Homework #6

Lindsey Dodd Gooch

December 3, 2019

Chapter 6

8 In this exercise, we will generate simulated data, and will then use this data to perform best subset selection. (a) Use the rnorm() function to generate a predictor X of length n = 100, as well as a noise vector ϵ of length n = 100.

```
set.seed(1214)
X <- rnorm(100)
e <- rnorm(100)
```

(b) Generate a response vector Y of length n = 100 according to the model $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$ where βs are coeffients of your choice.

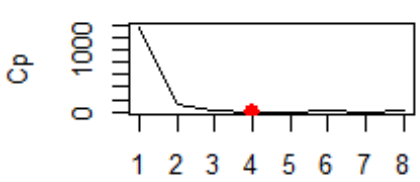
```
Y <- 10 + 2*X - 3*X^2 - 2*X^3 + e
```

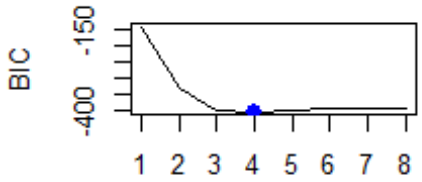
(c) Use the regsubsets() function to perform best subset selection in order to choose the best model containing the predictors $X, X^2, \ldots, X^{10}$. What is the best model obtained according to Cp, BIC, and adjusted R^2? Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained. Note you will need to use the data.frame() function to create a single data set containing both X and Y.
As shown in the plots and output below, the best model obtained according to Cp and BIC has 4 variables and for adjusted R^2 7 regressors. In all three plots it is evident that although the absolute minimum/maximum is achieved at a value greater than 3, the values do seem to taper off after 3 (the number of variables in the true model). The best model obtained is:

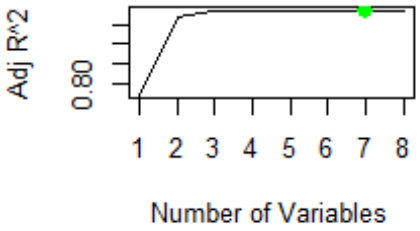
$$\hat{Y} = 9.8269 + 1.5519X - 2.8996X^2 - 1.648X^3 - 0.0640X^5$$

```
library(leaps)

## Warning: package 'leaps' was built under R version 3.5.3

df<-data.frame(Y, X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10)
regfit.full<-regsubsets(Y~., data=df)
reg.summary<-summary(regfit.full)

which.min(reg.summary$cp)

## [1] 4

which.min(reg.summary$bic)

## [1] 4

which.max(reg.summary$adjr2)
```

```
## [1] 7

par(mfrow=c(2,2))
plot(reg.summary$cp, xlab="Number of Variables", ylab="Cp", type='l')
points(4, reg.summary$cp[4], col="red", cex=2, pch=20)
plot(reg.summary$bic, xlab="Number of Variables", ylab="BIC", type='l')
points(4, reg.summary$bic[4], col="blue", cex=2, pch=20)
plot(reg.summary$adjr2, xlab="Number of Variables", ylab="Adj R^2", type='l')
points(7, reg.summary$adjr2[7], col="green", cex=2, pch=20)

coef(regfit.full, which.min(reg.summary$cp))

## (Intercept)           X         X.2         X.3         X.5
##  9.82692499  1.55190305 -2.89962469 -1.64810509 -0.06396451
```





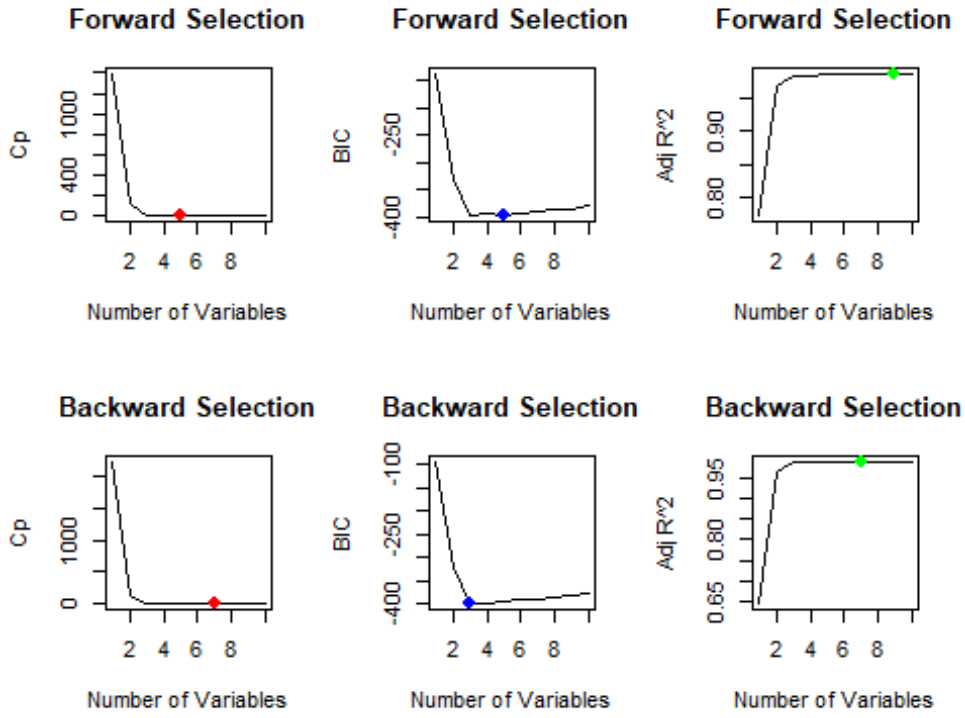


(d) Repeat (c), using forward stepwise selection and also using backwards stepwise selection. How does your answer compare to the results in (c)? **The best model by BIC for backward selection is the only one that has selected the correct number of variables with regard to the true model. Similar to the best subset selection in (c), the plots show that while the measure of fit may still continue to increase/decrease after 3, they do appear to flatten out after 3. The best model chosen from stepwise selection (backwards, by BIC) is:**

$$\hat{Y} = 9.9043 + 2.0895X - 2.9781X^2 - 2.0760X^3$$

```
regfit.fwd<-regsubsets(Y~.,data=df, method="forward", nvmax=10)
fwd.sum<-summary(regfit.fwd)
regfit.bwd<-regsubsets(Y~.,data=df, method="backward", nvmax=10)
```

```r
bwd.sum<-summary(regfit.bwd)

which.min(fwd.sum$cp)

## [1] 5

which.min(fwd.sum$bic)

## [1] 5

which.max(fwd.sum$adjr2)

## [1] 9

which.min(bwd.sum$cp)

## [1] 7

which.min(bwd.sum$bic)

## [1] 3

which.max(bwd.sum$adjr2)

## [1] 7

par(mfrow=c(2,3))

plot(fwd.sum$cp, main="Forward Selection", xlab="Number of Variables",
ylab="Cp", type='l')
points(5, fwd.sum$cp[5], col="red", cex=2, pch=20)
plot(fwd.sum$bic, main="Forward Selection", xlab="Number of Variables",
ylab="BIC", type='l')
points(5, fwd.sum$bic[5], col="blue", cex=2, pch=20)
plot(fwd.sum$adjr2, main="Forward Selection", xlab="Number of Variables",
ylab="Adj R^2", type='l')
points(9, fwd.sum$adjr2[9], col="green", cex=2, pch=20)

plot(bwd.sum$cp, main="Backward Selection", xlab="Number of Variables",
ylab="Cp", type='l')
points(7, bwd.sum$cp[7], col="red", cex=2, pch=20)
plot(bwd.sum$bic,  main="Backward Selection",xlab="Number of Variables",
ylab="BIC", type='l')
points(3, bwd.sum$bic[3], col="blue", cex=2, pch=20)
plot(bwd.sum$adjr2, main="Backward Selection", xlab="Number of Variables",
ylab="Adj R^2", type='l')
points(7, bwd.sum$adjr2[7], col="green", cex=2, pch=20)
```




```
coef(regfit.bwd, which.min(bwd.sum$bic))
```

```
## (Intercept)           X         X.2         X.3
##    9.904257    2.089520   -2.978072   -2.075996
```

(e) Now fit a lasso model to the simulated data, again using $X, X^2, \ldots, X^{10}$ as predictors. Use cross-validation to select the optimal value of λ. Create plots of the cross-validation error as a function of λ. Report the resulting coefficient estimates, and discuss the results obtained. **The MSE is nearly constant for log(Lambda) of -5 to 0, through which points the model is reduced from 5 to 3. The "best lambda" is achieved at 0.007149 with 5 variables in the model (though the coefficients on two of them are quite small). The model for the minimum lambda LASSO is:**

$$\hat{Y} = 9.8207 + 1.312X - 2.9053X^2 - 1.4584X^3 - 0.0960X^5 + 0.00008X^{10}$$

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.5.3
```

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

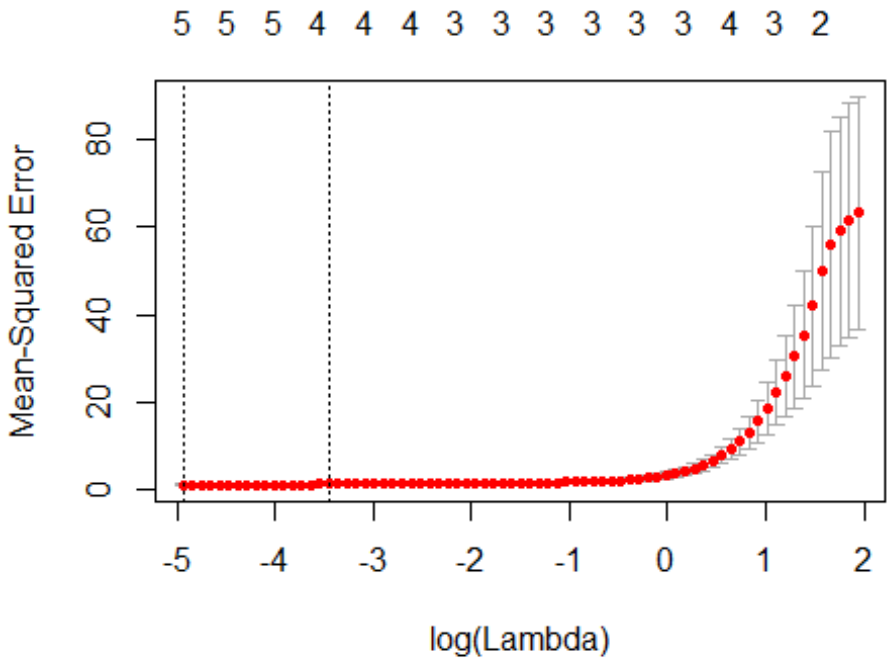
```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.5.3
```

```
## Loaded glmnet 2.0-18
```

```
set.seed(149)
x.matrix<-model.matrix(Y~.,data=df)[,-1]
```

```
lasso<-cv.glmnet(x.matrix,Y,alpha=1)
plot(lasso)
```




```
lasso$lambda.min

## [1] 0.007149413

coef(lasso, lasso$lambda.min)

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)  9.820742e+00
## X            1.312498e+00
## X.2         -2.905349e+00
## X.3         -1.458443e+00
## X.4          .
## X.5         -9.599247e-02
## X.6          .
## X.7          .
## X.8          .
## X.9          .
## X.10         8.314339e-05
```

(f) Now generate a response vector Y according to the model $Y = \beta_0 + \beta_7 X^7 + \epsilon$, and perform best subset selection and the lasso. Discuss the results obtained. **Both best subset selection and lasso were successful in selecting the number of**

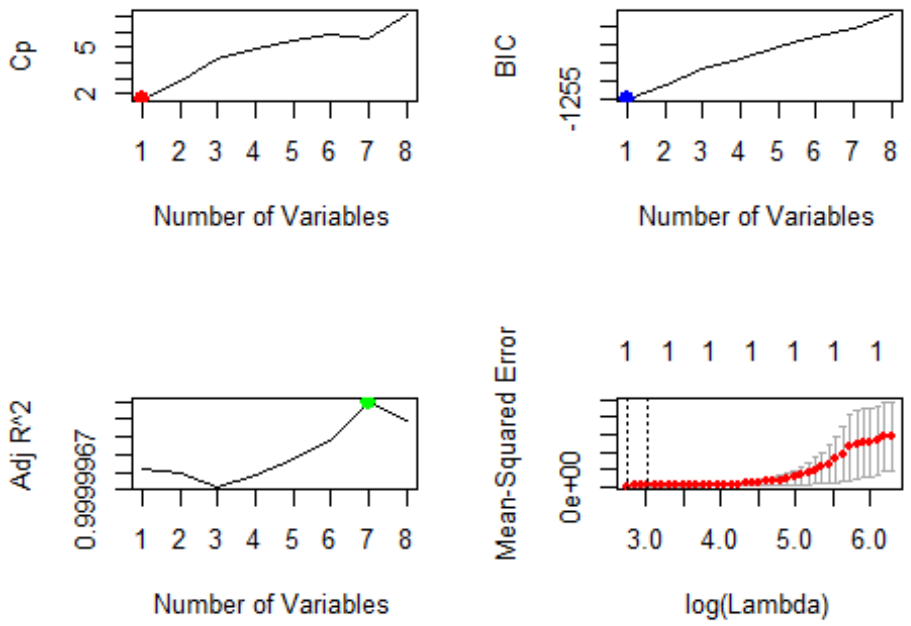
parameters in the model and both produced models that were very close to the true model, although best subset is closer (nearly perfect). Best Subset:

$$\hat{Y} = 9.9505 + 4.9977X^7$$

(g) Lasso:

$$\hat{Y} = 11.2210 + 4.8520X^7$$

```
Y_=10+5*X^7+e
df2<-data.frame(Y_, X, X^2, X^3, X^4, X^5, X^6, X^7, X^8, X^9, X^10)

bestsubset<-regsubsets(Y_~., data=df2)
bestsum<-summary(bestsubset)

par(mfrow=c(2,2))
plot(bestsum$cp, xlab="Number of Variables", ylab="Cp", type='l')
points(which.min(bestsum$cp), bestsum$cp[which.min(bestsum$cp)], col="red",
cex=2, pch=20)
plot(bestsum$bic, xlab="Number of Variables", ylab="BIC", type='l')
points(which.min(bestsum$bic), bestsum$bic[which.min(bestsum$bic)],
col="blue", cex=2, pch=20)
plot(bestsum$adjr2, xlab="Number of Variables", ylab="Adj R^2", type='l')
points(which.max(bestsum$adjr2), bestsum$adjr2[which.max(bestsum$adjr2)],
col="green", cex=2, pch=20)

coef(bestsubset, which.min(bestsum$bic))

## (Intercept)         X.7
##    9.950539    4.997684

set.seed(252)
x.matrix2<-model.matrix(Y_~.,data=df2)[,-1]
lasso2<-cv.glmnet(x.matrix2,Y_,alpha=1)
plot(lasso2)
```




```
lasso2$lambda.min

## [1] 15.64186

coef(lasso2, lasso2$lambda.min)

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 11.220962
## X            .
## X.2          .
## X.3          .
## X.4          .
## X.5          .
## X.6          .
## X.7          4.851999
## X.8          .
## X.9          .
## X.10         .
```

11 We will now try to predict per capita crime rate in the Boston data set. (a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
library(MASS)
data(Boston)
```

```r
set.seed(303)

bs_boston<-regsubsets(crim~.,data=Boston)
bs_sum<-summary(bs_boston)

which.min(bs_sum$cp)
```

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

```r
which.min(bs_sum$bic)
```

```
## [1] 3
```

```r
which.max(bs_sum$adjr2)
```

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

```r
coef(bs_boston, which.min(bs_sum$cp))
```

```
##   (Intercept)            zn           nox           dis           rad
##  19.683127801   0.043293393 -12.753707757  -0.918318253   0.532616533
##       ptratio         black         lstat          medv
##  -0.310540942  -0.007922426   0.110173124  -0.174207166
```

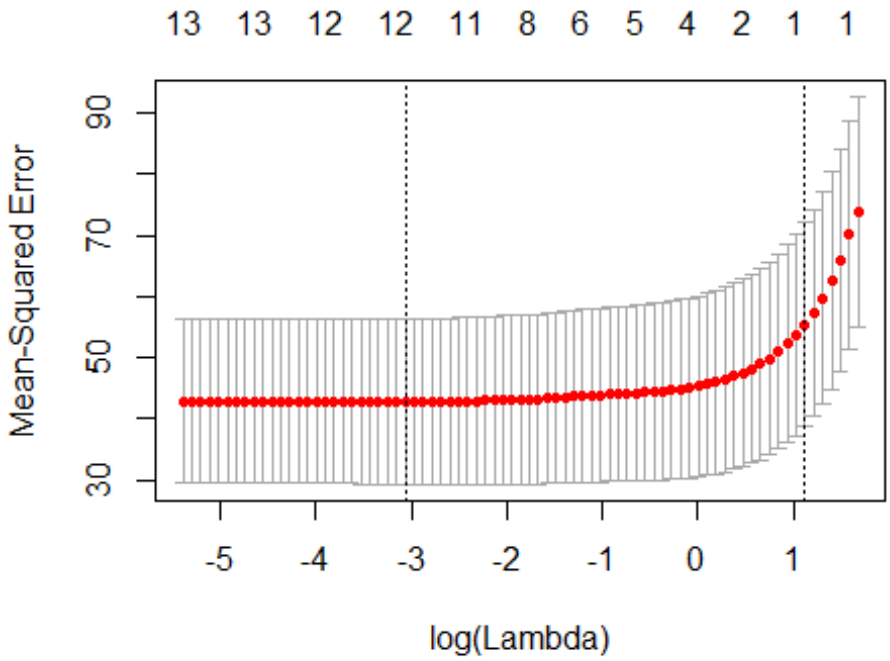
```r
coef(bs_boston, which.min(bs_sum$bic))
```

```
##  (Intercept)          rad        black        lstat
## -0.372585457  0.488172386 -0.009471639  0.213595700
```

```r
coef(bs_boston, which.max(bs_sum$adjr2))
```

```
##   (Intercept)            zn           nox           dis           rad
##  19.683127801   0.043293393 -12.753707757  -0.918318253   0.532616533
##       ptratio         black         lstat          medv
##  -0.310540942  -0.007922426   0.110173124  -0.174207166
```

```r
Y=Boston$crim
boston.matrix<-model.matrix(crim~.,data=Boston)[,-1]
lasso.boston<-cv.glmnet(boston.matrix,Y,alpha=1)
plot(lasso.boston)
```



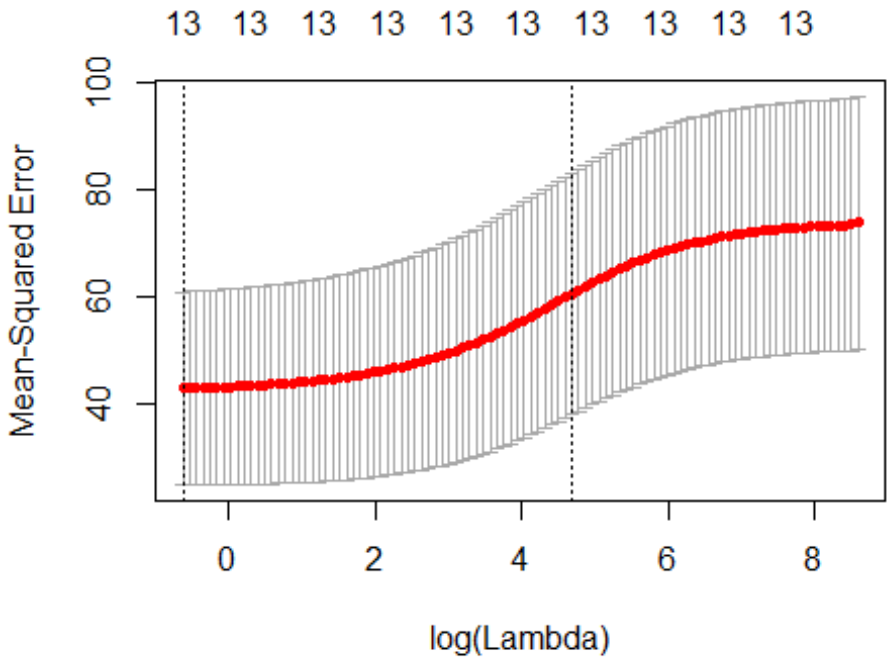

```
lasso.boston$lambda.min
```

```
## [1] 0.04674894
```

```
coef(lasso.boston, lasso.boston$lambda.min)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) 12.983186513
## zn           0.036699838
## indus       -0.072548500
## chas        -0.592035151
## nox         -7.272573553
## rm           0.248469779
## age          .
## dis         -0.805516462
## rad          0.516197063
## tax          .
## ptratio     -0.195655413
## black       -0.007546352
## lstat        0.125870714
## medv        -0.161001859
```

```
ridge.boston<-cv.glmnet(boston.matrix,Y,alpha=0)
plot(ridge.boston)
```




```
ridge.boston$lambda.min
```

```
## [1] 0.5374992
```

```
coef(ridge.boston, ridge.boston$lambda.min)
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  9.063048626
## zn           0.033002416
## indus       -0.082046152
## chas        -0.737684583
## nox         -5.393098481
## rm           0.335972073
## age          0.001962473
## dis         -0.702123641
## rad          0.422779054
## tax          0.003400607
## ptratio     -0.135911587
## black       -0.008483285
## lstat        0.142613436
## medv        -0.139604127
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.5.3
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

pcr.fit<-pcr(crim~., data=Boston, scale =TRUE, validation ="CV")
summary(pcr.fit)

## Data:    X dimension: 506 13
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             8.61    7.206    7.201    6.795    6.777    6.788    6.813
## adjCV          8.61    7.203    7.198    6.788    6.769    6.782    6.805
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       6.811    6.679    6.715     6.718     6.701     6.669     6.597
## adjCV    6.803    6.670    6.706     6.707     6.691     6.656     6.584
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       47.70    60.36    69.67    76.45    82.99    88.00    91.14
## crim    30.69    30.87    39.27    39.61    39.61    39.86    40.14
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.45    95.40     97.04     98.46     99.52     100.0
## crim    42.47    42.55     42.78     43.04     44.13      45.4
```

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, crossvalidation, or some other reasonable alternative, as opposed to using training error.

(c) Does your chosen model involve all of the features in the data set? Why or why not?

Chapter 10

1 This problem involves the K-means clustering algorithm. (a) Prove (10.12).

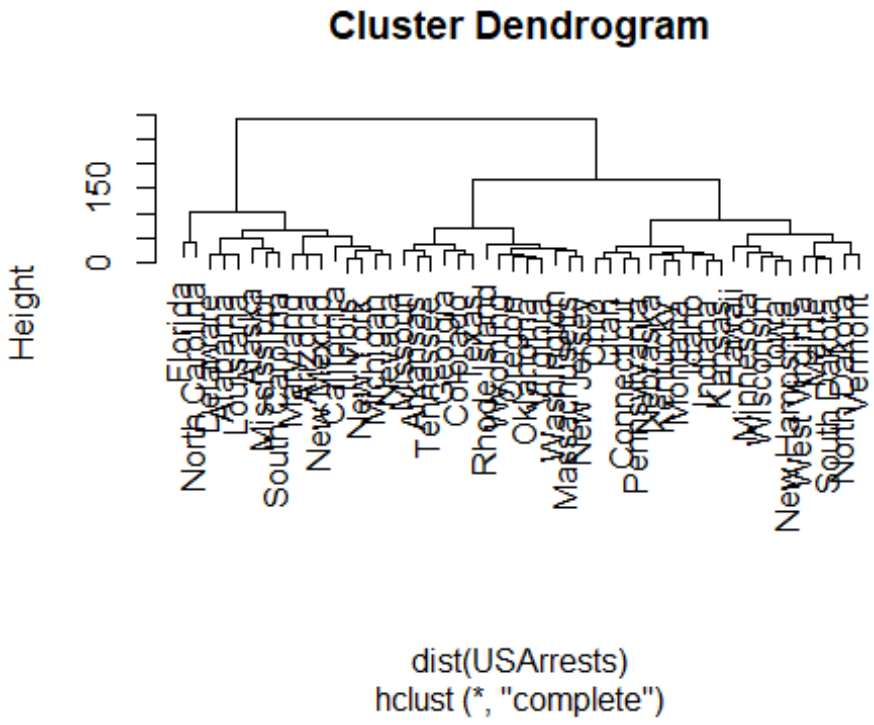
(b) On the basis of this identity, argue that the K-means clustering algorithm (Algorithm 10.1) decreases the objective (10.11) at each iteration. **2**

3

4

9 Consider the USArrests data. We will now perform hierarchical clustering on the states.
(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
hc.arrests <- hclust(dist(USArrests), method = "complete")
plot(hc.arrests)
```



(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters? **The states are shown below in alphabetical order with their cluster number below them. All states with the same number (1,2,3) belong to the same cluster.**

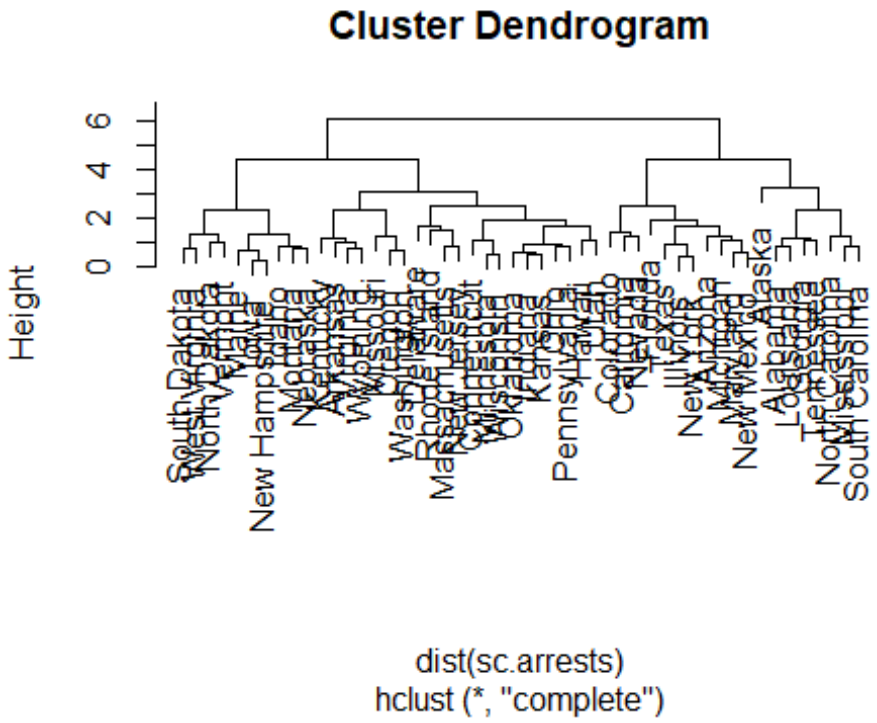
```
cutree(hc.arrests ,3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              1              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              2              1              3              1              2
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              3              3              1              3              2
##     New Mexico       New York North Carolina   North Dakota           Ohio
```

```
##             1             1             1             3             3
##      Oklahoma        Oregon  Pennsylvania  Rhode Island South Carolina
##             2             2             3             2             1
##  South Dakota     Tennessee         Texas          Utah       Vermont
##             3             2             2             3             3
##      Virginia    Washington West Virginia     Wisconsin       Wyoming
##             2             2             3             3             2
```

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
sc.arrests=scale(USArrests)
hc.arrests.scale <- hclust(dist(sc.arrests), method = "complete")
plot(hc.arrests.scale)
```




```
cutree(hc.arrests.scale,3)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              2              3              2
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              3              2              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              3              2              3              3
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              3              1              3              2
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              2              3              1              3
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
```

```
##              3              3              2              3              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              2              2              1              3              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              3              1              2              3              3
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              3              3              3
```

(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer. **The three clusters obtained using the scaled data are different from those obtained from the original data. For example, cluster 2 from the original data contains mostly values from cluster 3 of the scaled data, but also includes 2 values from each of clusters 1 and 2. This indicates that the variables have different units and scaling is advised.**

```
table(cutree(hc.arrests,3), cutree(hc.arrests.scale,3))
```

```
##
##      1  2  3
##   1  6  9  1
##   2  2  2 10
##   3  0  0 20
```

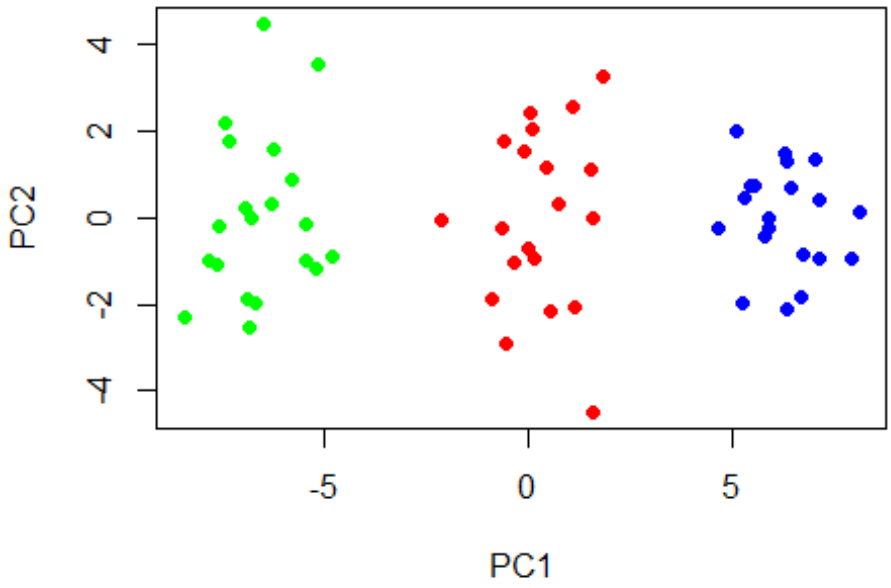
10 In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data. (a) Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Hint: There are a number of functions in R that you can use to generate data. One example is the rnorm() function; runif() is another option. Be sure to add a mean shift to the observations in each class so that there are three distinct classes. **Mean for class A is 0, for class B is -1 and for class C is 0.8.**

```
set.seed(936)

class <- rep(c("A", "B", "C"),20)
x <- matrix(rnorm(60*50), ncol=50)
x[class=="A",]= x[class=="A"]
x[class=="B",]= x[class=="B",] - 1
x[class=="C",]= x[class=="C",] + .8
```

(b) Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors. **The classes**

are well separatedm though their standard deviations appear to be slightly different.

```
pr.out=prcomp(x, scale=FALSE)

Cols=function(vec){
  cols=rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}

plot(pr.out$x[,1:2], col=Cols(class), pch =19)
```



(c) Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels? Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same. **The classes are perfectly separated by the k-means clustering. Class A is cluster 2, class B is cluster 1, and class C is cluster 3.**

```
km.out=kmeans(x,3, nstart=20)

table(km.out$cluster, class)

##    class
##      A  B  C
```

```
##   1  0  0 20
##   2 20  0  0
##   3  0 20  0
```

(d) Perform K-means clustering with K = 2. Describe your results. **Class B is entirely contained in cluster 2 along with one value from class A. The remaining class A and class C are all in cluster 1. It makes sense that A and C would end up in the same cluster if there are only 2 since the means for class A (0) and class C (0.8) are closer to each other than to the mean for class B (-1).**

```
km.2=kmeans(x,2,nstart=20)

table(km.2$cluster, class)

##    class
##      A  B  C
##   1 19  0 20
##   2  1 20  0
```

(e) Now perform K-means clustering with K = 4, and describe your results. **Class A is entirely contained in cluster 4 and class B is entirely contained in cluster 3 while class C is split between clusters 1 and 2. This is to be expected since the mean of class C is between the means classes A and B so there is greater separation for classes A and B and with 4 clusters class C would be split.**

```
km.4=kmeans(x,4,nstart=20)

table(km.4$cluster, class)

##    class
##      A  B  C
##   1  0  0 11
##   2  0 20  0
##   3  0  0  9
##   4 20  0  0
```

(f) Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results. **The k-means clusters are again perfectly separated into classes A, B, and C.**

```
km.pca=kmeans(pr.out$x[,1:2],3,nstart=20)

table(km.pca$cluster, class)

##    class
##      A  B  C
##   1 20  0  0
##   2  0 20  0
##   3  0  0 20
```

(g) Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain. **These results are the same as those in part (b) - perfect separation. It is likely that the standard deviations for the 3 groups were similar to begin with so scaling had little effect and is unnecessary in this case.**

```
km.sc=kmeans(scale(x),3,nstart=20)

table(km.sc$cluster, class)
```

```
##    class
##      A  B  C
##   1  0  0 20
##   2  0 20  0
##   3 20  0  0
```