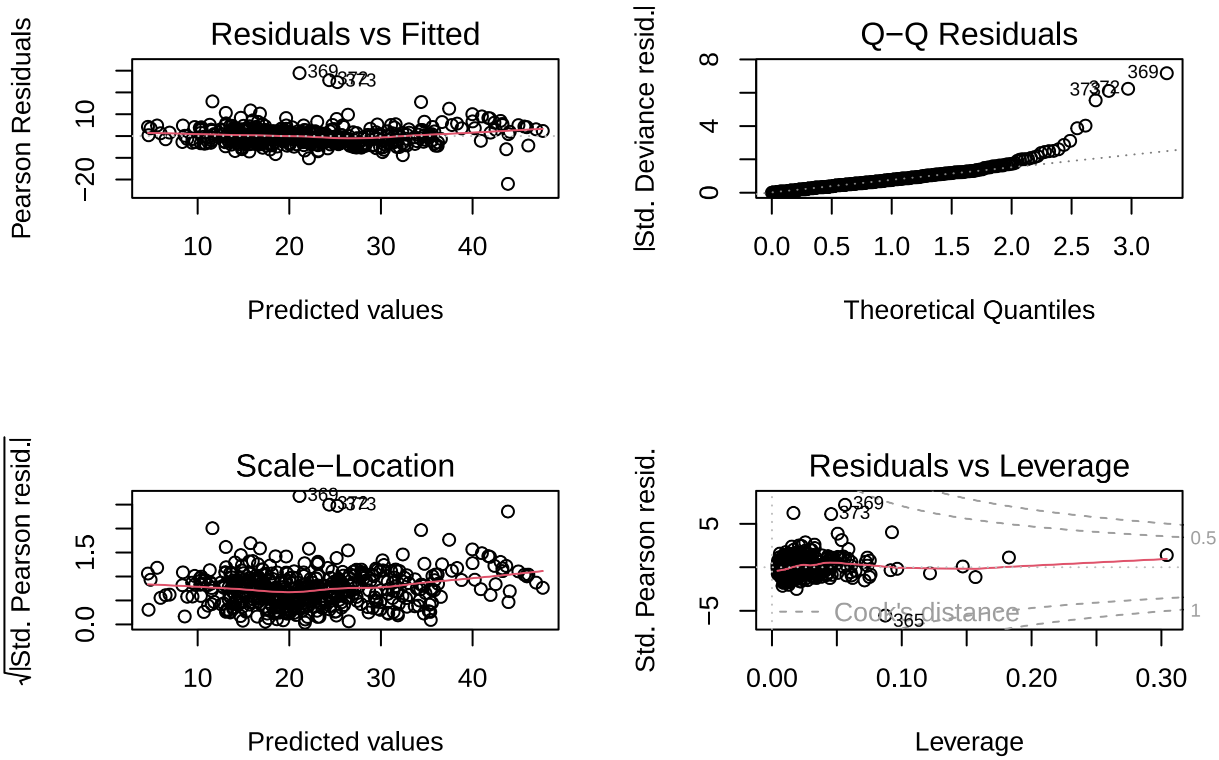
```r
par(mfrow=c(2,2))
m1 <- plot(simple_model_1)
```



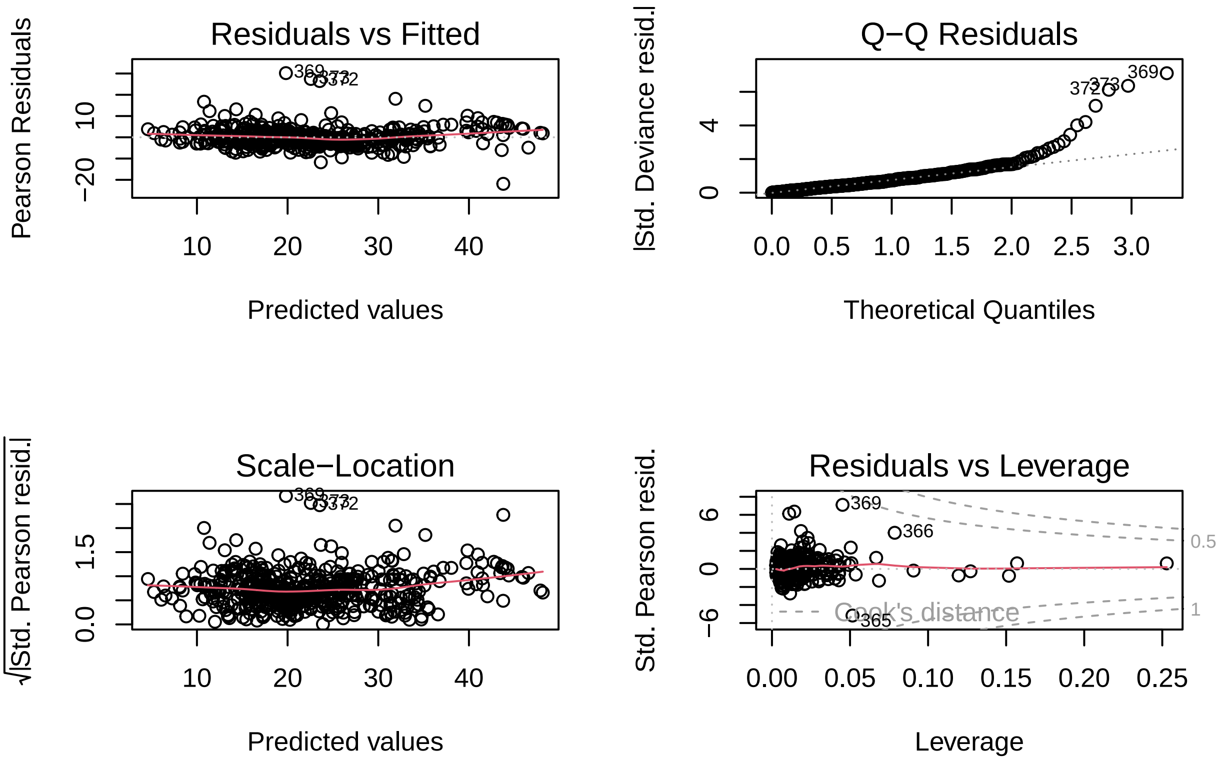

```
m2 <-plot(simple_model_2)
```




```
m3 <-plot(glm_model_1)
```




```
m4 <-plot(glm_model_2)
```




```
# Finding r-sqaured, AIC and BIC value to evaluate model
rsq_simple_model_1 <- summary(simple_model_1)$r.squared
rsq_simple_model_2 <-summary(simple_model_2)$r.squared
rsq_glm_model_1 <-with(summary(glm_model_1), 1 - deviance/null.deviance)
rsq_glm_model_2 <-with(summary(glm_model_2), 1 - deviance/null.deviance)


aic_simple_model_1 <- AIC(simple_model_1)
aic_simple_model_2 <- AIC(simple_model_2)
aic_glm_model_1 <- AIC(glm_model_1)
aic_glm_model_2 <- AIC(glm_model_2)

bic_simple_model_1 <- BIC(simple_model_1)
bic_simple_model_2 <- BIC(simple_model_2)
bic_glm_model_1 <- BIC(glm_model_1)
bic_glm_model_2 <- BIC(glm_model_2)

models_comparison <- data.frame(
  Model = c("Simple Model 1", "Simple Model 2", "GLM Model 1", "GLM Model 2"),
  R_squared = c(rsq_simple_model_1, rsq_simple_model_2, rsq_glm_model_1, rsq_glm_model_2),
  AIC_Value = c(aic_simple_model_1, aic_simple_model_2, aic_glm_model_1, aic_glm_model_2),
  BIC_Value = c(bic_simple_model_1, bic_simple_model_2, bic_glm_model_1, bic_glm_model_2)
)

print(models_comparison)
```

```
##            Model R_squared AIC_Value BIC_Value
## 1 Simple Model 1 0.4835255  3352.151  3364.831
## 2 Simple Model 2 0.5441463  3288.975  3301.655
## 3    GLM Model 1 0.8009243  2887.761  2938.479
## 4    GLM Model 2 0.7792536  2932.045  2965.858
```

3. Logistic Regression

```r
# Converting medv into binary outcomes (i.e. define a new attribute "high_value")
median_value <- median(Boston_copy$medv)
# Adopting the "best" model (glm_model_1) in our former linear regression, selecting same attributes, e
Boston_copy$high_value <- as.factor(ifelse(Boston$medv > median_value, 1, 0))
logistic_model <- glm(high_value ~ chas + nox + rm + dis + rad + tax + ptratio + lstat + rm:lstat,
                      family=binomial, data=Boston_copy)
summary(logistic_model)
```

```
##
## Call:
## glm(formula = high_value ~ chas + nox + rm + dis + rad + tax +
##     ptratio + lstat + rm:lstat, family = binomial, data = Boston_copy)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.232929   5.030110   0.046 0.963066
## chas1        1.855972   0.634682   2.924 0.003453 **
## nox         -8.308931   2.458643  -3.379 0.000726 ***
## rm           3.752230   0.766317   4.896 9.76e-07 ***
## dis         -0.508528   0.118333  -4.297 1.73e-05 ***
## rad          0.250172   0.058503   4.276 1.90e-05 ***
## tax         -0.010052   0.002941  -3.418 0.000630 ***
## ptratio     -0.554300   0.105596  -5.249 1.53e-07 ***
## lstat        0.892691   0.269023   3.318 0.000906 ***
## rm:lstat    -0.213094   0.047309  -4.504 6.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 701.39  on 505  degrees of freedom
## Residual deviance: 278.39  on 496  degrees of freedom
## AIC: 298.39
##
## Number of Fisher Scoring iterations: 7
```

```r
# Predicting and converting probabilities to binary outcome
fitted_results <- predict(logistic_model, type = "response")
fitted_results_bin <- ifelse(fitted_results > 0.5, 1, 0)

# Creating a confusion matrix
table(Boston_copy$high_value, fitted_results_bin)
```

```
##    fitted_results_bin
##       0   1
##   0 225  31
##   1  33 217
```

```r
# row by row: TN(0,0), FP(0,1), FN(1,0), TP(1,1)
```

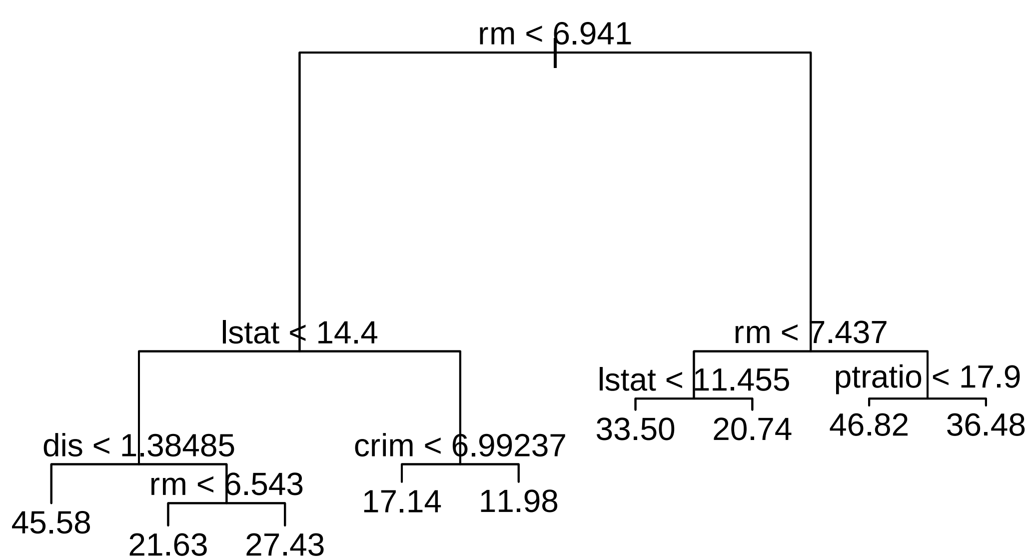
```r
# Accuracy Score
accuracy <- mean(fitted_results_bin == Boston_copy$high_value)
print(accuracy)
```

```
## [1] 0.8735178
```

4. Decision Tree

```r
# Fit the decision tree model, adopting glm_model_1 again but removing interaction term
tree_model <- tree(medv ~ crim + chas + nox + rm + dis + rad + tax + ptratio + lstat, data=Boston_copy)
```

```r
plot(tree_model)
text(tree_model, pretty=0)
```



```r
# Creating a train-test split
set.seed(121) # For reproducibility
train_indices <- sample(1:nrow(Boston_copy), nrow(Boston_copy) * 0.7)
train_data <- Boston_copy[train_indices, ]
test_data <- Boston_copy[-train_indices, ]

# Fit the model on training data
tree_model_train <- tree(medv ~ crim + chas + nox + rm + dis + rad + tax + ptratio + lstat, data=train_d

# Predict on test data
predictions <- predict(tree_model_train, test_data)

# Calculate RMSE or any other metric
rmse <- sqrt(mean((predictions - test_data$medv)^2))
print(rmse)
```

```
## [1] 4.451263
```