ISL_R_Practice

Ruchi Sharma

2022-08-25

An Introduction to Statistical Learning

This markdown has been created as a practice of some preliminary concepts by:
Ruchi Sharma [Graduate Student, UT Austin]

Chapter 2, Question 8

```
## (a)

library(ISLR)
summary(College)
```

```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate
##  Min.   : 10.00
##  1st Qu.: 53.00
```

```
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

```
names(College)
```

```
##  [1] "Private"     "Apps"        "Accept"      "Enroll"      "Top10perc"
##  [6] "Top25perc"   "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

```
dim(College) # 777 x 18
```

```
## [1] 777  18
```

```
## (b)
# fix(College)

college.rownames = rownames(College) # Creates Values in Envt

college = College[,-1]
# Creates new dataframe with 17 variables now instead of 18
names(college)
```

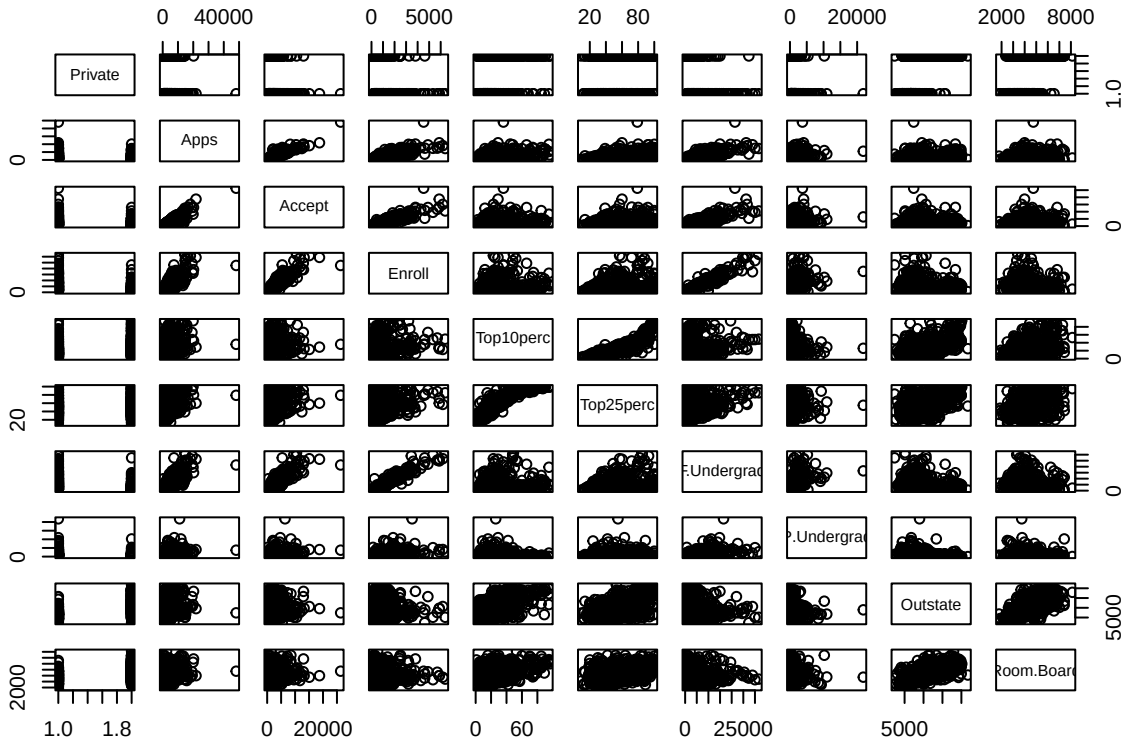
```
##  [1] "Apps"        "Accept"      "Enroll"      "Top10perc"   "Top25perc"
##  [6] "F.Undergrad" "P.Undergrad" "Outstate"    "Room.Board"  "Books"
## [11] "Personal"    "PhD"         "Terminal"    "S.F.Ratio"   "perc.alumni"
## [16] "Expend"      "Grad.Rate"
```

```
## (c)

summary(College)
```

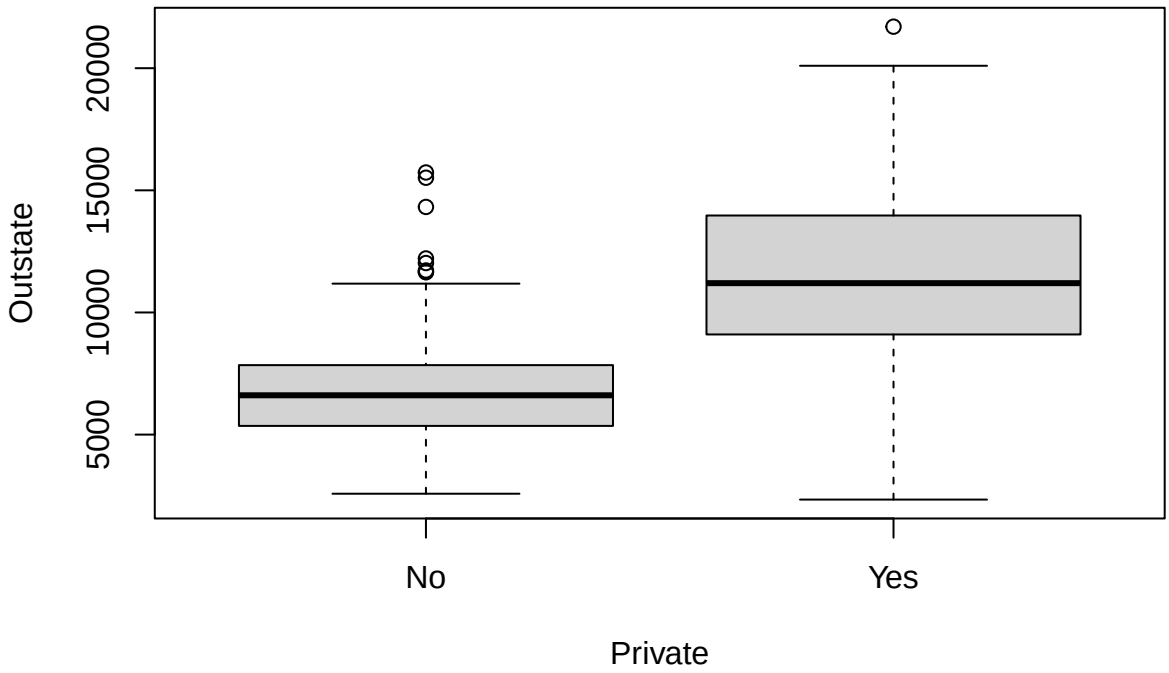
```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
```

```
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate
##  Min.   : 10.00
##  1st Qu.: 53.00
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

```
pairs(College[,1:10])
```




```
boxplot(Outstate~Private, data = College)
```




```
# binning

elite = rep("No", nrow(College))
elite[college$Top10perc>50] = "Yes"
elite = as.factor(elite)
College = data.frame(College, elite)
dim(College) # Another variable added to make it 19 from 18
```

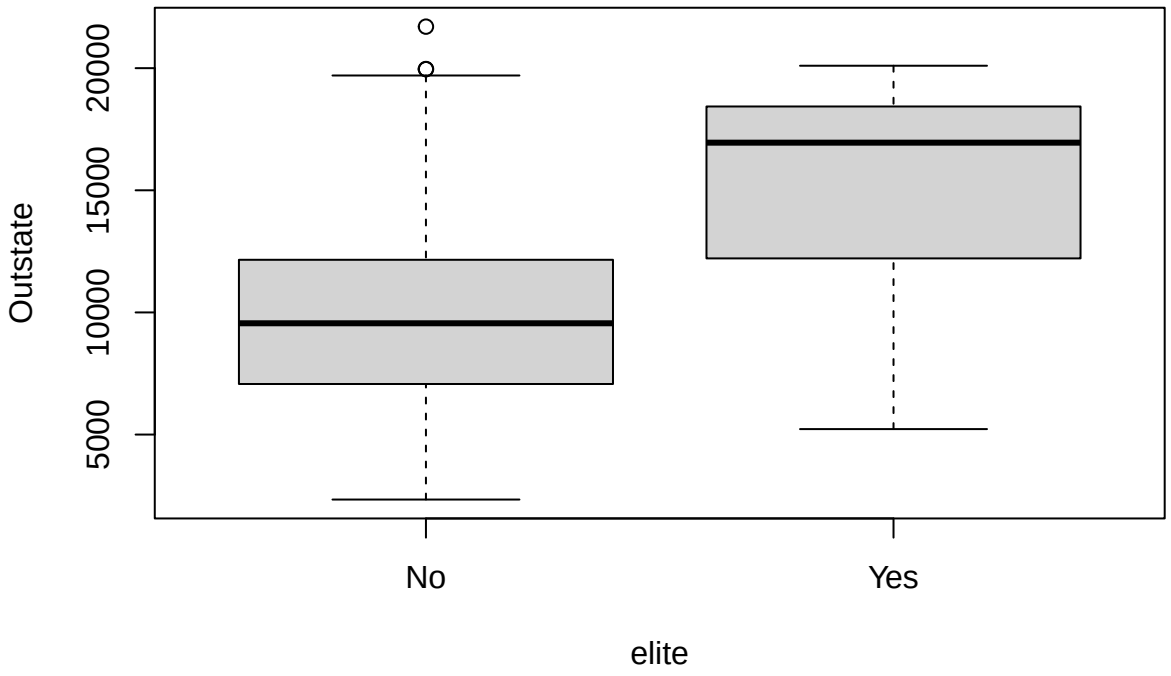
```
## [1] 777  19
```

```
summary(College) # There are 78 elite universities
```

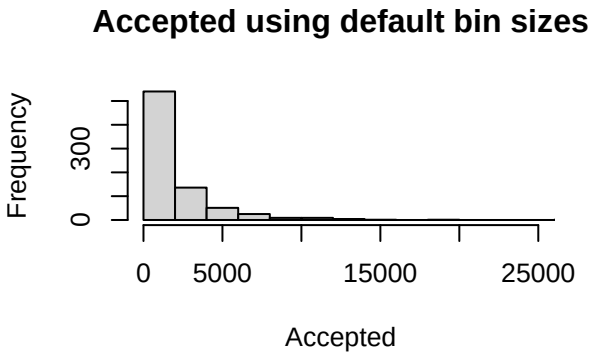
```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
```

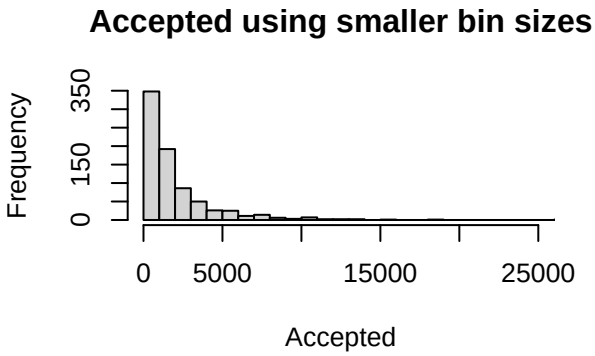
```
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate      elite
##  Min.   : 10.00   No :699
##  1st Qu.: 53.00   Yes: 78
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

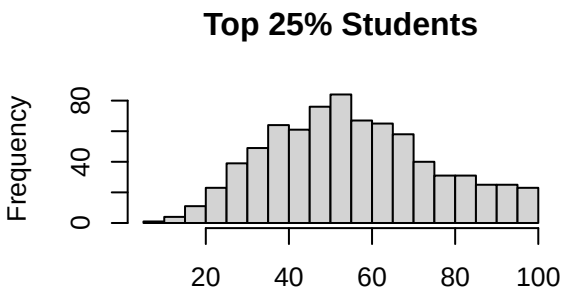
```
boxplot(Outstate~elite, data = College)
```

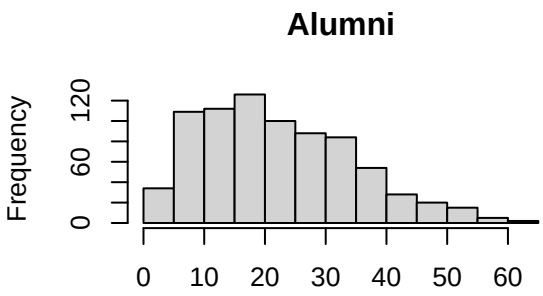



```
par(mfrow = c(2,2)) # divide window into 4 parts

hist(College$Accept, xlim = c(0, 25000), xlab = "Accepted", main = "Accepted using default bin sizes")
hist(College$Accept, xlim = c(0, 25000), breaks = 20, xlab = "Accepted", main = "Accepted using smaller
hist(College$Top25perc, breaks = 25, xlab = "%age of new students from 25% of High School", main = "Top
hist(College$perc.alumni, breaks = 10, xlab = "% alumni who donate", main = "Alumni")
```










```
# extending exploration

# Checking impact of Top10perc vs Grad.Rate

mean(College$Top10perc)
```

```
## [1] 27.55856
```

```
median(College$Top10perc)
```

```
## [1] 23
```

```
mean(College$Grad.Rate)
```

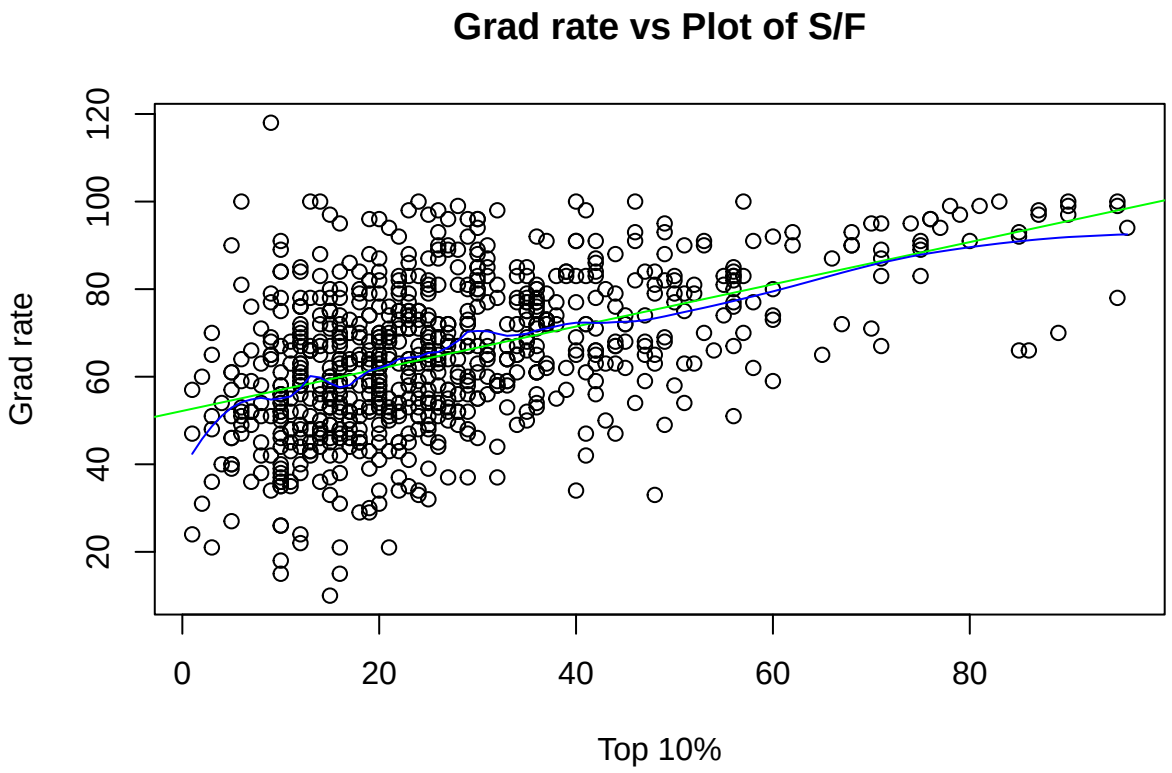
```
## [1] 65.46332
```

```
median(College$Grad.Rate)
```

```
## [1] 65
```

```
par(mfrow=c(1,1)) # re-setting par
plot(College$Top10perc, College$Grad.Rate, xlab = "Top 10%", ylab = 'Grad rate', main = "Grad rate vs P

# Plotting a linear regression line
abline(lm(College$Grad.Rate~College$Top10perc), col = "green")

# Although there is a direct relation but spread of the data around the line is huge
# Thus, we have high standard deviation or error

# Plotting local regression line with smoothing of 25%

loessMod = loess(Grad.Rate ~ Top10perc, data = College, span = 0.25)
```

```
j = order(College$Top10perc)
lines(College$Top10perc[j], loessMod$fitted[j], col = "blue")
```



Chapter 2, Question 9

```
library(ISLR)
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```
## (a)

summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
```

```
##                                          (Other)           :365
```

```
# Quantitative
# mpg, cylinder, displacement, horsepower, weight, acceleration, year

# Qualitative
# name, origin

# Although summary calculates stats for origin but it is evident that
# it is qualitative
temp = Auto$origin
# We can see the values taken by temp are not continuous quantities
```

```
## (b)

# range can also be seen in the max min of summary # Method 1
range(Auto$mpg) # Method 2
```

```
## [1]  9.0 46.6
```

```
sapply(Auto[,1:7], range) # Method 3
```

```
##       mpg cylinders displacement horsepower weight acceleration year
## [1,]  9.0         3           68         46   1613          8.0   70
## [2,] 46.6         8          455        230   5140         24.8   82
```

```
## (c)

sapply(Auto[,1:7], mean)
```

```
##          mpg    cylinders displacement   horsepower       weight acceleration
##    23.445918     5.471939   194.411990   104.469388  2977.584184    15.541327
##         year
##    75.979592
```

```
sapply(Auto[,1:7], sd)
```

```
##          mpg    cylinders displacement   horsepower       weight acceleration
##     7.805007     1.705783   104.644004    38.491160   849.402560     2.758864
##         year
##     3.683737
```

```
## (d)

Auto_Update <- Auto[-c(10:85),]
sapply(Auto_Update[,1:7], range)
```

```
##       mpg cylinders displacement horsepower weight acceleration year
## [1,] 11.0         3           68         46   1649          8.5   70
## [2,] 46.6         8          455        230   4997         24.8   82
```

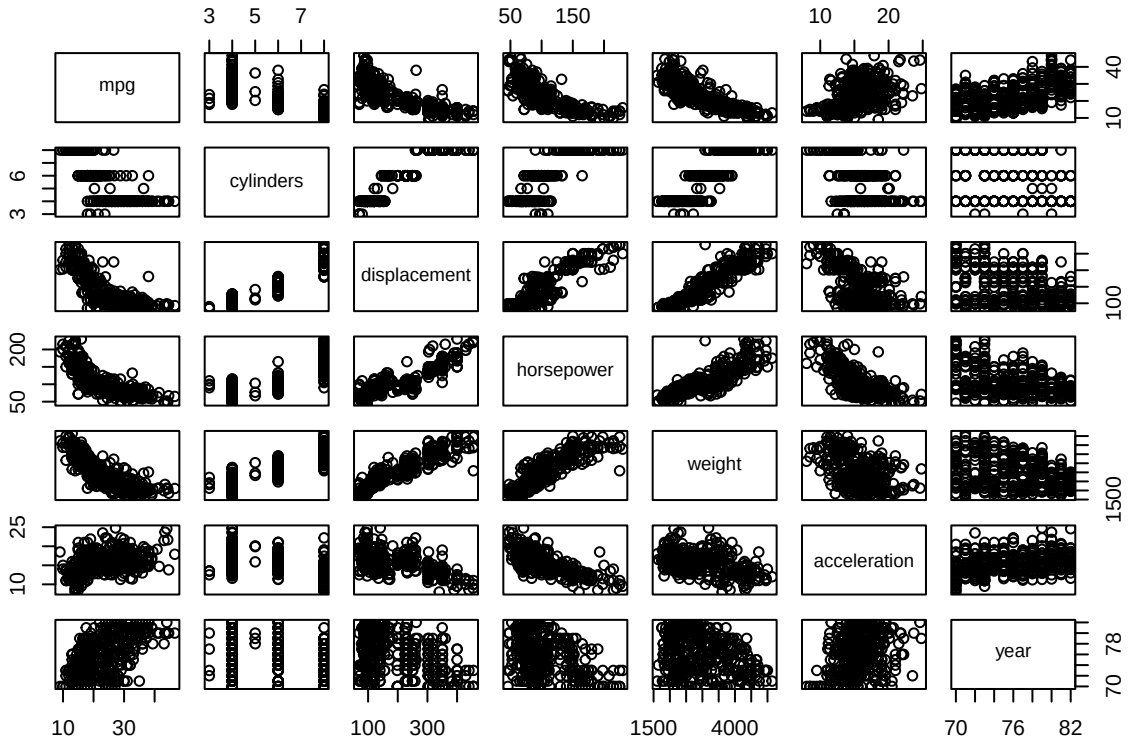
```
sapply(Auto_Update[,1:7], mean)
```

```
##          mpg    cylinders displacement   horsepower       weight acceleration
##    24.404430     5.373418   187.240506   100.721519  2935.971519    15.726899
##         year
##    77.145570
```

```r
sapply(Auto_Update[,1:7], sd)
```

```
##          mpg    cylinders displacement   horsepower       weight acceleration
##     7.867283     1.654179    99.678367    35.708853   811.300208     2.693721
##         year
##     3.106217
```

```r
## (e)

pairs(Auto[,1:7])
```




```r
# Cylinders seems to be independent of other parameters
# There are some linear relationships, mpg and disp seem negatively correlated

cor(Auto[,1:7])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
##              acceleration       year
## mpg             0.4233285  0.5805410
## cylinders      -0.5046834 -0.3456474
## displacement   -0.5438005 -0.3698552
## horsepower     -0.6891955 -0.4163615
## weight         -0.4168392 -0.3091199
## acceleration    1.0000000  0.2903161
```

```
## year            0.2903161  1.0000000
```

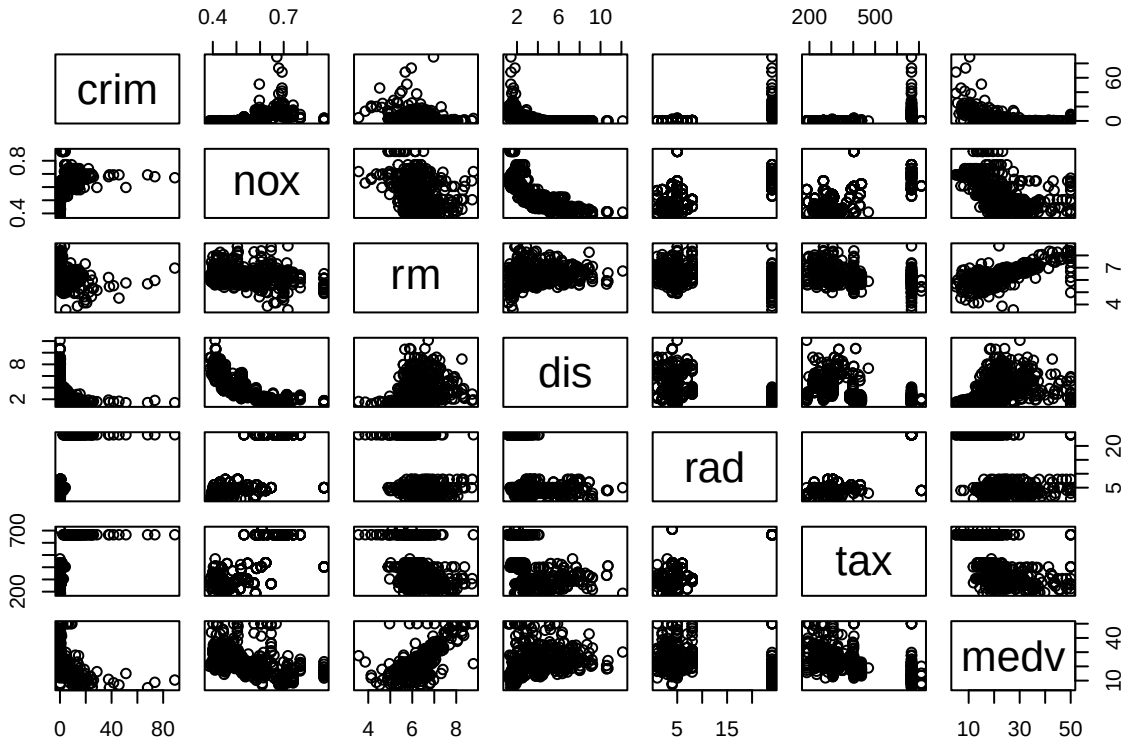
```
# High correlation bw cylinders and displacement, weight and displacement
```

```
## (f)

# Yes, can predict mpg based on other variables
# Inversely correlated: cylinders, displacement, horsepower, weight
# Can infer that as year increase i.e. newer models, mpg is higher
```

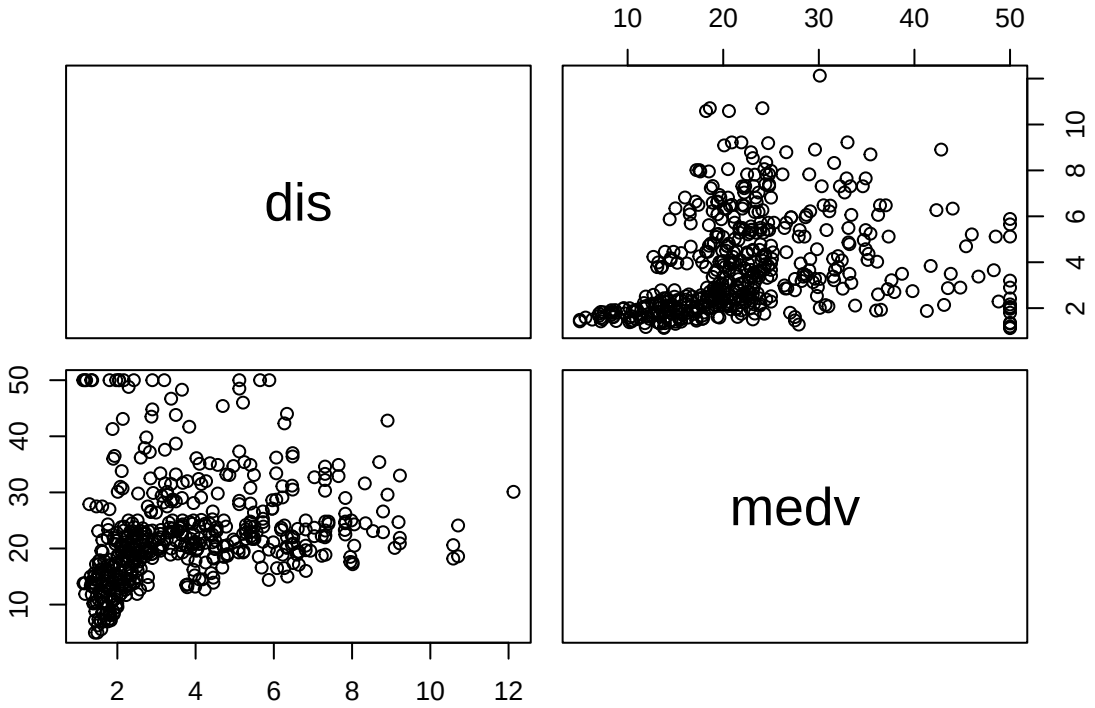
Chapter 2, Question 10

```
## (a)
library(MASS)
?Boston
# 506 rows X 14 columns - Housing Values in Suburbs of Boston
# rows - suburbs/towns
# Columns - predictors
  # crim: per capita crime rate in town
  # zn: proportion of residential land zoned for lots over 25,000 sq.ft.
  # indus: proportion of non-retail business acres per town
  # chas: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
  # nox: nitrogen oxides concentration (parts per 10 million)
  # rm: average number of rooms per dwelling
  # age: proportion of owner-occupied units built prior to 1940
  # dis: weighted mean of distances to five Boston employment centres
  # rad: index of accessibility to radial highways
  # tax: full-value property-tax rate per \$10,000
  # ptratio: pupil-teacher ratio by town
  # black: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
  # lstat: lower status of the population (percent)
  # medv: median value of owner-occupied homes in \$1000s
```

```
## (b)

# taking quantitative variables for initial exploration
pairs(~crim + nox + rm + dis + rad + tax + medv, data = Boston)
```




```
pairs(~dis + medv, data = Boston) # trial for checking some pairs individually
```




```
# Some observations from the plots:
# 1. -ve linear reln between crim and medv; crim and dis
# 2. +ve linear reln betwen dis and medv
# 3. Not very high correlation among variables (maybe)
```

```
## (c)
```

```
# checking corr of crim vs other predictors
cor(Boston[-1],Boston$crim)
```

```
##                [,1]
## zn      -0.20046922
## indus    0.40658341
## chas    -0.05589158
## nox      0.42097171
## rm      -0.21924670
## age      0.35273425
## dis     -0.37967009
## rad      0.62550515
## tax      0.58276431
## ptratio  0.28994558
## black   -0.38506394
## lstat    0.45562148
## medv    -0.38830461
```

```
# not very high magnitude of correlation
# +ve: indus, nox, age, rad, tax, ptratio, lstat (Highest: rad)
# -ve: zn, chas, rm, dis, black, medv (Lowest: medv)
```

```
## (d)

summary(Boston$crim) # Mean is greater than Median - Skewed Dist.
```

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

```
High_crime = Boston[which((Boston$crim) > mean(Boston$crim) + 2*sd(Boston$crim)),]
# 16 Obs in output i.e. outside 95% interval
# max = 88.97620 -> very far from mean = 3.61352 -> suggests very high crime rate
# wide range -> 0.006 to 88.976
High_crime_2 = Boston[which((Boston$crim) > mean(Boston$crim) + sd(Boston$crim)),]
# 43 Obs in output

summary(Boston$tax) # Mean and Median closer
```

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

```
High_tax = Boston[which((Boston$tax) > mean(Boston$tax) + 2*sd(Boston$tax)),]
# 0 Obs in output, nothing outside 95% interval
High_tax_2 = Boston[which((Boston$tax) > mean(Boston$tax) + sd(Boston$tax)),]
# 137 Obs in output
# range seems fine -> 187 to 711 with 330 Median and 408.2 Mean

summary(Boston$ptratio) # Mean and Median closer
```

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

```
pup_tea_r = Boston[which((Boston$ptratio) > mean(Boston$ptratio) + 2*sd(Boston$ptratio)),]
# 0 Obs in output
pup_tea_r_2 = Boston[which((Boston$ptratio) > mean(Boston$ptratio) + sd(Boston$ptratio)),]
# 56 Obs in output
```

```
# range is narrower -> 12.6 to 22 with 19.05 Median and 18.46 Mean
```

```
## (e)
sum(Boston$chas==1) # 35 suburbs are bounded by Charles river
```

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

```
## (f)
summary(Boston$ptratio) # median 19.05
```

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

```
## (g)
which(Boston$medv == min(Boston$medv)) # two suburbs - 399 & 406 has lowest medv
```

```
## [1] 399 406
```

```
Boston[399,] # compairing with summary(Boston)
```

```
##        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
```

```
  # crim 38.3518 - VERY HIGH compared to mean
  # zn 0
  # indus 18.1
  # chas 0
  # nox 0.693
  # rm 5.453
  # age 100
  # dis 1.4896
  # rad 24
  # tax 666
  # ptratio 20.2 - Closer to the MAX value in the dataset
  # black 396.9
  # lstat 30.59 - Closer to the MAX value in the dataset
  # medv 5

Boston[406,] # compairing with summary(Boston)
```

```
##        crim zn indus chas   nox    rm age    dis rad tax ptratio  black lstat
## 406 67.9208  0  18.1    0 0.693 5.683 100 1.4254  24 666    20.2 384.97 22.98
##     medv
## 406    5
```

```
  # crim 67.9208 - VERY HIGH compared to mean
  # zn 0
  # indus 18.1
  # chas 0
  # nox 0.693
  # rm 5.683
  # age 100
  # dis 1.4254
  # rad 24
  # tax 666
```

```
  # ptratio 20.2 - Closer to the MAX value in the dataset
  # black 384.97
  # lstat 22.98
  # medv 5

# a lot of predictors for 399, 406 match or are quite close
```

```
## (h)

sum(Boston$rm > 7) # 64
```

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

```
sum(Boston$rm > 8) # 13
```

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

```
summary(subset(Boston, rm > 8))
```

```
##       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
```

```
# for suburbs that average more than 8 rooms per dwelling:
# crim mean and max is very small as compared to total set
# lstat is also relatively lower
# medv is higher, min value and mean is higher than total set
```

Chapter 3, Question 8

```
## (a)

library(ISLR)
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##
##   acceleration        year           origin                      name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   amc matador       :  5
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   ford pinto        :  5
##  Median :15.50   Median :76.00   Median :1.000   toyota corolla    :  5
##  Mean   :15.54   Mean   :75.98   Mean   :1.577   amc gremlin       :  4
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000   amc hornet        :  4
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   chevrolet chevette:  4
##                                                  (Other)           :365
```

```
lin = lm(mpg ~ horsepower, Auto)
summary(lin)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

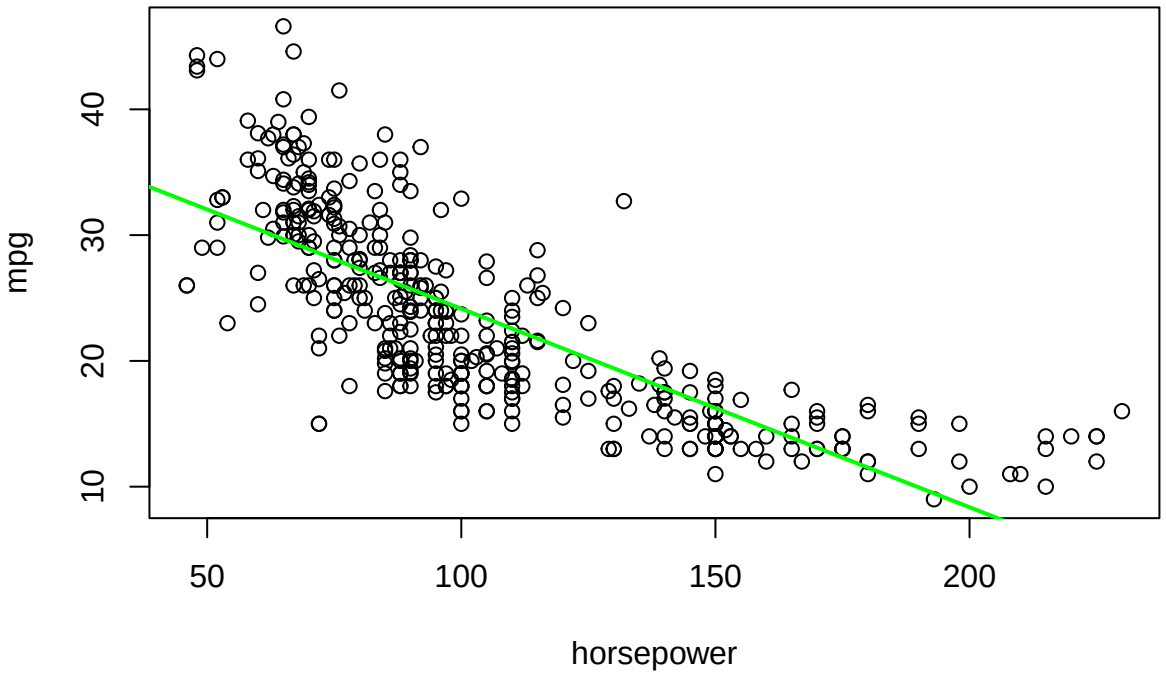
```
# p-value is close to zero -> strong relationship
# There's a negative relationship, negative slope, -0.15

predict(lin, data.frame(horsepower = 98, interval = "confidence"))
```

```
##        1
## 24.46708
```

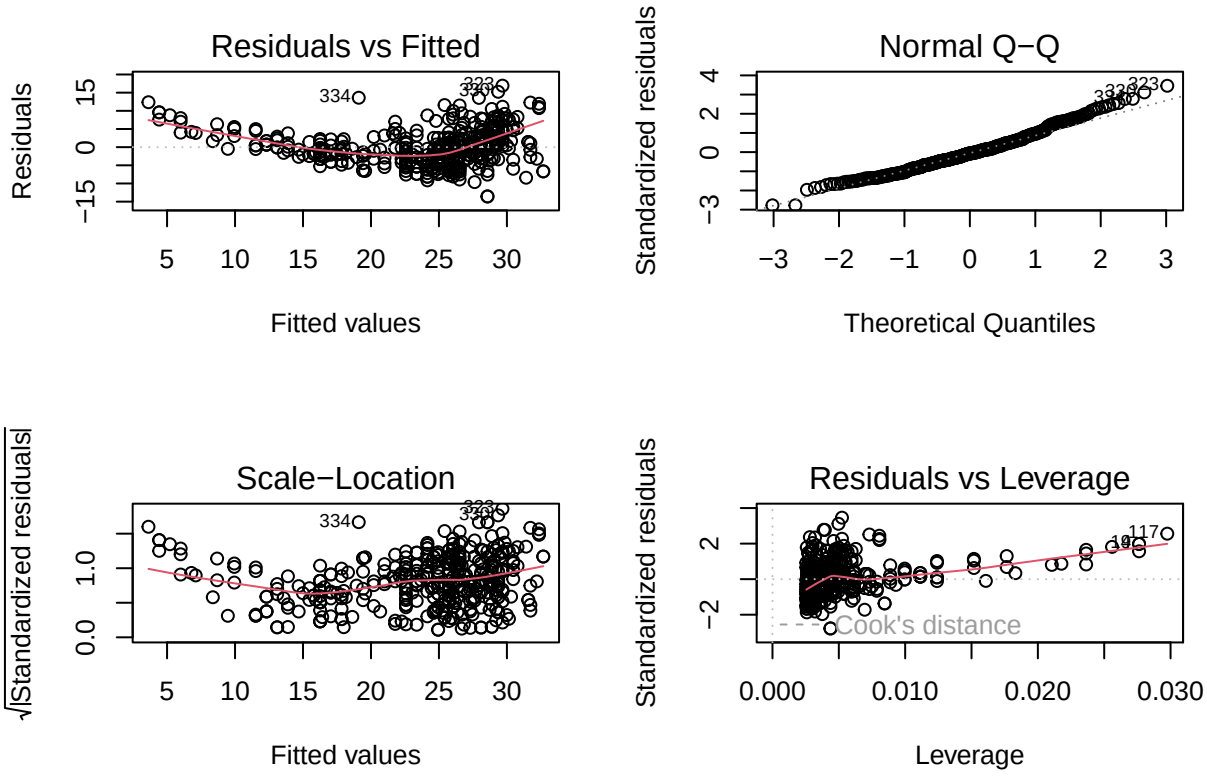
```
predict(lin, data.frame(horsepower = 98, interval = "prediction"))
```

```
##        1
## 24.46708
```

```
## (b)

plot(mpg ~ horsepower, data = Auto)
abline(lin, lwd = 2, col = "green")
```

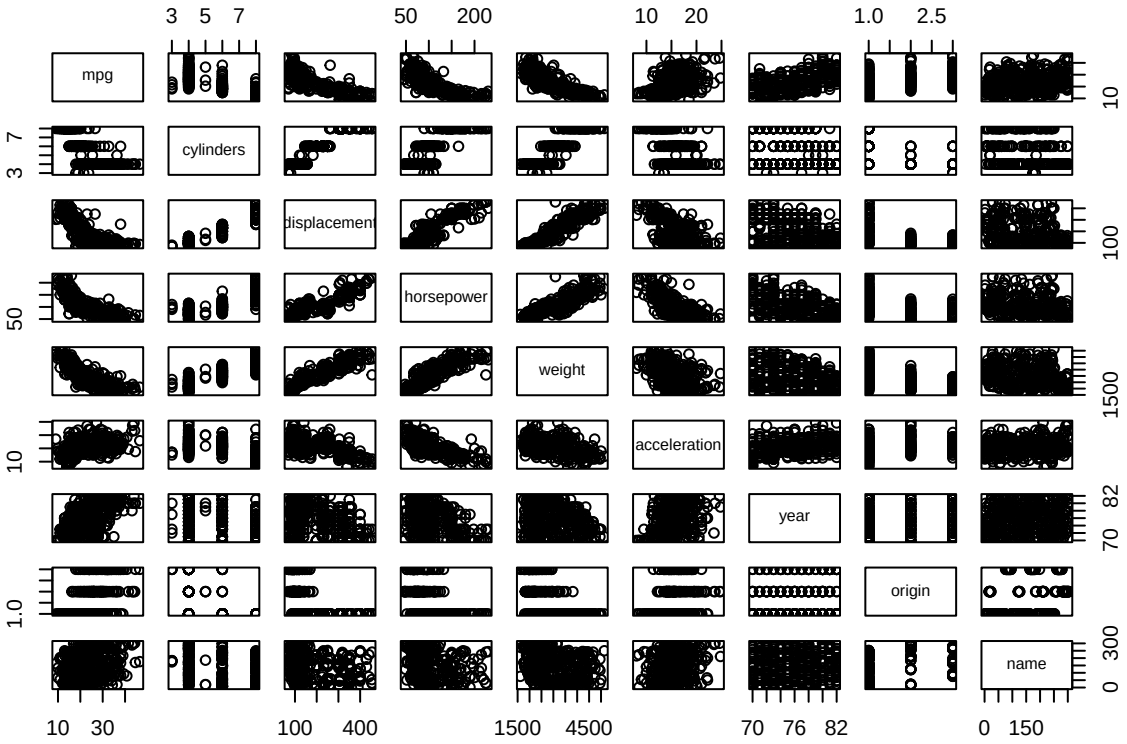



```
## (c)

par(mfrow = c(2,2))
plot(lin)
```

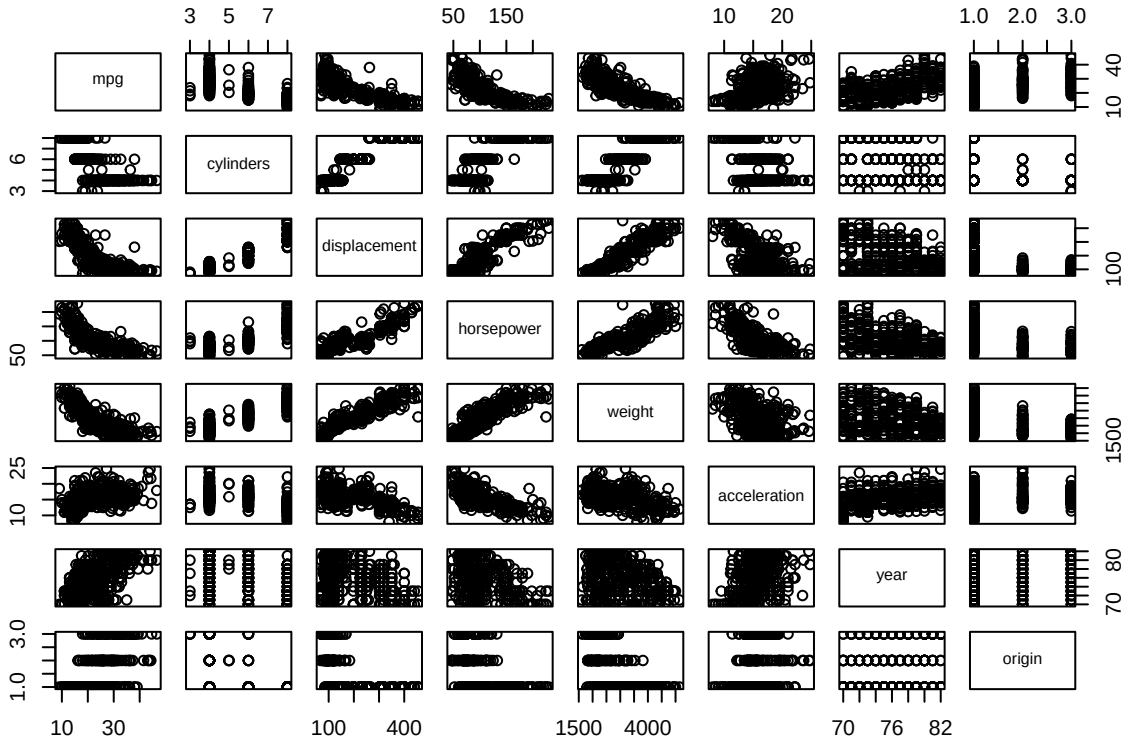


Chapter 3, Question 9

```
## (a)

pairs(Auto[,1:9])
```



```r
# but 9th is qualitative - "name"
pairs(Auto[,1:8])
```




```r
## (b)

cor(Auto[,1:8])
```

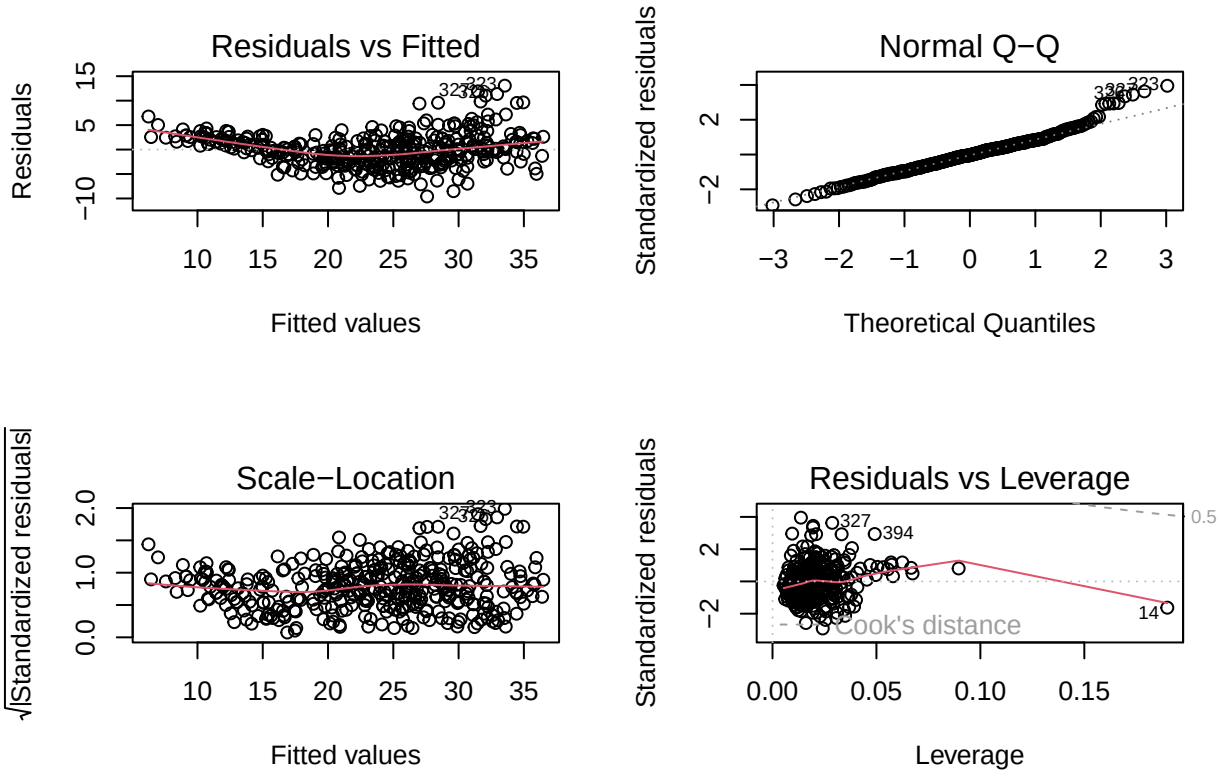
```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

```r
## (c)

reg = lm(mpg~.-name, data = Auto)
summary(reg)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

```
# p-value is quite small for the F-statistic obtained
# reject null hypothese - indicates strong relationship

# Predictors with statistically significant response:
# Looking at p-values - Displacement, Weight, Year and Origin

# Coeff of the year variable = 0.750773
# Implication: Keeping other variables constant, 1 unit increase in year,
# increases mpg by 0.75
```

```
## (d)

par(mfrow = c(2,2))
plot(reg)
```



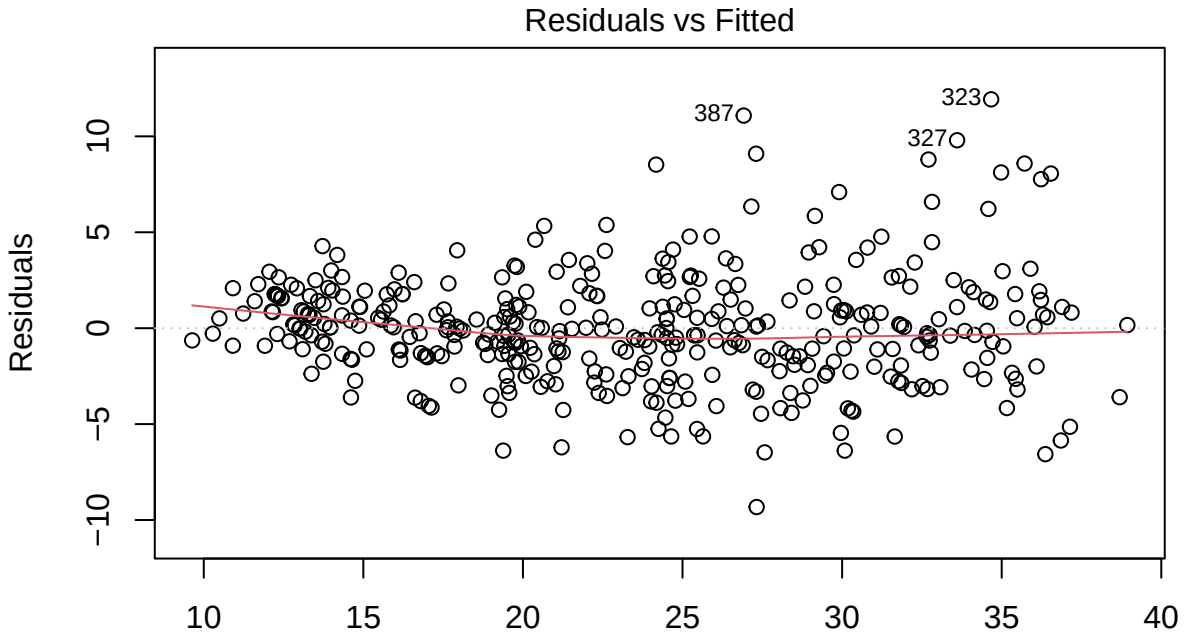

```
# Residual Vs Fitted - Non-Linear - Polynomial Reg could work better
```

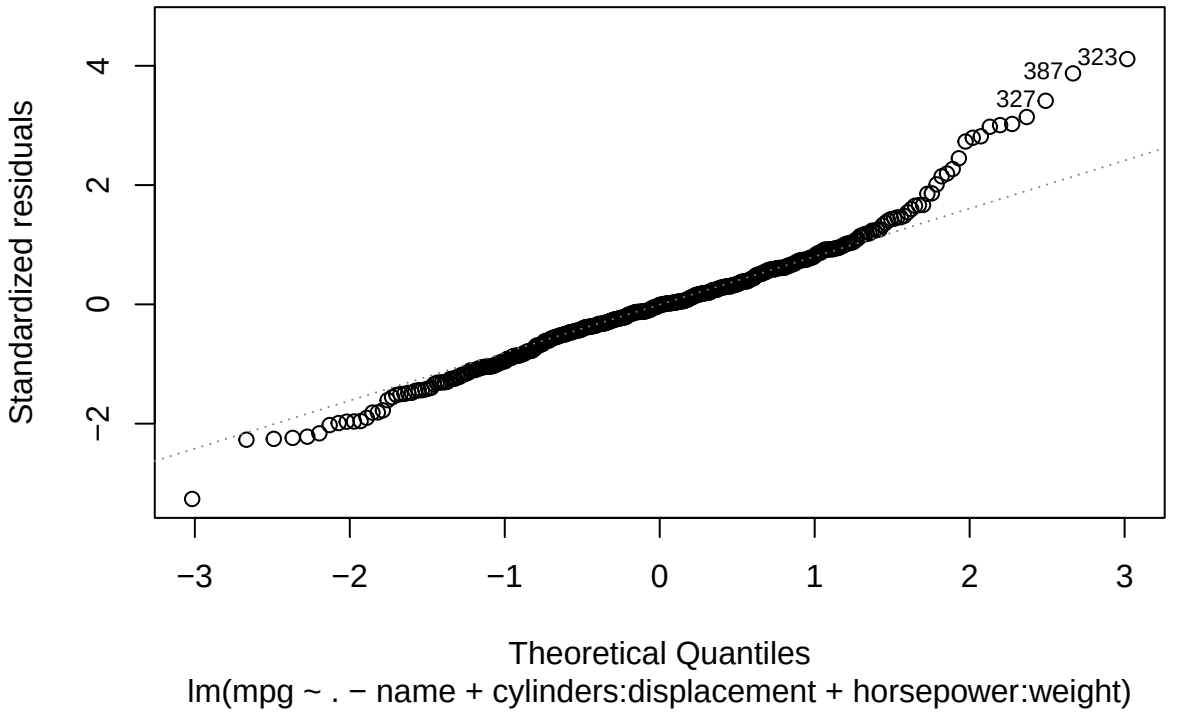
```
## (e)

interact = lm(mpg~.-name + cylinders:displacement + horsepower:weight, data = Auto)
summary(interact)
```

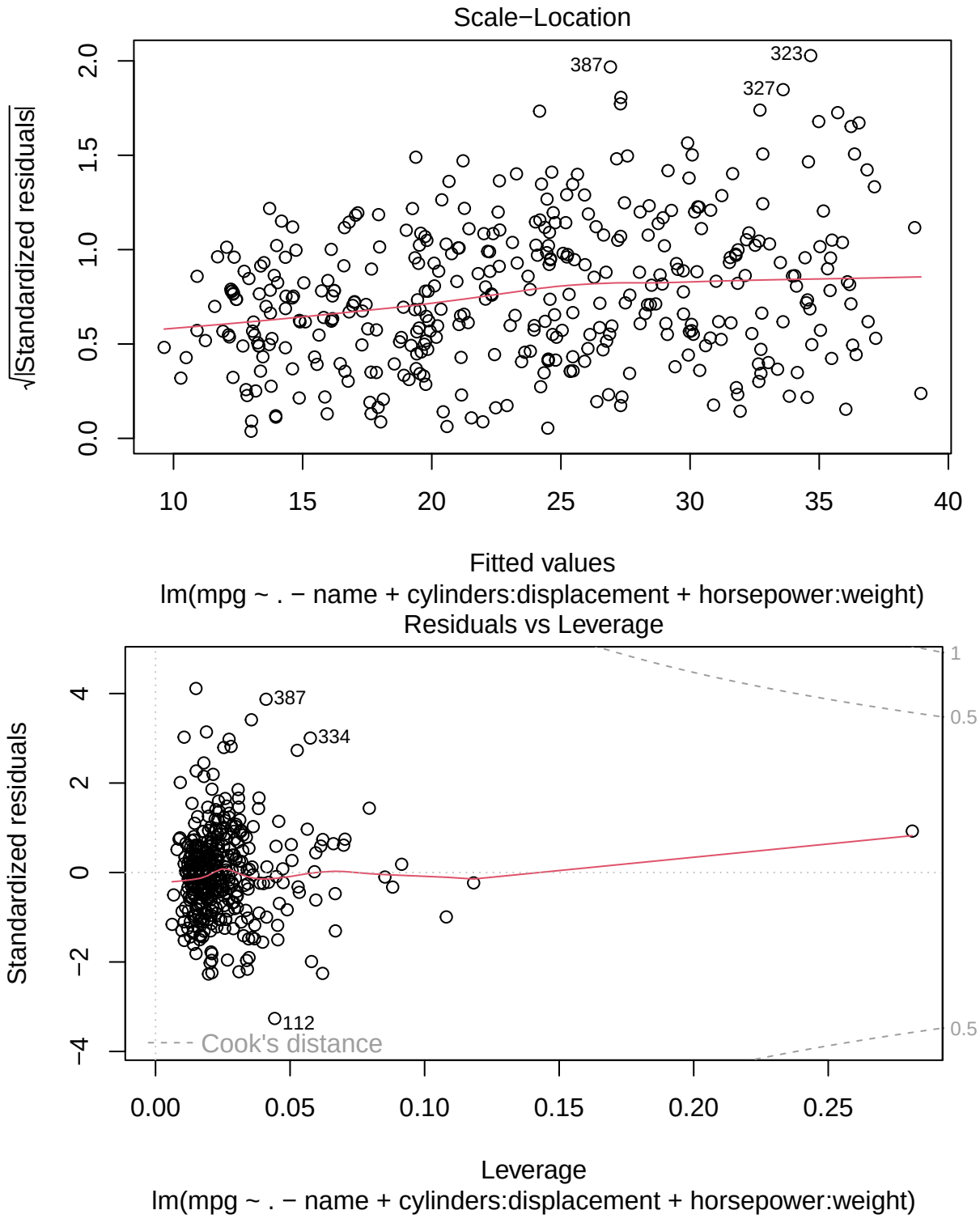
```
##
## Call:
## lm(formula = mpg ~ . - name + cylinders:displacement + horsepower:weight,
##     data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3228 -1.5862 -0.0291  1.5536 11.9296
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.053e+00  4.544e+00   0.892  0.37297
## cylinders              -7.357e-01  4.834e-01  -1.522  0.12883
## displacement           -2.010e-02  1.584e-02  -1.269  0.20531
## horsepower             -2.080e-01  2.683e-02  -7.754 8.17e-14 ***
## weight                 -1.014e-02  9.351e-04 -10.849  < 2e-16 ***
## acceleration           -7.057e-02  8.895e-02  -0.793  0.42805
## year                    7.692e-01  4.480e-02  17.168  < 2e-16 ***
## origin                  7.159e-01  2.589e-01   2.765  0.00597 **
## cylinders:displacement  3.932e-03  2.165e-03   1.816  0.07008 .
## horsepower:weight       4.699e-05  6.930e-06   6.781 4.55e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.923 on 382 degrees of freedom
## Multiple R-squared:  0.863,  Adjusted R-squared:  0.8598
## F-statistic: 267.4 on 9 and 382 DF,  p-value: < 2.2e-16
```

```
plot(interact)
```








```
# Observations:
# No significant change in p-value as compared with original model
# Multiple Rsq increased from 0.8215 to 0.863
# Adj Rsq increased from 0.8182 to 0.8598
# RSE decreased from 3.328 to 2.923
```

```r
# Conclusion: the combination used seems to be statistically significant
```

```r
## (f)

random_combination = lm(mpg~.-name + year:cylinders
                        + I(horsepower^2) + I(acceleration^2), data = Auto)

summary(random_combination)
```

```
##
## Call:
## lm(formula = mpg ~ . - name + year:cylinders + I(horsepower^2) +
##     I(acceleration^2), data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9986 -1.5525 -0.1194  1.4348 11.7722
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -3.394e+01  1.430e+01  -2.374 0.018075 *
## cylinders          8.481e+00  2.340e+00   3.624 0.000329 ***
## displacement      -1.106e-02  7.330e-03  -1.509 0.132051
## horsepower        -2.720e-01  3.531e-02  -7.703 1.16e-13 ***
## weight            -3.338e-03  6.812e-04  -4.900 1.42e-06 ***
## acceleration      -1.378e+00  5.421e-01  -2.542 0.011403 *
## year               1.272e+00  1.594e-01   7.982 1.71e-14 ***
## origin             1.027e+00  2.493e-01   4.121 4.63e-05 ***
## I(horsepower^2)    8.040e-04  1.140e-04   7.054 8.22e-12 ***
## I(acceleration^2)  3.351e-02  1.578e-02   2.124 0.034303 *
## cylinders:year    -1.056e-01  3.023e-02  -3.493 0.000533 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.935 on 381 degrees of freedom
## Multiple R-squared:  0.8622, Adjusted R-squared:  0.8586
## F-statistic: 238.3 on 10 and 381 DF,  p-value: < 2.2e-16
```

Chapter 3, Question 10

```r
## (a)

library(ISLR)
summary(Carseats)
```

```
##      Sales          CompPrice       Income        Advertising
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc        Age          Education
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0
```

```
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0
##  Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0
##  Mean   :264.8   Mean   :115.8                Mean   :53.32   Mean   :13.9
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00   3rd Qu.:16.0
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.0
##  Urban       US
##  No :118   No :142
##  Yes:282   Yes:258
##
##
##
##
```

```
predict_sales = lm(Sales~Price+Urban+US, data = Carseats)
summary(predict_sales)
```

```
##
## Call:
## lm(formula = Sales ~ Price + Urban + US, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9206 -1.6220 -0.0564  1.5786  7.0581
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
## Price       -0.054459   0.005242 -10.389  < 2e-16 ***
## UrbanYes    -0.021916   0.271650  -0.081    0.936
## USYes        1.200573   0.259042   4.635 4.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.472 on 396 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2335
## F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

```
## (b)

# Qual Predictors: US, Urban
# Quant Predictor: Price

# Interpreration of Model Coeff
# -ve coeff -> Price, UrbanYes; +ve coeff -> USYes
# For every unit increase in price, sales will fall by 54 (0.054*1000)

# Residual Error = 2.472 (Low)
# Multiple RSq = 0.2393 (low)
# For FStat 41.52 -> pvalue close to zero (good fit)

# UrbanYes -> not statistically significant
#(coeff = -0.021916, t value = -0.081)

## NEED TO CHECK ##
```

```
# USYes -> coeff = 1.2 -> If shop in US, there's an average increase in
# car seat sales of 1200 units
```

```
## (c)

attach(Carseats)
contrasts(Urban)
```

```
##     Yes
## No    0
## Yes   1
```

```
contrasts(US)
```

```
##     Yes
## No    0
## Yes   1
```

```
# y = b0 + b1x1 + b2x2 + b3x3
# Sales = 13.043469 + (-0.054459)*Price + (-0.021916)*Urban(Yes:1, No:0) +
# (1.200573)*US(Yes:1, No:0)
```

```
## (d)

all_vars = lm(Sales~., data = Carseats)
summary(all_vars)
```

```
##
## Call:
## lm(formula = Sales ~ ., data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8692 -0.6908  0.0211  0.6636  3.4115
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.6606231  0.6034487   9.380  < 2e-16 ***
## CompPrice        0.0928153  0.0041477  22.378  < 2e-16 ***
## Income           0.0158028  0.0018451   8.565 2.58e-16 ***
## Advertising      0.1230951  0.0111237  11.066  < 2e-16 ***
## Population       0.0002079  0.0003705   0.561    0.575
## Price           -0.0953579  0.0026711 -35.700  < 2e-16 ***
## ShelveLocGood    4.8501827  0.1531100  31.678  < 2e-16 ***
## ShelveLocMedium  1.9567148  0.1261056  15.516  < 2e-16 ***
## Age             -0.0460452  0.0031817 -14.472  < 2e-16 ***
## Education       -0.0211018  0.0197205  -1.070    0.285
## UrbanYes         0.1228864  0.1129761   1.088    0.277
## USYes           -0.1840928  0.1498423  -1.229    0.220
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 388 degrees of freedom
## Multiple R-squared:  0.8734, Adjusted R-squared:  0.8698
## F-statistic: 243.4 on 11 and 388 DF,  p-value: < 2.2e-16
```

```
# Null Hyp: No relation of any predictor with the result

# Looking at tstat vs Std Error

# CompPrice - low coeff but far from zero since tvalue is 22.378
# Income - low coeff but far from zero since tvalue is 8.565
# Advertising - low coeff but far from zero since tvalue is 11.066
# Population - very low coeff and also low tvalue = 0.561
# Price - low negative coeff, high negative tvaluye = -35.700
# ShelveLocGood - high coeff, high tvalue = 31.678
# ShelveLocMedium - high coeff, high tvalue = 15.516
# Age - low coeff, high negative tvalue = -14.42
# Education - low coeff, low negative tvalue = -1.070
# UrbanYes - low coeff, low tvalue = 1.088
# USYes - low coeff, low negative tvalue = -1.229

# No reln: Population, Education, UrbanYes, USYes
# Reln: Reject Null Hyp: CompPrice, Income, Advertising,
#                        Price, ShelveLocGood, ShelveLocMedium, Age
```

```
## (e)

new_model = lm(Sales~CompPrice + Income + Advertising + Price + ShelveLoc + Age, data = Carseats)
summary(new_model)
```

```
##
## Call:
## lm(formula = Sales ~ CompPrice + Income + Advertising + Price +
##     ShelveLoc + Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7728 -0.6954  0.0282  0.6732  3.3292
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.475226   0.505005   10.84   <2e-16 ***
## CompPrice         0.092571   0.004123   22.45   <2e-16 ***
## Income            0.015785   0.001838    8.59   <2e-16 ***
## Advertising       0.115903   0.007724   15.01   <2e-16 ***
## Price            -0.095319   0.002670  -35.70   <2e-16 ***
## ShelveLocGood     4.835675   0.152499   31.71   <2e-16 ***
## ShelveLocMedium   1.951993   0.125375   15.57   <2e-16 ***
## Age              -0.046128   0.003177  -14.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 392 degrees of freedom
## Multiple R-squared:  0.872,  Adjusted R-squared:  0.8697
## F-statistic: 381.4 on 7 and 392 DF,  p-value: < 2.2e-16
```

```
## (f)

summary(predict_sales)
```

```
##
## Call:
## lm(formula = Sales ~ Price + Urban + US, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9206 -1.6220 -0.0564  1.5786  7.0581
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
## Price       -0.054459   0.005242 -10.389  < 2e-16 ***
## UrbanYes    -0.021916   0.271650  -0.081    0.936
## USYes        1.200573   0.259042   4.635 4.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.472 on 396 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2335
## F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

```
summary(new_model)
```

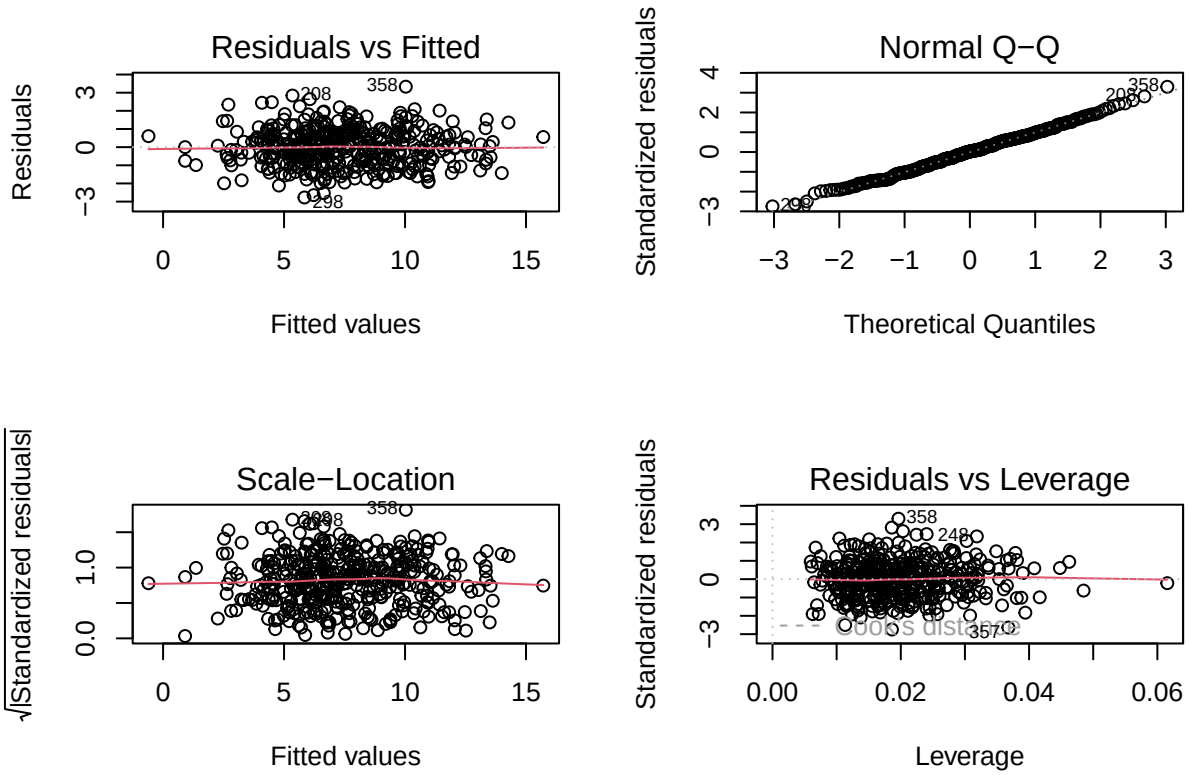
```
##
## Call:
## lm(formula = Sales ~ CompPrice + Income + Advertising + Price +
##     ShelveLoc + Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7728 -0.6954  0.0282  0.6732  3.3292
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.475226   0.505005   10.84   <2e-16 ***
## CompPrice        0.092571   0.004123   22.45   <2e-16 ***
## Income           0.015785   0.001838    8.59   <2e-16 ***
## Advertising      0.115903   0.007724   15.01   <2e-16 ***
## Price           -0.095319   0.002670  -35.70   <2e-16 ***
## ShelveLocGood    4.835675   0.152499   31.71   <2e-16 ***
## ShelveLocMedium  1.951993   0.125375   15.57   <2e-16 ***
## Age             -0.046128   0.003177  -14.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.019 on 392 degrees of freedom
## Multiple R-squared:  0.872,  Adjusted R-squared:  0.8697
## F-statistic: 381.4 on 7 and 392 DF,  p-value: < 2.2e-16
```

```
# Model in (a) Vs Model in (e) - the second model has:
  # lower range of residuals, lower standard errors, higher t-values
  # Lower RSE, Higher Rsq and Adj Rsq, much higher F Statistic, not much
    # difference in p-value

# Thus, model in (e) is much better than model in (a)
```

```
## (g)

confint(new_model)
```

```
##                       2.5 %      97.5 %
## (Intercept)      4.48236820  6.46808427
## CompPrice        0.08446498  0.10067795
## Income           0.01217210  0.01939784
## Advertising      0.10071856  0.13108825
## Price           -0.10056844 -0.09006946
## ShelveLocGood    4.53585700  5.13549250
## ShelveLocMedium  1.70550103  2.19848429
## Age             -0.05237301 -0.03988204
```

```
## (h)

par(mfrow=c(2,2))
plot(new_model)
```




```
# Fron Residuals vs Fitted chart - evident that good fit
# But there are some outliers - obs 358 (more than 3 residuals away)
```

Chapter 3, Question 11

```
set.seed(1)
x = rnorm(100)
y = 2*x + rnorm(100)

## (a)
```

```r
no_int_reg = lm(y~x+0)
summary(no_int_reg)
```

```
##
## Call:
## lm(formula = y ~ x + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9154 -0.6472 -0.1771  0.5056  2.3109
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## x   1.9939     0.1065   18.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9586 on 99 degrees of freedom
## Multiple R-squared:  0.7798, Adjusted R-squared:  0.7776
## F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

```r
## (b)

no_int_reg2 = lm(x~y+0)
summary(no_int_reg2)
```

```
##
## Call:
## lm(formula = x ~ y + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8699 -0.2368  0.1030  0.2858  0.8938
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## y  0.39111    0.02089   18.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4246 on 99 degrees of freedom
## Multiple R-squared:  0.7798, Adjusted R-squared:  0.7776
## F-statistic: 350.7 on 1 and 99 DF,  p-value: < 2.2e-16
```

```r
## (c)

# Observe same t-statistic and p-value for both
# Reg line is same with inverted axis
```

```r
## (d)

numer = (sqrt(length(x)-1))*sum(x*y)
denom = sqrt(sum(x^2)*sum(y^2) - sum(x*y)^2)

t_stat = numer/denom
```

```r
t_stat # Matches
```

```
## [1] 18.72593
```

```r
## (e)

# In (d) replacing y with x will give same equation, hence same t_stat
```

```r
## (f)

with_int_reg = lm(y~x)
summary(with_int_reg)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8768 -0.6138 -0.1395  0.5394  2.3462
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.03769    0.09699  -0.389    0.698
## x            1.99894    0.10773  18.556   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9628 on 98 degrees of freedom
## Multiple R-squared:  0.7784, Adjusted R-squared:  0.7762
## F-statistic: 344.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
# t statistic obtained is almost equal to that obtained with no intercept
# regression
```

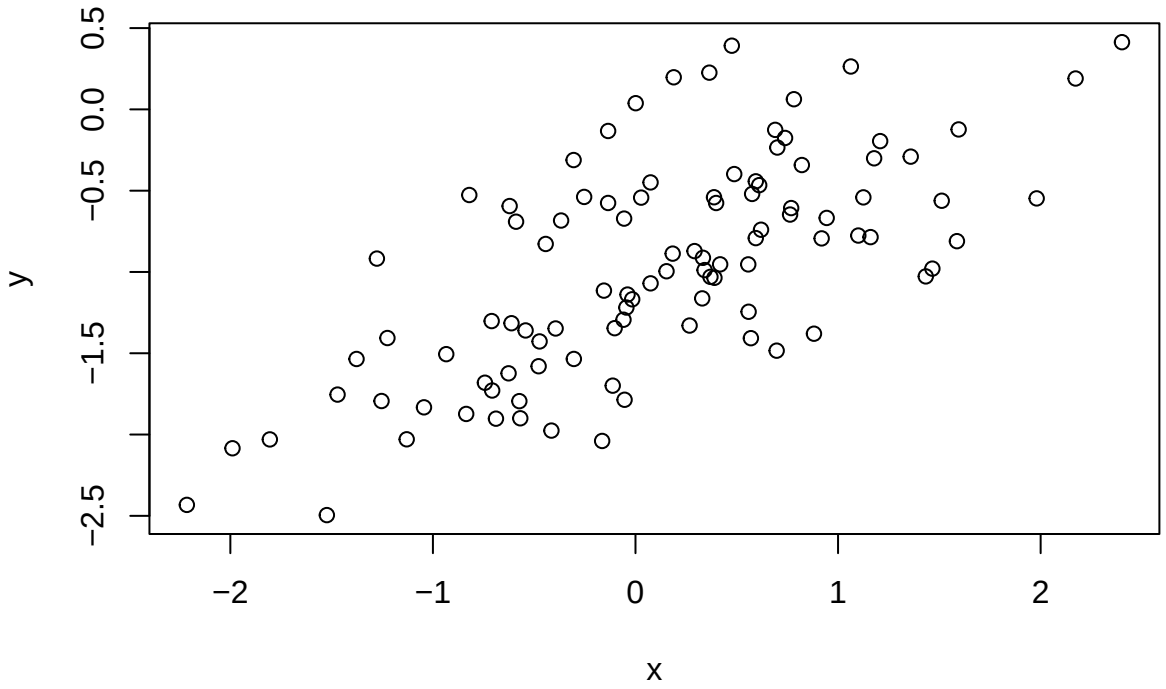
Chapter 3, Question 13

```r
set.seed(1)

## (a)

x = rnorm(100, mean = 0, sd = 1)
```

```r
## (b)

eps = rnorm(100, mean = 0, sd = 0.5) # variance = 0.25
```

```r
## (c)

y = -1 + 0.5*x + eps

# length of y = 100, Bo = -1, B1 = 0.5
```

```r
## (d)

plot(y~x)
```




```
# y and x seem to have a positive linear relationship
```

```
## (e)

fitline = lm(y~x)
summary(fitline)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

```
# B0_hat = -0.98, B1_hat = 0.47326
# values quite close to B0 and B1 respectively
```

```
## (f)

# abline(fitline, col = "blue")
```

```r
# abline(a = -1, b = 0.5, col = "red")

# regression line and population line are very close to each other
# p-val near zero and F-stat is large -> null hyp can be rejected
```

```r
## (g)

poly = lm(y~x + I(x^2))
summary(poly)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.98252 -0.31270 -0.06441  0.29014  1.13500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.97164    0.05883 -16.517  < 2e-16 ***
## x            0.50858    0.05399   9.420  2.4e-15 ***
## I(x^2)      -0.05946    0.04238  -1.403    0.164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.479 on 97 degrees of freedom
## Multiple R-squared:  0.4779, Adjusted R-squared:  0.4672
## F-statistic:  44.4 on 2 and 97 DF,  p-value: 2.038e-14
```

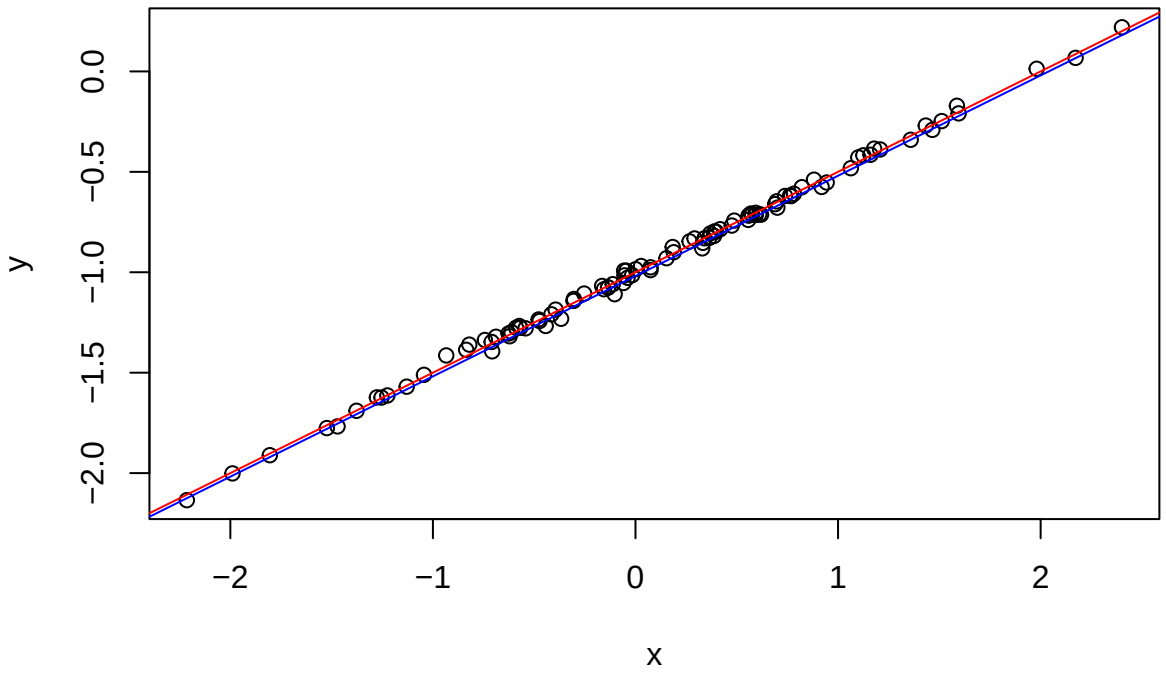
```r
# RSE for both cases is almost equal -> quadratic term did not improve the fit
```

```r
## (h)

eps_low = rnorm(100, mean = 0, sd = 0.02)
y = -1 + 0.5*x + eps_low

plot(y~x)

fitline2 = lm(y~x)
summary(fitline)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

```
abline(fitline, col = "blue")
abline(a = -1, b = 0.5, col = "red")
```



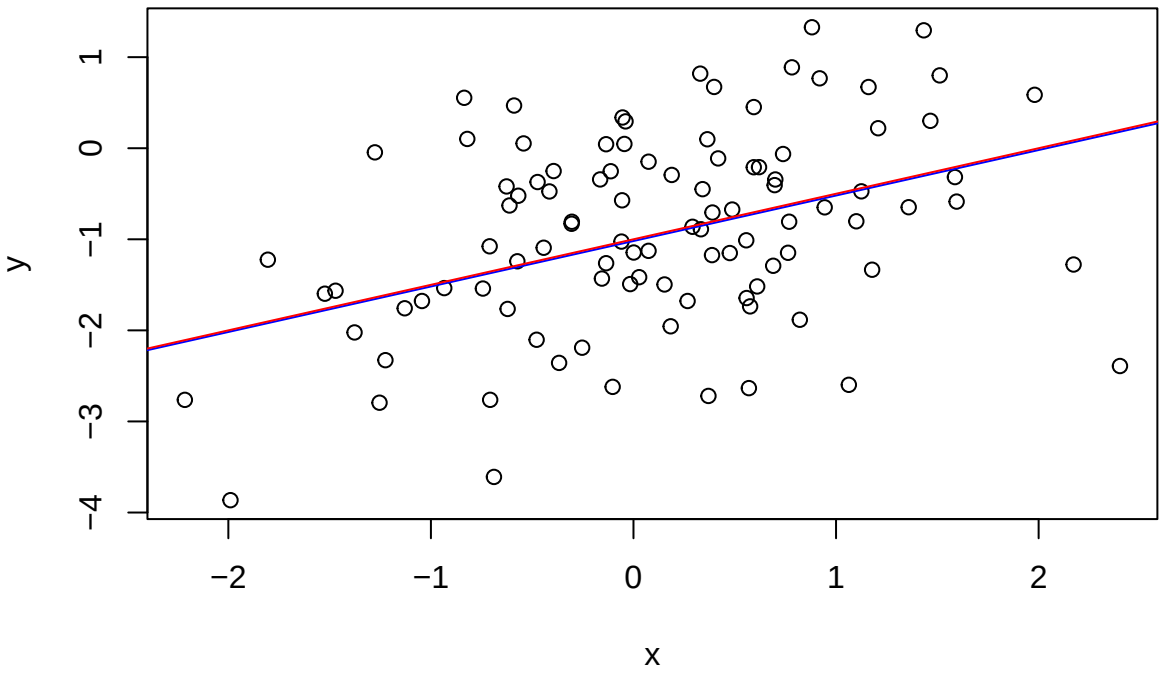

```
# RSE is very low in this case as noise is very low
```

```
## (i)

eps_high = rnorm(100, mean = 0, sd = 1)
y = -1 + 0.5*x + eps_high

plot(y~x)

fitline3 = lm(y~x)
summary(fitline)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93842 -0.30688 -0.06975  0.26970  1.17309
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.01885    0.04849 -21.010  < 2e-16 ***
## x            0.49947    0.05386   9.273 4.58e-15 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4814 on 98 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4619
## F-statistic: 85.99 on 1 and 98 DF,  p-value: 4.583e-15
```

```
abline(fitline, col = "blue")
abline(a = -1, b = 0.5, col = "red")
```




```
## (j)

confint(fitline)
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.1150804 -0.9226122
## x            0.3925794  0.6063602
```

```
confint(fitline2)
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.0036083 -0.9952970
## x            0.4958075  0.5050391
```

```
confint(fitline3)
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.1413399 -0.7433293
## x            0.2232721  0.6653558
```

Chapter 3, Question 14

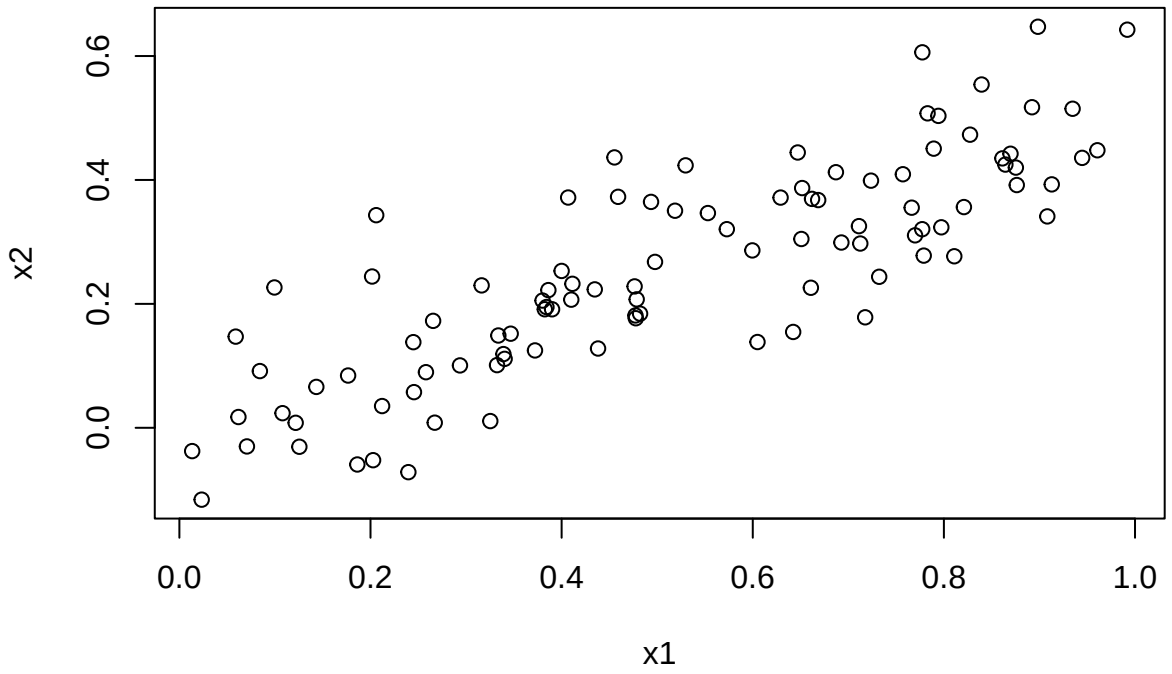
```
set.seed(1)
x1 = runif(100)
x2 = 0.5*x1+rnorm(100)/10
y = 2 + 2*x1 + 0.3*x2 + rnorm(100)
```

```r
## (a)

# B0 = 2; B1 = 2, B2 = 0.3
```

```r
## (b)

cor(x1,x2) # 0.835
```

```
## [1] 0.8351212
```

```r
plot(x2~x1)
```




```r
## (c)

modeling = lm(y~x1+x2)
summary(modeling)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8311 -0.7273 -0.0537  0.6338  2.3359
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1305     0.2319   9.188 7.61e-15 ***
## x1            1.4396     0.7212   1.996   0.0487 *
## x2            1.0097     1.1337   0.891   0.3754
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.056 on 97 degrees of freedom
## Multiple R-squared:  0.2088, Adjusted R-squared:  0.1925
## F-statistic:  12.8 on 2 and 97 DF,  p-value: 1.164e-05
```

```
# B0_hat = 2.1305, B1_hat = 1.4396, B2_hat = 1.0097
# RSE is high, Error increases from x1 to x2

# p value is less than 0.05 for x1, can reject H0 (null hypothesis)
# p value is much more than 0.05 for x2, cannot reject H0
```

```
## (d)

model_x1 = lm(y~x1)
summary(model_x1)
```

```
##
## Call:
## lm(formula = y ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89495 -0.66874 -0.07785  0.59221  2.45560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.1124     0.2307   9.155 8.27e-15 ***
## x1            1.9759     0.3963   4.986 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.055 on 98 degrees of freedom
## Multiple R-squared:  0.2024, Adjusted R-squared:  0.1942
## F-statistic: 24.86 on 1 and 98 DF,  p-value: 2.661e-06
```

```
# RSE and R2 are similar to when using x1+x2, F-statistic is slightly greater
# pval near zero for x1, can reject null hyp
```

```
## (e)

model_x2 = lm(y~x2)
summary(model_x2)
```

```
##
## Call:
## lm(formula = y ~ x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62687 -0.75156 -0.03598  0.72383  2.44890
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.3899     0.1949   12.26  < 2e-16 ***
## x2            2.8996     0.6330    4.58 1.37e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.072 on 98 degrees of freedom
## Multiple R-squared:  0.1763, Adjusted R-squared:  0.1679
## F-statistic: 20.98 on 1 and 98 DF,  p-value: 1.366e-05
```

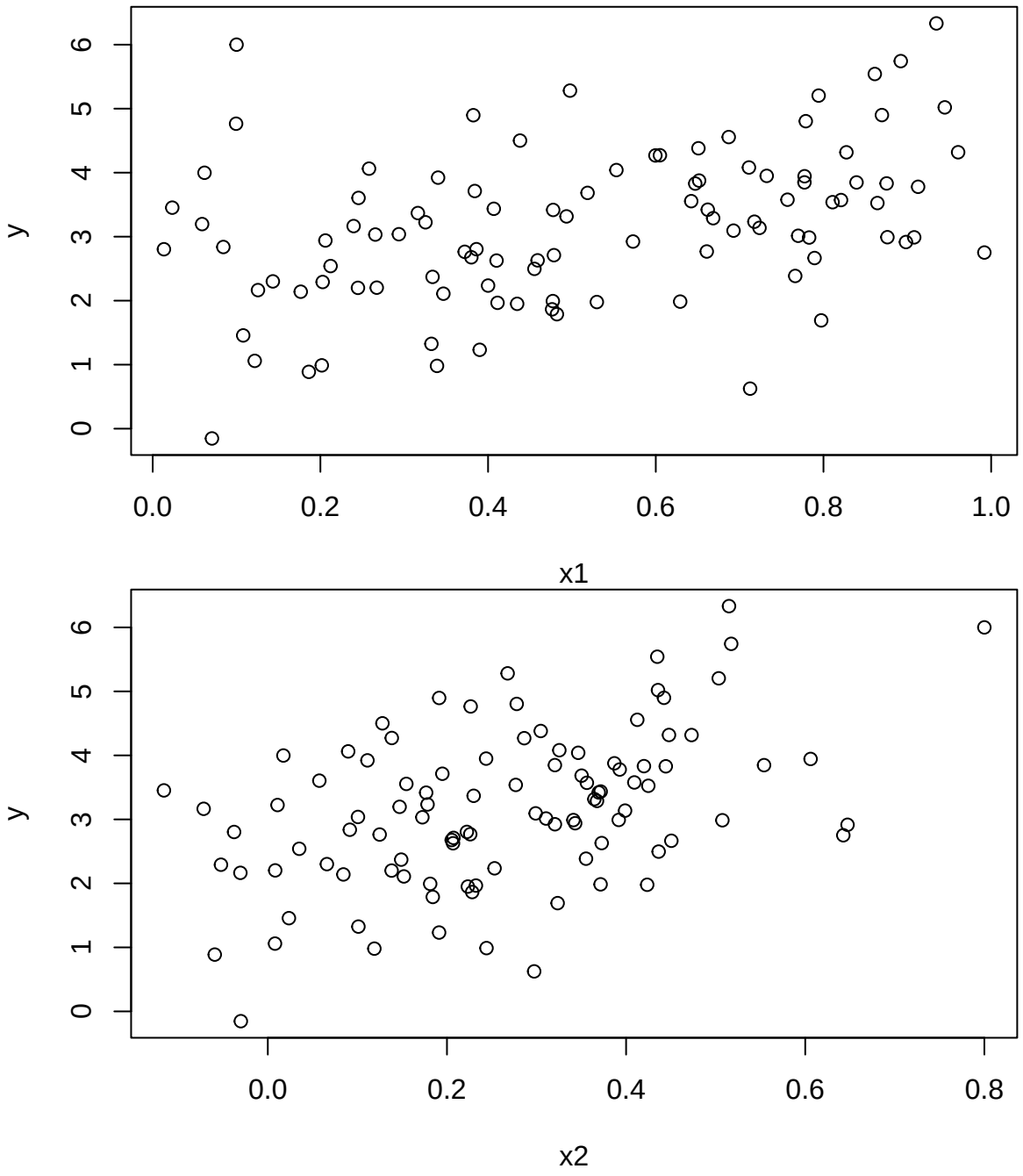
```
# pval near zero for x2, can reject null hyp

## (f)
```

```
# explained using collinearity

## (g)
```

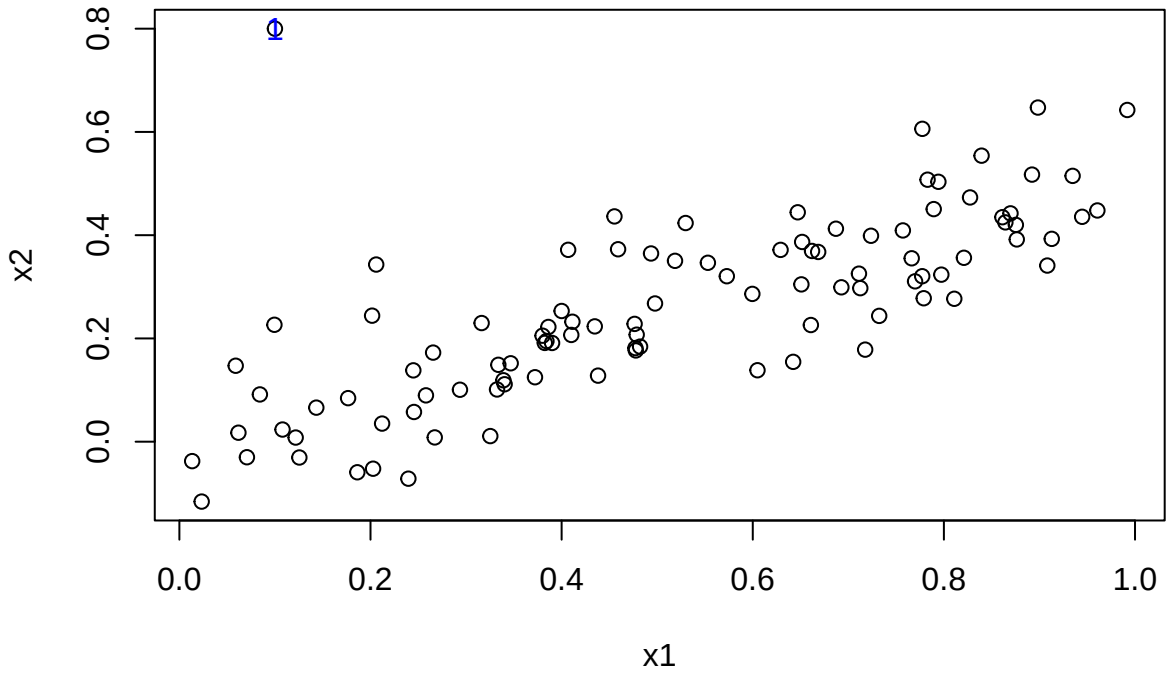
```
x1 = c(x1, 0.1)
x2 = c(x2, 0.8)
y = c(y,6)

modeling2 = lm(y~x1+x2)
summary(modeling2)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73348 -0.69318 -0.05263  0.66385  2.30619
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.2267     0.2314   9.624 7.91e-16 ***
## x1            0.5394     0.5922   0.911  0.36458
## x2            2.5146     0.8977   2.801  0.00614 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.075 on 98 degrees of freedom
## Multiple R-squared:  0.2188, Adjusted R-squared:  0.2029
## F-statistic: 13.72 on 2 and 98 DF,  p-value: 5.564e-06
```

```
plot(y~x1+x2)
```




```
plot(x1, x2)
text(x = 0.1, y = 0.8, col="blue")
```




```
# new point seems to be outlier
```

Chapter 3, Question 15

```
library(ISLR)
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
```

```
# Predicting Per Capita Crime Rate

# (a)

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

```
# Coeff: -0.07393
# PVal: 5.506e-06

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

```
# Coeff: 0.50978
# PVal: < 2.2e-16

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

```
# Coeff: -1.8928
# PVal: 0.2094

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

```
# Coeff: 31.249
# PVal: < 2.2e-16

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

```
# Coeff: -2.684
# PVal: 6.347e-07

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

```
# Coeff: 0.10779
# PVal: 2.855e-16

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

```
# Coeff: -1.5509
# PVal: < 2.2e-16

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

```
# Coeff: 0.61791
# PVal: < 2.2e-16

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

```
# Coeff: 0.029742
# PVal: < 2.2e-16

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

```
# Coeff: 1.1520
# PVal: 2.943e-11

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

```
##
## 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
# Coeff: -0.036280
# PVal: < 2.2e-16

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

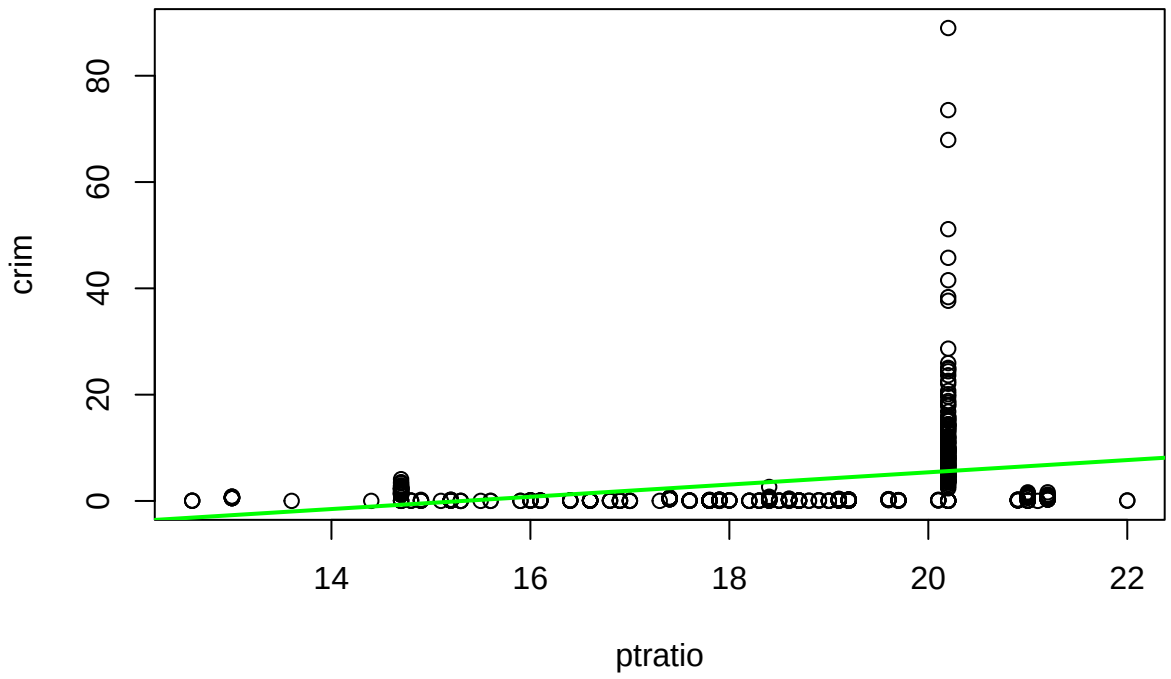
```
##
## Call:
## lm(formula = crim ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.925  -2.822  -0.664   1.079  82.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.33054    0.69376  -4.801 2.09e-06 ***
## lstat        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
# Coeff: 0.54880
# PVal: < 2.2e-16

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

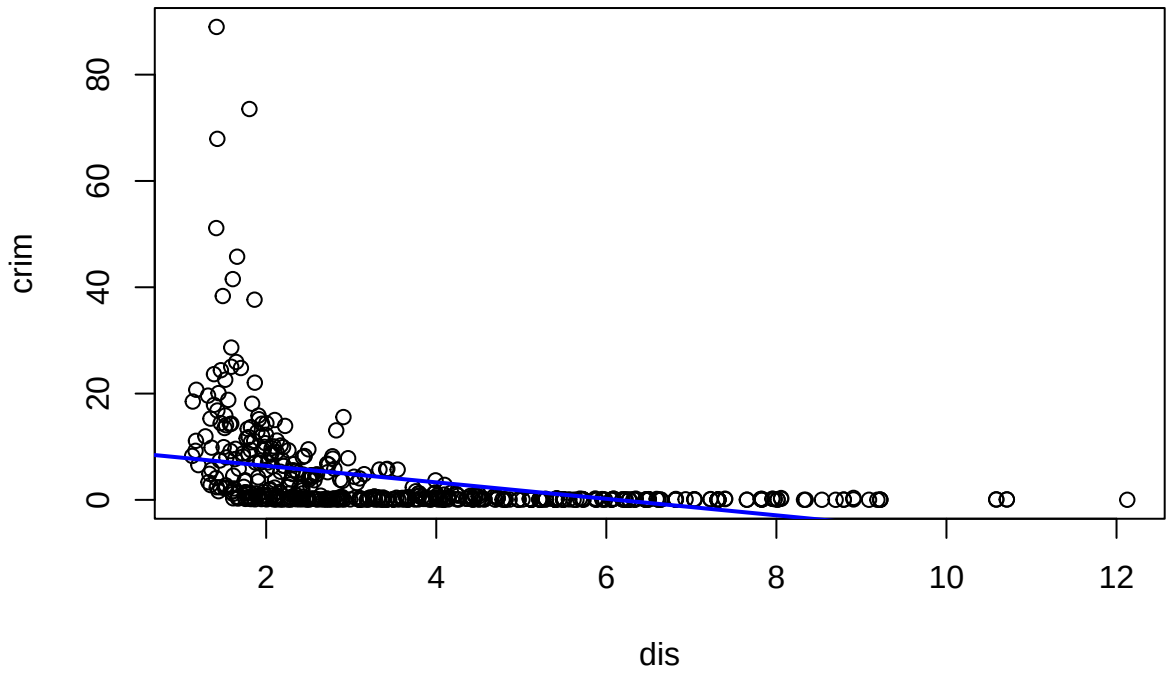
```
##
## Call:
## lm(formula = crim ~ medv, data = Boston)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.071 -4.022 -2.343  1.298 80.957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.79654    0.93419   12.63   <2e-16 ***
## medv        -0.36316    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
# Coeff: -0.36316
# PVal: < 2.2e-16

# Positive Relationship: indus, nox, age, rad, tax, lstat, ptratio
# Negative Relationship: zn, chas, rm, dis, black, medv
```

```
# Looking at pvalues, only chas is statistically insignificant

attach(Boston)

plot(ptratio, crim)
abline(model_ptratio, lwd = 2, col ="green")
```




```
plot(dis, crim)
abline(model_dis, lwd = 2, col ="blue")
```



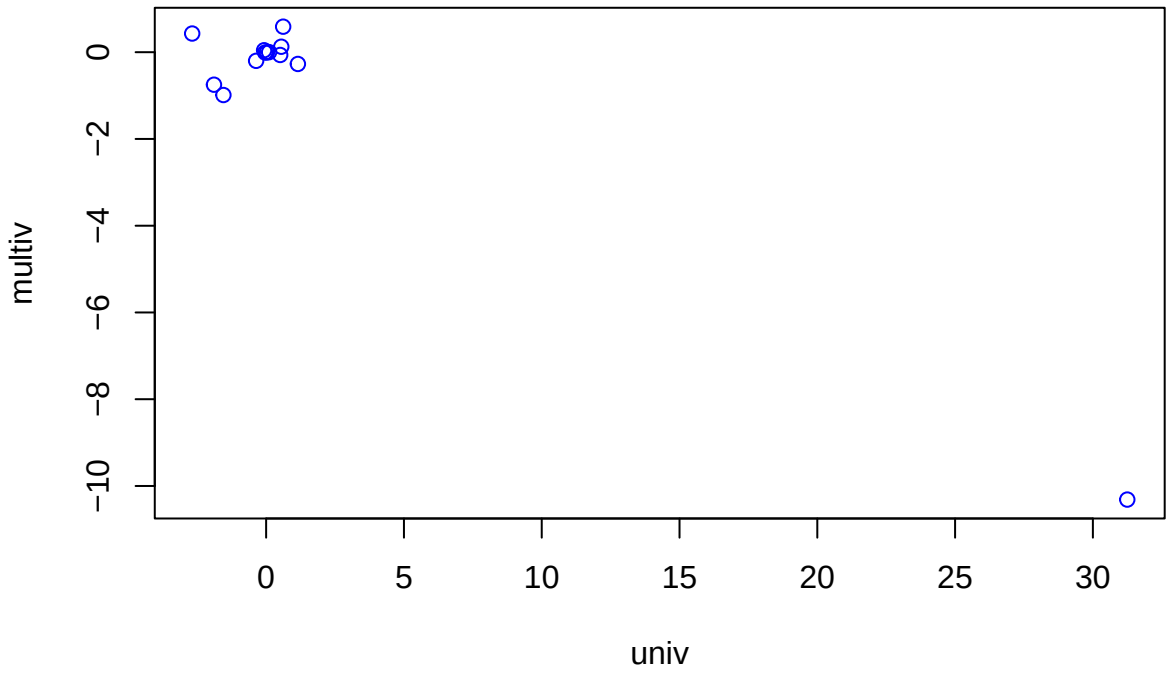
```r
# (b)

multi = lm(crim~., data = Boston)
summary(multi)
```

```
##
## 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
# Coeffs

# Zn -0.07393 -> 0.04485
# indus 0.50978 -> -0.063855
# chas -1.8928 -> -0.749134
# nox 31.249 -> -10.313535
# rm -2.684 -> 0.430131
# age 0.10779 -> 0.001452
# dis -1.5509 -> -0.987176
# rad 0.61791 -> 0.588209
# tax 0.029742 -> -0.003780
# ptratio 1.1520 -> -0.271081
# black -0.036280 -> -0.007538
# lstat 0.54880 -> 0.126211
# medv -0.36316 -> -0.198887

# We can see that for some of the variable the coefficients have not only
# changed in magnitude but also sign
```

```r
# In simple linear regression, all vriables are ignored while in multiple
# regression all others are kept constant while analysis the impact of
# predictor in question

# Thus this change in coefficients and their statistical significance
# is well justified by collinearity

# Null hypothesis can be rejected for pval < 0.05 -> zn, dis, rad, black, medv
```

```r
## (c)

# The results comparison between (a) and (b) has been already explained above.

univ <- vector(mode = "numeric", length = 0)
# creating empty vector to populate with coefficients

univ <- c(univ, model_zn$coefficients[2]) # R indexing begins from 1
univ <- c(univ, model_indus$coefficients[2])
univ <- c(univ, model_chas$coefficients[2])
univ <- c(univ, model_nox$coefficients[2])
univ <- c(univ, model_rm$coefficients[2])
univ <- c(univ, model_age$coefficients[2])
univ <- c(univ, model_dis$coefficients[2])
univ <- c(univ, model_rad$coefficients[2])
univ <- c(univ, model_tax$coefficients[2])
univ <- c(univ, model_ptratio$coefficients[2])
univ <- c(univ, model_black$coefficients[2])
univ <- c(univ, model_lstat$coefficients[2])
univ <- c(univ, model_medv$coefficients[2])

multiv <- vector(mode = "numeric", length = 0)
# creating empty vector to populate with coefficients

multiv <- c(multiv, multi$coefficients)
multiv <- multiv[-1]

par(mfrow = c(1, 1))
plot(univ, multiv, col = 'blue')
```




```
## (d)

# Checking Degree 3 Polynomial Fit

model_zn_3 = lm(crim~zn+I(zn^2)+I(zn^3), data=Boston)
summary(model_zn_3)
```

```
##
## 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
```

```
# zn

model_indus_3 = lm(crim~indus+I(indus^2)+I(indus^3), data=Boston)
summary(model_indus_3)
```

```
##
```

```
## 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
```

```
# indus, indus^2, indus^3

model_nox_3 = lm(crim~nox+I(nox^2)+I(nox^3), data=Boston)
summary(model_nox_3)
```

```
##
## 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
```

```
# nox, nox^2, nox^3

model_rm_3 = lm(crim~rm+I(rm^2)+I(rm^3), data=Boston)
summary(model_rm_3)
```

```
##
## 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
```

```
# None

model_age_3 = lm(crim~age+I(age^2)+I(age^3), data=Boston)
summary(model_age_3)
```

```
##
## 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
```

```
# age^2, age^3

model_dis_3 = lm(crim~dis+I(dis^2)+I(dis^3), data=Boston)
summary(model_dis_3)
```

```
##
## 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
```

```
# dis, dis^2, dis^3

model_rad_3 = lm(crim~rad+I(rad^2)+I(rad^3), data=Boston)
summary(model_rad_3)
```

```
##
## 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
```

```
# None

model_tax_3 = lm(crim~tax+I(tax^2)+I(tax^3), data=Boston)
summary(model_tax_3)
```

```
##
## 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
```

```
# None

model_ptratio_3 = lm(crim~ptratio+I(ptratio^2)+I(ptratio^3), data=Boston)
summary(model_ptratio_3)
```

```
##
## 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
```

```
# ptratio, ptratio^2, ptratio^3

model_black_3 = lm(crim~black+I(black^2)+I(black^3), data=Boston)
summary(model_black_3)
```

```
##
## 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
```

```r
# None

model_lstat_3 = lm(crim~lstat+I(lstat^2)+I(lstat^3), data=Boston)
summary(model_lstat_3)
```

```
##
## 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
```

```r
# None

model_medv_3 = lm(crim~medv+I(medv^2)+I(medv^3), data=Boston)
summary(model_medv_3)
```

```
##
## 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
```

```r
# medv, medv^2, medv^3

# From all above summary results we can see that following
```

```
# degree 3 coefficients are statistically significant:

# indus, nox, age, dis, ptratio, medv
```

Chapter 6, Question 9

```
library(ISLR)
data(College)

# install.packages("glmnet") # DONE
# install.packages("pls) # DONE
# install.packages("pcr") # DONE

library(glmnet)
```

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

```
## Loaded glmnet 4.1-4
```

```
require(pls)
```

```
## Loading required package: pls
```

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

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

```
set.seed(1)

x = model.matrix(Apps~., College)[,-1]
y = College$Apps

## (a)

train = sample(1:nrow(College), round(nrow(College) * 0.6))
test = (-train)

df_train = College[train, ]
df_test = College[-train, ]
```

```
## (b)

linear_model = lm(Apps ~ ., data = df_train)
linear_pred = predict(linear_model, df_test)
linear_mse = mean((linear_pred - df_test$Apps)^2) # Output: 1124481.5726
```

```
## (c)

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

cv1 = cv.glmnet(x[train,], y[train], alpha = 0)
lam = cv1$lambda.min # 382.88
```

```
ridge1 = glmnet(x[train,], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge_pred = predict(ridge1, s = lam, newx = x[test,])
ridge_err = mean((ridge_pred - y[test])^2) # Output = 1033013.5726
```

```
## (d)

cv2 = cv.glmnet(x[train,], y[train], alpha = 1)
lam2 = cv2$lambda.min

lasso1 = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
lasso_pred = predict(lasso1, s = lam2, newx = x[test,])
lasso_err = mean((lasso_pred - y[test])^2) # Output = 1113405.8410

lasso.coef = predict(lasso1, type = "coefficients", s = lam2)[1:18,]

length(lasso.coef[lasso.coef != 0]) # Ouput 17
```

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

```
length(lasso.coef[lasso.coef == 0]) # Ouput 1
```

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

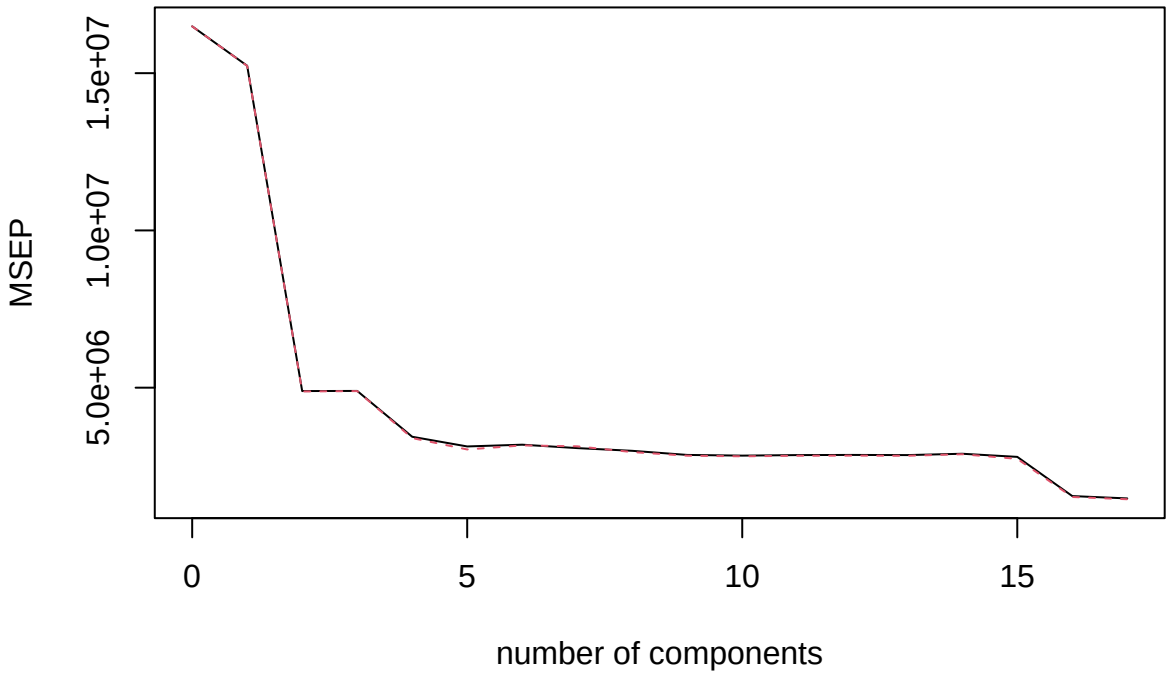
```
lasso.coef[lasso.coef == 0] # F.Undergrad
```

```
## F.Undergrad
##           0
```

```
lasso.coef
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -665.63058052 -442.72714002    1.70268040   -0.81051715   59.79404848
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
##  -18.44331561    0.00000000    0.06562778   -0.08672713    0.17166996
##         Books      Personal           PhD      Terminal     S.F.Ratio
##    0.39940807   -0.01122307  -12.29852997    0.93605355   21.66974536
##   perc.alumni        Expend     Grad.Rate
##    1.46131819    0.05675117    6.73310680
```

```
## (e)

pcr1 = pcr(Apps~., data = College, subset = train, scale = T, validation = "CV")
validationplot(pcr1, val.type = "MSEP")
```




```
# Observing from the graph - test for 5, 15, 16

pcr_pred = predict(pcr1, x[test,], ncomp = 5)
mean((pcr_pred - y[test])^2) # 1877569
```

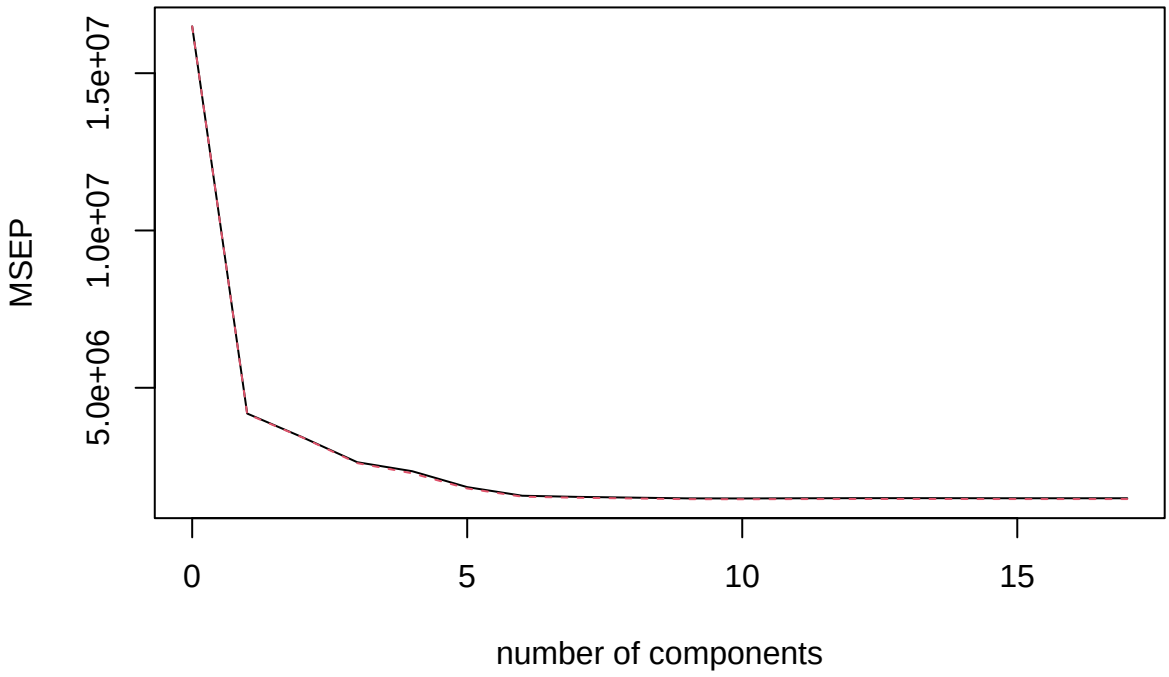
```
## [1] 1877569
```

```
pcr_pred = predict(pcr1, x[test,], ncomp = 15)
mean((pcr_pred - y[test])^2) # 1238541
```

```
## [1] 1238541
```

```
pcr_pred = predict(pcr1, x[test,], ncomp = 16)
pcr_err = mean((pcr_pred - y[test])^2) # 1206092 # Selecting for 16
```

```
## (f)

pls1 = plsr(Apps~., data = College, subset = train, scale = T, validation = "CV")
validationplot(pls1, val.type = "MSEP")
```




```
# Observing from the graph - test for 1, 5, 6, 7

pls_pred = predict(pls1, x[test,], ncomp = 1)
mean((pls_pred - y[test])^2) # 2391370
```

```
## [1] 2391370
```

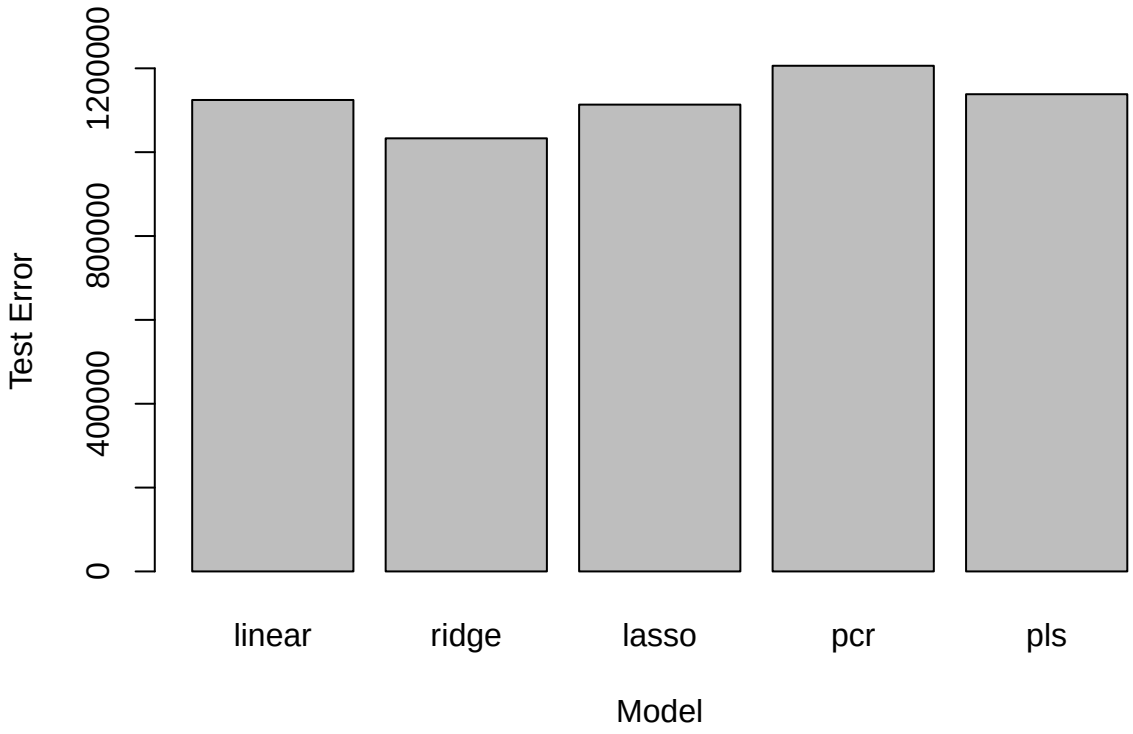
```
pls_pred = predict(pls1, x[test,], ncomp = 5)
mean((pls_pred - y[test])^2) # 1175328
```

```
## [1] 1175328
```

```
pls_pred = predict(pls1, x[test,], ncomp = 6)
pls_err = mean((pls_pred - y[test])^2) # 1138191 # Selecting for 6

pls_pred = predict(pls1, x[test,], ncomp = 7)
mean((pls_pred - y[test])^2) # 1150548
```

```
## [1] 1150548
```

```
## (g)

err = c(linear_mse, ridge_err, lasso_err, pcr_err, pls_err)
barplot(err, xlab = "Model", ylab="Test Error", names = c("linear", "ridge", "lasso", "pcr", "pls"))
```




```
# Order of error (highest to lowest)
# PCR, PLS, LINEAR, LASSO, RIDGE
# All models are quite close in performance

# Calculation R2 for accuracy

test_avg = mean(y[test])

linear_r2 = 1 - mean((linear_pred - y[test])^2) / mean((test_avg - y[test])^2)
# 0.9118

ridge_r2 = 1 - mean((ridge_pred - y[test])^2) / mean((test_avg - y[test])^2)
# 0.9189

lasso_r2 = 1 - mean((lasso_pred - y[test])^2) / mean((test_avg - y[test])^2)
# 0.9126

pcr_r2 = 1 - mean((pcr_pred - y[test])^2) / mean((test_avg - y[test])^2)
# 0.9054

pls_r2 = 1 - mean((pls_pred - y[test])^2) / mean((test_avg - y[test])^2)
# 0.9097

# R2 indicates predictions are close to correct
```

Chapter 6, Question 11

```
## (a) - TRY OUT REG METHODS IN CHAPTER - DISCUSS RESULTS

library(MASS)
```

```
data(Boston)
```

```
### BEST SUBSET SELECTION ###

names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```
dim(Boston)
```

```
## [1] 506  14
```

```
sum(is.na(Boston$crim))
```

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

```
library(leaps)
regfit.full = regsubsets(crim~., Boston)
s1 = summary(regfit.full)
names(s1)
```

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

```
s1$rsq
```

```
## [1] 0.3912567 0.4207965 0.4286123 0.4334892 0.4392738 0.4440173 0.4476594
## [8] 0.4504606
```

```
regfit.full2 = regsubsets(crim~., data = Boston, nvmax = 13)
s2 = summary(regfit.full2)
names(s2)
```

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

```
s2$rsq
```

```
##  [1] 0.3912567 0.4207965 0.4286123 0.4334892 0.4392738 0.4440173 0.4476594
##  [8] 0.4504606 0.4524408 0.4530572 0.4535605 0.4540031 0.4540104
```

```
par(mfrow=c(2,2))
plot(s2$rss ,xlab="Number of Variables ",ylab="RSS",
       type="l")
plot(s2$adjr2 ,xlab="Number of Variables ",
       ylab="Adjusted RSq",type="l")

# Plot makes it evident to keep all variables

set.seed(1)
train = sample(c(TRUE, FALSE), nrow(Boston), rep = TRUE)
test = (!train)

regfit.best = regsubsets(crim~., data = Boston[train,], nvmax = 13)
test.mat = model.matrix(crim~., data = Boston[test,])
val.errors = rep(NA, 13)
for (i in 1:13){
  coefi = coef(regfit.best, id = i)
  pred = test.mat[,names(coefi)]%*%coefi
  val.errors[i] = mean((Boston$crim[test]-pred)^2)
```

```
}

val.errors
```

```
##  [1] 53.58865 54.37254 52.66064 52.50837 51.53113 51.12540 51.04084 50.69984
##  [9] 50.43627 50.52382 50.71664 50.68209 50.65678
```

```
which.min(val.errors) # 9
```

```
## [1] 9
```

```
coef(regfit.best, 9)
```

```
## (Intercept)          zn       indus         nox         dis         rad
## 16.49784501  0.04428501 -0.11356470 -6.80041892 -0.87067024  0.48133294
##     ptratio       black       lstat        medv
## -0.17759119 -0.01438142  0.12943566 -0.13215744
```

```
regfit.best = regsubsets(crim~., data = Boston, nvmax = 13)
coef(regfit.best, 9)
```

```
##   (Intercept)            zn         indus           nox           dis
##  19.124636156   0.042788127  -0.099385948 -10.466490364  -1.002597606
##           rad       ptratio         black         lstat          medv
##   0.539503547  -0.270835584  -0.008003761   0.117805932  -0.180593877
```

```
k = 10
set.seed(1)
folds = sample(1:k, nrow(Boston), replace = TRUE)
cv.errors = matrix(NA, k, 13, dimnames = list(NULL, paste(1:13)))

predict.regsubsets = function (object ,newdata ,id ,...){
  form=as.formula(object$call [[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object ,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
  }

for(j in 1:k){
  best.fit = regsubsets(crim~., data = Boston[folds!=j,], nvmax = 13)
  for (i in 1:13){
    pred = predict(best.fit, Boston[folds == j,], id = i)
    cv.errors[j, i] = mean((Boston$crim[folds==j]-pred)^2)
  }
}

mean.cv.errors = apply(cv.errors, 2, mean)
mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 45.44573 43.87260 43.94979 44.02424 43.96415 43.96199 42.96268 42.66948
##        9       10       11       12       13
## 42.53822 42.73416 42.52367 42.46014 42.50125
```

```
par(mfrwo=c(1,1))
```

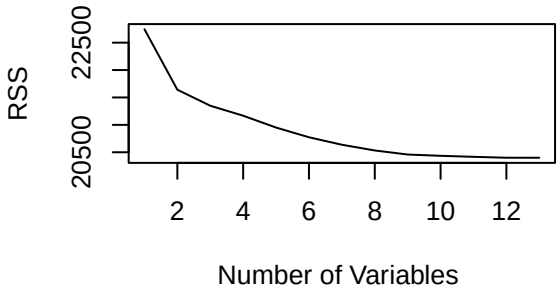
```
## Warning in par(mfrwo = c(1, 1)): "mfrwo" is not a graphical parameter
```

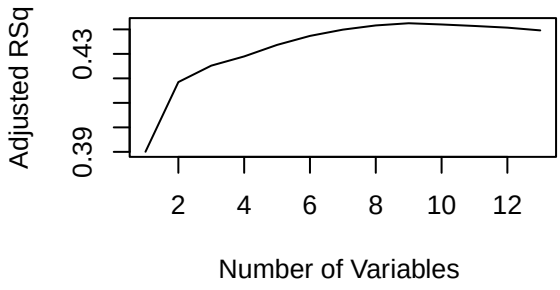
```
plot(mean.cv.errors, type = 'b')

# cv error lowest for model with 9 variables
mean.cv.errors[9] # Output: 42.53822
```

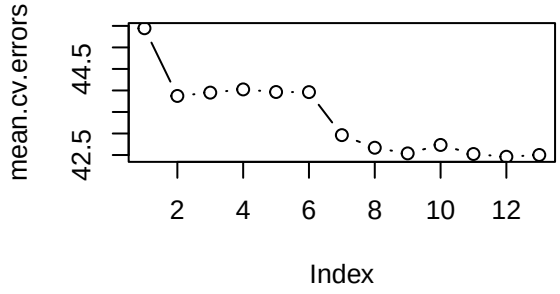
```
##        9
## 42.53822
```

```
reg.best = regsubsets(crim~., data = Boston, nvmax = 13)
coef(reg.best, 9)
```

```
##   (Intercept)            zn         indus           nox           dis
##  19.124636156   0.042788127  -0.099385948 -10.466490364  -1.002597606
##           rad       ptratio         black         lstat          medv
##   0.539503547  -0.270835584  -0.008003761   0.117805932  -0.180593877
```








```
### RIDGE REGRESSION ###

x = model.matrix(crim~., Boston)[,-1]
y = Boston$crim

library(glmnet)
grid = 10^seq(10, -2, length = 100)
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)
dim(coef(ridge.mod)) # 14 100
```

```
## [1]  14 100
```

```
ridge.mod$lambda[50] # 11497.57
```

```
## [1] 11497.57
```

```
coef(ridge.mod)[,50]
```

```
##   (Intercept)            zn         indus          chas           nox
##  3.589847e+00 -5.493033e-05  3.793684e-04 -1.412990e-03  2.325693e-02
##            rm           age           dis           rad           tax
## -1.996087e-03  8.018407e-05 -1.153934e-03  4.604414e-04  2.214944e-05
##       ptratio         black         lstat          medv
##  8.570813e-04 -2.702161e-05  4.083826e-04 -2.701952e-04
```

```
ridge.mod$lambda[60] # 705.4802
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##   (Intercept)            zn         indus          chas           nox
##  3.2516685945 -0.0007972592  0.0057051072 -0.0225316252  0.3514124032
##            rm           age           dis           rad           tax
## -0.0297634708  0.0012055897 -0.0174560167  0.0071446717  0.0003412896
##       ptratio         black         lstat          medv
##  0.0130090009 -0.0004177804  0.0062311863 -0.0041201724
```

```
predict(ridge.mod, s = 50, type = "coefficients")[1:14,]
```

```
##  (Intercept)           zn        indus         chas          nox           rm
##  0.838036330 -0.002577783  0.035801043 -0.249700752  2.367151149 -0.166588894
##          age          dis          rad          tax      ptratio        black
##  0.007679277 -0.121899702  0.065407357  0.002877986  0.091353127 -0.003626702
##        lstat         medv
##  0.047688080 -0.031157551
```

```
set.seed (1)
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]

ridge.mod = glmnet(x[train,], y[train], alpha = 0, lambda = grid, thresh = 1e-12)
ridge.pred = predict(ridge.mod, s = 4, newx = x[test,])
mean((ridge.pred-y.test)^2) # 40.77723
```

```
## [1] 40.77723
```

```
mean((mean(y[train])-y.test)^2) # 62.68428
```

```
## [1] 62.68428
```

```
ridge.pred = predict(ridge.mod, s = 1e10, newx = x[test,])
mean((ridge.pred-y.test)^2) # 62.68428
```

```
## [1] 62.68428
```

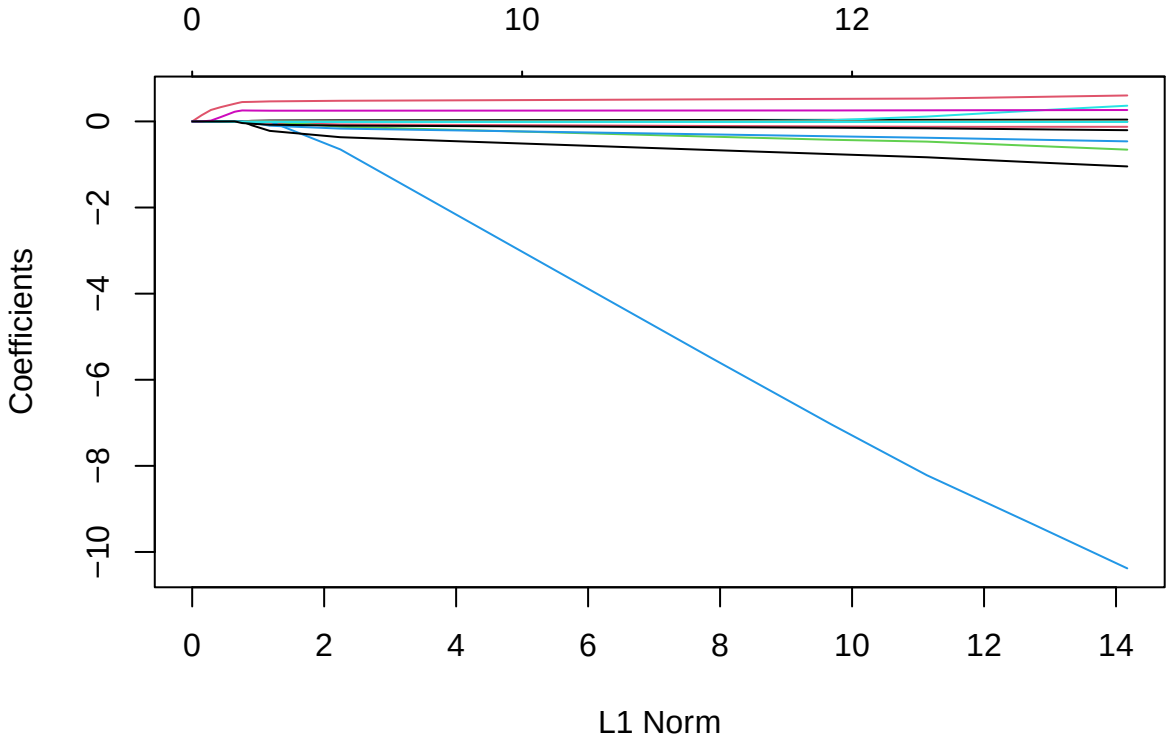
```
ridge.pred = predict(ridge.mod, s = 0, newx = x[test,])
mean((ridge.pred-y.test)^2) # 41.51969
```

```
## [1] 41.51969
```

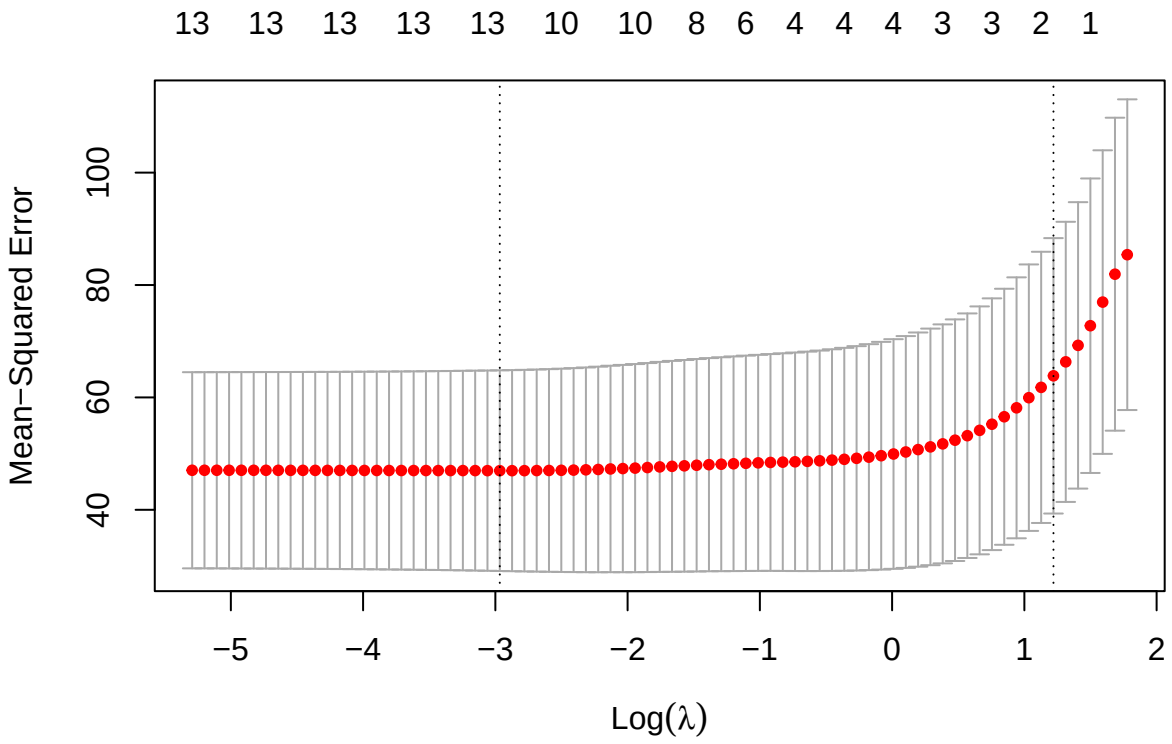
```
### LASSO ###

lasso.mod = glmnet(x[train,], y[train], alpha = 1, lambda = grid)
```

```
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
set.seed(1)
cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam, newx = x[test,])
mean((lasso.pred-y.test)^2) # 40.99066
```

```
## [1] 40.99066
```

```
out = glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:14,]
lasso.coef
```

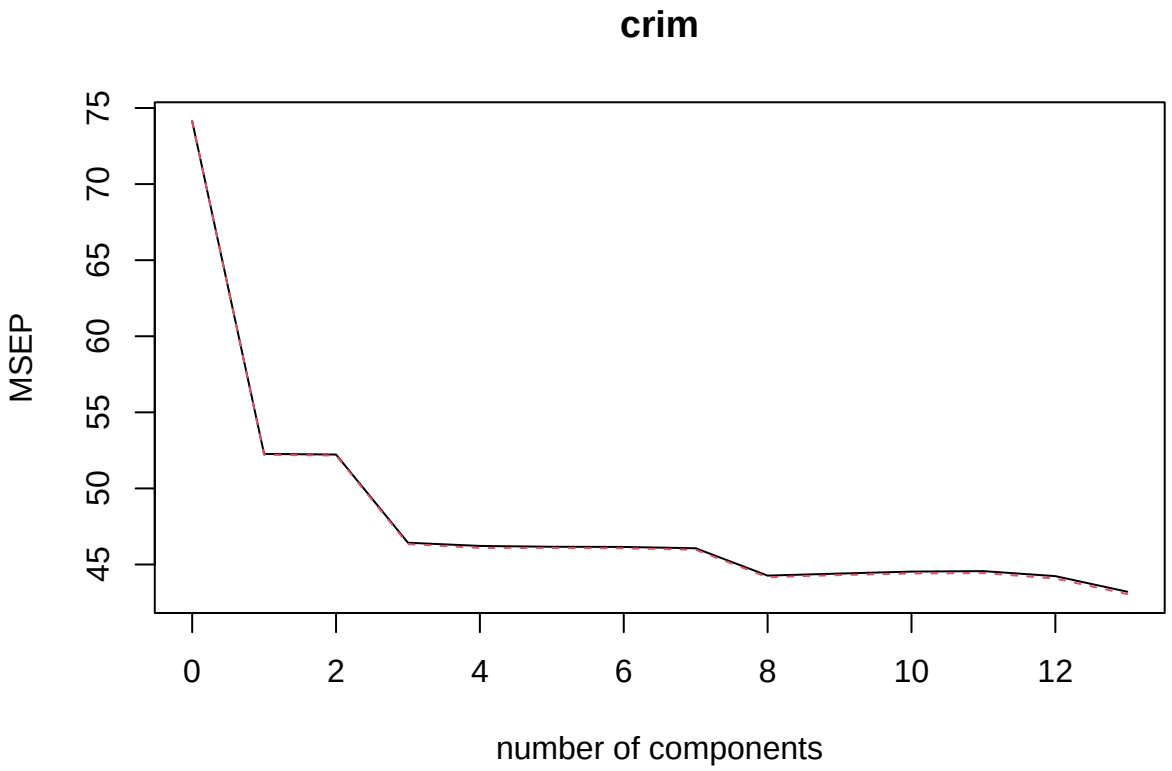
```
##   (Intercept)            zn         indus          chas           nox
##  1.269174e+01  3.628905e-02 -7.012417e-02 -5.842484e-01 -6.989138e+00
##            rm           age           dis           rad           tax
##  2.308650e-01  0.000000e+00 -7.886241e-01  5.146154e-01 -2.235567e-05
##       ptratio         black         lstat          medv
## -1.884305e-01 -7.544153e-03  1.248260e-01 -1.582852e-01
```

```
### PRINCIPAL COMPONENT REGRESSION ###

library(pls)
set.seed(2)
pcr.fit = pcr(crim~., data = Boston, scale = TRUE, validation = "CV")

summary(pcr.fit)
```

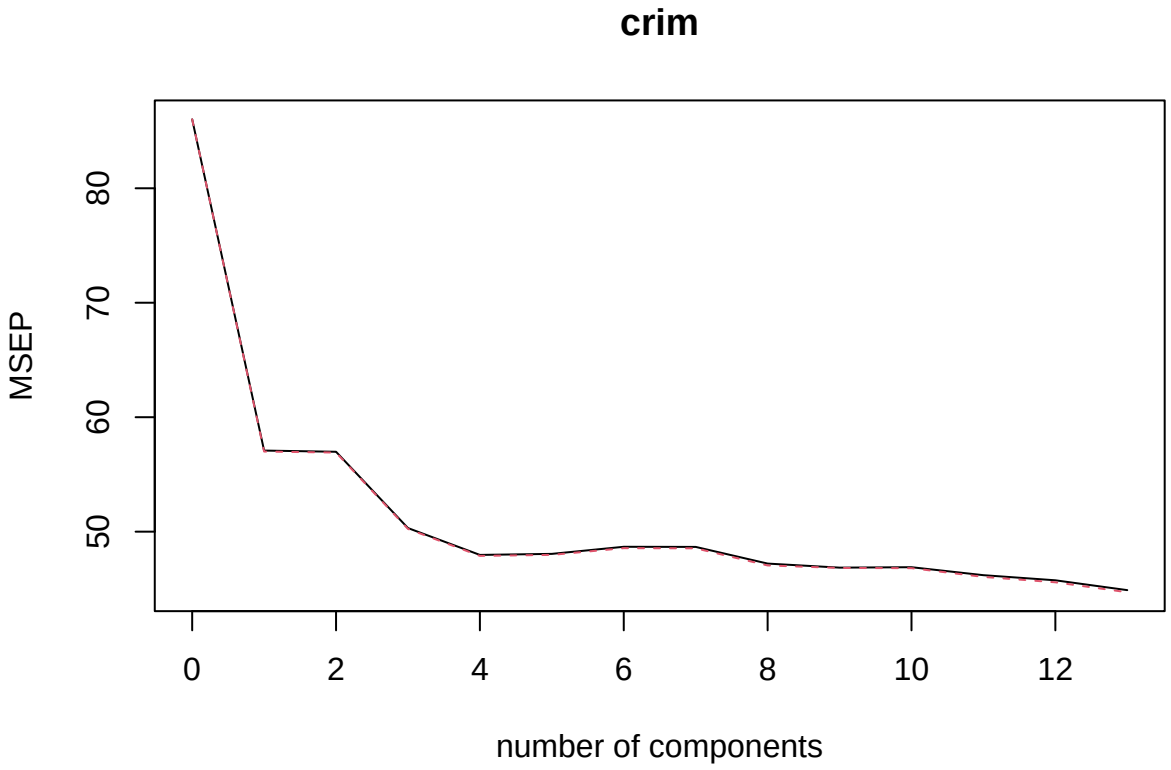
```
## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.229    7.227    6.814    6.799    6.795    6.794
## adjCV         8.61    7.225    7.222    6.807    6.789    6.788    6.787
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.787    6.654    6.664     6.673     6.676     6.651     6.573
## adjCV    6.780    6.645    6.656     6.664     6.666     6.639     6.562
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14    93.45
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14    42.47
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       95.40     97.04     98.46     99.52     100.0
## crim    42.55     42.78     43.04     44.13      45.4
```

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




```
set.seed(1)
pcr.fit = pcr(crim~., data = Boston, subset = train, scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```




```
# how to select ncomp from graph? It's decreasing throughout
# 8

pcr.pred = predict(pcr.fit, x[test,], ncomp = 8)
mean((pcr.pred-y.test)^2) # 43.21097
```

```
## [1] 43.21097
```

```
pcr.pred = predict(pcr.fit, x[test,], ncomp = 13)
mean((pcr.pred-y.test)^2) # 41.54639
```

```
## [1] 41.54639
```

```
pcr.pred = predict(pcr.fit, x[test,], ncomp = 4)
mean((pcr.pred-y.test)^2) # 43.91107
```

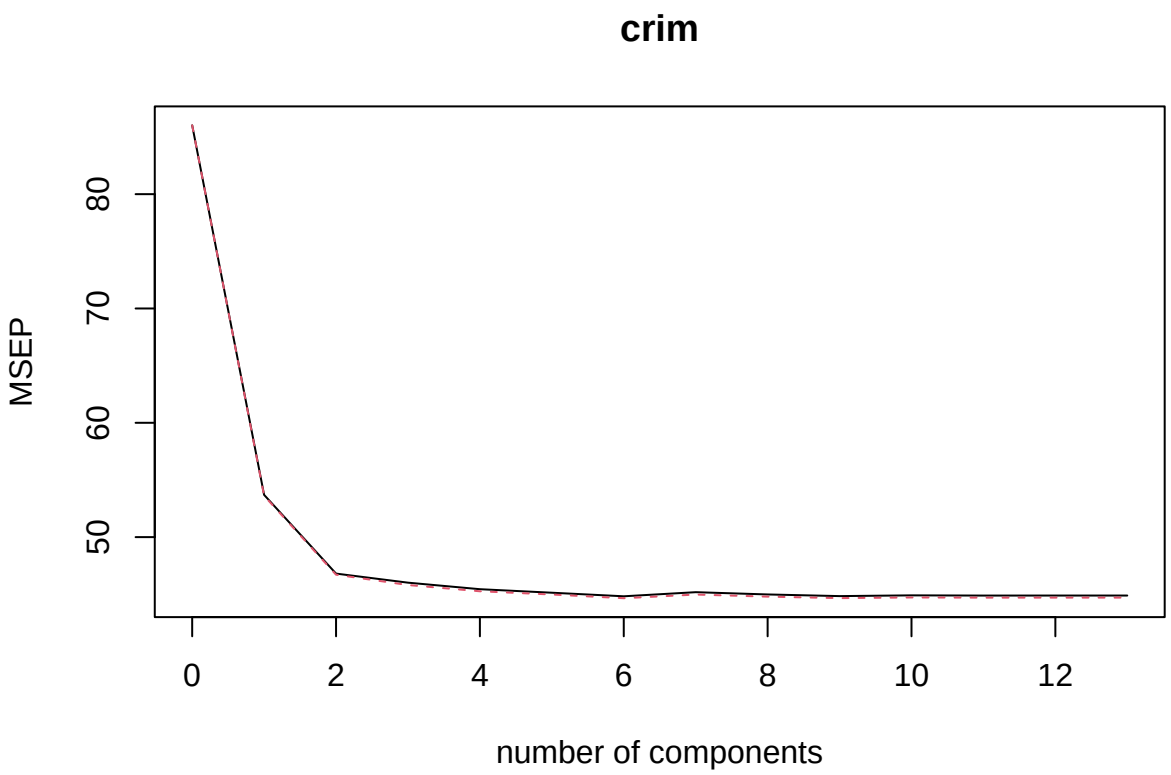
```
## [1] 43.91107
```

```
### PARTIAL LEAST SQUARES ###

set.seed(1)
pls.fit = plsr(crim~., data = Boston, subset = train, scale = TRUE, validation = "CV")
summary(pls.fit)
```

```
## Data:    X dimension: 253 13
##  Y dimension: 253 1
## Fit method: kernelpls
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
```

```
## CV           9.275    7.328    6.842    6.784    6.741    6.718    6.695
## adjCV        9.275    7.322    6.834    6.768    6.728    6.707    6.683
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.722    6.707    6.696     6.701     6.700     6.700     6.700
## adjCV    6.706    6.693    6.683     6.688     6.686     6.686     6.686
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       48.10    57.46    62.68    69.65    77.95    80.82    84.40    87.65
## crim    38.94    47.55    49.73    50.61    50.85    51.17    51.29    51.35
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       90.47     93.07     96.66     98.26    100.00
## crim    51.37     51.37     51.37     51.37     51.37
```

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




```
# how to select ncomp

pls.pred = predict(pls.fit, x[test,], ncomp = 2)
mean((pls.pred-y.test)^2) # 42.74086
```

```
## [1] 42.74086
```

```
pls.fot = plsr(crim~., data = Boston, scale = TRUE, ncomp = 2)
summary(pls.fit)
```

```
## Data:    X dimension: 253 13
##  Y dimension: 253 1
## Fit method: kernelpls
## Number of components considered: 13
##
```

```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           9.275    7.328    6.842    6.784    6.741    6.718    6.695
## adjCV        9.275    7.322    6.834    6.768    6.728    6.707    6.683
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.722    6.707    6.696     6.701     6.700     6.700     6.700
## adjCV    6.706    6.693    6.683     6.688     6.686     6.686     6.686
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       48.10    57.46    62.68    69.65    77.95    80.82    84.40    87.65
## crim    38.94    47.55    49.73    50.61    50.85    51.17    51.29    51.35
##       9 comps  10 comps  11 comps  12 comps  13 comps
## X       90.47     93.07     96.66     98.26    100.00
## crim    51.37     51.37     51.37     51.37     51.37
```

```
## (b) - Propose models that perform well, evaluate using cross validation
```

```
## (c) - does chosen model involve all features - why or why not

# I would choose the Lasso model, as it gives the lowest test mse.
# Lasso models are generally more interpretable.

# It results in a sparse model with 10 variables.
# Two variables whose effect on the response were below the required
# threshold were removed.
```

Chapter 8, Question 8

```
library(ISLR)
data(Carseats)
dim(Carseats)
```

```
## [1] 400  11
```

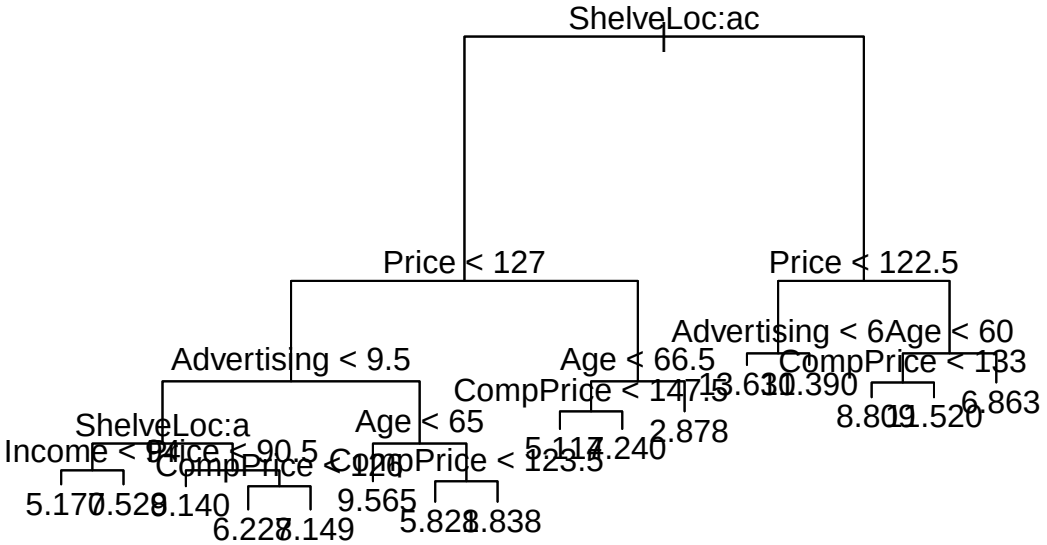
```
## (a)

set.seed(1)
train = sample(c(TRUE, FALSE), nrow(Carseats), rep = TRUE)
test = (!train)
```

```
## (b)

par(mfrow = c(1, 1))

library(tree)
tree_model = tree(Sales~., Carseats[train,])

summary(tree_model)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats[train, ])
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Advertising" "Income"      "CompPrice"
```

```
## [6] "Age"
## Number of terminal nodes:  16
## Residual mean deviance:  2.394 = 442.9 / 185
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.74400 -0.89700 -0.05937  0.00000  0.93060  6.66900
```

```
plot(tree_model)
text(tree_model)
```

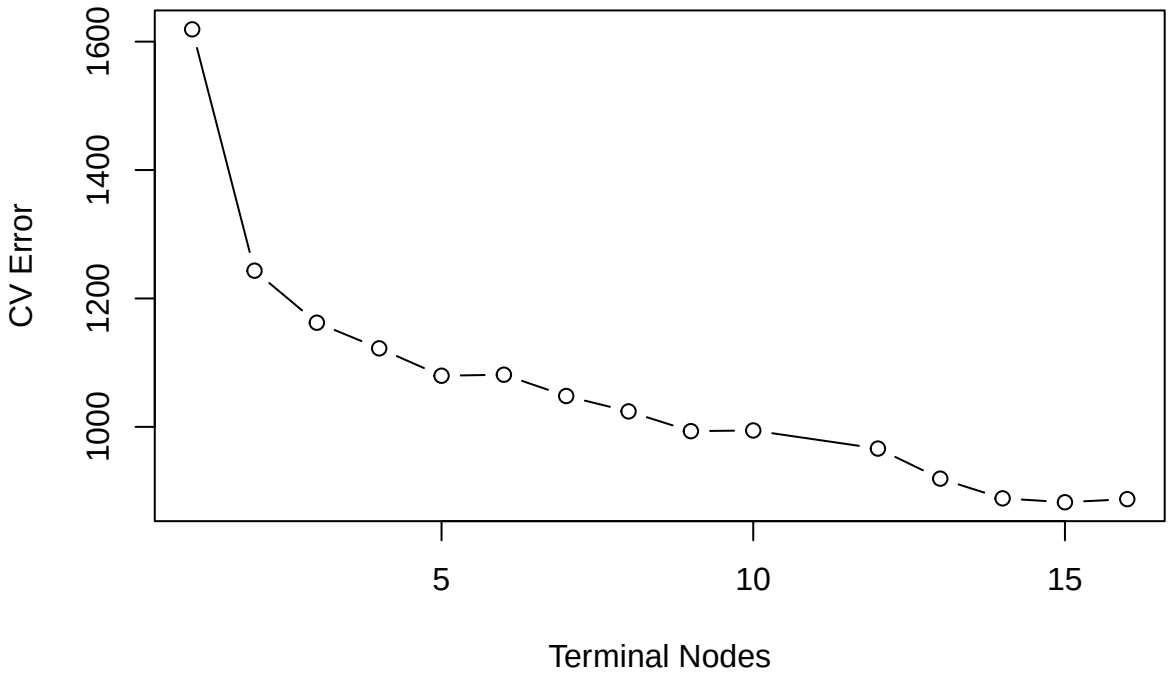



```
tree_pred = predict(tree_model, Carseats[test,])
test_mse = mean((tree_pred - Carseats[test,]$Sales)^2) # 5.207756

# Points to note:
# Variables actually used in the tree construction: ShelveLoc, Price,
# Advertising, Income, CompPrice, Age

# No. of Terminal Nodes: 16
# Test MSE = 5.2
```

```
## (c)

set.seed(2)
cv_car = cv.tree(tree_model)
summary(cv_car)
```

```
##        Length Class  Mode
## size   15     -none- numeric
## dev    15     -none- numeric
## k      15     -none- numeric
## method  1     -none- character
```

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




```
# Optimal value = 9 Terminal Nodes

prune_car = prune.tree(tree_model, best = 9)
tree_pred2 = predict(prune_car, Carseats[test,])
test_mse2 = mean((tree_pred2 - Carseats[test,]$Sales)^2) # 4.98545

# Pruning the tree slightly improved the MSE: 5.2 -> 4.98
```

```
## (d)

library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
set.seed(2)
bag_car = randomForest(Sales~., data = Carseats[train,], mtry = 10, importance = T)
importance(bag_car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   19.6211795    131.699726
## Income       7.5973882     97.853410
## Advertising 17.4351356    160.290717
## Population   2.4855317     71.032635
## Price       49.0567693    441.543542
## ShelveLoc   52.5432634    450.338179
## Age         13.5076447    133.766357
## Education    0.8161629     47.635165
## Urban        0.9596331     12.990954
## US           0.2780947      7.638444
```

```
bag_pred = predict(bag_car, Carseats[test,])
test_mse_bag = mean((bag_pred - Carseats[test,]$Sales)^2)
```

```
# randomForest is not available for this version for R
```

```
## (e)

set.seed(2)

car1 = randomForest(Sales~.,data=Carseats[train,],mtry=10/2,importance=T)
importance(car1)
```

```
##                %IncMSE IncNodePurity
## CompPrice   15.7914168     135.98350
## Income       4.0950976     117.45341
## Advertising 18.8770165     180.54746
## Population   3.8245971      92.60216
## Price       36.0921926     367.09517
## ShelveLoc   44.6241144     405.67345
## Age         12.2477216     152.44846
## Education   -1.6897207      55.36410
## Urban       -0.3899192      13.77597
## US           2.9518607      12.96404
```

```
car2 = randomForest(Sales~.,data=Carseats[train,],mtry=sqrt(10),importance=T)
importance(car2)
```

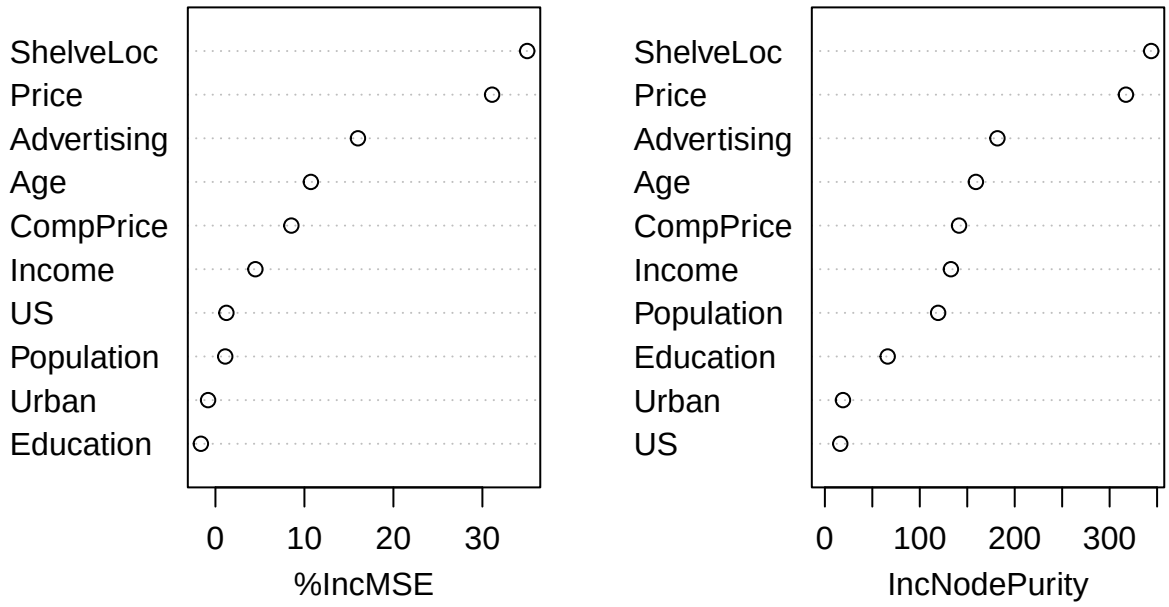
```
##                %IncMSE IncNodePurity
## CompPrice    8.5310606     141.37353
## Income       4.4891172     132.78327
## Advertising 16.0144551     181.80453
## Population   1.1012824     119.31739
## Price       31.0901404     317.19937
## ShelveLoc   35.0344742     343.81355
## Age         10.7246390     159.10963
## Education   -1.6358709      66.17645
## Urban       -0.8244829      19.05945
## US           1.2331181      16.22396
```

```
car3 = randomForest(Sales~.,data=Carseats[train,],mtry=10/4,importance=T)
importance(car3)
```

```
##                %IncMSE IncNodePurity
## CompPrice    7.6759698     134.76573
## Income       2.7391782     145.55262
## Advertising 15.0496256     178.94052
## Population  -0.8309692     130.85918
## Price       24.1485531     276.78857
## ShelveLoc   30.8297555     306.68708
## Age         10.9598221     161.34502
## Education   -1.8549693      76.27444
## Urban       -0.1916890      22.35433
## US           4.1864982      24.86787
```

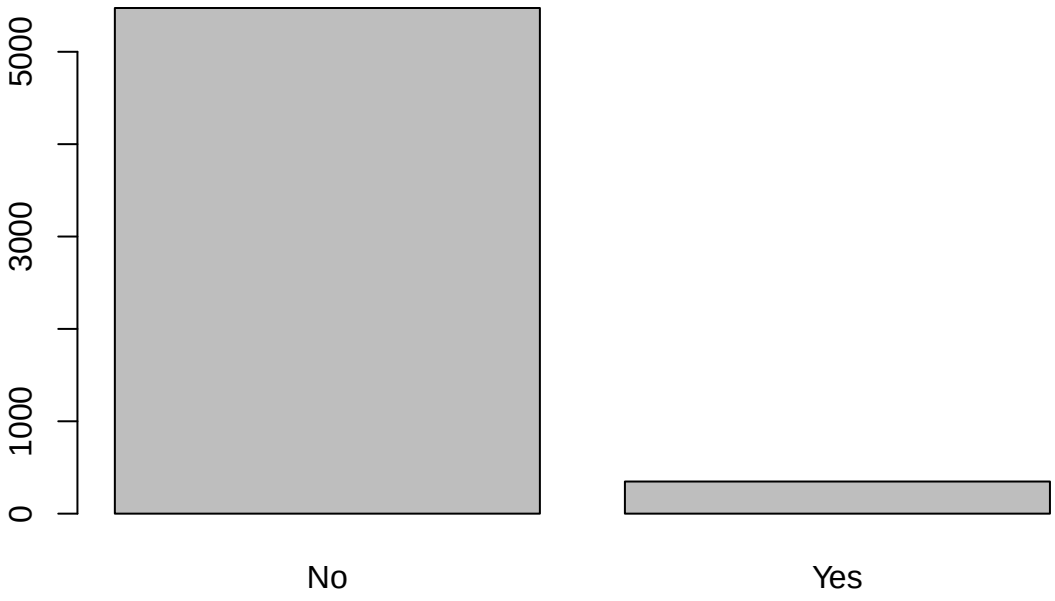
```
varImpPlot(car2)
```

car2



Chapter 8, Question 11

```
library(ISLR)
attach(Caravan)

?Caravan # Insurance Company Benchmark - 5822 records - 86 vars
plot(Caravan$Purchase)
```



```
## (a)

Caravan$PurchaseYES = rep(NA, 5822)
for (i in 1:5822) if (Caravan$Purchase[i] == "Yes")
  (Caravan$PurchaseYES[i]=1) else (Caravan$PurchaseYES[i]=0)

train_set = Caravan[1:1000,]
test_set = Caravan[1001:5822,]
```

```
## (b)

library(gbm)
```

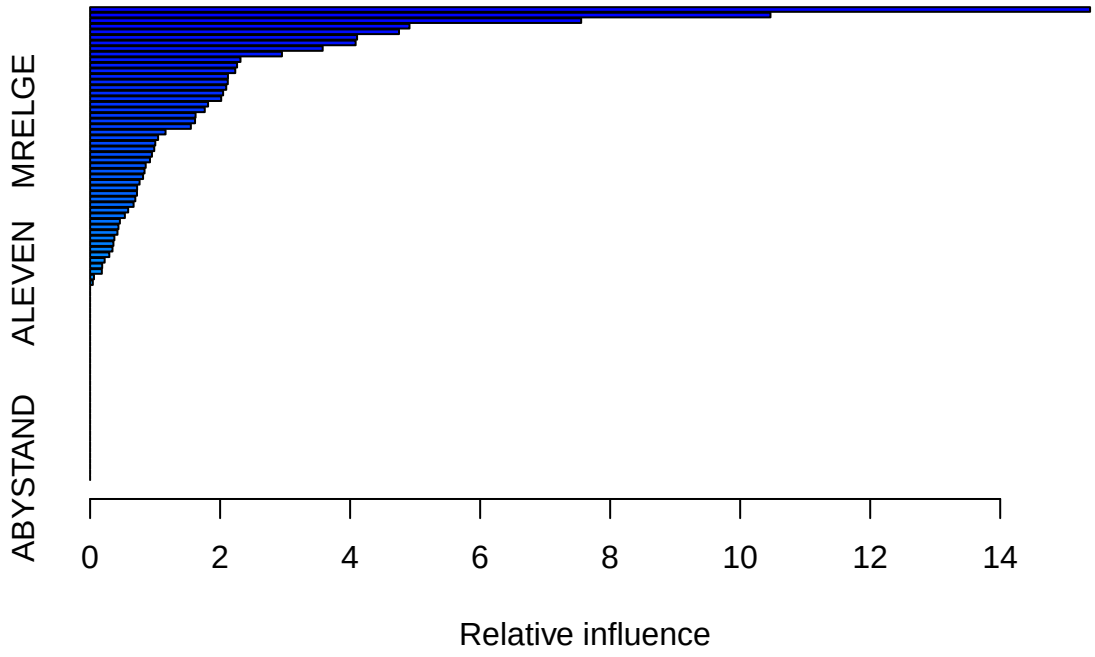
```
## Loaded gbm 2.1.8
```

```
set.seed(10)
boost_model = gbm(PurchaseYES~.-Purchase, data = train_set,
                  distribution = "bernoulli", n.trees = 1000, shrinkage = 0.01)
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 50: PVRAAUT has no variation.
```

```
## Warning in gbm.fit(x = x, y = y, offset = offset, distribution = distribution, :
## variable 71: AVRAAUT has no variation.
```

```
summary(boost_model)
```




```
##                var     rel.inf
## PPERSAUT PPERSAUT 15.38257285
## MKOOPKLA MKOOPKLA 10.46605387
## MOPLHOOG MOPLHOOG  7.55417441
## PBRAND     PBRAND  4.91302326
## MBERMIDD MBERMIDD  4.75162262
## ABRAND     ABRAND  4.10695497
## MGODGE     MGODGE  4.08460852
```

```
## MINK3045 MINK3045  3.57754339
## MOSTYPE   MOSTYPE  2.95142262
## MSKA         MSKA  2.31330767
## MSKC         MSKC  2.26066183
## MBERARBG MBERARBG  2.23298183
## PBYSTAND PBYSTAND  2.12038372
## PWAPART   PWAPART  2.11822298
## MAUT2       MAUT2  2.09432337
## MSKB1       MSKB1  2.04625792
## MAUT1       MAUT1  2.01682392
## MGODOV     MGODOV  1.81440468
## MGODPR     MGODPR  1.76389034
## MBERHOOG MBERHOOG  1.62253130
## MRELGE     MRELGE  1.61409689
## MINKGEM   MINKGEM  1.54973694
## MRELOV     MRELOV  1.16048896
## MHHUUR     MHHUUR  1.04601208
## MGODRK     MGODRK  1.00271859
## MFWEKIND MFWEKIND  0.98670997
## MAUT0       MAUT0  0.95127063
## MGEMLEEF MGEMLEEF  0.92105017
## MINKM30   MINKM30  0.85528510
## APERSAUT APERSAUT  0.83742532
## MINK7512 MINK7512  0.81474065
## MOPLMIDD MOPLMIDD  0.76091591
## MGEMOMV   MGEMOMV  0.72333895
## MFGEKIND MFGEKIND  0.72106858
## MOSHOOFD MOSHOOFD  0.69538457
## PLEVEN     PLEVEN  0.66896464
## MINK123M MINK123M  0.58561784
## PMOTSCO   PMOTSCO  0.53489044
## MBERBOER MBERBOER  0.45829619
## MZFONDS   MZFONDS  0.43597782
## MSKD         MSKD  0.42079495
## MINK4575 MINK4575  0.37339174
## MSKB2       MSKB2  0.35869222
## MZPART     MZPART  0.34433413
## MHKOOP     MHKOOP  0.29551164
## MBERARBO MBERARBO  0.22410217
## MBERZELF MBERZELF  0.18600145
## MRELSA     MRELSA  0.18042416
## MAANTHUI MAANTHUI  0.05935146
## ALEVEN     ALEVEN  0.04163979
## MFALLEEN MFALLEEN  0.00000000
## MOPLLAAG MOPLLAAG  0.00000000
## PWABEDR   PWABEDR  0.00000000
## PWALAND   PWALAND  0.00000000
## PBESAUT   PBESAUT  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
## 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
## AZEILPL   AZEILPL  0.00000000
## APLEZIER APLEZIER  0.00000000
## AFIETS     AFIETS  0.00000000
## AINBOED   AINBOED  0.00000000
## ABYSTAND ABYSTAND  0.00000000
```

```
# Most important predictor: PPERSAUT, MKOOPKLA, MOPLHOOG
```

```
## (c)

pro = predict(boost_model, newdata = test_set, n.trees = 1000, type = "response")

dim(test_set) # 4822 87
```

```
## [1] 4822   87
```

```
pred = rep("No, 4822")
pred[pro>0.20] = "Yes"

act = test_set$Purchase
table(act, pred)
```

```
##      pred
## act   No, 4822 Yes
##   No         1 108
##   Yes        0  31
```