Lee, Daniel
18379784
Stat 230A
4/21/17

Problem Set 5

Problem 1.

1. Augment Y_n and X_n as the following:

$$\underset{(n+p)}{\tilde{Y}} = \begin{bmatrix} \underset{n\times 1}{Y} \\ \underset{p\times 1}{0} \end{bmatrix}$$

$$\underset{(n+p)}{\tilde{X}} = \begin{bmatrix} \underset{n\times p}{X} \\ \sqrt{\lambda} I_p \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}_{p\times 1}$$

Then,

$$\tilde{X}^T\tilde{X} = \begin{bmatrix} X_n^T & \sqrt{\lambda} I_p \end{bmatrix} \begin{bmatrix} X_n \\ \sqrt{\lambda} I_p \end{bmatrix}$$

$$= X_n^T X_n + \sqrt{\lambda} I_p \sqrt{\lambda} I_p$$

$$= X_n^T X_n + \lambda I_p$$

$$(\tilde{X}^T\tilde{X})^{-1} = (X_n^T X_n + \lambda I_p)^{-1}$$

$$\tilde{X}^T\tilde{Y} = \begin{bmatrix} X_n^T & \sqrt{\lambda} I_p \end{bmatrix} \begin{bmatrix} Y \\ 0 \end{bmatrix}$$

$$= X_n^T Y$$

Therefore,

$$\hat{\tilde{\beta}} = (\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\tilde{Y}$$

$$= (X_n^T X_n + \lambda I_p)^{-1} X_n^T Y,$$

which is the ridge regression estimator.

Problem 1.

2. Augment X_n and Y_n as the following:

$$\tilde{X}_{(n+p)} = \begin{bmatrix} X \\ n+p \\ \sqrt{\lambda\alpha}\, I_p \end{bmatrix}$$

$$\tilde{Y}_{(n+p)} = \begin{bmatrix} Y_n \\ n\times 1 \\ 0 \\ p\times 1 \end{bmatrix}$$

Then,

$$\|\tilde{Y}_n - \tilde{X}_n \beta\|_2^2$$

$$= \left(\begin{bmatrix} Y_n \\ 0 \end{bmatrix} - \begin{bmatrix} X_n\beta \\ \sqrt{\lambda\alpha}\,\beta \end{bmatrix} \right)^T \left(\begin{bmatrix} Y_n \\ 0 \end{bmatrix} - \begin{bmatrix} X_n\beta \\ \sqrt{\lambda\alpha}\,\beta \end{bmatrix} \right)$$

$$= \left(\underset{1\times n}{\begin{bmatrix} Y_n^T \end{bmatrix}} \; \underset{1\times p}{\begin{bmatrix} 0^T \end{bmatrix}} - \underset{1\times n}{\begin{bmatrix} (X_n\beta)^T \end{bmatrix}} \; \underset{1\times p}{\begin{bmatrix} (\sqrt{\lambda\alpha}\,\beta)^T \end{bmatrix}} \right) \left(\underset{(n+p)\times 1}{\begin{bmatrix} Y_n \\ 0 \end{bmatrix}} - \underset{(n+p)\times 1}{\begin{bmatrix} X_n\beta \\ \sqrt{\lambda\alpha}\,\beta \end{bmatrix}} \right)$$

$$= \left(\underset{1\times (n+p)}{\begin{bmatrix} Y_n^T - \beta^T X_n^T & 0^T - \sqrt{\lambda\alpha}\,\beta^T \end{bmatrix}} \right) \left(\underset{(n+p)\times 1}{\begin{matrix} Y_n - X_n\beta \\ 0 - \sqrt{\lambda\alpha}\,\beta \end{matrix}} \right)$$

$$= (Y_n^T - \beta^T X_n^T)(Y_n - X_n\beta) + (0^T - \sqrt{\lambda\alpha}\,\beta^T)(0 - \sqrt{\lambda\alpha}\,\beta)$$

$$= Y_n^T Y_n - Y_n^T X_n \beta - \beta^T X_n^T Y_n + \beta^T X_n^T X_n \beta + \lambda\alpha\,\beta^T\beta$$

Problem 1. 2 (continued)

This can be rewritten as

$$\|Y_n - X_n^T \beta\|_2^2 + \lambda\alpha\|\beta\|_2^2.$$

Hence, I can simplify the following:

$$\min_\beta \|Y_n - X_n\beta\|_2^2 + \lambda\alpha\|\beta\|_2^2 + \lambda(1-\alpha)\|\beta\|_1$$

$$= \min_\beta \|\tilde{Y}_n - \tilde{X}_n\beta\|_2^2 + \lambda(1-\alpha)\|\beta\|_1$$

Hence, the augmented $\tilde{Y} = \begin{bmatrix} Y_n \\ 0 \end{bmatrix}$ and $\tilde{X} = \begin{bmatrix} X_n \\ \sqrt{\lambda\alpha} I_p \end{bmatrix}$

turns the elastic-net optimization problem into a LASSO problem with $\tilde{\lambda} = \lambda(1-\alpha)$.

Problem 2

2.1. Split the data set into a training set and a test set.

I will take a random 6/7 of the data (666 observations) as the training set and the remaining 1/7 of the data (111 observations) as the test set.

```
load("College.RData")

str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

```
x <- College[,-2]

y <- College$Apps

#Random 6/7th of the dataset is training set

train <- sample(x = 1:nrow(College), size = 6/7*nrow(College), replace = FALSE)

test <- (-train)

y.train <- College$Apps[train]

y.test <- College$Apps[test]

x.train <- x[train, ]

x.test <- x[test, ]
```

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

I will measure the error by the mean-squared error and the root-mean-squared error.

```r
linear <- lm(y.train ~ ., data = x.train)

summary(linear)
```

```
##
## Call:
## lm(formula = y.train ~ ., data = x.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5159.4  -424.6    -4.4   317.8  7427.2
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -472.20516  432.60774  -1.092  0.27544
## PrivateYes  -441.75374  144.07380  -3.066  0.00226 **
## Accept         1.62266    0.04154  39.061  < 2e-16 ***
## Enroll        -0.95538    0.19652  -4.861 1.46e-06 ***
## Top10perc     42.92113    5.75026   7.464 2.71e-13 ***
## Top25perc     -9.08407    4.61227  -1.970  0.04932 *
## F.Undergrad    0.06119    0.03340   1.832  0.06740 .
## P.Undergrad    0.03893    0.03356   1.160  0.24640
## Outstate      -0.09983    0.02058  -4.850 1.54e-06 ***
## Room.Board     0.15887    0.05171   3.073  0.00221 **
## Books         -0.08258    0.24759  -0.334  0.73885
## Personal       0.09059    0.07123   1.272  0.20392
## PhD           -7.63711    5.11710  -1.492  0.13606
## Terminal      -3.39915    5.50843  -0.617  0.53740
## S.F.Ratio      7.64950   13.77372   0.555  0.57883
## perc.alumni    0.63426    4.37992   0.145  0.88491
## Expend         0.07337    0.01306   5.619 2.86e-08 ***
## Grad.Rate      9.11670    3.10002   2.941  0.00339 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1019 on 648 degrees of freedom
## Multiple R-squared:  0.9333, Adjusted R-squared:  0.9315
## F-statistic:   533 on 17 and 648 DF,  p-value: < 2.2e-16
```

```r
# Calculate predictions
linear.pred <- predict(linear, newdata = x.test)

# Calculate MSE
linearMSE <- round(mean((linear.pred - y.test)^2), 3)

linearMSE
```

```
## [1] 1420125
```

```r
linearRMSE <- round(sqrt(linearMSE), 3)

linearRMSE
```

```
## [1] 1191.69
```

Test error obtained using ordinary least squares linear model is mse of 1.420125×10^6 and rmse of 1191.69

applications.

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

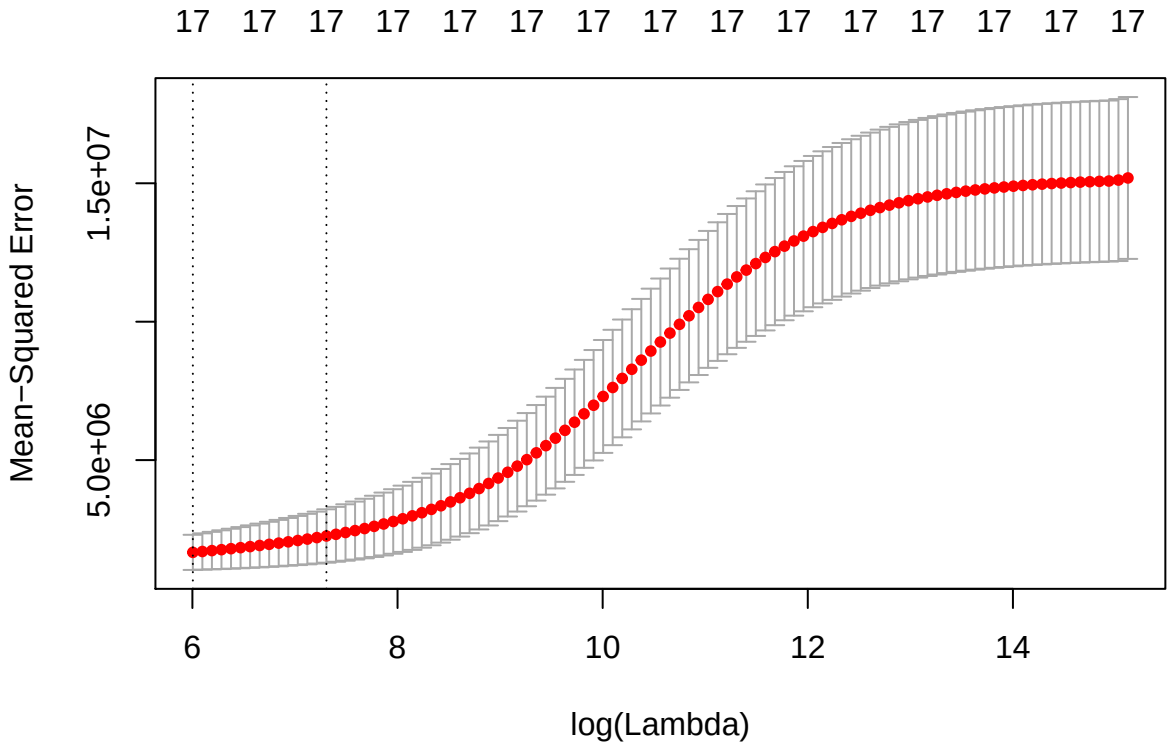
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
#create design matrix to accommodate for categorical variable

x.factors.prep <- model.matrix(y ~ x$Private)

x.factors <- as.matrix(data.frame(x[, -1], x.factors.prep))

x.train.factors <- x.factors[train, ]

x.test.factors <- x.factors[test, ]

cv.out <- cv.glmnet(x = x.train.factors, y = y.train, alpha = 0, family = "gaussian")
plot(cv.out)
```



```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 405.1599
```

```
ridge.pred = predict(cv.out, s = bestlam, newx = x.test.factors)

ridgeMSE <- round(mean((ridge.pred - y.test)^2), 3 )

ridgeMSE
```

```
## [1] 1364008
```

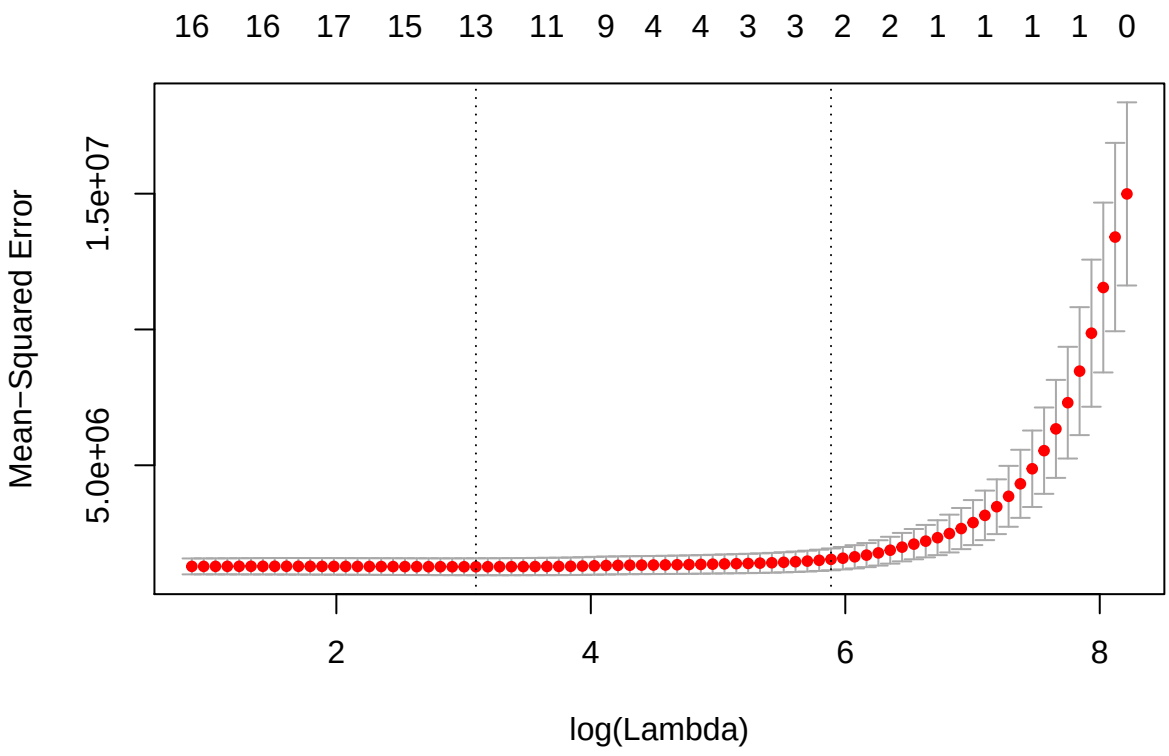
```
ridgeRMSE <- round(sqrt(ridgeMSE), 3)

ridgeRMSE
```

```
## [1] 1167.908
```

Test error obtained using ridge regression model is mse of 1.364008×10^6 and rmse of 1167.908 applications.

4. 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.

```
cv.out <- cv.glmnet(x = x.train.factors, y = y.train, alpha = 1, family = "gaussian")
plot(cv.out)
```




```
bestlam = cv.out$lambda.min
bestlam
```

```
## [1] 22.13096
```

```
lasso.pred = predict(cv.out, s = bestlam, newx = x.test.factors)

lassoMSE <- round( mean((lasso.pred - y.test)^2), 3 )

lassoMSE
```

```
## [1] 1466542
```

```
lassoRMSE <- round(sqrt(lassoMSE), 3)

lassoRMSE
```

```
## [1] 1211.009
```

```
grid = 10^ seq (10,-2, length =100)

out = glmnet(x = x.factors, y = y, alpha = 1, lambda = grid)

lasso.coef = predict(out, type = "coefficients", s = bestlam)
lasso.coef
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)  -609.64171910
```

```
## Accept          1.45210231
## Enroll         -0.18974012
## Top10perc      33.29076415
## Top25perc      -2.15270752
## F.Undergrad     .
## P.Undergrad     0.01955562
## Outstate       -0.05709551
## Room.Board      0.12390842
## Books           .
## Personal        0.00106854
## PhD            -5.49462294
## Terminal       -3.32092801
## S.F.Ratio       4.09031024
## perc.alumni    -0.96956602
## Expend          0.06938860
## Grad.Rate       5.07521110
## X.Intercept.    .
## x.PrivateYes -420.46714125
```

```
lasso.coef[lasso.coef !=0]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

```
##  [1] -609.64171910    1.45210231   -0.18974012   33.29076415   -2.15270752
##  [6]    0.01955562   -0.05709551    0.12390842    0.00106854   -5.49462294
## [11]   -3.32092801    4.09031024   -0.96956602    0.06938860    5.07521110
## [16] -420.46714125
```

```
length(lasso.coef[lasso.coef !=0])
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

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

Test error obtained using lasso model is mse of 1.4665424×10^6 and rmse of 1211.009 applications. There are 16 non-zero coefficient estimates.

5. 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 three approaches?

Models	Mean-Squared Error	Root MSE (applications)
Linear Regression	1.420125×10^6	1191.69
Ridge Regression	1.364008×10^6	1167.908
Lasso Regression	1.4665424×10^6	1211.009

From the output, I see that ridge regression has the lowest test error while lasso regression has the highest test error. Based on these three approaches, we can predict the number of college applications a college would receive to within 1167.908 to 1211.009 applications. However, this only makes sense for colleges that receive many applications. Based on the data, the median number of applications received by the schools in the data is 1558. So, some of the schools would not even receive a total of 1167.908 applications, which makes our prediction accuracy of 1167.908 to 1211.009 pretty meaningless.

Problem 3

3.a. Generate a data set with $p = 20$ features, $n = 1000$ observations, and an associated quantitative response vector generated according to the model

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

where β has some elements that are exactly equal to zero. Use the below codes for simulation.

```
set.seed(1)
x <- matrix(rnorm(1000 * 20), 1000, 20)
b <- rnorm(20)
b[3] <- 0
b[4] <- 0
b[9] <- 0
b[19] <- 0
b[10] <- 0
eps <- rnorm(1000)
y <- x%*% b + eps
```

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

```
# generate random indices for the training set
train <- sample(x = 1:nrow(x), size = 100, replace = FALSE)

test <- (-train)

x <- data.frame(x)

y.train <- y[train]

y.test <- y[test]

x.train <- x[train, ]

x.test <- x[test, ]
```

3.c. Perform best subset selection on the training set, and plot the training set MSE (Mean Squared Error) associated with the best model of each size.

```
library(leaps)

regfit.best.subset <- regsubsets(y.train ~ .,data = x.train, nvmax = 20, intercept = FALSE)

summary(regfit.best.subset)
```

```
## Subset selection object
## Call: regsubsets.formula(y.train ~ ., data = x.train, nvmax = 20, intercept = FALSE)
## 20 Variables
##     Forced in Forced out
```

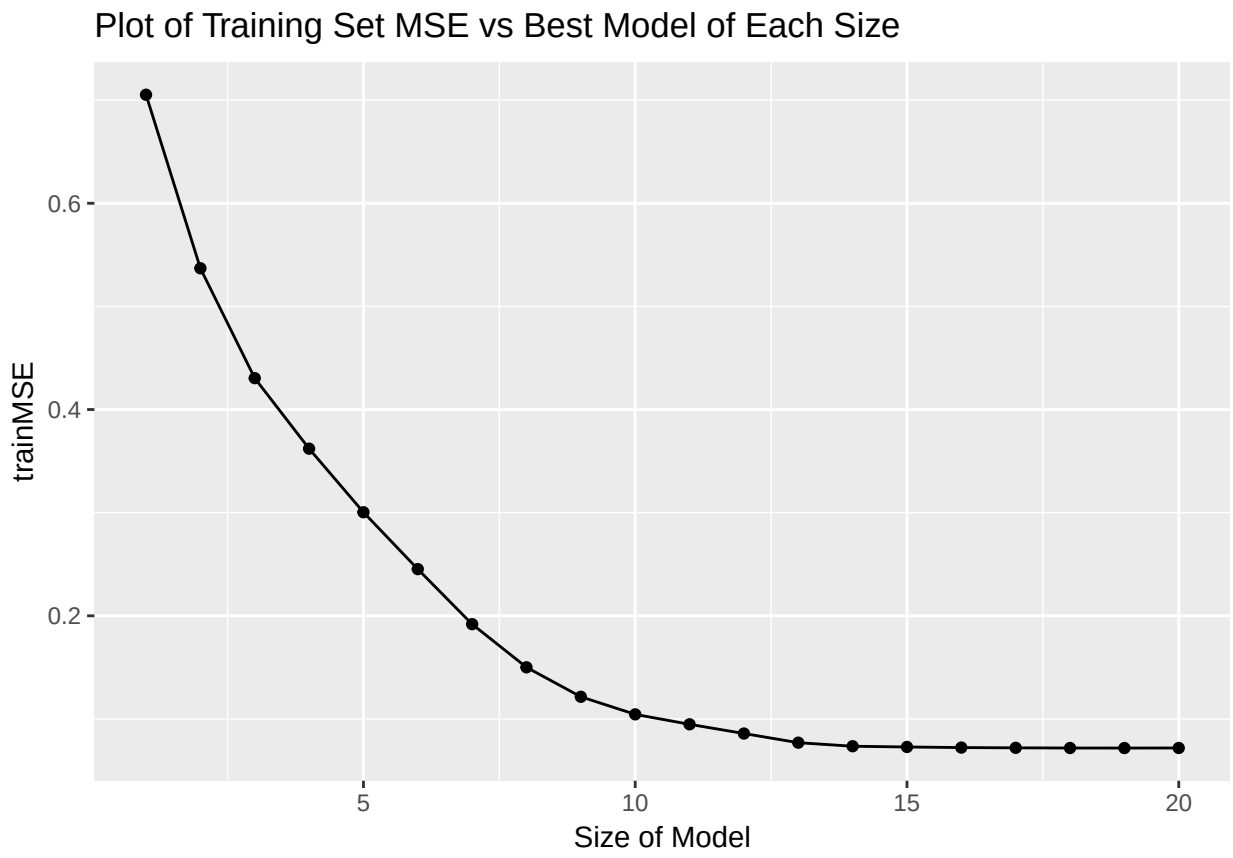
```
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## X11     FALSE      FALSE
## X12     FALSE      FALSE
## X13     FALSE      FALSE
## X14     FALSE      FALSE
## X15     FALSE      FALSE
## X16     FALSE      FALSE
## X17     FALSE      FALSE
## X18     FALSE      FALSE
## X19     FALSE      FALSE
## X20     FALSE      FALSE
## 1 subsets of each size up to 20
## Selection Algorithm: exhaustive
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 X16
## 1  ( 1 )  " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
## 2  ( 1 )  " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " " " "
## 3  ( 1 )  " " " " " " " " " " " " " " "*" "*" " " " " " " " " " " " " " "
## 4  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " " " " " " " " " " "
## 5  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " " " " " " " "*" " "
## 6  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " "*" " " " " " "*" " "
## 7  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " "*" " " " " " " "*" "
## 8  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " "*" " " " " " "*" "*"
## 9  ( 1 )  " " " " " " " " "*" " " " "*" "*" " " " " " "*" "*" " " " "*" "*"
## 10  ( 1 ) "*" " " " " " " " "*" " " " "*" "*" " " " " " "*" "*" " " " "*" "*"
## 11  ( 1 ) "*" " " " " " " " "*" " " " "*" "*" " " " " " "*" "*" "*" "*" "*"
## 12  ( 1 ) "*" "*" " " " " " "*" " " " "*" "*" " " " " " "*" "*" " " " "*" "*"
## 13  ( 1 ) "*" "*" " " " " " "*" " " " "*" "*" " " " " " "*" "*" "*" "*" "*"
## 14  ( 1 ) "*" "*" " " " " " "*" " " " "*" "*" " " " " " "*" "*" "*" "*" "*"
## 15  ( 1 ) "*" "*" " " " " " "*" " " " "*" "*" "*" " " " "*" "*" "*" "*" "*"
## 16  ( 1 ) "*" "*" "*" " " " "*" " " " "*" "*" "*" " " " "*" "*" "*" "*" "*"
## 17  ( 1 ) "*" "*" "*" " " " "*" " " " "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 18  ( 1 ) "*" "*" "*" " " " "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 19  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
## 20  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
##           X17 X18 X19 X20
## 1  ( 1 )  " " "*" " " " "
## 2  ( 1 )  " " "*" " " " "
## 3  ( 1 )  " " "*" " " " "
## 4  ( 1 )  " " "*" " " " "
## 5  ( 1 )  " " "*" " " " "
## 6  ( 1 )  " " "*" " " " "
## 7  ( 1 )  " " "*" " " "*"
## 8  ( 1 )  " " "*" " " "*"
## 9  ( 1 )  " " "*" " " "*"
## 10  ( 1 ) " " "*" " " "*"
## 11  ( 1 ) " " "*" " " "*"
```

```
## 12  ( 1 ) "*" "*" " " " " "*"
## 13  ( 1 ) "*" "*" " " " " "*"
## 14  ( 1 ) "*" "*" " " " " "*"
## 15  ( 1 ) "*" "*" " " " " "*"
## 16  ( 1 ) "*" "*" " " " " "*"
## 17  ( 1 ) "*" "*" " " " " "*"
## 18  ( 1 ) "*" "*" " " " " "*"
## 19  ( 1 ) "*" "*" " " " " "*"
## 20  ( 1 ) "*" "*" "*" "*" "*"
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
trainMSE = rep(NA, ncol(x))

for(i in 1:ncol(x)){

  coefi = coef(regfit.best.subset, id=i)

  pred = as.matrix(x.train[ , names(coefi)]) %*% coefi

  trainMSE[i]= sum((y.train - pred)^2) / (nrow(x) - i)

}

ggplot(data.frame(trainMSE), aes(x = 1:ncol(x.train), y = trainMSE)) +
  geom_point() +
  geom_line() +
  xlab("Size of Model") +
  ggtitle("Plot of Training Set MSE vs Best Model of Each Size")
```



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

```
testMSE = rep(NA, ncol(x))

for(i in 1:ncol(x)){

  coefi = coef(regfit.best.subset, id=i)

  pred = as.matrix(x.test[ , names(coefi)]) %*% coefi

  testMSE[i]= mean((y.test - pred)^2)

}

ggplot(data.frame(testMSE), aes(x = 1:(ncol(x.test)), y = testMSE)) +
  geom_point() +
  geom_line() +
  xlab("Size of Model") +
  ggtitle("Plot of Test Set MSE vs Best Model of Each Size")
```

3.e. For which model size does the test set MSE take on its minimum value?

```
which(testMSE == min(testMSE))
```

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

The test set MSE takes on its minimum value when $p = 14$. The corresponding MSE value is 1.4006556.

3.f. How does the model at which the test set MSE is minimized compare to the true model used to generate the data?

```
b.values <- data.frame(b, idx = names(coef(regfit.best.subset, id=20)))
```

```
best_model <- data.frame(coef = round(coef(regfit.best.subset, id = which(testMSE == min(testMSE))), 3))

best_model <- data.frame(best_model, idx = rownames(best_model))

require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
```

```
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
comparison <- full_join(b.values, best_model, by = "idx")
```

```
## Warning in full_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

```
comparison[is.na(comparison)] <- 0

comparison <- mutate(comparison, diff = round(abs(b - coef), 3))

b <- comparison$b

best_model <- comparison$coef

diff <- comparison$diff
```

The model at which the test set MSE is minimized is pretty close to the β values of true model used to generate the data.

j	True β_j	Best Subset $\hat{\beta}_j$	Difference
1	0.2353485	0.38	0.145
2	0.244825	0.323	0.078
3	0	0	0
4	0	0	0
5	1.0386957	1.06	0.021
6	−0.2835501	0	0.284
7	−1.4097291	−1.292	0.118
8	0.7231804	0.833	0.11
9	0	0	0
10	0	0	0
11	0.8791534	0.684	0.195
12	0.5545564	0.574	0.019
13	−0.2845811	−0.362	0.077
14	−0.674658	−0.826	0.151
15	−0.7154889	−0.573	0.142
16	−0.2705279	−0.198	0.073
17	0.3129646	0.308	0.005
18	1.6698068	1.563	0.107
19	0	0	0
20	−1.0154889	−0.808	0.207

When the true model βs are zero, the best subset model also has $\hat{\beta} = 0$ except for β_6. The true model βs and the best subset model $\hat{\beta}$s have the same signs.

3.g. Create a plot displaying $\sqrt{\sum_{j=1}^{p}(\beta_j - \hat{\beta}_j^r)^2}$ for a range of values or r, where $\hat{\beta}_j^r$ is the jth coefficient estimate for the best model containing r coefficients. Comment on what you boserve. How does this compare to the test MSE plot from (d)?

```
require(reshape)
```

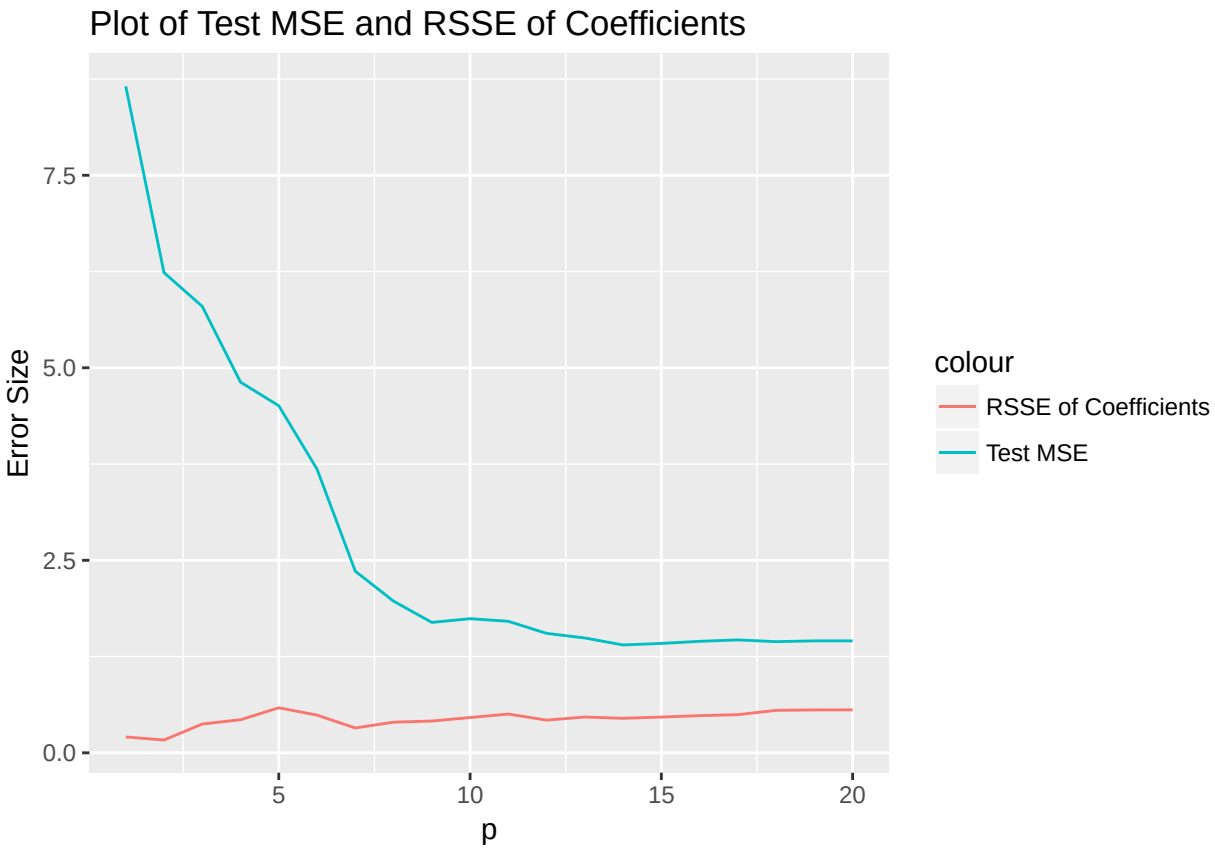
```
## Loading required package: reshape
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
coefRSSE = rep(NA, ncol(x))

colnames(x.test) <- 1:20

for(i in 1:ncol(x)){

  coefi = coef(regfit.best.subset, id=i)

  true.beta = b[as.numeric(gsub("X", "", names(coefi)))]

  coefRSSE[i] = sqrt(sum((true.beta - coefi)^2))

}

p <- 1:ncol(x.test)

coef.test <- data.frame(coefRSSE = coefRSSE, testMSE, p = 1:20)

ggplot(coef.test, aes(p)) +
  geom_line(aes(y = coefRSSE, colour = "RSSE of Coefficients")) +
  geom_line(aes(y = testMSE, colour = "Test MSE")) +
  ylab("Error Size") +
  ggtitle("Plot of Test MSE and RSSE of Coefficients")
```



The root sum of squares of the coefficient values do not necessarily correlate with the test MSE. Test MSE might decrease even though the root sum of squares of the coefficient values may increase.

Ridge and LASSO Regression

a. Ridge Regression

Prior : $p(\beta) = \prod_{i=1}^{p} p(\beta_i)$, $p(\beta_i) \sim N(\mu = 0, \sigma^2 = c)$

$$= \prod_{i=1}^{p} \left[\frac{1}{\sqrt{2\pi c}} \exp\left(-\frac{\beta_i^2}{2c} \right) \right]$$

$$= \left(\frac{1}{\sqrt{2\pi c}} \right)^p \exp\left(-\frac{1}{2c} \sum_{i=1}^{p} \beta_i^2 \right)$$

$$f(\beta | X, Y) \propto f(Y | X, \beta) p(\beta | X) = f(Y | X, \beta) p(\beta)$$

$$f(Y | X, \beta) p(\beta) = \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 \right)$$

$$* \left(\frac{1}{\sqrt{2\pi c}} \right)^p \exp\left(-\frac{1}{2c} \sum_{i=1}^{p} \beta_i^2 \right)$$

$$\log f(Y | X, \beta) p(\beta)$$

$$= \log\left[\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \left(\frac{1}{\sqrt{2\pi c}} \right)^p \right] - \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 + \frac{1}{2c} \sum_{i=1}^{p} \beta_i^2 \right)$$

$$\underset{\beta}{\arg\max}\ f(\beta | X, Y)$$

$$= \underset{\beta}{\arg\max} \log\left[\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \left(\frac{1}{\sqrt{2\pi c}} \right)^p \right] - \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 + \frac{1}{2c} \sum_{i=1}^{p} \beta_i^2 \right)$$

$$\Leftrightarrow \underset{\beta}{\arg\min} \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 + \frac{1}{2c} \sum_{i=1}^{p} \beta_i^2 \right)$$

$$= \underset{\beta}{\arg\min} \left(\frac{1}{2\sigma^2} \right) \left(\sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 + \frac{\sigma^2}{c} \sum_{i=1}^{p} \beta_i^2 \right)$$

Let $\lambda = \frac{\sigma^2}{c}$

$$= \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \right) \left(\sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 + \lambda \sum_{i=1}^{p} \beta_i^2 \right)$$

This is the ridge regression model. The larger the variance of the prior, c, the smaller the λ.

b. LASSO regression

Prior: $p(\beta) = \frac{1}{2b} \exp(-|\beta|/b)$

where $p(\beta_i) = \frac{1}{2b} e^{-|\beta_i|/b}$

The prior is the Laplace distribution.

Likelihood:

$$\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(- \frac{\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2}{2\sigma^2} \right)$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 \right)$$

where $Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \sim N(0, \sigma^2)$

$f(\beta|X,Y) \propto f(Y|X,\beta) p(\beta|X) = f(Y|X,\beta) p(\beta)$

$$f(Y|X,\beta) p(\beta) = \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \left[Y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij} \right) \right]^2 \right) * \left(\frac{1}{2b} \exp(-|\beta|/b) \right)$$

$$f(Y|X,\beta)p(\beta) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \left(\frac{1}{2b}\right) \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 - \frac{|\beta|}{b}\right)$$

$$\log f(Y|X,\beta)p(\beta) =$$

$$\log\left[\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \left(\frac{1}{2b}\right)\right] - \left(\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \frac{|\beta|}{b}\right)$$

$$\underset{\beta}{\operatorname{argmax}}\, f(\beta|X,Y) =$$

$$\underset{\beta}{\operatorname{argmax}} \log\left[\left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \left(\frac{1}{2b}\right)\right] - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \frac{|\beta|}{b}$$

$$\Leftrightarrow \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \frac{|\beta|}{b}$$

$$= \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \frac{1}{b}\sum_{j=1}^{p}|\beta_j|$$

$$= \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \frac{2\sigma^2}{b}\sum_{j=1}^{p}|\beta_j|\right)$$

Let $\lambda = \frac{2\sigma^2}{b}$

$$= \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}\left[Y_i - \left(\beta_0 + \sum_{j=1}^{p}\beta_j X_{ij}\right)\right]^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right)$$

This is the LASSO regression model. The larger the scale parameter b, the smaller the λ.

Question 5. Elastic Net: Simulation Study

5.a. Provide numerical and graphical summaries of the simulation model and of the learning set.

```
require(MASS)
require(graphics)
require(glmnet)
require(RColorBrewer)
require(ggplot2)
require(reshape)
require(reshape2)
source("multiplot.R")
```

```
myPaletteSeq <- brewer.pal(10, "Paired")
set2 <- c("#ced64a",
          "#6042bb",
          "#70d454",
          "#c14bcb",
          "#83d7a3",
          "#c7407f",
          "#5b8842",
          "#d04338",
          "#76c4d1",
          "#c67a3b",
          "#6b84cd",
          "#c4ad5d",
          "#492b5b",
          "#c8bbae",
          "#642f2b",
          "#c58dc8",
          "#404d2d",
          "#c1777c",
          "#546d7d")

set3 <- c("#281a59",
"#a0d44e",
"#320172",
"#94e97a",
"#78007e",
"#43b94f",
"#6b4fc3",
"#dad969",
"#340a67",
"#7feaa3",
"#aa0078",
"#53edb6",
"#ba0053",
"#00842d",
"#df7aed",
"#527400",
"#0045ac",
"#d3860c",
```

```
"#003f88",
"#bf7300",
"#62baff",
"#b35500",
"#bea1ff",
"#8a8400",
"#ffa4f0",
"#00570e",
"#ff527d",
"#02b28d",
"#e85342",
"#b9e189",
"#5a1c52",
"#ffb46c",
"#900044",
"#85742e",
"#ff86b8",
"#8f2e00",
"#ff9a96",
"#7e0012",
"#ff8f68",
"#ff6b75")

rho <- 0.5
J <- 10
beta <- c(-J/2 + 1:5, sort(J/2 - 1:5))/10
sigma = 2

lambda <- 0:100
rev_lambda <- rev(lambda)
nlambda <- length(rev_lambda)
```

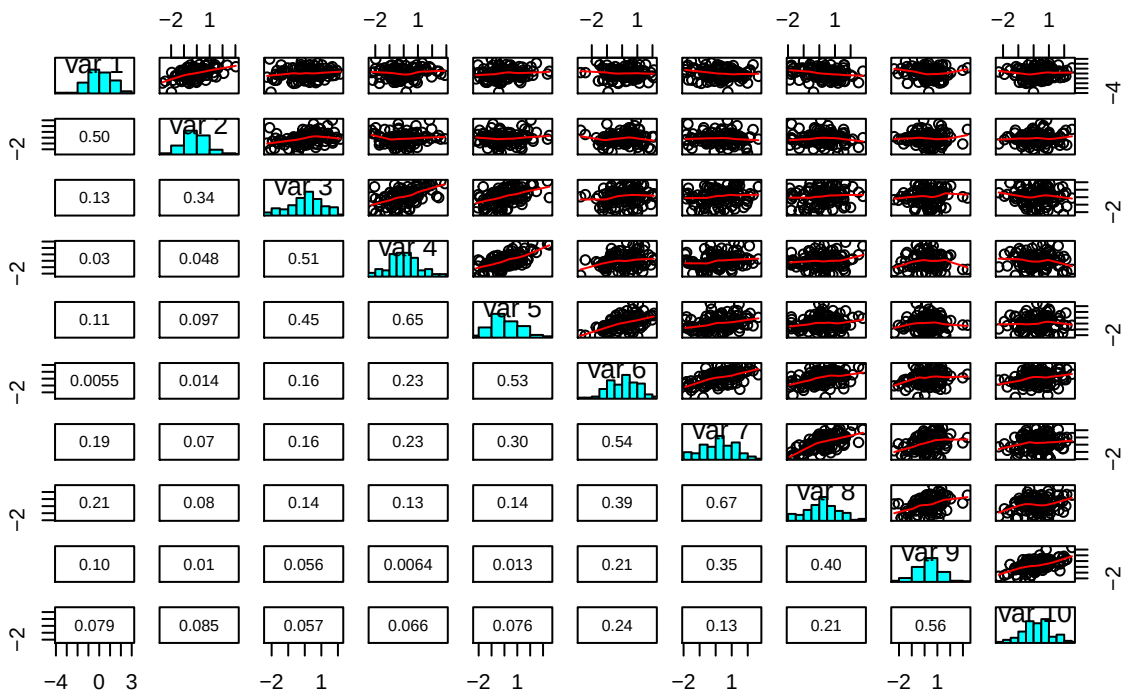
I will use the reverse of 0, 1, ..., 100 for the λ values for many of the computations and plots because the `glmnet` function's default setting makes it so that the λ's orders are decreasing when I input increasing orders of λ.

```
#Covariance matrix
Covariance <- matrix(NA, 10, 10)

for(i in 1:J){
  for(j in 1:J){
    Covariance[i,j] <- rho^(abs(i-j))
  }
}

#Learning Set
LS_sample_size = 100

X_LS <- mvrnorm(LS_sample_size, mu = rep(0, J), Sigma = Covariance)

mu_given_X_LS <- X_LS %*% beta

Y_LS <- rnorm(LS_sample_size, mean = mu_given_X_LS, sd = sigma)
```

```r
#Test Set

TS_sample_size <- 1000

X_TS <- mvrnorm(TS_sample_size, mu = rep(0, J), Sigma = Covariance)

mu_given_X_TS <- X_TS %*% beta

Y_TS <- rnorm(TS_sample_size, mean = mu_given_X_TS, sd = sigma)
```

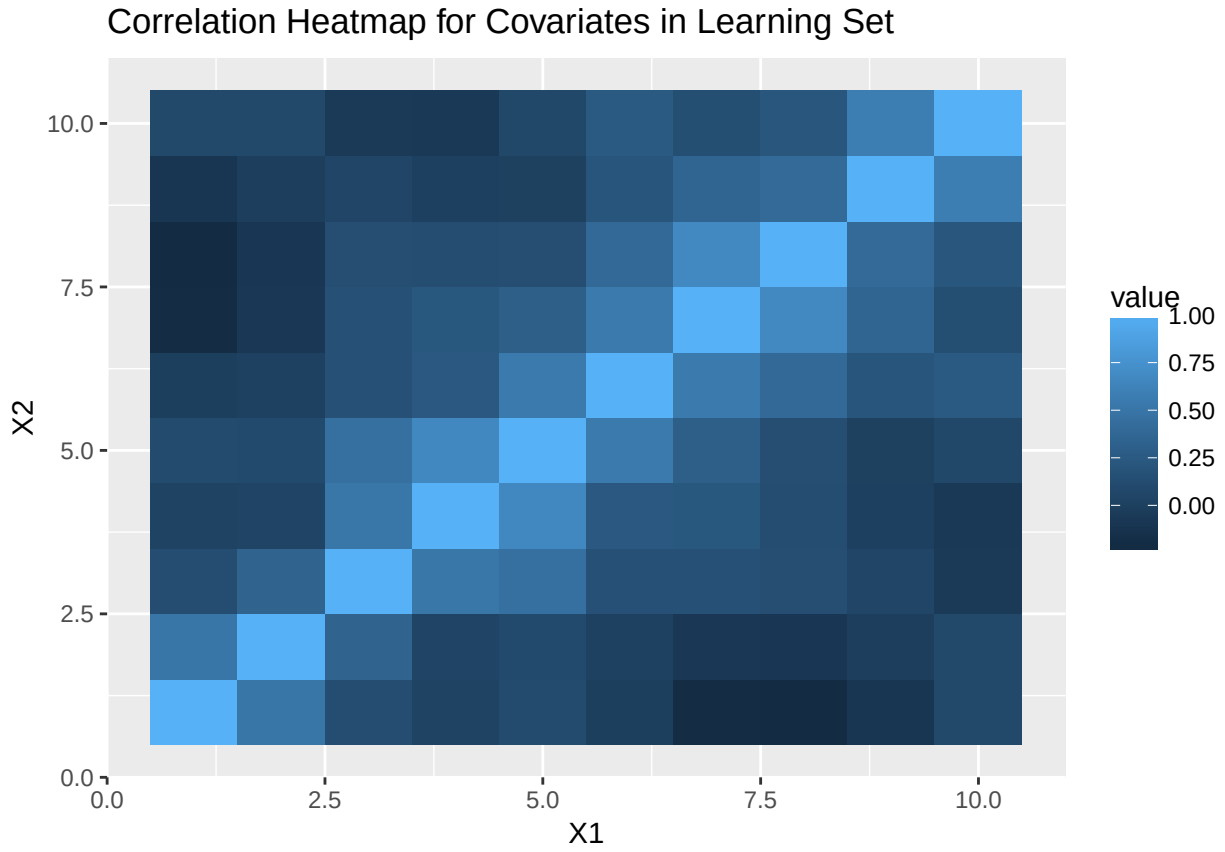
Numerical and Graphical Summaries of the Learning Set

```r
panel.hist <- function(x, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = 0.8)
}

pairs(X_LS, diag.panel = panel.hist, upper.panel = panel.smooth,
      lower.panel = panel.cor,
      main = "Pairwise Graphical Summary \n Learning Set Covariates")
```



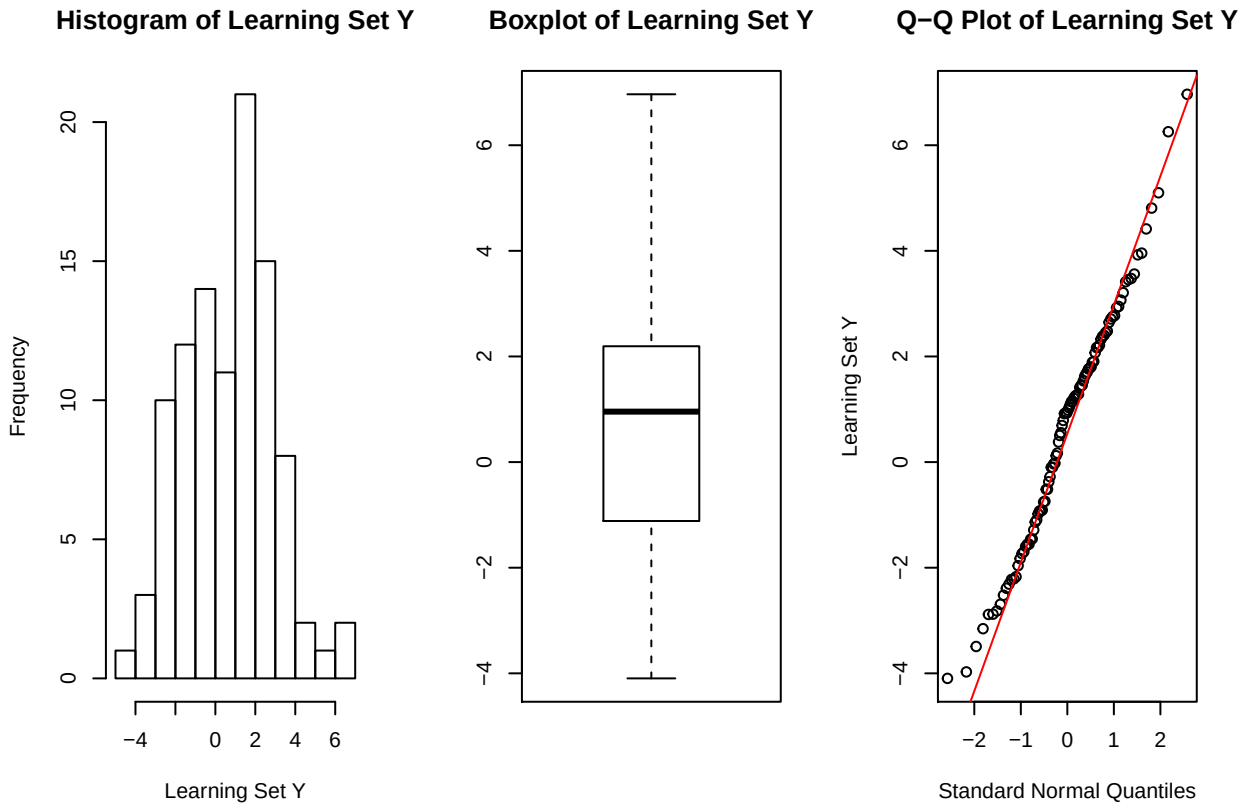
Based on the pairwise graphical summaries, you can see that each X_i has a normal distribution. The correlations between the covariates are also shown. The scatter plots show that the pairwise relationship between two variables is a bivariate normal distribution.

```
ggplot(melt(cor(X_LS)), aes(x = X1, y = X2)) +
  geom_tile(aes(fill = value)) +
  ggtitle("Correlation Heatmap for Covariates in Learning Set")
```



As expected, the X variables closer to each other are more highly correlated. That is, X_j and X_{j+i} are more highly correlated when the values of i are smaller.

```
#Histogram of Y

par(mfrow = c(1, 3))
hist(Y_LS, xlab = "Learning Set Y", main = "Histogram of Learning Set Y")
boxplot(Y_LS, main = "Boxplot of Learning Set Y")
qqnorm(Y_LS, main = "Q-Q Plot of Learning Set Y", ylab = "Learning Set Y",
       xlab = "Standard Normal Quantiles")
qqline(Y_LS, col = "red")
```

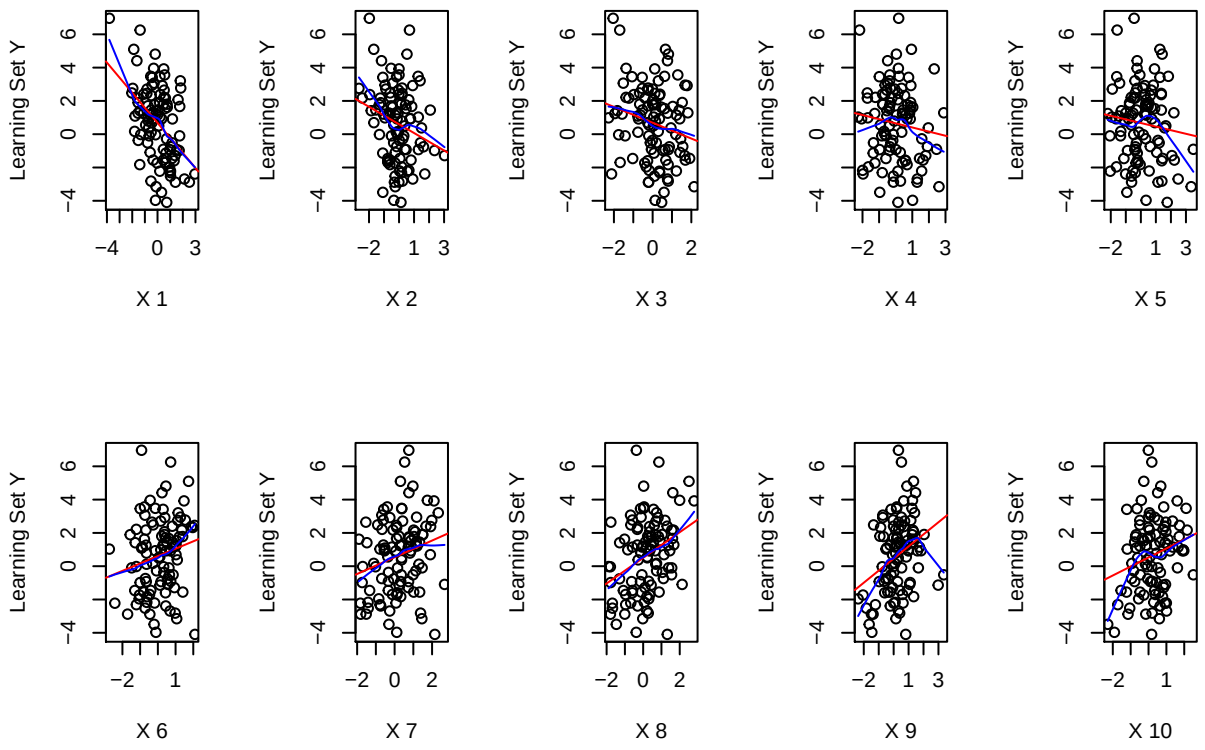


The histogram, qq-plot, and boxplot of the learning set of Y suggest that the Y's have a normal distribution as well. It seems as if the approximate mean of Y is zero.

```
summary(Y_LS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.0950 -1.1050  0.9555  0.6532  2.1830  6.9650
```

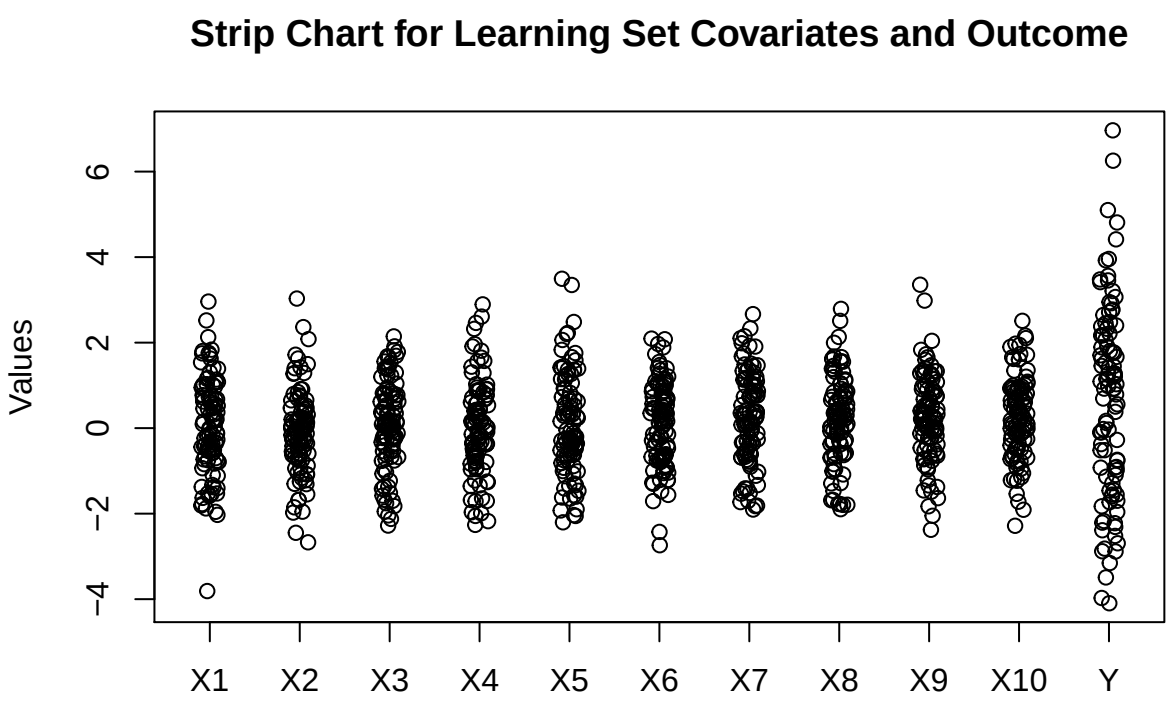
```
par(mfrow = c(2, 5))
for(i in 1:10){
  plot(X_LS[ , i], Y_LS, xlab = paste("X", i), ylab = "Learning Set Y")
  abline(lm(Y_LS ~ X_LS[, i]), col="red") # regression line (y~x)
  lines(lowess(X_LS[, i],Y_LS), col="blue") # lowess line (x,y)
}
```




```
par(mfrow = c(1, 1))
```

The scatterplot of the outcome Y vs. each of the 10 covariates show that the mean of Y seems to be zero for every X_i. Therefore, I will not be centering or scaling Y as centering and scaling will not make much of a difference. The Xs also seem to be centered at zero, which is what we expect based on the simulation model that generated the data.

```
#stripchart
stripchart_list <- as.list(rep(NA, J + 1))

for(i in 1:J){
  stripchart_list[[i]] <- X_LS[ , i]
}
stripchart_list[[J + 1]] <- Y_LS

stripchart(stripchart_list,
 main="Strip Chart for Learning Set Covariates and Outcome",
 xlab="",
 ylab = "Values",
 group.names=c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10", "Y"),
 vertical=TRUE,
 pch=1,
 method = "jitter"
)
```

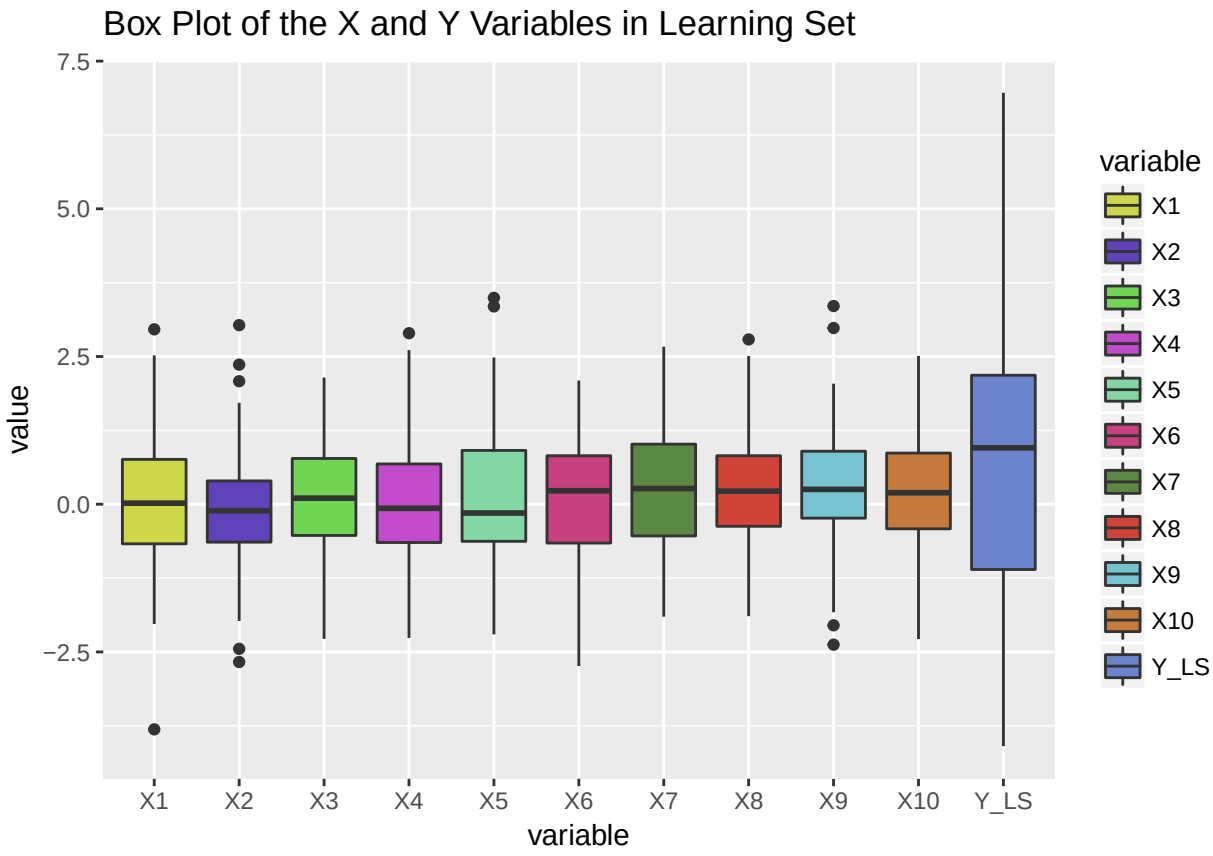


The stripchart further confirms that the Xs and the Y seem to be centered at zero. The covariates also seem to be scaled. So there is no need to transform the data before analysis.

```
LS_plot <- data.frame(X_LS, Y_LS)

LS_melt_plot <- melt(LS_plot)
```

```
## Using  as id variables
```

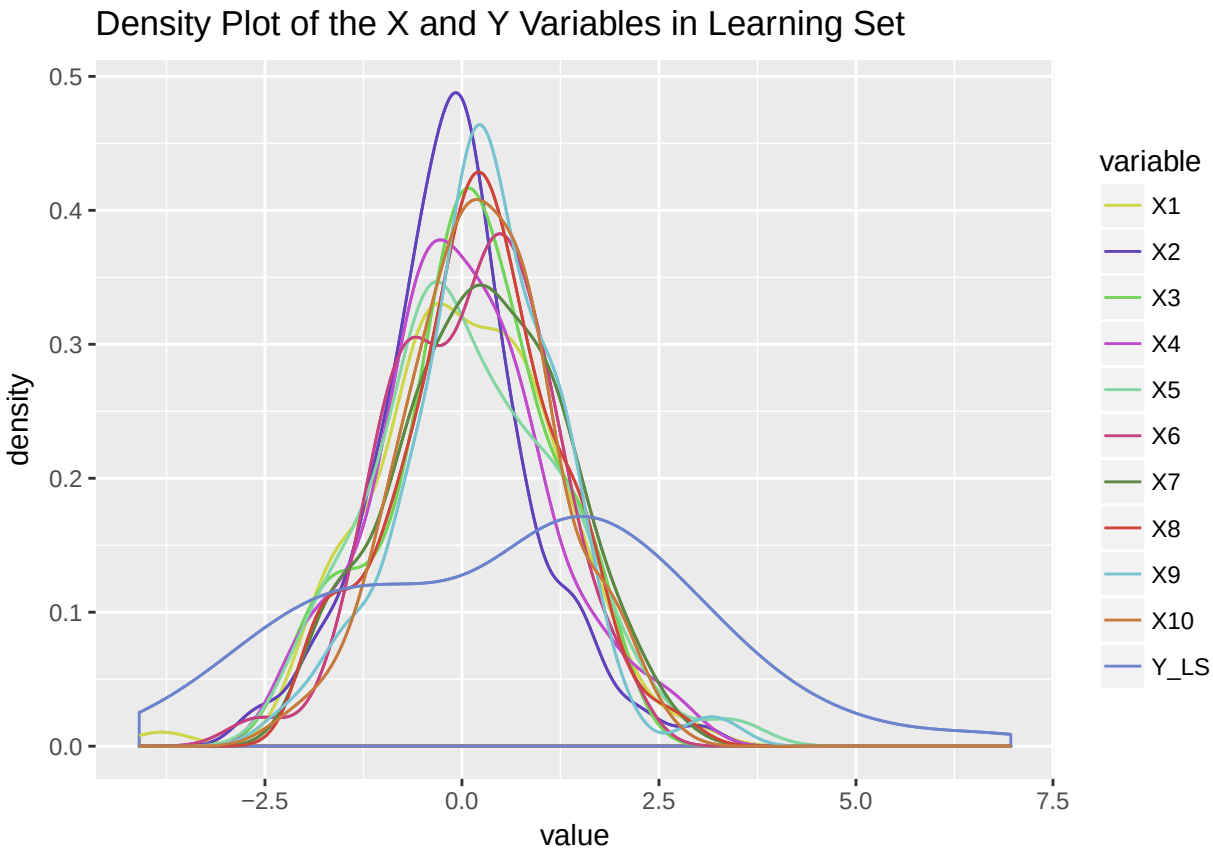
```
ggplot(LS_melt_plot, aes(x = variable, y = value, fill = variable)) +
  geom_boxplot() +
  scale_fill_manual(values = set2) +
  ggtitle("Box Plot of the X and Y Variables in Learning Set")
```



Boxplot of the learning set further confirms my findings from the stripchart above.

```
ggplot(LS_melt_plot, aes(x = value, group = variable, col = variable)) +
  geom_density(show_guide = FALSE) +
  scale_colour_manual(values = set2) +
  ggtitle("Density Plot of the X and Y Variables in Learning Set") +
  stat_density(aes(x = value, group = variable, col = variable), geom="line",position="identity")
```

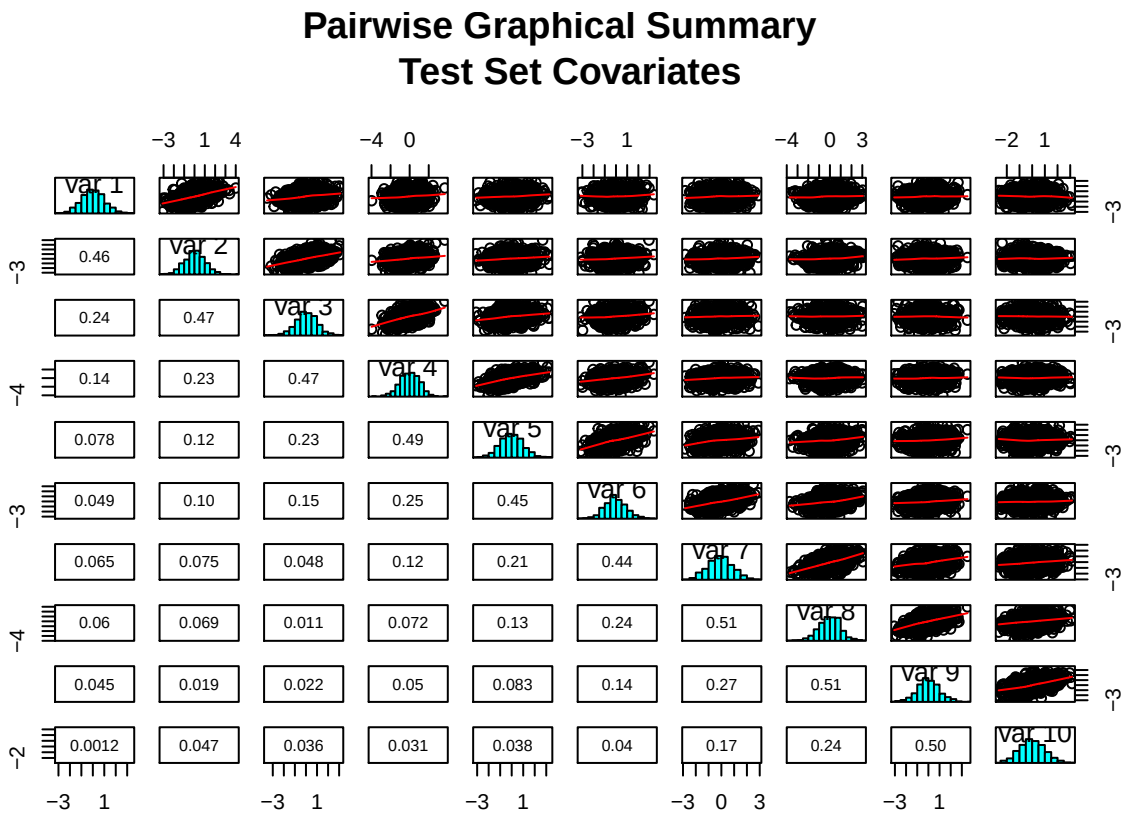
```
## Warning: `show_guide` has been deprecated. Please use `show.legend`
## instead.
```



The density plot shows that the Xs have normal distributions with center at zero. Their variances are very similar as well, which suggests that they are scaled. The Y is also normally distributed, but has a larger variance.

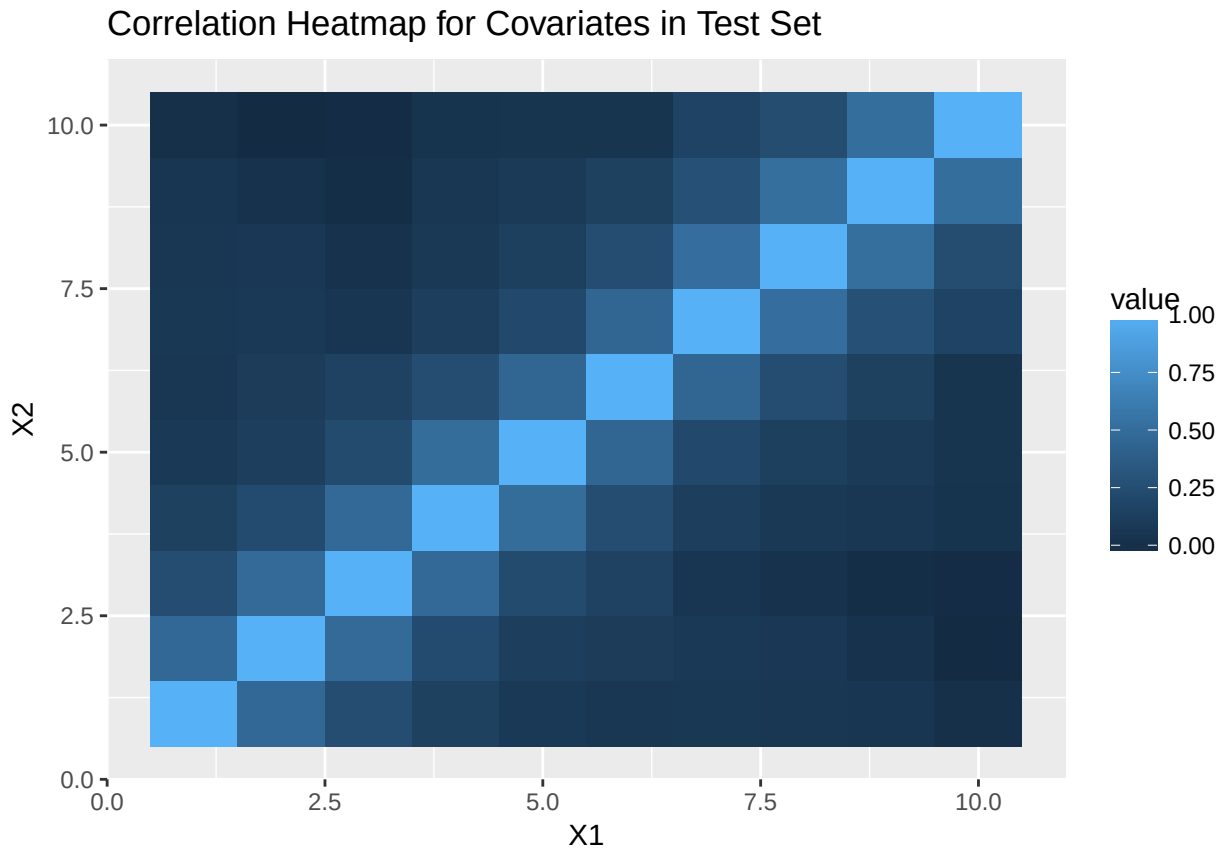
Numerical and Graphical Summaries of the Test Set

```
pairs(X_TS, diag.panel = panel.hist, upper.panel = panel.smooth,
      lower.panel = panel.cor,
      main = "Pairwise Graphical Summary \n Test Set Covariates")
```



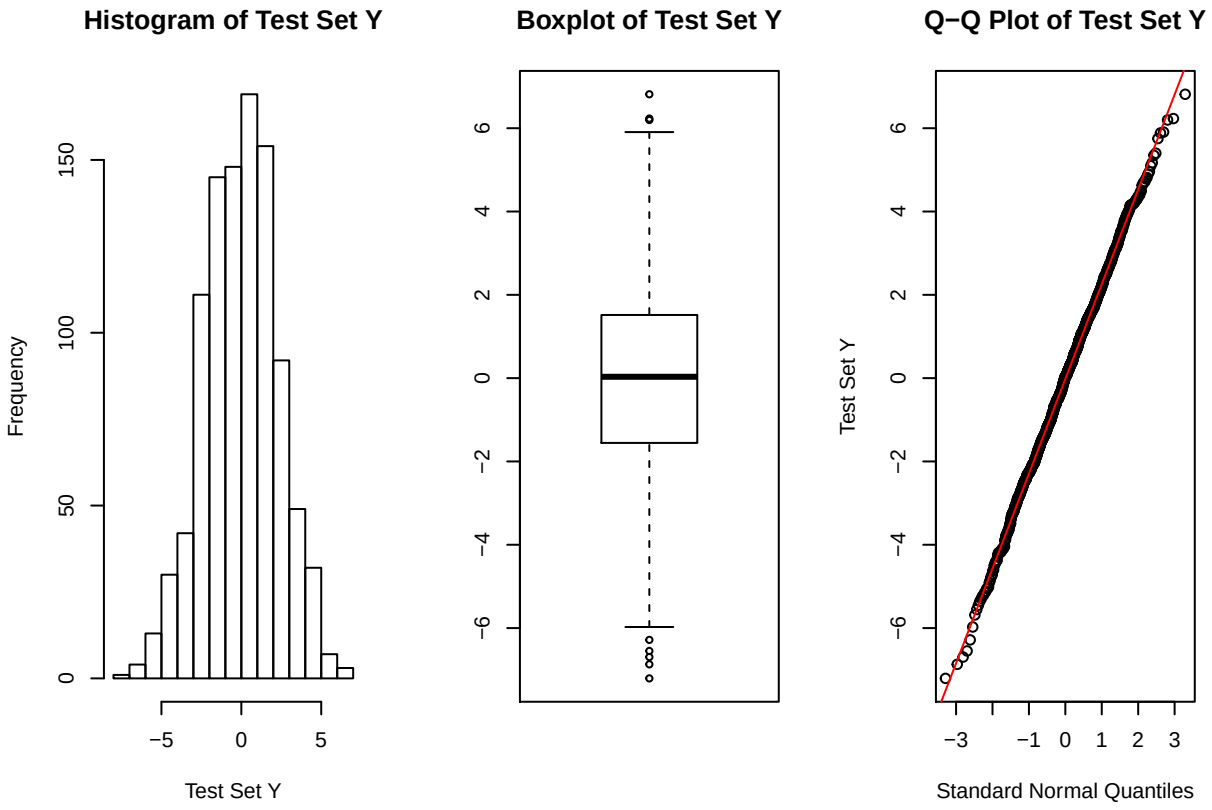
Based on the pairwise graphical summaries, you can see that each X_i has a normal distribution. The correlations between the covariates are also shown. The scatter plots show that the pairwise relationship between two variables is a bivariate normal distribution.

```
ggplot(melt(cor(X_TS)), aes(x = X1, y = X2)) +
  geom_tile(aes(fill = value)) +
  ggtitle("Correlation Heatmap for Covariates in Test Set")
```



As expected, the X variables closer to each other are more highly correlated. That is, X_j and X_{j+i} are more highly correlated when the values of i are smaller.

```
#Histogram of Y

par(mfrow = c(1, 3))
hist(Y_TS, xlab = "Test Set Y", main = "Histogram of Test Set Y")
boxplot(Y_TS, main = "Boxplot of Test Set Y")
qqnorm(Y_TS, main = "Q-Q Plot of Test Set Y", ylab = "Test Set Y",
       xlab = "Standard Normal Quantiles")
qqline(Y_TS, col = "red")
```

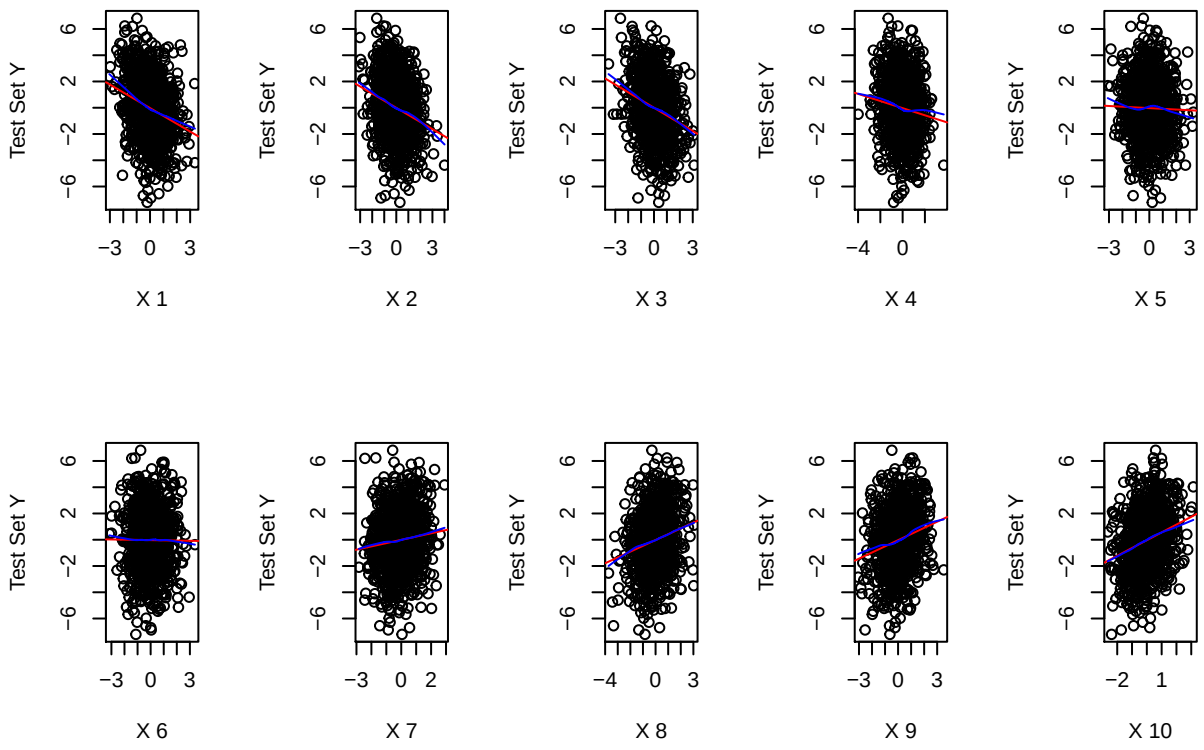


The histogram, qq-plot, and boxplot of the test set of Y suggest that the Y's have a normal distribution as well. It seems as if the approximate mean of Y is zero.

```
summary(Y_TS)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -7.20700 -1.55200  0.03250 -0.02851  1.51600  6.81500
```

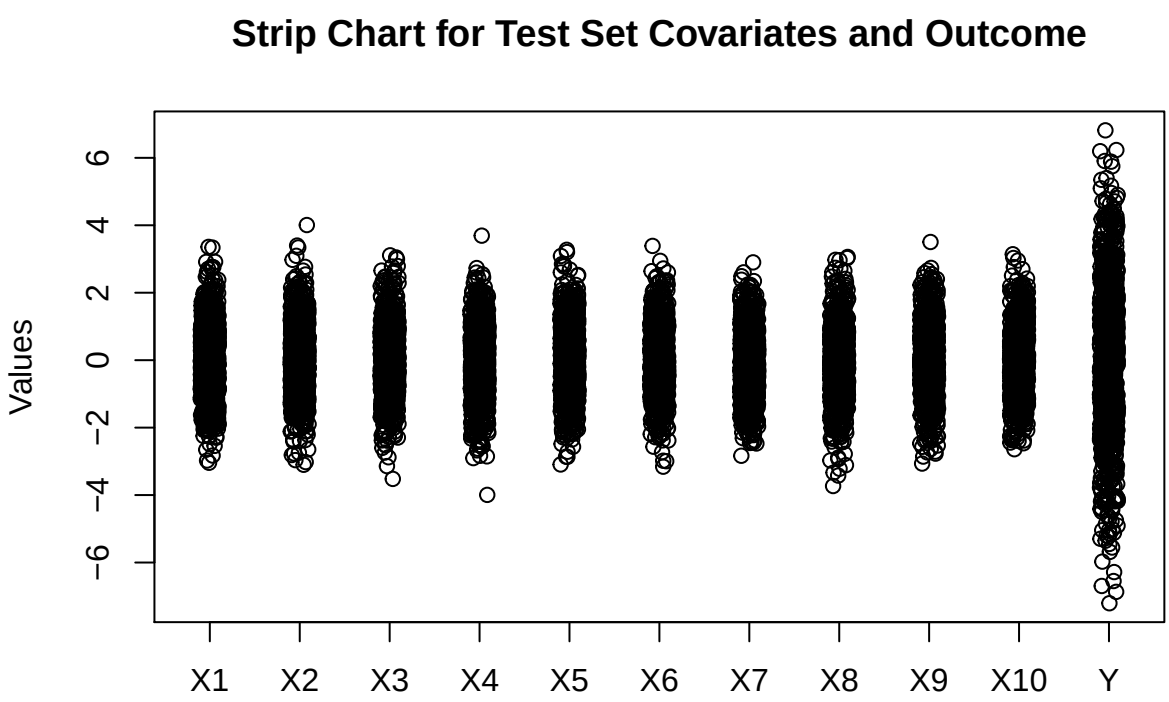
```
par(mfrow = c(2, 5))
for(i in 1:10){
  plot(X_TS[ , i], Y_TS, xlab = paste("X", i), ylab = "Test Set Y")
  abline(lm(Y_TS ~ X_TS[, i]), col="red") # regression line (y~x)
  lines(lowess(X_TS[, i],Y_TS), col="blue") # lowess line (x,y)
}
```




```
par(mfrow = c(1, 1))
```

The scatterplot of the outcome Y vs. each of the 10 covariates show that the mean of Y seems to be zero for every X_i. Therefore, I will not be centering or scaling Y as centering and scaling will not make much of a difference. The Xs also seem to be centered at zero, which is what we expect based on the simulation model that generated the data.

```
#stripchart
stripchart_list <- as.list(rep(NA, J + 1))

for(i in 1:J){
  stripchart_list[[i]] <- X_TS[ , i]
}
stripchart_list[[J + 1]] <- Y_TS

stripchart(stripchart_list,
 main="Strip Chart for Test Set Covariates and Outcome",
 xlab="",
 ylab = "Values",
 group.names=c("X1","X2","X3","X4","X5","X6","X7","X8","X9","X10", "Y"),
 vertical=TRUE,
 pch=1,
 method = "jitter"
)
```

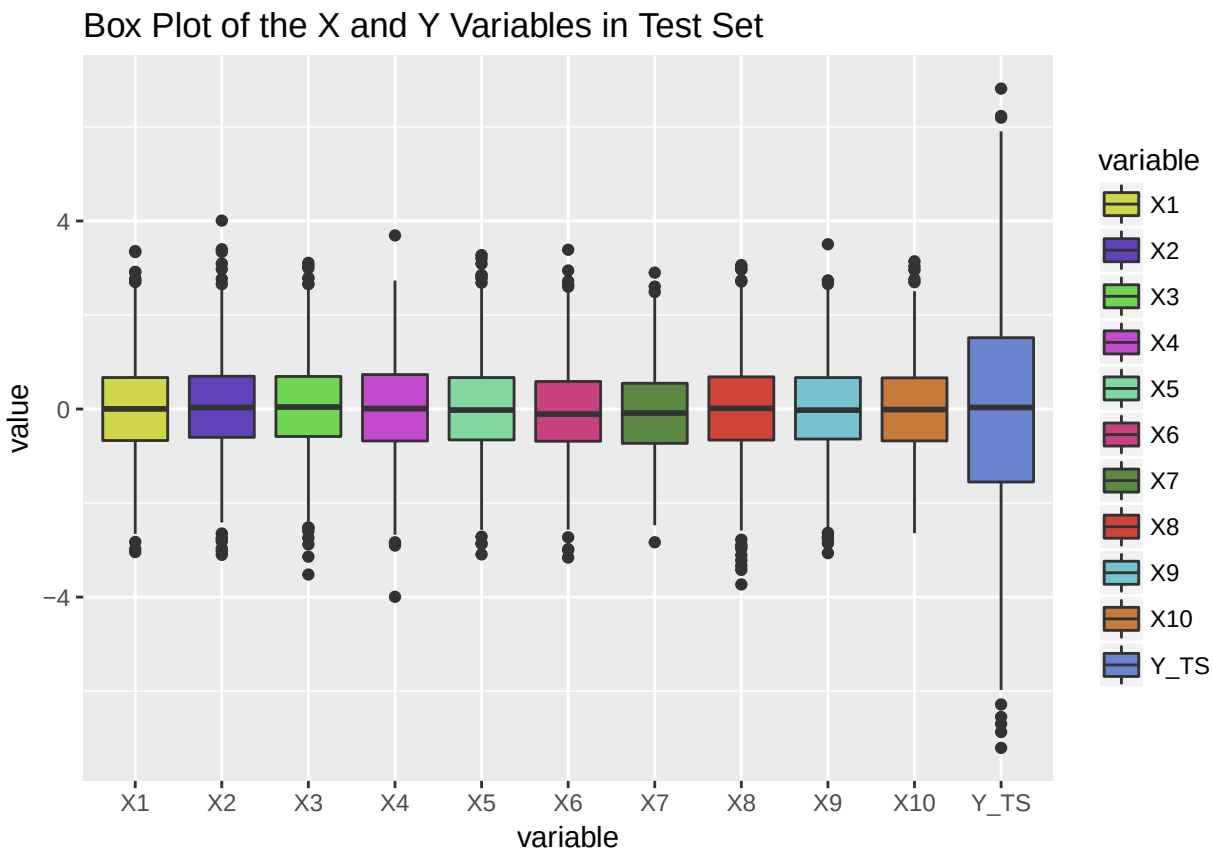


The stripchart further confirms that the Xs and the Y seem to be centered at zero. The covariates also seem to be scaled. So there is no need to transform the data before analysis.

```
TS_plot <- data.frame(X_TS, Y_TS)

TS_melt_plot <- melt(TS_plot)
```

```
## Using  as id variables
```

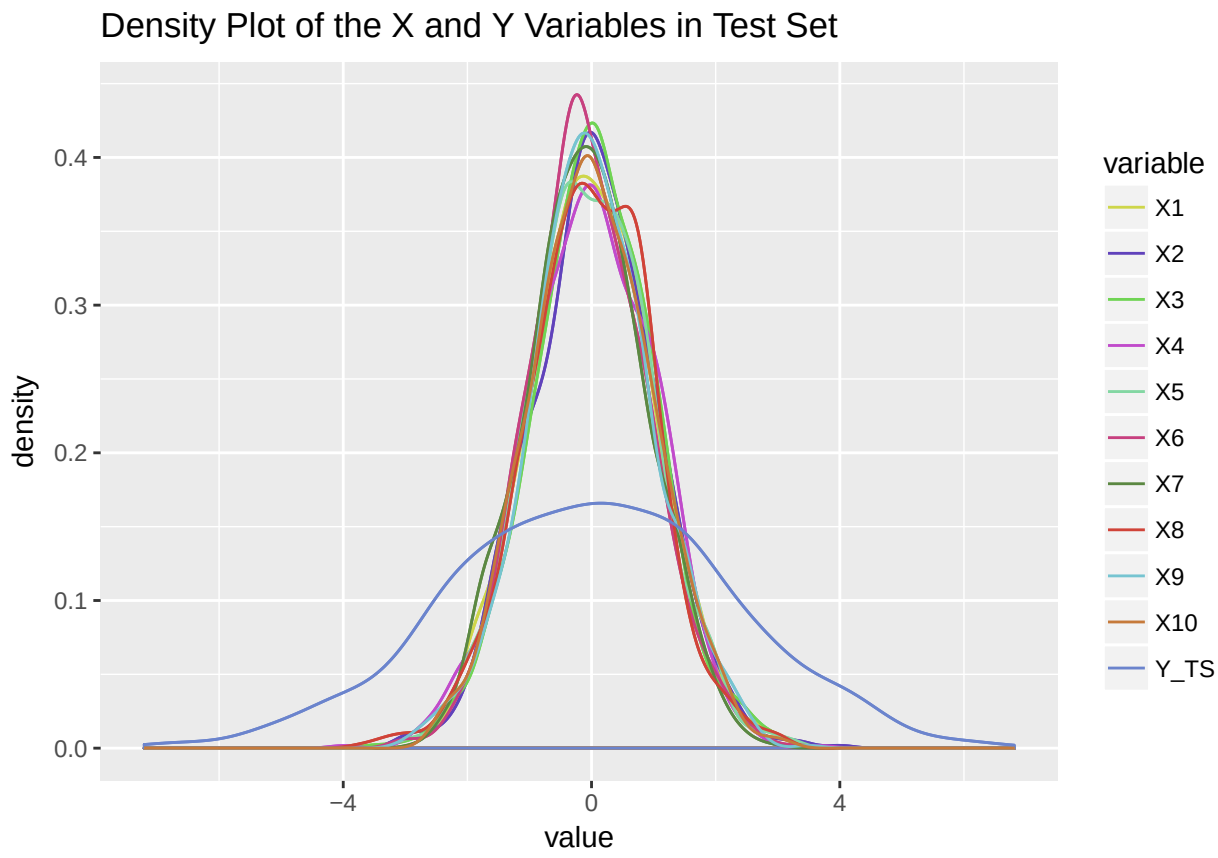
```
ggplot(TS_melt_plot, aes(x = variable, y = value, fill = variable)) +
  geom_boxplot() +
  scale_fill_manual(values = set2) +
  ggtitle("Box Plot of the X and Y Variables in Test Set")
```



Boxplot of the learning set further confirms my findings from the stripchart above.

```
ggplot(TS_melt_plot, aes(x = value, group = variable, col = variable)) +
  geom_density(show_guide = FALSE) +
  scale_colour_manual(values = set2) +
  ggtitle("Density Plot of the X and Y Variables in Test Set") +
  stat_density(aes(x = value, group = variable, col = variable), geom="line",position="identity")
```

```
## Warning: `show_guide` has been deprecated. Please use `show.legend`
## instead.
```



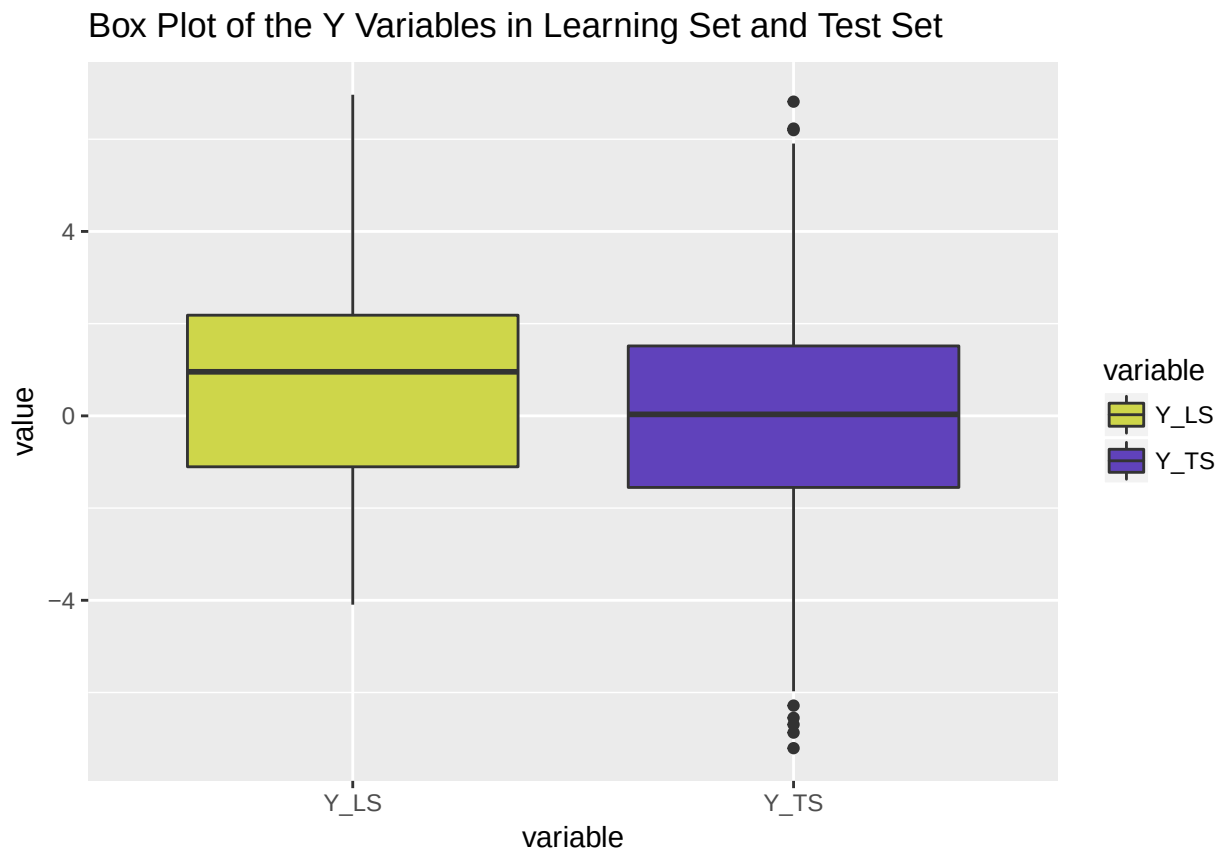
The density plot shows that the Xs have normal distributions with center at zero. Their variances are very similar as well, which suggests that they are scaled. The Y is also normally distributed, but has a larger variance.

Numerical and Graphical Summaries of the Learning Set vs the Test Set

```
Y_plot <- data.frame(Y_LS, Y_TS)

Y_melt_plot <- melt(Y_plot)
```

```
## Using  as id variables
```

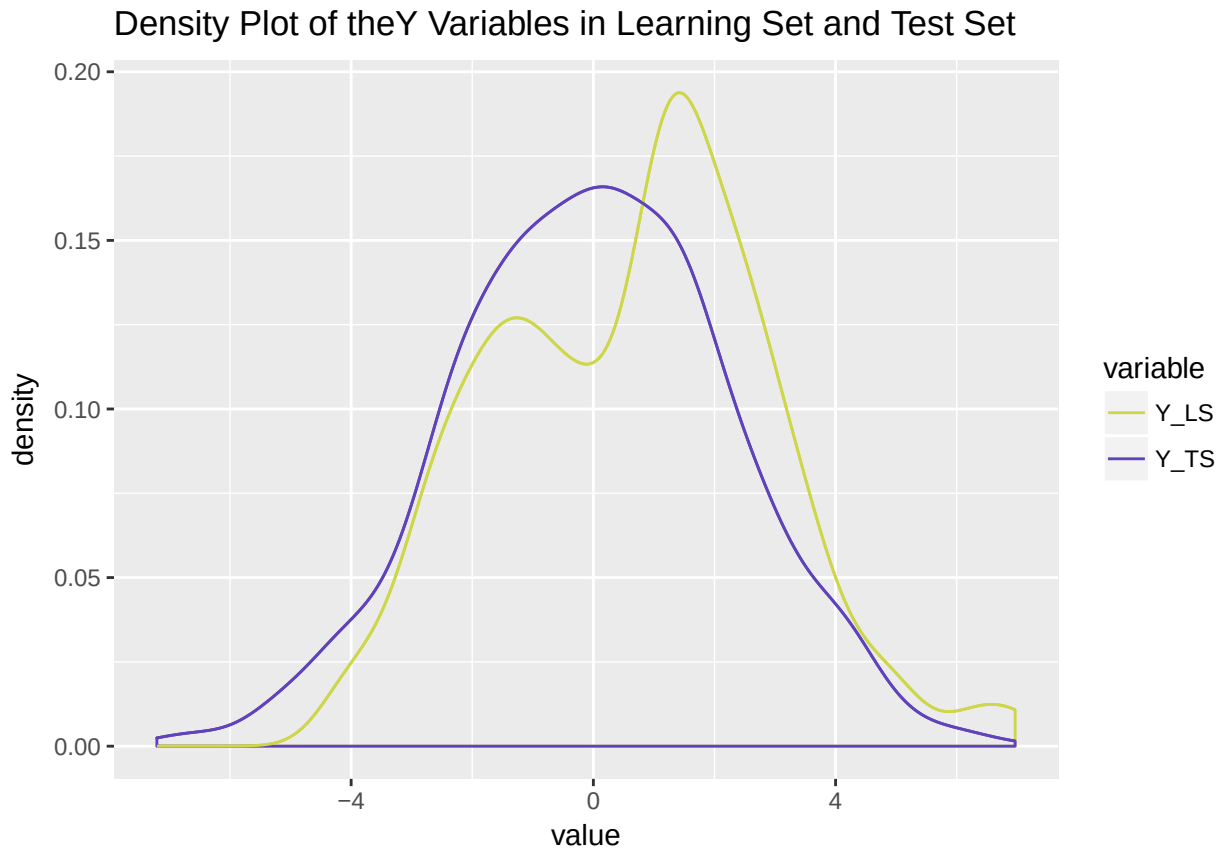
```
ggplot(Y_melt_plot, aes(x = variable, y = value, fill = variable)) +
  geom_boxplot() +
  scale_fill_manual(values = set2) +
  ggtitle("Box Plot of the Y Variables in Learning Set and Test Set")
```



The Y values for the learning set and the test set have similar median values and spread. However, since the test set has more values, it has more outliers.

```
ggplot(Y_melt_plot, aes(x = value, group = variable, col = variable)) +
  geom_density(show_guide = FALSE) +
  scale_colour_manual(values = set2) +
  ggtitle("Density Plot of theY Variables in Learning Set and Test Set") +
  stat_density(aes(x = value, group = variable, col = variable), geom="line",position="identity")
```

```
## Warning: `show_guide` has been deprecated. Please use `show.legend`
## instead.
```



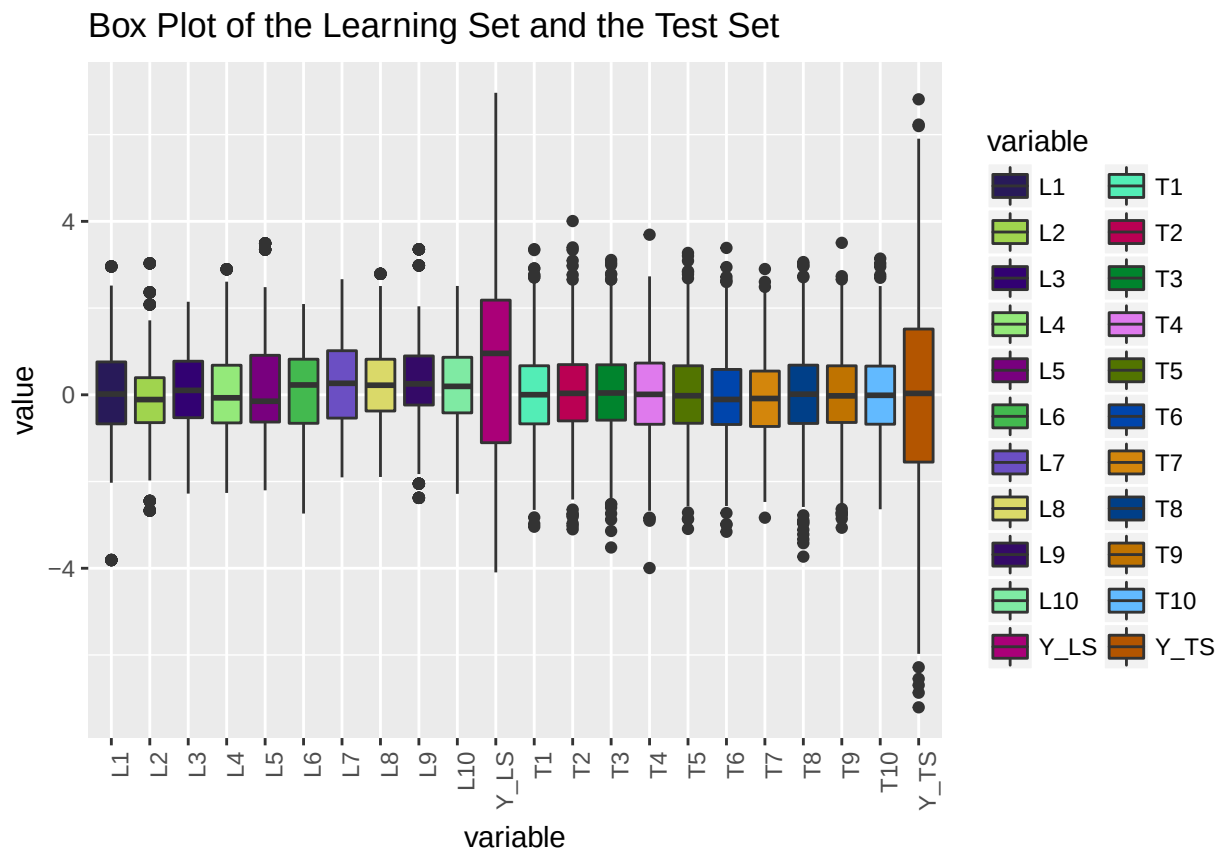
The overlay of the densities show that the distributions of the Y values of the learning set and the test set are both approximately normal and have similar centers and spread.

Now, let's look at the combined distribution of the test set and the learning set.

```
XLS <- data.frame(X_LS)

colnames(XLS) <- c("L1", "L2", "L3", "L4", "L5", "L6", "L7", "L8", "L9", "L10")

XTS <- data.frame(X_TS)

colnames(XTS) <- c("T1", "T2", "T3", "T4", "T5", "T6", "T7", "T8", "T9", "T10")

combined <- data.frame(XLS, Y_LS, XTS, Y_TS)


combined_melt <- melt(combined)
```

```
## Using  as id variables
```

```
ggplot(combined_melt, aes(x = variable, y = value, fill = variable)) +
  geom_boxplot() +
  scale_fill_manual(values = set3) +
  ggtitle("Box Plot of the Learning Set and the Test Set") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



The side-by-side boxplots show that the Xs in the learning set and the test set are centered at zero and also scaled with approximately constant variance. The Y for the learning set and the test set are also centered at zero but have wider spread. Generally, there are more outliers in the test set variables because there are more points in the test set.

```
summary(combined)
```

```
##        L1                  L2                  L3
##  Min.   :-3.81081   Min.   :-2.6705   Min.   :-2.27863
##  1st Qu.:-0.67024   1st Qu.:-0.6418   1st Qu.:-0.52821
##  Median : 0.01757   Median :-0.1101   Median : 0.10254
##  Mean   : 0.03207   Mean   :-0.1081   Mean   : 0.07561
##  3rd Qu.: 0.75872   3rd Qu.: 0.3947   3rd Qu.: 0.77416
##  Max.   : 2.96000   Max.   : 3.0310   Max.   : 2.14574
##        L4                   L5                  L6
##  Min.   :-2.262884   Min.   :-2.20280   Min.   :-2.7384
##  1st Qu.:-0.647790   1st Qu.:-0.62834   1st Qu.:-0.6570
##  Median :-0.069063   Median :-0.14872   Median : 0.2271
##  Mean   : 0.005923   Mean   : 0.08922   Mean   : 0.1047
##  3rd Qu.: 0.681936   3rd Qu.: 0.91081   3rd Qu.: 0.8210
##  Max.   : 2.894761   Max.   : 3.49326   Max.   : 2.0938
##        L7                  L8                  L9                 L10
##  Min.   :-1.9038   Min.   :-1.8955   Min.   :-2.3773   Min.   :-2.2839
##  1st Qu.:-0.5372   1st Qu.:-0.3737   1st Qu.:-0.2358   1st Qu.:-0.4165
##  Median : 0.2652   Median : 0.2212   Median : 0.2518   Median : 0.1944
##  Mean   : 0.2329   Mean   : 0.1958   Mean   : 0.2647   Mean   : 0.2400
##  3rd Qu.: 1.0166   3rd Qu.: 0.8207   3rd Qu.: 0.8964   3rd Qu.: 0.8657
```

```
##  Max.   : 2.6664   Max.   : 2.7893   Max.   : 3.3545   Max.   : 2.5098
##       Y_LS               T1                  T2
##  Min.   :-4.0955   Min.   :-3.042460   Min.   :-3.10132
##  1st Qu.:-1.1048   1st Qu.:-0.670398   1st Qu.:-0.60205
##  Median : 0.9555   Median : 0.002263   Median : 0.03359
##  Mean   : 0.6532   Mean   : 0.006395   Mean   : 0.04265
##  3rd Qu.: 2.1832   3rd Qu.: 0.669774   3rd Qu.: 0.69640
##  Max.   : 6.9648   Max.   : 3.358641   Max.   : 4.00684
##        T3                  T4                  T5
##  Min.   :-3.52074   Min.   :-3.99442   Min.   :-3.092632
##  1st Qu.:-0.58477   1st Qu.:-0.68015   1st Qu.:-0.656442
##  Median : 0.04203   Median : 0.00982   Median :-0.021456
##  Mean   : 0.04736   Mean   :-0.01013   Mean   :-0.006224
##  3rd Qu.: 0.69189   3rd Qu.: 0.73217   3rd Qu.: 0.669555
##  Max.   : 3.10899   Max.   : 3.69220   Max.   : 3.272345
##        T6                  T7                  T8
##  Min.   :-3.15854   Min.   :-2.83389   Min.   :-3.73162
##  1st Qu.:-0.68513   1st Qu.:-0.73067   1st Qu.:-0.66062
##  Median :-0.10750   Median :-0.08502   Median : 0.01403
##  Mean   :-0.04802   Mean   :-0.08390   Mean   :-0.01830
##  3rd Qu.: 0.58529   3rd Qu.: 0.54771   3rd Qu.: 0.68402
##  Max.   : 3.38773   Max.   : 2.90054   Max.   : 3.06104
##        T9                 T10                 Y_TS
##  Min.   :-3.06557   Min.   :-2.640212   Min.   :-7.2075
##  1st Qu.:-0.63830   1st Qu.:-0.678099   1st Qu.:-1.5522
##  Median :-0.02444   Median :-0.011546   Median : 0.0325
##  Mean   : 0.01145   Mean   : 0.009053   Mean   :-0.0285
##  3rd Qu.: 0.66856   3rd Qu.: 0.662550   3rd Qu.: 1.5165
##  Max.   : 3.50443   Max.   : 3.141329   Max.   : 6.8148
```

```
median_values_combined <- apply(combined, 2, median)
median_values_combined
```

```
##          L1          L2          L3          L4          L5          L6
##  0.01756459 -0.11010383  0.10253678 -0.06906300 -0.14872019  0.22712601
##          L7          L8          L9         L10        Y_LS          T1
##  0.26518565  0.22115033  0.25176269  0.19438757  0.95545338  0.00226334
##          T2          T3          T4          T5          T6          T7
##  0.03358724  0.04202842  0.00982018 -0.02145632 -0.10750036 -0.08501581
##          T8          T9         T10        Y_TS
##  0.01402994 -0.02443606 -0.01154555  0.03249715
```

```
mean_values_combined <- apply(combined, 2, mean)
mean_values_combined
```

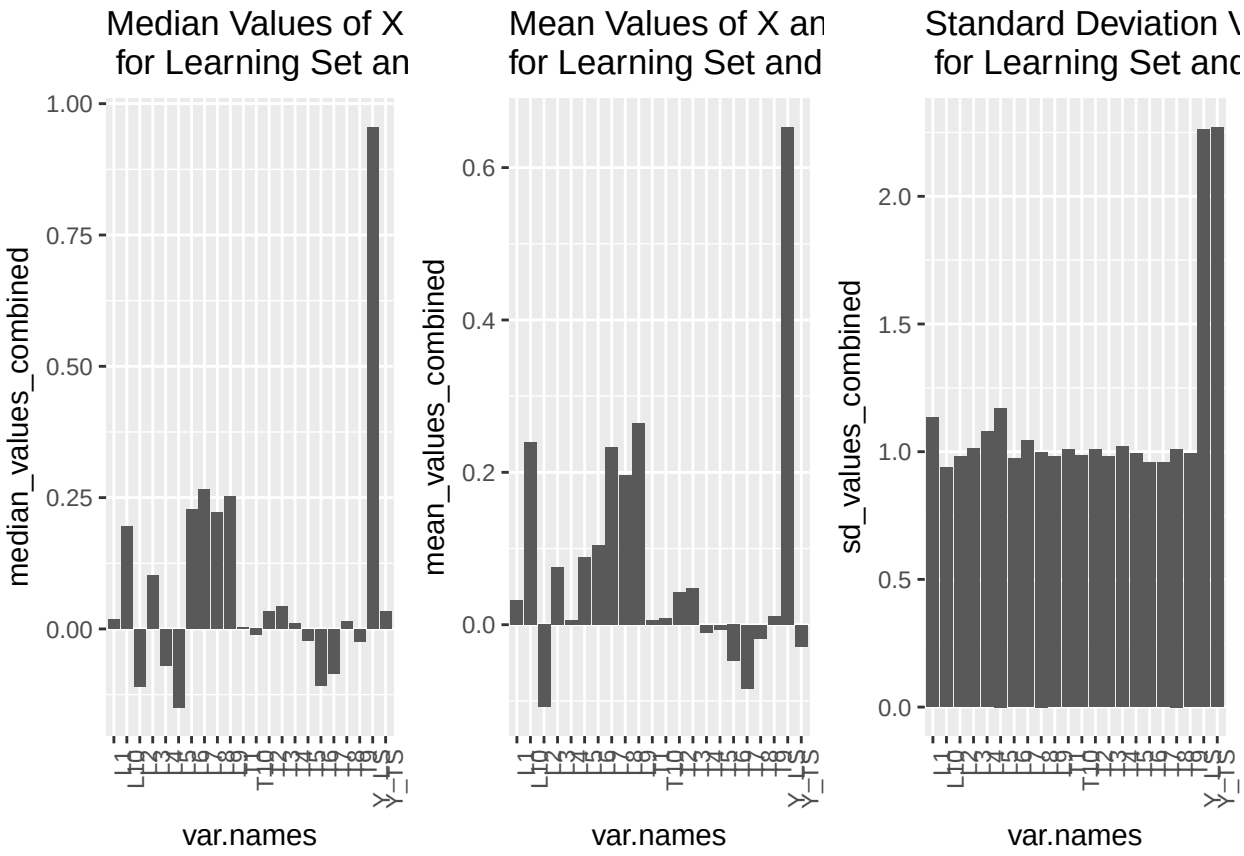
```
##           L1           L2           L3           L4           L5
##  0.032074719 -0.108092659  0.075607439  0.005923298  0.089214688
##           L6           L7           L8           L9          L10
##  0.104699806  0.232913629  0.195814293  0.264721729  0.239994440
##         Y_LS           T1           T2           T3           T4
##  0.653156440  0.006395463  0.042654674  0.047363443 -0.010125141
##           T5           T6           T7           T8           T9
## -0.006224161 -0.048016854 -0.083904736 -0.018302560  0.011451575
##          T10         Y_TS
##  0.009052806 -0.028505045
```

```r
sd_values_combined <- apply(combined, 2, sd)
sd_values_combined
```

```
##        L1        L2        L3        L4        L5        L6        L7
## 1.1324225 0.9816884 1.0131093 1.0800733 1.1713490 0.9718900 1.0440624
##        L8        L9       L10      Y_LS        T1        T2        T3
## 0.9989706 0.9819083 0.9386467 2.2630848 1.0082428 1.0072748 0.9796254
##        T4        T5        T6        T7        T8        T9       T10
## 1.0193520 0.9940922 0.9563539 0.9563436 1.0105554 0.9920520 0.9843599
##      Y_TS
## 2.2711190
```

```r
numbers_combined <- data.frame(median_values_combined, mean_values_combined, sd_values_combined)

var.names <- rownames(numbers_combined)

numbers_combined <- data.frame(numbers_combined, var.names)

p1 <- ggplot(numbers_combined, aes(x = var.names, y = median_values_combined)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Median Values of X and Y\n for Learning Set and Test Set")

p2 <- ggplot(numbers_combined, aes(x = var.names, y = mean_values_combined)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Mean Values of X and Y\nfor Learning Set and Test Set")

p3 <- ggplot(numbers_combined, aes(x = var.names, y = sd_values_combined)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = set2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Standard Deviation Values of X and Y\n for Learning Set and Test Set")

multiplot(p1, p2, p3, cols = 3)
```

```
## Loading required package: grid
```



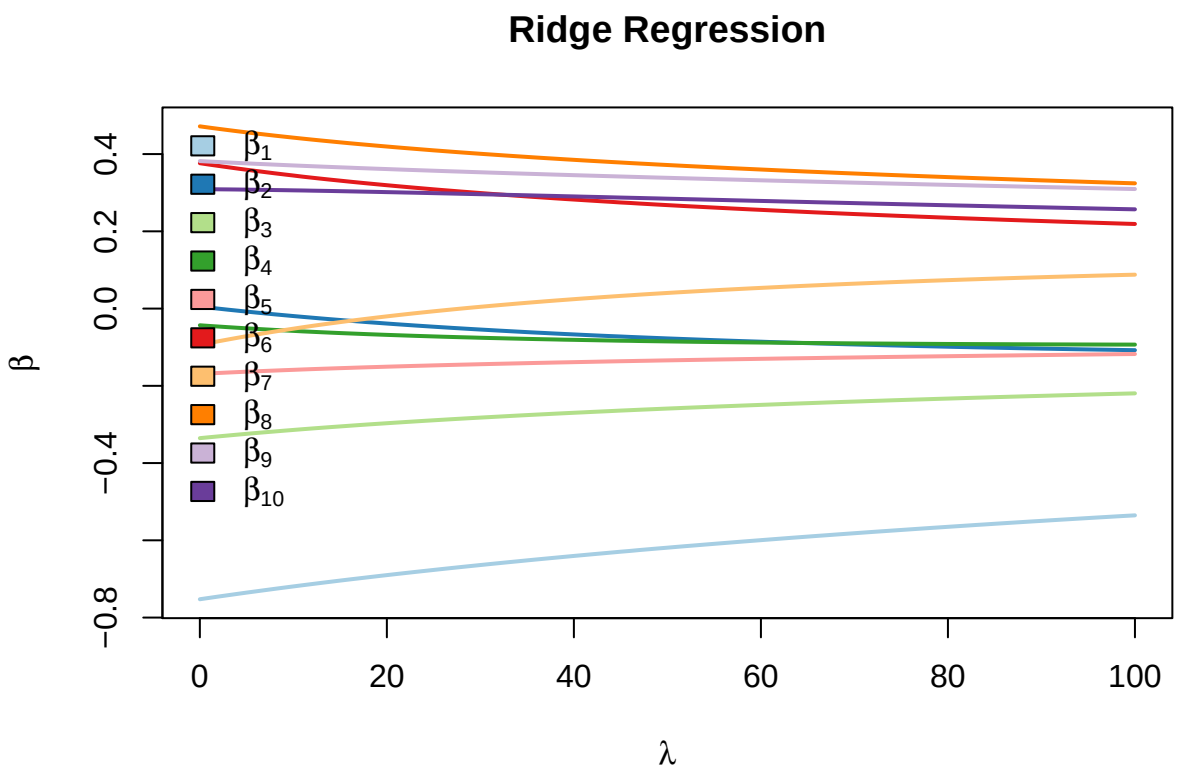
These numerical summaries and barplots further confirm that the Xs and the Ys for the learning set and the test are centered approximately at zero. The Xs for the learning set and the test set have constanst standard deviation. The Ys have larger standard deviation but they are constant for the learning set and the test set.

5.b. Obtain ridge, LASSO, and elastic net estimators of the regression coefficients β, for $\lambda \in \{0, 1, ..., 100\}$ based on the learning set simulated in (a).

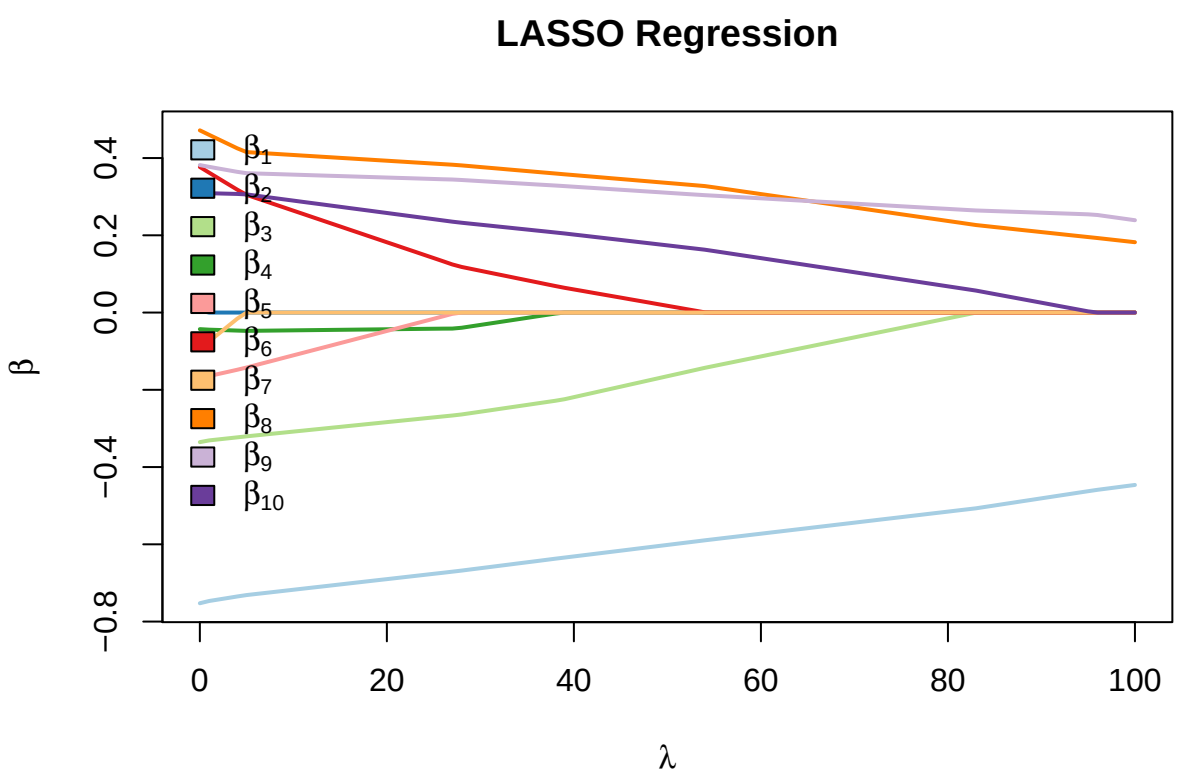
```
X_LS <- as.matrix(X_LS)

# Ridge Regression
lambda_ridge <- lambda / LS_sample_size
ridge.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 0,
                    lambda = lambda_ridge,
                    intercept = FALSE,
                    standardize = FALSE
                    )

# LASSO regression
lambda_lasso <- lambda / (2 * LS_sample_size)
lasso.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 1,
                    lambda = lambda_lasso,
```

```
                    intercept = FALSE,
                    standardize = FALSE
                    )

# Elastic Net Regression
lambda_elnet <- lambda * 3 / (4 * LS_sample_size)
elnet.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 1/3,
                    lambda = lambda_elnet,
                    intercept = FALSE,
                    standardize = FALSE
                    )

beta_ridge.mod <- ridge.mod$beta
beta_lasso.mod <- lasso.mod$beta
beta_elnet.mod <- elnet.mod$beta
```

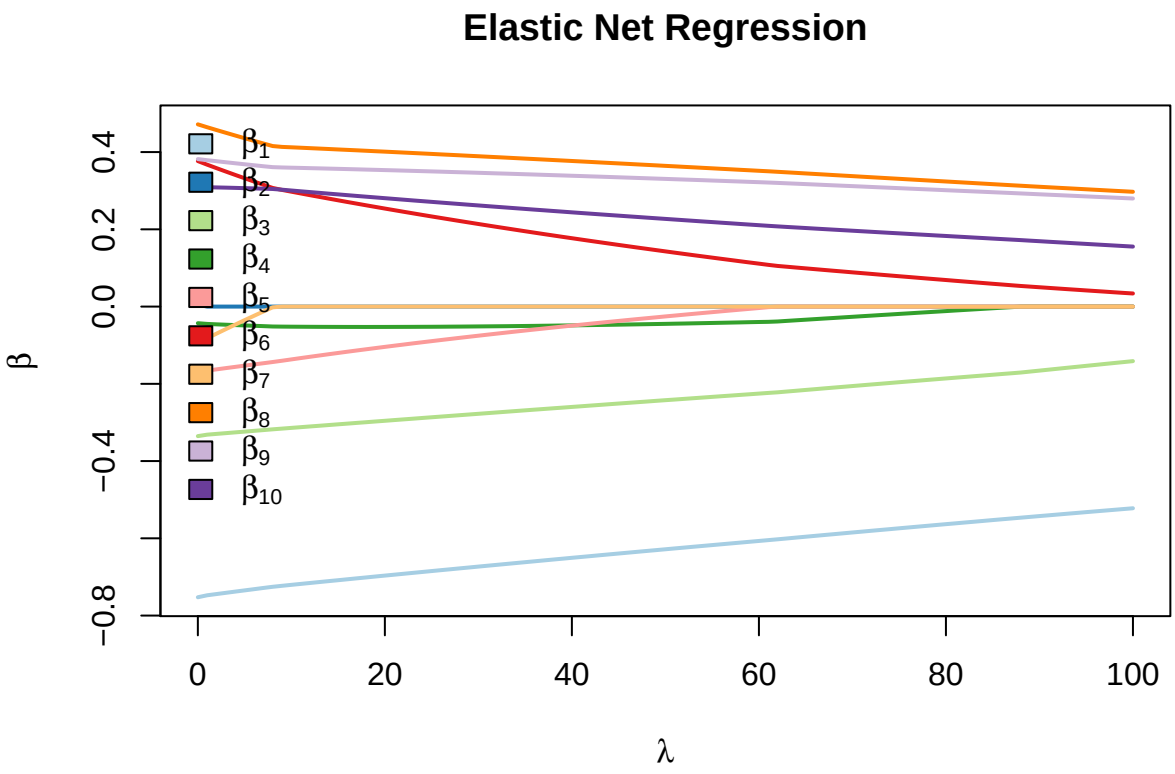
```
coefficient_labels <- c(expression(beta[1]),
  expression(beta[2]),
  expression(beta[3]),
  expression(beta[4]),
  expression(beta[5]),
  expression(beta[6]),
  expression(beta[7]),
  expression(beta[8]),
  expression(beta[9]),
  expression(beta[10]))

#ridge regression
matplot(rev_lambda, t(beta_ridge.mod),
        type = "l",
        xlab = expression(lambda),
        ylab = expression(beta),
        lty = 1, lwd = "2",
        main = "Ridge Regression",
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```




```r
#lasso regression
matplot(rev_lambda, t(beta_lasso.mod),
        type = "l",
        xlab = expression(lambda),
        ylab = expression(beta),
        lty = 1, lwd = "2",
        main = "LASSO Regression",
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```




```r
#elastic net regression
matplot(rev_lambda, t(beta_elnet.mod),
        type = "l",
        xlab = expression(lambda),
        ylab = expression(beta),
        lty = 1, lwd = "2",
        main = "Elastic Net Regression",
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



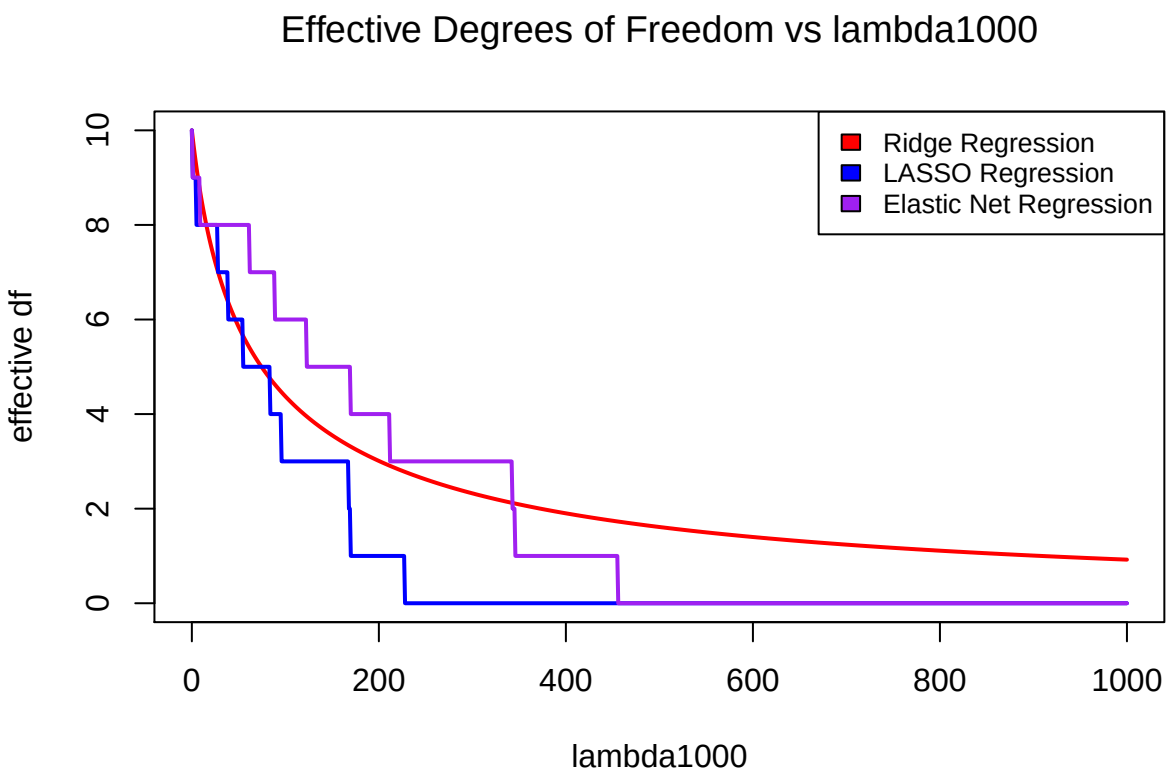
For all three plots, the values of β decrease as λ increase. This is expected because the regularized regression models are adding penalties to the empirical risk to prevent the model from overfitting the data. So, these regularized regression models prevent the β values from becoming too large. That is why as λ increase, the β values go to zero.

5.b. (continued) For each type of estimator, provide and comment on plots of the effective degrees of freedom versus the shrinkage parameter.

For the sake of illustration, I will use $\lambda \in \{0, 1, ..., 1000\}$ instead of $\lambda \in \{0, 1, ..., 1000\}$.

```
# For the sake of illustration, I increase my lambda1000 values from 100 to 1000.
lambda1000 <- 0:1000
rev_lambda1000 <- rev(lambda1000)
nlambda1000 <- length(rev_lambda1000)

X_LS <- as.matrix(X_LS)

# Ridge Regression
lambda1000_ridge <- lambda1000 / LS_sample_size
ridge.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 0,
                    lambda = lambda1000_ridge,
                    intercept = FALSE,
                    standardize = FALSE
                    )
```

```r
# LASSO regression
lambda1000_lasso <- lambda1000 / (2 * LS_sample_size)
lasso.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 1,
                    lambda = lambda1000_lasso,
                    intercept = FALSE,
                    standardize = FALSE
                    )

# Elastic Net Regression
lambda1000_elnet <- lambda1000 * 3 / (4 * LS_sample_size)
elnet.mod <- glmnet(X_LS,
                    Y_LS,
                    alpha = 1/3,
                    lambda = lambda1000_elnet,
                    intercept = FALSE,
                    standardize = FALSE
                    )

# ridge regression degrees of freedom
L_j <- svd(X_LS)$d

effective_df <- function(lambda1000){
  eff_df <- c()
  for(i in 1:length(lambda1000)){
    eff_df <- c(eff_df, sum(L_j^2/(L_j^2 + lambda1000[i])))
  }
  return(eff_df)
}

eff_df <- effective_df(lambda1000)

#effective df plot
plot(lambda1000, elnet.mod$df, type="n",
     xlab = expression(lambda1000),
     ylab = "effective df",
     ylim = c(0,10),
     xlim = c(0, 1000),
     main = expression(paste("Effective Degrees of Freedom vs ",
                     lambda1000)))
lines(lambda1000, eff_df, col = "red", lwd = 2) #ridge df
lines(rev_lambda1000, lasso.mod$df, col = "blue", lwd = 2) #lasso df
lines(rev_lambda1000, elnet.mod$df, col = "purple", lwd = 2) #elastic net df
legend("topright",
       c("Ridge Regression", "LASSO Regression", "Elastic Net Regression"),
       col=c("red", "blue", "purple"),
       cex=0.8,
       fill=c("red", "blue", "purple"))
```

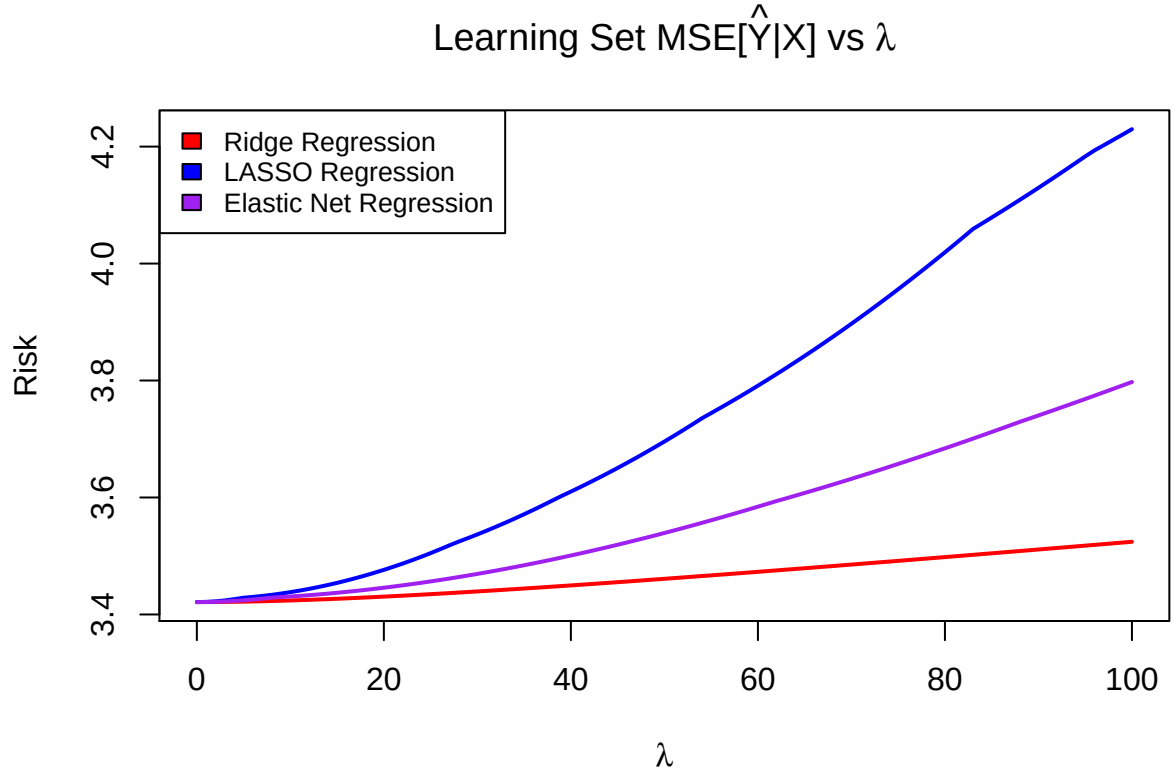


The degrees of freedom decreases for all three of the models as λ increases. The degrees of freedom for LASSO and elastic net regression models correspond to the number of non-zero coefficients. The effective degrees of freedom for ridge regression was computed separately. As you can see, as $\lambda \to \infty$, the degrees of freedom goes to zero. Also, the degrees of freedom plot shows that LASSO and elastic net produce sparse results, while ridge regresion does not.

5.b. For each type of estimator, obtain the learning set risk for the squared error loss function, i.e., the mean squared error (MSE). Provide and comment on plots of the MSE versus the shrinkage parameter and report which values of the shrinkage parameter minimize risk.

```
#ridge MSE
MSE_ridge_LS <- apply((Y_LS - X_LS %*% beta_ridge.mod),
                      2,
                      function(x) mean(x^2))

#LASSO MSE
MSE_lasso_LS <- apply((Y_LS - X_LS %*% beta_lasso.mod),
                      2,
                      function(x) mean(x^2))

#Elastic Net MSE
MSE_elnet_LS <- apply((Y_LS - X_LS %*% beta_elnet.mod),
                      2,
```

```
                      function(x) mean(x^2))

#Plot of MSE and Lambda Learning Set
plot(lambda, MSE_lasso_LS, type="n",
     xlab = expression(lambda),
     ylab = "Risk",
     main = expression(paste("Learning Set MSE[",hat(Y),"|X] vs ", lambda)))
lines(rev_lambda, MSE_ridge_LS, col = "red", lwd = 2) #ridge MSE
lines(rev_lambda, MSE_lasso_LS, col = "blue", lwd = 2) #lasso MSE
lines(rev_lambda, MSE_elnet_LS, col = "purple", lwd = 2) #elastic net MSE
legend("topleft",
       c("Ridge Regression", "LASSO Regression", "Elastic Net Regression"),
       col=c("red", "blue", "purple"),
       cex=0.8,
       fill=c("red", "blue", "purple"))
```




```
#lambda values that correspond to the minimum MSE
min_lambda_ridge_MSE_LS <- nlambda - which(MSE_ridge_LS == min(MSE_ridge_LS))
min_lambda_ridge_MSE_LS
```

```
## s100
##    0
```

```
min_lambda_lasso_MSE_LS <- nlambda - which(MSE_lasso_LS == min(MSE_lasso_LS))
min_lambda_lasso_MSE_LS
```

```
## s100
```

```
##    0
```

```
min_lambda_elnet_MSE_LS <- nlambda - which(MSE_elnet_LS == min(MSE_elnet_LS))
min_lambda_elnet_MSE_LS
```

```
## s100
##    0
```

```
#corresponding beta values for minimum lambda values with minimum MSE
beta_ridge.mod[ , nlambda - min_lambda_ridge_MSE_LS]
```

```
##           V1           V2           V3           V4           V5
## -0.752570530  0.005160694 -0.335514929 -0.043241198 -0.169074101
##           V6           V7           V8           V9          V10
##  0.376412331 -0.093998429  0.471810839  0.381861797  0.309198824
```

```
beta_lasso.mod[ , nlambda - min_lambda_lasso_MSE_LS]
```

```
##           V1           V2           V3           V4           V5
## -0.752540932  0.005047792 -0.335405602 -0.042921621 -0.169563692
##           V6           V7           V8           V9          V10
##  0.376872264 -0.094117313  0.471740942  0.381808537  0.309239002
```

```
beta_elnet.mod[ , nlambda - min_lambda_elnet_MSE_LS]
```

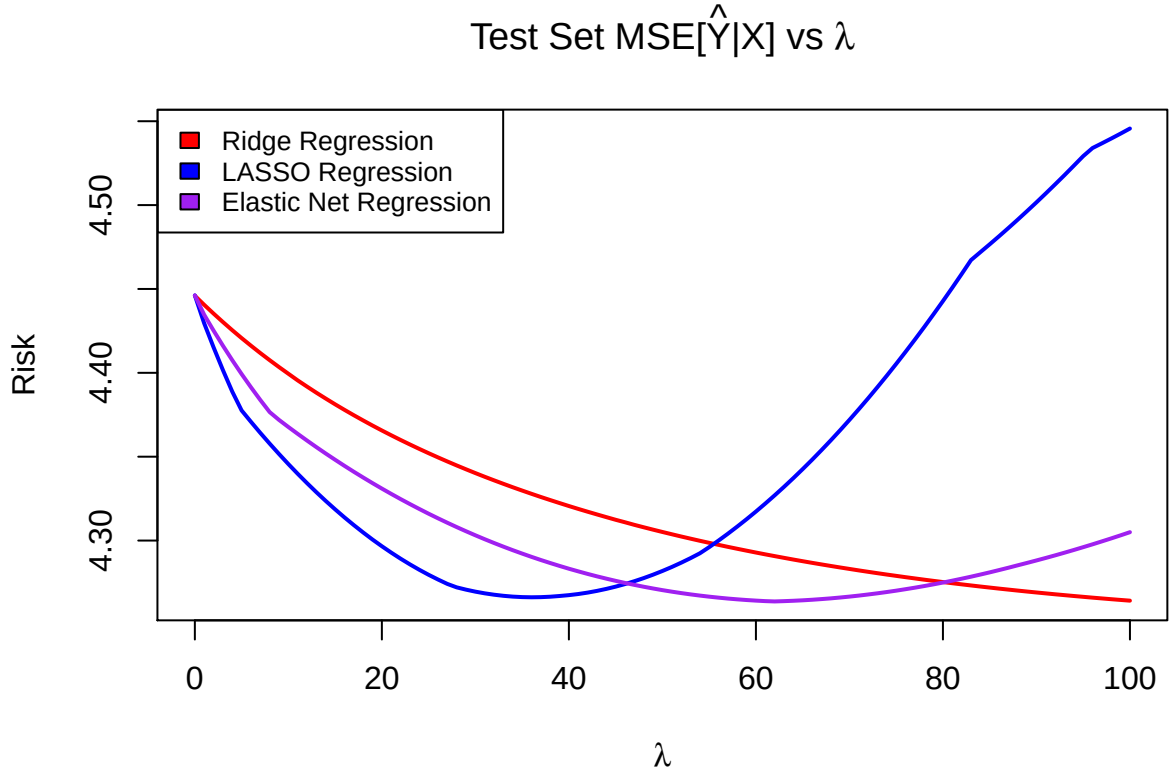
```
##           V1           V2           V3           V4           V5
## -0.752544145  0.005076604 -0.335426902 -0.043047468 -0.169373261
##           V6           V7           V8           V9          V10
##  0.376739520 -0.094126617  0.471781808  0.381871202  0.309193486
```

The mean squared error increases with λ for all three models. This is expected because in the learning set, the model is most data-adaptive when λ is zero. The MSE is minimized when λ is zero for all regression models. When $\lambda = 0$, the regression models become the OLS estimator. In the OLS estimator model, all the residuals sum to zero. However, as the shrinkage parameter values increase, the sum of the residuals are no longer zero as bias is introduced into the model. This can be seen in the plot above.

5.c. For each estimator in (b), obtain the test set risk MSE for the squared error los function. Provide an dcomment on plots of risk versus the shrinkage parameter and report which values of the shrinkage parameter minimize risk. Examine the corresponding three "optiml" estimators of the regression coefficients.

```
#ridge MSE
MSE_ridge_TS <- apply((Y_TS - X_TS %*% beta_ridge.mod),
                      2,
                      function(x) mean(x^2))

#LASSO MSE
MSE_lasso_TS <- apply((Y_TS - X_TS %*% beta_lasso.mod),
                      2,
                      function(x) mean(x^2))

#Elastic Net MSE
MSE_elnet_TS <- apply((Y_TS - X_TS %*% beta_elnet.mod),
                      2,
                      function(x) mean(x^2))
```

```r
#Plot of MSE and Lambda Test Set
plot(lambda, MSE_ridge_TS, type="n",
     xlab = expression(lambda),
     ylab = "Risk",
     main = expression(paste("Test Set MSE[",hat(Y),
                             "|X] vs ", lambda)),
     ylim = c(min(MSE_ridge_TS, MSE_lasso_TS, MSE_elnet_TS),
              max(MSE_ridge_TS, MSE_lasso_TS, MSE_elnet_TS)))
lines(rev_lambda, MSE_ridge_TS, col = "red", lwd = 2) #ridge MSE
lines(rev_lambda, MSE_lasso_TS, col = "blue", lwd = 2) #lasso MSE
lines(rev_lambda, MSE_elnet_TS, col = "purple", lwd = 2) #elastic net MSE
legend("topleft",
       c("Ridge Regression", "LASSO Regression", "Elastic Net Regression"),
       col=c("red", "blue", "purple"),
       cex=0.8,
       fill=c("red", "blue", "purple"))
```



MSE[$\hat{\beta}$|X] for the test set decreases and then increases as λ increases for LASSO and elastic net regression. From the plot, the ridge regression MSE is seemingly only decreasing. However, when the λ values increase, the MSE for ridge regression will clearly increase as well.

```r
#lambda values that correspond to the minimum MSE
min_lambda_ridge_MSE_TS <- nlambda - which(MSE_ridge_TS == min(MSE_ridge_TS))
min_lambda_ridge_MSE_TS
```

```
##  s0
## 100
```

```
min_lambda_lasso_MSE_TS <- nlambda - which(MSE_lasso_TS == min(MSE_lasso_TS))
min_lambda_lasso_MSE_TS
```

```
## s64
##  36
```

```
min_lambda_elnet_MSE_TS <- nlambda - which(MSE_elnet_TS == min(MSE_elnet_TS))
min_lambda_elnet_MSE_TS
```

```
## s38
##  62
```

```
#corresponding beta values for minimum lambda values with minimum MSE
beta_ridge.mod[ , nlambda - min_lambda_ridge_MSE_TS]
```

```
##          V1          V2          V3          V4          V5          V6
## -0.53535320 -0.10775468 -0.21949277 -0.09328068 -0.11767259  0.21904379
##          V7          V8          V9         V10
##  0.08775485  0.32426274  0.30948463  0.25695578
```

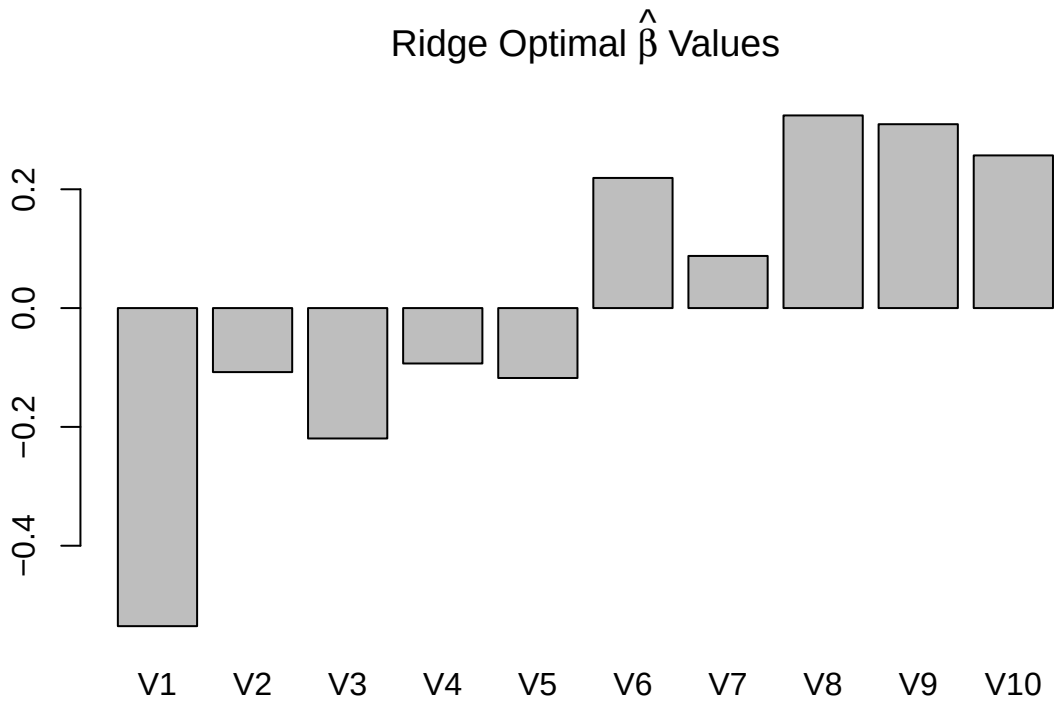
```
beta_lasso.mod[ , nlambda - min_lambda_lasso_MSE_TS]
```

```
##          V1          V2          V3          V4          V5          V6
## -0.64345035  0.00000000 -0.23548828 -0.01002005  0.00000000  0.07864123
##          V7          V8          V9         V10
##  0.00000000  0.36422233  0.33171273  0.21241811
```

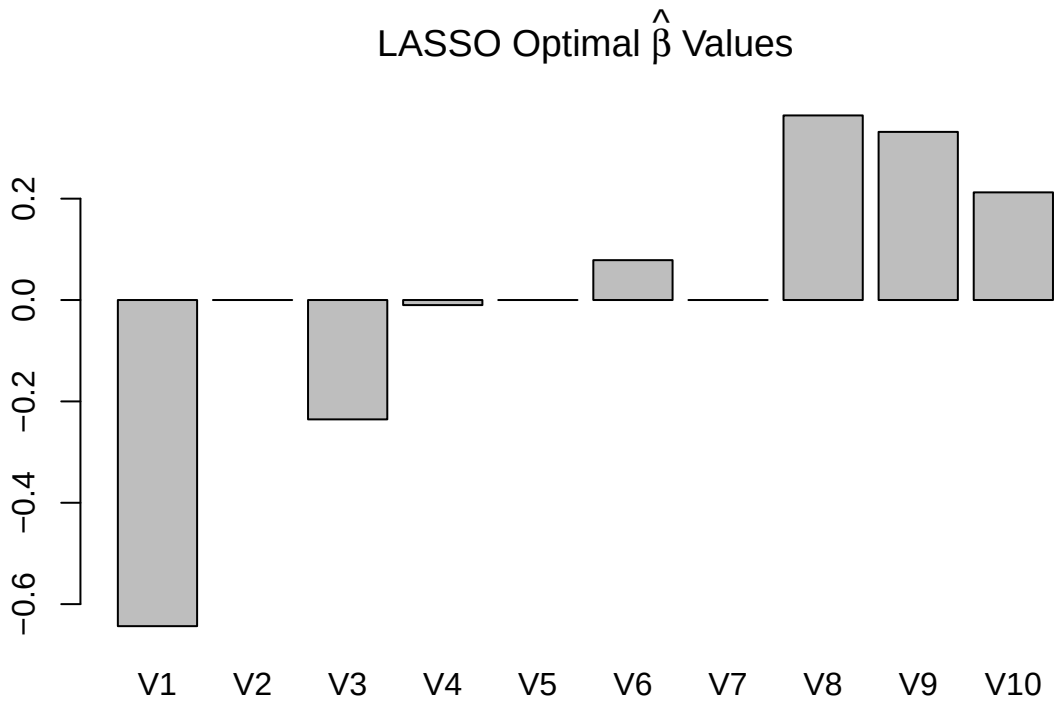
```
beta_elnet.mod[ , nlambda - min_lambda_elnet_MSE_TS]
```

```
##          V1          V2          V3          V4          V5          V6
## -0.60253294  0.00000000 -0.22196877 -0.03858729  0.00000000  0.10519904
##          V7          V8          V9         V10
##  0.00000000  0.34894459  0.32022349  0.20755469
```

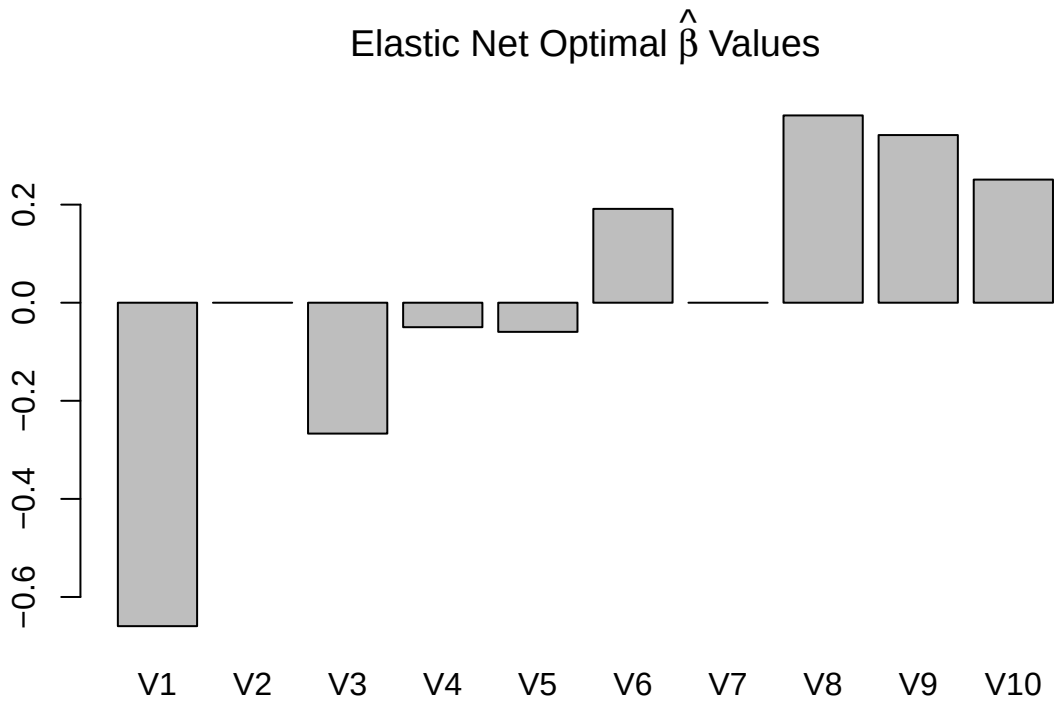
```
barplot(beta_ridge.mod[ , nlambda - min_lambda_ridge_MSE_TS],
        main = expression(paste("Ridge Optimal ", hat(beta), " Values")))
```




```
barplot(beta_lasso.mod[ , nlambda - min_lambda_lasso_MSE_TS],
        main = expression(paste("LASSO Optimal ", hat(beta), " Values")))
```




```
barplot(beta_elnet.mod[ , nlambda - min_lambda_lasso_MSE_TS],
        main = expression(paste("Elastic Net Optimal ", hat(beta), " Values")))
```



From the MSE plot for the test set and the outputs above, I conclude that the optimal values for λ increases from LASSO to elastic net to ridge regression as expected. The optimal λ values for LASSO, elastic net, and ridge regression are 36, 62, and 100, respectively.

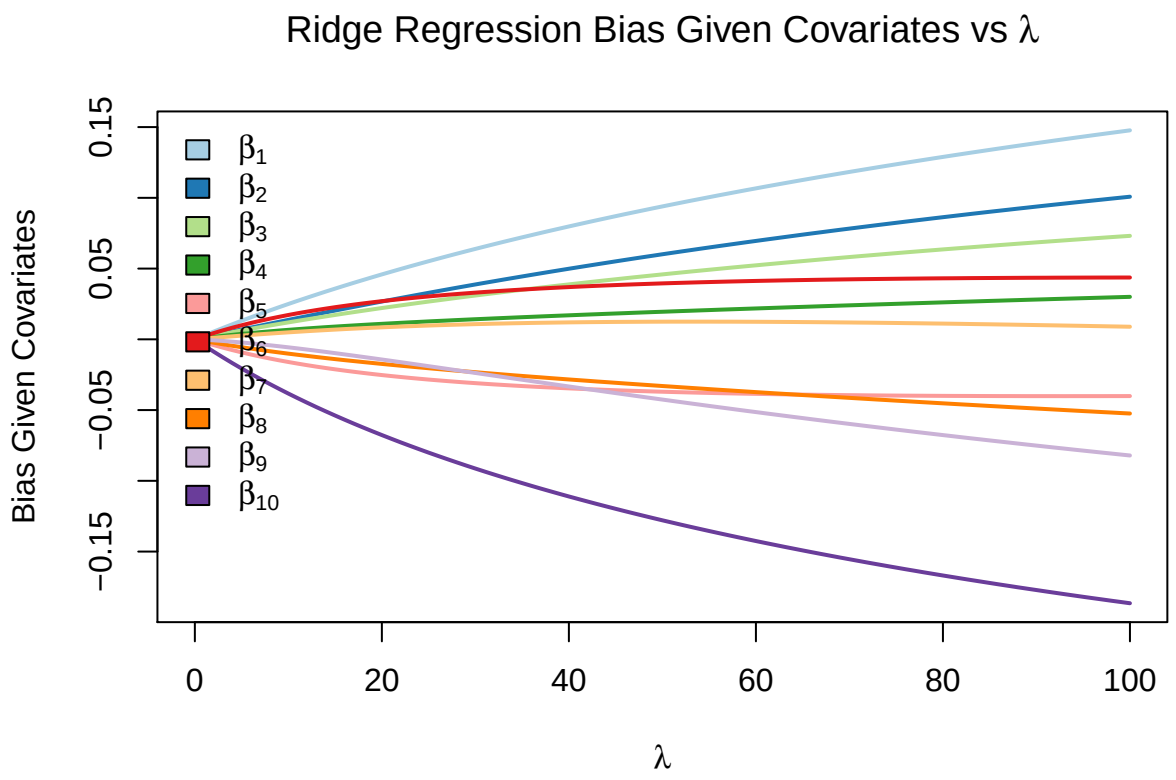
5.d. Derive the bias, variance, and mean squared error of the ridge estimators of the regression coefficients. Be specific about assumptions and which variables you are conditioning on. For the simulation model of (a), provide and comment on graphical displays of the bias, variance, and MSE of the ridge estimators based on the learning set. For each coefficient, provide the value of the shrinkage parameter λ minimizing the MSE and the corresponding estimate.

The variables I am conditioning on are the covariates X. The assumptions of linearity and constant variance of the errors do not have to be true for ridge regression. I do not have to make any distributional assumptions of the errors.

```
I <- diag(J)

beta_matrix <- matrix(beta, nrow = 10, ncol = 1)

bias <- c()

#Ridge regression Bias, Variance, and MSE

bias_ridge <- matrix(NA, nrow = J, ncol = nlambda)
variance_ridge <- matrix(NA, nrow = J, ncol = nlambda)
```

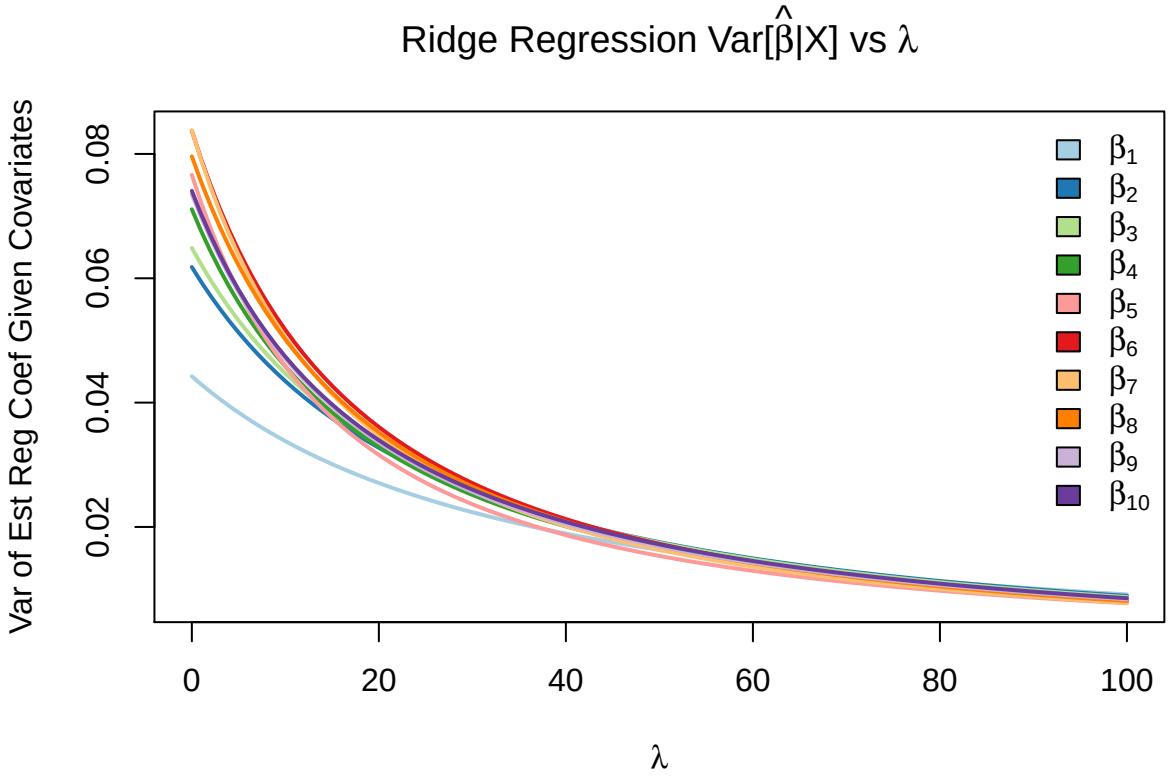
```r
#bias
for(i in 1:nlambda){
  bias_ridge[, i] <- solve(t(X_LS) %*% X_LS + lambda[i] * I) %*%
    t(X_LS) %*% X_LS %*% beta - beta
}

#variance
for(i in 1:nlambda){
  variance_ridge[, i] <- diag(4 * solve(t(X_LS) %*% X_LS + lambda[i] * I) %*%
                                t(X_LS) %*% X_LS %*% solve(t(X_LS) %*% X_LS +
                                                             lambda[i] * I))
}

#MSE
MSE_ridge <- bias_ridge^2 + variance_ridge
```

```r
#Plot of bias and lambda
matplot(lambda, t(bias_ridge),
        type = "l",
        xlab = expression(lambda),
        ylab = "Bias Given Covariates", lty=1, lwd = "2",
        main = expression(paste("Ridge Regression Bias Given Covariates vs ",
                                lambda)),
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



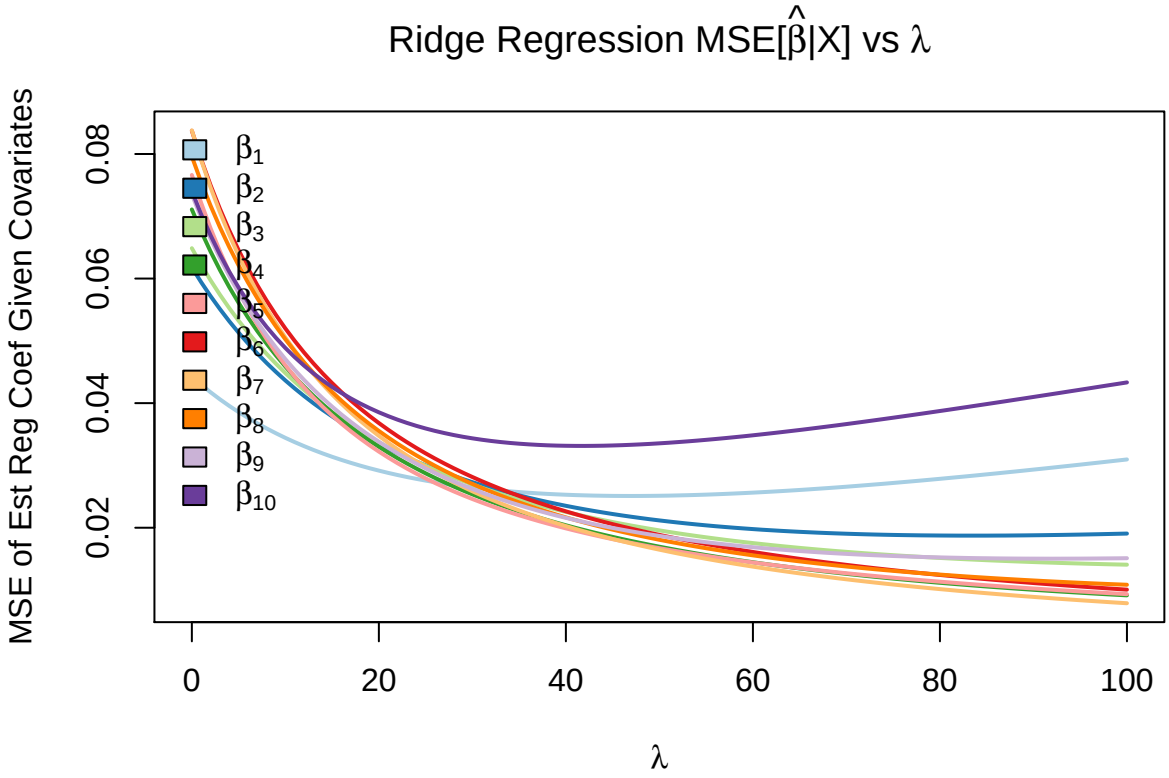
The plot of bias vs. λ shows that the bias increases as λ increases. This is expected because when we add penalties to the risk, we do a trade off between bias and variance. When $\lambda = 0$, bias $= 0$ (OLS). However, as λ increases, the estimates of β will no longer be unbiased.

```
#Plot of variance and lambda
matplot(lambda, t(variance_ridge),
        type = "l",
        xlab = expression(lambda),
        ylab = "Var of Est Reg Coef Given Covariates",
        lty = 1, lwd = "2",
        main = expression(paste("Ridge Regression Var[",
                                hat(beta), "|X] vs " , lambda)),
        col = myPaletteSeq)
legend("topright", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



The plot of variance vs. λ shows that the variance decreases as λ increases. This is expected because when we add penalties to the risk, we increase the bias of the estimates. However, we decrease the variance of the $\hat{\beta}$s.

```
#Plot of MSE and lambda
matplot(lambda, t(MSE_ridge),
        type = "l",
        xlab = expression(lambda),
        ylab = "MSE of Est Reg Coef Given Covariates",
        lty = 1, lwd = "2",
        main = expression(paste("Ridge Regression MSE[",
                                hat(beta), "|X] vs " , lambda)),
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



The plot of mean squared error vs. λ shows that the MSE initially decreases for all the $\hat{\beta}$. However, for some of the covariates, the MSE decreases and then increases. This is expected, because we want to find the optimal values of λ that balances the bias and variance of the estimates. Too large values of λ will have high MSEs for some of the estimates.

```
#lambda values that correspond to the minimum MSE

lambda_min_MSE_ridge <- c()
est_beta_min_MSE_ridge <- c()

for(i in 1:J){

  lambda_min_MSE <- which(MSE_ridge[i,] == min(MSE_ridge[i,]))
  lambda_min_MSE_ridge <- c(lambda_min_MSE_ridge,
                            lambda_min_MSE - 1)

  est_beta_min_MSE_ridge <- c(est_beta_min_MSE_ridge,
                          beta_ridge.mod[i , nlambda - lambda_min_MSE + 1])
}
```

The value of the shrinkage parameter λ minimizing the MSE for each coefficient $\hat{\beta}_j$ is given in the following output from β_1 to β_10:

```
lambda_min_MSE_ridge
```

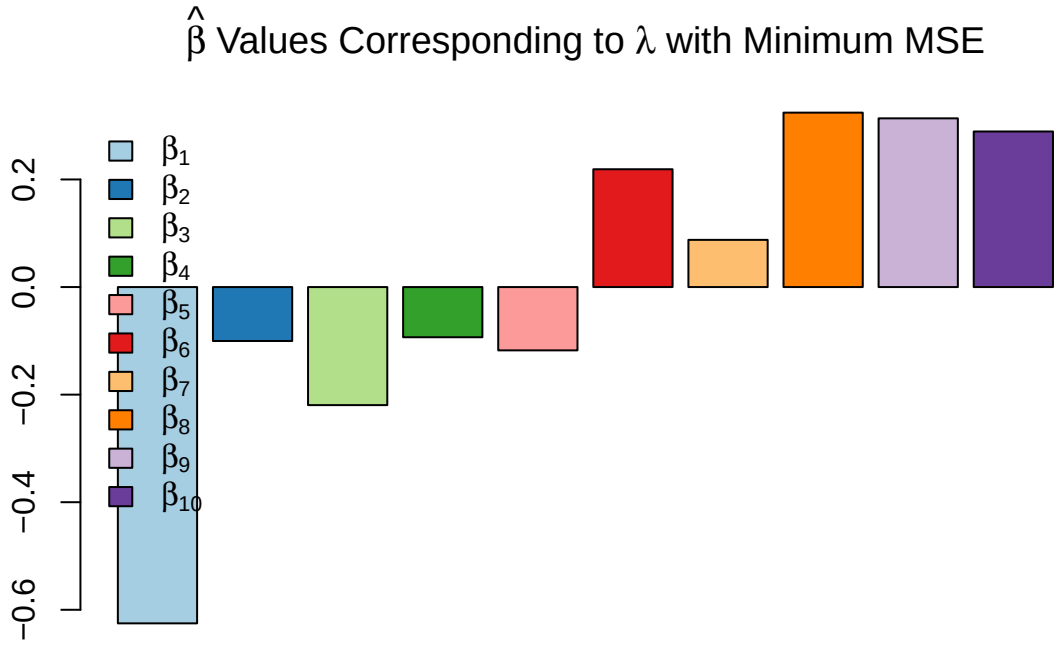
```
##  [1]  47  83 100 100 100 100 100 100  92  42
```

The β values that correspond to the λ with minimum MSE is showin in the following output from β_1 to β_10:

```
est_beta_min_MSE_ridge
```

```
##  [1] -0.62523751 -0.10031544 -0.21949277 -0.09328068 -0.11767259
##  [6]  0.21904379  0.08775485  0.32426274  0.31368976  0.28921367
```

```
barplot(est_beta_min_MSE_ridge, col = myPaletteSeq,
        main = expression(paste(hat(beta), " Values Corresponding to ",
                                lambda, " with Minimum MSE")))
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



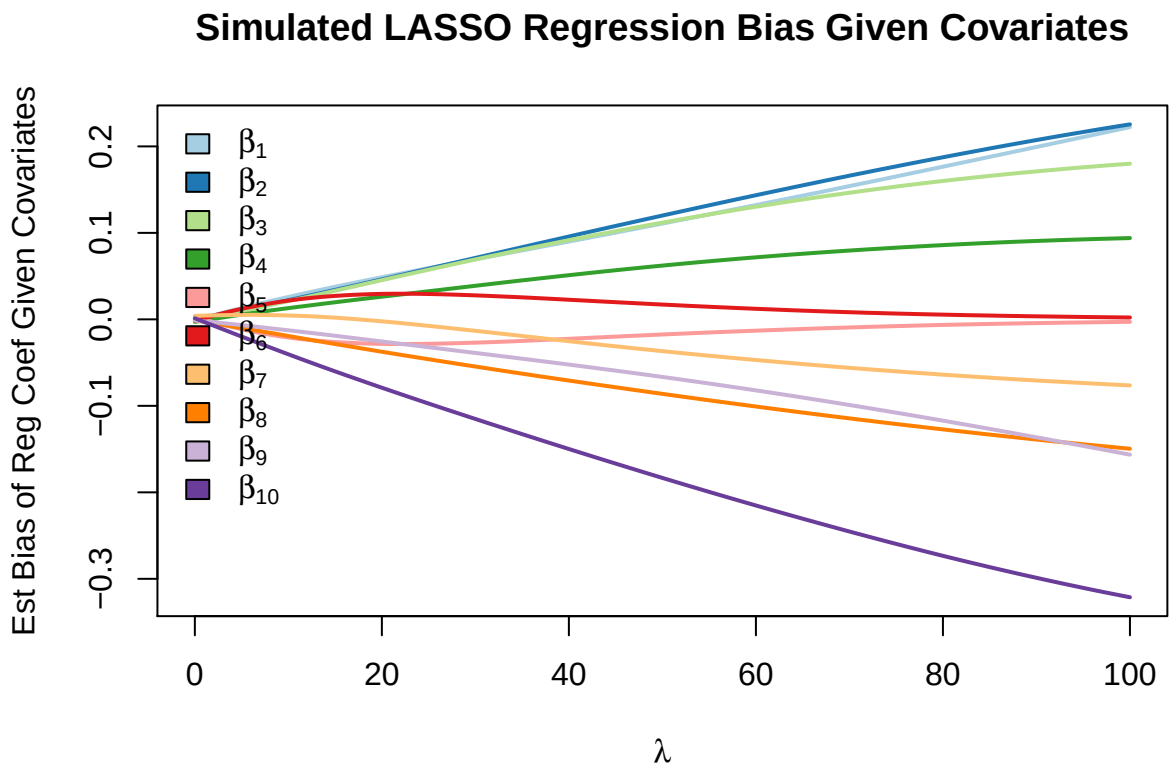
5.e. Describe how one can estimate these quantities using the simulation model of (a). In particular, provide and comment on graphical displays of the bias, variance, and MSE of the LASSO estimators based on the learning set. For each coefficient, provide the value of the shrinkage parameter λ minimizing the MSE and the corresponding estimate. Again, be specific about assumptions and which variables you are conditioning on.

The variables I am conditioning on are the covariates X. The assumptions of linearity and constant variance of the errors do not have to be true for LASSO regression. I do not have to make any distributional assumptions of the errors.

To calculate bias, variance, and MSE of the LASSO estimators, I will use simulation. Simulate many sets of

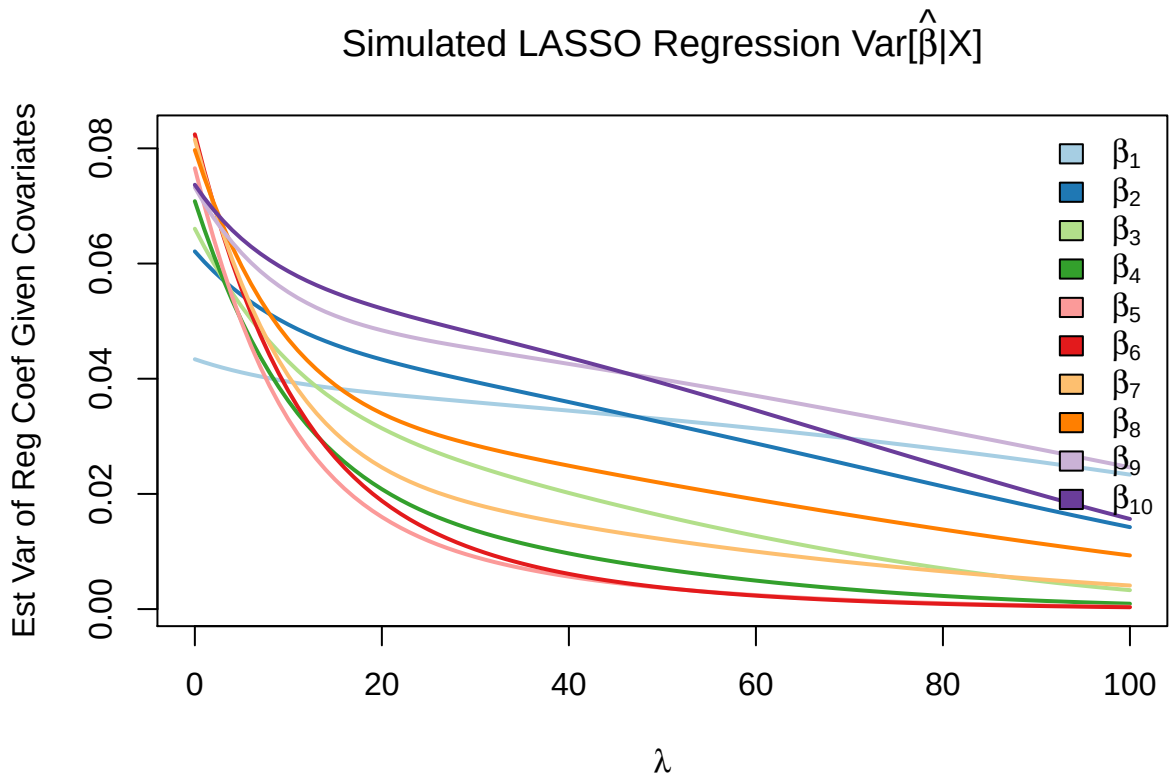
Y values by sampling from a normal distribution with the proper mean and standard deviation as specified by the model in part a. Then, estimate the β values for each of the simulation sets. Calculate the bias, variance, and MSE for each set. Then, take the average of these values to come up the estimate of bias, variance, and MSE for LASSO estimator.

```
B <- 10000

#Simulate expected value of beta_hat given X

Y_LS_star <- matrix(rnorm(B * LS_sample_size, mean = mu_given_X_LS, sd = sigma),
                    nrow = B, ncol = LS_sample_size, byrow = TRUE)

beta_star <- apply(Y_LS_star, 1, function(Y) {

  glmnet(X_LS,
         Y,
         alpha = 1,
         lambda = lambda_lasso,
         intercept = FALSE,
         standardize = FALSE)$beta

})

mean_beta_star <- Reduce("+", beta_star) / B

bias_lasso <- mean_beta_star - beta

squared_beta_star <- list()

for(i in 1:B){
  squared_beta_star[[i]] <- beta_star[[i]]^2
}

mean_squared_beta_star <- Reduce("+", squared_beta_star) / B

variance_lasso <- mean_squared_beta_star - mean_beta_star^2

MSE_lasso <- bias_lasso^2 + variance_lasso

#Plot of bias and lambda
matplot(rev_lambda, t(bias_lasso),
        type = "l",
        xlab = expression(lambda),
        ylab = "Est Bias of Reg Coef Given Covariates", lty=1, lwd = "2",
        main = "Simulated LASSO Regression Bias Given Covariates",
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



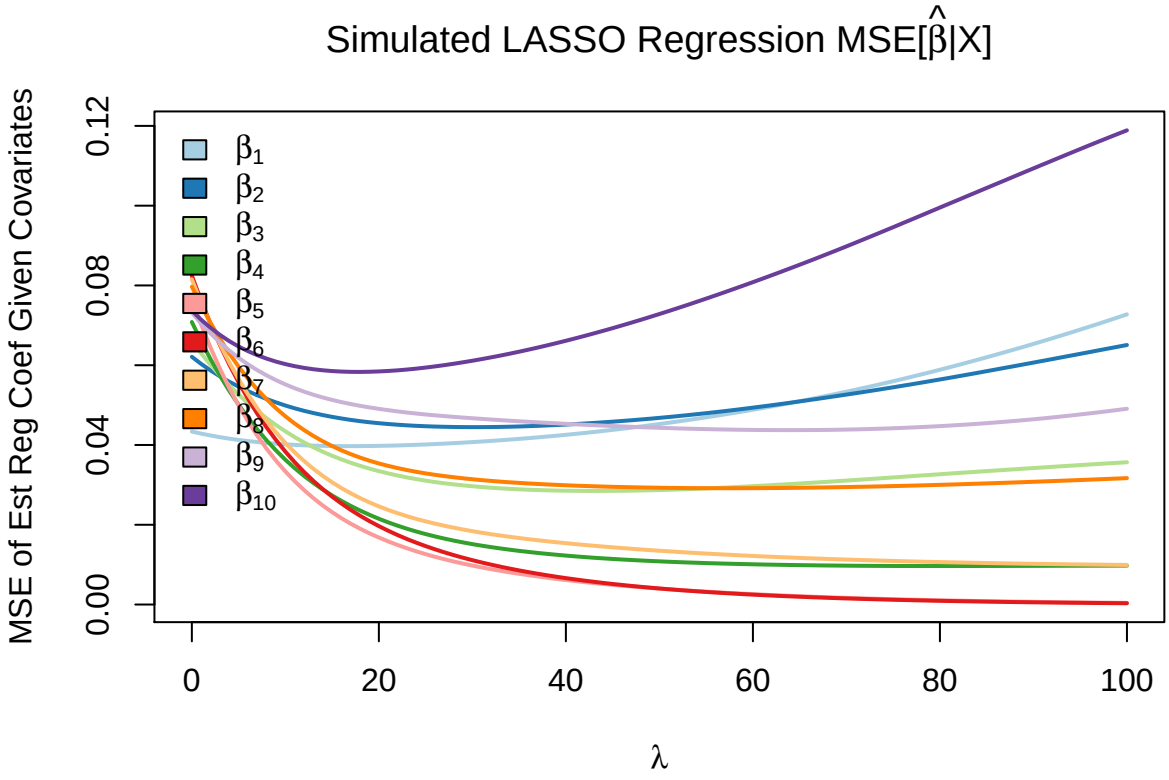
The plot of bias vs. λ shows that the bias increases as λ increases. This is expected because when we add penalties to the risk, we do a trade off between bias and variance. When $\lambda = 0$, bias $= 0$ (OLS). However, as λ increases, the estimates of β will no longer be unbiased.

```
#Plot of variance and lambda
matplot(rev_lambda, t(variance_lasso),
        type = "l",
        xlab = expression(lambda),
        ylab = "Est Var of Reg Coef Given Covariates", lty = 1, lwd = "2",
        main = expression(paste("Simulated LASSO Regression Var[",
                                hat(beta), "|X]")),
        col = myPaletteSeq)
legend("topright", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



The plot of variance vs. λ shows that the variance decreases as λ increases. This is expected because when we add penalties to the risk, we increase the bias of the estimates. However, we decrease the variance of the $\hat{\beta}$s.

```
#Plot of MSE and lambda
matplot(rev_lambda, t(MSE_lasso),
        type = "l",
        xlab = expression(lambda),
        ylab = "MSE of Est Reg Coef Given Covariates",
        lty = 1, lwd = "2",
        main = expression(paste("Simulated LASSO Regression MSE[",
                                hat(beta), "|X]")),
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```



The plot of mean squared error vs. λ shows that the MSE initially decreases for all the $\hat{\beta}$. However, for some of the covariates, the MSE decreases and then increases. This is expected, because we want to find the optimal values of λ that balances the bias and variance of the estimates. Too large values of λ will have high MSEs for some of the estimates.

```
#lambda values that correspond to the minimum MSE

lambda_min_MSE_lasso <- c()
est_beta_min_MSE_lasso <- c()

for(i in 1:J){

  lambda_min_MSE <- nlambda - which(MSE_lasso[i,] == min(MSE_lasso[i,]))
  lambda_min_MSE_lasso <- c(lambda_min_MSE_lasso,
                            lambda_min_MSE)

  est_beta_min_MSE_lasso <- c(est_beta_min_MSE_lasso,
                          beta_lasso.mod[i , nlambda - lambda_min_MSE + 1])
}
```

The value of the shrinkage parameter λ minimizing the MSE for each coefficient $\hat{\beta}_j$ is given in the following output from β_1 to β_10:

```
lambda_min_MSE_lasso
```

```
## s83 s70 s57 s21  s0  s0  s0 s42 s36 s82
##  17  30  43  79 100 100 100  58  64  18
```

The β values that correspond to the λ with minimum MSE is showin in the following output from β_1 to β_10:

```
est_beta_min_MSE_lasso
```

```
##  [1] -0.7011301  0.0000000 -0.2081141  0.0000000  0.0000000  0.0000000
##  [7]  0.0000000  0.3172261  0.2916459  0.2674074
```

```
barplot(est_beta_min_MSE_lasso, col = myPaletteSeq,
        main = expression(paste(hat(beta), " Values Corresponding to ",
                                lambda, " with Minimum MSE")))
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       fill=myPaletteSeq,
       bty = "n")
```