Ch.5 Exercises: Resampling Methods

Conceptual

1.

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \qquad (5.6)$$

We have: $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(Y)$, $\sigma_{XY} = COV(XY)$

Using the following statistical property of variance:

$$Var(aX + bY) = a^2Var(X) + b^2Var(Y) + 2abCOV(XY)$$

We can show that α minimizes $Var(\alpha X + (1-\alpha)Y)$.

$$\begin{aligned} Var(aX + bY) &= a^2\sigma_X^2 + b^2\sigma_Y^2 + 2ab\sigma_{XY} \\ Var(\alpha X + (1-\alpha)Y) &= \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + 2(\alpha(1-\alpha))\sigma_{XY} \\ f &= \alpha^2\sigma_X^2 + \sigma_Y^2 - 2\alpha\sigma_Y^2 + \alpha^2\sigma_Y^2 + 2\alpha\sigma_{XY} - 2\alpha^2\sigma_{XY} \end{aligned}$$

Finding the critical points, simplifying and rearranging:

$$\begin{aligned} \frac{\partial(f)}{\partial\alpha} &= 0 \\ 2\alpha\sigma_X^2 - 2\sigma_Y^2 + 2\alpha\sigma_Y^2 + 2\sigma_{XY} - 4\alpha\sigma_{XY} &= 0 \\ \alpha &= \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}} \end{aligned}$$

2. (a)

When using random sampling with replacement to obtain a bootstrap sample from n observations:

If n=3, with observations: 1,2,3 we have:

Total permutations: $n^n = n^3 = 27$. Total times an observation will appear in a particular order(say as the first observation): $n^{n-1} = n^2 = 9$. So, the probability 1st bootstrap observation is the jth observation is: $1/n$.

Therefore, the probability the 1st bootstrap observation is not the jth observation is: $1 - 1/n$.

(b)

$1 - 1/n$

(c)

Total times an observation will not appear in a bootstrap sample: $(n-1)^n$.

Therefore, the probability that an observation is not in the bootstrap sample: $\frac{(n-1)^n}{n^n} = (1 - 1/n)^n$.

(d)

$p = 1 - (1 - 1/5)^5 = 0.672$

(e)

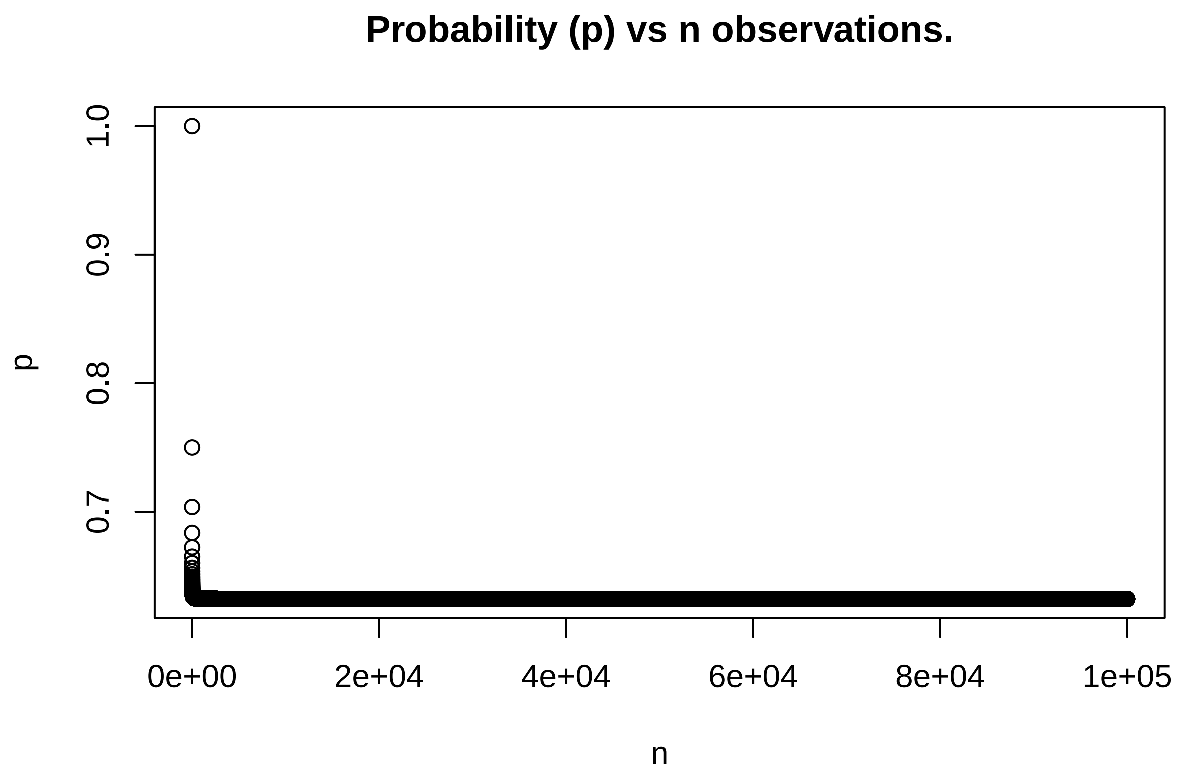
$p = 1 - (1 - 1/100)^{100} = 0.634$

(f)

$p = 1 - (1 - 1/10000)^{10000} = 0.632$

(g)

```r
n = 1:100000
p = 1-(1-1/n)^n
plot(n,p, main="Probability (p) vs n observations.")
```



- Probability decreases rapidly to an asymptote around 0.63 as n increases.

(h)

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

```
## [1] 0.6298
```

- The result is similar to the value calculated using the formula in 2(e).

3. (a)

In k-fold cross validation, a training set is randomly divided into k groups of equal size. The first group (or fold) is treated as the validation set, and the model is fit on the remaining k-1 groups (training set).The MSE is calculated using the validation set. This procedure is repeated k times and on each occasion the validation set and training sets will be different than the previous one. We then take the average of all the MSE's as the final MSE.

(b)

i. Less variance in test error estimate. A more accurate test error, as entire dataset is used.

ii. Computational advantage, as less computing resources required than LOOCV. If K=10 then only 10 models need to be fitted, unlike with LOOCV where n models need fitting. Higher bias than LOOCV, as fewer observations are used, but tends to have lower variance. A choice of K=5 or 10 gives a test error estimate that has neither high bias or variance, and so tends to give more accurate estimates in general.

(4)

- Create a function that outputs the response Y given X.
- Use bootstrapping to make multiple predictions of Y and the associated Standard Error(SE).
- Calculate SD by using : $SD = SE\sqrt{n}$

Applied

5. (a) (b)

```r
library(ISLR)
library(MASS)
```

```r
set.seed(1)

require(caTools)
```

```
## Loading required package: caTools
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```r
df = Default
sample_data = sample.split(df$default, SplitRatio = 0.50)
training.set = subset(df, sample_data==TRUE)
test.set = subset(df, sample_data==FALSE)
test.default = test.set$default

# Logistic regression model
lr.fit = glm(default ~ income+balance, data=training.set, family=binomial)

lr.probs = predict(lr.fit,test.set, type="response")
lr.preds = rep("No", length(test.set$default))
lr.preds[lr.probs>0.5] = "Yes"

table(lr.preds,test.default)
```

```
##          test.default
## lr.preds   No  Yes
##       No  4820  111
##       Yes   13   56
```

- Test error rate of 2.5%.

(c)

```r
# Assuming the question is referring to separate random splits rather than
# reducing or increasing the validation set.
set.seed(12)

sample_data = sample.split(df$default, SplitRatio = 0.50)
training.set = subset(df, sample_data==TRUE)
test.set = subset(df, sample_data==FALSE)
test.default = test.set$default

# Logistic regression model
lr.fit = glm(default ~ income+balance, data=training.set, family=binomial)

lr.probs = predict(lr.fit,test.set, type="response")
lr.preds = rep("No", length(test.set$default))
lr.preds[lr.probs>0.5] = "Yes"

table(lr.preds,test.default)
```

```
##          test.default
## lr.preds   No  Yes
##       No  4810  113
##       Yes   23   54
```

- Test error rate of 2.7%

```r
set.seed(123)
sample_data = sample.split(df$default, SplitRatio = 0.50)
training.set = subset(df, sample_data==TRUE)
test.set = subset(df, sample_data==FALSE)
test.default = test.set$default

# Logistic regression model
lr.fit = glm(default ~ income+balance, data=training.set, family=binomial)

lr.probs = predict(lr.fit,test.set, type="response")
lr.preds = rep("No", length(test.set$default))
lr.preds[lr.probs>0.5] = "Yes"

table(lr.preds,test.default)
```

```
##          test.default
## lr.preds   No  Yes
##       No  4818  106
##       Yes   15   61
```

- Test error rate of 2.4%

```r
set.seed(1234)
sample_data = sample.split(df$default, SplitRatio = 0.50)
training.set = subset(df, sample_data==TRUE)
test.set = subset(df, sample_data==FALSE)
test.default = test.set$default

# Logistic regression model
lr.fit = glm(default ~ income+balance, data=training.set, family=binomial)

lr.probs = predict(lr.fit,test.set, type="response")
lr.preds = rep("No", length(test.set$default))
lr.preds[lr.probs>0.5] = "Yes"

table(lr.preds,test.default)
```

```
##          test.default
## lr.preds   No  Yes
##       No  4807  110
##       Yes   26   57
```

- Test error rate of 2.7%
- The results are similar to each other (low variance). The minor differences can be explained by the fact that we used separate observations for each model.

(d)

```r
# Logistic regression model, dummy variable used for student.
lr.fit = glm(default ~ income+balance+student, data=training.set, family=binomial)

lr.probs = predict(lr.fit,test.set, type="response")
lr.preds = rep("No", length(test.set$default))
lr.preds[lr.probs>0.5] = "Yes"

contrasts(Default$student)
```

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

```r
table(lr.preds,test.default)
```

```
##          test.default
## lr.preds   No  Yes
##       No  4809  111
##       Yes   24   56
```

- Test error of 2.7%, which is unchanged.

6. (a) (b)

```r
set.seed(111)
# Logistic regression model
lr.fit2 = glm(default ~ income+balance, data=df, family=binomial)
summary(lr.fit2)$coefficients[2:3,2]
```

```
##       income      balance 
## 4.985167e-06 2.273731e-04
```

```r
boot.fn = function(data, index){
  default = data$default[index]
  income = data$income[index]
  balance = data$balance[index]
  lr.fit2 = glm(default ~ income + balance, family = binomial)
  return(summary(lr.fit2)$coefficients[2:3,2])
}
```

```r
boot.fn(df,1:length(df$default))
```

```
##       income      balance 
## 4.985167e-06 2.273731e-04
```

(c)

```r
# Using boot() with R=100.
library(boot)
boot(df,boot.fn,100)
```

```
## 
## ORDINARY NONPARAMETRIC BOOTSTRAP
## 
## 
## Call:
## boot(data = df, statistic = boot.fn, R = 100)
## 
## 
## Bootstrap Statistics :
##         original         bias     std. error
## t1* 4.985167e-06 2.169422e-08 1.506607e-07
## t2* 2.273731e-04 4.540846e-07 1.084125e-05
```

(d)

- The standard errors are slightly lower for both income and balance coefficients. In this case, using bootstrapping reduces the standard error for coefficient estimates.

7. (a) (b)

```r
set.seed(222)
lr.fit3 = glm(Direction ~ Lag1+Lag2, data=Weekly, family=binomial)
summary(lr.fit3)
```

```
##
## 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
```

(c)

```
# Logistic regression using all observations except the first.
lr.fit3 = glm(Direction ~ Lag1+Lag2, data=Weekly[-c(1),], family=binomial)

# Predicting the direction of the first observation.
lr.probs3 = predict(lr.fit3,Weekly[1,], type="response")
if(lr.probs3 > 0.5) "Up" else "Down"
```

```
## [1] "Up"
```

- Prediction is "Up" and this is incorrect.

(d)

```
lr.preds = rep(NA, length(Weekly$Direction))
lr.err = rep(NA, length(Weekly$Direction))
for (i in 1:length(Weekly$Direction)){
  lr.fit=glm(Direction ~ Lag1+Lag2, data=Weekly[-c(i),], family=binomial)
  lr.probs = predict(lr.fit,Weekly[i,], type="response")
  #Make predictions for row i (left out of training set) and compare to actual prediction,
  #then use if else statement to state 1 if incorrect and 0 if correct.

  if(lr.probs > 0.5) lr.preds[i]="Up" else lr.preds[i]="Down"
  if(lr.preds[i]==Weekly$Direction[i]) lr.err[i]=0 else lr.err[i]=1
  }
```

(e)

Count of 1's divided by n:

```
#LOOCV Error rate
cv.err = sum(lr.err==1)/length(Weekly$Direction)
print(cv.err)
```

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

```
table(Weekly$Direction)
```

```
##
## Down   Up
##  484  605
```

- LOOCV error rate is 44.9%. This shows that the model is correct in 55% of its predictions, which is slightly better than random guessing.

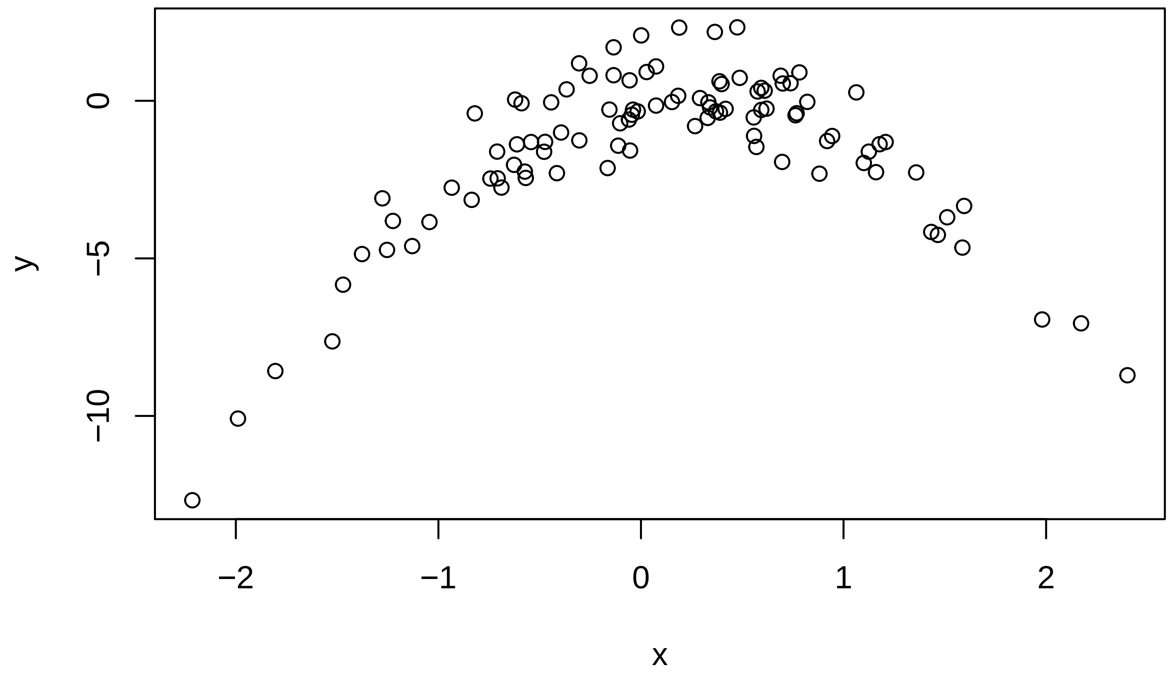
Applied

8. (a)

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

- $n = 100$, $p = 2$, $Y = X - 2X^2 + e$ **(b)**

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



- The dataset is non-linear and the relationship between X and Y is roughly quadratic.

(c)

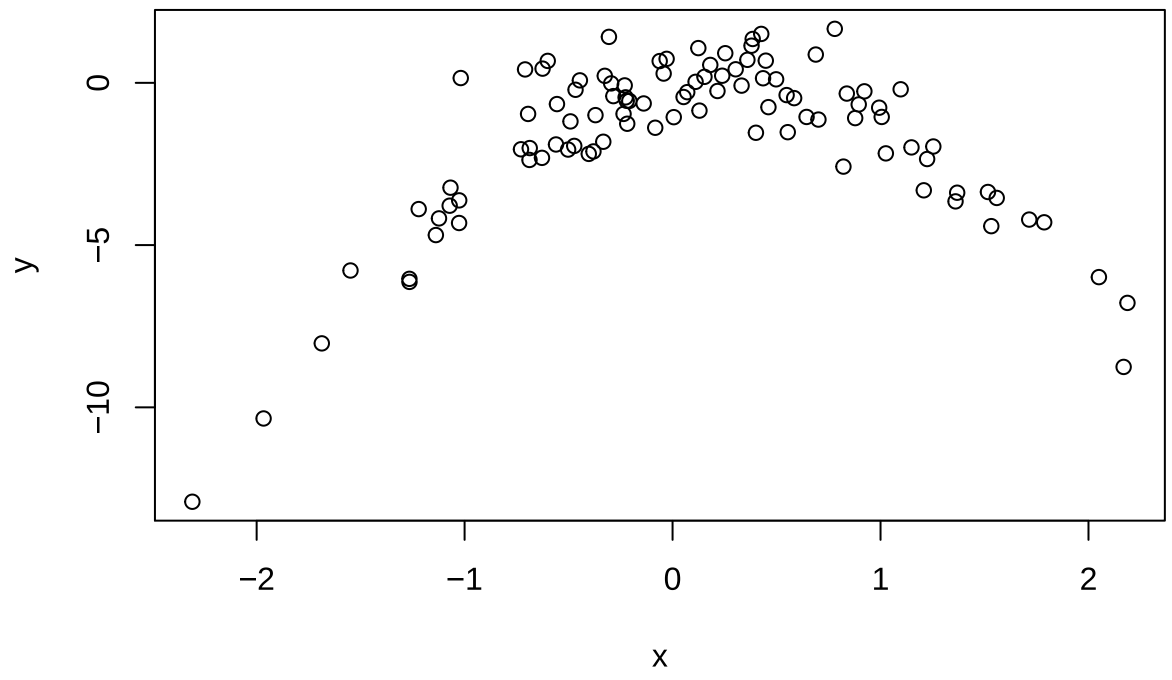
```
set.seed(11)
x = c(rnorm(100))
y = c(x-2*x^2+rnorm(100))
df = data.frame(x,y)
cv.err = rep(0,4)

for (i in 1:4){
  lr.fit = glm(y~poly(x,degree=i,raw=TRUE))
  cv.err[i] = cv.glm(df, lr.fit)$delta[1]
}
cv.err
```

```
## [1] 6.630651 1.032469 1.090550 1.071305
```

- As expected, there is a sharp drop in LOOCV estimate for MSE when a quadratic term is added. The error increases as higher polynomial terms are used, again confirming the quadratic nature of the dataset.

(d)

```
set.seed(123)
x = c(rnorm(100))
y = c(x-2*x^2+rnorm(100))
plot(x,y)
```



```
df = data.frame(x,y)
cv.err = rep(0,4)#
for (i in 1:4){
  lr.fit = glm(y~poly(x,degree=i,raw=TRUE))
  cv.err[i] = cv.glm(df, lr.fit)$delta[1]
}
cv.err
```

```
## [1] 6.9752118 0.9664678 1.0000174 0.9993215
```

- The results are similar to (c). As can be seen from the chart, the dataset is similar to before, and so the lowest error is given by the model with a quadratic term.

(e)

- The model with the quadratic term had the lowest LOOCV error. This is expected as the the dataset is roughly quadratic in shape - as show in (b).

(f)

```
summary(glm(y~poly(x,degree=1,raw=TRUE)))
```

```
##
## Call:
```

```
## glm(formula = y ~ poly(x, degree = 1, raw = TRUE))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -10.083   -1.048    0.712    1.740    3.288
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -1.7263     0.2561  -6.741 1.09e-09 ***
## poly(x, degree = 1, raw = TRUE)     0.4760     0.2806   1.696    0.093 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.494276)
##
##     Null deviance: 655.13  on 99  degrees of freedom
## Residual deviance: 636.44  on 98  degrees of freedom
## AIC: 474.86
##
## Number of Fisher Scoring iterations: 2
```

```
summary(glm(y~poly(x,degree=2,raw=TRUE)))
```

```
##
## Call:
## glm(formula = y ~ poly(x, degree = 2, raw = TRUE))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9368  -0.7291  -0.1191   0.6544   3.3320
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -0.03039    0.12033  -0.253    0.801
## poly(x, degree = 2, raw = TRUE)1  0.96856    0.10879   8.903 3.13e-14 ***
## poly(x, degree = 2, raw = TRUE)2 -2.08920    0.08684 -24.058  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9417574)
##
##     Null deviance: 655.13  on 99  degrees of freedom
## Residual deviance:  91.35  on 97  degrees of freedom
## AIC: 282.74
##
## Number of Fisher Scoring iterations: 2
```

```
summary(glm(y~poly(x,degree=4,raw=TRUE)))
```

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

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8630  -0.7018  -0.1688   0.6240   3.4230
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       0.05304    0.14328   0.370    0.712
## poly(x, degree = 4, raw = TRUE)1  0.93261    0.19220   4.852 4.78e-06 ***
## poly(x, degree = 4, raw = TRUE)2 -2.33497    0.24306  -9.606 1.15e-15 ***
## poly(x, degree = 4, raw = TRUE)3  0.01954    0.06701   0.292    0.771
## poly(x, degree = 4, raw = TRUE)4  0.05998    0.05587   1.073    0.286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9491413)
##
##     Null deviance: 655.130  on 99  degrees of freedom
## Residual deviance:  90.168  on 95  degrees of freedom
## AIC: 285.44
##
## Number of Fisher Scoring iterations: 2
```

- Model (ii): Degree=1 and 2 coefficients are statistically significant. This is expected and matches the CV results, where this model (with predictors X and X^2) provides the lowest test error.
- Model (iv): Degree=1 and 2 coefficients are statistically significant. This matches model (ii).

(9) (a) (b)

```r
# Mean of medv
mu = mean(Boston$medv)

# Standard deviation of mean
length = length(Boston$medv)
sd = sqrt(sum((Boston$medv - mu)^2)/(length-1))

# Standard error of mean
se = sd/sqrt(length)
se
```

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

(c)

```r
# Standard error using bootstrapping
boot.fn2 = function(data, index){
  X=data$medv[index]
  mu2 = mean(X)
  return(mu2)
}
```

```r
set.seed(1)
boot(Boston,boot.fn2,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston, statistic = boot.fn2, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.007650791   0.4106622
```

- The standard error using bootstrapping with R=1000 is 0.41, which is almost the same as in (c).

(d)

```
mu-2*se #lower bound
```

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

```
mu+2*se #higher bound
```

```
## [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
```

- The confidence intervals nearly the same when using the approximation formula or using one sample t-test.

(e) (f)

```
# Median value of medv
median.medv = median(Boston$medv)

# Bootstrap estimate of the standard error of the median value.
boot.fn3 = function(data, index){
  X=data$medv[index]
  Y = median(X)
  return(Y)
}

boot(Boston,boot.fn3,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston, statistic = boot.fn3, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0386   0.3770241
```

- SE(median) = 0.37, which is relatively small when compared to the median value of 21.2. So we can be reasonably confident of the estimate.

(g)

```r
tenth.percentile = quantile(Boston$medv, 0.1)

boot.fn4 = function(data, index){
  X=data$medv[index]
  Y = quantile(X, 0.1)
  return(Y)
}

boot(Boston,boot.fn4,1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston, statistic = boot.fn4, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75  0.0186   0.4925766
```

- The standard error 0.499, which is relatively low when compared to the 10th percentile value of 12.75.