Wentao Zhu

Homework 2

July 17, 2018

1 Chapter 4, Exercise 4 (p. 168)

1.1 a

$$(0.95 - 0.05) * 0.1 + 2\int_0^{0.05}(x + 0.05)dx = 0.09 + 0.0075 = 0.0975$$

(If we ignore $X < 0.05$ and $X > 0.95$, the answer should be 0.1)

1.2 b

Ignoring corner cases, the answer is 0.01.

1.3 c

Ignoring corner cases, the answer is $0.1^{100} = 10^{-100}$

1.4 d

When p is large (compared to n), n observations would be very sparse in the p-dimension space, making KNN unreliable.

1.5 e

Supppose the length is l, then

$$l^p = 0.1$$
$$l = 0.1^{\frac{1}{p}}$$

- $p = 1, l = 0.100$
- $p = 2, l = 0.316$
- $p = 100, l = 0.977$

As p grows larger, l converges to 1, which means almost the whole space is needed to make a reasonable prediction.

2 Chapter 4, Exercise 6 (p. 170)

2.1 a

$$P(Y=1|X_1,X_2) = \frac{e\beta_0+\beta_1X_1+\beta_2X_2}{1+e\beta_0+\beta_1X_1+\beta_2X_2} = 0.3775$$

In [1]:
```
x1 = 40
x2 = 3.5
b0 = -6
b1 = 0.05
b2 = 1
p = exp(b0+b1*x1+b2*x2)/(1+exp(b0+b1*x1+b2*x2))
p
```

0.377540668798145

2.2 b

Solve the equation about X_1:

$$P(Y=1|X_1,X_2) = \frac{e\beta_0+\beta_1X_1+\beta_2X_2}{1+e\beta_0+\beta_1X_1+\beta_2X_2} = 0.5$$

That is,

$$log[\frac{P(Y=1|X_1,X_2)}{P(Y=0|X_1,X_2)}] = \beta_0+\beta_1X_1+\beta_2X_2 = 0$$

We have $X_1 = 50$.

In [2]:
```
x2 = 3.5
b0 = -6
b1 = 0.05
b2 = 1
x1 = (-b2*x2-b0)/b1
x1
```

50

3 Chapter 4, Exercise 8 (p. 170)

Performing 1-nearest neighbors on training set always produces correct result. Therefore the test errror rate of 1-nearest neighbors = 18%/0.5 $=$ 36%. The error rate of logistic regression is 36%, so it is better for this case.

4 Chapter 4, Exercise 10 (p. 171)

In [3]:
```
library(ISLR)
data(Weekly)
```

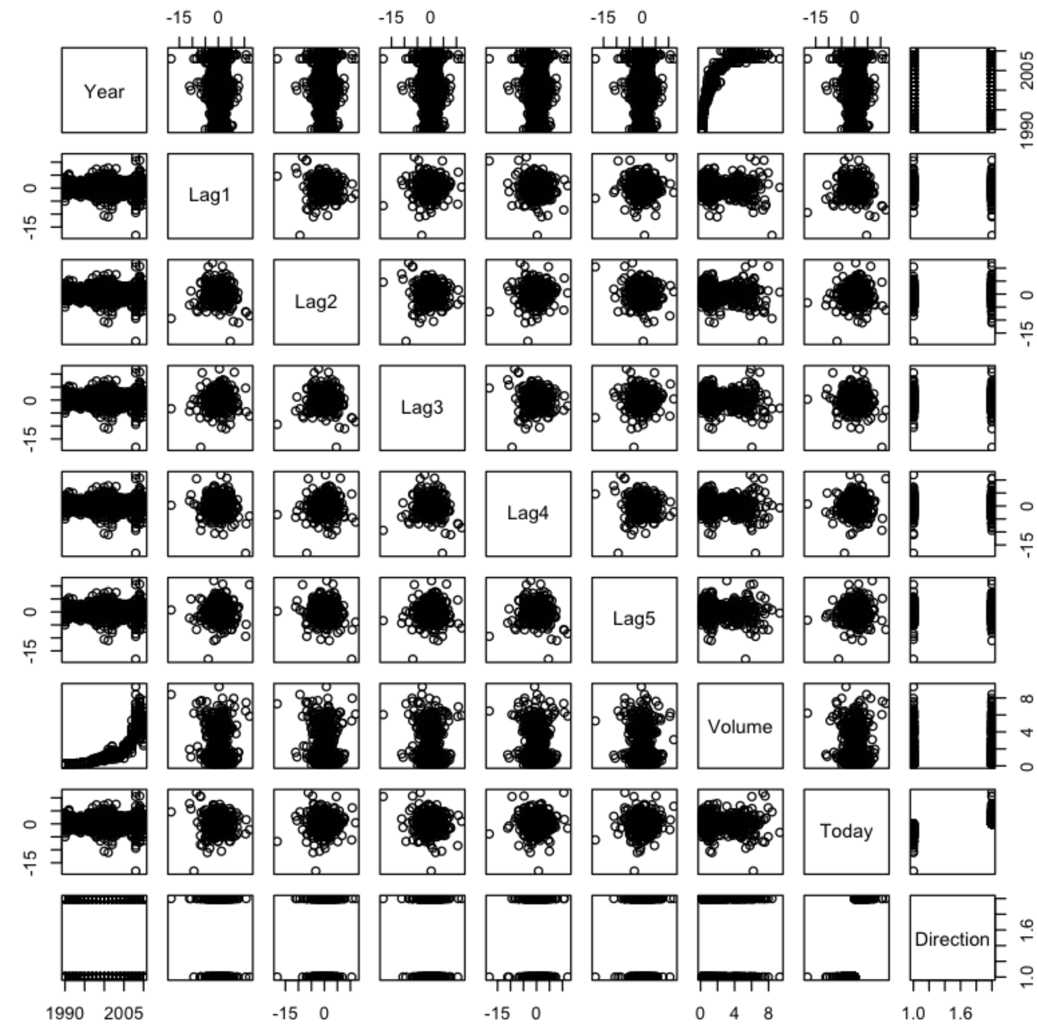
```
Warning message:
package ISLR was built under R version 3.4.2
```

4.1 a

```
In [4]: summary(Weekly)
        pairs(Weekly)
        round(cor(Weekly[,-9]), 3)
```

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

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Year	1.000	-0.032	-0.033	-0.030	-0.031	-0.031	0.842	-0.032
Lag1	-0.032	1.000	-0.075	0.059	-0.071	-0.008	-0.065	-0.075
Lag2	-0.033	-0.075	1.000	-0.076	0.058	-0.072	-0.086	0.059
Lag3	-0.030	0.059	-0.076	1.000	-0.075	0.061	-0.069	-0.071
Lag4	-0.031	-0.071	0.058	-0.075	1.000	-0.076	-0.061	-0.008
Lag5	-0.031	-0.008	-0.072	0.061	-0.076	1.000	-0.059	0.011
Volume	0.842	-0.065	-0.086	-0.069	-0.061	-0.059	1.000	-0.033
Today	-0.032	-0.075	0.059	-0.071	-0.008	0.011	-0.033	1.000



Year and Volume could have a positive correlation.

4.2 b

```
In [5]: fit1 = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly,
            family = binomial)
        summary(fit1)
```

```
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Deviance Residuals:
```

```
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

Lag2 has more statistical significance.

4.3 c

In [6]:
```
attach(Weekly)
threshold = 0.5
fit1.ans = predict(fit1, type = "response")
fit1.preds = ifelse(fit1.ans > 0.5, "Up", "Down")
table(fit1.preds, Direction)
mean(fit1.preds == Direction)
```

```
          Direction
fit1.preds Down  Up
      Down   54  48
      Up    430 557
```

0.561065197428834
Overall fraction of correct predictions = 56.1%.
Most incorrect predictions mistake `Down` for `Up` ($\frac{430}{430+48} = 90.0\%$).

In [7]:
```
train = (Year <= 2008)
train_data = Weekly[train,]
test_data = Weekly[!train,]
```

```
fit2 = glm(Direction ~ Lag2, data = train_data, family = binomial)
fit2.ans = predict(fit2, test_data, type = "response")
fit2.preds = ifelse(fit2.ans > 0.5, "Up", "Down")
table(fit2.preds, test_data$Direction)
mean(fit2.preds == test_data$Direction)
detach(Weekly)
```

```
fit2.preds Down Up
      Down    9  5
      Up     34 56
```

0.625
Overall fraction of correct predictions for the held out data = 62.5%.

5 Chapter 5, Exercise 5 (p. 198)

```
In [8]: library(ISLR)
        attach(Default)
        # summary(Default)
```

5.1 a

```
In [9]: fit1 = glm(default ~ income + balance, data = Default, family = binomial)
        summary(fit1)
```

```
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
```

```
AIC: 1585

Number of Fisher Scoring iterations: 8
```

5.2 b

```
In [10]: TrainAndEvaluate = function(train_ratio) {
             set.seed(2)
             # i.
             train = sample(nrow(Default), train_ratio*nrow(Default))
             train_data = Default[train,]
             validate_data = Default[-train,]
             # ii.
             fit = glm(default ~ income + balance, data = train_data, family = binomial)
             # iii.
             fit.ans = predict(fit, validate_data, type = "response")
             fit.preds = ifelse(fit.ans > 0.5, "Yes", "No")
             # iv.
             return(mean(fit.preds != validate_data$default))
         }

         TrainAndEvaluate(0.8)
```

0.0215

5.3 c

```
In [11]: TrainAndEvaluate(0.5)
         TrainAndEvaluate(0.6)
         TrainAndEvaluate(0.7)
```

0.0276
0.02725
0.024
As the ratio of training data increases, the validation error generally reduces.

5.4 d

```
In [12]: TrainAndEvaluate = function(train_ratio) {
             set.seed(2)
             # i.
             train = sample(nrow(Default), train_ratio*nrow(Default))
             train_data = Default[train,]
             validate_data = Default[-train,]
             # ii.
             fit = glm(default ~ income + student + balance, data = train_data,
                       family = binomial)
```

```
        # iii.
        fit.ans = predict(fit, validate_data, type = "response")
        fit.preds = ifelse(fit.ans > 0.5, "Yes", "No")
        # iv.
        return(mean(fit.preds != validate_data$default))
    }

    TrainAndEvaluate(0.5)
    TrainAndEvaluate(0.6)
    TrainAndEvaluate(0.7)
    TrainAndEvaluate(0.8)
```

0.0286
0.02825
0.025
0.022
Including a dummy variable for student does not lead to a reduction in the test error rate.

6 Chapter 5, Exercise 6 (p. 199)

6.1 a

In [13]:
```
fit1 = glm(default ~ income + balance, data = Default, family = binomial)
summary(fit1)
```

```
Call:
glm(formula = default ~ income + balance, family = binomial,
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585
```

```
Number of Fisher Scoring iterations: 8
```

Estimated standard error: $4.985e-06$ for `income`, $2.274e-04$ for `balance`

6.2 b

```
In [14]: boot.fn = function(data, index) {
             return(coef(glm(default ~ income + balance,
             data = data, family = binomial, subset = index)))
         }
```

6.3 c

```
In [15]: library(boot)
         set.seed(1)
         boot(Default, boot.fn, 50)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 50)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01  1.181200e-01 4.202402e-01
t2*  2.080898e-05 -5.466926e-08 4.542214e-06
t3*  5.647103e-03 -6.974834e-05 2.282819e-04
```

6.4 d

There are no significant difference between the estimated standard errors produced by two methods.

7 Chapter 5, Exercise 8 (p. 200)

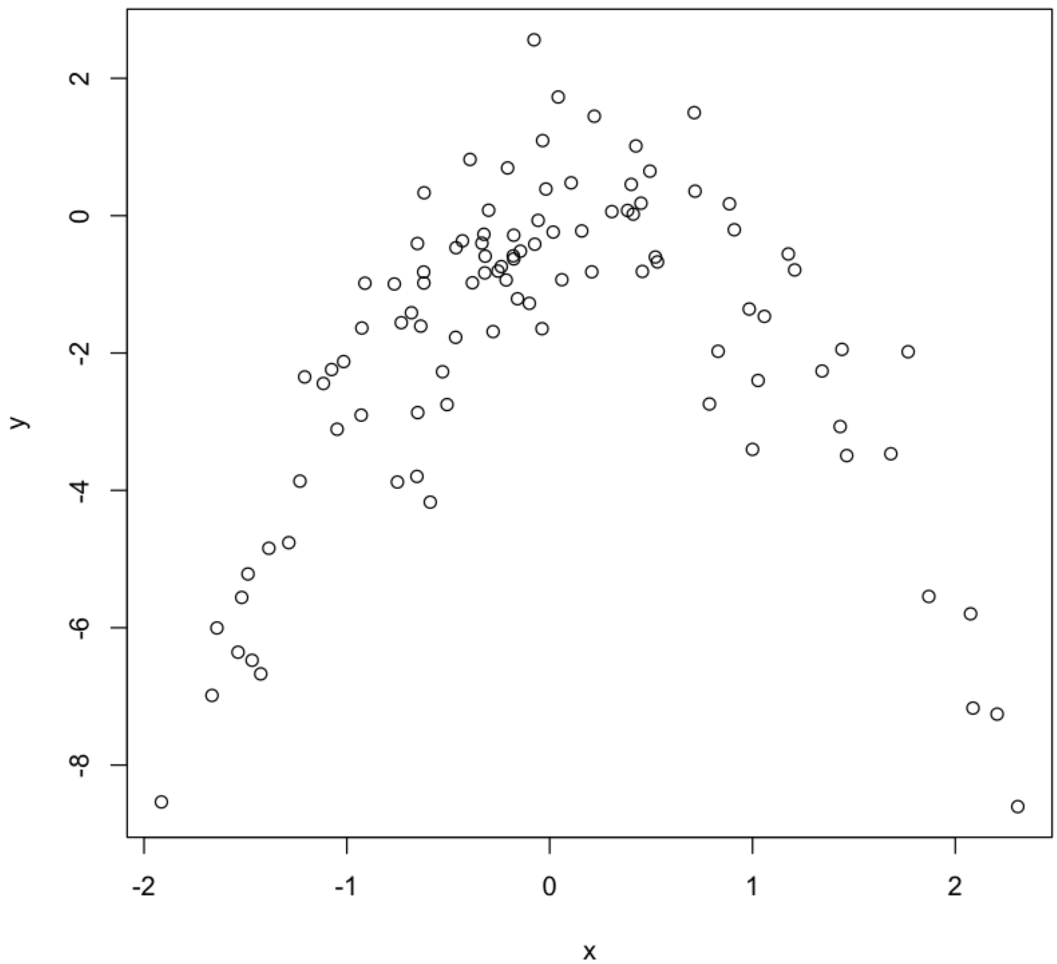
7.1 a

```
In [16]: set.seed (1)
         y=rnorm(100)
         x=rnorm(100)
         y=x-2*x^2+ rnorm(100)
```

- $n = 100$ (observations)

- $p = 2$ (predictors)
- $Y = -2X^2 + X + \epsilon$ (model)

```
In [17]: plot(x,y)
```



Quadratic relationship between X and Y with noises.

7.2 c

```
In [18]: polyfits = function() {
             Data = data.frame(x, y)
             # i.
             glm.fit1 = glm(y ~ x)
             print(cv.glm(Data, glm.fit1)$delta)
```

```
    # ii.
    glm.fit2 = glm(y ~ poly(x, 2, raw=TRUE))
    print(cv.glm(Data, glm.fit2)$delta)
    # iii.
    glm.fit3 = glm(y ~ poly(x, 3, raw=TRUE))
    print(cv.glm(Data, glm.fit3)$delta)
    # iv.
    glm.fit4 = glm(y ~ poly(x, 4, raw=TRUE))
    print(cv.glm(Data, glm.fit4)$delta)
}

set.seed(1)
polyfits()
```

```
[1] 5.890979 5.888812
[1] 1.086596 1.086326
[1] 1.102585 1.102227
[1] 1.114772 1.114334
```

```
In [19]: set.seed(233)
         polyfits()
```

```
[1] 5.890979 5.888812
[1] 1.086596 1.086326
[1] 1.102585 1.102227
[1] 1.114772 1.114334
```

Resuts are same, because LOOCV predicts n observations separately using rest of the data, therefore no randomness is introduced.

7.3 e

The quadratic model has the smallest error, as we expected. The data are generated using quadratic model, so the model should perform better than others.

7.4 f

```
In [20]: glm.fit1 = glm(y ~ x)
         glm.fit2 = glm(y ~ poly(x, 2, raw=TRUE))
         glm.fit3 = glm(y ~ poly(x, 3, raw=TRUE))
         glm.fit4 = glm(y ~ poly(x, 4, raw=TRUE))
         summary(glm.fit1)
         summary(glm.fit2)
         summary(glm.fit3)
         summary(glm.fit4)
```

```
Call:
glm(formula = y ~ x)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-7.3469  -0.9275   0.8028   1.5608   4.3974

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.8185     0.2364  -7.692 1.14e-11 ***
x             0.2430     0.2479   0.981    0.329
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for gaussian family taken to be 5.580018)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 546.84  on 98  degrees of freedom
AIC: 459.69

Number of Fisher Scoring iterations: 2
```

```
Call:
glm(formula = y ~ poly(x, 2, raw = TRUE))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.89884  -0.53765   0.04135   0.61490   2.73607

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             -0.09544    0.13345  -0.715    0.476
poly(x, 2, raw = TRUE)1  0.89961    0.11300   7.961 3.24e-12 ***
poly(x, 2, raw = TRUE)2 -1.86665    0.09151 -20.399  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for gaussian family taken to be 1.06575)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 103.38  on 97  degrees of freedom
AIC: 295.11

Number of Fisher Scoring iterations: 2
```

```
Call:
glm(formula = y ~ poly(x, 3, raw = TRUE))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.87250  -0.53881   0.02862   0.59383   2.74350

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             -0.09865    0.13453  -0.733    0.465
poly(x, 3, raw = TRUE)1  0.95551    0.22150   4.314  3.9e-05 ***
poly(x, 3, raw = TRUE)2 -1.85303    0.10296 -17.998  < 2e-16 ***
poly(x, 3, raw = TRUE)3 -0.02479    0.08435  -0.294    0.769
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for gaussian family taken to be 1.075883)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 103.28  on 96  degrees of freedom
AIC: 297.02

Number of Fisher Scoring iterations: 2



Call:
glm(formula = y ~ poly(x, 4, raw = TRUE))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8914  -0.5244   0.0749   0.5932   2.7796

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             -0.13897    0.15973  -0.870 0.386455
poly(x, 4, raw = TRUE)1  0.90980    0.24249   3.752 0.000302 ***
poly(x, 4, raw = TRUE)2 -1.72802    0.28379  -6.089  2.4e-08 ***
poly(x, 4, raw = TRUE)3  0.00715    0.10832   0.066 0.947510
poly(x, 4, raw = TRUE)4 -0.03807    0.08049  -0.473 0.637291
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

```
(Dispersion parameter for gaussian family taken to be 1.084654)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 103.04  on 95  degrees of freedom
AIC: 298.78

Number of Fisher Scoring iterations: 2
```

Linear and quadratic terms have statistical significance, as shown by the p-values. The summary results agree with the CV results.

8 Chapter 5, Exercise 9

```
In [21]: library(MASS)
         summary(Boston)
         attach(Boston)
```

```
      crim                zn             indus            chas
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
 1st Qu.: 0.08204   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
```

8.1 a

```
In [22]: medv.mean = mean(medv)
         medv.mean
```

22.5328063241107

8.2 b

```
In [23]: medv.err = sd(medv)/sqrt(nrow(Boston))
         medv.err
```

0.408861147497535

8.3 c

```
In [24]: set.seed(1)
         library(boot)
         boot.fn = function(data, index){
           return(mean(data[index]))
         }
         medv.bstrap = boot(medv, boot.fn, 1000)
         medv.bstrap
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original      bias    std. error
t1* 22.53281 0.008517589   0.4119374
```

The two results are very close (~0.7% diffrence).

8.4 d

```
In [25]: c(medv.bstrap$t0 - sd(medv.bstrap$t) * 2, medv.bstrap$t0 + sd(medv.bstrap$t) * 2)
         t.test(Boston$medv)
```

1. 21.708931444342 2. 23.3566812038793

```
One Sample t-test

data:  Boston$medv
```

```
t = 55.111, df = 505, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.72953 23.33608
sample estimates:
mean of x
 22.53281
```

The two results are very close (~0.08% diffrence).

8.5 e

```
In [26]: medv.med = median(medv)
         medv.med
```

21.2

8.6 f

```
In [27]: set.seed(1)
         library(boot)
         boot.fn.med = function(data, index){
           return(median(data[index]))
         }
         med.bstrap = boot(medv, boot.fn.med, 1000)
         med.bstrap
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn.med, R = 1000)


Bootstrap Statistics :
    original   bias    std. error
t1*     21.2 -0.01615   0.3801002
```

We get a result with reasonably small standard error.

8.7 g

```
In [28]: medv.tp = quantile(medv, 0.1)
         medv.tp
```

10\%: 12.75

8.8 h

```
In [29]: set.seed(1)
         library(boot)
         boot.fn.tentp = function(data, index){
           return(quantile(data[index], 0.1))
         }
         tentp.bstrap = boot(medv, boot.fn.tentp, 1000)
         tentp.bstrap
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn.tentp, R = 1000)


Bootstrap Statistics :
    original  bias    std. error
t1*    12.75 0.01005    0.505056
```

The standard error of estimating tenth percentile is larger then that of estimating median.