Chapter 5 Exercises

Name: Jiangxue Han

Computing ID: jh6rg

2.

(a) 1-(1/n)

Since there are n observations in total, the probability of not obtaining one observation is 1-(1/n).

(b) 1-(1/n)

Because the bootstrap sampling is performed with replacement, therefore the probability is the same with that of the first observation.

(c)

There are n observations in total and the probability that the chance of jth observation not being in the bootstrap sample is always (1-(1/n)) for each drawing. Thus, the probability is (1-(1/n))^n.

(d) 67.2%

1 - (1-1/5)^5 = 67.2%

(e) 63.4%

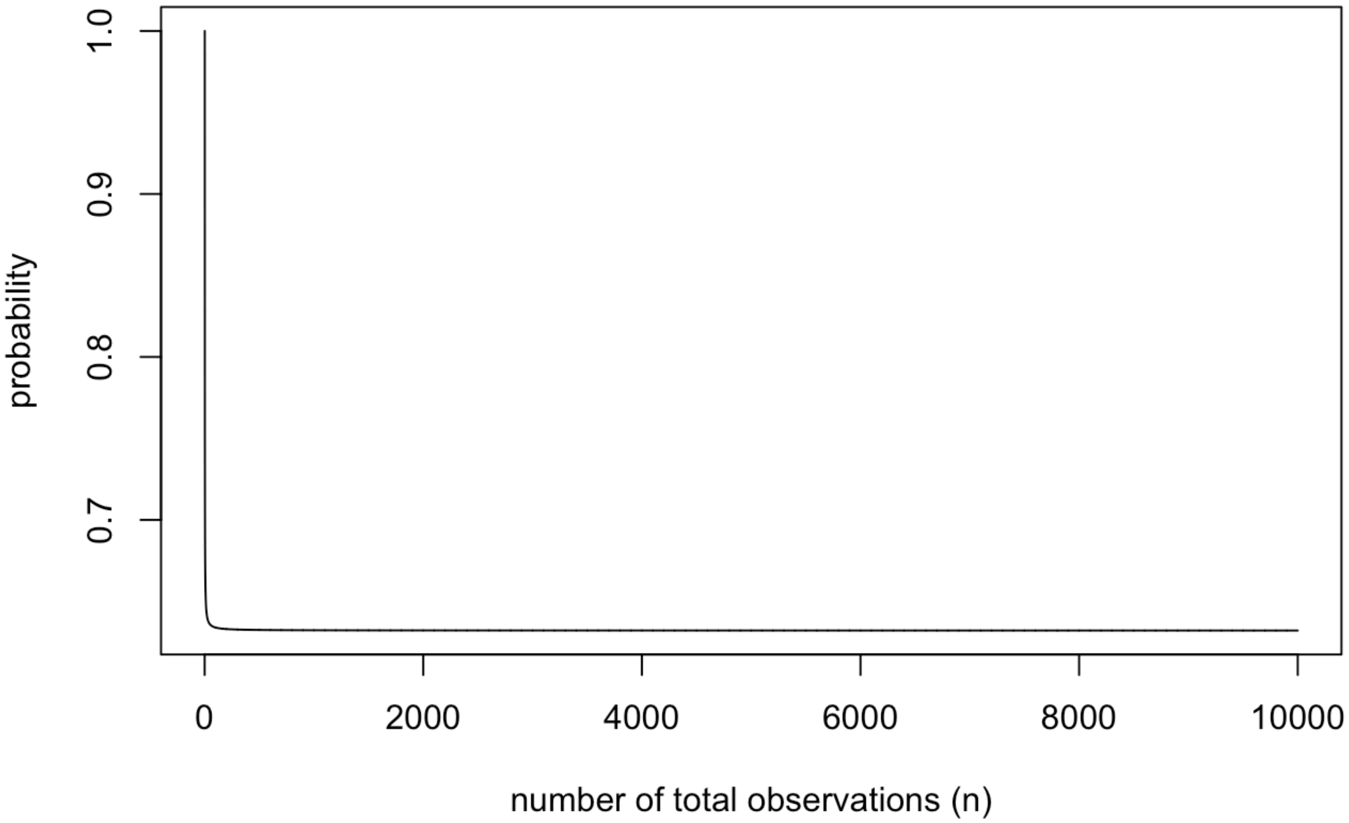
1-(1-1/5)^100 = 63.4%

(f) 63.2%

1-(1-1/10000)^10000 = 63.2%

(g)

```
x = seq(1,10000)
y = 1-(1-1/x)^x
plot(x, y, type="l",xlab="number of total observations (n)",ylab = "probability")
```



The probability decreases to 63.2% quickly and stays the same even if the number of total observations is increasing.

(h)

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

```
## [1] 0.6373
```

After repeating creating bootstrap samples 10,000 times, the mean of probability that a bootstrap sample of size n = 100 contains the jth observation is 63.4%. This is consistent with the result in (g).

3.

a. k-fold cross-validation first divides the entire training set into k subsets. Then we choose one subset as the validation set and train on the rest k-1 sets. Next we calculate the MSE for the held-out fold. Each time, a different set is treated as the validation set. We repeat this procedure for k times and finally compute the average of all MSEs. The aim for k-fold CV is to find the minimum point in the estimated test MSE curve and identify the correspending level of flexibility.

b.

i. The validation set approach. Advantages: The test error rate is less variable which results in smaller variance. k-fold also has a smaller bias, because it each training set contains (k-1)n/k observations. Disadvantages: k-fold is more expensive to implement.

ii. LOOCV. Advantages: 1. Less Variance, because the outputs are less correlated with each other. 2. k-fold can be applied to almost any statistical learning method while LOOCV has the potential to be computationally expensive. Disadvantages: k-fold tends to have a higher bias than LOOCV.

5.

(a)

```r
library(ISLR)
default.lg <- glm(default ~ income+balance, data=Default, family=binomial)
```

(b)

```r
set.seed(1)
n <- nrow(Default)
split <- sample(n,n/2)
train <- Default[split,]
test <- Default[-split,]
default.lg <- glm(default ~ income+balance, data=train, family=binomial)
probs <- predict(default.lg, test, type="response")
pred <- rep("No", n/2)
pred[probs > 0.5] <- "Yes"
mean(pred != test$default)
```

```
## [1] 0.0286
```

(c)

```
default.glm <- function(percent) {
  set.seed(1)
  n <- nrow(Default)
  split <- sample(n,n*percent)
  train <- Default[split,]
  test <- Default[-split,]
  default.lg <- glm(default ~ income+balance, data=train, family=binomial)
  probs <- predict(default.lg, test, type="response")
  pred <- rep("No", nrow(test))
  pred[probs > 0.5] <- "Yes"
  return (mean(pred != test$default))
}
default.glm(0.6)
```

```
## [1] 0.02775
```

```
default.glm(0.7)
```

```
## [1] 0.028
```

```
default.glm(0.8)
```

```
## [1] 0.026
```

The validation set error decreases with the increasing proportion in training set.

(d)

```
default.glm2 <- function(percent) {
  set.seed(1)
  split <- sample(n,n*percent)
  train <- Default[split,]
  test <- Default[-split,]
  default.lg <- glm(default ~ income+balance+student, data=train, family=binomial)
  probs <- predict(default.lg, test, type="response")
  pred <- rep("No", nrow(test))
  pred[probs > 0.5] <- "Yes"
  mean(pred != test$default)
}
default.glm2(0.5)
```

```
## [1] 0.0288
```

```
default.glm2(0.6)
```

```
## [1] 0.02775
```

```
default.glm2(0.7)
```

```
## [1] 0.02766667
```

```
default.glm2(0.8)
```

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

It seems that adding the student into the model does not reduce the validation set error significantly.

8.

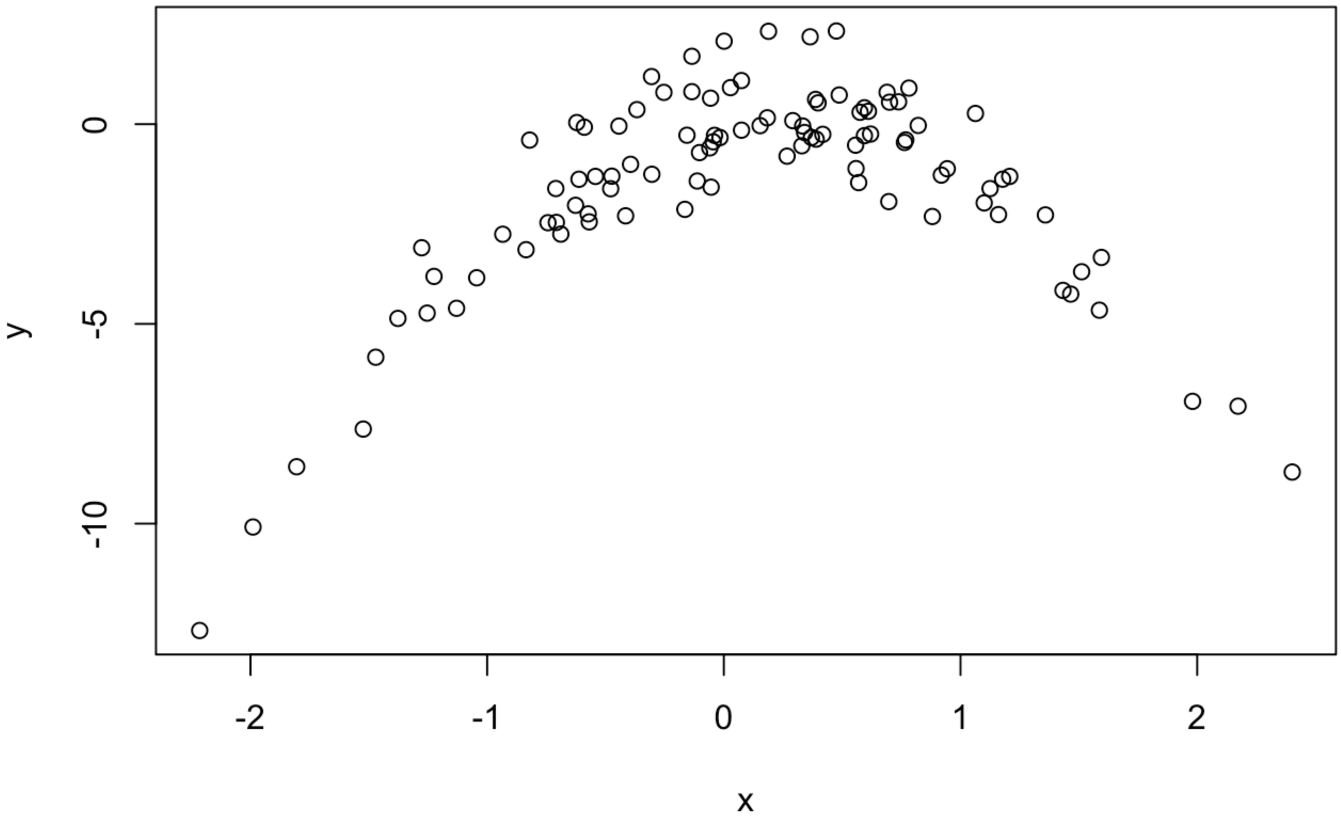
(a)

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

n = 100, p = 2. y = x - 2x^2 + є

(b)

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



x is from -2 to 2, and y is from -10 to 5.

(c)

```
library(boot)
set.seed(1)
data8 <- data.frame(x,y)
cv.error <- rep(0, 4)
for (i in 1:4) {
  glm.fit <- glm(y~poly(x, i), data=data8)
  cv.error[i] <- cv.glm(data8, glm.fit)$delta[1]
}
cv.error
```

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

(d)

```
set.seed(12)
data8 <- data.frame(x,y)
cv.error <- rep(0, 4)
for (i in 1:4) {
  glm.fit <- glm(y~poly(x, i), data=data8)
  cv.error[i] <- cv.glm(data8, glm.fit)$delta[1]
}
cv.error
```

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

The result are the same, because there is no randomness in the training/validation set splits.

(e) The quadratic model had the lowest LOOCV error. Because the linear model has a higher bias and the higher order polynomial is overfitting the training set.

(f)

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = y ~ poly(x, i), data = data8)
##
## 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, i)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, i)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, i)3   0.26411    0.95905   0.275    0.784
## poly(x, i)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
```

x and x^2 is stronger than the other variables. This is consistent with the cross validation result.