HDS Exercise set 1

Shabbeer Hassan

Problem 1 - Solutions

(a)

```r
library(MASS)
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
anyNA(Boston)
```

```
## [1] FALSE
```

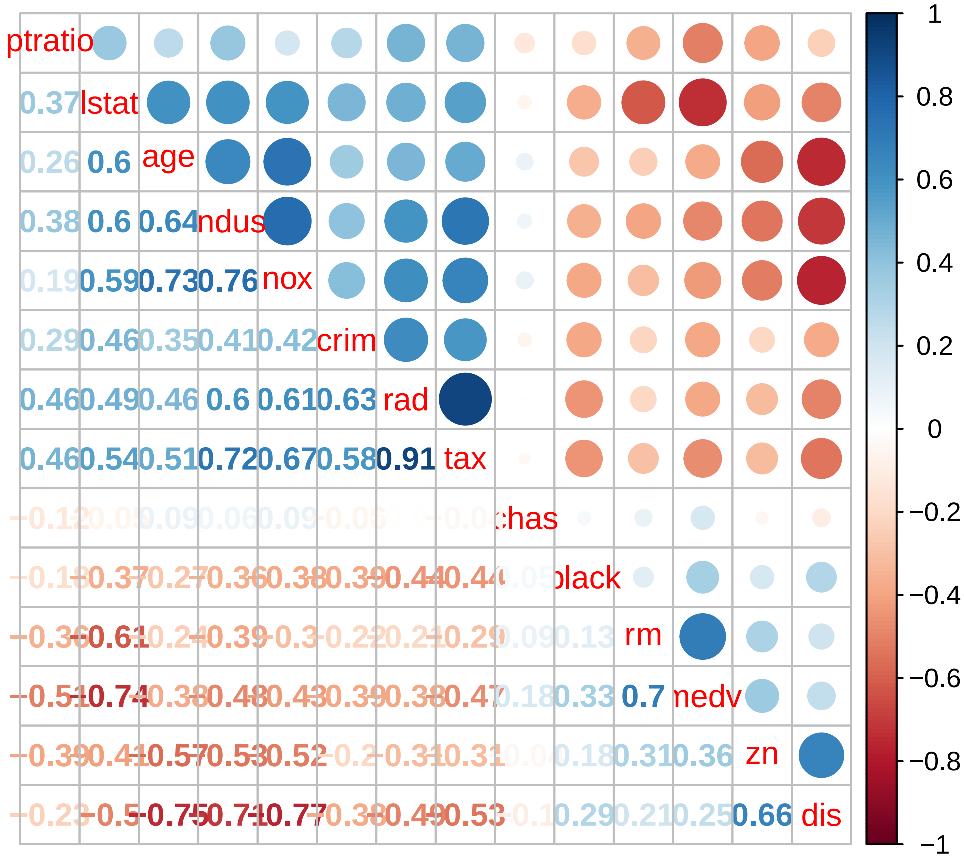
Boston dataset contains n=506 and p=13, without any missing values. The dataset contains all numerical variables and no non-numerical ones

(b)

```r
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
corr.matrix = cor(Boston)
corrplot.mixed(corr.matrix, order = "hclust")
```



Based on the corrplot above, we see that two variables - Index of accessibility to raidal highways ("rad") & Full-value property-tax rate per $10000 (tax) are highly correlated (r > 0.9).

For variable "medv", based on the correlation values alone we see that variables such as - Average number of rooms per dwelling ("rm") & Lower status of the population ("lstat") could be potential predictors (r>0.7, either direction)

(c)

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ---------------------------------------------------------------------------------- tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
library(broom)




## Function to extract data out of linear regression model, and return important values ( adj R-squares
```

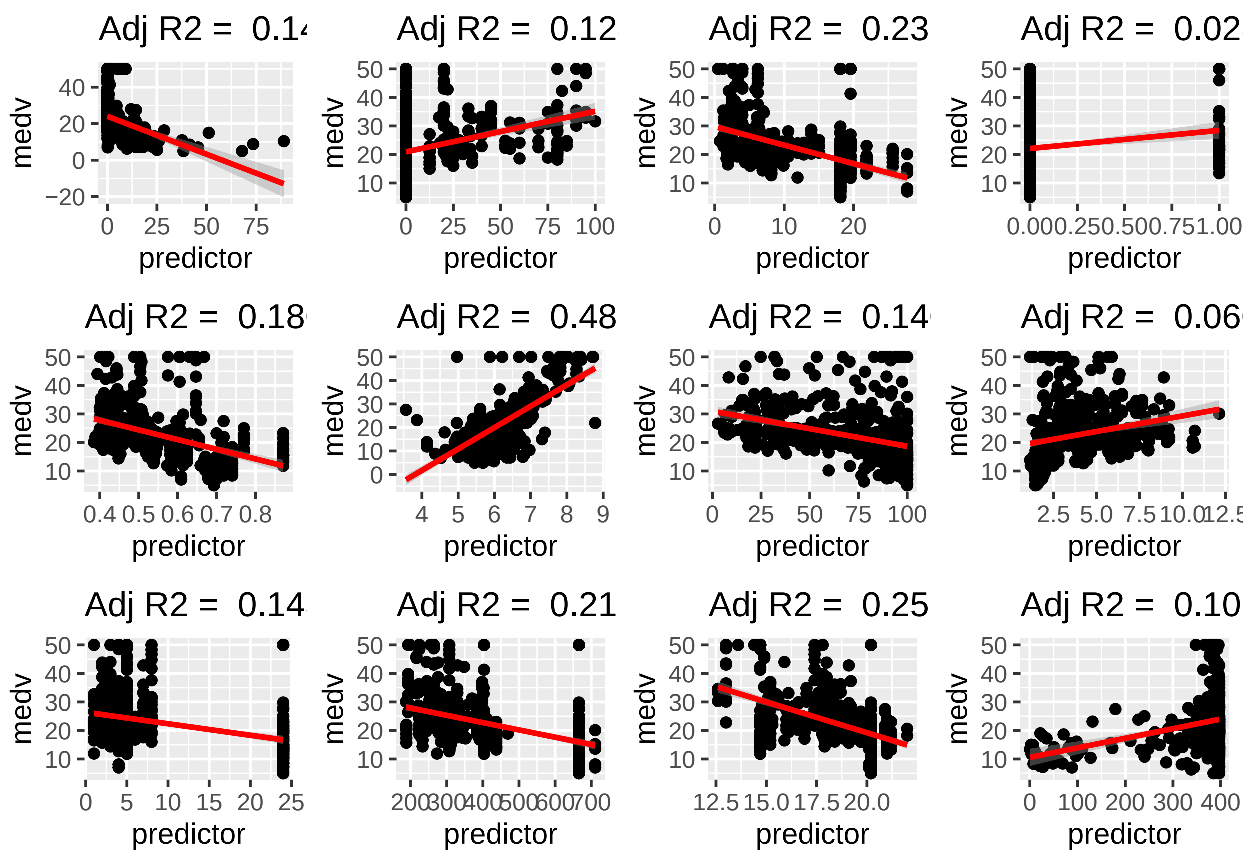
```r
ggplotRegression <- function (fit) {

require(ggplot2)

ggplot(fit$model, aes_string(x = names(fit$model)[2], y = names(fit$model)[1])) +
  geom_point() +
  stat_smooth(method = "lm", col = "red") +
  labs(title = paste("Adj R2 = ",signif(summary(fit)$adj.r.squared, 5)))
}


## Use a for loop to run through the columns of Boston matrix wrt "medv" variable

library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```r
plotlist = list()
Boston_lm <- Boston[,-4] # Remoce chas

cols_pred <- ncol(Boston_lm) - 1
cols_name <- colnames(Boston_lm[, -13])

for (i in 1:cols_pred) {
  predictor = Boston[,i]
  p <- ggplotRegression(lm(medv ~ predictor, data = Boston))
  pname <- paste0("Medv_vs_", cols_name[i])
  ggsave(paste0(pname,".png"),p)
  plotlist[[i]] = p
}
```

```
## Saving 6.5 x 4.5 in image

## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
```

```r
## USe gridExtra to generate different figures from plotlist
p <- grid.arrange(grobs = plotlist, ncol = 4)
```




```r
ggsave("LM_plot.png",p)
```

```
## Saving 6.5 x 4.5 in image
```

(d)

```r
lm.fit = lm(medv ~ lstat + rm, data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.076  -3.516  -1.010   1.909  28.131
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.35827    3.17283  -0.428    0.669
## lstat       -0.64236    0.04373 -14.689   <2e-16 ***
## rm           5.09479    0.44447  11.463   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.54 on 503 degrees of freedom
## Multiple R-squared:  0.6386, Adjusted R-squared:  0.6371
## F-statistic: 444.3 on 2 and 503 DF,  p-value: < 2.2e-16
```

```r
confint(lm.fit) # getting CI
```

```
##                  2.5 %     97.5 %
## (Intercept) -7.5919003  4.8753547
## lstat       -0.7282772 -0.5564395
## rm           4.2215504  5.9680255
```

The predictors are related to median house value in a significant fashion. To start off with this model explains 64% of the variance in the medain house values. With "lstat", it is negatively associated with an estimate of -0.64 whereas "rm" is positively associated with 5.09 estimate. What this means simply is, If all predictors remain constant, then 1 unit change in "lstat" reduves Median house values by -0.64 and for "rm", a unit change brings an increase of Median values by 5.09

Predict function

```r
predict(lm.fit, data.frame(lstat = c(7, 17), rm = c(5, 5)),interval="confidence")
```

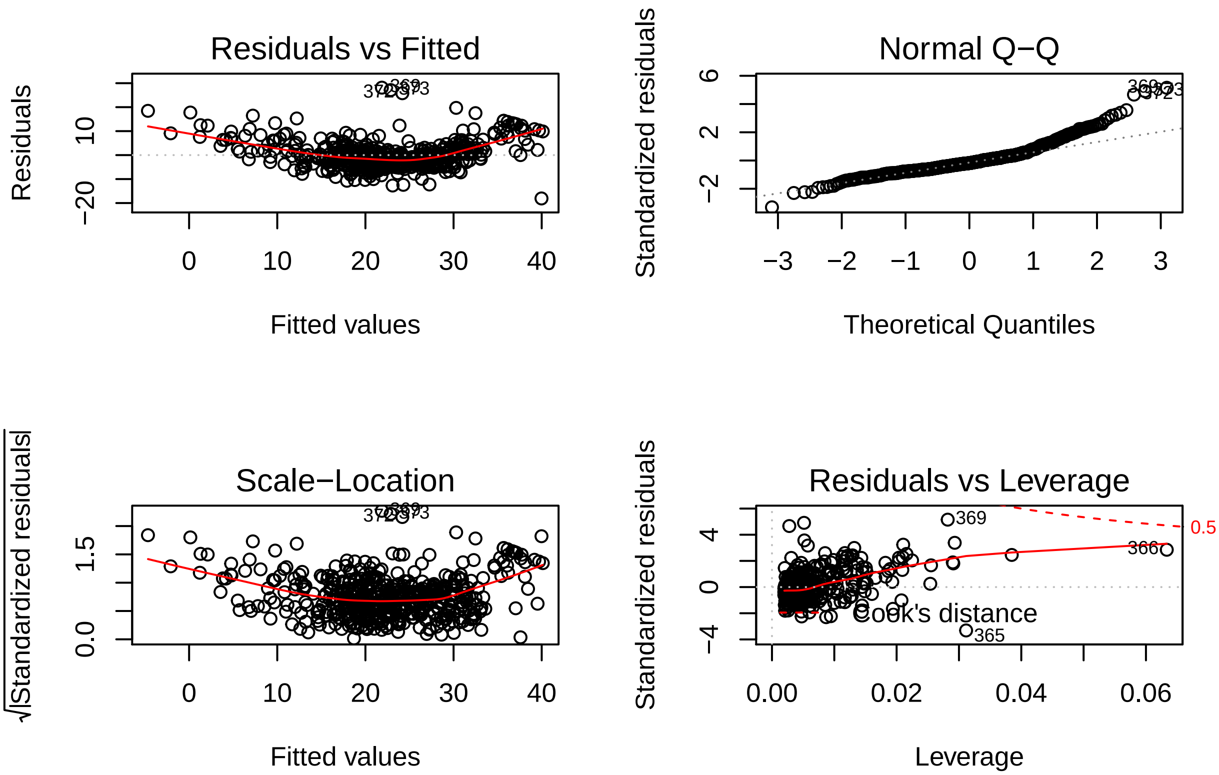
```
##        fit      lwr     upr
## 1 19.61916 18.07082 21.1675
## 2 13.19558 12.13835 14.2528
```

Problem 2 - Solutions

Let's continue with the linear model `medv ~ lstat + rm` in `Boston` dataset.

(a)

```r
par(mfrow=c(2,2)) # Change the panel layout to 2 x 2
plot(lm.fit)
```



The diagnostic plots suggest that the influence of outliers is very strongly present here.One observation is far beyond Cook's distance lines, while the other residuals appear clustered on the left because the last plot is scaled to show larger area than the Scale-Location plot. The plot identified the influential observation as #366

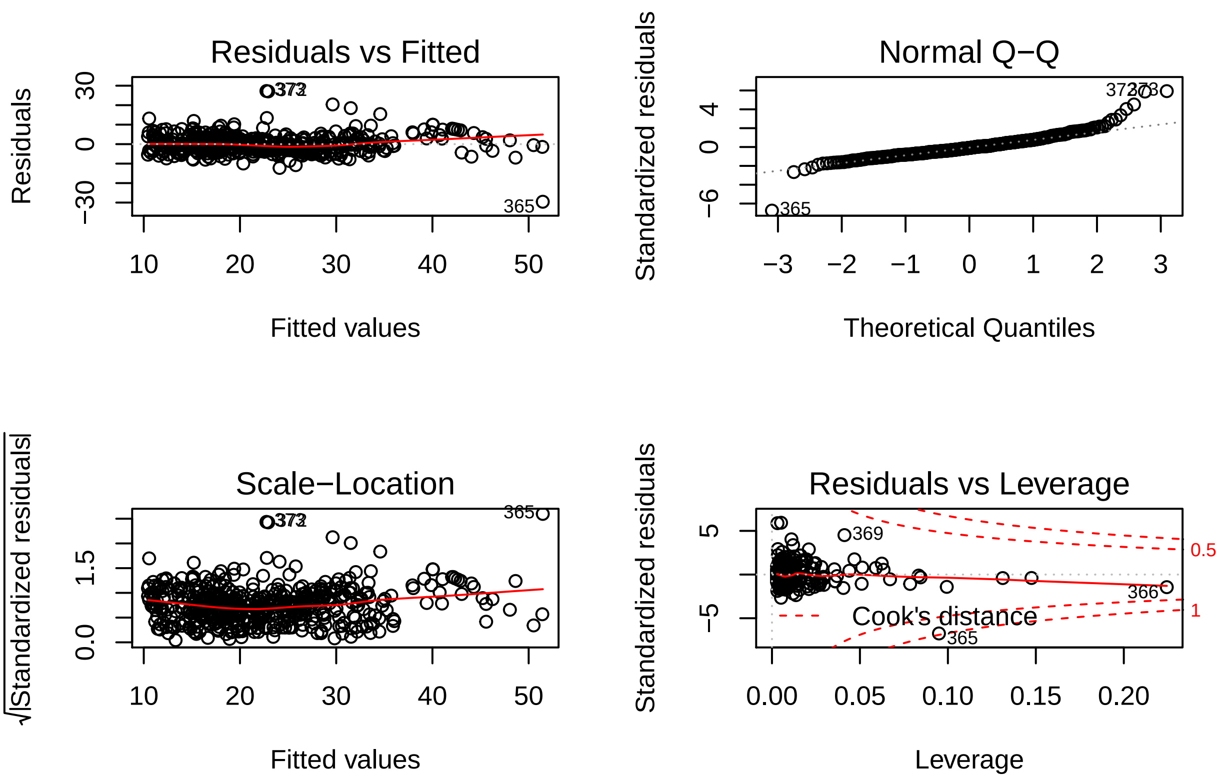
(b)

```r
lm.quad = lm(medv ~ rm + I(rm ^ 2) + lstat + I(lstat ^ 2), data = Boston)
summary(lm.quad)
```

```
##
## Call:
## lm(formula = medv ~ rm + I(rm^2) + lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.5670  -2.8232  -0.4123   2.2523  27.2530
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 105.084032   9.938929  10.573  < 2e-16 ***
## rm          -26.009362   3.103058  -8.382 5.30e-16 ***
## I(rm^2)       2.356069   0.239998   9.817  < 2e-16 ***
## lstat        -1.416229   0.120312 -11.771  < 2e-16 ***
## I(lstat^2)    0.021850   0.003515   6.217 1.07e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 4.608 on 501 degrees of freedom
## Multiple R-squared:  0.751,	Adjusted R-squared:  0.749
## F-statistic: 377.7 on 4 and 501 DF,  p-value: < 2.2e-16
```

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

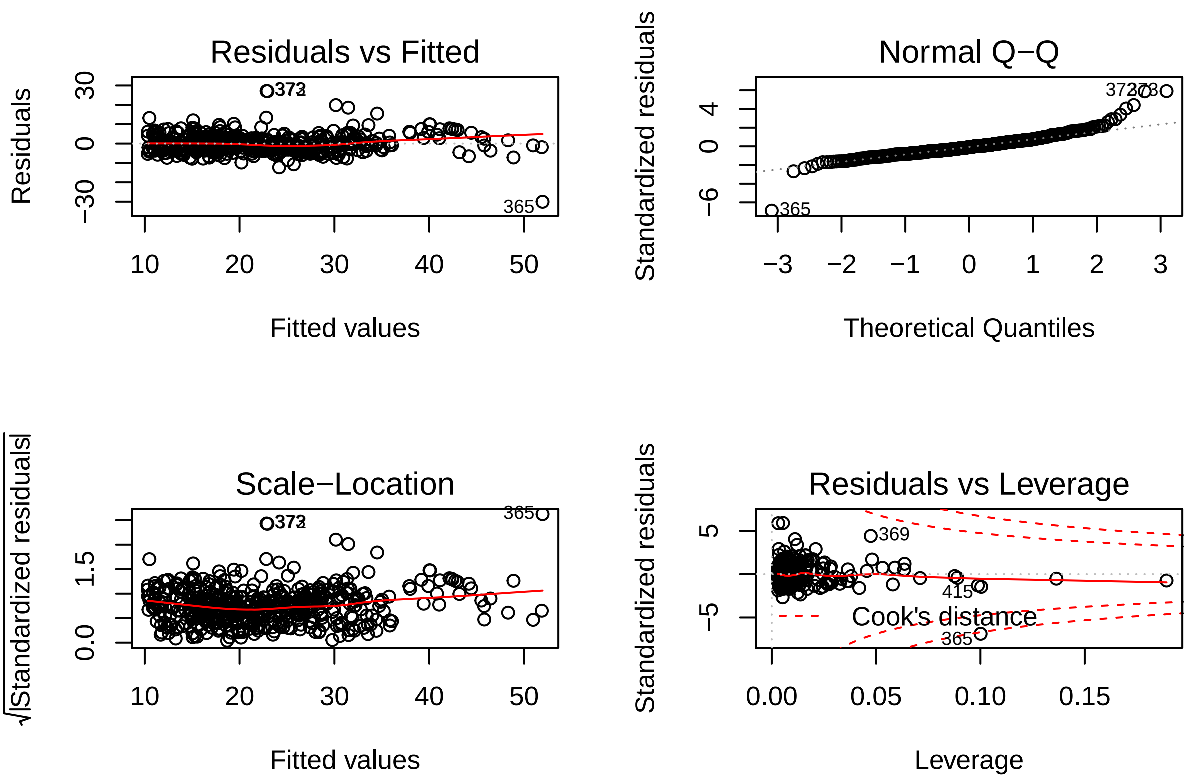



```
### Try fitting model after removing obs. 366
lm.quad.new = lm(medv ~ rm + I(rm ^ 2) + lstat + I(lstat ^ 2), data = Boston[-366,])
summary(lm.quad.new)
```

```
##
## Call:
## lm(formula = medv ~ rm + I(rm^2) + lstat + I(lstat^2), data = Boston[-366,
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -30.0469  -2.7561  -0.4159   2.2261  27.1512
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 112.476905  11.165621  10.074  < 2e-16 ***
## rm          -28.170920   3.440869  -8.187 2.24e-15 ***
## I(rm^2)       2.512307   0.262930   9.555  < 2e-16 ***
## lstat        -1.408802   0.120290 -11.712  < 2e-16 ***
```

```
## I(lstat^2)    0.021209   0.003539   5.994 3.93e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.603 on 500 degrees of freedom
## Multiple R-squared:  0.7519, Adjusted R-squared:  0.7499
## F-statistic: 378.8 on 4 and 500 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm.quad.new)
```



The added quadratic terms improve the variance explained of Median house values from 64% to 75% and all of them being highly significant.

Observation from row 366 in Boston dataset could be an influential variable, as its removal made the R2 values jum,p to 78% and the residence plot being linear with x-axis at 0 value of y-axis.

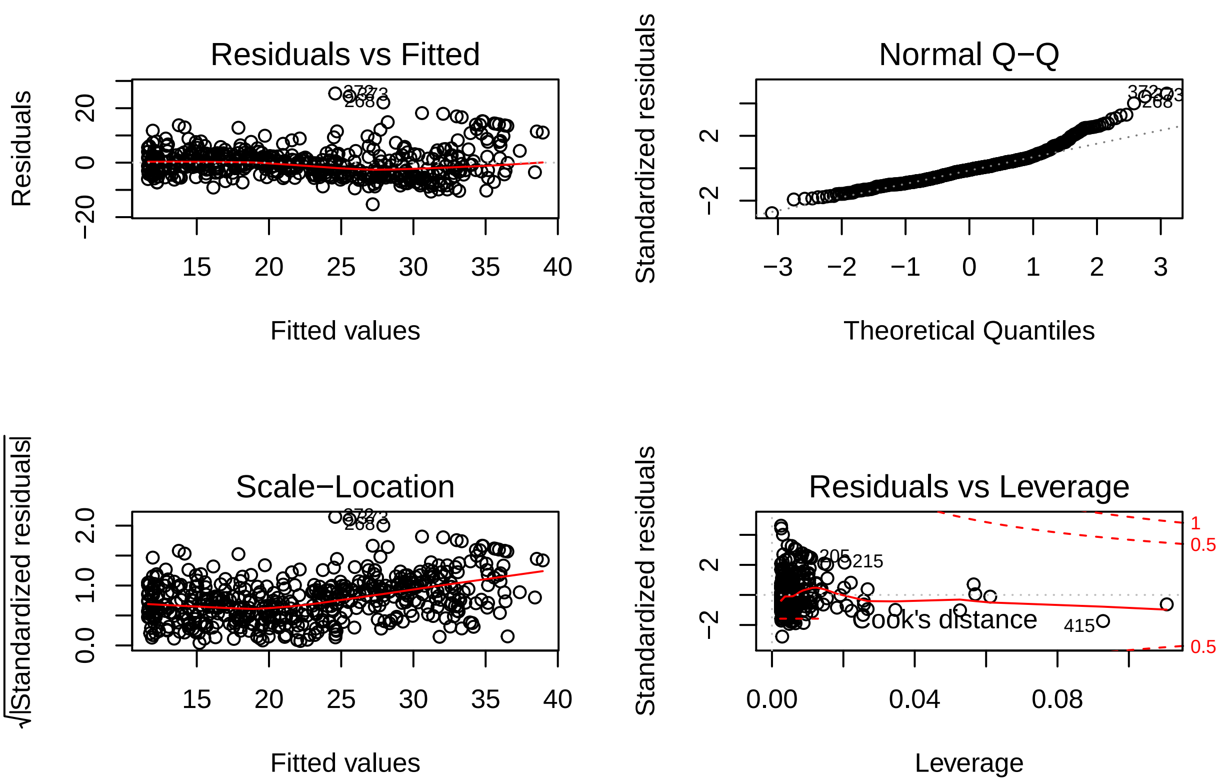
(c)

```
### Figure out the role of inlfuential observations

# "lstat"
lm.lstat = lm(medv ~ lstat + I(lstat ^ 2), data = Boston)
summary(lm.lstat)
```

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

```
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

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

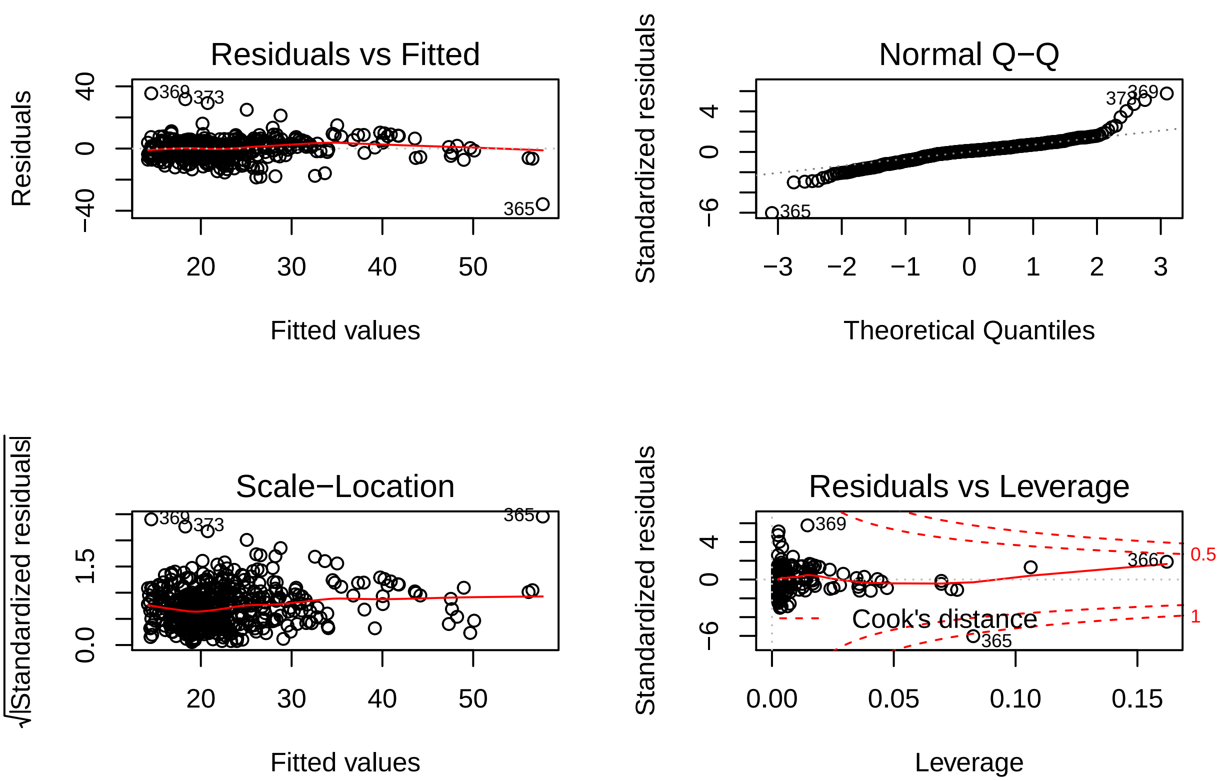



```r
# "rm"
lm.rm = lm(medv ~ rm + I(rm ^ 2), data = Boston)
summary(lm.rm)
```

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

```
## lm(formula = medv ~ rm + I(rm^2), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.769  -2.752   0.619   3.003  35.464
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  66.0588    12.1040   5.458 7.59e-08 ***
## rm          -22.6433     3.7542  -6.031 3.15e-09 ***
## I(rm^2)       2.4701     0.2905   8.502  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.193 on 503 degrees of freedom
## Multiple R-squared:  0.5484, Adjusted R-squared:  0.5466
## F-statistic: 305.4 on 2 and 503 DF,  p-value: < 2.2e-16
```

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



The two models above show differing results with regards to Cook's distance. Obnservation 366, whichwas the iunfliential one in the multiple regression model is NOT so anymore when regression is done with only "rm" variable as a predictor

Problem 3 - SOlution

```
## Generate a set of p=100, P-values using command
pval = c(runif(80), rbeta(20, 1, 100))

## Step by step
sort_pval <- sort(pval)
manual_BH <- 0.5 * (1:length(sort_pval))/length(sort_pval)


## Use function
func_BH <- p.adjust(pval, method="hochberg")
```

Problem 4 - SOlutions

(a)

```
n = 1000
p = 10000
m = 100
b = sqrt( 0.01 / (1 - 0.01) ) #This means each predictor explains 1%
#indicator for non-null effects
eff = c(rep(T, m), rep(F, p-m))
#Prop_adjust <- data.frame(matrix(nrow = 1000, ncol = 1))
#Prop_BF <- data.frame(matrix(nrow = 1000, ncol = 1))
#Prop_BH <- data.frame(matrix(nrow = 1000, ncol = 1))

#for (i in 1:1000) {

#pval[i] = pchisq( rnorm(p, b*sqrt(n)*as.numeric(eff), 1)^2, df = 1, lower = F)

# Non-adjusted p-values
#Prop_adjust[i,] <- length(pval[i][(pval[i]<0.05)])/n

# Bonferroni correction
## Get Bonferroni corrected P-value
#BF_pval = 0.05/p
#Prop_BF[i,] <- length(pval[i][(pval[i]<BF_pval)])/n

# Benjamini and Hochberg FDR at alpha=0.05
#BH_pval = p.adjust(pval[i], "BH")
#Prop_BH[i,] <- length(pval[i][(pval[i]<BH_pval)])/n

#  }


## Histograms
#hist(Prop_adjust)
```

Part (b).

Problem 5.

When we have a large number p of predictors x_j collected to $n \times p$ matrix $\boldsymbol{X}$ that we want to use to predict outcome y, the first step is often to fit p simple linear models of type $y \sim \mu_j + x_j\beta_j$. In lecture 1 we did this by applying `lm()` on each column of $\boldsymbol{X}$ separately:

```
#by mean-centering y and each x, we can ignore intercept terms (since they are 0, see Lecture 0)
X = as.matrix( scale(X, scale = F) ) #mean-centers columns of X to have mean 0
y = as.vector( scale(y, scale = F) )
#apply lm to each column of X separately and without intercept (see Lecture 0.)
lm.res = apply(X, 2 , function(x) summary(lm(y ~ -1 + x))$coeff[1,])
```

In this exercise, we do the same much more efficiently by using direct matrix-vector operations.

Part (a).

We know that the formula for the regression coefficient for mean-centered model is

$$\widehat{\beta}_j = \frac{\boldsymbol{x}_j^T \boldsymbol{y}}{\boldsymbol{x}_j^T \boldsymbol{x}_j}.$$

How can you efficiently compute the vector $\widehat{\boldsymbol{\beta}} = (\widehat{\beta_1}, \ldots, \widehat{\beta_p})^T$ using only matrix-matrix and matrix-vector operations in R? (No for-loops, no apply-functions.)

Demonstrate your method on a data set generated as

```
n = 100
p = 1000
X = matrix(rnorm(n*p, 0, 1), nrow = n)
y = rnorm(n)
```

Compare your β-estimates to the ones given by `apply()` as above to see that they agree (up to precision 1e-16).

Part (b).

Standard errors from the mean-centered simple linear model are of the form

$$s_j = \sqrt{\frac{\sigma_j^2}{\boldsymbol{x}_j^T \boldsymbol{x}_j}},$$

where σ_j^2 is the error variance of the simple linear regression with x_j as the predictor. How can you estimate these efficiently for all j using $\boldsymbol{X}$, $\boldsymbol{y}$ and $\widehat{\boldsymbol{\beta}}$? Again, verify your results by comparing to the standard errors given by `apply()` function.

(Extra: Once everything works, you can test by increasing p from 1,000 to 30,000 that the matrix operations still produce results instantaneously but with `apply()` function it starts to take annoyingly long.)