HW 5 Q2

Ben Howell

3/24/2022

```
suppressMessages(require(tidyverse))
suppressMessages(require(janitor))
suppressMessages(require(purrr))
suppressMessages(require(leaps))
```

```
## Warning: package 'leaps' was built under R version 4.1.3
```

```
suppressMessages(require(ggthemes))

# generate p = 20, n = 1000

set.seed(123)

df <- matrix(rnorm(n = 1000 * 20, mean = 15, sd = 3), 1000, 20)
b <- rnorm(20)
b[1] <- 0
b[5] <- 0
b[7] <- 0
b[13] <- 0
b[17] <- 0

error <- rnorm(1000, mean = 2, sd = 1)

y <- df%*%b + error
# df <- df %>%
#   clean_names()

# df <- df %>%
#   dplyr::mutate(beta = ifelse(rowSums(.) < 320,
#                               rnorm(1000, mean = 5, sd = 1), 0),
#                  y = beta * rowSums(.) +
#                    rnorm(1000, mean = 100, sd = 10))
  # if the rowSum is low, that total gets multiplied by a random number drawn from a distribution with
  # and a standard deviation of 1, otherwise it gets multiplied by 0
  # each observation then gets the error added to it, which is a number from the distribution where mea
```

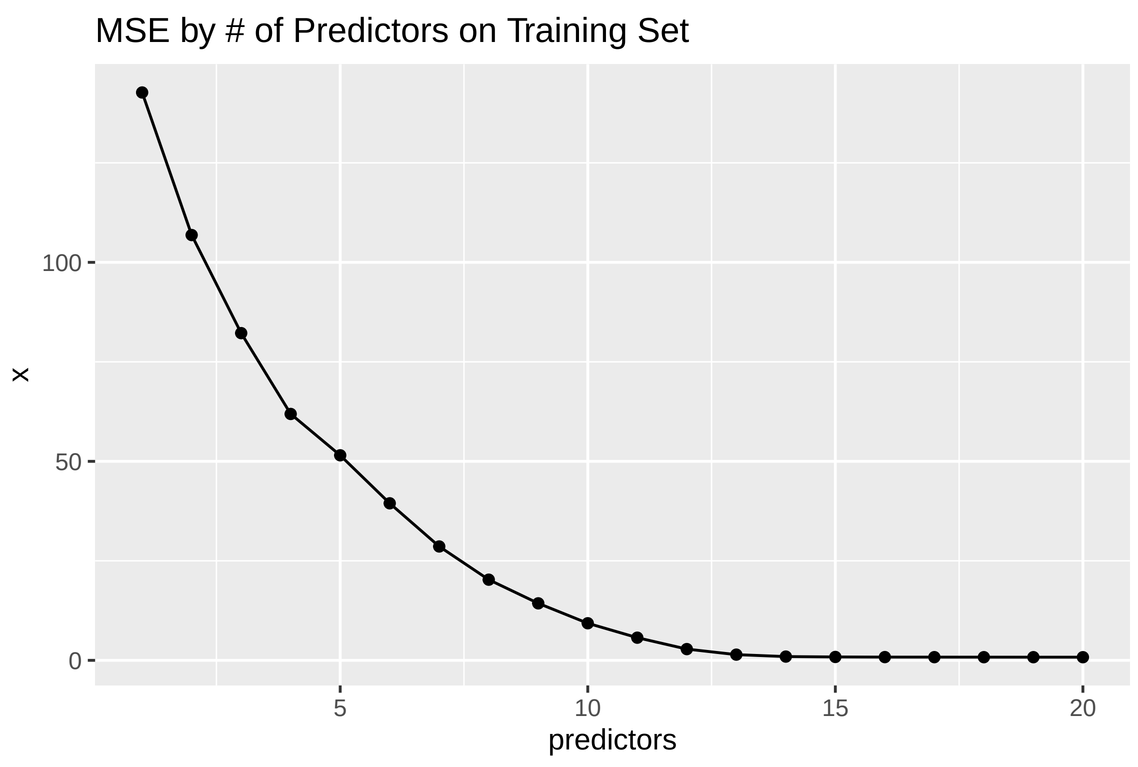
```
train <- sample(nrow(df), 100)
x_train <- df[train,]
y_train <- y[train]
```

```r
x_test <- df[-train,]
y_test <- y[-train]
```

```r
train <- data.frame(x_train, y_train)
test <- data.frame(x_test, y_test)
model <- regsubsets(y_train ~ ., data = train, nvmax = 20)
tmat <- model.matrix(y_train ~., data = train, nvmax = 20)

err <- rep(NA, 20)

for (i in 1:20) {

  cf <- coef(model, id = i)
  pred <- tmat[, names(cf)]%*%cf
  err[i] <- mean((y_train - pred)^2)

  # print(i)
}
```

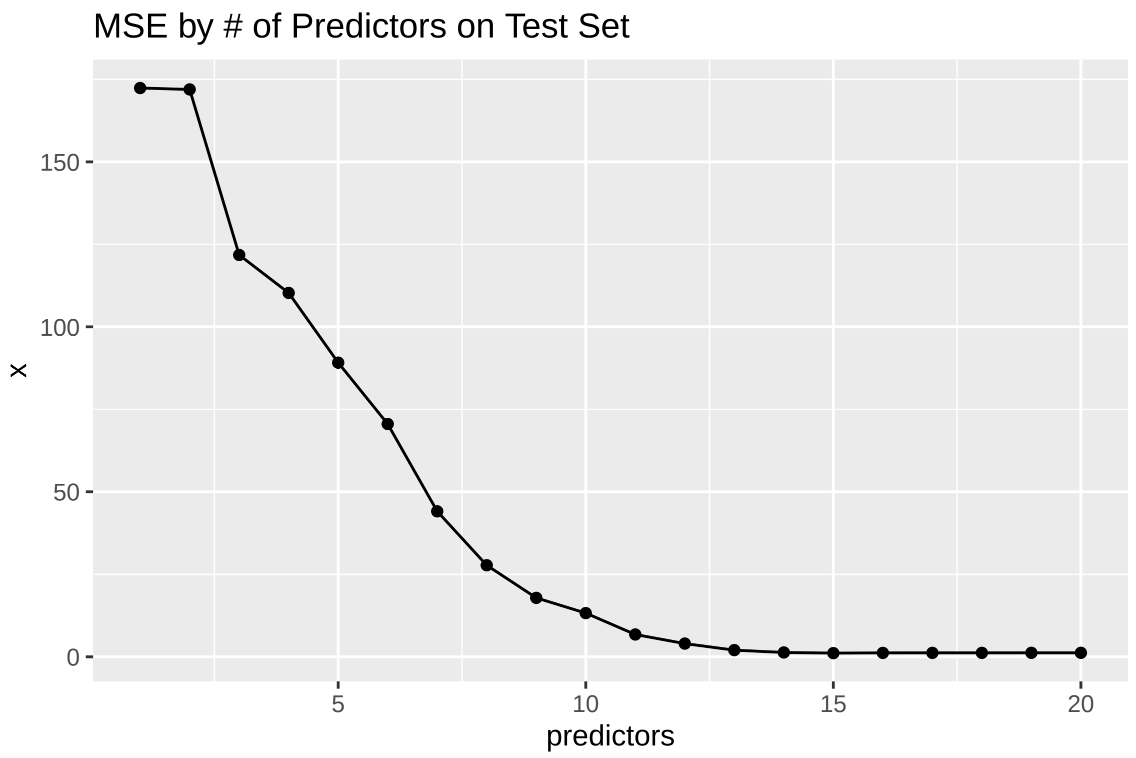
```r
# sum <- summary(model)
# names(sum)
#
# sum$rsq
#
# train_mat <- model.matrix(y ~ ., data = x_train)
# errors <- rep(NA, 20)
#
# for (i in 1:20) {
#
#   cf <- coef(model, id = i)
#   pred <- train_mat[, names(cf)]%*%cf
#   errors[i] <- mean((y_train - pred)^2)
#
#   # print(i)
# }
```

```r
err %>%
  data.frame() %>%
  clean_names() %>%
  mutate(predictors = row_number()) %>%
  ggplot() +
  geom_point(aes(x = predictors, y = x)) +
  geom_line(aes(x = predictors, y = x)) +
  labs(title = "MSE by # of Predictors on Training Set")
```



```r
test_nat <- model.matrix(y_test ~ ., data = test)
test_errors <- rep(NA, 20)

for (i in 1:20) {

  cf <- coef(model, id = i)
  pred <- test_nat[, names(cf)]%*%cf
  test_errors[i] <- mean((y_test - pred)^2)

  # print(i)
}
```

```r
test_errors %>%
  data.frame() %>%
  clean_names() %>%
  mutate(predictors = row_number()) %>%
  ggplot() +
  geom_point(aes(x = predictors, y = x)) +
  geom_line(aes(x = predictors, y = x)) +
  labs(title = "MSE by # of Predictors on Test Set")
```



```
n <- which.min(test_errors)
print(paste0("The test set's MSE is minimized at n = ", n))
```

```
## [1] "The test set's MSE is minimized at n = 15"
```

The test set MSE was minimized at n = 15. While this is close to all the predictors that were possible from the dataset, it was not quite all of them.

```
coef(model, which.min(test_errors))
```

```
## (Intercept)          X2          X3          X4          X6          X8 
##   2.2102961  -0.2633063  -2.0952458  -1.7425503  -1.6516235   1.5532524 
##          X9         X10         X11         X12         X14         X15 
##  -0.3843552   0.8321505   0.7450386   1.0700472  -1.2344482   1.6837671 
##         X16         X18         X19         X20 
##   1.5251852   0.1209883  -0.4902608  -1.2816242
```

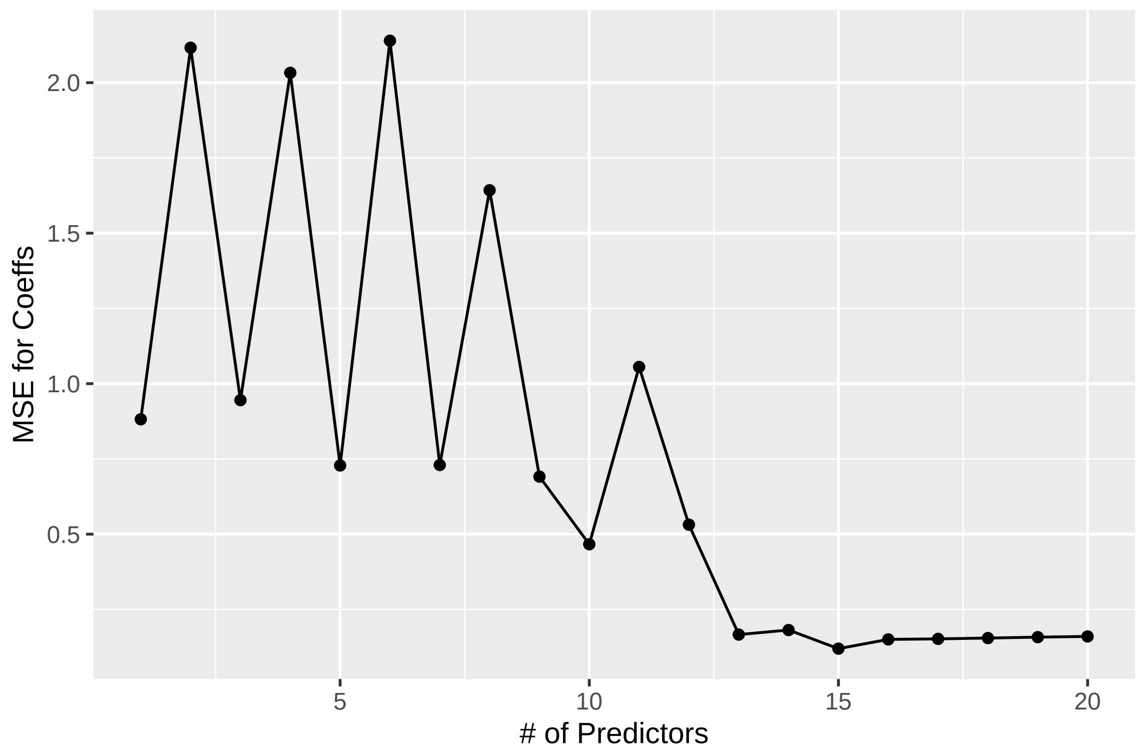
```
print(b)
```

```
##  [1]  0.0000000 -0.2205730 -2.1035148 -1.6678075  0.0000000 -1.6656212
##  [7]  0.0000000  1.5593029 -0.4046239  0.7860261  0.7391292  1.0371054
## [13]  0.0000000 -1.2053098  1.6687057  1.5361579  0.0000000  0.1327737
## [19] -0.5262407 -1.2642772
```

By comparing the coeffiecients that were taken to the `beta` parameter where some were zero-ed out, it appears that many of the zero-ed out `betas` were not deemed to be good predictors for the model.

Honestly, it's hard to see much of a relationship between the way the data was generated and what the coefficients of the model ended up being.

```r
val_err <- rep(NA, 20)

df_cols = colnames(df, do.NULL = FALSE, prefix = "X")

for (i in 1:20) {
    cf <- coef(model, id = i)
    val_err[i] <- sqrt(sum((b[df_cols %in% names(cf)] - cf[names(cf) %in% df_cols])^2) + sum(b[!(df_col
}

val_err %>%
  data.frame() %>%
  clean_names() %>%
  mutate(ord = row_number()) %>%
  ggplot() +
  geom_point(aes(x = ord, y = x)) +
  geom_line(aes(x = ord, y = x)) +
  labs(x = "# of Predictors", y = "MSE for Coeffs")
```



The error between estimated coefficients is the lowest at n = 15, which happens to be the model that reduced the test MSE the most, indicating that the model that most accuractely estimated the parameters also does the best job of predicting values from the test set.