# DS303 HW4

Shigeki Kanamori

10/28/2020

```
library(ISLR)
```

```
### Problem 1 ###
head(Hitters)
```

```
##                   AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits
## -Andy Allanson      293   66     1   30  29    14     1    293    66
## -Alan Ashby         315   81     7   24  38    39    14   3449   835
## -Alvin Davis        479  130    18   66  72    76     3   1624   457
## -Andre Dawson       496  141    20   65  78    37    11   5628  1575
## -Andres Galarraga   321   87    10   39  42    30     2    396   101
## -Alfredo Griffin    594  169     4   74  51    35    11   4408  1133
##                   CHmRun CRuns CRBI CWalks League Division PutOuts Assists
## -Andy Allanson         1    30   29     14      A        E     446      33
## -Alan Ashby           69   321  414    375      N        W     632      43
## -Alvin Davis          63   224  266    263      A        W     880      82
## -Andre Dawson        225   828  838    354      N        E     200      11
## -Andres Galarraga     12    48   46     33      N        E     805      40
## -Alfredo Griffin      19   501  336    194      A        W     282     421
##                   Errors Salary NewLeague
## -Andy Allanson        20     NA         A
## -Alan Ashby           10  475.0         N
## -Alvin Davis          14  480.0         A
## -Andre Dawson          3  500.0         N
## -Andres Galarraga      4   91.5         N
## -Alfredo Griffin      25  750.0         A
```

```
Hitters = na.omit(Hitters)
n = nrow(Hitters)
X = model.matrix(Salary ~.,data=Hitters)[,-1]

Y = Hitters$Salary

set.seed(30)
train = sample(1:nrow(X), nrow(X)/2)
test=(-train)
Y.test = Y[test]
```

```
#(b) least squares model
```

```r
ls = lm(Salary~., data=Hitters[train,])
summary(ls)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = Hitters[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -615.76 -172.03  -31.47  124.90 1712.35
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  255.6645   125.0451   2.045 0.043262 *
## AtBat         -3.0027     0.8878  -3.382 0.000994 ***
## Hits          11.1568     3.7698   2.960 0.003767 **
## HmRun         -3.8772     8.6465  -0.448 0.654727
## Runs         -10.9112     4.3580  -2.504 0.013745 *
## RBI            4.7941     3.6886   1.300 0.196394
## Walks         10.1544     2.8891   3.515 0.000638 ***
## Years        -34.0117    16.7885  -2.026 0.045174 *
## CAtBat         0.1122     0.2049   0.547 0.585199
## CHits          0.2413     1.1097   0.217 0.828233
## CHmRun         6.4298     2.6202   2.454 0.015683 *
## CRuns          1.4350     1.0569   1.358 0.177307
## CRBI          -2.0985     1.1305  -1.856 0.066062 .
## CWalks        -1.0434     0.5329  -1.958 0.052728 .
## LeagueN       -8.8687   119.8827  -0.074 0.941161
## DivisionW   -120.9895    57.5613  -2.102 0.037822 *
## PutOuts        0.4812     0.1196   4.024 0.000105 ***
## Assists        0.7209     0.3085   2.336 0.021261 *
## Errors        -8.5448     6.8071  -1.255 0.212010
## NewLeagueN   130.8622   119.9319   1.091 0.277575
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 300.3 on 111 degrees of freedom
## Multiple R-squared:  0.6675, Adjusted R-squared:  0.6105
## F-statistic: 11.73 on 19 and 111 DF,  p-value: < 2.2e-16
```

```r
yhat = predict(ls,newdata=Hitters[test,])

mean((yhat-Y.test)^2)
```

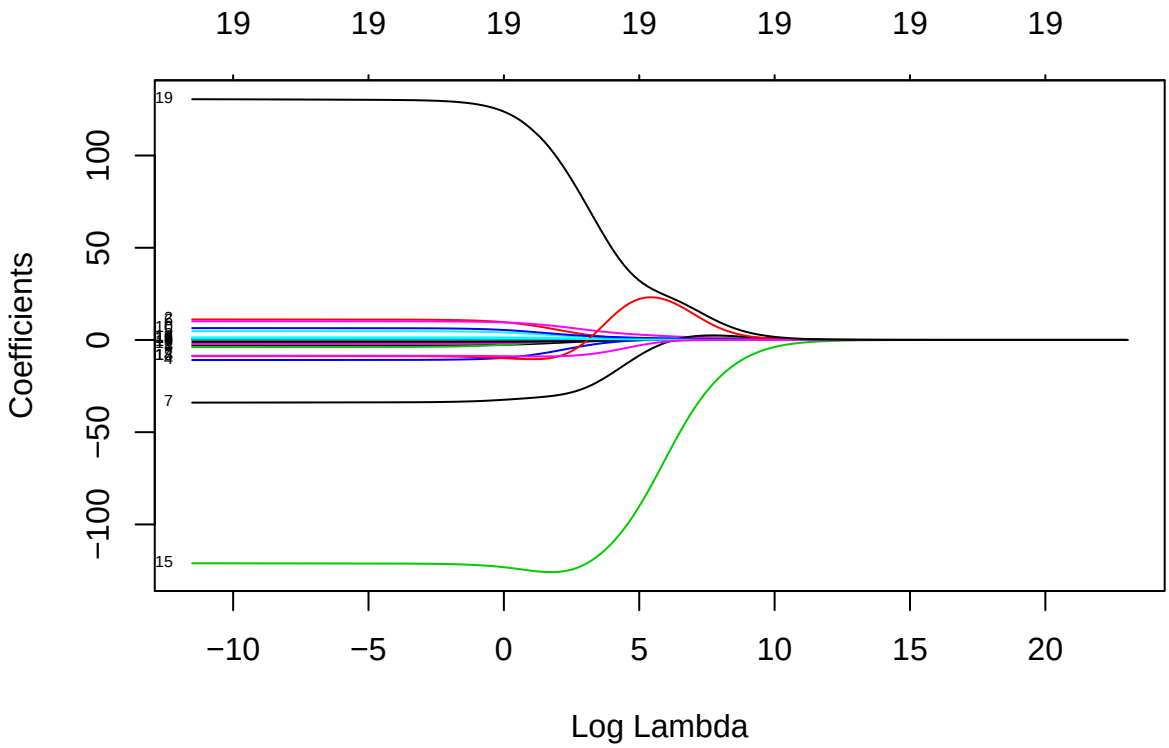
```
## [1] 196387.4
```

```r
#(d)

library(glmnet)
```

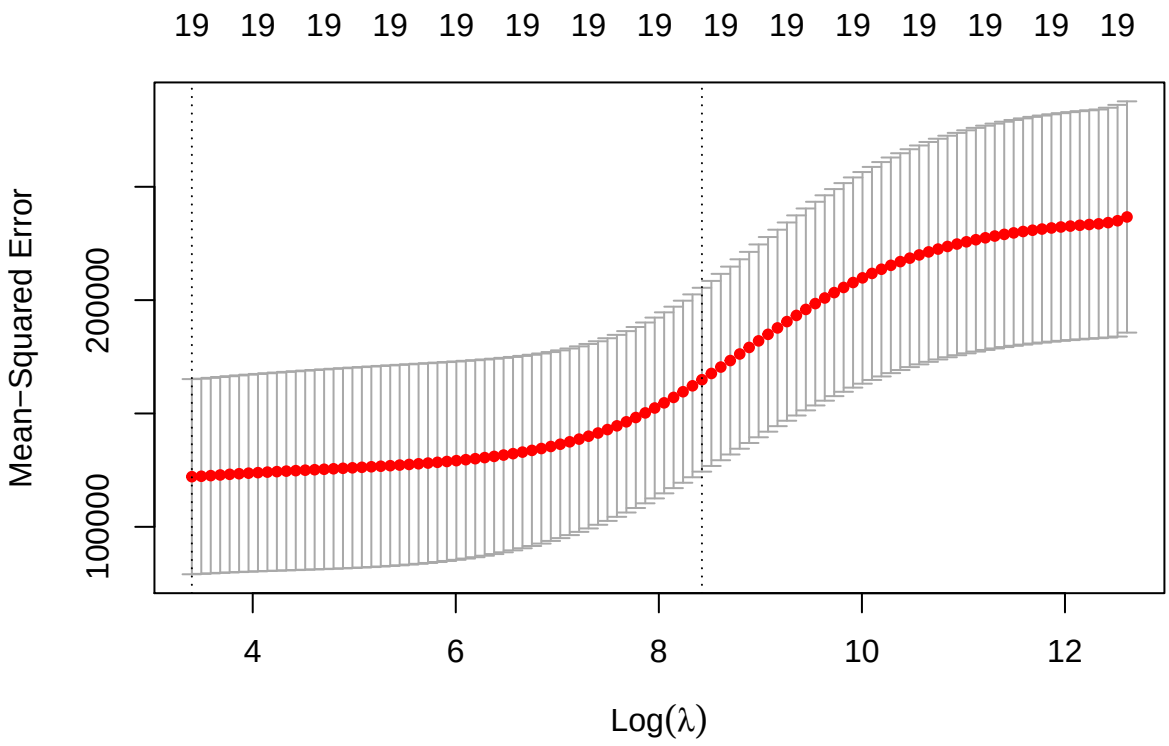
```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
lambdagrid = 10^seq(10,-5,length=200)

ridge.train = glmnet(X[train,],Y[train],alpha=0,lambda=lambdagrid)
plot(ridge.train,xvar="lambda",label=TRUE)
```




```
#(e)
cv.out = cv.glmnet(X[train,],Y[train],alpha=0)
plot(cv.out)
```



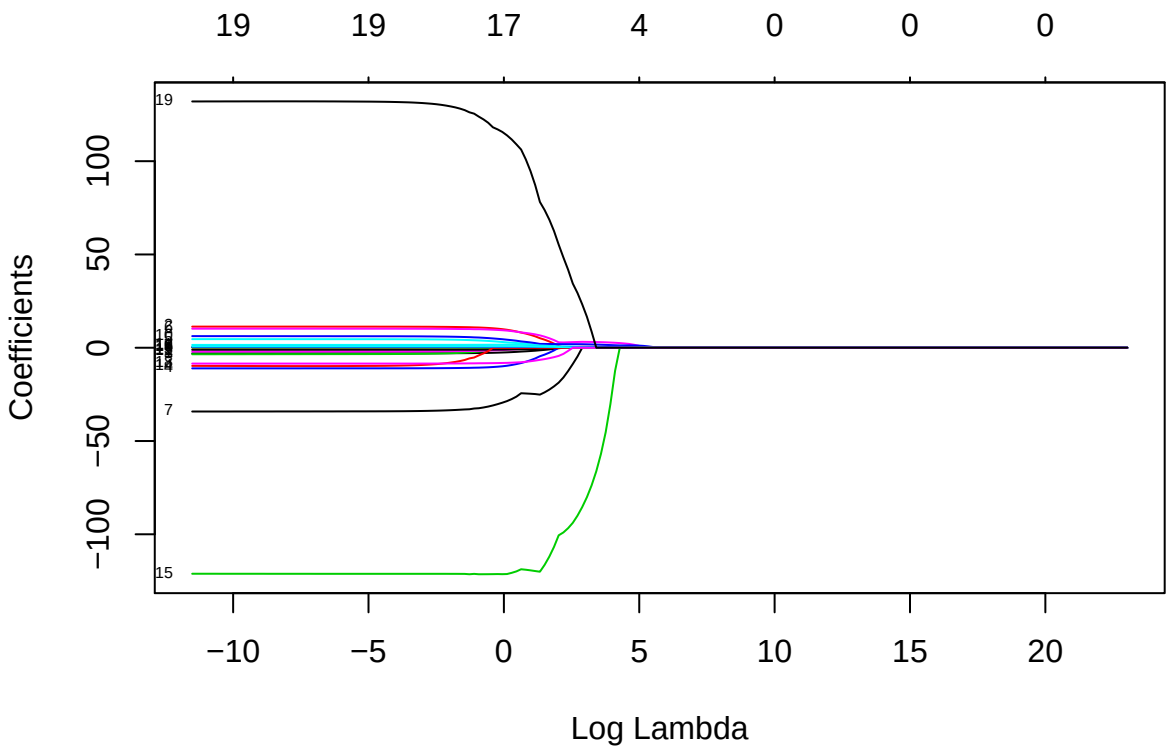

```
lambda_min = cv.out$lambda.min
lambda_min
```

```
## [1] 29.98169
```

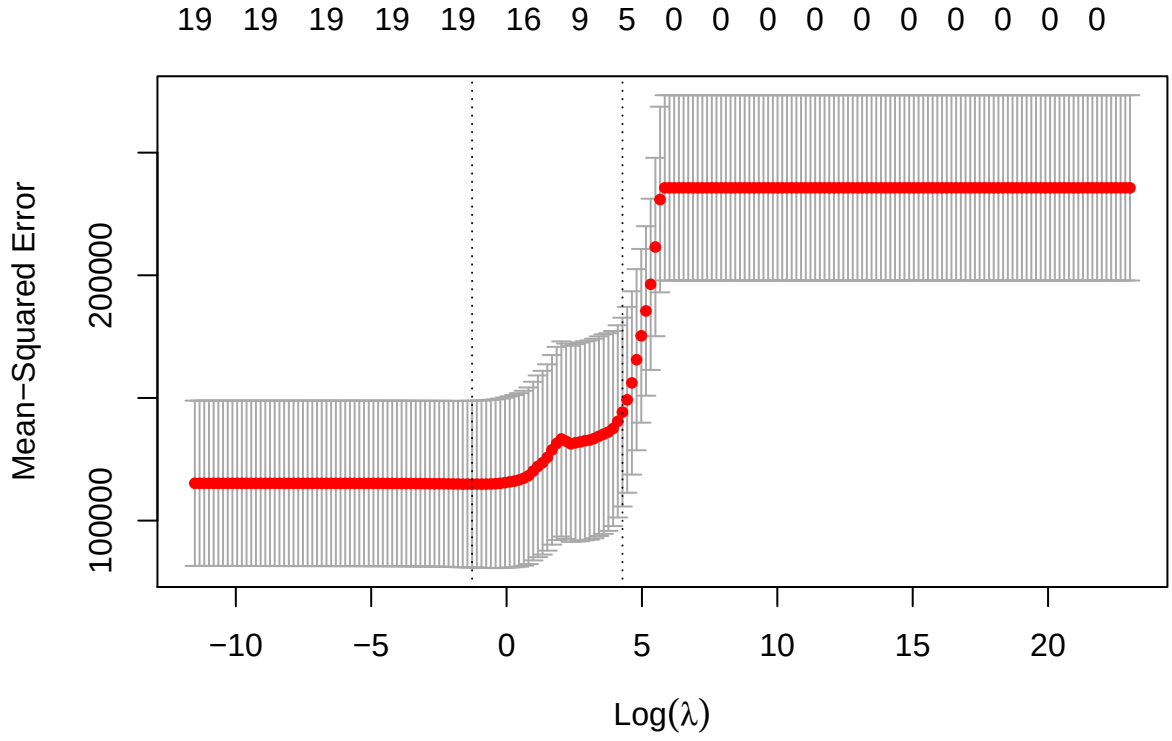
```
#(f)
lambda_1se = cv.out$lambda.1se
lambda_1se
```

```
## [1] 4556.95
```

```
#(g)
lasso.train = glmnet(X[train,],Y[train],alpha=1,lambda=lambdagrid,)
plot(lasso.train,xvar="lambda",label=TRUE )
```




```
#(h)

cv.out.lasso = cv.glmnet(X[train,],Y[train],alpha=1, lambda = lambdagrid)
plot(cv.out.lasso)
```



```
lambda_lasso_min = cv.out.lasso$lambda.min
lambda_lasso_min
```

```
## [1] 0.2800504
```

```
lambda_lasso_1se = cv.out.lasso$lambda.1se
lambda_lasso_1se
```

```
## [1] 72.32634
```

```
#(i)
ridge.pred = predict(ridge.train,s = lambda_min,newx=X[test,])
mean((ridge.pred-Y.test)^2)
```

```
## [1] 140199.1
```

```
ridge.pred2 = predict(ridge.train,s = lambda_1se,newx=X[test,])
mean((ridge.pred2-Y.test)^2)
```

```
## [1] 123127.9
```

```
lasso.pred1 = predict(lasso.train,s = lambda_lasso_min,newx=X[test,])
mean((lasso.pred1-Y.test)^2)
```

```
## [1] 189087
```

```
lasso.pred2 = predict(lasso.train,s = lambda_lasso_1se,newx=X[test,])
mean((lasso.pred2-Y.test)^2)
```

```
## [1] 126704.4
```

```
### Problem 2 ###
spam = read.csv('spambase/spambase.data',header=FALSE)

head(spam)
```

```
##     V1   V2   V3 V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14  V15
## 1 0.00 0.64 0.64  0 0.32 0.00 0.00 0.00 0.00 0.00 0.00 0.64 0.00 0.00 0.00
## 2 0.21 0.28 0.50  0 0.14 0.28 0.21 0.07 0.00 0.94 0.21 0.79 0.65 0.21 0.14
## 3 0.06 0.00 0.71  0 1.23 0.19 0.19 0.12 0.64 0.25 0.38 0.45 0.12 0.00 1.75
## 4 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
## 5 0.00 0.00 0.00  0 0.63 0.00 0.31 0.63 0.31 0.63 0.31 0.31 0.31 0.00 0.00
## 6 0.00 0.00 0.00  0 1.85 0.00 0.00 1.85 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##    V16  V17  V18  V19  V20  V21 V22  V23  V24 V25 V26 V27 V28 V29 V30 V31
## 1 0.32 0.00 1.29 1.93 0.00 0.96   0 0.00 0.00   0   0   0   0   0   0   0
## 2 0.14 0.07 0.28 3.47 0.00 1.59   0 0.43 0.43   0   0   0   0   0   0   0
## 3 0.06 0.06 1.03 1.36 0.32 0.51   0 1.16 0.06   0   0   0   0   0   0   0
## 4 0.31 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0
```

```
## 5 0.31 0.00 0.00 3.18 0.00 0.31   0 0.00 0.00   0   0   0   0   0   0   0
## 6 0.00 0.00 0.00 0.00 0.00 0.00   0 0.00 0.00   0   0   0   0   0   0   0
##   V32 V33 V34 V35 V36  V37 V38 V39  V40 V41 V42  V43 V44  V45  V46 V47 V48
## 1   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 2   0   0   0   0   0 0.07   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 3   0   0   0   0   0 0.00   0   0 0.06   0   0 0.12   0 0.06 0.06   0   0
## 4   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 5   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
## 6   0   0   0   0   0 0.00   0   0 0.00   0   0 0.00   0 0.00 0.00   0   0
##    V49   V50 V51   V52   V53   V54   V55 V56  V57 V58
## 1 0.00 0.000   0 0.778 0.000 0.000 3.756  61  278   1
## 2 0.00 0.132   0 0.372 0.180 0.048 5.114 101 1028   1
## 3 0.01 0.143   0 0.276 0.184 0.010 9.821 485 2259   1
## 4 0.00 0.137   0 0.137 0.000 0.000 3.537  40  191   1
## 5 0.00 0.135   0 0.135 0.000 0.000 3.537  40  191   1
## 6 0.00 0.223   0 0.000 0.000 0.000 3.000  15   54   1
```

```
table(spam$V58)
```

```
##
##    0    1
## 2788 1813
```

```
1813/(2788+1813)
```

```
## [1] 0.3940448
```

```
standardized.X = as.data.frame(scale(spam[,-58]))
var(standardized.X[,1])
```

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

```
# training and testing set
set.seed(1)
spam.index = which(spam$V58 ==1)
nonspam.index = which(spam$V58 ==0)
```

```
#4601/2

num_yes = round(2301*(1813/4601))
num_yes
```

```
## [1] 907
```

```
num_no = round(2301*(2788/4601))
num_no
```

```
## [1] 1394
```

```r
train.yes = sample(spam.index, num_yes)
train.no = sample(nonspam.index , num_no)
train = c(train.yes,train.no) #index

train.X = standardized.X[train,]
dim(train.X)
```

```
## [1] 2301   57
```

```r
train.Y = spam$V58[train]
train.data = as.data.frame(cbind(train.X,train.Y))

test.X = standardized.X[-train,]
dim(test.X)
```

```
## [1] 2300   57
```

```r
test.Y = spam$V58[-train]
test.data = as.data.frame(cbind(test.X,test.Y))
```

```r
#logistic
log.fit = glm(train.Y~.,data=train.data,family='binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
probs = predict(log.fit,test.X,type='response')
log.pred = rep(0, 2300)
log.pred[probs>0.5] = 1
```

```r
table(log.pred, test.Y)
```

```
##         test.Y
## log.pred    0    1
##        0 1319   93
##        1   75  813
```

```r
mean(log.pred!=test.Y)
```

```
## [1] 0.07304348
```

```r
log.pred = rep(0, 2300)
log.pred[probs>0.7] = 1

table(log.pred, test.Y)
```

```
##         test.Y
## log.pred    0    1
##        0 1351  153
##        1   43  753
```

```r
mean(log.pred!=test.Y)
```

```
## [1] 0.08521739
```