Pete-Davis–pmd734–Exam.R

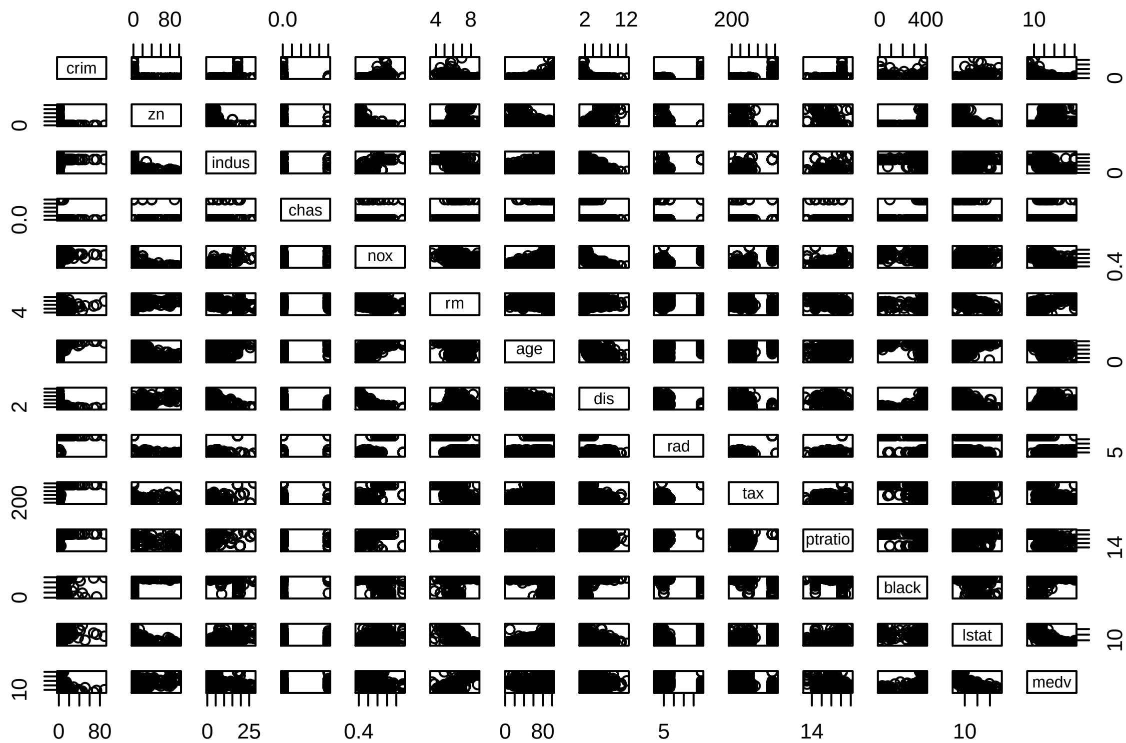
19727

2020-08-02

```r
# BOOK PROBLEMS

## CHAPTER 2: PROBLEM 10
#_____________________________________________________________________________________________
rm(list=ls())


library(MASS)

attach(Boston)

Boston = data.frame(Boston)

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
############
### a
############
n = dim(Boston)

############
### b
############
pairs(Boston)
```




```
############
### c
############
cor.test(crim, zn)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and zn
## t = -4.5938, df = 504, p-value = 5.506e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2826979 -0.1153156
## sample estimates:
##        cor
## -0.2004692
```

```
cor.test(crim, indus)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and indus
## t = 9.9908, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3311512 0.4768518
```

```
## sample estimates:
##       cor
## 0.4065834
```

```r
cor.test(crim, chas)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and chas
## t = -1.2567, df = 504, p-value = 0.2094
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.14236665  0.03143023
## sample estimates:
##         cor
## -0.05589158
```

```r
cor.test(crim, nox)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and nox
## t = 10.419, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3465187 0.4901539
## sample estimates:
##       cor
## 0.4209717
```

```r
cor.test(crim, zn)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and zn
## t = -4.5938, df = 504, p-value = 5.506e-06
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2826979 -0.1153156
## sample estimates:
##        cor
## -0.2004692
```

```r
cor.test(crim, rm)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and rm
## t = -5.0448, df = 504, p-value = 6.347e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3006692 -0.1346514
## sample estimates:
```

```
##        cor
## -0.2192467
```

```r
cor.test(crim, age)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and age
## t = 8.4628, df = 504, p-value = 2.855e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2739900 0.4267805
## sample estimates:
##       cor
## 0.3527343
```

```r
cor.test(crim, dis)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and dis
## t = -9.2135, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4518835 -0.3025132
## sample estimates:
##        cor
## -0.3796701
```

```r
cor.test(crim, rad)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and rad
## t = 17.998, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5693817 0.6758248
## sample estimates:
##       cor
## 0.6255051
```

```r
cor.test(crim, tax)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and tax
## t = 16.099, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5221186 0.6375464
## sample estimates:
##       cor
```

```
## 0.5827643
```

```r
cor.test(crim, ptratio)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and ptratio
## t = 6.8014, df = 504, p-value = 2.943e-11
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2080348 0.3678180
## sample estimates:
##       cor
## 0.2899456
```

```r
cor.test(crim, black)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and black
## t = -9.367, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4568967 -0.3082415
## sample estimates:
##        cor
## -0.3850639
```

```r
cor.test(crim, lstat)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and lstat
## t = 11.491, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3836915 0.5220562
## sample estimates:
##       cor
## 0.4556215
```

```r
cor.test(crim, medv)
```

```
##
##  Pearson's product-moment correlation
##
## data:  crim and medv
## t = -9.4597, df = 504, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4599064 -0.3116859
## sample estimates:
##        cor
## -0.3883046
```

```r
############
### d
############

#### crime
summary(crim)
```

```
##      Min.  1st Qu.   Median      Mean  3rd Qu.     Max.
##  0.00632  0.08204  0.25651  3.61352  3.67708 88.97620
```

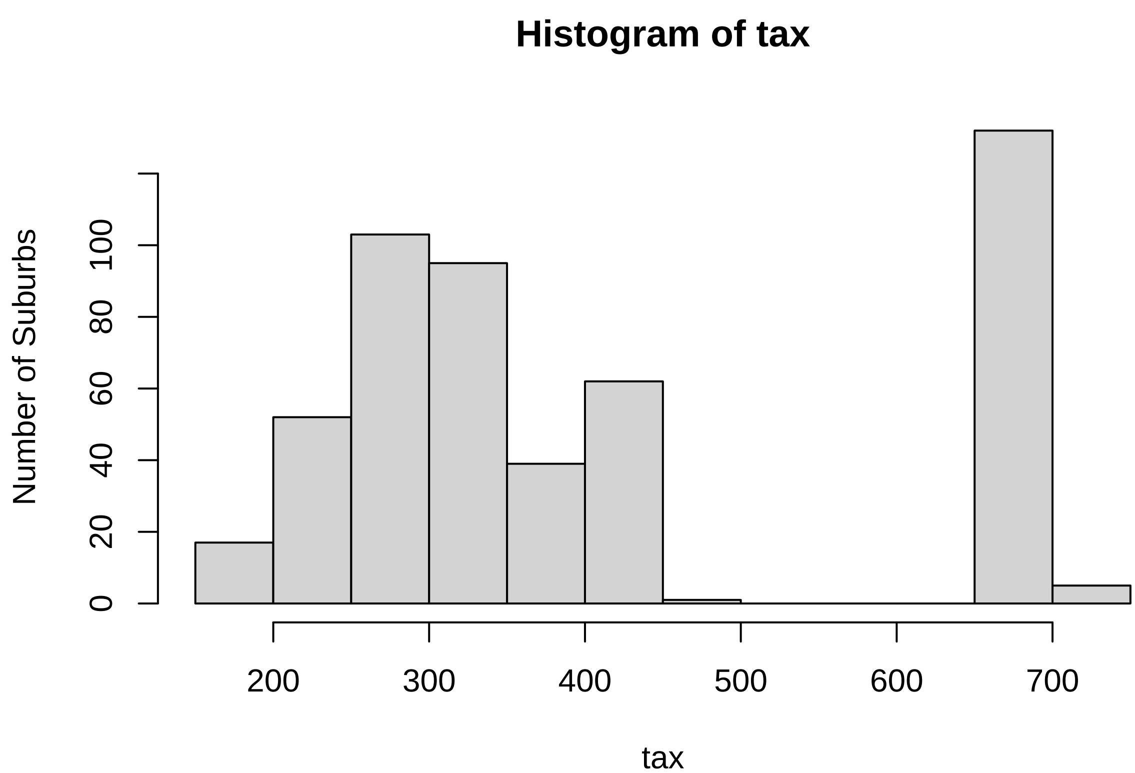
```r
highcrime = subset(Boston, crim > 10)
dim(highcrime)[1] / dim(Boston)[1]
```

```
## [1] 0.1067194
```

```r
#### tax
summary(tax)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   187.0   279.0   330.0   408.2   666.0   711.0
```

```r
hist(tax, ylab = "Number of Suburbs")
```



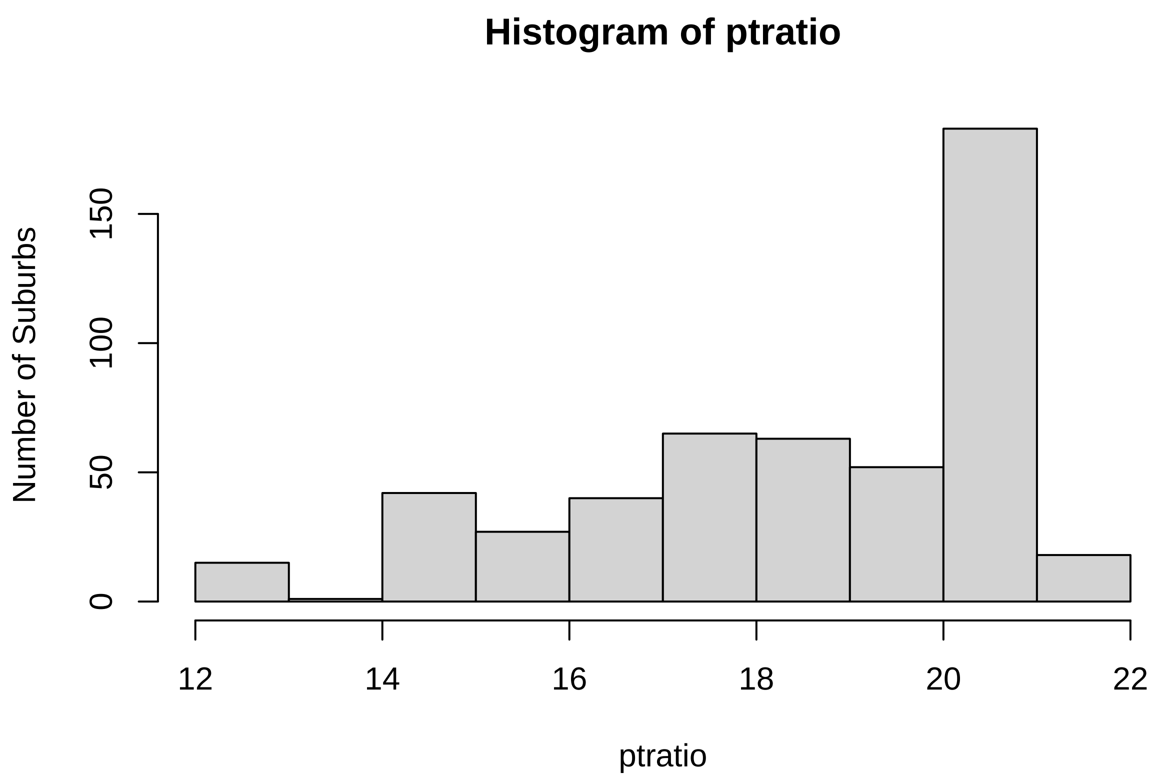

```r
hightax = subset(Boston, tax >= 666)
dim(hightax)[1] / dim(Boston)[1]
```

```
## [1] 0.270751
```

```r
#### pupil-teacher ratio by town
summary(ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   17.40   19.05   18.46   20.20   22.00
```

```r
hist(ptratio, ylab = "Number of Suburbs")
```




```r
############
### e
############
bound = subset(Boston, chas == 1)
dim(bound)[1]
```

```
## [1] 35
```

```r
############
### f
############
summary(ptratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   17.40   19.05   18.46   20.20   22.00
```

```r
############
### g
############
summary(medv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   17.02   21.20   22.53   25.00   50.00
```

```r
Boston[order(medv),][1,]
```

```
##        crim zn indus chas   nox    rm age    dis rad tax ptratio black lstat
## 399 38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 396.9 30.59
##     medv
## 399    5
```

```r
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          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
############
### h
############
dim(subset(Boston, rm > 7)[1])
```

```
## [1] 64  1
```

```r
highrm = (subset(Boston, rm > 8))
dim(highrm)[1]
```

```
## [1] 13
```

```r
summary(highrm)
```

```
##       crim               zn            indus             chas       
##  Min.   :0.02009   Min.   : 0.00   Min.   : 2.680   Min.   :0.0000  
##  1st Qu.:0.33147   1st Qu.: 0.00   1st Qu.: 3.970   1st Qu.:0.0000
```

```
##  Median :0.52014   Median : 0.00   Median : 6.200   Median :0.0000
##  Mean   :0.71879   Mean   :13.62   Mean   : 7.078   Mean   :0.1538
##  3rd Qu.:0.57834   3rd Qu.:20.00   3rd Qu.: 6.200   3rd Qu.:0.0000
##  Max.   :3.47428   Max.   :95.00   Max.   :19.580   Max.   :1.0000
##       nox               rm             age              dis
##  Min.   :0.4161   Min.   :8.034   Min.   : 8.40   Min.   :1.801
##  1st Qu.:0.5040   1st Qu.:8.247   1st Qu.:70.40   1st Qu.:2.288
##  Median :0.5070   Median :8.297   Median :78.30   Median :2.894
##  Mean   :0.5392   Mean   :8.349   Mean   :71.54   Mean   :3.430
##  3rd Qu.:0.6050   3rd Qu.:8.398   3rd Qu.:86.50   3rd Qu.:3.652
##  Max.   :0.7180   Max.   :8.780   Max.   :93.90   Max.   :8.907
##       rad              tax            ptratio          black
##  Min.   : 2.000   Min.   :224.0   Min.   :13.00   Min.   :354.6
##  1st Qu.: 5.000   1st Qu.:264.0   1st Qu.:14.70   1st Qu.:384.5
##  Median : 7.000   Median :307.0   Median :17.40   Median :386.9
##  Mean   : 7.462   Mean   :325.1   Mean   :16.36   Mean   :385.2
##  3rd Qu.: 8.000   3rd Qu.:307.0   3rd Qu.:17.40   3rd Qu.:389.7
##  Max.   :24.000   Max.   :666.0   Max.   :20.20   Max.   :396.9
##      lstat          medv
##  Min.   :2.47   Min.   :21.9
##  1st Qu.:3.32   1st Qu.:41.7
##  Median :4.14   Median :48.3
##  Mean   :4.31   Mean   :44.2
##  3rd Qu.:5.12   3rd Qu.:50.0
##  Max.   :7.44   Max.   :50.0
```

```r
#_____________________________________________________________________________________________________
## CHAPTER 3: PROBLEM 15
#_____________________________________________________________________________________________________
############
### a
############

summary(lm(crim ~ zn, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ zn, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.429 -4.222 -2.620  1.250 84.523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.45369    0.41722  10.675  < 2e-16 ***
## zn          -0.07393    0.01609  -4.594 5.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.435 on 504 degrees of freedom
## Multiple R-squared:  0.04019,    Adjusted R-squared:  0.03828
## F-statistic:  21.1 on 1 and 504 DF,  p-value: 5.506e-06
```

```r
summary(lm(crim ~ indus, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ indus, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.972  -2.698  -0.736   0.712  81.813
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.06374    0.66723  -3.093  0.00209 **
## indus        0.50978    0.05102   9.991  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.866 on 504 degrees of freedom
## Multiple R-squared:  0.1653, Adjusted R-squared:  0.1637
## F-statistic: 99.82 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ chas, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ chas, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.738 -3.661 -3.435  0.018 85.232
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.7444     0.3961   9.453   <2e-16 ***
## chas         -1.8928     1.5061  -1.257    0.209
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.597 on 504 degrees of freedom
## Multiple R-squared:  0.003124,   Adjusted R-squared:  0.001146
## F-statistic: 1.579 on 1 and 504 DF,  p-value: 0.2094
```

```r
summary(lm(crim ~ nox, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ nox, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.371  -2.738  -0.974   0.559  81.728
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -13.720      1.699  -8.073 5.08e-15 ***
```

```
## nox           31.249      2.999  10.419  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.81 on 504 degrees of freedom
## Multiple R-squared:  0.1772, Adjusted R-squared:  0.1756
## F-statistic: 108.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ rm, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ rm, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.604 -3.952 -2.654  0.989 87.197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   20.482      3.365   6.088 2.27e-09 ***
## rm            -2.684      0.532  -5.045 6.35e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.401 on 504 degrees of freedom
## Multiple R-squared:  0.04807,    Adjusted R-squared:  0.04618
## F-statistic: 25.45 on 1 and 504 DF,  p-value: 6.347e-07
```

```r
summary(lm(crim ~ age, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ age, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.789 -4.257 -1.230  1.527 82.849
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.77791    0.94398  -4.002 7.22e-05 ***
## age          0.10779    0.01274   8.463 2.85e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.057 on 504 degrees of freedom
## Multiple R-squared:  0.1244, Adjusted R-squared:  0.1227
## F-statistic: 71.62 on 1 and 504 DF,  p-value: 2.855e-16
```

```r
summary(lm(crim ~ dis, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ dis, data = Boston)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.708 -4.134 -1.527  1.516 81.674
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4993     0.7304  13.006   <2e-16 ***
## dis          -1.5509     0.1683  -9.213   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.965 on 504 degrees of freedom
## Multiple R-squared:  0.1441, Adjusted R-squared:  0.1425
## F-statistic: 84.89 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ rad, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ rad, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.164  -1.381  -0.141   0.660  76.433
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.28716    0.44348  -5.157 3.61e-07 ***
## rad          0.61791    0.03433  17.998  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.718 on 504 degrees of freedom
## Multiple R-squared:  0.3913, Adjusted R-squared:   0.39
## F-statistic: 323.9 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ tax, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ tax, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.513  -2.738  -0.194   1.065  77.696
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.528369   0.815809  -10.45   <2e-16 ***
## tax          0.029742   0.001847   16.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.997 on 504 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3383
```

```
## F-statistic: 259.2 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ ptratio, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ ptratio, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.654 -3.985 -1.912  1.825 83.353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.6469     3.1473  -5.607 3.40e-08 ***
## ptratio       1.1520     0.1694   6.801 2.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.24 on 504 degrees of freedom
## Multiple R-squared:  0.08407,    Adjusted R-squared:  0.08225
## F-statistic: 46.26 on 1 and 504 DF,  p-value: 2.943e-11
```

```r
summary(lm(crim ~ black, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ black, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.756  -2.299  -2.095  -1.296  86.822
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.553529   1.425903  11.609   <2e-16 ***
## black       -0.036280   0.003873  -9.367   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.946 on 504 degrees of freedom
## Multiple R-squared:  0.1483, Adjusted R-squared:  0.1466
## F-statistic: 87.74 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ lstat, data = Boston))
```

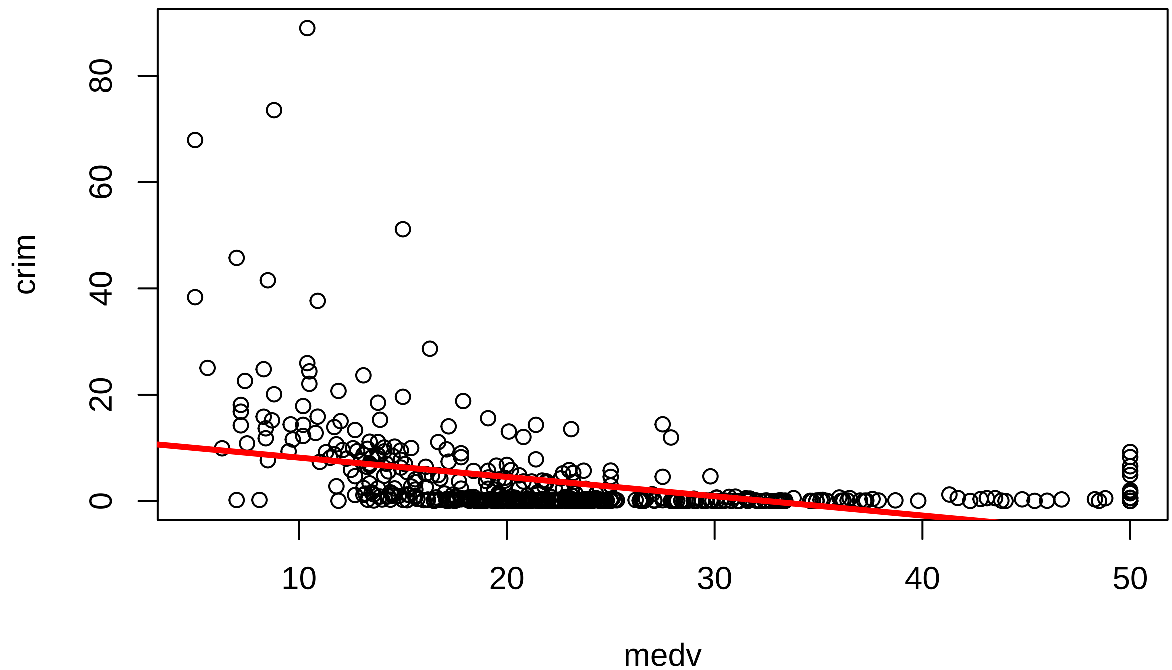
```
##
## Call:
## lm(formula = crim ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.925  -2.822  -0.664   1.079  82.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat        0.54880    0.04776  11.491  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.664 on 504 degrees of freedom
## Multiple R-squared:  0.2076, Adjusted R-squared:  0.206
## F-statistic:   132 on 1 and 504 DF,  p-value: < 2.2e-16
```

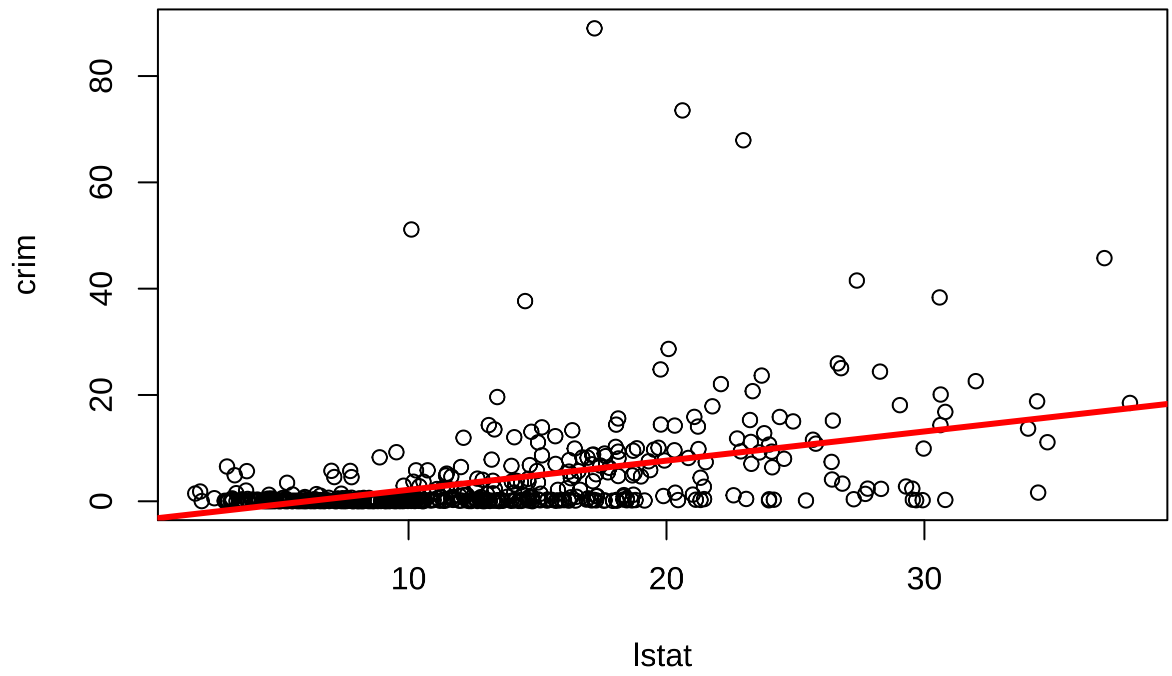
```r
summary(lm(crim ~ medv, data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ medv, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.071 -4.022 -2.343  1.298 80.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316    0.03839   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.934 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

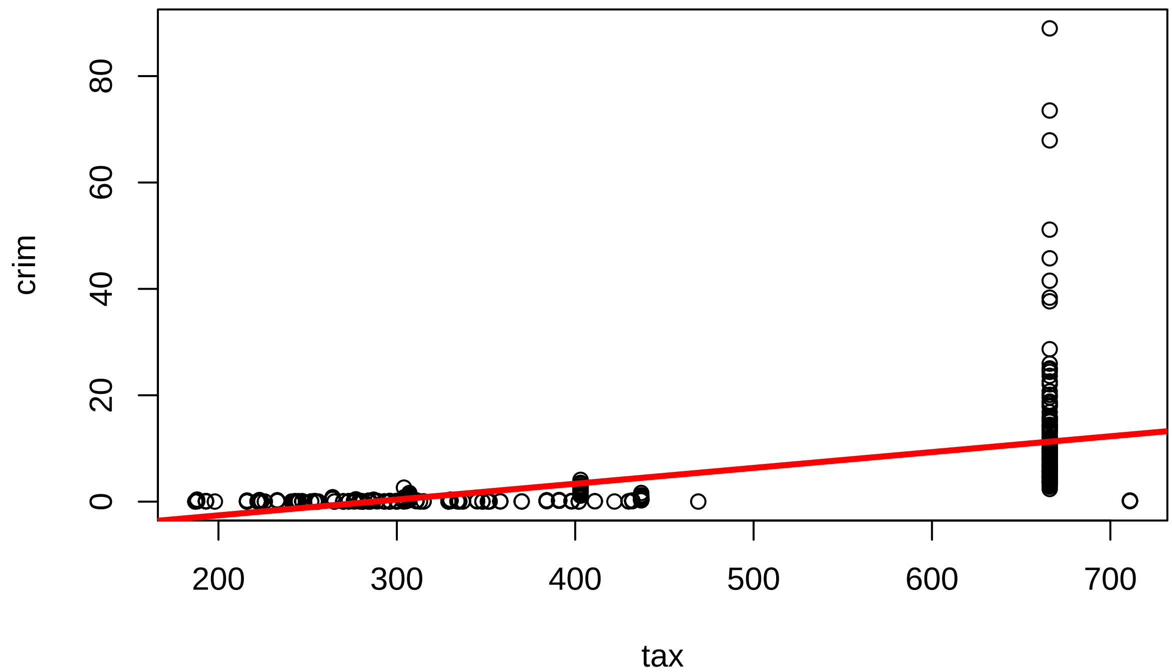
```r
lm.fit=lm(crim~medv)
plot(medv,crim)
abline(lm.fit,lwd = 3,col = "red")
```




```r
lm.fit2=lm(crim~lstat)
plot(lstat,crim)
abline(lm.fit2,lwd = 3,col = "red")
```




```
lm.fit3=lm(crim~tax)
plot(tax,crim)
abline(lm.fit3,lwd = 3,col = "red")
```




```r
############
### b
############
lm.fit4 = lm(crim ~., data = Boston)
summary(lm.fit4)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.924 -2.120 -0.353  1.019 75.051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.033228   7.234903   2.354 0.018949 *
## zn            0.044855   0.018734   2.394 0.017025 *
## indus        -0.063855   0.083407  -0.766 0.444294
## chas         -0.749134   1.180147  -0.635 0.525867
## nox         -10.313535   5.275536  -1.955 0.051152 .
## rm            0.430131   0.612830   0.702 0.483089
## age           0.001452   0.017925   0.081 0.935488
## dis          -0.987176   0.281817  -3.503 0.000502 ***
## rad           0.588209   0.088049   6.680 6.46e-11 ***
## tax          -0.003780   0.005156  -0.733 0.463793
```

```
## ptratio      -0.271081   0.186450  -1.454 0.146611
## black        -0.007538   0.003673  -2.052 0.040702 *
## lstat         0.126211   0.075725   1.667 0.096208 .
## medv         -0.198887   0.060516  -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454,  Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
```

```r
############
### c
############
univcof <- lm(crim ~ zn, data = Boston)$coefficients[2]
univcof <- append(univcof, lm(crim ~ indus, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ chas, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ nox, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ rm, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ age, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ dis, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ rad, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ tax, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ ptratio, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ black, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ lstat, data = Boston)$coefficients[2])
univcof <- append(univcof, lm(crim ~ medv, data = Boston)$coefficients[2])
fooBoston <- (lm(crim ~ . - crim, data = Boston))
fooBoston$coefficients[2:14]
```

```
##           zn         indus          chas           nox            rm
##  0.044855215  -0.063854824  -0.749133611 -10.313534912   0.430130506
##          age           dis           rad           tax       ptratio
##  0.001451643  -0.987175726   0.588208591  -0.003780016  -0.271080558
##        black         lstat          medv
## -0.007537505   0.126211376  -0.198886821
```

```r
plot(univcof, fooBoston$coefficients[2:14], main = "Univariate vs. Multiple Regression Coefficients",
     xlab = "Univariate Coefficient", ylab = "Multiple Coefficient")

############
### c
############
summary(lm(crim ~ zn + I(zn^2) + I(zn^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ zn + I(zn^2) + I(zn^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.821 -4.614 -1.294  0.473 84.130
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.846e+00  4.330e-01  11.192  < 2e-16 ***
```

```
## zn          -3.322e-01  1.098e-01  -3.025  0.00261 **
## I(zn^2)      6.483e-03  3.861e-03   1.679  0.09375 .
## I(zn^3)     -3.776e-05  3.139e-05  -1.203  0.22954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.372 on 502 degrees of freedom
## Multiple R-squared:  0.05824,    Adjusted R-squared:  0.05261
## F-statistic: 10.35 on 3 and 502 DF,  p-value: 1.281e-06
```

```
summary(lm(crim ~ indus + I(indus^2) + I(indus^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ indus + I(indus^2) + I(indus^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.278 -2.514  0.054  0.764 79.713
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.6625683  1.5739833   2.327   0.0204 *
## indus       -1.9652129  0.4819901  -4.077 5.30e-05 ***
## I(indus^2)   0.2519373  0.0393221   6.407 3.42e-10 ***
## I(indus^3)  -0.0069760  0.0009567  -7.292 1.20e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.423 on 502 degrees of freedom
## Multiple R-squared:  0.2597, Adjusted R-squared:  0.2552
## F-statistic: 58.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
summary(lm(crim ~ chas + I(chas^2) + I(chas^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ chas + I(chas^2) + I(chas^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.738 -3.661 -3.435  0.018 85.232
##
## Coefficients: (2 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.7444     0.3961   9.453   <2e-16 ***
## chas         -1.8928     1.5061  -1.257    0.209
## I(chas^2)         NA         NA      NA       NA
## I(chas^3)         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.597 on 504 degrees of freedom
## Multiple R-squared:  0.003124,    Adjusted R-squared:  0.001146
## F-statistic: 1.579 on 1 and 504 DF,  p-value: 0.2094
```

```r
summary(lm(crim ~ nox + I(nox^2) + I(nox^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ nox + I(nox^2) + I(nox^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.110 -2.068 -0.255  0.739 78.302
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   233.09      33.64   6.928 1.31e-11 ***
## nox         -1279.37     170.40  -7.508 2.76e-13 ***
## I(nox^2)     2248.54     279.90   8.033 6.81e-15 ***
## I(nox^3)    -1245.70     149.28  -8.345 6.96e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.234 on 502 degrees of freedom
## Multiple R-squared:  0.297,  Adjusted R-squared:  0.2928
## F-statistic: 70.69 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ rm + I(rm^2) + I(rm^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ rm + I(rm^2) + I(rm^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.485  -3.468  -2.221  -0.015  87.219
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 112.6246    64.5172   1.746   0.0815 .
## rm          -39.1501    31.3115  -1.250   0.2118
## I(rm^2)       4.5509     5.0099   0.908   0.3641
## I(rm^3)      -0.1745     0.2637  -0.662   0.5086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.33 on 502 degrees of freedom
## Multiple R-squared:  0.06779,    Adjusted R-squared:  0.06222
## F-statistic: 12.17 on 3 and 502 DF,  p-value: 1.067e-07
```

```r
summary(lm(crim ~ age + I(age^2) + I(age^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ age + I(age^2) + I(age^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.762 -2.673 -0.516  0.019 82.842
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.549e+00  2.769e+00  -0.920  0.35780
## age          2.737e-01  1.864e-01   1.468  0.14266
## I(age^2)    -7.230e-03  3.637e-03  -1.988  0.04738 *
## I(age^3)     5.745e-05  2.109e-05   2.724  0.00668 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.84 on 502 degrees of freedom
## Multiple R-squared:  0.1742, Adjusted R-squared:  0.1693
## F-statistic: 35.31 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
summary(lm(crim ~ dis + I(dis^2) + I(dis^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ dis + I(dis^2) + I(dis^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.757  -2.588   0.031   1.267  76.378
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.0476     2.4459  12.285  < 2e-16 ***
## dis         -15.5543     1.7360  -8.960  < 2e-16 ***
## I(dis^2)      2.4521     0.3464   7.078 4.94e-12 ***
## I(dis^3)     -0.1186     0.0204  -5.814 1.09e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.331 on 502 degrees of freedom
## Multiple R-squared:  0.2778, Adjusted R-squared:  0.2735
## F-statistic: 64.37 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
summary(lm(crim ~ rad + I(rad^2) + I(rad^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ rad + I(rad^2) + I(rad^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.381  -0.412  -0.269   0.179  76.217
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.605545   2.050108  -0.295    0.768
## rad          0.512736   1.043597   0.491    0.623
## I(rad^2)    -0.075177   0.148543  -0.506    0.613
## I(rad^3)     0.003209   0.004564   0.703    0.482
##
## Residual standard error: 6.682 on 502 degrees of freedom
```

```
## Multiple R-squared:    0.4,  Adjusted R-squared:  0.3965
## F-statistic: 111.6 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ tax + I(tax^2) + I(tax^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ tax + I(tax^2) + I(tax^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.273  -1.389   0.046   0.536  76.950
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.918e+01  1.180e+01   1.626    0.105
## tax         -1.533e-01  9.568e-02  -1.602    0.110
## I(tax^2)     3.608e-04  2.425e-04   1.488    0.137
## I(tax^3)    -2.204e-07  1.889e-07  -1.167    0.244
##
## Residual standard error: 6.854 on 502 degrees of freedom
## Multiple R-squared:  0.3689, Adjusted R-squared:  0.3651
## F-statistic:  97.8 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
summary(lm(crim ~ ptratio + I(ptratio^2) + I(ptratio^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ ptratio + I(ptratio^2) + I(ptratio^3), data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.833 -4.146 -1.655  1.408 82.697
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  477.18405  156.79498   3.043  0.00246 **
## ptratio      -82.36054   27.64394  -2.979  0.00303 **
## I(ptratio^2)   4.63535    1.60832   2.882  0.00412 **
## I(ptratio^3)  -0.08476    0.03090  -2.743  0.00630 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.122 on 502 degrees of freedom
## Multiple R-squared:  0.1138, Adjusted R-squared:  0.1085
## F-statistic: 21.48 on 3 and 502 DF,  p-value: 4.171e-13
```

```r
summary(lm(crim ~ black + I(black^2) + I(black^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ black + I(black^2) + I(black^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.096  -2.343  -2.128  -1.439  86.790
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.826e+01  2.305e+00   7.924  1.5e-14 ***
## black       -8.356e-02  5.633e-02  -1.483    0.139
## I(black^2)   2.137e-04  2.984e-04   0.716    0.474
## I(black^3)  -2.652e-07  4.364e-07  -0.608    0.544
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.955 on 502 degrees of freedom
## Multiple R-squared:  0.1498, Adjusted R-squared:  0.1448
## F-statistic: 29.49 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
summary(lm(crim ~ lstat + I(lstat^2) + I(lstat^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ lstat + I(lstat^2) + I(lstat^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.234  -2.151  -0.486   0.066  83.353
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.2009656  2.0286452   0.592   0.5541
## lstat       -0.4490656  0.4648911  -0.966   0.3345
## I(lstat^2)   0.0557794  0.0301156   1.852   0.0646 .
## I(lstat^3)  -0.0008574  0.0005652  -1.517   0.1299
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.629 on 502 degrees of freedom
## Multiple R-squared:  0.2179, Adjusted R-squared:  0.2133
## F-statistic: 46.63 on 3 and 502 DF,  p-value: < 2.2e-16
```

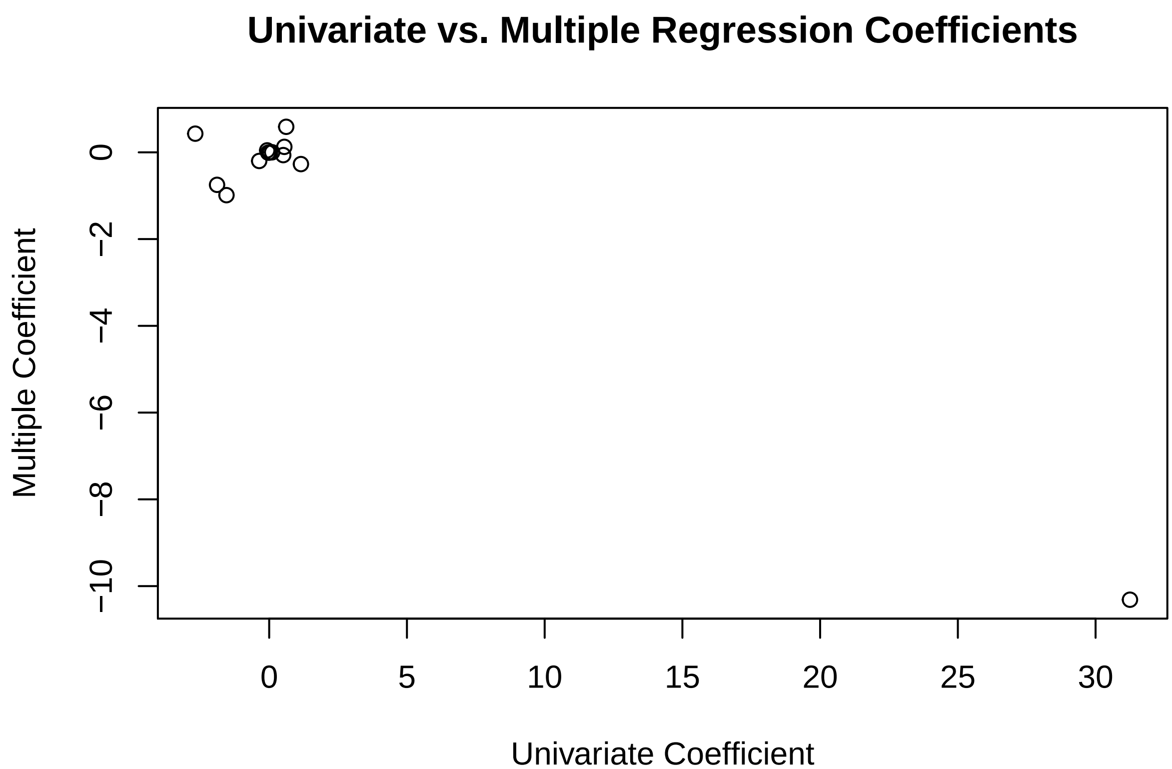
```
summary(lm(crim ~ medv + I(medv^2) + I(medv^3), data = Boston))
```

```
##
## Call:
## lm(formula = crim ~ medv + I(medv^2) + I(medv^3), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.427  -1.976  -0.437   0.439  73.655
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 53.1655381  3.3563105  15.840  < 2e-16 ***
## medv        -5.0948305  0.4338321 -11.744  < 2e-16 ***
## I(medv^2)    0.1554965  0.0171904   9.046  < 2e-16 ***
## I(medv^3)   -0.0014901  0.0002038  -7.312 1.05e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.569 on 502 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4167
## F-statistic: 121.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```
#_____________________________________________________________________________________________
## CHAPTER 6: PROBLEM 9
#_____________________________________________________________________________________________
rm(list=ls()) #Removes every object from your environment

set.seed(1)

############
### a
############

library(ISLR)
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```




```
attach(College)

train = data.frame(College)
test = data.frame(College)
```

```r
n = dim(train)[1]

#Sample (in this case with uniform distribution)
tr = sample(1:777, #The values that will be sampled
            size = 600, #The size of the sample
            replace = FALSE) #without replacement

train = train[tr,] #the rows of train will be the ones sampled
test = test[-tr,] #and test will be everything else (thus, out-of-sample)

preObj <- preProcess(train, method = c('center', 'scale'))

train <- predict(preObj, train)
test <- predict(preObj, test)


############
### b
############

model = lm(Apps ~ ., data = train)
summary(model)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47220 -0.11152  0.00149  0.08434  1.95712
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.079011   0.031318   2.523  0.01191 *
## PrivateYes  -0.106531   0.039717  -2.682  0.00752 **
## Accept       1.052604   0.027919  37.702  < 2e-16 ***
## Enroll      -0.272456   0.048265  -5.645 2.58e-08 ***
## Top10perc    0.238470   0.027905   8.546  < 2e-16 ***
## Top25perc   -0.075054   0.024818  -3.024  0.00260 **
## F.Undergrad  0.091719   0.044116   2.079  0.03805 *
## P.Undergrad  0.026672   0.013936   1.914  0.05612 .
## Outstate    -0.085439   0.021722  -3.933 9.39e-05 ***
## Room.Board   0.042331   0.015174   2.790  0.00545 **
## Books        0.008789   0.011267   0.780  0.43565
## Personal     0.001930   0.012105   0.159  0.87339
## PhD         -0.045952   0.021266  -2.161  0.03112 *
## Terminal     0.001104   0.021080   0.052  0.95826
## S.F.Ratio    0.019977   0.015124   1.321  0.18708
## perc.alumni  0.003823   0.014670   0.261  0.79447
## Expend       0.082223   0.018369   4.476 9.15e-06 ***
## Grad.Rate    0.030948   0.014625   2.116  0.03476 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.2605 on 582 degrees of freedom
## Multiple R-squared:  0.9341, Adjusted R-squared:  0.9322
## F-statistic: 485.1 on 17 and 582 DF,  p-value: < 2.2e-16
```

```r
pred = predict(model, test)
RMSE_linear = sqrt(mean((test$Apps-pred)^2))
print(RMSE_linear)
```

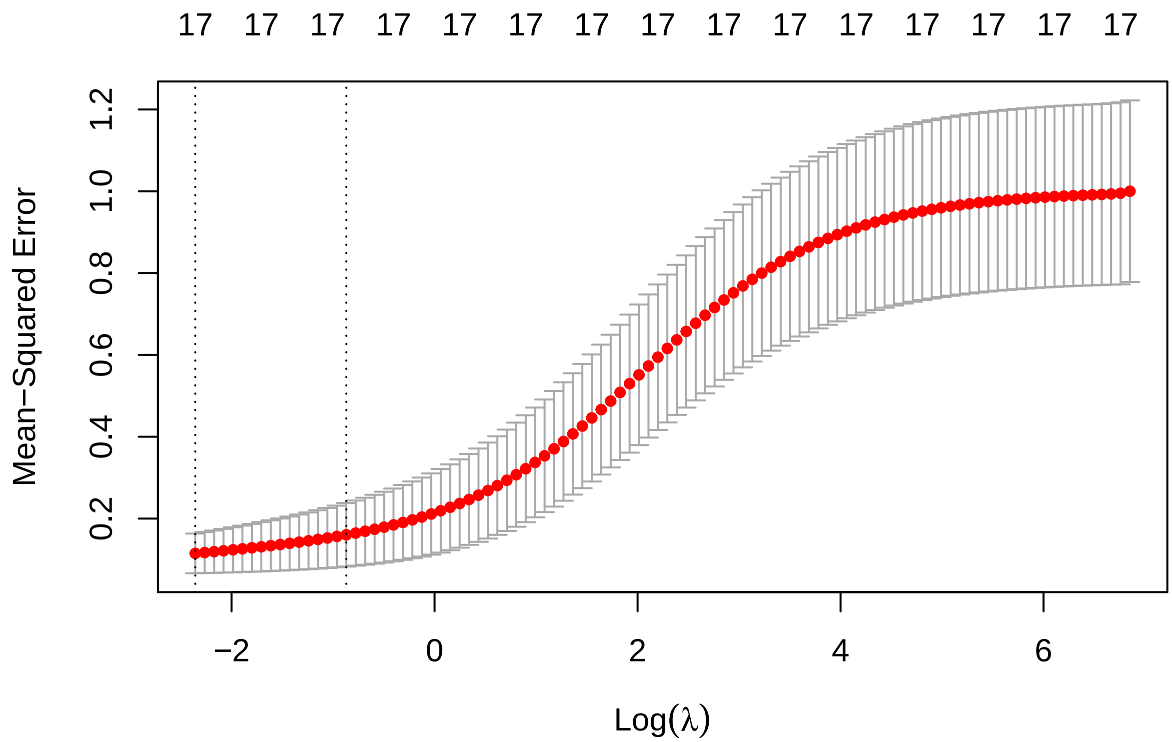
```
## [1] 0.3109753
```

```r
############
### c
############

library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

```r
xtrain = model.matrix (Apps ~ ., train)[,-1]
ytrain = train$Apps
xtest = model.matrix(Apps ~ ., test)[,-1]
ytest = test$Apps

      #ridge regression !
set.seed (1)
CV.R = cv.glmnet(xtrain, ytrain, alpha = 0)
plot(CV.R)
```




```
# finding the best lambda from the cross validation
R.minlam = CV.R$lambda.min
print(R.minlam)
```

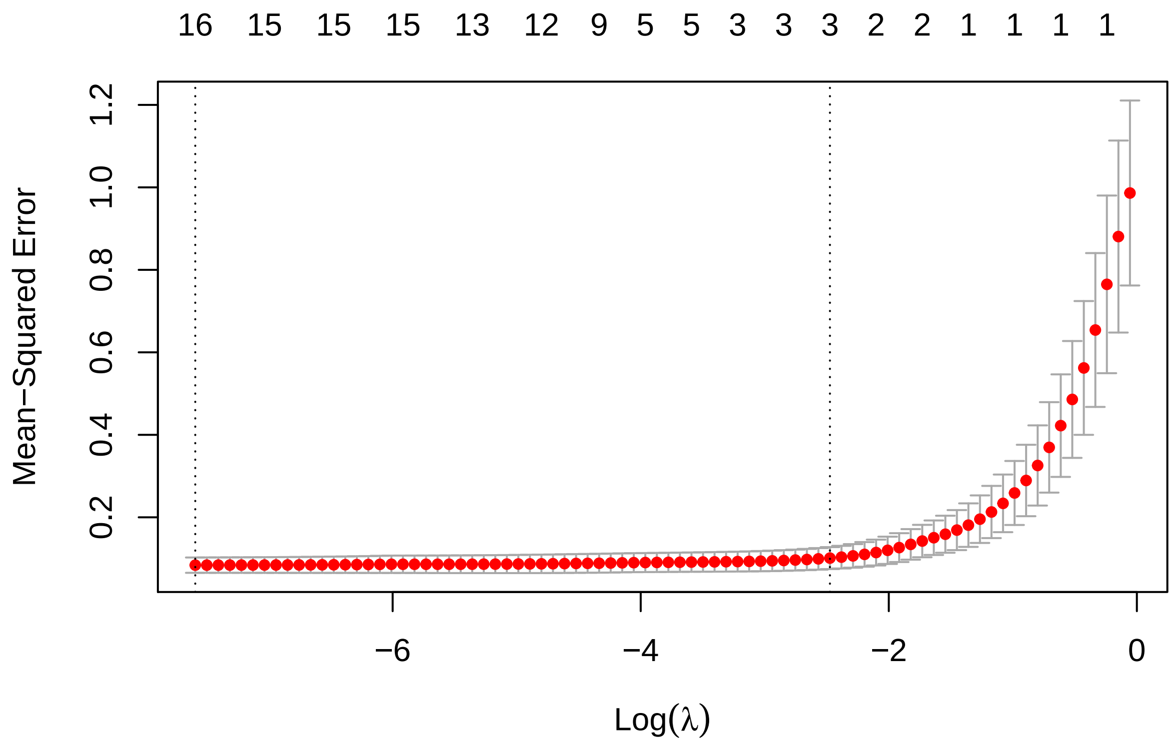
```
## [1] 0.09459227
```

```
# Creating training model using ridge regression!
model.R = glmnet(xtrain, ytrain, alpha=0,lambda = R.minlam)


pred = predict(model.R, s = R.minlam, newx = xtest)
# Calculating Accuracy
RMSE_ridge = sqrt(mean((test$Apps-pred)^2))
print(RMSE_ridge)
```

```
## [1] 0.2946376
```

```
############
### d
############

xtrain = model.matrix (Apps ~ ., train)[,-1]
ytrain = train$Apps
xtest = model.matrix(Apps ~ ., test)[,-1]
ytest = test$Apps

#lasso regression !
set.seed (1)
```

```
CV.L = cv.glmnet(xtrain, ytrain, alpha = 1)
plot(CV.L)
```




```
# finding the best lambda from the cross validation
L.minlam = CV.L$lambda.min
print(L.minlam)
```

```
## [1] 0.0005048105
```

```
# Creating training model using ridge regression!
model.L = glmnet(xtrain, ytrain, alpha = 1,lambda = L.minlam)


pred = predict(model.L, s = L.minlam, newx = xtest)
# Calculating Accuracy
RMSE_lasso = sqrt(mean((test$Apps-pred)^2))
print(RMSE_lasso)
```

```
## [1] 0.3095178
```

```
# see the number of non-zero coefficients
coef.L = predict(CV.L,type="coefficients",s=L.minlam)[1:length(model.L$beta),]
coef.L[coef.L != 0]
```

```
##  (Intercept)   PrivateYes       Accept       Enroll    Top10perc    Top25perc
##  0.078346018 -0.105635081  1.043147424 -0.246983852  0.231625523 -0.069057287
##  F.Undergrad  P.Undergrad     Outstate   Room.Board        Books     Personal
##  0.075301063  0.026230568 -0.082423691  0.041796920  0.008386538  0.001448003
```

```
##          PhD    S.F.Ratio  perc.alumni       Expend
## -0.043990552  0.018782824  0.001483850  0.081435171
```

```
############
### e
############

library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings
```

```
#pcr method
pcr_fit <- train(x = xtrain, y = ytrain,
                   method = 'pcr',
                   trControl = trainControl(method = 'cv', number = 10),
                   tuneGrid = expand.grid(ncomp = 1:10))
#this will show the error of the prediction
(pcr_info = postResample(predict(pcr_fit, xtest), ytest))
```

```
##      RMSE  Rsquared       MAE
## 0.3771579 0.8801608 0.2238508
```

```
# this will show a summary of the prediction with the number of components
summary(pcr_fit)
```

```
## Data:    X dimension: 600 17
##  Y dimension: 600 1
## Fit method: svdpc
## Number of components considered: 10
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           33.28    57.12    64.59    70.44    76.01    81.16    84.99
## .outcome    10.07    73.52    73.77    80.76    83.68    83.73    83.88
##           8 comps  9 comps  10 comps
## X           88.51    91.65     93.96
## .outcome    84.86    85.28     85.45
```

```
############
### f
############

#pls method
pls_fit <- train(x = xtrain, y = ytrain,
                 method = 'pls',
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = expand.grid(ncomp = 1:10))
#this will show the error of the prediction
(pls_info = postResample(predict(pls_fit, xtest), ytest))
```

```
##      RMSE  Rsquared       MAE
## 0.3091268 0.9139673 0.1723655
```

```
# this will show a summary of the prediction with the number of components
summary(pls_fit)
```

```
## Data:    X dimension: 600 17
##  Y dimension: 600 1
## Fit method: oscorespls
## Number of components considered: 10
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           25.91    53.00    62.65    65.36    69.71    74.32    77.89
## .outcome    77.27    82.13    87.46    90.95    92.56    93.07    93.17
##           8 comps  9 comps  10 comps
## X           81.16    83.29     86.22
## .outcome    93.27    93.35     93.37
```

```
############
### g
############
# compared using data I already calculated on questions above!



#____________________________________________________________________________________________________
## CHAPTER 6: PROBLEM 11
#____________________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment

set.seed(1)
library(ISLR)
library(caret)
library(MASS)
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 11):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn
```

```
############
### a
############

train = data.frame(Boston)
test = data.frame(Boston)

n = dim(train)[1]

#Sample (in this case with uniform distribution)
tr = sample(1:506, #The values that will be sampled
            size = 400, #The size of the sample
            replace = FALSE) #without replacement
```

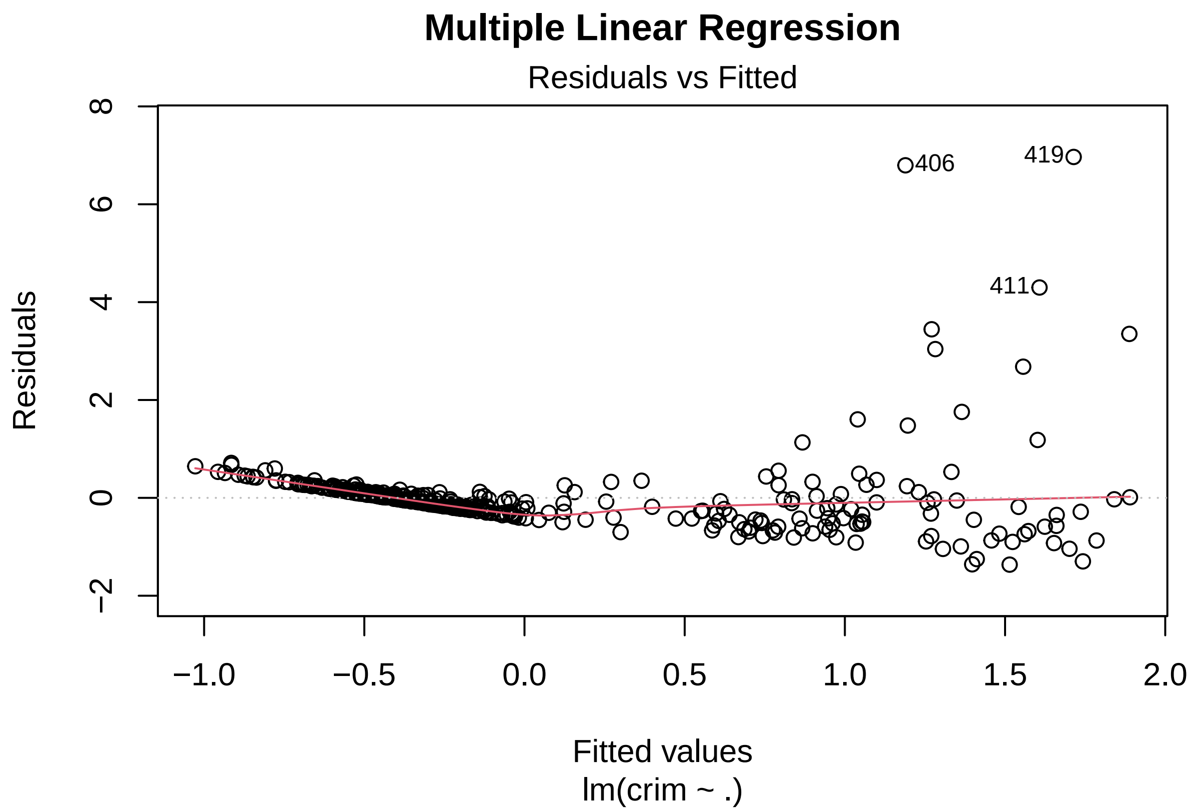
```r
train = train[tr,] #the rows of train will be the ones sampled
test = test[-tr,] #and test will be everything else (thus, out-of-sample)

preObj <- preProcess(train, method = c('center', 'scale'))

train <- predict(preObj, train)
test <- predict(preObj, test)


###### Multiple Linear Regression

model = lm(crim ~ ., data = train)
summary(model)
```

```
##
## Call:
## lm(formula = crim ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3645 -0.2236 -0.0413  0.1088  6.9664
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.775e-16  3.672e-02   0.000 1.000000
## zn           1.019e-01  5.494e-02   1.854 0.064439 .
## indus       -5.748e-02  7.185e-02  -0.800 0.424263
## chas        -2.022e-02  3.882e-02  -0.521 0.602847
## nox         -1.286e-01  7.754e-02  -1.658 0.098055 .
## rm          -1.027e-02  5.613e-02  -0.183 0.854917
## age          2.143e-02  6.510e-02   0.329 0.742151
## dis         -2.014e-01  7.641e-02  -2.635 0.008749 **
## rad          5.666e-01  9.713e-02   5.833 1.15e-08 ***
## tax         -5.929e-02  1.087e-01  -0.546 0.585712
## ptratio     -7.283e-02  5.185e-02  -1.405 0.160923
## black       -1.672e-01  4.375e-02  -3.822 0.000154 ***
## lstat        1.081e-01  6.838e-02   1.581 0.114783
## medv        -1.550e-01  7.149e-02  -2.168 0.030745 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7344 on 386 degrees of freedom
## Multiple R-squared:  0.4783, Adjusted R-squared:  0.4607
## F-statistic: 27.22 on 13 and 386 DF,  p-value: < 2.2e-16
```

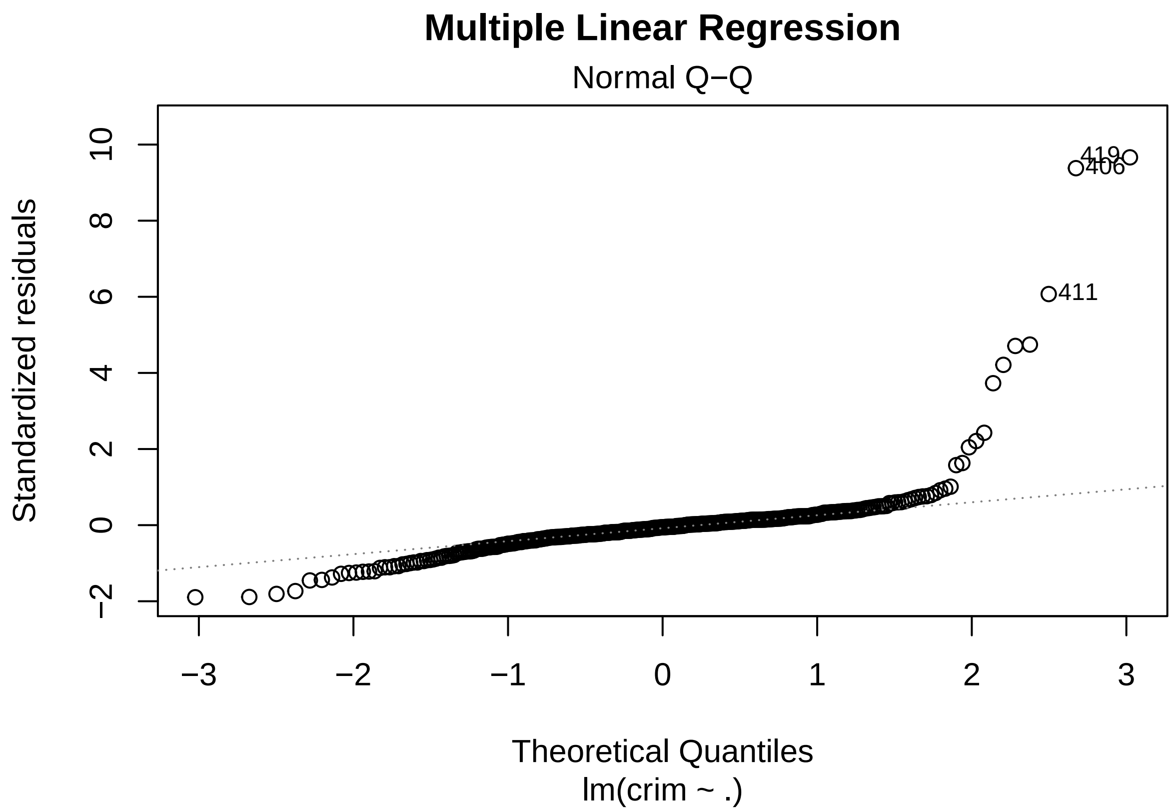
```r
pred = predict(model, test)
RMSE_linear = sqrt(mean((test$crim-pred)^2))
print(RMSE_linear)
```

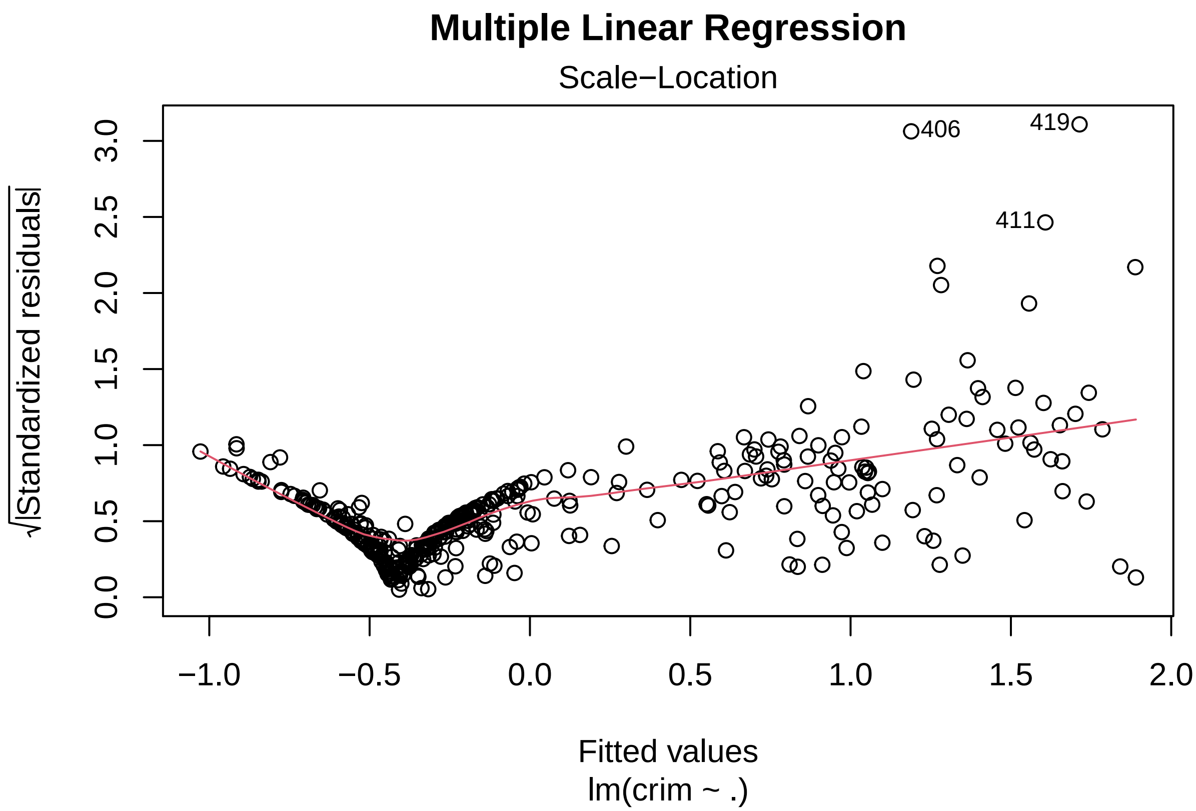
```
## [1] 1.012721
```

```r
plot(model, main = "Multiple Linear Regression")
```

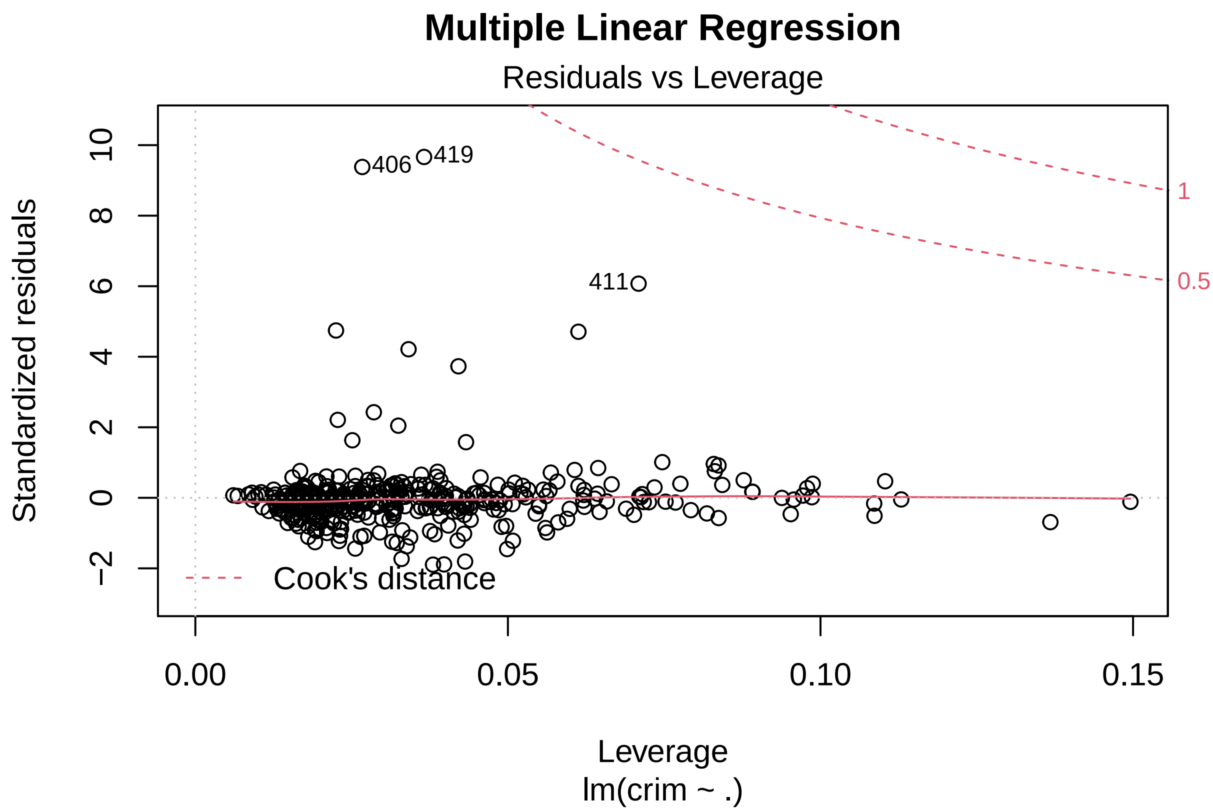


Multiple Linear Regression
Residuals vs Fitted
Residuals
8
6
4
2
0
−2
406
419
411
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Fitted values
lm(crim ~ .)

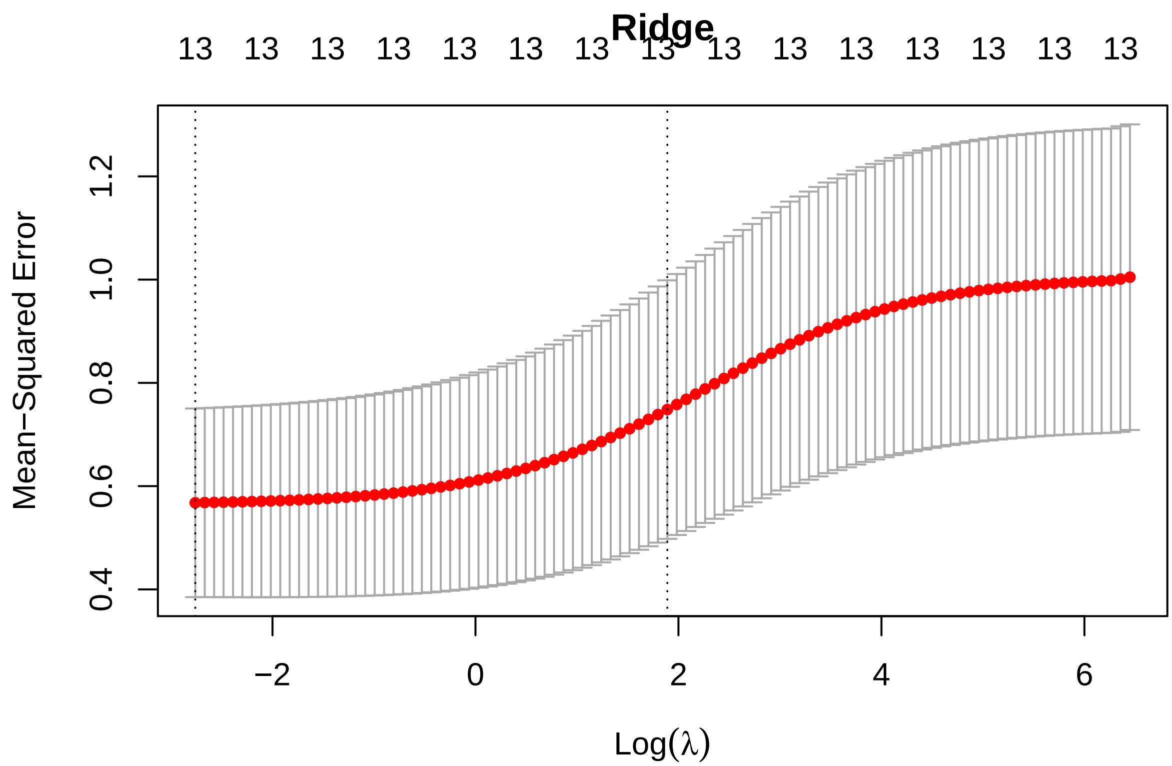

Multiple Linear Regression
Normal Q−Q
Standardized residuals
−2
0
2
4
6
8
10
419
406
411
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(crim ~ .)



Multiple Linear Regression
Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
406
419
411
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Fitted values
lm(crim ~ .)




```
###### Ridge

library(glmnet)

xtrain = model.matrix (crim ~ ., train)[,-1]
ytrain = train$crim
xtest = model.matrix(crim ~ ., test)[,-1]
ytest = test$crim

#ridge regression !
set.seed (1)
CV.R = cv.glmnet(xtrain, ytrain, alpha = 0)
plot(CV.R, main = "Ridge")
```



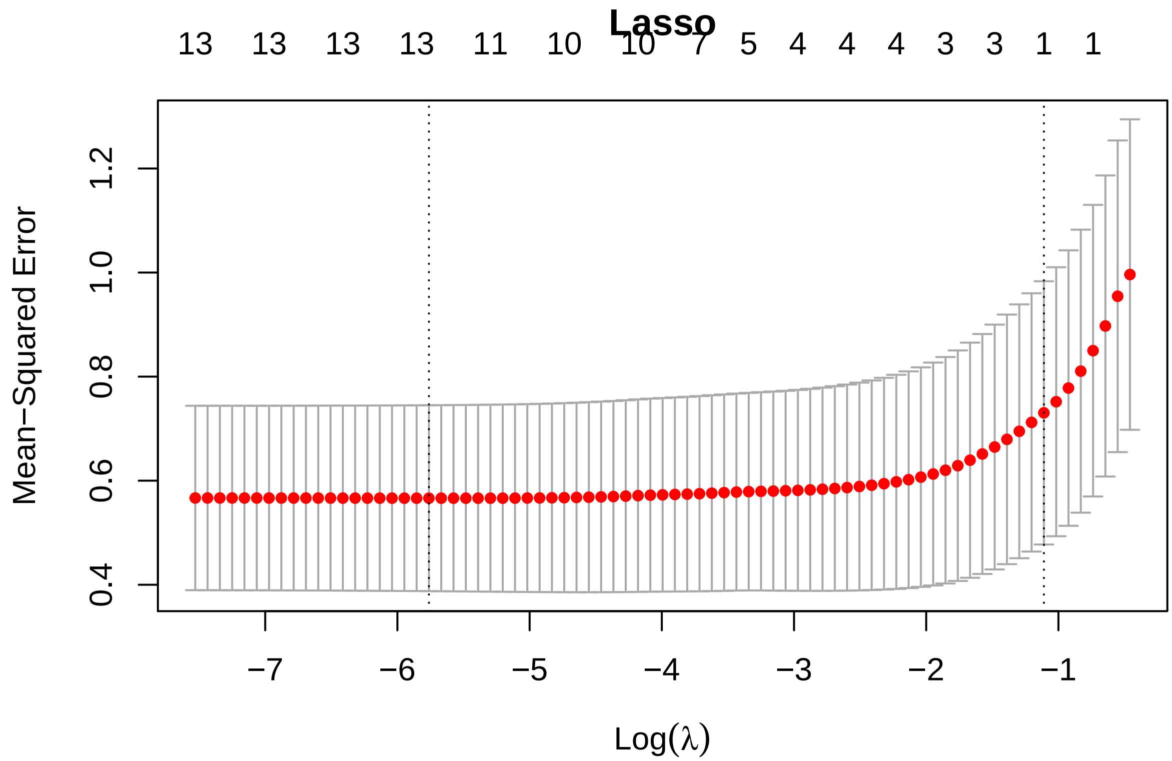

```
# finding the best lambda from the cross validation
R.minlam = CV.R$lambda.min
print(R.minlam)
```

```
## [1] 0.0632114
```

```
# Creating training model using ridge regression!
model.R = glmnet(xtrain, ytrain, alpha=0,lambda = R.minlam)


pred = predict(model.R, s = R.minlam, newx = xtest)
# Calculating Accuracy
RMSE_ridge = sqrt(mean((test$crim-pred)^2))
print(RMSE_ridge)
```

```
## [1] 1.024805
```

```
######## Lasso

xtrain = model.matrix (crim ~ ., train)[,-1]
ytrain = train$crim
xtest = model.matrix(crim ~ ., test)[,-1]
ytest = test$crim

#lasso regression !
set.seed (1)
CV.L = cv.glmnet(xtrain, ytrain, alpha = 1)
plot(CV.L, main = 'Lasso')
```




```r
# finding the best lambda from the cross validation
L.minlam = CV.L$lambda.min
print(L.minlam)
```

```
## [1] 0.003146046
```

```r
# Creating training model using ridge regression!
model.L = glmnet(xtrain, ytrain, alpha = 1,lambda = L.minlam)


pred = predict(model.L, s = L.minlam, newx = xtest)
# Calculating Accuracy
RMSE_lasso = sqrt(mean((test$crim-pred)^2))
print(RMSE_lasso)
```

```
## [1] 1.014697
```

```r
# see the number of non-zero coefficients
coef.L = predict(CV.L,type="coefficients",s=L.minlam)[1:length(model.L$beta),]
coef.L[coef.L != 0]
```

```
##   (Intercept)            zn         indus          chas           nox
## -4.949752e-17  8.868277e-02 -5.914716e-02 -1.605760e-02 -1.044215e-01
##            rm           age           dis           rad           tax
## -2.029476e-03  9.477202e-03 -1.784304e-01  5.187712e-01 -1.215178e-02
##       ptratio         black         lstat
## -6.064609e-02 -1.657503e-01  1.155395e-01
```

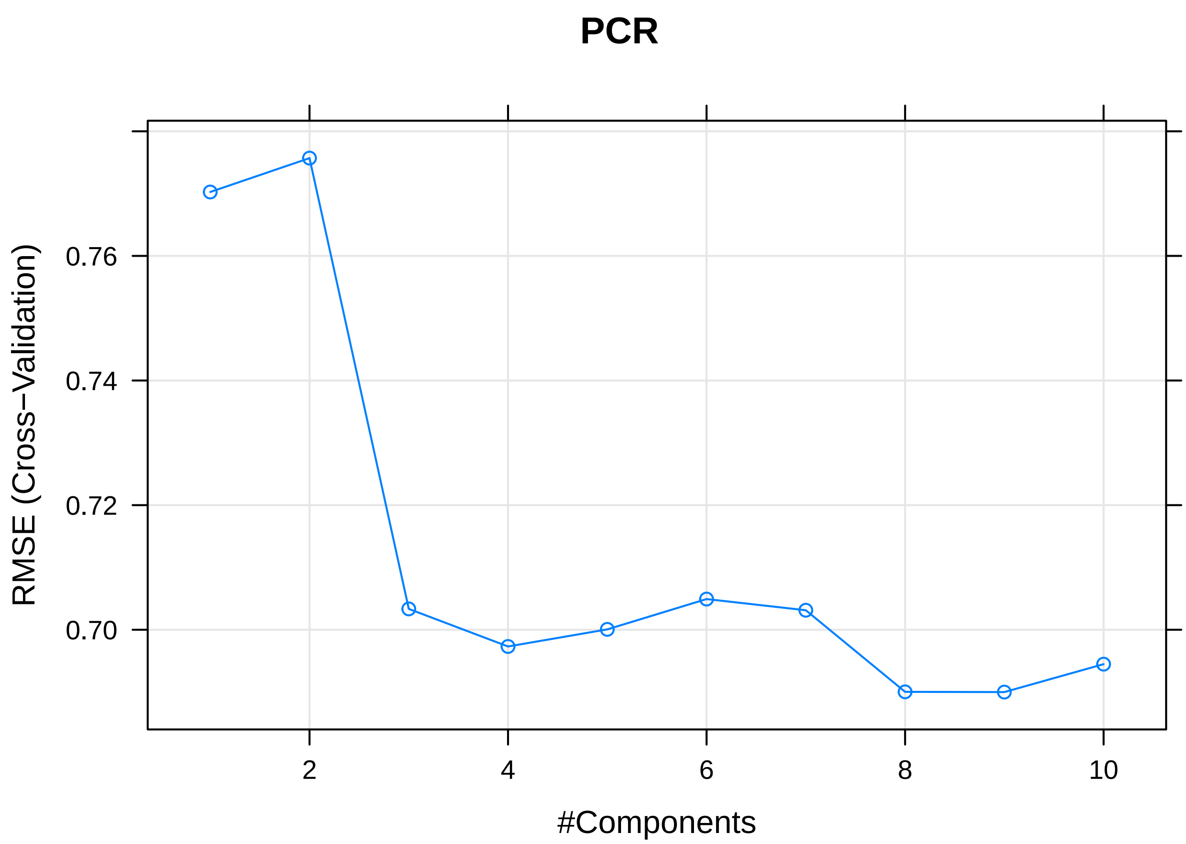
```r
######## PCR

library(pls)

#pcr method
pcr_fit <- train(x = xtrain, y = ytrain,
                 method = 'pcr',
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = expand.grid(ncomp = 1:10))
#this will show the error of the prediction
(pcr_info = postResample(predict(pcr_fit, xtest), ytest))
```

```
##      RMSE  Rsquared       MAE
## 1.0462918 0.3448423 0.3602000
```

```r
# this will show a summary of the prediction with the number of components
summary(pcr_fit)
```

```
## Data:    X dimension: 400 13
##  Y dimension: 400 1
## Fit method: svdpc
## Number of components considered: 9
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           48.13    60.81    70.11    76.79    82.95    87.85    91.12
## .outcome    32.00    32.24    42.24    43.15    43.20    43.20    43.47
##           8 comps  9 comps
## X           93.41    95.31
## .outcome    45.34    45.68
```

```r
plot(pcr_fit, main = "PCR")
```



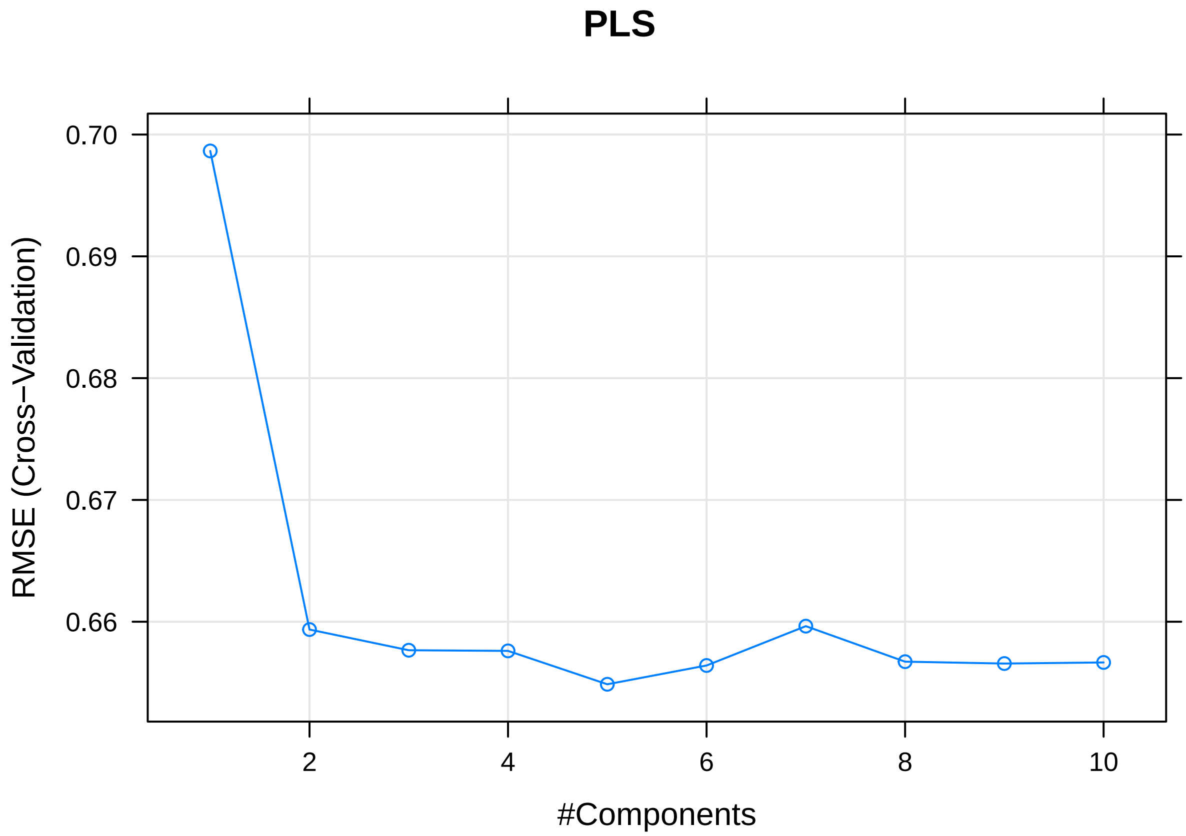

```
####### PLS

#pls method
pls_fit <- train(x = xtrain, y = ytrain,
                 method = 'pls',
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = expand.grid(ncomp = 1:10))
#this will show the error of the prediction
(pls_info = postResample(predict(pls_fit, xtest), ytest))
```

```
##      RMSE  Rsquared       MAE
## 1.0172880 0.3843102 0.3478014
```

```
# this will show a summary of the prediction with the number of components
summary(pls_fit)
```

```
## Data:    X dimension: 400 13
##  Y dimension: 400 1
## Fit method: oscorespls
## Number of components considered: 5
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps
## X           47.63    57.25    62.68    71.57    77.93
## .outcome    36.27    44.84    46.57    47.16    47.41
```

```r
plot(pls_fit, main = "PLS")
```




```r
############
### b
############
print(RMSE_linear)
```

```
## [1] 1.012721
```

```r
print(RMSE_ridge)
```

```
## [1] 1.024805
```

```r
print(RMSE_lasso)
```

```
## [1] 1.014697
```

```r
print(pcr_info)
```

```
##      RMSE  Rsquared       MAE
## 1.0462918 0.3448423 0.3602000
```

```r
print(pls_info)
```

```
##      RMSE  Rsquared       MAE
## 1.0172880 0.3843102 0.3478014
```

```r
############
### c
############
```

```r
summary(model)
```

```
##
## Call:
## lm(formula = crim ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3645 -0.2236 -0.0413  0.1088  6.9664
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.775e-16  3.672e-02   0.000 1.000000
## zn           1.019e-01  5.494e-02   1.854 0.064439 .
## indus       -5.748e-02  7.185e-02  -0.800 0.424263
## chas        -2.022e-02  3.882e-02  -0.521 0.602847
## nox         -1.286e-01  7.754e-02  -1.658 0.098055 .
## rm          -1.027e-02  5.613e-02  -0.183 0.854917
## age          2.143e-02  6.510e-02   0.329 0.742151
## dis         -2.014e-01  7.641e-02  -2.635 0.008749 **
## rad          5.666e-01  9.713e-02   5.833 1.15e-08 ***
## tax         -5.929e-02  1.087e-01  -0.546 0.585712
## ptratio     -7.283e-02  5.185e-02  -1.405 0.160923
## black       -1.672e-01  4.375e-02  -3.822 0.000154 ***
## lstat        1.081e-01  6.838e-02   1.581 0.114783
## medv        -1.550e-01  7.149e-02  -2.168 0.030745 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7344 on 386 degrees of freedom
## Multiple R-squared:  0.4783, Adjusted R-squared:  0.4607
## F-statistic: 27.22 on 13 and 386 DF,  p-value: < 2.2e-16
```

```r
#___________________________________________________________________________________________________________
## CHAPTER 4: PROBLEM 10    ------> omit e and f
#___________________________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment

set.seed(1)
library(ISLR)
library(caret)
library(kknn)
```

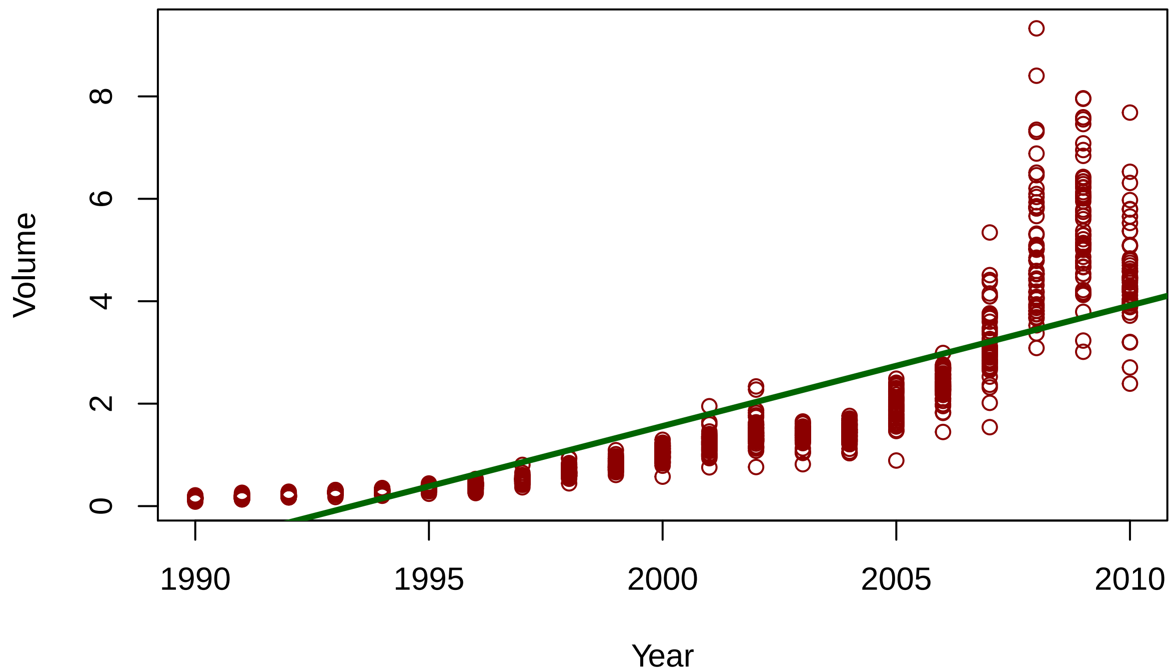
```
##
## Attaching package: 'kknn'

## The following object is masked from 'package:caret':
##
##     contr.dummy
```

```r
library(class)
attach(Weekly)

############
```

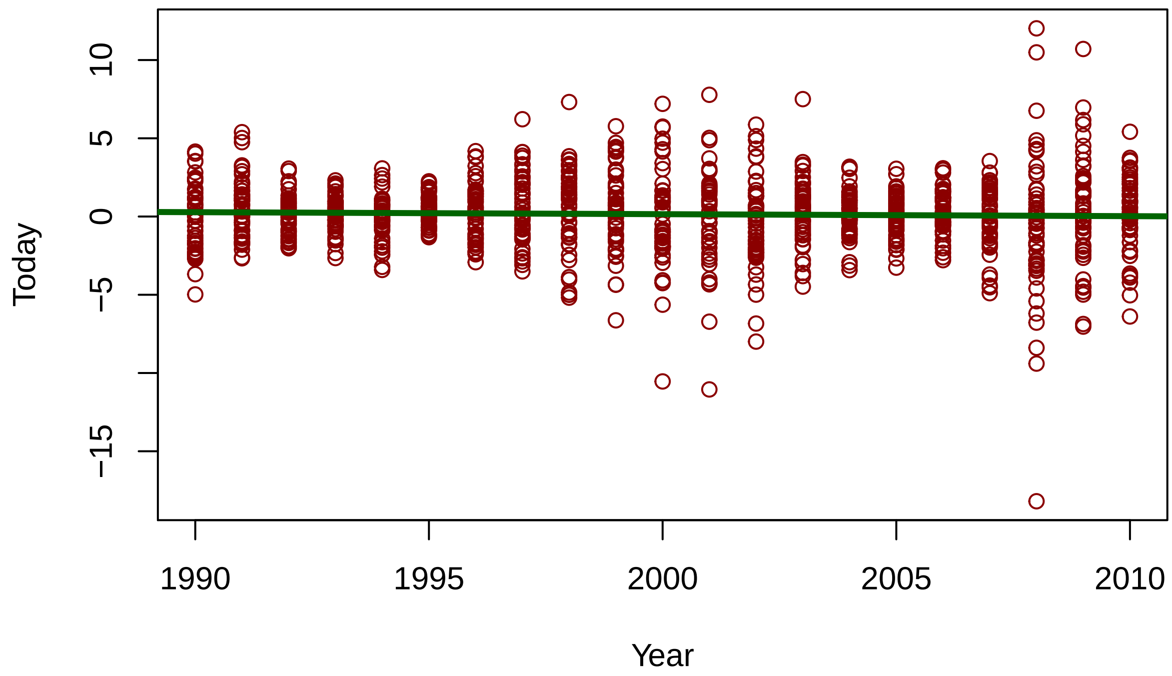
```r
### a
############

df = data.frame(Weekly)
summary(Weekly)
```

```
##       Year           Lag1              Lag2              Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
##
```

```r
plot(Volume~Year, col="darkred", data=Weekly)
simplelm = lm(Volume~Year, data=Weekly)
abline(simplelm, lwd= 3, col= "darkgreen")
```




```
plot(Today~Year, col="darkred", data=Weekly)
simplelm = lm(Today~Year, data=Weekly)
abline(simplelm, lwd= 3, col= "darkgreen")
```




```
############
### b
############

log_reg = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = "binomial", data = Weekly)
summary(log_reg)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

```r
############
### c
############
p = predict(log_reg, type = "response")
prediction = rep("Down", 1089)
prediction[p > 0.5] = "Up"
table(prediction, Direction)
```

```
##           Direction
## prediction Down  Up
##       Down   54  48
##       Up    430 557
```

```r
############
### d
############


train = Weekly[Year<2009,]
test = Weekly[Year>2008,]
log_reg2 = glm(Direction ~ Lag2, data= train, family = "binomial")
summary(log_reg2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = "binomial", data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```r
p2 = predict(log_reg2, type = "response")
prediction2 = rep("Down", 1089)
prediction2[p2 > 0.5] = "Up"
table(prediction2, Direction)
```

```
##            Direction
## prediction2 Down  Up
##        Down   23  22
##        Up    461 583
```

```r
############
### g
############

x.train = cbind(train$Lag2)
y.train = cbind(train$Direction)
x.test = cbind(test$Lag2)
prediction_knn = knn(x.train, x.test, y.train, k = 1)
table(prediction_knn, test$Direction)
```

```
##
## prediction_knn Down Up
##              1   21 30
##              2   22 31
```

```r
############
### i
############

# logistic regression with lag1, lag2, and lag4
log_reg3 = glm(Direction ~ Lag1 + Lag2 + Lag4, family = "binomial", data = train)
# summary(log_reg3)
p3 = predict(log_reg3, type = "response")
prediction3 = rep("Down", 1089)
prediction3[p3 > 0.5] = "Up"
table(prediction3, Direction)
```

```
##            Direction
## prediction3 Down  Up
##        Down   58  53
##        Up    426 552
```

```r
# logistic regression with lag1 and lag2
log_reg4 = glm(Direction ~ Lag1 + Lag2, family = "binomial", data = train)
# summary(log_reg4)
p4 = predict(log_reg4, type = "response")
prediction4 = rep("Down", 1089)
prediction4[p4 > 0.5] = "Up"
table(prediction4, Direction)
```

```
##            Direction
## prediction4 Down  Up
##        Down   47  49
##        Up    437 556
```

```r
# different knn values

# k = 3
prediction_knn3 = knn(x.train, x.test, y.train, k = 3)
table(prediction_knn3, test$Direction)
```

```
##
## prediction_knn3 Down Up
##               1   16 19
##               2   27 42
```

```r
# k = 5
prediction_knn5 = knn(x.train, x.test, y.train, k = 5)
table(prediction_knn5, test$Direction)
```

```
##
## prediction_knn5 Down Up
##               1   16 21
##               2   27 40
```

```r
# k = 7
prediction_knn7 = knn(x.train, x.test, y.train, k = 7)
table(prediction_knn7, test$Direction)
```

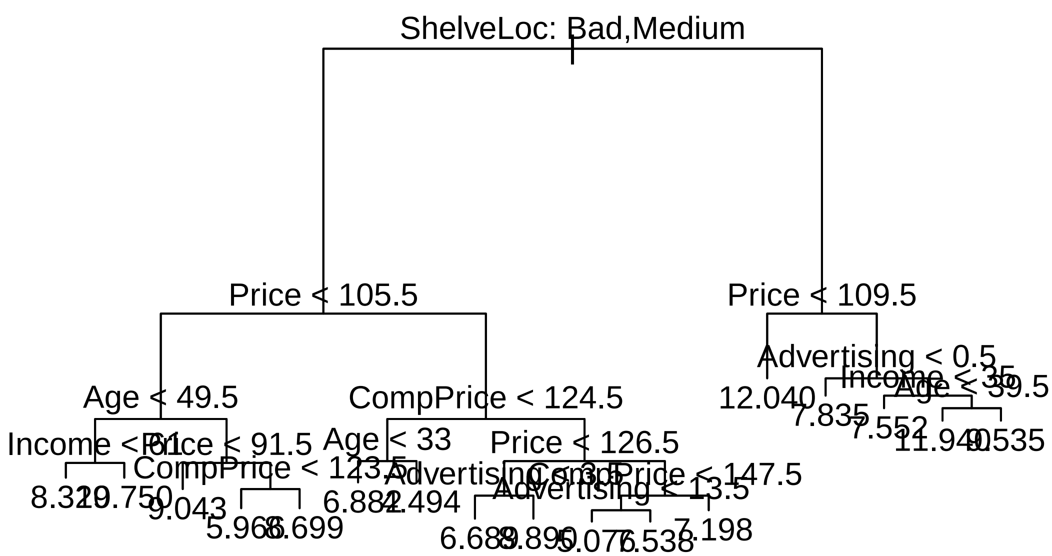
```
##
## prediction_knn7 Down Up
##               1   16 19
##               2   27 42
```

```r
#_____________________________________________________________________________________________________
## CHAPTER 8: PROBLEM 8
#_____________________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment
set.seed(1)
library(tree)
library(ISLR)
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
attach(Carseats)
library(kknn)

############
### a
############

n = dim(Carseats)[1]
```

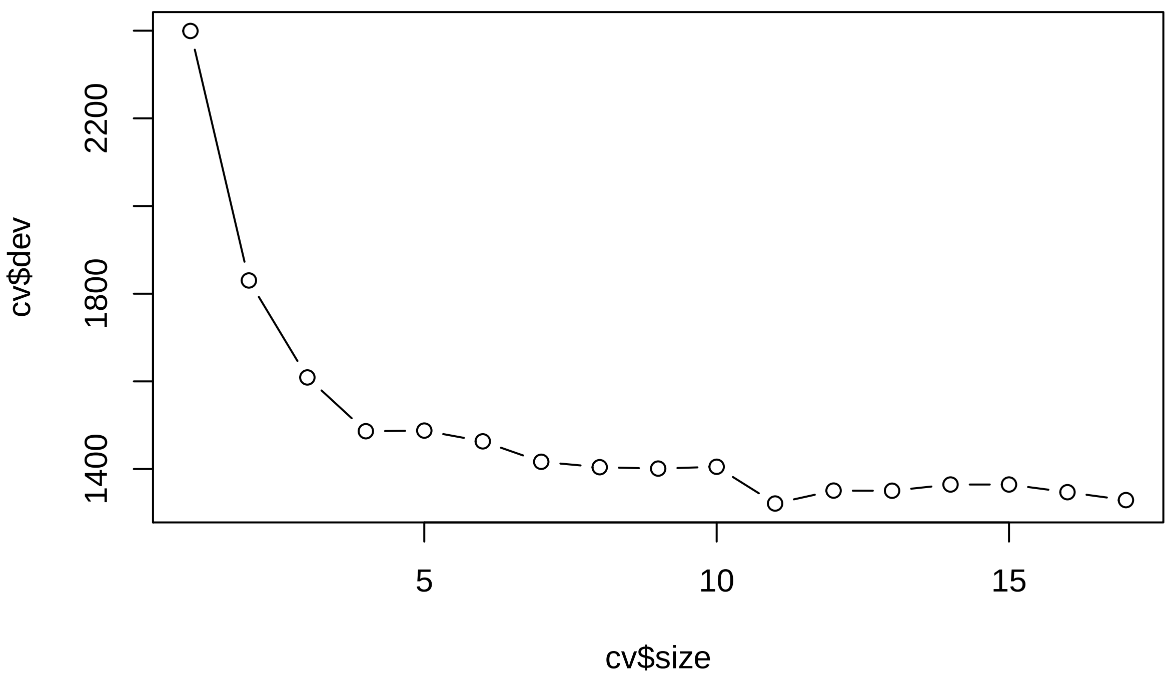
```r
#Sample (in this case with uniform distribution)
tr = sample(1:400, #The values that will be sampled
            size = 300, #The size of the sample
            replace = FALSE) #without replacement

# train and test set of the Carseats data
train = Carseats[tr,]
test = Carseats[-tr,]

############
### b
############

# creates the tree with sales as the y variable
tree = tree(Sales ~ ., data = train)

#plots the tree
plot(tree)
text(tree, pretty = 0)
```



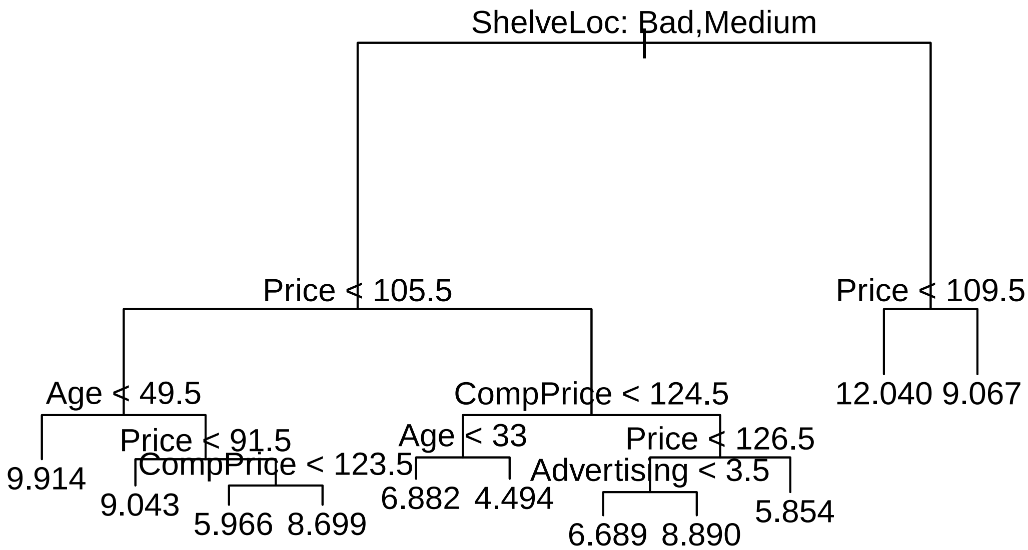

```r
# predicts the validation set
tree.pred = predict(tree, newdata = test)

#finds the MSE
mean((tree.pred - test$Sales)^2)
```

```
## [1] 4.910268
```

```
############
### c
############


cv = cv.tree(tree)
plot(cv$size, cv$dev, type = "b")
```




```
#tree.min = which.min(cv$dev)

pruned = prune.tree(tree, best = 11)
plot(pruned)
text(pruned, pretty = 0)
```



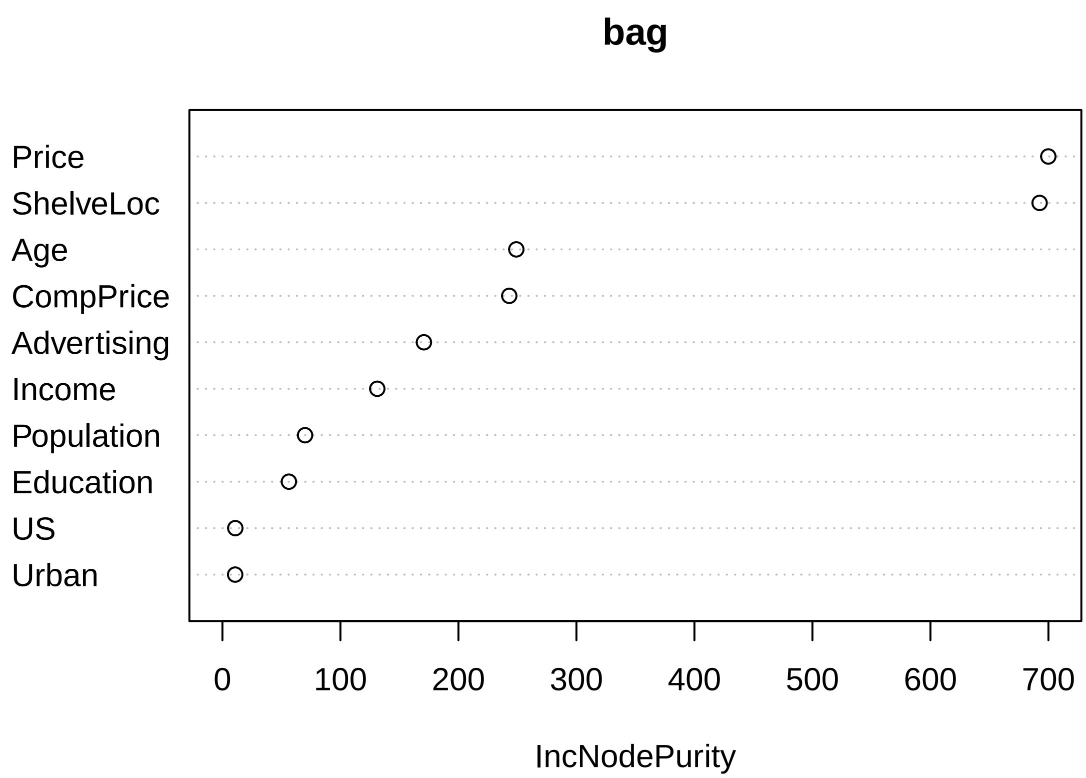

```r
# predicts the validation set
pruned.pred = predict(pruned, newdata = test)

#finds the MSE
mean((pruned.pred - test$Sales)^2)
```

```
## [1] 5.236696
```

```r
############
### d
############

# bagging with 10 variables tried at each split
bag = randomForest(Sales ~ ., data = train, mtry = 10, importance = TRUE)

prediction_bag = predict(bag, newdata = test)

#Find the MSE of the bagging prediction
mean((prediction_bag - test$Sales)^2)
```

```
## [1] 2.819484
```

```r
# Use importance function to find out important variables
importance = importance(bag)
# plot the most important variables
varImpPlot(bag, type = 2)
```

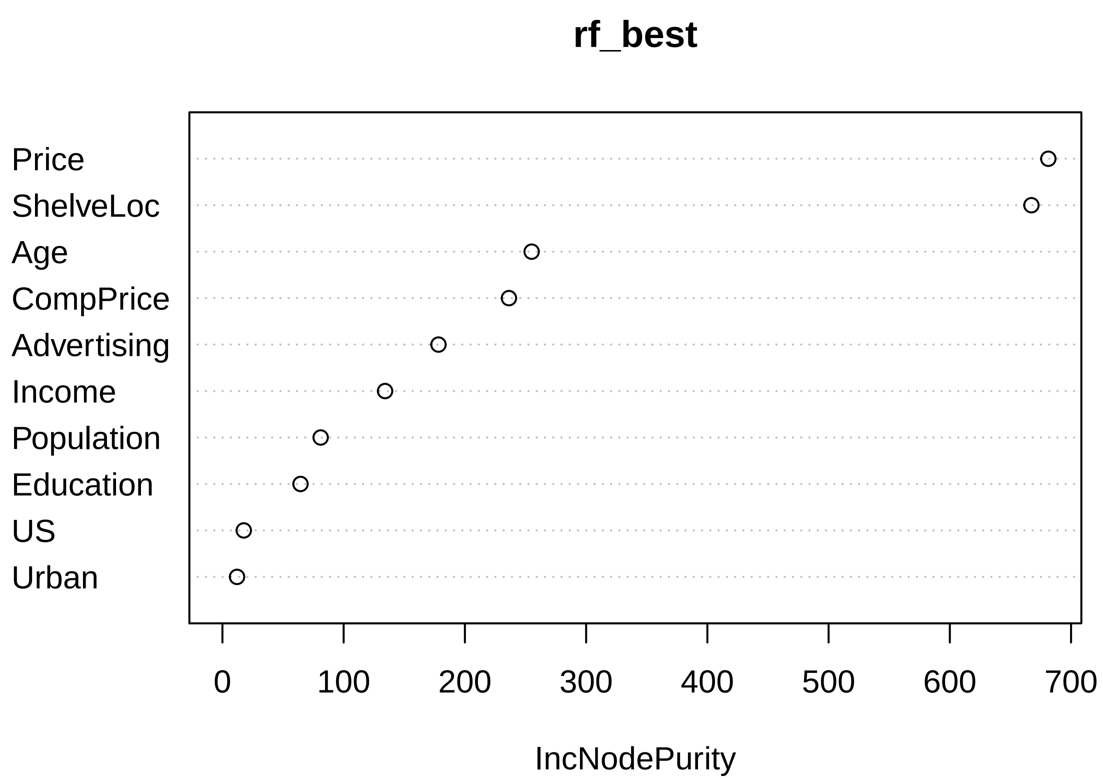



```
############
### e
############

# random forest with different numbers of mtry
m = c(1:10)
for (i in m){
  rf = randomForest(Sales ~ ., data = train, mtry = i, importance = TRUE)
  prediction_rf = predict(rf, newdata = test)
  print(i)
  MSE = (mean((prediction_rf - test$Sales)^2))
  print(MSE)
}
```

```
## [1] 1
## [1] 4.862325
## [1] 2
## [1] 3.584137
## [1] 3
## [1] 3.173027
## [1] 4
## [1] 2.922094
## [1] 5
## [1] 2.80836
## [1] 6
## [1] 2.795388
## [1] 7
```

```
## [1] 2.761671
## [1] 8
## [1] 2.769803
## [1] 9
## [1] 2.794291
## [1] 10
## [1] 2.806043
```

```
# randomForest with the lowest MSE
rf_best = randomForest(Sales ~ ., data = train, mtry = 7, importance = TRUE)
importance = importance(rf_best)
varImpPlot(rf_best, type = 2)
```

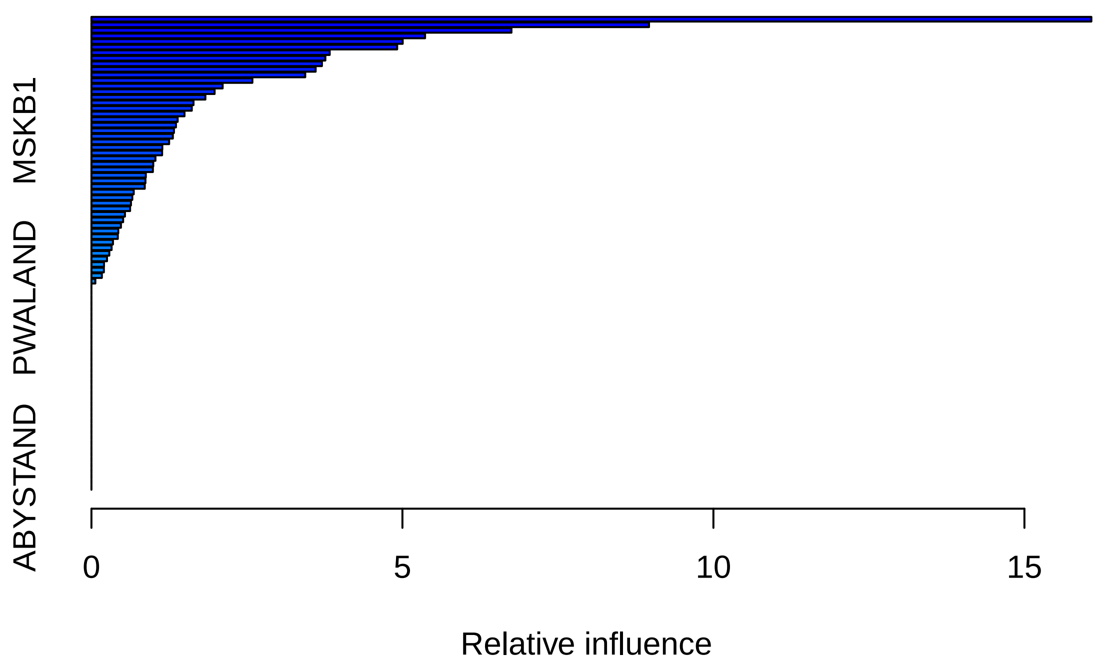



```
#______________________________________________________________________________________________
## CHAPTER 8: PROBLEM 11
#______________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment
set.seed(1)
library(tree)
library(ISLR)
library(randomForest)
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
attach(Caravan)
```

```r
############
### a
############

n = dim(Caravan)[1]
#Sample (in this case with uniform distribution)
tr = sample(1:5822, #The values that will be sampled
            size = 1000, #The size of the sample
            replace = FALSE) #without replacement

# train and test set of the Carseats data
train = Caravan[tr,]
test = Caravan[-tr,]

############
### b
############

boost <- gbm(Purchase ~ ., data = train, distribution = "gaussian", n.trees = 1000, shrinkage = 0.01)
summary(boost)
```




```
##               var     rel.inf
## PPERSAUT PPERSAUT 16.07625883
## MAUT2       MAUT2  8.96456460
## ALEVEN     ALEVEN  6.75239489
## MINKGEM   MINKGEM  5.36396870
```

```
## MBERMIDD MBERMIDD  5.00423947
## MBERHOOG MBERHOOG  4.91524769
## MHHUUR     MHHUUR  3.83078669
## PBRAND     PBRAND  3.76076515
## MGODGE     MGODGE  3.70550061
## MHKOOP     MHKOOP  3.60461639
## MSKC         MSKC  3.43716884
## MOPLHOOG MOPLHOOG  2.58754336
## MOSTYPE   MOSTYPE  2.10937153
## MAUT0       MAUT0  1.98037320
## MOPLLAAG MOPLLAAG  1.83177006
## MINK3045 MINK3045  1.64099813
## MKOOPKLA MKOOPKLA  1.61340922
## MINK4575 MINK4575  1.49618146
## MFALLEEN MFALLEEN  1.38606496
## MINK7512 MINK7512  1.35881683
## MSKB1       MSKB1  1.32650463
## MGODRK     MGODRK  1.30907202
## MINKM30   MINKM30  1.24894123
## MRELOV     MRELOV  1.13895658
## APERSAUT APERSAUT  1.13601480
## MGODOV     MGODOV  1.02884025
## MSKB2       MSKB2  0.99363868
## MFWEKIND MFWEKIND  0.98780019
## MFGEKIND MFGEKIND  0.87399354
## MGODPR     MGODPR  0.86813100
## MBERARBG MBERARBG  0.85766489
## MSKD         MSKD  0.68015715
## MZFONDS   MZFONDS  0.65679024
## MOPLMIDD MOPLMIDD  0.63539585
## MSKA         MSKA  0.62142253
## MGEMLEEF MGEMLEEF  0.53883819
## MBERARBO MBERARBO  0.50918267
## MBERBOER MBERBOER  0.47432034
## MINK123M MINK123M  0.43117655
## MOSHOOFD MOSHOOFD  0.42385474
## PLEVEN     PLEVEN  0.34599034
## PWAPART   PWAPART  0.32417626
## MAUT1       MAUT1  0.28718933
## MRELSA     MRELSA  0.25038462
## MRELGE     MRELGE  0.20132669
## MZPART     MZPART  0.19999609
## MBERZELF MBERZELF  0.16759445
## MAANTHUI MAANTHUI  0.06260555
## MGEMOMV   MGEMOMV  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  0.00000000
## PMOTSCO   PMOTSCO  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PAANHANG PAANHANG  0.00000000
## PTRACTOR PTRACTOR  0.00000000
## PWERKT     PWERKT  0.00000000
## PBROM       PBROM  0.00000000
```

```
## PPERSONG PPERSONG  0.00000000
## PGEZONG   PGEZONG  0.00000000
## PWAOREG   PWAOREG  0.00000000
## PZEILPL   PZEILPL  0.00000000
## PPLEZIER PPLEZIER  0.00000000
## PFIETS     PFIETS  0.00000000
## PINBOED   PINBOED  0.00000000
## PBYSTAND PBYSTAND  0.00000000
## AWAPART   AWAPART  0.00000000
## AWABEDR   AWABEDR  0.00000000
## AWALAND   AWALAND  0.00000000
## ABESAUT   ABESAUT  0.00000000
## AMOTSCO   AMOTSCO  0.00000000
## AVRAAUT   AVRAAUT  0.00000000
## AAANHANG AAANHANG  0.00000000
## ATRACTOR ATRACTOR  0.00000000
## AWERKT     AWERKT  0.00000000
## ABROM       ABROM  0.00000000
## APERSONG APERSONG  0.00000000
## AGEZONG   AGEZONG  0.00000000
## AWAOREG   AWAOREG  0.00000000
## ABRAND     ABRAND  0.00000000
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

```
############
### c
############

# using boost model
probs <- predict(boost, test, n.trees = 1000, type = "response")

pred_boost <- ifelse(probs > 0.2, 1, 0)
table(test$Purchase, pred_boost)
```

```
##      pred_boost
##          1
##   No  4526
##   Yes  296
```

```
#logistic reg
log_caravan = glm(Purchase ~ ., data = train, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
probs2 <- predict(log_caravan, test, type = "response")
```

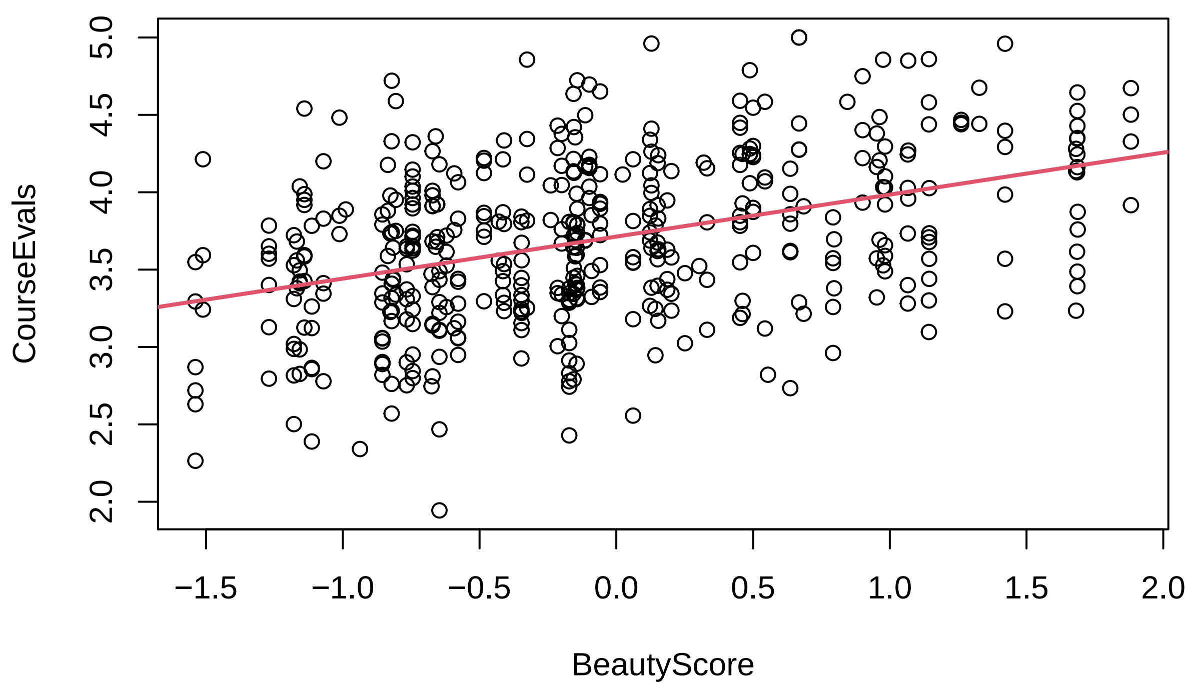
```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
pred2 =  ifelse(probs2 > .2, 1, 0)
table(test$Purchase, pred2)
```

```
##      pred2
```

```
##          0    1
##   No  4207  319
##   Yes  244   52
```

```
## NON-BOOK PROBLEMS ------------------------------------------------------------------------------
#_________________________________________________________________________________________________
## PROBLEM 1: BEAUTY PAYS
#_________________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment
#Read data
beautyData = read.csv("BeautyData.csv",header=T)
attach(beautyData)

# linear regression
lm.1 = lm(CourseEvals ~ BeautyScore, data = beautyData)
plot(BeautyScore, CourseEvals)
abline(lsfit(BeautyScore,CourseEvals), #lsfit can be used instead of lm()
       lwd=2, #Line width
       col=2) #Line color
```




```
summary(lm.1)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore, data = beautyData)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5936 -0.3346  0.0097  0.3702  1.2321
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.71340    0.02249 165.119   <2e-16 ***
## BeautyScore  0.27148    0.02837   9.569   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4809 on 461 degrees of freedom
## Multiple R-squared:  0.1657, Adjusted R-squared:  0.1639
## F-statistic: 91.57 on 1 and 461 DF,  p-value: < 2.2e-16
```

```
lm.all = lm(CourseEvals ~ ., data = beautyData)
summary(lm.all)
```

```
##
## Call:
## lm(formula = CourseEvals ~ ., data = beautyData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31385 -0.30202  0.01011  0.29815  1.04929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.06542    0.05145  79.020  < 2e-16 ***
## BeautyScore  0.30415    0.02543  11.959  < 2e-16 ***
## female      -0.33199    0.04075  -8.146 3.62e-15 ***
## lower       -0.34255    0.04282  -7.999 1.04e-14 ***
## nonenglish  -0.25808    0.08478  -3.044  0.00247 **
## tenuretrack -0.09945    0.04888  -2.035  0.04245 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4273 on 457 degrees of freedom
## Multiple R-squared:  0.3471, Adjusted R-squared:  0.3399
## F-statistic: 48.58 on 5 and 457 DF,  p-value: < 2.2e-16
```

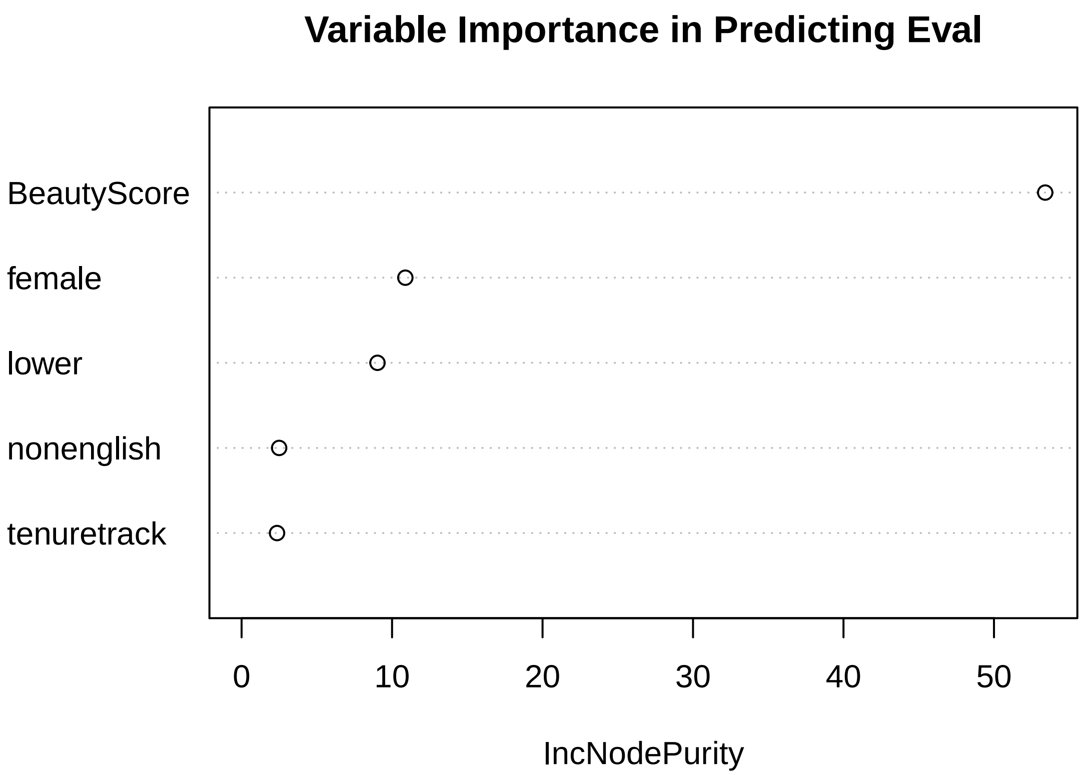
```
#variable importance using bagging
bag_beauty = randomForest(CourseEvals ~ ., data = beautyData, mtry = 10, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
importance(bag_beauty)
```

```
##               %IncMSE IncNodePurity
## BeautyScore 64.115483     53.404892
## female      45.202210     10.882564
## lower       45.889089      9.030989
## nonenglish  16.873553      2.497086
## tenuretrack  8.444722      2.356280
```

```r
varImpPlot(bag_beauty, type = 2, main = "Variable Importance in Predicting Eval")
```




```r
#__________________________________________________________________________________________________
## PROBLEM 2: Housing Price structure
#__________________________________________________________________________________________________

rm(list=ls()) #Removes every object from your environment
#Read data
set.seed(1)
mc.data = read.csv("MidCity.csv",header=T)
attach(mc.data)
```

```
## The following object is masked from Carseats:
##
##     Price
```

```r
############
### 1
############
brick = ifelse(mc.data$Brick == "Yes", 1, 0)
mc.data = mc.data[-5]
mc.data = cbind(mc.data, brick)

lm.mc = lm(Price ~ ., data = mc.data)
summary(lm.mc)
```

```
##
```

```
## Call:
## lm(formula = Price ~ ., data = mc.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24940.6  -8383.0    430.7   7430.4  31371.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9814.663   9858.884  -0.996  0.32149
## Home            6.187     28.973   0.214  0.83128
## Nbhd         9832.281   1821.869   5.397 3.47e-07 ***
## Offers      -8351.794   1267.428  -6.590 1.24e-09 ***
## SqFt           49.811      6.769   7.359 2.53e-11 ***
## Bedrooms     5671.911   1840.979   3.081  0.00256 **
## Bathrooms    8243.545   2449.897   3.365  0.00103 **
## brick       15601.818   2261.896   6.898 2.66e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11540 on 120 degrees of freedom
## Multiple R-squared:  0.8256, Adjusted R-squared:  0.8154
## F-statistic: 81.15 on 7 and 120 DF,  p-value: < 2.2e-16
```

```r
############
### 2
############

# makes new column with dummy variable for houses in neighborhood 3
nbhd_3 = ifelse(mc.data$Nbhd == 3, 1, 0)
mc.data = mc.data[-2]
mc.data = cbind(mc.data, nbhd_3)

lm.mc3 = lm(Price ~ ., data = mc.data)
summary(lm.mc3)
```

```
##
## Call:
## lm(formula = Price ~ ., data = mc.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27191.9  -6372.7   -154.2   5739.9  26880.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3049.72    8778.83   0.347  0.72890
## Home           -8.68      25.03  -0.347  0.72940
## Offers      -8061.22    1023.98  -7.872 1.73e-12 ***
## SqFt           52.56       5.72   9.190 1.46e-15 ***
## Bedrooms     3971.91    1601.85   2.480  0.01454 *
## Bathrooms    7874.35    2124.72   3.706  0.00032 ***
## brick       17051.46    1950.02   8.744 1.64e-14 ***
## nbhd_3      21929.82    2491.57   8.802 1.20e-14 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10030 on 120 degrees of freedom
## Multiple R-squared:  0.8683, Adjusted R-squared:  0.8606
## F-statistic:   113 on 7 and 120 DF,  p-value: < 2.2e-16
```

```
############
### 3
############

# makes a new data frame with only houses in neighborhood 3
mc.data_3 = mc.data[ which(mc.data$nbhd_3 == 1),]
lm.brick3 = lm(Price ~ ., data = mc.data_3)
summary(lm.brick3)
```

```
##
## Call:
## lm(formula = Price ~ ., data = mc.data_3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13495.7  -3201.1   -448.8   2110.9  22960.0
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15446.321  15272.354   1.011   0.3194
## Home          -63.735     48.568  -1.312   0.1988
## Offers      -8552.987   1863.469  -4.590 6.52e-05 ***
## SqFt           56.647      9.383   6.037 9.75e-07 ***
## Bedrooms     5280.851   2526.808   2.090   0.0447 *
## Bathrooms    7158.602   3348.565   2.138   0.0403 *
## brick       24262.184   3205.391   7.569 1.27e-08 ***
## nbhd_3             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8388 on 32 degrees of freedom
## Multiple R-squared:  0.8486, Adjusted R-squared:  0.8202
## F-statistic: 29.89 on 6 and 32 DF,  p-value: 8.596e-12
```

```
############
### 4
############

##### refer to 2

#_______________________________________________________________________________________________________
## PROBLEM 4: Neural Nets
#_______________________________________________________________________________________________________
rm(list=ls()) #Removes every object from your environment

set.seed(1)
library(ISLR)
library(caret)
library(MASS)
```

```r
#Read in the data
attach(Boston)
```

```
## The following objects are masked from Boston (pos = 13):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn

## The following objects are masked from Boston (pos = 22):
##
##     age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad,
##     rm, tax, zn
```

```r
df = data.frame(Boston)
#Summary of the data
summary(Boston)
```

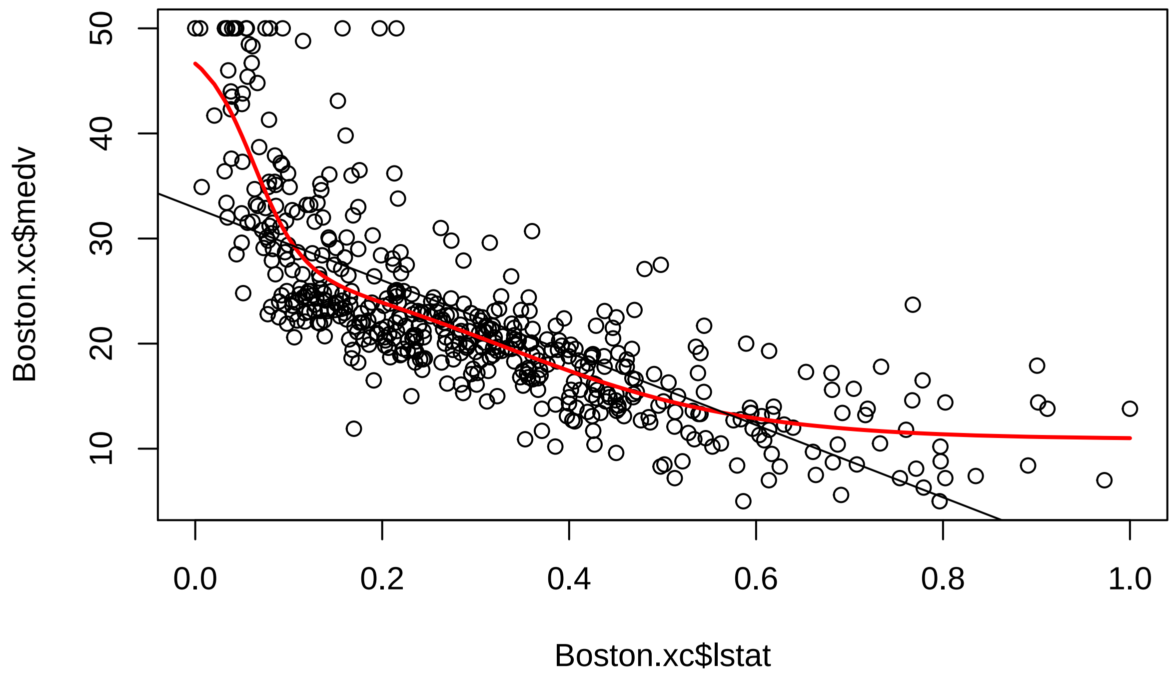
```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  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          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```r
#Standardize the x's (the first 3 columns)
minv = rep(0,13) #Create vector which will hold the minimum
maxv = rep(0,13) #Create vector which will hold the maximum
Boston.xc = Boston #Create auxiliary copy of the matrix
for(i in 1:13) {
  minv[i] = min(Boston[[i]]) #Save the minimum
  maxv[i] = max(Boston[[i]]) #Save the maximum
  Boston.xc[[i]] = (Boston[[i]]-minv[i])/(maxv[i]-minv[i]) #Standardize the values
}
```

```r
# nn library
library(nnet)

#Fit nn with just one x=lstat
set.seed(1) #Seed to guarantee the same results

#Create the model
znn = nnet(medv ~ lstat, #Formula
           data = Boston.xc, #Data frame with the traning set
           size=3, #Units in the hidden layer
           decay=0.1, #Parameter for weight decay
           linout=T) #Linear output
```

```
## # weights:  10
## initial  value 287195.498148
## iter  10 value 42866.685420
## iter  20 value 18869.758682
## iter  30 value 14335.290324
## iter  40 value 13718.867845
## iter  50 value 13546.473874
## iter  60 value 13544.478878
## iter  70 value 13542.662065
## iter  80 value 13542.312221
## iter  90 value 13542.198012
## iter  90 value 13542.197913
## iter  90 value 13542.197911
## final  value 13542.197911
## converged
```

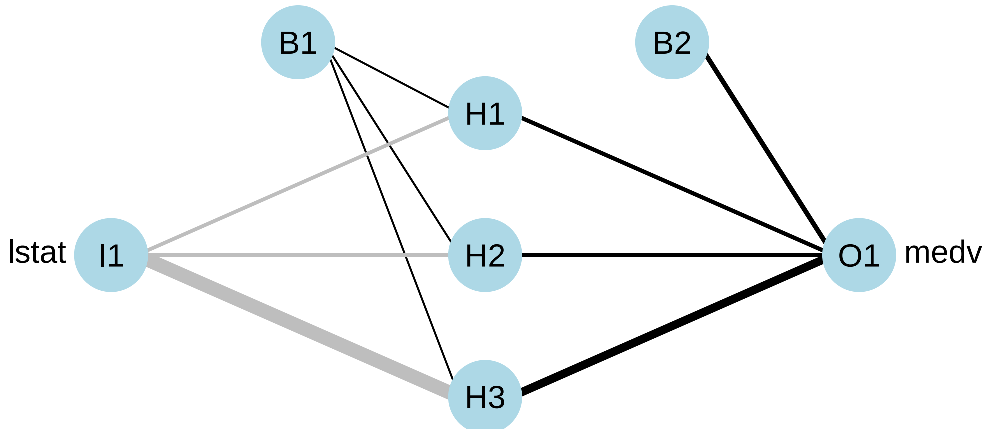
```r
#Get fits, print summary, and plot fit
fznn = predict(znn,Boston.xc) #Gets the models fits for the data
plot(Boston.xc$lstat,Boston.xc$medv) #Dispersion plot of lstat and medv
oo = order(Boston.xc$lstat) #Get the indices that will order the column lstat
lines(Boston.xc$lstat[oo],fznn[oo],col="red",lwd=2) #Line of the fits
abline(lm(medv~lstat,Boston.xc)$coef) #Compare with the OLS fit
```




```r
#What does this mean? Try to interpret looking at the Neural network
summary(znn)
```

```
## a 1-3-1 network with 10 weights
## options were - linear output units  decay=0.1
##  b->h1 i1->h1
##   2.52  -7.65
##  b->h2 i1->h2
##   2.52  -7.63
##  b->h3 i1->h3
##   2.38 -38.23
##   b->o  h1->o  h2->o  h3->o
##  10.90   8.90   8.80  21.15
```

```r
NeuralNetTools::plotnet(znn)
```




```
#Now let us try a model with 5 units in the hidden layer
set.seed(99)
znn = nnet(medv ~ lstat, #Formula
           data = Boston.xc, #Data frame with the traning set
           size=5, #Units in the hidden layer
           decay=0.1, #Parameter for weight decay
           linout=T) #Linear output
```

```
## # weights:  16
## initial  value 296339.511159
## iter  10 value 19548.919269
## iter  20 value 14981.907690
## iter  30 value 13558.378791
## iter  40 value 13548.587652
## iter  50 value 13544.846614
## iter  60 value 13542.122924
## iter  70 value 13540.590160
## iter  80 value 13536.856495
## iter  90 value 13535.843727
## iter 100 value 13535.454826
## final  value 13535.454826
## stopped after 100 iterations
```

```
print(summary(znn))
```

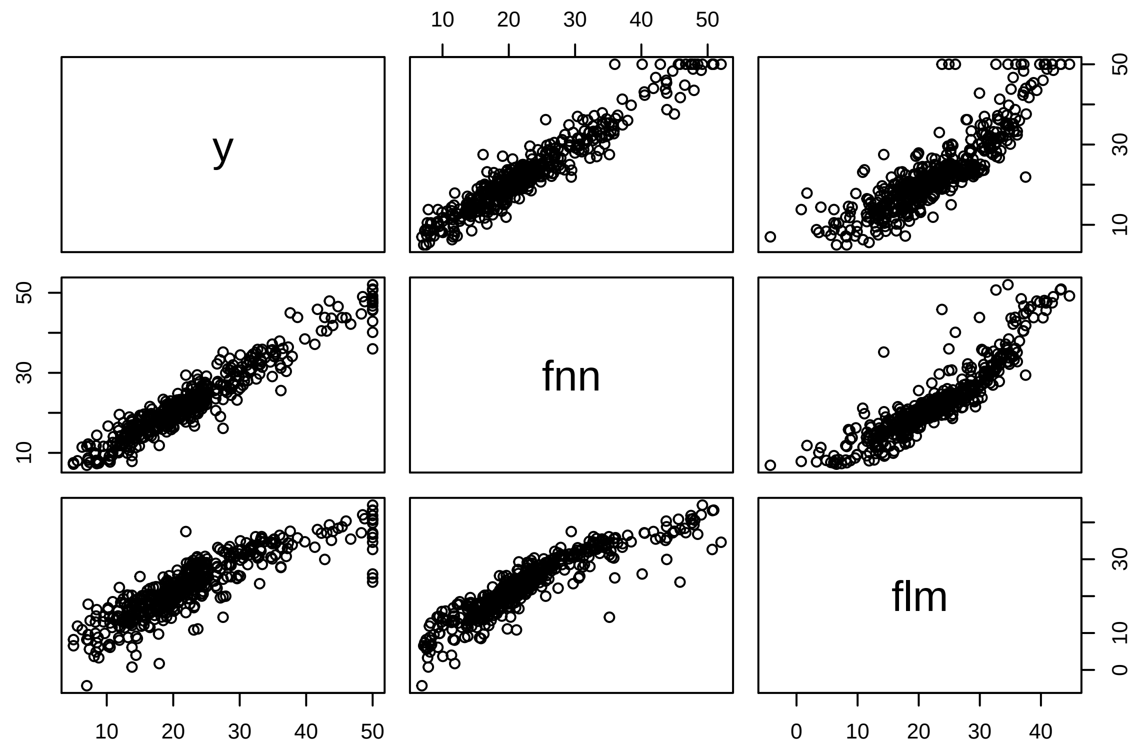
```
## a 1-5-1 network with 16 weights
## options were - linear output units  decay=0.1
```

```
##  b->h1 i1->h1
##   1.49   0.59
##  b->h2 i1->h2
##   2.48  -7.61
##  b->h3 i1->h3
##   1.59   0.58
##  b->h4 i1->h4
##   2.54  -7.61
##  b->h5 i1->h5
##   2.38 -38.17
##   b->o  h1->o  h2->o  h3->o  h4->o  h5->o
##   4.27   3.79   9.05   3.76   9.05  21.21
```

```r
#Now, let us estimate a model with all covariates
znn = nnet(medv ~ ., #Formula
           data = Boston.xc, #Data frame with the traning set
           size=5, #Units in the hidden layer
           decay=0.1, #Parameter for weight decay
           linout=T) #Linear output
```

```
## # weights:  76
## initial  value 325072.622693
## iter  10 value 13407.516705
## iter  20 value 7344.877324
## iter  30 value 5965.779266
## iter  40 value 5499.374428
## iter  50 value 5225.689306
## iter  60 value 4967.736976
## iter  70 value 4849.858048
## iter  80 value 4555.015904
## iter  90 value 4130.707856
## iter 100 value 3976.829624
## final  value 3976.829624
## stopped after 100 iterations
```

```r
fznn = predict(znn,Boston.xc) #Gets the models fits for the data
zlm = lm(medv~.,Boston.xc) #Estimating medv using OLS
fzlm = predict(zlm,Boston.xc) #Gets the OLS fits for the data
temp = data.frame(y=Boston.xc$medv,fnn=fznn,flm=fzlm) #Data frame of results
pairs(temp) #Matrix of scatterplots
```




```r
print(cor(temp)) #Correlation matrix
```

```
##             y       fnn       flm
## y   1.0000000 0.9580994 0.8606060
## fnn 0.9580994 1.0000000 0.9041531
## flm 0.8606060 0.9041531 1.0000000
```

```r
#Let us modify the number of nodes in the hidden layer and decay values

#Four different fits
set.seed(1)
znn1 = nnet(medv~lstat,Boston.xc,size=3,decay=.5,linout=T)
```

```
## # weights:  10
## initial  value 287196.222000
## iter  10 value 36138.805744
## iter  20 value 17210.283364
## iter  30 value 14355.037349
## iter  40 value 14304.719643
## iter  50 value 14301.723285
## final  value 14301.600601
## converged
```

```r
znn2 = nnet(medv~lstat,Boston.xc,size=3,decay=.00001,linout=T)
```

```
## # weights:  10
## initial  value 283943.029641
## iter  10 value 32767.126611
```

```
## iter  20 value 13683.923628
## iter  30 value 13465.797936
## iter  40 value 13366.208738
## iter  50 value 13211.278091
## iter  60 value 13201.534216
## iter  70 value 13199.318717
## iter  80 value 13199.260950
## final  value 13199.260019
## converged
```

```
znn3 = nnet(medv~lstat,Boston.xc,size=50,decay=.5,linout=T)
```

```
## # weights:  151
## initial  value 317291.580293
## iter  10 value 16833.903019
## iter  20 value 14854.528559
## iter  30 value 14629.216414
## iter  40 value 14462.533054
## iter  50 value 14362.657153
## iter  60 value 14316.558902
## iter  70 value 14290.923232
## iter  80 value 14278.375563
## iter  90 value 14268.608134
## iter 100 value 14263.539631
## final  value 14263.539631
## stopped after 100 iterations
```

```
znn4 = nnet(medv~lstat,Boston.xc,size=50,decay=.00001,linout=T)
```

```
## # weights:  151
## initial  value 404716.949723
## iter  10 value 14948.852525
## iter  20 value 13573.387977
## iter  30 value 13394.855224
## iter  40 value 13263.870282
## iter  50 value 13197.797529
## iter  60 value 13181.502464
## iter  70 value 13166.596868
## iter  80 value 13158.974473
## iter  90 value 13152.144083
## iter 100 value 13148.334356
## final  value 13148.334356
## stopped after 100 iterations
```

```
temp = data.frame(medv = Boston.xc$medv, lstat = Boston.xc$lstat) #The data

#The predictions of each model for the data
znnf1 = predict(znn1,temp)
znnf2 = predict(znn2,temp)
znnf3 = predict(znn3,temp)
znnf4 = predict(znn4,temp)

#Plotting the fits
par(mfrow=c(2,2)) #Plot window: 2 row, 2 columns

plot(Boston.xc$lstat,Boston.xc$medv, xlab = "lstat", ylab = "medv") #Scatterplot
```

```r
lines(Boston.xc$lstat[oo],znnf1[oo],lwd=2) #Adding the lines of predicted values
title("size=3, decay=.5")

plot(Boston.xc$lstat,Boston.xc$medv, xlab = "lstat", ylab = "medv")
lines(Boston.xc$lstat[oo],znnf2[oo],lwd=2)
title("size=3, decay=.00001")

plot(Boston.xc$lstat,Boston.xc$medv, xlab = "lstat", ylab = "medv")
lines(Boston.xc$lstat[oo],znnf3[oo],lwd=2)
title("size = 50, decay = .5")

plot(Boston.xc$lstat,Boston.xc$medv, xlab = "lstat", ylab = "medv")
lines(Boston.xc$lstat[oo],znnf4[oo],lwd=2)
title("size = 50, decay = .00001")
```