101c-hw4

Lucy Shao,id 304575686,lec 2

May 16, 2017

Q1) Use the appropriate transformation to the y variable first, then use Backwards Stepwise regression to determine a Least Squares model that predicts the price of a house. (houses.csv). Do this using set.seed(2628) (a) Validation Set cross-validation (b) Leave One Out cross-validation (c) 10-fold cross-validation (d) Mallows-Cp Write up a paragraph explaining the relative advantages and disadvantages of these methods.

For cross-validation, by spliting data into 70% for train data and 30% for test data, use test data to test the model resulting from train data which is simpler compared to other ways. However, we could not repeat it and it may be less accurate.LOOCV has better accuracy than cross-validation, since it uses multiple rounds of cross-validation to reduce the potential error. It helps to reduce the variability to find the simplest model. It also prevent overfitted condition. However, LOOCV has to repeat for several times which cost more time. 10-fold cross-validation use nearest 10 points to predict. It is for each case so that it could minimize the MSE. However, k is an unfixed number which is not certained. We need trial and error to reproduce the best k. Mallow-cp is the best way overall to predict with this given data, however it does not considered as prediction power.

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.2
```

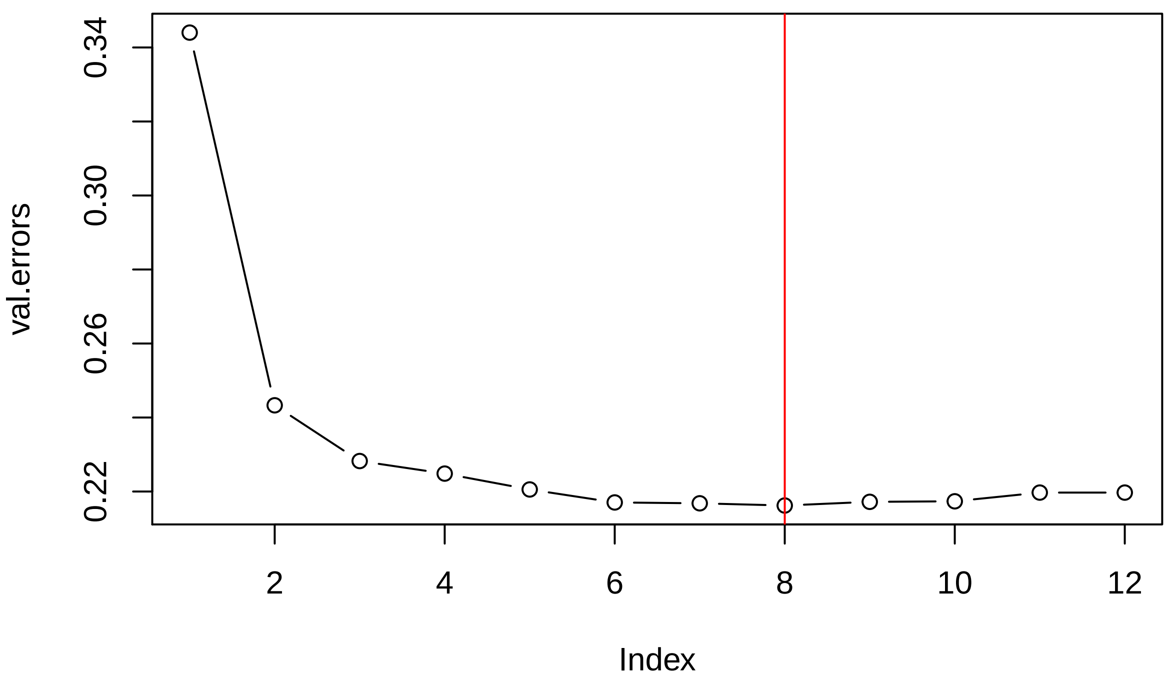
```
dat<-read.csv("/Users/lucy/Downloads/houses.csv",header=TRUE)
dat=dat[complete.cases(dat),]
dat$price=log(dat$price)#since skewed
set.seed(2628)

#training and testing
train=sample(5981,4186)
test=-train
regfit.best=regsubsets(price~.,data=dat[train,],nvmax=12,method="backward")
test.mat=model.matrix(price~.,data=dat[test,])
#self-defined function to get predicted value
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat<-model.matrix(form,newdata)
  coefi<-coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
val.errors=vector()
for(i in 1:12){
  pred=predict(regfit.best,dat[test,],id=i)
  val.errors[i]=mean((dat$price[test]-pred)^2)
}
val.errors
```

```
##  [1] 0.3440151 0.2433001 0.2282386 0.2248428 0.2205533 0.2170459 0.2168152
##  [8] 0.2162193 0.2172086 0.2173584 0.2197100 0.2197096
```

```r
best=which.min(val.errors)
best
```

```
## [1] 8
```

```r
plot(val.errors,type='b')
abline(v=best,col='red')
```




```r
regfit.best=regsubsets(price~.,data=dat,nvmax=12,method="backward")
coef(regfit.best,best)
```

```
##       (Intercept)        bldg.full        total.full     percent.brick
##     -8.679991e+00    -6.243556e-06      6.155526e-06      2.354677e-03
## total.living.area    basement.area         bathrooms          bedrooms
##      3.367805e-04     2.250918e-04      4.084940e-02      5.012083e-02
##        year.built
##      9.804289e-03
```

```r
#k fold
nrow(dat)
```

```
## [1] 5981
```

```r
k=10
folds=sample(1:k,nrow(dat),replace = TRUE)
table(folds)
```

```
## folds
##   1   2   3   4   5   6   7   8   9  10
## 631 608 651 578 605 546 617 579 583 583
```

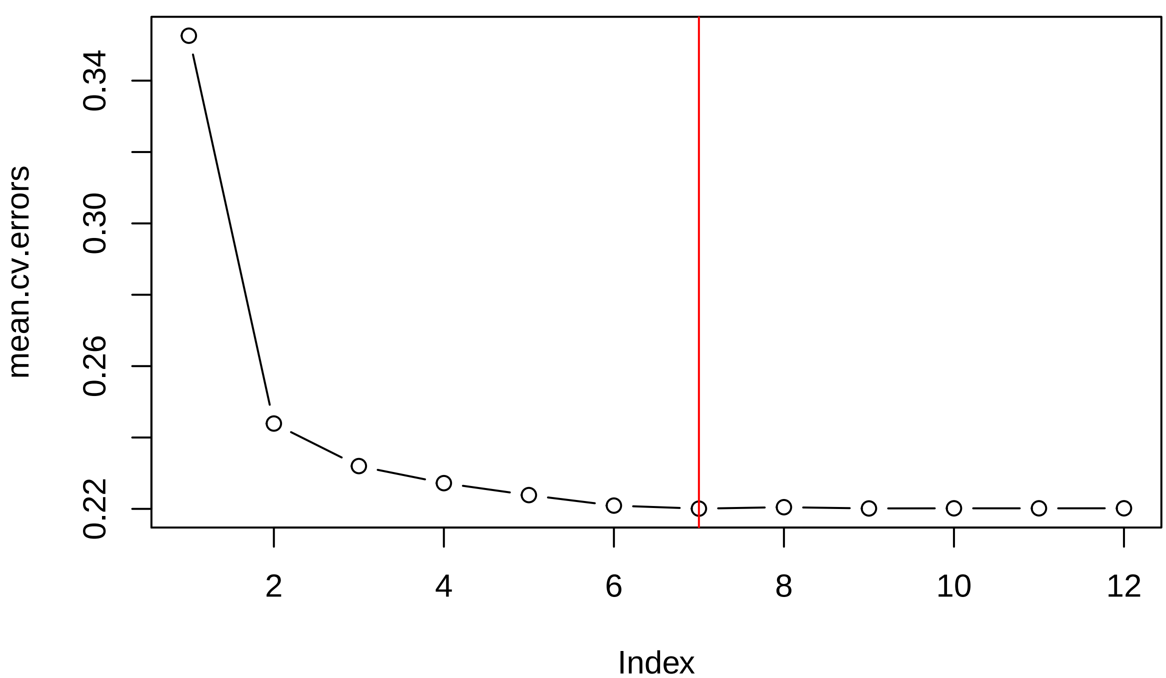
```
cv.errors=matrix(NA,k,12,dimnames=list(NULL,paste(1:12)))
for(j in 1:k){
  best.fit=regsubsets(price~.,data=dat[folds!=j,],nvmax=12,method="backward")
  for(p in 1:12){
    pred=predict(best.fit,dat[folds==j,],id=p)
    cv.errors[j,p]=mean((dat$price[folds==j]-pred)^2)
  }
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
##         1         2         3         4         5         6         7
## 0.3525920 0.2439127 0.2319934 0.2272130 0.2238597 0.2209176 0.2200680
##         8         9        10        11        12
## 0.2204665 0.2201214 0.2201582 0.2201522 0.2201646
```

```
best=which.min(mean.cv.errors)
best
```

```
## 7
## 7
```

```
plot(mean.cv.errors,type='b')
abline(v=best,col="red")
```



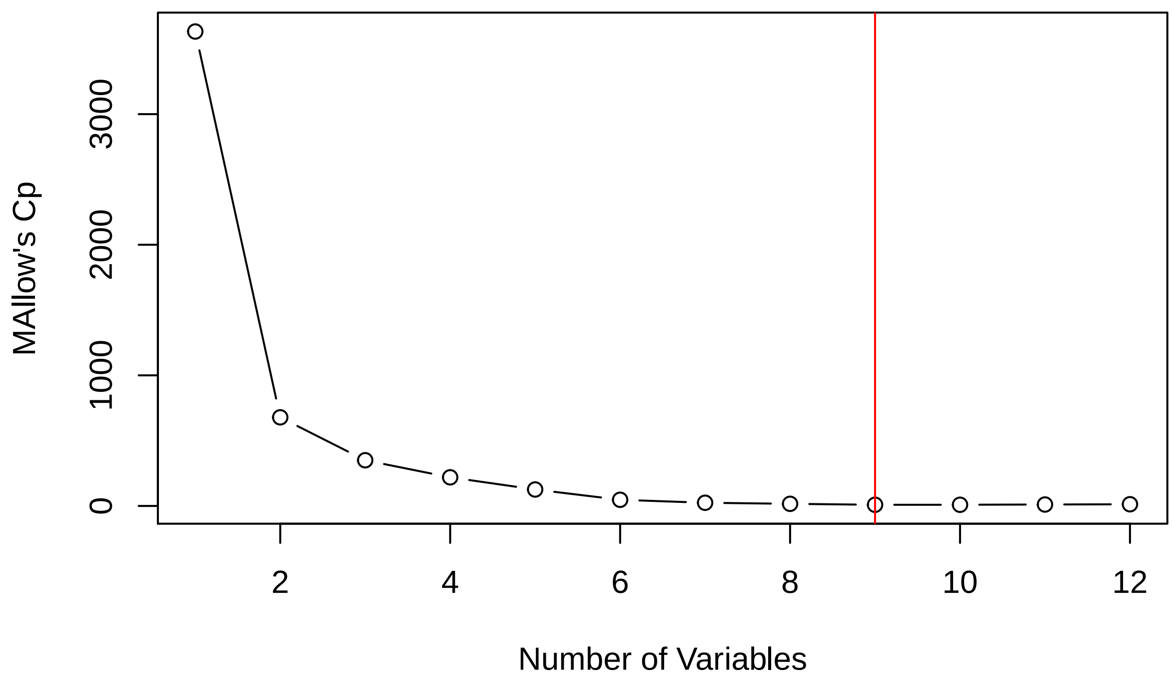
```r
reg.best=regsubsets(price~.,data=dat,nvmax=12,method="backward")
coef(reg.best,best)
```

```
##      (Intercept)        bldg.full        total.full    percent.brick
##    -9.277406e+00    -6.179780e-06      6.124659e-06     2.297808e-03
## total.living.area   basement.area         bedrooms       year.built
##     3.572355e-04     2.365283e-04      4.972832e-02     1.012144e-02
```

```r
#Mallow's Cp
best.fit=regsubsets(price~.,data=dat,nvmax=12,method="backward")
reg.summary=summary(best.fit)
best=which.min(reg.summary$cp)
best
```

```
## [1] 9
```

```r
plot(reg.summary$cp,xlab="Number of Variables",
     ylab = "MAllow's Cp", type="b")
abline(v=best,col="red")
```



###Q2) Using set.seed(2628) (a) Use Ridge Regression to predict the price of a house. Interpret the resulting model. (b) Make a plot that shows how the ratio of the size of the coefficients for Ridge Regression to the size of the coefficients for LS Change as lambda gets bigger. Your x-axis should have the values of lambda (from 10(-2) to 10(10) . The y-axis should have the ratio of the L2 norm of the Ridge Regression coefficients divided by the L2 Norm of the Least Squares coefficients. (Hint: you'll need to do a LS Regression with the variables standardized. Do not drop any terms from the model.) (c) Use Lasso to predict the price of a house and interpret. (d) Repeat (b) Using Lasso

Answer:Comparing to least square line, Lasso and Ridge Regression have less flexibility and greater biasness since the beta is limited. the Ridge regression has a penalty term on sum of

square values of beta(s). In the ridge regression the highly-variable parameters are shrunk, and it has less variability. Improve variability makes the bias gets worse. A lower MSE will help if bias gets worse more slowly than variability gets better. The LASSO model has a penalty term on sum of absolute values of beta(s). So it penalizes beta with large absolute value. It sacrifices unbiaseness for lower varaince. It will also increase prediction accuracy when the speed of increasing bias is slower than the speed of decreasing variance.

```
set.seed(2628)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.2

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-10
```

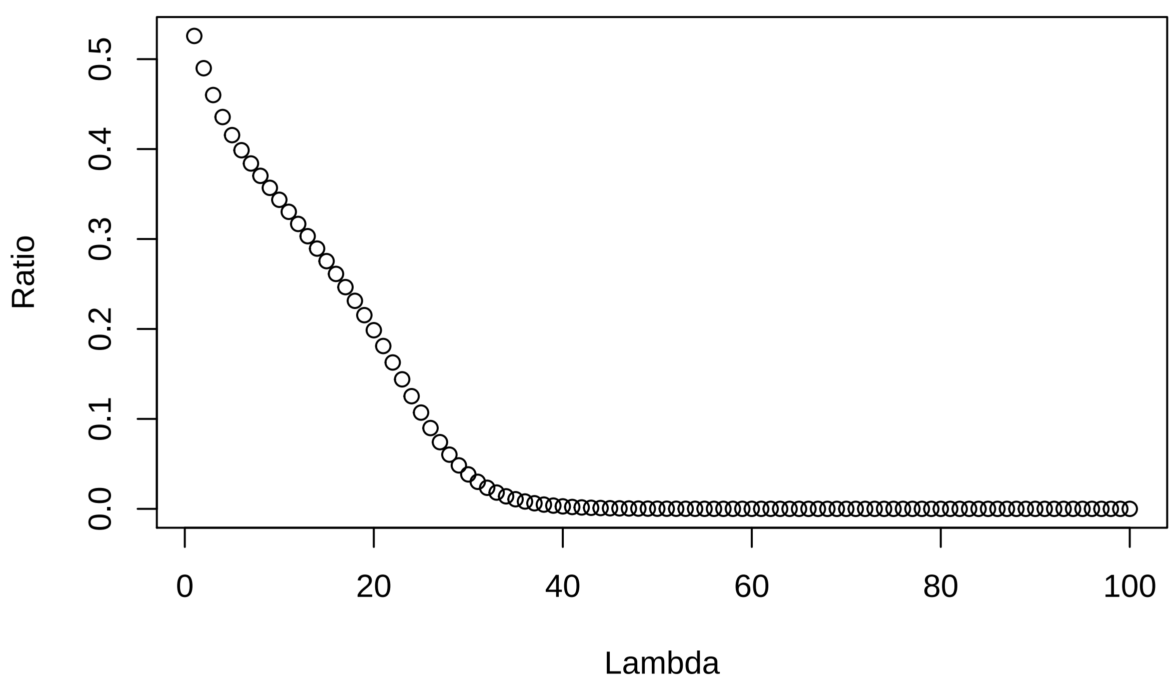
```
dat.std<-as.data.frame(scale(dat))
x=model.matrix(price~.,data=dat)[,-1]
y=dat$price
x.std=model.matrix(price~.,data = dat.std)[,-1]
y.std=dat.std$price
cv.out=cv.glmnet(x.std[train,],y.std[train],alpha=0)
best.lambda=cv.out$lambda.min
ridge.mod=glmnet(x.std[train,],y.std[train],alpha=0,lambda = best.lambda)
coef(ridge.mod)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)       0.001120623
## bldg.full         0.017355101
## total.full        0.113876171
## land.acres        0.041358664
## percent.brick     0.061539126
## main.living.area  0.024081735
## total.living.area 0.167842608
## basement.area     0.105831028
## att.garage.area   0.079578612
## bathrooms         0.045721012
## bedrooms          0.056777309
## rooms             0.032388295
## year.built        0.366307235
```

```
ridge.pred=predict(ridge.mod,newx=x.std[test,])
mean((ridge.pred-y.std[test])^2)
```

```
## [1] 0.3591787
```

```r
#b
lambdas<-10^seq(10,-2,length=100)
ridge.mod=glmnet(x.std,y.std,alpha=0,lambda = lambdas)
beta<-as.matrix(coef(ridge.mod))[-1,]
size=vector()
for(i in 1:100){
  size[i]=sqrt(sum(beta[,i]^2))
}
beta.lm<-coef(lm(y.std~x.std))[-1]
size.lm<-sqrt(sum(beta.lm^2))
ratio<-size/size.lm
plot(rev(ratio),xlab="Lambda",ylab="Ratio",type="b")
```

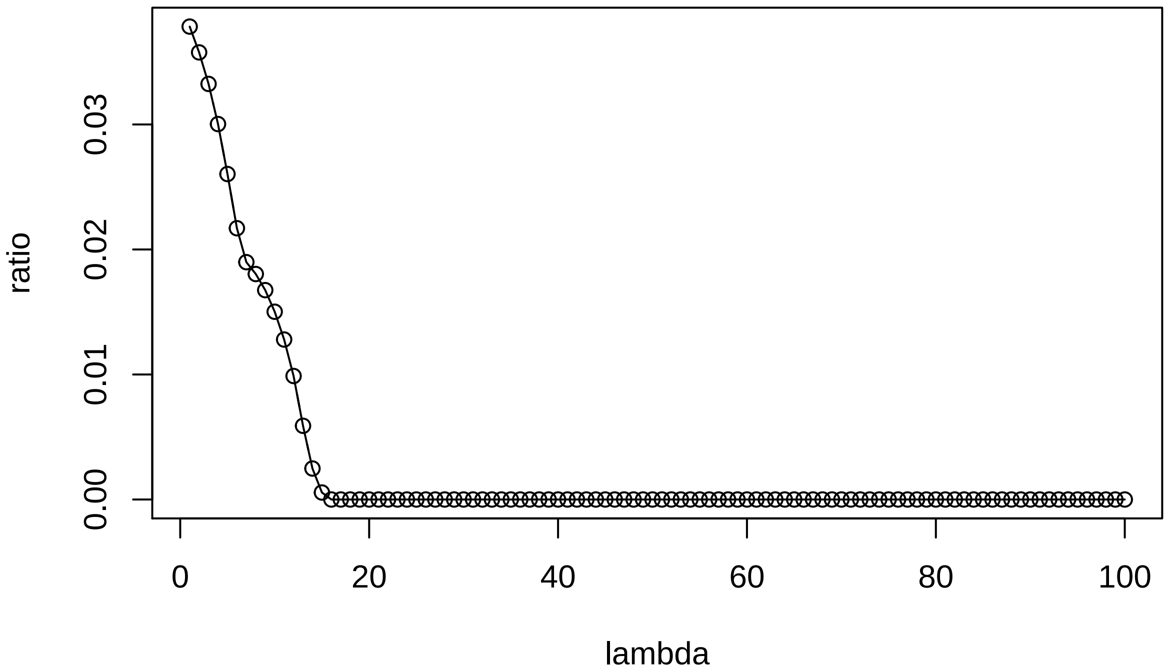



```r
#c lasso
cv.out=cv.glmnet(x.std[train,],y.std[train],alpha=1)
best.lambda=cv.out$lambda.min
lasso.mod<-glmnet(x.std,y.std,alpha=1,lambda=best.lambda)
lasso.pred=predict(lasso.mod,newx=x.std[test,])
mean((lasso.pred-y.std[test])^2)
```

```
## [1] 0.336162
```

```r
#d lasso ratio
lambdas<-10^seq(10,-2,length=100)
lasso.mod<-glmnet(x,y,alpha=1,lambda = lambdas)
beta<-as.matrix(coef(lasso.mod))[-1,]
size=vector()
for(i in 1:100){
  size[i]=sqrt(sum(beta[,i]^2))
}
ratio<-size/size.lm
```

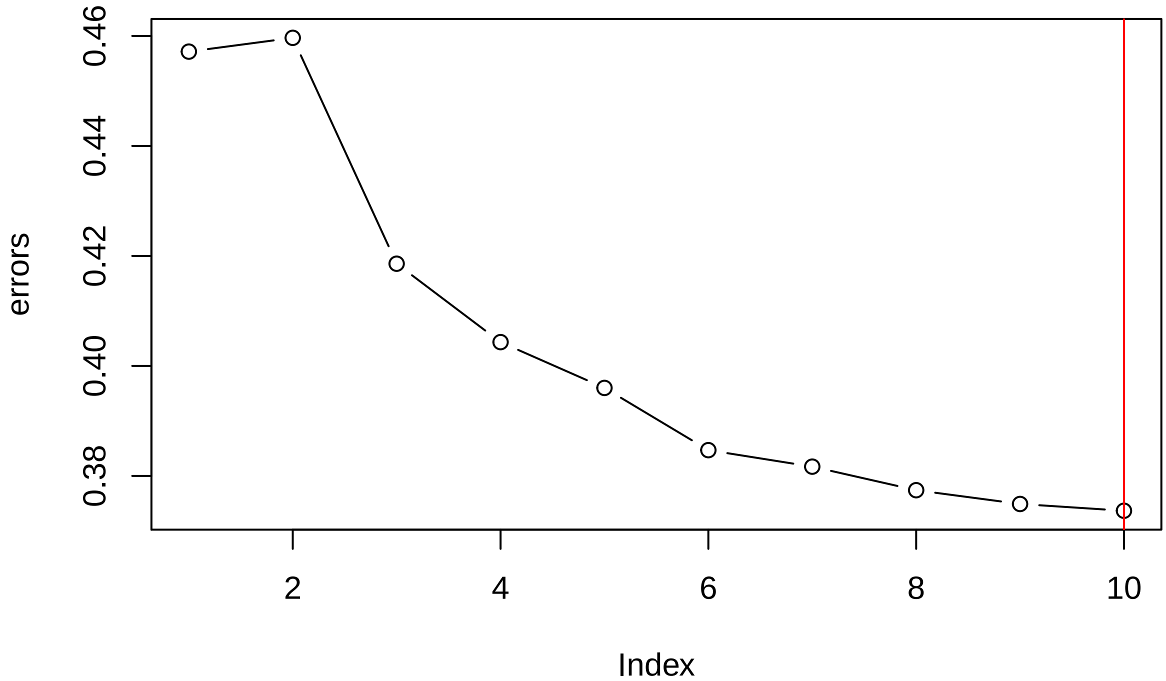
```
plot(rev(ratio),xlab="lambda",ylab = "ratio",type="b")
lines(rev(ratio))
```



Q3)

Use morehouses.csv to answer this question: set.seed(2628) Use K-nearest neighbors to predict the type of foundation, using the other variables. Use 10-fold cross-validation (you'll need to write a function) to choose the best value of K. Base the CV on the overall misclassification rate (so Choose the value of K that minimizes the overall misclassification rate). Consider K=1 through 10.

```
library(class)
set.seed(2628)
dat<-read.csv("/Users/lucy/Downloads/morehouses.csv")
dat=dat[complete.cases(dat),]

#10-fold cross-validation
k<-10
folds<-sample(1:k,nrow(dat),replace = TRUE)
cv.errors=matrix(NA,k,10,dimnames=list(NULL,paste(1:10)))
for(i in 1:10){
  for(j in 1:10){
    m=knn(train=dat[folds!=j,2:14],test=dat[folds==j,2:14],cl=dat[folds!=j,1],k=i)
    t=table(dat[folds==j,1],m);
    error=1-sum(diag(t))/sum(t)
    cv.errors[j,i]=error
  }
}
errors=apply(cv.errors,2,mean)
plot(errors,type='b')
best=which.min(errors)
abline(v=best,col="red")
```



5.4.7

(a) Fit a logistic regression model that predicts Direction using Lag1 and Lag2.
(b) Fit a logistic regression model that predicts Direction using Lag1 and Lag2 using all but the first observation.
(c) Use the model from (b) to predict the direction of the first obser- vation. You can do this by predicting that the first observation will go up if P(Direction="Up"|Lag1, Lag2) > 0.5. Was this ob- servation correctly classified?
(d) Writeaforloopfromi=1toi=n,wherenisthenumberof observations in the data set, that performs each of the following steps:

i. Fit a logistic regression model using all but the ith obser- vation to predict Direction using Lag1 and Lag2.
ii. Compute the posterior probability of the market moving up for the ith observation.
iii. Use the posterior probability for the ith observation in order to predict whether or not the market moves up.
iv. Determine whether or not an error was made in predicting the direction for the ith observation. If an error was made, then indicate this as a 1, and otherwise indicate it as a 0.

(e) Take the average of the n numbers obtained in (d)iv in order to obtain the LOOCV estimate for the test error. Comment on the results.

```
library(ISLR)
summary(Weekly)
```

```
##       Year          Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
```

```
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
#a
glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly, family = binomial)
summary(glm.fit)
```

```
##
## 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
```

```r
#b
glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-1, ], family = binomial)
summary(glm.fit)
```

```
##
## 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
```

```r
#c
predict.glm(glm.fit, Weekly[1, ], type = "response") > 0.5
```

```
##    1
## TRUE
```

```r
#note that Prediction is UP, and true Direction is DOWN
#
#d
count = rep(0, dim(Weekly)[1])
for (i in 1:(dim(Weekly)[1])) {
    glm.fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)
    is_up = predict.glm(glm.fit, Weekly[i, ], type = "response") > 0.5
    is_true_up = Weekly[i, ]$Direction == "Up"
    if (is_up != is_true_up)
        count[i] = 1
}
sum(count)
```

```
## [1] 490
```

```r
#there are 490 errors
#
#e
mean(count)
```

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

```r
#LOOCV estimates a test error rate of 45%.
```

6.8.2

a. iii. Less flexible and better predictions because of less variance, more bias
b. Same as part a, iii.
c. ii. More flexible, less bias, more variance