SDM_Assignment3_1

Sri Balaji Muruganandam

22/10/2021

Setting Working Directory

```
rm(list = ls())
setwd("G:\\SDM_Sem01\\Assignment3")
```

Importing necessary libraries

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.1.1
```

1. We have seen that as the number of features used in a model increase, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

(a) Generate a data set with p = 20 features, n = 1,000 observations, and an associated quantitative response vector generated according to the model: Y = Xβ + ϵ, where β has some elements that are exactly equal to zero.

Generating a random point of 20000 and converting it into a data set of 1000 * 20

```
set.seed(21)
#data_points = 0.001 * rnorm(20000)
#data_points = 0.1 * rnorm(20000)
data_points = rnorm(20000)

X_data = matrix(data_points, 1000, 20)
dim(X_data)
```

```
## [1] 1000   20
```

Setting names for the column

```
colnames(X_data) = paste('C', 1:20, sep="")
head(X_data,2)
```

```
##             C1          C2         C3         C4         C5         C6
## [1,] 0.7930132  0.09516123  0.2817064 -2.0142477 0.28560785 -1.3511055
## [2,] 0.5222513 -0.03745645 -0.6907325 -0.3251423 0.09009744 -0.8141718
##              C7        C8        C9        C10        C11        C12       C13
## [1,] -0.7677244 0.1297478 0.9159966 -0.1897123  0.6785635 -0.2289234  1.026472
## [2,]  1.2013020 0.2615262 0.1905610 -1.9731269 -0.8519017 -0.9168313 -1.665080
##             C14        C15        C16        C17         C18       C19
## [1,] -0.4402043  0.3588432  0.5347137 -0.8276196  0.05579728  2.177254
## [2,] -1.5515003 -0.5922448 -1.0281673 -0.9216680 -0.33865202 -1.105036
##             C20
## [1,] -0.2624241
## [2,]  1.5296461
```

Generating coefficient values(Beta) and setting a few of the values to zero to make certain predictors as irrelavant to the model

```
coeff = runif(20)
coeff[c(5,8,11,16,18)] = 0
print(coeff)
```

```
##  [1] 0.39454815 0.30628884 0.01713109 0.07614966 0.00000000 0.17411393
##  [7] 0.43311309 0.00000000 0.04602152 0.27767837 0.00000000 0.22062573
## [13] 0.95674004 0.96267233 0.04108773 0.00000000 0.58364469 0.00000000
## [19] 0.69284816 0.18600950
```

Generating Error

```
error = 0.00001 * rnorm(1000)
```

Using Matrix multiplication to get the values of Y

X - 1000 * 20 & coeff - 20 * 1

So we get 1000 * 1 values for the Y column

```
Y_data = (X_data %*% coeff) + error
print(Y_data[1:5])
```

```
## [1]  1.1143944 -4.3348773 -1.7024174  1.6636357  0.4122339
```

Combining Y_data as 21st column of the original data

```
X_data = cbind(X_data, Y_data)
colnames(X_data)[21] = "C21"
head(X_data, 2)
```

```
##             C1          C2         C3         C4         C5         C6
## [1,] 0.7930132  0.09516123  0.2817064 -2.0142477 0.28560785 -1.3511055
## [2,] 0.5222513 -0.03745645 -0.6907325 -0.3251423 0.09009744 -0.8141718
##              C7        C8        C9        C10        C11        C12       C13
## [1,] -0.7677244 0.1297478 0.9159966 -0.1897123  0.6785635 -0.2289234  1.026472
## [2,]  1.2013020 0.2615262 0.1905610 -1.9731269 -0.8519017 -0.9168313 -1.665080
##             C14        C15        C16        C17         C18       C19
## [1,] -0.4402043  0.3588432  0.5347137 -0.8276196  0.05579728  2.177254
## [2,] -1.5515003 -0.5922448 -1.0281673 -0.9216680 -0.33865202 -1.105036
##             C20       C21
## [1,] -0.2624241  1.114394
## [2,]  1.5296461 -4.334877
```

(b) Split your data set into a training set containing 900 observations and a test set containing 100 observations.

```
set.seed(23)
random_index = sample(c(1:1000), size = round(9/10 * 1000), replace = FALSE)
train_data <- X_data[random_index,]
test_data <- X_data[-random_index,]
train_data = data.frame(train_data)
test_data = data.frame(test_data)
X_data = data.frame(X_data)

y_train_data <- train_data$C21
y_test_data <- test_data$C21
dim(train_data)
```

```
## [1] 900  21
```

```
dim(test_data)
```

```
## [1] 100  21
```

(c) Perform subset selection (best, forward or backwards) on the training set, and plot the training set MSE associated with the best model of each size.

Performing Forward Subset Selection

Creating objects to store errors

```
train_error_st <- matrix(rep(NA,20))
test_error_st <- matrix(rep(NA,20))
```

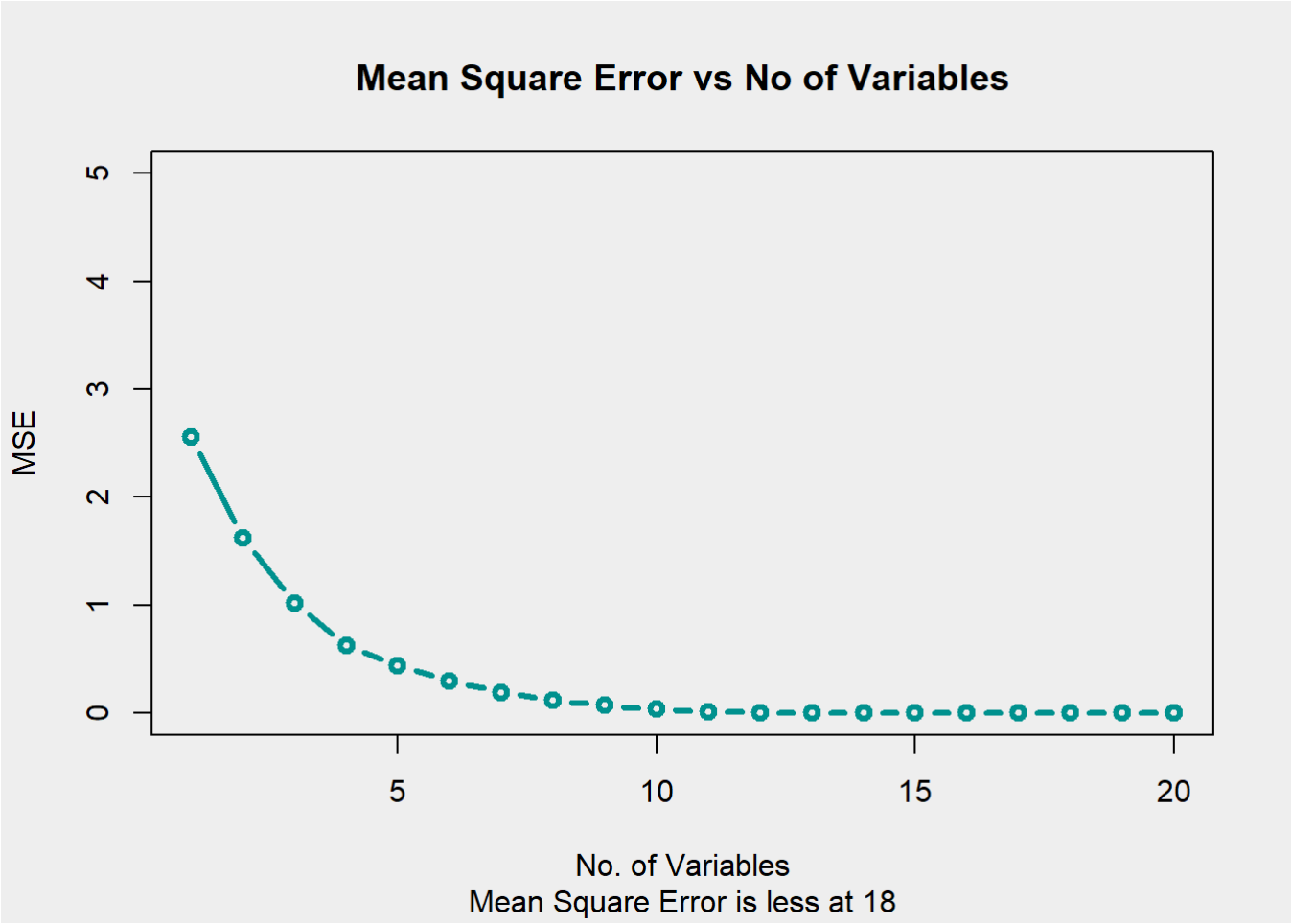
```
predict.regsubsets = function(object, newdata, id){
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object,id=id)
    xvars=names(coefi)
    mat[,xvars]%*%coefi
}
```

```
fwd_subset <- regsubsets(C21~., data = X_data, nbest = 1, nvmax = 20, method = "forward")
```

```
for(i in 1:20){
  y_hat_train = predict(fwd_subset, newdata = train_data, id = i)
  y_hat_test = predict(fwd_subset, newdata = test_data, id = i)

  train_error_st[i] = (1/length(y_train_data))*sum((y_train_data-y_hat_train)^2)
  test_error_st[i] = (1/length(y_test_data))*sum((y_test_data-y_hat_test)^2)
}
```

Plotting Test Error of Forward Subset Selection

```
par(bg = '#EEEEEE')
plot(train_error_st, col="#00918E", type = "b", xlab = "No. of Variables", ylab = "MSE", ylim
= c(0,5.0), main = "Mean Square Error vs No of Variables", sub="Mean Square Error is less at
 18", lwd = 3.0)
```




```
#lines(test_error_st, col = "#00918E", type = "b", lwd = 3.0)
```

Subset with minimum error

```
which(train_error_st == min(train_error_st))
```

```
## [1] 18
```

```
print(min(train_error_st))
```

```
## [1] 1.055713e-10
```

Performing Exhaustive Selection - Best Subset Selection

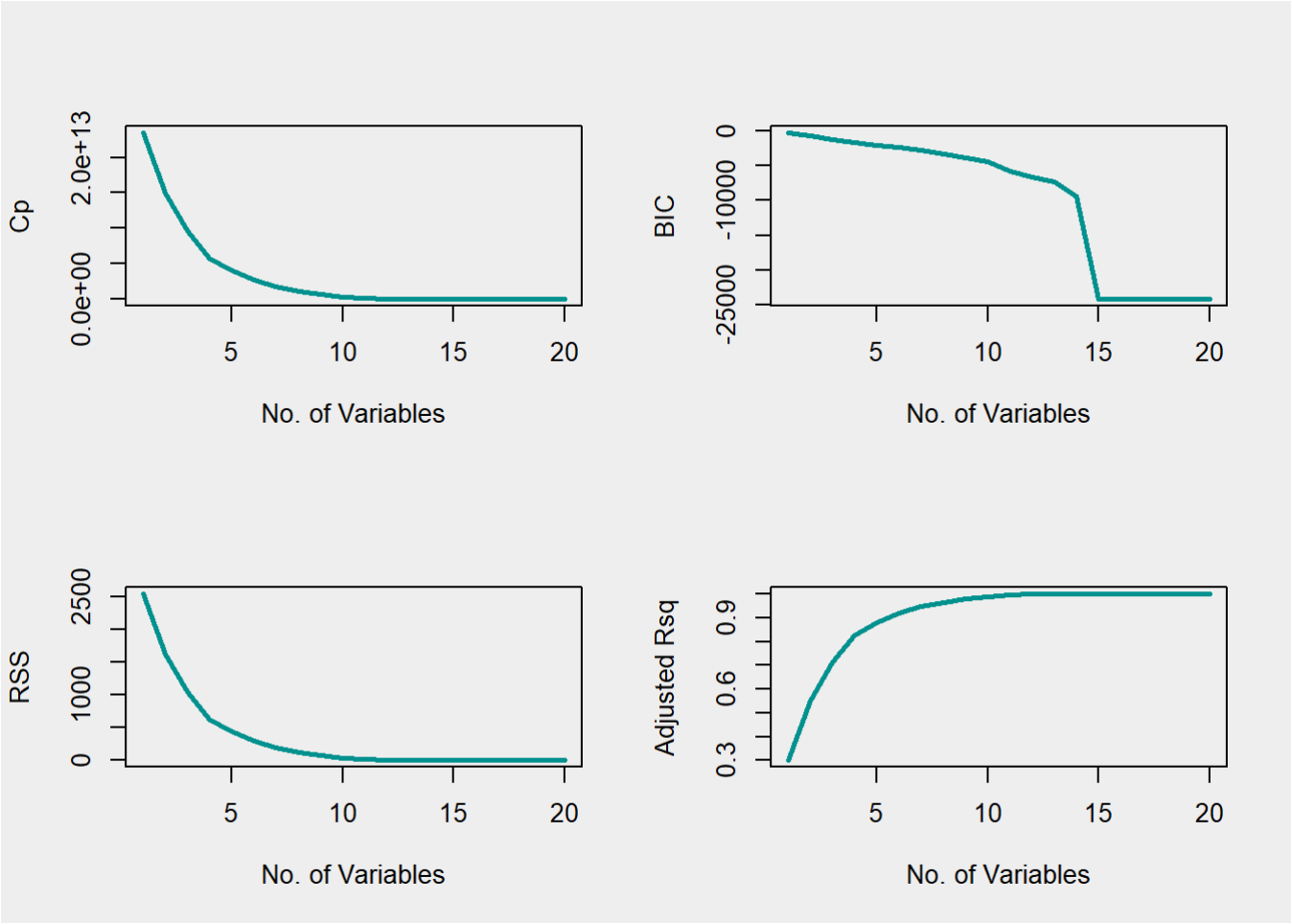
Defining the model

```
ex_subset <- regsubsets(C21~., data = X_data, nbest = 1, nvmax = 20, method = "exhaustive")
ex_summary <- summary(ex_subset)
names(ex_summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

Plotting Exhaustive Selection Measurements - Errors

```
par(mfrow = c(2,2),bg = '#EEEEEE')
plot(ex_summary$cp, xlab = "No. of Variables", ylab = "Cp", type = "l",col = "#00918E",lwd =
2.5)
plot(ex_summary$bic, xlab = "No. of Variables", ylab = "BIC", type = "l",col = "#00918E",lwd
 = 2.5)
plot(ex_summary$rss, xlab = "No. of Variables", ylab = "RSS", type = "l",col = "#00918E",lwd
 = 2.5)
plot(ex_summary$adjr2, xlab = "No. of Variables", ylab = "Adjusted Rsq", type = "l",col = "#0
0918E",lwd = 2.5)
```



Finding the optimal model measures selection

```
which(ex_summary$cp == min(ex_summary$cp))
```

```
## [1] 17
```

```
which(ex_summary$bic == min(ex_summary$bic))
```

```
## [1] 16
```

```
which(ex_summary$rss == min(ex_summary$rss))
```

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

```
which(ex_summary$adjr2 == max(ex_summary$adjr2))
```

```
## [1] 17
```

```
print(min(ex_summary$rss))
```

```
## [1] 1.059863e-07
```

```
print(max(ex_summary$adjr2))
```

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

Performing Backward Subset Selection

Creating objects to store errors

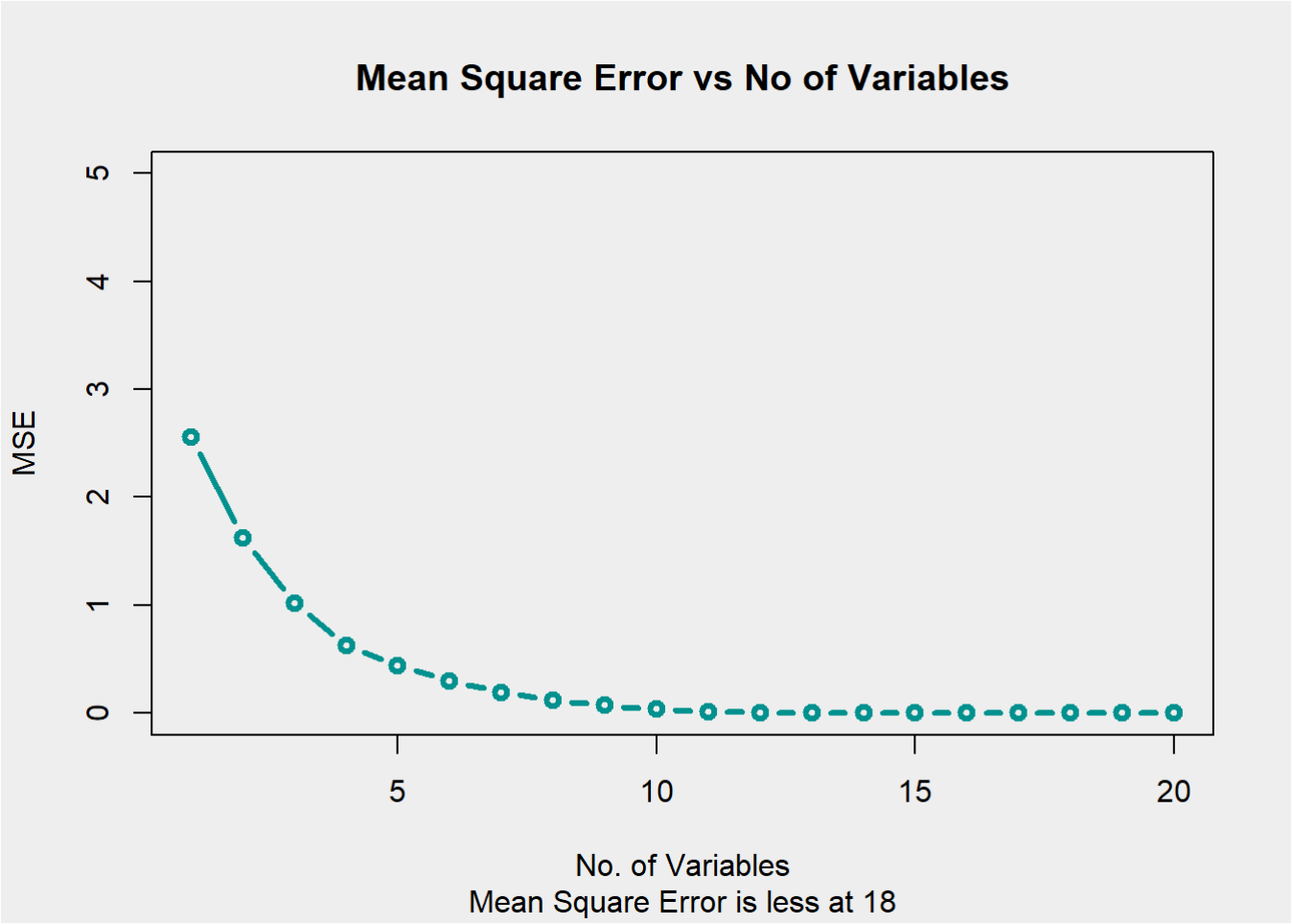
```
train_error_bk <- matrix(rep(NA,20))
test_error_bk <- matrix(rep(NA,20))
```

```
fwd_subset <- regsubsets(C21~., data = X_data, nbest = 1, nvmax = 20, method = "backward")
```

```
for(i in 1:20){
  y_hat_train = predict(fwd_subset, newdata = train_data, id = i)
  y_hat_test = predict(fwd_subset, newdata = test_data, id = i)

  train_error_bk[i] = (1/length(y_train_data))*sum((y_train_data-y_hat_train)^2)
  test_error_bk[i] = (1/length(y_test_data))*sum((y_test_data-y_hat_test)^2)
}
```

Plotting Test Error of Backward Subset Selection

```
par(bg = '#EEEEEE')
plot(train_error_bk, col="#00918E", type = "b", xlab = "No. of Variables", ylab = "MSE", ylim
= c(0,5.0), main = "Mean Square Error vs No of Variables", sub="Mean Square Error is less at
 18", lwd = 3.0)
```




```
#lines(test_error_bk, col = "#00918E", type = "b", lwd = 3.0)
```

Subset with minimum error

```
which(train_error_bk == min(train_error_bk))
```

```
## [1] 18
```

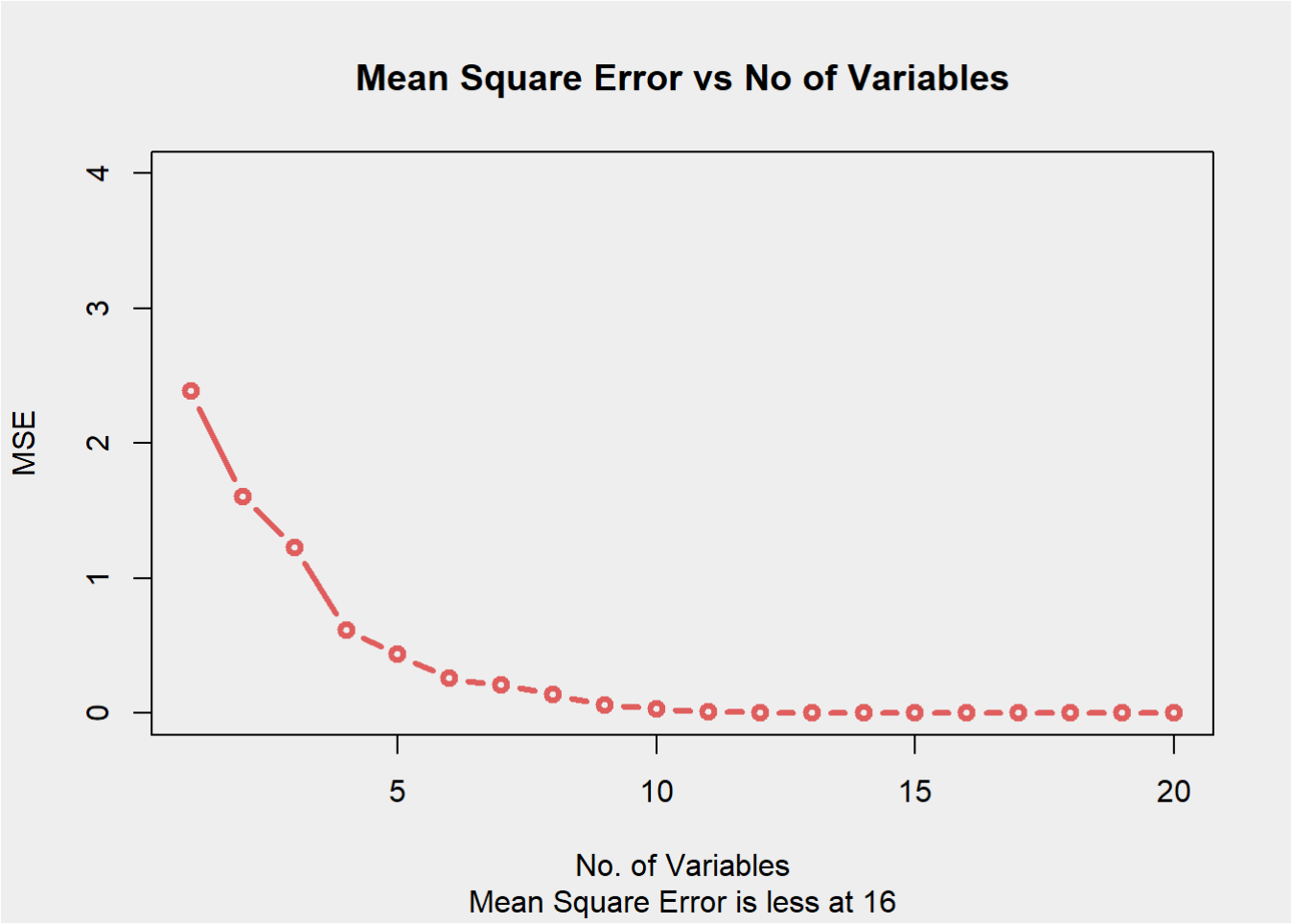
```
print(min(train_error_bk))
```

```
## [1] 1.055713e-10
```

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

Test Error of Forward Subset Selection

```
par(bg = '#EEEEEE')
plot(test_error_st, col="#E05D5D", type = "b", xlab = "No. of Variables", ylab = "MSE", ylim
 = c(0,4.0), main = "Mean Square Error vs No of Variables", sub="Mean Square Error is less at
16", lwd = 3.0)
```



Subset with minimum error

```
which(test_error_st == min(test_error_st))
```

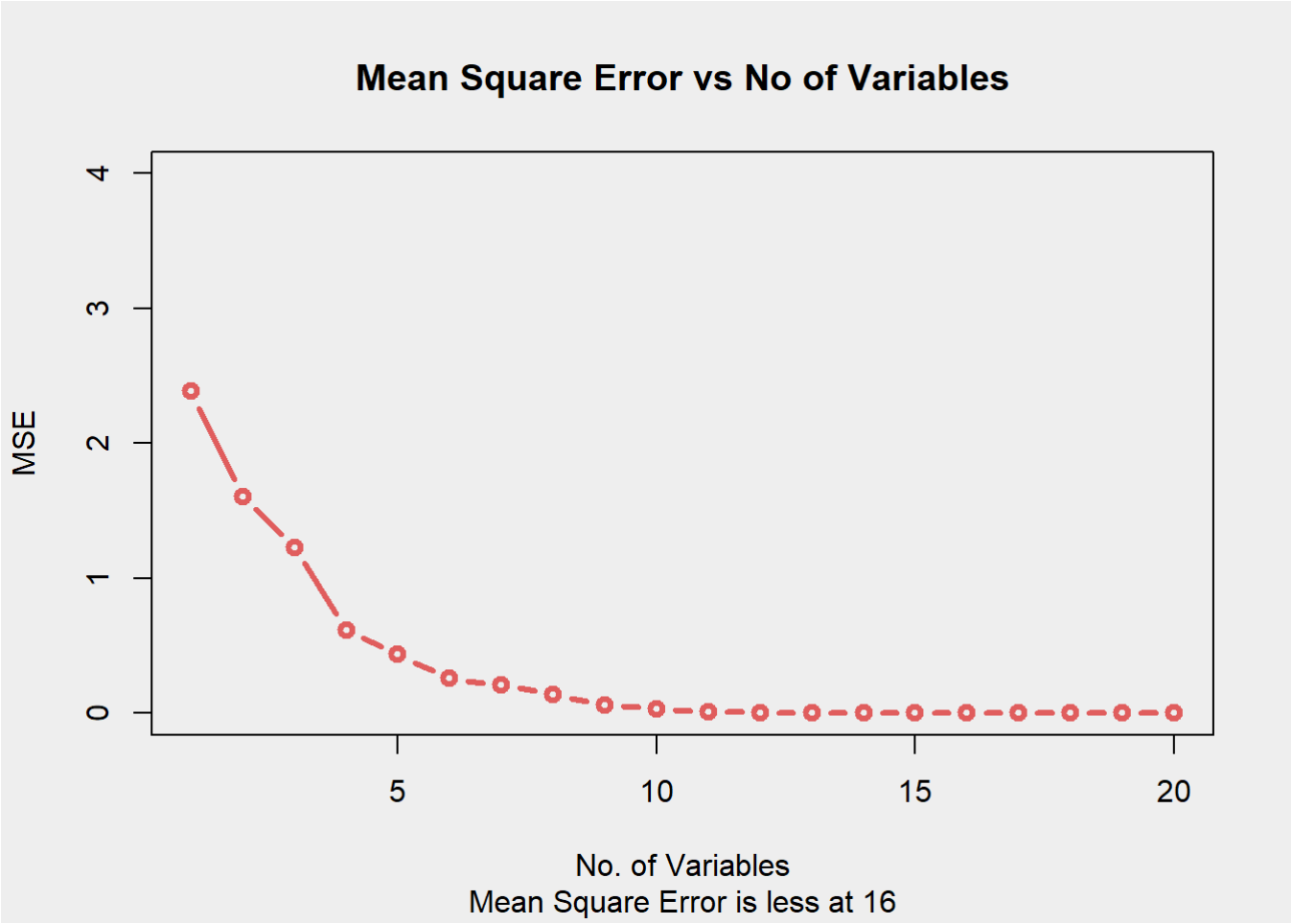
```
## [1] 16
```

```
print(min(test_error_st))
```

```
## [1] 1.085376e-10
```

Test Error of Backward Subset Selection

```
par(bg = '#EEEEEE')
plot(test_error_bk, col="#E05D5D", type = "b", xlab = "No. of Variables", ylab = "MSE", ylim
 = c(0,4.0), main = "Mean Square Error vs No of Variables", sub="Mean Square Error is less at
16", lwd = 3.0)
```



Subset with minimum error

```
which(test_error_bk == min(test_error_bk))
```

```
## [1] 16
```

```
print(min(test_error_bk))
```

```
## [1] 1.085376e-10
```

(e) For which model size does the test set MSE take on its minimum value? Comment on your results. If it takes on its minimum value for a model containing only an intercept a model containing all the features, then play around with the way that you are generating the data in (a) until you come up with a scenario in which the test set MSE is minimized for an intermediate model size.

• The test error is minimum at the model size of 16 for forward subset selection with an error of 1.085376e-10

• Similarly, the test error is minimum at the model size of 16 for backward subset selection with an error of 1.085376e-10

• For Exhaustive Subset Selection, the test error is low at the model size of 17

(f) 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.

The model size of 16 is chosen from the subset selection as the best size of model

Actual Coefficient (Randomly generated)

```
print(coeff)
```

```
##  [1] 0.39454815 0.30628884 0.01713109 0.07614966 0.00000000 0.17411393
##  [7] 0.43311309 0.00000000 0.04602152 0.27767837 0.00000000 0.22062573
## [13] 0.95674004 0.96267233 0.04108773 0.00000000 0.58364469 0.00000000
## [19] 0.69284816 0.18600950
```

Coefficient of the best model with size 16

```
esti_coeff = coef(fwd_subset, id = 16)
print(esti_coeff)
```

```
##   (Intercept)            C1            C2            C3            C4
## -7.953846e-07  3.945485e-01  3.062885e-01  1.713138e-02  7.615006e-02
##            C5            C6            C7            C9           C10
## -1.095377e-06  1.741139e-01  4.331134e-01  4.602135e-02  2.776784e-01
##           C12           C13           C14           C15           C17
##  2.206258e-01  9.567402e-01  9.626718e-01  4.108775e-02  5.836444e-01
##           C19           C20
##  6.928480e-01  1.860100e-01
```

Initally we have set 0 for 5th,8th,11th,16th,18th values of coefficient. but the model has a small value for the same

Variation in coefficient

```
coeff_diff = (esti_coeff - coeff)
```

```
## Warning in esti_coeff - coeff: longer object length is not a multiple of shorter
## object length
```

```
print(coeff_diff)
```

```
##  [1] -3.945489e-01  8.825966e-02  2.891574e-01 -5.901828e-02  7.615006e-02
##  [6] -1.741150e-01 -2.589992e-01  4.331134e-01 -1.766649e-07  1.833409e-08
## [11]  2.206258e-01  7.361144e-01  5.931753e-03 -9.215846e-01  5.425567e-01
## [16]  6.928480e-01 -3.976347e-01 -7.953846e-07 -2.982997e-01  1.202790e-01
```

(g) Create a plot displaying $\sum$ the ($\beta - \beta$) for a range of values, jth coefficient estimate for the best model containing r, where r coefficients. Comment on what you observe. How do these result compare to part D.

Calculating all the difference in the coeffiecients

```
diff_vector = rep(0,20)
for(i in 1:20){
  esti_coeff = coef(fwd_subset, id = i)
  esti_square = (coeff-esti_coeff)**2
  diff_vector[i] = sqrt(sum(esti_square))
}
```

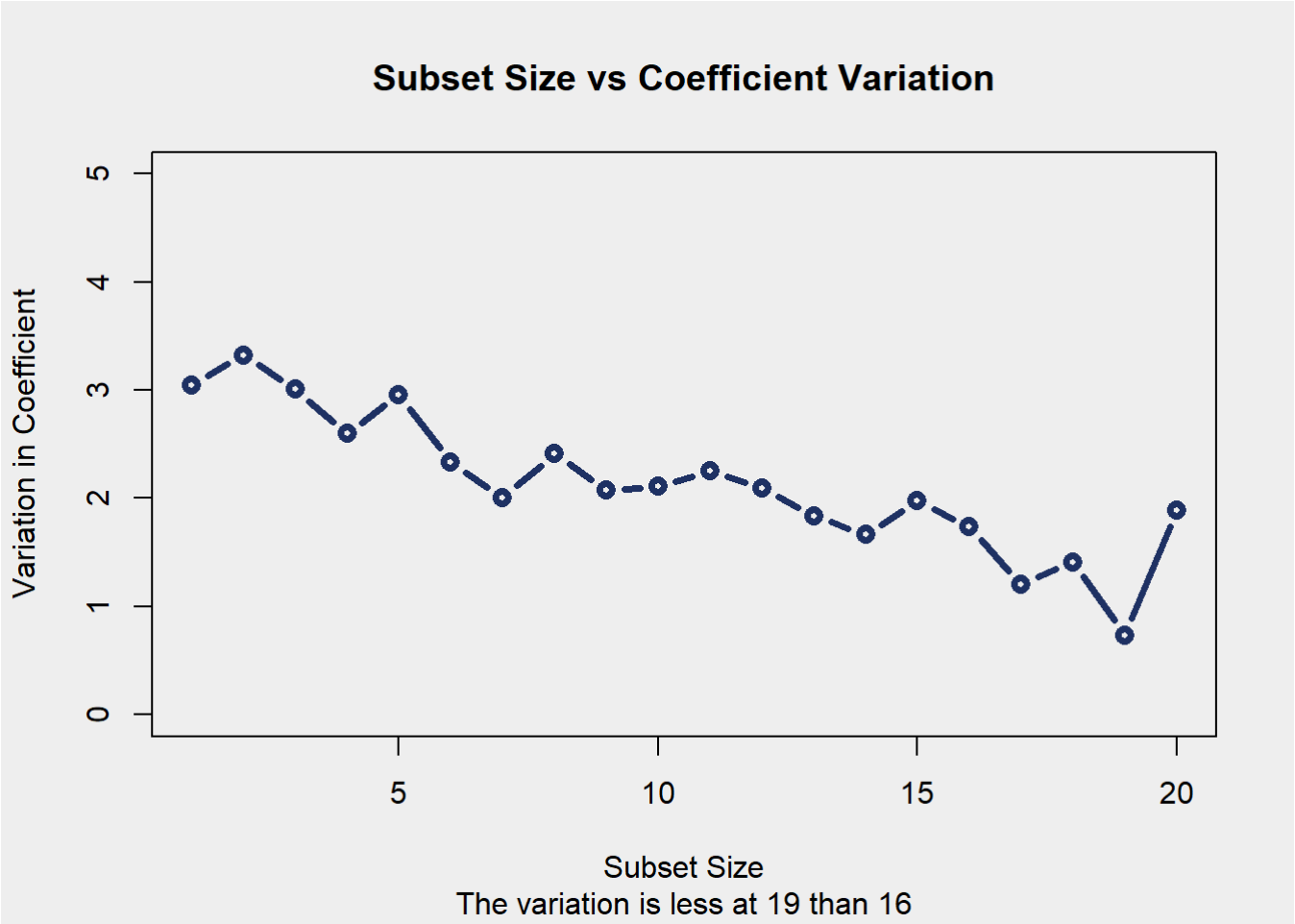
```
## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length

## Warning in coeff - esti_coeff: longer object length is not a multiple of shorter
## object length
```

```
print(diff_vector)
```

```
##  [1] 3.0459321 3.3240266 3.0097482 2.5980146 2.9578333 2.3292123 2.0053495
##  [8] 2.4112171 2.0780180 2.1071695 2.2542468 2.0943113 1.8356624 1.6624203
## [15] 1.9741088 1.7370087 1.2050023 1.4121407 0.7357557 1.8914276
```

Plotting the variation of coefficient

```
x_axis = c(1:20)
par(bg = '#EEEEEE')
plot(x_axis, diff_vector, col="#1E3163", type = "b", xlab = "Subset Size", ylab = "Variation
 in Coefficient", ylim = c(0,5.0), main = "Subset Size vs Coefficient Variation", sub="The va
riation is less at 19 than 16", lwd = 3.6)
```



When we plot the test error in part (d) the subset with minimum error is 16 whereas when we calculated the variation in the coefficient, the subset with size 19 seems to be less.