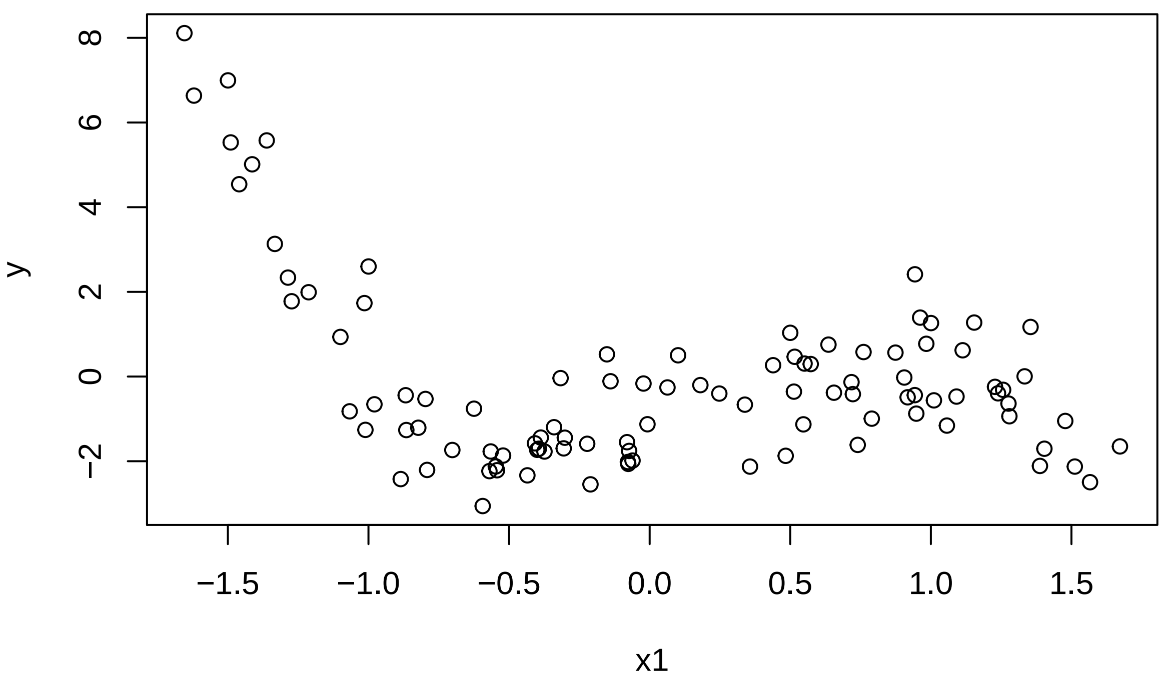
Homework 6

Drew Dahlquist

3/19/2022

1)

```r
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)
```



(a)

x_1, x_2, and x_3 should be found to be associated with the response variable.

(b)

1-predictor: x_5

3-predictor: x_1, x_2, x_3

5-predictor: x_1, x_4, x_5, x_6, x_9

```r
library(leaps)

regfit.full = regsubsets(y ~ ., data=data_df, nvmax=10)
summary(regfit.full)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data_df, nvmax = 10)
## 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 10
## 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 )  "*" "*" "*" "*" " " "*" "*" "*" "*" " "
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" "*" " "
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

(c)

Cp: 5 predictor model

BIC: 3 predictor model

Adj R^2: 6 predictor model

```r
reg.summary = summary(regfit.full)

crit = matrix(NA,3,10)
colnames(crit) = c('1','2','3','4','5','6','7','8','9','10')
rownames(crit) = c('Cp','BIC','Adj R^2')

crit[1,] = reg.summary$cp
crit[2,] = reg.summary$bic
crit[3,] = reg.summary$adjr2

crit
```

```
##                   1           2           3           4           5
## Cp      192.9947822   39.432147   6.5456192   4.5700953   3.4338473
## BIC     -50.0473358 -122.723720 -148.4259008 -147.9385577 -146.7059454
## Adj R^2   0.4414574    0.739456   0.8055722   0.8114517   0.8157653
```

```
##                  6            7            8            9           10
## Cp         4.159326    5.5538280    7.1508258    9.0003467   11.0000000
## BIC     -143.504509 -139.5731905 -135.4190471 -130.9828112 -126.3780305
## Adj R^2    0.816380    0.8156307    0.8144434    0.8126984    0.8105946
```

(d)

Cp: -1.0804923, 1.4200656, 1.3717432, -1.2374129, -0.317148, 0.0884622

BIC: -1.3123085, 1.988786, 1.4872258, -1.9318133

Adj R^2: -1.0466359, 1.537114, -1.7340376, 2.3224985, 0.4136039, -1.5442441, 0.2800249

(e)

The model with 4 predictors has the best BIC score. x_1, x_2, x_5, and x_9 are included in the model.

```r
regfit.fwd = regsubsets(y ~ ., data=data_df, nvmax=10, method="forward")

summary(regfit.fwd)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data_df, nvmax = 10, method = "forward")
## 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 10
## Selection Algorithm: forward
##           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 )  "*" "*" "*" "*" "*" "*" " " "*" "*" " "
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" "*" " "
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```r
bic = matrix(NA,1,10)
colnames(bic) = c('1','2','3','4','5','6','7','8','9','10')
rownames(bic) = c('BIC')

bic[1,] = summary(regfit.fwd)$bic

bic
```

```
##             1         2         3         4         5         6         7
## BIC -50.04734 -122.7237 -134.8986 -146.8954 -143.7764 -142.1926 -139.4302
```

```
##              8         9       10
## BIC -135.2293 -130.9828 -126.378
```

(f)

The model with 4 predictors has the best Cp score. x_1, x_3, x_4, and x_6 are included in the model.

```
regfit.bwd = regsubsets(y ~ ., data=data_df, nvmax=10, method="backward")

summary(regfit.bwd)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = data_df, nvmax = 10, method = "backward")
## 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 10
## Selection Algorithm: backward
##           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 )  "*" "*" "*" "*" " " "*" "*" "*" "*" " "
## 9  ( 1 )  "*" "*" "*" "*" "*" "*" "*" "*" "*" " "
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

```
cp = matrix(NA,1,10)
colnames(cp) = c('1','2','3','4','5','6','7','8','9','10')
rownames(cp) = c('Cp')

cp[1,] = summary(regfit.fwd)$cp

cp
```

```
##           1        2        3        4        5        6       7        8
## Cp 192.9948 39.43215 20.81984 5.561733 6.152039 5.349925 5.68194 7.320154
##           9 10
## Cp 9.000347 11
```

(g)

The best models, according to Cp, BIC, and Adj R^2, from full subset selection had 5, 3, and 6 predictors, respectively. The best models from forward and backwards subset selection both has 4 predictors, although the individual predictors were not the same. Knowing the true form of the data, using full subset selection with the BIC score would yield the best model as it only includes the 3 true variables that determine the response variable. A disadvantage of both forward and backward subset selection is that they make "greedy" decisions when fitting each model, that they are then stuck with as they add more predictors. Full subset selection avoids this but is more computationally expensive.

2)

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

(a)

```
set.seed(1)
size=100
train=sample(size, 0.7*size)
```

(b)

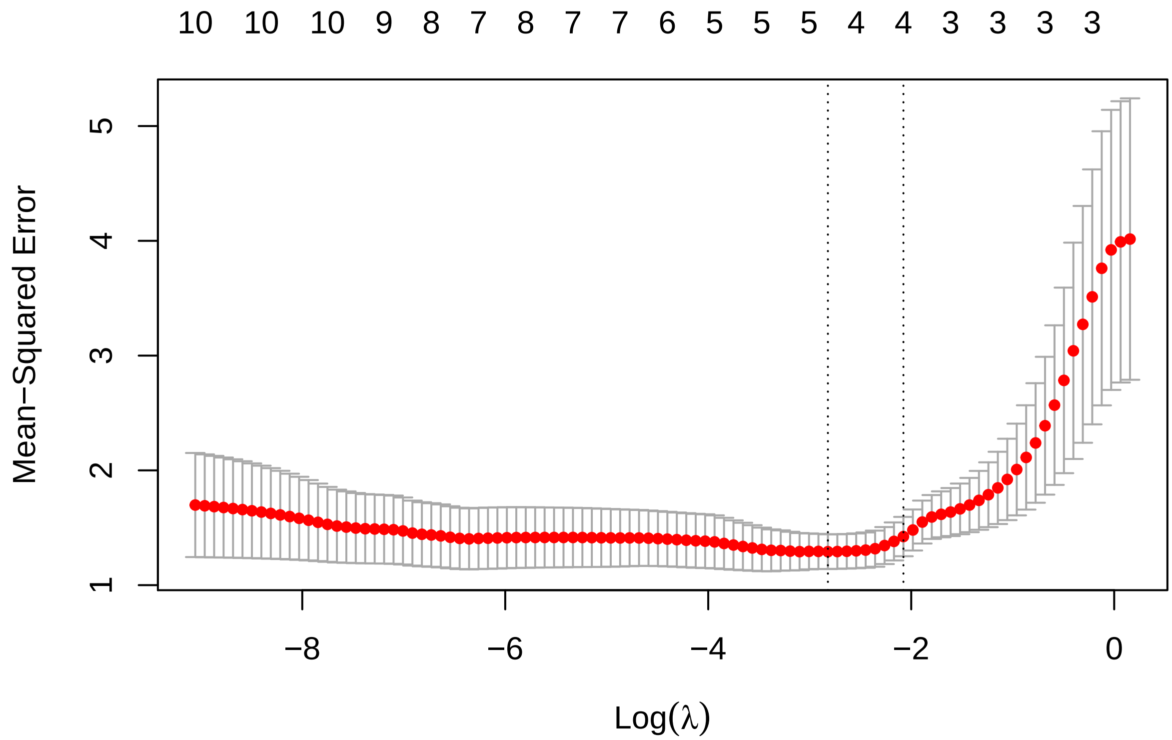
The best value of λ is approximately 4.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-3
```

```
x=data_mat[train,2:11]
y=data_mat[train,1]

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

## CV to find best lambda
set.seed(2)
cv.out=cv.glmnet(x,y,alpha=1)
plot(cv.out)
```




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

(c)

```
x=data_mat[-train,2:11]
y=data_mat[-train,1]
lasso.pred=predict(lasso.mod,s=bestlam,newx=x)
mean((lasso.pred-y)^2)
```

```
## [1] 1.011545
```

Test MSE = 1.0115449.

(d)

The regression coefficients of the model with the best lambda parameter isn't too far off the true data generating process. The intercept, and x_1 and x_2 coefficients are near the true values, however x_3 is left out in favor of x_4 and x_5.

```
out=glmnet(data_mat[,2:11],data_mat[,1],alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:10,]
lasso.coef
```

```
## (Intercept)          x1          x2          x3          x4          x5
## -1.20476328  0.49388048  1.09932056 -0.02011887  0.14719977 -0.51405244
##          x6          x7          x8          x9
##  0.00000000  0.00000000  0.00000000  0.00000000
```

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

```
## (Intercept)          x1          x2          x3          x4          x5
## -1.20476328  0.49388048  1.09932056 -0.02011887  0.14719977 -0.51405244
```

3)

(a)

5 principal components has the best MSEP as shown by the validation plot, but we could also likely consider 3 principal components since it's about within 1 standard deviation from the 5 principal component MSEP.

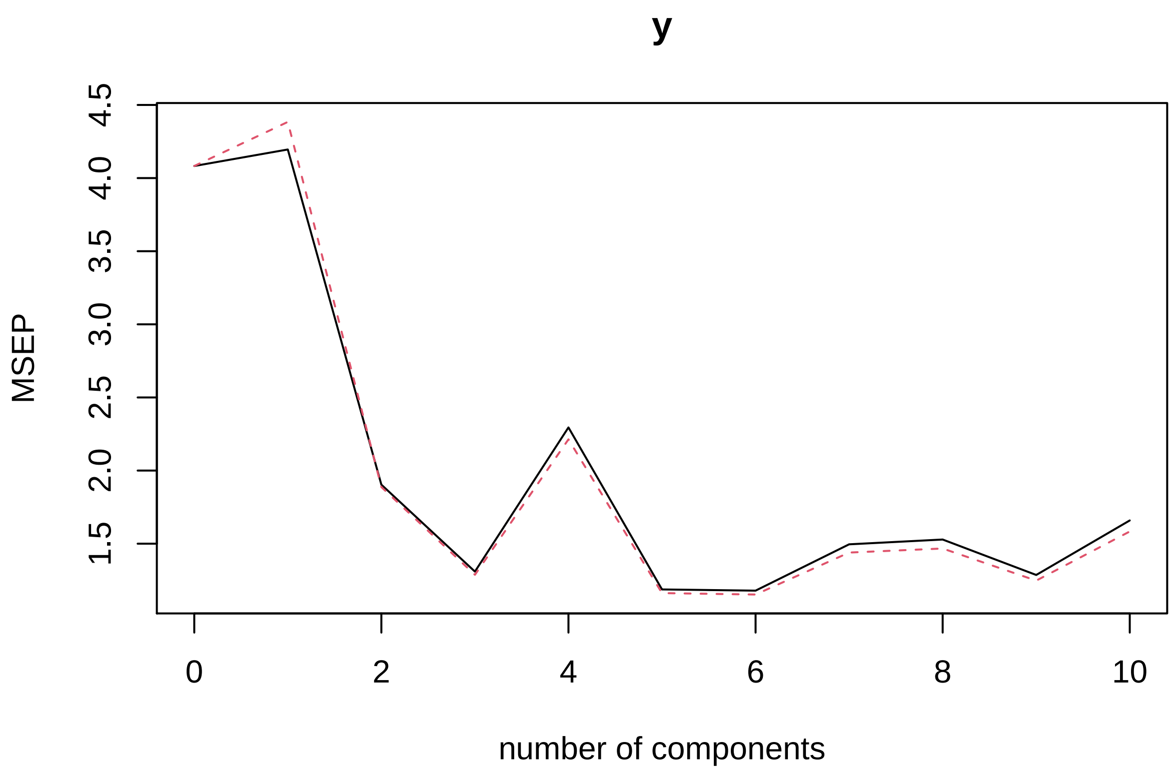
```r
library(pls)
```

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

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

```r
set.seed(3)
pcr.fit= pcr(y ~ ., data=data_df,subset=train,scale=TRUE, validation="CV")

summary(pcr.fit)
```

```
## 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  5 comps  6 comps
## CV           2.021    2.048    1.380    1.144    1.515    1.090    1.086
## adjCV        2.021    2.094    1.374    1.135    1.488    1.078    1.073
##        7 comps  8 comps  9 comps  10 comps
## CV       1.223    1.236    1.134     1.288
## adjCV    1.200    1.211    1.117     1.258
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X  55.3091    91.48    96.90    99.48    99.85    99.99   100.00   100.00
## y   0.3512    55.41    70.22    70.28    78.66    79.40    79.59    79.96
##    9 comps  10 comps
## X   100.00    100.00
## y    80.24     80.41
```

```r
validationplot(pcr.fit,val.type="MSEP")
```



(b)

```
pcr.pred=predict(pcr.fit, x[-train,], ncomp=3)
mean((pcr.pred-y[-train])^2)
```

```
## [1] 1.789957
```

Test MSE = 1.7899573.

(c)

96.29% variability of the predictors is explained by the PCs. 72.19% variability of the response variable is explained by the PCs.

```
pcr.fit=pcr(y~., data=data_df, scale=TRUE,ncomp=3)
summary(pcr.fit)
```

```
## Data:    X dimension: 100 10
##  Y dimension: 100 1
## Fit method: svdpc
## Number of components considered: 3
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps
## X    47.15    91.25    96.29
## y    54.76    65.06    72.19
```