Assignment_ML

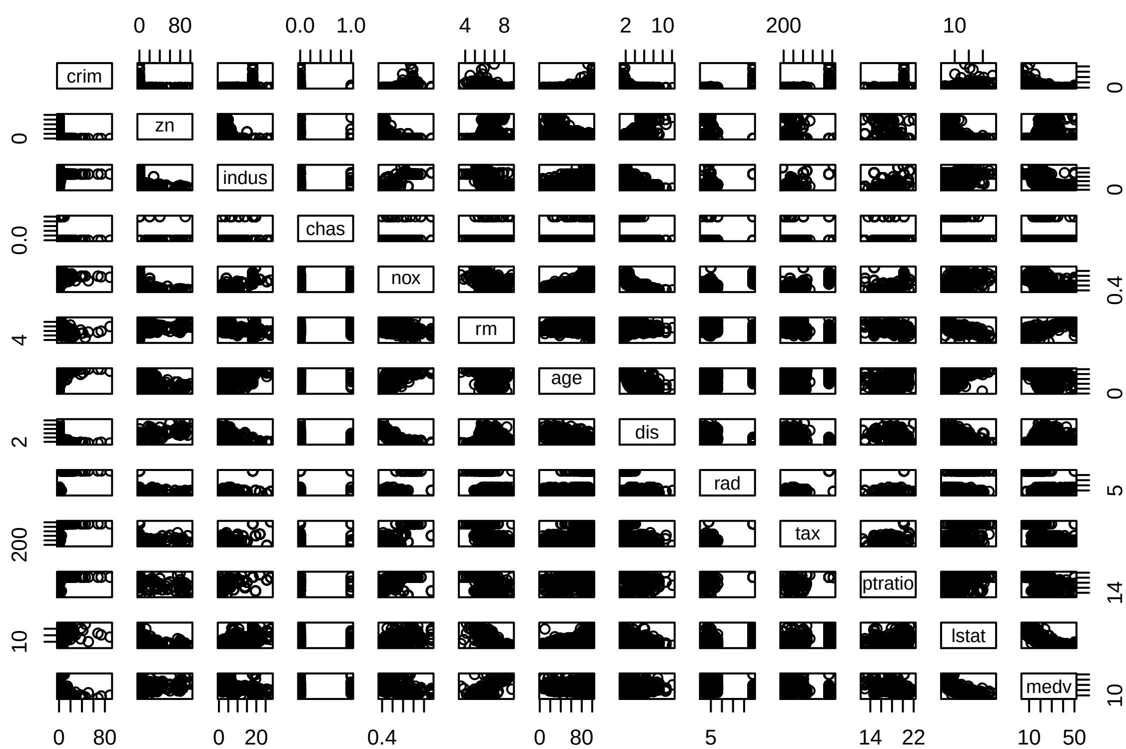
2023-07-26

Question 1 : Chapter 2: #10

(a)

- **No. of Rows** : 506
- **No. of Columns**: 13
- Rows represent the **suburbs** of Boston, and Columns represent the **variables**(crim, zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, lstat, medv).

```r
pairs(Boston)
```



(b) **Findings:**

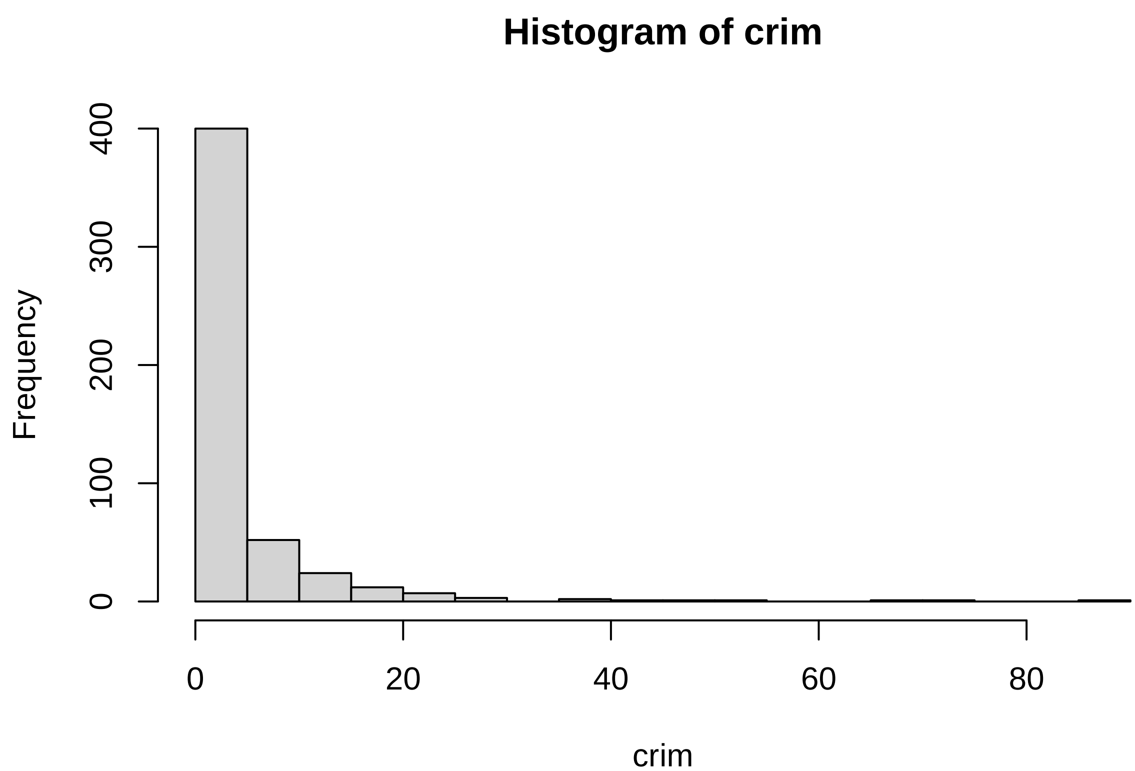
- Per capita crime rate by town(crim) has negative linear relationship with weighted mean of distances to five Boston employment centres(dis), and positive linear relationship with the proportion of owner-occupied units built prior to 1940(age). It has been the same case with (nox)nitrogen oxides concentration (parts per 10 million) as well.

- Lower status of the population (percent)(lstat) has a negative linear relationship with median value of owner-occupied homes in $1000s (medv)

(c) **Per capita crime rate by town(crim)** has negative linear relationship with weighted mean of distances to five Boston employment centres(dis), and positive linear relationship with the proportion of owner-occupied units built prior to 1940(age). It also has positive linear relationship with (rad)index of accessibility to radial highways, (tax)full-value property-tax rate per $10,000, and (ptratio)pupil-teacher ratio by town.

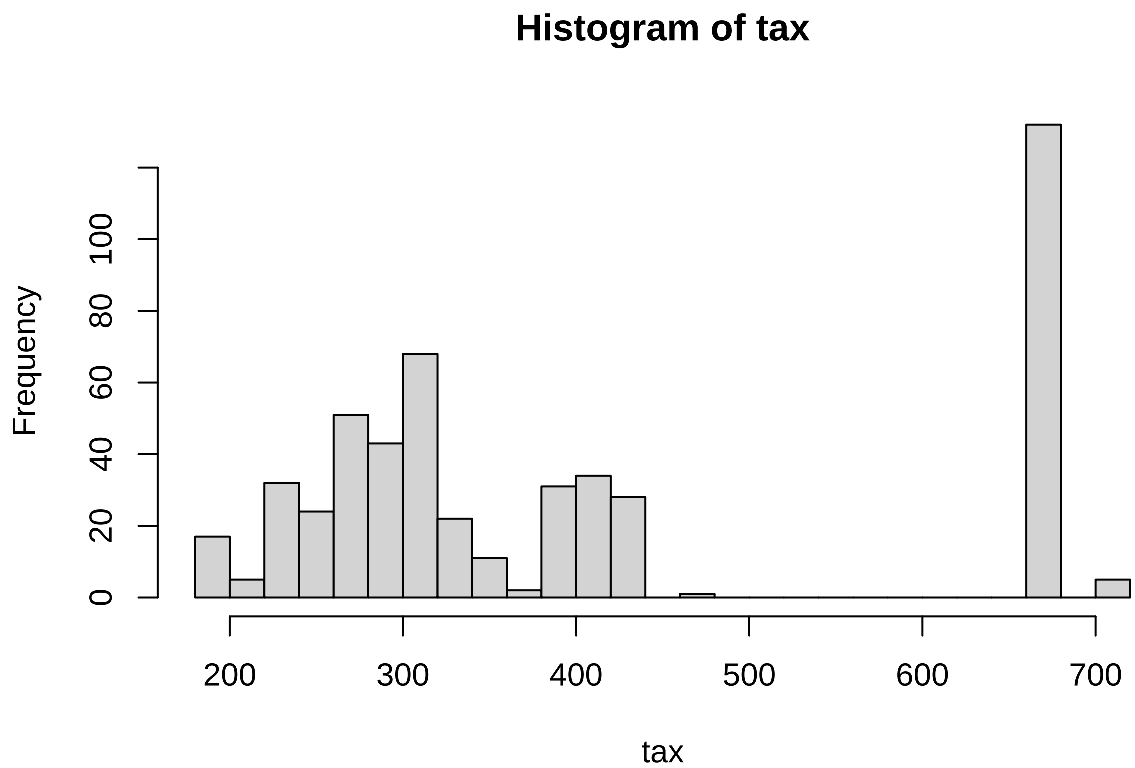
(d) **High crime rates, tax rates and Pupil-teacher ratios**

```
hist(crim, breaks = 20)
```



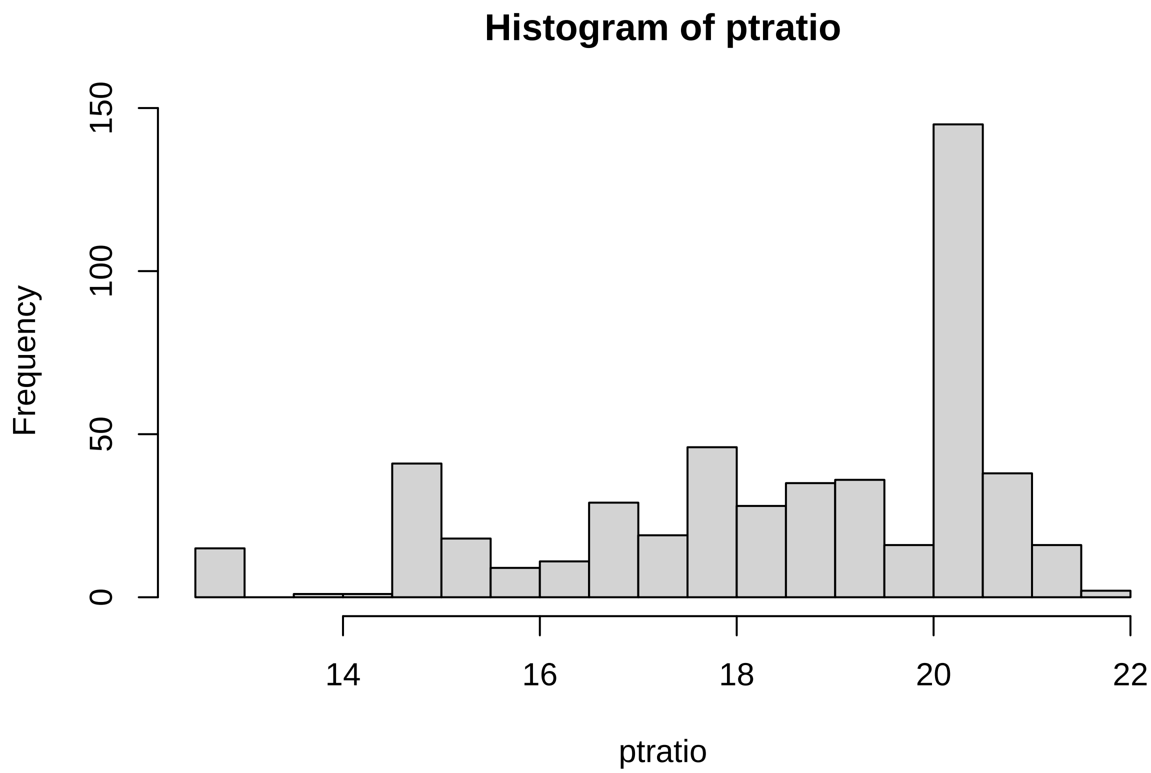
A very few census tracts of Boston appear to have particularly high crime rates

```
hist(tax, breaks = 20)
```



Above 120 census tracts of Boston appear to have particularly high tax rates

```
hist(ptratio, breaks = 20)
```



Around 150 census tracts of Boston appear to have particularly high Pupil-teacher ratios.

(e) **Census tracts in this data set bound the Charles river** = 35

(f) **Median pupil-teacher ratio among the towns in this data set** = 19.05

(g) **Census tract of Boston that has lowest median value of owner-occupied homes:** = 399, 406

```r
subset(Boston, medv == min(medv))
```

```
##        crim zn indus chas   nox    rm age    dis rad tax ptratio lstat medv
## 399 38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 30.59    5
## 406 67.9208  0  18.1    0 0.693 5.683 100 1.4254  24 666    20.2 22.98    5
```

Overall ranges for other predictors:

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

Comparison Findings

For the particular census tracts that has lowest median value of owner-occupied homes, the crime rate is very high. 'indus', 'nox', 'rm', 'ptratio' are almost same.

(h) ***Census tracts average more than seven rooms per dwelling*** = 64

Census tracts average more than eight rooms per dwelling = 13

Findings on the census tracts that average more than eight rooms per dwelling:

- The crime rate is relatively less in the census tracts that average more than eight rooms per dwelling.
- Pupil-teacher ratio by town and nitrogen oxides concentration (parts per 10 million) are almost the same when compared to the overall range.

Question 2 : Chapter 3: #15

(a) ***For each predictor, fit a simple linear regression model to predict the response***

```
library(MASS)
```

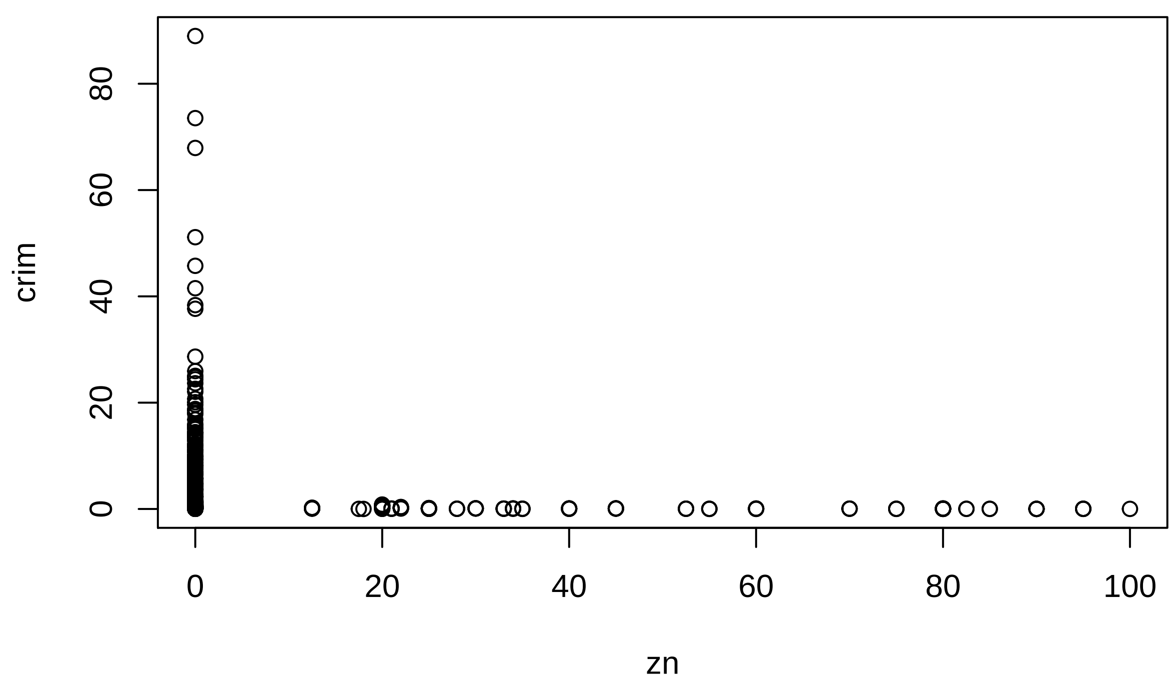
```
##
## Attaching package: 'MASS'

## The following object is masked _by_ '.GlobalEnv':
##
##     Boston

## The following object is masked from 'package:dplyr':
##
##     select

## The following object is masked from 'package:ISLR2':
##
##     Boston
```

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

```
##
## 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
```

```
plot(zn, crim)
```

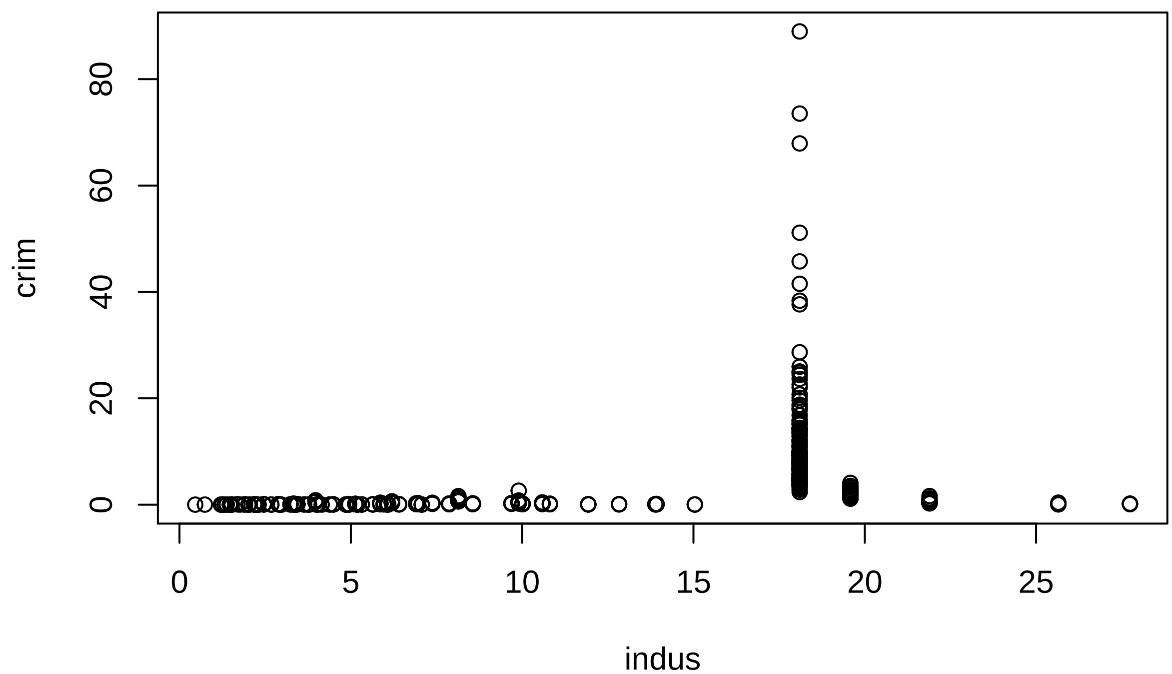



```
lm.indus <- lm(crim ~ indus , data = Boston)
summary(lm.indus)
```

```
##
## 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
plot(indus, crim)
```

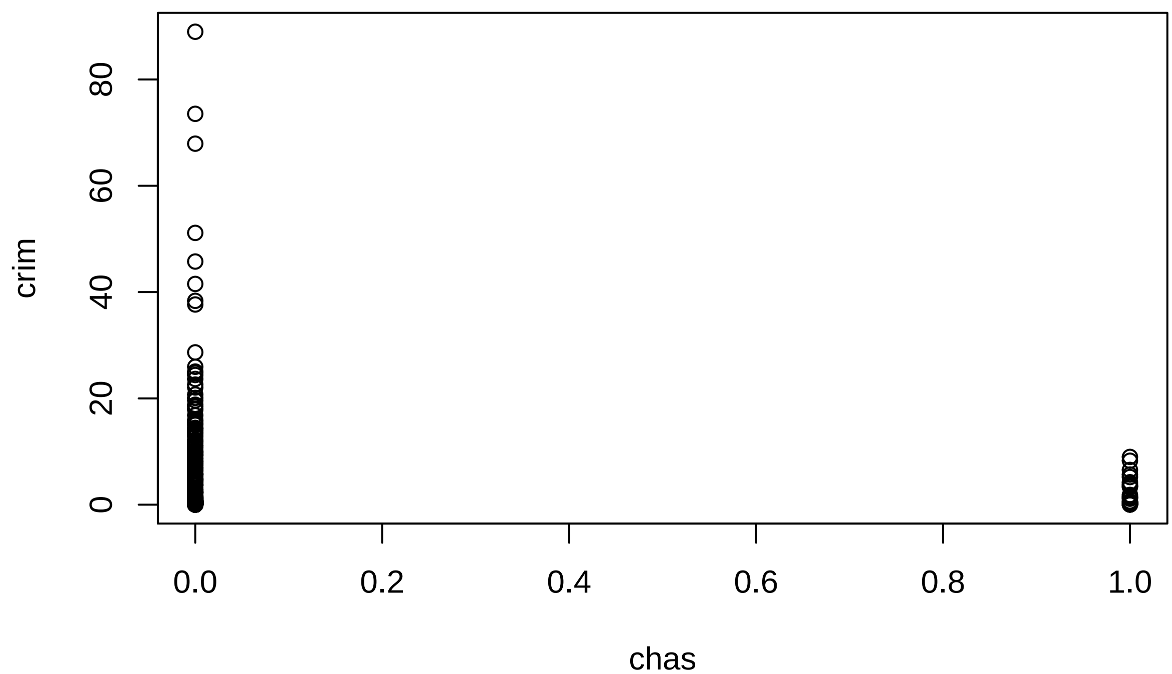



```r
lm.chas <- lm(crim ~ chas , data = Boston)
summary(lm.chas)
```

```
##
## 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
plot(chas, crim)
```

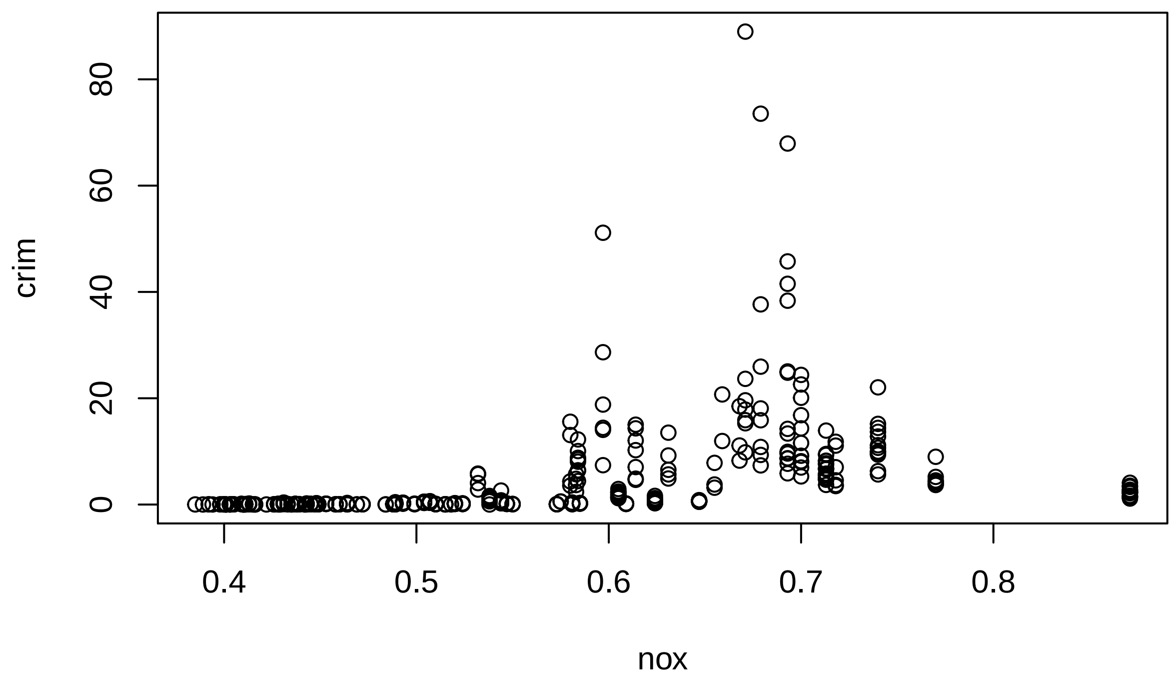



```r
lm.nox <- lm(crim ~ nox , data = Boston)
summary(lm.nox)
```

```
##
## 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
plot(nox, crim)
```

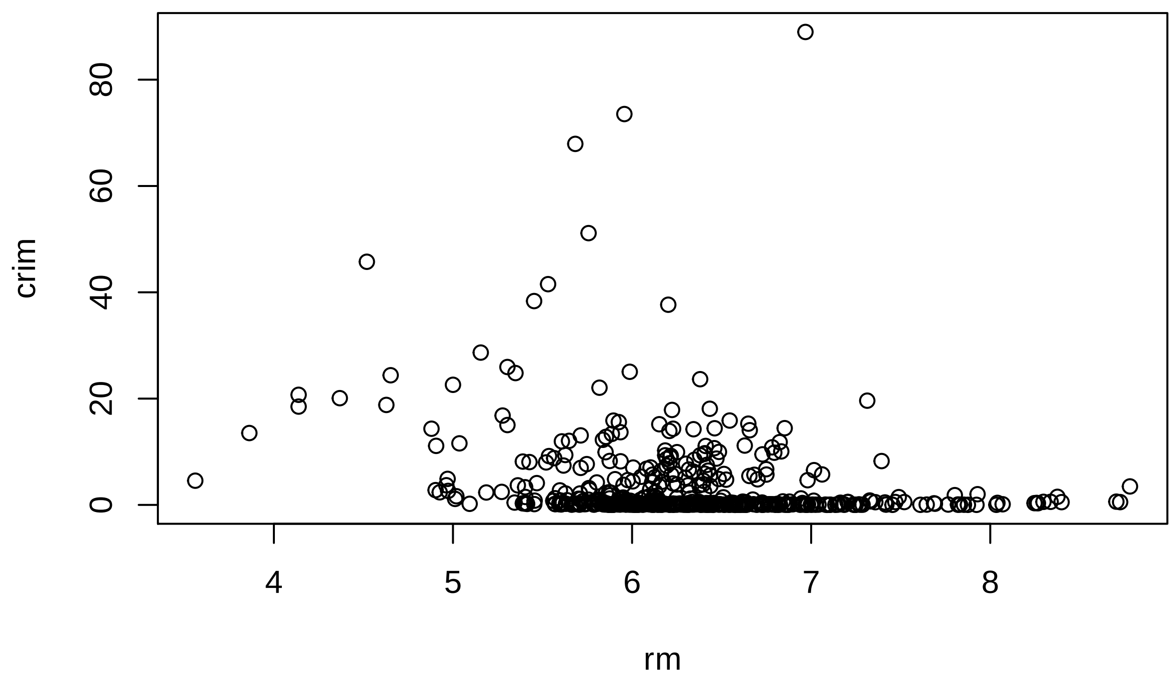



```r
lm.rm <- lm(crim ~ rm , data = Boston)
summary(lm.rm)
```

```
##
## 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
```

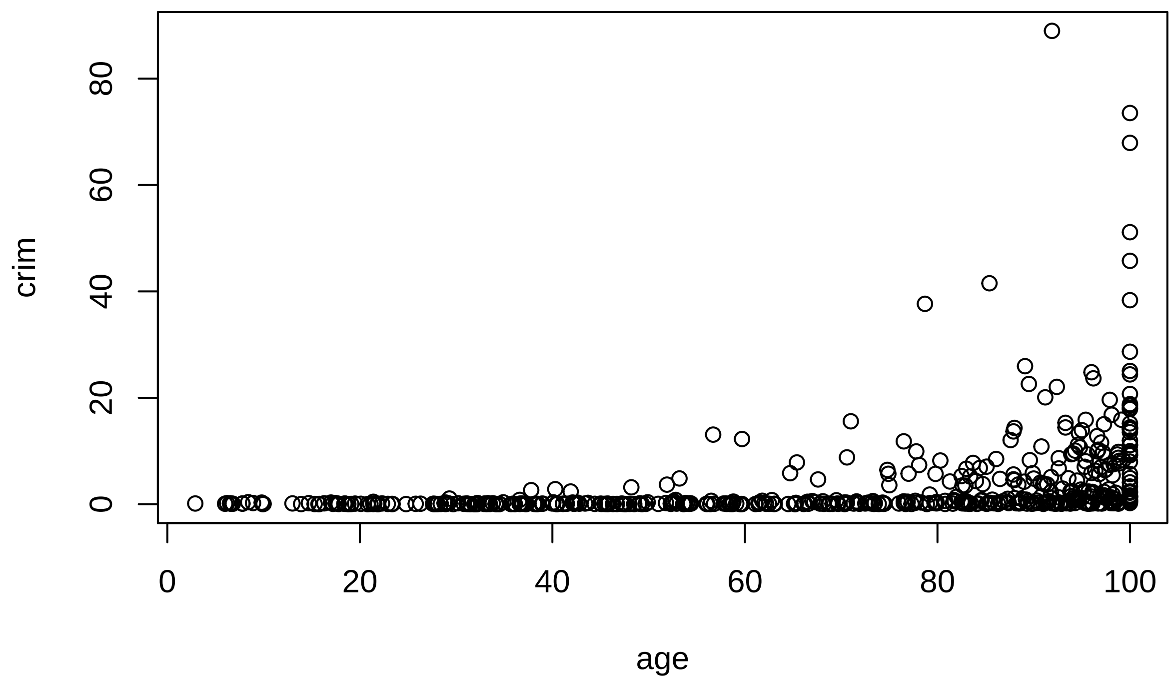
```
plot(rm, crim)
```




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

```
##
## 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
```

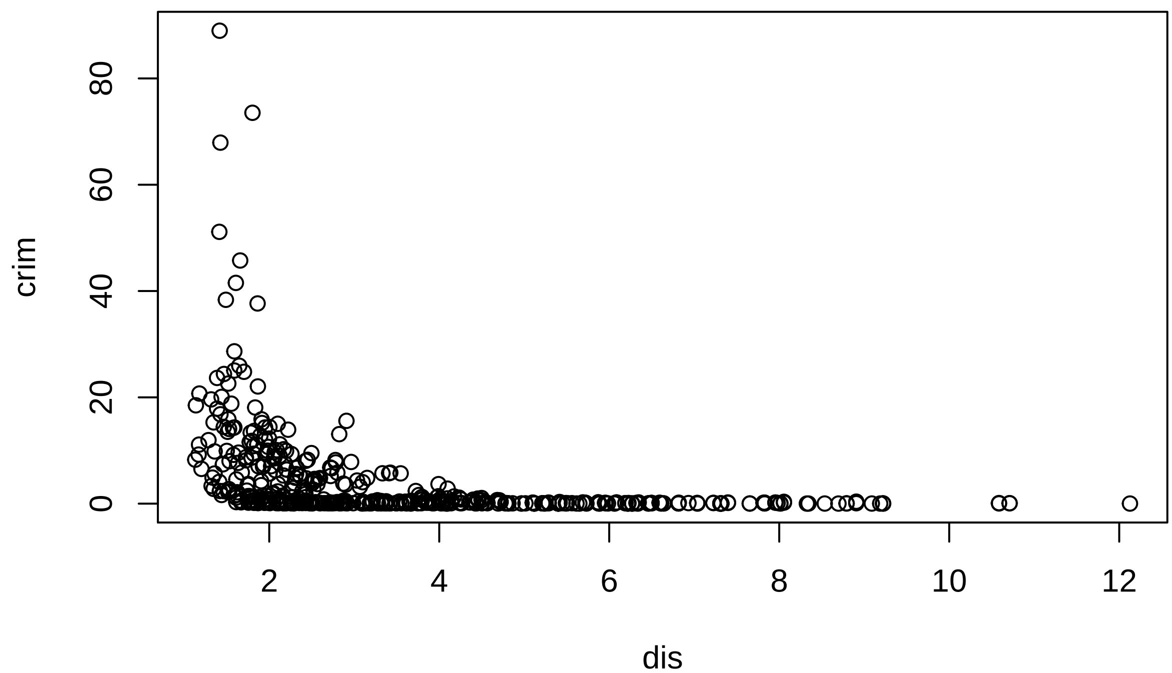
```r
plot(age, crim)
```




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

```
##
## 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
```

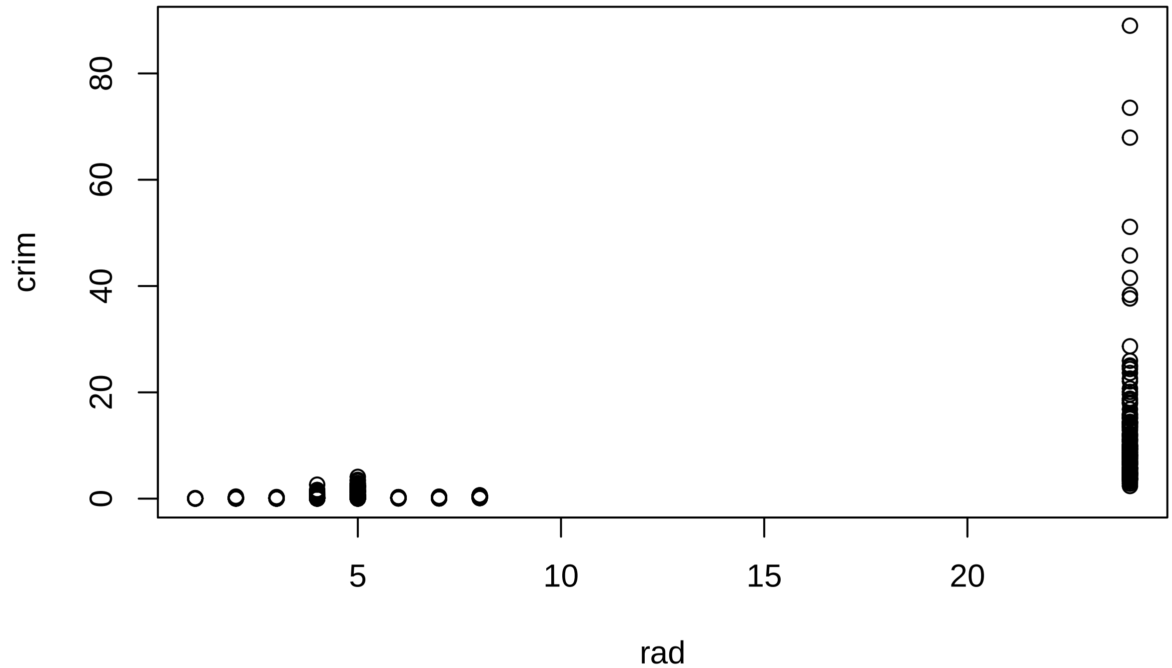
```r
plot(dis, crim)
```




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

```
##
## 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
```

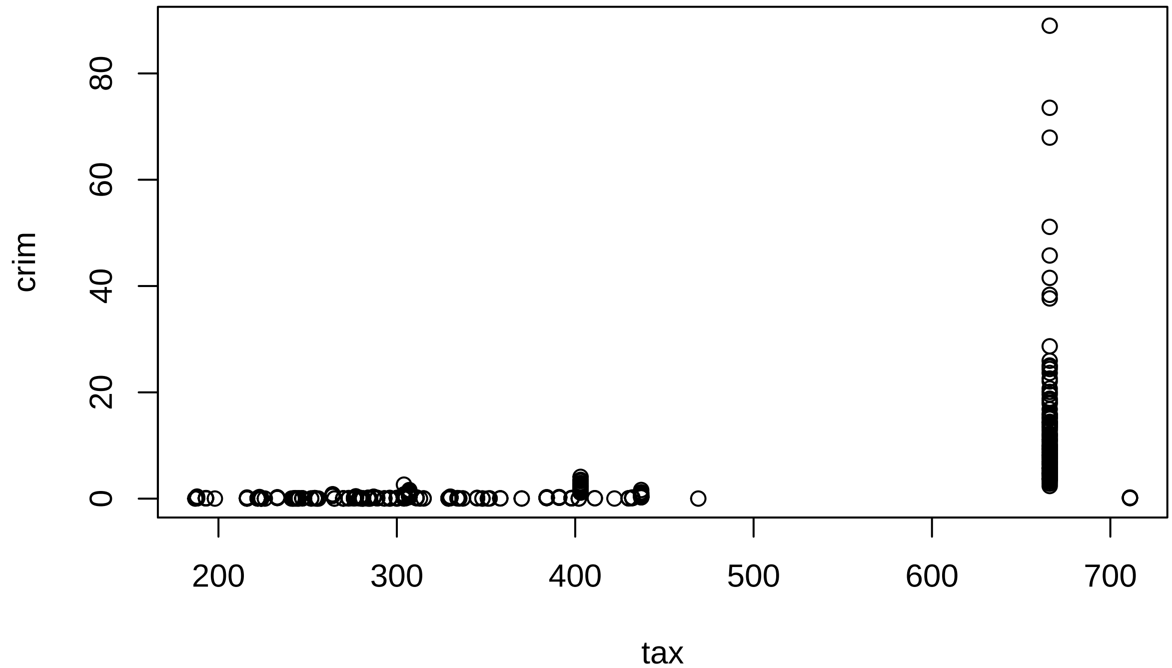
```r
plot(rad, crim)
```




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

```
##
## 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
```

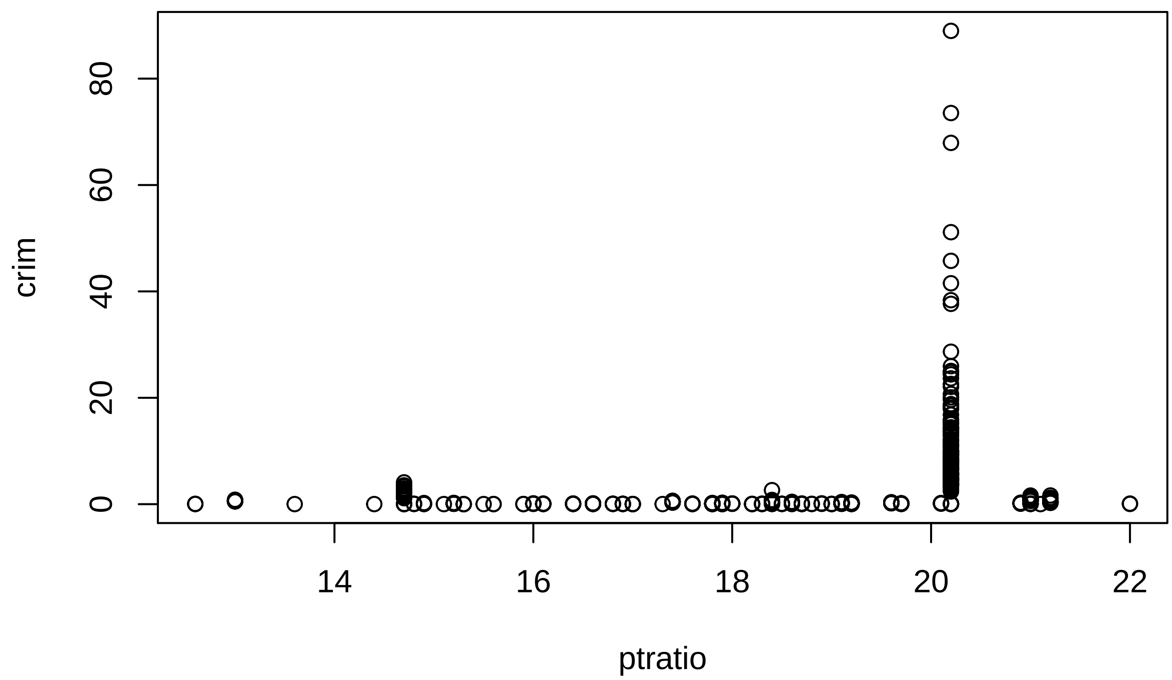
```r
plot(tax, crim)
```




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

```
##
## 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
```

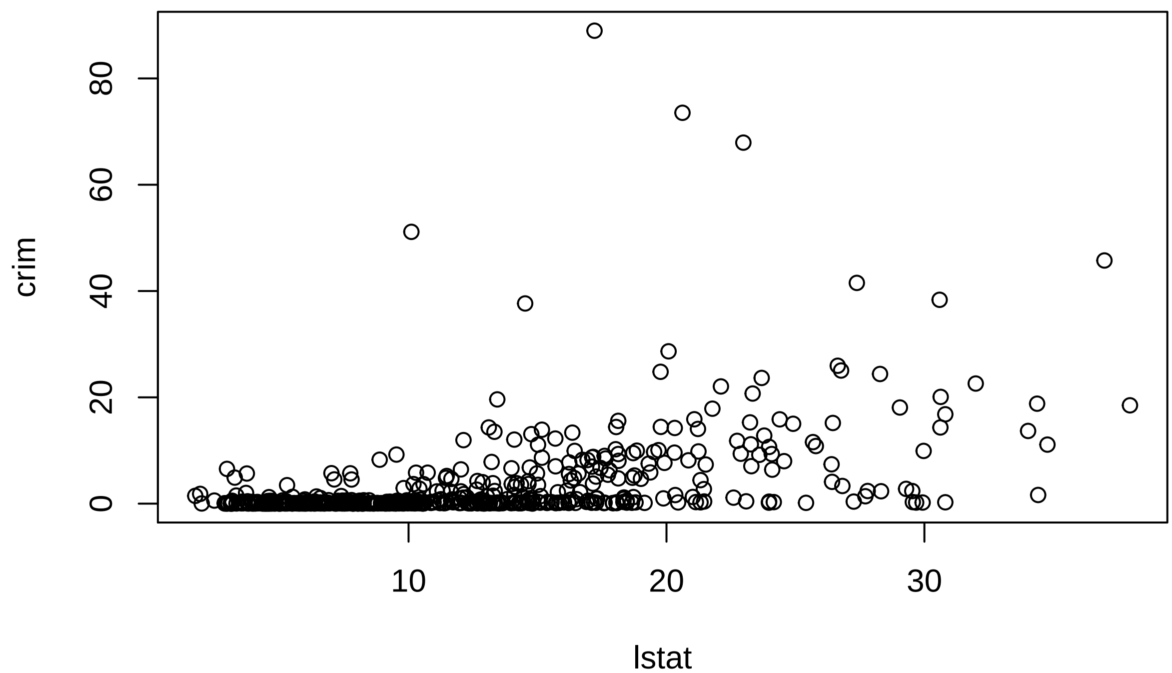
```r
plot(ptratio, crim)
```




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

```
##
## 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
```

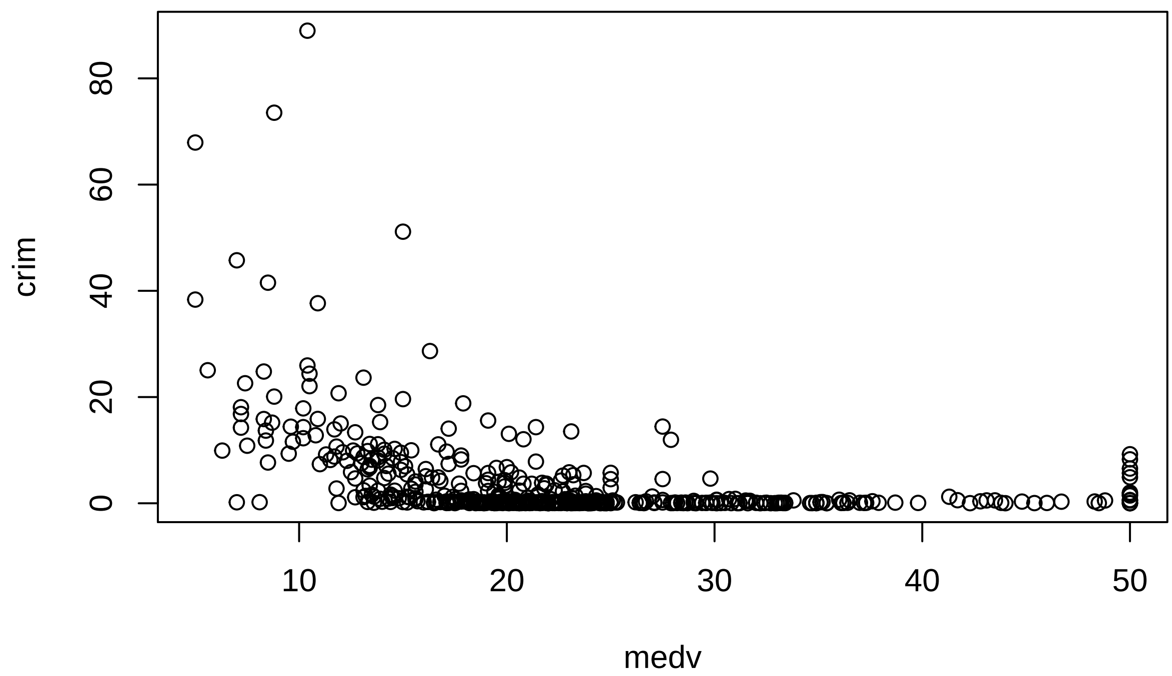
```r
plot(lstat, crim)
```




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

```
##
## 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
plot(medv, crim)
```



Models that has statistically significant association between the predictor and the response:

zn, indus, nox, rm, age, dis, rad, tax, ptratio, lstat, medv

(b) ***Fit a multiple regression model to predict the response using all of the predictors***

```r
lm.all <- lm(crim ~ . , data = Boston)
summary(lm.all)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.534 -2.248 -0.348  1.087 73.923
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.7783938  7.0818258   1.946 0.052271 .
## zn           0.0457100  0.0187903   2.433 0.015344 *
## indus       -0.0583501  0.0836351  -0.698 0.485709
## chas        -0.8253776  1.1833963  -0.697 0.485841
## nox         -9.9575865  5.2898242  -1.882 0.060370 .
## rm           0.6289107  0.6070924   1.036 0.300738
## age         -0.0008483  0.0179482  -0.047 0.962323
```

```
## dis         -1.0122467  0.2824676  -3.584 0.000373 ***
## rad          0.6124653  0.0875358   6.997 8.59e-12 ***
## tax         -0.0037756  0.0051723  -0.730 0.465757
## ptratio     -0.3040728  0.1863598  -1.632 0.103393
## lstat        0.1388006  0.0757213   1.833 0.067398 .
## medv        -0.2200564  0.0598240  -3.678 0.000261 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.46 on 493 degrees of freedom
## Multiple R-squared:  0.4493, Adjusted R-squared:  0.4359
## F-statistic: 33.52 on 12 and 493 DF,  p-value: < 2.2e-16
```

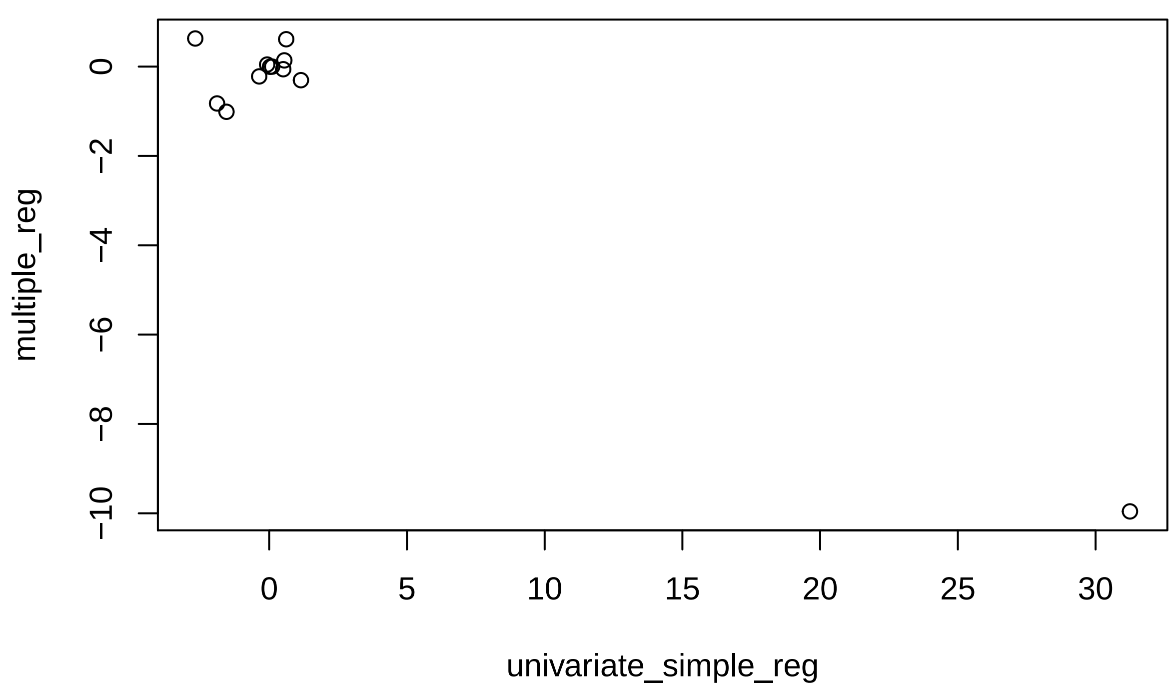
Predictors that we can reject the null hypothesis H0 = 0 are:

zn, dis, rad, medv

(c) ***Comparing (a) and (b)***

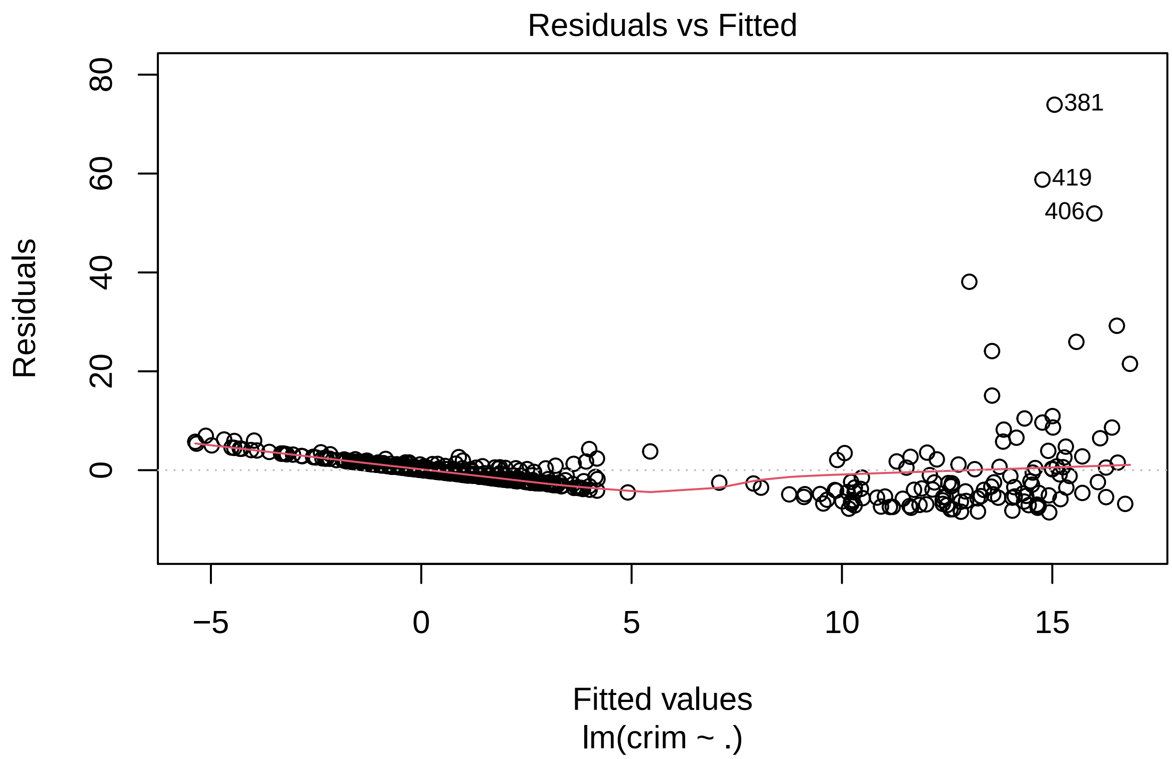
When observed independently, 11 predictors(zn, indus, nox, rm, age, dis, rad, tax, ptratio, lstat, medv) has statistically significant association between the predictor and the response. But when taken overall range of predictors, only 4 predictors(zn, dis, rad, medv) has statistically significant association between the predictor and the response.

```r
multiple_reg = c(coef(lm.all)[2:13])
univariate_simple_reg = c(coef(lm.zn)[2],coef(lm.indus)[2],coef(lm.chas)[2],coef(lm.nox)[2],coef(lm.rm)

plot(univariate_simple_reg, multiple_reg)
```



(d) Non-linear association between the predictors and the response exists.

```r
plot(lm.all, which =1)
```



Non-linear association of each predictor:

```r
lm.zn_poly <- lm(crim ~ poly(zn, 3))
summary(lm.zn_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(zn, 3))
##
## 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)    3.6135     0.3722   9.709  < 2e-16 ***
## poly(zn, 3)1 -38.7498     8.3722  -4.628  4.7e-06 ***
## poly(zn, 3)2  23.9398     8.3722   2.859  0.00442 **
## poly(zn, 3)3 -10.0719     8.3722  -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
```

```
lm.indus_poly <- lm(crim ~ poly(indus, 3))
summary(lm.indus_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(indus, 3))
##
## 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)        3.614      0.330  10.950  < 2e-16 ***
## poly(indus, 3)1   78.591      7.423  10.587  < 2e-16 ***
## poly(indus, 3)2  -24.395      7.423  -3.286  0.00109 **
## poly(indus, 3)3  -54.130      7.423  -7.292  1.2e-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
```

```
lm.chas_poly <- lm(crim ~ poly(chas, 1))
summary(lm.chas_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(chas, 1))
##
## 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)     3.6135     0.3822   9.455   <2e-16 ***
## poly(chas, 1) -10.8036     8.5966  -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
```

```
lm.nox_poly <- lm(crim ~ poly(nox, 3))
summary(lm.nox_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(nox, 3))
##
## 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)      3.6135     0.3216  11.237  < 2e-16 ***
## poly(nox, 3)1  81.3720     7.2336  11.249  < 2e-16 ***
## poly(nox, 3)2 -28.8286     7.2336  -3.985 7.74e-05 ***
## poly(nox, 3)3 -60.3619     7.2336  -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
lm.rm_poly <- lm(crim ~ poly(rm, 3))
summary(lm.rm_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(rm, 3))
##
## 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)     3.6135     0.3703   9.758  < 2e-16 ***
## poly(rm, 3)1 -42.3794     8.3297  -5.088 5.13e-07 ***
## poly(rm, 3)2  26.5768     8.3297   3.191  0.00151 **
## poly(rm, 3)3  -5.5103     8.3297  -0.662  0.50858
## ---
## 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
lm.age_poly <- lm(crim ~ poly(age, 3))
summary(lm.age_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(age, 3))
##
## 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)      3.6135     0.3485  10.368  < 2e-16 ***
## poly(age, 3)1  68.1820     7.8397   8.697  < 2e-16 ***
```

```
## poly(age, 3)2  37.4845     7.8397   4.781 2.29e-06 ***
## poly(age, 3)3  21.3532     7.8397   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
```

```r
lm.dis_poly <- lm(crim ~ poly(dis, 3))
summary(lm.dis_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(dis, 3))
##
## 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)     3.6135     0.3259  11.087  < 2e-16 ***
## poly(dis, 3)1 -73.3886     7.3315 -10.010  < 2e-16 ***
## poly(dis, 3)2  56.3730     7.3315   7.689 7.87e-14 ***
## poly(dis, 3)3 -42.6219     7.3315  -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
```

```r
lm.rad_poly <- lm(crim ~ poly(rad, 3))
summary(lm.rad_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(rad, 3))
##
## 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)     3.6135     0.2971  12.164  < 2e-16 ***
## poly(rad, 3)1 120.9074     6.6824  18.093  < 2e-16 ***
## poly(rad, 3)2  17.4923     6.6824   2.618  0.00912 **
## poly(rad, 3)3   4.6985     6.6824   0.703  0.48231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 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
lm.tax_poly <- lm(crim ~ poly(tax, 3))
summary(lm.tax_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(tax, 3))
##
## 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)      3.6135     0.3047  11.860  < 2e-16 ***
## poly(tax, 3)1 112.6458     6.8537  16.436  < 2e-16 ***
## poly(tax, 3)2  32.0873     6.8537   4.682 3.67e-06 ***
## poly(tax, 3)3  -7.9968     6.8537  -1.167    0.244
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## 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
lm.ptratio_poly <- lm(crim ~ poly(ptratio, 3))
summary(lm.ptratio_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(ptratio, 3))
##
## 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)           3.614      0.361  10.008  < 2e-16 ***
## poly(ptratio, 3)1    56.045      8.122   6.901 1.57e-11 ***
## poly(ptratio, 3)2    24.775      8.122   3.050  0.00241 **
## poly(ptratio, 3)3   -22.280      8.122  -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
lm.lstat_poly <- lm(crim ~ poly(lstat, 3))
summary(lm.lstat_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(lstat, 3))
```

```
##
## 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)       3.6135     0.3392  10.654   <2e-16 ***
## poly(lstat, 3)1  88.0697     7.6294  11.543   <2e-16 ***
## poly(lstat, 3)2  15.8882     7.6294   2.082   0.0378 *
## poly(lstat, 3)3 -11.5740     7.6294  -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
```

```r
lm.medv_poly <- lm(crim ~ poly(medv, 3))
summary(lm.medv_poly)
```

```
##
## Call:
## lm(formula = crim ~ poly(medv, 3))
##
## 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)       3.614      0.292  12.374  < 2e-16 ***
## poly(medv, 3)1  -75.058      6.569 -11.426  < 2e-16 ***
## poly(medv, 3)2   88.086      6.569  13.409  < 2e-16 ***
## poly(medv, 3)3  -48.033      6.569  -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
```

Question 3 : Chapter 6: #9

(a) Split the data into training and testing sets:

```r
set.seed (1)
num_rows <- nrow(College)
num_train_rows <- round(0.8 * num_rows)
train_indices <- sample(1:num_rows, size = num_train_rows, replace = FALSE)
College_train <- College[train_indices, ]
College_test <- College[-train_indices, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```r
lm_apps <- lm(Apps ~ ., data = College_train)
lm.pred <- predict(lm_apps, newdata = College_test)
mse_linear <- mean((College_test$Apps - lm.pred)^2)
mse_linear
```

```
## [1] 1578073
```

MSE for a linear model using least squares on the training set:

```r
mse_linear
```

```
## [1] 1578073
```

(c) Fit a ridge regression model on the training set, with lambda chosen by cross-validation. Report the test error obtained.

```r
x <- model.matrix(Apps~., data = College_train)[, -1]
y <- College_train$Apps

str(College_train)
```

```
## 'data.frame':    622 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 1 2 1 2 2 1 2 2 2 2 ...
##  $ Apps       : num  1401 344 4216 427 2929 ...
##  $ Accept     : num  1239 264 2290 385 1834 ...
##  $ Enroll     : num  605 97 736 143 622 ...
##  $ Top10perc  : num  10 11 20 18 20 10 27 50 62 13 ...
##  $ Top25perc  : num  34 42 52 38 56 35 50 77 93 33 ...
##  $ F.Undergrad: num  3716 500 4296 581 2738 ...
##  $ P.Undergrad: num  675 331 1027 533 1662 ...
##  $ Outstate   : num  7100 12600 5130 12700 12600 ...
##  $ Room.Board : num  4380 5520 4690 5800 5610 ...
##  $ Books      : num  540 630 600 450 450 537 450 525 500 570 ...
##  $ Personal   : num  2948 2250 1450 700 3160 ...
##  $ PhD        : num  63 77 73 81 90 77 77 76 94 66 ...
##  $ Terminal   : num  88 80 75 85 90 84 98 92 96 83 ...
##  $ S.F.Ratio  : num  19.4 10.4 17.9 10.3 15.1 21 21.5 10.1 9.6 16 ...
##  $ perc.alumni: num  0 7 18 37 9 16 21 57 20 14 ...
##  $ Expend     : num  5389 9773 5125 11758 9084 ...
##  $ Grad.Rate  : num  36 43 56 84 84 54 64 77 93 66 ...
```

```r
cv.ridge <- cv.glmnet(x, y, alpha =0)
bestlam_ridge <- cv.ridge$lambda.min
bestlam_ridge
```

```
## [1] 362.6608
```

```r
grid <- 10^ seq (10, -2, length = 100)

ridge.mod <- glmnet (x, y, alpha = 0, lambda = grid, thresh = 1e-12)

ridge.pred <- predict (ridge.mod , s = bestlam_ridge , newx = model.matrix(Apps~., data = College_test)

mse_ridge <- mean ((ridge.pred - College_test$Apps)^2)
mse_ridge
```

```
## [1] 1446029
```

MSE for a ridge regression model on the training set:

```
mse_ridge
```

```
## [1] 1446029
```

(d) Fit a lasso model on the training set, with lambda chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates

```
cv.lasso <- cv.glmnet (x, y, alpha = 1)

bestlam_lasso <- cv.lasso$lambda.min
lasso.pred <- predict (cv.lasso , s = bestlam_lasso , newx = model.matrix(Apps~., data = College_test)[

mse_lasso <- mean ((lasso.pred - College_test$Apps)^2)
print(mse_lasso)
```

```
## [1] 1565220
```

```
lasso_coef <- coef(cv.lasso, s = bestlam_lasso)[-1]
num_nonzero <- sum(lasso_coef !=0)
num_nonzero
```

```
## [1] 17
```

MSE for a lasso model on the training set:

```
mse_lasso
```

```
## [1] 1565220
```

Number of non-zero coefficient estimates:

```
num_nonzero
```

```
## [1] 17
```

Hence, there are no non-zero coefficients in the lasso model.

(e) Fit a PCR model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation

```
library (pls)
```

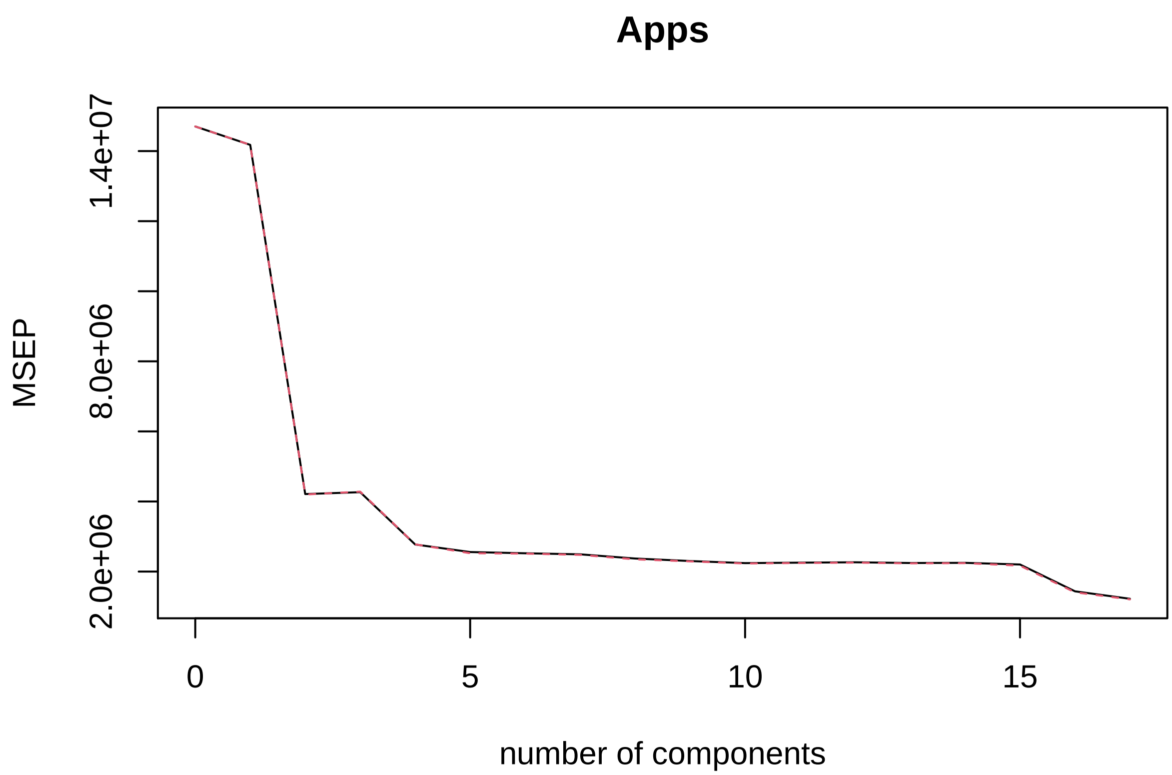
```
##
## Attaching package: 'pls'

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

```
set.seed (1)
cv.pcr <- pcr(Apps ~ ., data = College_train , scale = TRUE , validation = "CV")

summary(cv.pcr)
```

```
## Data:    X dimension: 622 17
##  Y dimension: 622 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3834     3765     2052     2065     1665     1599     1588
## adjCV         3834     3765     2049     2069     1667     1589     1585
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1579     1540     1517      1497      1501      1505      1498
## adjCV     1574     1533     1514      1494      1498      1502      1495
##        14 comps  15 comps  16 comps  17 comps
## CV         1500      1484      1199      1106
## adjCV      1497      1473      1187      1099
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X      32.019    57.05    64.13    70.01    75.36    80.38    84.09    87.44
## Apps    4.315    72.01    72.02    81.89    83.65    83.73    83.98    85.12
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       90.48     92.84     94.92     96.78     97.86     98.72     99.36
## Apps    85.40     85.75     85.75     85.76     85.88     85.94     89.94
##       16 comps  17 comps
## X        99.83    100.00
## Apps     92.88     93.47
```

```r
validationplot (cv.pcr , val.type = "MSEP")
```




```r
pcr.pred <- predict(cv.pcr, newdata = College_test, ncomp = 4)
mse_pcr <- mean((pcr.pred - College_test$Apps)^2)
print(mse_pcr)
```

```
## [1] 2691474
```

MSE for a PCR model on the training set:

```
mse_pcr
```

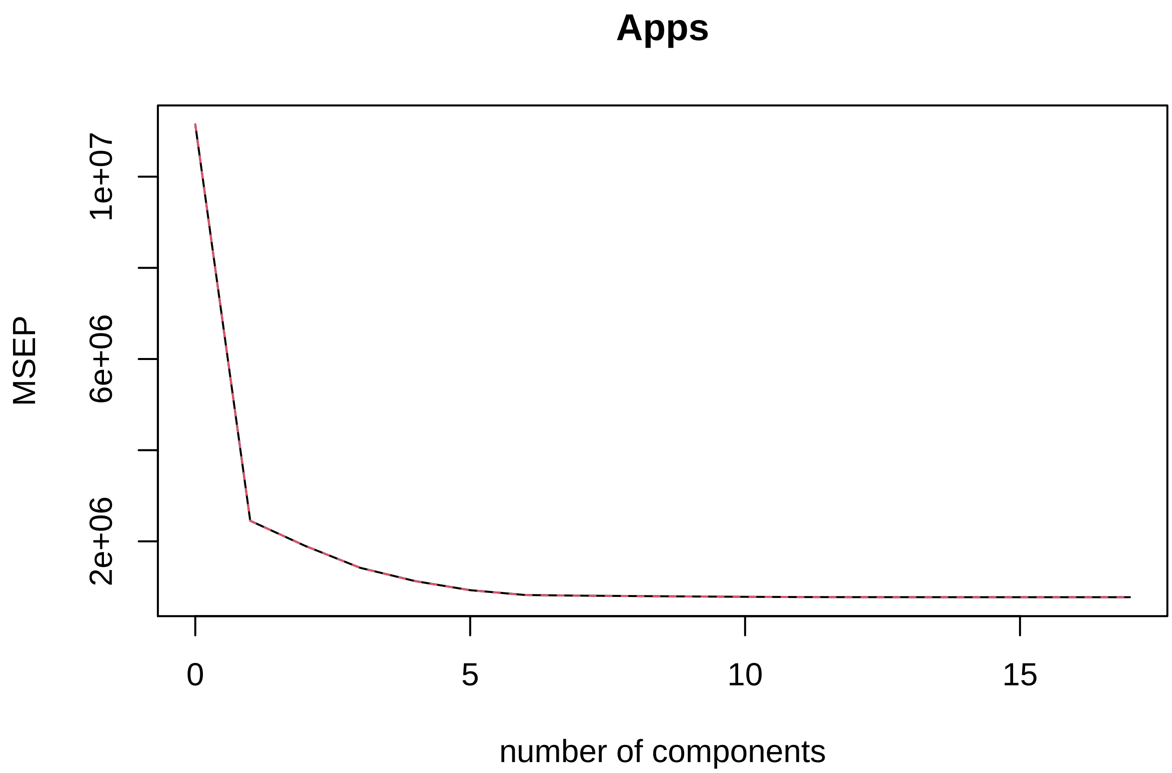
```
## [1] 2691474
```

(f) Fit a PLS model on the training set, with M chosen by cross-validation. Report the test error obtained, along with the value of M selected by cross-validation

```
set.seed (1)
cv.pls <- plsr(Apps ~ ., data=College, subset=model.matrix(Apps~., data = College_train)[, -1] , scale =

summary(cv.pls)
```

```
## Data:    X dimension: 6734 17
##  Y dimension: 6734 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3339     1566     1378     1191     1062    963.5    907.1
## adjCV         3339     1565     1378     1191     1061    962.4    906.2
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       899.3    894.1    889.3     884.6     881.9     880.8     880.4
## adjCV    898.5    893.3    888.6     883.8     881.3     880.1     879.7
##        14 comps  15 comps  16 comps  17 comps
## CV        880.3     880.3     880.2     880.2
## adjCV     879.6     879.6     879.6     879.6
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       25.05    47.04    62.30    66.57    70.50    73.67    78.20    80.90
## Apps    78.18    83.17    87.53    90.22    91.97    92.85    92.96    93.03
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X       83.14     84.67     87.83     91.12     91.93     93.72     95.41
## Apps    93.11     93.19     93.22     93.23     93.24     93.24     93.24
##       16 comps  17 comps
## X        97.12    100.00
## Apps     93.24     93.24
```

```
validationplot (cv.pls, val.type = "MSEP")
```




```r
pls.pred <- predict (cv.pls , newdata = College_test, ncomp = 2)
mse_pls <- mean((pls.pred - College_test$Apps)^2)
print(mse_pls)
```

```
## [1] 2666841
```

MSE for a PLS model on the training set:

```r
mse_pls
```

```
## [1] 2666841
```

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```r
test.avg = mean(College_test$Apps)
lm.r2 = 1 - mean((lm.pred - College_test$Apps)^2) / mean((test.avg - College_test$Apps)^2)
print(lm.r2)
```

```
## [1] 0.9011472
```

```r
ridge.r2 = 1 - mean((ridge.pred - College_test$Apps)^2) / mean((test.avg - College_test$Apps)^2)
print(ridge.r2)
```

```
## [1] 0.9094186
```

```r
lasso.r2 = 1 - mean((lasso.pred - College_test$Apps)^2) / mean((test.avg - College_test$Apps)^2)
print(lasso.r2)
```

```
## [1] 0.9019524
```

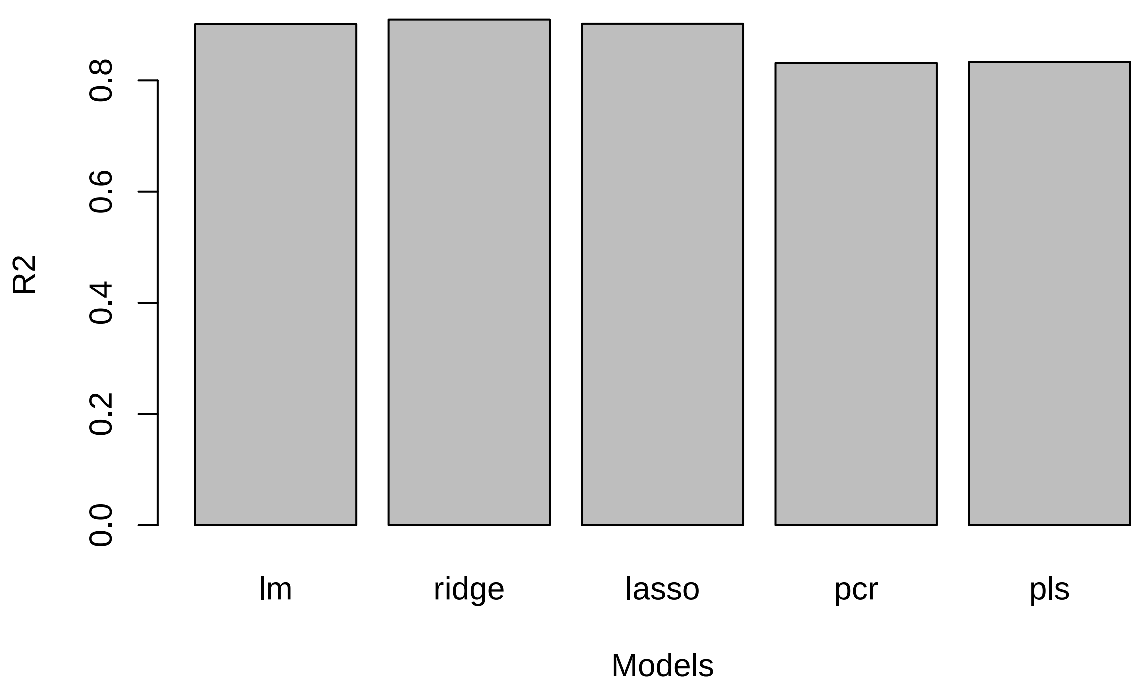
```
pcr.r2 = 1 - mean((pcr.pred - College_test$Apps)^2) / mean((test.avg - College_test$Apps)^2)
print(pcr.r2)
```

```
## [1] 0.8314022
```

```
pls.r2 = 1 - mean((pls.pred - College_test$Apps)^2) / mean((test.avg - College_test$Apps)^2)
print(pls.r2)
```

```
## [1] 0.8329452
```

```
barplot(c(lm.r2, ridge.r2, lasso.r2, pcr.r2, pls.r2), xlab="Models", ylab="R2",names=c("lm", "ridge", "l
```



All models have almost same R-squared value, of around 0.85 and above. So, we the results are pretty accurate to predict the number of applications received.

Question 4 : Chapter 6: #11

(a)

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.0.5
```

```
set.seed (1)

Boston_num_rows <- nrow(Boston)
Boston_num_train_rows <- round(0.8 * Boston_num_rows)
train_indices <- sample(1:Boston_num_rows, size = Boston_num_train_rows, replace = FALSE)
Boston_train <- Boston[train_indices, ]
```

```r
Boston_test <- Boston[-train_indices, ]

Boston_x <- model.matrix(crim~., data = Boston_train)[, -1]
Boston_y <- Boston_train$crim


best_subset <- regsubsets(crim ~ ., data = Boston_train[, -c(14)], method = "exhaustive", nvmax = ncol(B
summary_best_subset <- summary(best_subset)
names(best_subset)
```

```
##  [1] "np"        "nrbar"     "d"         "rbar"      "thetab"    "first"
##  [7] "last"      "vorder"    "tol"       "rss"       "bound"     "nvmax"
## [13] "ress"      "ir"        "nbest"     "lopt"      "il"        "ier"
## [19] "xnames"    "method"    "force.in"  "force.out" "sserr"     "intercept"
## [25] "lindep"    "nullrss"   "nn"        "call"
```

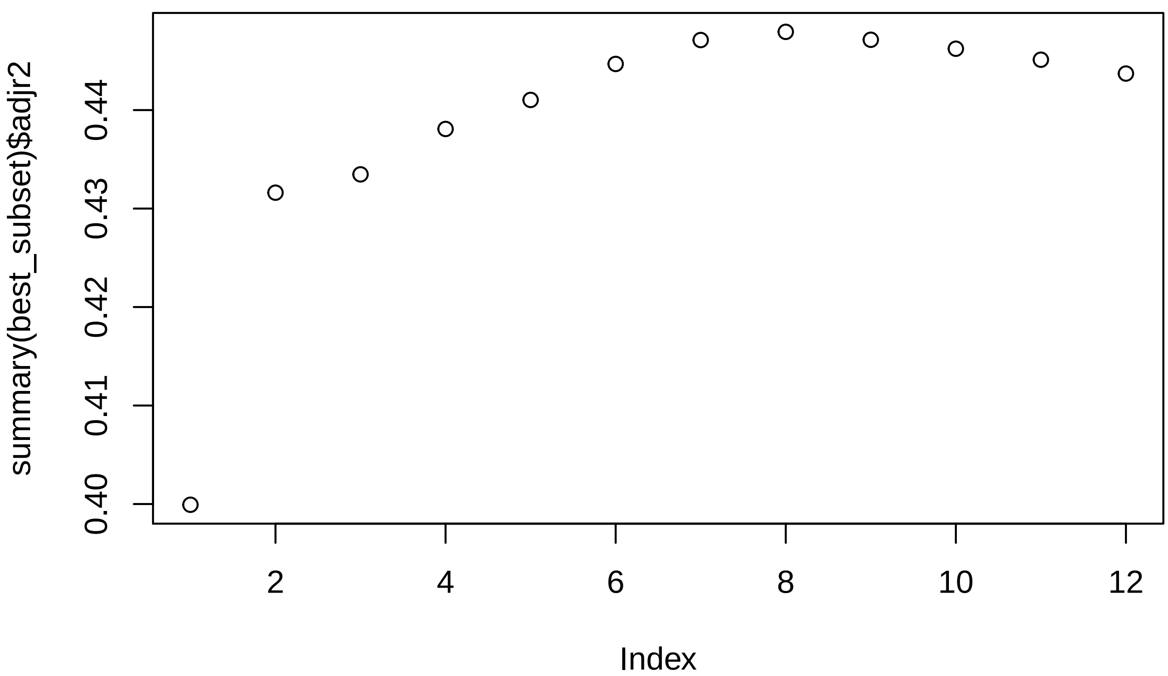
```r
names(summary(best_subset))
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

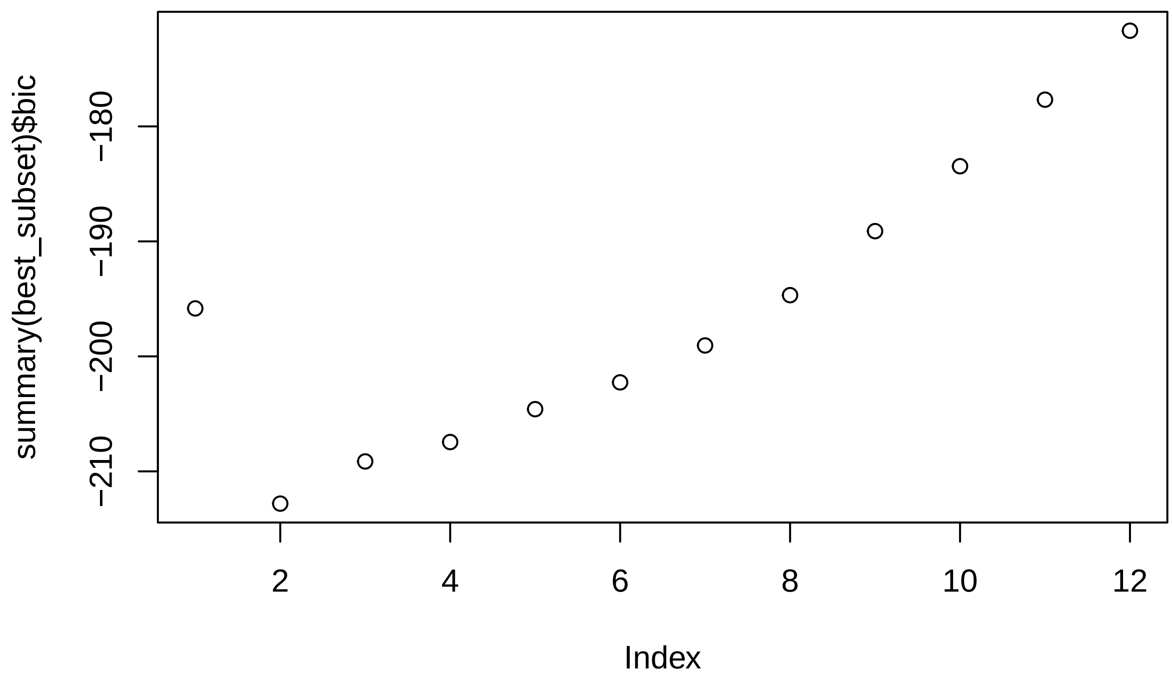
```r
summary(best_subset)$adjr2
```

```
##  [1] 0.3999267 0.4316155 0.4334748 0.4380825 0.4410310 0.4446849 0.4471137
##  [8] 0.4479435 0.4471402 0.4462307 0.4451137 0.4437097
```

```r
plot(summary(best_subset)$adjr2)
```



```r
plot(summary(best_subset)$bic)
```




```r
best_subset_model <- which.min(summary_best_subset$cp)
print(best_subset_model)
```

```
## [1] 7
```

```r
best_subset_variables <- names(coef(best_subset, id = best_subset_model))
print(best_subset_variables)
```

```
## [1] "(Intercept)" "zn"          "nox"         "dis"         "rad"
## [6] "ptratio"     "lstat"       "medv"
```

```r
#Lasso

Boston_lasso_model <- cv.glmnet(Boston_x, Boston_y, alpha = 1)
best_lambda_lasso <- Boston_lasso_model$lambda.min
print(best_lambda_lasso)
```

```
## [1] 0.0585163
```

```r
Boston_lasso_pred <- predict(Boston_lasso_model, newx = model.matrix(crim ~., data = Boston_test)[,-1],

Boston_mse_lasso <- mean((Boston_lasso_pred - Boston_test$crim)^2)
print(Boston_mse_lasso)
```

```
## [1] 65.82537
```

```r
Boston_lasso_coef <- coef(Boston_lasso_model, s = best_lambda_lasso)[-1]
print(length(Boston_lasso_coef))
```

```
## [1] 12
```

```r
Boston_num_nonzero <- sum(Boston_lasso_coef !=0)
print(Boston_num_nonzero)
```

```
## [1] 10
```

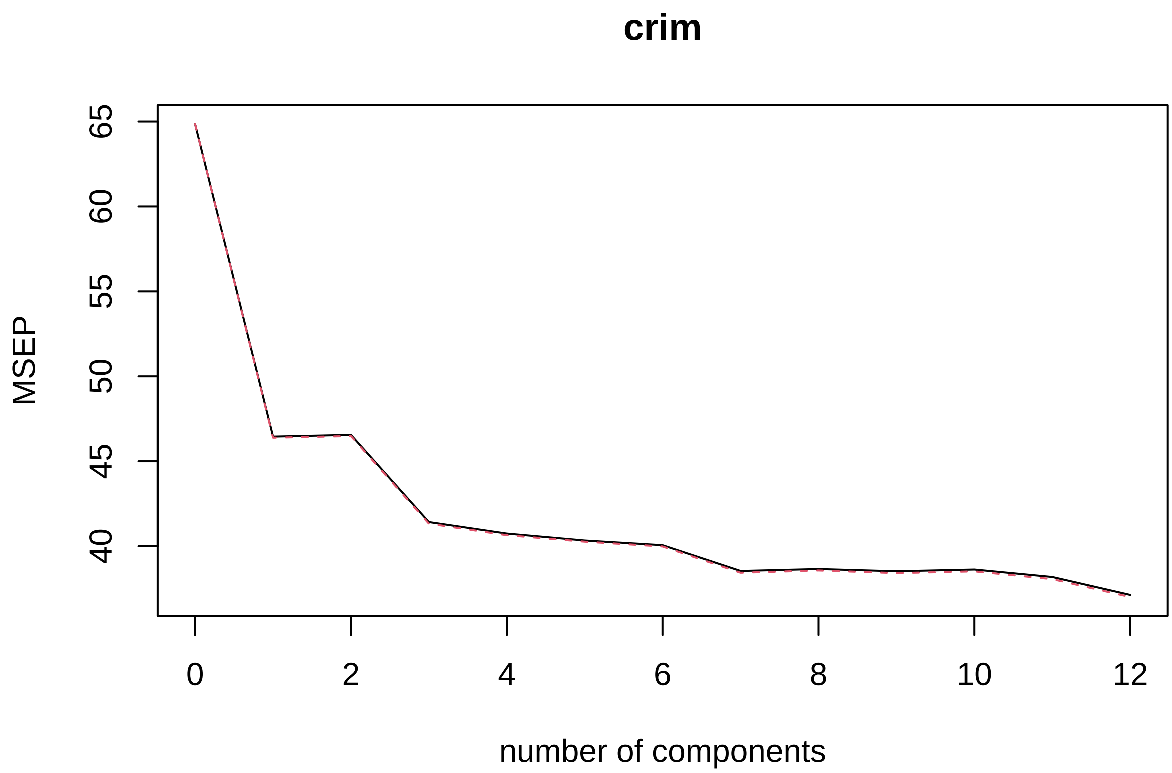
```r
#Ridge

Boston_ridge_model <- cv.glmnet(Boston_x, Boston_y, alpha = 0)
best_lambda_ridge <- Boston_ridge_model$lambda.min

Boston_ridge_pred <- predict(Boston_ridge_model, newx = model.matrix(crim ~., data = Boston_test)[,-1],

Boston_mse_ridge <- mean((Boston_ridge_pred - Boston_test$crim)^2)
print(Boston_mse_ridge)
```

```
## [1] 66.28151
```

```r
#PCR

set.seed (1)
Boston_pcr_model <- pcr(crim ~ ., data = Boston_train , scale = TRUE , validation = "CV")

summary(Boston_pcr_model)
```

```
## Data:    X dimension: 405 12
##  Y dimension: 405 1
## Fit method: svdpc
## Number of components considered: 12
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           8.053    6.815    6.823    6.436    6.383    6.351    6.329
## adjCV        8.053    6.811    6.818    6.429    6.376    6.346    6.323
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
## CV       6.208    6.217    6.207     6.215     6.180     6.094
## adjCV    6.200    6.211    6.198     6.206     6.169     6.084
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       50.32    63.99    73.14    80.26    86.58    90.13    92.69    94.85
## crim    29.80    29.98    37.79    38.91    39.64    40.18    42.82    43.06
##       9 comps  10 comps  11 comps  12 comps
## X       96.70     98.23     99.44    100.00
## crim    43.45     43.51     44.51     46.02
```

```r
validationplot (Boston_pcr_model , val.type = "MSEP")
```




```r
Boston_pcr_pred <- predict(Boston_pcr_model, newdata = Boston_test, ncomp = 3)
Boston_mse_pcr <- mean((Boston_pcr_pred - Boston_test$crim)^2)
print(Boston_mse_pcr)
```

```
## [1] 71.57538
```

The MSE's obtained for the models are as below:

Lasso: 70.34441

Ridge: 71.32964

PCR: 75.44626

(b) ***Chosen Model: Lasso***

When compared to the other models, Lasso is giving out the least MSE value, hence it is chosen as the best model to predict the per capita crime rate in the Boston data set.

(c) ***Does chosen model involve all of the features in the data set?***

The Lasso model typically involves only a subset of the features in the data set, and it automatically performs feature selection by setting some coefficients to zero based on the strength of the regularization parameter(lambda).

The Lasso penalty (L1 regularization) encourages sparsity in the model, meaning that it will tend to shrink some coefficients to exactly zero, effectively excluding the corresponding features from the model. As a result, the Lasso model will only involve a subset of the features that have non-zero coefficients.

When observed, the non-zero coefficients are 11, out of 13 total coeffficients. So all the features of the data set are not involved in the chosen model.

Question 5 : Chapter 8: #8

(a)

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.0.5
```

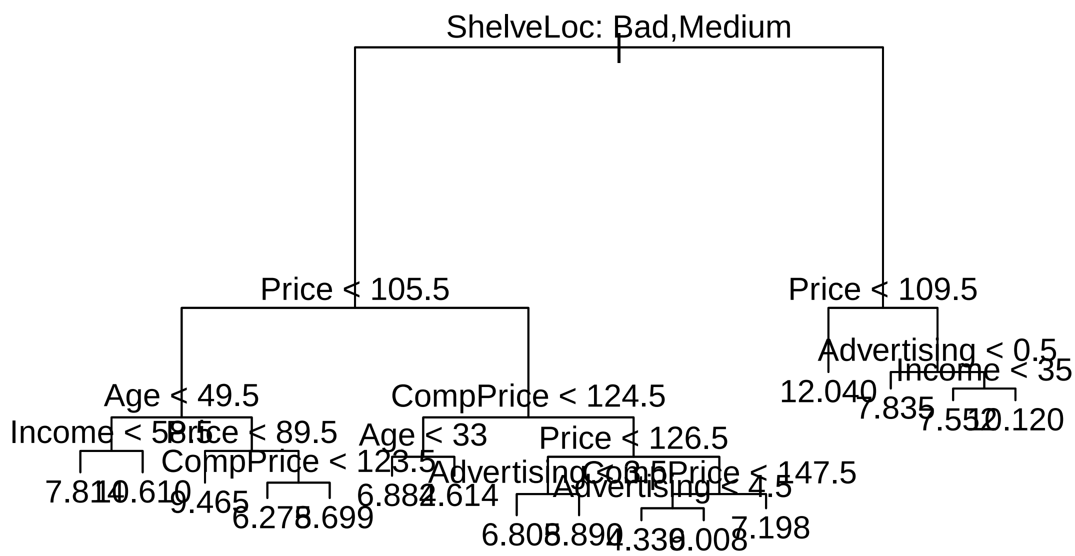
```
data(Carseats)

set.seed (1)

Carseats_num_rows <- nrow(Carseats)
Carseats_num_train_rows <- round(0.8 * Carseats_num_rows)
Carseats_train_indices <- sample(1:Carseats_num_rows, size = Carseats_num_train_rows, replace = FALSE)
Carseats_train <- Carseats[Carseats_train_indices, ]
Carseats_test <- Carseats[-Carseats_train_indices, ]
```

(b) ***Regression Tree***

```
Carseats_tree <- tree(Sales ~ ., data = Carseats, subset = Carseats_train_indices)
summary(Carseats_tree)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats, subset = Carseats_train_indices)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising"
## Number of terminal nodes:  16
## Residual mean deviance:  2.572 = 781.9 / 304
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.45400 -1.07000 -0.05544  0.00000  1.14500  4.69600
```

```
plot(Carseats_tree)
text (Carseats_tree , pretty = 0)
```

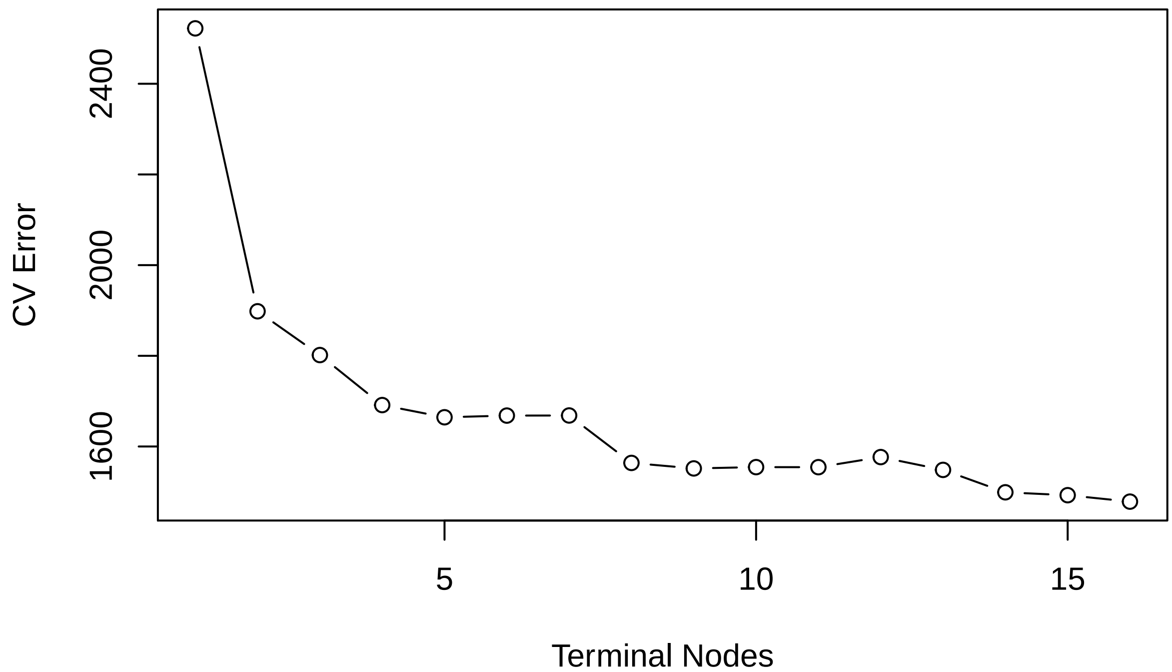


```
Carseats_tree_pred <- predict(Carseats_tree, Carseats_test)
Carseats_tree_mse <- mean((Carseats_tree_pred - Carseats_test$Sales)^2)
Carseats_tree_mse
```

```
## [1] 4.936081
```

(c) ***Cross Validation and Pruning***

```
Carseats_model <- cv.tree(Carseats_tree)

plot (Carseats_model$size , Carseats_model$dev, xlab="Terminal Nodes",ylab="CV Error", type = "b")
```



CV Error can be observed with 10 terminal nodes.

```
set.seed(1)
Carseats_prune <- prune.tree (Carseats_tree , best = 10)

Carseats_prune_pred <- predict(Carseats_prune, Carseats_test)
Carseats_prune_mse <- mean((Carseats_prune_pred - Carseats_test$Sales)^2)
Carseats_prune_mse
```

```
## [1] 5.088731
```

The MSE after pruning is slightly higher than the initial MSE.

(d) ***Bagging***

```
library(randomForest)
```

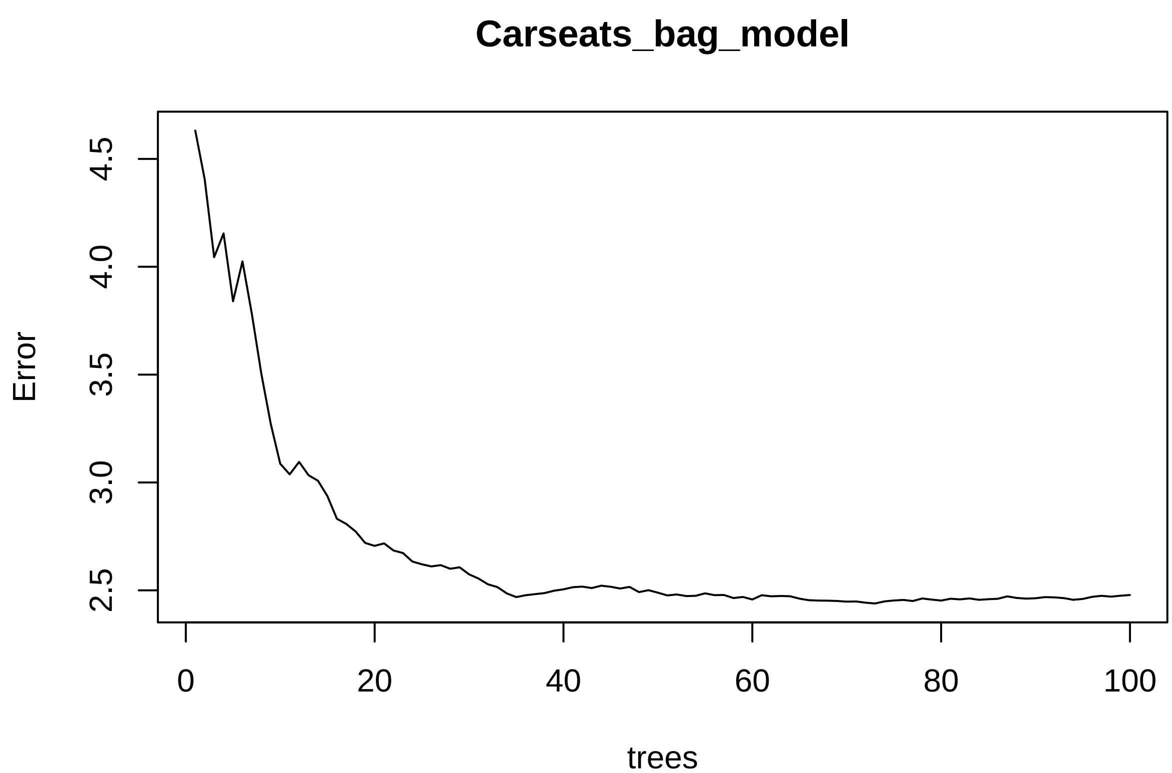
```
## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

```
set.seed (1)
Carseats_bag_model <- randomForest(Sales ~ ., data = Carseats, subset = Carseats_train_indices, mtry = 1
```

```r
summary(Carseats_bag_model)
```

```
##                 Length Class  Mode
## call              7    -none- call
## type              1    -none- character
## predicted       320    -none- numeric
## mse             100    -none- numeric
## rsq             100    -none- numeric
## oob.times       320    -none- numeric
## importance       20    -none- numeric
## importanceSD     10    -none- numeric
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               320    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```r
plot(Carseats_bag_model)
```



```r
Carseats_bag_imp <- importance(Carseats_bag_model)
print(Carseats_bag_imp)
```

```
##                %IncMSE IncNodePurity
## CompPrice   17.4478747    260.129519
## Income       3.4574929    145.060122
## Advertising 11.9178883    191.246961
## Population  -0.3592669     68.053683
## Price       35.6721781    730.264239
## ShelveLoc   34.8108576    711.031795
## Age         10.5239463    234.152513
## Education    0.4924419     63.399781
## Urban        0.4347503      8.039634
## US           1.2090286     13.022635
```

```r
Carseats_bag_pred <- predict(Carseats_bag_model, Carseats_test)
Carseats_bag_mse <- mean((Carseats_bag_pred - Carseats_test$Sales)^2)
Carseats_bag_mse
```

```
## [1] 2.953512
```

The most important variables are: "Price" and "ShelveLoc"

(e) ***Random Forest:***

```r
set.seed(1)
Carseats_rf1 <- randomForest(Sales ~ ., data = Carseats, subset = Carseats_train_indices, mtry = 10/2, i

Carseats_rf1_pred <- predict(Carseats_rf1, Carseats_test)
Carseats_rf1_mse <- mean((Carseats_rf1_pred - Carseats_test$Sales)^2)
Carseats_rf1_mse
```

```
## [1] 2.904885
```

```r
Carseats_rf2 <- randomForest(Sales ~ ., data = Carseats, subset = Carseats_train_indices, mtry = sqrt(1

Carseats_rf2_pred <- predict(Carseats_rf2, Carseats_test)
Carseats_rf2_mse <- mean((Carseats_rf2_pred - Carseats_test$Sales)^2)
Carseats_rf2_mse
```

```
## [1] 3.341028
```

```r
Carseats_rf3 <- randomForest(Sales ~ ., data = Carseats, subset = Carseats_train_indices, mtry = 10/4, i

Carseats_rf3_pred <- predict(Carseats_rf3, Carseats_test)
Carseats_rf3_mse <- mean((Carseats_rf3_pred - Carseats_test$Sales)^2)
Carseats_rf3_mse
```

```
## [1] 3.953503
```

```r
print(importance(Carseats_rf1))
```

```
##                %IncMSE IncNodePurity
## CompPrice   11.2062781     239.61956
## Income       5.2288146     164.48914
## Advertising  9.0728677     202.83535
## Population  -1.4633468     101.28015
## Price       26.2406889     670.88874
```

```
## ShelveLoc   32.1576694     644.20626
## Age         10.9095507     260.39518
## Education    1.2588148      79.66607
## Urban       -0.8358164      13.09046
## US           1.4865156      27.15364
```

```r
print(importance(Carseats_rf2))
```

```
##                 %IncMSE IncNodePurity
## CompPrice    8.3283018     219.24796
## Income       3.5092441     201.06354
## Advertising  7.3574402     197.87131
## Population  -0.6946125     149.00691
## Price       21.0410869     618.45313
## ShelveLoc   22.1877818     551.84473
## Age          8.8105245     279.70681
## Education    0.7254375     106.33700
## Urban       -0.9235665      20.65741
## US           2.4686358      32.88532
```

```r
print(importance(Carseats_rf3))
```

```
##                  %IncMSE IncNodePurity
## CompPrice    6.08869190     229.23544
## Income       2.85179888     196.10808
## Advertising  7.28546443     194.44014
## Population  -0.05440636     161.92170
## Price       18.38158146     535.53835
## ShelveLoc   17.62532191     460.48652
## Age          7.04992736     284.15561
## Education    0.40664814     123.44442
## Urban        0.82220261      24.76613
## US           3.39568022      41.45076
```

The most important variables are still "Price" and "ShelveLoc"

Number of variables considered at each split:

Effect of 'm': MSE has increased as the m value decreased.

(f) ***BART:***

```r
library(BART)
```

```
## Warning: package 'BART' was built under R version 4.0.5

## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## Loading required package: nnet

## Loading required package: survival
```

```r
set.seed(1)
```

```r
Carseats_x <- Carseats[, 1:11]
Carseats_y <- Carseats[, "Sales"]

Carseats_xtrain <- Carseats_x[Carseats_train_indices, ]
Carseats_ytrain <- Carseats_y[Carseats_train_indices]

Carseats_xtest <- Carseats_x[-Carseats_train_indices, ]
Carseats_ytest <- Carseats_y[-Carseats_train_indices]

bart_model <- gbart(Carseats_xtrain, Carseats_ytrain,x.test = Carseats_xtest)
```

```
## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 320, 15, 80
## y1,yn: 2.739219, 1.409219
## x1,x[n*p]: 10.360000, 1.000000
## xp1,xp[np*p]: 10.810000, 1.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 1
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset: 2,0.95,0.276302,3,2.63651e-30,7.62078
## *****sigma: 0.000000
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,15,0
## *****printevery: 100
##
## MCMC
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 11s
## trcnt,tecnt: 1000,1000
```

```r
yhat.bart <- bart_model$yhat.test.mean
bart_mse <- mean ((Carseats_ytest - yhat.bart)^2)
print(bart_mse)
```

```
## [1] 0.1031486
```

MSE through BART model is 0.1031486

Question 6 : Chapter 8: #11

(a) ***Create a training set consisting of the first 1,000 observations, and a test set consisting of the remaining observations***

```r
library(dplyr)

data(Caravan)
attach(Caravan)
Caravan$Purchase <- if_else(Caravan$Purchase =="No", 0, 1)
summary(Caravan$Purchase)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.05977 0.00000 1.00000
```

```r
Caravan_train_indices <- sample(1:1000, replace = FALSE)
Caravan_train <- Caravan[Caravan_train_indices, ]
Caravan_test <- Caravan[-Caravan_train_indices, ]

summary(Caravan_train_indices)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0   250.8   500.5   500.5   750.2  1000.0
```

```r
dim(Caravan_train)
```

```
## [1] 1000   86
```

```r
dim(Caravan_test)
```

```
## [1] 4822   86
```

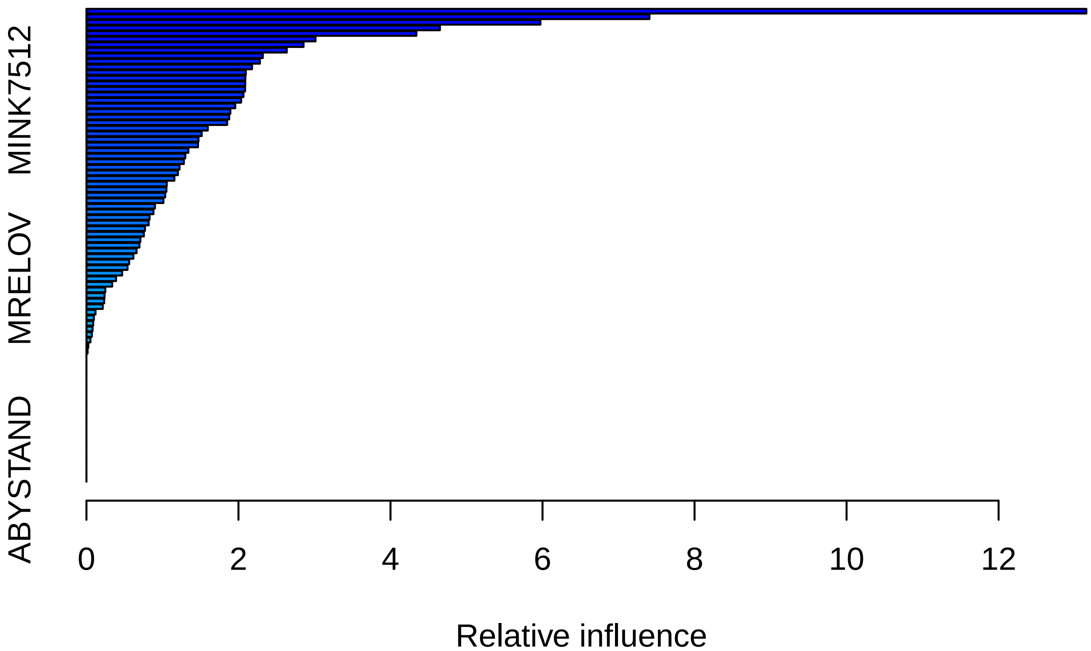
(b) ***Boosting:***

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.5
```

```
## Loaded gbm 2.1.8
```

```r
set.seed(1)

Caravan_boost_model <- gbm(Purchase ~ ., data = Caravan_test, n.trees =1000, interaction.depth = 4, shri

summary(Caravan_boost_model)
```




```
##                 var     rel.inf
## PPERSAUT PPERSAUT 13.15448009
## PBRAND     PBRAND  7.40675804
## PPLEZIER PPLEZIER  5.97181311
## MOSTYPE   MOSTYPE  4.65009324
## MOPLLAAG MOPLLAAG  4.34063426
## MBERMIDD MBERMIDD  3.01341938
## MINKGEM   MINKGEM  2.85647336
## MGODGE     MGODGE  2.63550663
## MBERHOOG MBERHOOG  2.32042762
## MAUT1       MAUT1  2.28180901
## MSKC         MSKC  2.17879114
## PBYSTAND PBYSTAND  2.09687058
## APERSAUT APERSAUT  2.09048005
## MINK3045 MINK3045  2.08862681
## MOPLHOOG MOPLHOOG  2.08742823
## MGODPR     MGODPR  2.06518725
## MINK7512 MINK7512  2.03472787
## MKOOPKLA MKOOPKLA  1.95766117
## MFGEKIND MFGEKIND  1.89379092
## MBERARBG MBERARBG  1.87800310
## ALEVEN     ALEVEN  1.85097834
## MRELGE     MRELGE  1.59652275
## MOPLMIDD MOPLMIDD  1.51616869
## PWAPART   PWAPART  1.47418180
## MSKB1       MSKB1  1.46762093
```

```
## MSKA         MSKA  1.34068795
## MGODOV     MGODOV  1.30032332
## MINK4575 MINK4575  1.28168102
## MBERARBO MBERARBO  1.22501212
## MINKM30   MINKM30  1.20343327
## MFWEKIND MFWEKIND  1.15674791
## MGEMLEEF MGEMLEEF  1.05668365
## MAUT2       MAUT2  1.05201628
## PFIETS     PFIETS  1.03614205
## MGODRK     MGODRK  1.01260064
## MZFONDS   MZFONDS  0.90132512
## MSKB2       MSKB2  0.88159619
## MAUT0       MAUT0  0.83084974
## MHKOOP     MHKOOP  0.81865006
## MBERZELF MBERZELF  0.76993179
## MZPART     MZPART  0.75529882
## MOSHOOFD MOSHOOFD  0.70950818
## PLEVEN     PLEVEN  0.69725115
## MHHUUR     MHHUUR  0.65952387
## MFALLEEN MFALLEEN  0.61762733
## AFIETS     AFIETS  0.56209436
## MRELSA     MRELSA  0.53980800
## MSKD         MSKD  0.46959900
## MRELOV     MRELOV  0.38987318
## MGEMOMV   MGEMOMV  0.33859236
## MBERBOER MBERBOER  0.24812533
## PMOTSCO   PMOTSCO  0.23851532
## PGEZONG   PGEZONG  0.23369117
## MINK123M MINK123M  0.21496659
## PINBOED   PINBOED  0.11944352
## PWAOREG   PWAOREG  0.09781305
## PWABEDR   PWABEDR  0.08841471
## MAANTHUI MAANTHUI  0.08053701
## PBROM       PBROM  0.07152588
## PAANHANG PAANHANG  0.05254905
## PTRACTOR PTRACTOR  0.02414746
## PWALAND   PWALAND  0.01495918
## PBESAUT   PBESAUT  0.00000000
## PVRAAUT   PVRAAUT  0.00000000
## PWERKT     PWERKT  0.00000000
## PPERSONG PPERSONG  0.00000000
## PZEILPL   PZEILPL  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
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

PPERSAUT, PBRAND are the most important predictors

(c)

```r
Caravan_boost_pred <- predict (Caravan_boost_model ,Caravan_test, type = "response", n.trees = Caravan_

Caravan_pred_20 <- if_else(Caravan_boost_pred> 0.2, "Yes", "No")

Caravan_actual <- Caravan_test$Purchase

conf_mat_table <- table(Caravan_pred_20, Caravan_actual)
conf_mat_table
```

```
##                  Caravan_actual
## Caravan_pred_20    0    1
##              No  4374  167
##              Yes  159  122
```

```r
people_fraction <- conf_mat_table["Yes", "1"] / sum(conf_mat_table[, "1"])
print(people_fraction)
```

```
## [1] 0.4221453
```

Fraction of the people predicted to make a purchase do in fact make one: 42%

Question 7 : Chapter 10: #7

```r
library(keras)
library(ISLR2)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.4
## v ggplot2   3.4.2     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x nlme::collapse()        masks dplyr::collapse()
## x randomForest::combine() masks dplyr::combine()
## x tidyr::expand()         masks Matrix::expand()
## x dplyr::filter()         masks stats::filter()
## x dplyr::lag()            masks stats::lag()
## x ggplot2::margin()       masks randomForest::margin()
## x tidyr::pack()           masks Matrix::pack()
## x MASS::select()          masks dplyr::select()
## x tidyr::unpack()         masks Matrix::unpack()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
default_data <- Default
```

```r
default_data$default <- as.numeric(default_data$default) - 1
default_data$student <- as.numeric(default_data$student)-1

set.seed(42)
Default_train_index <- sample(1:nrow(default_data), 0.8 * nrow(default_data))
Default_train_data <- default_data[Default_train_index, ]
Default_test_data <- default_data[-Default_train_index, ]

X_Default_train_data <- subset(Default_train_data, select = -c(default))
X_Default_test_data <- subset(Default_test_data, select = -c(default))

Y_Default_train_data <- Default_train_data$default
Y_Default_test_data <- Default_test_data$default

modnn <- keras_model_sequential()

modnn %>%
  layer_dense(units = 10, activation = 'relu', input_shape = c(ncol(X_Default_train_data))) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 1, activation = "sigmoid")


modnn %>%
  compile(optimizer=optimizer_rmsprop(),
  loss='binary_crossentropy',
  metrics='accuracy')

history <- modnn %>%
  fit(
    x = as.matrix(X_Default_train_data),
    y = Y_Default_train_data,
    epochs = 30,
    batch_size = 512,
    validation_data = list(as.matrix(X_Default_test_data), Y_Default_test_data)
  )


metrics <- modnn %>% evaluate(x = as.matrix(X_Default_test_data), y = Y_Default_test_data)

test_accuracy <- metrics[2]

cat("Neural Network Test Accuracy:", test_accuracy, "\n")
```

```
## Neural Network Test Accuracy: 0.9695
```

```r
Default_ll_reg <- glm(default ~ student+balance+income,family="binomial",data=Default_train_data)

Default_ll_pred <- predict(Default_ll_reg, data=Default_test_data, type='response') > 0.5
Default_ll_accuracy = mean(Default_ll_pred == Y_Default_test_data)
Default_ll_accuracy
```

```
## [1] 0.955
```

Problem 1: Beauty Data : Beauty Pays!

```r
library(readr)
Beauty_data <- read_csv("BeautyData.csv")
```

```
## Rows: 463 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (6): CourseEvals, BeautyScore, female, lower, nonenglish, tenuretrack
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
str(Beauty_data)
```

```
## spc_tbl_ [463 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ CourseEvals: num [1:463] 3.24 3.23 3.65 3.37 4.29 ...
##  $ BeautyScore: num [1:463] 0.202 -0.826 -0.66 -0.766 1.421 ...
##  $ female     : num [1:463] 1 0 0 1 1 0 1 1 1 0 ...
##  $ lower      : num [1:463] 0 0 0 0 0 0 0 0 0 0 ...
##  $ nonenglish : num [1:463] 0 0 0 0 0 0 0 0 0 0 ...
##  $ tenuretrack: num [1:463] 1 1 1 1 1 1 1 1 1 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   CourseEvals = col_double(),
##   ..   BeautyScore = col_double(),
##   ..   female = col_double(),
##   ..   lower = col_double(),
##   ..   nonenglish = col_double(),
##   ..   tenuretrack = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
model <- lm(CourseEvals ~ BeautyScore + female + lower + nonenglish + tenuretrack, data = Beauty_data)

summary(model)
```

```
##
## Call:
## lm(formula = CourseEvals ~ BeautyScore + female + lower + nonenglish +
##     tenuretrack, data = Beauty_data)
##
## 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
```

(1)

Analysis:

Beauty Score: The analysis shows that there is a significant positive relationship between instructor "beauty" (as measured by BeautyScore) and course evaluation ratings. As instructors are perceived as more beautiful, their course ratings tend to be higher.

Female: The analysis suggests that there is a significant negative relationship between the gender of the instructor (female) and course evaluation ratings. Courses taught by female instructors tend to receive lower ratings compared to their male counterparts.

Lower: The analysis shows that there is a significant negative relationship between the course level (lower) and course evaluation ratings. Introductory or lower-level courses tend to receive lower ratings compared to higher-level courses.

Non-English: The analysis suggests that there is a significant negative relationship between the language of instruction (non-English) and course evaluation ratings. Courses taught in languages other than English tend to receive lower ratings.

Tenure Track: The analysis suggests that there is a significant negative relationship between the tenure track status of the instructor and course evaluation ratings. Tenure-track instructors tend to receive slightly lower ratings compared to non-tenure-track instructors.

In summary, the analysis reveals that instructor "beauty" (BeautyScore) has a significant positive effect on course evaluation ratings. However, the effect size of "beauty" is relatively small compared to other factors such as the instructor's gender, course level, language of instruction, and tenure track status. These other factors also have significant influences on course evaluations.

(2)

Dr. Hamermesh's statement highlights the complexity of studying human behavior and the limitations of statistical analyses in establishing causation. While his research shows a significant positive relationship between instructor "beauty" and course evaluation ratings, it does not provide a definitive answer to whether this effect is primarily due to actual teaching productivity or potential discrimination based on appearance.

In social science research, it can be challenging to isolate and measure specific causal factors, especially when multiple variables are involved, and there may be unobservable or confounding factors at play. While regression analyses can help control for certain variables, they cannot completely address all potential determinants and interactions between factors.

To gain deeper insights, further research and experimental studies might be necessary to disentangle the specific effects of productivity and discrimination on course evaluation ratings. Additionally, exploring qualitative aspects, such as gathering feedback from students, conducting focus groups, or interviewing instructors, may provide valuable context to better understand the observed relationships. Nevertheless, Dr. Hamermesh's research sheds light on an interesting and relevant aspect of the evaluation process in education and highlights the importance of considering multiple factors when interpreting research findings.

Problem 2: MidCity : Housing Price Structure

```r
library(readr)
library(dplyr)
MidCity_data <- read_csv("MidCity.csv")
```

```
## Rows: 128 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Brick
## dbl (7): Home, Nbhd, Offers, SqFt, Bedrooms, Bathrooms, Price
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
str(MidCity_data)
```

```
## spc_tbl_ [128 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Home     : num [1:128] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Nbhd     : num [1:128] 2 2 2 2 2 1 3 3 2 2 ...
##  $ Offers   : num [1:128] 2 3 1 3 3 2 3 2 3 3 ...
##  $ SqFt     : num [1:128] 1790 2030 1740 1980 2130 1780 1830 2160 2110 1730 ...
##  $ Brick    : chr [1:128] "No" "No" "No" "No" ...
##  $ Bedrooms : num [1:128] 2 4 3 3 3 3 3 4 4 3 ...
##  $ Bathrooms: num [1:128] 2 2 2 2 3 2 3 2 2 3 ...
##  $ Price    : num [1:128] 114300 114200 114800 94700 119800 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Home = col_double(),
##   ..   Nbhd = col_double(),
##   ..   Offers = col_double(),
##   ..   SqFt = col_double(),
##   ..   Brick = col_character(),
##   ..   Bedrooms = col_double(),
##   ..   Bathrooms = col_double(),
##   ..   Price = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
MidCity_data$BrickYes <- if_else(MidCity_data$Brick == "Yes", 1, 0)
MidCity_data$N2 <- if_else(MidCity_data$Nbhd == 2, 1, 0)
MidCity_data$N3 <- if_else(MidCity_data$Nbhd == 3, 1, 0)

MidCity_model <- lm(Price ~ Offers + SqFt + BrickYes +N2 +N3 + Bathrooms + Bedrooms, data = MidCity_data
summary(MidCity_model)
```

```
##
## Call:
## lm(formula = Price ~ Offers + SqFt + BrickYes + N2 + N3 + Bathrooms +
##     Bedrooms, data = MidCity_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27337.3  -6549.5    -41.7   5803.4  27359.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2159.498   8877.810   0.243  0.80823
## Offers      -8267.488   1084.777  -7.621 6.47e-12 ***
## SqFt           52.994      5.734   9.242 1.10e-15 ***
## BrickYes    17297.350   1981.616   8.729 1.78e-14 ***
```

```
## N2          -1560.579   2396.765  -0.651  0.51621
## N3          20681.037   3148.954   6.568 1.38e-09 ***
## Bathrooms    7883.278   2117.035   3.724  0.00030 ***
## Bedrooms     4246.794   1597.911   2.658  0.00894 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10020 on 120 degrees of freedom
## Multiple R-squared:  0.8686, Adjusted R-squared:  0.861
## F-statistic: 113.3 on 7 and 120 DF,  p-value: < 2.2e-16
```

(1)

Coefficient estimate: 17297.350.

This means, on average, a brick house tends to sell for $17,297.35 more than a non-brick house, holding all other variables constant

p-value: 1.78e-14 (<0.05) - Statistically significant

So we can say that People pay a premium for a Brick house.

(2)

The Estimate for N3 is 20681.037, which means that the average difference in selling price between houses in Neighborhood 3 and houses in the reference neighborhood (usually Neighborhood 1) is $20,681.037.

The confidence interval for N3 is [14446.33, 26915.75]. Since the entire confidence interval is greater than zero, it indicates that people pay a premium to live in Neighborhood 3, even when accounting for other variables in the model.

Therefore, based on this regression analysis, we can conclude that there is a premium for houses in Neighborhood 3, and people are willing to pay more to live in this modern, newer, and more prestigious part of the town.

(3)

```r
MidCity_model_N3B <- lm(Price ~ Offers + SqFt + BrickYes +N2 +N3 + BrickYes:N3 + Bathrooms + Bedrooms, 
summary(MidCity_model_N3B)
```

```
##
## Call:
## lm(formula = Price ~ Offers + SqFt + BrickYes + N2 + N3 + BrickYes:N3 +
##     Bathrooms + Bedrooms, data = MidCity_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26939.1  -5428.7   -213.9   4519.3  26211.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3009.993   8706.264   0.346  0.73016
## Offers       -8401.088   1064.370  -7.893 1.62e-12 ***
## SqFt            54.065      5.636   9.593  < 2e-16 ***
## BrickYes     13826.465   2405.556   5.748 7.11e-08 ***
## N2            -673.028   2376.477  -0.283  0.77751
## N3           17241.413   3391.347   5.084 1.39e-06 ***
## Bathrooms     6463.365   2154.264   3.000  0.00329 **
## Bedrooms      4718.163   1577.613   2.991  0.00338 **
## BrickYes:N3 10181.577   4165.274   2.444  0.01598 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9817 on 119 degrees of freedom
## Multiple R-squared:  0.8749, Adjusted R-squared:  0.8665
## F-statistic:   104 on 8 and 119 DF,  p-value: < 2.2e-16
```

Coefficient Estimate for BrickYes:N3 : 10181.577.

This means that, in neighborhood three, the premium for a brick house is an additional $10,181.577 compared to non-brick houses, after accounting for the effect of other variables and the interaction between brick and neighborhood three.

p-value for BrickYes:N3 : 0.01598(<0.05)

Since the entire confidence interval is greater than zero, it indicates that people do pay an extra premium for brick houses in neighborhood three.

Therefore, based on this regression analysis, we can conclude that there is an extra premium for brick houses in neighborhood three. People are willing to pay more for a brick house in neighborhood three compared to other neighborhoods, even when considering the interaction effect and other variables in the model.

Overall, the analysis suggests that both brick houses and neighborhood three have positive effects on the selling price, and there is an additional premium for brick houses specifically in neighborhood three.

(4)

```r
MidCity_data$OlderNeighborhood <- if_else(MidCity_data$Nbhd %in% c(1, 2), 1, 0)

MidCity_model <- lm(Price ~ BrickYes + OlderNeighborhood + Offers + SqFt + Bedrooms + Bathrooms, data =

summary(MidCity_model)
```

```
##
## Call:
## lm(formula = Price ~ BrickYes + OlderNeighborhood + Offers +
##     SqFt + Bedrooms + Bathrooms, data = MidCity_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26810.5  -5953.6   -266.5   5662.9  26793.0
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        25005.043   9658.403   2.589 0.010807 *
## BrickYes           17058.771   1942.805   8.780 1.28e-14 ***
## OlderNeighborhood -21937.572   2482.393  -8.837 9.39e-15 ***
## Offers             -8019.003   1013.011  -7.916 1.32e-12 ***
## SqFt                  52.149      5.572   9.359 5.44e-16 ***
## Bedrooms            4070.005   1570.921   2.591 0.010751 *
## Bathrooms           7810.698   2109.060   3.703 0.000322 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9995 on 121 degrees of freedom
## Multiple R-squared:  0.8682, Adjusted R-squared:  0.8616
## F-statistic: 132.8 on 6 and 121 DF,  p-value: < 2.2e-16
```

The OlderNeighborhood coefficient is -21937.572, indicating that, on average, houses in the "old" neighborhood (combining neighborhoods 1 and 2) are associated with a decrease of $21,937.572 in price compared to houses in neighborhood 3, holding other variables constant.

Problem 3: What causes what??

(1)

The approach of obtaining data from different cities and running a regression analysis between "Crime" and "Police" may lead to misleading results due to potential confounding factors. It may show a positive correlation between police presence and crime rates, but this could be because cities with higher crime rates tend to hire more police officers in response.

To address this issue, it's crucial to consider alternative methods. Conducting controlled experiments in specific neighborhoods, exploring natural experiments resulting from policy changes, or employing longitudinal studies within the same city can provide more reliable insights. Additionally, incorporating other relevant variables in a multivariate regression or utilizing geospatial analysis can help control for confounding factors and obtain a clearer understanding of the relationship between police presence and crime rates. Qualitative research, such as interviews and surveys, can also complement the quantitative analysis by providing deeper context and perspectives from stakeholders.

(2)

The researchers at UPENN were able to isolate the effect of increased police presence on crime rates by utilizing a natural experiment. They collected data on crime in DC and correlated it with days when there was a higher alert for potential terrorist attacks. During these high-alert days, the DC mayor was required by law to deploy more police officers in the streets. Since this decision to increase police presence was not directly related to crime but rather a response to the high-alert status, it served as an effective natural experiment to study the impact of more cops on crime rates independently.

(3)

Controlling for METRO ridership was necessary to capture the potential influence of people's behavior on crime rates during high-alert days. If individuals were less likely to be out in public places, such as the subway, during those days due to the high-alert status, there would naturally be fewer opportunities for crimes to occur. This decrease in crime would not be directly attributed to increased police presence but rather a result of reduced opportunities for criminal activities. By factoring in ridership data, the researchers could isolate the specific effect of more police officers on crime rates and discern the true impact of increased police presence on crime independent of other external factors. The results from the analysis indicated that even after controlling for ridership, more police presence had a negative impact on crime rates, further supporting the hypothesis that increased police deployment contributed to a reduction in crime during high-alert days.

(4)

In Table 4 of the research paper, the researchers further refined their analysis to investigate whether the effect of high alert days on crime rates varied across different areas of the town (districts). They introduced interactions between location (districts) and high alert days to examine the specificity of the impact.

The conclusion drawn from this analysis was that the effect of high alert days on crime was primarily significant in District 1. This finding aligns with the logical expectation that potential terrorist targets in DC would be concentrated in District 1, leading to a higher likelihood of increased police deployment in that area during high alert days. In other districts, the effect of high alert days on crime was also negative, indicating a potential decrease in crime rates, but the impact was relatively small and could still be considered as having a possibility of being zero, given the standard error and confidence intervals.

Problem 4: Project Contribution:

Project: Ecommerce Customer Churn Analysis and Prediction

Group members:

JAHNAVI ANGATI, MEGHAVI SINGHANIYA, ANUBHAV NEHRU, ANUKUL KUMAR SINGH, HAYOUNG KIM

What?

Utilize customer-level attributes such as Tenure, Cashback Amount, and Warehouse to Home to predict Customer churn on an eCommerce Platform.

Why?

Customer Churn is the percentage of customers that drop out of the platform during a certain period of time. Predicting if a customer would exit their engagement on the platform helps in creating correct retention strategies to retarget the customers and build consumer intelligence.

How?

Using the eCommerce dataset, we ran the following Classification models to predict the churn : 1. Logistic Regression 2. Decision Tree 3. KNN 4. Random Forest Search 5. Boosting

My Contribution: KNN Classification

- The features in the training set and testing set are standardized using StandardScaler
- A KNN classifier is created and set to use k=5, for which it will consider 5 nearest neighbors for each data point during prediction.
- The code runs a loop from k=1 to k=31 to find the optimal value of k for the KNN model.
- For each k, a new KNN model is trained, and its error rate on the test set is calculated.
- The error rates are plotted against the k values, helping identify the best k value that minimizes the error rate.
- The best k value was found at k=1 with the least error rate, but was not considered as it is over-fitting the model.
- After determining the optimal k=4, The second best value for k, a new KNN model is created and trained on the scaled training data.
- The evaluation process is repeated for this optimized model, including recall, accuracy, and ROC-AUC curve.

ROC Curve:

- It is observed here that Decision Tree has the highest AUC among all the models, and hence chosen as the best model.
- Based on our data, we found that Tenure, Complain, Warehouse to Home distance and cash back amount are the key variables that dictate a customer's churn from the platform.
- Although we have chosen Decision Tree model, a similar variable importance is observed in all the other models as well.

Conclusion:

- The key metric that we used to compare the models was the Recall.
- Customers with longer tenure, farther warehouse-to-home distance, or lower cash back amounts are more likely to churn from the platform.
- Leveraging this information, the platform can devise targeted strategies to retain customers, such as offering personalized incentives to those with longer tenure, closer proximity to warehouses, or providing attractive cash back offers by focusing on customer experience and lower complain rates.