homework6

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

```
library(ISLR)

set.seed(23456)
College = na.omit(College)
index = sample(nrow(College), size = trunc(0.70 * nrow(College)))
College_train = College[index,]
College_test = College[-index,]
```

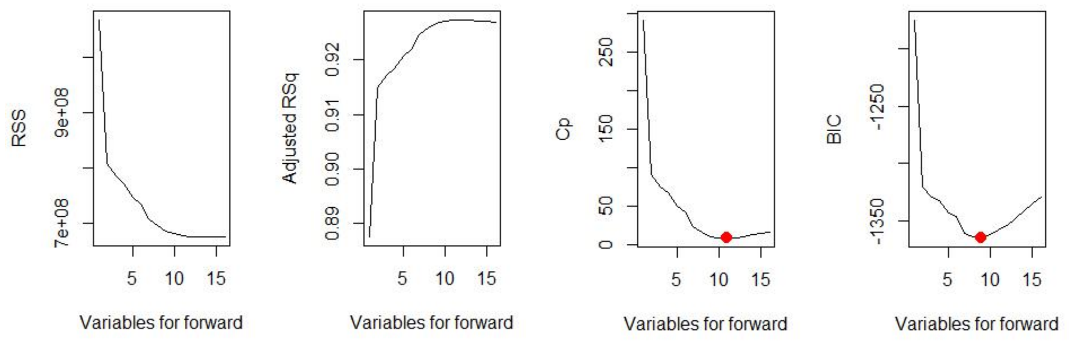
(b) Fit a linear model using least squares on the training set

```
lm.mod = lm(Apps~.,data = College_train)
summary(lm.mod)
## Call:
## lm(formula = Apps ~ ., data = College_train)
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.664e+02  5.399e+02  -0.679 0.497675
## PrivateYes  -4.579e+02  1.796e+02  -2.550 0.011043 *
## Accept       1.637e+00  4.970e-02  32.946  < 2e-16 ***
## Enroll      -1.280e+00  2.459e-01  -5.207 2.76e-07 ***
## Top10perc    6.196e+01  7.163e+00   8.649  < 2e-16 ***
## Top25perc   -2.105e+01  5.866e+00  -3.588 0.000364 ***
## F.Undergrad  1.172e-01  4.335e-02   2.704 0.007080 **
## P.Undergrad  6.332e-02  4.217e-02   1.502 0.133815
## Outstate    -1.082e-01  2.451e-02  -4.414 1.23e-05 ***
## Room.Board   1.943e-01  6.178e-02   3.145 0.001753 **
## Books       -1.878e-01  3.290e-01  -0.571 0.568235
## Personal     8.372e-03  8.190e-02   0.102 0.918620
## PhD         -9.278e+00  6.126e+00  -1.515 0.130484
## Terminal    -3.241e+00  6.539e+00  -0.496 0.620324
## S.F.Ratio    1.625e+01  1.848e+01   0.879 0.379640
## perc.alumni  2.622e+00  5.375e+00   0.488 0.625806
## Expend       7.751e-02  1.655e-02   4.683 3.60e-06 ***
## Grad.Rate    1.030e+01  3.826e+00   2.693 0.007311 **
##
## Residual standard error: 1127 on 525 degrees of freedom
## Multiple R-squared:  0.9298, Adjusted R-squared:  0.9276
## F-statistic: 409.3 on 17 and 525 DF,  p-value: < 2.2e-16
lm.pred_train = predict(lm.mod,College_train)
mean((College_train$Apps-lm.pred_train)^2)

## [1] 1228844
lm.pred_test = predict(lm.mod,College_test)
mean((College_test$Apps-lm.pred_test)^2)

## [1] 738611.2
```

Comment: According to the calculation, the training error is 1228844 and the test error is 738611.2.

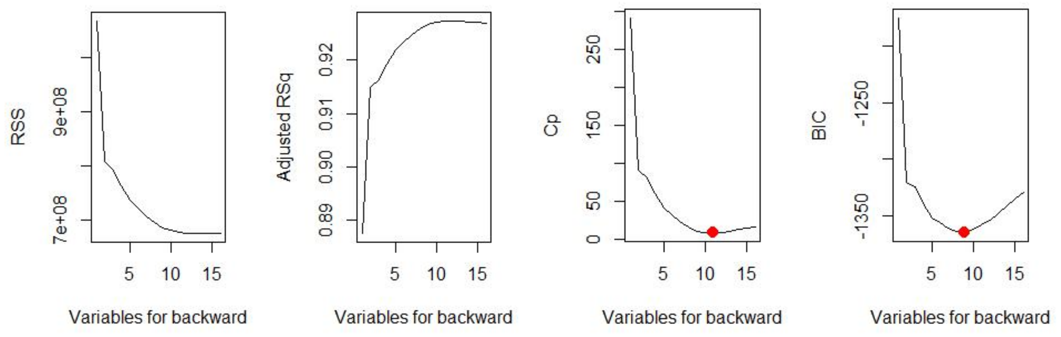
(c) Perform forward and backward selection on the previous model

```
library(leaps)
regfit.fwd=regsubsets(Apps~.,data=College_train[,-1],nvmax=16,method="for
ward")
regfit.fwd.summary = summary(regfit.fwd)
regfit.bwd=regsubsets(Apps~.,data=College_train[,-1],nvmax=16,method="bac
kward")
regfit.bwd.summary = summary(regfit.bwd)
par(mfrow=c(1,2))
plot(regfit.fwd.summary$rss,xlab="Variables for forward",ylab="RSS",type=
"l")
plot(regfit.fwd.summary$adjr2,xlab="Variables for forward",ylab="Adjusted
 RSq",type="l")

which.max(regfit.fwd.summary$adjr2)

## [1] 12

par(mfrow=c(1,2))
plot(regfit.fwd.summary$cp,xlab="Variables for forward",ylab="Cp",type='l
')
which.min(regfit.fwd.summary$cp)

## [1] 11

points(11,regfit.fwd.summary$cp[11],col="red",cex=2,pch=20)
which.min(regfit.fwd.summary$bic)

## [1] 9

plot(regfit.fwd.summary$bic,xlab="Variables for forward",ylab="BIC",type=
'l')
points(9,regfit.fwd.summary$bic[9],col="red",cex=2,pch=20)
```



```
par(mfrow=c(1,2))
plot(regfit.bwd.summary$rss,xlab="Variables for backward",ylab="RSS",type
="l")
```

```
plot(regfit.bwd.summary$adjr2,xlab="Variables for backward",ylab="Adjuste
d RSq",type="l")

which.max(regfit.bwd.summary$adjr2)

## [1] 12

par(mfrow=c(1,2))
plot(regfit.bwd.summary$cp,xlab="Variables for backward",ylab="Cp",type='
l')
which.min(regfit.bwd.summary$cp)

## [1] 11

points(11,regfit.bwd.summary$cp[11],col="red",cex=2,pch=20)
which.min(regfit.bwd.summary$bic)

## [1] 9

plot(regfit.bwd.summary$bic,xlab="Variables for backward",ylab="BIC",type
='l')
points(9,regfit.bwd.summary$bic[9],col="red",cex=2,pch=20)
```



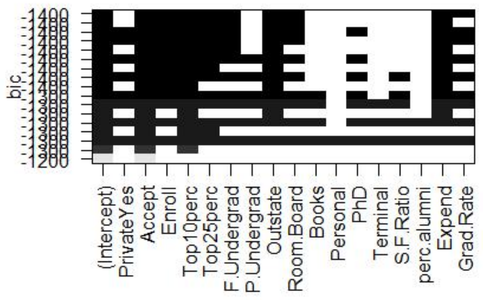
```
library(SignifReg)
forward = SignifReg(Apps~., College_train, alpha = 0.05, direction = "for
ward",criterion = "p-value", correction = "None")
backward = SignifReg(Apps~., College_train, alpha = 0.05, direction = "ba
ckward",criterion = "p-value", correction = "None")
summary(forward)
## Call:
## lm(formula = reg, data = data)
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -168.46633  344.43207  -0.489  0.62496
## Accept         1.62541    0.04837  33.603  < 2e-16 ***
## Top10perc     61.19885    7.06873   8.658  < 2e-16 ***
## Top25perc    -21.53481    5.77008  -3.732  0.00021 ***
## Enroll        -1.26307    0.24408  -5.175 3.24e-07 ***
## F.Undergrad    0.12958    0.04178   3.101  0.00203 **
## Outstate      -0.10889    0.02336  -4.660 3.99e-06 ***
## Expend         0.07325    0.01519   4.823 1.85e-06 ***
## Room.Board     0.18898    0.05979   3.161  0.00166 **
## Grad.Rate     10.37948    3.63969   2.852  0.00452 **
## PrivateYes  -479.65395  174.96506  -2.741  0.00632 **
## PhD          -10.82813    4.12725  -2.624  0.00895 **
## Residual standard error: 1125 on 531 degrees of freedom
## Multiple R-squared:  0.9293, Adjusted R-squared:  0.9279
## F-statistic: 634.9 on 11 and 531 DF,  p-value: < 2.2e-16

summary(backward)
## Call:
## lm(formula = reg, data = data)
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -168.46633  344.43207  -0.489  0.62496
## PrivateYes  -479.65395  174.96506  -2.741  0.00632 **
## Accept         1.62541    0.04837  33.603  < 2e-16 ***
## Enroll        -1.26307    0.24408  -5.175 3.24e-07 ***
## Top10perc     61.19885    7.06873   8.658  < 2e-16 ***
## Top25perc    -21.53481    5.77008  -3.732  0.00021 ***
## F.Undergrad    0.12958    0.04178   3.101  0.00203 **
## Outstate      -0.10889    0.02336  -4.660 3.99e-06 ***
## Room.Board     0.18898    0.05979   3.161  0.00166 **
## PhD          -10.82813    4.12725  -2.624  0.00895 **
## Expend         0.07325    0.01519   4.823 1.85e-06 ***
## Grad.Rate     10.37948    3.63969   2.852  0.00452 **
## Residual standard error: 1125 on 531 degrees of freedom
## Multiple R-squared:  0.9293, Adjusted R-squared:  0.9279
## F-statistic: 634.9 on 11 and 531 DF,  p-value: < 2.2e-16

coef.forward = coef(forward)
coef.backward = coef(backward)
# calculate the train error and test error
test.mat=model.matrix(Apps~.,data=College_test)
train.mat=model.matrix(Apps~.,data=College_train)
# forward checking
pred_train.forward=train.mat[,names(coef.forward)]%*%coef.forward
pred_test.forward=test.mat[,names(coef.forward)]%*%coef.forward
val.errors_train=mean((College_train$Apps-pred_train.forward)^2)
val.errors_test=mean((College_test$Apps-pred_test.forward)^2)
val.errors_train

## [1] 1237635

val.errors_test

## [1] 724578

# backward checking
pred_train.backward=train.mat[,names(coef.backward)]%*%coef.backward
pred_test.backward=test.mat[,names(coef.backward)]%*%coef.backward
val.errors_train=mean((College_train$Apps-pred_train.backward)^2)
val.errors_test=mean((College_test$Apps-pred_test.backward)^2)
val.errors_train
```

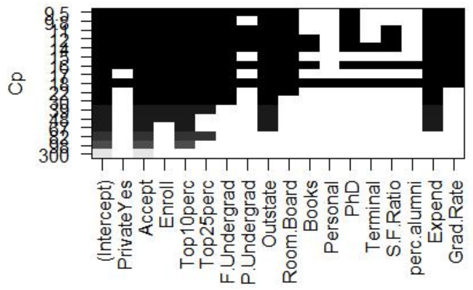
```
## [1] 1237635

val.errors_test

## [1] 724578
```

Comment: According to the result, the suitable number of variables in the final model would be among 9 and 11. Forward and backward selection both indicate that the final model should contain 11 variables respectively. Corresponding variables and their coefficients have been shown above. Corresponding train error and test error are 1237635 and 724578.

(d) Use AIC and BIC to select a potentially smaller model

```
regfit.AICBIC=regsubsets(Apps~.,data=College_train,nvmax=17)
regfit.AICBIC.summary = summary(regfit.AICBIC)
which.min(regfit.AICBIC.summary$cp)

## [1] 12

which.min(regfit.AICBIC.summary$bic)

## [1] 9

plot(regfit.AICBIC,scale="Cp")

plot(regfit.AICBIC,scale="bic")
```





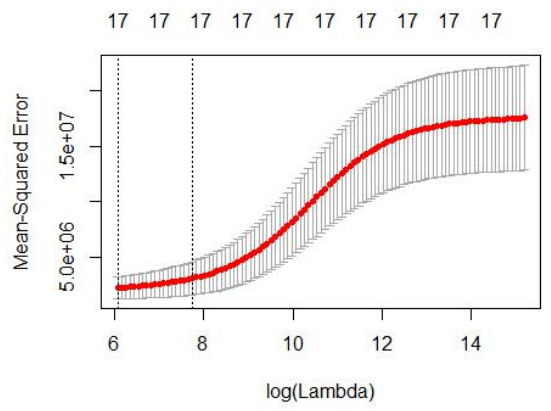
```
# AIC choice
coef.AIC = coef(regfit.AICBIC,12)
pred_train=train.mat[,names(coef.AIC)]%*%coef.AIC
pred_test=test.mat[,names(coef.AIC)]%*%coef.AIC
val.errors_train=mean((College_train$Apps-pred_train)^2)
val.errors_test=mean((College_test$Apps-pred_test)^2)
val.errors_train

## [1] 1232282

val.errors_test

## [1] 732162.9
```

```r
# BIC choice
coef.BIC = coef(regfit.AICBIC,9)
pred_train=train.mat[,names(coef.BIC)]%*%coef.BIC
pred_test=test.mat[,names(coef.BIC)]%*%coef.BIC
val.errors_train=mean((College_train$Apps-pred_train)^2)
val.errors_test=mean((College_test$Apps-pred_test)^2)
val.errors_train

## [1] 1263519

val.errors_test

## [1] 760046.7
```

Comment: According to the result, there will be 12 variables in the final model through AIC and 9 variables in the final model through BIC. For AIC, the variables contain PrivateYes, Accept, Top10perc, Top25perc, Enroll, F.undergrad, Outstate, Room.Board, Grad.Rate, P.Undergrad, PhD and Expend. For BIC, the variables contain Accept, Top10perc, Top25perc, Enroll, F.undergrad, Outstate, Room.Board, Expend and Grad.Rate. For AIC, the train error and test error are 1232282 and 732162.9. For BIC, the train error and test error are 1263519 and 760046.7.

(e) Fit a ridge regression model on the training set

```r
library(glmnet)

X_train = model.matrix(Apps~., College_train)[, -1]
y_train = College_train$Apps
X_test = model.matrix(Apps~., College_test)[, -1]
y_test = College_test$Apps
grid=10^seq(10,-2,length=100)
ridge.mod=glmnet(X_train,y_train,alpha=0,lambda=grid)

set.seed(23456)
cv.out=cv.glmnet(X_train,y_train,alpha=0)
plot(cv.out)
```



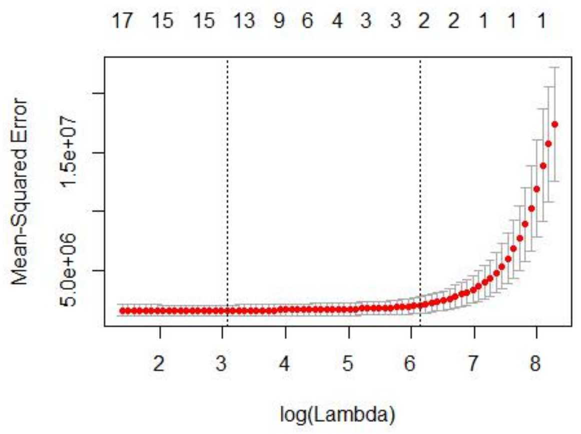
```r
bestlam=cv.out$lambda.min
bestlam
```

```
## [1] 432.7762

predict(ridge.mod,s=433,type="coefficients")[1:17,]

##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -1.697484e+03 -4.544814e+02  9.979736e-01  4.450977e-01  2.864644e+01
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
##  1.152299e-01  8.965724e-02  1.491390e-02 -3.591888e-02  2.355121e-01
##         Books      Personal           PhD      Terminal     S.F.Ratio
##  1.245528e-01 -4.226640e-02 -3.504955e+00 -5.064701e+00  1.649866e+01
##   perc.alumni        Expend
## -8.023658e+00  7.786768e-02
ridge.pred_train=predict(ridge.mod,s=bestlam,newx=X_train)
mean((ridge.pred_train-y_train)^2)

## [1] 1641351
ridge.pred_test=predict(ridge.mod,s=bestlam,newx=X_test)
mean((ridge.pred_test-y_test)^2)

## [1] 720118.4
```

Comment: According to the result, corresponding variables' coefficients have been shown above. Lambda is 433 and train error and test error are 1641351 and 720118.4.

(f) Fit a lasso model on the training set, with lambda chosen by cross-validation. Report which variables are included in the model, and the training and test errors obtained.

```
lasso.mod=glmnet(X_train,y_train,alpha=1,lambda=grid)

set.seed(23456)
cv.out=cv.glmnet(X_train,y_train,alpha=1)
plot(cv.out)
```



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

## [1] 21.53937
```

```
predict(lasso.mod,s=bestlam,type="coefficients")[1:18,]

##   (Intercept)    PrivateYes        Accept        Enroll     Top10perc
## -656.11856013 -408.27537845    1.50117188   -0.30367854   44.29500408
##     Top25perc   F.Undergrad   P.Undergrad      Outstate    Room.Board
##   -7.40068856    0.00000000    0.03793485   -0.07966233    0.16434804
##         Books      Personal           PhD      Terminal     S.F.Ratio
##    0.00000000    0.00000000   -5.83721313   -2.91491665    4.61594741
##   perc.alumni        Expend     Grad.Rate
##    0.00000000    0.06705392    6.26811084

# training MSE
lasso.pred_train=predict(lasso.mod,s=bestlam,newx=X_train)
mean((lasso.pred_train-y_train)^2)

## [1] 1286184

# test MSE
lasso.pred_test=predict(lasso.mod,s=bestlam,newx=X_test)
mean((lasso.pred_test-y_test)^2)

## [1] 620801.4
```

Comment: According to the result, corresponding variables' coefficients have been shown above. Lambda is 21.53937 and train error and test error are 1286184 and 620801.4.

(g) Comment on the results obtained.

Comment: According to the result shown above, The training error and test error for linear regression are 1228844 and 738611.2.Training error and test error for forward and backward selection are 1237635 and 724578. Training error and test error for AIC are 1232282 and 732162.9.Training error and test error for BIC are 1263519 and 760046.7. Training error and test error for ridge regression are 1641351 and 720118.4.Training error and test error for Lasso regression are 1286184 and 620801.4. Since the test error for Lasso regression is smaller, Lasso regression method is more suitable for this data set here.