Homework 3

Casey Gibson

3/1/2018

Problem 1

a

Best subset selection will have the smallest training RSS, because it is able to search accross all models to choose the best one. Forward and backward selection may have the same training RSS, but not lower.

b

Since any of these selection techniques are prone to overfitting there is no way to know a-priori which will have the lowest test RSS.

c

i

True - forward selection can only add predictors to the existing k-variable model.

ii

True - the k-variable is obtained by removing a variable from the $k+1$-variable model.

iii

False - No link exists between forward and backward selection.

iv

False - see above

v

False - Best subset selection is allowed to vary all predictors when going from k-variable to $k+1$-variable models.

Problem 2

a)

iii - Less flexible because we have imposed a constraint on the βs and improved prediction accuracy. This constraint also reduces the variance, and as long as the bias increase is smaller than the variance decrease, the MSE will go down.

b)

iii - Less flexible because we have imposed a constraint on the βs and improved prediction accuracy. This constraint also reduces the variance, and as long as the bias increase is smaller than the variance decrease, the MSE will go down.

Problem 3

a)

iv - As we relax the constraints on β the model is more flexible so we expect a decrease on the training data.

b)

ii - As we relax the constraints on β the model is more flexible so we expect a decrease on the testing data at first, until we overfit.

c)

iii - Variance increases as the model becomes more flexible because we have more possible β values with positive probability.

d)

iv - Bias decreases as the model becomes more flexible, since the OLS estimate is unbiased and the fully unconstrained estimate = OLS.

Problem 4

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

We can add and subtract 0 in the following way

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \bar{x_j}\beta_j + \sum_{j=1}^{p} \bar{x_j}\beta_j - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \sum_{j=1}^{p} \bar{x_j}\beta_j) - \sum_{j=1}^{p}(x_{ij} - \bar{x_j})\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

So we can see from this that

$$\beta_0^c = \beta_0 + \sum_{j=1}^{p} \bar{x_j}\beta_j$$

and all other $\beta_j^c = \beta_j \ for \ j \neq 0$.

And that by minimizing the first, we are indeed minimizing the second. We could have also shown this using the partial derivative hint, but this seems easier.

Problem 5

a

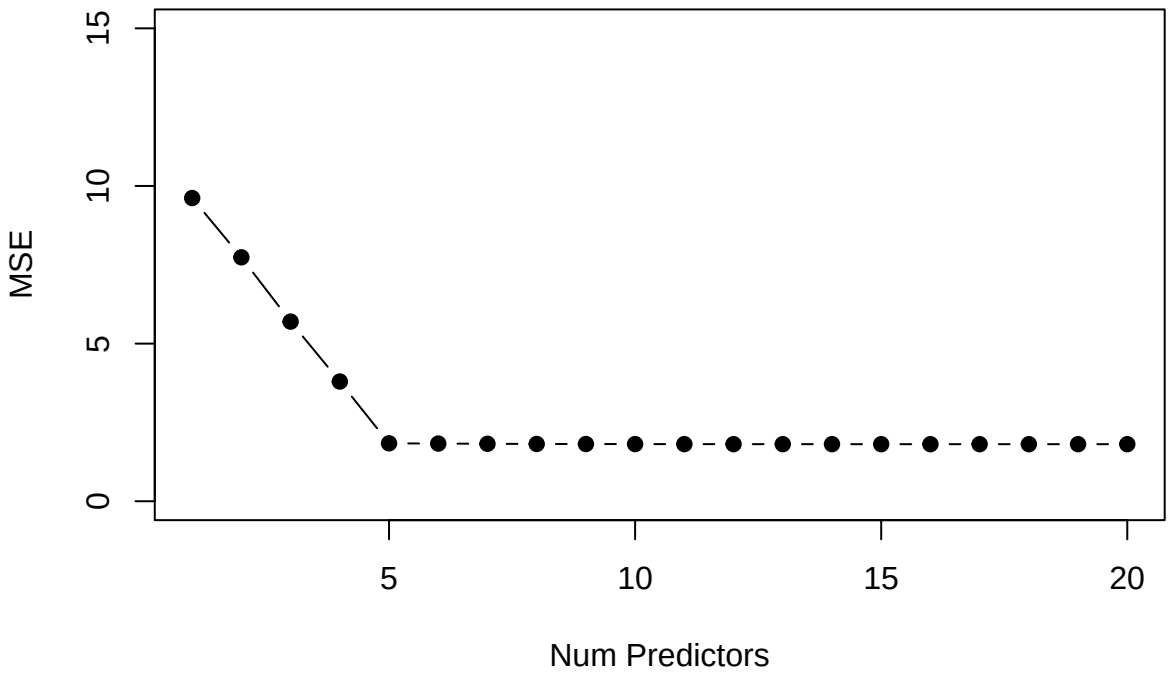
```
set.seed(1)
p = 20
n = 1000

beta <- c(rep(1,5),rep(0,15))
xs <- matrix(rnorm(n*p,0,sqrt(2)),nrow=n,ncol=p)
error <- rnorm(n,0,sqrt(2))

y <- xs%*%beta + error
```

b

```
train_indexes <- sample(n,size=500,replace = FALSE)
train_y <- y[train_indexes]
test_y <- y[-train_indexes]

train_x <- xs[train_indexes,]
test_x <- xs[-train_indexes,]
```

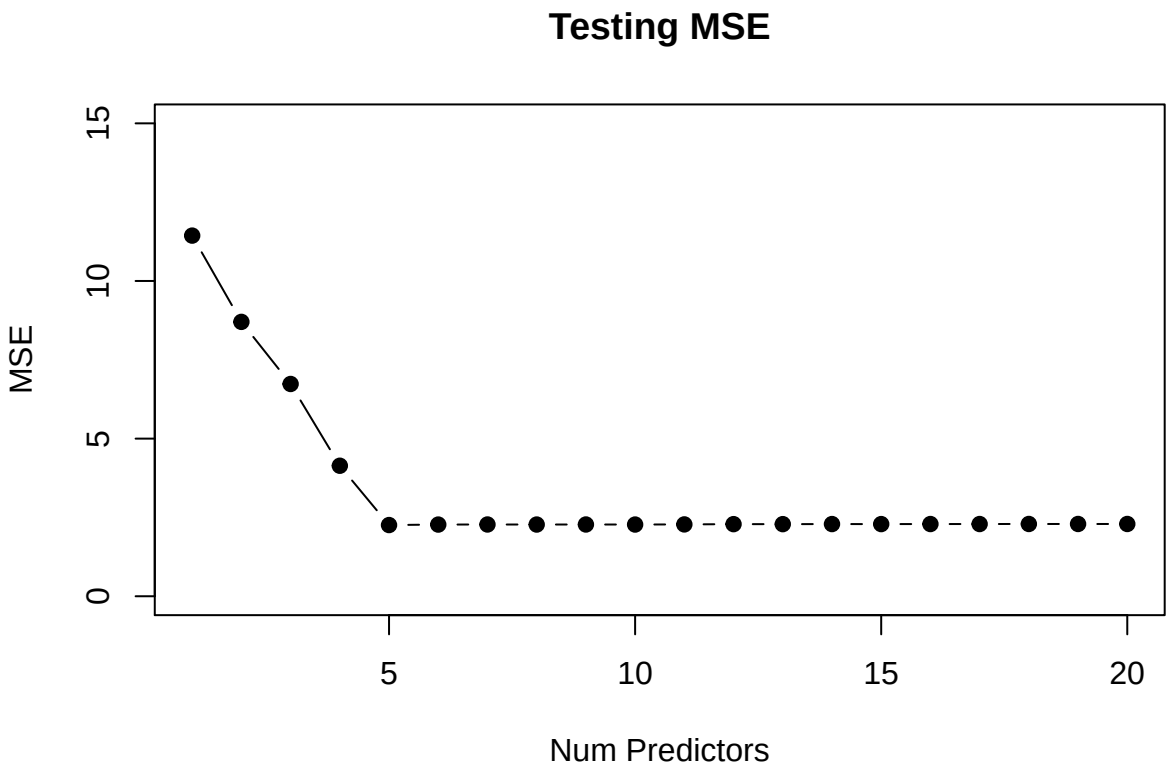
c

```
library(ISLR)
library(leaps)

data.train =data.frame(y = train_y,x = train_x)
data.test =data.frame(y = test_y,x = test_x)

regfit = regsubsets(y ~. ,data=data.train,nvmax=20)

data.train$"(Intercept)" <- rep(0,500)
errors <- rep(NA,20)

for (i in 1:20){
  coef = coef(regfit, id=i)
  pred <- as.matrix(data.train[,names(coef)])%*%matrix(coef,ncol=1)
 # print (pred)
  errors[i]=mean((data.train$y-pred)^2)
}

plot(errors,type="b",main="Training MSE",xlab="Num Predictors",ylab="MSE",pch=19,ylim=c(0,15))
```



d

```
library(ISLR)
library(leaps)

data.train =data.frame(y = train_y,x = train_x)
regfit = regsubsets(y ~. ,data=data.train,nvmax=20)

data.test$"(Intercept)" <- rep(1,500)
errors <- rep(NA,20)

for (i in 1:20){
  coef = coef(regfit, id=i)
  pred <- as.matrix(data.test[,names(coef)])%*%matrix(coef,ncol=1)
  errors[i]=mean((data.test$y-pred)^2)
}

plot(errors,type="b",pch=19,main="Testing MSE",xlab="Num Predictors",ylab="MSE",ylim=c(0,15))
```



e

```
print (which.min(errors))
```

```
## [1] 5
```

We can see that the model with 5 predictors has the lowest test MSE, which is consistent with the true underlying data generating mechanism since we selected 5 active predictors to generate the data.

f

```
print (coef(regfit, id=20))
```

```
##  (Intercept)          x.1          x.2          x.3          x.4
##  0.078514549  1.070602149  1.028086271  1.005139850  0.953969105
##          x.5          x.6          x.7          x.8          x.9
##  1.080606089  0.031072049  0.062768208  0.003624725  0.028478852
##         x.10         x.11         x.12         x.13         x.14
##  0.021670769  0.013077318  0.053591860  0.032750655  0.008445044
##         x.15         x.16         x.17         x.18         x.19
## -0.004185222 -0.007736939 -0.003121689  0.057019065 -0.006997981
##         x.20
##  0.003992506
```

We can see the coefficients for the first 5 predictors are much higher than those of the remaining 15, which is again consistent with how we generated the data.

Problem 6

a

```
library(MASS)
require(glmnet)  # ridge and lasso
```

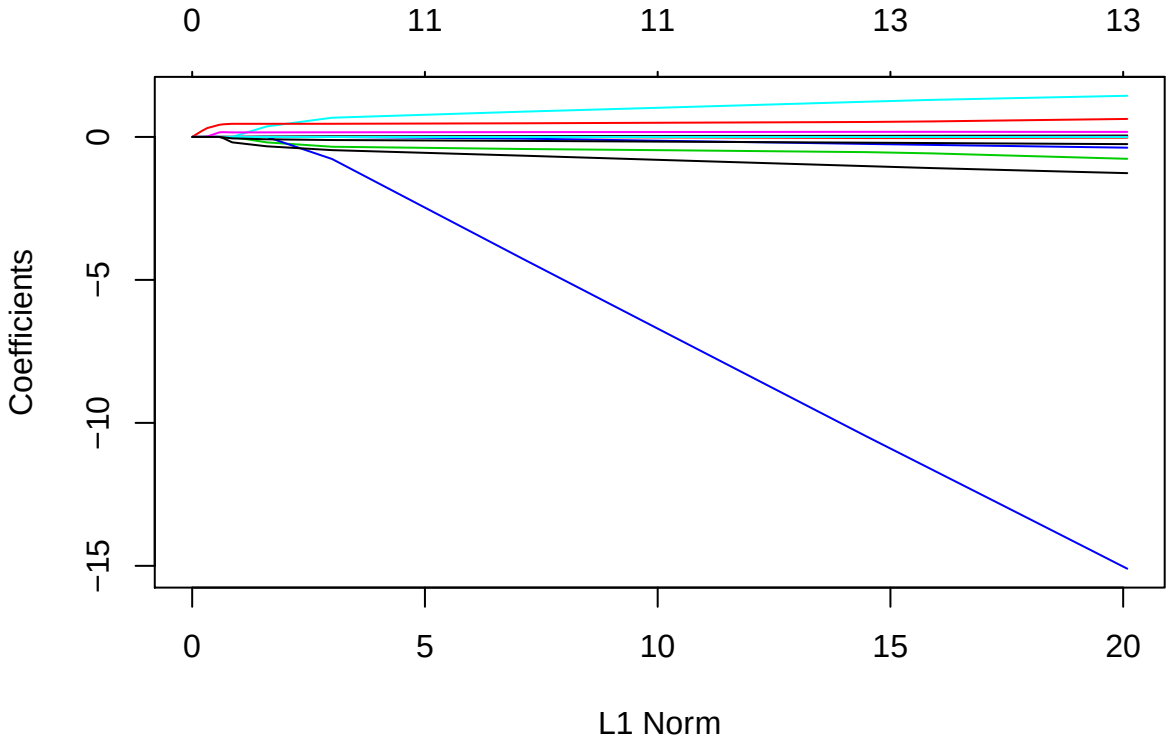
```
## Loading required package: glmnet
```

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

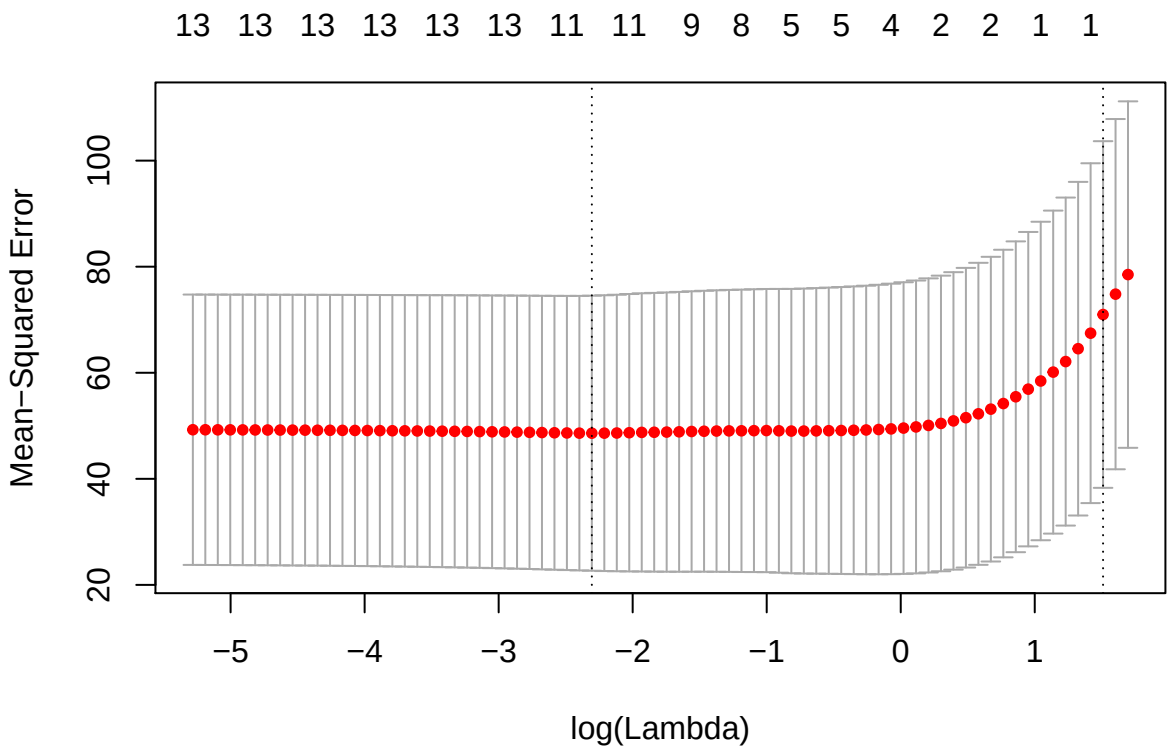
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-12
```

```
grid=10^seq(10,-2,length=100)
data(Boston)
set.seed(1)

x <- model.matrix(crim~., data=Boston)[,-1]
y <- Boston$crim
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]

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




```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




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

lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])

out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)
print (c("Best Lambda", bestlam))
```

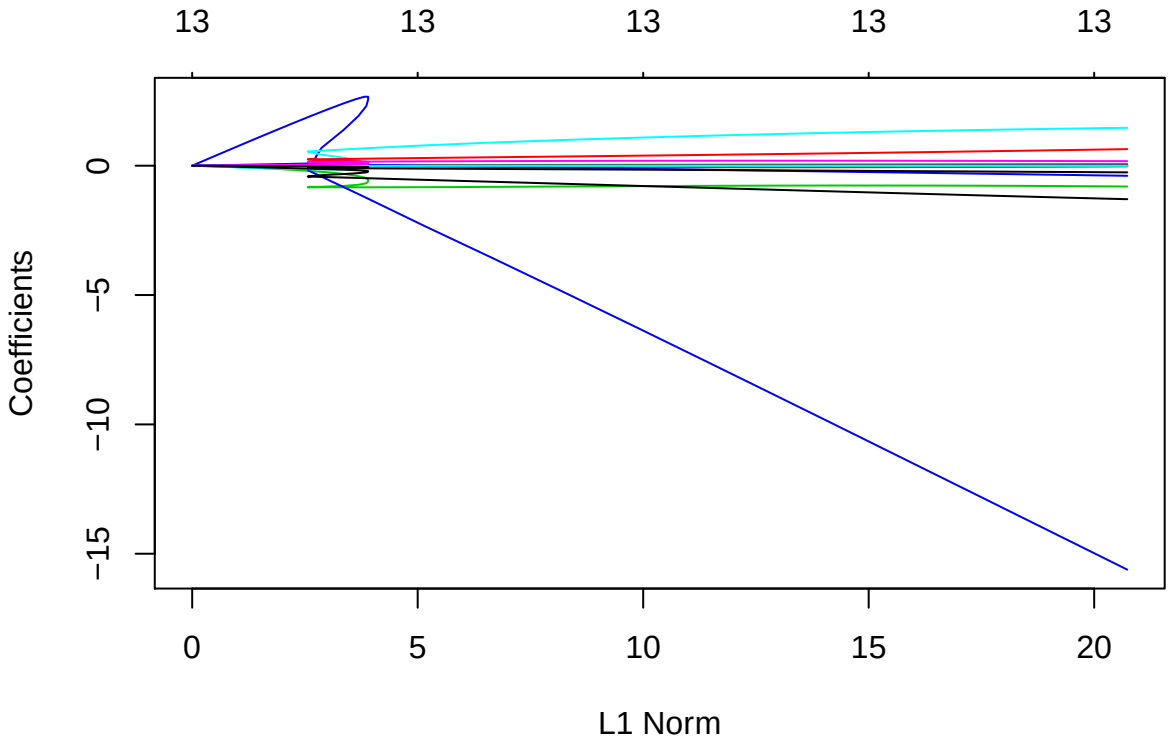
```
## [1] "Best Lambda"        "0.0997955327377658"
```

```
lasso.coef
```

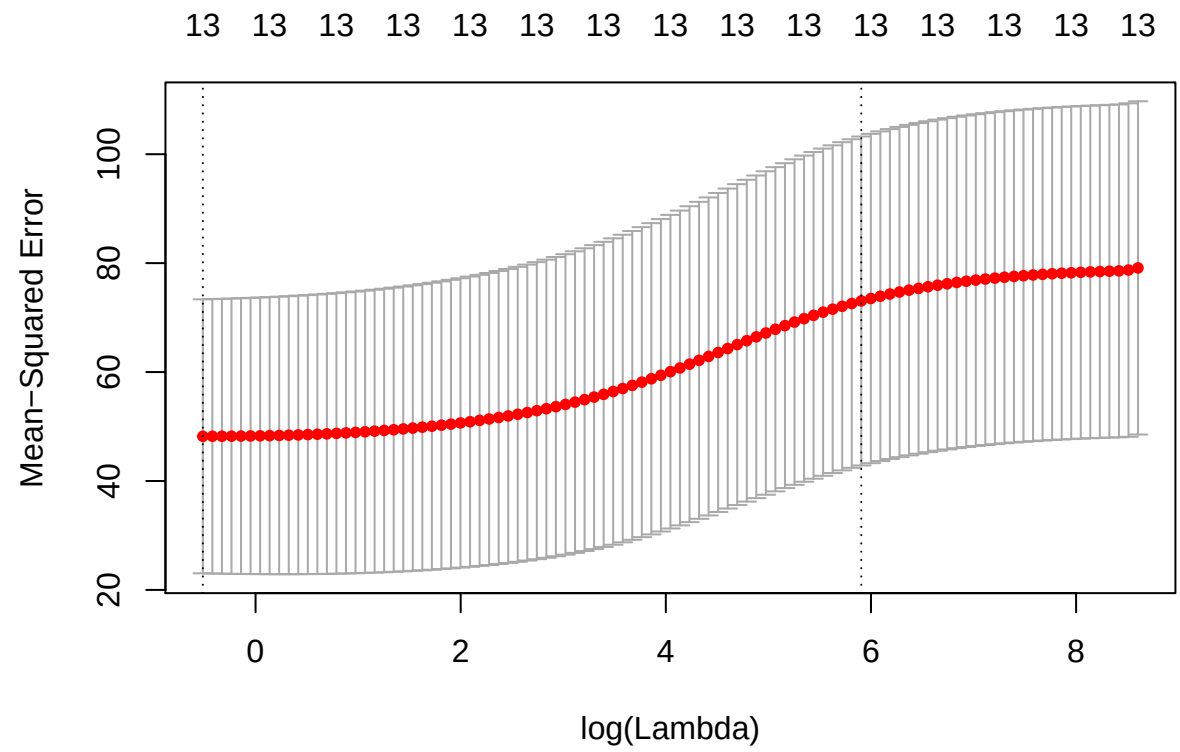
```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  9.262700913
## zn           0.031356409
## indus       -0.051023135
## chas        -0.512648901
## nox         -3.755451657
## rm           0.041320041
## age          .
## dis         -0.600700390
## rad          0.494793892
## tax          .
## ptratio     -0.107509984
## black       -0.007556396
## lstat        0.118431941
## medv        -0.126165598
```

b

```
ridge.mod=glmnet(x[train,],y[train],alpha=0,lambda=grid)
plot(ridge.mod)
```




```
cv.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```



```
bestlam=cv.out$lambda.min
ridge.pred=predict(ridge.mod,s=bestlam,newx=x[test,])
ridge.coef = predict(out,type="coefficients",s=bestlam)
print (c("Best Lambda", bestlam))
```

```
## [1] "Best Lambda"       "0.598258501284789"
```

```
ridge.coef
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  0.577054605
## zn           .
## indus        .
## chas         .
## nox          .
## rm           .
## age          .
## dis         -0.013300001
## rad          0.454844884
## tax          .
## ptratio      .
## black       -0.005284489
## lstat        0.126370091
## medv        -0.043078735
```

c

```
print (mean((lasso.pred-y.test)^2))
```

```
## [1] 38.3096
```

```
print (mean((ridge.pred-y.test)^2))
```

```
## [1] 38.36464
```

d

We see that both age and tax are zeroed out in the lasso model, which has a lower MSE than ridge.