Homework #3

Corbin Charpentier

```r
#  include=FALSE}
rm(list=ls())

pp <- function(...) {
    print(paste0(...))
}
```

1. In this problem, we will make use of the Auto data set, which is part of the ISLR2 package.

Instructions:

You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

On this assignment, some of the problems involve random number generation. Be sure to set a random seed (using the command `set.seed()`*) before you begin.*

1. In this problem, we'll see a (very!!) simple simulated example where a least squares linear model is "too flexible".

(a) First, generate some data with $n = 100$ and $p = 10{,}000$ features, and a quantitative response, using the following `R` commands:

$$\mathtt{y} < -\mathtt{rnorm}(100)$$

$$\mathtt{x} < -\mathtt{matrix}(\mathtt{rnorm}(10000 * 100), \mathtt{ncol} = 10000)$$

Write out an expression for the model corresponding to this data generation procedure. For instance, it might look something like $Y = 2X_1 + 3X_2 + \epsilon, \epsilon \sim N(0, 1)$.

```r
y <- rnorm(100)
X <- matrix(rnorm(10000*100), ncol=10000)
df1 <- as.data.frame(X)
df1['y'] <- y
```

Let $Y, \beta, \epsilon \in \mathbb{R}^{100}$ and $X \in \mathbb{R}^{10000X100}$. Then the expression for the model corresponding to the above R code is

$$Y = X^T\beta + \epsilon$$

(b) What is the value of the irreducible error?

The irreducible error for each Y_i is ϵ, which is distributed $\epsilon\ N(0,1)$.

(c) Consider a very simple model-fitting procedure that just predicts 0 for every observation. That is, $\hat{f}(x) = 0$ for all x.

i. What is the bias of this procedure?

The bias of the error of model for a prediction is:

$$\text{Bias} = E\left[\hat{f}(x)\right] - f(x) = -f(x)$$

Since the data is standard-normally distributed, the bias of error is pretty close to 0.

ii. What is the variance of this procedure?

Since the model is constant, the variance is 0.

iii. What is the expected prediction error of this procedure?

Generally, expected error is:

$$\text{Err}(x) = (E\left[\hat{f}(x)\right] - f(x))^2 + E\left[(\hat{f}(x) - E[\hat{f}(x)])^2\right] + \sigma_e^2$$

Plugging in the parameters of this model, we get:

$$\text{Err}(x) = (f(x))^2 + 0 + \epsilon$$

Given the training data, the expected error will tend towards ϵ.

iv. Use the validation set approach to estimate the test error of this procedure. What answer do you get?

Since the model is constant and 0, the estimated error depends entirely on how the training and test sets are partitioned, that is, ϵ.

```r
splitData <- function(df, trainProportion) {
    df <- df %>% mutate(id = row_number())

    set.seed(1)
    train <- sample_frac(df, trainProportion)
    test <- anti_join(df, train, by='id')

    df <- df %>% mutate(id=NULL)

    return(list(train=train, test=test))
}

split <- splitData(df1, 0.5)
model <- lm(y ~ 0, split$train)

pp("Test error: ", mean((df1$y - predict(model, split$test))^2))
```

```
## [1] "Test error: 0.810550927470378"
```

v. Comment on your answers to (iii) and (iv). Do your answers agree with each other? Explain.

They do agree. In the end, the error for each strategy depends on random chance, i.e. $/epsilon$.

(d) Now use the validation set approach to estimate the test error of a least squares linear model using $X_1, ..., X_{10,000}$ to predict Y . What is the estimated test error?

In practice, we tend to over-estimate the bias because training set is smaller than the entire data set. For example, a training set consisting of only two data points could generate an extremely biased model. Here, however, since the model is constant,

```
model <- lm(y ~ ., split$train)

pp("Test error: ", mean((df1$y - predict(model, split$test))^2))
```

```
## [1] "Test error: 30.0570770718004"
```

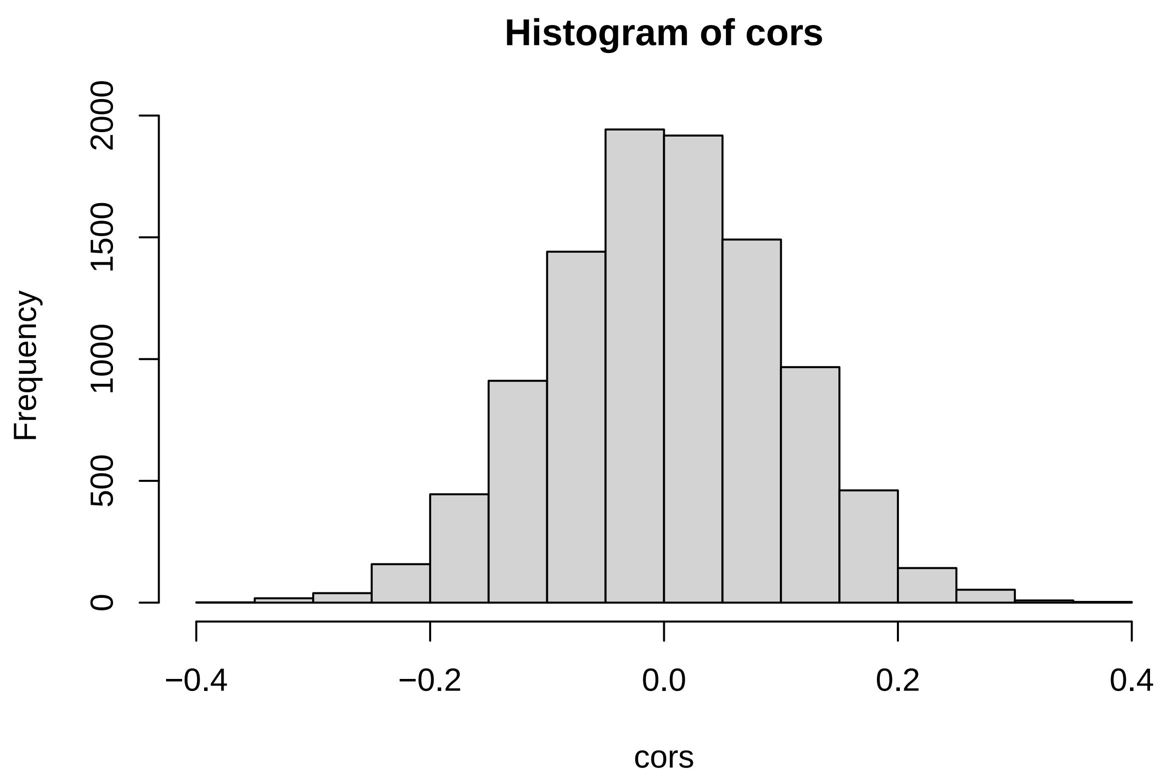
(e) Comment on your answers to (c) and (d). Which of the two procedures has a smaller estimated test error? higher bias? higher variance? In answering this question, be sure to think carefully about how the data were generated.

The model for (c) has lower error because, while having high bias, the model exactly represents the mean of the population ($\mu = 0$), and since (c) is constant, it has lower variance of the two. That $ p > n$ does not matter for (c) since it uses none of the independent variables to predict the response. Model (d), however, very likely over fits the training data, hence the higher test error. Another way to say this is that (d) has much greater variance.

2. In lecture during Week 5, we discussed "Option 1" and "Option 2": two possible ways to perform the validation set approach for a modeling strategy where you identify the q features most correlated with the response, and then fit a least squares linear model to predict the response using just those q features. If you missed that lecture, then please familiarize yourself with the lecture notes (posted on Canvas) before you continue. Here, we are going to continue to work with the simulated data from the previous problem, in order to illustrate the problem with Option 1.

(a) Calculate the correlation between each feature and the response. Make a histogram of these correlations. What are the values of the 10 largest absolute correlations?

```
xcols <- !names(df1) %in% c("y")
cors <- cor(df1[, xcols], df1["y"])

hist(cors)
```



```r
top10Cors <- head(sort(abs(cors), decreasing=TRUE), 10)
pp("Top 10 most correlated: ")
```

```
## [1] "Top 10 most correlated: "
```

```r
pp(top10Cors)
```

```
##  [1] "0.398328069907005" "0.394921380308979" "0.368630382335929"
##  [4] "0.352901242404994" "0.347135747162808" "0.346625342259251"
##  [7] "0.345284483302047" "0.338422783279816" "0.338313294683231"
## [10] "0.337169169163078"
```

(b) Now try out "Option 1" with $q = 10$. What is the estimated test error?

```r
calcTopN <- function(df, n) {
    dfCors <- as.data.frame(cor(df[, xcols], df["y"]))
    dfCors <- dfCors %>% mutate(y=abs(y)) %>% arrange(desc(y)) %>% head(10)
    labs <- labels(dfCors)[[1]]
    return(df %>% select(append(labs, "y")))
}

df01 <- calcTopN(df1, 10)

split01 <- splitData(df01, 0.5)

model01 <- lm(y ~ ., data=split01$train)

pp("Test error: ", mean((split01$test$y - predict(model01, split01$test))^2))
```

```
## [1] "Test error: 0.47523262446835"
```

(c) Now try out "Option 2" with $q = 10$. What is the estimated test error?

```
train02 <- calcTopN(split$train, 10)

model02 <- lm(y ~ ., data=train02)

test02 <- select(split$test, colnames(train02))
pp("Test error: ", mean((test02$y - predict(model02, test02))^2))
```

```
## [1] "Test error: 1.17497072952889"
```

(d) Comment on your results in (b) and (c). How does this relate to the discussion of Option 1 versus Option 2 from lecture? Explain how you can see that Option 1 gave you a useless (i.e. misleading, inaccurate, wrong) estimate of the test error.

Option 1 undermines the entire motivation for having split training and testing sets. We want to ensure the model generalizes to unseen data. By calculating correlations with the whole data set (and then using the results of that calculating for feature selection), we effectively expose the model to the test data during training. This is why Option 1 has lower error than Option 2–it cheated! Option 2 does not suffer this problem. Option 2's feature selection is performed exclusively on the training data.

3. In this problem, you will analyze a (real, not simulated) dataset of your choice with a quantitative response Y , and $p \geq 50$ quantitative predictors.

(a) Describe the data. Where did you get it from? What is the meaning of the response, and what are the meanings of the predictors?

```
dfNews <- read.csv("OnlineNewsPopularity/OnlineNewsPopularity.csv")
head(dfNews)
```

```
##                                                              url timedelta
## 1   http://mashable.com/2013/01/07/amazon-instant-video-browser/       731
## 2    http://mashable.com/2013/01/07/ap-samsung-sponsored-tweets/       731
## 3 http://mashable.com/2013/01/07/apple-40-billion-app-downloads/       731
## 4      http://mashable.com/2013/01/07/astronaut-notre-dame-bcs/       731
## 5              http://mashable.com/2013/01/07/att-u-verse-apps/       731
## 6              http://mashable.com/2013/01/07/beewi-smart-toys/       731
##   n_tokens_title n_tokens_content n_unique_tokens n_non_stop_words
## 1             12              219       0.6635945                1
## 2              9              255       0.6047431                1
## 3              9              211       0.5751295                1
## 4              9              531       0.5037879                1
## 5             13             1072       0.4156456                1
## 6             10              370       0.5598886                1
##   n_non_stop_unique_tokens num_hrefs num_self_hrefs num_imgs num_videos
## 1                0.8153846         4              2        1          0
## 2                0.7919463         3              1        1          0
## 3                0.6638655         3              1        1          0
## 4                0.6656347         9              0        1          0
## 5                0.5408895        19             19       20          0
## 6                0.6981982         2              2        0          0
##   average_token_length num_keywords data_channel_is_lifestyle
## 1             4.680365            5                         0
## 2             4.913725            4                         0
## 3             4.393365            6                         0
## 4             4.404896            7                         0
## 5             4.682836            7                         0
## 6             4.359459            9                         0
##   data_channel_is_entertainment data_channel_is_bus data_channel_is_socmed
## 1                             1                   0                      0
## 2                             0                   1                      0
## 3                             0                   1                      0
## 4                             1                   0                      0
## 5                             0                   0                      0
## 6                             0                   0                      0
##   data_channel_is_tech data_channel_is_world kw_min_min kw_max_min kw_avg_min
## 1                    0                     0          0          0          0
## 2                    0                     0          0          0          0
## 3                    0                     0          0          0          0
## 4                    0                     0          0          0          0
## 5                    1                     0          0          0          0
## 6                    1                     0          0          0          0
##   kw_min_max kw_max_max kw_avg_max kw_min_avg kw_max_avg kw_avg_avg
## 1          0          0          0          0          0          0
## 2          0          0          0          0          0          0
## 3          0          0          0          0          0          0
## 4          0          0          0          0          0          0
## 5          0          0          0          0          0          0
## 6          0          0          0          0          0          0
##   self_reference_min_shares self_reference_max_shares
## 1                       496                       496
## 2                         0                         0
## 3                       918                       918
## 4                         0                         0
## 5                       545                     16000
## 6                      8500                      8500
##   self_reference_avg_sharess weekday_is_monday weekday_is_tuesday
## 1                    496.000                 1                  0
## 2                      0.000                 1                  0
## 3                    918.000                 1                  0
## 4                      0.000                 1                  0
## 5                   3151.158                 1                  0
## 6                   8500.000                 1                  0
##   weekday_is_wednesday weekday_is_thursday weekday_is_friday
## 1                    0                   0                 0
## 2                    0                   0                 0
## 3                    0                   0                 0
## 4                    0                   0                 0
## 5                    0                   0                 0
## 6                    0                   0                 0
##   weekday_is_saturday weekday_is_sunday is_weekend     LDA_00     LDA_01
## 1                   0                 0          0 0.50033120 0.37827893
## 2                   0                 0          0 0.79975569 0.05004668
## 3                   0                 0          0 0.21779229 0.03333446
## 4                   0                 0          0 0.02857322 0.41929964
## 5                   0                  0          0 0.02863281 0.02879355
## 6                   0                  0          0 0.02224528 0.30671758
##       LDA_02     LDA_03     LDA_04 global_subjectivity
## 1 0.04000468 0.04126265 0.04012254           0.5216171
## 2 0.05009625 0.05010067 0.05000071           0.3412458
## 3 0.03335142 0.03333354 0.68218829           0.7022222
## 4 0.49465083 0.02890472 0.02857160           0.4298497
## 5 0.02857518 0.02857168 0.88542678           0.5135021
## 6 0.02223128 0.02222429 0.62658158           0.4374086
##   global_sentiment_polarity global_rate_positive_words
## 1                0.09256198                 0.04566210
## 2                0.14894781                 0.04313725
## 3                0.32333333                 0.05687204
## 4                0.10070467                 0.04143126
## 5                0.28100348                 0.07462687
## 6                0.07118419                 0.02972973
##   global_rate_negative_words rate_positive_words rate_negative_words
## 1                0.013698630           0.7692308           0.2307692
## 2                0.015686275           0.7333333           0.2666667
## 3                0.009478673           0.8571429           0.1428571
## 4                0.020715631           0.6666667           0.3333333
## 5                0.012126866           0.8602151           0.1397849
## 6                0.027027027           0.5238095           0.4761905
##   avg_positive_polarity min_positive_polarity max_positive_polarity
## 1             0.3786364            0.10000000                   0.7
## 2             0.2869146            0.03333333                   0.7
## 3             0.4958333            0.10000000                   1.0
## 4             0.3859652            0.13636364                   0.8
## 5             0.4111274            0.03333333                   1.0
## 6             0.3506100            0.13636364                   0.6
##   avg_negative_polarity min_negative_polarity max_negative_polarity
## 1            -0.3500000                -0.600            -0.2000000
## 2            -0.1187500                -0.125            -0.1000000
## 3            -0.4666667                -0.800            -0.1333333
## 4            -0.3696970                -0.600            -0.1666667
## 5            -0.2201923                -0.500            -0.0500000
## 6            -0.1950000                -0.400            -0.1000000
##   title_subjectivity title_sentiment_polarity abs_title_subjectivity
## 1          0.5000000               -0.1875000             0.00000000
## 2          0.0000000                0.0000000             0.50000000
## 3          0.0000000                0.0000000             0.50000000
## 4          0.0000000                0.0000000             0.50000000
## 5          0.4545455                0.1363636             0.04545455
## 6          0.6428571                0.2142857             0.14285714
##   abs_title_sentiment_polarity shares
## 1                    0.1875000    593
## 2                    0.0000000    711
## 3                    0.0000000   1500
## 4                    0.0000000   1200
## 5                    0.1363636    505
## 6                    0.2142857    855
```

```
dfNews <- dfNews %>% mutate(url=NULL, timedelta=NULL, n_non_stop_words=NULL)
```

(b) Fit a least squares linear model to the data, and provide an estimate of the test error. (Explain how you got this estimate.)

```r
# V128 is the response
library(regclass)
predictAndError <- function(ys, xs, model) {
    mean((ys - predict(model, xs))^2)
}

tmp <- cor(dfNews)
tmp[upper.tri(tmp)] <- 0
diag(tmp) <- 0

dfNews1 <-
  dfNews[, !apply(tmp, 2, function(x) any(abs(x) > 0.90, na.rm = TRUE))]

splitNews <- splitData(dfNews1, 0.7)
model <- lm(shares ~ ., data=splitNews$train)

model$rank
```

```
## [1] 54
```

```r
print(nrow(dfNews1))
```

```
## [1] 39644
```

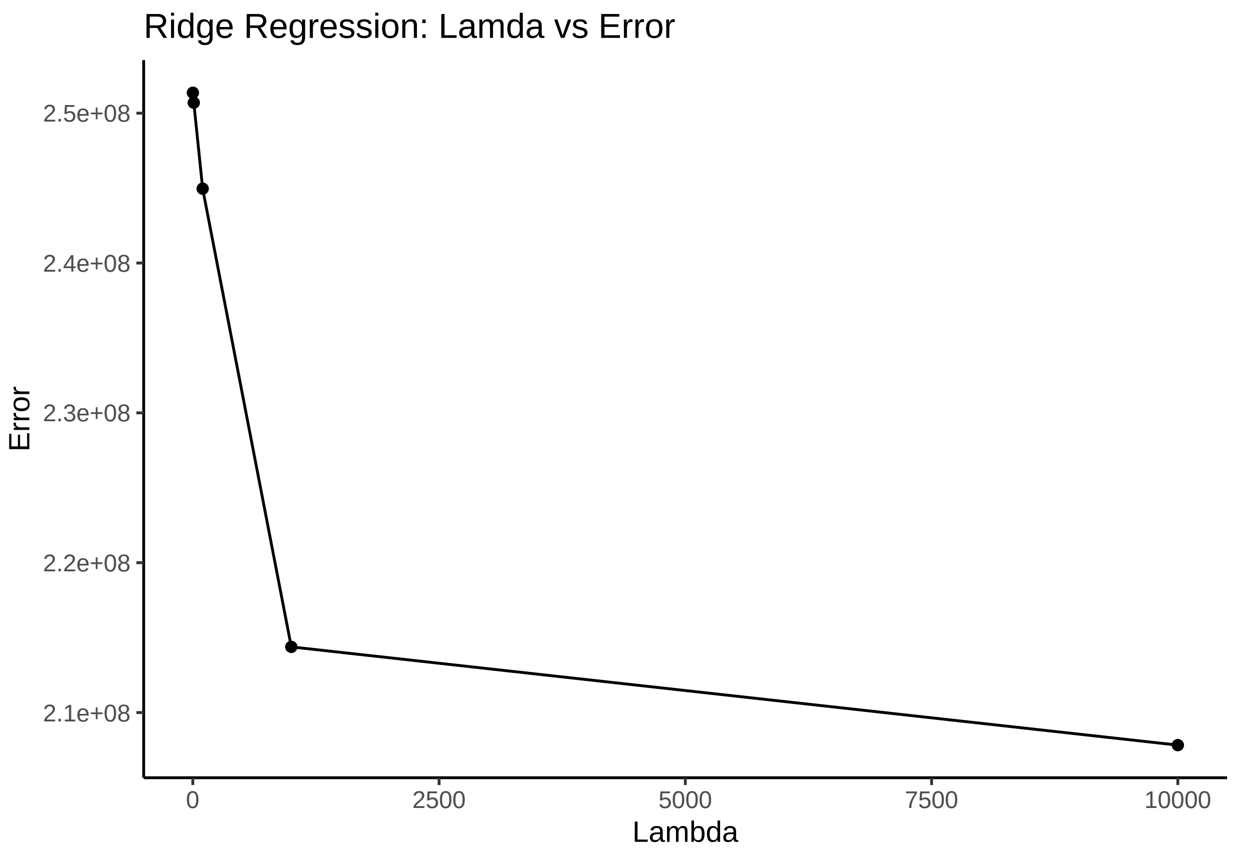
```r
pp("Test error: ", predictAndError(splitNews$test$shares, splitNews$test, model))
```

```
## [1] "Test error: 255437942.09694"
```

```r
# I do not know why I'm getting the rank-deficient warning.
```

(c) Fit a ridge regression model to the data, with a range of values of the tuning parameter λ. Make a plot like the left-hand panel of Figure 6.4 in the textbook.

```r
library(glmnetUtils)


trainXss <- as.matrix(splitNews$train %>% select(-shares))
trainYs  <- as.matrix(splitNews$train["shares"])

i <- 1
m <- matrix(nrow=5, ncol=2)
for(lam in list(1, 10, 100, 1000, 10000)) {
    model <- glmnet(shares ~ ., data=splitNews$train, alpha=0, lambda=lam)
    err <- predictAndError(splitNews$test$shares, splitNews$test, model)

    m[i, ] <- c(lam, err)
    i <- i + 1
}

dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Ridge Regression: Lamda vs Error",x="Lambda", y = "Error") +
    theme_classic()
```



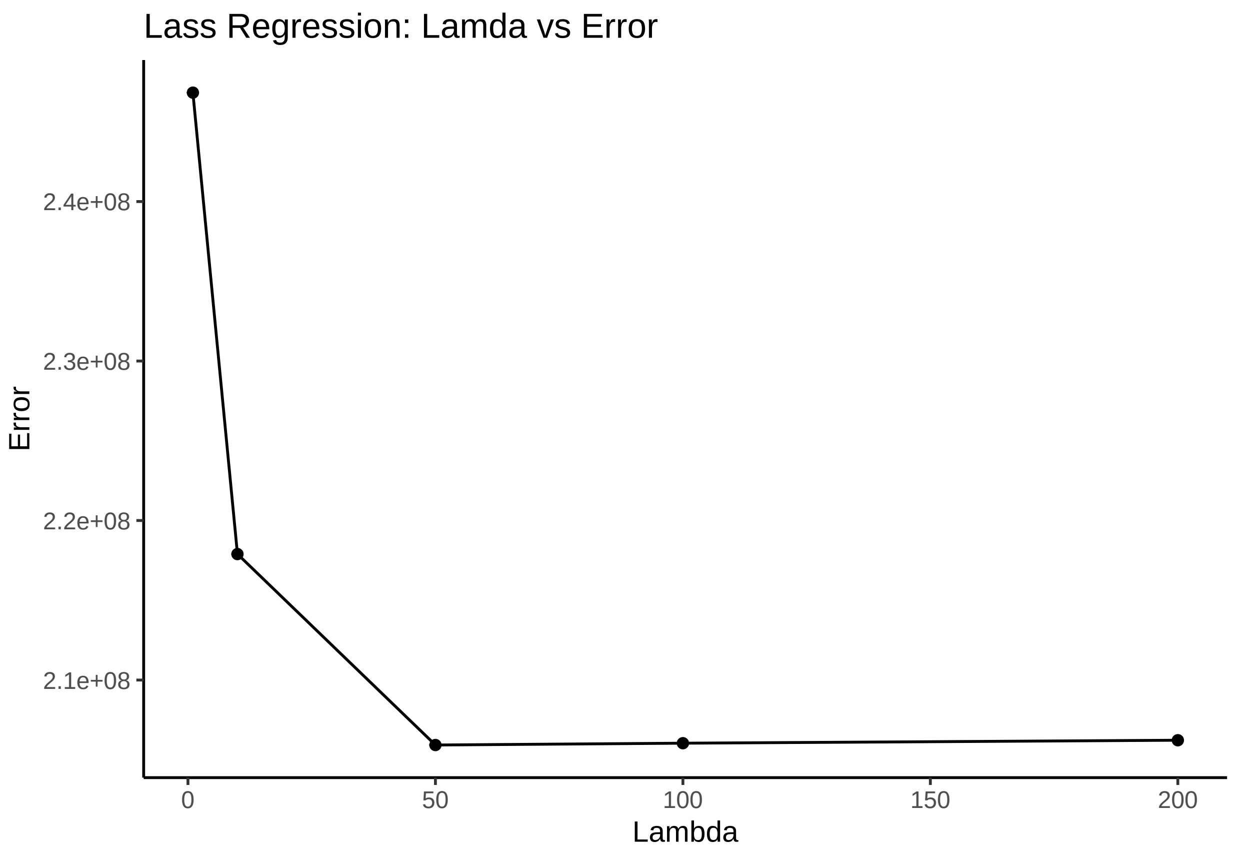
(d) What value of λ in the ridge regression model provides the smallest estimated test error? Report this estimate of test error. (Also, explain how you estimated test error.)

The optimal error for this data set and split appears to reside at $0.01 < \alpha < 0.05$. Error was estimated as follows:

1. Split data set into training and testing partitions, 70|30 split.
2. Train model on training data
3. Use predictions of trained model on test data to calculate Mean Squared Error.

(e) Repeat (c), but for a lasso model.

```
i <- 1
m <- matrix(nrow=5, ncol=2)
for(lam in list(1, 10, 50, 100, 200)) {
    model <- glmnet(shares ~ ., data=splitNews$train, alpha=1, lambda=lam)
    err <- predictAndError(splitNews$test$shares, splitNews$test, model)

    m[i, ] <- c(lam, err)
    i <- i + 1
}

dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Lass Regression: Lamda vs Error",x="Lambda", y = "Error") +
    theme_classic()
```



(f) Repeat (d), but for a lasso model. Which features are included in this lasso model?

All 56 features are included in this model.

4. In this problem, you may use the function in the `glmnet` package that performs cross-validation. Consider using the `Auto` data set to predict `mpg` using polynomial functions of `horsepower` in a least squares linear regression.

(a) Perform the validation set approach, and produce a plot like the one in the right-hand panel of Figure 5.2 of the textbook. Your answer won't look *exactly* the same as the results in Figure 5.2, since you'll be starting with a different random seed. Discuss your findings. What degree polynomial is best, and why?

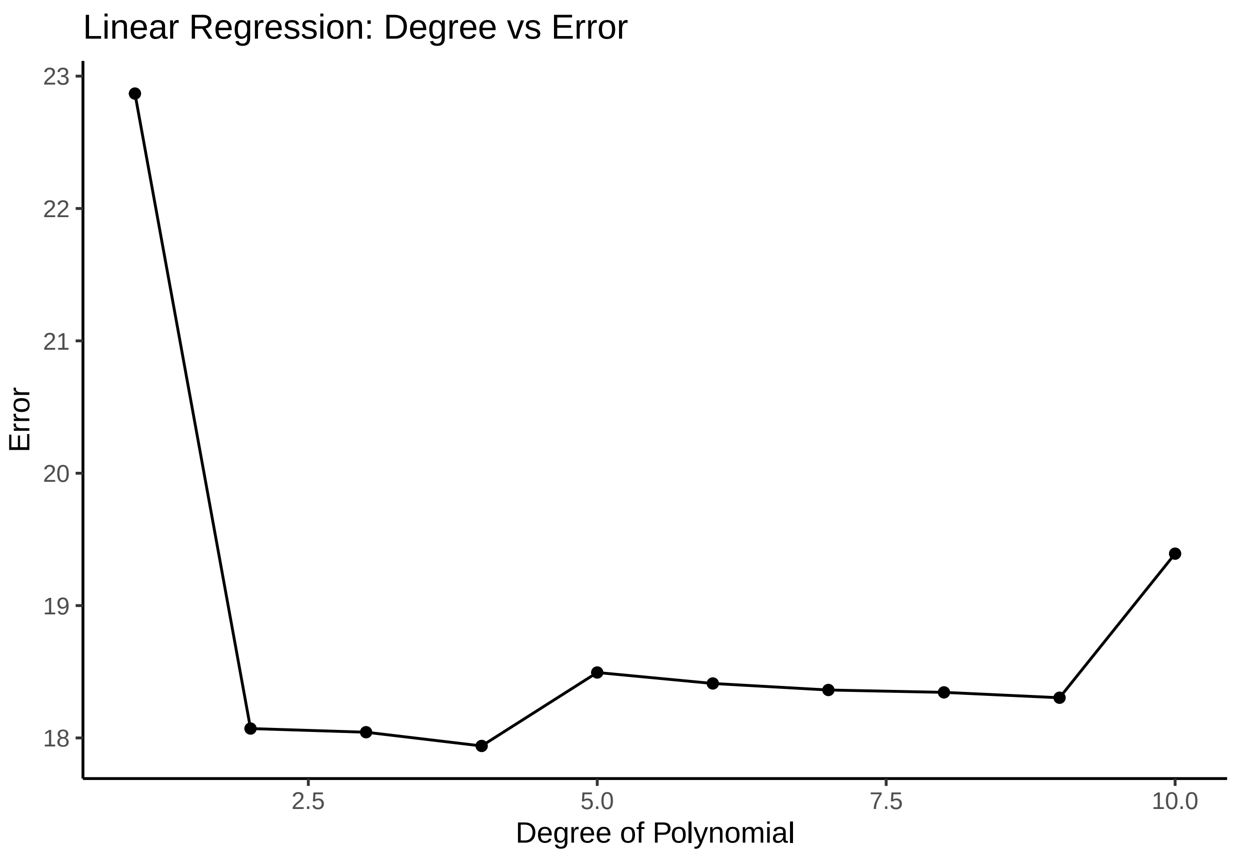
```
library(ISLR2)
dfAuto <- Auto

head(dfAuto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

```
splitAuto <- splitData(dfAuto, 0.7)

m <- matrix(nrow=10, ncol=2)
i <- 1
for(d in 1:10) {
    model <- lm(mpg ~ poly(horsepower, d), data=splitAuto$train)
    err <- predictAndError(splitAuto$test$mpg, splitAuto$test, model)
    print(err)
    m[i, ] <- c(d, err)
    i <- i + 1
}
```

```
## [1] 22.86834
## [1] 18.07092
## [1] 18.04324
## [1] 17.93951
## [1] 18.49449
## [1] 18.41162
## [1] 18.36217
## [1] 18.3444
## [1] 18.30377
## [1] 19.39223
```

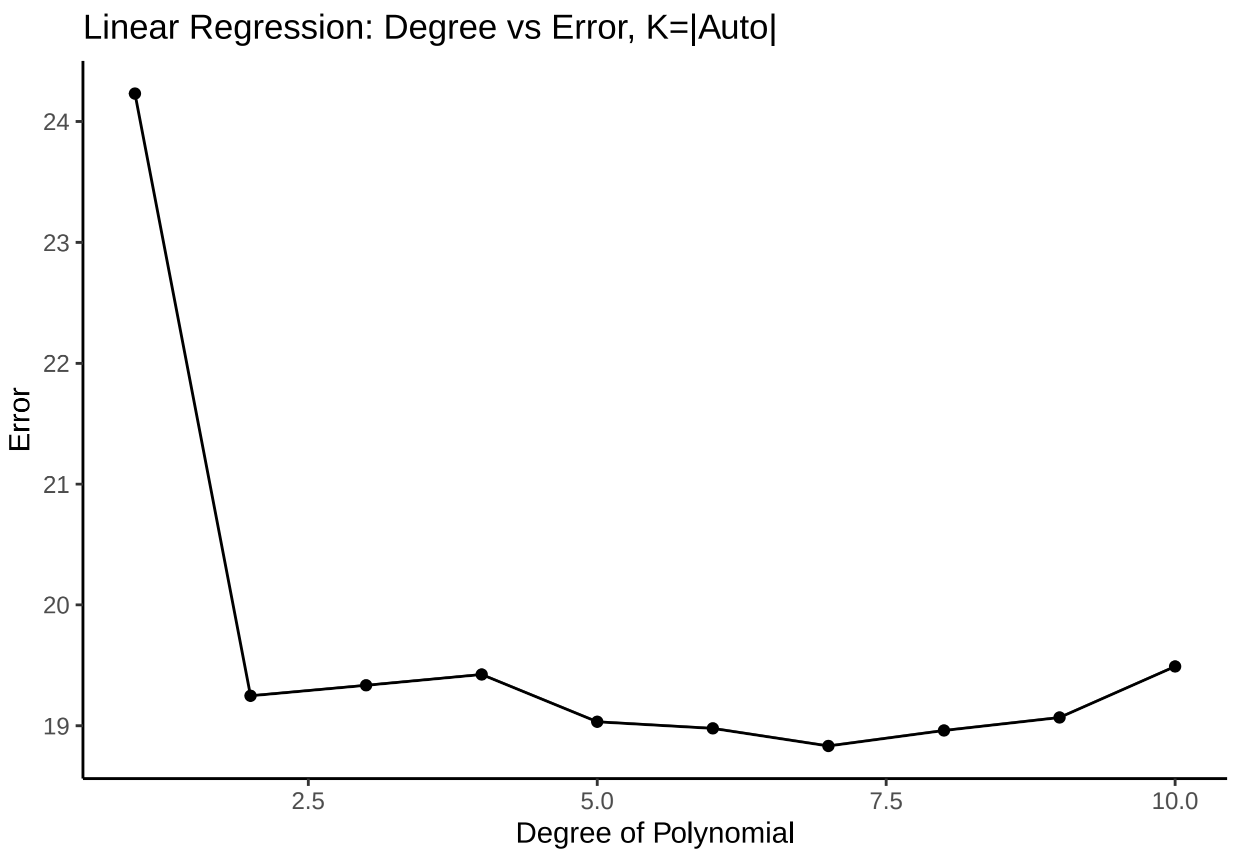
```
dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Linear Regression: Degree vs Error",x="Degree of Polynomial", y = "Error") +
    theme_classic()
```



The degree that minimizes error is somewhere between 2 and 4, inclusive (with this particular seed, it was 4). This implies that the true relationship between mpg and horsepower is more quadratic than linear.

(b) Perform leave-one-out cross-validation, and produce a plot like the one in the left-hand panel of Figure 5.4 of the textbook. Discuss your findings. What degree polynomial is best, and why?

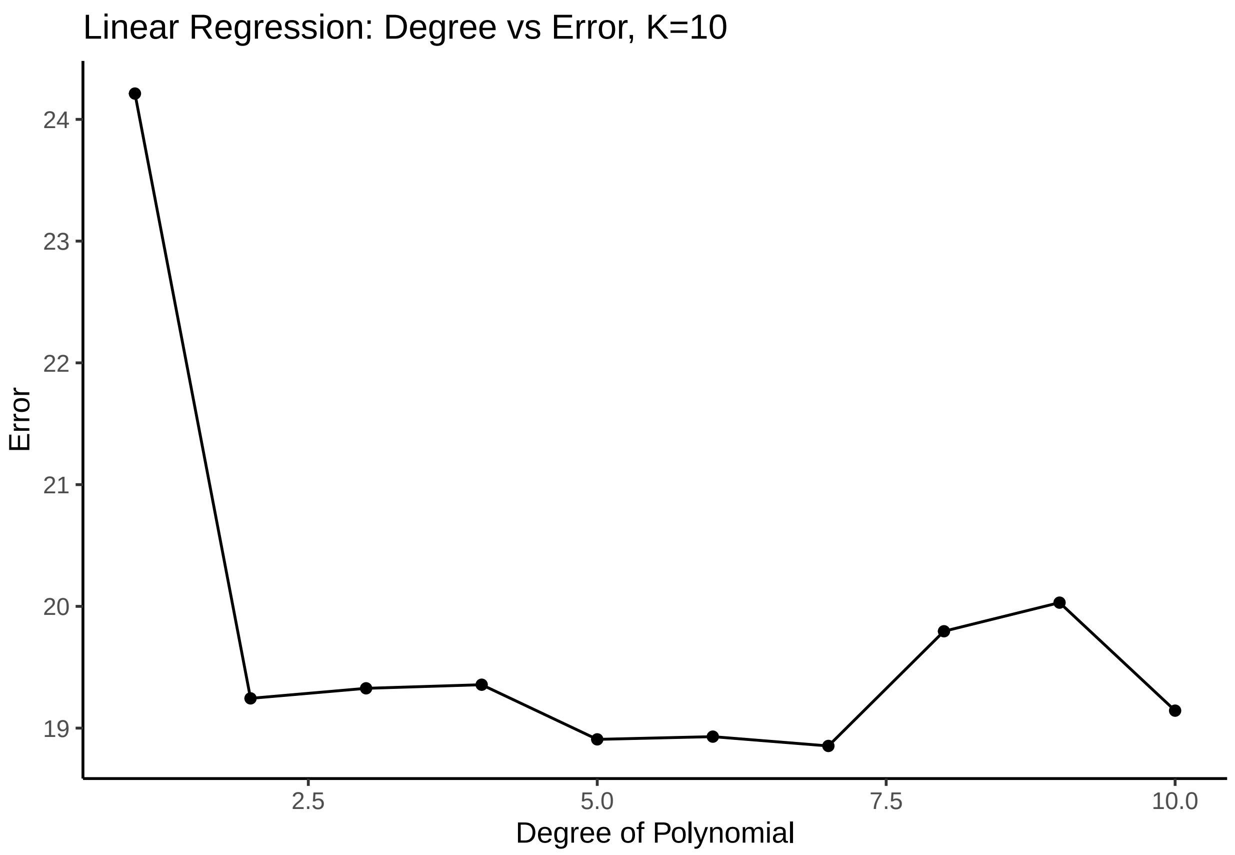
```r
library(boot)
m <- matrix(nrow=10, ncol=2)
i <- 1
for(d in 1:10) {
    model <- glm(mpg ~ poly(horsepower, d), data=dfAuto)
    err <- cv.glm(dfAuto, model, K=nrow(dfAuto))$delta[1]
    m[i, ] <- c(d, err)
    i <- i + 1
}

dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Linear Regression: Degree vs Error, K=|Auto|",x="Degree of Polynomial", y = "Error") +
    theme_classic()
```



We see the error initially minimized at degree two and then oscillating around that until finding the least error at degree. I think the higher degree models (above $d = 2$) have slightly lower error because they are overfitting. Indeed, they include the second degree term already.

(c) Perform 10-fold cross-validation, and produce a plot like the one in the right-hand panel of Figure 5.4 of the textbook. Discuss your findings. What degree polynomial is best, and why?

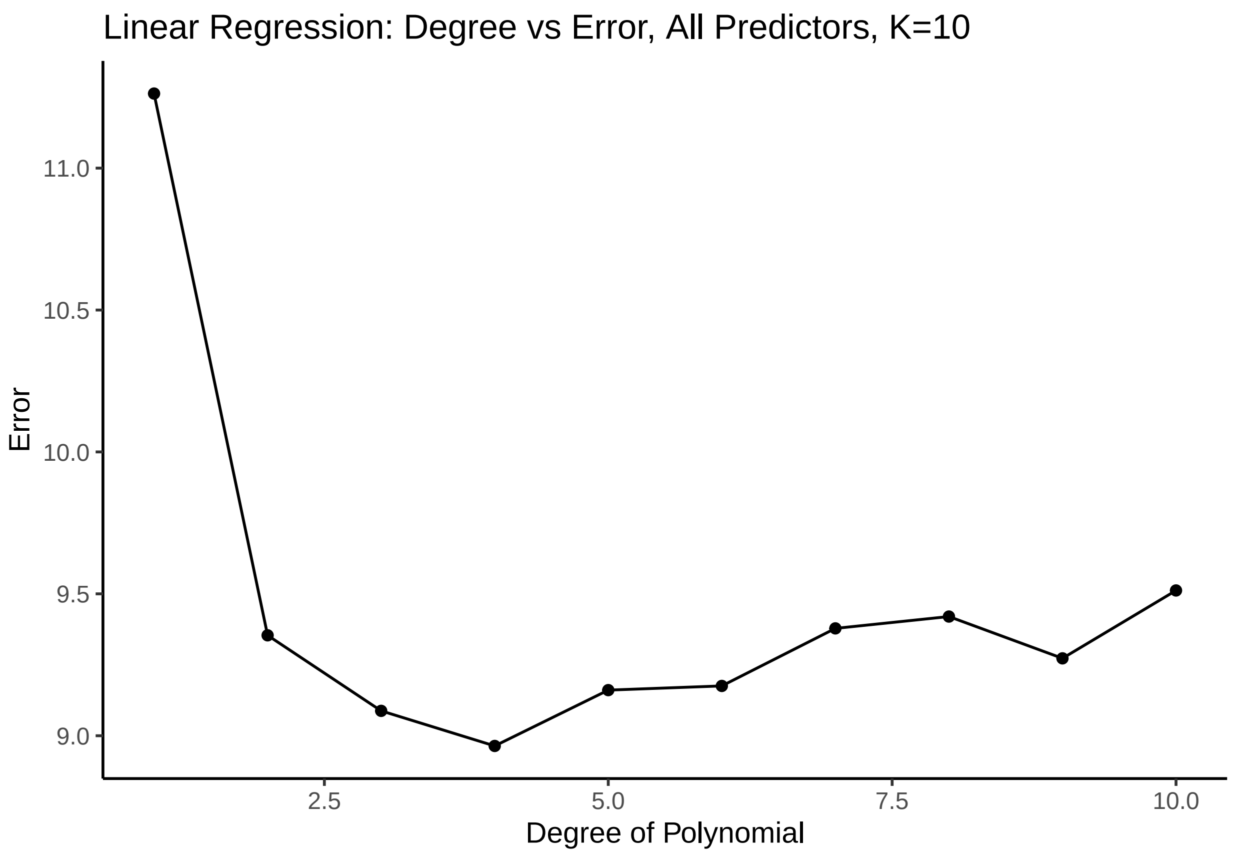
```
m <- matrix(nrow=10, ncol=2)
i <- 1
for(d in 1:10) {
    model <- glm(mpg ~ poly(horsepower, d), data=dfAuto)
    err <- cv.glm(dfAuto, model, K=10)$delta[1]
    m[i, ] <- c(d, err)
    i <- i + 1
}

dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Linear Regression: Degree vs Error, K=10",x="Degree of Polynomial", y = "Error") +
    theme_classic()
```



Results here largely align with what we saw in (b), althought slightly more exaggerated because of the smaller folds.

(d) Fit a least squares linear model to predict mpg using polynomials of degrees from 1 to 10, using all available observations. Make a plot showing "Degree of Polynomial" on the *x*-axis, and "Training Set Mean Squared Error" on the *y*-axis. Discuss your findings.

```
m <- matrix(nrow=10, ncol=2)
i <- 1
for(d in 1:10) {
    model <- glm(mpg ~  cylinders + displacement + weight + acceleration + year + factor(origin) + poly
    err <- cv.glm(dfAuto, model, K=10)$delta[1]
    m[i, ] <- c(d, err)
    i <- i + 1
}

dfResults <- as.data.frame(m)

ggplot(data=dfResults, aes(x=V1, y=V2, group=1)) +
    geom_line() +
    geom_point() +
    labs(title="Linear Regression: Degree vs Error, All Predictors, K=10",x="Degree of Polynomial", y =
    theme_classic()
```



The primary difference between this and the (a) through (c) is the increase in error as degree increases (after the precipitous drop), although we still see the error largely leveling off after $d = 2$.

(e) Fit a least squares linear model to predict `mpg` using a degree-10 polynomial, using all available observations. Using the `summary` command in `R`, examine the output. Comment on the output, and discuss how this relates to your findings in (a)–(d).

```
model <- lm(mpg ~  cylinders + displacement + weight + acceleration + year + factor(origin) + poly(horse

summary(model)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + weight + acceleration +
##     year + factor(origin) + poly(horsepower, 10), data = dfAuto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3299 -1.6241  0.0355  1.3281 11.8177
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.827e+01  4.073e+00  -4.485 9.69e-06 ***
## cylinders              -7.970e-02  3.732e-01  -0.214 0.830980
## displacement           -2.491e-03  8.023e-03  -0.310 0.756401
## weight                 -3.765e-03  7.028e-04  -5.358 1.47e-07 ***
## acceleration          -2.531e-01  1.019e-01  -2.483 0.013484 *
## year                   7.528e-01  4.842e-02  15.547  < 2e-16 ***
## factor(origin)2        1.289e+00  5.486e-01   2.349 0.019354 *
## factor(origin)3        1.758e+00  5.187e-01   3.390 0.000774 ***
## poly(horsepower, 10)1  -3.801e+01  1.071e+01  -3.548 0.000437 ***
## poly(horsepower, 10)2   3.263e+01  3.932e+00   8.299 1.94e-15 ***
## poly(horsepower, 10)3  -1.259e+01  3.510e+00  -3.588 0.000378 ***
## poly(horsepower, 10)4  -2.839e+00  3.194e+00  -0.889 0.374609
## poly(horsepower, 10)5   4.402e+00  3.136e+00   1.404 0.161146
## poly(horsepower, 10)6   5.010e-01  3.202e+00   0.156 0.875747
## poly(horsepower, 10)7  -1.316e+00  3.041e+00  -0.433 0.665581
## poly(horsepower, 10)8  -1.146e+00  3.114e+00  -0.368 0.713148
## poly(horsepower, 10)9   1.504e+00  3.036e+00   0.496 0.620487
## poly(horsepower, 10)10  1.116e+00  2.998e+00   0.372 0.709928
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.963 on 374 degrees of freedom
## Multiple R-squared:  0.8622, Adjusted R-squared:  0.8559
## F-statistic: 137.6 on 17 and 374 DF,  p-value: < 2.2e-16
```

Polynomials with $d > 3$ are insignificant! By far the most significant degree is $d = 2$. This corresponds with our findings above. Adding higher degree polynomials amounts to adding noise.

5. ***Extra Credit!*** Let's consider doing least squares and ridge regression under a very simple setting, in which $p = 1$, and $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i = 0$. We consider regression without an intercept. (It's usually a bad idea to do regression without an intercept, but if our feature and response each have mean zero, then it is okay to do this!)

Hint: For this problem, you might want to brush up on some basic properties of means and variances! For instance, if $Cov(Z, W) = 0$, then $Var(Z + W) = Var(Z) + Var(W)$. And if a is a constant, then $Var(aW) = a^2 Var(W)$, and $Var(a + W) = Var(W)$.

(a) The least squares solution is the value of $\beta \in \mathbb{R}$ that minimizes $\sum_{i=1}^{n}(y_i - \beta x_i)^2$. Write out an analytical (closed-form) expression for this least squares solution. Your answer should be a function of $x_1, ..., x_n$ and $y_1, ..., y_n$. *Hint: Calculus!!*

(b) For a given value of λ, the ridge regression solution minimizes $\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda\beta^2$. Write out an analytical (closed-form) expression for the ridge regression solution, in terms of $x_1, ..., x_n$ and $y_1, ..., y_n$ and λ.

(c) Suppose that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero, and X is fixed (non-random). What is the expectation of the least squares estimator from (a)? Is it biased or unbiased?

(d) Suppose again that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero, and X is fixed (non-random). What is the expectation of the ridge regression estimator from (b)? Is it biased or unbiased? Explain how the bias changes as a function of λ.

(e) Suppose that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero and variance σ^2, and X is fixed (non-random), and also $Cov(\epsilon_i, \epsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the least squares estimator from (a)?

(f) Suppose that the true data-generating model is $Y = 3X + \epsilon$, where ϵ has mean zero and variance σ^2, and X is fixed (non-random), and also $Cov(\epsilon_i, \epsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the ridge estimator from (b)? How does the variance change as a function of λ?

(g) In light of your answers to parts (d) and (f), argue that λ in ridge regression allows us to control model complexity by trading off bias for variance.