Assignment 3

Statistics and Data Science 365/565

Due: October 9 (before midnight)

I worked with Jackson Simon on Problem 1, Part 2 and Problem 2, Part B.

This homework treats variable selection and shrinkage. Problem 1 is a conceptual question, and Problems 2 and 3 are applied problems. Throughout this assignment, remember that for a vector $\beta \in \mathbb{R}^p$,

$$\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$$

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

Problem 1 (25 points) LASSO Risk

Suppose we estimate β in a linear regression model by minimizing

$$\widehat{\beta}^{(s)} = \underset{\beta \in \mathbb{R}^{p+1}}{\arg\min} \|Y - X\beta\|_2^2 \text{ subject to } \|\beta\|_1 \leq s$$

for a particular value of s.

Part 1.

For (a) through (e), indicate which of (i) through (v) is correct, and justify your answer.

Part a.

As we increase s from 0, the training RSS of $\widehat{\beta}^{(s)}$ will:

(i) increase initially, and then eventually start decreasing in an inverted U shape

(ii) decrease initially, and then eventually start increasing in a U shape

(iii) steadily increase

(iv) steadily decrease

(v) remain constant

Solution: (iv) steadily decrease As s increases from zero, we are allowing the size of our β_j coefficients to increases. Starting at s zero, we are only using the intercept as our $\widehat{\beta}$ and forcing other coefficients to be zero. Adding more parameters to our model by allowing the other coefficients to be nonzero, we are reducing RSS. With training data, adding more and more parameters into our model will continously decrease RSS

Part b.

Repeat (a) for the test RSS of $\widehat{\beta}^{(s)}$.

Solution: (ii) decrease initially, and then eventually start increase in a U shape As stated above, s increasing from zero, we are adding more parameters to our model by allowing the other coefficients to be nonzero, we are reducing RSS. However, since we fit our model to the training data, increasing s and allowing too many parameters we will begin to overfit our test data and RSS will begin to increase again. Think of how 1-NN fits the training data perfectly but overfits the test data.

Part c.

Repeat (a) for the variance of $\widehat{\beta}^{(s)}$.

Solution: (iii) steadily increase As our model becomes more complex (i.e. more predictor variables used), variance increases. So as s increases and we include more parameters in our model, variance will steadily increase.

Part d.

Repeat (a) for the squared bias of $\widehat{\beta}^{(s)}$.

Solution: (iv) steadily decrease Contrary to above, as our model becomes more complex, squared bias decreases. So as s increases and we include more parameters in our model, squared bias will steadily increase. For s approaching infinity (or lambda=0), then we have an unbiased estimator for the OLS model.

Part e.

Repeat (a) for the irreducible error of $\widehat{\beta}^{(s)}$.

Solution: (v) remain constant As the name suggests, the irreducible error is thus irreducible error. Our data is inherently noisy as thus is modeled as $y_i = f(X_i) + \epsilon_i$ where epsilon is irreducible error. No matter what function we choose for $f(X_i)$ (i.e. what parameters and their coefficients we choose to include by changing the values of s), these errors will remain constant.

Part 2.

Now we will compute the lasso estimates on simulated data. That is, we will find the minimizer

$$\widehat{\beta}^{(\lambda)} = \underset{\beta \in \mathbb{R}^{p+1}}{\arg\min} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

where λ is some fixed constant.

Part a.

What do you think is the relationship between s in Part 1 and λ? As we increase λ from 0, what will happen to the bias and variance of our estimator $\widehat{\beta}^{(\lambda)}$?

Solution: Comparing this equation to the one in Part 1, we can see that large values of λ correspond to small values of s. As we increase the weight of the penalty (λ), we want to decrease the size of the beta coefficients (s) to compensate for the penalty increase and continue to minimize RSS. Thus as we increase λ from 0, we will be including less and less parameters in our model according to the following:

$$\widehat{\beta} = \begin{cases} Y-\lambda, & \text{if } Y > \lambda \\ 0, & \text{if } Y \leq |\lambda| \\ Y+\lambda, & \text{if } Y < -\lambda \end{cases}$$

Therefore model complexity will decrease, thus variance will decrease and squared bias will increase (the opposite of what we see when s increases).

Part b.

Generate independent predictors, X_1, X_2, X_3, X_4, and store them in the $n \times 4$ matrix X, as follows:

```
n <- 1000
p <- 4
X <- matrix(rnorm(n*p, mean=1), nrow = n, ncol = p)
```

Also generate an outcome vector $Y \in \mathbb{R}^n$ according to the model:

$$Y_i = 17 + (.005) * X_{i1} + 117 * X_{i2} + .6 * X_{i3} + 52 * X_{i4} + \epsilon_i$$

where the $\epsilon_i \sim N(0,1)$.

For various values of λ (between 0 and 0.1), fit a lasso model (using the `glmnet` package) to the generated data, and store each fitted lasso estimate $\widehat{\beta}^{(\lambda)}$. For B times, repeat the process of generating data and, for the same set of values for lambda, fitting a lasso and storing the result. (It is up to you to decide how big B should be. i.e. How many repetitions to do.) To estimate the bias and variance, let

$$\widehat{\mathbb{E}}\widehat{\beta}^{(\lambda)} = \frac{1}{B}\sum_{b=1}^{B}\widehat{\beta}^{(b,\lambda)}$$

where $\widehat{\beta}^{(b,\lambda)}$ represents the vector of lasso coefficients on round b and penalization parameter λ.

Estimate the squared bias via:

$$\widehat{\text{Bias}^2}(\widehat{\beta}^{(\lambda)}) = \|\widehat{\mathbb{E}}\widehat{\beta}^{(\lambda)} - \beta\|_2^2$$

And estimate the variance via:

$$\widehat{\text{Var}}(\widehat{\beta}^{(\lambda)}) = \frac{1}{B}\sum_{b=1}^{B}\|\widehat{\beta}^{(b,\lambda)} - \widehat{\mathbb{E}}\widehat{\beta}^{(\lambda)}\|_2^2$$

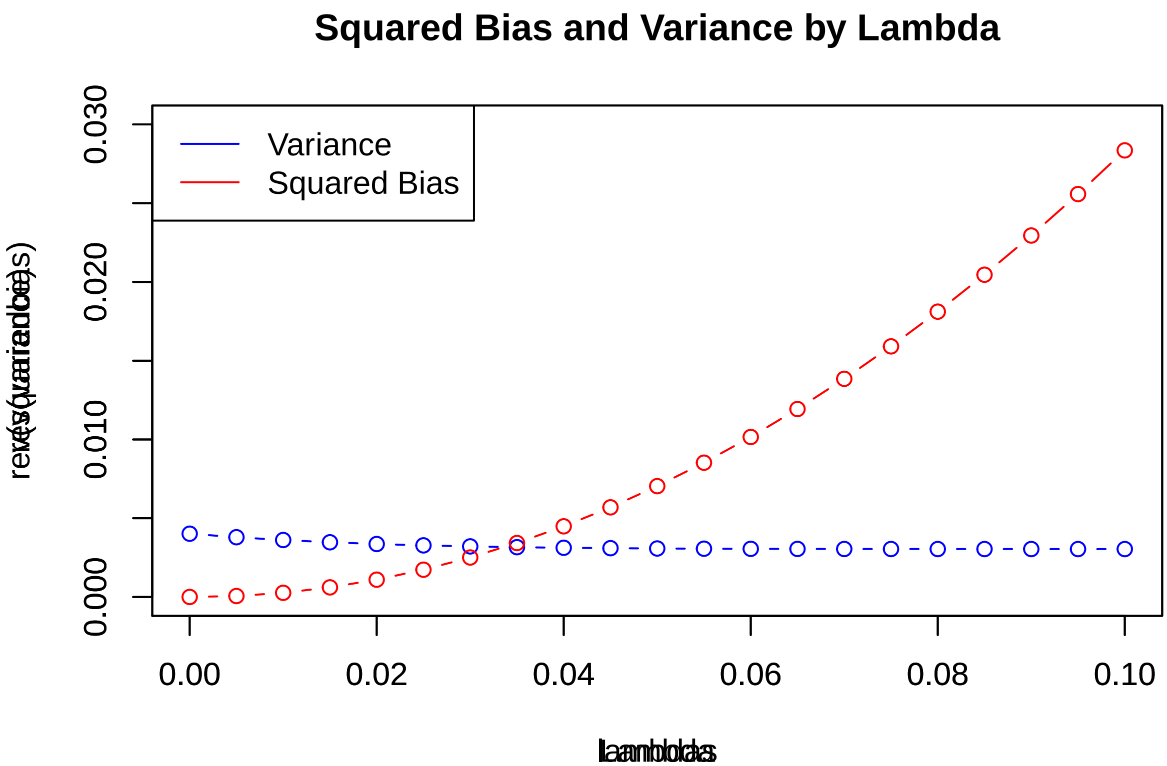
Plot the estimated bias and variance for each λ. For a large enough number of repetitions B, you should see a clear pattern. Summarize your results and compare them to Part (a).

```
#Generate our y values, we are going to need to do this B times, lets just set B to 100 first.
B <- 1000
lambdas <- seq(0,0.1,.005)
coefficients <- array(data = rep(0,B*length(lambdas)*4), dim =c(B,length(lambdas),4))
beta_vec <- as.vector(c(.005,117,.6,52))

for (i in 1:B) {
  errors <- rnorm(1000)
  y <- 17 + X%*%beta_vec + errors
  mod <- glmnet(X, y, family="gaussian", standardize=TRUE, alpha=1, lambda=lambdas)
  for (j in c(1:length(lambdas))) {
    coefficients[i,j,] <- coef(mod)[2:5,j]
  }
}
```

```
squaredbias <- rep(0,length(lambdas))
variance <- rep(0,length(lambdas))
expectations <- array(data = rep(0,length(lambdas)*4), dim = c(length(lambdas),4))
for (i in 1:length(lambdas)) {
  expectations[i,] <- colMeans(coefficients[,i,])
  squaredbias[i] <- sum((expectations[i,] - beta_vec)^2)
  variance[i] <- sum(colMeans(t(apply(coefficients[,i,], 1, function(x) (x-expectations[i,])^2))))
}

plot(lambdas,rev(variance),col="blue",type="b",ylim=c(0,.03),main="Squared Bias and Variance by Lambda")
par(new=T)
plot(lambdas,rev(squaredbias),ylim=c(0,.03),col="red",type="b",xlab="Lambda")
legend("topleft",legend=c("Variance", "Squared Bias"),col=c("blue","red"),lty=1)
```



Solution: Acording to my results here, squared bias increase and variance decreases as lambda increases (s decreases). This is in agreement with what I argue in part 1, that variance constantly increases and squared bias constantly decreases.

Problem 2 (15 points) Boston Housing Values

The value of homes in Boston is a dataset included in the `MASS` library in R with the name `Boston`. You may use the `glmnet` package to fit the lasso and ridge regression models.

Part a.

Start by doing some data exploration. What are n and p in this dataset? Are there any missing data points, outliers, or explanatory variables that seem highly correlated with each other? What variables seem most closely associated with the outcome variable, `medv`? You are not limited to these questions; examine whatever you think is interesting and reasonable. Show plots, accompanied with comments, that illustrate your findings.

```
dim(Boston)
```

```
## [1] 506  14
```

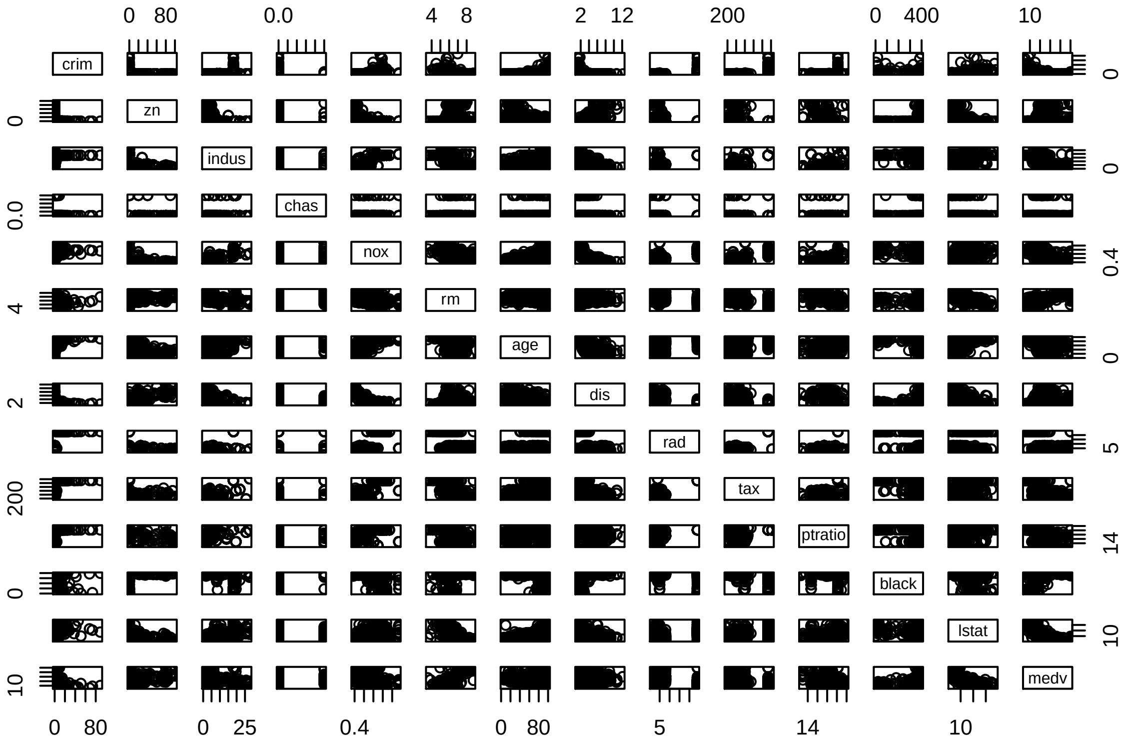
```
summary(is.na(Boston))
```

```
##     crim            zn            indus            chas        
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:506       FALSE:506       FALSE:506       FALSE:506      
##     nox             rm             age             dis         
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:506       FALSE:506       FALSE:506       FALSE:506      
##     rad            tax            ptratio          black        
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical  
##  FALSE:506       FALSE:506       FALSE:506       FALSE:506      
##    lstat            medv        
##  Mode :logical   Mode :logical  
##  FALSE:506       FALSE:506      
```

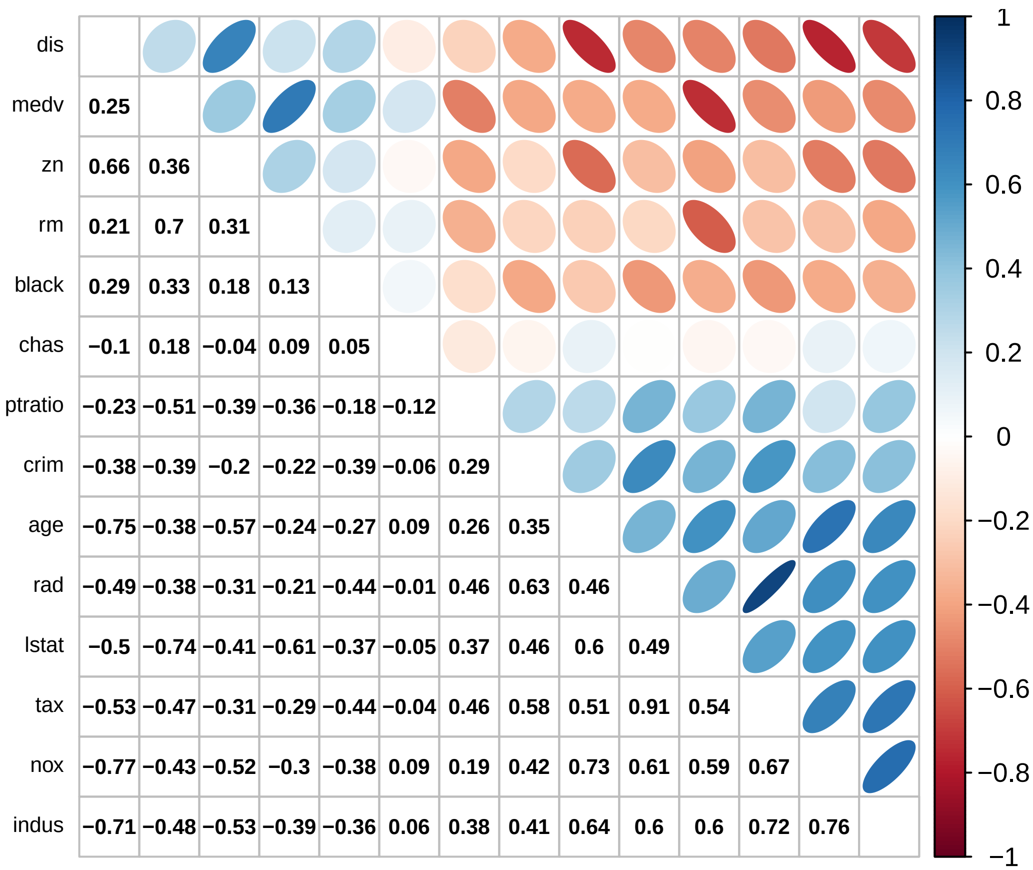
```
summary(Boston)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00  
```

```
plot(Boston)
```




```
cor1 <- round(cor(Boston, use = "pairwise.complete.obs"),2)
corrplot.mixed(cor1,lower.col="black", upper = "ellipse", tl.col = "black", number.cex=.7,
               order = "FPC", tl.pos = "lt", tl.cex=.7, sig.level = .05)
```



Comments: By using the `dim()` function, we see that n is 506 (number of observations) and p is 14 (number of predictor variables). Additionally, there are no missing values in the dataset. Then using the `corrplot.mixed` function from the `corrplot` library, we see that our strongest associations between outcome variable `medv` and the predictor variables is a 0.7 R-squared with the numbers of rooms (`rm`) and a -0.74 correlation with the percent in the "lower status" of the populations (`lstat`). These seem to make sense, I am surprised to see that there is a strong negative correlation (-0.51) with the pupil-to-teacher ratio in local schools (`ptratio`). It is worth noting that many predictor variables are correlated with each other. For example, `tax` and `rad` (index of highway accessibility) are strongly positively correlated (0.91) while `dis` and `indus` are strongly negatively correlated (-0.71), this second example suggests that the largest employment centers are retail oriented. Observing the summary data for the Boston housing data, I do not see many extreme values.

Part b.

Randomly sample 100 rows of the Boston dataset to use as the test set, and use the rest for training. Fit lasso and ridge regression models to predict `medv` with all other variables as predictors. For various values of λ, use cross-validation to select an optimal values to use in the lasso and ridge regression models. (You may use the `cv.glmnet()` in the `glmnet` package to do the cross-validation.) Also, plot the model coefficients against the values of λ. How does each coefficient change as λ increases in the two models? How is this behavior similar or different between the lasso and ridge regression approaches?

```
set.seed(1)
testrows <- sample(nrow(Boston),100)
test <- Boston[testrows,]
train <- Boston[-testrows,]

#lasso
cvfit <- cv.glmnet(x=as.matrix(train[-14]),y=as.matrix(train[14]),standardize=TRUE,nfold=10,alpha=1)
cvfit$lambda.min
```

```
## [1] 0.01773275
```

```
predict(cvfit, newx = as.matrix(test[-14]), s = "lambda.min")
```

```
##             1
## 135 13.428197
## 188 33.042812
## 289 27.138334
## 457 12.834164
## 102 25.373309
## 451 16.752445
## 473 21.595768
## 330 24.817806
## 314 25.256494
## 31  11.916388
## 103 20.328959
## 88  25.636672
## 340 21.055558
## 190 34.347536
## 379 15.280124
## 245 16.865574
## 352 21.780879
## 486 21.466260
## 186 24.534903
## 492 14.185122
```

```
## 455 15.619387
## 496 16.729627
## 316 20.426500
## 61  18.003028
## 129 18.720548
## 488 20.333664
## 7   23.244214
## 184 30.484882
## 416 10.114886
## 163 40.626795
## 230 30.587811
## 285 31.672446
## 234 36.974534
## 89  30.635964
## 391 16.238724
## 315 25.381856
## 374  5.456154
## 51  21.599244
## 339 21.934629
## 193 33.138876
## 383 12.588280
## 301 31.036182
## 364 19.730106
## 257 36.992520
## 491  4.478407
## 464 21.573125
## 11  19.827419
## 220 29.788818
## 336 20.652702
## 317 17.811675
## 218 28.039414
## 392 16.498874
## 199 35.018057
## 111 20.466985
## 32  18.216110
## 45  23.120256
## 143 15.112473
## 233 37.862886
## 297 27.568242
## 182 27.203705
## 408 18.843133
## 131 19.806944
## 204 41.163340
## 148  8.619078
## 288 26.912300
## 114 20.730302
## 211 22.849600
## 337 20.052932
## 37  22.018248
## 466 17.087189
## 443 17.876000
## 366 12.547373
## 151 20.706581
## 145  8.787821
```

```
## 206 22.383808
## 385  2.826338
## 372 23.659503
## 168 23.147581
## 333 23.712177
## 411 15.328508
## 481 22.552406
## 303 28.644318
## 170 26.538121
## 138 19.135789
## 320 21.323181
## 86  27.740210
## 299 29.258463
## 469 16.382184
## 485 18.655429
## 60  21.011262
## 100 32.287963
## 25  15.919708
## 266 27.458186
## 362 18.151930
## 321 24.743018
## 328 19.534411
## 187 35.610790
## 429 14.171556
## 331 22.382446
## 247 20.464967
```

```r
#ridge
cvfit2 <- cv.glmnet(x=as.matrix(train[-14]),y=as.matrix(train[14]),standardize=TRUE,nfold=10,alpha=0)
cvfit2$lambda.min
```

```
## [1] 0.7499609
```

```r
predict(cvfit2, newx = as.matrix(test[-14]), s = "lambda.min")
```

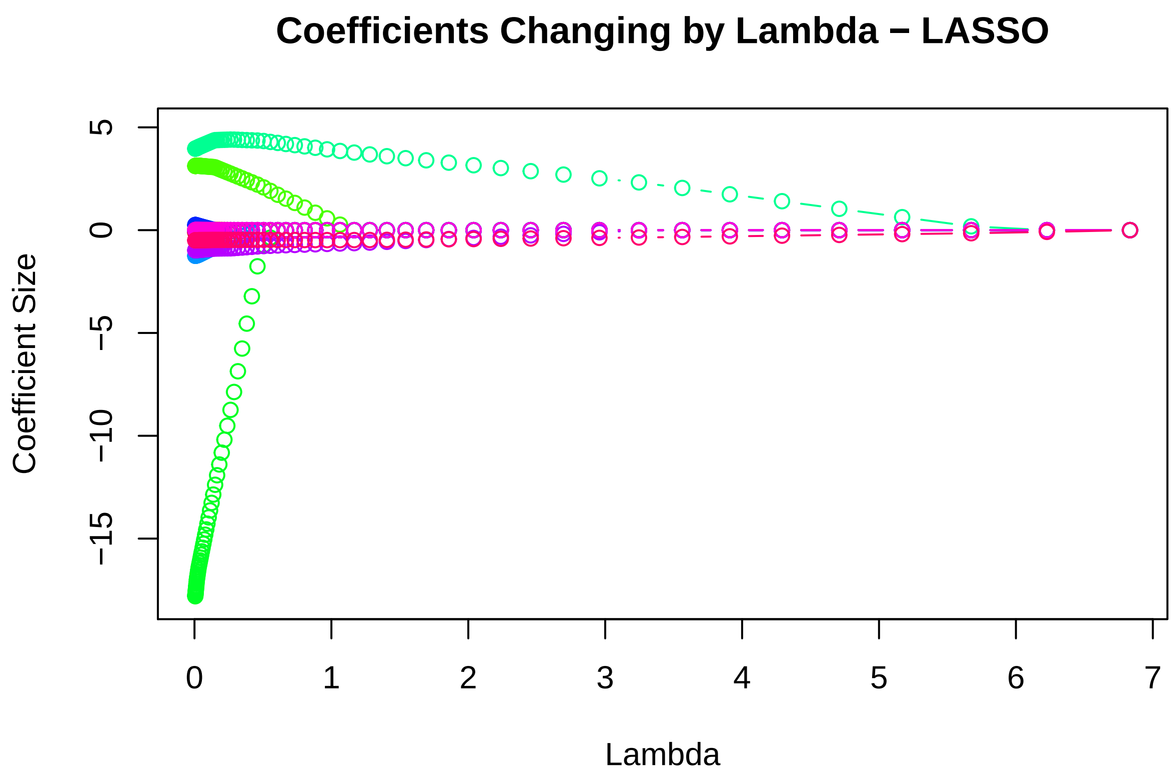
```
##             1
## 135 14.065919
## 188 32.439413
## 289 26.809812
## 457 12.943005
## 102 25.473553
## 451 16.811247
## 473 21.074537
## 330 25.551243
## 314 25.012835
## 31  12.884089
## 103 20.385685
## 88  25.219023
## 340 21.154120
## 190 33.963301
## 379 15.385854
## 245 17.400515
## 352 23.341650
## 486 21.133595
## 186 24.163133
## 492 15.580229
```

```
## 455 15.817688
## 496 17.378313
## 316 20.512539
## 61  18.267057
## 129 19.211275
## 488 19.756576
## 7   23.444163
## 184 29.565731
## 416 10.536844
## 163 40.018625
## 230 29.810279
## 285 31.685748
## 234 36.373414
## 89  30.373916
## 391 16.031372
## 315 25.337309
## 374  5.734321
## 51  21.822012
## 339 22.003145
## 193 33.214111
## 383 12.613615
## 301 31.015032
## 364 19.885319
## 257 36.079083
## 491  6.343983
## 464 21.439839
## 11  20.642092
## 220 29.352433
## 336 21.041362
## 317 18.284488
## 218 27.475722
## 392 16.519279
## 199 34.940209
## 111 20.800415
## 32  18.696790
## 45  23.182934
## 143 16.590046
## 233 37.244496
## 297 27.297927
## 182 26.460052
## 408 18.002708
## 131 20.242091
## 204 39.798027
## 148 10.059654
## 288 26.590124
## 114 20.991302
## 211 22.979552
## 337 20.257692
## 37  21.764018
## 466 16.943828
## 443 17.902941
## 366 11.516250
## 151 21.521131
## 145 10.177929
```

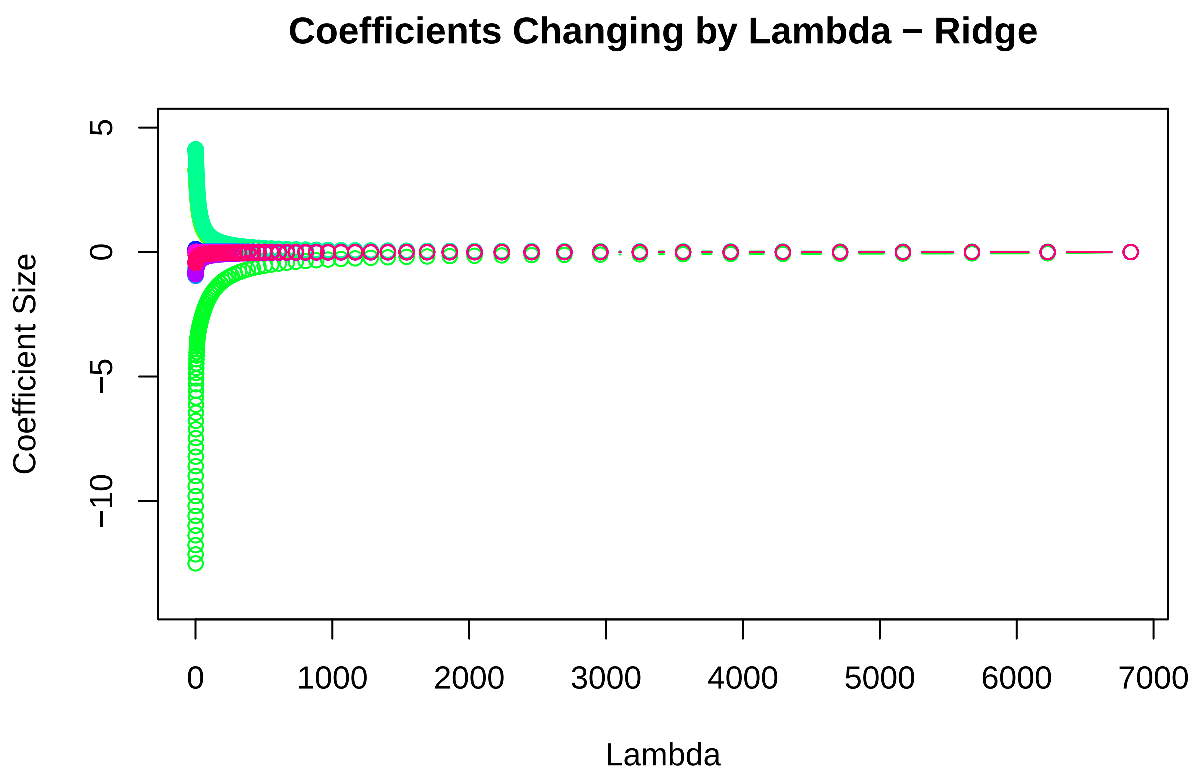
```
## 206 22.246113
## 385  3.142551
## 372 22.572811
## 168 22.800943
## 333 24.122567
## 411 14.471592
## 481 21.790508
## 303 28.214483
## 170 26.147989
## 138 19.599097
## 320 21.606944
## 86  27.491503
## 299 29.190184
## 469 16.125460
## 485 18.346520
## 60  21.066863
## 100 32.060089
## 25  16.634002
## 266 26.754371
## 362 18.286173
## 321 24.705911
## 328 20.028764
## 187 35.029155
## 429 14.165937
## 331 23.325692
## 247 21.057087
```

```
#Color Vector
colors <- rainbow(n=length(Boston))

#plot lasso coefficients by lambda value
plot(x=as.vector(unlist(cvfit$glmnet.fit[5])), y=as.vector(coef(cvfit$glmnet.fit)[2,]),
     xlab = "Lambda", ylab="Coefficient Size", main="Coefficients Changing by Lambda - LASSO",
     type = "b",ylim=c(-18,5),col=colors[1])
for (i in 3:length(Boston)) {
  lines(x=as.vector(unlist(cvfit$glmnet.fit[5])), y=as.vector(coef(cvfit$glmnet.fit)[i,]),
        type = "b",ylim=c(-20,30),col=colors[i])
}
```




```r
#plot ridge coefficients by lambda value
plot(as.vector(unlist(cvfit2$glmnet.fit[5])), as.vector(coef(cvfit2$glmnet.fit)[2,]),
     xlab = "Lambda", ylab="Coefficient Size", main="Coefficients Changing by Lambda - Ridge",
     type = "b",ylim=c(-14,5),col=colors[1])
for (i in 3:length(Boston)) {
  lines(as.vector(unlist(cvfit2$glmnet.fit[5])), as.vector(coef(cvfit2$glmnet.fit)[i,]),
        type = "b",col=colors[i])
}
```



Comments: As lambda increases, our coefficients all shirnk in size. In ridge regression, the coefficients decrease in more curved shapes and assympotically approach zero. With lasso, coefficients decrease in a spline shape and some coefficients are set to zero. This is because lasso does coefficient by using the ℓ_1 norm of the beta coefficients.

Problem 3 (25 points)

In this exercise, we will predict the number of applications received using the other variables in the `College` data set in the `ISLR` package.

Part a.

Randomly sample 20 percent of the rows of the dataset, and set this aside as a test set. Let the remainder be the training set.

```
#Lets convert the private variable to a numeric binary indicator.
College$Private <- ifelse(College$Private=="Yes",1,0)
```

```
set.seed(365)
testrows <- sample(nrow(College),round(nrow(College)*.20))
test <- College[testrows,]
train <- College[-testrows,]
```

Part b.

Fit a linear model using least squares on the training set, and report the test error obtained.

```
set.seed(365)
mod1 <- lm(Apps ~ ., data=train)
predapps <- predict(mod1, newdata=test[-2])
MSE_lm <- sum((test[2]-predapps)^2)/nrow(test)
MSE_lm
```

```
## [1] 1033568
```

Part c.

Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
set.seed(1)
cvmod2_ridge <- cv.glmnet(x=as.matrix(train[-2]), y=as.matrix(train[2]),
                          standardize=TRUE, nfold=10, alpha=0)
predapps_ridge <- predict(cvmod2_ridge, newx=as.matrix(test[-2]), s="lambda.min")
MSE_ridge <- sum((test[2]-predapps_ridge)^2)/nrow(test)
MSE_ridge
```

```
## [1] 885803.7
```

Part d.

Fit a lasso model on the training set, with λ chosen by cross-validation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
set.seed(1)
cvmod3_lasso <- cv.glmnet(x=as.matrix(train[-2]), y=as.matrix(train[2]),
                          standardize=TRUE, nfold=10, alpha=1)
predapps_lasso <- predict(cvmod3_lasso, newx=as.matrix(test[-2]), s="lambda.min")
MSE_lasso <- sum((test[2]-predapps_lasso)^2)/nrow(test)
MSE_lasso
```

```
## [1] 998732.2
```

Part e.

Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these approaches? Which model do you think is better?

Comments: The mean squared error is highest for the linear model, decreases for the LASSO model and decreases more with the ridge regression model (1,033,000 $>$ 999,000 $>$ 886,000). No there is no a significant difference, based on a different seed we would get different values and thus likely a different order in which models produce the lowest MSE. According to set.seed(1) ridge regression is the best model.