HW2 Tianchun Jiang

SUID ending in 0710

tcjiang108@gmail.com

Load necessary libraries

Problem 1

Chapter 4, Exercise 4 (p. 168).

a)

For the cases where 0.5 < X < 0.95, the average will be 10%. Otherwise, we form an integral as such:

$$\int_0^{0.05} 100x + 5dx$$

Which equals 0.375, multiplied by 2 for two intervals: when x < 0.05 and when x > 0.95 Thus on average our prediction is (0.1 * 0.9 + 0.00375 * 2) * 100 = 9.75%

b)

0.0975^2 * 100 = 0.95%

c)

0.0975^100 * 100 = (7.95 e-100)%

d)

Our results show that as dimensionality increases, the number of datapoints that are close in all dimensions to the response variable decreases exponentially.

e)

The length will be 0.1^(1/p) e.g. for p=1, l = 0.1^(1) = 0.1 for p=100, l = 0.1^(1/100)

Problem 2

Chapter 4, Exercise 6 (p. 170).

a)

$$\frac{e^{B_0+B1X_1+...+B_pX_p}}{1+e^{B_0+B_1X_1+...+B_pX_p}}\frac{e^{-6+0.05*40+3.5}}{1+e^{-6+0.05*40+3.5}} = 0.3775$$

b)

$$0.5 = \frac{e^{-6+0.05*X_1+3.5}}{1+e^{-6+0.05*X_1+3.5}}$$
$$0.5*(1+e^{0.05*X_1-2.5}) = e^{0.05*X_1-2.5}$$
$$0.5 = 0.5e^{0.05*X_1-2.5}$$
$$log(1) = log(e^{0.05*X_1-2.5})$$
$$0 = 0.05*X_1 - 2.5$$
$$X_1 = 2.5/0.05 = 50 hours$$

The student in part a) needs to study 50 hours to have a 50% chance of earning an A

Problem 3

Chapter 4, Exercise 8 (p. 170).

With K=1, the decision boundary is highly flexible. In general, as flexibility increases, the training error will decline but the testing error will increase. A KNN classifier with K=1 has a training error of 0, as every observation will simply cluster with itself. This means that the test error was 36%.

Thus, I would choose the 30% test error logistic regression classifier.

Problem 4

Chapter 4, Exercise 10 (p. 171). In part (i), please be concise; only describe and provide the output of your best prediction. Updated: only a, b, c, d are required

a)

Quick look at correlation matrix

```
names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
dim(Weekly)
```

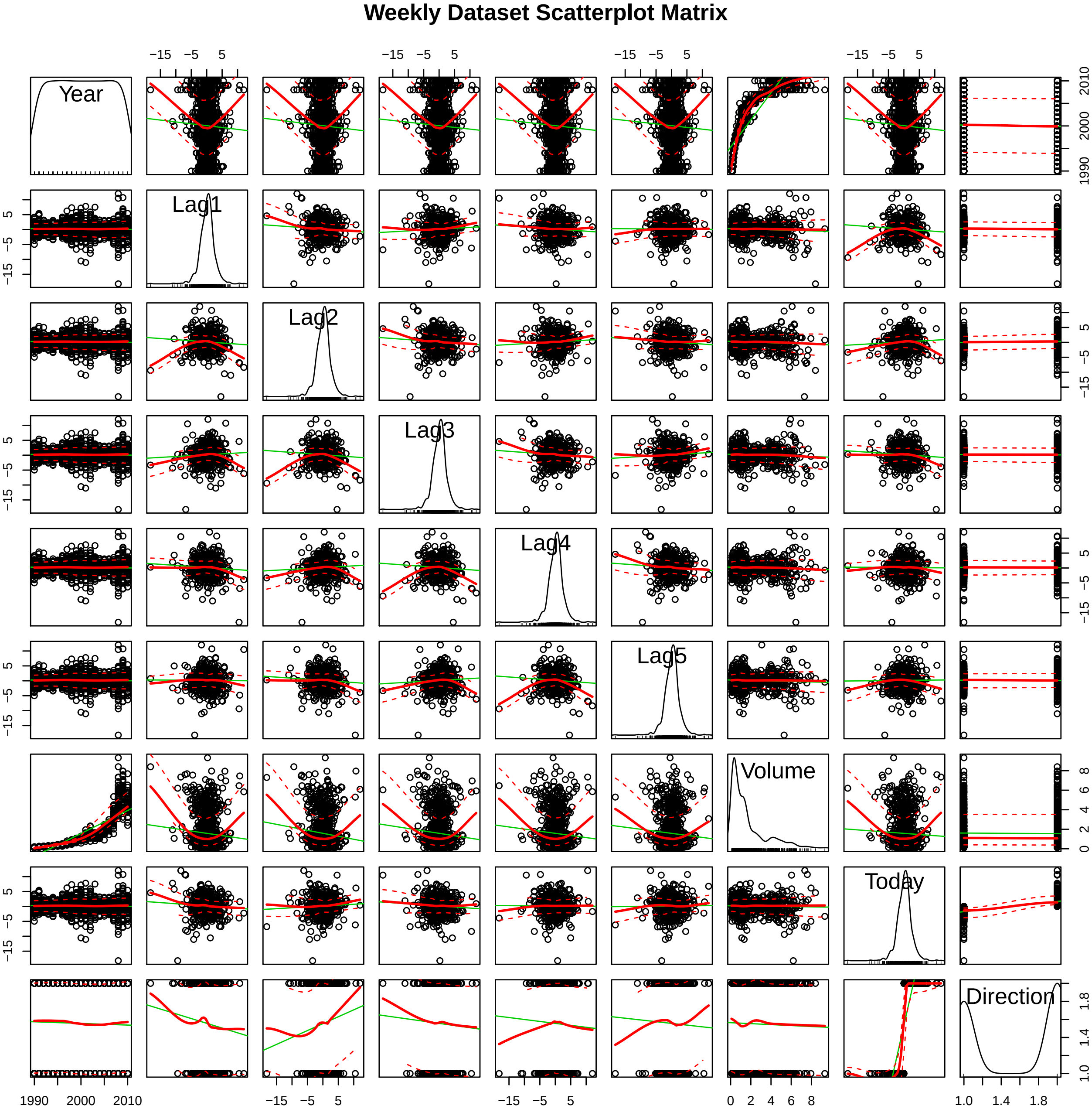
```
## [1] 1089    9
```

```
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
```

```
## Mean   :  0.1458   Mean   :  0.1399   Mean    :1.57462
## 3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
## Max.   : 12.0260   Max.   : 12.0260   Max.    :9.32821
##      Today          Direction
## Min.    :-18.1950   Down:484
## 1st Qu.: -1.1540    Up  :605
## Median :  0.2410
## Mean   :  0.1499
## 3rd Qu.:  1.4050
## Max.   : 12.0260
```

```r
scatterplotMatrix(Weekly, main="Weekly Dataset Scatterplot Matrix")
```



```r
Weekly_numeric <- Weekly[, sapply(Weekly, is.numeric)]
cor(Weekly_numeric)
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

Besides the 'year' and 'volume' features, there does not appear to be significant correlation between variables. In other words, volume of shares traded weekly increased between 1990 and 2010.

```r
m1<-lm(Year~Volume, data=Weekly)
summary(m1)
```

```
##
## Call:
## lm(formula = Year ~ Volume, data = Weekly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.3999  -2.2495   0.5893   2.6037   7.4944
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 1.995e+03  1.350e-01 14775.21   <2e-16 ***
## Volume      3.012e+00  5.854e-02    51.45   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.257 on 1087 degrees of freedom
## Multiple R-squared:  0.7089, Adjusted R-squared:  0.7086
## F-statistic:  2647 on 1 and 1087 DF,  p-value: < 2.2e-16
```

b)

```r
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume , data=Weekly ,family=binomial)
summary(glm.fit)
```

```
##
## 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
```

Lag 2 appears to be statistically significant with a p value of 0.02, pointing to an association between Lag2 and direction

c)

```r
glm.probs <- predict(glm.fit, type="response")
contrasts(Weekly$Direction)
```

```
##      Up
## Down  0
## Up    1
```

```r
glm.pred=rep("Down", 1089)
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Weekly$Direction)
```

```
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```r
mean(glm.pred == Weekly$Direction)
```

```
## [1] 0.5610652
```

The confusion matrix tells us that the model correctly predicted the weekly market movement 56.1% of the time. Since we trained and tested the model using the same 1089 observations, 100-56.1=43.9 is the training error rate. Training error rate tends to underestimate test error rate.

d)

```r
train <- (Weekly$Year < 2008)
Weekly.2008 <- Weekly[!train,]
dim(Weekly.2008)
```

```
## [1] 156   9
```

```r
Direction.2008 <- Weekly$Direction[!train]
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly ,family=binomial,subset=train)
glm.probs <- predict(glm.fit, Weekly.2008, type="response")
glm.pred=rep("Down",156)
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Direction.2008)
```

```
##         Direction.2008
## glm.pred Down Up
##     Down   22 26
##     Up     50 58
```

```r
train_err <- mean(glm.pred==Direction.2008)
test_err <- 1-train_err
```

```r
sprintf('training error: %0.4f', train_err)
```

```
## [1] "training error: 0.5128"
```

```r
sprintf('test error: %0.4f', test_err)
```

```
## [1] "test error: 0.4872"
```

The fraction of correct predictions in our test set of observations between 2009 and 2010 was 48.72%, worse than random guessing.

Problem 5

Chapter 5, Exercise 5 (p. 198).

a)

```r
set.seed(1) # seed interpretor's RNG for reproducibility of results
glm.fit <- glm(default~income+balance, data=Default, family=binomial)
summary(glm.fit)
```

```
##
## 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
```

b)

```r
# i. split data into training and validation set
train <- sample(dim(Default)[1], dim(Default)[1]/2)

# ii fit multiple logistic regression with only training set
glm.fit <- glm(default~income+balance, data=Default, family=binomial, subset=train)

# iii obtain prediction of default status
glm.probs <- predict(glm.fit, newdata=Default[-train, ], type="response")
glm.pred <- rep("No", length(glm.probs))
glm.pred[glm.probs > 0.5] <- "Yes"

# iv
sprintf("test error rate with validation set approach is %0.3f%%",
        100*mean(glm.pred != Default[-train,]$default))
```

```
## [1] "test error rate with validation set approach is 2.860%"
```

c)

```r
for(i in 1:3) {
  train <- sample(dim(Default)[1], dim(Default)[1]/2)
  glm.fit <- glm(default~income+balance, data=Default, family=binomial, subset=train)
  glm.probs <- predict(glm.fit, newdata=Default[-train, ], type="response")
  glm.pred <- rep("No", length(glm.probs))
  glm.pred[glm.probs > 0.5] <- "Yes"
  tmpstr <- sprintf("run %d, test error rate with validation set approach is %0.3f%%", i
                    , 100*mean(glm.pred != Default[-train,]$default))
  print(tmpstr)
}
```

```
## [1] "run 1, test error rate with validation set approach is 2.360%"
## [1] "run 2, test error rate with validation set approach is 2.800%"
## [1] "run 3, test error rate with validation set approach is 2.680%"
```

Here we see that there is some variance in test error rate depending on validation set, but average is around 2.6%

d)

```
train <- sample(dim(Default)[1], dim(Default)[1]/2)
glm.fit <- glm(default~student+income+balance, data=Default, family=binomial, subset=train)
glm.probs <- predict(glm.fit, newdata=Default[-train, ], type="response")
glm.pred <- rep("No", length(glm.probs))
glm.pred[glm.probs > 0.5] <- "Yes"
tmpstr <- sprintf("test error rate with dummy student variable is %0.3f%%", i,
                  100*mean(glm.pred != Default[-train,]$default))
print(tmpstr)
```

```
## [1] "test error rate with dummy student variable is 3.000%"
```

We've seen that there is some variance in the test error rate baesd on how the train set/validation set is sampled. However, the test error rate with a dummy student variable included is perhaps slightly lower.

Problem 6

Chapter 5, Exercise 6 (p. 199)

a)

```
set.seed(1) # seed interpretor's RNG for reproducibility of results
glm.fit <- glm(default~income+balance, data=Default, family=binomial)
summary(glm.fit)
```

```
##
## 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
```

b)

```r
boot.fn <- function(data,index) {
  return(coef(glm(default~income+balance,data=data,family="binomial",subset=index)))
}
```

c)

```r
boot(Default, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01 -8.008379e-03 4.239273e-01
## t2*  2.080898e-05  5.870933e-08 4.582525e-06
## t3*  5.647103e-03  2.299970e-06 2.267955e-04
```

d)

The results between bootstrap method and logistic regression are not significantly different

Problem 7

Chapter 5, Exercise 8 (p. 200)

a)

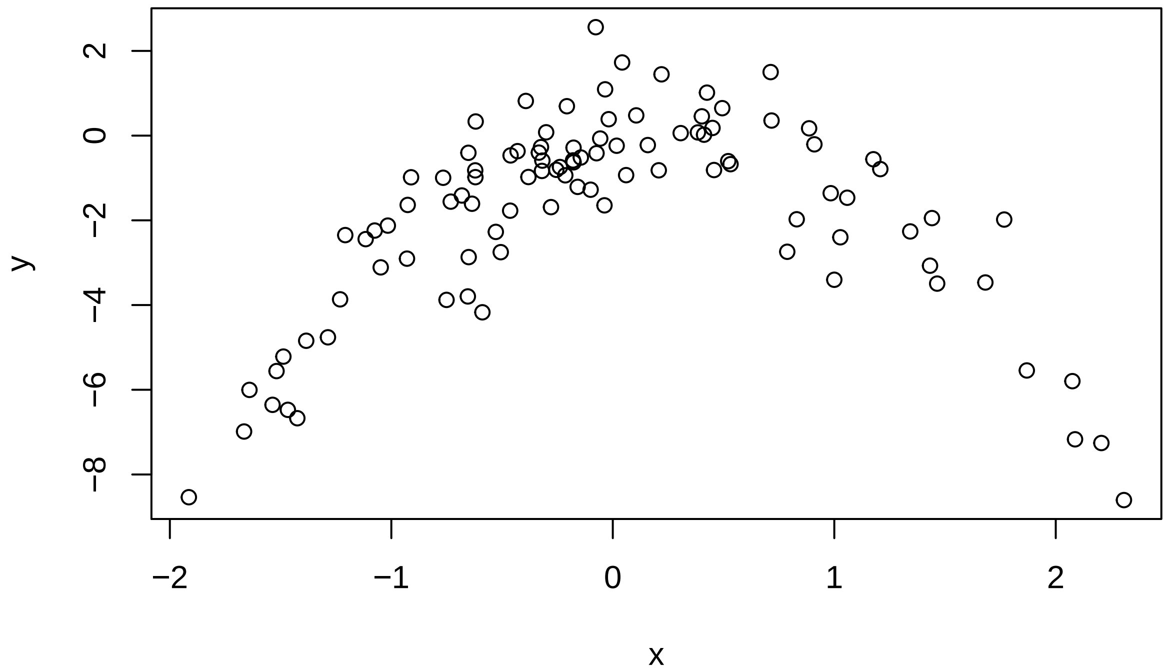
```r
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

n is 100 p is 2, x and x squared

$$Y = X - 2X^2 +$$

b)

```r
plot(x, y)
```



The relationship between y and x appears to be parabolic

c)

```
set.seed(1)
Data <- data.frame(x, y)

# i.
glm.fit.1 <- glm(y ~ x)
cv.glm(Data, glm.fit.1)$delta[1]
```

```
## [1] 5.890979
```

```
# ii.
glm.fit.2 = glm(y ~ poly(x, 2))
cv.glm(Data, glm.fit.2)$delta[1]
```

```
## [1] 1.086596
```

```
# iii.
glm.fit.3 = glm(y ~ poly(x, 3))
cv.glm(Data, glm.fit.3)$delta[1]
```

```
## [1] 1.102585
```

```
# iv.
glm.fit.4 = glm(y ~ poly(x, 4))
cv.glm(Data, glm.fit.4)$delta[1]
```

```
## [1] 1.114772
```

d)

```r
set.seed(2)
Data <- data.frame(x, y)
glm.fit.1 <- glm(y ~ x)
cv.glm(Data, glm.fit.1)$delta[1]
```

```
## [1] 5.890979
```

```r
glm.fit.2 = glm(y ~ poly(x, 2))
cv.glm(Data, glm.fit.2)$delta[1]
```

```
## [1] 1.086596
```

```r
glm.fit.3 = glm(y ~ poly(x, 3))
cv.glm(Data, glm.fit.3)$delta[1]
```

```
## [1] 1.102585
```

```r
glm.fit.4 = glm(y ~ poly(x, 4))
cv.glm(Data, glm.fit.4)$delta[1]
```

```
## [1] 1.114772
```

Results using a different random seed are identical. This makes sence because the above methodology does not have any probabilistic component of validation set generation.

e)

The second order polyfit yielded the smallest LOOCV error. This is what I expected due to the parabolic shape plotted in b)

f)

```r
summary(glm.fit.4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## 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)  -1.8277     0.1041 -17.549   <2e-16 ***
## poly(x, 4)1   2.3164     1.0415   2.224   0.0285 *
## poly(x, 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
## poly(x, 4)3  -0.3048     1.0415  -0.293   0.7704
## poly(x, 4)4  -0.4926     1.0415  -0.473   0.6373
## ---
## 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
```

Based on p-values, the second order term is very significant, which confirms my statement in e).

Problem 8

Chapter 5, Exercise 9 (p. 201)

a)

```r
sample_u <- mean(Boston$medv)
sample_u
```

```
## [1] 22.53281
```

b)

```r
sample_sde <- sd(Boston$medv) / sqrt(dim(Boston)[1])
sample_sde
```

```
## [1] 0.4088611
```

Roughly speaking, the standard error tells us the average amount this estimated mean differs from the actual mean. 'medv' is the median value of owner occupied homes in thousands of dollars. This means that on average the sample mean differs from the population mean by about $408.86. This is a relatively small error compared to the true mean.

c)

```r
set.seed(1)
boot.fn <- function(data, idx) {
  ret <- mean(data[idx])
  return (ret)
}
boot(Boston$medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.008517589   0.4119374
```

The result we got here using bootstrap is very close to result from b)

d)

```
l = sample_u - 2*sample_sde
u = sample_u + 2*sample_sde
l
```

```
## [1] 21.71508
```

```
u
```

```
## [1] 23.35053
```

```
t.test(Boston$medv)
```

```
##
##  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
```

Results calculated are very similar to those we get from t.test()

e)

```
med <- median(Boston$medv)
med
```

```
## [1] 21.2
```

f)

```
boot.fn <- function(data, idx) {
  ret = median(data[idx])
  return (ret)
}
boot(Boston$medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0098   0.3874004
```

The std error for median is small compared to the median value. Bootstrap was able to accurately calculate the median.

g)

```r
mu_hat_0.1 <- quantile(Boston$medv, 0.1)
mu_hat_0.1
```

```
##   10%
## 12.75
```

h)

```r
boot.fn <- function(data, idx){
  ret = quantile(data[idx], c(0.1))
  return (ret)
}

boot(Boston$medv, boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.00515   0.5113487
```

The std error for tenth percentile is small compared to the actual value.