STAT 435 Homework 3

Jaiden Atterbury

2024-05-03

Note:
In this homework, I briefly collaborated with Aarav Vishesh Dewangan and Annabel Wade.

Exercise 1
In this exercise, we will assume that we want to predict a quantitative response y, the concentration of a vitamin in the blood stream an hour after ingestion using x_1 and x_2. In this case, x_1 is the weight of a vitamin in grams, and x_2 is the weight of the vitamin in micrograms (μg) (1 gram = 10^6 micrograms). The linear model is described by the following equation

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon. \tag{1}$$

We will further suppose that we are given that $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$ are the least squares solutions to the model in Equation (1). The main goal of this problem is to develop some intuition for why ridge/lasso regression works better than usual linear regression in high-dimensional settings with linearly dependent features.

Part a
In this part of the exercise, we will start off by describing why the least squares solution for the above regression problem is not unique. The least squares solution is not unique because x_1 and x_2 are linearly dependent features. Since multiple linear regression uses matrix algebra to estimate coefficients, if one or more of the columns in the design matrix X are linearly dependent, then we know that the corresponding linear system does not have a unique solution. From a more intuitive lens, it makes sense that $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique, as they both correspond to features that measure the weight of a vitamin, meaning one could easily be scaled to get the other.

The next goal of this part of the exercise is to derive a general expression for the set of least squares coefficient estimates. This set of values will be written in terms of $\hat{\beta}_0$, $\hat{\beta}_1$, and $\hat{\beta}_2$. In order to do this, we will notice from the problem description that since one gram equals 10^6 micrograms, we can rewrite this in terms of our variables as $10^6 x_1 = x_2$. Replacing x_2 in Equation (1) with $10^6 x_1$, we obtain the following equation

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon \\ &= \beta_0 + \beta_1 x_1 + \beta_2 \left(10^6 x_1\right) + \epsilon \\ &= \beta_0 + \beta_1 x_1 + 10^6 \beta_2 x_1 + \epsilon \\ &= \beta_0 + \left(\beta_1 + 10^6 \beta_2\right) x_1 + \epsilon \end{aligned}$$

If we let $\beta_3 = \beta_1 + 10^6 \beta_2$, then our least squares equation then becomes $y = \beta_0 + \beta_3 x_1$. We now have a simple linear regression model. In order to find the values of $\hat{\beta}_0$ and $\hat{\beta}_3$, we would find the values of β_0 and β_3 that minimize the RSS. However, in Homework 1 Exercise 4 we already computed the general form for the ordinary least squares estimators in simple linear regression. In particular, the estimator for β_0 was written as

$$\begin{aligned} \hat{\beta}_0 &= \bar{y} - \hat{\beta}_3 \bar{x} \\ &= \bar{y} - \left(\hat{\beta}_1 + 10^6 \hat{\beta}_2\right) \bar{x} \end{aligned}$$

As shown above, in terms of $\hat{\beta}_1$ and $\hat{\beta}_2$, $\hat{\beta}_0$ can be written as $\hat{\beta}_0 = \bar{y} - \left(\hat{\beta}_1 + 10^6\hat{\beta}_2\right)\bar{x}$. We will now move onto finding expressions for $\hat{\beta}_1$ and $\hat{\beta}_2$ (even though finding one would be enough because they are a linear combination of each other). With that being said, as found in Homework 1 Exercise 4, the estimator for β_3 was written as

$$\begin{aligned}\hat{\beta}_3 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}\\ \hat{\beta}_1 + 10^6\hat{\beta}_2 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}\\ \hat{\beta}_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} - 10^6\hat{\beta}_2\end{aligned}$$

As shown above, in terms of $\hat{\beta}_2$, $\hat{\beta}_1$ can be written as $\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^n (x_i-\bar{x})^2} - 10^6\hat{\beta}_2$. This implies that, in terms of $\hat{\beta}_1$, $\hat{\beta}_2$ can be written as $\hat{\beta}_2 = \frac{\sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})}{10^6\sum_{i=1}^n (x_i-\bar{x})^2} - \frac{\hat{\beta}_1}{10^6}$. Thus, as computed above, a general expression for the set of least squares coefficient estimates of Equation (1) can be written as

$$\begin{aligned}\hat{\beta}_0 &= \bar{y} - \left(\hat{\beta}_1 + 10^6\hat{\beta}_2\right)\bar{x}\\ \hat{\beta}_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} - 10^6\hat{\beta}_2\end{aligned}$$

Obviously, $\hat{\beta}_1$ and $\hat{\beta}_2$ are not unique. In office hours, it was confirmed that the above values serve as an alternative way to display the correct answer. However, it was also mentioned that we should include the following information as well, since this is what will end up being in the answer key. Namely, since $10^6x_1 = x_2$, as shown above, we can rewrite our linear model as $y = \beta_0 + (\beta_1 + 10^6\beta_2)x_1 + \epsilon$. Therefore we can construct a family of solutions $\{\beta_0, \beta_1, \beta_2\}$ using $\{\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2\}$ via the following constraint

$$\begin{cases}\beta_0 = \hat{\beta}_0\\ \beta_1 + 10^6\beta_2 = \hat{\beta}_1 + 10^6\hat{\beta}_2\end{cases}$$

Where the above is an alternative way to write the set of solutions that we computed using the formulas from Homework 1 Problem 4.

Part b
For the rest of the exercise, we will assume that we have n data points $\{(x_{1i}, x_{2i}, y_i)\}_{i=1}^n$. Furthermore, we will attempt to fit a regularized regression model with tuning parameter λ. For the sake of simplicity we will assume that we no longer have an intercept term. That is, $\beta_0 = 0$. Our new linear model takes the form

$$y = \beta_1 x_1 + \beta_2 x_2 + \epsilon \tag{2}$$

In this part of the exercise, we will perform regularization using the model described by Equation (2). We will start by writing down what expression ridge regression is trying to minimize. According to the notes, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values of β_1 and β_2 that minimize the following

$$\sum_{i=1}^n \left(y_i - \sum_{j=1}^2 \beta_j x_{ji}\right)^2 + \lambda\sum_{j=1}^2 \beta_j^2 = \sum_{i=1}^n (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 + \lambda\left(\beta_1^2 + \beta_2^2\right)$$

Where $\lambda \geq 0$ is a tuning parameter. Notice that the above expression could be rewritten as $RSS + \lambda\left(\beta_1^2 + \beta_2^2\right)$. We will now write down what expression lasso regression is trying to minimize. According to the notes, the lasso regression coefficient estimates $\hat{\beta}^L$ are the values of β_1 and β_2 that minimize the following

$$\sum_{i=1}^n \left(y_i - \sum_{j=1}^2 \beta_j x_{ji}\right)^2 + \lambda\sum_{j=1}^2 |\beta_j| = \sum_{i=1}^n (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 + \lambda\left(|\beta_1| + |\beta_2|\right)$$

Where $\lambda \geq 0$ is a tuning parameter. Notice that the above expression could be rewritten as $RSS + \lambda\left(|\beta_1| + |\beta_2|\right)$.

The parameters that are being varied to find this minimum are β_1, β_2 and λ. In particular, λ is a parameter that is used to shrink the coefficient estimates towards zero. In lasso regression some of these coefficient values can be exactly zero due to the ℓ_1 penalty. As mentioned above, since increasing λ shrinks the coefficient estimates, changing the value of λ directly impacts the values of β_1 and β_2.

In part (a), we mentioned that x_1 and x_2 were linearly dependent. Furthermore, we went a step further and stated that $10^6 x_1 = x_2$, was the exact linear dependence of these two variables. Furthermore, from this, we found out that our linear model became $y = \beta_0 + (\beta_1 + 10^6\beta_2)x_1 + \epsilon$. This led to the following constraint/set of solutions

$$\begin{cases} \beta_0 = \hat{\beta}_0 \\ \beta_1 + 10^6\beta_2 = \hat{\beta}_1 + 10^6\hat{\beta}_2 \end{cases}$$

However, we got rid of the intercept, so the constraint that still carries over from part (a) is $\beta_1 + 10^6\beta_2 = \hat{\beta}_1 + 10^6\hat{\beta}_2$. Thus, while λ varies, the values of β_1 and β_2 will change together dependently.

Part c

In this part of the exercise, we will now define $\beta_1 + 10^6\beta_2 = c$, where c is some fixed constant. This implies that $\beta_1 = c - 10^6\beta_2$. We will also consider just the ridge regression problem defined in part (b). We will first rewrite it in terms of c, β_2, λ, x_{1i}, and y_i. From part (a), we also defined $10^6 x_{1i} = x_{2i}$. Using the above facts, we will rewrite the ridge regression problem as minimizing the following in terms of β_1 and β_2 (although in our case we will only be minimizing in terms of β_2 since they are related)

$$\begin{aligned}
&\sum_{i=1}^{n} \left(y_i - \beta_1 x_{1i} - \beta_2 x_{2i}\right)^2 + \lambda\left(\beta_1^2 + \beta_2^2\right) \\
&\sum_{i=1}^{n} \left(y_i - (c - 10^6\beta_2)x_{1i} - 10^6\beta_2 x_{1i}\right)^2 + \lambda\left((c - 10^6\beta_2)^2 + \beta_2^2\right) \\
&\sum_{i=1}^{n} \left(y_i - cx_{1i} + 10^6\beta_2 x_{1i} - 10^6\beta_2 x_{1i}\right)^2 + \lambda\left(c^2 - 2c10^6\beta_2 + 10^{12}\beta_2^2 + \beta_2^2\right) \\
&\sum_{i=1}^{n} \left(y_i - cx_{1i}\right)^2 + \lambda\left(c^2 - 2c10^6\beta_2 + (10^{12}+1)\beta_2^2\right) \\
&\sum_{i=1}^{n} \left(y_i - cx_{1i}\right)^2 + \lambda c^2 - \lambda 2c10^6\beta_2 + \lambda(10^{12}+1)\beta_2^2
\end{aligned}$$

Thus, as can be seen above, the ridge regression problem rewritten in terms of c, β_2, λ, x_{1i}, and y_i corresponds to minimizing $\sum_{i=1}^{n} \left(y_i - cx_{1i}\right)^2 + \lambda c^2 - \lambda 2c10^6\beta_2 + \lambda(10^{12}+1)\beta_2^2$ in terms of β_2. We will now find the coefficients that solve this ridge regression problem. To start off, we will take the derivative with respect to β_2 of the above ridge regression problem

$$\frac{d}{d\beta_2}\left(\sum_{i=1}^{n} \left(y_i - cx_{1i}\right)^2 + \lambda c^2 - \lambda 2c10^6\beta_2 + \lambda(10^{12}+1)\beta_2^2\right) = -\lambda 2c10^6 + 2\lambda(10^{12}+1)\beta_2$$

As computed above, the derivative with respect to β_2 of the above ridge regression problem equals $-\lambda 2c10^6 +$

$2\lambda(10^{12}+1)\beta_2$. We will now set this expression equal to zero and solve for β_2

$$\begin{aligned}
0 &= -\lambda 2c10^6 + 2\lambda(10^{12}+1)\beta_2 \\
0 &= -\lambda c10^6 + \lambda(10^{12}+1)\beta_2 \\
\lambda c10^6 &= \lambda(10^{12}+1)\beta_2 \\
\beta_2 &= \frac{\lambda c10^6}{\lambda(10^{12}+1)} \\
\beta_2 &= \frac{c10^6}{10^{12}+1}
\end{aligned}$$

Therefore, we have shown that $\beta_2 = \frac{c10^6}{10^{12}+1}$, and hence we can write β_1 as

$$\begin{aligned}
\beta_1 &= c - 10^6 \cdot \left(\frac{c10^6}{10^{12}+1}\right) \\
&= c - \frac{c10^{12}}{10^{12}+1} \\
&= c\left(1 - \frac{10^{12}}{10^{12}+1}\right) \\
&= c\left(\frac{10^{12}+1}{10^{12}+1} - \frac{10^{12}}{10^{12}+1}\right) \\
&= c\left(\frac{10^{12}+1-10^{12}}{10^{12}+1}\right) \\
&= \frac{c}{10^{12}+1}
\end{aligned}$$

Therefore, as shown above, $\beta_1 = \frac{c}{10^{12}+1}$. Since c is a fixed constant, we have that $\hat{\beta}_1$ and $\hat{\beta}_2$ are unique. These coefficients are non-zero as long as c does not equal zero. However, given the fact that ridge regression does not force coefficients to be exactly zero, it is safe to assume that these coefficients will never be zero. Note that, since we assume the function is convex, we do not need to do the second derivative test.

Part d

In this part of the exercise, as in (c), we will let $\beta_1 + 10^6\beta_2 = c$. The goal of this exercise is to justify why LASSO will drop one of the covariates. That is, we will explain why LASSO would choose $\hat{\beta}_1 = 0$ and $\hat{\beta}_2 = 10^{-6}c$ over $\hat{\beta}_1 \neq 0$ and $\hat{\beta}_2 \neq 0$.

First off, as shown in the figure on slide 37 of the chapter 7 notes, we can't use our normal calculus techniques to prove why LASSO will drop a covariate because the places in which a covariate equals zero are non-differentiable (due to the ℓ_1 penalty). Thus, we will have to find a different strategy to prove the result. We will start by taking advantage of the hint and figure out what LASSO is minimizing.

As stated in the exercise description, we define $\beta_1 + 10^6\beta_2 = c$, where c is some fixed constant. This implies that $\beta_1 = c - 10^6\beta_2$. We will also consider the LASSO regression problem defined in part (b). From part (a), we also defined $10^6 x_{1i} = x_{2i}$. Using the above facts, we will rewrite the LASSO regression problem as minimizing the following in terms of β_1 and β_2

$$\begin{aligned}
&\sum_{i=1}^{n} (y_i - \beta_1 x_{1i} - \beta_2 x_{2i})^2 + \lambda (|\beta_1| + |\beta_2|) \\
&\sum_{i=1}^{n} \left(y_i - (c - 10^6\beta_2) x_{1i} - 10^6 \beta_2 x_{1i}\right)^2 + \lambda (|\beta_1| + |\beta_2|) \\
&\sum_{i=1}^{n} \left(y_i - c x_{1i} + 10^6\beta_2 x_{1i} - 10^6 \beta_2 x_{1i}\right)^2 + \lambda (|\beta_1| + |\beta_2|) \\
&\sum_{i=1}^{n} (y_i - c x_{1i})^2 + \lambda (|\beta_1| + |\beta_2|)
\end{aligned}$$

Thus, as can be seen above, the LASSO regression problem rewritten corresponds to minimizing $\sum_{i=1}^{n} (y_i - cx_{1i})^2 + \lambda (|\beta_1| + |\beta_2|)$ in terms of β_1 and β_2. However, since $\sum_{i=1}^{n} (y_i - cx_{1i})^2$ does not contain β_1 or β_2, the LASSO regression problem corresponds to minimizing $\lambda (|\beta_1| + |\beta_2|)$. Thus, in order to justify why LASSO will drop a covariate, we must show that LASSO would choose $\hat{\beta}_1 = 0$ and $\hat{\beta}_2 = 10^{-6}c$ over $\hat{\beta}_1 \neq 0$ and $\hat{\beta}_2 \neq 0$. In order to do this, we can show that $\lambda \left(|\hat{\beta}_1| + |\hat{\beta}_2|\right) < \lambda (|\beta_1| + |\beta_2|))$, where $\hat{\beta}_1 = 0$, $\hat{\beta}_2 = 10^{-6}c$, $\beta_1 \neq 0$, and $\beta_2 \neq 0$. This proof is shown below.

$$\begin{aligned}
\lambda \left(|\hat{\beta}_1| + |\hat{\beta}_2|\right) &= \lambda \left(|0| + |10^{-6}c|\right) \\
&= \lambda \left(|10^{-6}c|\right) \\
&= \lambda \left(|10^{-6}(\beta_1 + 10^6\beta_2)|\right) \\
&= \lambda \left(|10^{-6}\beta_1 + \beta_2)|\right) \\
&\leq \lambda \left(|10^{-6}\beta_1| + |\beta_2)|\right) \quad \text{(By the Triangle Inequality)} \\
&< \lambda (|\beta_1| + |\beta_2)|) \quad \text{(Since } \beta_1 \neq 0\text{)}
\end{aligned}$$

Thus, when $\hat{\beta}_1 = 0$, $\hat{\beta}_2 = 10^{-6}c$, $\beta_1 \neq 0$, and $\beta_2 \neq 0$, it follows that $\lambda \left(|\hat{\beta}_1| + |\hat{\beta}_2|\right) < \lambda (|\beta_1| + |\beta_2|))$. Therefore, LASSO chooses $\hat{\beta}_1 = 0$ and $\hat{\beta}_2 = 10^{-6}c$ over $\hat{\beta}_1 \neq 0$ and $\hat{\beta}_2 \neq 0$.

Exercise 2
In this exercise, we will construct 95% bootstrap intervals. Before getting started, we must recall that a large sample two-sided 95% confidence interval for a regression coefficient β_j in multiple linear regression is approximately

$$\left(\hat{\beta}_j \pm 1.96 * SE(\hat{\beta}_j)\right) \tag{3}$$

In this exercise, we will work with the `Auto` data set in `Auto.csv`. The goal of this exercise is to predict the mileage (mpg) using weight and horsepower as predictors. To reduce skewness, we will do a log transformation of the mileage variable. We will use the linear model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$$

where y represents log(mpg), x_1 represents weight, and x_2 represents horsepower. Since the coding methods in this problem are important, we will show the code for each different bootstrap method (as was confirmed in Ed post #81).

Part a
In this part of the exercise, we will compute and report the standard errors and standard regression 95% confidence intervals for β_1 and β_2. This will be done using the `summary()` and `confint()` function in `R`. This is done below.

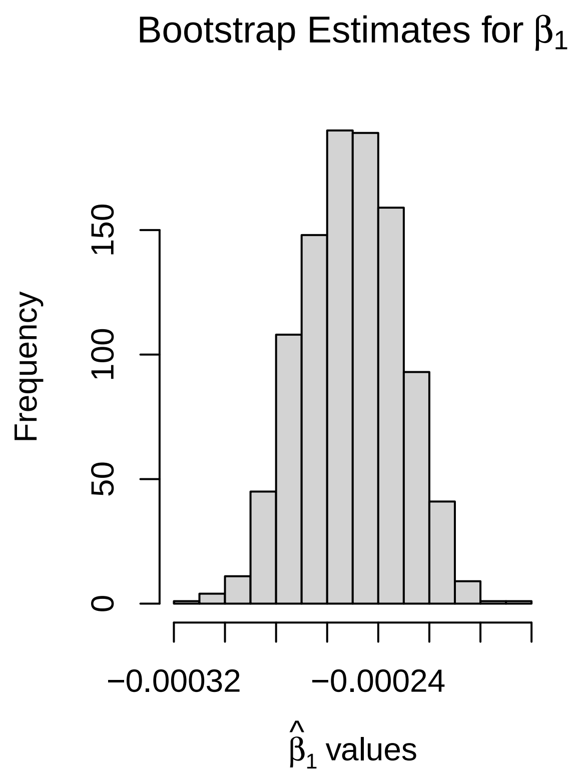
```
##
## Call:
## lm(formula = Auto$logmpg ~ Auto$weight + Auto$horsepower)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.45865 -0.09660 -0.00648  0.10038  0.49885
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.1109472  0.0293674 139.983  < 2e-16 ***
## Auto$weight     -0.0002504  0.0000186 -13.462  < 2e-16 ***
## Auto$horsepower -0.0025573  0.0004104  -6.231  1.2e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.157 on 389 degrees of freedom
## Multiple R-squared:  0.7879, Adjusted R-squared:  0.7869
## F-statistic: 722.7 on 2 and 389 DF,  p-value: < 2.2e-16
```

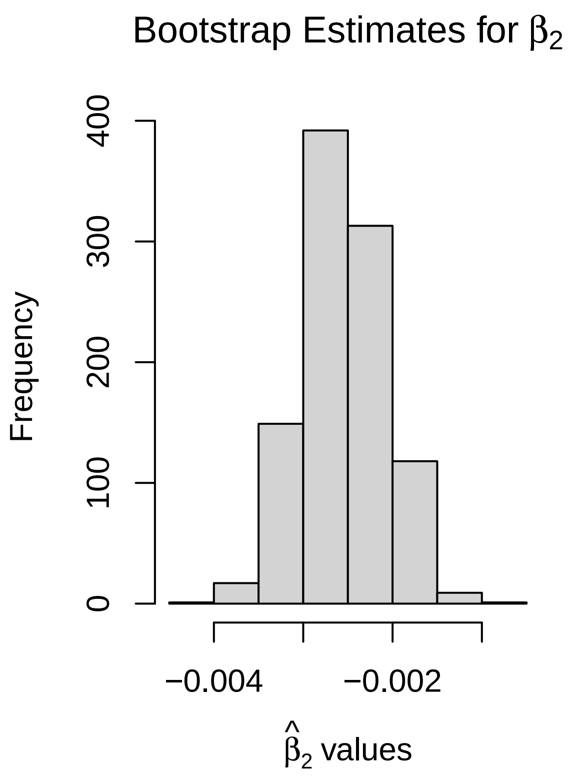
As can be seen above, the estimated coefficient corresponding to the weight predictor is $\hat{\beta}_1 = -0.0002504$ with a standard error of $SE(\hat{\beta}_1) = 0.0000186$. The corresponding 95% confidence interval for the true coefficient of the weight predictor β_1 is [-0.0002869, -0.0002138]. Similarly, the estimated coefficient corresponding to the horsepower predictor is $\hat{\beta}_2 = -0.0025573$ with a standard error of $SE(\hat{\beta}_2) = 0.0004104$. The corresponding 95% confidence interval for the true coefficient of the horsepower variable β_2 is [-0.0033642, -0.0017504].

Part b
In this part of the exercise, we will construct 95% bootstrap percentile intervals for β_1 and β_2. We construct such intervals by using the 0.025 and 0.975 quantiles of the bootstrap regression coefficient estimates. In particular, we will set the seed to 123, and implement this using the `sample()` and `quantile()` function in `R`. We will use $R = 1000$ bootstrap replicates. In the end we will report the bootstrap percentile intervals for β_1 and β_2 as well as plot the histograms of the bootstrap estimates $\hat{\beta}_1$ and $\hat{\beta}_2$. As mentioned in the exercise description, and confirmed in Ed, we will show our code for the bootstrap simulation, as well as the code for constructing the intervals. All of this is done below.

```r
# Set the seed for reproducibility
set.seed(123)

# Set the number of replications R to 1000
R <- 1000

# Create a matrix to store the resulting coefficient values
boot_coefs <- matrix(data=NA, nrow=R, ncol=2)

# Create a for loop to create the 1000 bootstrap simulations
for(i in 1:R) {
  # Create a bootstrap sample
  boot_sample = sample(x=nrow(Auto), size=nrow(Auto), replace=TRUE)

  # Fit a linear model on the bootstrap sample
  boot_model <- lm(logmpg~weight+horsepower, data=Auto, subset=boot_sample)

  # Add the coefficients to the matrix
  boot_coefs[i,] <- coef(boot_model)[2:3]
}

# Find the 0.025 and 0.975 quantile for the weight coefficient
weight_quantile <- quantile(x=boot_coefs[,1], probs=c(0.025, 0.975))

# Find the 0.025 and 0.975 quantile for the horsepower coefficient
horse_quantile <- quantile(x=boot_coefs[,2], probs=c(0.025, 0.975))
```

As calculated using the non-parametric bootstrap, the 95% bootstrap percentile interval for the true coefficient of the weight predictor, β_1, is [-0.0002869, -0.0002147]. Similarly, the corresponding 95% bootstrap percentile interval for the true coefficient of the horsepower variable β_2 is [-0.003463, -0.001649]. We will now display the histograms of the bootstrap estimates.





As can be seen above, both histograms are roughly symmetric around values of approximately -0.00026 and -0.0025, respectively. This would make sense, because the estimators in OLS are supposed to be approximately normally distributed around the true parameters values, and that is the exact distribution that we are estimating when taking bootstrap samples.

Part c

In this part of the exercise, we will create approximate confidence intervals using a different method. Namely, another way to construct 95% bootstrap confidence interval is to estimate the standard error of the regression coefficient $SE(\hat{\beta}_j)$ in Equation (3) using the bootstrap. In particular, we will first set the seed to 123, then implement the bootstrap using the `boot()` function in `R`. Furthermore, we will use $R = 1000$ bootstrap replicates. After this is all done we will report our bootstrap confidence intervals for β_1 and β_2. As mentioned in the exercise description, and confirmed in Ed, we will show our code for the bootstrap simulation, as well as the code for constructing the intervals. All of this is done below.

```r
# Set the seed for reproducibility
set.seed(123)

# Create a function to return bootstrap estimates
coeff_estimation <- function(data, index) {
  model <- lm(logmpg~weight+horsepower, data=data, subset=index)
  return(coef(model)[2:3])
}

# Use the boot function to perform non-parametric bootstrap
boot_result <- boot(data=Auto, statistic=coeff_estimation, R=1000)

# Print the output from boot result
```

```
boot_result
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = coeff_estimation, R = 1000)
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* -0.000250361 -1.760405e-07 0.0000194129
## t2* -0.002557337 -3.388400e-06 0.0004839747
```

As can be seen above, our bootstrap estimates for the standard error of our estimated coefficients are 0.0000194129 and 0.0004839747, respectively. We will use these standard error values along with the estimated coefficients in part (a) to compute our bootstrap confidence intervals using Equation (3). This is done below.

```
# Compute the bootstrap confidence interval for weight
weight_upper <- model_1$coefficients[2]+1.96*0.0000194129
weight_lower <- model_1$coefficients[2]-1.96*0.0000194129

# Compute the bootstrap confidence interval for horsepower
horse_upper <- model_1$coefficients[3]+1.96*0.0004839747
horse_lower <- model_1$coefficients[3]-1.96*0.0004839747
```

As calculated using the non-parametric bootstrap, the 95% bootstrap confidence interval for the true coefficient of the weight predictor β_1 is [-0.0002884, -0.0002123]. Similarly, the corresponding 95% bootstrap percentile interval for the true coefficient of the horsepower variable β_2 is [-0.0035059, -0.0016087].

Part d
In this part of the exercise, we will compare the intervals from (a), (b), and (c), and the standard errors from (a) and (c). We will start by comparing the intervals.

As calculated in parts (a), (b), and (c), the confidence intervals for the true coefficient of the weight predictor were [-0.0002869, -0.0002138], [-0.0002869, -0.0002147], and [-0.0002884, -0.0002123], respectively. Furthermore, as calculated in part (a), (b), and (c), the confidence intervals for the true coefficient of the horsepower predictor were [-0.0033642, -0.0017504], [-0.003463, -0.001649], and [-0.0035059, -0.0016087], respectively. For both coefficients, the intervals computed in parts (a)-(c) were very similar. Furthermore, the intervals for each coefficient, were the widest in part (c), this may be due to the fact that the value of 1.96 is rounded quite a bit. Since there were quite a few estimates used to create the intervals in parts (b) and (c), it makes sense that they differ a little bit from the "true" confidence interval from part (a). Lastly, since we used different bootstrap methods in parts (b) and (c), it makes sense that there are differences between these two intervals as well.

As calculated in parts (a) and (b), the standard error for the estimated coefficient of the weight predictors were 0.0000186 and 0.0000194129, respectively. Similarly, the standard error for the estimated coefficient of the weight predictors were 0.0004104 and 0.0004839747, respectively. For both coefficients, the standard errors computed in parts (a) and (c) were similar. Furthermore, the standard error for each of the estimated coefficients was larger in part (c). Lastly, since there were quite a few estimates used to compute the estimated standard error in part (c), it makes sense that they differ a little bit from the "true" standard error from part (a).

Exercise 3
In this exercise, we will analyze a dataset of our choice. For this we have two options. The first option, labeled (I), is to choose a real dataset with a quantitative response y and $p \geq 30$ predictors. The second option, labeled (II), is to generate our own data with a quantitative response y and $p \geq 30$ predictors. However, in option (II) we would also need to ensure that at least two of the p predictors are linearly dependent.

Part a
In this part of the exercise, since we chose option (II), we will give an overview of how we generated the data and how we introduced linear dependencies between predictors. Due to the fact that I generated my own data, it will be easier to visualize what I actually did if I explicitly show my code. This is shown below.

```r
# Set the seed for reproducibility
set.seed(123)

# Set the number of data points
n <- 100

# Set the number of initial predictors
p <- 30

# Create the data for the first p predictors
X <- matrix(data=rnorm(n=n*p), nrow=n, ncol=p, byrow=FALSE)

# Create an error term for our model
epsilon <- rnorm(n)

# Create the data for the quantitative response y
y <- 1+2*X[,1]+5*X[,15]-2*X[,23]+0.5*X[,30]+epsilon

# Create the data as a dataframe
data <- data.frame(y=y, X)

# Create three linearly dependent predictors
data$X31 <- 2*data$X1+1
data$X32 <- 2*data$X15-3
data$X33 <- 0.5*data$X12
```

The above code starts by creating 100 data points for 30 predictors which come from a $N(0,1)$ distribution. These predictors are called $X_1 - X_{30}$. Once I had my initial predictors, I simulated the errors for my quantitative response y_i from a $N(0,1)$ distribution. After this, my y_i was created using the true model of $y_i = 1 + 2x_{i1} + 5x_{i15} - 2x_{i23} + 0.5x_{i30} + \epsilon_i$. In order to create linear dependencies, I created 3 new predictors that were linear combinations of the 30 existing predictors. These three predictors were defined as $X_{31} = 2X_1 + 1$, $X_{32} = 2X_{15} - 3$, and $X_{33} = 0.5X_{12}$. Therefore, in total, I have 33 predictors, 100 data points, and 3 distinct linear dependencies.

Part b
In this part of the exercise, we will start by splitting our data into a training and a testing set. We are given the choice of using either an 80/20 split or a 90/10 split. In particular, we will opt to go for an 80/20 split. In order to get this training and testing split we will use the `sample.split()` function from the `caTools` package (which I learned in STAT 302). Since $n = 100$ and we're using a 80/20 split, this would imply that $n_{\text{train}} = 80$. Since the number of predictors we have is $p = 34$, it follows that $n_{\text{train}} > p$.

We are now tasked with using the `regsubsets` function in the `leaps` package using forward or backward selection to fit a reduced linear model on the training data. We will opt to use backward selection (which will work since $n_{\text{train}} > p$). In the `regsubsets` function, we will use `nvmax=30` in order to get a model for all sizes (up to the original number of simulated predictors). Furthermore, we will use 10-fold cross validation to choose the best model out of the 30 models of size $1 - 30$. This is done below.

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 3 linear dependencies found
```

Usually when writing up homework's I would suppress all warning messages, however, it is interesting to see that the `regsubsets()` function specifically mentioned that my predictors had 3 linear dependencies, which seems to back up my simulation from part (a). Moving back to the task of the exercise, backward selection using 10-fold cross-validation chose the model of size 29. Below we will show which predictors `regsubsets` chose and their corresponding coefficients

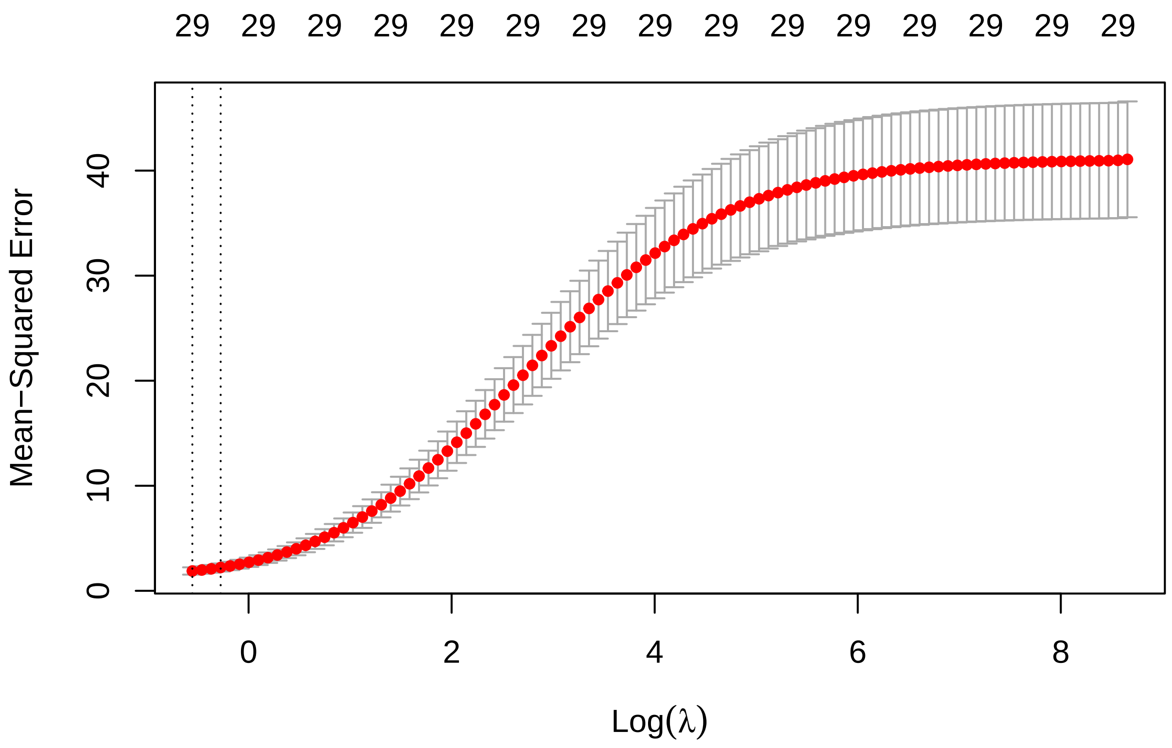
```
##  (Intercept)           X1           X2           X3           X4           X5 
##  0.869395127  1.696619324 -0.182743428 -0.035715328 -0.008889434 -0.102940934 
##           X6           X7           X8           X9          X10          X11 
## -0.055904000  0.040049715  0.079508964  0.271589758 -0.030883833  0.120614445 
##          X12          X13          X14          X15          X16          X17 
##  0.132681062 -0.312326511 -0.270066761  5.077543763  0.143656281 -0.044434194 
##          X18          X19          X20          X21          X22          X23 
##  0.261337432  0.114969771 -0.182403271 -0.074625826 -0.027597816 -1.996723441 
##          X24          X25          X26          X28          X29          X30 
## -0.046113493  0.010681470 -0.148174649  0.008919216  0.017051671  0.574892570
```

As can be seen from the above output, `regsubsets` dropped $X_{31} - X_{33}$, the three linear dependent predictors, as well as X_{27}. Therefore, the rest of the 29 predictors are in the selected model. We will now use this selected model with the untouched test data to compute and report the test error. This will be done below.

As computed in `R` the test MSE for our best model selected using backward selection was 3.3061591.

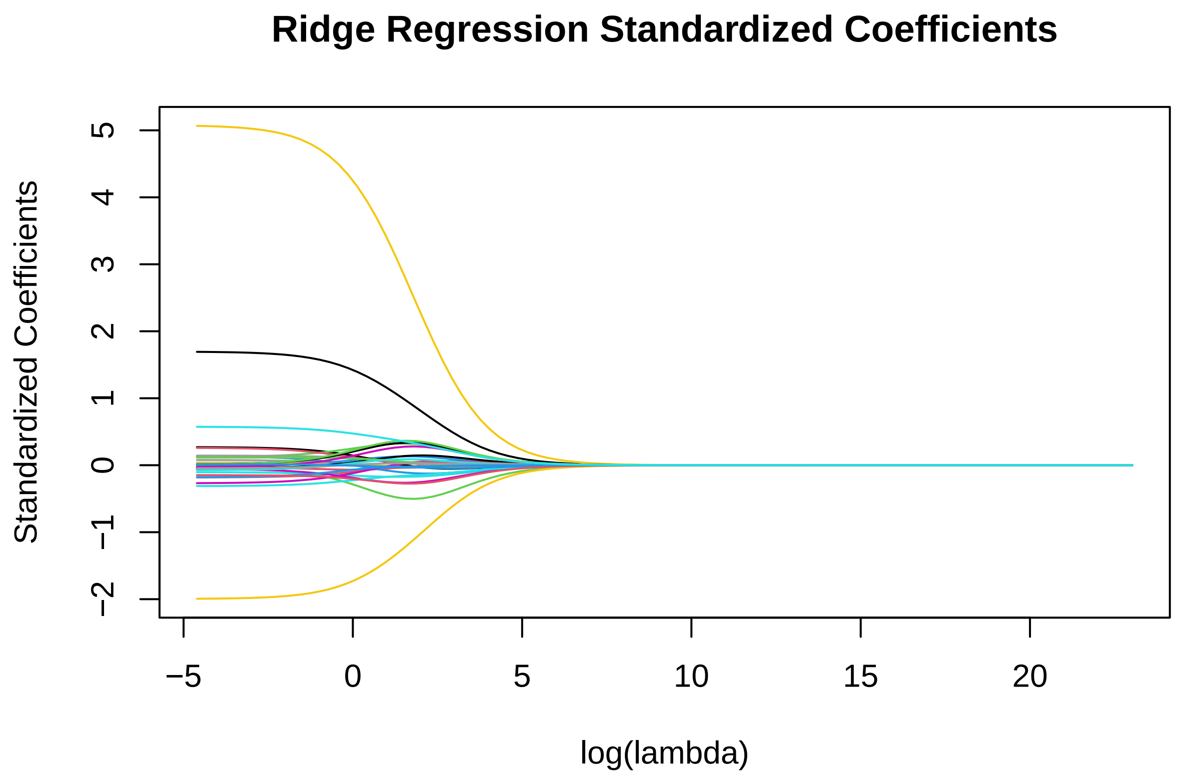
Part c
In this part of the exercise, using the subset of predictors identified in part (b), we will fit a ridge regression model on the training data with a range of values for the regularization parameter λ. We will use the `grid` function to get the range of λ values. Specifically, we will use the sequence of λ from Lab 7, defined as `10^seq(10, -2, length = 100)`. Furthermore, we will use 10-fold cross-validation to find the optimal λ value. Since the `glmnet` function automatically standardizes our predictors, we will not have to worry about standardizing our features before performing ridge regression. Lastly, we will use the training data that we created in part (b).

However, we will start by plotting the MSE from 10-fold cross-validation against $\log \lambda$. Since we have very large values of λ, this is why we will instead plot $\log \lambda$ on the horizontal axis. This is done below.



As seen above the optimal $\log \lambda$ value is quite small. In particular, using 10-fold cross validation it was calculated to be 0.5746816.

We will now generate a plot that resembles the left panel of Figure 6.4 in ISLR. Since we have very large values of λ, we will instead plot $\log \lambda$ on the horizontal axis. This is done below.



From this plot, we can see how the standardized coefficients change as the value of $\log \lambda$ increases. Since there are two standardized coefficients noticeably larger than the rest of the standardized coefficients when $\log \lambda$ is at its lowest value, this is why the plot is hard to read (of course this is also hard to read due to the large amounts of coefficients).

Part d

In this part of the exercise, we will find the value of λ that provides the smallest cross-validation error from part (c). Furthermore, we will find the test error of this ridge regression model. We will then compare this with the error from part (b)

As computed in part (c), using 10-fold cross validation, the optimal value of λ was calculated to be 0.5746816. We will now use this value to find the test error of this ridge regression model. This will be done in `R` using our test set from part (b).

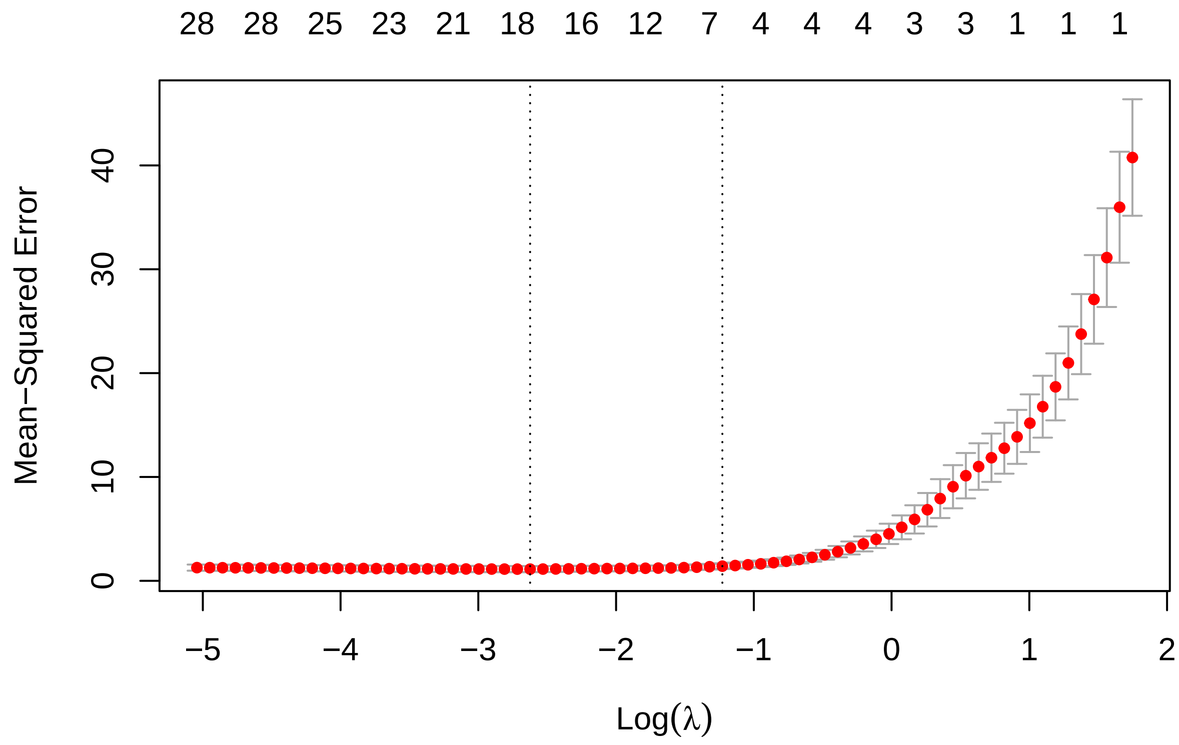
As calculated in `R` using the `predict` function, the test error for the ridge regression model using the optimal λ value was 3.0203645. Note that this is noticeably different than the error we found in part (b) for the model without the shrinkage procedure, which was calculated as 3.3061591. Therefore, ridge regression has successfully lowered the testing MSE.

Part e

In this part of the exercise, we will repeat part (c) and (d) using LASSO regression. At the end we will check and see which features, according to LASSO regression, have a non-zero estimated coefficient (using the optimal lambda value of course).

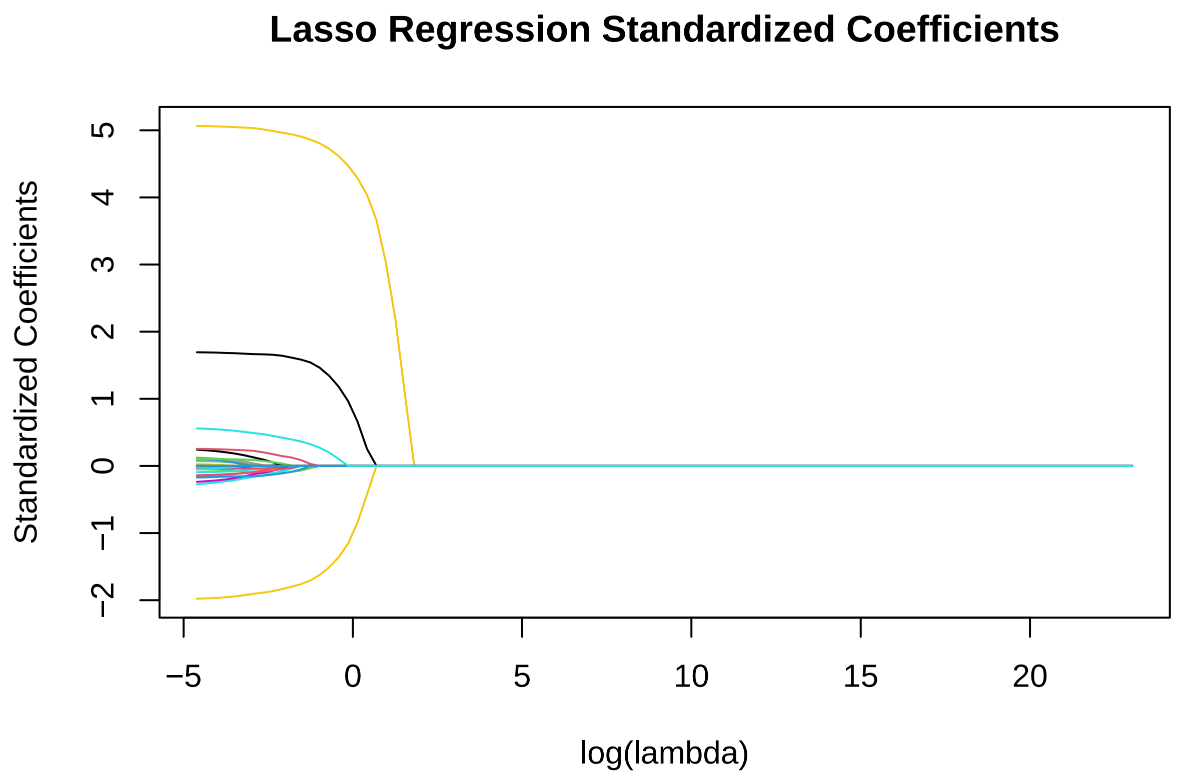
We will start by repeating part (c). In this part of the exercise, using the subset of predictors identified in part (b), we will fit a LASSO regression model on the training data with a range of values for the regularization parameter λ. We will use the `grid` function to get the range of λ values. Specifically, we will use the sequence of λ from Lab 7, defined as `10^seq(10, -2, length = 100)`. Furthermore, we will use 10-fold cross-validation to find the optimal λ value. Since the `glmnet` function automatically standardizes our predictors, we will not have to worry about standardizing our features before performing LASSO regression. Lastly, we will use the training data that we created in part (b).

However, we will start by plotting the MSE from 10-fold cross-validation against $\log \lambda$. Since we have very large values of λ, this is why we will instead plot $\log \lambda$ on the horizontal axis. This is done below.



As seen above the optimal $\log \lambda$ value is quite small. In particular, using 10-fold cross validation it was calculated to be 0.0725166.

We will now generate a plot that resembles the left panel of Figure 6.4 in ISLR. Since we have very large values of λ, we will instead plot $\log \lambda$ on the horizontal axis. This is done below.



From this plot, we can see how the standardized coefficients change as the value of $\log\lambda$ increases. Since there are two standardized coefficients noticeably larger than the rest of the standardized coefficients when $\log\lambda$ is at its lowest value, this is why the plot is hard to read (of course this is also hard to read due to the large amounts of coefficients).

We will now move on to repeating part (d). In this part of the exercise, we will find the value of λ that provides the smallest cross-validation error from part (c). Furthermore, we will find the test error of this LASSO regression model. We will then compare this with the error from part (b)

As computed in part (c), using 10-fold cross validation, the optimal value of λ was calculated to be 0.0725166. We will now use this value to find the test error of this ridge regression model. This will be done in `R` using our test set from part (b).

As calculated in `R` using the `predict` function, the test error for the LASSO regression model using the optimal λ value was 2.3014636. Note that this is noticeably smaller than the error we found in part (b) for the model without the shrinkage procedure, which was calculated as 3.3061591. It is also noticeably smaller than the error from the optimal ridge regression in part (d), which was calculated as 3.0203645. Therefore, LASSO regression has successfully lowered the testing MSE. This is expected since we have many predictors that were not part of the "true" generation process of y_i.

Lastly, we will now take a look at the coefficients from the LASSO regression using the optimal λ to see which of them are non-zero. This is done below.

```
## 30 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  0.977427907
## X1           1.662670900
## X2          -0.064038941
## X3          -0.108213009
## X4           .
## X5          -0.039872595
## X6           .
## X7           .
## X8           0.003592446
## X9           0.091420499
## X10         -0.035608620
## X11          .
## X12          .
## X13         -0.135608329
## X14         -0.098391713
## X15          5.010662818
## X16          0.008312161
## X17          0.002307431
## X18          0.200574691
## X19          0.072993420
## X20         -0.143014704
## X21          .
## X22          .
## X23         -1.885655445
## X24          .
## X25          .
## X26         -0.079985744
## X28          .
## X29          .
## X30          0.470299221
```

As can be seen above, there are 18 predictors in our LASSO regression model using the optimal λ value that had a non-zero estimated coefficient. These predictors include $X_1 - X_3$, X_5, X_7, $X_8 - X_{10}$, $X_{13} - X_{20}$, X_{23}, X_{26}, and X_{30}. It is important to note that LASSO selected all of the predictors that are truly important to y_i, but it also kept 14 predictors that are truly unrelated to y_i.