Jeffrey Kerley
Jakcqc
3/24/2022

Homework 6

Problem 1

(a) According to the code generating the data, which predictor variables, among x1,⋯,x10, are desired to be found to be associated with the response variable by model selection methods?
X1, x2, and x3 are found to be associated with the response variable.

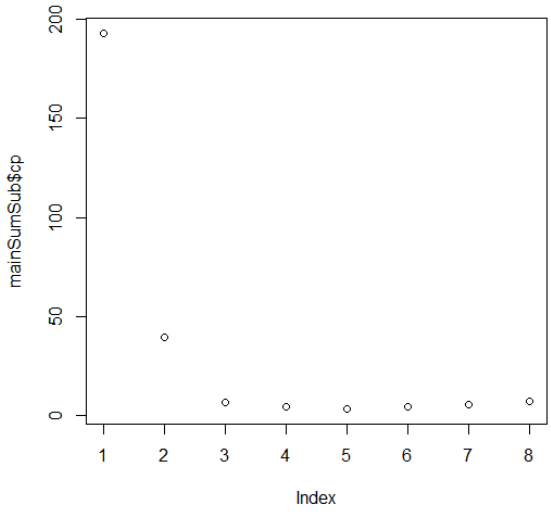
(b) Use the function regsubsets() in the library leaps to perform the best subset selection (using data_df). Show the summary of the outcome. Which variables are included in the best 1-predictor, 3-predictor, and 5-predictor models?

```
> sFit = regsubsets(y ~ ., data = data_df)
> summary(sFit)
Subset selection object
Call: regsubsets.formula(y ~ ., data = data_df)
10 Variables  (and intercept)
   Forced in Forced out
x1      FALSE     FALSE
x2      FALSE     FALSE
x3      FALSE     FALSE
x4      FALSE     FALSE
x5      FALSE     FALSE
x6      FALSE     FALSE
x7      FALSE     FALSE
x8      FALSE     FALSE
x9      FALSE     FALSE
x10     FALSE     FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         x1  x2  x3  x4  x5  x6  x7  x8  x9  x10
1  ( 1 ) " " " " "*" " " " " " " " " " " " " "
2  ( 1 ) " " "*" "*" " " " " " " " " " " " " "
3  ( 1 ) "*" "*" "*" " " " " " " " " " " " " "
4  ( 1 ) "*" " " "*" "*" " " " " " " " " " " "
5  ( 1 ) "*" " " " " "*" "*" "*" " " " " "*" " "
6  ( 1 ) "*" " " " " " " " " "*" "*" "*" "*" " " "*"
7  ( 1 ) "*" "*" " " " " "*" "*" "*" "*" " " " " " "*"
8  ( 1 ) "*" "*" "*" "*" " " " " "*" "*" "*" "*" " " "
```

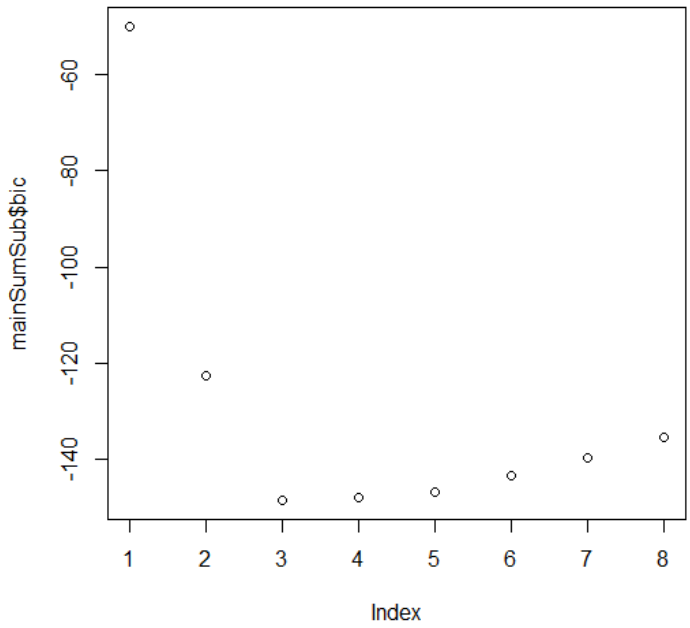
For 1 predictor no variables are included, for 3, x1,x2,x3 are included, and for 5, we see x1, x4, x5, x6, x9 are included.

(c) What is the best model obtained according to Cp, BIC, and adjusted R2? Show plots for each criterion to provide evidence for your answer.

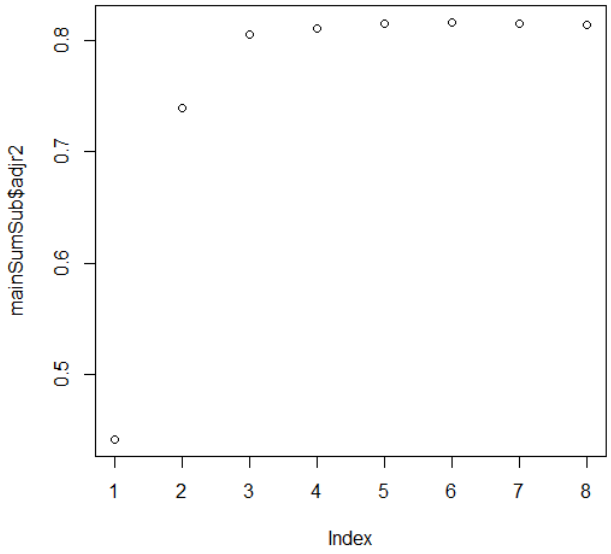
The best model according to Cp is degree 4, but 3 is almost equal to 4, so we say 3 in practice.



BIC says 3 degrees



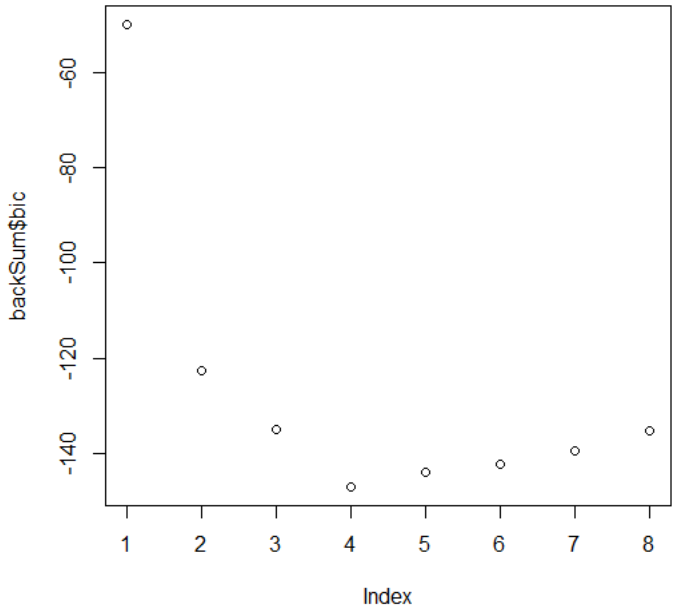
R2 says 6 degrees, but degree 3 to 6 has few differences.



(d) Using 'coef()', report the coefficients of the best models obtained by Cp, BIC, and adjusted R2, respectively.

```
> coef(sFit, 4)
(Intercept)          x1          x3          x4          x6
 -1.0496665   2.0386545  -1.9544633   1.3341706  -0.3031902
> coef(sFit, 3)
(Intercept)          x1          x2          x3
  -1.312308    1.988786    1.487226   -1.931813
> coef(sFit, 6)
(Intercept)          x1          x5          x6          x7          x8         x10
 -1.0466359   1.5371140  -1.7340376   2.3224985   0.4136039  -1.5442441   0.2800249
```

(e) Find the best model using forward stepwise selection. At this time, use only BIC to determine the best model. Which variables are included in the model?



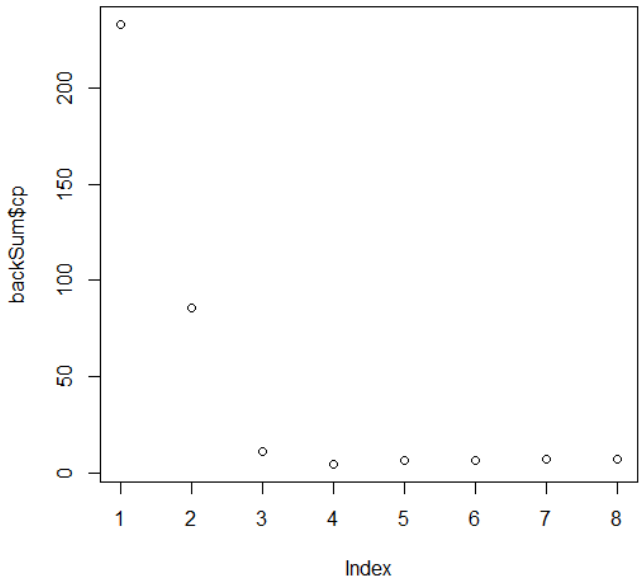
This model includes degree 4, the variables are:

> coef(sFitFor, 4)

(Intercept) x1 x2 x5 x9

-1.33969146 1.36699727 1.49984649 -1.20853482 0.08518351

(e) Find the best model using backward stepwise selection. At this time, use only Cpto to determine the best model. Which variables are included in the model?



> coef(sFitBack, 4)

(Intercept) x1 x3 x4 x6

-1.0496665 2.0386545 -1.9544633 1.3341706 -0.3031902

(f) Compare the best model you obtained from best subset selection, forward stepwise and backward stepwise methods with the true underlying model. Briefly describe the advantages and disadvantages of those methods based on what you observed from the outcome.

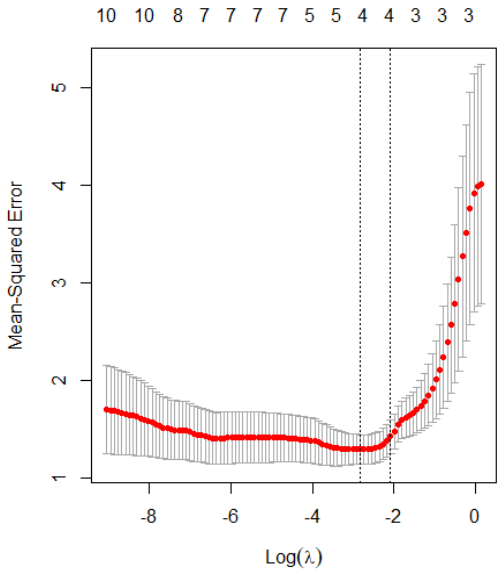
The advantages of using forward and backward stepwise does not reside in their ability to provide the most accurate model fit. This is clear from how the subset selection method yielded the most accurate fit to our true model. Forward and backward do require much less computational time, which is where their strength lies. Overall, we can rate their effectiveness as forward < backward < subset selection in terms of accurate model fit.

Problem 2

(a) Split the data into training (70%) and testing (30%) sets using the seed number 1. You can simply prepare train and test vectors that include row indices for each set.

```
> set.seed(1)
> ttData = sort(sample(nrow(data_mat), nrow(data_mat)*.7))
> trainD = data_mat[ttData,]
> testD = data_mat[-ttData,]
```

(b) Create lasso.mod by fitting lasso regression using the training set. Then use 10-fold CV to find the best λ with the seed number 2. Plot the outcome of cross-validation. Which value of λ is the best?



The best Lambda is:
bestL
[1] 0.0595452

(c) Make predictions for the test set using the fitted model lasso.mod with the best λ. Compute the test MSE.

lasso.pred = predict(lasso.mod, s = bestL, newx = x[testD,])
> mean((lasso.pred-y.test)^2)
[1] 1.011547

(d) Refit the model on the full data set with the best λ. Extract the regression coefficients estimates. Compare the outcome with the true model.

> fullFit = glmnet(x,y,alpha=1, lambda=grid)
> lCoef = predict(fullFit, type = "coefficients", s=bestL)[1:20]
> lCoef[lCoef!=0]
[1] -1.20476328 0.49388048 1.09932056 -0.02011887 0.14719977
[6] -0.51405244 -1.20476328 0.49388048 1.09932056 -0.02011887
[11] 0.14719977 -0.51405244

We can see that the Lasso method overestimates the number of degrees/coefficients that are needed in the final fit.

Problem 3
(a) Perform model training for PCR using the training set. Use cross-validation to determine the number of principal components to be used, with the seed number 3. Show the summary of fit and validation plot. How many PC is the best?

> pFit = pcr(y~., data = data_df, subset=trainD, scale = TRUE, validation="CV")
> summary(pFit)
Data: X dimension: 70 10
Y dimension: 70 1
Fit method: svdpc
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 10 random segments.
(Intercept) 1 comps 2 comps 3 comps 4 comps

CV 2.021 2.045 1.382 1.210 1.530
adjCV 2.021 2.087 1.378 1.198 1.503
5 comps 6 comps 7 comps 8 comps 9 comps
CV 1.113 1.119 1.121 1.287 1.307
adjCV 1.100 1.104 1.105 1.257 1.275
10 comps
CV 1.955
adjCV 1.879

TRAINING: % variance explained
1 comps 2 comps 3 comps 4 comps 5 comps 6 comps
X 55.3091 91.48 96.90 99.48 99.85 99.99
y 0.3512 55.41 70.22 70.28 78.66 79.40
7 comps 8 comps 9 comps 10 comps
X 100.00 100.00 100.00 100.00
y 79.59 79.96 80.24 80.41

From the above, we see that anywhere from 5 to 6 components are good, with 6 being the best fit.

(b) Test the model with the test set using the best number of PC. Compute test MSE.

```
> pPred = predict(pFit, x[testD,], ncomp=6)
> mean((pPred-y.test)^2)
[1] 0.8816928
```

(c) Refit the PCR on the full data set with the best number of PC. Show the summary of the fit. How much of the variability of predictors is explained by the PCs? How much of the variability of the response variable is explained by the PC regression?

```
> pFitFull =pcr(y~., data=data_df, scale=TRUE,ncomp=6)
> summary(pFitFull)
Data:  X dimension: 100 10
       Y dimension: 100 1
Fit method: svdpc
Number of components considered: 6
TRAINING: % variance explained
  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X  47.15    91.25    96.29    99.46    99.83    99.98
```

y 54.76 65.06 72.19 74.28 81.88 82.22
> plot(pFitFull)

We can see that at 6 components almost 100% of the variability of the predictors is explained. Yet, at only 4 components we are already at 99.46 percent explanation. In our response variable, we only see an 82.22 percent explanation.

CODE:

```
library(leaps)
library(ISLR)
library(glmnet)
library(pls)
set.seed(1)
x1 <- runif(100, -1.7, 1.7)
x2 <- x1^2; x3 <- x1^3; x4 <- x1^4; x5 <- x1^5
x6 <- x1^6; x7 <- x1^7; x8 <- x1^8; x9 <- x1^9
x10 <- x1^10
y <- -1.3 + 2*x1 + 1.5*x2 - 2*x3 + rnorm(100)
data_df <- data.frame(y, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10)
data_mat <- as.matrix(data_df)
# We prepare the data set in two different objects: data_df (data frame),
data_mat (matrix)
plot(x1,y)

?regsubsets
sFit = regsubsets(y ~ ., data = data_df)
mainSumSub = summary(sFit)

plot(mainSumSub$cp)
plot(mainSumSub$bic)
plot(mainSumSub$adjr2)

coef(sFit, 4)
coef(sFit, 3)
coef(sFit, 6)

sFitBack = regsubsets(y ~ ., data = data_df, method = "backward")
```

```r
backSum = summary(sFitBack)
plot(backSum$cp)
coef(sFitBack, 4)

sFitFor = regsubsets(y ~ ., data = data_df, method = "forward")
forSum = summary(sFitFor)
plot(forSum$bic)
coef(sFitFor, 4)


#problem 2
grid=10^seq(10,-2,length=100)

x = model.matrix(y~.,data_df)[,-1]
newY = data_df[1]

set.seed(1)
trainD = sample(1:nrow(x), nrow(x)*.7)
testD = (-trainD)
y.test = y[testD]
lasso.mod = glmnet(x[trainD,], y[trainD], alpha=1)
plot(lasso.mod)

set.seed(2)

crossV = cv.glmnet(x[trainD,], y[trainD], alpha=1)
plot(crossV)
bestL = crossV$lambda.min
bestL

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

fullFit = glmnet(x,y,alpha=1, lambda=grid)
lCoef = predict(fullFit, type = "coefficients", s=bestL)[1:20]
lCoef[lCoef!=0]


set.seed(3)
```

```r
pFit = pcr(y~., data = data_df, subset=trainD, scale = TRUE, validation="CV")
summary(pFit)
validationplot(pFit,val.type="MSEP")

pPred = predict(pFit, x[testD,], ncomp=6)
mean((pPred-y.test)^2)

pFitFull =pcr(y~., data=data_df, scale=TRUE,ncomp=6)
summary(pFitFull)
```