Resampling Methods

Hrishabh Khakurel

6/14/2020

```
library(ISLR)
library(boot)
library(ggplot2)
```

Contents

5.4.5

In this problem we will be using the *Default* dataset and estimate the test error of the logistic regression model using the **Validation Set Approach**.

Loading the dataset

```
default <- Default
# set seed for consistent result
set.seed(1)
```

(a)

We will create a logistic regression model in this section.

```
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)

In this section we will use the validation set approach.

i.

First, we will split the dataset into training set and validation set.

```
train <- sample(10000,5000)
```

ii.

We will use the training set to train the model.

```
glm.fit2 <- glm(default ~ income + balance, data = default, subset = train, 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
```

iii.

In this step, we will predict the outcome for validation set.

```
pred_prob <- predict(glm.fit2, newdata = default[-train,])
pred <- rep('No',5000)
pred[pred_prob > 0.5] = 'Yes'
```

iv.

In this set we will calculate the validation set error.

```
t <- as.matrix(table(pred, default[-train,]$default))
error <- round((1 - (t[1,1] + t[2,2])/5000)*100,2)
paste0('The validation set error is ',error, '%')
```

```
## [1] "The validation set error is 2.64%"
```

Thus, we obtain a validation set error of 2.64%.

(d)

We will now show the inconsistency of the validation set method by doing the process three times.

```
error_list <- vector()
for(i in 1:3){
  set.seed(i)
  train <- sample(10000,5000)
  glm.fit3 <- glm(default ~ income + balance, data = default, subset = train, family = binomial)
  pred_probs3 <- predict(glm.fit3, default[-train,])
  pred3 <- rep('No',5000)
  pred3[pred_probs3 > 0.5] <- 'Yes'
  mat <- table(pred3, default[-train,]$default)
  error[i] <- round((1 - (mat[1,1] + mat[2,2])/5000)*100,2)
}

kableExtra::kable(data.frame(Sampling = c(1,2,3), error))
```

Sampling	error
1	2.64
2	2.60
3	2.60

As we see, we get three different validation errors for three different samples taken.

(d)

In this section we will add a new independent variable student to our model.

```
set.seed(1)
train <- sample(10000,5000)
glm.fit4 <- glm(default ~ income + balance + student, data = default, subset = train, family = binomial
pred_probs4 <- predict(glm.fit4, default[-train,])
pred4 <- rep('No',5000)
pred4[pred_probs4 > 0.5] <- 'Yes'
mat <- table(pred4, default[-train,]$default)
error4 <- round((1 - (mat[1,1] + mat[2,2])/5000)*100,2)
paste0('The validation set error is ',error4, '%')
```

```
## [1] "The validation set error is 2.6%"
```

Yes the error goes down when we include the student variable.

5.4.6

We will be using the same default dataset in this exercise. We will be computing the estimates for the standard errors in the following two ways:
1. Using the bootstrap.
2. Using the standard formula.

(a)

```
set.seed(1)
summary(glm(default ~ income + balance, data = default, family = binomial))$coef
```

```
##                   Estimate   Std. Error    z value      Pr(>|z|)
## (Intercept) -1.154047e+01 4.347564e-01 -26.544680 2.958355e-155
## income       2.080898e-05 4.985167e-06   4.174178  2.990638e-05
## balance      5.647103e-03 2.273731e-04  24.836280 3.638120e-136
```

The above output is associated with the calculation of estimate and their standard errors using the standard formula.

(b)

In this section we will construct the function to calculate the statistics needed for bootstrap.

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

(c)

In this section we will perform the bootstrap using the function defined in the above section.

```
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 -3.945460e-02 4.344722e-01
## t2*  2.080898e-05  1.680317e-07 4.866284e-06
## t3*  5.647103e-03  1.855765e-05 2.298949e-04
```

(d)

As we can see, both the standard function and the bootstrap method gives the same coefficient estimates but differ in the std. error. As discussed in the book, the SE from the standard function is not accurate as the formula used to calculate the SE depends on underlying assumptions of linearity in the data. This may not be satisfied. Hence, the SE obtained from the bootstrap is a better estimate.

5.4.7

In this section, we will perform LOOCV without using the *cv.glm* function. This can be accomplished by using loops. We will work with the *Weekly* dataset.

Loading the Data

```
weekly <- Weekly
kableExtra::kable(head(weekly,10))
```

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
1990	0.816	1.572	-3.936	-0.229	-3.484	0.1549760	-0.270	Down
1990	-0.270	0.816	1.572	-3.936	-0.229	0.1485740	-2.576	Down
1990	-2.576	-0.270	0.816	1.572	-3.936	0.1598375	3.514	Up
1990	3.514	-2.576	-0.270	0.816	1.572	0.1616300	0.712	Up
1990	0.712	3.514	-2.576	-0.270	0.816	0.1537280	1.178	Up
1990	1.178	0.712	3.514	-2.576	-0.270	0.1544440	-1.372	Down
1990	-1.372	1.178	0.712	3.514	-2.576	0.1517220	0.807	Up
1990	0.807	-1.372	1.178	0.712	3.514	0.1323100	0.041	Up
1990	0.041	0.807	-1.372	1.178	0.712	0.1439720	1.253	Up
1990	1.253	0.041	0.807	-1.372	1.178	0.1336350	-2.678	Down

(a)

We will fit a logistic model using Lag1 and Lag2 to predict the direction.

```
glm.fit5 <- glm(Direction ~ Lag1 + Lag2, data = weekly, family = binomial)
summary(glm.fit5)
```

```
##
## 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
```

We can see that the dependent variable are not very statistically significant.

(b)

In this section we will fit the logistic regression model using all but the first observation.

```
temp_data <- weekly[-1,]
glm.fit4 <- glm(Direction ~ Lag1 + Lag2, data = temp_data, family = binomial)
summary(glm.fit4)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = temp_data)
##
## 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
```

(c)

In this section we will predict the result for the first observation.

```
if (predict(glm.fit4,weekly[1,]) > 0.5){
  print('The Direction is predicted to be Up.')
} else {
  print('The Direction is predicted to be Down.')
}
```

```
## [1] "The Direction is predicted to be Down."
```

```
paste0('The true outcome is ', weekly[1,]$Direction, '.' )
```

```
## [1] "The true outcome is Down."
```

Thus, the prediction was correct.

(d)

In this section we will create a loop to perform LOOCV.

```
count <- rep(0,1089)
for (i in 1:1089){
  temp_data2 <- weekly[-i,]
  temp.fit <- glm(Direction ~ Lag1 + Lag2, data = temp_data2, family = binomial)
  if(predict(temp.fit, weekly[i,]) > 0.5){
    temp.pred <- 'Up'
  } else {
    temp.pred <- 'Down'
  }
  if (temp.pred != weekly[i,]$Direction){
    count[i] <- 1
  }
}
```

(e)

In this section, we will find the error rate.

```
paste0('The LOOCV error is ',round((sum(count)/nrow(weekly))*100,2), '%')
```

```
## [1] "The LOOCV error is 54.55%"
```

The error is very high. So the model is not accurate.

5.4.8

In this section we will perform cross-validation on a simulated dataset.

(a)

We will first simulate some data.

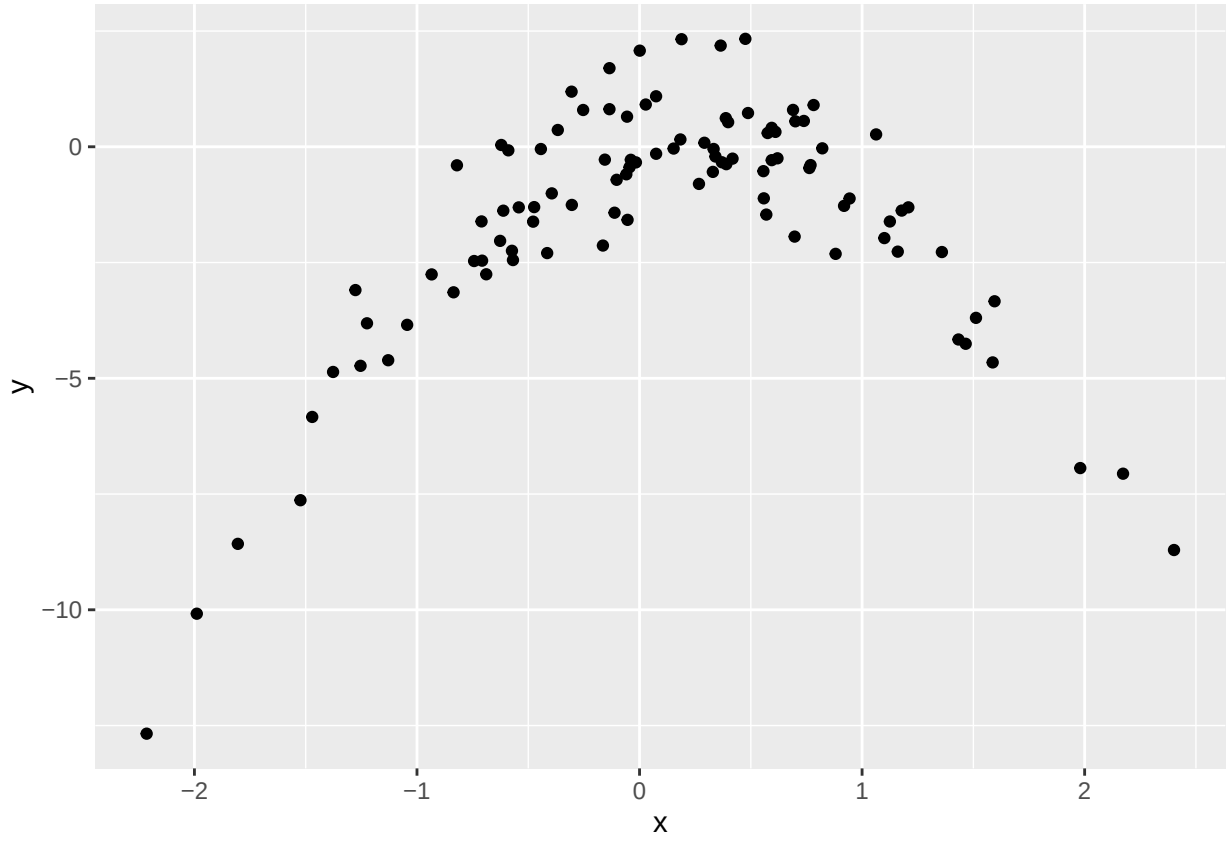
```
set.seed(1)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

Here, n is 100 and p is 2.

(b)

Visual representation of the data

```r
ggplot(data.frame(x,y), aes(x,y)) +
  geom_point()
```

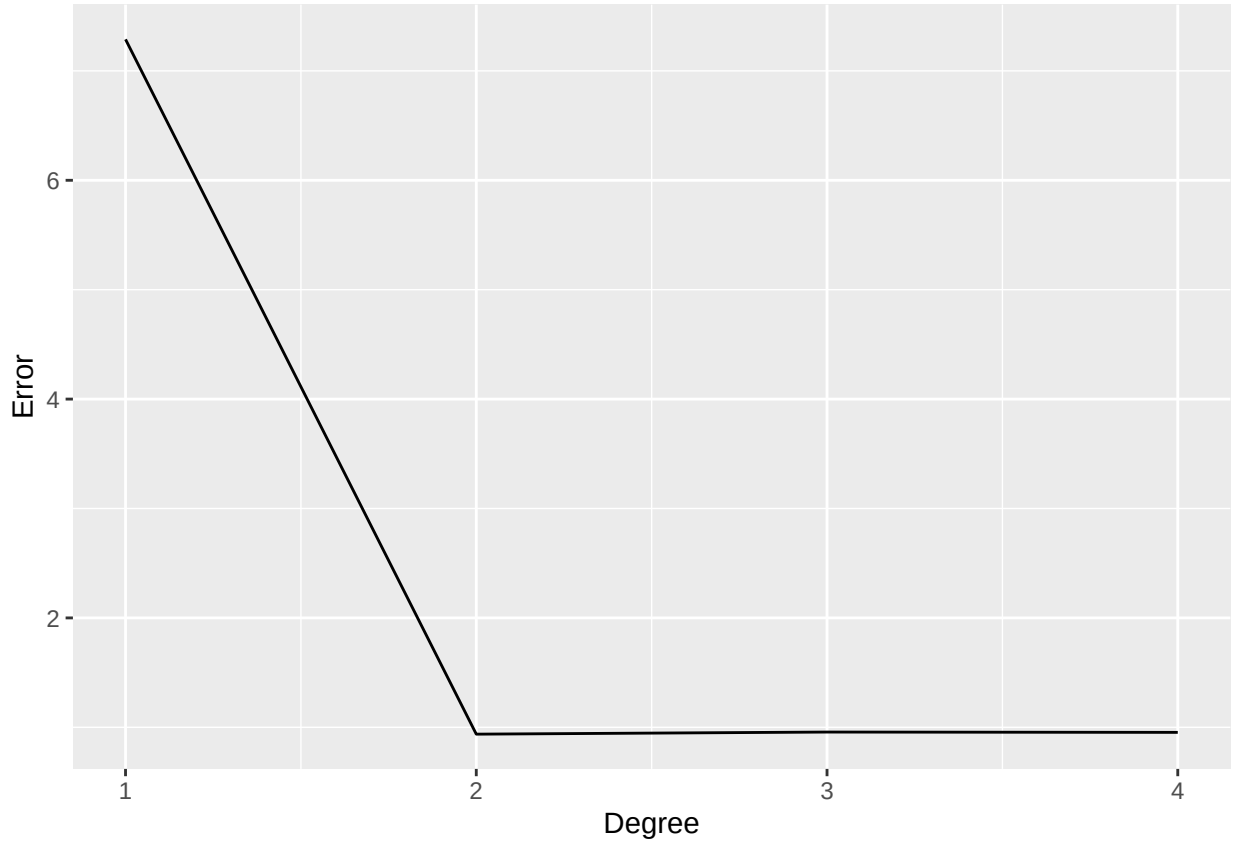


This model is quadratic.

(c)

In this section, we will compute the four given models nad their LOOCV statistics. We will automate this process using loops.

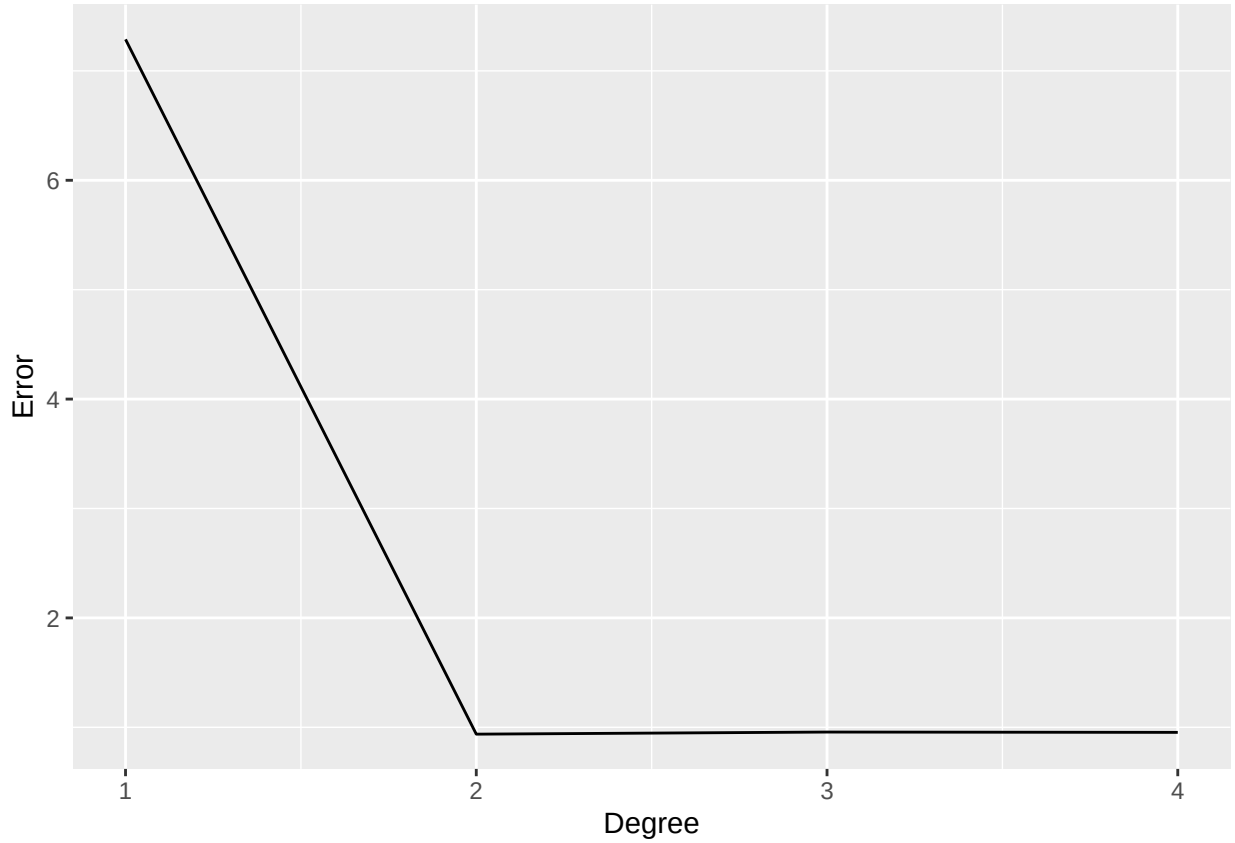
```r
data <- data.frame(x,y)
set.seed(5)
loocv_error <- vector()
for(i in 1:4){
  glm.fit <- glm(y ~ poly(x,i), data = data)
  loocv_error[i] <- cv.glm(data, glm.fit)$delta[1]
}
ggplot(data.frame(Degree = c(1,2,3,4), Error = loocv_error), aes(x = Degree, y = Error))+
  geom_line()
```



As we see, the loocv error drops sharply when the degree increases from 1 to 2. After that the error remains constant. This is because, the data had a quadratic relationship. Once that is attained, their cannot be further improvement in the error.

(d)

```
set.seed(100)
loocv_error2 <- vector()
for(i in 1:4){
  glm.fit <- glm(y ~ poly(x,i), data = data)
  loocv_error2[i] <- cv.glm(data, glm.fit)$delta[1]
}
ggplot(data.frame(Degree = c(1,2,3,4), Error = loocv_error2), aes(x = Degree, y = Error))+
  geom_line()
```



We obtain the same results as before. This is because there is no randomness in the training/validation set split in the LOOCV method.

(e)

As we see, the loocv error drops sharply when the degree increases from 1 to 2. After that the error remains constant. This is because, the data had a quadratic relationship. Once that is attained, their cannot be further improvement in the error. Thus, the best model is the quadratic model.

(f)

In this section we will look at the statistical significance of the coefficient described in each model.

```
for (i in 1:4){
  glm.fit <- glm(y ~ poly(x,i), data = data)
  print(summary(glm.fit))
}
```

```
##
## Call:
## glm(formula = y ~ poly(x, i), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.550      0.260  -5.961 3.95e-08 ***
## poly(x, i)     6.189      2.600   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, i)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, i)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, i)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, i)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, i)3   0.26411    0.96263   0.274    0.784
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = data)
##
## 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
```

As we can see, only the coefficients of intercept, x and x^2 is statistically significant. This result is the same as the result we obtained from crossvalidation.

5.4.9

In this section we will be using the *Boston* dataset from the *MASS* package.

Loading the Dataset

```
boston <- MASS::Boston
kableExtra::kable(head(boston, 10))
```

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

(a)

```
paste0('The estimate for the population mean is ', round(mean(boston$medv),2))
```

```
## [1] "The estimate for the population mean is 22.53"
```

(b)

```
paste0('The estimate for the standard error is ', round(sqrt(var(boston$medv)/nrow(boston)),3))
```

```
## [1] "The estimate for the standard error is 0.409"
```

(c)

We will use the bootstrap function to calculate the standard error.

Step 1: Creating the function to compute the statistics

```
boot.fn2 <- function(data,index)
  return(sd(data[index])/sqrt(length(data[index])))
```

Step 2: Perform the Bootstrap

```
t <- boot(boston$medv, boot.fn2, R = 1000)
t
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boston$medv, statistic = boot.fn2, R = 1000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.4088611 -0.0001041296  0.01712303
```

The estimated std. error is same as the previously calculated std. error.

(d)

```
CI <- c(mean(boston$medv) - 2 * t$t0,mean(boston$medv) + 2 * t$t0)
paste0('The 95% Confidence Interval obtained from bootstrap is: ')
```

```
## [1] "The 95% Confidence Interval obtained from bootstrap is: "
```

```
CI
```

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

As we can see, our results are similar to the ones obtained using t-test.

(e)

```r
paste0('The estimate for median of medv is ',median(boston$medv))
```

```
## [1] "The estimate for median of medv is 21.2"
```

(f)

We are using bootstrap to calculate the the standard error of the median estimator.

```r
boot.fn3 <- function(data,index){
  return(median(data[index]))
}
boot(boston$medv,boot.fn3, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boston$medv, statistic = boot.fn3, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     21.2 -0.0218   0.3863535
```

The standard error for the median is 0.3839.

(g)

```r
paste0('The tenth percentile of medv is ', quantile(boston$medv,0.1))
```

```
## [1] "The tenth percentile of medv is 12.75"
```

(h)

```r
boot.fn4 <- function(data,index){
  return(quantile(data[index],0.1))
}
boot(boston$medv,boot.fn4, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boston$medv, statistic = boot.fn4, R = 1000)
```

```
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75  0.0181   0.5096343
```

The standard error for the 10 percentile is 0.4869.