Chapter 5 Exercises

Cross-validation and Bootstrap Lab

Conceptual

1)

$Var(\alpha X + (1-\alpha)Y) = \alpha^2 Var(X) + (1-\alpha)^2 Var(Y) + 2\alpha(1-\alpha)Cov(XY)$

$= \alpha^2\sigma_X^2 + (1-\alpha)^2\sigma_Y^2 + (2\alpha - 2\alpha^2)\sigma_{XY}$

$\frac{df}{d\alpha} = 2\alpha\sigma_X^2 + (2\alpha-2)\sigma_Y^2 + (2-4\alpha)\sigma_{XY}$

Setting the derivative to zero and solving for alpha gives equation 5.6

Second derivative:

$\frac{d^2f}{d^2\alpha} = 2\sigma_X^2 + 2\sigma_Y^2 - 4\sigma_{XY} = 2Var(X-Y) \geq 0$

Hence, the extreme value achieved with alpha set to equation 5.6 is indeed a minimum.

2)

a) As the jth observation is one observation in n, the chance of the bootstrap picking it first is $\frac{1}{n}$ meaning the chance of not picking it is $\frac{n-1}{n}$

b) As the bootstrap select samples *with replacement*, the samples selected prior to the k^{th} selection have no effect on th k^{th} selection. Hence, the probability is still $\frac{n-1}{n}$

c) Because the probability of selecting a specific observation is not affected by the prior selections. The probability that j is not selected at all is simply the probability of not selecting j to the power of the total number of observations (assuming that is the bootstrap is selecting n observations in total)

For answers d to f the answers will be $1 - (\frac{n-1}{n})^n$ for n = 5,100 and 10000

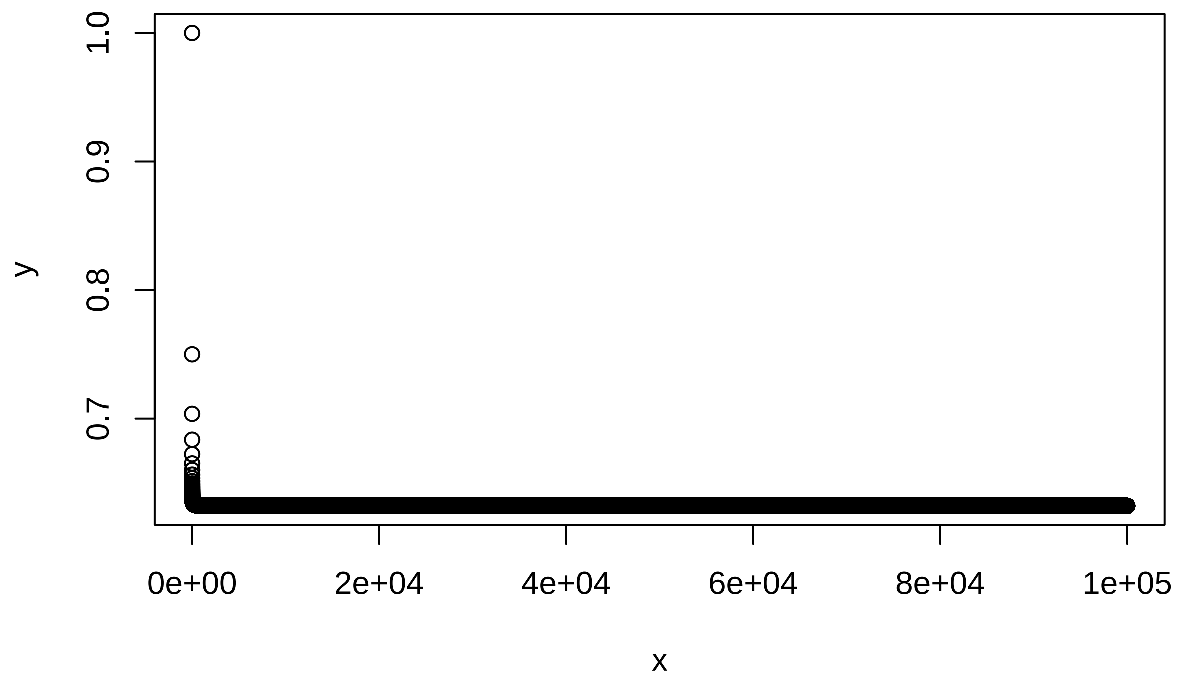
g)

Generating the plot we see below shows that the probability of j being in the bootstrap plateaus near 0.6 as the total number of samples increases.

```r
x <- c(5,100,10000)
y <- 1 - ((x-1)/(x))^x
data.frame(Question = c("d","e","f"),
N =c(5,100,10000), Prob = y)
```

```
##   Question     N      Prob
## 1        d     5 0.6723200
## 2        e   100 0.6339677
## 3        f 10000 0.6321390
```

```r
x <- 1:100000
y <- 1 - ((x-1)/(x))^x
plot(x,y)
```



h)

Much like the plot above, a numerical investigation reaches a similar conclusion. As the number of observations increases, the bootstrap method's probability of including a sample in the bootstrap plateaus near 60%.

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

```
## [1] 0.6362
```

3)

a) k-fold cross-validation is implemented by splitting the data intro k equal-sized groups. Then,each group will be used as a test data set with the remaining groups used to train the model. This will lead to k models being built allowing for accurate estimations of the parameters of the model. In particular, it enables a more accurate estimation of the model's test error.

b) - K-fold enables us to generate a more accurate estimate of the test error as we can generate multiple models and use different samples in the training and validation set (reduces the variability in the estimates). This helps give us a better idea of the model's accuracy. On the other hand, by dividing the data into smaller groups, we inevitably end up with a smaller validation set to use against the data.
   - LOOCV leads to much more models being built (n models vs k). The key advantage k-folds has here is that it is less computationally intensive as it only builds k models vs n (where n > k). As a result it returns estimates much faster. LOOCV will have less bias than k-folds as it includes almost every sample (n-1) but this means the variance is higher as the training sets for each model are very similar.

4)

We can use the bootstrap method to make B number of estimates of the prediction. We can then use equation 5.8 to determine the SD of the prediction.

Applied

5)

```
# A
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

```
attach(Default)
log.fit <- glm(default ~ income + balance, data = Default, family = binomial)
# B
set.seed(1)
train <- sample(c(TRUE,FALSE),nrow(Default),replace = TRUE)
log.fit <- glm(default ~ income + balance,
                data = Default, family = binomial, subset = train)
log.prob <- predict(log.fit, Default[!train,], type = "response")
log.pred <- rep("No",length(log.prob))
log.pred[log.prob > 0.5] <- "Yes"
mean(log.pred != Default[!train,"default"])
```

```
## [1] 0.02651515
```

```
#We get a ~2.5% error rate

# C
for (i in 1:3){
  train <- sample(c(TRUE,FALSE),nrow(Default),replace = TRUE)
  log.fit <- glm(default ~ income + balance,
                  data = Default, family = binomial, subset = train)
  log.prob <- predict(log.fit, Default[!train,], type = "response")
  log.pred <- rep("No",length(log.prob))
  log.pred[log.prob > 0.5] <- "Yes"
  print(mean(log.pred != Default[!train,"default"]))
}
```

```
## [1] 0.02684426
## [1] 0.02692998
## [1] 0.02528317
```

```
# We see the variability in the error based on the exact samples used in the
#training model

#D
for (i in 1:3){
  train <- sample(c(TRUE,FALSE),nrow(Default),replace = TRUE)
  log.fit <- glm(default ~ income + balance + student,
                 data = Default, family = binomial, subset = train)
  log.prob <- predict(log.fit, Default[!train,], type = "response")
  log.pred <- rep("No",length(log.prob))
  log.pred[log.prob > 0.5] <- "Yes"
  print(mean(log.pred != Default[!train,"default"]))
}
```

```
## [1] 0.02854861
## [1] 0.02725826
## [1] 0.02940024
```

```
#Seem to get slightly lower test error rates when student
#is included although with enough repeats the averages are likely to be
#identical
```

6)

```
#A

set.seed(2022)
log.fit <- glm(default ~ income + balance, data = Default,family = binomial)
summary(log.fit)$coeff[,2]
```

```
##  (Intercept)       income      balance
## 4.347564e-01 4.985167e-06 2.273731e-04
```

```
#B
boot.fn <- function(data, index){
  fit <- glm(default ~ income + balance, data = Default,
             family = binomial,subset = index)
  fit$coef
}

#C
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 -3.549550e-02 4.213490e-01
## t2*  2.080898e-05  4.743080e-08 4.888646e-06
## t3*  5.647103e-03  1.542289e-05 2.249716e-04
```

```
#D
#The estimates for SE seem to be very similar between bootstrap and summary
```

7)

```
library(ISLR2)
#Using Weekly data set

# A
log.fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
# B
log.fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-1,], family = binomial)
# C
pred <- predict(log.fit,Weekly[1,],type = "response")
ifelse(pred > 0.5,"Up","Down")
```

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

```
# D
ans_d <- sapply(1:nrow(Weekly), function(i) {
  log.fit <- glm(Direction ~ Lag1 + Lag2, data = Weekly[-i,], family = binomial)
  prob <- predict(log.fit,Weekly[i,],type = "response")
  pred <-ifelse(prob > 0.5,"Up","Down")
  ifelse(pred == Weekly[i,"Direction"],0,1)
})

#E
mean(ans_d)
```

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

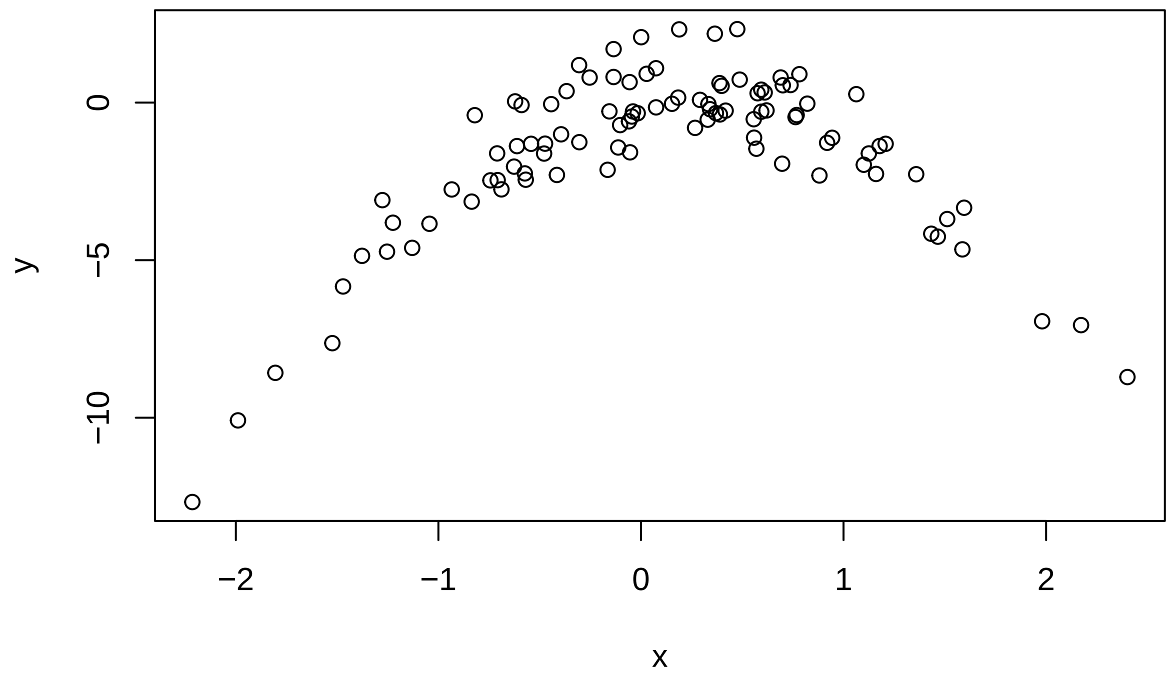
```
#We get an error rate of ~45%
```

8) A)

N is 100 (100 x/y observations), p is 1 (we have a polynomial linear regression)

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

# B
plot(x,y)
```



```r
# See an inverted parabola that  has a global maximum at the origin

# C
library(boot)
set.seed(1)
q_8 <- data.frame(x = x,y = y)
fit <- glm(y ~ x)
lin <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,2))
sqrd <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,3))
cube <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,4))
quad <- cv.glm(q_8,fit)$delta[1]
rbind(c("Poly(1)","Poly(2)","Poly(3)","Poly(4)"),c(lin,sqrd,cube,quad))
```

```
##      [,1]                 [,2]                 [,3]
## [1,] "Poly(1)"          "Poly(2)"           "Poly(3)"
## [2,] "7.28816160667281" "0.937423637615552" "0.95662183010894"
##      [,4]
## [1,] "Poly(4)"
## [2,] "0.953904892744804"
```

```
# D
set.seed(2022)
fit <- glm(y ~ x)
lin <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,2))
sqrd <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,3))
cube <- cv.glm(q_8,fit)$delta[1]
fit <- glm(y ~ poly(x,4))
quad <- cv.glm(q_8,fit)$delta[1]
rbind(c("Poly(1)","Poly(2)","Poly(3)","Poly(4)"),c(lin,sqrd,cube,quad))
```

```
##      [,1]               [,2]                [,3]
## [1,] "Poly(1)"          "Poly(2)"           "Poly(3)"
## [2,] "7.28816160667281" "0.937423637615552" "0.956621830108939"
##      [,4]
## [1,] "Poly(4)"
## [2,] "0.953904892744804"
```

```
#The errors seem to be fairlt similar across seeds

#E

#The second degree polynomial model has the lowest error which
#makes sense as it is closest to the true relationship between
#x and y

# F
#Observing the summary stats for the models in c shows that
#the only significant coefficients are those of x and x^2,
#indicating that the least squares results agree with LOOCV
#that the best model only uses the second degree polynomial.
```

9)

```
# A
attach(Boston)
mean(medv)
```

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

```
# B
sd(medv)/sqrt(length(medv))
```

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

```r
# C
boot.fn <- function(data,index){
  mean(data[index])
}
boot(medv,boot.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 22.53281 0.01130771   0.4145107
```

```r
#The answer is fairly close to B

# D
conf <- c(mean(medv) - 2*0.4119,mean(medv) + 2*0.4119)
t <- t.test(medv)
rbind(conf,t$conf.int[1:2])
```

```
##          [,1]     [,2]
## conf 21.70901 23.35661
##      21.72953 23.33608
```

```r
## The confidence intervals are approximately the same

# E
median(medv)
```

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

```r
# F
med.fn <- function(data,index){
  median(data[index])
}
boot(medv,med.fn,R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = med.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*     21.2 -0.01895   0.3682241
```

```r
#The bootstrap returns a median that matches our estimate and
#a relatively small standard error

# G
quantile(medv,0.1)
```

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

```r
# H
quant.fn <- function(data, index){
  quantile(data[index],0.1)
}
boot(medv,quant.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = medv, statistic = quant.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*    12.75 -0.00095   0.5265633
```

```r
# The bootstrap returns a 10th percentile estimate that matches
# our own and also has a small standard error.
```