HW3

1. 2 from section 6.8

(a)

iii. Lasso regression has restrictions on the parameters while least squares don’t. The restrictions prevent the model from overfitting so bias will increase, and variance will decrease.

(b)

iii. Ridge regression has restrictions on the parameters while least squares don’t. The restrictions prevent the model from overfitting so bias will increase, and variance will decrease.

(c)

ii Non-linear methods are more flexible and can fit the training set better than linear models. So bias will decrease and variance increase.

2. Read the lab 6.6 , page 251-255, and repeat the steps for ridge and lasso regression

1. Import and prepare the data.

```
library (ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.6.2
```

```
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
```

```
# Drop missing rows
Hitters =na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```
sum(is.na(Hitters$Salary))
```

```
## [1] 0
```

2. Ridge Regression

```
library (glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

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

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

```
grid=10^seq(10,-2, length =100)
# generate training set and testing set
set.seed(666)
train=sample (1: nrow(Hitters), nrow(Hitters)/2)
test=(-train)
# Split x and y
Hitters.train=Hitters[train,]
Hitters.test=Hitters[test,]
x.train=model.matrix(Salary ~ .,Hitters.train)[,-1]
y.train=Hitters.train$Salary
x.test=model.matrix(Salary ~ .,Hitters.test)[,-1]
y.test=Hitters.test$Salary

# basic ridge model
ridge.mod=glmnet(x.train,y.train,alpha=0,lambda=grid, thresh=1e-12)

# randomly choose a lambda
ridge.pred1=predict(ridge.mod, x=x.train,y=y.train,s=4, newx=x.test)
mean((ridge.pred1 -y.test)^2)
```

```
## [1] 137923.8
```

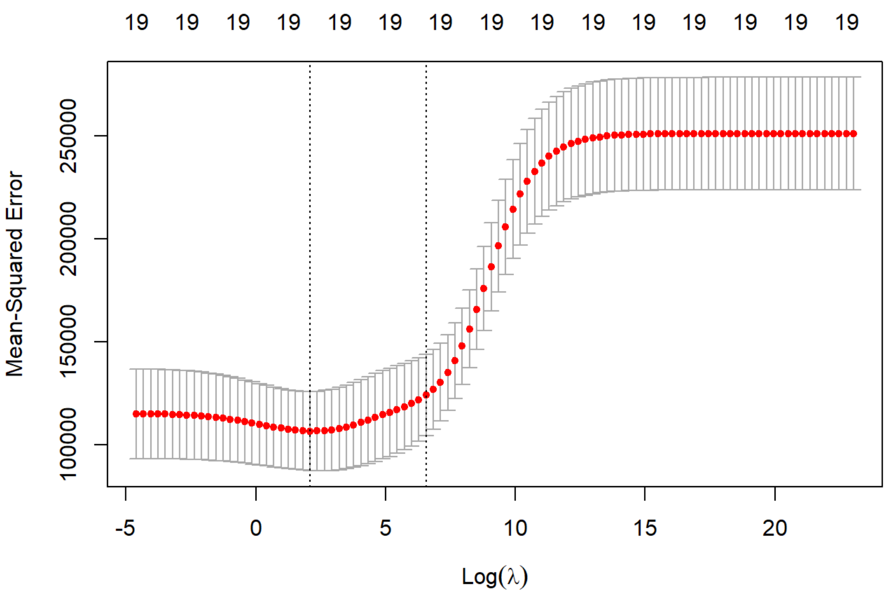
```
ridge.pred2=predict(ridge.mod, x=x.train,y=y.train,s=0, newx=x.test,exact = TRUE)
mean((ridge.pred2 -y.test)^2)
```

```
## [1] 140249.4
```

```
ridge.pred3=predict(ridge.mod, x=x.train,y=y.train,s=10000, newx=x.test,exact = TRUE)
mean((ridge.pred3 -y.test)^2)
```

```
## [1] 128746.4
```

```
# use cross validation to choose the best lambda
set.seed(666)
cv.out=cv.glmnet(x.train,y.train,alpha=0,lambda=grid, thresh=1e-12)
plot(cv.out)
```




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

```
## [1] 8.111308
```

```
ridge.pred=predict (ridge.mod ,s=bestlam ,newx=x.test)
mean((ridge.pred -y.test)^2)
```

```
## [1] 136333.2
```

```
# fit the Ridge Regression model with the best lambda and total data
x=model.matrix(Salary ~ .,Hitters)[,-1]
ridge.tot=glmnet(x,Hitters$Salary,alpha=0,lambda=bestlam)
ridge.coef = predict(ridge.tot ,type="coefficients",s=bestlam)[1:20,]
ridge.coef
```

```
##    (Intercept)         AtBat          Hits         HmRun          Runs
##   129.94224681   -1.31685188    4.64575618   -0.19109424    0.27390385
##            RBI         Walks         Years        CAtBat         CHits
##     0.30629715    4.61893745  -10.81131717   -0.02973118    0.17084274
##         CHmRun         CRuns          CRBI        CWalks       LeagueN
##     0.68251377    0.52542940    0.34983683   -0.49772302   59.69106459
##      DivisionW       PutOuts       Assists        Errors    NewLeagueN
## -123.99632137    0.27566152    0.25069713   -3.83941842  -26.25974228
```

```
ridge.tot.pred = model.matrix(Salary ~ .,Hitters)%*%array(ridge.coef)
mean((ridge.tot.pred -Hitters$Salary)^2)
```

```
## [1] 94585.52
```

3. Ordinary Least Squares Estimates The OLS estimates were almost the same as the coefficients of the Ridge Regression with lambda = 0. Tiny computation errors existed.

```
lm.coef=lm(Salary~., data=Hitters,subset=train)$coef
lm.coef
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 159.79768727  -2.43401688   8.45130111  -4.82357820  -2.73196262
##          RBI        Walks        Years       CAtBat        CHits
##   1.22824576   8.55663773 -34.92423092  -0.03635762   0.28071915
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.83357704   1.38639713  -0.02720391  -0.85514701 -17.98928857
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -84.90231011   0.43333581  -0.08036251   3.63392059  17.21556115
```

```
predict(x=x.train,y=y.train, ridge.mod ,s=0,exact=T,type="coefficients")[1:20,]
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs
## 159.79514839  -2.43386922   8.45066733  -4.82412057  -2.73163118
##          RBI        Walks        Years       CAtBat        CHits
##   1.22851272   8.55629326 -34.92228332  -0.03642196   0.28103915
##       CHmRun        CRuns         CRBI       CWalks      LeagueN
##   0.83405000   1.38622448  -0.02741165  -0.85505027 -17.98418511
##    DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -84.90214180   0.43333263  -0.08033589   3.63337438  17.21182010
```

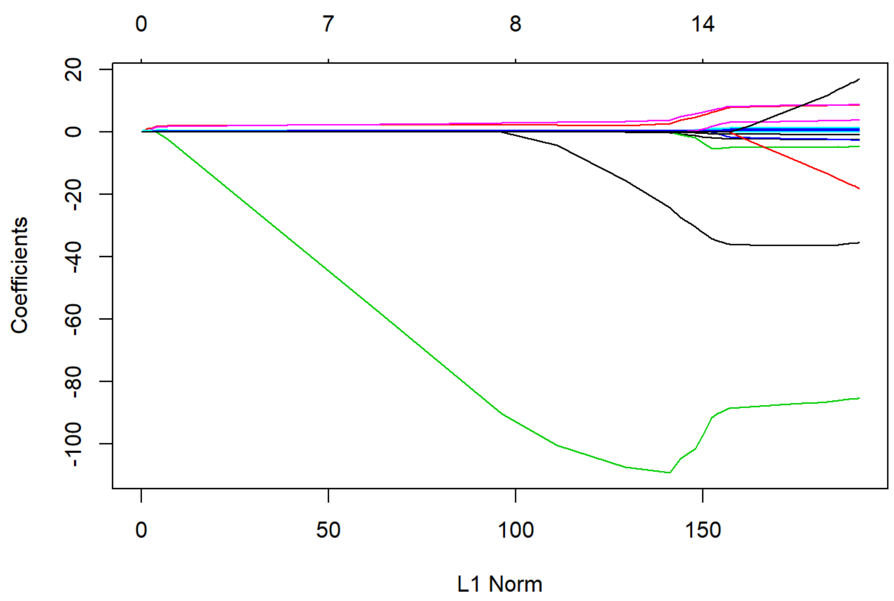
```
lm.tot.pred = model.matrix(Salary ~ .,Hitters)%*%array(lm.coef)
mean((lm.tot.pred -Hitters$Salary)^2)
```

```
## [1] 108026.4
```

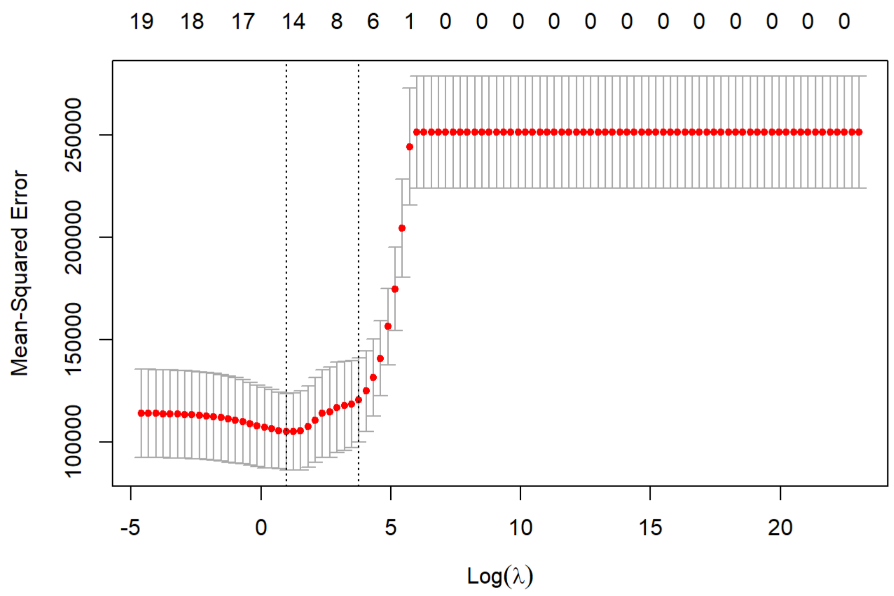
4. Lasso Regression

```
lasso.mod=glmnet(x.train,y.train,alpha=1,lambda=grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```




```
# cross-validation
set.seed(666)
cv.out=cv.glmnet(x.train,y.train,alpha=1,lambda=grid)
plot(cv.out)
```




```
la.bestlam =cv.out$lambda.min
la.bestlam
```

```
## [1] 2.656088
```

```
lasso.pred=predict (lasso.mod ,s=la.bestlam ,newx=x[test,])
mean((lasso.pred -y.test)^2)
```

```
## [1] 135844.8
```

```
la.tot=glmnet(x,Hitters$Salary,alpha=1,lambda=grid)
lasso.coef=predict (la.tot ,type="coefficients",s=la.bestlam)[1:20,]
lasso.coef
```

```
##   (Intercept)        AtBat         Hits        HmRun         Runs
##   124.0894873   -1.5600984    5.6931685    0.0000000    0.0000000
##           RBI        Walks        Years       CAtBat        CHits
##     0.0000000    4.7505395   -9.5180241    0.0000000    0.0000000
##        CHmRun        CRuns         CRBI       CWalks      LeagueN
##     0.5191611    0.6604074    0.3915415   -0.5326868   32.1125493
##     DivisionW      PutOuts      Assists       Errors   NewLeagueN
## -119.2583540    0.2726207    0.1748164   -2.0567432    0.0000000
```

```
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)        AtBat         Hits        Walks        Years
##   124.0894873   -1.5600984    5.6931685    4.7505395   -9.5180241
##        CHmRun        CRuns         CRBI       CWalks      LeagueN
##     0.5191611    0.6604074    0.3915415   -0.5326868   32.1125493
##     DivisionW      PutOuts      Assists       Errors
## -119.2583540    0.2726207    0.1748164   -2.0567432
```

```
la.tot.pred = model.matrix(Salary ~ .,Hitters)%*%array(lasso.coef)
mean((la.tot.pred -Hitters$Salary)^2)
```

```
## [1] 94525.63
```

3) 9 (a-d) from section 6.8

(a) Split the data set into a training set and a test set.

```
# Import data College
college <- read.csv("D:/luxinyve/00 Linear graph/HW3/College.csv",header=T,na.strings ="?")
#str(college)
sum(is.na(college))
```

```
## [1] 0
```

```
#rownames(college) = college[,1] #fix(college)
college = college[,-1]#fix(college)
# Split the data into test:train = 1:2
set.seed(123)
train=sample (1: nrow(college), nrow(college)/3*2)
test=-train
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

The mean squared error on the testing set was 779551.

```
b.lm = lm(Apps~.,data=college[train,])
summary(b.lm)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = college[train, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3098.1  -435.7   -32.6   326.9  6524.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -320.63000  483.82540  -0.663 0.507830
## PrivateYes  -631.06608  166.38884  -3.793 0.000167 ***
## Accept         1.22765    0.05907  20.782  < 2e-16 ***
## Enroll         0.07342    0.22242   0.330 0.741483
## Top10perc     45.28449    6.30692   7.180 2.54e-12 ***
## Top25perc    -12.88783    5.12008  -2.517 0.012144 *
## F.Undergrad    0.02496    0.04024   0.620 0.535386
## P.Undergrad    0.03394    0.03505   0.968 0.333304
## Outstate      -0.06350    0.02155  -2.947 0.003361 **
## Room.Board     0.20100    0.05392   3.728 0.000215 ***
## Books          0.16346    0.27890   0.586 0.558084
## Personal      -0.03987    0.07418  -0.537 0.591204
## PhD           -6.76818    5.36695  -1.261 0.207866
## Terminal      -5.29390    5.82889  -0.908 0.364201
## S.F.Ratio     -0.13458   14.77294  -0.009 0.992735
## perc.alumni   -7.16431    4.68079  -1.531 0.126506
## Expend         0.08032    0.01338   6.005 3.69e-09 ***
## Grad.Rate      9.82319    3.37117   2.914 0.003730 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 980.1 on 500 degrees of freedom
## Multiple R-squared:  0.918,  Adjusted R-squared:  0.9153
## F-statistic: 329.5 on 17 and 500 DF,  p-value: < 2.2e-16
```

```
y.predict = predict(b.lm,newdata = college[test,]) # newdata=test is equal to x.test
lm.test.err = mean((college[test,]$Apps-y.predict)^2)
lm.test.err
```

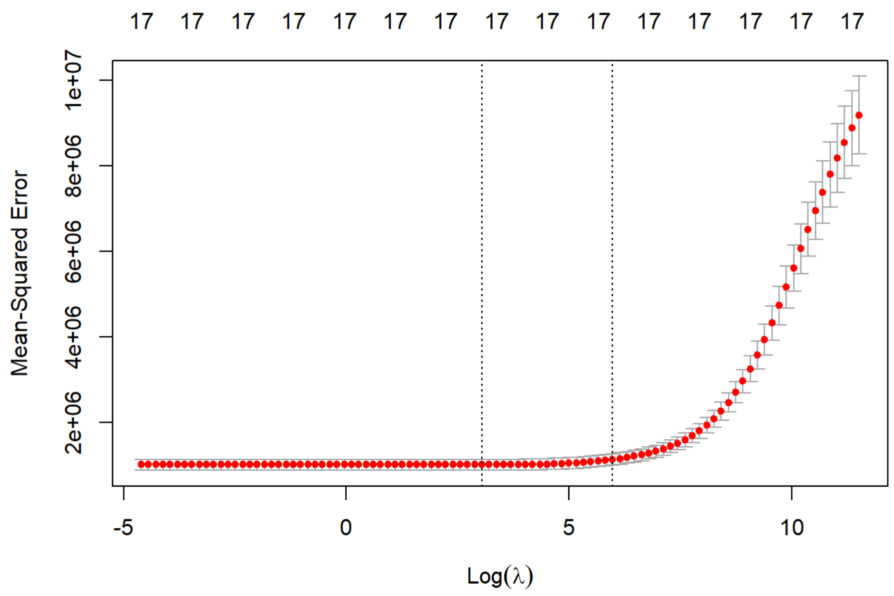
```
## [1] 1684049
```

c. Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

The best lambda for lasso chosen by cross validation was 0.01, which was rather small. The mean squared error on the testing set was 779536, which was a little bit smaller than the one for the ordinary linear model. Overall, for this data set, ridge regression's restriction was not strong so that it produced similar results as the ordinary linear regression. From the cross validation plot against lambda, we could see that regulization had little effect on reducing the testing error.

```
library(glmnet)
x.train=model.matrix(Apps~., data=college[train,])
x.test=model.matrix(Apps~., data=college[test,])
y.train=college[train,]$Apps
y.test=college[test,]$Apps
grid = 10 ^ seq(5, -2, length=100)

set.seed(123)
ridge.cv = cv.glmnet(x.train,y.train,alpha=0,lambda=grid,thresh=1e-12)
plot(ridge.cv)
```




```
lambda.best=ridge.cv$lambda.min
lambda.best
```

```
## [1] 21.04904
```

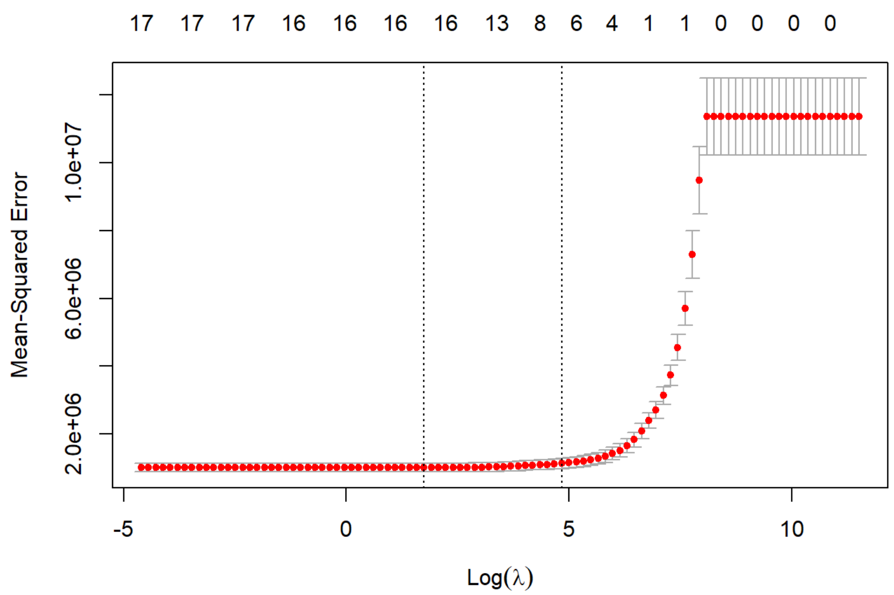
```
y.predict = predict(ridge.cv,newx = x.test,s=lambda.best) # newdata=test is equal to x.test
ridge.test.err = sum((y.test-y.predict)^2)/nrow(x.test)
ridge.test.err
```

```
## [1] 1800248
```

d. Fit a lasso model on the training set, with λ chosen by cross validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

The best lambda for lasso chosen by cross validation was 17.88. The mean squared error on the testing set was 748371, which was larger than the linear regression model and ridge model. There were 15 non-zero coefficient estimates. They were: PrivateYes, Accept, Enroll, Top10perc, Top25perc, P.Undergrad, Outstate, Room.Board, Personal, PhD, Terminal, S.F.Ratio, perc.alumni, Expend, Grad.Rate

```
# choose best lasso lambda with cross validation.
set.seed(123)
lasso.cv = cv.glmnet(x.train,y.train,alpha=1,lambda=grid,thresh=1e-12)
plot(lasso.cv)
```



```
lasso.lambda.best=lasso.cv$lambda.min
lasso.lambda.best
```

```
## [1] 5.722368
```

```
y.predict = predict(lasso.cv,newx = x.test,s=lasso.lambda.best) # newdata=test is equal to x.test
lasso.test.err = sum((y.test-y.predict)^2)/nrow(x.test)
lasso.test.err
```

```
## [1] 1685840
```

```
# fit a total lasso model with the best lasso lambda
x.tot=model.matrix(Apps~., data=college)
lasso.tot=glmnet(x.tot,college$Apps,alpha=1,lambda=lasso.lambda.best,thresh=1e-12)
lasso.coef=predict(lasso.tot,type='coefficients',s=lasso.lambda.best,thresh=1e-12)[,1]
lasso.coef[lasso.coef!=0]
```

```
##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -4.910457e+02 -4.869666e+02  1.552215e+00 -6.079926e-01  4.566857e+01
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
## -1.093780e+01  2.123086e-02  4.345524e-02 -7.941763e-02  1.451606e-01
##         Books      Personal           PhD      Terminal     S.F.Ratio
##  6.237793e-03  2.548149e-02 -7.962928e+00 -3.104548e+00  1.290359e+01
##   perc.alumni        Expend     Grad.Rate
## -2.368895e-01  7.561850e-02  7.661742e+00
```

4) for the prostate cancer data from the previous homework

```
# import data
library('lasso2')
```

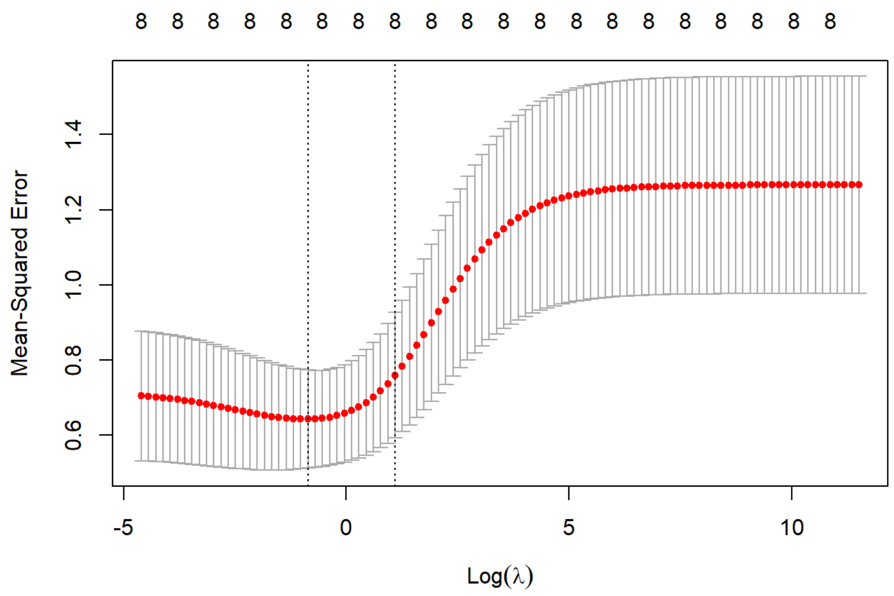
```
## R Package to solve regression problems while imposing
##   an L1 constraint on the parameters. Based on S-plus Release 2.1
## Copyright (C) 1998, 1999
## Justin Lokhorst   <jlokhors@stats.adelaide.edu.au>
## Berwin A. Turlach <bturlach@stats.adelaide.edu.au>
## Bill Venables     <wvenable@stats.adelaide.edu.au>
##
## Copyright (C) 2002
## Martin Maechler <maechler@stat.math.ethz.ch>
```

```
data(Prostate)
```

(a) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

The best lambda for lasso chosen by cross validation was 0.689. The mean squared error on the testing set was 0.6243547. From the cross validation plot against lambda, we could see that regulization could produce smaller mean squared test error than ordinary linear regression with appropriate lambda.

```
# split train and test
set.seed(999)
train=sample (1: nrow(Prostate), nrow(Prostate)/2)
test=(-train)
P.x.train=model.matrix(lpsa~., data=Prostate[train,])
P.x.test=model.matrix(lpsa~., data=Prostate[test,])
P.y.train=Prostate[train,]$lpsa
P.y.test=Prostate[test,]$lpsa
# cross-validation
set.seed(999)
P.ridge.cv = cv.glmnet(P.x.train,P.y.train,alpha=0,lambda=grid,thresh=1e-12)
plot(P.ridge.cv)
```




```
P.lambda.best=P.ridge.cv$lambda.min
P.lambda.best
```

```
## [1] 0.4229243
```

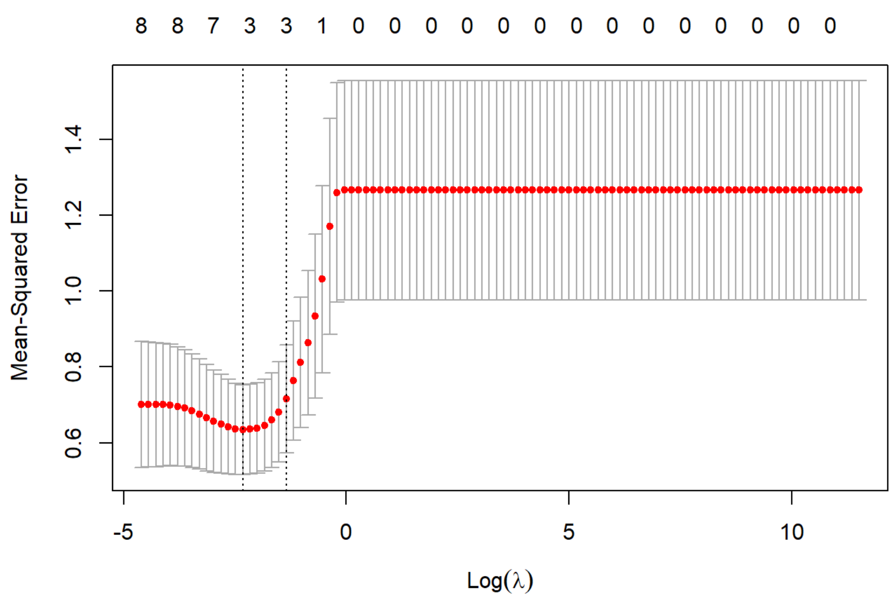
```
P.y.predict = predict(P.ridge.cv,newx = P.x.test,s=P.lambda.best) # newdata=test is equal to x.test
P.ridge.test.err = mean((P.y.test-P.y.predict)^2)
P.ridge.test.err
```

```
## [1] 0.5724469
```

(b) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

The best lambda for lasso chosen by cross validation was 0.135. The mean squared error on the testing set was 0.53473, which was larger than the linear regression model and ridge model. There were 10 non-zero coefficient estimates. They were:lcavol, lweight, lbph, svi, pgg45.

```
set.seed(999)
P.lasso.cv = cv.glmnet(P.x.train,P.y.train,alpha=1,lambda=grid,thresh=1e-12)
plot(P.lasso.cv)
```




```
P.lasso.lambda.best=P.lasso.cv$lambda.min
P.lasso.lambda.best
```

```
## [1] 0.097701
```

```
P.y.predict = predict(P.lasso.cv,newx = P.x.test,s=P.lasso.lambda.best)
P.lasso.test.err = mean((P.y.test-P.y.predict)^2)
P.lasso.test.err
```

```
## [1] 0.5564063
```

```
# fit a total lasso model with the best lasso lambda
x.tot=model.matrix(lpsa~., data=Prostate)
P.lasso.tot=glmnet(x.tot,Prostate$lpsa,alpha=1,lambda=P.lasso.lambda.best,thresh=1e-12)
P.lasso.coef=predict(P.lasso.tot,type='coefficients',s=P.lasso.lambda.best,thresh=1e-12)[,1]
P.lasso.coef[P.lasso.coef!=0]
```

```
##  (Intercept)       lcavol      lweight         lbph          svi
## 0.5444509114 0.5047028358 0.3062920464 0.0297849260 0.5102961676
##        pgg45
## 0.0008337464
```