Lab Assignment #4

Math 437 - Modern Data Analysis

Due February 27, 2023

Instructions

The purpose of this lab is to review simple linear regression and multiple linear regression strategies from Math 338/439.

In this lab, we will be working with the Boston housing dataset (`Boston` in the `ISLR2` library). This dataset has 506 rows and 13 variables.

```
library(ISLR2)
library(ggplot2)
library(dplyr)
library(car) # For problem 3
```

This lab assignment is worth a total of **19.5 points**.

Problem 1: Bootstrap Estimation of Standard Error

Part a (Code: 0.5 pts)

Run the code in the first half of ISLR Lab 5.3.4, "Estimating the Accuracy of a Statistic of Interest." Put each chunk from the textbook in its own chunk.

If you are in the actuarial science concentration, you should be familiar with (or will at some point see) this formula! For the rest of us, note that X and Y are assumed to be the yearly return of two different financial assets, and α, the quantity to be estimated, is the fraction of money to be invested in X such that the variance (risk) of the total investment $\alpha X + (1 - \alpha)Y$ is minimized. In this problem α is not the significance level!

```
alpha.fn <- function (data , index) {
  X <- data$X[index]
  Y <- data$Y[index]
  ( var (Y) - cov (X, Y)) / ( var (X) + var (Y) - 2 * cov (X, Y))
}
```

```
alpha.fn(Portfolio, 1:100)
```

```
## [1] 0.5758321
```

```
set.seed(7)
alpha.fn(Portfolio, sample(100, 100, replace = T))
```

```
## [1] 0.5385326
```

```
#install.packages("boot")
library(boot)
```

```
##
## Attaching package: 'boot'

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

```
boot(Portfolio, alpha.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.5758321 0.0007959475  0.08969074
```

Part b (Code: 2 pts)

According to the instructions for Lab 5.3.4, "We can implement a bootstrap analysis by performing this command [alpha.fn on a bootstrap sample] many times, recording all of the corresponding estimates for α, and computing the resulting standard deviation."

Write a code chunk that performs all of those steps and prints out the standard deviation. Use 1000 bootstrap samples.

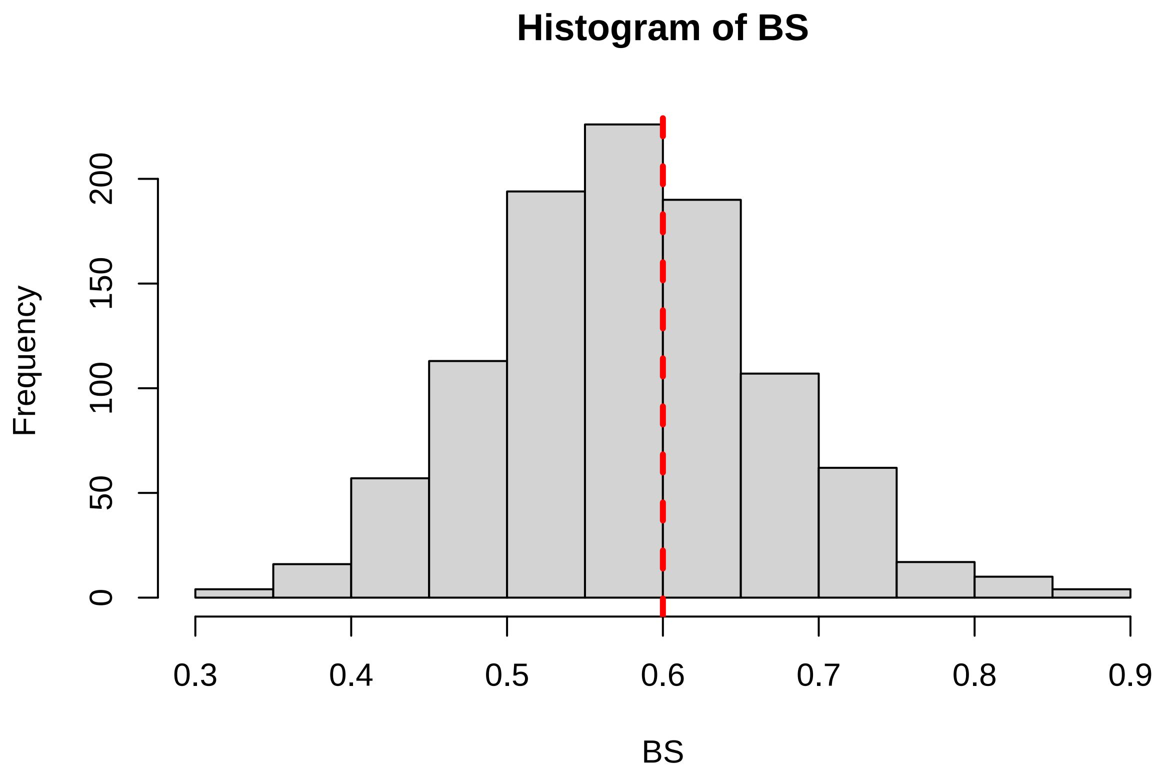
```
# 1. get a bootstrap sample
## Use sample()
#bootSample <- sample[,], replace = T)
# 2. Call alpha.fn on bootstrap sample and store the output (alpha)
## use alpha.fn()
#alpha.fn(Portfolio, )
# 3. Repeat Steps 1 and 2 many times (to get many estimates of alpha)
## use for loop
# 4. Get SD of the alpha-estimates
## Use sd()
set.seed(7)
BS <- matrix(nrow=1,ncol=1000)
for (i in 1:1000) {
  BS[i] = alpha.fn(Portfolio, sample(100, 100, replace = T))
}
sd(BS)
```

```
## [1] 0.0902068
```

Part c (Code: 1 pt)

Replicate the center panel of textbook Figure 5.10: a histogram of the bootstrap estimates of α (from Part b) with a solid pink (or red) line at the true value of $\alpha = 0.6$. You may use either base R plotting commands (which uses `abline` to add the vertical line) or the `ggplot2` package (which adds a `geom_vline` to the plot).

```
hist(BS)
abline(v=0.6, col="red", lwd=3, lty=2)
```




```
#ggplot(data=Boston, aes(x=BS)) + geom_vline(v=0.6)
```

Part d (Explanation: 1 pt)

Note that the distribution you graphed in Part c is a sampling distribution of $\hat{\alpha}$. Explain why it would be appropriate to use this sampling distribution to construct a confidence interval for α, but not to obtain a p-value for a hypothesis test of $H_0 : \alpha = 0.6$ against $H_a : \alpha \neq 0.6$.

It would be appropriate to use this sampling distribution to construct a confidence interval for $\hat{\alpha}$ because we used bootstrap to get this distribution. We can't obtain a p-value because we didn't do a hypothesis test. This also means we don't have a value of mu, so we don't need alpha either.

Problem 2: Domain Knowledge and Exploratory Data Analysis

Part a (Explanation: 1 pt)

Do an Internet search for "Boston housing dataset" and answer the following questions as best you can.

- Who collected this data? How old is this dataset?

The data was collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts. This data is from the 1970 census, so about 53 years old.

- What does one row in this dataset represent?

One row in this dataset represents housing in a Boston suburb or town.

Part b (Explanation: 1.5 pts)

In your search, you should eventually come across references to a *fourteenth* variable, `B`, which the textbook authors have removed from the dataset. What does this mysterious variable represent?

The 14th variable `B` is $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks(sic) by town.

Suppose you are a data scientist at Zillow or a similar company whose housing price models are often used as a reference when people decide how much to offer to buy or sell a home for. What ethical issues would arise from using the variable `B` in your model?

The original description of the variable `B` might be offensive since it is *the proportion of blacks by town.* It's also unethical to use race as a predictor for housing price.

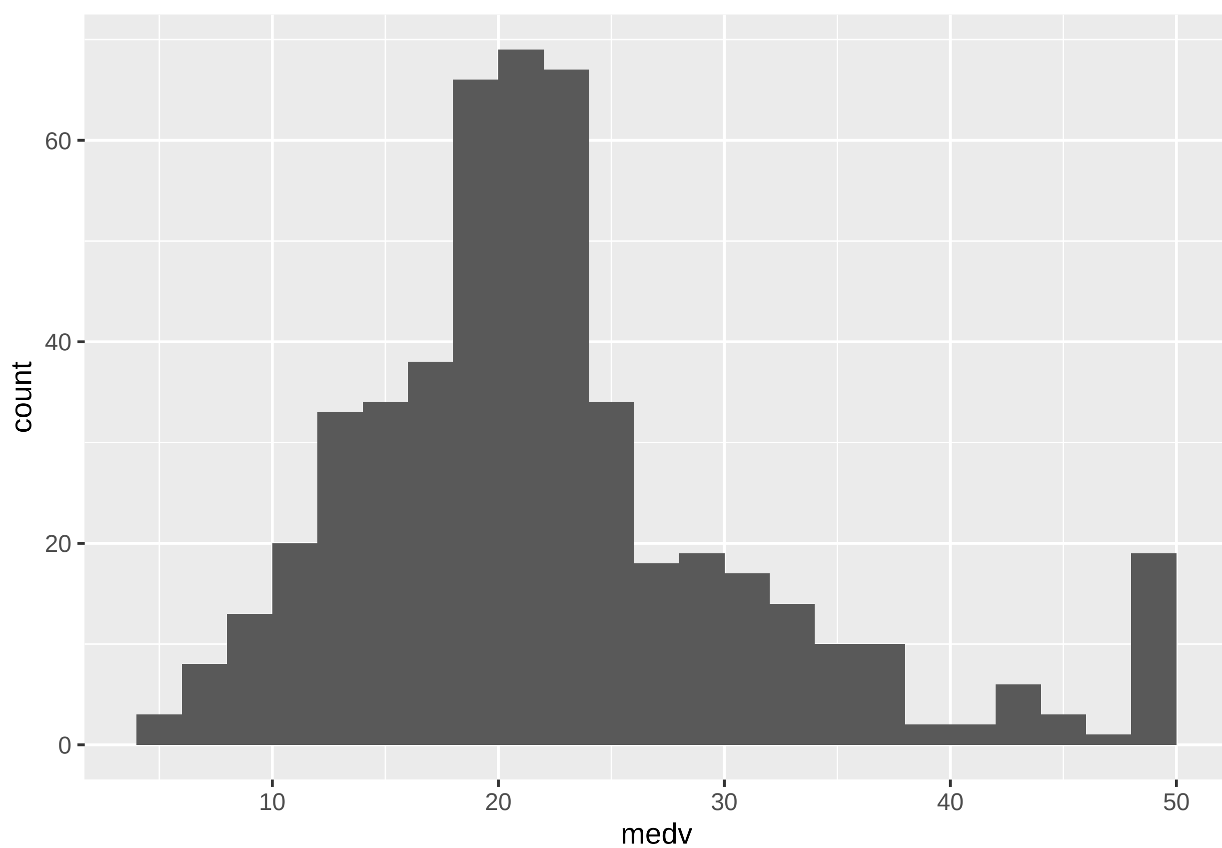
Part c (Code: 1 pt; Explanation: 1 pt)

In the next problem we will be trying to predict `medv` from `lstat`. What does the variable `medv` represent? What are the measurement units?

`medv` represents the median value of owner-occupied homes in $1000's (thousands of dollars).

Using the `ggplot2` package, create a histogram of the variable `medv`. Use a `center` of 35 and a `binwidth` of 2.

```
ggplot(data=Boston, aes(x=medv)) + geom_histogram(center=35, binwidth=2)
```



What looks a bit off about this histogram? Try filling in the `filter` function in the chunk below to confirm your suspicions.

The graph is supposed to be skewed right or normal for the most part, but there is the bin at 50,000 with almost 20 observations that makes it look unusual. Using the filter function, we see that there are exactly 16 observations at 50,000.

```
Boston %>%
  filter(medv > 48) %>%
  count() # getting sample size without having to summarize
```

Part d (Code: 1 pt; Explanation: 1 pt)

The full documentation for this dataset is somewhat confusing and raises more questions than answers. For example, `lstat` is defined as "$\frac{1}{2}$ (proportion of adults without some high school education and proportion of male workers classified as laborers)" (whatever that means), and `rad` represents the "index of accessibility to radial highways" as determined by something called the "MIT Boston Project."

Other variables are sensibly defined, but are counterintuitive to what we would expect. Pick either the variable `age` or `rm`, and answer the following questions:

I pick `age`

- What do you expect this variable would represent, if the observational units were houses?

I expect `age` to represent how old the house is.

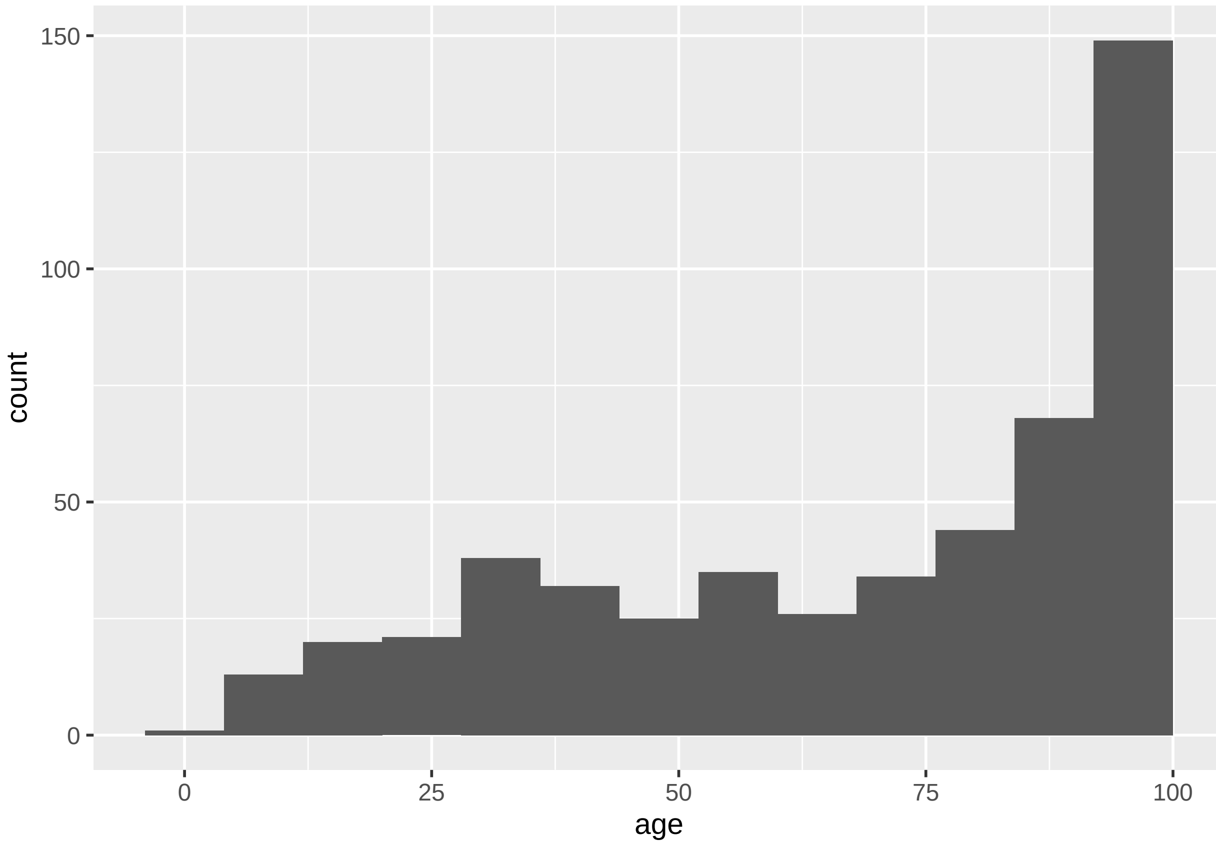
- What does this variable actually represent?

`age` actually represents the proportion of owner-occupied units built prior to 1940.

- What is the distribution of this variable in the dataset? Include at least one graph to support your answer.

The distribution of `age` is left-skewed. There are many old houses and not as many new houses.

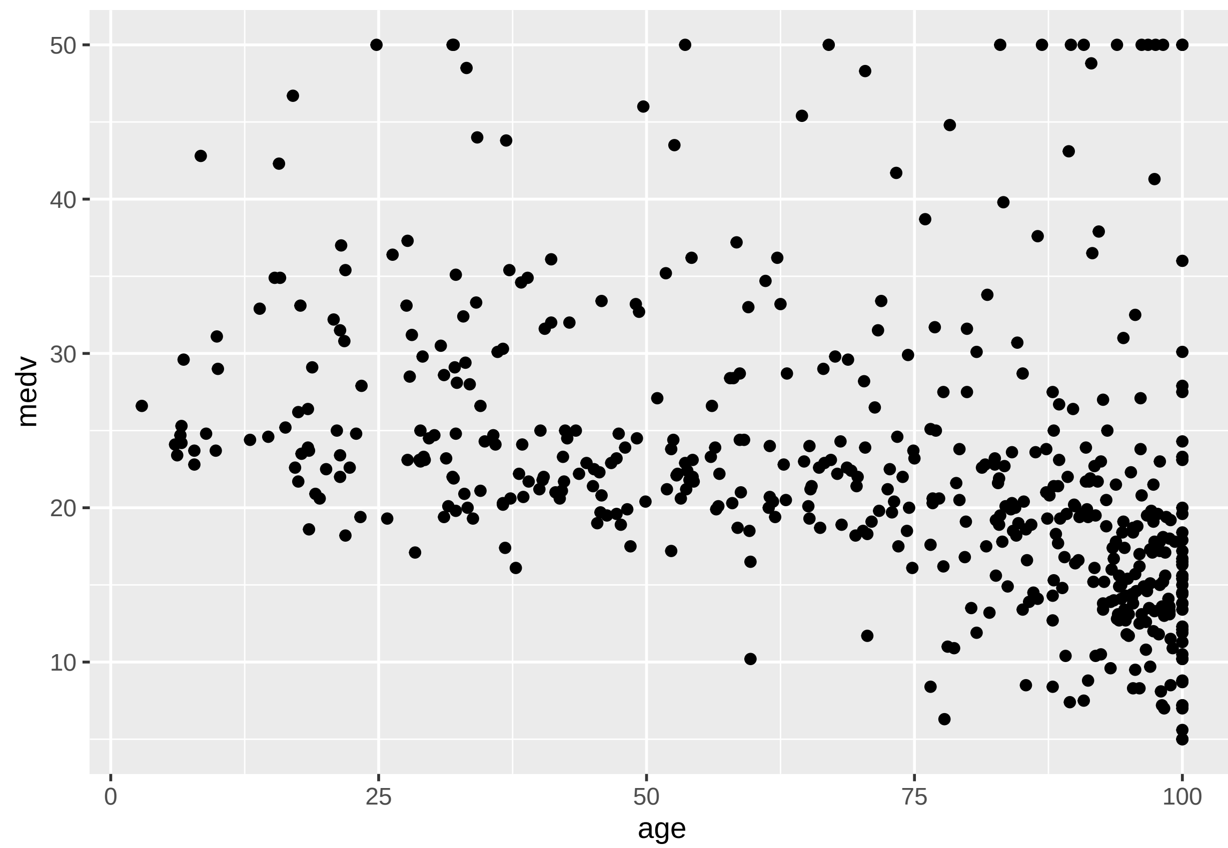
```
ggplot(data=Boston, aes(x=age)) + geom_histogram(binwidth=8)
```



- Does this variable appear to have a relationship with the response variable `medv`? Include at least one graph to support your answer.

Looking at the scatterplot, there are more clusters of points on the bottom right corner of the graph. Comparing medv and age, this implies that the oldest neighborhoods in Boston have less value in thousands, compared to houses built between 0 and 50 years from 1940. However, there are quite a bit of outliers of neighborhoods built at least 80 years ago that are more valuable than most of the newer houses. I believe this is because there are some neighborhoods that could possibly have nicer looking houses or have been renovated.

```
ggplot(data=Boston, aes(x=age, y=medv)) + geom_point() #+ geom_abline(aes(intercept=, slope=))
```



Problem 3: Simple Linear Regression

Part a (Code: 0.5 pts; Explanation: 1 pt)

Run the code in ISLR Lab 3.6.2. Put each chunk from the textbook in its own chunk.

```
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
```

```
?Boston
```

```
lm.fit <- lm(medv~lstat)
```

```
lm.fit <- lm(medv ~ lstat, data = Boston)
attach(Boston)
lm.fit <- lm(medv ~ lstat)
```

```
lm.fit
```

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

```
## lm(formula = medv ~ lstat)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```r
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## 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
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```r
#we can extract quantities by name (e.g. lm.fit$coefficients) but it is safer to use extractor function
coef(lm.fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

```r
#Confidence intervals for coefficient estimates
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

```r
#95% Confidence interval for the prediction of medv for a given value of lstat
predict(lm.fit, data.frame(lstat = (c(5, 10, 15))), interval = "confidence")
```

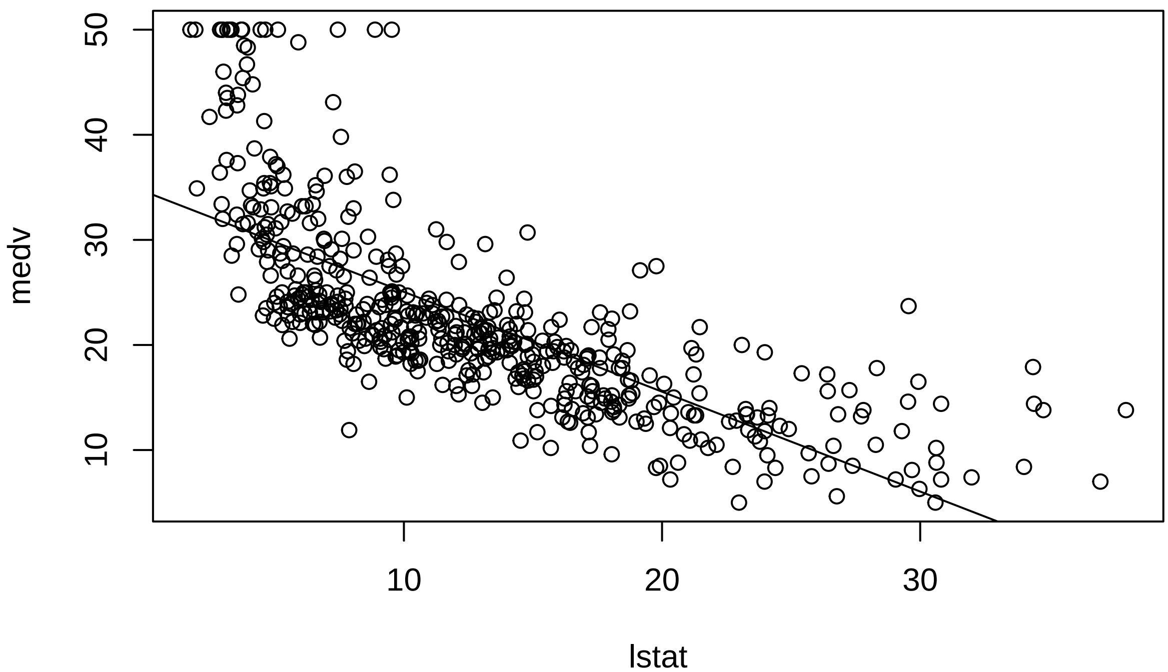
```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```r
#95% Prediction interval for the prediction of medv for a given value of lstat
#prediction interval is wider than confidence interval
predict(lm.fit, data.frame(lstat = (c(5, 10, 15))), interval = "prediction")
```

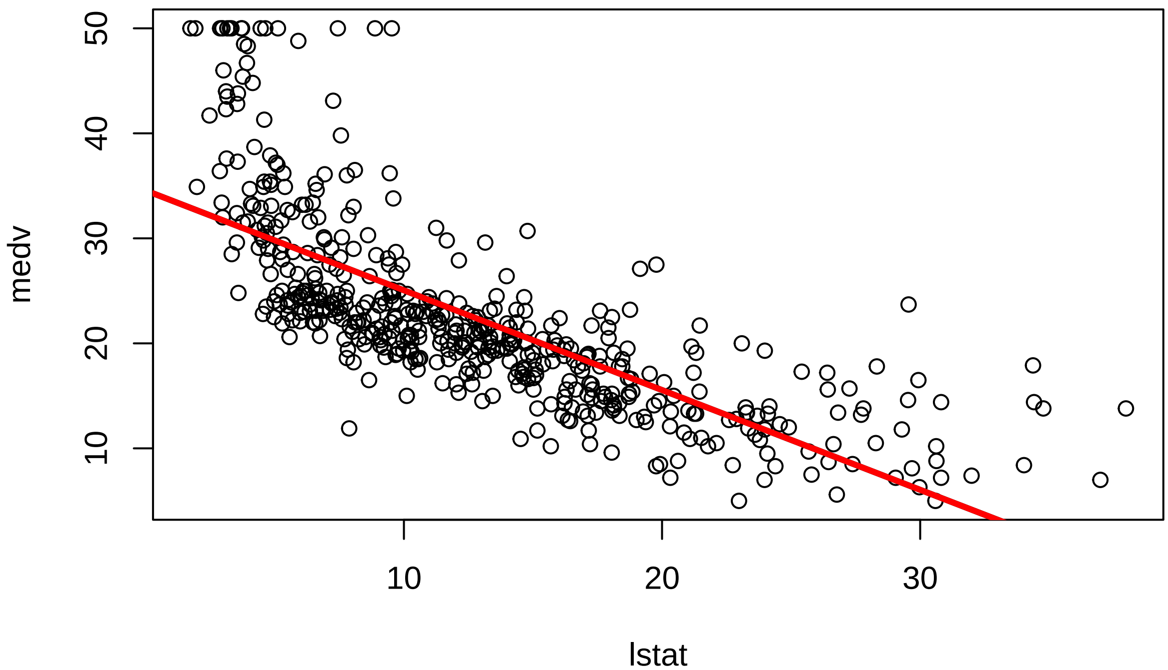
```
##        fit       lwr      upr
```

```
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

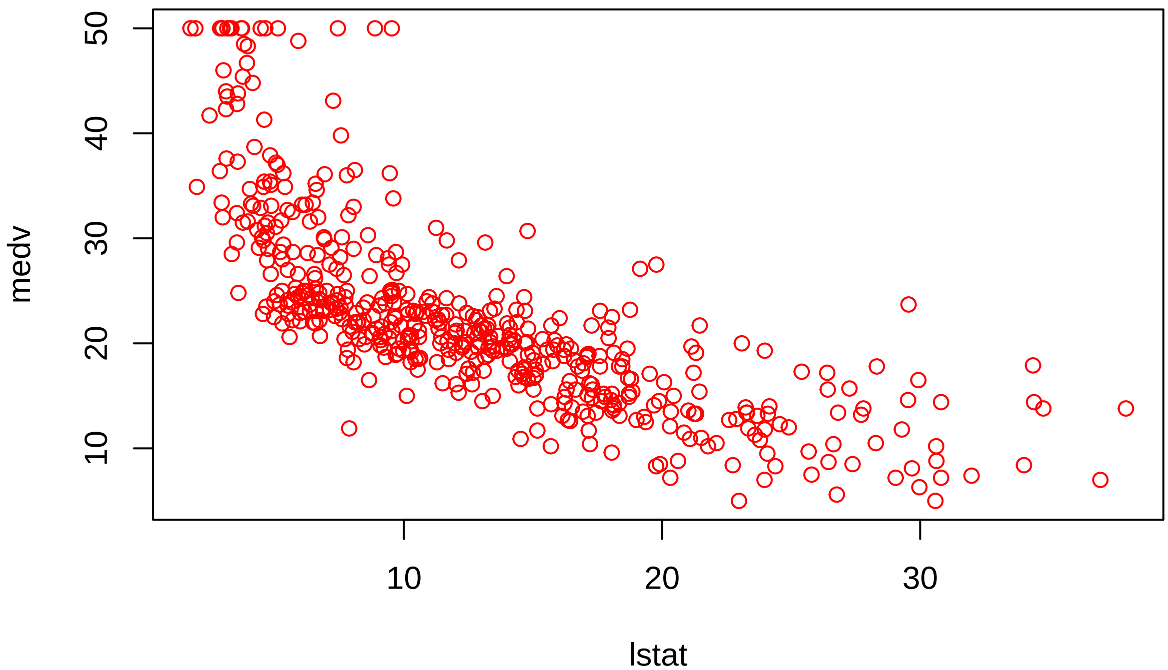
```
plot(lstat, medv)
abline(lm.fit)
```



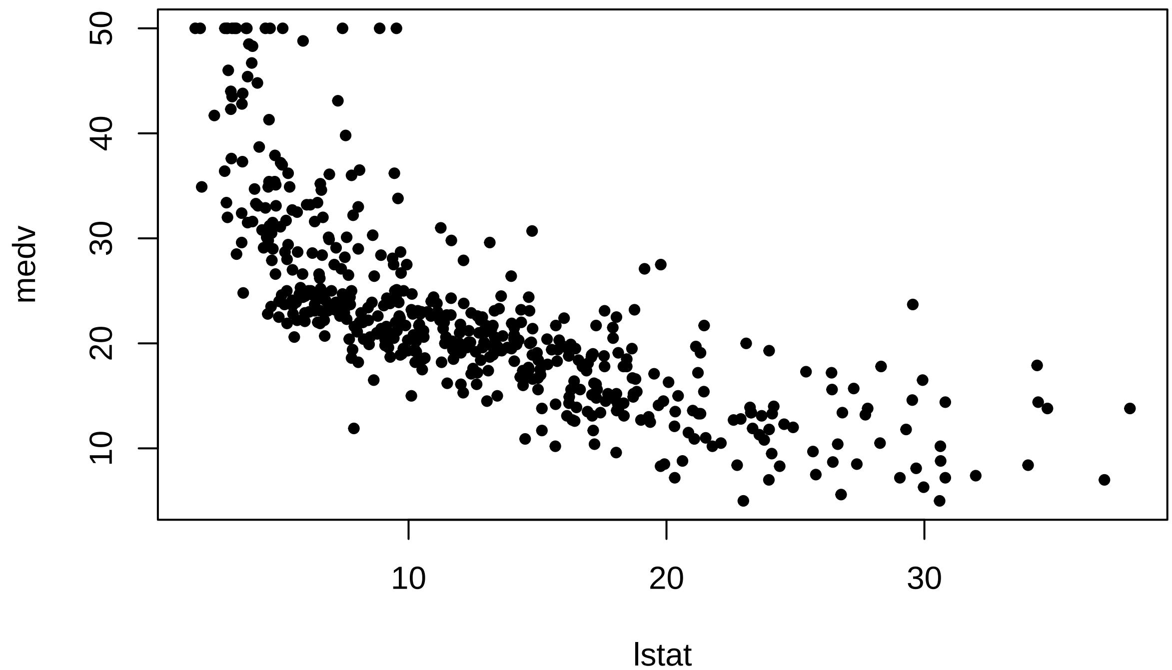

```
plot(lstat, medv)
abline(lm.fit, lwd = 3) #can be used to draw any line, not just LS regression line
abline(lm.fit, lwd = 3, col = "red")
```



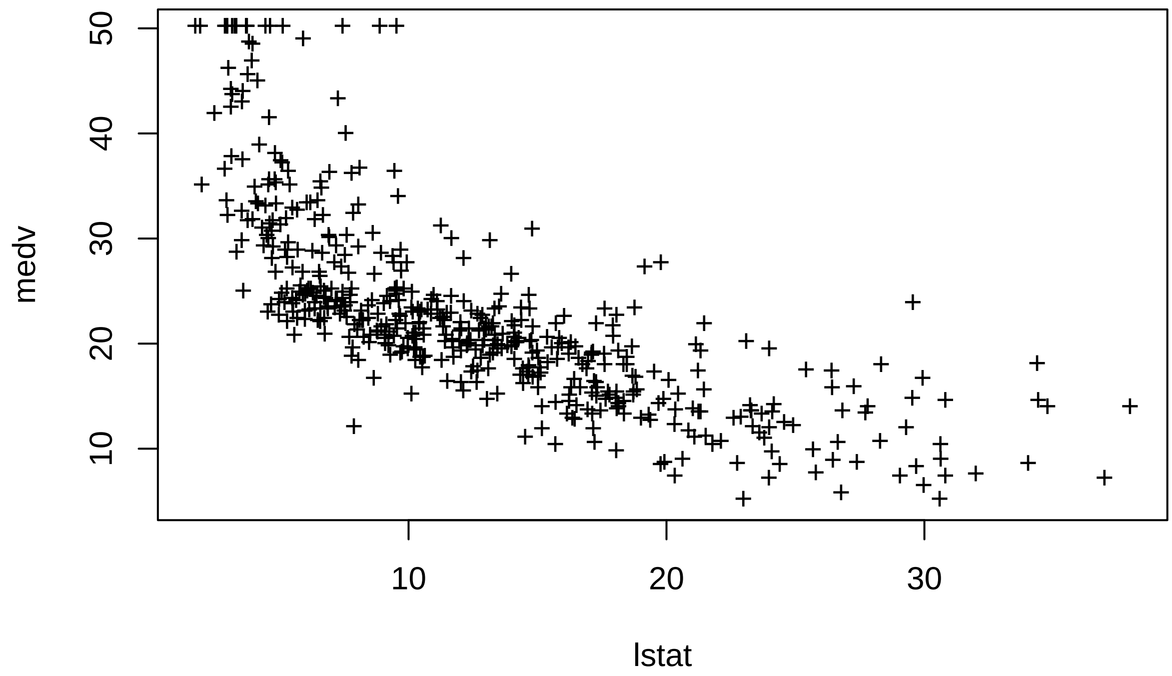

```
plot(lstat, medv, col = "red")
```



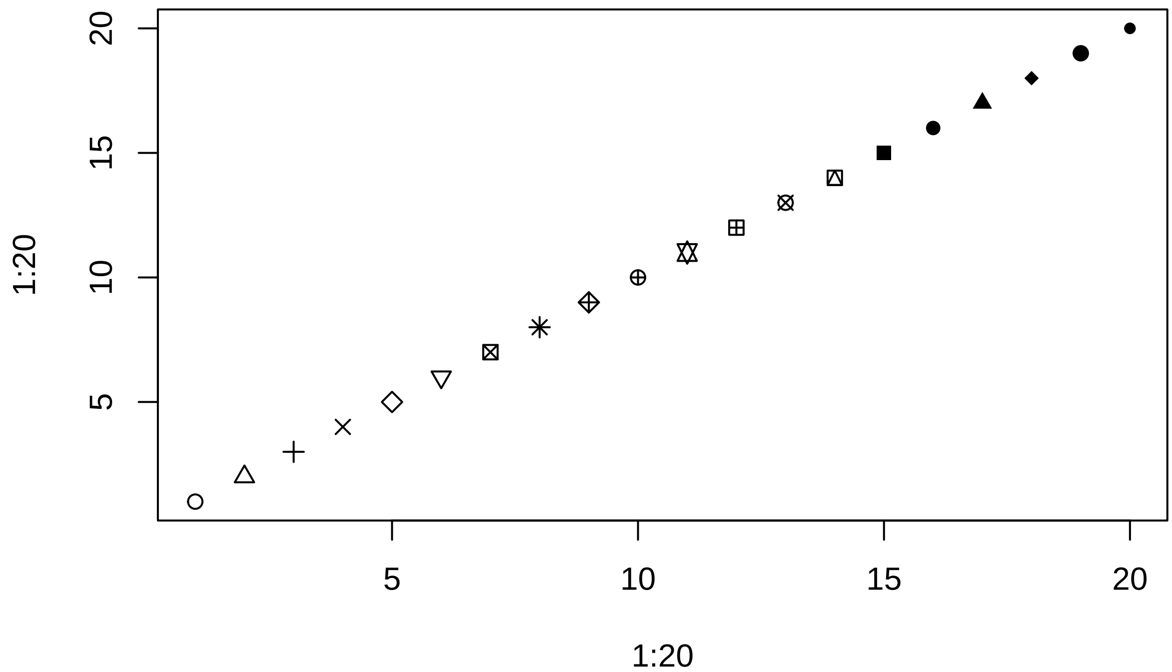

```r
plot(lstat, medv, pch = 20)
```



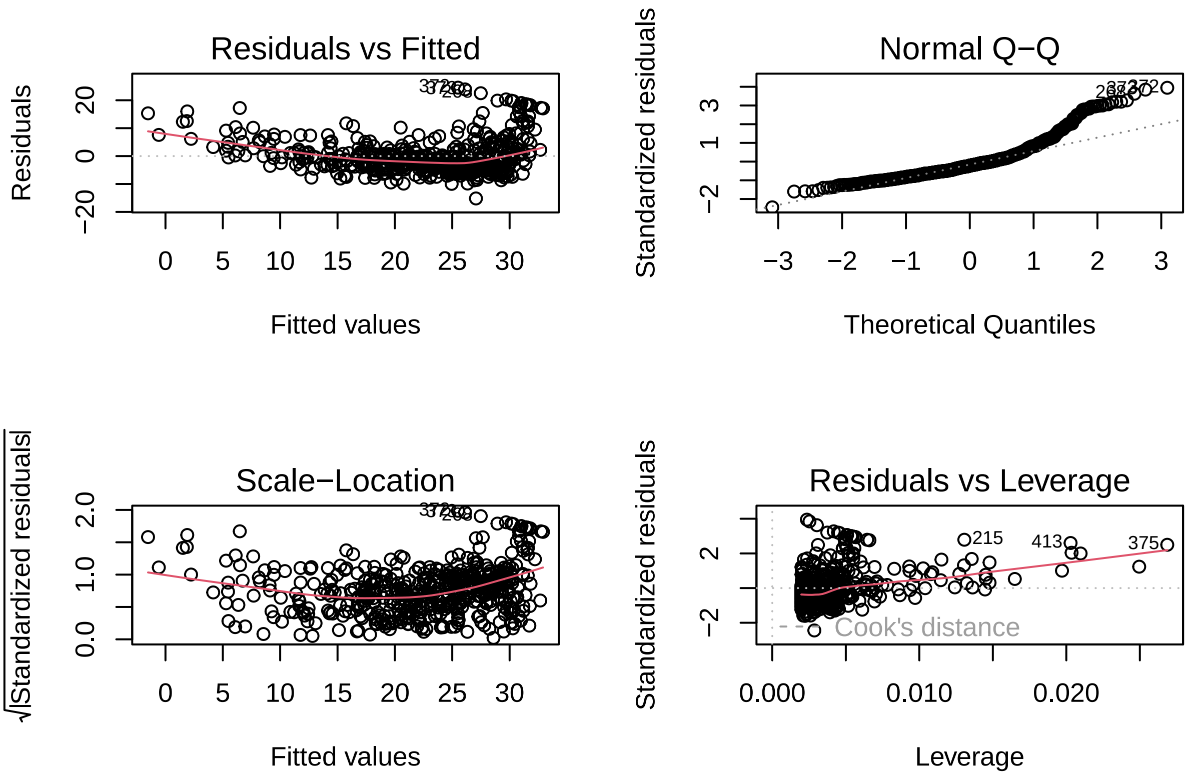

```
plot(lstat, medv, pch = "+")
```



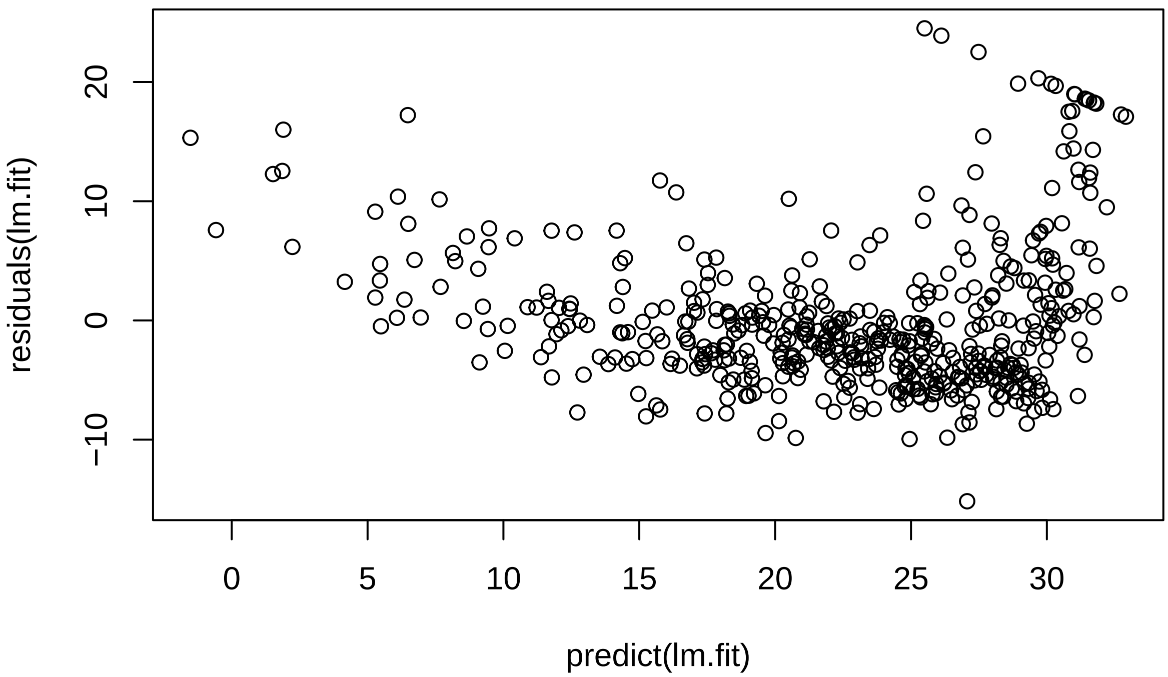

```
plot(1:20, 1:20, pch = 1:20)
```



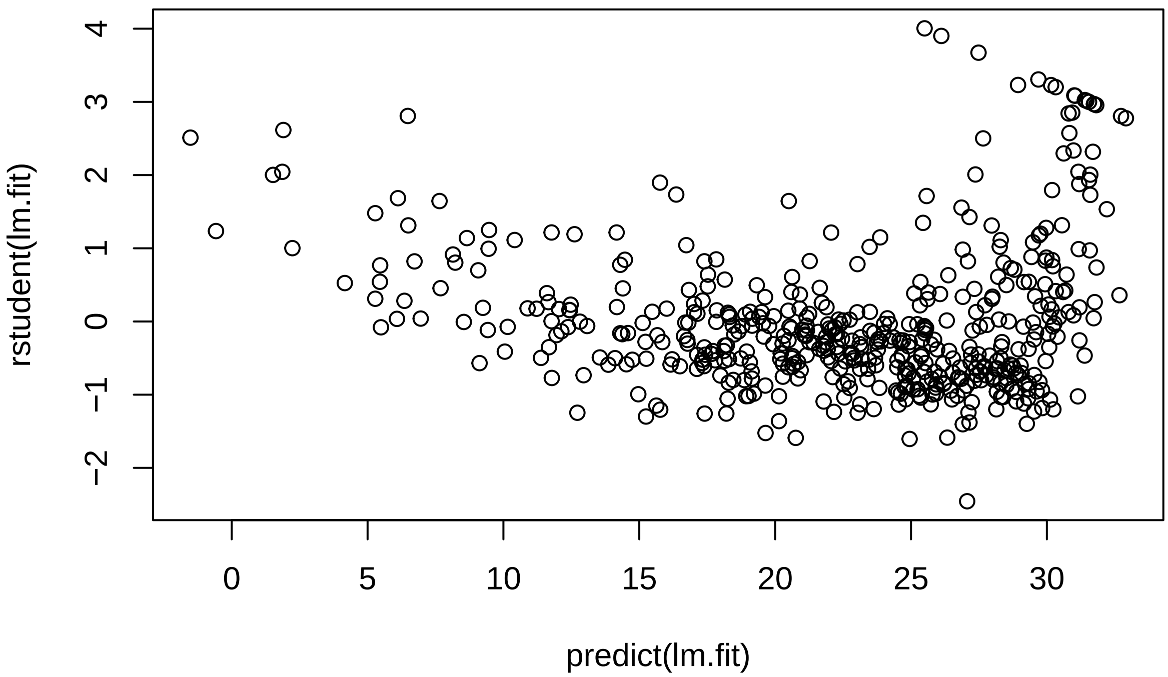

```r
par(mfrow = c(2,2))
plot(lm.fit)
```



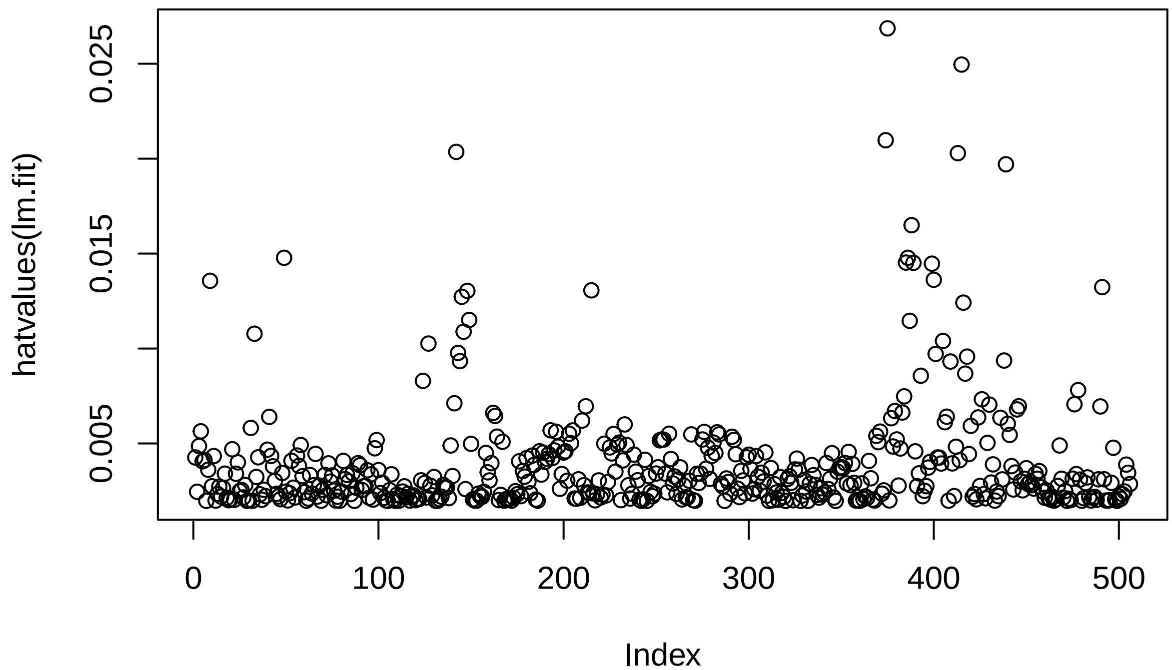

```
plot(predict(lm.fit), residuals(lm.fit))
```




```
plot(predict(lm.fit), rstudent(lm.fit))
```




```
plot(hatvalues(lm.fit))
```




```
which.max(hatvalues(lm.fit))
```

```
## 375
## 375
```

Briefly explain what the `confint()` and `predict()` functions output when applied to a linear model.

The confint() command helps us obtain confidence intervals for the coefficient estimates. The predict() command produces confident intervals and prediction intervals for a prediction of medv given a value of lstat.

Part b (Explanation: 2.5 pts)

Write out the equation of the least-squares line relating `lstat` and `medv`. Write two sentences interpreting the parameter estimates (one for slope, one for intercept) in the context of the data. Remember to use the right observational units!

The least-squares line relating lstat and medv is lstat = 34.55(medv) - 0.95 The intercept is 34.55, which means that when we are not accounting for any percentage of the lower status of the Boston population, there are about $ 34,500 houses with owners. In regards to the slope, the smaller the percentage of lower class, the greater the median value of houses owned in thousands of dollars.

Given the issue raised in Problem 1d with the `lstat` interpretation, let's just say in our interpretations that `lstat` represents the percentage of people in the neighborhood considered lower class.

Part c (Explanation: 1.5 pts)

Refer to the diagnostic plots you created in Part (a) to answer the following questions:

- Why do the lab instructions claim that "there is some evidence of non-linearity"?

Based on the residual vs fitted graph, linearity is shown when there is a horizontal line. However, the general relationship of the data is curved, so the red line is also more curved as well.

- Do you believe that the residuals are normally distributed? Why or why not?

The line of best fit on the q-q plot is at an angle less than 45 degrees, which convinces me to believe that the residuals are not normally distributed.

- Do you believe that the response variable is homoskedastic (the residuals have roughly constant variance across the entire predictor range)? Why or why not?

I believe the response variable is not homoskedastic because the variance is not constant on the scale-location graph. The majority of the data points are on the right side of the graph, despite the line of best fit. Thus the variance is not constant and there is no homoskedasticity.

Problem 4: Multiple Linear Regression

Part a (Code: 0.5 pts; Explanation: 1 pt)

Run the code in ISLR Lab 3.6.3. Put each chunk from the textbook in its own chunk. (Note that you will have to install the `car` package.)

```
lm.fit <- lm(medv ~ lstat + age , data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
lm.fit <- lm(medv ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1304  -2.7673  -0.5814   1.9414  26.2526
##
## 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
## chas          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
##
## Residual standard error: 4.798 on 493 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7278
## F-statistic: 113.5 on 12 and 493 DF,  p-value: < 2.2e-16
```

```
library(car)
vif(lm.fit)
```

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.767486 2.298459 3.987181 1.071168 4.369093 1.912532 3.088232 3.954037
##      rad      tax  ptratio    lstat
## 7.445301 9.002158 1.797060 2.870777
```

```
lm.fit1 <- lm(medv ~ . - age , data = Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1851  -2.7330  -0.6116   1.8555  26.3838
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.525128   4.919684   8.441 3.52e-16 ***
## crim         -0.121426   0.032969  -3.683 0.000256 ***
## zn            0.046512   0.013766   3.379 0.000785 ***
## indus         0.013451   0.062086   0.217 0.828577
## chas          2.852773   0.867912   3.287 0.001085 **
## nox         -18.485070   3.713714  -4.978 8.91e-07 ***
## rm            3.681070   0.411230   8.951  < 2e-16 ***
## dis          -1.506777   0.192570  -7.825 3.12e-14 ***
## rad           0.287940   0.066627   4.322 1.87e-05 ***
## tax          -0.012653   0.003796  -3.333 0.000923 ***
## ptratio      -0.934649   0.131653  -7.099 4.39e-12 ***
## lstat        -0.547409   0.047669 -11.483  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.794 on 494 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7284
## F-statistic: 124.1 on 11 and 494 DF,  p-value: < 2.2e-16
```

```
lm.fit1 <- update (lm.fit , ~ . - age)
```

Briefly explain what the `vif()` and `update()` functions do when applied to a linear model.

The variance inflation function is used to check how cautious we need to be to make sure the assumptions are not violated. The higher a vif value, the greater colinearity or multicolinearity there is present.

The update function is used to update an already existing variable by editing it. In our case, the update function was used to output a regression for all the predictor variables in our Boston dataset except for age because it had a high p-value.

Part b (Code: 0.5 pts; Explanation: 1 pt)

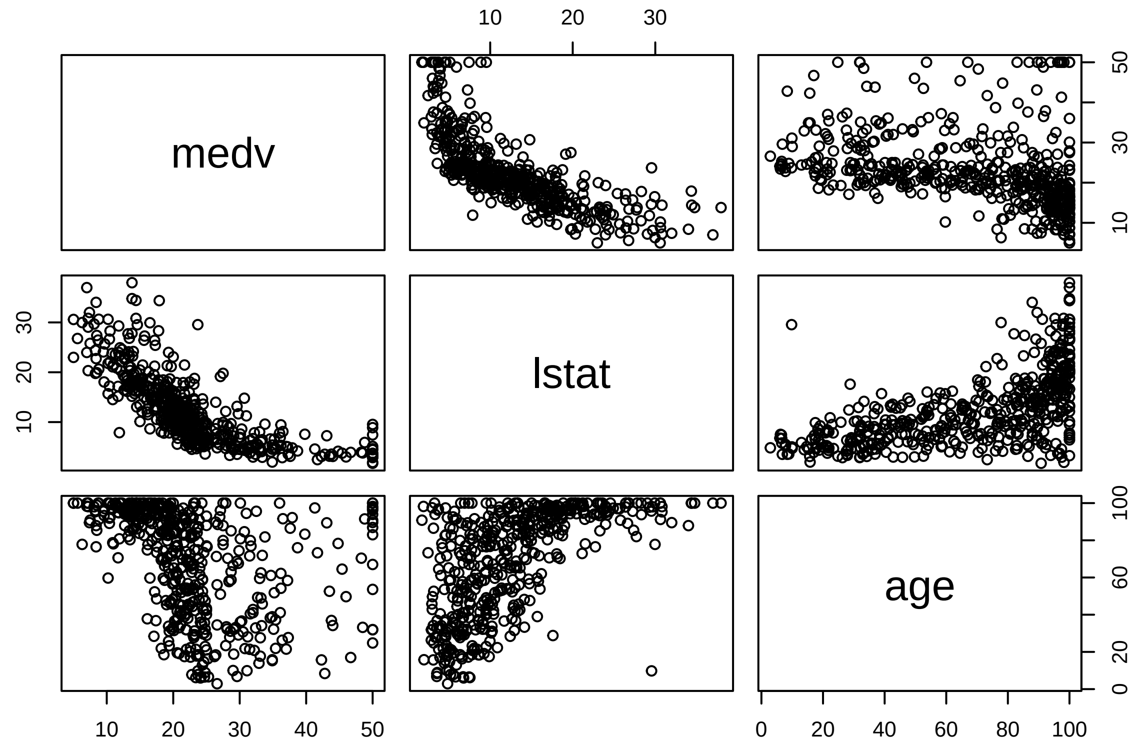
Jumping straight into modeling without looking at the data is a very bad idea. Create a scatterplot matrix showing only the three variables in the first `lm.fit` object (`medv`, `lstat`, `age`).

Do you see any evidence of nonlinearity? Any evidence of collinearity? Explain your reasoning.

Medv is the response variable, so we are looking at the top middle and top right graph. The lstat vs age graph is to answer colinearity.

There is evidence of nonlinearity between `medv`, `lstat`, and `age` as none of the scatterplots seem to follow a clearly linear pattern. Since we see that `medv`, `lstat`, and `age` don't have a linear relationship, then there is also no evidence of collinearity.

```
pairs(~ medv +  lstat + age, data = Boston)
```


medv
lstat
age
10
20
30
50
40
0
60
80
100