Assignment 2

Daniel Lee

Question 1. Ridge and LASSO regression:

1. To obtain $\hat{\beta}_n^{OLS}$,

Start with $\hat{\beta}_n^{OLS} = (X_n^T X)^{-1} X_n^T Y_n$

$= (I_J)^{-1} X_n^T Y_n$

$= X_n^T Y_n$

$\hat{\beta}_n^{ridge} = (X_n^T X_n + \lambda I_J)^{-1} X_n^T Y_n$

$= (I_J + \lambda I_J)^{-1} X_n^T Y_n$

$= (1+\lambda)^{-1} I_J X_n^T Y_n$

$= (1+\lambda)^{-1} (X_n^T X_n)^T X_n^T Y_n$

$= (1+\lambda)^{-1} \hat{\beta}_n^{OLS}$

$E[\hat{\beta}_n^{ridge} | X] = (1+\lambda)^{-1} E[\hat{\beta}_n^{OLS} | X]$

$= (1+\lambda)^{-1} \beta$

So $E[\hat{\beta}_n^{ridge} | X] - \beta$

$= (1+\lambda)^{-1} \beta - \beta$

$= ((1-\lambda)^{-1} - 1) \beta$

$$\mathrm{Cov}\left[\hat{\beta}_n^{\text{ridge}} \mid X\right] = \mathrm{Cov}\left(\left(\frac{1}{\lambda+1}\right)\hat{\beta}_n^{\text{OLS}}\right)$$

$$= \mathrm{Cov}\left[\left(\frac{1}{\lambda+1}\right)X_n^T Y_n \mid X_n\right]$$

$$= \left[\frac{1}{\lambda+1}X_n^T\right]\mathrm{Cov}(Y_n \mid X_n)\left[\frac{1}{\lambda+1}X_n^T\right]^T$$

$$= \sigma^2\left(\frac{1}{\lambda+1}\right)^2\left[X_n^T\right]\left(X_n^T\right)^T$$

$$= \sigma^2\left(\frac{1}{\lambda+1}\right)^2 X_n^T X_n$$

$$= \left(\frac{\sigma}{\lambda+1}\right)^2 I_J$$

$$df(\lambda) = df(Y_n, \hat{Y}_n) = \frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}\left[Y_i, \hat{Y}_i \mid X_n\right]$$

$$= \mathrm{tr}\, X_n\left(X_n^T X_n + \lambda I_J\right)^{-1} X_n^T$$

$$= \mathrm{tr}\, X_n\left(I_J + \lambda I_J\right)^{-1} X_n^T$$

$$= \mathrm{tr}\left(I_J + \lambda I_J\right)^{-1} I_J$$

$$= \sum_{j=1}^{J}\left(\frac{1}{1+\lambda}\right) = J\left(\frac{1}{1+\lambda}\right)$$

$$\hat{\beta}_n^{LASSO} \equiv \underset{\beta \in \mathbb{R}^J}{\operatorname{argmin}} \| Y_n - X_n \beta \|_2^2 + \lambda_1 \|\beta\|_1$$

$$= \underset{\beta \in \mathbb{R}^J}{\operatorname{argmin}} (Y_n - X_n \beta)^T (Y_n - X_n \beta) + \lambda \|\beta\|_1$$

$$O_{LASSO} = X_n^T Y_n + \lambda \sum_{j=1}^{J} |\beta_j|$$

$$\frac{d\, O_{LASSO}}{d\beta} = -2 X_n^T Y_n + 2\beta + \lambda \sin(\beta)$$

where $\operatorname{sign}(\beta) = \begin{bmatrix} \sin \beta_1 \\ \vdots \\ \sin \beta_J \end{bmatrix}$

Set to 0:

$$-2 X_n^T Y_n + 2\beta + \lambda \operatorname{sig}(\beta) = 0$$

$$2\beta + \lambda \operatorname{sign}(\beta) = 2 X_n^T Y_n$$

Signs must match on both sides of equation.

So $2\beta + \lambda \operatorname{sign}(X_n^T Y_n) = 2 X_n^T Y_n$

$$2\beta = 2 X_n^T Y_n - \lambda \operatorname{sign}(X_n^T Y_n)$$

$$\beta = X_n^T Y_n - \frac{\lambda}{2} \operatorname{sign}(X_n^T Y_n)$$

Question 2. Ridge and LASSO Regression

a. Ridge Regression

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

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

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

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

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

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

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

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

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

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

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

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

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.

Assignment 2

Daniel Lee

November 5, 2016

Question 3. Elastic Net: Simulation Study

a. Simulation Model

```
require(MASS)
require(graphics)
require(glmnet)
require(RColorBrewer)
```

```
myPaletteSeq <- brewer.pal(10, "Paired")

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)
```

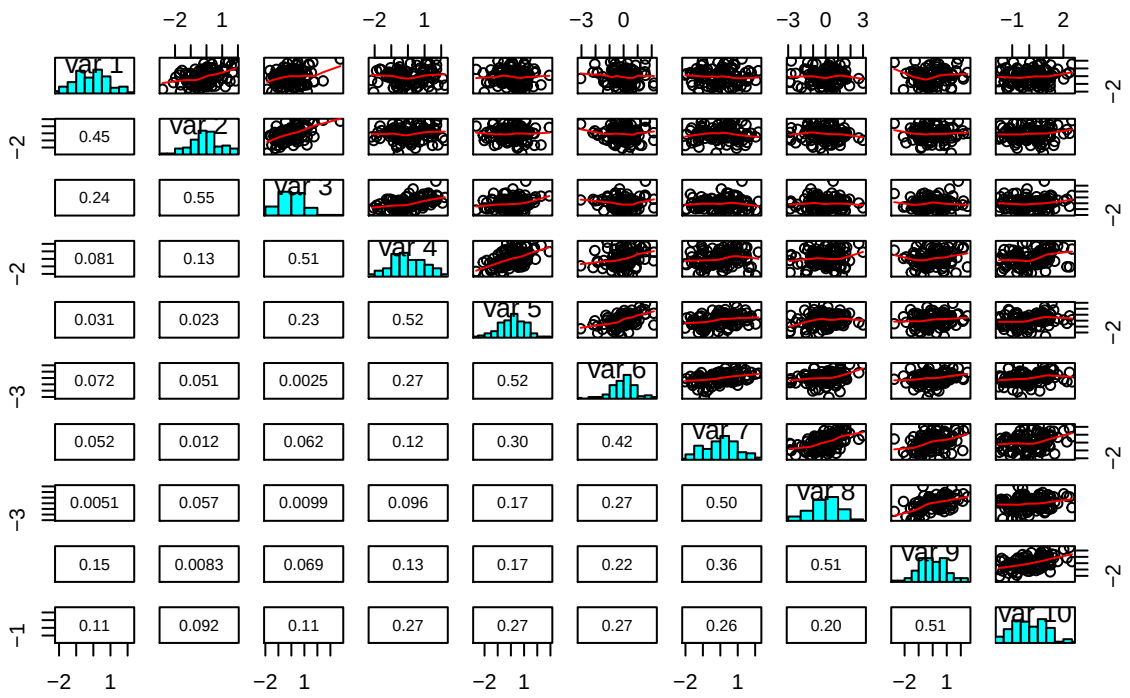
I will use the reverse of 0, 1, ..., 100 for the λ values for many of the computations and plots because the `glmnet` function's default setting makes it so that the λ's orders are decreasing when I input increasing orders of λ.

```
#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)

#Test Set
```

```r
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)
```

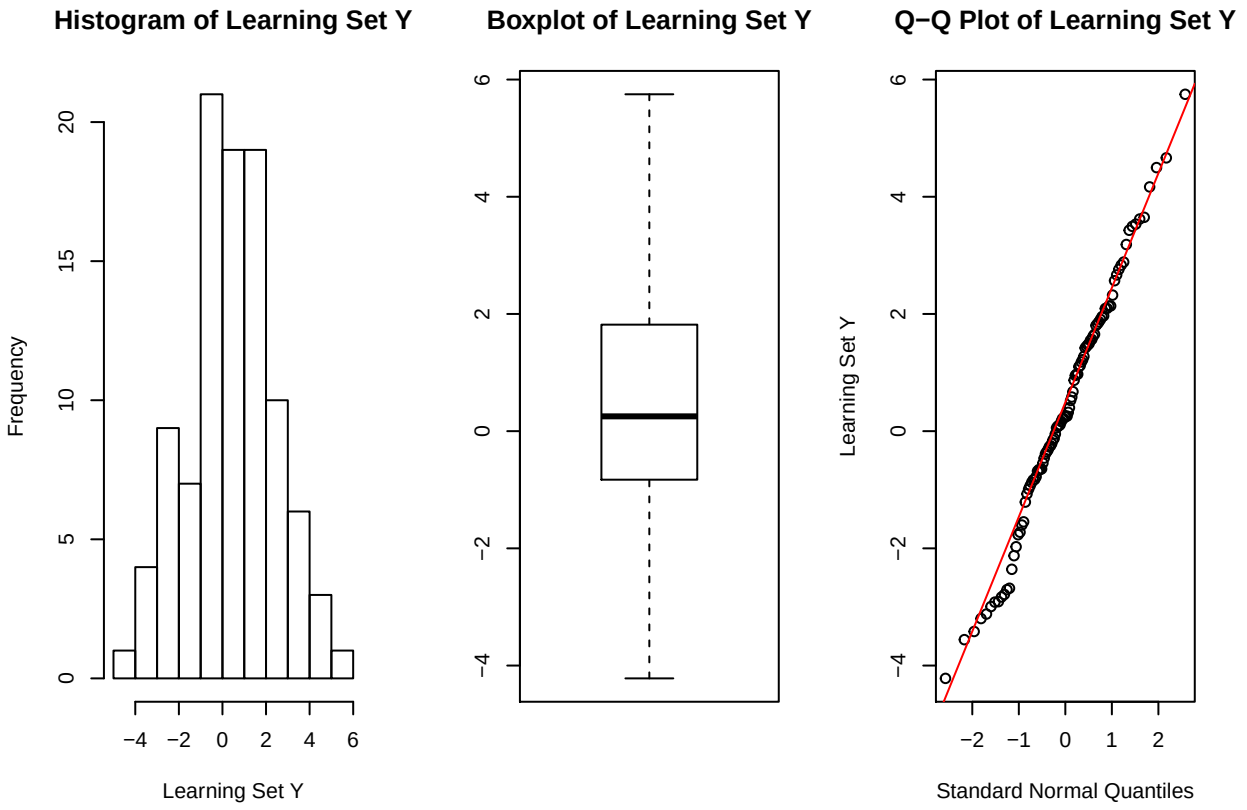
Numerical and Graphical Summaries of the Learning Set

```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.

```
#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")
```

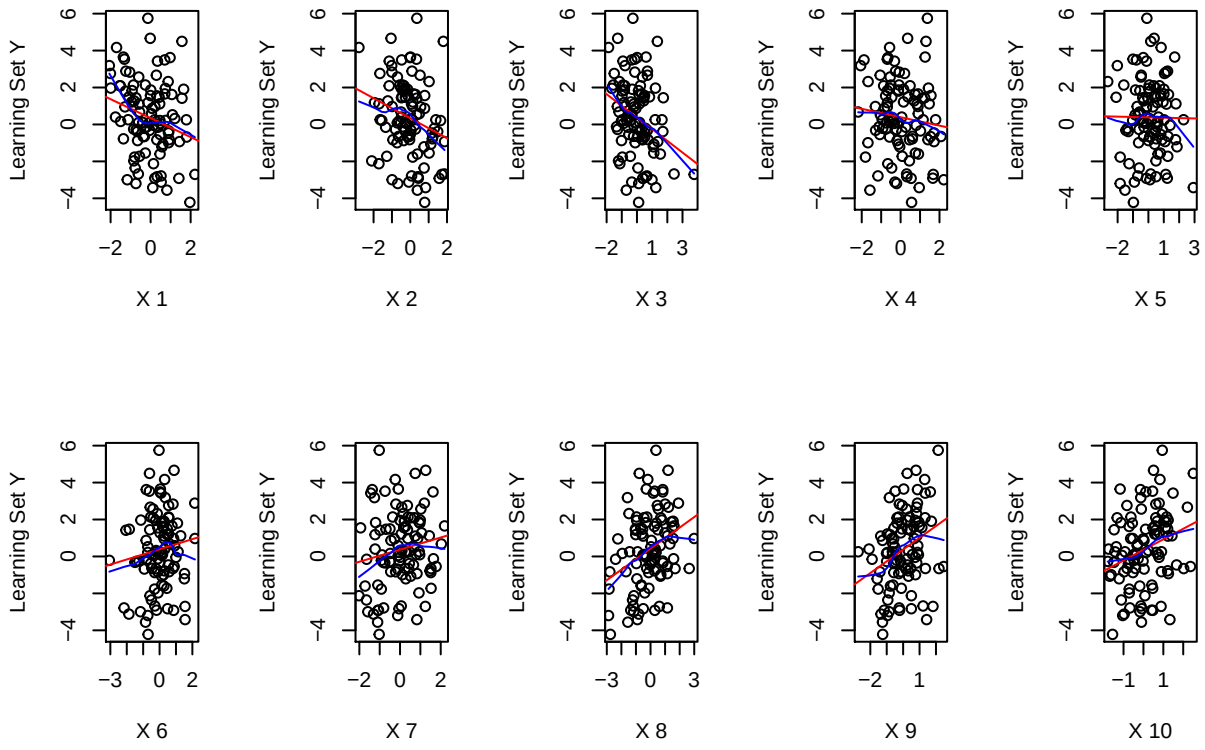


The histogram, qq-plot, and boxplot of the learning set of Y suggest that the Y's have a normal distribution as well. It seems as if the approximate mean of Y is zero.

```
summary(Y_LS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4.2190 -0.8263  0.2524  0.3820  1.8100  5.7490
```

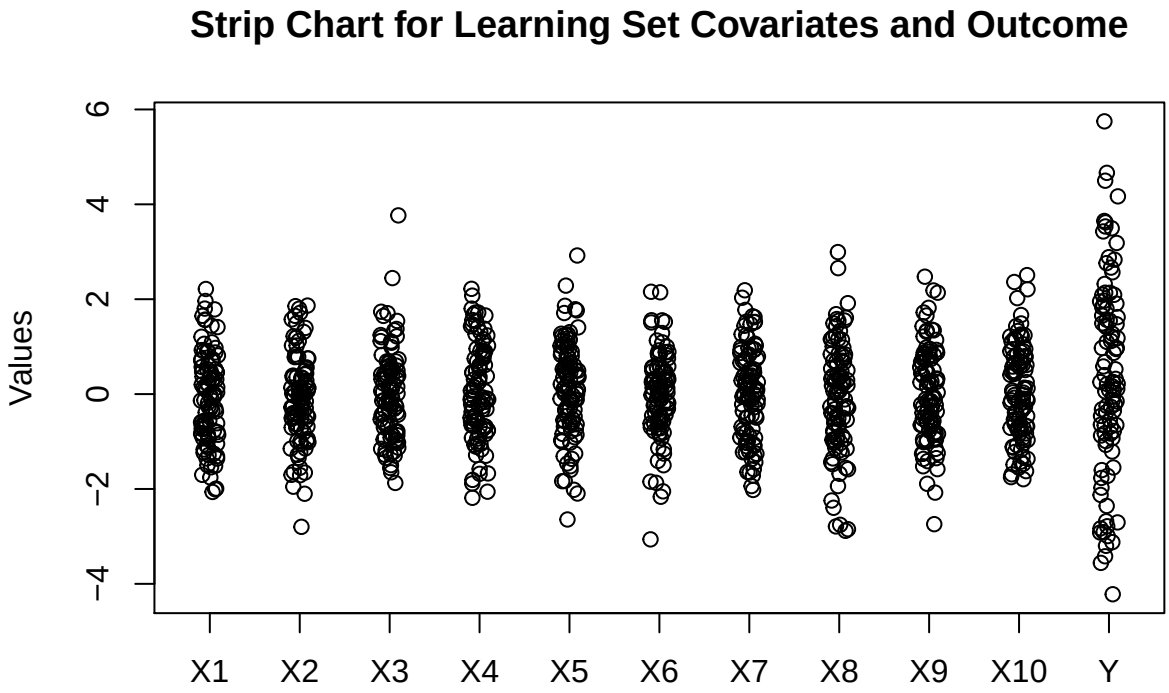
```
par(mfrow = c(2, 5))
for(i in 1:10){
  plot(X_LS[ , i], Y_LS, xlab = paste("X", i), ylab = "Learning Set Y")
  abline(lm(Y_LS ~ X_LS[, i]), col="red") # regression line (y~x)
  lines(lowess(X_LS[, i],Y_LS), col="blue") # lowess line (x,y)
}
```




```
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.

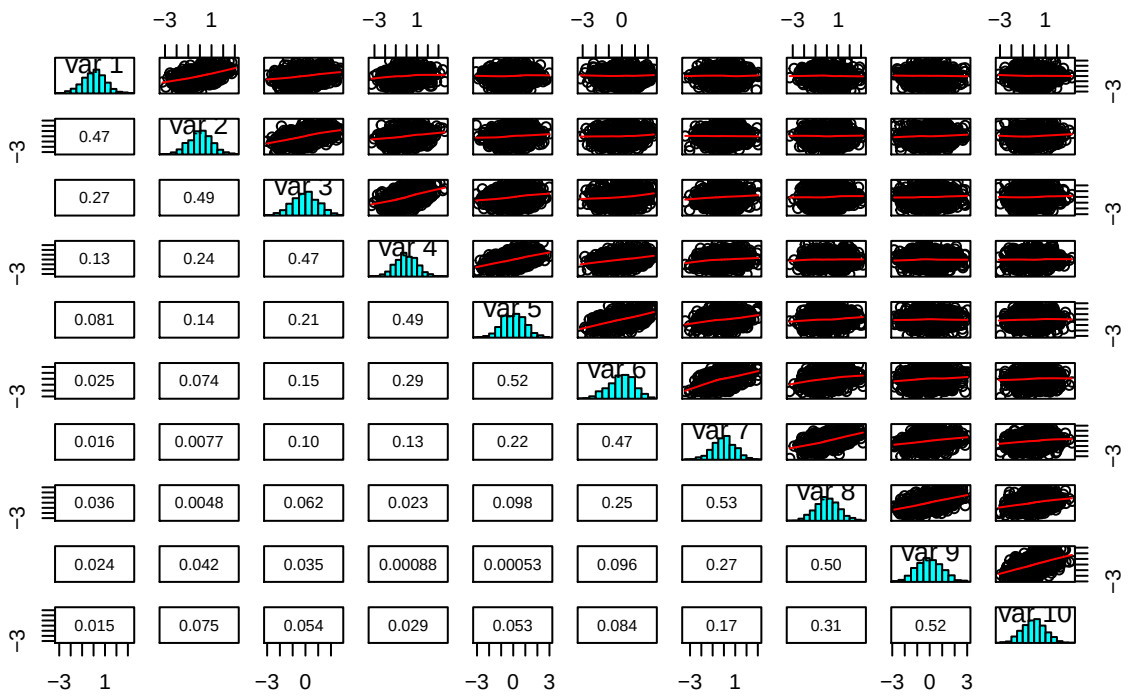
```
#stripchart
stripchart_list <- as.list(rep(NA, J + 1))

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.

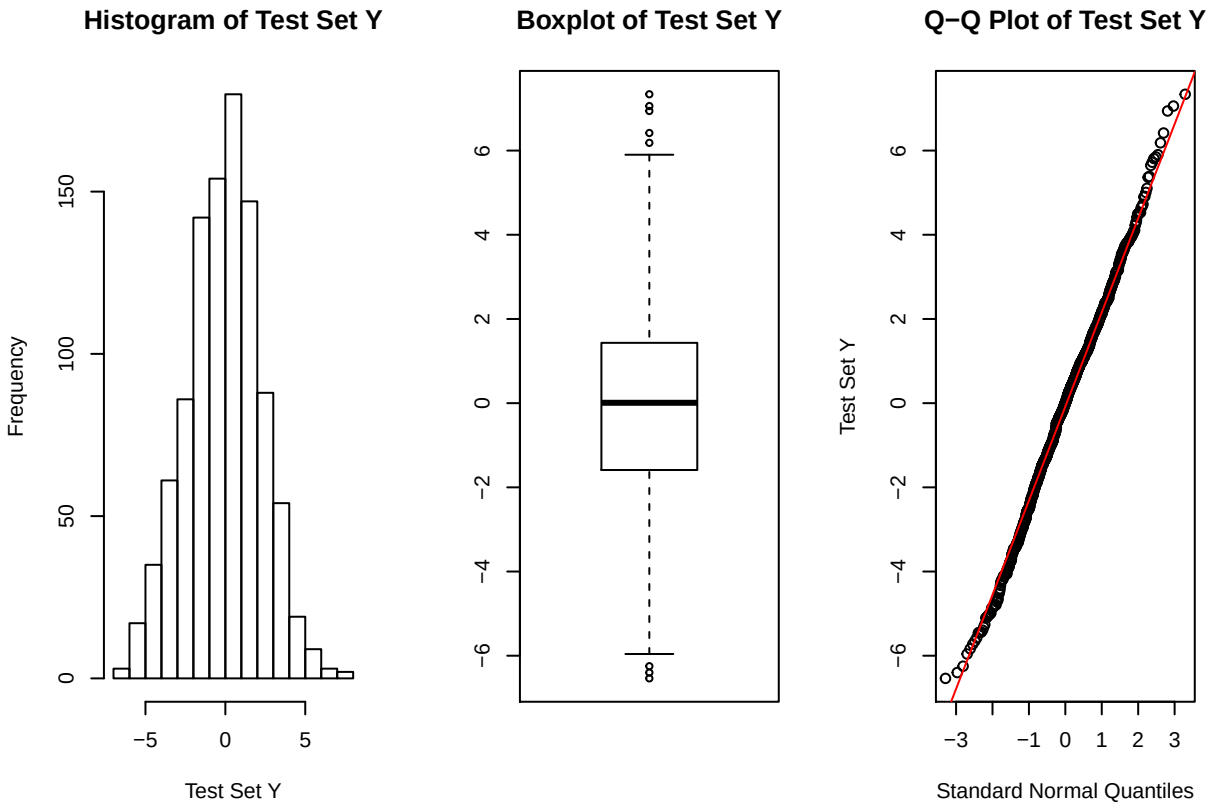
Numerical and Graphical Summaries of the Test Set

```
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.

```
#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")
```

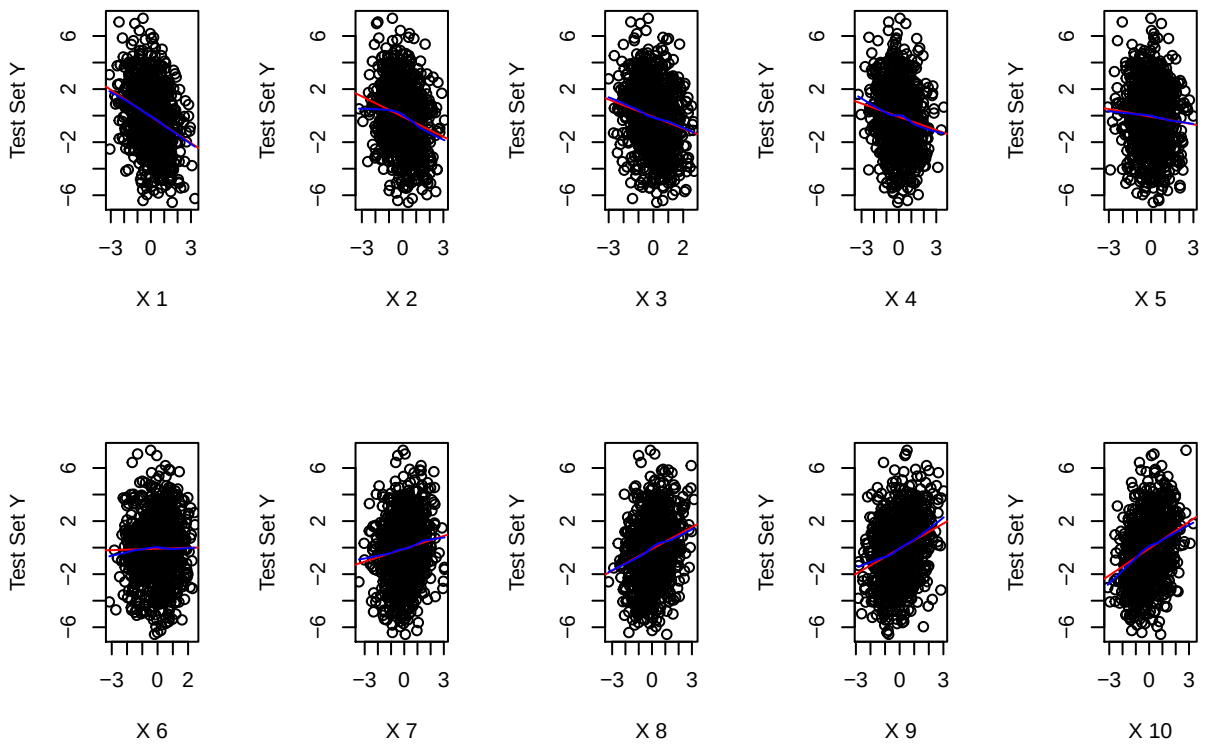


The histogram, qq-plot, and boxplot of the test set of Y suggest that the Y's have a normal distribution as well. It seems as if the approximate mean of Y is zero.

```
summary(Y_TS)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -6.538000 -1.588000  0.008789 -0.087310  1.428000  7.337000
```

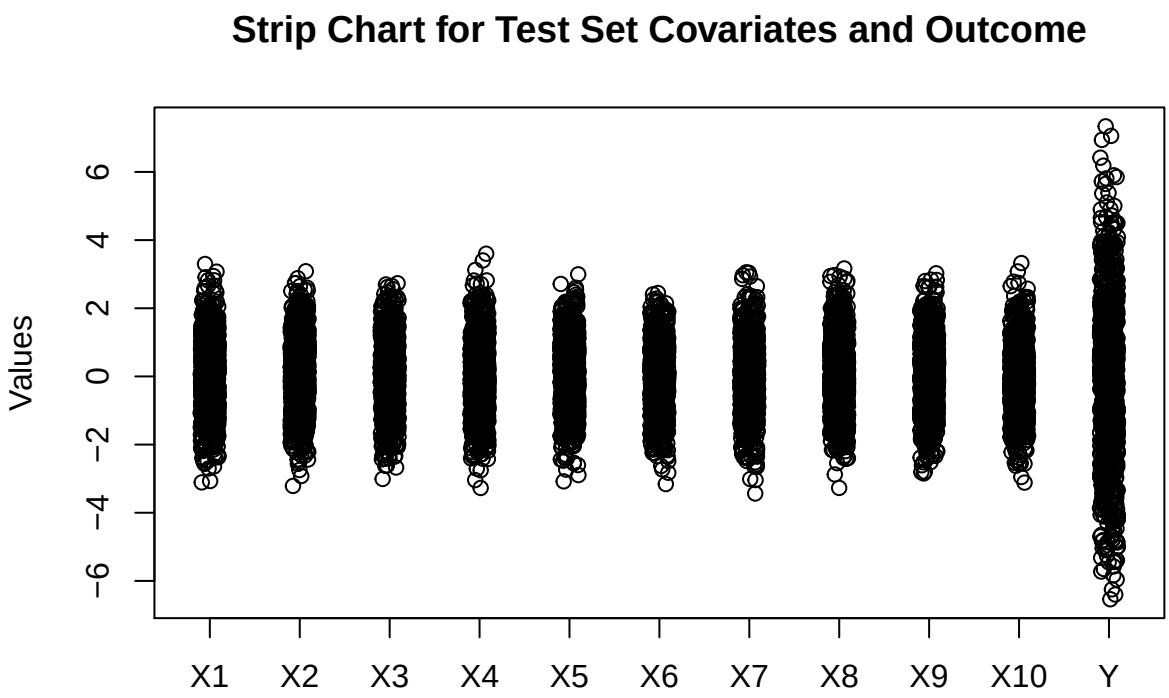
```
par(mfrow = c(2, 5))
for(i in 1:10){
  plot(X_TS[ , i], Y_TS, xlab = paste("X", i), ylab = "Test Set Y")
  abline(lm(Y_TS ~ X_TS[, i]), col="red") # regression line (y~x)
  lines(lowess(X_TS[, i],Y_TS), col="blue") # lowess line (x,y)
}
```




```
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.

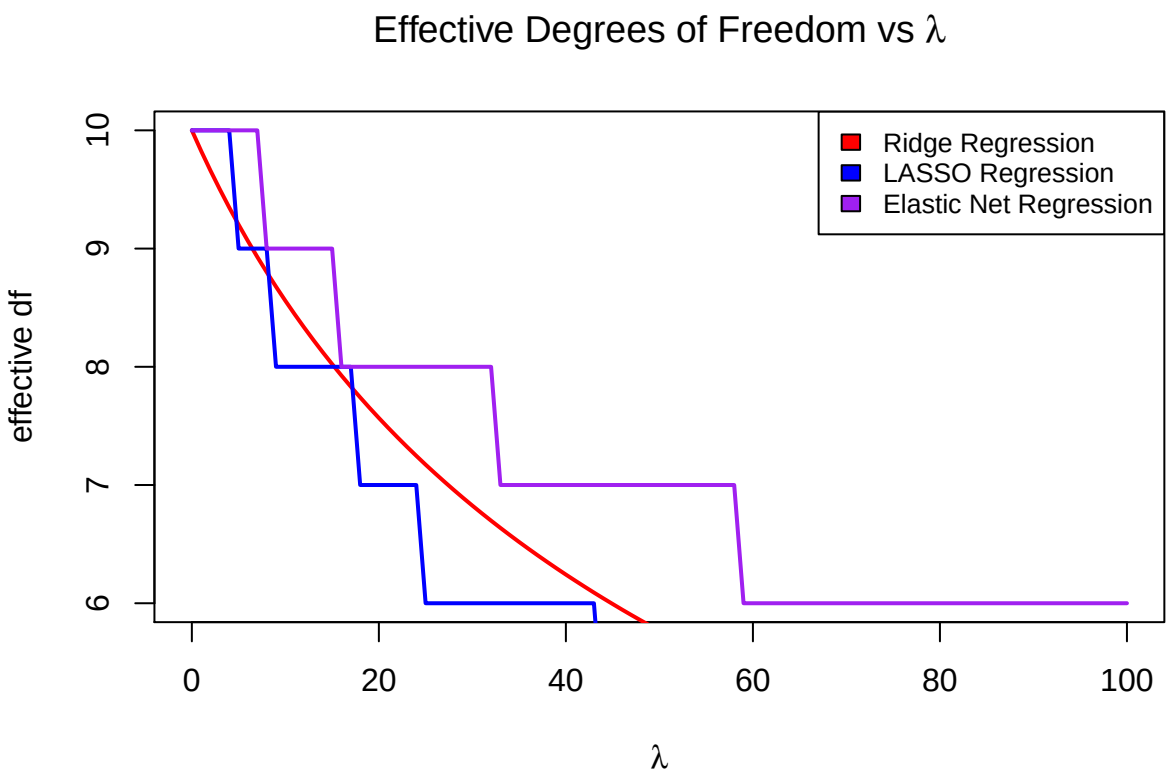
b. Elastic Net Regression on Learning Set

```
# 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
                    )
```

```r
# 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
```

The ridge, LASSO, and elastic net regression models were built using the `glmnet` package. For ridge regression, the alpha was set to zero to match the ridge regression model. The lambda values were divided by 100 to scale the model properly. For LASSO regression, alpha was set to one and the lambda values were divided by 200. For elastic net regression, alpha was set to 1/3 and the lambda values were multiplied by 3/400. Based on the numerical and graphical summaries, I decided that it was not necessary to fit an intercept or to standardize the data.

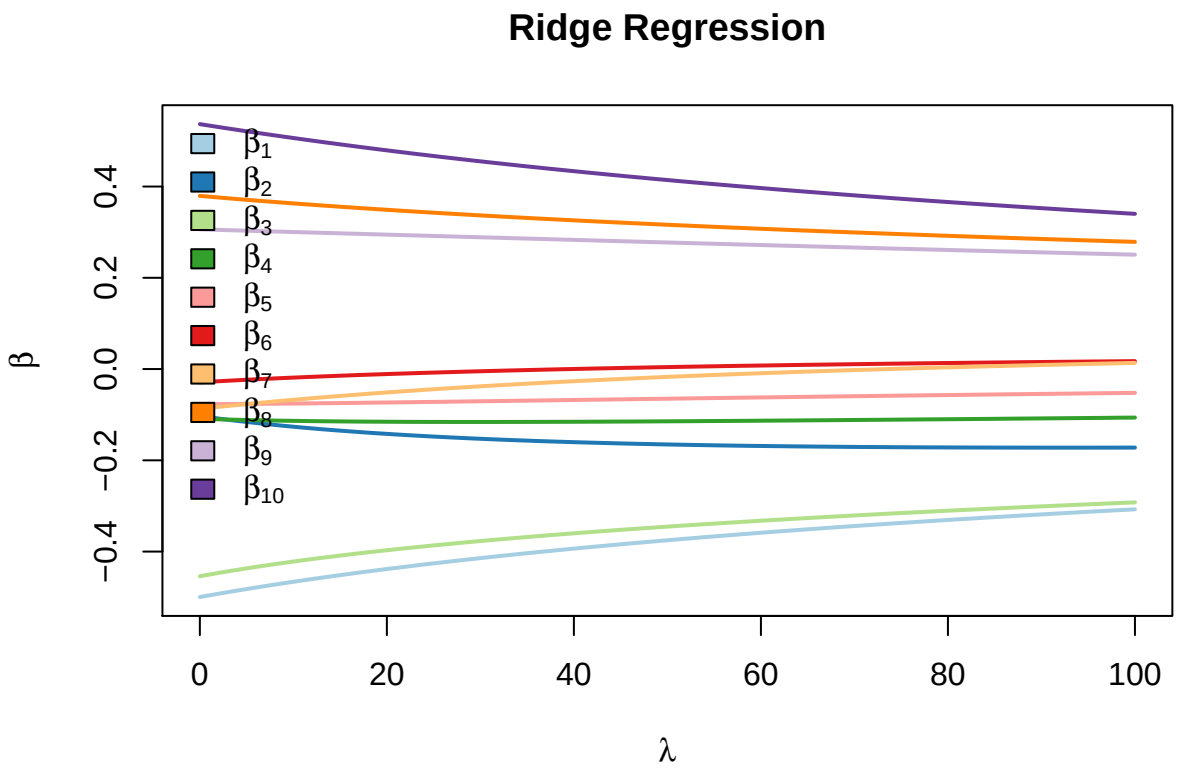
```r
# ridge regression degrees of freedom
L_j <- svd(X_LS)$d

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

eff_df <- effective_df(lambda)

#effective df plot
plot(lambda, elnet.mod$df, type="n",
     xlab = expression(lambda),
     ylab = "effective df",
     main = expression(paste("Effective Degrees of Freedom vs ",
                     lambda)))
lines(lambda, eff_df, col = "red", lwd = 2) #ridge df
lines(rev_lambda, lasso.mod$df, col = "blue", lwd = 2) #lasso df
lines(rev_lambda, 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"))
```



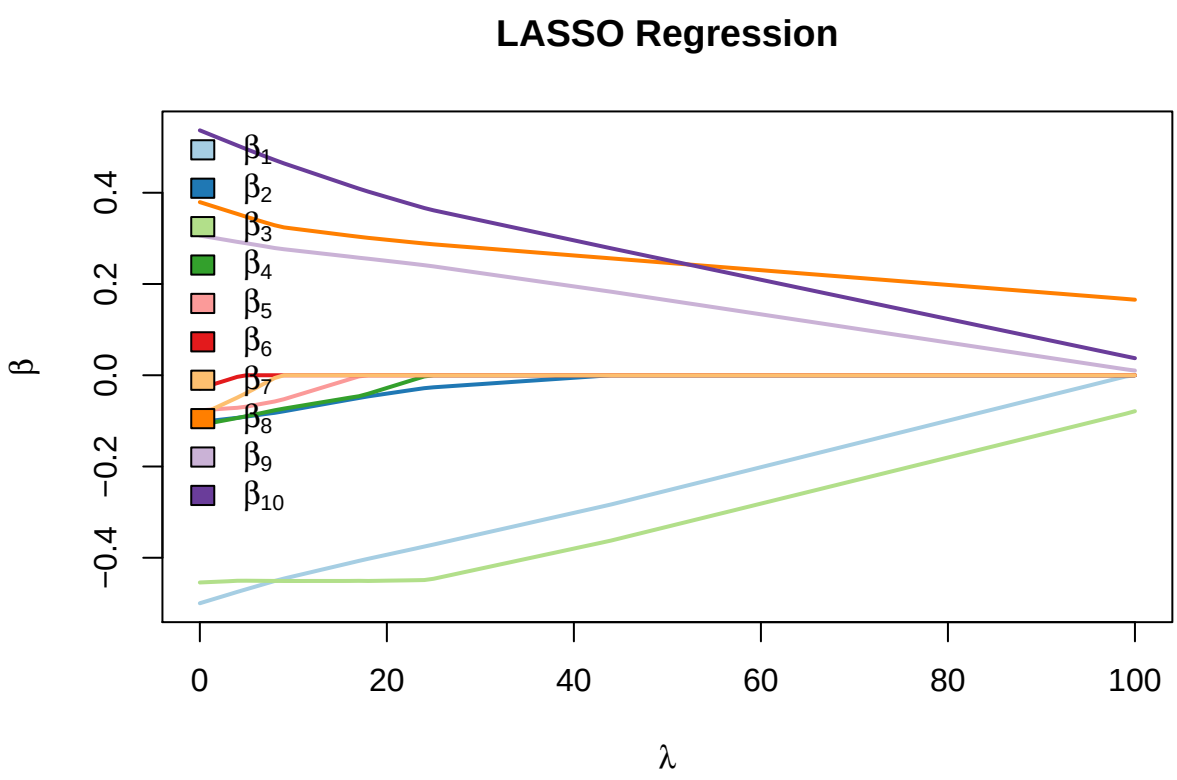
The degrees of freedom decreases for all three of the models as λ increases. The degrees of freedom for LASSO and elastic net regression models correspond to the number of non-zero coefficients. The effective degrees of freedom for ridge regression was computed separately.

```
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,
```

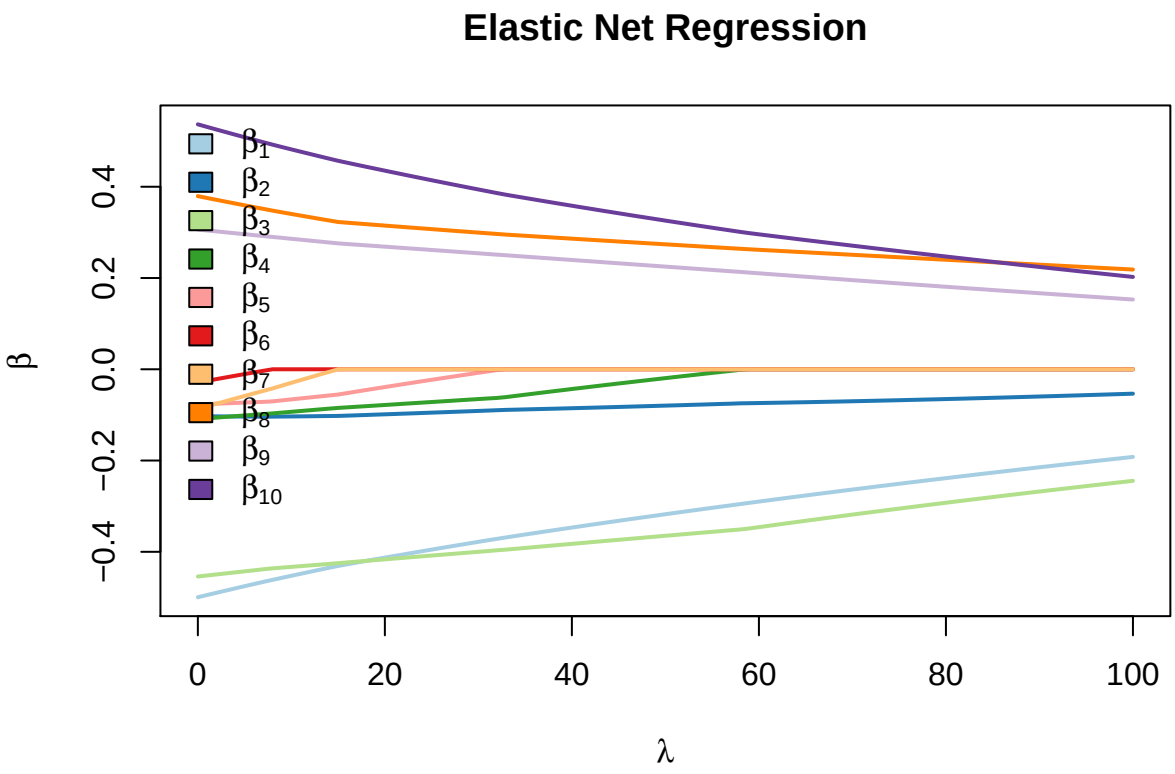
```
       bty = "n")
```




```
#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")
```



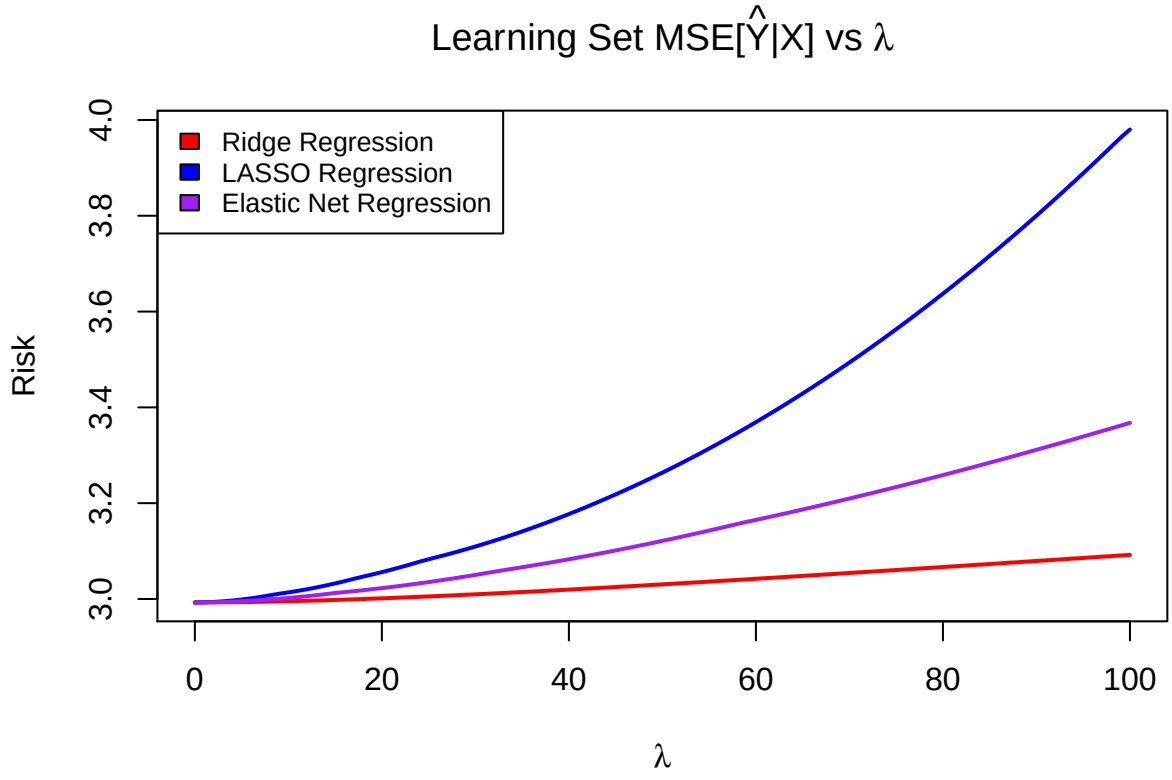

```
#elastic net regression
matplot(rev_lambda, t(beta_elnet.mod),
        type = "l",
        xlab = expression(lambda),
        ylab = expression(beta),
        lty = 1, lwd = "2",
        main = "Elastic Net Regression",
        col = myPaletteSeq)
legend("topleft", coefficient_labels,
       col=myPaletteSeq,
       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. By doing this, the model is minimizing the 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          V6
## -0.49953205 -0.10283707 -0.45406006 -0.10905028 -0.07696236 -0.02899815
##          V7          V8          V9         V10
## -0.08720819  0.37951124  0.30611266  0.53675543
```

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

```
##          V1          V2          V3          V4          V5          V6
## -0.49966104 -0.10268615 -0.45414597 -0.10899188 -0.07697404 -0.02898509
##          V7          V8          V9         V10
## -0.08725118  0.37945464  0.30618058  0.53673900
```

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

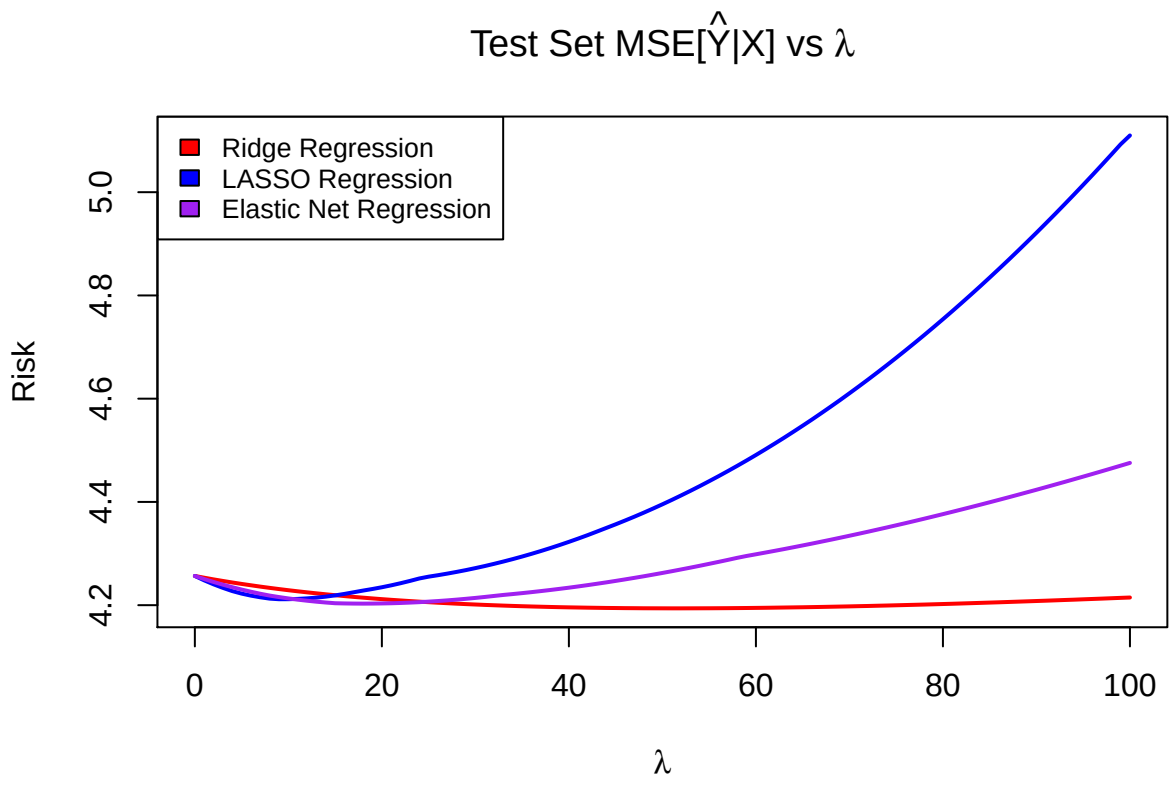
```
##          V1          V2          V3          V4          V5          V6
## -0.49956378 -0.10275421 -0.45411889 -0.10901932 -0.07695934 -0.02896748
##          V7          V8          V9         V10
## -0.08717866  0.37942990  0.30615066  0.53673317
```

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. 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.

c. Performance Assessment on Test Set

```
#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))

#Plot of MSE and Lambda Test Set
```

```r
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"))
```




```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
```

```
## s48
##  52
```

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

```
## s91
##   9
```

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

```
## s82
##  18
```

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

```
##           V1           V2           V3           V4           V5
## -0.371716183 -0.166022228 -0.342580913 -0.114267235 -0.064376271
##           V6           V7           V8           V9          V10
##  0.005056233 -0.015441451  0.314404755  0.276245960  0.410558030
```

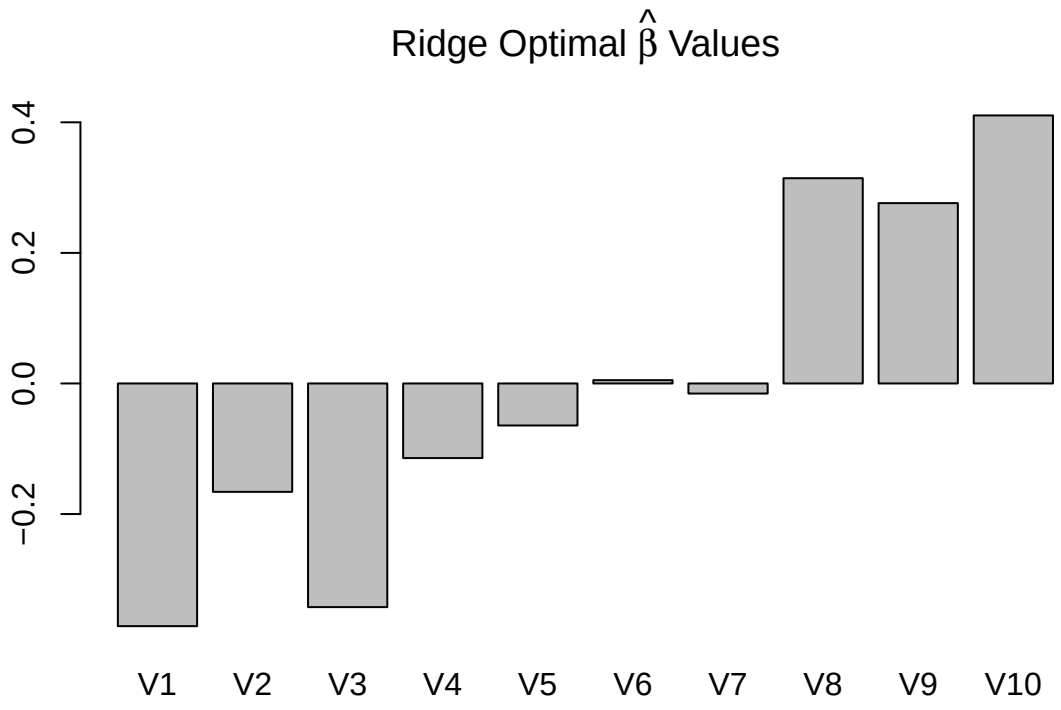
```r
beta_lasso.mod[ , nlambda - min_lambda_lasso_MSE_TS]
```

```
##          V1          V2          V3          V4          V5          V6
## -0.44554402 -0.07889686 -0.45089668 -0.07306061 -0.05248055  0.00000000
##          V7          V8          V9         V10
##  0.00000000  0.32399665  0.27615570  0.46431180
```

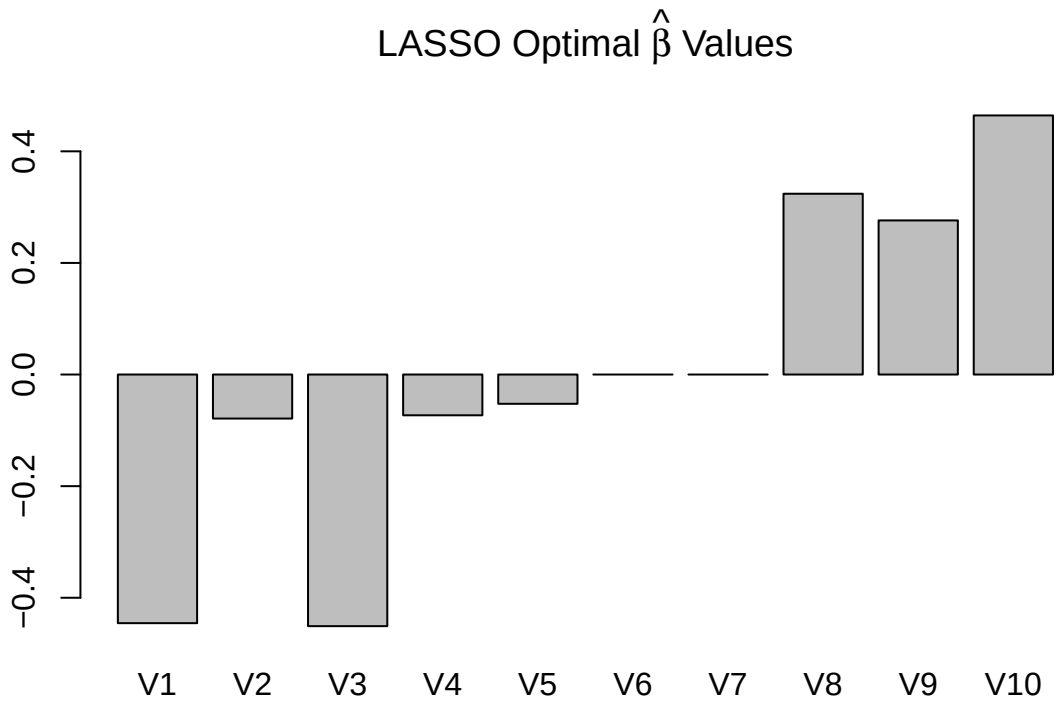
```r
beta_elnet.mod[ , nlambda - min_lambda_elnet_MSE_TS]
```

```
##          V1          V2          V3          V4          V5          V6
## -0.41993919 -0.10014027 -0.41988671 -0.08081050 -0.04546509  0.00000000
##          V7          V8          V9         V10
##  0.00000000  0.31799552  0.27158090  0.44379370
```

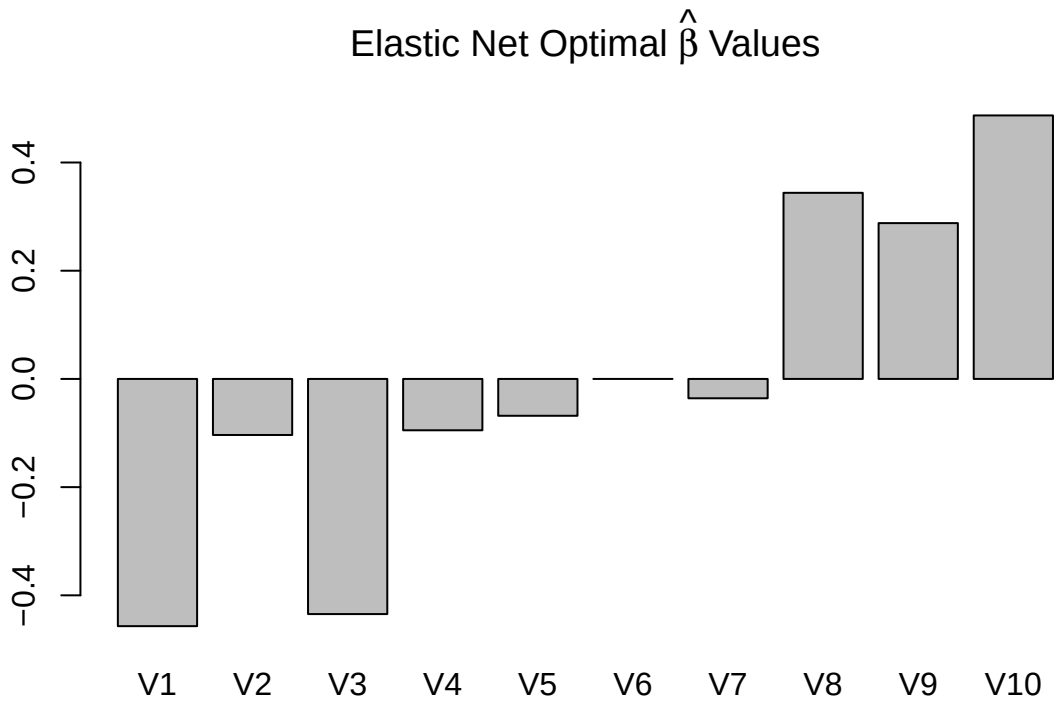
```r
barplot(beta_ridge.mod[ , nlambda - min_lambda_ridge_MSE_TS],
        main = expression(paste("Ridge Optimal ", hat(beta), " Values")))
```




```
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")))
```



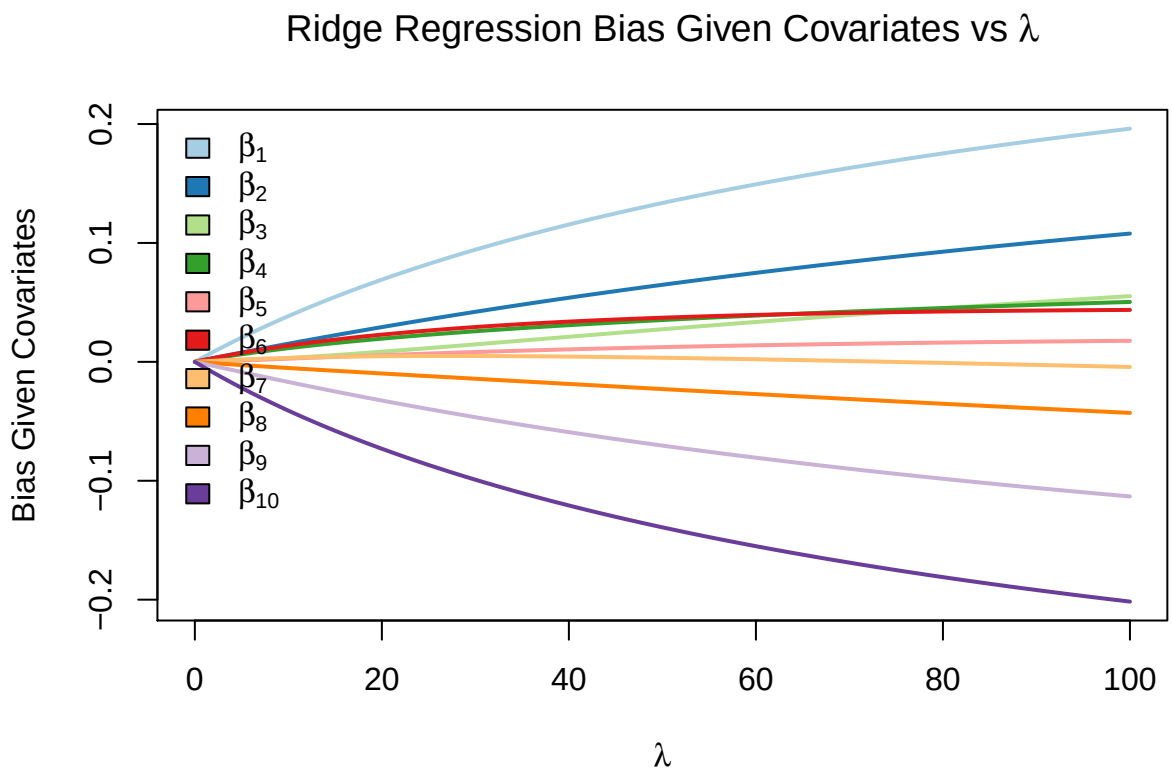
MSE[$\hat{\beta}$|X] for LASSO and elastic net regression decreases and then increases as λ increases. The MSE[$\hat{\beta}$|X] for ridge regression decreases as λ increases. From the plot and the outputs, I conclude that the optimal values for λ increases from LASSO to elastic net to ridge regression as expected.

d. Ridge Regression: Bias, Variance, and Mean Squared Error of Estimated Regression Coefficients

```
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)

#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
}
```

```r
#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
```

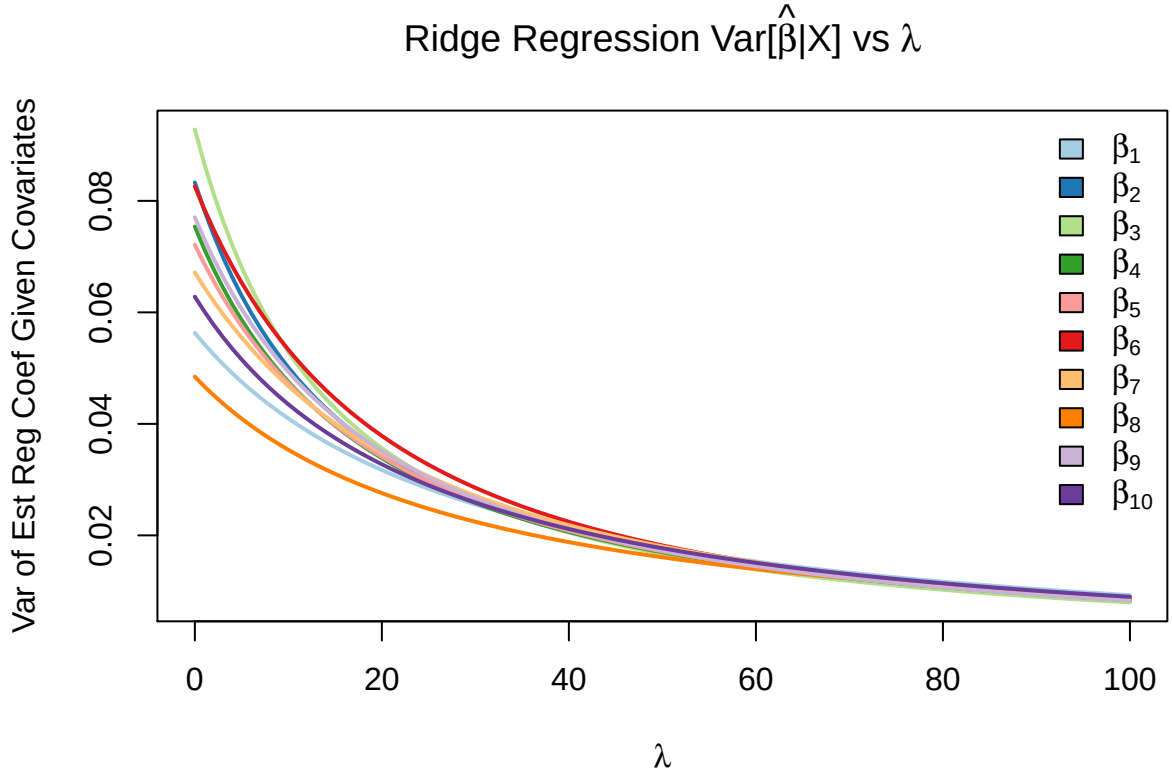
```r
#Plot of bias and lambda
matplot(lambda, t(bias_ridge),
        type = "l",
        xlab = expression(lambda),
        ylab = "Bias Given Covariates", lty=1, lwd = "2",
        main = expression(paste("Ridge Regression Bias Given Covariates vs ",
                                lambda)),
        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.

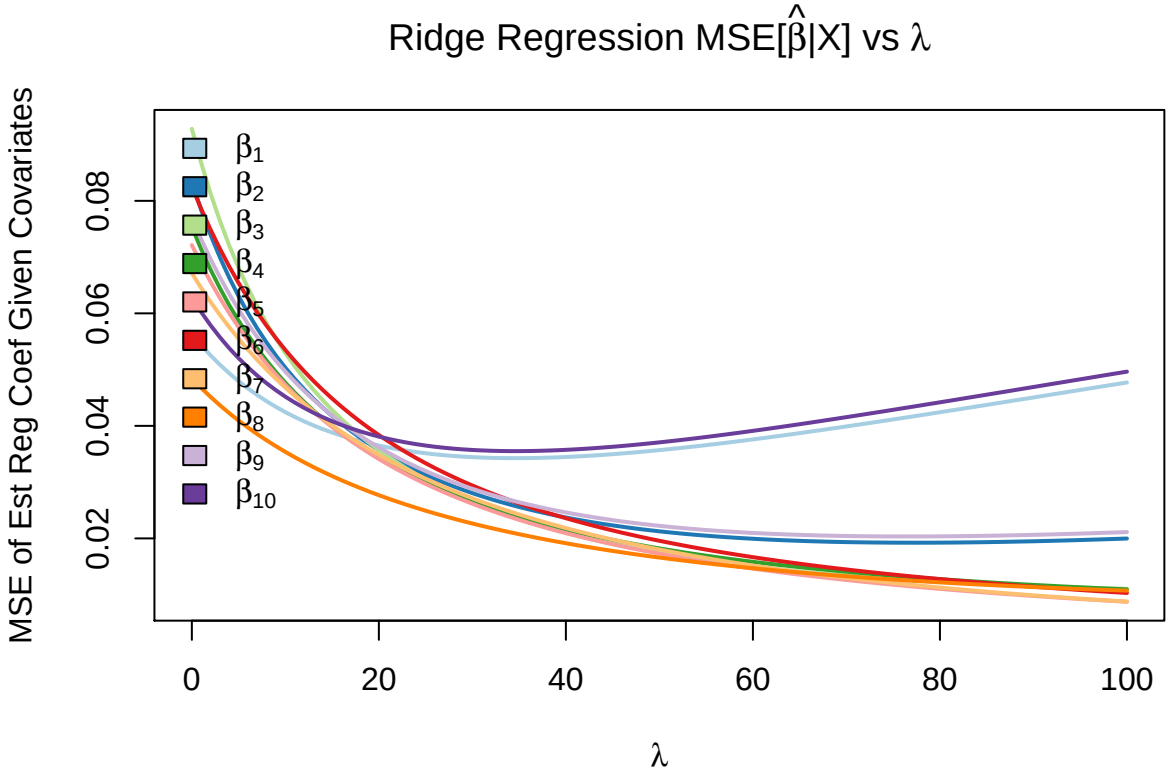
```
#Plot of variance and lambda
matplot(lambda, t(variance_ridge),
        type = "l",
        xlab = expression(lambda),
        ylab = "Var of Est Reg Coef Given Covariates",
        lty = 1, lwd = "2",
        main = expression(paste("Ridge Regression Var[",
                                hat(beta), "|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])
}
```

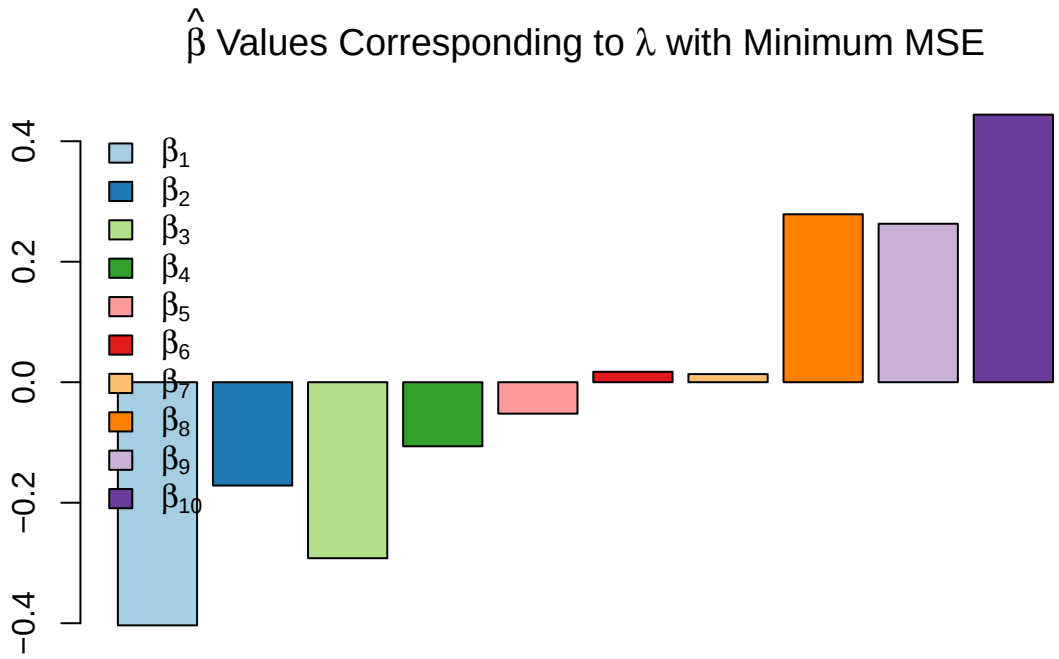
```
lambda_min_MSE_ridge
```

```
##  [1]  35  77 100 100 100 100 100 100  76  35
```

```
est_beta_min_MSE_ridge
```

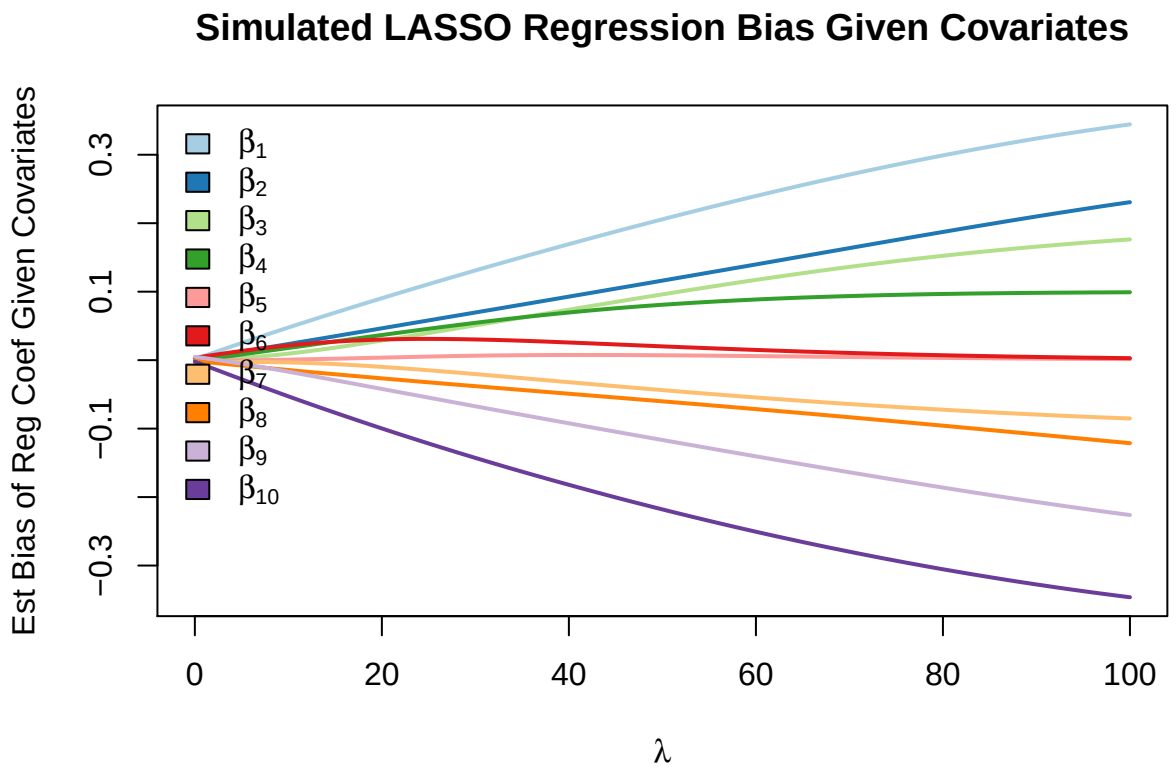
```
##  [1] -0.40355656 -0.17150431 -0.29213163 -0.10636002 -0.05216621
##  [6]  0.01735776  0.01344333  0.27873449  0.26305334  0.44411742
```

```
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")
```



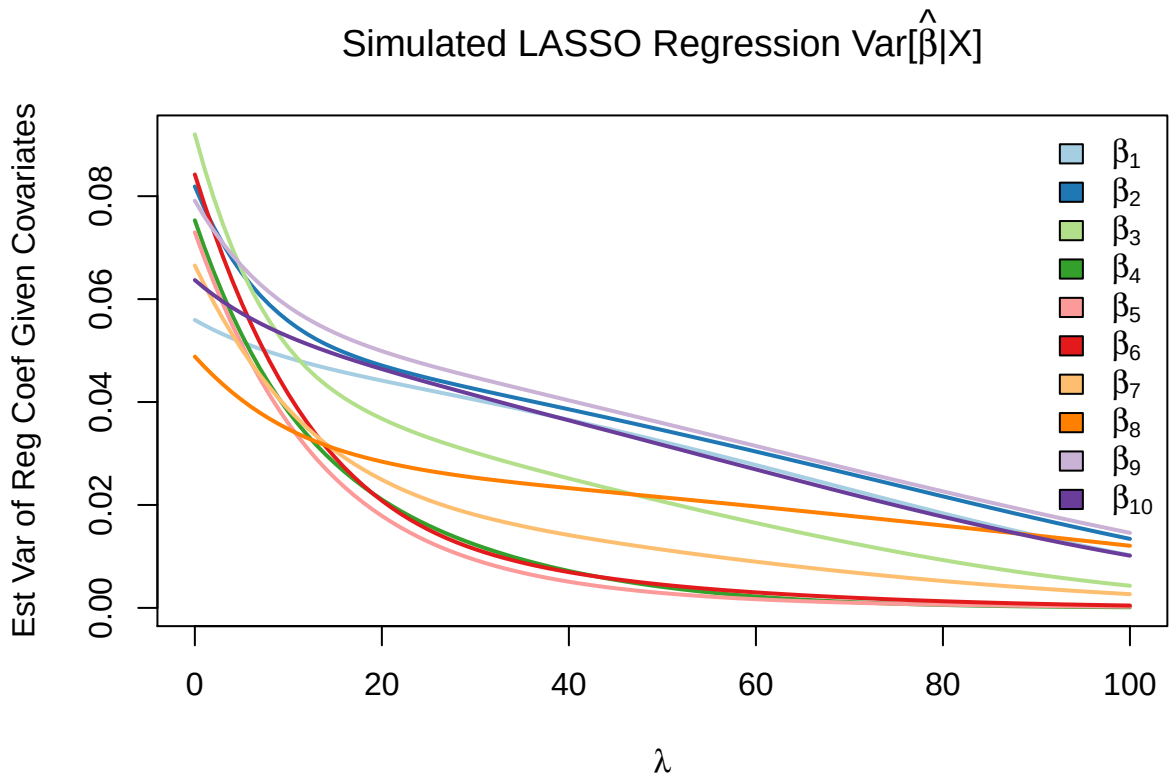
e. LASSO Regression: Bias, Variance, and Mean Squared Error of Estimated Regression Coefficients

```r
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")
```



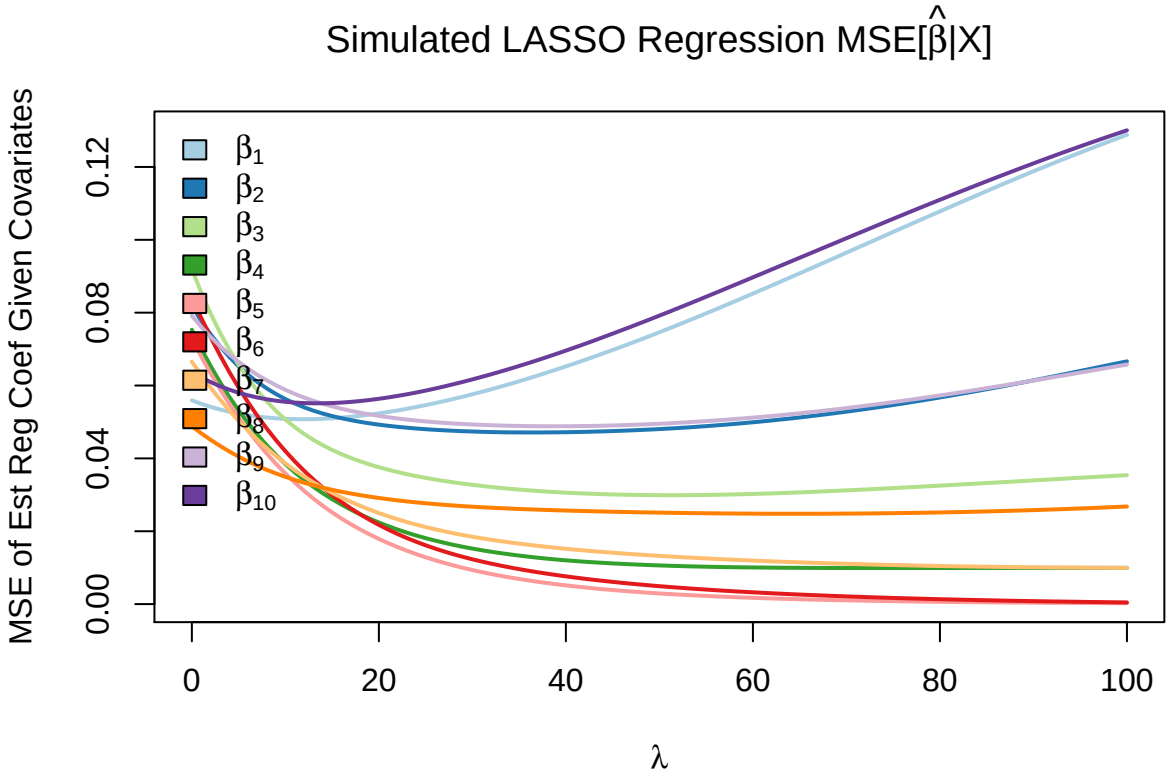
The plot of bias vs. λ shows that the bias increases as λ increases. This is expected because when we add penalties to the risk, we do a trade off between bias and variance. When $\lambda = 0$, bias $= 0$ (OLS). However, as λ increases, the estimates of β will no longer be unbiased.

```
#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])
}

lambda_min_MSE_lasso
```

```
## s88 s64 s49 s26  s0  s0  s0 s35 s61 s87
##  12  36  51  74 100 100 100  65  39  13
```

```
est_beta_min_MSE_lasso
```

```
##  [1] -0.43589025 -0.01245704 -0.33186376  0.00000000  0.00000000
##  [6]  0.00000000  0.00000000  0.22379891  0.20059475  0.44354652
```

```
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")
```