STAT_101C_HW4

Junhyuk Jang

5/6/2017

SID : 004 728 134 LEC : 2 DIS : 2B

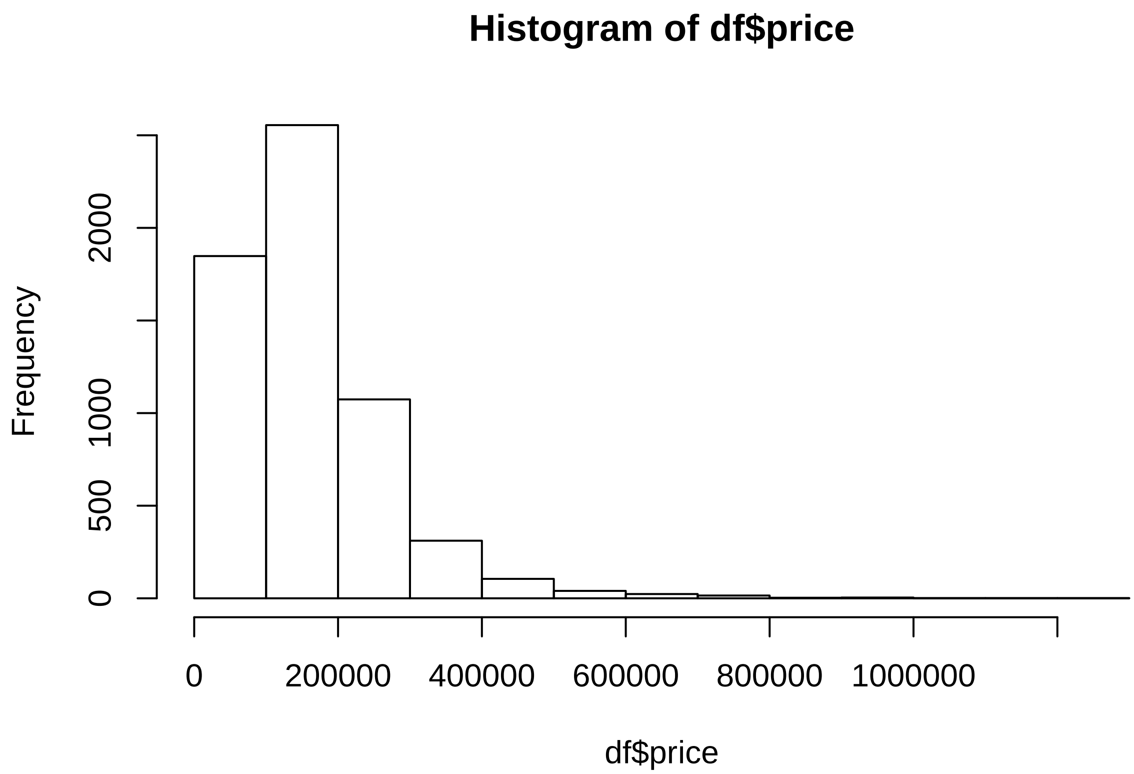
```r
# install.packages("glmnet")
library("readr")
```

```
## Warning: package 'readr' was built under R version 3.2.5
```

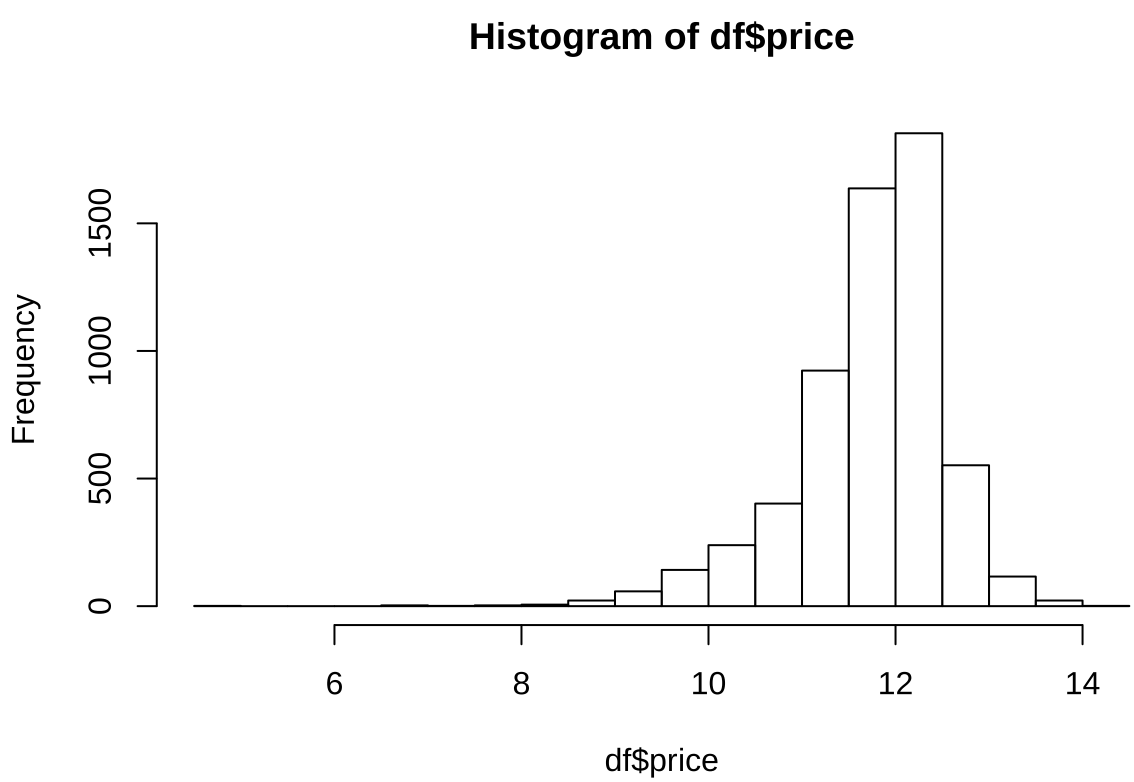
```r
library("leaps")
df <- read.csv("~/Desktop/UCLA_Academic/Spring 2017/STAT 101_C/HW/houses.csv")
df <- na.omit(df)
head(df)
```

```
##    price bldg.full total.full land.acres percent.brick main.living.area
## 1 196000     79800     158300      3.540             0              956
## 2 390000    292900     415400      2.900            50             2415
## 3  78000     81800     119200      0.244             0              860
## 4  90000     57900     111500      0.636             0             1016
## 5 132500     73400     123100      0.603             0              988
## 6 239900    268900     323000      0.282             0             1874
##   total.living.area basement.area att.garage.area bathrooms bedrooms rooms
## 1              1463           567               0         1        3     6
## 2              3852          2263             725         3        5    10
## 3              1300           800               0         1        3     5
## 4              1643           936               0         1        1     6
## 5              1531           988               0         2        4     6
## 6              1874          1678             780         2        1     8
##   year.built
## 1       1910
## 2       1973
## 3       1951
## 4       1900
## 5       1900
## 6       2004
```

```r
# Q1
hist(df$price)
```




```r
df$price <- log(df$price)
hist(df$price)
```



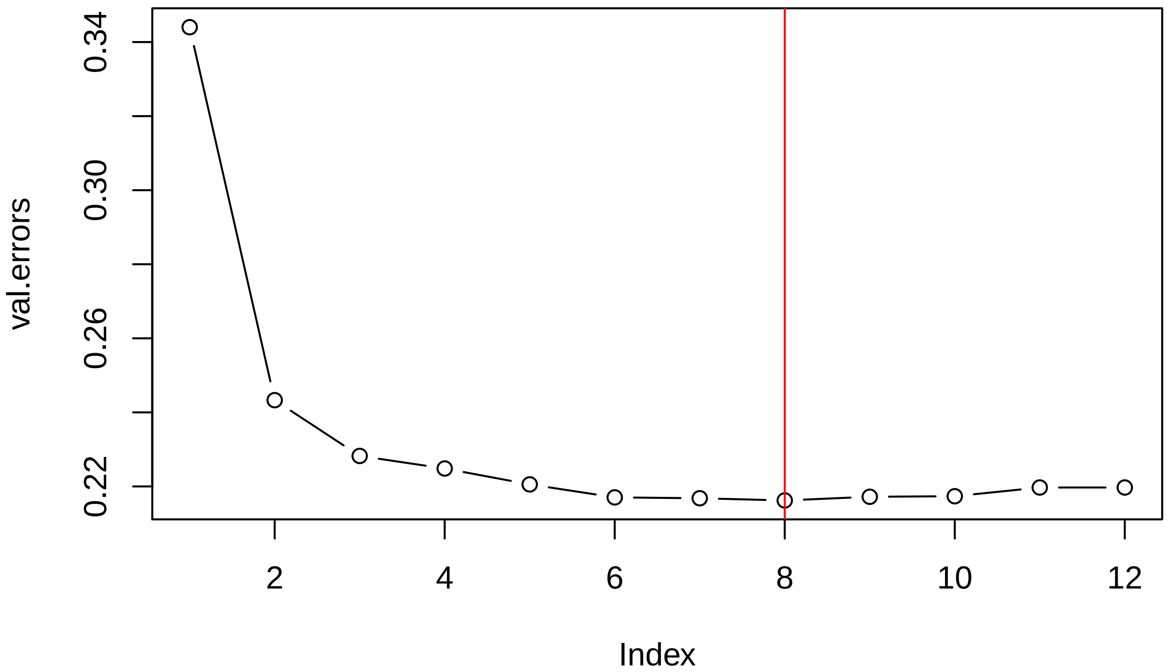
```r
set.seed(2628)

# (a)
# Validation set cross - validation
train <- sample(5981,4186)
test <- (-train)

# Backward stepwise
set.seed(2628)
regfit.best <- regsubsets(price~.,data = df[train,],nvmax = 12,method = "backward" )
test.mat <- model.matrix(price~.,data = df[test,])

predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}

val.errors <- vector()
for (i in 1:12) {
        pred <- predict(regfit.best,df[test,],id=i)
        val.errors[i] <- mean((df$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))
```

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

```r
# model with 8 predictors is the best
plot(val.errors,type = "b")
abline(v = best,col = "red")
```




```
regfit.best=regsubsets(price~.,data=df,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
```

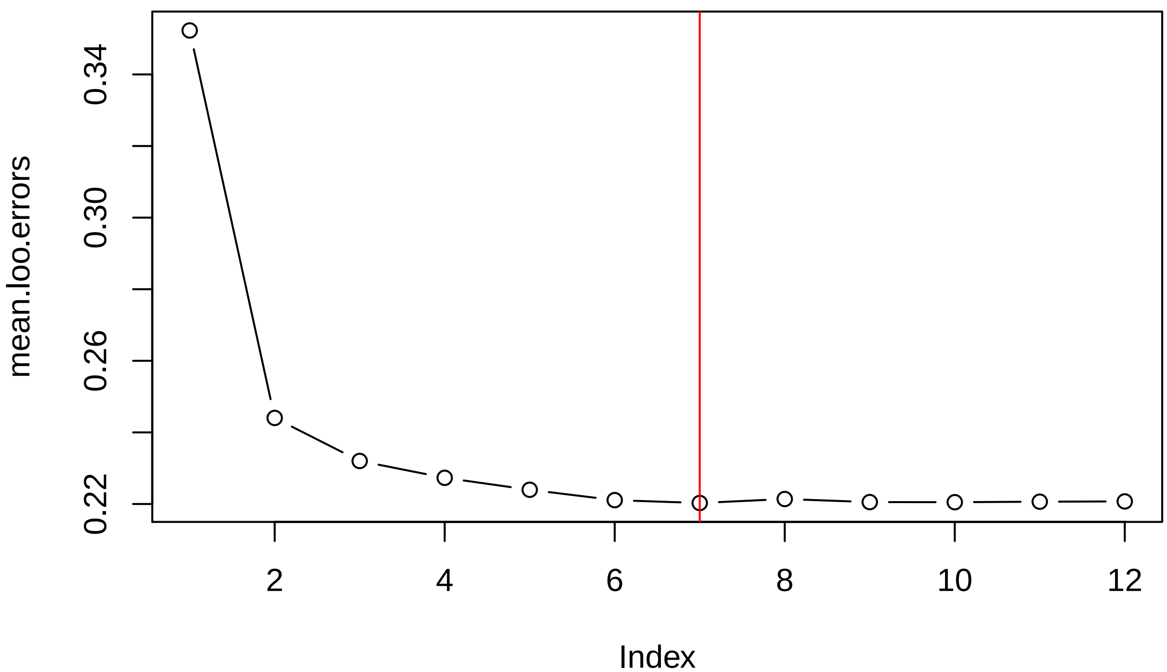
```
# (b) LOOCV
loo.errors <- matrix(NA,nrow(df),12,dimnames = list(NULL,paste(1:12)))
for (j in 1:nrow(df)) {
        best.fit <- regsubsets(price~.,data = df[-j,],nvmax = 12,method = "backward")
        for (i in 1:12) {
                pred = predict(best.fit,df[j,],id = i)
                loo.errors[j,i] = mean((df$price[j] - pred)^2)
        }
}
(mean.loo.errors <- apply(loo.errors, 2, mean))
```

```
##         1         2         3         4         5         6         7
## 0.3522793 0.2440412 0.2320200 0.2273186 0.2239744 0.2210835 0.2202794
##         8         9        10        11        12
## 0.2213914 0.2205354 0.2205086 0.2206457 0.2207239
```

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

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

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




```r
reg.best <- regsubsets(price~.,data = df,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
# model with 7 predictors is the best


# (c) 10 FOLD CROSS VALIDATION
set.seed(2628)
k=10
folds=sample(1:k,nrow(df),replace = TRUE);table(folds)
```

```
## folds
##   1   2   3   4   5   6   7   8   9  10
## 630 607 622 630 553 553 614 593 589 590
```

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

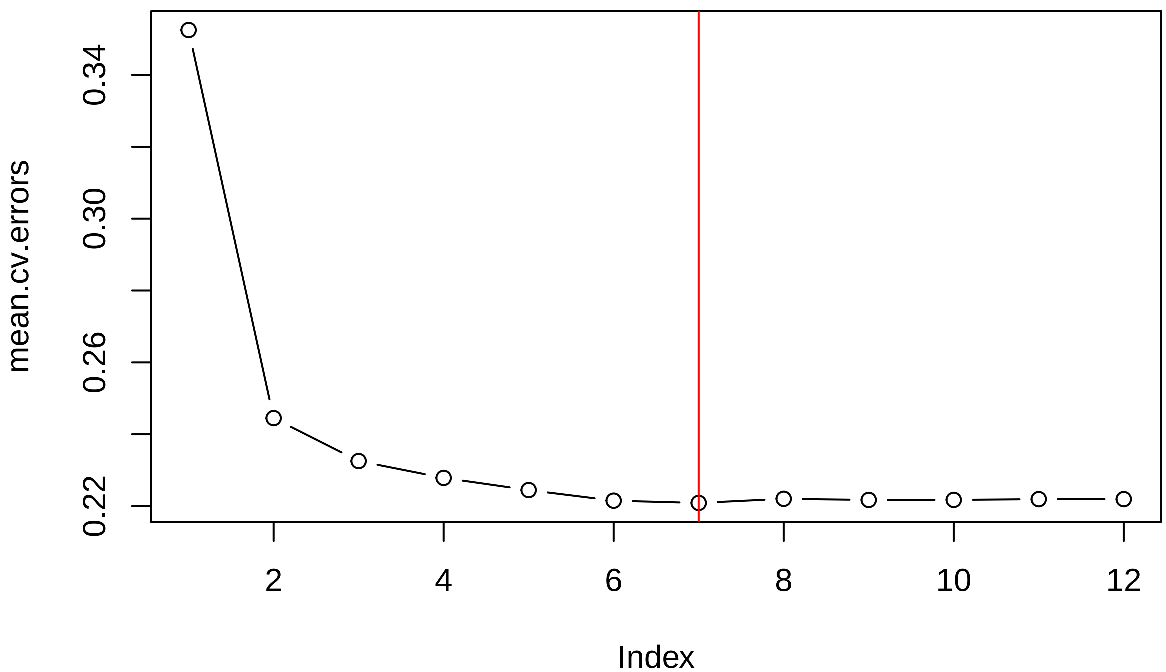
```r
}
mean.cv.errors=apply(cv.errors,2,mean)
mean.cv.errors
```

```
##         1         2         3         4         5         6         7
## 0.3524750 0.2445052 0.2325506 0.2278610 0.2244760 0.2215248 0.2208983
##         8         9        10        11        12
## 0.2220425 0.2217330 0.2217395 0.2219527 0.2219596
```

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

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

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




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

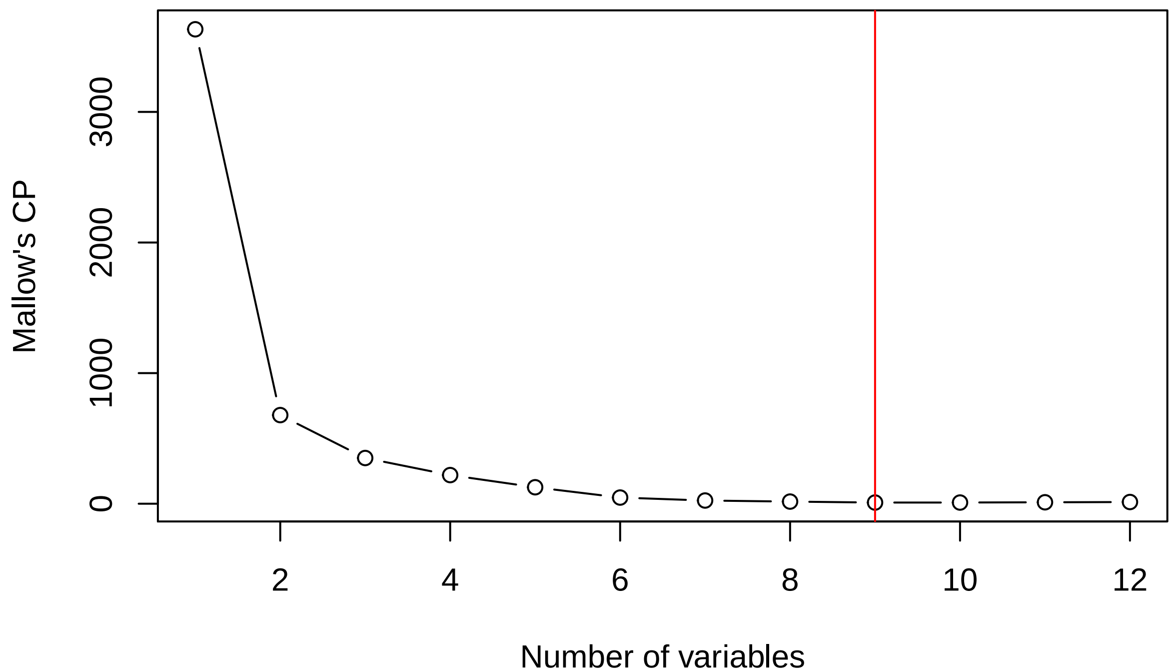
```
##       (Intercept)         bldg.full        total.full     percent.brick
##     -9.277406e+00     -6.179780e-06      6.124659e-06      2.297808e-03
## total.living.area     basement.area          bedrooms        year.built
##      3.572355e-04      2.365283e-04      4.972832e-02      1.012144e-02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

```r
# (d) Mallow-CP
best.fit <- regsubsets(price~.,data = df,nvmax = 12,method = "backward")
reg.summary <- summary(best.fit)
(best <- which.min(reg.summary$cp))
```

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

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

# validation set apprach is easy to implement but denping on which observation
# is included in traing and testing dataset, it can be highly variable.
# LOOCV is not only hard to comupte but also lower accuracy rate than K-fold c.v
# since LOOCV's simulating error is high. Also, LOOCV has redundency problem.
# The main disadvantage of Mallows-CP need to supply an estimate of variance.
# The main disadvantage of K-fold cv is that the training algorithm has to be
# rerun k times.

# Q2
# (a) Ridge regression
library("glmnet")
```

```
## Warning: package 'glmnet' was built under R version 3.2.5

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-10
```




```r
library("class")
df.std <- as.data.frame(scale(df))
x = model.matrix(price~.,data = df)[,-1]
y = df$price
x.std <- model.matrix(price~.,data = df.std)[,-1]
```

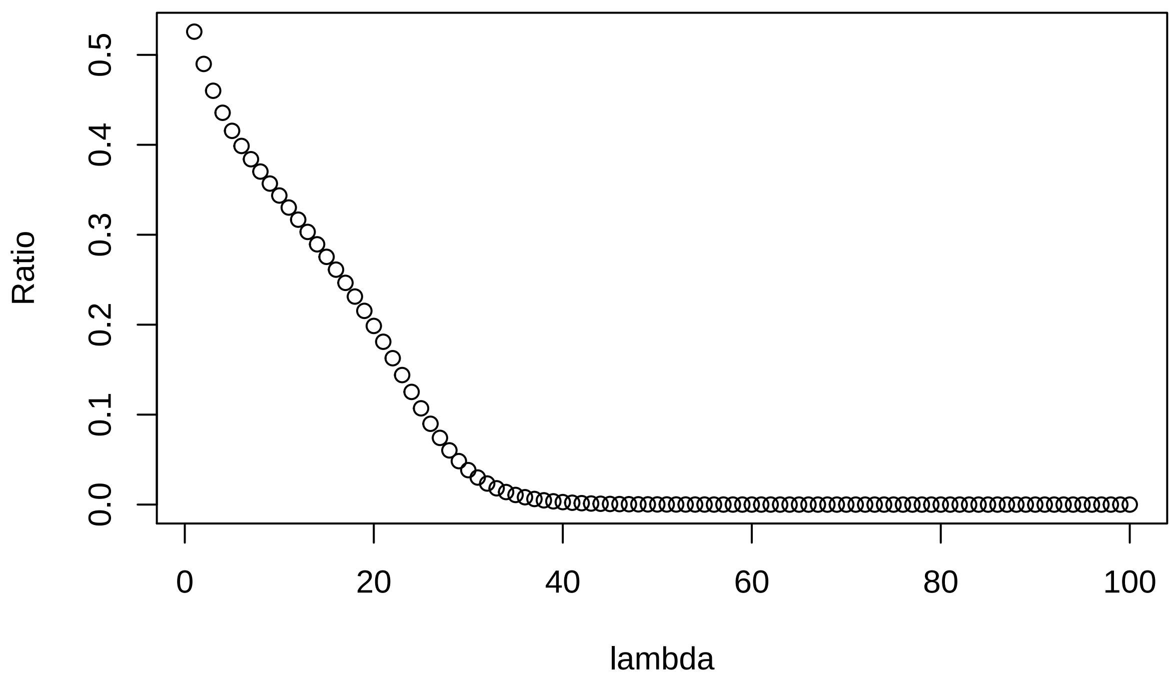
```r
y.std <- df.std$price
cv.out <- cv.glmnet(x.std[train,],y.std[train],alpha=0)
best.lambda <- cv.out$lambda.min
ridge.mod <- glmnet(x.std[train,],y.std[train],alpha = 0,lambda = best.lambda)
coef(ridge.mod)
```

```
## 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
```

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

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

```r
# Since all the estimated coefficient is positive, we can say all predictors have
# increasing effect on response variable which is a log(price).

# (b) Ridge ratio
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")
```



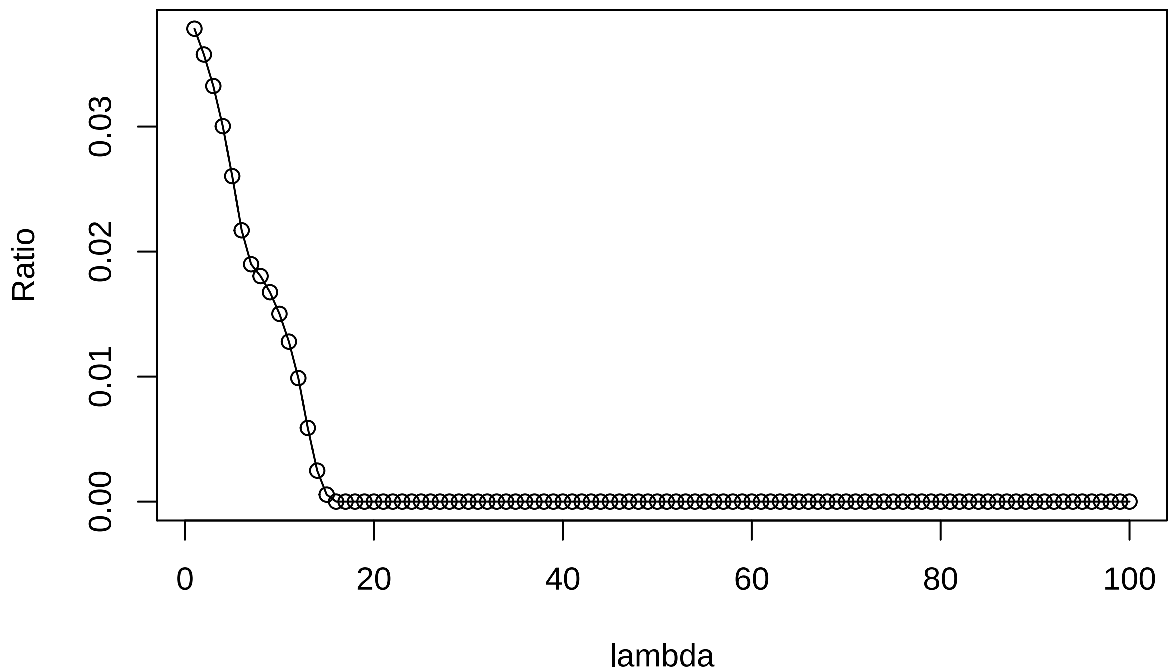

```r
# As lamda increases and passes a certain point it is not changing and
# keep constatly spreading out. Since ridge regression is not a model
# selection statistical method. There is no reduced model.

# (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)
coef(lasso.mod)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)        7.987436e-16
## bldg.full         -6.672560e-01
## total.full         7.934877e-01
## land.acres        -2.207221e-02
## percent.brick      7.220202e-02
## main.living.area   5.624208e-04
## total.living.area  2.479855e-01
## basement.area      1.317573e-01
## att.garage.area    2.068475e-02
## bathrooms          3.398851e-02
## bedrooms           4.758672e-02
## rooms              3.029741e-03
## year.built         4.168472e-01
```

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

```
## [1] 0.3361912
```

```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))
```

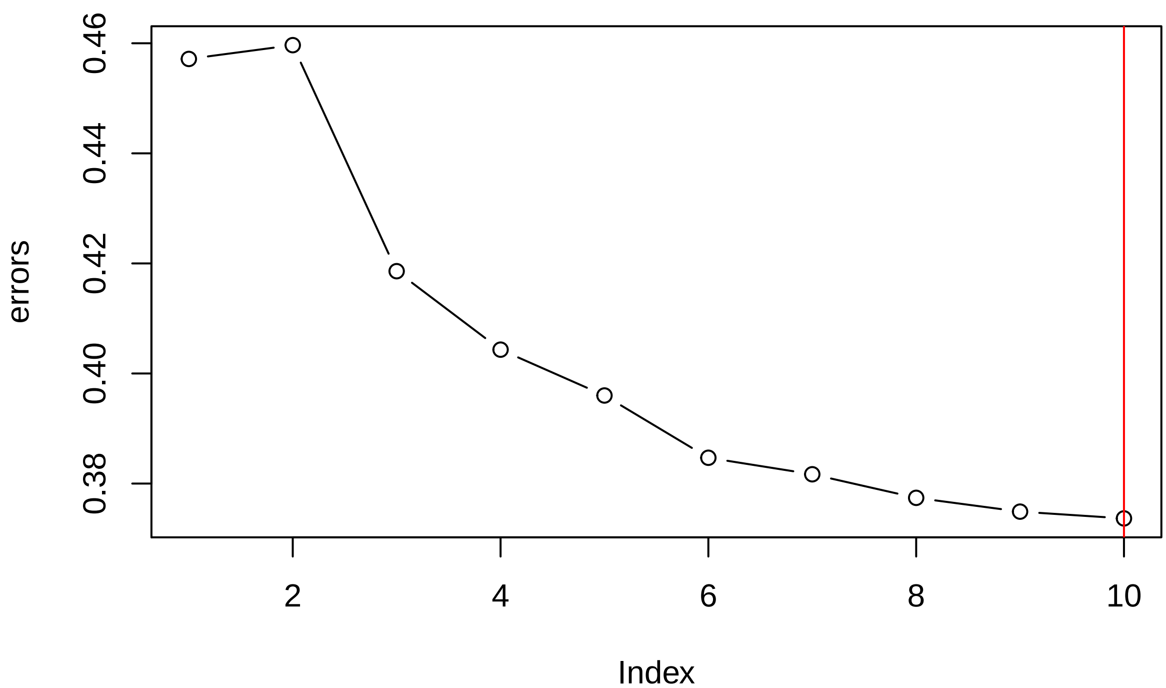



```r
# Except predictors bldg.full and land.acres, other predictors have a increasing
# effect on log(price). In other words, as bldg.full and land.acres increases,
# log(price) will decrease.
# Also unlike to ridge regression, lasso have a power to reduce the model.
# Lasso can shrink coefficient equals to zero.

# Q3
library(class)
set.seed(2628)
df1 <- read.csv("~/Desktop/UCLA_Academic/Spring 2017/STAT 101_C/HW/morehouses.csv")
df1 <- na.omit(df1)
#k=10 fold k from 1 to 10
k=10
folds=sample(1:k,nrow(df1),replace=TRUE)
cv.errors=matrix(NA,k,10,dimnames = list(NULL, paste(1:10)))
for(i in 1:10){
  for(j in 1:k){
    m = knn(train=df1[folds!=j,2:14],test=df1[folds==j,2:14],
            cl=df1[folds!=j,1],k=i)
    t=table(df1[folds==j,1],m)
```

```r
    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);best
```

```
## 10
## 10
```

```r
abline(v=best,col="red")
```




```r
# Q4
# (a)
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
```

```
attach(Weekly)

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
```

```
# (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
# prediction is greater 0.5 but true direction is not.

# (d)
v = rep(0, dim(Weekly)[1])
for (i in 1:(dim(Weekly)[1])) {
    glm_fit = glm(Direction ~ Lag1 + Lag2, data = Weekly[-i, ], family = binomial)
    up = predict.glm(glm_fit, Weekly[i, ], type = "response") > 0.5
    true_up = Weekly[i, ]$Direction == "Up"
    if (up != true_up)
        v[i] = 1
}
sum(v)
```

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

```r
# We have 490 errors

# (e)
mean(v)
```

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

```r
# Q5
# (a)
# Lasso is less flexible method so it gives good result in high dimensionality
# situation. In this question, the answer is (iii). Since Lasso is a inflexible
# statistical learning method, if we make it more flexible, the variance would be
# decrease and the bias would be increase. So we have to find the point WHERE minimizes
# the MSE.
# (b)
# same as (a). Ridege regression is also infelxible statistical learning method.
```

```
# Answer would be (III)
# (c)
# Non-linear method is more flexible realative to least square. So it has higher
# variance and lower bias than least squares. So we can get the best result when
# increase in variance is less than its decrese in bias.
```