Chapter 5 Exercises

From *An Introduction to Statistical Learning with Applications in R*

Jacob Zeiher

May 24, 2018

Contents

Conceptual

Problem 1

Let's first show briefly that $\text{Cov}(\alpha X, \beta Y) = \alpha\beta\text{Cov}(X, Y)$. By the definition of covariance, we have

$$\begin{aligned}
\text{Cov}(\alpha X, \beta Y) &= \text{E}[\alpha X \beta Y] - \text{E}[\alpha X]\text{E}[\beta Y] \\
&= \alpha\beta\text{E}[XY] - \alpha E[X]\beta\text{E}[Y] \\
&= \alpha\beta\text{E}[XY] - \alpha\beta\text{E}[X]\text{E}[Y] \\
&= \alpha\beta\left(\text{E}[XY] - \text{E}[X]\text{E}[Y]\right) \\
&= \alpha\beta\text{Cov}(X, Y).
\end{aligned}$$

Now we can derive equation 5.6. First note that by the properties of variance and covariance we have

$$\begin{aligned}
\text{Var}(\alpha X + (1-\alpha)Y) &= \text{Var}(\alpha X) + \text{Var}((1-\alpha)Y) + 2\text{Cov}(\alpha X, (1-\alpha)Y) \\
&= \alpha^2\text{Var}(X) + (1-\alpha)^2\text{Var}(Y) + 2\alpha(1-\alpha)\text{Cov}(X, Y) \\
&= \alpha^2\sigma_X^2 + (1 - 2\alpha + \alpha^2)\sigma_Y^2 + 2\alpha(1-\alpha)\sigma_{XY} \\
&= \alpha^2\sigma_X^2 + \sigma_Y^2 - 2\alpha\sigma_Y^2 + \alpha^2\sigma_Y^2 + 2\alpha\sigma_{XY} - 2\alpha^2\sigma_{XY} \\
&= \alpha^2(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) + \alpha(-2\sigma_Y^2 + 2\sigma_{XY}) + \sigma_Y^2.
\end{aligned}$$

Taking a derivative of this last equation with respect to α and setting it equal to zero we have

$$\begin{aligned}2\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) + (-2\sigma_Y^2 + 2\sigma_{XY}) &= 0\\ 2\alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) &= 2\sigma_Y^2 - 2\sigma_{XY}\\ \alpha(\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}) &= \sigma_Y^2 - \sigma_{XY}\\ \alpha &= \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}.\end{aligned}$$

The equation above is precisely equation 5.6. We could verify this value of α is a local minimum using the second derivative test, but we will not do so here.

Problem 2

Part a

Each observation from the original sample has an equal probability of being selected for each bootstrap observation since we are sampling with replacement. Since there are n observations in the original sample, the probability the j-th observation in the original sample is not the first observation in the bootstrap sample is given by

$$\frac{n-1}{n} = 1 - \frac{1}{n}.$$

Part b

Since we are sampling the original sample with replacement, this is the same probability as for part a. The probability neither the first nor the second bootstrap observation is the j-th original observation is given by:

$$\left(\frac{n-1}{n}\right)^2.$$

Part c

Since we are sampling the original sample with replacement, each bootstrap observation is selected independently from every other bootstrap observation. Hence, for each bootstrap observation, observation j in the original sample has a probability $\frac{n-1}{n}$ of not being selected. If we take a bootstrap sample with n observations, then observation j has probability

$$\left(\frac{n-1}{n}\right)^n = \left(1 - \frac{1}{n}\right)^n$$

of not being in the bootstrap sample.

Part d

The probability the jth observation is not in the sample is $\left(1 - \frac{1}{n}\right)^n$, so the probability the jth observation is in the sample is $1 - \left(1 - \frac{1}{n}\right)^n$. Hence, for $n = 5$, the probability the jth observation is in the sample is given by:

$$1 - (1 - \frac{1}{5})^5 = 1 - (\frac{4}{5})^5 = 0.67232.$$

Part e

Using the same logic as in part d, the probability the jth observation is in the sample with $n = 100$ is given by:

$$1 - (1 - \frac{1}{100})^{100} = 1 - (\frac{99}{100})^{100} \approx 0.633969.$$

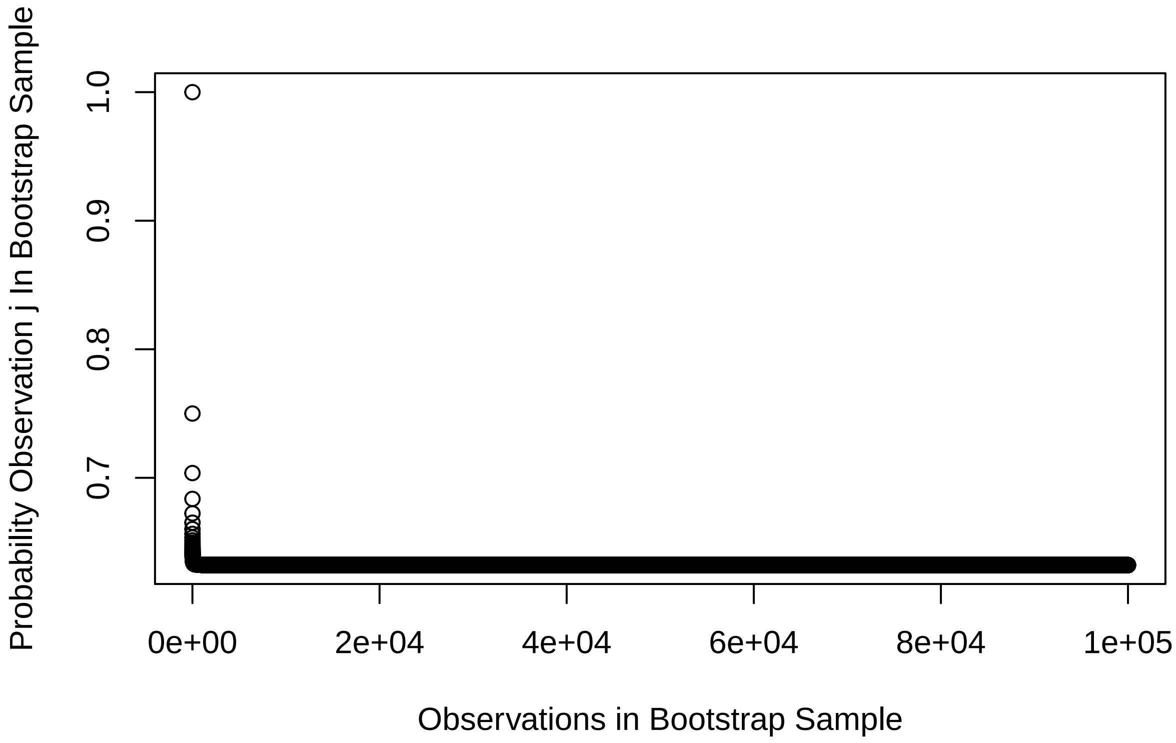
Part f

Using the same logic as in part d, the probability the jth observation is in the sample with $n = 10,000$ is given by:

$$1 - (1 - \frac{1}{100,00})^{10,000} = 1 - (\frac{9,999}{10,000})^{10,000} \approx 0.632139.$$

Part g

```r
probs <- rep(0,100000)
for (i in 1:100000){
  probs[i] <- 1 - ((i-1)/i)^i
}
plot(1:100000,probs, xlab="Observations in Bootstrap Sample",
     ylab="Probability Observation j In Bootstrap Sample")
```



The probability the jth observation is in the data set decreases monotonically as n increases. It converges rapidly to a lower bound somewhere around 0.63.

Part h

```r
store <- rep(NA,10000)
for (i in 1:10000) {
  store[i] <- sum(sample(1:100, rep=TRUE)==4)>0
}
mean(store)
```

```
## [1] 0.6326
```

This confirms empirically what we found from part g.

Problem 3

Part a

To implement k-fold cross-validation, first randomly divide/partition the data into k equally sized partitions. Fit the model on partitions $2, ..., k$, and use partition 1 as test data. Repeat this process using partitions 1 and $3, ..., k$ as training data and partition 2 as test data. Repeat for the remaining partitions. The estimated test error is the average test error of these k models.

Part b

The advantage of k-fold cross-validation (CV) relative to the validation set approach is it has lower bias. The disadvantage of k-fold CV relative to the validation set approach is k-fold CV requires more computation because we have to fit k different models.

The advantage of k-fold CV relative to LOOCV is k-fold CV requires less computation because we are only fitting k models instead of n, and k-fold CV has lower variance than LOOCV. The disadvantage of k-fold CV relative to LOOCV is k-fold CV has higher bias than LOOCV.

Problem 4

We could estimate the standard deviation of this prediction using a bootstrap method. Take a random observation of X, and use it to predict Y. Record this prediction of Y. Sample another observation of X with replacement, and use it to predict Y. Again, record this prediction of Y. Continue sampling observations of X with replacement, using this value to predict Y, and recording the results. Then just take the standard deviation of all these predictions.

Applied

Problem 5

Part a

```r
library(ISLR)
#Fit logistic regression
  glm.fit1 <- glm(default~income+balance, data=Default, family=binomial)
  summary(glm.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
```

Part b

```
#Partition the data
  set.seed(500)
  sample_size <- floor(0.80 * nrow(Default))
  train_ind <- sample(seq_len(nrow(Default)), size=sample_size, replace=FALSE)
  train <- Default[train_ind,]
  test <- Default[-train_ind,]
#Fit logitistic regression using the training data
  glm.fit2 <- glm(default~income+balance, data=train, family=binomial)
  glm.probs2 <- predict(glm.fit2, test, type="response")
  glm.class2 <- ifelse(glm.probs2>0.5,"Yes","No")
#Compute validation set error
  table(glm.class2, test$default)
```

```
##
## glm.class2   No  Yes
##        No  1917   52
##        Yes   11   20
```

```
  mean(glm.class2 != test$default)
```

```
## [1] 0.0315
```

Part c

```
#Partition the data
  set.seed(274)
  sample_size <- floor(0.80 * nrow(Default))
  train_ind <- sample(seq_len(nrow(Default)), size=sample_size, replace=FALSE)
  train <- Default[train_ind,]
  test <- Default[-train_ind,]
#Fit logitistic regression using the training data
  glm.fit3 <- glm(default~income+balance, data=train, family=binomial)
  glm.probs3 <- predict(glm.fit3, test, type="response")
  glm.class3 <- ifelse(glm.probs3>0.5,"Yes","No")
#Compute validation set error
  table(glm.class3, test$default)
```

```
##
## glm.class3   No  Yes
##        No  1932   43
##        Yes    8   17
```

```
  mean(glm.class3 != test$default)
```

```
## [1] 0.0255
```

```
#Partition the data
  set.seed(769)
  sample_size <- floor(0.80 * nrow(Default))
  train_ind <- sample(seq_len(nrow(Default)), size=sample_size, replace=FALSE)
  train <- Default[train_ind,]
  test <- Default[-train_ind,]
#Fit logitistic regression using the training data
  glm.fit4 <- glm(default~income+balance, data=train, family=binomial)
  glm.probs4 <- predict(glm.fit4, test, type="response")
  glm.class4 <- ifelse(glm.probs4>0.5,"Yes","No")
#Compute validation set error
  table(glm.class4, test$default)
```

```
##
## glm.class4   No  Yes
##        No  1941   33
##        Yes    8   18
```

```
  mean(glm.class4 != test$default)
```

```
## [1] 0.0205
```

```
#Partition the data
  set.seed(211)
  sample_size <- floor(0.80 * nrow(Default))
  train_ind <- sample(seq_len(nrow(Default)), size=sample_size, replace=FALSE)
  train <- Default[train_ind,]
  test <- Default[-train_ind,]
#Fit logitistic regression using the training data
  glm.fit5 <- glm(default~income+balance, data=train, family=binomial)
  glm.probs5 <- predict(glm.fit5, test, type="response")
  glm.class5 <- ifelse(glm.probs5>0.5,"Yes","No")
#Compute validation set error
  table(glm.class5, test$default)
```

```
##
## glm.class5   No  Yes
##        No  1938   40
##        Yes    4   18
```

```
  mean(glm.class5 != test$default)
```

```
## [1] 0.022
```

Part d

```
#Partition the data
  set.seed(7)
  sample_size <- floor(0.80 * nrow(Default))
  train_ind <- sample(seq_len(nrow(Default)), size=sample_size, replace=FALSE)
  train <- Default[train_ind,]
  test <- Default[-train_ind,]
#Fit the model with student added as a predictor
  glm.fit6 <- glm(default~income+balance+student, data=train, family=binomial)
  glm.probs6 <- predict(glm.fit6, test, type="response")
  glm.class6 <- ifelse(glm.probs6>0.5,"Yes","No")
  mean(glm.class6 != test$default)
```

```
## [1] 0.025
```

The new test error is 2.5%. It is hard, however to tell whether the test error is really decreased because we only have one validation set that was randomly selected. We either need to repeat this process several times or adopt another method to determine wether the test error decreases with the addition of "student" as a predictor.

Problem 6

Part a

```
rm(list=ls()) #Clear the workspace from last problem
library(ISLR)
set.seed(1)
#Fit logistic regression model
  glm.fit1 <- glm(default~income+balance, data=Default, family="binomial")
  summary(glm.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
```

Part b

```
#Write boot statistic function and test
  boot.fn <- function(data, index) {
    return(coef(glm(default~income+balance, data=data, family="binomial", subset=index)))
  }
  boot.fn(Default, 1:nrow(Default))
```

```
##   (Intercept)        income       balance
## -1.154047e+01  2.080898e-05  5.647103e-03
```

Part c

```
#Estimate coefficient standard error using bootstrap
  library(boot)
  boot(Default, boot.fn, R=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
```

Part d

The standard errors are roughly the same in both methods.

Problem 7

Part a

```r
rm(list=ls()) #Clear workspace from last problem
library(ISLR)
#Fit logistic regression model using all data
  logit.fit1 <- glm(Direction~Lag1+Lag2, data=Weekly, family="binomial")
  summary(logit.fit1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## 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: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

Part b

```r
contrasts(Weekly$Direction) #"Up" gets coded as a 1
```

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

```r
#Fit logistic regression excluding first observation
  logit.fit2 <- glm(Direction~Lag1+Lag2, data=Weekly[-1,], family="binomial")
  summary(logit.fit2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = "binomial", data = Weekly[-1,
##     ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6258  -1.2617   0.9999   1.0819   1.5071
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22324    0.06150   3.630 0.000283 ***
## Lag1        -0.03843    0.02622  -1.466 0.142683
## Lag2         0.06085    0.02656   2.291 0.021971 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1494.6  on 1087  degrees of freedom
## Residual deviance: 1486.5  on 1085  degrees of freedom
## AIC: 1492.5
##
## Number of Fisher Scoring iterations: 4
```

Part c

```
#Predict first observation using model fit with rest of data
  logit.prob2 <- predict(logit.fit2, Weekly[1,], type="response")
  logit.class2 <- ifelse(logit.prob2>0.5,"Up","Down")
  logit.class2
```

```
##    1
## "Up"
```

```
  (logit.class2 == Weekly[1,9])
```

```
##     1
## FALSE
```

Part d

```
#Write for loop to compute LOOCV
  errors=0
  for (i in 1:nrow(Weekly)) {
    logit <- glm(Direction~Lag1+Lag2, data=Weekly[-i,], family="binomial")
    logit.prob <- predict(logit, Weekly[i,], type="response")
    logit.class <- ifelse(logit.prob>0.5, "Up", "Down")
    if (logit.class != Weekly[i,9]) {
      errors = errors+1
    }
  }
```

Part e

```
#Calculate LOOCV estimate by dividing errors by n
  errors/nrow(Weekly)
```

```
## [1] 0.4499541
```

The LOOCV error estimate is 45%.

Problem 8

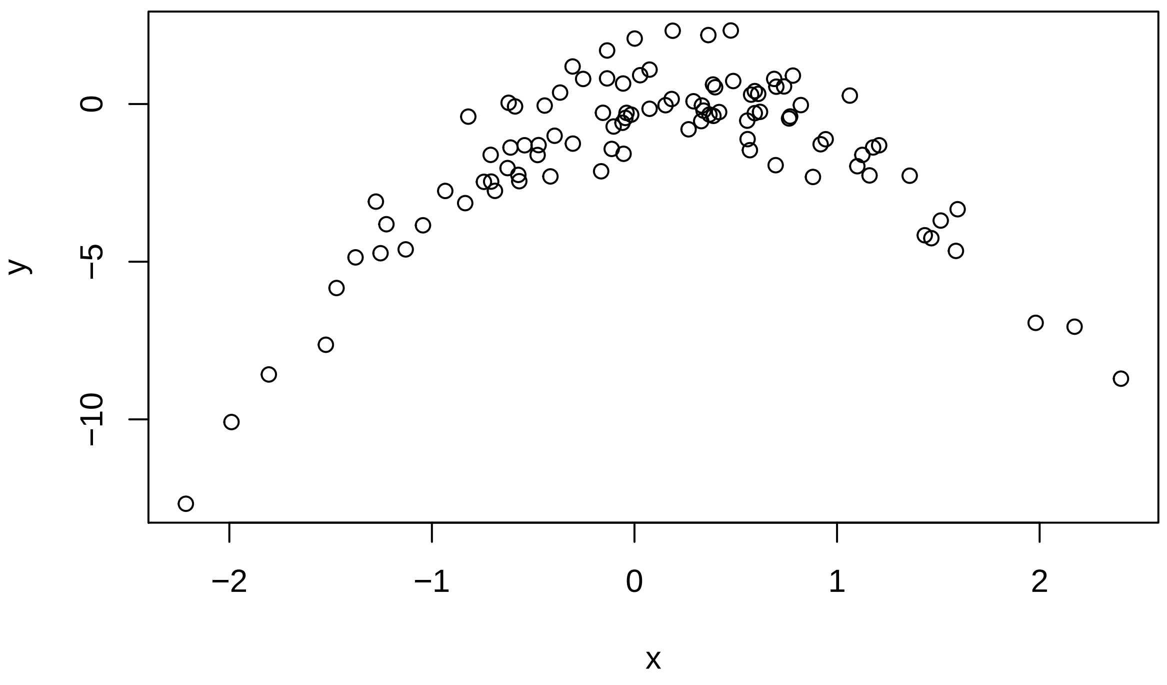
Part a

```
rm(list=ls()) #Clear workspace from previous problem
#Generate data
  set.seed(1)
  x <- rnorm(100)
  y <- x-2*x^2+rnorm(100)
```

In this model $n = 100$ and $p = 2$. The model has the form $y = x + 2x^2 + \epsilon$.

Part b

```
#Plot x vs y
  plot(x, y)
```



There is a quadratic relationship – an "inverted U-shaped" relationship – between x and y.

Part c

```
#Estimate
library(boot)
set.seed(123)
#Use for loop to compute LOOCV on several polynomial models
  cv.error <- rep(0,4)
  for (i in 1:4) {
    glm.fit <- glm(y ~ poly(x,i))
    cv.error[i] <- cv.glm(data.frame(x,y), glm.fit)$delta[1]
  }
  cv.error
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

```
  summary(glm(y ~ poly(x,4)))
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

Part d

```
#Repeat previous part with different random seed
  set.seed(246)
  cv.error2 <- rep(0,4)
  for (i in 1:4) {
    glm.fit <- glm(y ~ poly(x,i))
    cv.error2[i] <- cv.glm(data.frame(x,y), glm.fit)$delta[1]
  }
  cv.error2
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

The LOOCV test error estimate is the same in both cases because, no matter what, LOOCV will fit n models where a different observation is left out in each one. The test error is then just the average error across the models. There is nothing random about this, and our error estimate would only change if we re-generated the data with a different seed.

Part e

The model iv (the quintic polynomial) has the lowest LOOCV estimate. This is expected because more predictors does mean a lower training error, even if it is only a marginal improvement.

Part f

The LOOCV estimate for model iv might be the lowest, but the coefficients on powers of x stop being significant at x^3.

Problem 9

Part a

```r
rm(list=ls()) #Clear workspace from previous problem
set.seed(1)
library(MASS)
#Mean of medv
  mean(Boston$medv)
```

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

From the above we have that $\hat{\mu} = 22.53281$.

Part b

```r
#Estimate the standard error
  sd(Boston$medv)/sqrt(nrow(Boston))
```

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

Part c

```r
library(boot)
#Write function for mean
  mean.boot <- function(data, index){
    return(mean(data[index]))
  }
  mean.boot(Boston$medv, sample(nrow(Boston),nrow(Boston), replace=TRUE))
```

```
## [1] 22.78241
```

```r
#Estimate sd of mean using bootstrap
  boot.results1 <- boot(Boston$medv, mean.boot, R=1000)
```

Bootstrap standard deviation of 0.4127 with 1000 replications is very closed to 0.4089 from part c.

Part d

```r
lower.bound <- boot.results1$t0 - 2*sd(boot.results1$t)
lower.bound
```

```
## [1] 21.70734
```

```
upper.bound <- boot.results1$t0 + 2*sd(boot.results1$t)
upper.bound
```

```
## [1] 23.35827
```

```
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
```

The 95% confidence interval is given by $[21.70734, 23.35827]$. The confidence interval from the bootstrap is slightly larger than the confidence interval from t.test.

Part e

```
#Calculate median from the data
  median(Boston$medv)
```

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

Part f

```
#Write function for median
  median.boot <- function(data, index){
    return(median(data[index]))
  }
  median.boot(Boston$medv, sample(nrow(Boston),nrow(Boston), replace=TRUE))
```

```
## [1] 21.7
```

```
#Perform bootstrap with median
  boot(Boston$medv, median.boot, R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = median.boot, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     21.2 -0.01275   0.3865868
```

The standard error estimate for the median from the bootstrap is about 0.3865. This is smaller than the estimated standard error for the mean.

Part g

```r
#Compute tenth percentile for medv
  quantile(Boston$medv, 0.1)
```

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

Part h

```r
#Write function for 10th percentile
  quant.boot <- function(data, index){
    return(quantile(data[index], 0.1))
  }
#Perform bootstrap with 10th percentile
  quant.boot(Boston$medv, sample(nrow(Boston),nrow(Boston), replace=TRUE))
```

```
##   10%
## 13.35
```

```r
  boot(Boston$medv, quant.boot, R=1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = quant.boot, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    12.75 0.00345   0.5139794
```

The standard error estimate for the 10th percentile is about 0.514. This is higher than the standard deviation of mean or median.