Solution to Q2-Q4 in Homework 2

2023-10-3

Problem 2 (9 pts)

Part 1. (3 pts)

Generate the training data for $\rho = 0.1$ and $n = 100$.

```r
# clear memory
rm(list=ls())

# you need MASS package to generate multivariate normal distribution
library(MASS)
set.seed(0)

# define a function of generating the data
data_Gen <- function(n, rho) {
  mu <- rep(0, 3)
  Sigma <- matrix(c(1, 0, 0, 0, 1, rho, 0, rho, 1), byrow = T, nrow = 3)
  X <- mvrnorm(n, mu, Sigma)
  epsilon <- rnorm(n, 0, 1)
  beta <- c(0.5, 0.5, 0)
  y <- X %*% beta + epsilon
  return(list(y = y, X = X))
}

n_train <- 100
rho <- 0.1

train_data <- data_Gen(n_train, rho)
y_train <- train_data$y
X_train <- train_data$X
```

Fit each predictor and compute their training MSEs.

```r
# fit predictors
lm_1 <- lm(y_train ~ X_train[,c(1,2)] - 1)
lm_2 <- lm(y_train ~ X_train[,c(1,2,3)] - 1)
lm_3 <- lm(y_train ~ X_train[,c(1,3)] - 1)

# compute training MSEs
cat("The training MSEs are: f1:", sum(lm_1$residuals ** 2) / n_train,
    " f2:", sum(lm_2$residuals ** 2) / n_train,
    " f3:", sum(lm_3$residuals ** 2) / n_train)
```

```
## The training MSEs are: f1: 0.8744107  f2: 0.8620776  f3: 1.223834
```

We obtain the same conclusion between $\hat{f}_1$ and $\hat{f}_2$. For comparison between $\hat{f}_1$ and $\hat{f}_3$, we see that $\hat{f}_1$ has a smaller training MSE. (Note the latter comparison could be different due to numerical randomness.)

Part 2. (3 pts)

Now repeat step (a) $N = 100$ times and compute the training MSEs.

```r
N <- 100
# We use a N by 3 matrix to store the training of each predictor
train_MSEs <- matrix(0, N, 3)

for (i in 1:N) {
  # generate the data
  train_data <- data_Gen(n_train, rho)
  y_train <- train_data$y
  X_train <- train_data$X

  lm_1 <- lm(y_train ~ X_train[,c(1,2)] - 1)
  lm_2 <- lm(y_train ~ X_train[,c(1,2,3)] - 1)
  lm_3 <- lm(y_train ~ X_train[,c(1,3)] - 1)

  # compute and store the training MSEs
  train_MSEs[i,] <- c(sum(lm_1$residuals ** 2) / n_train,
                      sum(lm_2$residuals ** 2) / n_train,
                      sum(lm_3$residuals ** 2) / n_train)
}

table(apply(train_MSEs, 1, which.min))
```

```
##
##   2
## 100
```

```r
table(apply(train_MSEs, 1, which.max))
```

```
##
##   3
## 100
```

```r
cat("The averaged training MSEs are",
    paste(c("f1:,", "f2:", "f3:"), round(apply(train_MSEs, 2, mean), 3)))
```

```
## The averaged training MSEs are f1:, 0.994 f2: 0.982 f3: 1.233
```

We find that the comparison among three predictors stays the same across repetitions. The comparison between $\hat{f}_1$ and $\hat{f}_2$ is as expected since the proof of $\mathrm{MSE}(\hat{f}_1) \geq \mathrm{MSE}(\hat{f}_2)$ holds for any training data. The comparison between $\hat{f}_1$ and $\hat{f}_3$ is also expected as $\hat{f}_1$ uses the true features X_1 and X_2 whereas $\hat{f}_3$ uses X_1 and X_3. Since the correlation between X_2 and X_3 is small, we should expect a difference between them for explaining the variances in the response. (Note the latter comparison could be different due to numerical randomness.)

Part 3. (3 pts)

Let us re-do step (b) for $\rho = 0.95$.

```r
N <- 100
# We use a N by 3 matrix to store the training MSEs of each predictor
train_MSEs <- matrix(0, N, 3)

for (i in 1:N) {
  # generate the data
  train_data <- data_Gen(n_train, rho = 0.99)
  y_train <- train_data$y
  X_train <- train_data$X

  lm_1 <- lm(y_train ~ X_train[,c(1,2)] - 1)
  lm_2 <- lm(y_train ~ X_train[,c(1,2,3)] - 1)
  lm_3 <- lm(y_train ~ X_train[,c(1,3)] - 1)

  # compute and store the training MSEs
  train_MSEs[i,] <- c(sum(lm_1$residuals ** 2) / n_train,
                      sum(lm_2$residuals ** 2) / n_train,
                      sum(lm_3$residuals ** 2) / n_train)
}


table(apply(train_MSEs, 1, which.min))
```

```
##
##   2
## 100
```

```r
table(apply(train_MSEs, 1, which.max))
```

```
##
##  1  3
## 38 62
```

```r
cat("The averaged training MSEs are",
    paste(c("f1:,", "f2:", "f3:"), round(apply(train_MSEs, 2, mean), 3)))
```

```
## The averaged training MSEs are f1:, 0.974 f2: 0.961 f3: 0.979
```

We find the same conclusion between $\hat{f}_1$ and $\hat{f}_2$. Differently, we see that the comparison between MSE($\hat{f}_1$) and MSE($\hat{f}_3$) varies per repetition. This is because X_2 and X_3 are very correlated, rendering the difference between $\hat{f}_1$ and $\hat{f}_2$ small.

Problem 3 (17 pts)

1. (2 pts)

Generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model $y = X\beta + \epsilon$, where β satisfies $\beta_1 = \beta_2 = \beta_3 = 2, \beta_4 = \beta_5 = 0.5$ and $\beta_6 = \cdots = \beta_{20} = 0$. The design matrix $X \in \mathbb{R}^{n\times p}$ has entries generated as i.i.d. realizations of $N(0, 1)$. The error $\epsilon \in \mathbb{R}^n$ also contains entries generated as i.i.d. realizations of $N(0, 1)$. (Note: for reproducibility, you need to specify \verb*|set.seed(0)| at the beginning before generating the data.)

```r
# clear memory
rm(list=ls())

p = 20
n = 1000
beta = c(c(2,2,2,0.5,0.5), rep(0,15))
set.seed(0)
X = matrix(rnorm(n*p),nrow = n,ncol = p)
xi = rnorm(n)
y = X %*% beta + xi
```

2. (1 pt)

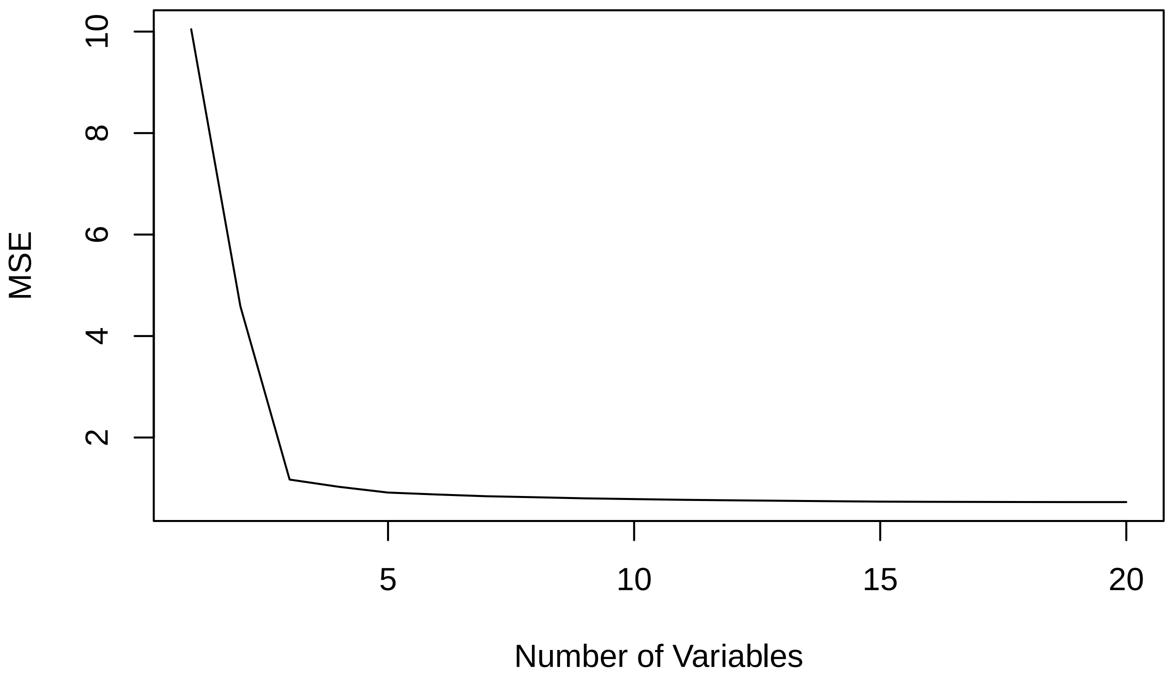
Randomly split your dataset into a training set containing 100 observations and a test set containing 900 observations.

```r
set.seed(0)
indx = sample(1:n,100)
y.train = y[indx]
x.train = X[indx,]
y.test = y[-indx]
x.test = X[-indx,]
```

3. (2 pts)

Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

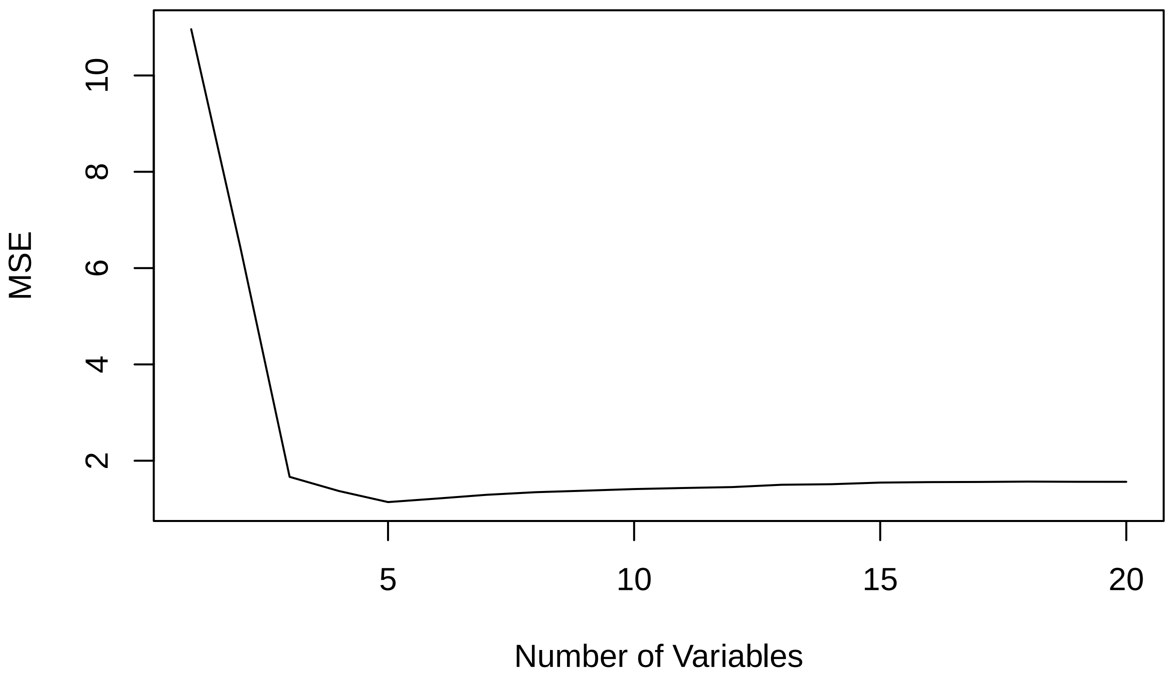
```r
library(leaps)
reg.fit = regsubsets(x = x.train,y = y.train,nvmax = 20)
reg.summary = summary(reg.fit)
plot(reg.summary$rss/length(y.train),xlab="Number of Variables",ylab="MSE",type="l")
```



4. (2 pts)

Plot the test set MSE associated with the best model of each size.

```r
x.test = cbind(rep(1,900),x.test)
colnames(x.test) = names(coef(reg.fit,id=20))

test.mse <- rep(NA,20)
for(i in 1:20){
  coefi <- coef(reg.fit,id=i)
  pred <- x.test[,names(coefi)]%*%coefi
  test.mse[i] <- mean((y.test-pred)^2)
}
plot(test.mse,xlab="Number of Variables",ylab="MSE",type="l")
```



5. (2 pts)

For which model size do the training set MSE and test set MSE take on their minimum value? Comment on your results.

```r
opt.train = which.min(reg.summary$rss)
opt.test = which.min(test.mse)
```

The training MSE has the minimal value when the model size is 20, while the test MSE has the minimal value when the model size is 5. The training error always decreases as the number of predictors increases. Choosing the model based on the training MSE leads to overfitting. The selected model based on the test MSE corresponds to the true model.

6. (2 pts)

How does the model at which the test set MSE is minimized compare to the true model used to generate the data? Comment on the coefficient values

```r
coef(reg.fit,id=opt.test)
```

```
##   (Intercept)            a            b            c            d            e
## -0.008975307  2.202528759  1.806976503  2.035352336  0.388887113  0.376186602
```

the model at which the test set MSE is minimized is

$$y = -0.009 + 2.20X_1 + 1.81X_2 + 2.04X_3 + 0.39X_4 + 0.38X_5$$

while the true model is

$$y = 2X_1 + 2X_2 + 2X_3 + 0.5X_4 + 0.5X_5$$

The model selected based on the test set MSE identifies the non-zero β. The coefficients between the estimated model and the true model are close but different. It's because that the estimated parameters in linear regression are random, depending on the data.

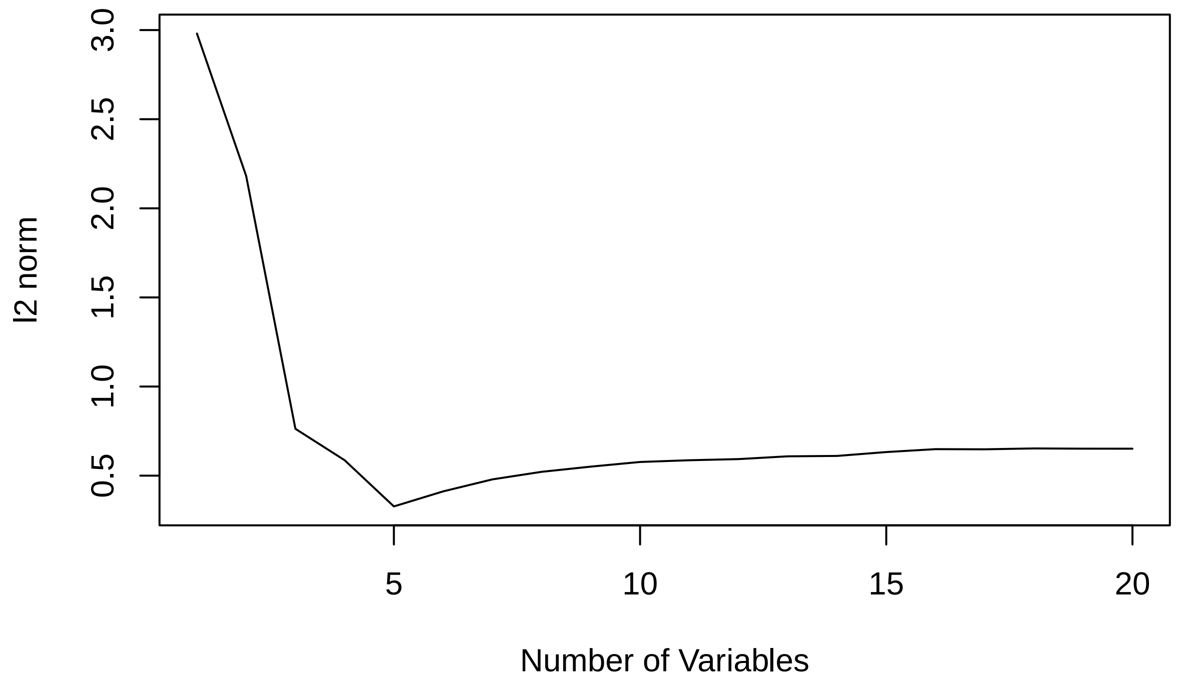
7. (2 pts)

Create a plot displaying

$$\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^{(k)})^2}$$

for a range of values of k, where $\hat{\beta}_j^{(k)}$ is the jth coefficient estimate for the best model containing k coefficients. Comment on what you observe. How does this compare to the test MSE plot from part 4?

```r
vals = rep(NA,20)
for(i in 1:20){
  coefi <- coef(reg.fit,id=i)
  betai = rep(0,21)
  names(betai) = colnames(x.test)
  betai[names(coefi)] = coefi
  vals[i] = sqrt(sum((beta - betai[-1])^2))
}
plot(vals,xlab="Number of Variables",ylab="l2 norm",type="l")
```

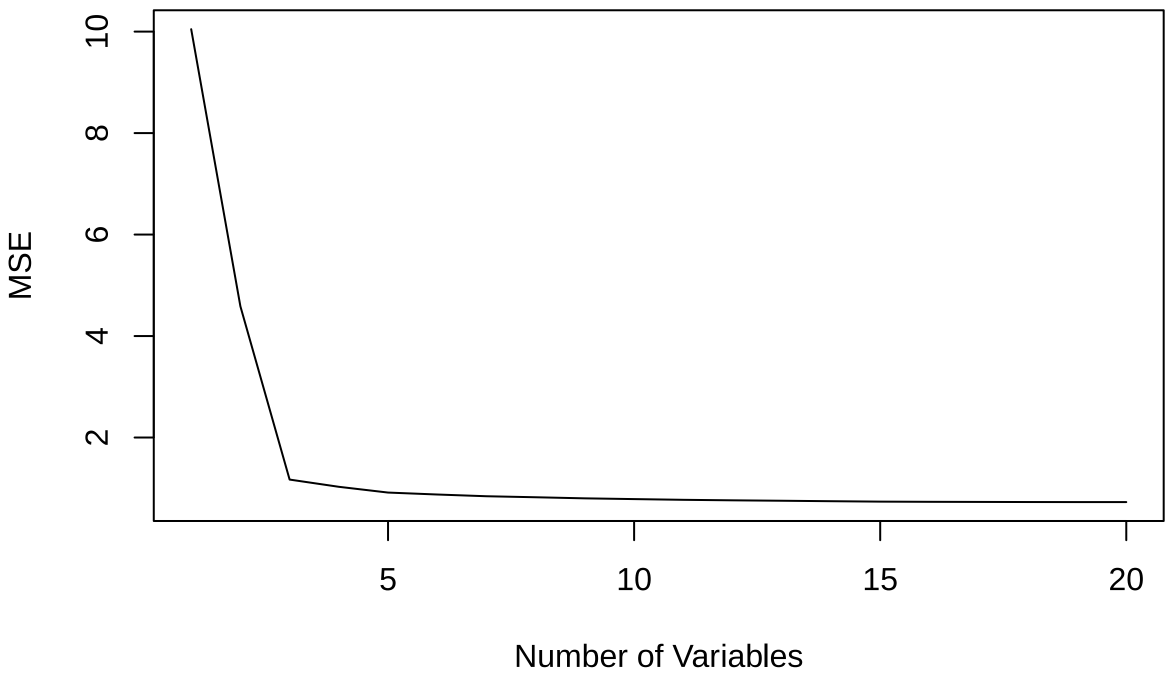


As the number of variables increases, the ℓ_2 norm of the estimation error first decreases, achieves its minimum when the model size is 5, and then increases. It has the same trend as the test MSE plot.

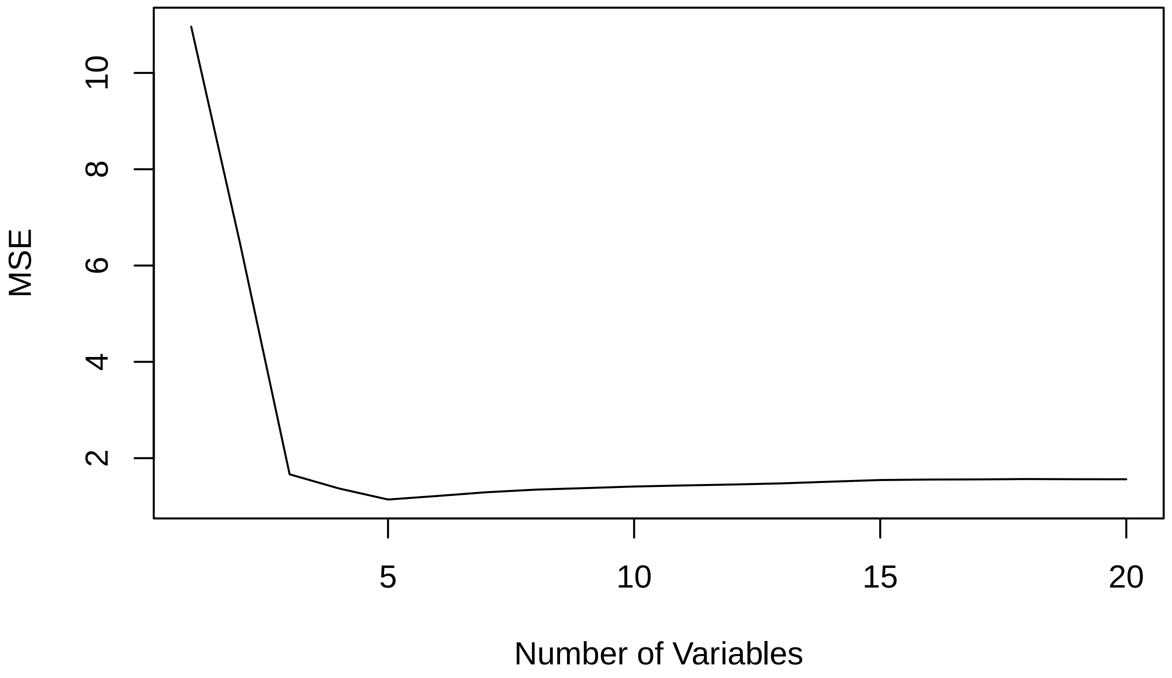
8. (2 pts)

Repeat steps 3 - 6 for forward stepwise selection.

```r
reg.fit.forward = regsubsets(x = x.train,y = y.train,nvmax = 20,method="forward")
reg.summary.forward = summary(reg.fit.forward)
plot(reg.summary.forward$rss/length(y.train),xlab="Number of Variables",ylab="MSE",type="l")
```



```r
test.mse <- rep(NA,20)
for(i in 1:20){
   coefi <- coef(reg.fit.forward,id=i)
   pred <- x.test[,names(coefi)]%*%coefi
   test.mse[i] <- mean((y.test-pred)^2)
}
plot(test.mse,xlab="Number of Variables",ylab="MSE",type="l")
```



```r
opt.train = which.min(reg.summary.forward$rss)
opt.test = which.min(test.mse)
```

We see the same result as the best subset selection. The training MSE has the minimal value when the model size is 20, while the test MSE has the minimal value when the model size is 5.

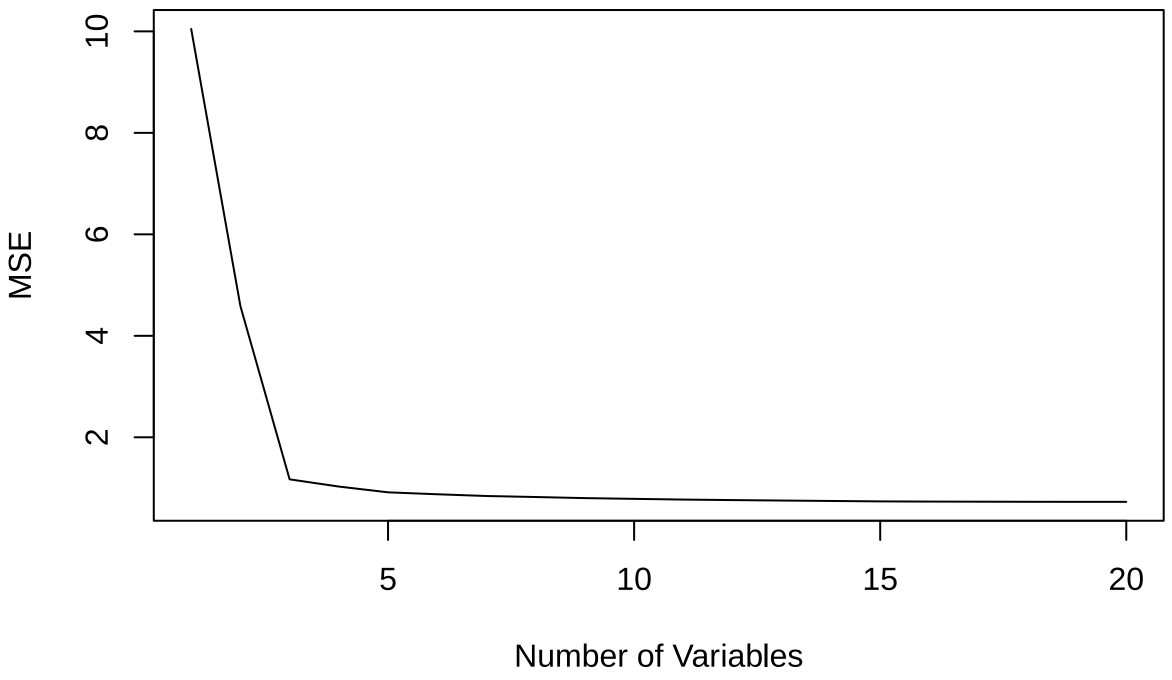
```r
coef(reg.fit.forward,id=opt.test)
```

```
##  (Intercept)            a            b            c            d            e
## -0.008975307  2.202528759  1.806976503  2.035352336  0.388887113  0.376186602
```

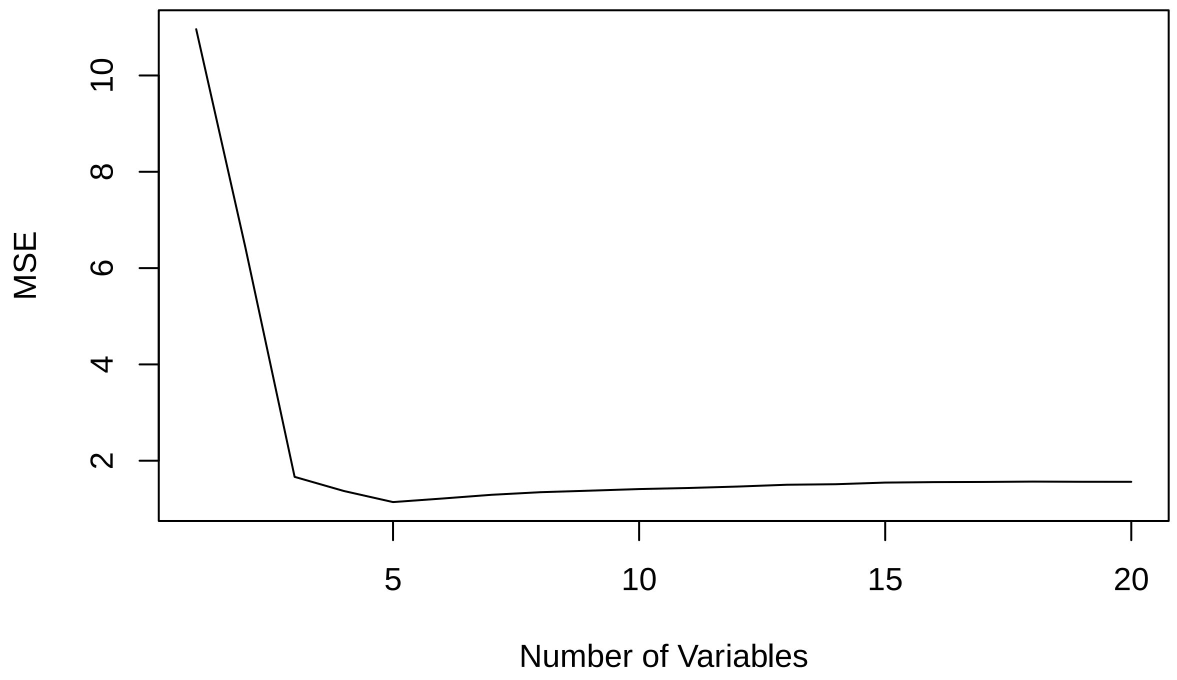
9. (2 pts)

Repeat steps 3 - 6 for backward stepwise selection. \end{enumerate}

```r
reg.fit.backward = regsubsets(x = x.train,y = y.train,nvmax = 20,method="backward")
reg.summary.backward = summary(reg.fit.backward)
plot(reg.summary.backward$rss/length(y.train),xlab="Number of Variables",ylab="MSE",type="l")
```



```r
test.mse <- rep(NA,20)
for(i in 1:20){
   coefi <- coef(reg.fit.backward,id=i)
   pred <- x.test[,names(coefi)]%*%coefi
   test.mse[i] <- mean((y.test-pred)^2)
}

plot(test.mse,xlab="Number of Variables",ylab="MSE",type="l")
```



```
opt.train = which.min(reg.summary.backward$rss)
opt.test = which.min(test.mse)
```

We see the same result as the best subset selection. The training MSE has the minimal value when the model size is 20, while the test MSE has the minimal value when the model size is 5.

```
coef(reg.fit.backward,id=opt.test)
```

```
##  (Intercept)            a            b            c            d            e
## -0.008975307  2.202528759  1.806976503  2.035352336  0.388887113  0.376186602
```

Problem 4 (12 pts)

Steps (a) – (d).

```r
# clear memory
rm(list=ls())
# set global variable

n <- 1100
p <- 50

set.seed(0)

# generate X and epsilon

X <- matrix(rnorm(n*p), nrow=n, ncol=p, byrow=TRUE)

epsilon <- rnorm(n)

# generate beta and fit Y
beta <- c(rep(2,5), rep(0,45))

Y <- X%*%beta + epsilon

# Train/Test split
train_index <-sample(c(1:nrow(Y)),100)
train_y <- Y[train_index]
train_x <- X[train_index,]

test_y <- Y[-train_index]
test_x <- X[-train_index,]

# set grid of lambda

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

Part 1. (3 pts)

Fit both the ridge regression and the lasso with λ selected by cross validation on the grid generated as above. Which method leads to a smaller test set MSE?

```r
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```r
# ridge regression

cv.ridge <- cv.glmnet(train_x,train_y,alpha=0,lambda=grid)
bestlam <- cv.ridge$lambda.min
ridge.mod <- glmnet(train_x,train_y,alpha=0,lambda=bestlam)
pred.ridge <- predict(ridge.mod,test_x, lambda = bestlam)
```

```r
ridge.mse <- mean((test_y - pred.ridge)^2)
paste('ridge regression MSE is ', ridge.mse)
```

```
## [1] "ridge regression MSE is  1.51215287321105"
```

```r
# lasso regression
cv.lasso <- cv.glmnet(train_x,train_y,alpha=1,lambda=grid)
bestlam <- cv.lasso$lambda.min
lasso.mod <- glmnet(train_x,train_y,alpha=1,lambda=bestlam)
pred.lasso <- predict(lasso.mod,test_x, lambda = bestlam)

lasso.mse <- mean((test_y - pred.lasso)^2)
paste('lasso regression MSE is ', lasso.mse)
```

```
## [1] "lasso regression MSE is  1.11141570184101"
```

We can see that the lasso gives a smaller test MSE here.

Part 2. (3 pts)

Repeat steps (a){(d) for generating the data by using different seeds

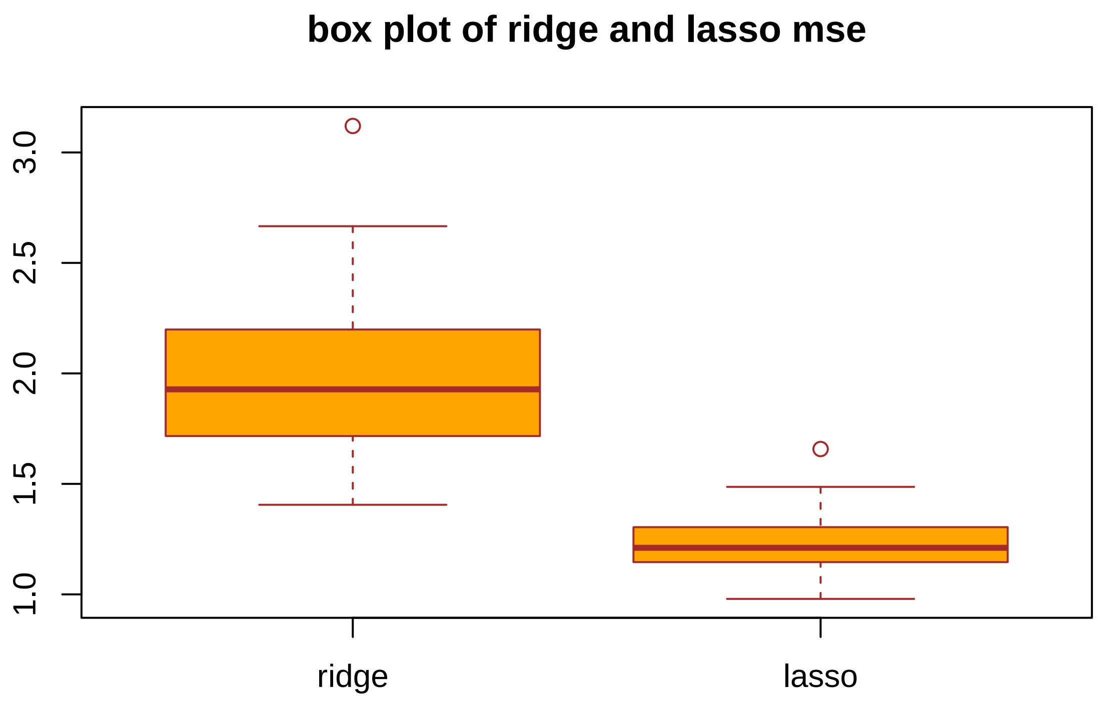
```
set.seed(2),...,set.seed(50)
```

and also repeat part 1 for each seed. Save the test error for both, lasso and ridge for all seeds. Together with the results from part 1, this should give you 50 test MSEs for ridge and lasso.

```r
# we can convert all above to be a function of seed numbers only:

mse_gen <- function(seed_num){
  set.seed(seed_num)
  X <- matrix(rnorm(n*p), nrow=n, ncol=p, byrow=TRUE)
  epsilon <- rnorm(n)
  beta <- c(rep(2,5), rep(0,45))
  Y <- X%*%beta + epsilon

  train_index <-sample(c(1:nrow(Y)),100)
  train_y <- Y[train_index]
  train_x <- X[train_index,]

  test_y <- Y[-train_index]
  test_x <- X[-train_index,]

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

  #ridge regression
  cv.ridge <- cv.glmnet(train_x,train_y,alpha=0,lambda=grid)
  bestlam <- cv.ridge$lambda.min
  ridge.mod <- glmnet(train_x,train_y,alpha=0,lambda=bestlam)
  pred.ridge <- predict(ridge.mod,test_x, lambda = bestlam)

  ridge.mse <- mean((test_y - pred.ridge)^2)

  # lasso regression
  cv.lasso <- cv.glmnet(train_x,train_y,alpha=1,lambda=grid)
  bestlam <- cv.lasso$lambda.min
  lasso.mod <- glmnet(train_x,train_y,alpha=1,lambda=bestlam)
  pred.lasso <- predict(lasso.mod,test_x, lambda = bestlam)

  lasso.mse <- mean((test_y - pred.lasso)^2)

  return(c(ridge.mse, lasso.mse))
}
```

```
# run for seed number 2 to 50

MSE_ridge <- c()
MSE_lasso <- c()

for (i in 2:50){
  m <- mse_gen(i)
  MSE_ridge <- c(MSE_ridge,m[1])
  MSE_lasso <- c(MSE_lasso,m[2])
}
```

```
# concatenate with what we have from q1

MSE_ridge <- c(ridge.mse,MSE_ridge)
MSE_lasso <- c(lasso.mse,MSE_lasso)

# now make the box_plot

boxplot(MSE_ridge,MSE_lasso,
        names = c('ridge','lasso'),
        main = 'box plot of ridge and lasso mse',
        y_lab = 'mse',
        col="orange",
        border="brown")
```



- On average, the lasso has smaller test MSE than the ridge. The variance of test MSEs of the lasso is also smaller than that of the ridge regression. In this case, the Lasso visibly outperforms the ridge.
- This is expected as the true model is sparse, that is, the true coefficients have zeroes.

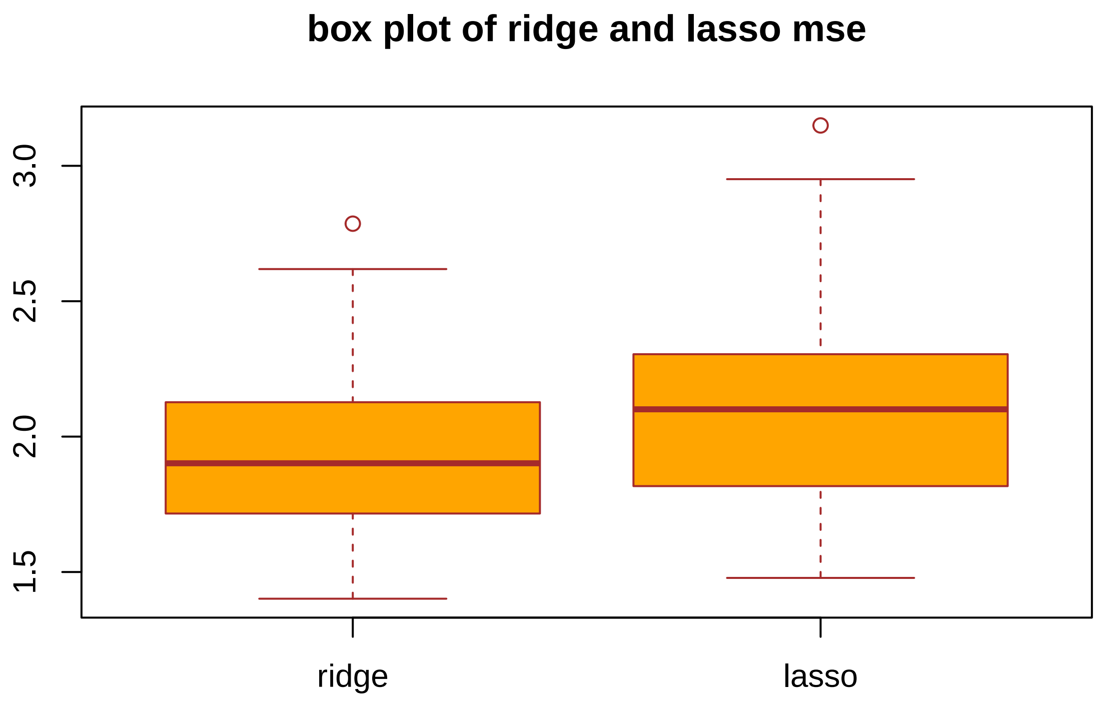
Part 3. (6 pts)

Redo parts 1 and 2 by using $\beta_j = 0.5$, for all $j = 1, \ldots, 50$, in step (b).

```r
# again redo the function:

mse_gen_new <- function(seed_num){
  set.seed(seed_num)
  X <- matrix(rnorm(n*p), nrow=n, ncol=p, byrow=TRUE)
  epsilon <- rnorm(n)
  beta <- rep(0.5,50)
  Y <- X%*%beta + epsilon

  train_index <-sample(c(1:nrow(Y)),100)
  train_y <- Y[train_index]
  train_x <- X[train_index,]

  test_y <- Y[-train_index]
  test_x <- X[-train_index,]

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

  #ridge regression
  cv.ridge <- cv.glmnet(train_x,train_y,alpha=0,lambda=grid)
  bestlam <- cv.ridge$lambda.min
  ridge.mod <- glmnet(train_x,train_y,alpha=0,lambda=bestlam)
  pred.ridge <- predict(ridge.mod,test_x, lambda = bestlam)

  ridge.mse <- mean((test_y - pred.ridge)^2)

  # lasso regression
  cv.lasso <- cv.glmnet(train_x,train_y,alpha=1,lambda=grid)
  bestlam <- cv.lasso$lambda.min
  lasso.mod <- glmnet(train_x,train_y,alpha=1,lambda=bestlam)
  pred.lasso <- predict(lasso.mod,test_x, lambda = bestlam)

  lasso.mse <- mean((test_y - pred.lasso)^2)

  return(c(ridge.mse, lasso.mse))
}
```

```r
#and fit the model:
#run for seed number 0 to 50

MSE_ridge_new <- c()
MSE_lasso_new <- c()

for (i in 0:50){
  m <- mse_gen_new(i)
  MSE_ridge_new <- c(MSE_ridge_new,m[1])
  MSE_lasso_new <- c(MSE_lasso_new,m[2])
}

# leave set.seed(1) out
MSE_ridge_new <- MSE_ridge_new[-2]
MSE_lasso_new <- MSE_lasso_new[-2]
```

```
# and now make the box_plot

boxplot(MSE_ridge_new,MSE_lasso_new,
        names = c('ridge','lasso'),
        main = 'box plot of ridge and lasso mse',
        y_lab = 'mse',
        col="orange",
        border="brown")
```



- The ridge regression, on average, has slightly smaller test MSEs than the Lasso. But their variances of test MSEs are nearly the same. The advantage of the ridge over the lasso in this case seems not substantial.
- This is also expected as the true coefficients have some small but non-zero entries, whence the ridge should perform (slightly) better.