Comparing Best Subset, Forward and Backward Selection

Alexandre H.

Part 1: Load libraries and set seed

```r
library("leaps")
```

```
## Warning: package 'leaps' was built under R version 4.3.2
```

```r
library("ggplot2")
set.seed(0)

p <- 20
n <- 1000
# desing matrix X
X <- matrix(rnorm(n*p, 0, 1), nrow=n, ncol=p)
beta <- c(2, 2, 2, 0.5, 0.5, rep(0, p-5))
epsilon <- rnorm(n, 0, 1)
y <- X %*% beta + epsilon
```

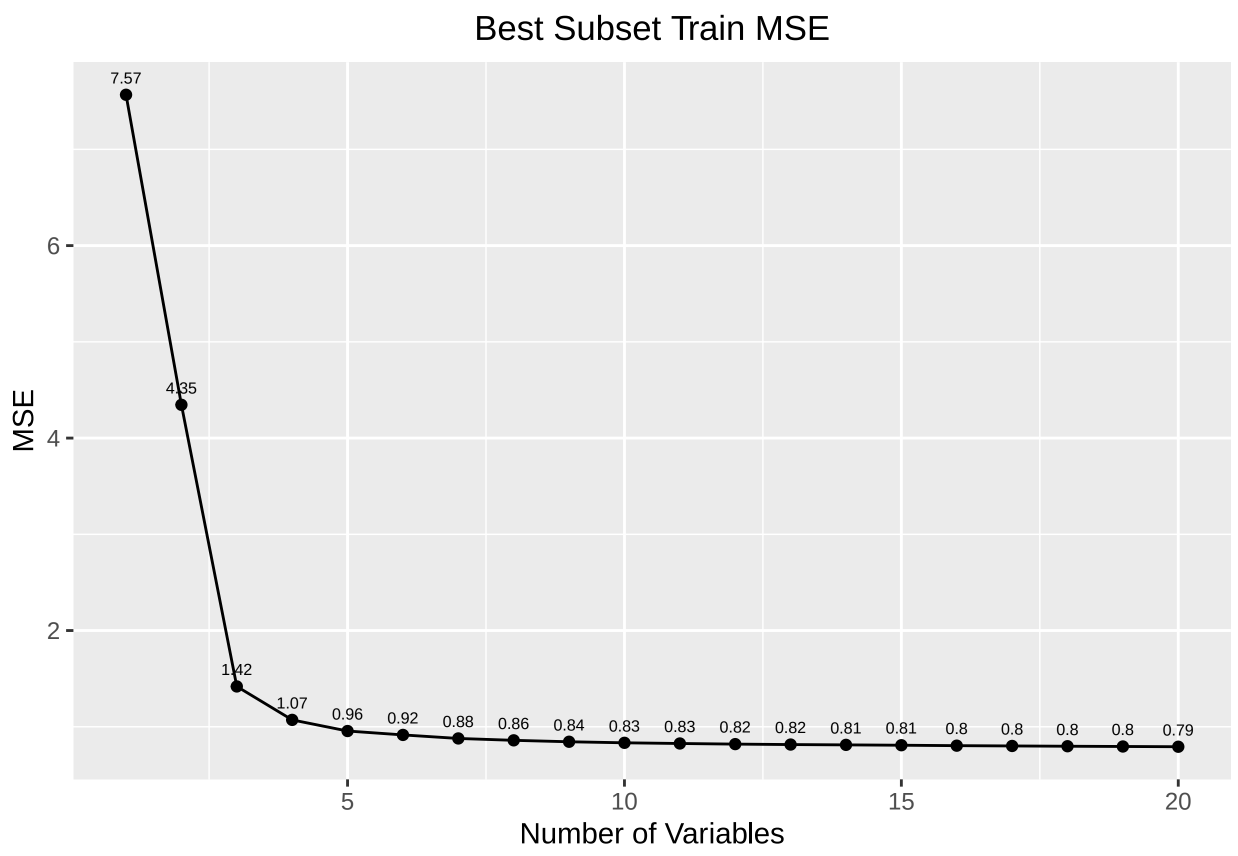
Part 2: Split data into training and test sets

```r
df <- data.frame(y=y, X=X)
train_id <- sample(seq_len(nrow(df)), 100)
df_train <- df[train_id,]
df_test <- df[-train_id,]
```

Part 3: Best Subset Selection

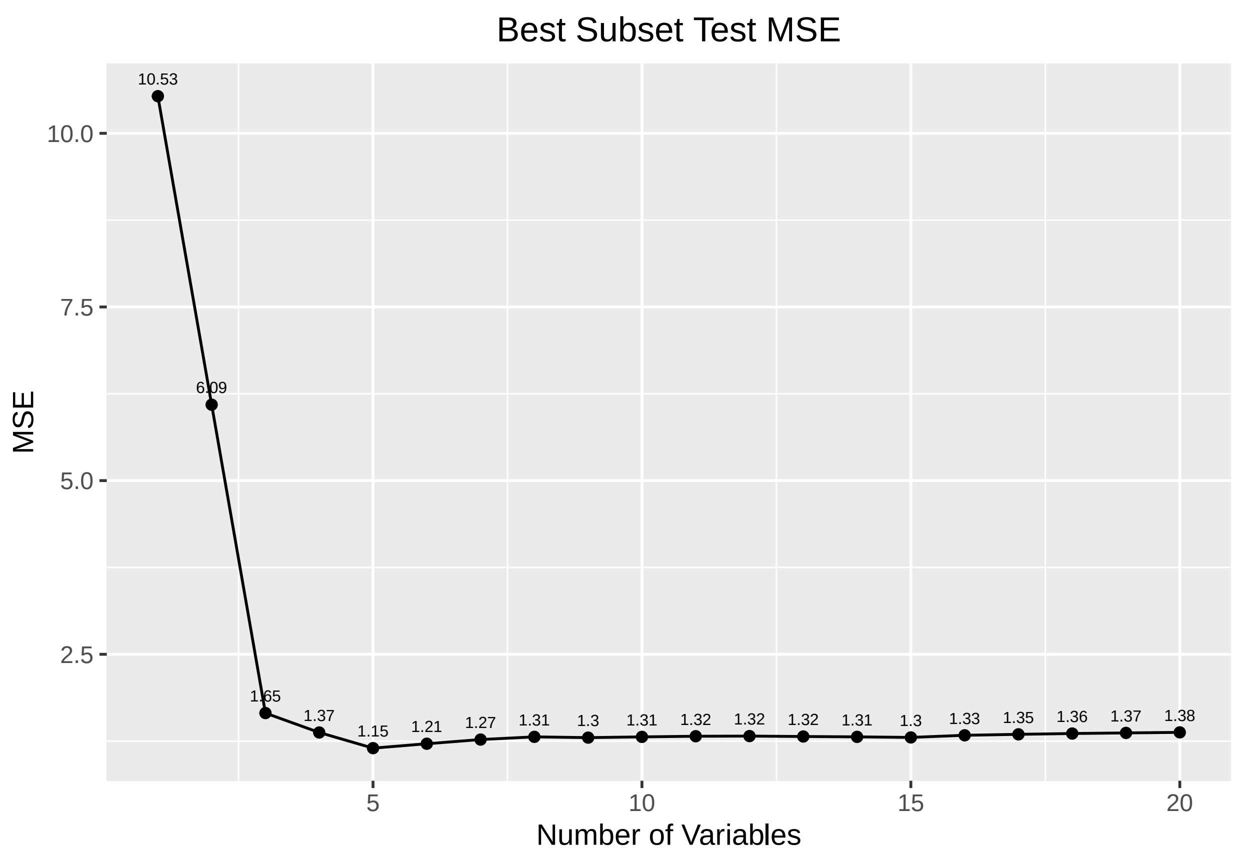
```r
regfit.full <- regsubsets(y~., df_train, nvmax = p, method="exhaustive")

train_mse <- rep(NA, p)
for (i in 1:p) {
   coefi <- coef(regfit.full,id=i)
   train_pred <- model.matrix(y~.,data=df_train)[, names(coefi)] %*% coefi
   train_mse[i] <- mean((df_train$y - train_pred)^2)
}
```

```r
ggplot(data.frame(train_mse, x=1:p), aes(x=x, y=train_mse)) +
  geom_line() +
  ggtitle("Best Subset Train MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(train_mse, 2)), vjust=-1, size=2)
```



Part 4: Test MSE

```r
test_mse <- rep(NA, p)
for (i in 1:p) {
   coefi <- coef(regfit.full,id=i)
   test_pred <- model.matrix(y~.,data=df_test)[, names(coefi)] %*% coefi
   test_mse[i] <- mean((df_test$y - test_pred)^2)
}

ggplot(data.frame(test_mse, x=1:p), aes(x=x, y=test_mse)) +
  geom_line() +
  ggtitle("Best Subset Test MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(test_mse, 2)), vjust=-1, size=2)
```



Part 5: Minimizer

```
which.min(train_mse)
```

```
## [1] 20
```

```
which.min(test_mse)
```

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

The training MSE takes on its minimum value for a model with a size that has the maximum number of variables being 20. The test MSE takes on its minimum value with a model size that has 5 variables.

These results are in line with theory. The training MSE will always decrease as the number of variables increases. However, it is not always the case for the test error. We can clearly make sense of why only variable X1,. . . ,X5 are included. They are the only ones which have a relationship with y. We defined the other variables X6,. . . ,X20 to have no relationship with y by setting beta to zero. Indeed, the test MSE nicely captures that.

Part 6: Coefficients

```
coef(regfit.full,id=which.min(train_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5 
##  0.28990753  2.05423046  1.99413286  2.09437466  0.72569091  0.28749549 
##         X.6         X.7         X.8         X.9        X.10        X.11 
##  0.12376248  0.06853343  0.13864798 -0.08691245  0.10623419  0.05562583 
##        X.12        X.13        X.14        X.15        X.16        X.17 
## -0.07658784 -0.11245578 -0.09466100 -0.15193611  0.07098179 -0.24179777 
##        X.18        X.19        X.20 
##  0.06361559  0.11499179  0.12959617
```

```
coef(regfit.full,id=which.min(test_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5 
##   0.3013784   2.0218917   2.0285077   2.0873708   0.6831486   0.3341665
```

The coefficient values of X1,. . . ,X5 for the model with 5 variables vs the model with 20 variables resemble each other. Both models seem to have values close to 2 for X1,. . . X3 and 0.5 for X4,X5. Although, we do not exactly have these values due to the variance in X, and the added random noise. This is in line with how we have defined y.

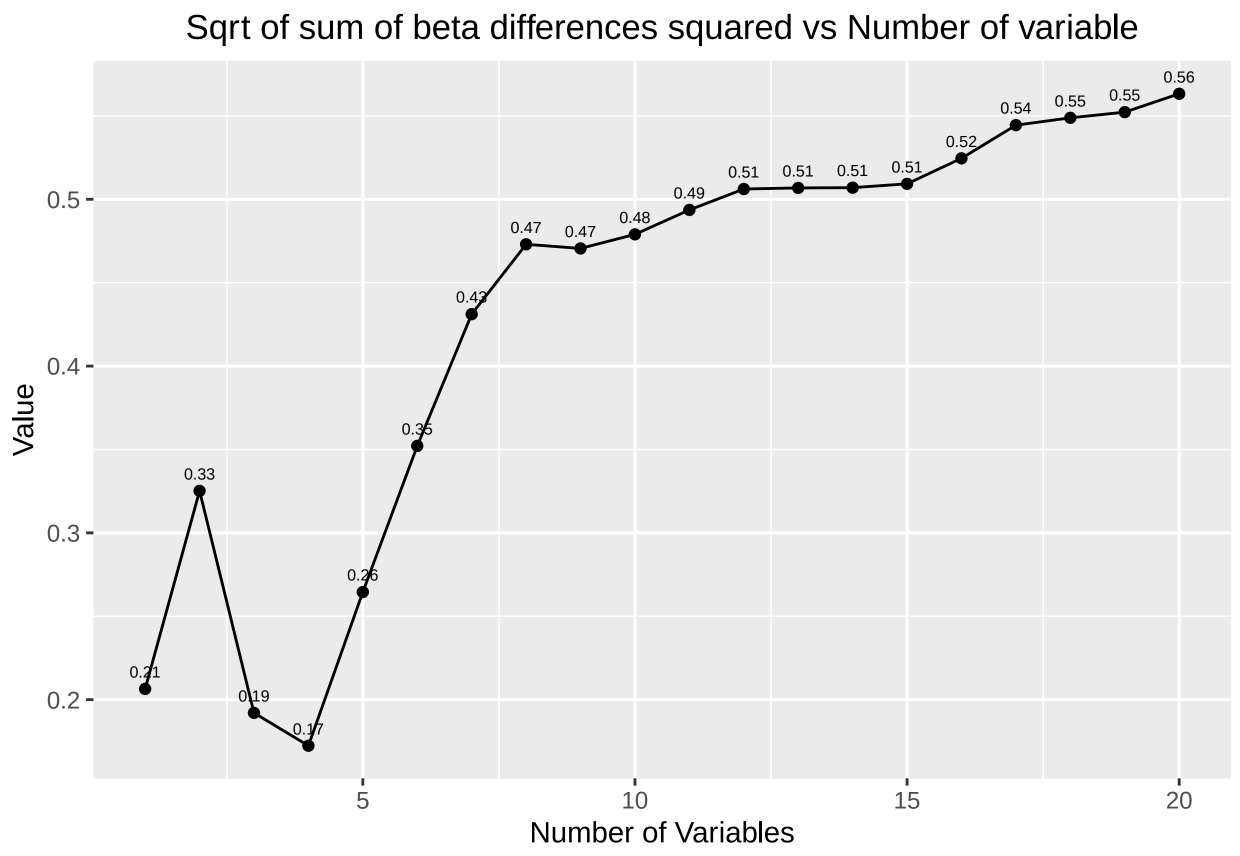
As for X6 to X20 in the model with 20 variables, these variables together vary around 0. Taking the average of all estimated coefficients beta 6 to 20, we find that this average is close to zero: 0.0071759. Each not being exactly zero is due to the variance in X, and the added random noise.

Part 7: Beta differences

```
for (i in 1:p){
   coef_names <- names(coef(regfit.full,id=i))
   beta[as.integer(sub("^X\\.", "", coef_names[2:length(coef_names)]))]
}

beta_diffs <- rep(NA, p)

for (i in 1:p){
   coefi <- coef(regfit.full,id=i)
   coef_names <- names(coef(regfit.full,id=i))
   coef_names <- coef_names[2:length(coef_names)]
   beta_diffs[i] <- sqrt(sum(( beta[as.integer(sub("^X\\.", "", coef_names))] - coefi[coef_names] )^2))
}

beta_diffs
```

```
##  [1] 0.2064585 0.3251780 0.1920717 0.1723701 0.2645180 0.3521221 0.4311250
##  [8] 0.4730100 0.4705271 0.4789780 0.4936149 0.5061668 0.5068153 0.5069707
## [15] 0.5092734 0.5245607 0.5444532 0.5488462 0.5522562 0.5632713
```

```r
ggplot(data.frame(beta_diffs, x=1:p), aes(x=x, y=beta_diffs)) +
  geom_line() +
  ggtitle("Sqrt of sum of beta differences squared vs Number of variable") +
  xlab("Number of Variables") +
  ylab("Value") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(beta_diffs, 2)), vjust=-1, size=2)
```

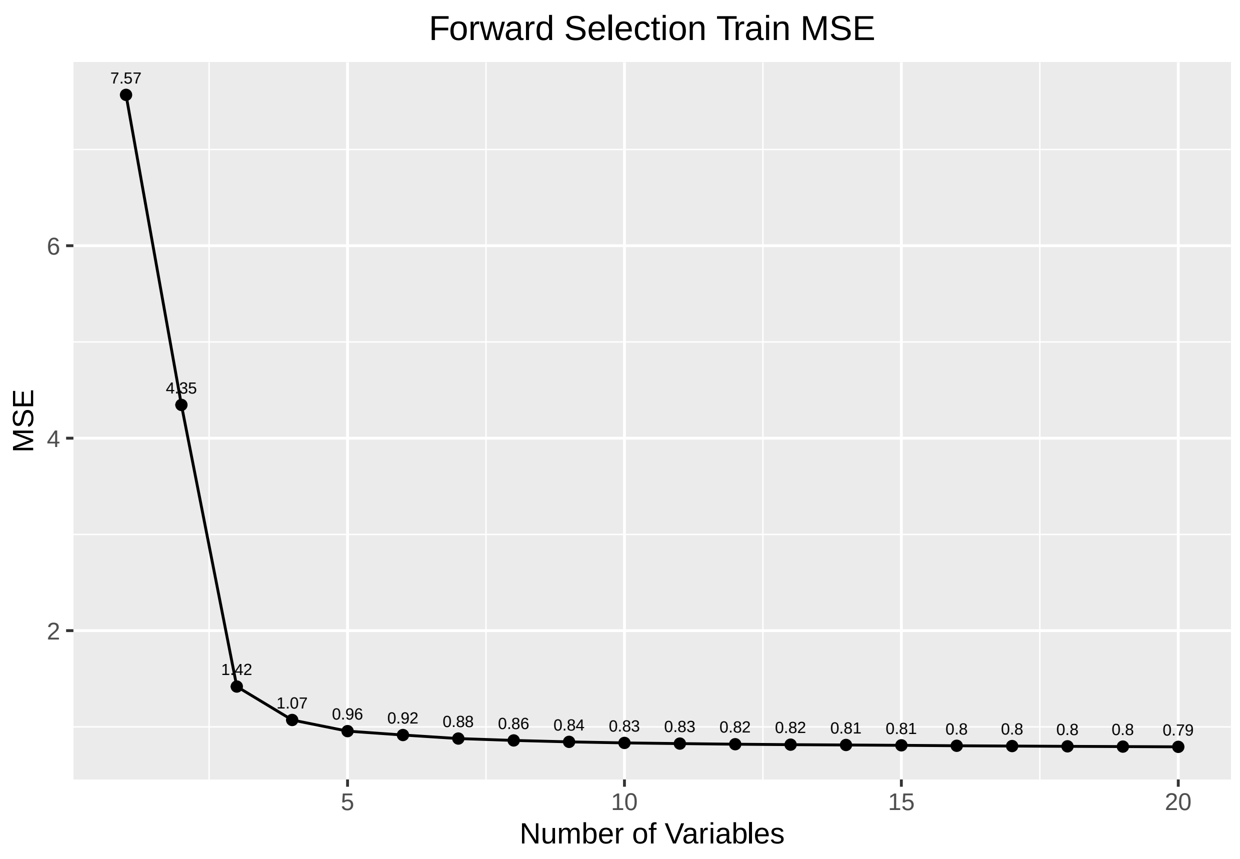


The plot shows the square root of the sum of the beta differences squared vs Number of variables. We observe that from the best subsets containing 4 to 20 variables, the value almost only increases. If the estimated betas for variables shared across models with different number of variables were the same, we can expect this graph to only increase due to the cumulative nature of the function. However, there are cases where the value decreases as p increases. This is due to the fact that the estimated beta for a variable in a model with p variables is closer to the true beta than the estimated beta for the same variable in a model with p-1 variables. This can be due to the amount of random error that varies across models. When comparing this plot to the test MSE in part 4), we can see the values at which the minimum occurs are close. It is 4 for this plot, and 5 for the test MSE. This makes sense as the differences being calculated are not the same. The MSE difference is between the true beta and the estimated beta, and this difference is between y and estimated y. But, the closeness of the minimizer is not surprising as the estimated y is dependent on the estimated betas.

Part 8: Forward Selection

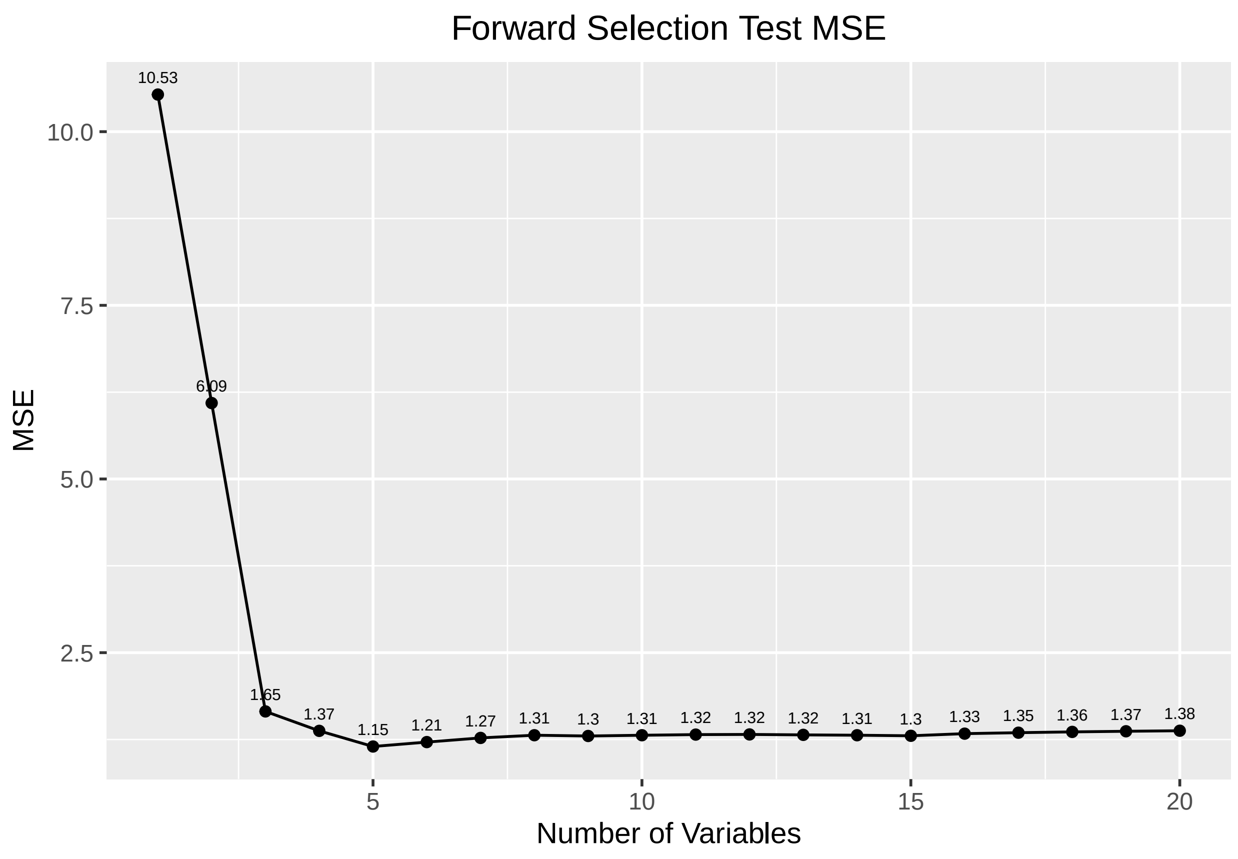
Part 8.3: Train MSE

```r
regfit.forward <- regsubsets(y~., df_train, nvmax = p, method="forward")

train_mse <- rep(NA, p)
for (i in 1:p) {
   coefi <- coef(regfit.forward,id=i)
   train_pred <- model.matrix(y~.,data=df_train)[, names(coefi)] %*% coefi
   train_mse[i] <- mean((df_train$y - train_pred)^2)
}

ggplot(data.frame(train_mse, x=1:p), aes(x=x, y=train_mse)) +
  geom_line() +
  ggtitle("Forward Selection Train MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(train_mse, 2)), vjust=-1, size=2)
```



Part 8.4: Test MSE

```r
test_mse <- rep(NA, p)

for (i in 1:p) {
   coefi <- coef(regfit.forward,id=i)
   test_pred <- model.matrix(y~.,data=df_test)[, names(coefi)] %*% coefi
   test_mse[i] <- mean((df_test$y - test_pred)^2)
}

ggplot(data.frame(test_mse, x=1:p), aes(x=x, y=test_mse)) +
  geom_line() +
  ggtitle("Forward Selection Test MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(test_mse, 2)), vjust=-1, size=2)
```



Part 8.5: Minimizer

```r
which.min(train_mse)
```

```
## [1] 20
```

```r
which.min(test_mse)
```

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

The training MSE takes on its minimum value for a model with a size that has the maximum number of variables being 20. The test MSE takes on its minimum value with a model size that has 5 variables.

These results are in line with theory. The training MSE will always decrease as the number of variables increases. However, it is not always the case for the test error. We can clearly make sense of why only variable X1,...,X5 are included. They are the only ones which have a relationship with y. We defined the other variables X6,...,X20 to have no relationship with y by setting beta to zero. Indeed, the test MSE nicely captures that.

Part 8.6: Coefficients

```r
coef(regfit.forward,id=which.min(train_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5 
##  0.28990753  2.05423046  1.99413286  2.09437466  0.72569091  0.28749549 
##         X.6         X.7         X.8         X.9        X.10        X.11 
##  0.12376248  0.06853343  0.13864798 -0.08691245  0.10623419  0.05562583 
##        X.12        X.13        X.14        X.15        X.16        X.17 
## -0.07658784 -0.11245578 -0.09466100 -0.15193611  0.07098179 -0.24179777 
##        X.18        X.19        X.20 
##  0.06361559  0.11499179  0.12959617
```

```r
coef(regfit.forward,id=which.min(test_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5 
##   0.3013784   2.0218917   2.0285077   2.0873708   0.6831486   0.3341665
```

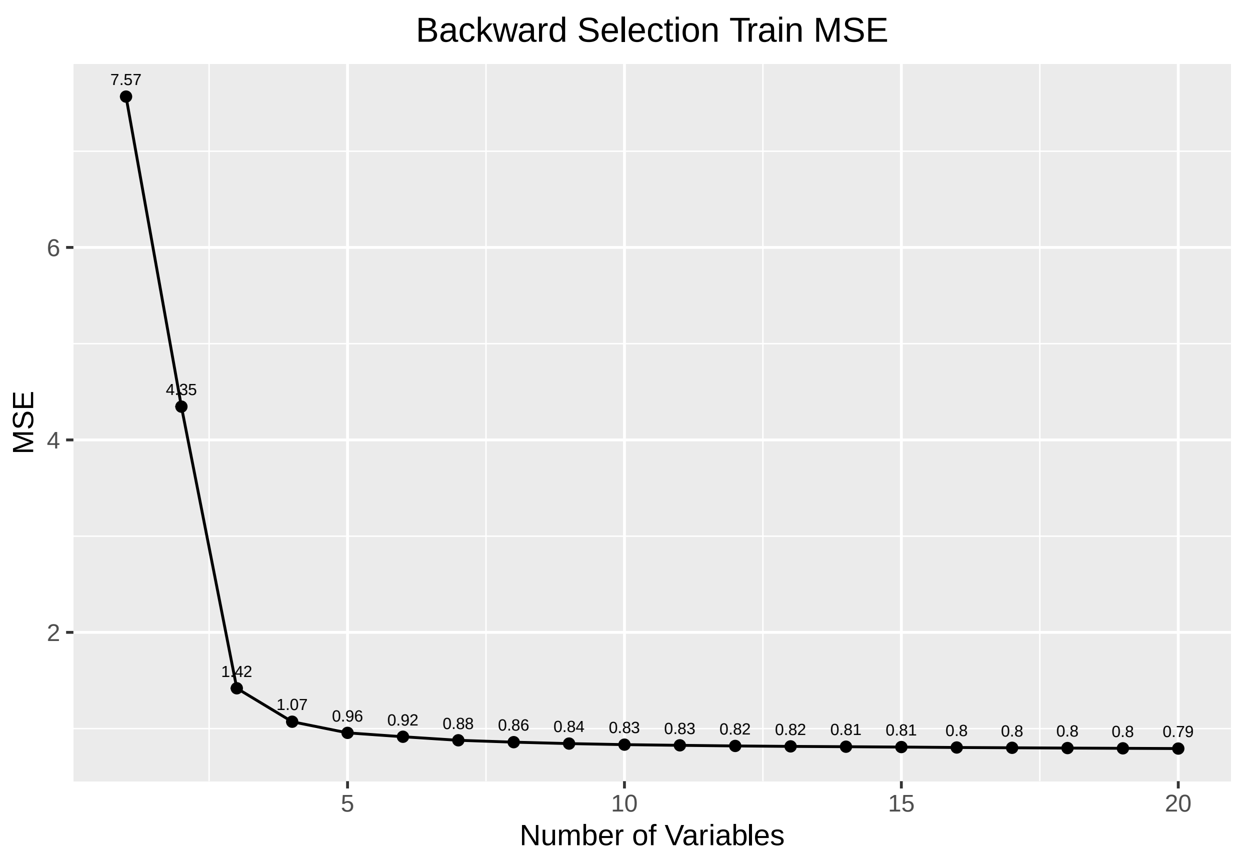
The coefficient values of X1,...,X5 for the model with 5 variables vs the model with 20 variables resemble each other. Both models seem to have values close to 2 for X1,...X3 and 0.5 for X4,X5. Although, we do not exactly have these values due to the variance in X, and the added random noise. This is in line with how we have defined y.

As for X6 to X20 in the model with 20 variables, these variables together vary around 0. Taking the average of all estimated coefficients beta 6 to 20, we find that this average is close to zero: 0.0071759. Each not being exactly zero is due to the variance in X, and the added random noise.

Part 9: Backward Selection

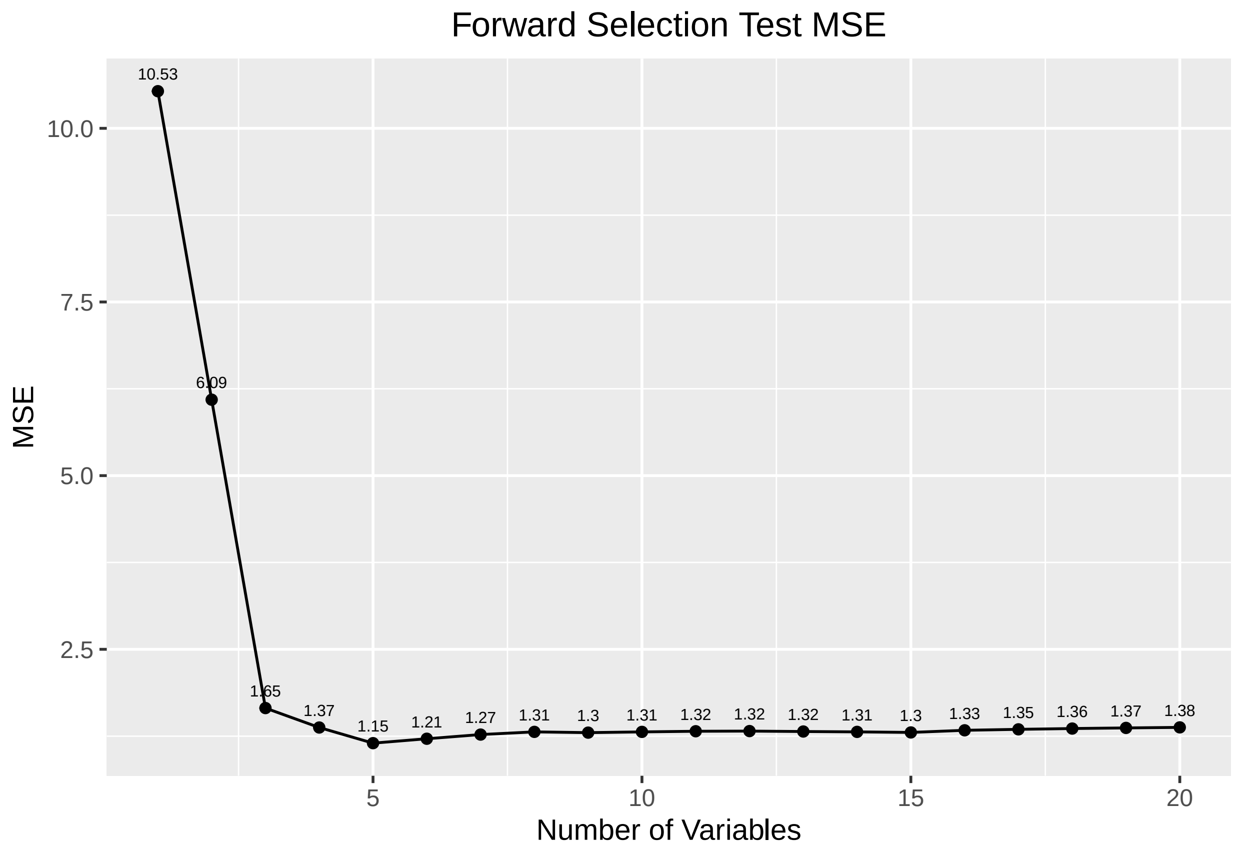
Part 9.3: Train MSE

```
regfit.backward <- regsubsets(y~., df_train, nvmax = p, method="backward")

train_mse <- rep(NA, p)
for (i in 1:p) {
   coefi <- coef(regfit.backward,id=i)
   train_pred <- model.matrix(y~.,data=df_train)[, names(coefi)] %*% coefi
   train_mse[i] <- mean((df_train$y - train_pred)^2)
}

ggplot(data.frame(train_mse, x=1:p), aes(x=x, y=train_mse)) +
  geom_line() +
  ggtitle("Backward Selection Train MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(train_mse, 2)), vjust=-1, size=2)
```



Part 9.4: Test MSE

```
test_mse <- rep(NA, p)

for (i in 1:p) {
    coefi <- coef(regfit.backward,id=i)
    test_pred <- model.matrix(y~.,data=df_test)[, names(coefi)] %*% coefi
    test_mse[i] <- mean((df_test$y - test_pred)^2)
}

ggplot(data.frame(test_mse, x=1:p), aes(x=x, y=test_mse)) +
  geom_line() +
  ggtitle("Forward Selection Test MSE") +
  xlab("Number of Variables") +
  ylab("MSE") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_text(aes(label=round(test_mse, 2)), vjust=-1, size=2)
```



Part 9.5: Minimizer

```
which.min(train_mse)
```

```
## [1] 20
```

```
which.min(test_mse)
```

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

The training MSE takes on its minimum value for a model with a size that has the maximum number of variables being 20. The test MSE takes on its minimum value with a model size that has 5 variables.

These results are in line with theory. The training MSE will always decrease as the number of variables increases. However, it is not always the case for the test error. We can clearly make sense of why only variable X1,. . . ,X5 are included. They are the only ones which have a relationship with y. We defined the other variables X6,. . . ,X20 to have no relationship with y by setting beta to zero. Indeed, the test MSE nicely captures that.

Part 9.6: Coefficients

```r
coef(regfit.backward,id=which.min(train_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5
##  0.28990753  2.05423046  1.99413286  2.09437466  0.72569091  0.28749549
##         X.6         X.7         X.8         X.9        X.10        X.11
##  0.12376248  0.06853343  0.13864798 -0.08691245  0.10623419  0.05562583
##        X.12        X.13        X.14        X.15        X.16        X.17
## -0.07658784 -0.11245578 -0.09466100 -0.15193611  0.07098179 -0.24179777
##        X.18        X.19        X.20
##  0.06361559  0.11499179  0.12959617
```

```r
coef(regfit.backward,id=which.min(test_mse))
```

```
## (Intercept)         X.1         X.2         X.3         X.4         X.5
##   0.3013784   2.0218917   2.0285077   2.0873708   0.6831486   0.3341665
```

The coefficient values of X1,. . . ,X5 for the model with 5 variables vs the model with 20 variables resemble each other. Both models seem to have values close to 2 for X1,. . . X3 and 0.5 for X4,X5. Although, we do not exactly have these values due to the variance in X, and the added random noise. This is in line with how we have defined y.

As for X6 to X20 in the model with 20 variables, these variables together vary around 0. Taking the average of all estimated coefficients beta 6 to 20, we find that this average is close to zero: 0.0071759. Each not being exactly zero is due to the variance in X, and the added random noise.