PH C240D/STAT C245D: Assignment 2

Rachael Phillips

10/19/2017

Regularized Regression

Question 1. Ridge and LASSO regression: Orthonormal covariates.

Consider the linear regression model for which $E[\mathbf{Y}_n|\mathbf{X}_n] = \mathbf{X}_n\beta$ and $Cov[\mathbf{Y}_n|\mathbf{X}_n] = \sigma^2\mathbf{I}_n$. Derive closed-form expressions for the ordinary least squares (OLS), ridge, and LASSO estimators of the regression coefficients β in the special case of *orthonormal covariates*, i.e., $\mathbf{X}_n^\top\mathbf{X}_n = \mathbf{I}_J$. Provide the effective degrees of freedom, bias, and covariance matrix of the ridge regression estimator.

Solution:

Note that for orthonormal covariates, $\mathbf{X}_n^\top\mathbf{X}_n = \mathbf{I}_J$, $\det(\mathbf{X}_n)^2 = 1$. In particular, $\det(\mathbf{X}_n) \neq 0$ so $\mathbf{X}_n$ is non-singular and has full rank.

OLS

We will begin with derivation of a closed-form expression for the ordinary least squares (OLS) estimator of the regression coefficients β. According to equation (10) on the 'Regularized Regression' lecture slides, for a design matrix of full column rank, $\hat{\beta}_n^{\text{OLS}} = (\mathbf{X}_n^\top\mathbf{X}_n)^{-1}\mathbf{X}_n^\top\mathbf{Y}_n$. Because our covariates are orthonormal (i.e. linearly independent) we have that $\hat{\beta}_n^{\text{OLS}} = \mathbf{X}_n^\top\mathbf{Y}_n$. For the parameter $\beta = (\beta_j : j = 1, ..., J) \in \mathbf{R}^J$, a J-dimensional column vector of regression coefficients with β_j the jth regression coefficient and $X = (X_j : j = 1, ..., J) \in \mathbf{R}^J$, a J-dimensional row vector of covariates, with X_j the jth covariate, we have that the OLS estimator of the jth regression coefficient $\beta_{n,j}$ is $\hat{\beta}_{n,j}^{\text{OLS}} = \sum_{i=1}^n X_{i,j}Y_i$.

LASSO

Next, we consider the derivation of closed-form expression for the LASSO regression estimator of the regression coefficients β. According to equation (27) on the 'Regularized Regression' lecture slides, $\hat{\beta}_n^{\text{LASSO}} \equiv \text{ argmin }_{\beta\in\mathbf{R}^J} \|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2 + \lambda\|\beta\|_1 = \text{ argmin }_{\beta\in\mathbf{R}^J} \sum_{i=1}^n \left(Y_i - \sum_{j=1}^J \beta_j X_{i,j}\right)^2 + \lambda\sum_{j=1}^J |\beta_j|$ where $\lambda \geq 0$. Equation (28) on the 'Regularized Regression' lecture slides points out that an equivalent definition of the LASSO estimator, which makes the constraint on the L1-norm more explicit, is $\hat{\beta}_n^{\text{LASSO}} = \text{ arg min }_{\beta\in\mathbf{R}^J} \|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2$ subject to $\|\beta\|_1 \leq k$ where there is a one-to-one correspondence between the shrinkage parameter, λ and the Lagrange multiplier, k. Note that this is also equivalent to $\hat{\beta}_n^{\text{LASSO}} = \text{ arg min }_{\beta\in\mathbf{R}^J} \sum_{i=1}^n \left(Y_i - \sum_{j=1}^J \beta_j X_{i,j}\right)^2$ subject to $\sum_{j=1}^J |\beta_j| \leq k$. We let $\hat{\beta}_j^{\text{OLS}}$ be the full least squares estimates and we let $k_0 = \sum_{j=1}^J |\hat{\beta}_j^{\text{OLS}}|$. Values of $k < k_0$ will cause shrinkage towards zero and if $k > k_0$ then $\hat{\beta}_j^{\text{LASSO}} = \hat{\beta}_j^{\text{OLS}}$. According to the 2006 Tibshirani article, *Regression Shrinkage and Selection via the Lasso*, for orthonormal covariates, the LASSO estimator has the form, $\hat{\beta}_j^{\text{LASSO}} = \text{sign}(\hat{\beta}_j^{\text{OLS}})(|\hat{\beta}_j^{\text{OLS}}| - \gamma)^+$ where γ is determined by the condition $\lambda = \sum_{j=1}^J |\hat{\beta}_j^{\text{LASSO}}|$. This is called a 'soft threshold' estimator by Donoho and Johnstone (1994). We can denote our estimator with respect to n as $\hat{\beta}_{n,j}^{\text{LASSO}} = \mathcal{S}(\hat{\beta}_{n,j}^{\text{OLS}}, \lambda)$ where the 'soft-thresholding operator', $\mathcal{S}$, is defined as

$$\mathcal{S}(z, \lambda) = \begin{cases} z - \lambda & z > \lambda \\ 0 & |z| \leq \lambda \\ z + \lambda & z < -\lambda \end{cases}$$

Ridge

Finally, we consider the derivation of closed-form expression for the ridge estimator of the regression coefficients β. We also provide the effective degrees of freedom, bias, and covariance matrix of the ridge regression estimator. According to equation (20) on the 'Regularized Regression' lecture slides, $\hat{\beta}_n^{\text{ridge}} = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} \mathbf{X}_n^\top \mathbf{Y}_n$. Since $\mathbf{X}_n^\top \mathbf{X}_n = \mathbf{I}_J$ we have that $\hat{\beta}_n^{\text{ridge}} = (1+\lambda)^{-1} \mathbf{X}_n^\top \mathbf{Y}_n = \frac{1}{1+\lambda} \mathbf{X}_n^\top \mathbf{Y}_n = \frac{1}{1+\lambda} \hat{\beta}_n^{OLS}$. Thus, for the parameter $\beta = (\beta_j : j = 1, ..., J) \in \mathbf{R}^J$, a J-dimensional column vector of regression coefficients with β_j the jth regression coefficient and $X = (X_j : j = 1, ..., J) \in \mathbf{R}^J$, a J-dimensional row vector of covariates, with X_j the jth covariate, we have that the ridge regression estimator of the jth regression coefficient $\beta_{n,j}$ is $\hat{\beta}_{n,j}^{\text{ridge}} = \frac{1}{1+\lambda} \sum_{i=1}^n X_{i,j} Y_i$

According to equation (21) on the 'Regularized Regression' lecture slides, $E[\hat{\beta}_n^{\text{ridge}} | \mathbf{X}_n] = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} \mathbf{X}_n^\top \mathbf{X}_n \beta = \frac{1}{1+\lambda} \beta$. The bias is as follows, $\text{Bias}[\hat{\beta}_n^{\text{ridge}} | \mathbf{X}_n] = E[\hat{\beta}_n^{\text{ridge}} | \mathbf{X}_n] - E[\mathbf{Y}_n | \mathbf{X}_n] = E[\hat{\beta}_n^{\text{ridge}} | \mathbf{X}_n] - \beta = \frac{1}{1+\lambda} \beta - \beta = (\frac{1}{1+\lambda} - 1) \beta = -\frac{\lambda}{1+\lambda} \beta$.

According to equation (22) on the 'Regularized Regression' lecture slides, the covariance matrix of the ridge regression estimator, $Cov[\hat{\beta}_n^{\text{ridge}} | \mathbf{X}_n] = \sigma^2 (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} \mathbf{X}_n^\top \mathbf{X}_n (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} = \sigma^2 \frac{1}{1+\lambda} \mathbf{I}_J \frac{1}{1+\lambda} = \frac{\sigma^2 \mathbf{I}_J}{(1+\lambda)^2}$.

According to equation (26) on the 'Regularized Regression' lecture slides, for ridge regression, the effective degrees of freedom, $df^{\text{ridge}}(\lambda) = \sum_{j=1}^J \frac{L_j^2}{L_j^2 + \lambda}$ where L_J are the singular values of $\mathbf{X}_n$. Thus, for our orthonormal covariates, $df^{\text{ridge}}(\lambda) = \sum_{j=1}^J \frac{1}{1+\lambda} = \frac{J}{1+\lambda}$.

Question 2. Ridge and LASSO regression: Bayesian interpretation.

Ridge and LASSO estimators also arise within a *Bayesian* framework, as the *mode of a posterior distribution* for the regression coefficients β, with a suitably-chosen *prior distribution*. Consider the Gaussian linear regression model $\mathbf{Y}_n | \mathbf{X}_n, \beta \sim N(\mathbf{X}_n \beta, \sigma^2 \mathbf{I}_n)$, with σ^2 known.

a) Ridge regression.

Propose a prior distribution for β for which the ridge regression estimator of β is the mode of the posterior distribution for β. Comment on how the prior parameters control shrinkage.

Solution:

The posterior in β is proportional to the likelihood (Gaussian) times the prior (Gaussian); $p(\beta | \mathbf{Y}_n, \mathbf{X}_n) \propto p(\mathbf{Y}_n | \mathbf{X}_n, \beta) p(\beta)$. Gaussians $\times$ Gaussian is still Gaussian. So the posterior is a Gaussian with a mean β_G and a covariance Σ_G. Now we will show that $\beta_G = \hat{\beta}_n^{\text{ridge}}$. We know the posterior is a Gaussian. So the mode=mean=maximum. Therefore $\beta_G = \text{argmax}_\beta \mathcal{L}(\mathbf{Y}_n | \mathbf{X}_n, \beta) P(\beta)$ where $\mathcal{L}(\mathbf{Y}_n | \mathbf{X}_n, \beta)$ is the likelihood of the data parameterized as defined above and $P(\beta)$ is the prior, $N(0, \mathbf{I}_J \delta^2)$. Recall that the log function is a monotone transformation thus $\beta_G = \text{argmax}_\beta log \mathcal{L}(\mathbf{Y}_n | \mathbf{X}_n, \beta) + log P(\beta)$. Removing constants this leads to $\beta_G = \text{argmax}_\beta - \frac{1}{2\sigma^2} (\mathbf{Y}_n - \mathbf{X}_n \beta)^\top (\mathbf{Y}_n - \mathbf{X}_n \beta) - \frac{1}{2\delta^2} \beta^\top \beta$. This becomes a minimization problem when we change the sign and getting rid of additive constants and multiplicative terms we have that $\beta_G = \text{argmin}_\beta \frac{1}{\sigma^2} (\beta^\top \mathbf{X}_n^\top \mathbf{X}_n \beta - 2\beta^\top \mathbf{X}_n^\top \mathbf{Y}_n) + \frac{1}{\delta^2} \beta^\top \beta$. Here, we see that we are getting close to ridge (recall, $\hat{\beta}_n^{\text{ridge}} = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} \mathbf{X}_n^\top \mathbf{Y}_n$). After multiplying the objective by σ^2, $\beta_G = \text{argmin}_\beta \beta^\top \Big(\mathbf{X}_n^\top \mathbf{X}_n + \frac{\sigma^2}{\delta^2} \mathbf{I}_J \Big) \beta - 2\beta^\top \mathbf{X}_n^\top \mathbf{Y}_n$ and setting the gradient in β to zero yields $\Big(\mathbf{X}_n^\top \mathbf{X}_n + \frac{\sigma^2}{\delta^2} \mathbf{I}_J \Big) \beta_G = \mathbf{X}_n^\top \mathbf{Y}_n$. Thus, $\beta_G = \Big(\mathbf{X}_n^\top \mathbf{X}_n + \frac{\sigma^2}{\delta^2} \mathbf{I}_J \Big)^{-1} \mathbf{X}_n^\top \mathbf{Y}_n = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1} \mathbf{X}_n^\top \mathbf{Y}_n = \hat{\beta}_n^{\text{ridge}} \Leftrightarrow \lambda = \frac{\sigma^2}{\delta^2}$ and the variance of the prior distribution inversely controls the shrinkage parameter. That is, as the variance of the prior increases the shrinkage parameter decreases.

b) LASSO regression.

Propose a prior distribution for β for which the LASSO regression estimator of β is the mode of the posterior distribution for β. Comment on how the prior parameters control shrinkage.

Solution:

We take the prior for β_j to be independent Laplace (as suggested in Tibshirani 1996) with mean zero and some scale δ so $P(\beta|\delta) \propto e^{-\frac{1}{2\delta}\sum_{j=1}^{J}|\beta_j|}$ and we have the same model for the data as in part a with the same likelihood function. Like part a, this becomes a minimization problem when we change the sign and we get rid of additive constants and take the log transform of the posterior distribution to obtain $\frac{1}{2\sigma^2}(\mathbf{Y}_n - \mathbf{X}_n\beta)^\top(\mathbf{Y}_n - \mathbf{X}_n\beta) + \frac{1}{\delta}\sum_{j=1}^{J}|\beta_j| = \frac{1}{2\sigma^2}\|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2 + \frac{1}{\delta}\|\beta\|_1 = \|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2 + \lambda\|\beta\|_1 = \hat{\beta}_n^{\text{LASSO}} \Leftrightarrow \lambda = \frac{2\sigma^2}{\delta}$. Just like part a, the Maximum A Posteriori (MAP) estimator for β minimizes the above equation. Thus, here, the MAP estimator for β is LASSO and the scale of the prior distribution inversely controls the shrinkage parameter. That is, as the sccale of the prior increases the shrinkage parameter decreases.

Question 3. Elastic net: Simulation study.

a) Simulation model.

Consider the data structure $(X, Y) \sim P$, where $Y \in \mathbb{R}$ is a scalar outcome and $X = (X_j : j = 1, ..., J) \in \mathbb{R}^J$ a J-dimensional vector of covariates. Assume the following Gaussian linear regression model

$$Y|X \sim N(X^T\beta, \sigma^2) \text{ and } X \sim N(0_{J\times 1}, \Gamma), (1)$$

where $0_{J\times 1}$ is a J-dimensional column vector of zeros and the covariance matrix $\Gamma = (\gamma_{j,j'} : j, j' = 1, ..., J)$ of the covariates has an autocorrelation of order 1, i.e., AR(1), structure,

$$\gamma_{j,j'} = \rho^{|j-j'|},$$

for $\rho \in (-1, 1)$. Set the parameter values to $J = 10$, $\beta = (-J/2+1, ..., -2, -1, 0, 0, 1, 2, ..., J/2-1)/10$, $\sigma = 2$, and $\rho = 0.5$.

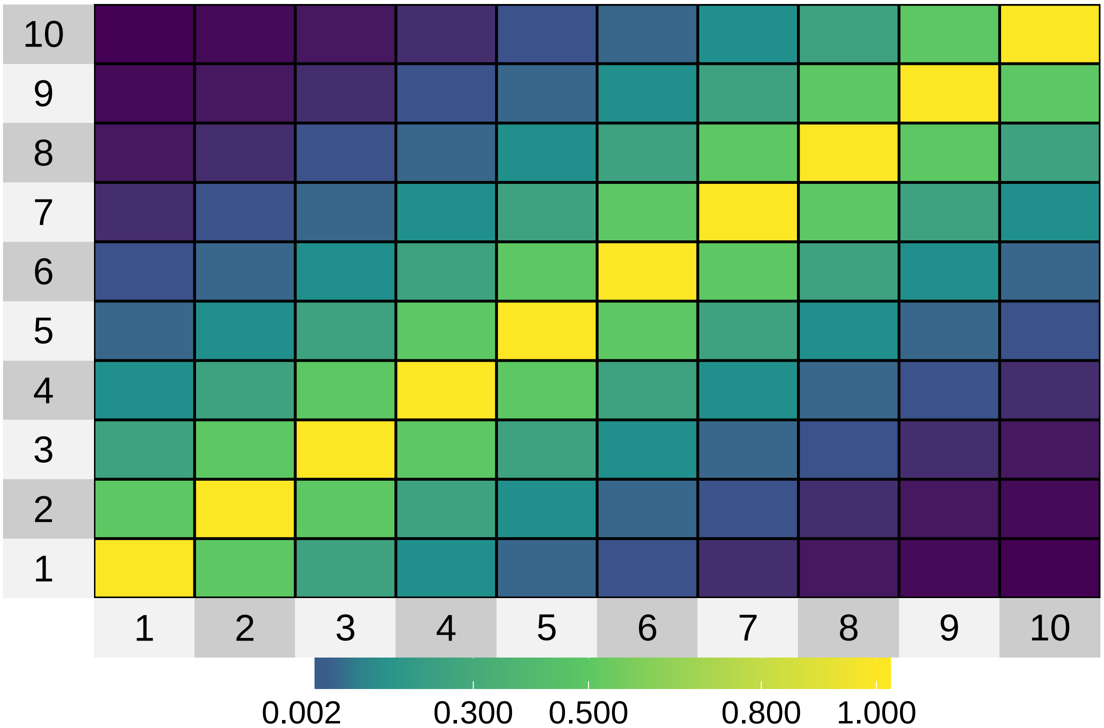
Simulate a learning set $\mathcal{L}_n = (X_i, Y_i) : i = 1, ..., n$ of $n = 100$ independent and identically distributed (IID) random variables $(X_i, Y_i) \sim P, i = 1, ..., n$. Also simulate an independent test set $\mathcal{T}_{n_{TS}} = \{(X_i, Y_i) : i = 1, ..., n_{TS}\}$ of $n_{TS} = 1,000$ IID $(X_i, Y_i) \sim P$, $i = 1, ..., n_{TS}$.

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

Solution:

```
##########################
# Given Information
#########################

# Thank you, Kelly!
J <- 10
rho <- .5
sigma <- 2

# Variance of the Covariates, gamma
gamma <- sapply(1:10, function(i){
  sapply(1:10, function(j){
```

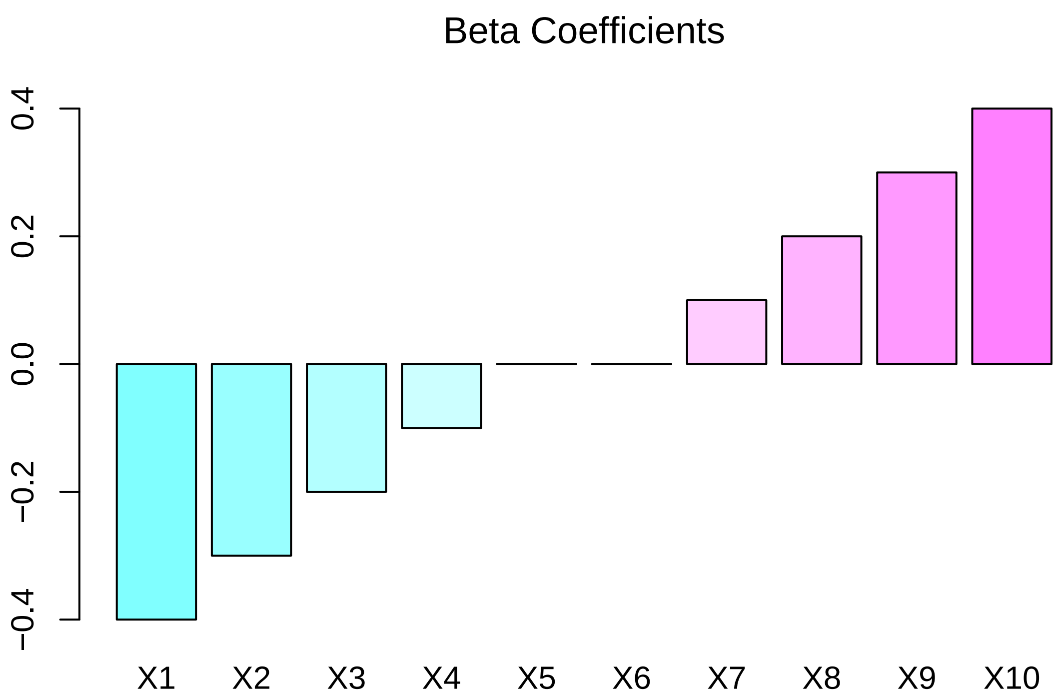
```
    rho^(abs(i-j))
  })
})
# visualize this 10 X 10 symmetric matrix
# with i columns
superheat(gamma)
```




```
# regression coefficients, beta
Beta <- c((-J/2+1):0,0:(J/2-1))/10
Beta
```

```
##  [1] -0.4 -0.3 -0.2 -0.1  0.0  0.0  0.1  0.2  0.3  0.4
```

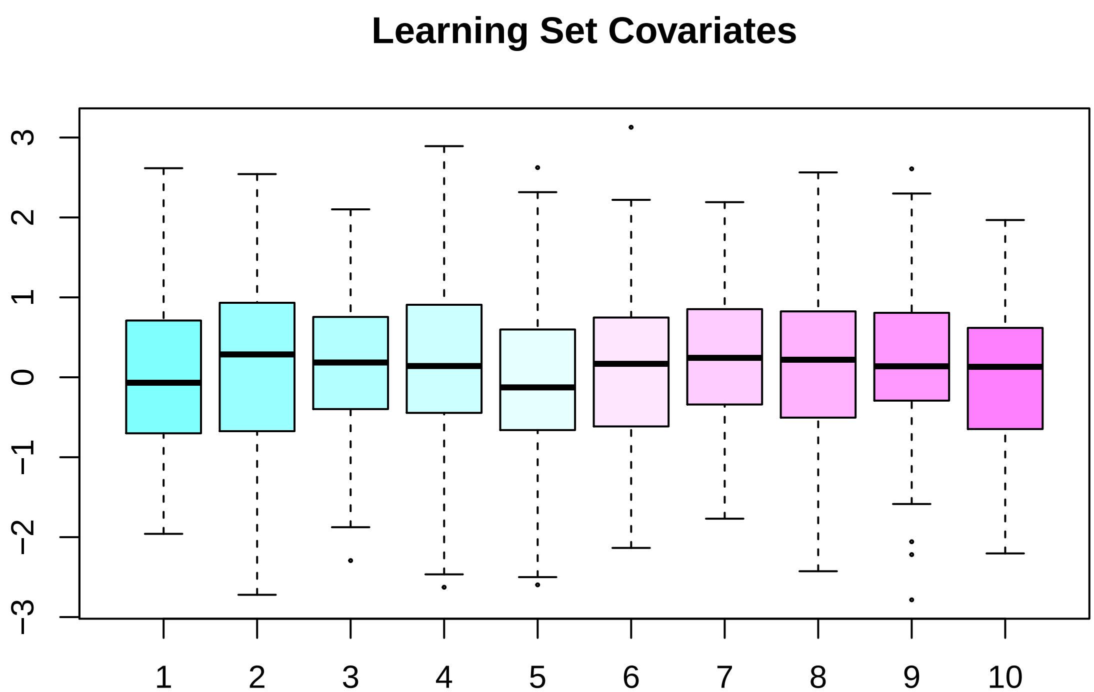
```
barplot(Beta, main = expression("Beta Coefficients"),
col=cm.colors(10), names.arg = c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                  "X8", "X9", "X10"))
```



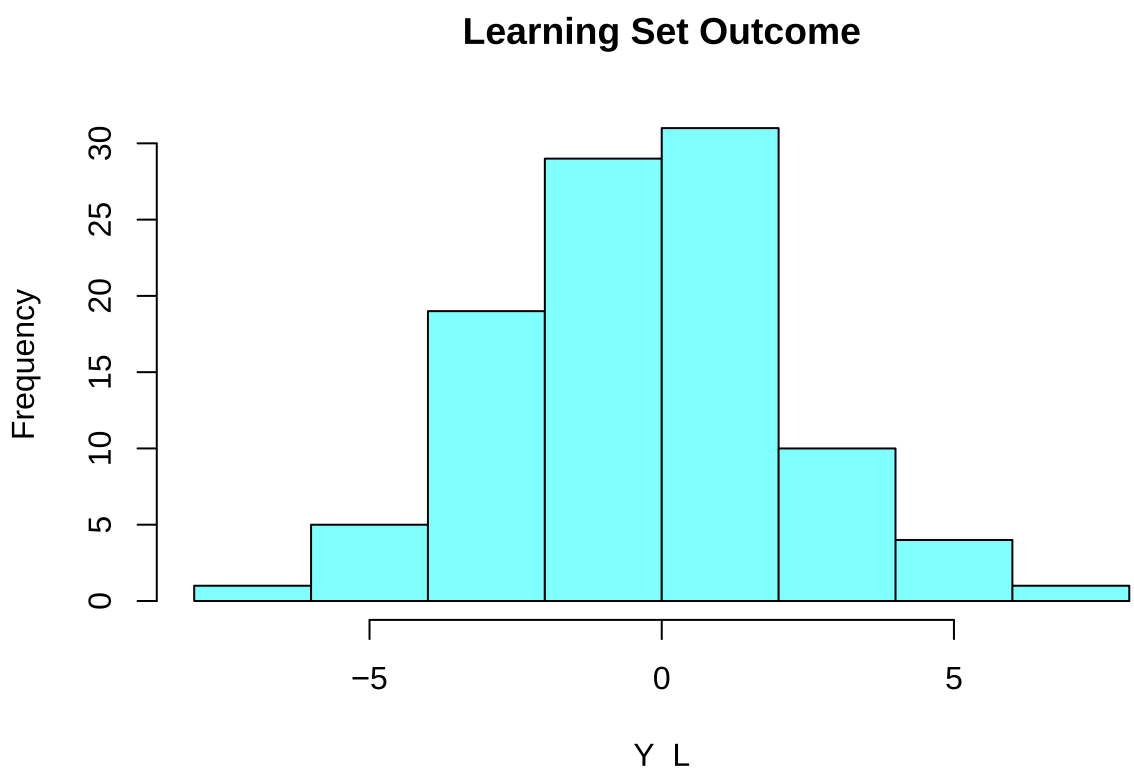

```r
##########################
# Learning Set Simulation
##########################

n <- 100

# Covariates
X_L <- mvrnorm(n = n, mu = rep(0,J), Sigma = gamma)
dim(X_L)
```

```
## [1] 100  10
```

```r
boxplot(X_L, cex=.2, col=cm.colors(10),
        main = "Learning Set Covariates")
```




```
# Outcome
Y_L <- rnorm(n = n, mean = X_L %*% Beta, sd = sigma)
hist(Y_L,col=cm.colors(1), main = "Learning Set Outcome")
```



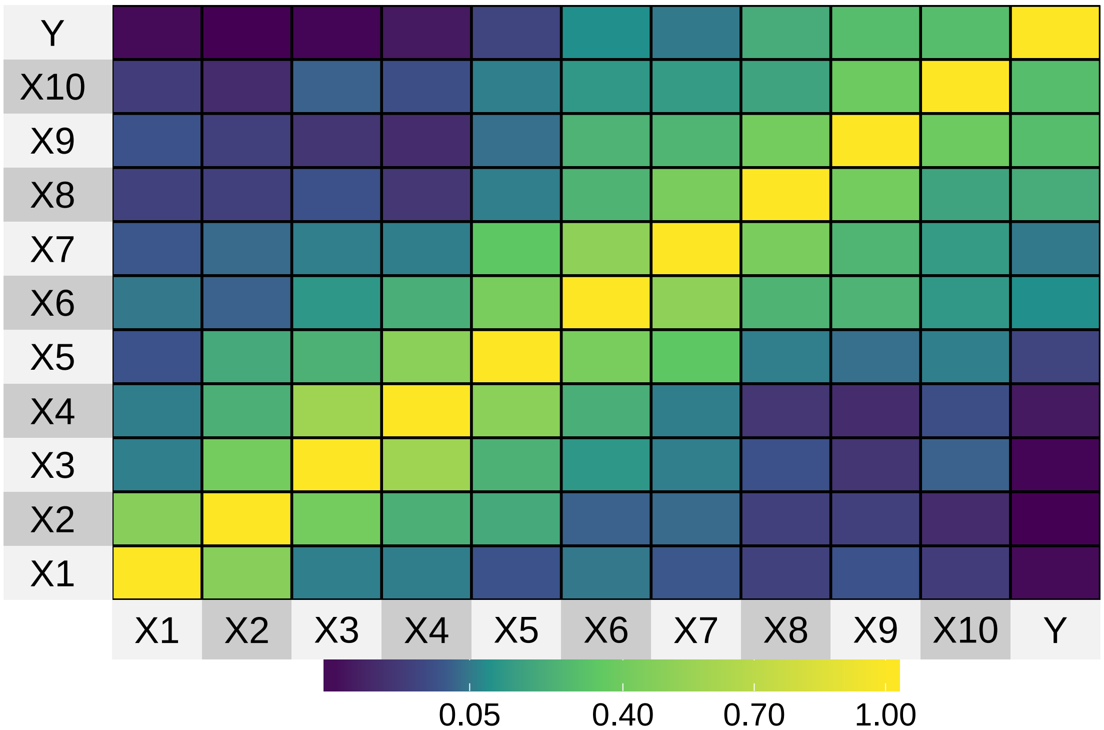

```
summary(Y_L)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -7.7704 -1.9703 -0.3373 -0.2294  1.4615  6.2551
```

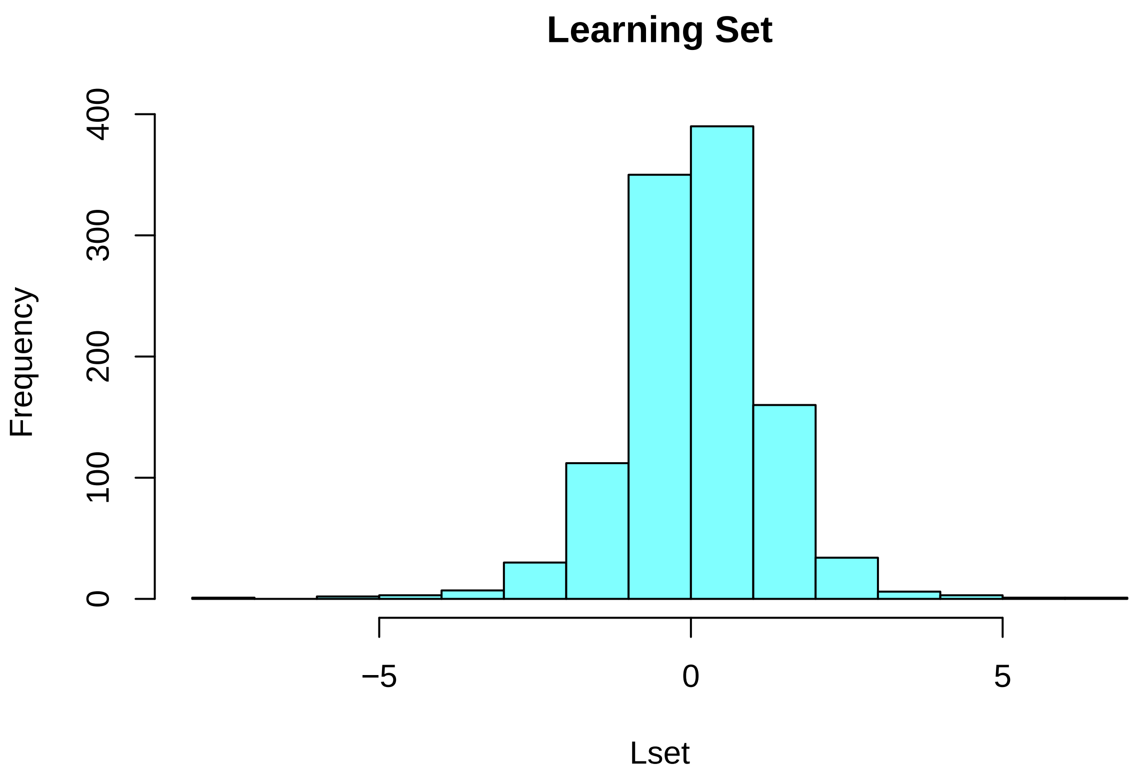
```r
# Combine Outcome and Covariates to make Learning Set
# we know there should be negative correlation here
Lset_X <- X_L[(1:100),]
Lset_Y <- Y_L[(1:100)]
Lset <- cbind(Lset_X,Lset_Y)
colnames(Lset) <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                 "X8", "X9", "X10", "Y")
summary(Lset)
```

```
##        X1                 X2                X3                X4
##  Min.   :-1.95874   Min.   :-2.7213   Min.   :-2.2930   Min.   :-2.6267
##  1st Qu.:-0.69534   1st Qu.:-0.6581   1st Qu.:-0.3952   1st Qu.:-0.4249
##  Median :-0.06796   Median : 0.2861   Median : 0.1852   Median : 0.1412
##  Mean   :-0.03344   Mean   : 0.1335   Mean   : 0.1663   Mean   : 0.1764
##  3rd Qu.: 0.70768   3rd Qu.: 0.9254   3rd Qu.: 0.7546   3rd Qu.: 0.9027
##  Max.   : 2.61621   Max.   : 2.5425   Max.   : 2.1014   Max.   : 2.8924
##        X5                X6                X7                X8
##  Min.   :-2.5969   Min.   :-2.13479   Min.   :-1.7691   Min.   :-2.4266
##  1st Qu.:-0.6601   1st Qu.:-0.60028   1st Qu.:-0.3398   1st Qu.:-0.5041
##  Median :-0.1262   Median : 0.16903   Median : 0.2436   Median : 0.2203
##  Mean   :-0.0312   Mean   : 0.09747   Mean   : 0.2553   Mean   : 0.2037
##  3rd Qu.: 0.5899   3rd Qu.: 0.73927   3rd Qu.: 0.8498   3rd Qu.: 0.8139
##  Max.   : 2.6238   Max.   : 3.12858   Max.   : 2.1915   Max.   : 2.5634
##        X9               X10                 Y
##  Min.   :-2.7842   Min.   :-2.20344   Min.   :-7.7704
##  1st Qu.:-0.2883   1st Qu.:-0.64708   1st Qu.:-1.9703
##  Median : 0.1375   Median : 0.13122   Median :-0.3373
##  Mean   : 0.1998   Mean   : 0.02295   Mean   :-0.2294
##  3rd Qu.: 0.7767   3rd Qu.: 0.61816   3rd Qu.: 1.4615
##  Max.   : 2.6084   Max.   : 1.96785   Max.   : 6.2551
```

```r
# we can visualize the correlations in a heatmap
superheat(cor(Lset))
```

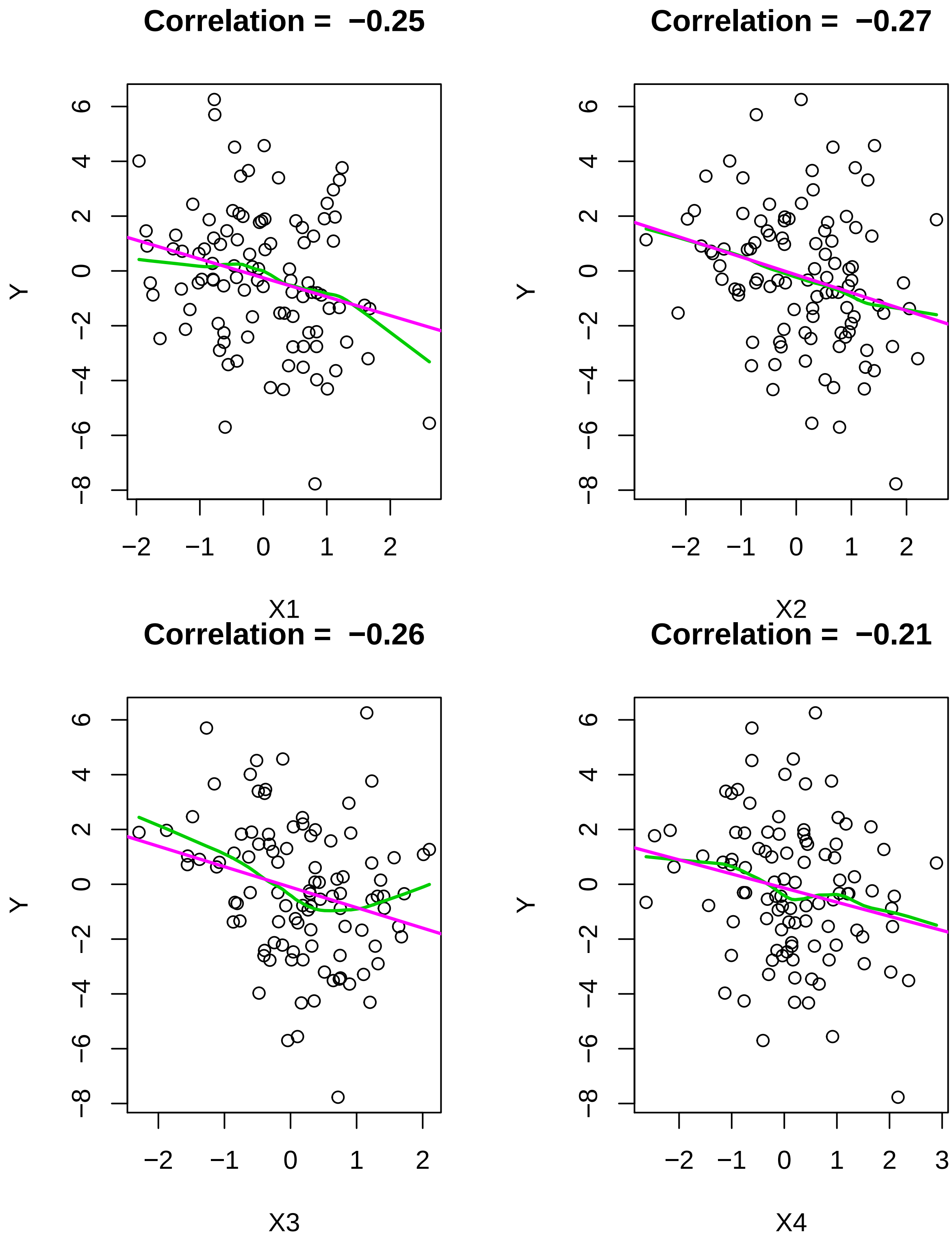


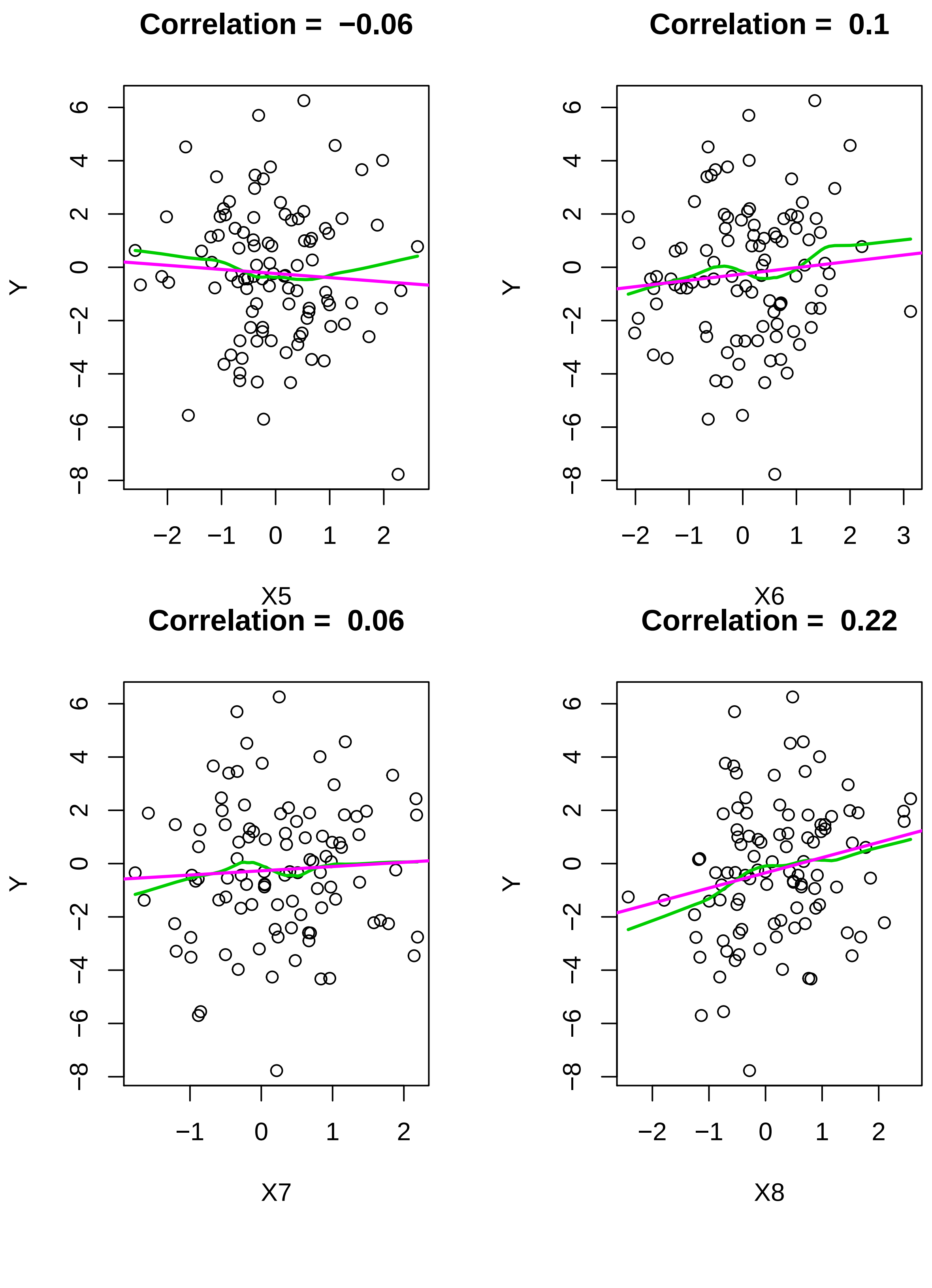

```
hist(Lset, col=cm.colors(1), main = "Learning Set")
```

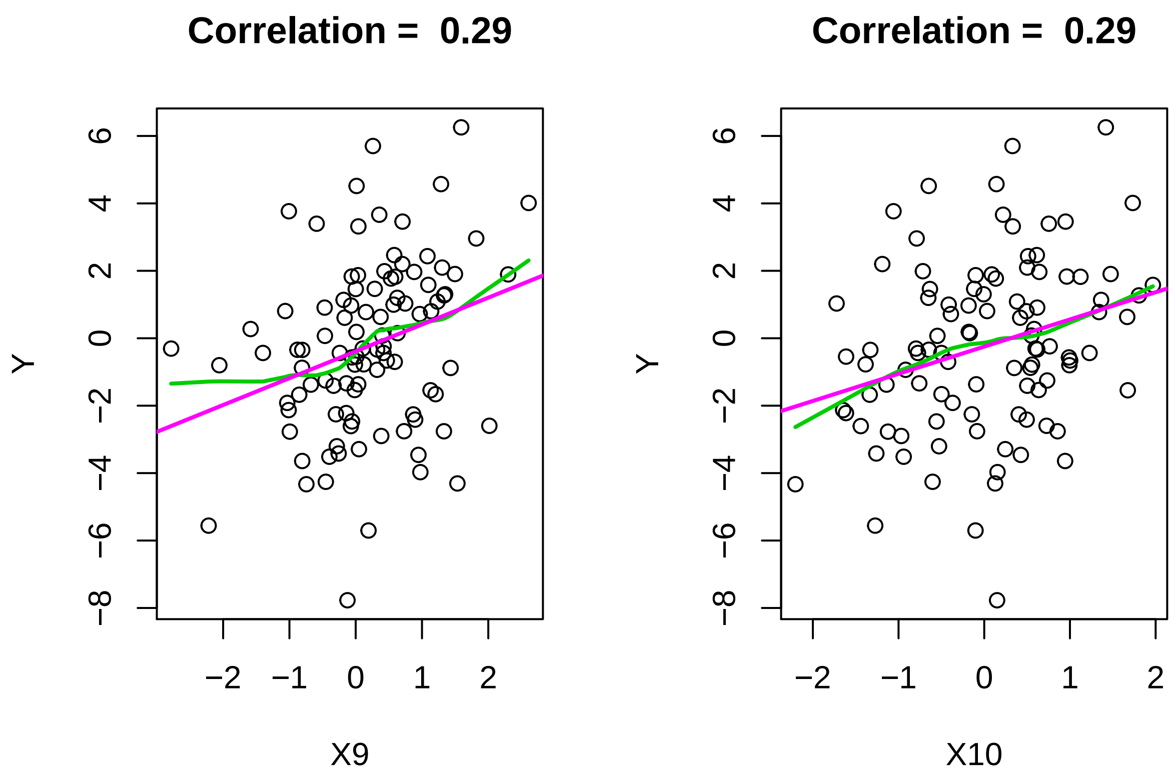



```
# Adapted from Sandrine's 'Regularized Regression:Example' code
par(mfrow=c(1,2))
for(J in 1:10){
  plot(Lset[,J], Lset_Y, xlab = colnames(Lset)[J],
       ylab = "Y", main = paste("Correlation =  ", round(cor(Lset[,c(J,11)])[1,2],2), sep = ""))
```

```
  lines(lowess(Y_L ~ X_L[,J]), col=123, lwd =2)
  abline(lm(Y_L ~ X_L[,J])$coef, col = 54, lwd=2)
}
```




Correlation = −0.06
Correlation = 0.1
Correlation = 0.06
Correlation = 0.22
Y
X5
X6
X7
X8
6
4
2
0
−2
−4
−6
−8
−2
−1
0
1
2
3




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

###########################
# Test Set Simulation
###########################

n <- 1000

# Covariates
X_T <- mvrnorm(n = n, mu = rep(0,J), Sigma = gamma)
dim(X_T)
```

```
## [1] 1000   10
```

```
# Outcome
Y_T <- rnorm(n = n, mean = X_T %*% Beta, sd = sigma)

# Combine Outcome and Covariates to make Test Set
Tset_X <- X_T[(1:1000),]
Tset_Y <- Y_T[(1:1000)]
Tset <- cbind(Tset_X,Tset_Y)
colnames(Tset) <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                    "X8", "X9", "X10", "Y")
```

b) Elastic net regression on learning set.

The *elastic net* estimator of the regression coefficients β is defined as

$$\hat{\beta}_n^{\text{enet}} \equiv \arg\min_{\beta \in \mathbb{R}^J} \|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$$

$$= \arg\min_{\beta \in \mathbb{R}^J} \sum_{i=1}^{n} \left(Y_i - \sum_{j=1}^{J} \beta_j X_{i,j} \right)^2$$

$$+\lambda_1 \sum_{j=1}^{J} |\beta_j| + \lambda_2 \sum_{j=1}^{J} \beta_j^2$$

where the shrinkage parameters $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are tuning parameters that control the strength of the penalty terms, i.e., the complexity or shrinking of the coe cients towards zero.

Obtain ridge ($\lambda_1 = 0, \lambda_2 = \lambda$), LASSO ($\lambda_1 = \lambda, \lambda_2 = 0$), and elastic net ($\lambda_1 = \lambda_2 = \lambda/2$) estimators of the regression coefficients β, for $\lambda \in \{0, 1, ..., 100\}$, based on the learning set simulated in a).

In particular, for each type of estimator, provide and comment on plots of the effective degrees of freedom versus the shrinkage parameter λ and plots of the estimated regression coefficients versus the shrinkage parameter.

For each type of estimator, obtain the learning set risk for the squared error loss function, i.e., the mean squared error (MSE),

$$MSE(\hat{\beta}_n; \mathcal{L}_n) = \frac{1}{n} \|\mathbf{Y}_n - \mathbf{X}_n \hat{\beta}_n\|_2^2.$$

Provide and comment on plots of the MSE versus the shrinkage parameter and report which values of the shrinkage parameter minimize risk.

Hint. You may use the `glmnet` function from the `glmnet` package, but be mindful of centering and scaling, of the handling of the intercept, and of the parameterization of the elastic net penalty.

Solution:

```
############################
# Setting up
############################

# Directly from Sandrine's 'Regularized Regression:Example' code

## Elastic net
## N.B. alpha = lambda1/(lambda1+2*lambda2), lambda = (lambda1+2*lambda2)/(2*n)
myGlmnet <- function(x,y,x.new=NULL,intercept=TRUE,scale=TRUE,alpha=0,lambda=0,thresh=1e-12)
  {
    n <- nrow(x)
    J <- ncol(x)
    xx <- scale(x,center=TRUE,scale=scale)
    beta0.hat <- mean(y)
    y.new <- NULL

    res <- glmnet(xx,y/sd(y),alpha=alpha,lambda=lambda,intercept=FALSE,standardize=FALSE,thresh=thresh)
    if(alpha == 0)
      df <- sapply(lambda*n, function(l) sum(diag(xx%*%solve(crossprod(xx)+l*diag(J))%*%t(xx)))) + inte
    else
      df <- rev(res$df) + intercept
    beta.hat <- as.matrix(t(coef(res)[-1,length(lambda):1])*sd(y))
    rownames(beta.hat) <- NULL
    y.hat <- t(predict(res,newx=xx,s=lambda)*sd(y))
    if(intercept)
      {
        beta.hat <- cbind(rep(beta0.hat,length(lambda)),beta.hat)
```

```r
      y.hat <- y.hat + beta0.hat
    }

  e <- scale(y.hat,center=y,scale=FALSE)
  mse <- rowMeans(e^2)

  if(!is.null(x.new))
    y.new <- t(predict(res,newx=scale(x.new,center=TRUE,scale=scale),s=lambda))*sd(y)+beta0.hat*interc

  res <- list(df=df,beta.hat=beta.hat,mse=mse,y.hat=y.hat,e=e,y.new=y.new)
  res

}

# Beta vs. lambda
myPlotBeta <- function(x,beta,type="l",lwd=2,lty=1,col=1:ncol(beta),
                       xlab=expression(lambda),ylab="",labels=paste(1:ncol(beta)),
                       zero=TRUE,right=FALSE,main="",...)
 {
  matplot(x,beta,type=type,lwd=lwd,lty=lty,col=col,xlab=xlab,ylab=ylab,main=main,...)
  if(right)
    text(x[length(x)],beta[length(x),],labels=labels,col=col)
  if(!right)
    text(x[1],beta[1,],labels=labels,col=col)
  if(zero)
    abline(h=0,lty=2)
}

###########################
# Estimators
###########################

lambda <- seq(0,100,by=1)

ridge <- myGlmnet(x = Lset_X, y = Lset_Y, x.new = Tset_X, alpha = 0, lambda = lambda/(nrow(Lset_X)))
lasso <- myGlmnet(x = Lset_X, y = Lset_Y, x.new = Tset_X, alpha = 1, lambda = lambda/(2*nrow(Lset_X)))
enet <- myGlmnet(x = Lset_X, y = Lset_Y, x.new = Tset_X, alpha = 1/2, lambda = (3*lambda)/(4*nrow(Lset_

# Compare to lm
lm <- lm(Lset_Y~ scale(Lset_X), center = TRUE, scale = FALSE)
summary(lm)
```

```
##
## Call:
## lm(formula = Lset_Y ~ scale(Lset_X), center = TRUE, scale = FALSE)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2043 -1.5272 -0.0269  1.1703  6.7727
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -0.22936    0.23221  -0.988   0.3260
## scale(Lset_X)1  -0.51600    0.27455  -1.879   0.0635 .
```

```
## scale(Lset_X)2  -0.05614    0.30792  -0.182   0.8558
## scale(Lset_X)3  -0.50256    0.31396  -1.601   0.1130
## scale(Lset_X)4  -0.12197    0.32704  -0.373   0.7101
## scale(Lset_X)5  -0.09036    0.30321  -0.298   0.7664
## scale(Lset_X)6   0.29198    0.29509   0.989   0.3251
## scale(Lset_X)7  -0.20904    0.29772  -0.702   0.4844
## scale(Lset_X)8   0.27444    0.27891   0.984   0.3278
## scale(Lset_X)9   0.32710    0.28078   1.165   0.2471
## scale(Lset_X)10  0.50604    0.25841   1.958   0.0533 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.322 on 89 degrees of freedom
## Multiple R-squared:  0.2468, Adjusted R-squared:  0.1622
## F-statistic: 2.916 on 10 and 89 DF,  p-value: 0.003334
```

```r
ridge$beta.hat[1,]
```

```
##                     V1          V2          V3          V4          V5
## -0.22935632 -0.51600124 -0.05613818 -0.50255491 -0.12196832 -0.09035874
##          V6          V7          V8          V9         V10
##  0.29197504 -0.20904487  0.27443703  0.32710005  0.50603746
```

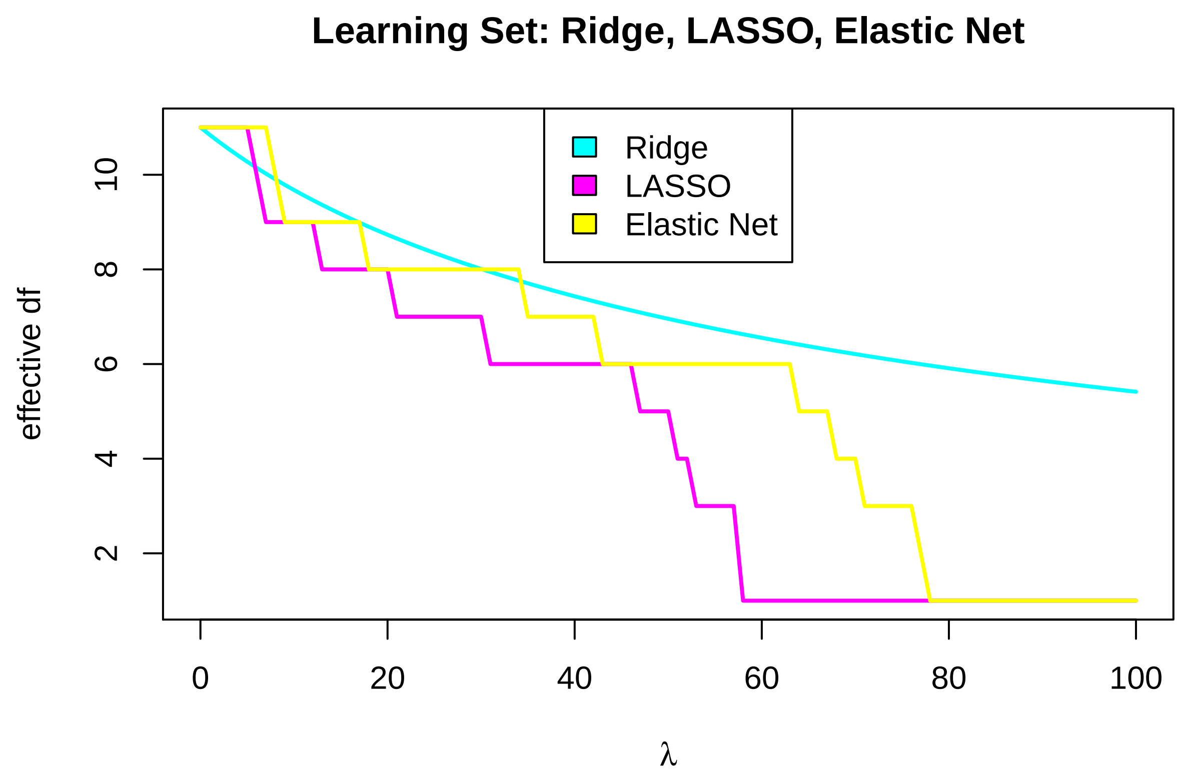
```r
lasso$beta.hat[1,]
```

```
##                     V1          V2          V3          V4          V5
## -0.22935632 -0.51600349 -0.05613554 -0.50255703 -0.12196704 -0.09035998
##          V6          V7          V8          V9         V10
##  0.29197582 -0.20904499  0.27443741  0.32709948  0.50603788
```

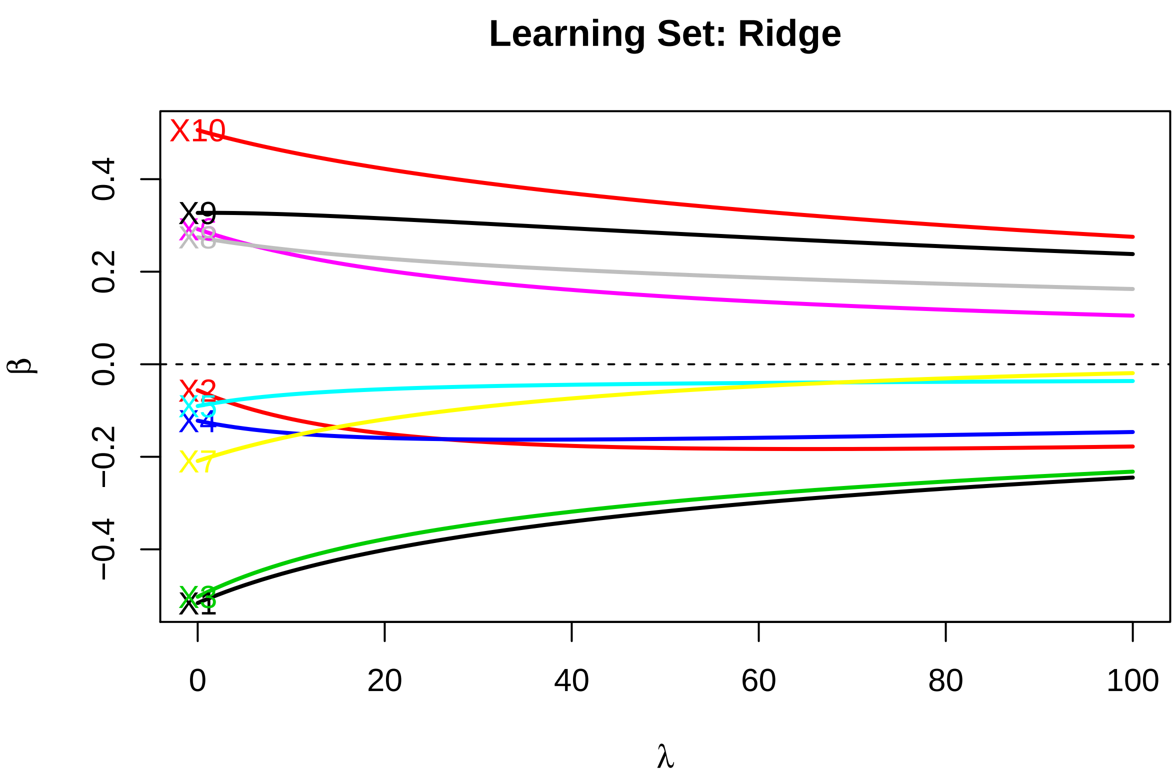
```r
enet$beta.hat[1,]
```

```
##                     V1          V2          V3          V4          V5
## -0.22935632 -0.51600173 -0.05613761 -0.50255547 -0.12196777 -0.09035895
##          V6          V7          V8          V9         V10
##  0.29197518 -0.20904499  0.27443705  0.32710017  0.50603749
```

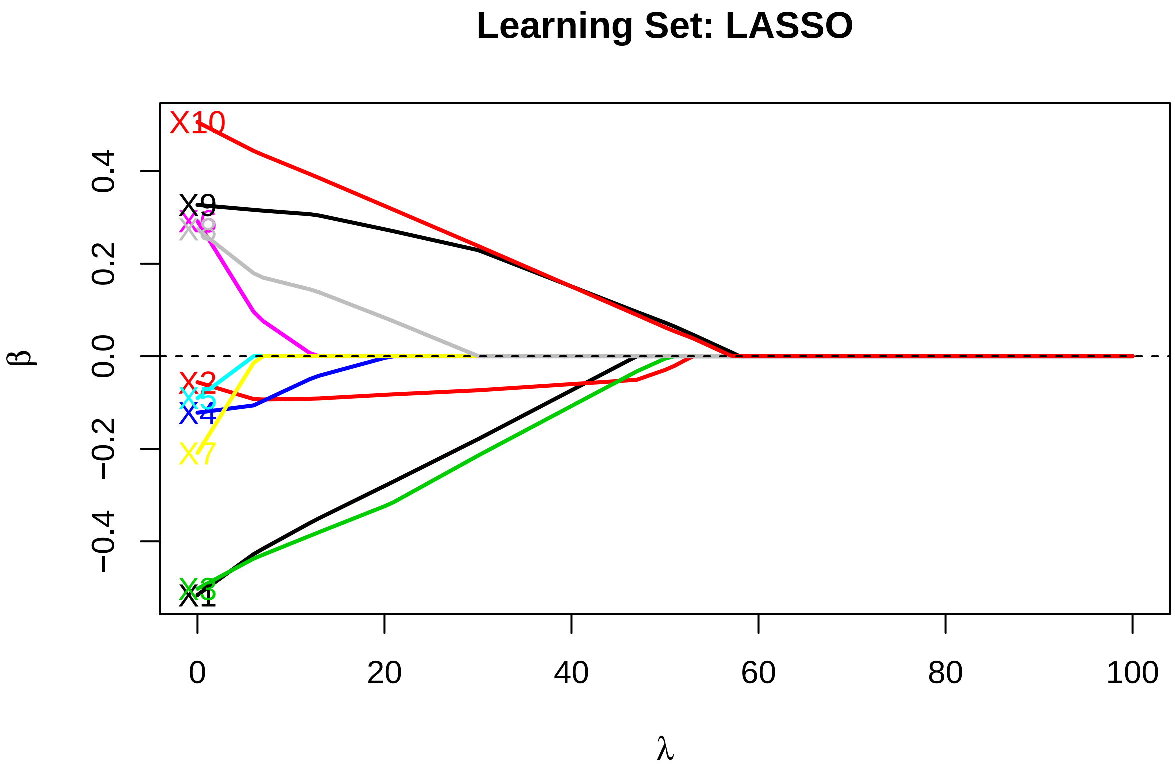
```r
###########################
# Plots
###########################

# Effective df vs. lambda
matplot(lambda, cbind(ridge$df,lasso$df, enet$df),
        type="l", lwd=2, lty=1, col=5:7,
        xlab=expression(lambda), ylab = "effective df",
        main="Learning Set: Ridge, LASSO, Elastic Net")
legend("top", c("Ridge","LASSO", "Elastic Net"), fill=5:7)
```



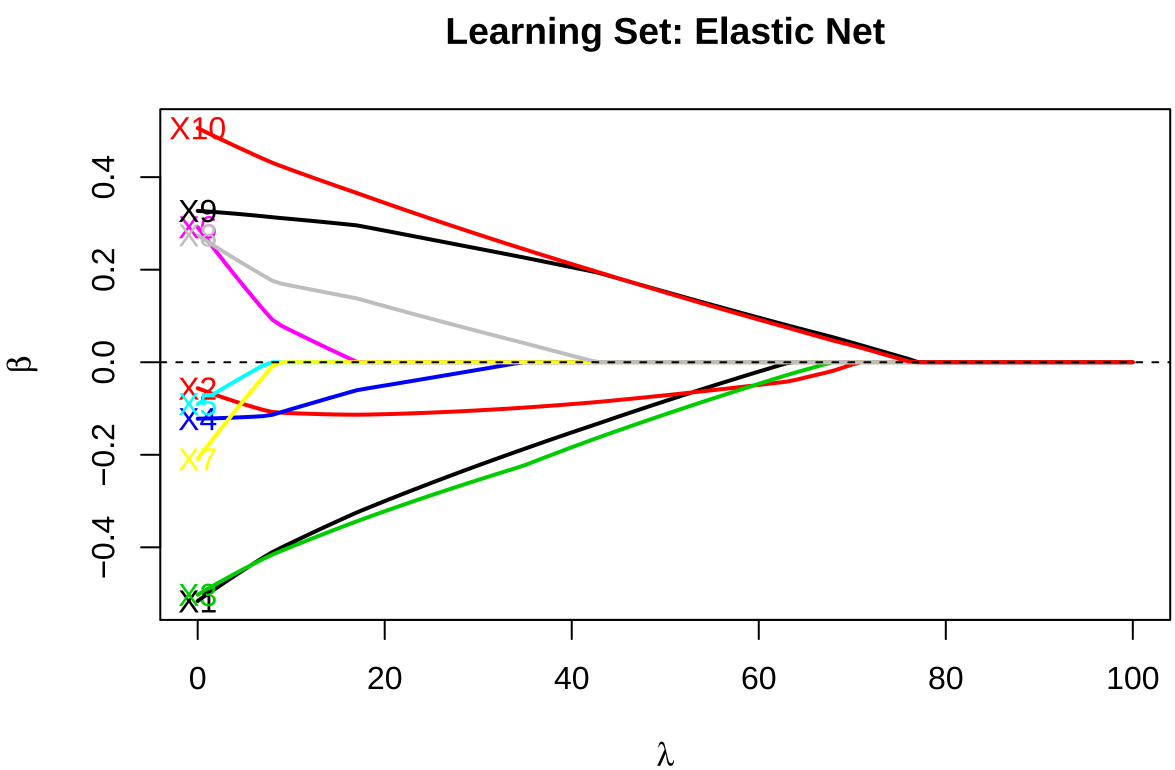

```
myPlotBeta(lambda, ridge$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
           labels=colnames(Lset[,1:10]), main="Learning Set: Ridge")
```



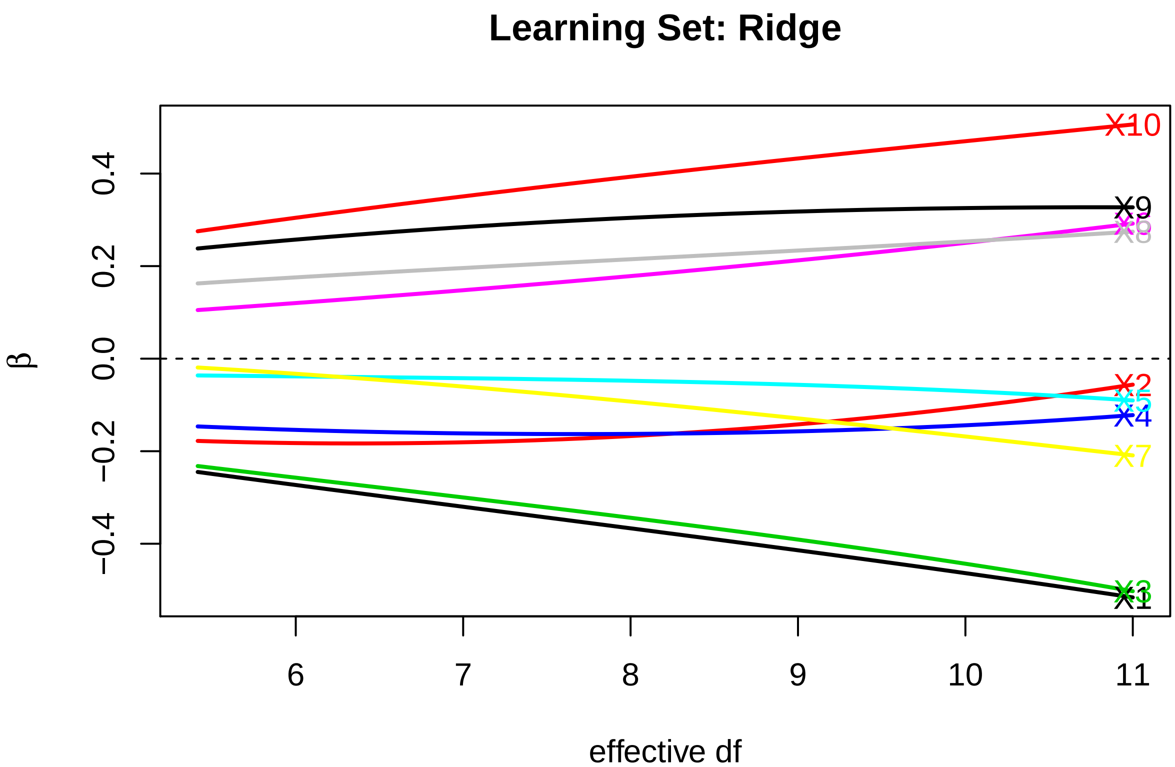

```
myPlotBeta(lambda, lasso$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
           labels=colnames(Lset[,1:10]), main="Learning Set: LASSO")
```



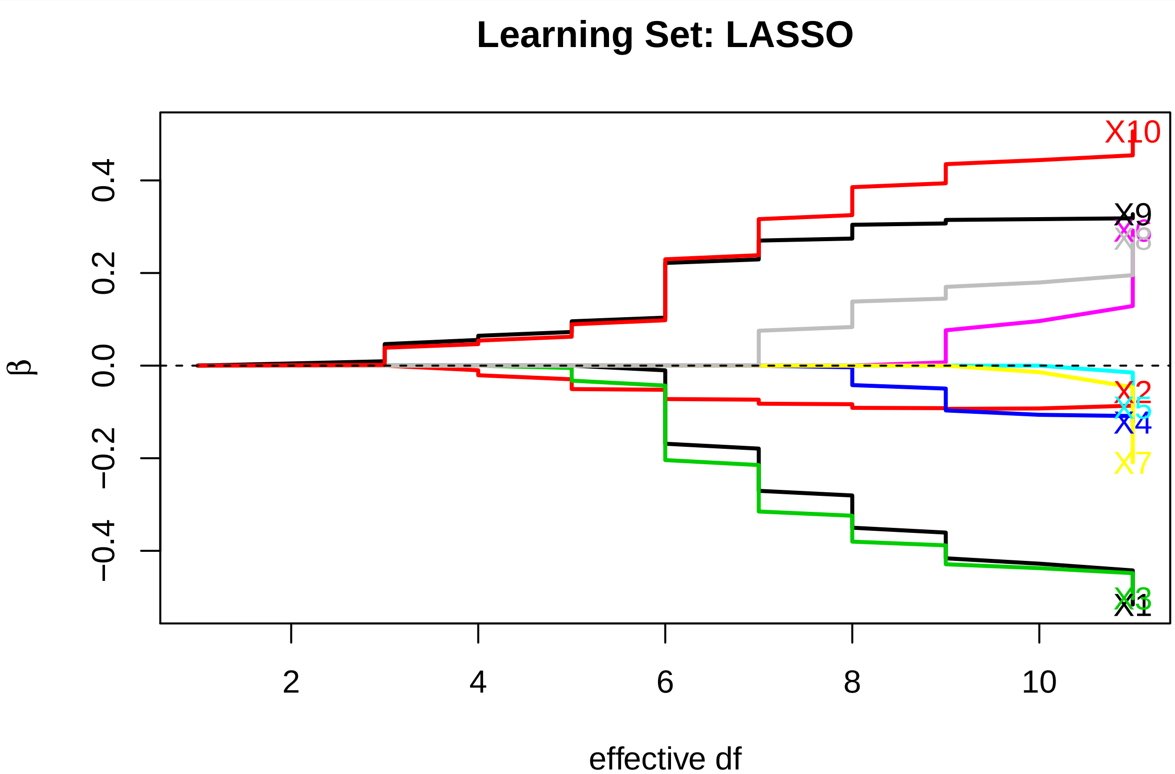

```
myPlotBeta(lambda, enet$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
          labels=colnames(Lset[,1:10]), main="Learning Set: Elastic Net")
```



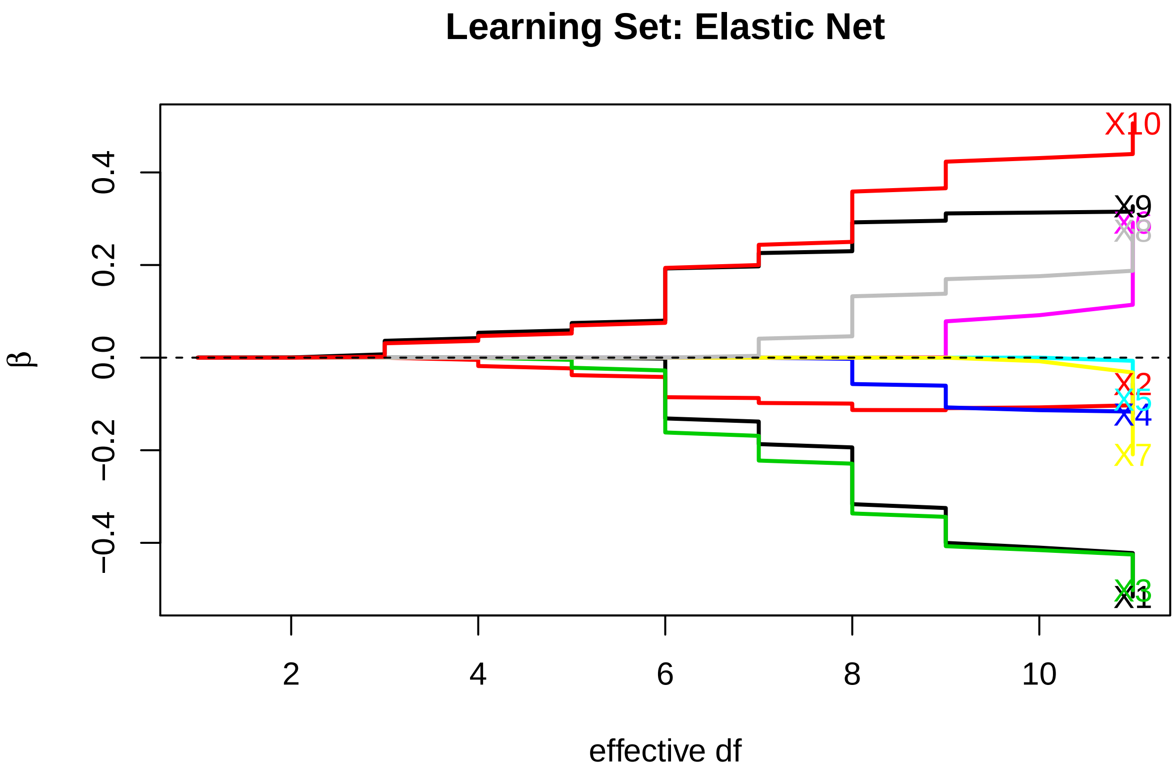

```
# Beta vs. effective df
myPlotBeta(ridge$df,ridge$beta.hat[,-1],type="l",lwd=2,xlab="effective df",
          ylab=expression(hat(beta)),labels=colnames(Lset[,1:10]),main="Learning Set: Ridge")
```




```r
myPlotBeta(lasso$df,lasso$beta.hat[,-1],type="l",lwd=2,xlab="effective df",
           ylab=expression(hat(beta)),labels=colnames(Lset[,1:10]),main="Learning Set: LASSO")
```



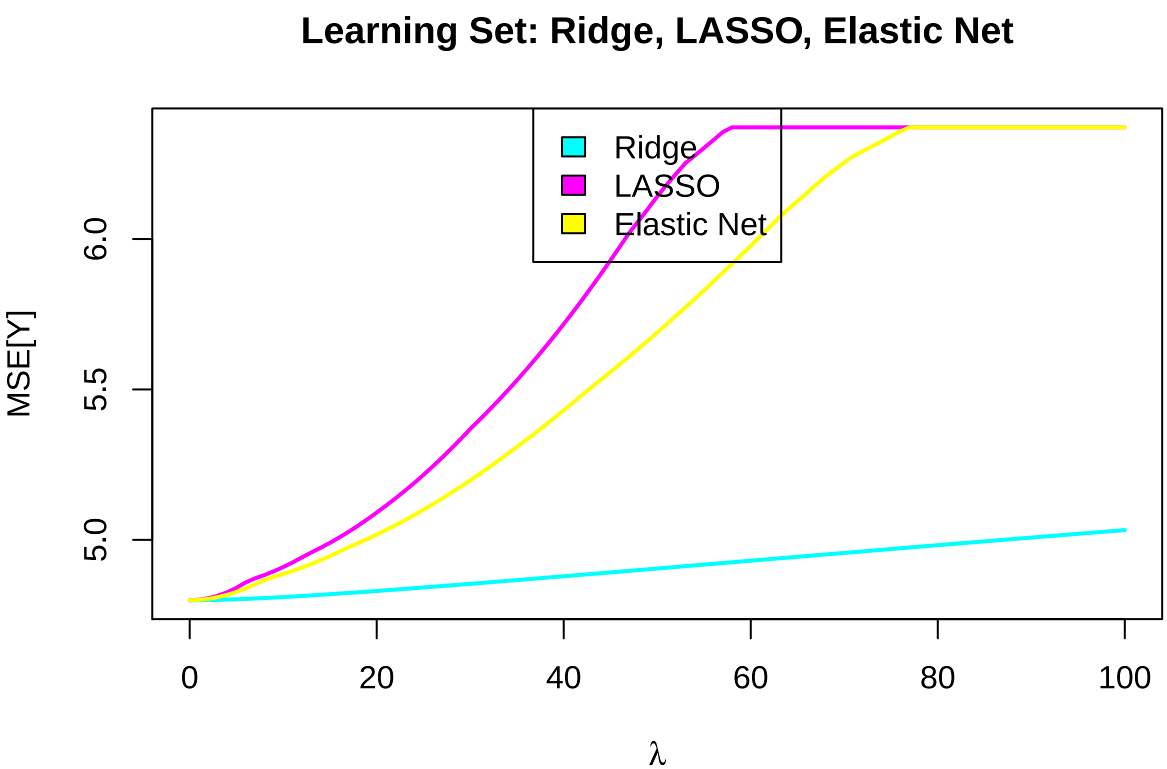
```r
myPlotBeta(enet$df,enet$beta.hat[,-1],type="l",lwd=2,xlab="effective df",
           ylab=expression(hat(beta)),labels=colnames(Lset[,1:10]),main="Learning Set: Elastic Net")
```




```r
###########################
# MSE
###########################

mse.Lset <- cbind("Ridge" = ridge$mse,"LASSO" = lasso$mse, "Elastic Net" = enet$mse)

summary(mse.Lset)
```

```
##       Ridge           LASSO         Elastic Net
##  Min.   :4.799   Min.   :4.799   Min.   :4.799
##  1st Qu.:4.842   1st Qu.:5.216   1st Qu.:5.100
##  Median :4.905   Median :6.142   Median :5.689
##  Mean   :4.907   Mean   :5.828   Mean   :5.677
##  3rd Qu.:4.969   3rd Qu.:6.372   3rd Qu.:6.342
##  Max.   :5.032   Max.   :6.372   Max.   :6.372
```

```r
matplot(lambda, mse.Lset, type="l",lwd=2,lty=1,col=5:7,
        xlab=expression(lambda), ylab="MSE[Y]",
        main="Learning Set: Ridge, LASSO, Elastic Net")
legend("top",c("Ridge","LASSO", "Elastic Net"),fill=5:7)
```



We notice an inverse relationship between the effective degrees of freedom and the shrinkage parameter across all three methods. For LASSO and elastic net, this happens in a stepwise manner since these are step-wise functions. Additionally, across all three methods, as the shrinkage parameter increases, the estimated regression coefficient shrinks towards zero. In fact, for large enough shrinkage parameters, the regression coefficients are set to zero. We see this occuring for the LASSO and elastic net estimators. We also note that as the effective degrees of freedom increase the estimated regression coefficient blows up, as expected. Lastly and as expected, the mean squared error (MSE) of the fitted values of the learning set increase as the shrinkage parameter increases, corresponding to the estimators become less data-adaptive. We also see that the MSE is minimized for all three types of estimators when the shrinkage parameter is set to zero.

c) Performance assessment on test set.

For each estimator in b), obtain the test set risk $MSE(\hat{\beta}_n; \mathcal{T}_{n_{TS}})$ for the squared error loss function (i.e., MSE). Provide and comment on plots of risk versus the shrinkage parameter and report which values of the shrinkage parameter minimize risk. Examine the corresponding three "optimal" estimators of the regression coefficients.

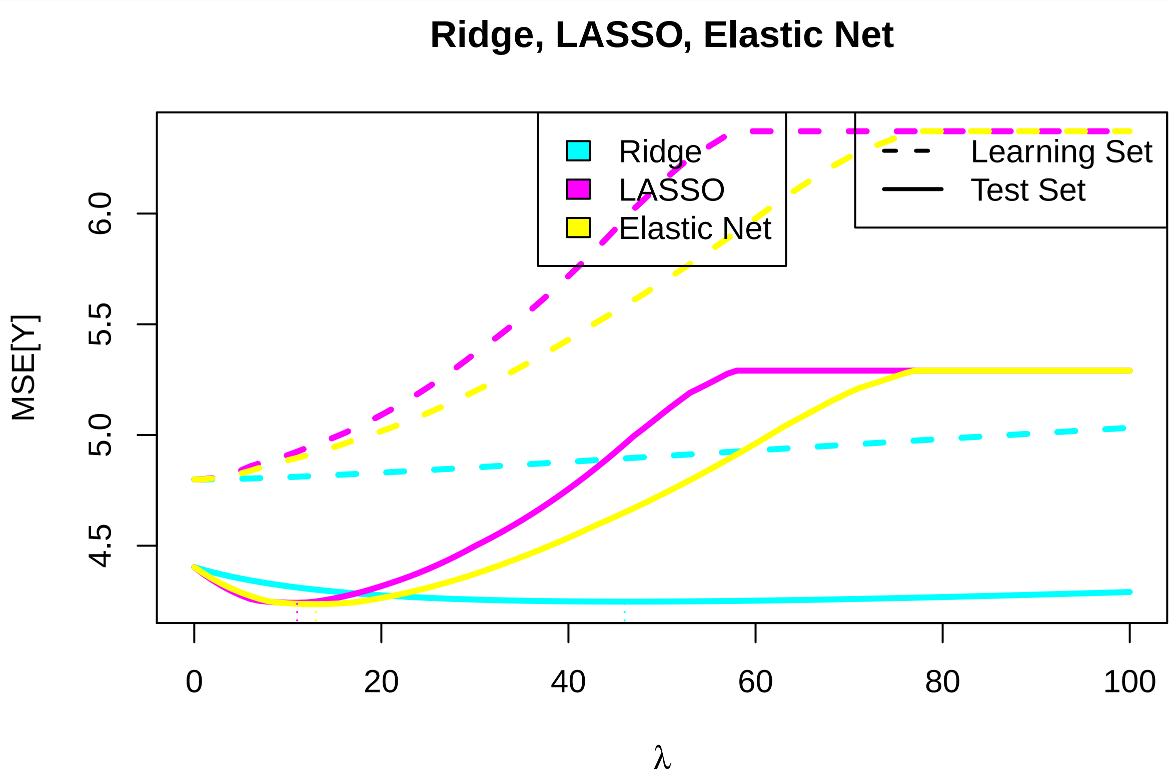
Solution:

```
############################
# MSE
############################

mse.ridge.Tset <- rowMeans(scale(ridge$y.new, center=Tset[,11], scale=FALSE)^2)
mse.lasso.Tset <- rowMeans(scale(lasso$y.new, center=Tset[,11], scale=FALSE)^2)
mse.enet.Tset <- rowMeans(scale(enet$y.new, center=Tset[,11], scale=FALSE)^2)

l1 <- which.min(mse.ridge.Tset)
l2 <- which.min(mse.lasso.Tset)
```

```r
l3 <- which.min(mse.enet.Tset)

mse.Tset <- cbind("Ridge" = mse.ridge.Tset,
                  "LASSO" = mse.lasso.Tset,
                  "Elastic Net" = mse.enet.Tset)
summary(mse.Tset)
```

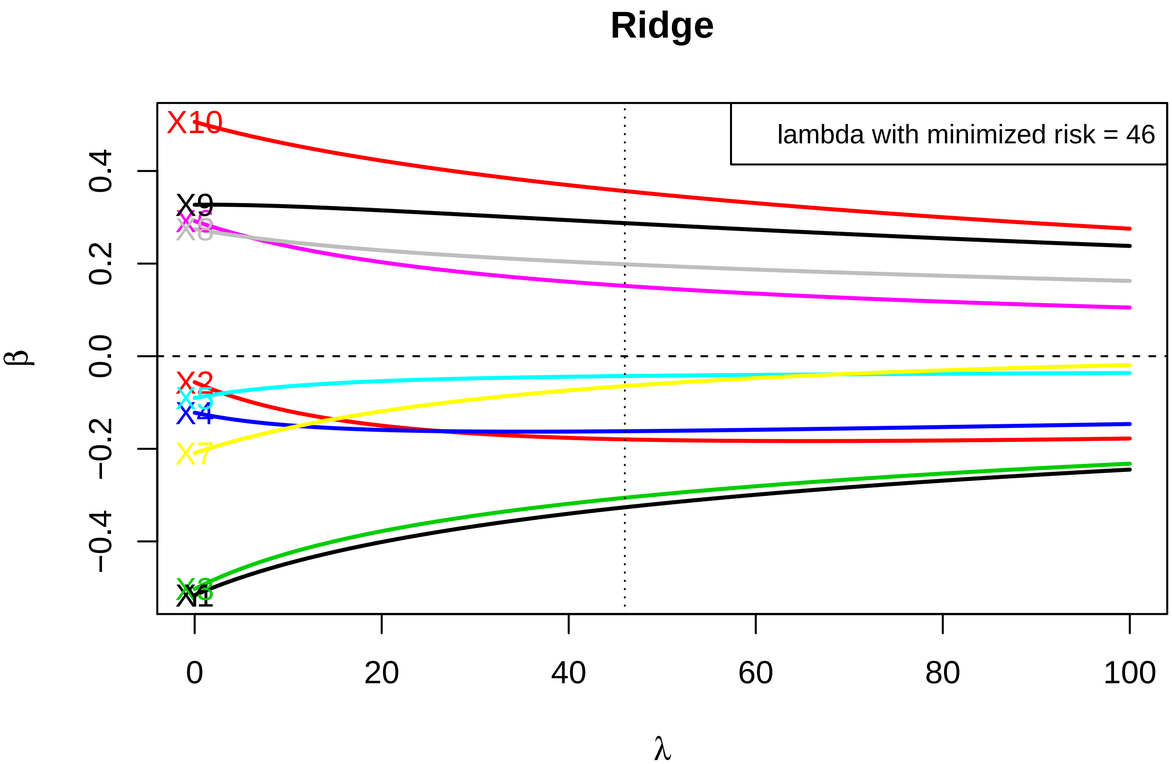
```
##      Ridge           LASSO         Elastic Net
##  Min.   :4.248   Min.   :4.242   Min.   :4.236
##  1st Qu.:4.252   1st Qu.:4.403   1st Qu.:4.346
##  Median :4.264   Median :5.095   Median :4.731
##  Mean   :4.275   Mean   :4.888   Mean   :4.771
##  3rd Qu.:4.282   3rd Qu.:5.291   3rd Qu.:5.265
##  Max.   :4.403   Max.   :5.291   Max.   :5.291
```

```r
matplot(lambda, cbind(ridge$mse, mse.ridge.Tset,
                      lasso$mse, mse.lasso.Tset,
                      enet$mse, mse.enet.Tset),
        type="l", lwd=3,col=rep(5:7, each=2), lty=rep(2:1,2),
        xlab=expression(lambda), ylab="MSE[Y]",
        main="Ridge, LASSO, Elastic Net")
legend("topright", c("Learning Set", "Test Set"), lty=2:1, lwd=2)
legend("top", c("Ridge","LASSO", "Elastic Net"), fill=5:7)
lines(lambda[rep(l1, 2)], c(0,min(mse.ridge.Tset)), col=5, lty=3)
lines(lambda[rep(l2, 2)], c(0,min(mse.lasso.Tset)), col=6, lty=3)
lines(lambda[rep(l3, 2)], c(0,min(mse.enet.Tset)), col=7, lty=3)
```

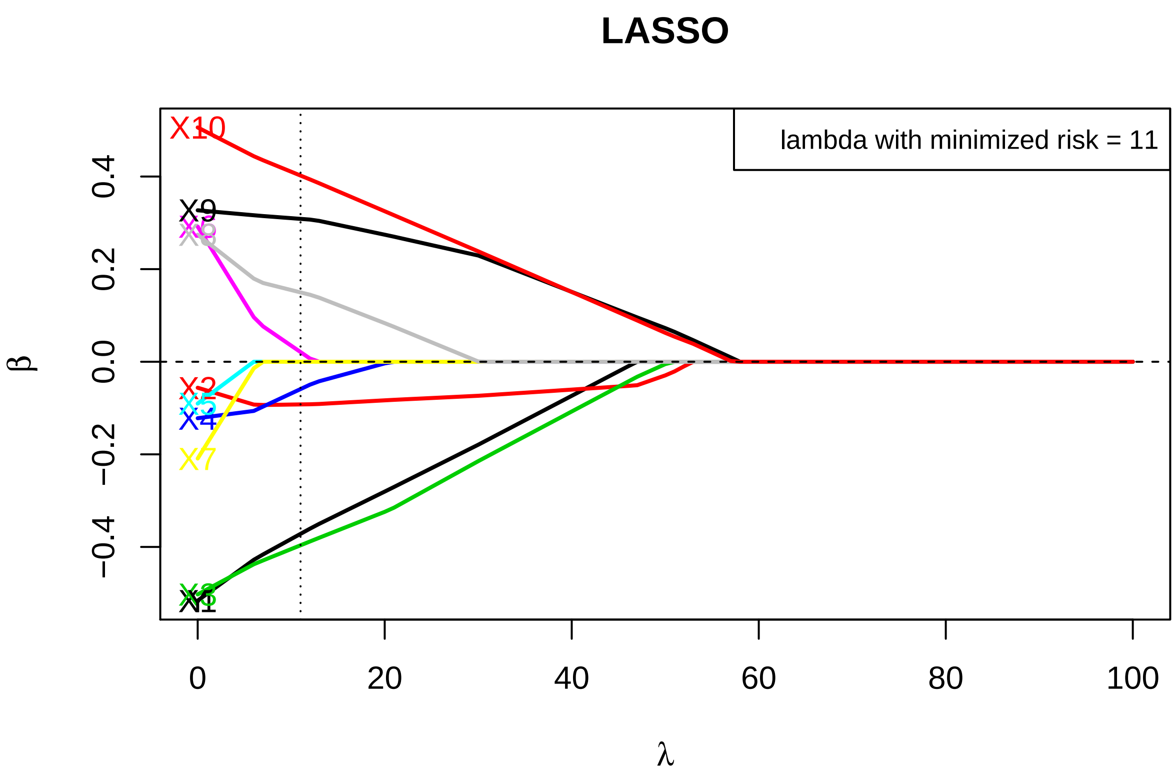



```r
############################
# Risk v Lambda
############################
```

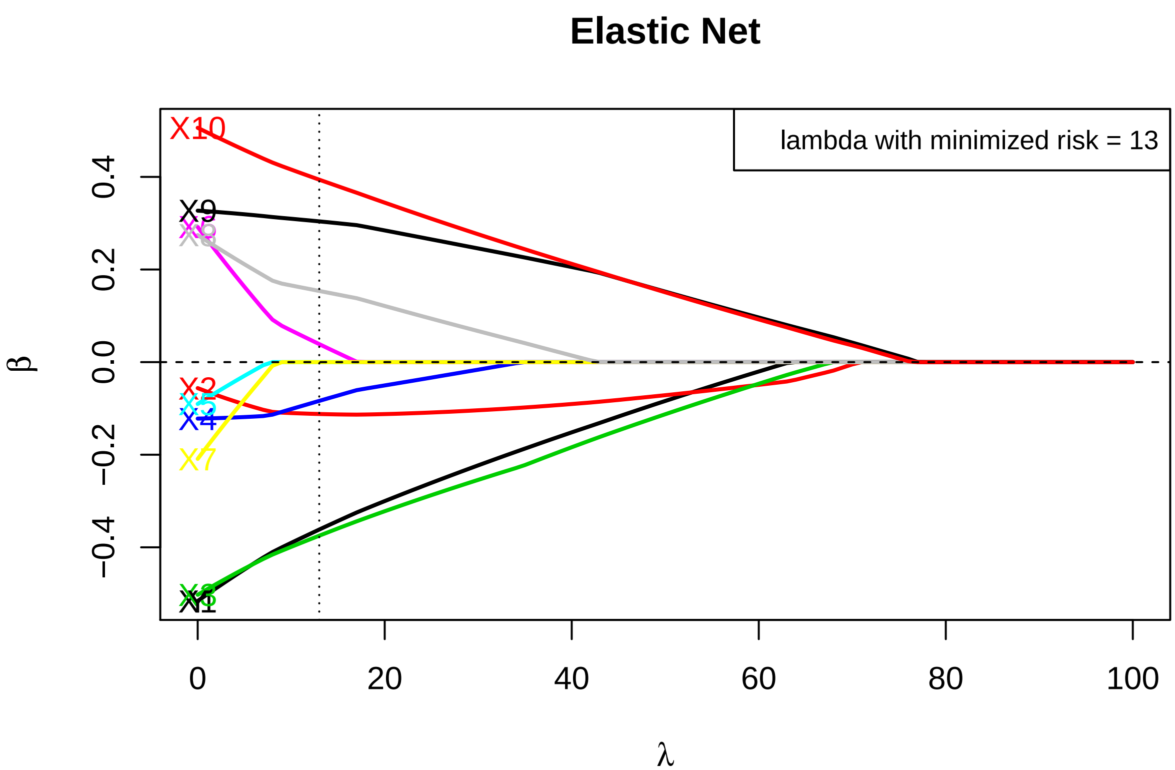
```r
myPlotBeta(lambda,ridge$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="Ridge")
abline(v = lambda[l1],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[l1], sep = "")))
```



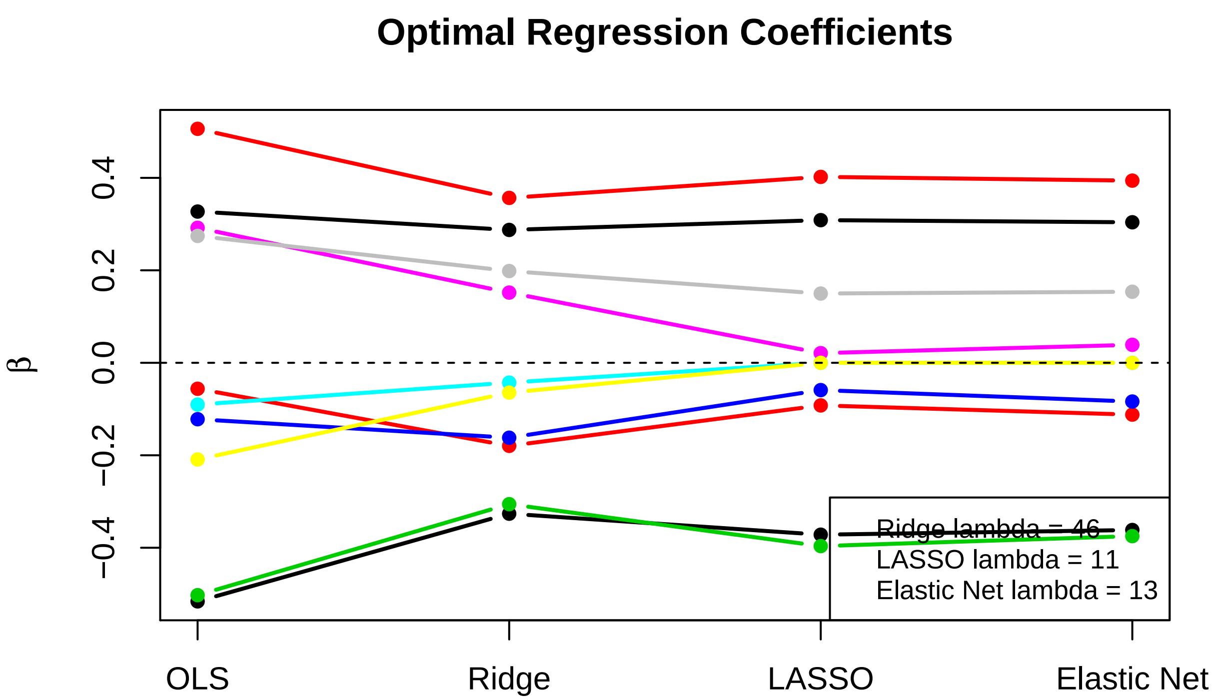

```r
myPlotBeta(lambda, lasso$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="LASSO")
abline(v = lambda[l2],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[l2], sep = "")))
```



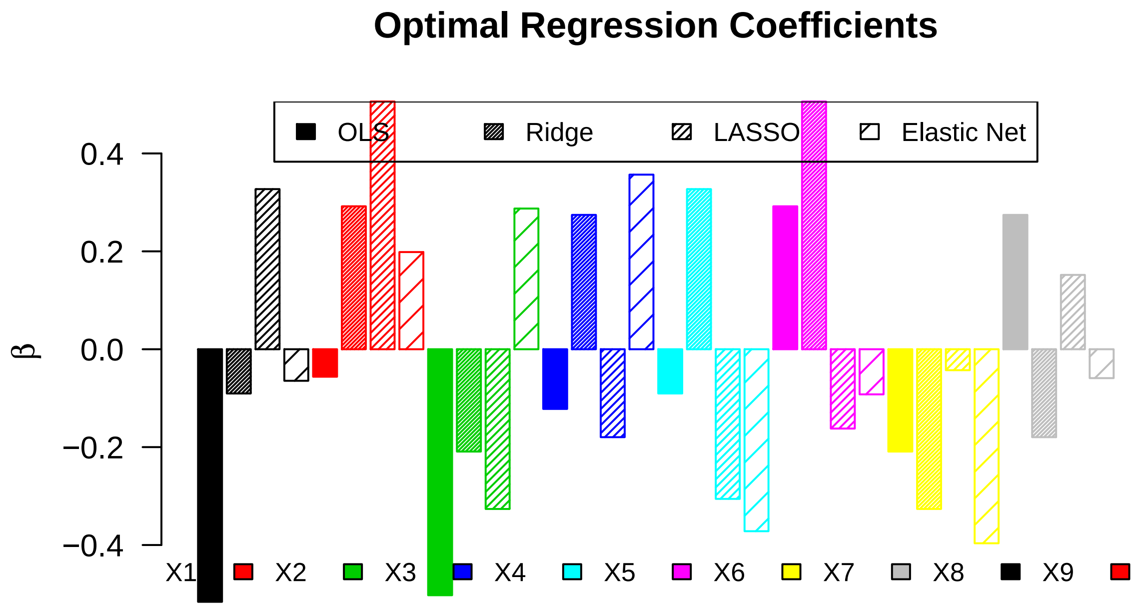

```r
myPlotBeta(lambda, enet$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="Elastic Net")
abline(v = lambda[13],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[13], sep = "")))
```



```r
############################
# Optimal Beta
############################

matplot(t(cbind(ridge$beta.hat[1,-1], ridge$beta.hat[l1,-1],
                lasso$beta.hat[l2,-1], enet$beta.hat[l3,-1])),
        type="b", lty=1, lwd=2, pch=16, col=1:ncol(X_T),
        ylab=expression(hat(beta)), axes=FALSE,
        main="Optimal Regression Coefficients")
box()
axis(1, at=1:4, c("OLS", "Ridge", "LASSO", "Elastic Net"))
axis(2)
abline(h=0, lty=2)
legend("bottomright", pt.cex = 1, cex=.8,
       c(paste("Ridge lambda = ", lambda[l1], sep = ""),
         paste("LASSO lambda = ", lambda[l2], sep = ""),
         paste("Elastic Net lambda = ", lambda[l3], sep = "")))
```




```r
barplot(c(ridge$beta.hat[1,-1],ridge$beta.hat[l1,-1],lasso$beta.hat[l2,-1],
          enet$beta.hat[l3,-1])[as.vector(sapply(1:8,function(x)c(x,x+4,x+8,x+16)))],
       col=rep(1:ncol(X_T),each=4),density=rep(c(-9,66,33,10),ncol(X_T)),
       border=TRUE, ylab=expression(hat(beta)), names.arg="",
       las=2, main="Optimal Regression Coefficients")
legend("bottom", colnames(Lset)[1:10], fill=1:ncol(X_T),
       pt.cex = 1, cex=.8, bty = "n", horiz = TRUE)
legend("top", density=c(-9,66,33,10),
       c("OLS", "Ridge", "LASSO", "Elastic Net"), pt.cex = 1, cex=.8, horiz = TRUE)
```




```r
df <- data.frame("OLS" = c(ridge$beta.hat[1,-1]), "Ridge" = c(ridge$beta.hat[11,-1]),
           "LASSO" = c(ridge$beta.hat[12,-1]),
           "Elastic Net" = c(ridge$beta.hat[12,-1]))
rownames(df) <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                  "X8", "X9", "X10")
df
```

```
##             OLS       Ridge      LASSO Elastic.Net
## X1  -0.51600124 -0.32634367 -0.4419509  -0.4419509
## X2  -0.05613818 -0.17963570 -0.1224965  -0.1224965
## X3  -0.50255491 -0.30575592 -0.4199443  -0.4199443
## X4  -0.12196832 -0.16202351 -0.1506623  -0.1506623
## X5  -0.09035874 -0.04284147 -0.0635732  -0.0635732
## X6   0.29197504  0.15184663  0.2334610   0.2334610
## X7  -0.20904487 -0.06448791 -0.1510744  -0.1510744
## X8   0.27443703  0.19851948  0.2447056   0.2447056
## X9   0.32710005  0.28739904  0.3229681   0.3229681
## X10  0.50603746  0.35666592  0.4540948   0.4540948
```

The plots of the risk versus the shrinkage parameter show us that the risk is minimized for smaller values of the shrinkage parameter for the LASSO and Elastic Net regression estimators in comparison to the Ridge regression estimator. We examine the corresponding "optimal" estimators of the regression coefficients that we constructed as well as OLS with plots and a table. These visuals show us that the optimal estimators of the regression coefficients across all of the covariates are most similar for the Ridge, LASSO, and Elastic Net regression estimators and differ widely from the optimal OLS estimators of the regression coefficients across all of the covariates.

d) Ridge regression: Bias, variance, and mean squared error of estimated regression coefficients.

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.

Solution:

According to equation (21) on the 'Regularized Regression' lecture slides, $E[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top E[\mathbf{Y}_n|\mathbf{X}_n] = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n \beta$. The bias is as follows, $\text{Bias}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = E[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] - \beta = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n \beta - \beta = \Big((\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n - 1\Big)\beta$.

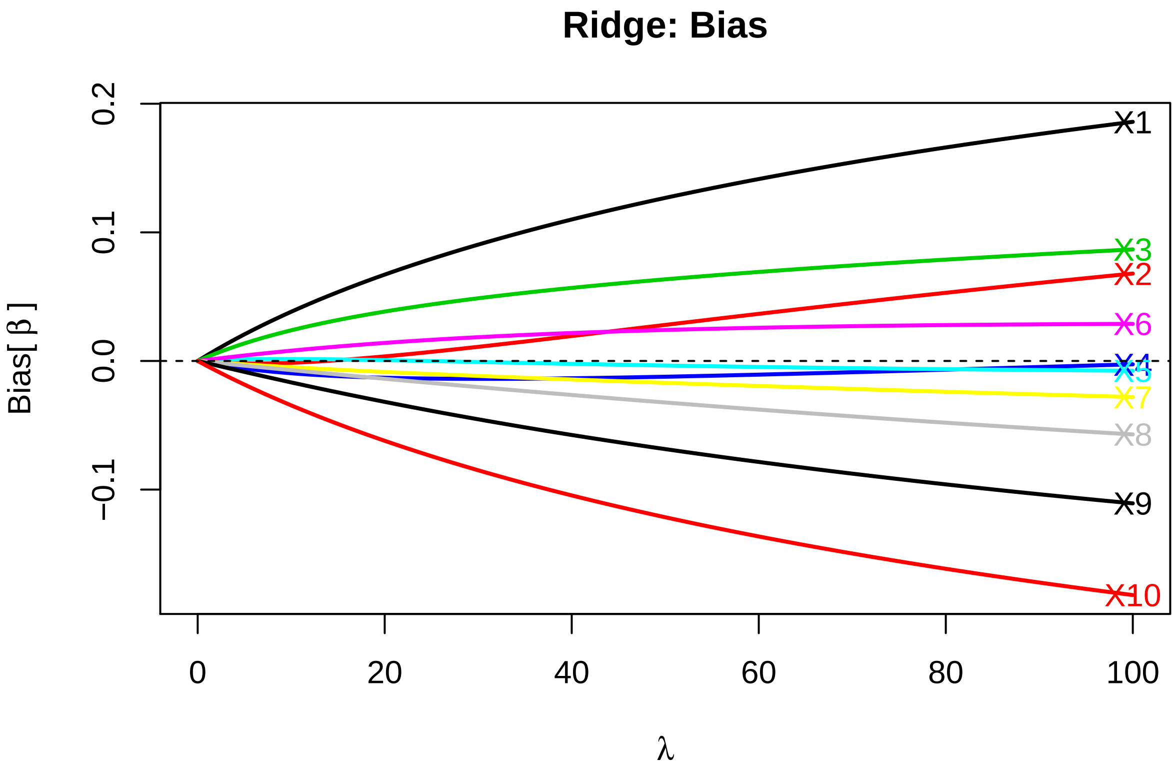
According to equation (22) on the 'Regularized Regression' lecture slides, the covariance matrix of the ridge regression estimator, $\text{Cov}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = \sigma^2(\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n(\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}$.

Thus, for the parameter $\beta = (\beta_j : j = 1, ..., J) \in \mathbf{R}^J$, a J-dimensional column vector of regression coefficients we have a J-dimensional vector of mean squared errors for each β_j is $\text{MSE}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = \text{Var}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] + \big(\text{Bias}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n]\big)^2$.

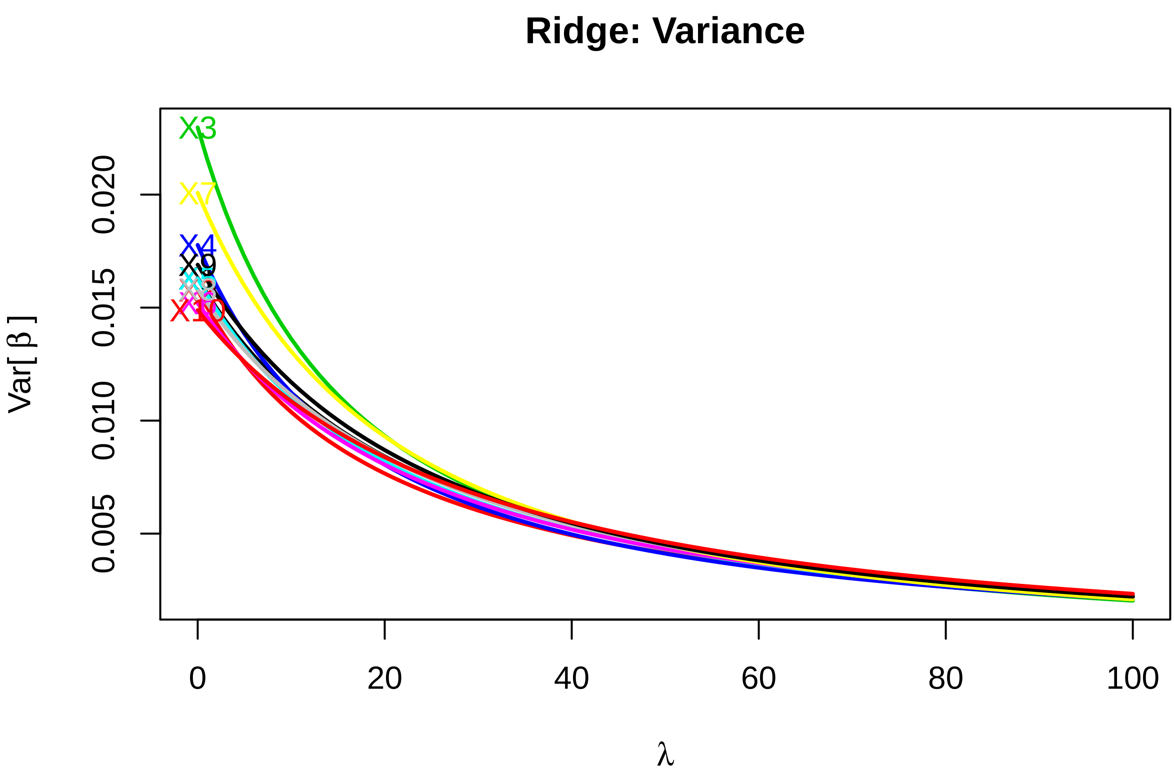
According to the slides, and we can see here, that the ridge estimator is biased. As the shrinkage parameter increases, the bias tends to increase while variance tends to decrease. This is because we become more data-adaptive and less smooth as we increase the shrinkage parameter, highlighting the bias-variance trade-off of the ridge regression estimator. It should be noted that we assume the model in Equation (1) and the bias and covariance matrices of the ridge regression estimator are conditional on the design matrix of the learning set.

```
###########################
# Setting up
###########################

# Directly from Sandrine's 'Regularized Regression:Example' code

## Ridge regression: Bias, variance, and MSE
## N.B. Do not fit intercept.

myRidgePerf <- function(x,y,beta=0,sigma=1,scale=FALSE,lambda=0)
  {
    n <- nrow(x)
    J <- ncol(x)
    df <- rep(NA,length(lambda))
    beta.hat <- bias <- var <- mse <- matrix(NA,length(lambda),J)
    cov <- array(NA,c(length(lambda),J,J))

    xx <- scale(x,center=TRUE,scale=scale)

    for(l in 1:length(lambda))
      {
        a <- solve(crossprod(xx)+lambda[l]*diag(J))
        df[l] <- sum(diag(xx%*%a%*%t(xx)))
        beta.hat[l,] <- a%*%crossprod(xx,y)
        bias[l,] <- a%*%t(xx)%*%x%*%beta - beta
        cov[l,,] <- sigma^2*a%*%crossprod(xx)%*%a
        var[l,]  <- diag(cov[l,,])
        mse[l,] <- var[l,] + bias[l,]^2
      }

    res <- list(df=df,beta.hat=beta.hat,bias=bias,cov=cov,var=var,mse=mse)
    res
  }

###########################
```

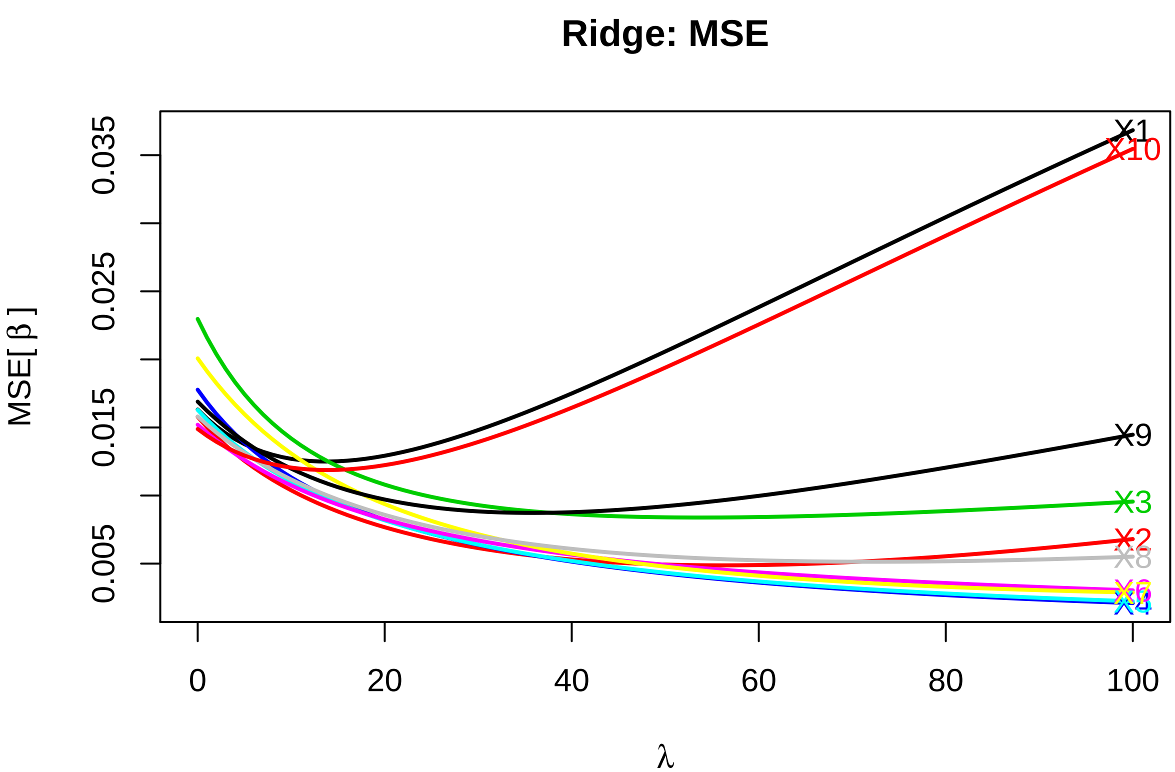
```r
# Plots
############################

ridge_perf <- myRidgePerf(x=Lset_X, y=Lset_Y, beta=Beta, sigma=gamma, lambda=lambda)

# Bias

myPlotBeta(lambda, ridge_perf$bias, labels=colnames(Lset[,1:10]), right=TRUE,
           ylab=expression("Bias["~ hat(beta) ~ "]"),
           main = "Ridge: Bias")
```



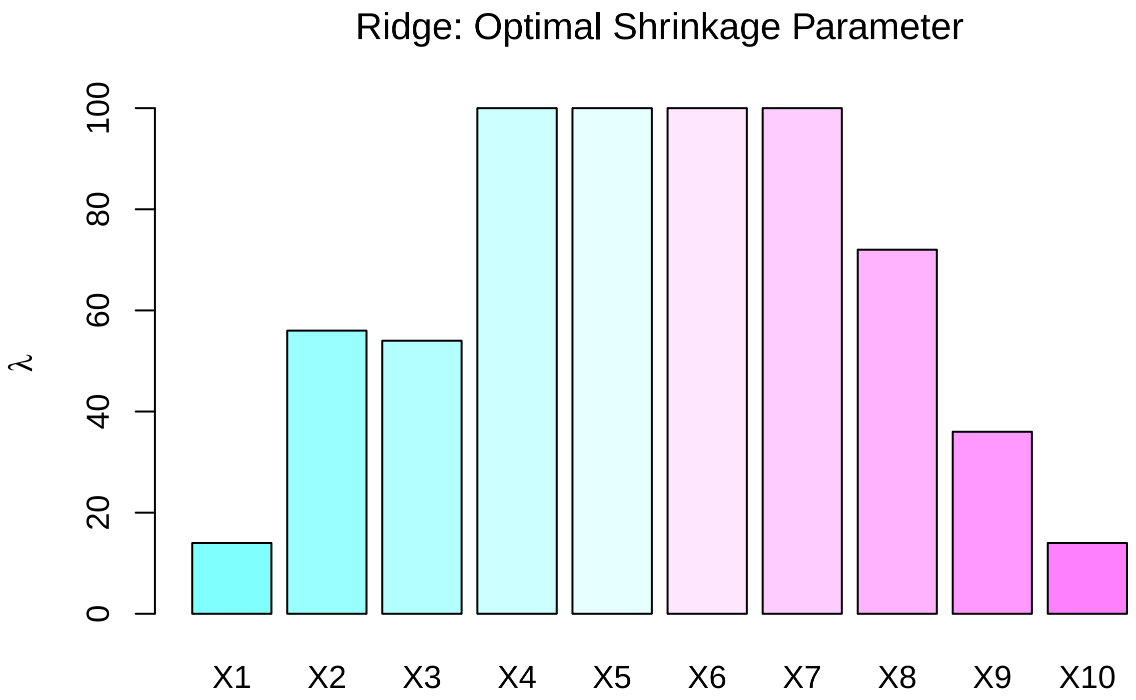

```r
# Variance

myPlotBeta(lambda, ridge_perf$var, labels=colnames(Lset[,1:10]), right=FALSE,
           ylab=expression("Var["~ hat(beta) ~ "]"),
           main = "Ridge: Variance")
```



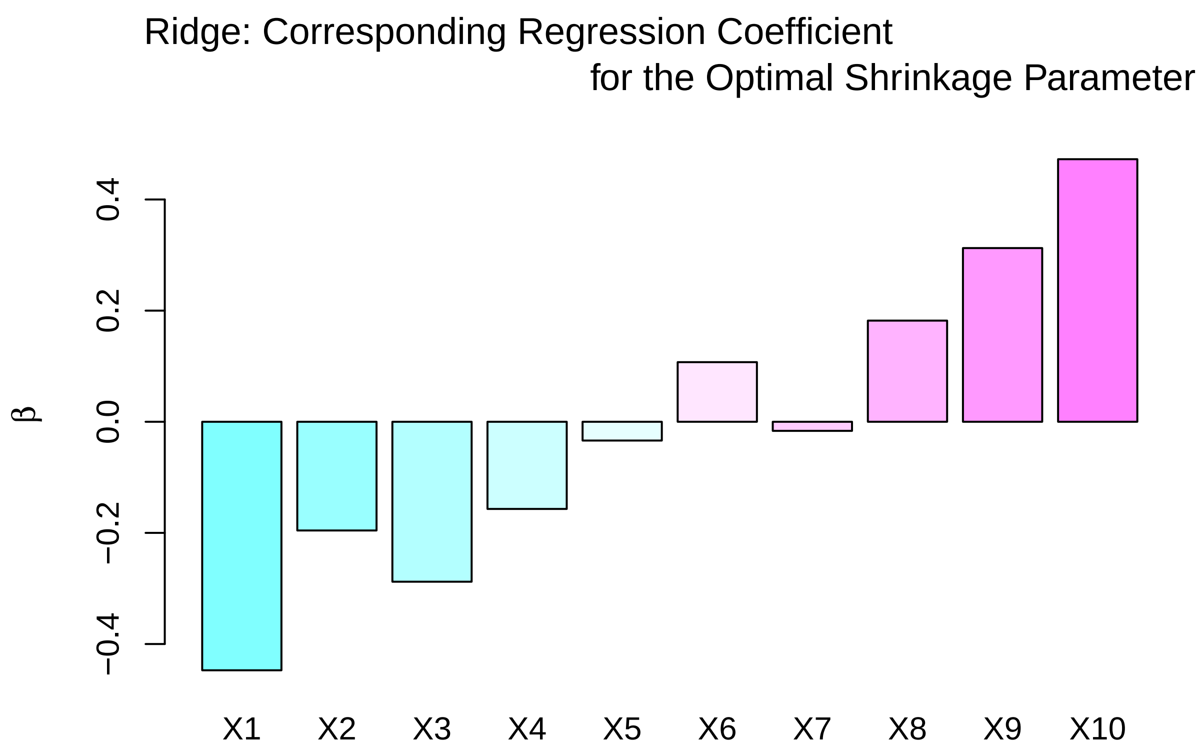

```r
# MSE

myPlotBeta(lambda, ridge_perf$mse, labels=colnames(Lset[,1:10]), right=TRUE,
           ylab=expression("MSE["~ hat(beta) ~ "]"),
           main = "Ridge: MSE")
```



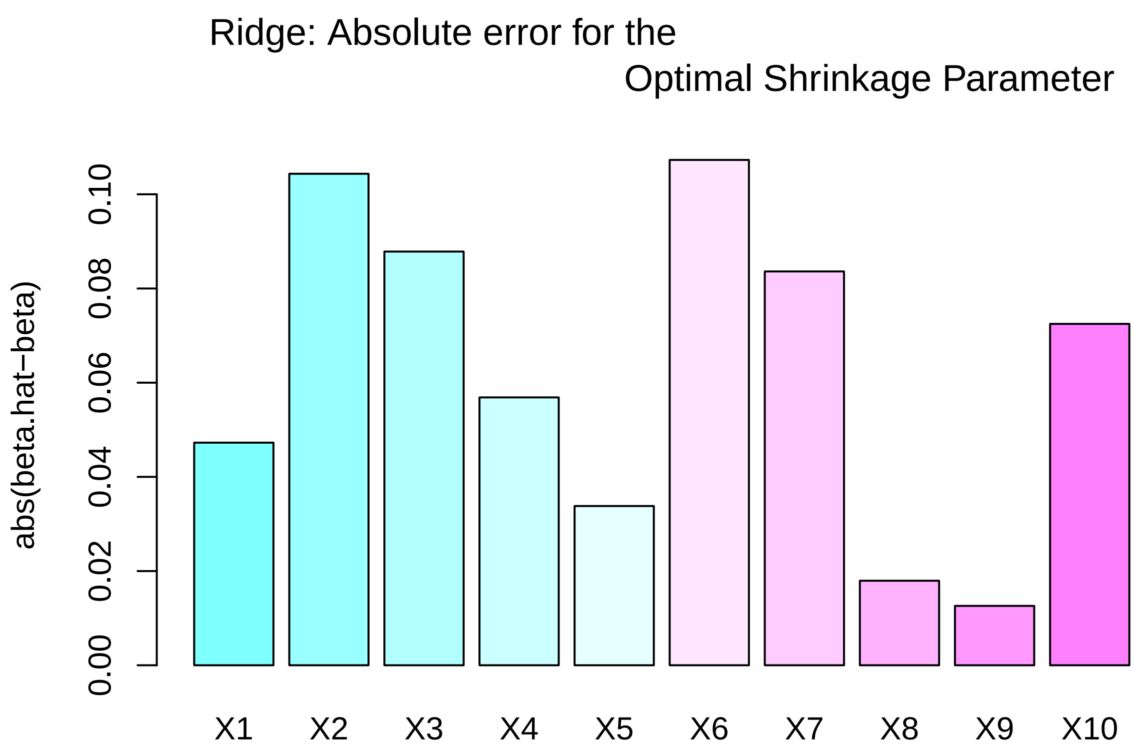
```r
############################
# Optimal lambda with beta
############################

index <- c()
mse_optimal <- c()
beta_optimal <- c()
lambda_optimal <- c()
abs_error <- c()

for(j in 1:J){
  index <- c(index, which.min(ridge_perf$mse[,j]))
}
for(j in 1:J){
  mse_optimal <- c(mse_optimal, ridge_perf$mse[index[j],j])
  lambda_optimal <- c(lambda_optimal, lambda[index[j]])
  beta_optimal <- c(beta_optimal, ridge_perf$beta.hat[index[j],j])
  abs_error <- c(abs_error, abs(abs(beta_optimal[j]) - abs(Beta[j])))
}

df <- data.frame(mse_optimal, lambda_optimal, beta_optimal, beta=Beta, abs_error)
rownames(df) <- c("X1","X2","X3","X4","X5",
                  "X6","X7","X8","X9","X10")

# plot of optimal lambda

barplot(df$lambda_optimal, main = expression("Ridge: Optimal Shrinkage Parameter"), ylab = expression(la
```




```r
barplot(df$beta_optimal, main = expression("Ridge: Corresponding Regression Coefficient
                                           for the Optimal Shrinkage Parameter"),
        ylab = expression(hat(beta)), col=cm.colors(10), names.arg = rownames(df))
```




```r
barplot(df$abs_error, main = expression("Ridge: Absolute error for the
                                        Optimal Shrinkage Parameter"),
        ylab = "abs(beta.hat-beta)", col=cm.colors(10), names.arg = rownames(df))
```




```r
df
```

```
##     mse_optimal lambda_optimal beta_optimal beta   abs_error
## X1  0.012497875             14  -0.44726409 -0.4 0.04726409
```

```
## X2  0.004873294              56  -0.19564693 -0.3 0.10435307
## X3  0.008384466              54  -0.28783932 -0.2 0.08783932
## X4  0.002099583             100  -0.15687336 -0.1 0.05687336
## X5  0.002233798             100  -0.03381018  0.0 0.03381018
## X6  0.003036670             100   0.10729402  0.0 0.10729402
## X7  0.002875275             100  -0.01637967  0.1 0.08362033
## X8  0.005136162              72   0.18204989  0.2 0.01795011
## X9  0.008725148              36   0.31260797  0.3 0.01260797
## X10 0.011872934              14   0.47248325  0.4 0.07248325
```

The graphs nicely display the bias-variance trade-off mentioned in the beginning of this solution. We see that as we increase the shrinkage parameter the bias increases and the variance decreases. The MSE plot shows us that there surely exist optimal values of the shrinkage parameter; as we increase the shrinkage parameter the MSE decrease a bit across all covariates and then it increases if the shrinkage parameter increases too much. We find the values of the optimal shrinkage parameter (those that minimize the MSE) with the corresponding regression coefficient estimate and compare this to the true regression coefficients in the last plot. There is a noticable amount of variablility for the absolute error (absolute difference from the estimate to the truth) across the covariates.

e) LASSO regression: Bias, variance, and mean squared error of estimated regression coefficients.

For the LASSO, there are no closed-form expressions for the bias, variance, and mean squared error of the estimators of the regression coefficients.

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.

Solution:

Since the lasso estimate is a non-linear and non-differentiable function of the response values even for a fixed value of the shrinkage parameter, it is not straightforward to obtain accurate estimates for the bias, variance, and mean squared error of the regression coefficients. Tibshirani, 2006 suggests the bootstrap. This is a resampling method that mimics the availability of several datasets by resampling from the same unique dataset. Here is the general procedure:

1. Select a random sample (of size n), with replacement, from the observations in the original sample. This is called a bootstrap sample.
2. Perform the original regression procedure on the bootstrap sample, and obtain the estimate of interest.
3. Repeat the sampling with replacement a large number (B) of times, and for each new bootstrap sample, obtain the estimate of interest, so that we have a collection of bootstrap estimates.

We want to choose $n = 100$ (corresponding to Step 1) so we generate a bootstrapped sample that is the same size as our learning set and has the same distribution as the learning set. Next, we perform LASSO regression to estimate the coefficients (Step 2). We perform this $B = 10000$ times (Step 3). So, we obtain 10,000 vectors of LASSO estimated regression coefficients (i.e. a collection of bootstrap estimates of the LASSO regression coefficients).

To estimate bias, covariance, and MSE we need to estimate the conditional mean, $E[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n]$. The bootstrap allows us to do this empirically by estimating the conditional mean as the average the B bootstrap estimates of the LASSO regression coefficients, yielding smoothed LASSO estimates of the regression coefficients. This method is clearly explained and suggested by Efron in 2014 in the article *Estimation and Accuracy after Model Selection*.

Now we can express the estimates of the bias, covariance, and MSE as a function of these smoothed coefficients.

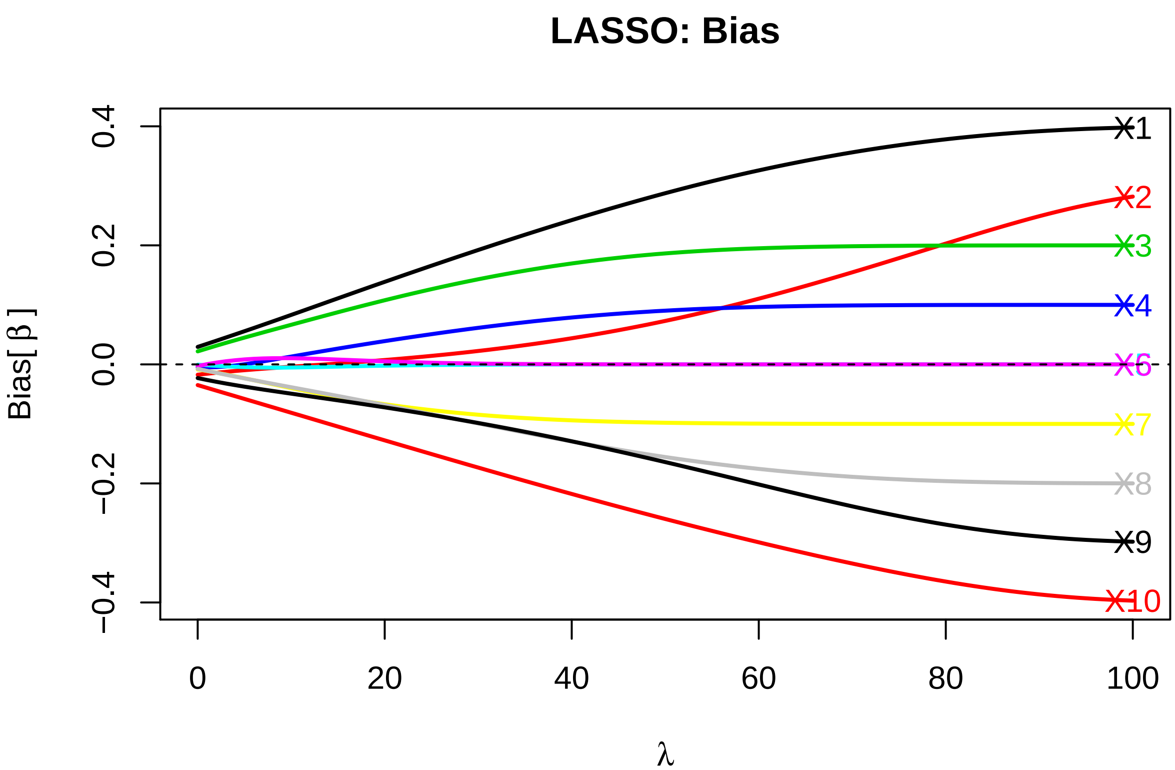
$\hat{\text{Bias}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n] = E[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n] - \beta = \frac{1}{B}\sum_{b=1}^{B}\hat{\beta}_{n,b}^{\text{LASSO}} - \beta = \bar{\hat{\beta}}_{n,b}^{\text{LASSO}} - \beta$

$\hat{\text{Cov}}_{\text{unbiased}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n] = \frac{1}{B-1}\sum_{b=1}^{B}(\hat{\beta}_{n,b}^{\text{LASSO}} - \bar{\hat{\beta}}_n^{\text{LASSO}})(\hat{\beta}_{n,b}^{\text{LASSO}} - \bar{\hat{\beta}}_n^{\text{LASSO}})^\top$.

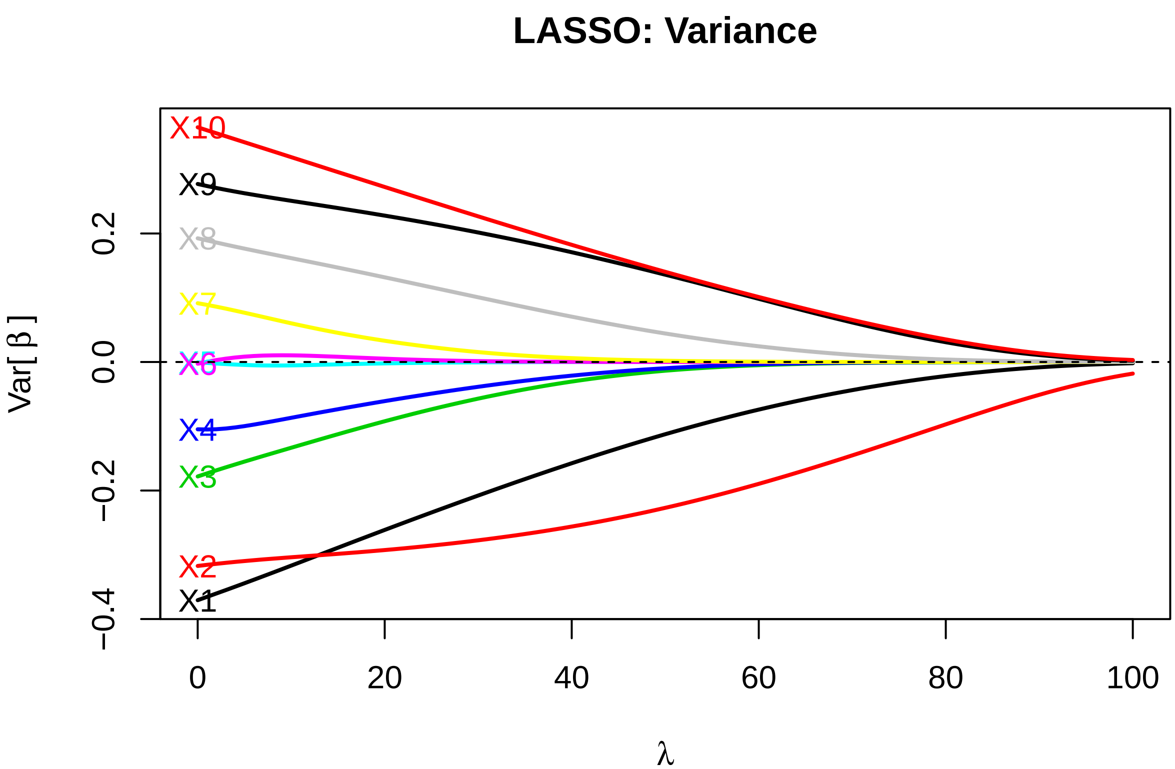
$\hat{\text{MSE}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n] = \hat{\text{Var}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n] + \left(\hat{\text{Bias}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n]\right)^2$.

Regarding the assumption and the variables we condition on, we have the same case as in part D. That is, the bias and covariance matrices of the bootstrapped LASSO regression estimators are conditional on the design matrix of the learning set and assume the model from Equation (1).

```r
# Thank you, Kelly for your help on this question!

B <- 10000
beta_sim <- array(data = NA, dim = c(B, length(lambda), J))

for(b in 1:B){
  n <- 100
  Y_sim <- X_L %*% Beta + rnorm(n, sigma)
  lasso_sim <- myGlmnet(x = X_L[(1:100),], y = Y_sim,
                        alpha = 1, lambda = lambda/(2*n))
  beta_sim[b,,] <- lasso_sim$beta.hat[,-1]
}

bias_hat <- scale(apply(beta_sim, 2:3, mean),
                  center = Beta, scale = FALSE)
var_hat <- apply(beta_sim, 2:3, mean)
mse_hat <- var_hat + (bias_hat)^2

###########################
# Plots
###########################

myPlotBeta(lambda, bias_hat, labels=colnames(Lset[,1:10]),
           right=TRUE, ylab=expression("Bias["~ hat(beta) ~ "]"),
           main = "LASSO: Bias")
```

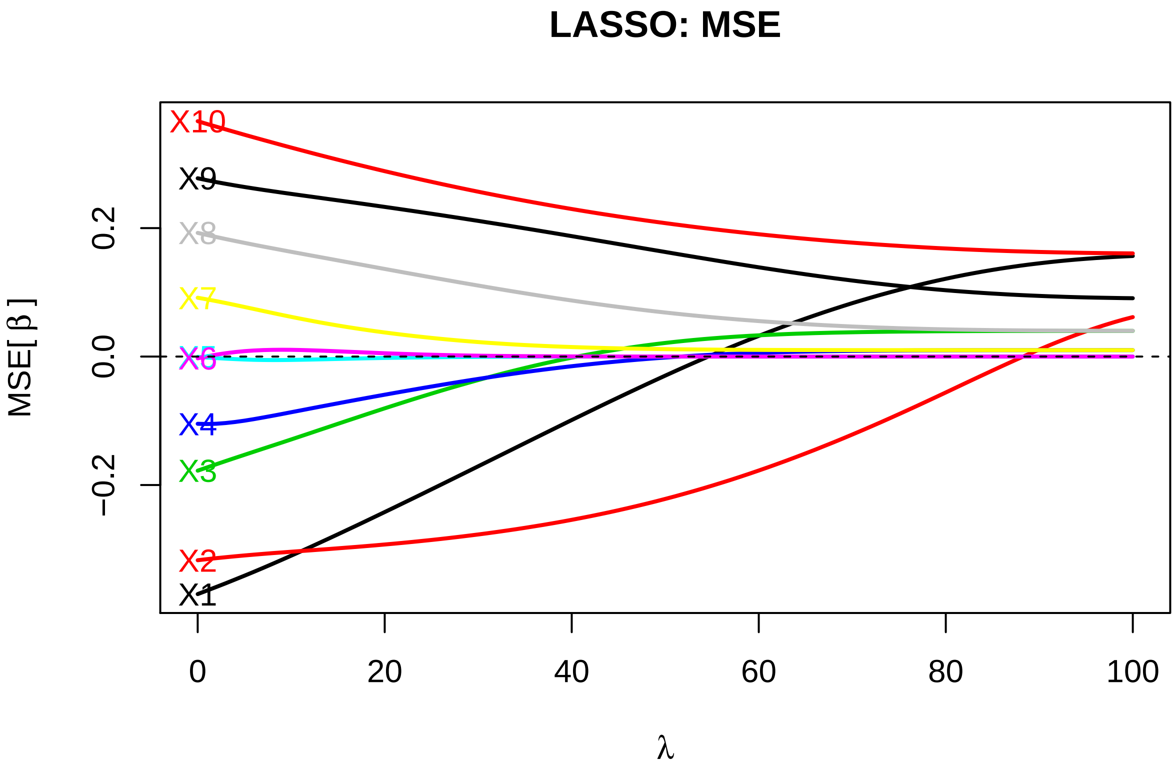



```
myPlotBeta(lambda, var_hat, labels=colnames(Lset[,1:10]),
           right=FALSE, ylab=expression("Var["~ hat(beta) ~ "]"),
           main = "LASSO: Variance")
```

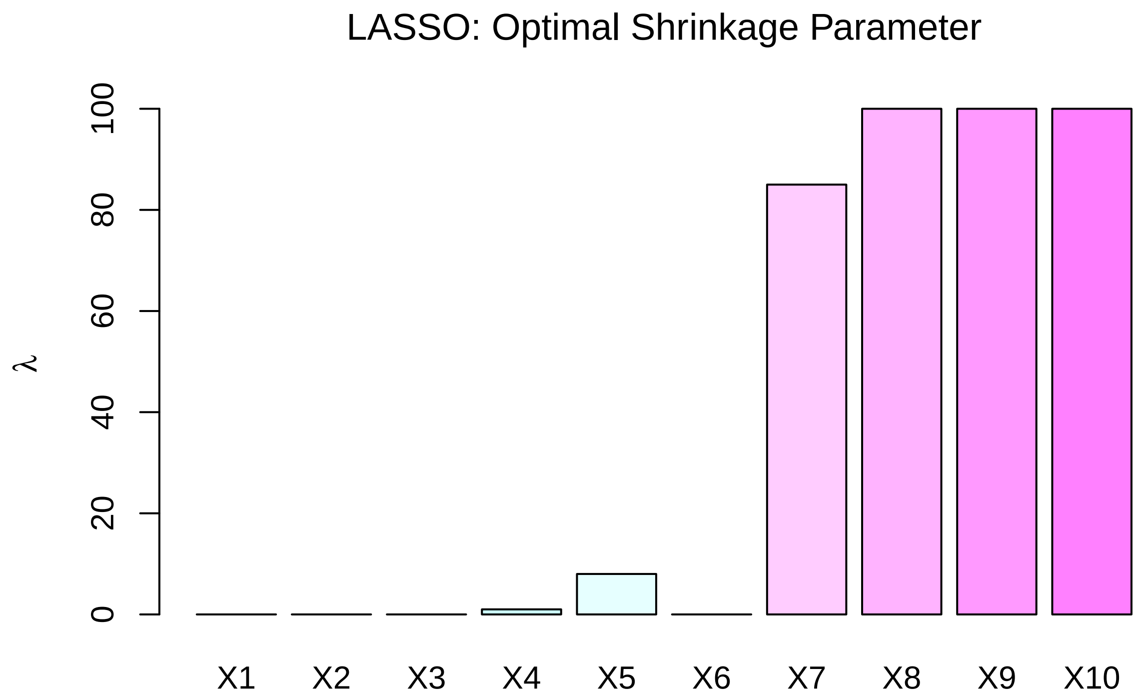



```
myPlotBeta(lambda, mse_hat, labels=colnames(Lset[,1:10]),
           right=FALSE, ylab=expression("MSE["~ hat(beta) ~ "]"),
```

```
          main = "LASSO: MSE")
```

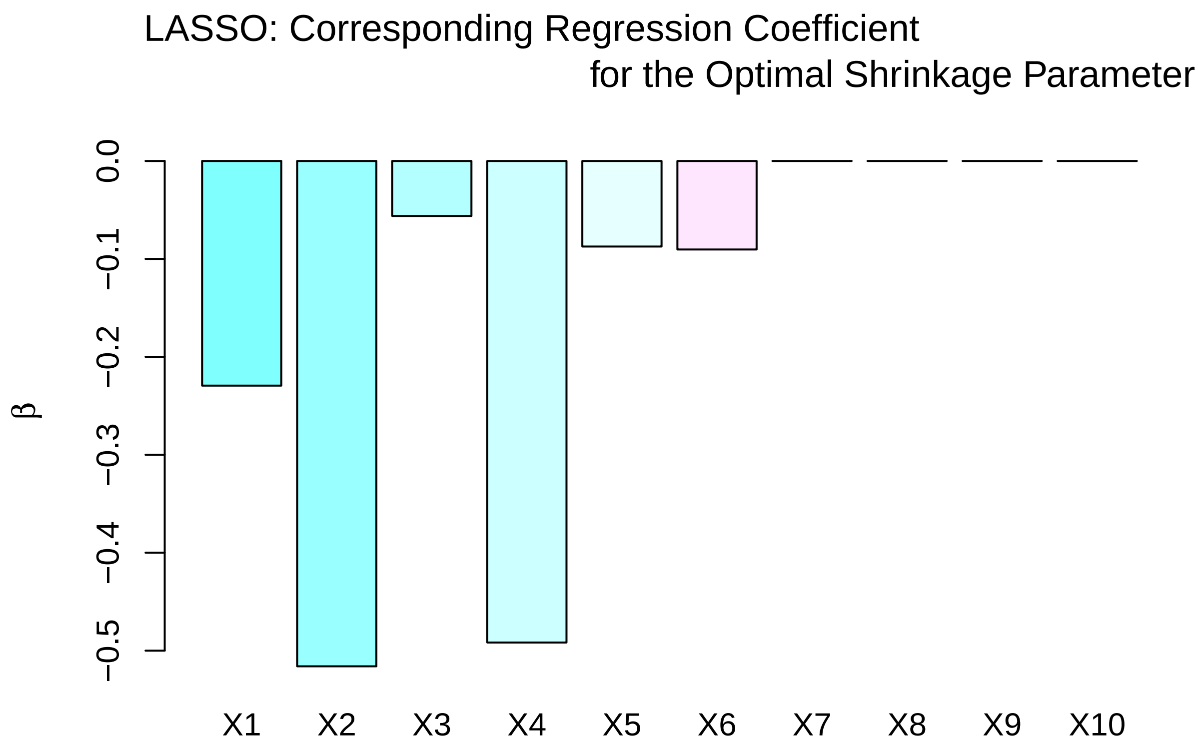



```
###########################
# Optimal lambda with beta
###########################

index <- c()
mse_optimal <- c()
beta_optimal <- c()
lambda_optimal <- c()
abs_error <- c()

for(j in 1:J){
  index <- c(index, which.min(mse_hat[,j]))
}
for(j in 1:J){
  mse_optimal <- c(mse_optimal, mse_hat[index[j],j])
  lambda_optimal <- c(lambda_optimal, lambda[index[j]])
  beta_optimal <- c(beta_optimal, lasso$beta.hat[index[j],j])
  abs_error <- c(abs_error, abs((abs(beta_optimal[j]) - abs(Beta[j]))))
}

df <- data.frame(mse_optimal, lambda_optimal, beta_optimal, beta=Beta, abs_error)
rownames(df) <- c("X1","X2","X3","X4","X5",
                  "X6","X7","X8","X9","X10")

# plot of optimal lambda

barplot(df$lambda_optimal, main = expression("LASSO: Optimal Shrinkage Parameter"), ylab = expression(la
```



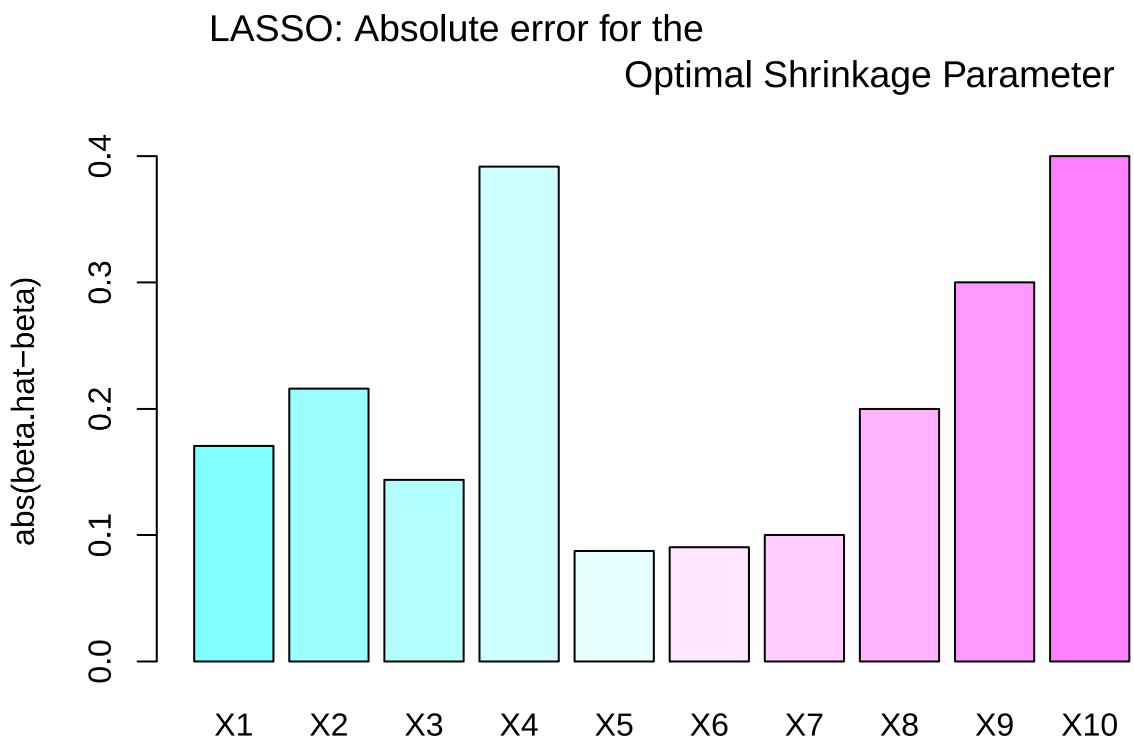

```r
barplot(df$beta_optimal, main = expression("LASSO: Corresponding Regression Coefficient
                                            for the Optimal Shrinkage Parameter"),
        ylab = expression(hat(beta)), col=cm.colors(10), names.arg = rownames(df))
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## font metrics unknown for character 0xa

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## font metrics unknown for character 0xa
```



```r
barplot(df$abs_error, main = expression("LASSO: Absolute error for the
                                        Optimal Shrinkage Parameter"),
        ylab = "abs(beta.hat-beta)", col=cm.colors(10), names.arg = rownames(df))
```

```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## font metrics unknown for character 0xa

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## font metrics unknown for character 0xa
```




```r
df
```

```
##      mse_optimal lambda_optimal beta_optimal beta  abs_error
## X1  -0.369777641              0  -0.22935632 -0.4 0.17064368
## X2  -0.317056711              0  -0.51600349 -0.3 0.21600349
## X3  -0.177626491              0  -0.05613554 -0.2 0.14386446
## X4  -0.104994254              1  -0.49169139 -0.1 0.39169139
## X5  -0.005412870              8  -0.08734287  0.0 0.08734287
## X6  -0.002723464              0  -0.09035998  0.0 0.09035998
## X7   0.010000000             85   0.00000000  0.1 0.10000000
## X8   0.040071416            100   0.00000000  0.2 0.20000000
## X9   0.090844985            100   0.00000000  0.3 0.30000000
## X10  0.160626564            100   0.00000000  0.4 0.40000000
```

We see the bias-variance tradeoff in the graphs of the estimated bias versus the shrinkage parameterand thte estimated variance versus the shrinkage parameter. The estimated bias increases across all covariates as we increase the shrinkage parameter whereas the estimated variance decreases as we increase the shrinakge parameter. The MSE plot shows the variability within the covariates; some eventually reach an optimal level of zero, others have a generally maintained MSE around zero, and some covariates don't even attain an MSE of zero. Specifically, covariates X5 and X6 have a very low MSE throughout all levels of the shrinkage parameter. The MSE of covariates like X1 and X2 steadily decrease as the shrinakge parameter increases but then increase if the shrinkage parameter gets too high. The MSE of covariates X8, X9, and X10 steadily decrease as the shrinkage parameter increases and we do not see the MSE reach zero for these covariates. Estimating the optimal shrinkage parameter within this range of the shrinkage parameter is not ideal for those covariates that did not acheive an "optimal" MSE (i.e. an MSE of zero). We still proceed to find the values of the optimal shrinkage parameter (those that minimize the MSE) with the corresponding regression coefficient estimate and compare this to the true regression coefficients in the last plot. There is a noticable amount of variablility for the absolute error (absolute difference from the estimate to the truth) across the covariates. We expect a larger error for covariates X8, X9, and X10 but the large error term coming from other covariates that did reach an optimal MSE is probably due to the fact that we used the betas from the original lasso regression performed in step b to define the absolute error term. If we knew of a clearly defined way to choose a beta from the bootstrap sample, then we could have attempted that and might have gotten closer to the truth. The table displays that these optimal shrinkage parameters did truly have very small MSE values.

Ideally, we would like to address the limitations mentioned by increasing the range of the shrinkage parameter to see if the covariates X8, X9, and X10 do end up reaching an MSE of 0. Also, we would like to be able to chose the beta corresponding to the optimal shrinkage parameter in a different manner than we did above.

Collaborators & Resources

Tommy Carpenito, Kelly Street for coding help

PH240D 'Regularized Regression' Lecture Slides

PH240D 'Regularized Regression' Example with R Script

Tibshirani, 1996 *Regression Shrinkage and Selection via the LASSO*

Efron, 2014 *Estimation and Accuracy after Model Selection*

StackExchange helpful proofs:

https://math.stackexchange.com/questions/1518335/prove-the-estimator-hatb-of-ridge-regression-mean-of-the-posterior-distri

https://stats.stackexchange.com/questions/182098/why-is-lasso-penalty-equivalent-to-the-double-exponential-laplace-prior