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, $\hat{\beta}_n^{\text{LASSO}} \equiv \text{ arg min }_{\beta\in\mathbf{R}^J} \|\mathbf{Y}_n - \mathbf{X}_n\beta\|_2^2 + \lambda\|\beta\|_1 = \text{ arg min }_{\beta\in\mathbf{R}^J} \sum_{i=1}^n \left(Y_i - \sum_{j=1}^J \beta_j X_{i,j}\right)^2 + \lambda \sum_{j=1}^J |\beta_j|$ 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 $\hat{\beta}_n^{\text{LASSO}} = \text{ arg min }_{\beta\in\mathbf{R}^J} \sum_{i=1}^n \big(Y_i - \sum_{j=1}^J \beta_j X_{i,j}\big)^2$ 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 $Y_n | X_n, \sim N(X_n \beta, \sigma^2 I_n)$, 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:

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:

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,

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

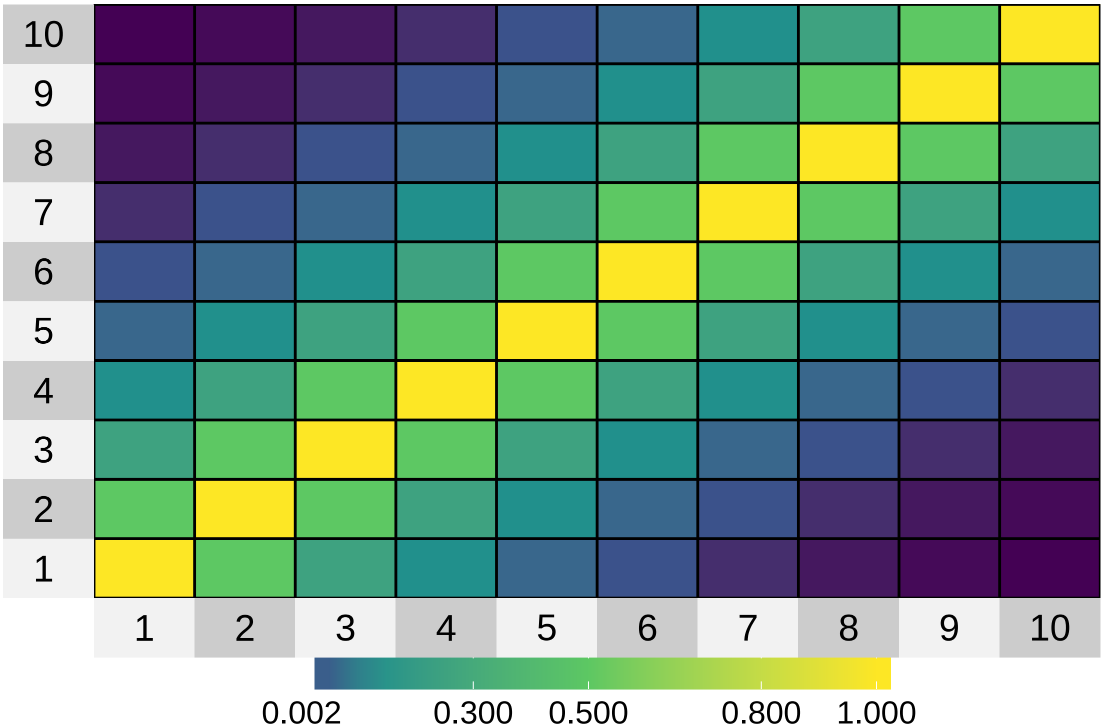
for $\rho \in (-1, 1)$. Set the parameter values to $J = 10$, $\beta = (-J/2+1, ..., -2, -1, 0, 0, 1, 2, ..., J/2-1)/10$, $\sigma = 2$, and $\rho = 0.5$.

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:

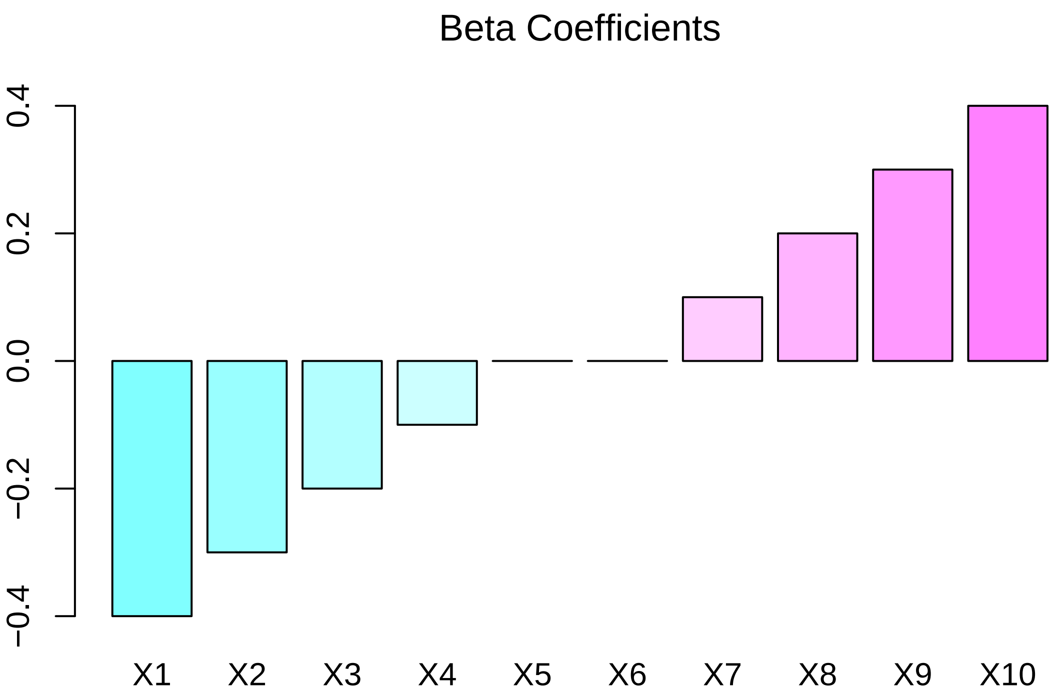
```
##########################
# Given Information
#########################

# 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)
```




```r
# 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
```

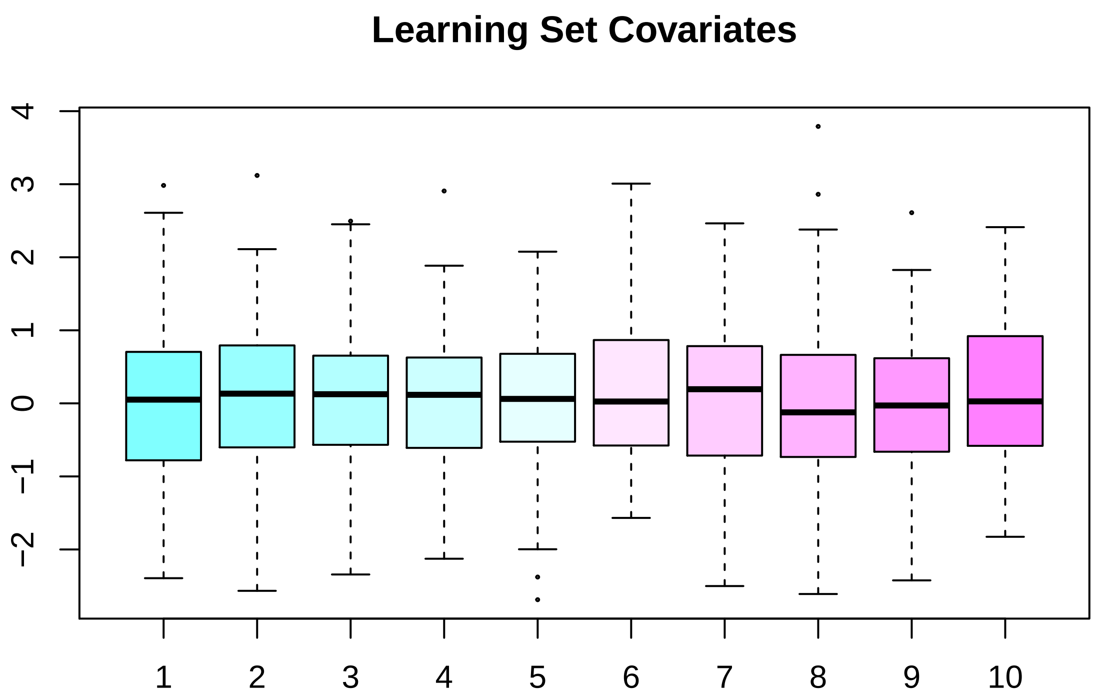
```r
barplot(Beta, main = expression("Beta Coefficients"),
col=cm.colors(10), names.arg = c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                 "X8", "X9", "X10"))
```



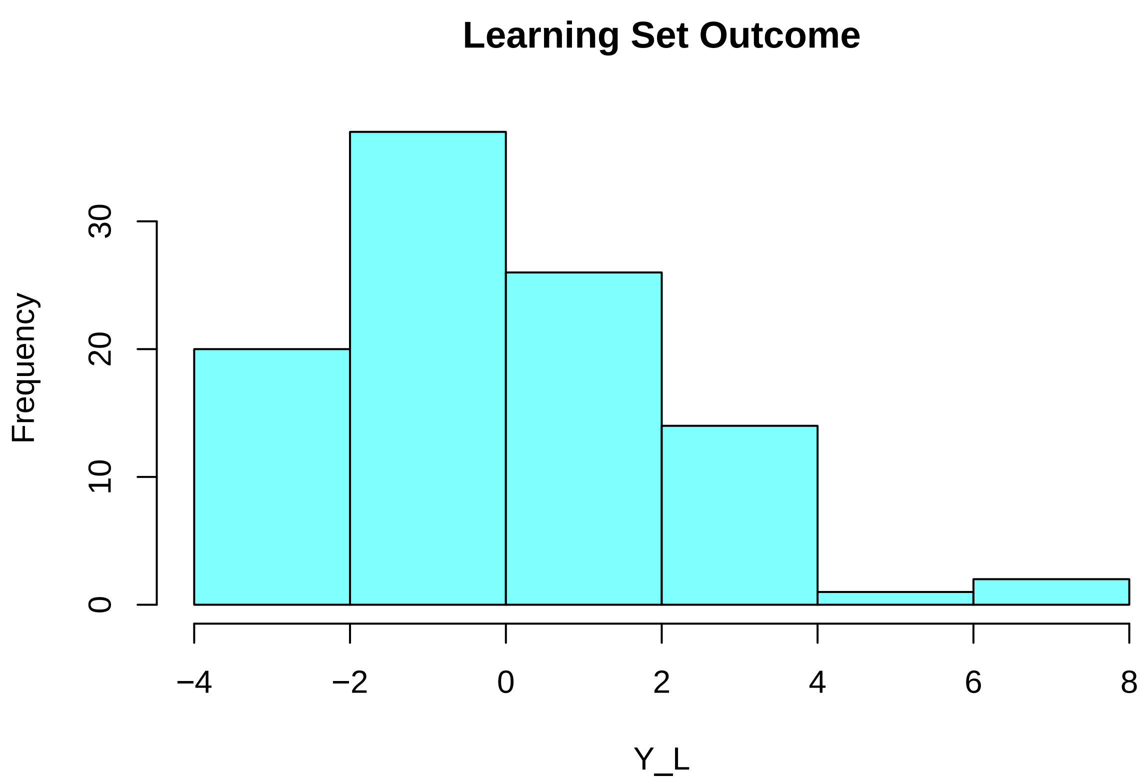
```r
###########################
# Learning Set Simulation
###########################

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")
```




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




```r
summary(Y_L)
```

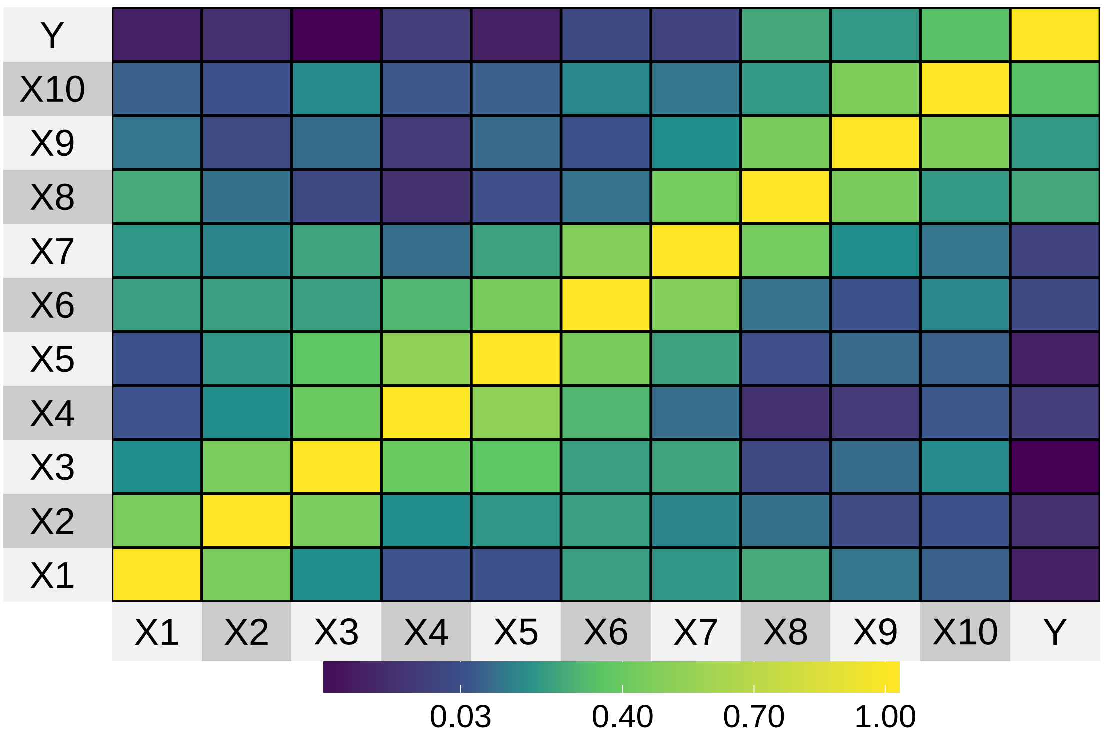
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -3.98890 -1.53167 -0.32770 -0.08656  1.22870  7.47284
```

```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)
```

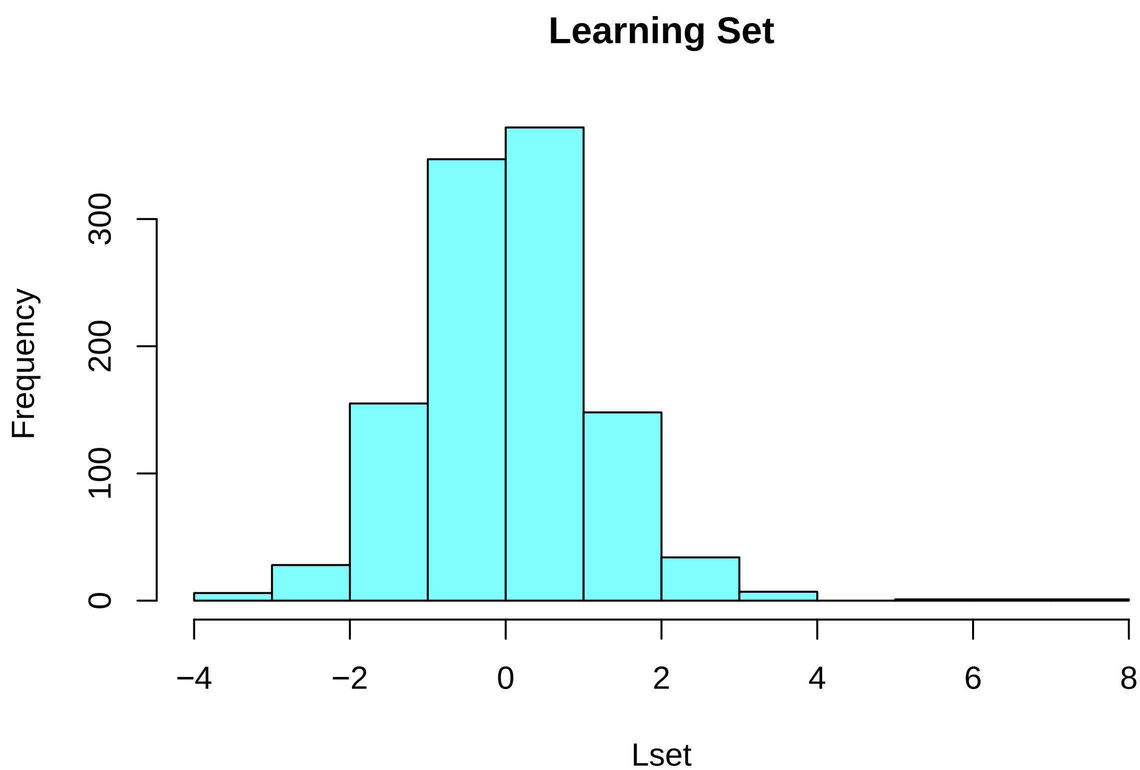
```
##        X1                 X2                X3
##  Min.   :-2.39294   Min.   :-2.5661   Min.   :-2.34250
##  1st Qu.:-0.75134   1st Qu.:-0.5981   1st Qu.:-0.55312
##  Median : 0.05113   Median : 0.1331   Median : 0.12556
##  Mean   : 0.04526   Mean   : 0.1342   Mean   : 0.06867
##  3rd Qu.: 0.69626   3rd Qu.: 0.7789   3rd Qu.: 0.64637
##  Max.   : 2.98346   Max.   : 3.1200   Max.   : 2.49498
##        X4                 X5                 X6
##  Min.   :-2.12672   Min.   :-2.68745   Min.   :-1.56827
##  1st Qu.:-0.59241   1st Qu.:-0.52401   1st Qu.:-0.56839
##  Median : 0.11697   Median : 0.06095   Median : 0.02543
##  Mean   : 0.02128   Mean   : 0.06060   Mean   : 0.07895
##  3rd Qu.: 0.62572   3rd Qu.: 0.66862   3rd Qu.: 0.86036
##  Max.   : 2.90738   Max.   : 2.07655   Max.   : 3.00805
##        X7                X8                 X9                X10
##  Min.   :-2.5008   Min.   :-2.60969   Min.   :-2.42310   Min.   :-1.8264
##  1st Qu.:-0.7134   1st Qu.:-0.73190   1st Qu.:-0.65838   1st Qu.:-0.5757
```

```
##  Median : 0.1938   Median :-0.12297   Median :-0.02910   Median : 0.0271
##  Mean   : 0.1060   Mean   :-0.06177   Mean   :-0.03624   Mean   : 0.1762
##  3rd Qu.: 0.7601   3rd Qu.: 0.65779   3rd Qu.: 0.61718   3rd Qu.: 0.9134
##  Max.   : 2.4642   Max.   : 3.79094   Max.   : 2.61036   Max.   : 2.4125
##        Y
##  Min.   :-3.98890
##  1st Qu.:-1.53167
##  Median :-0.32770
##  Mean   :-0.08656
##  3rd Qu.: 1.22870
##  Max.   : 7.47284
```

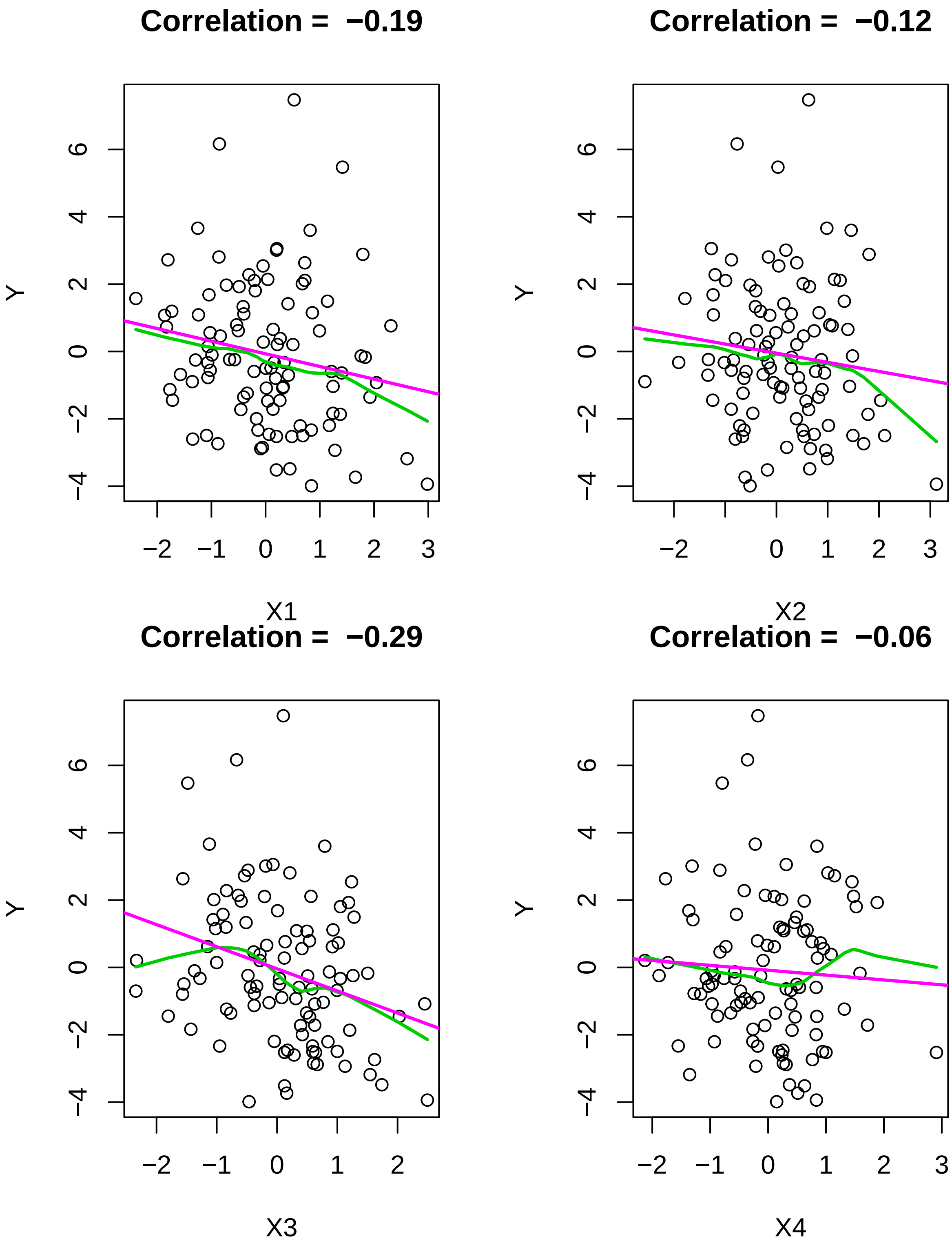
```
# we can visualize the correlations in a heatmap
superheat(cor(Lset))
```

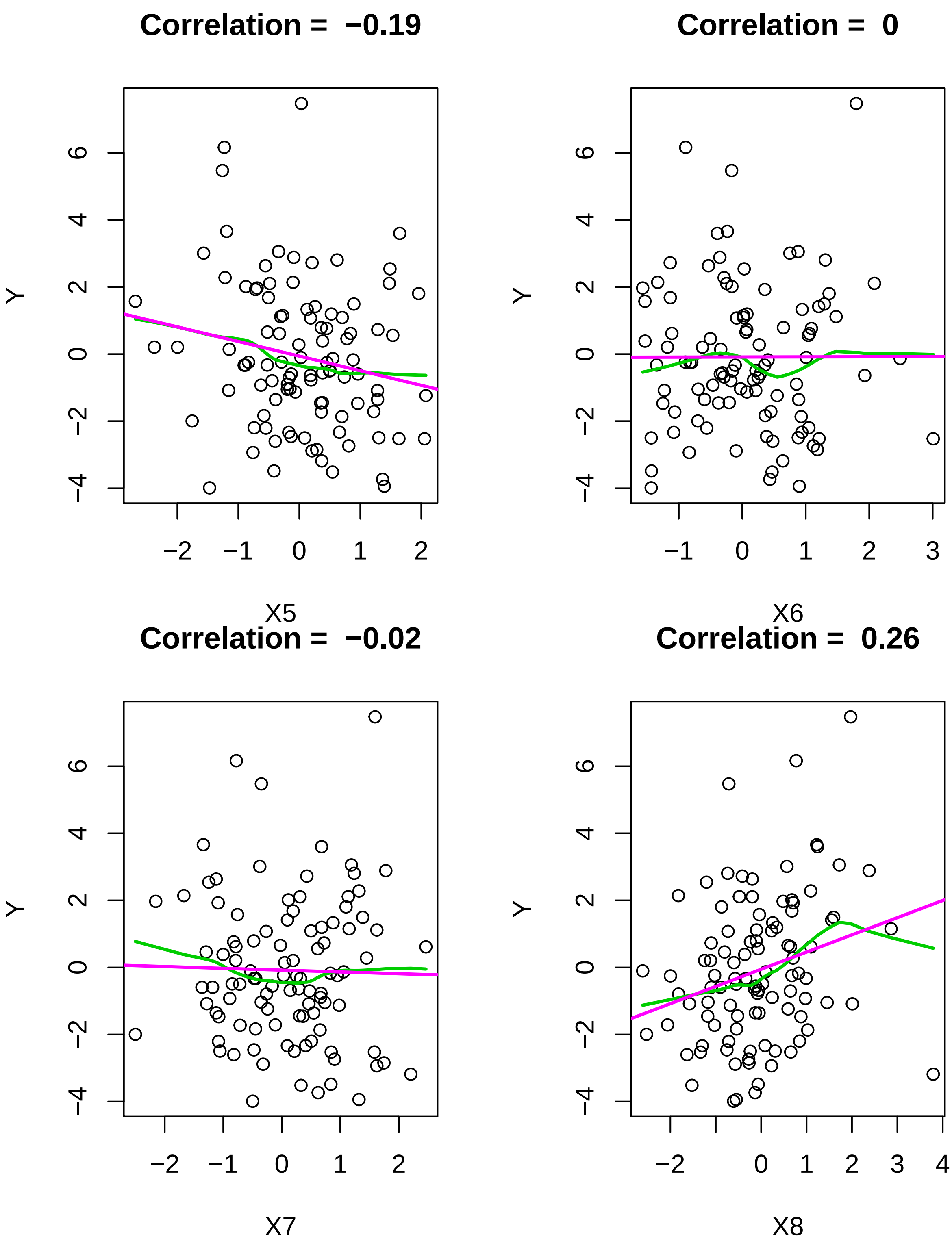


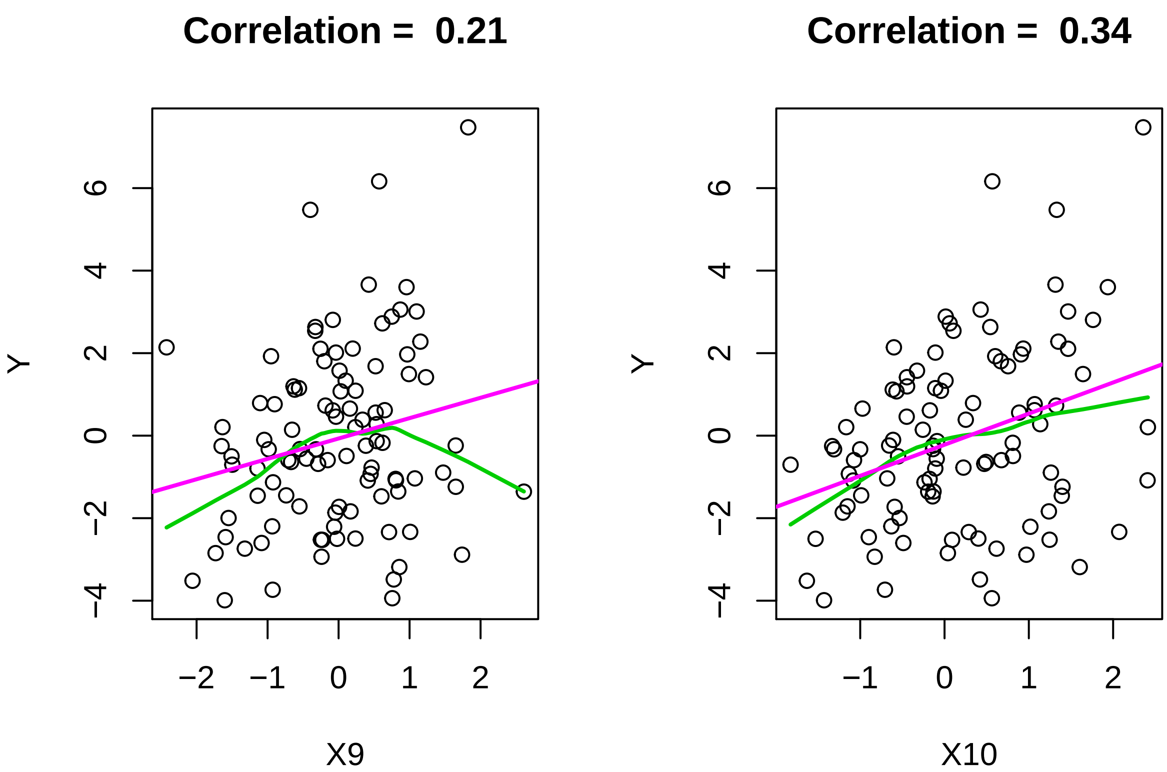

```
hist(Lset, col=cm.colors(1), main = "Learning Set")
```




```
# 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 = −0.19
Correlation = −0.12
Correlation = −0.29
Correlation = −0.06
Y
X1
X2
X3
X4
−4
−2
0
2
4
6
−2
−1
0
1
2
3


Correlation = −0.19
Y
X5
Correlation = 0
Y
X6
Correlation = −0.02
Y
X7
Correlation = 0.26
Y
X8




```r
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
```

```r
# 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

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

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

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

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),

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

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)
```

```
        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
## -3.1950 -1.1266 -0.0831  1.0421  4.6172
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -0.08656    0.18104  -0.478 0.633750
## scale(Lset_X)1  -0.65221    0.21398  -3.048 0.003032 **
```

```
## scale(Lset_X)2   0.29272    0.22848   1.281 0.203477
## scale(Lset_X)3  -0.78061    0.23090  -3.381 0.001075 **
## scale(Lset_X)4   0.41663    0.22691   1.836 0.069676 .
## scale(Lset_X)5  -0.54669    0.23542  -2.322 0.022508 *
## scale(Lset_X)6   0.27959    0.23529   1.188 0.237887
## scale(Lset_X)7  -0.19875    0.23547  -0.844 0.400893
## scale(Lset_X)8   0.60963    0.23285   2.618 0.010390 *
## scale(Lset_X)9   0.12921    0.22995   0.562 0.575588
## scale(Lset_X)10  0.72760    0.21110   3.447 0.000868 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.81 on 89 degrees of freedom
## Multiple R-squared:  0.3813, Adjusted R-squared:  0.3118
## F-statistic: 5.485 on 10 and 89 DF,  p-value: 2.613e-06
```

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

```
##                      V1          V2          V3          V4          V5
## -0.08655711 -0.65221095  0.29271767 -0.78061309  0.41663362 -0.54668579
##          V6          V7          V8          V9         V10
##  0.27959319 -0.19875382  0.60963492  0.12920916  0.72760344
```

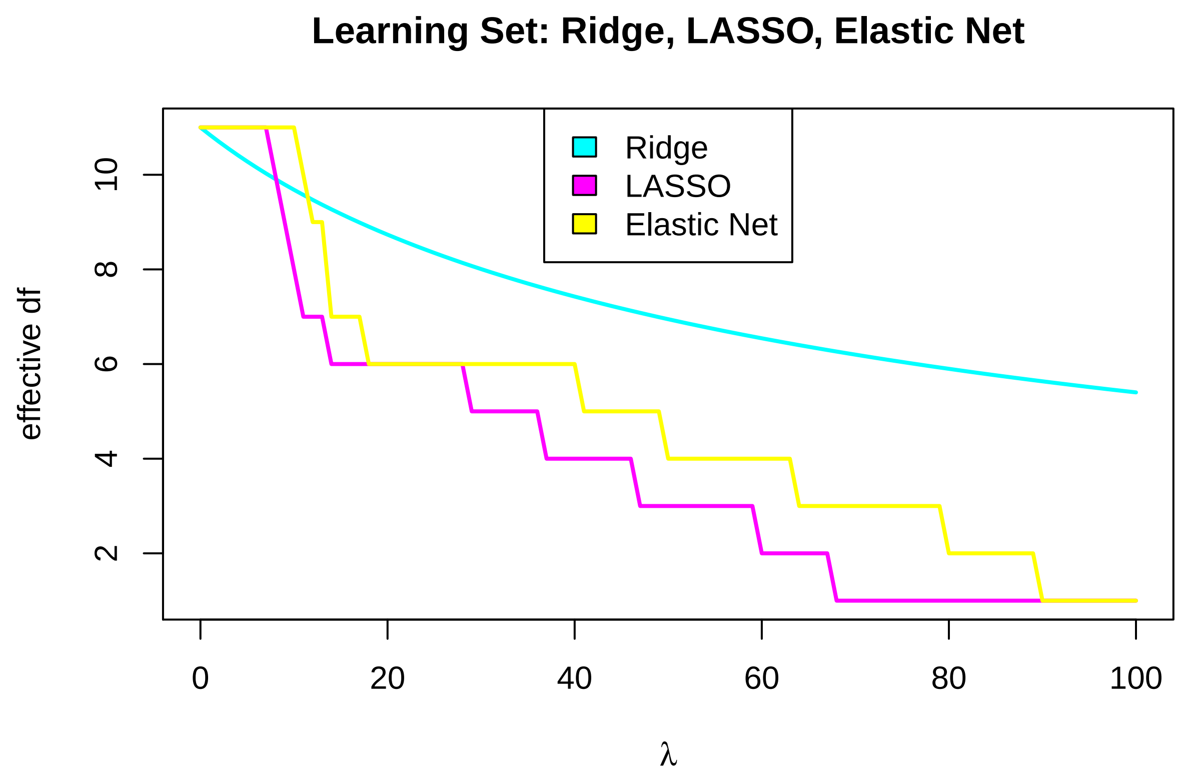
```
lasso$beta.hat[1,]
```

```
##                      V1          V2          V3          V4          V5
## -0.08655711 -0.65221064  0.29271758 -0.78061365  0.41663392 -0.54668536
##          V6          V7          V8          V9         V10
##  0.27959274 -0.19875378  0.60963494  0.12920874  0.72760372
```

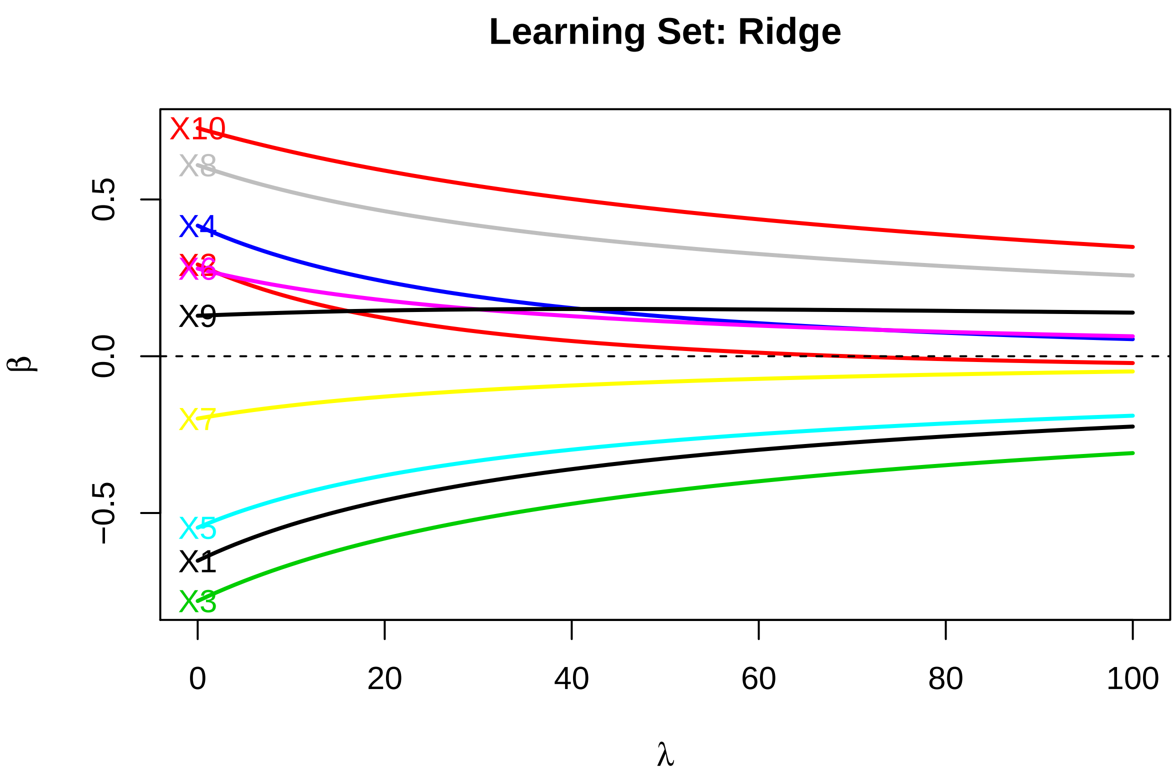
```
enet$beta.hat[1,]
```

```
##                      V1          V2          V3          V4          V5
## -0.08655711 -0.65221066  0.29271759 -0.78061365  0.41663391 -0.54668538
##          V6          V7          V8          V9         V10
##  0.27959279 -0.19875380  0.60963494  0.12920876  0.72760371
```

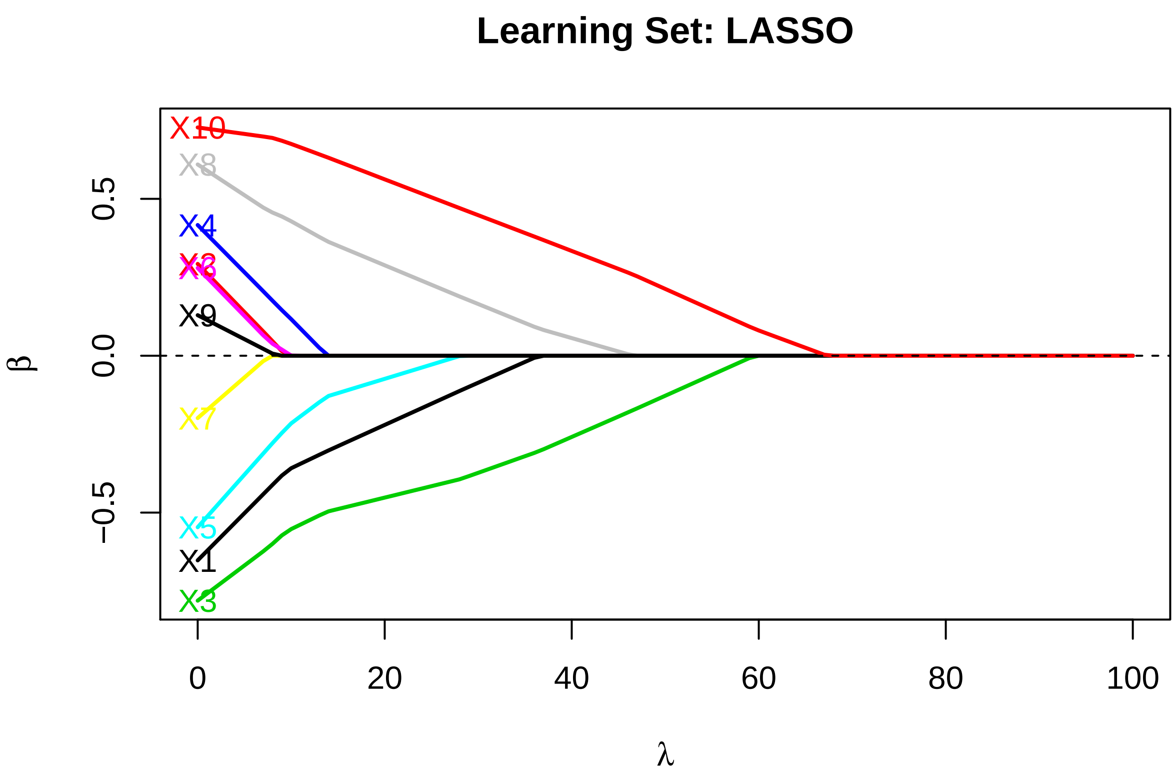
```
###########################
# Plots
###########################

# 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)
```



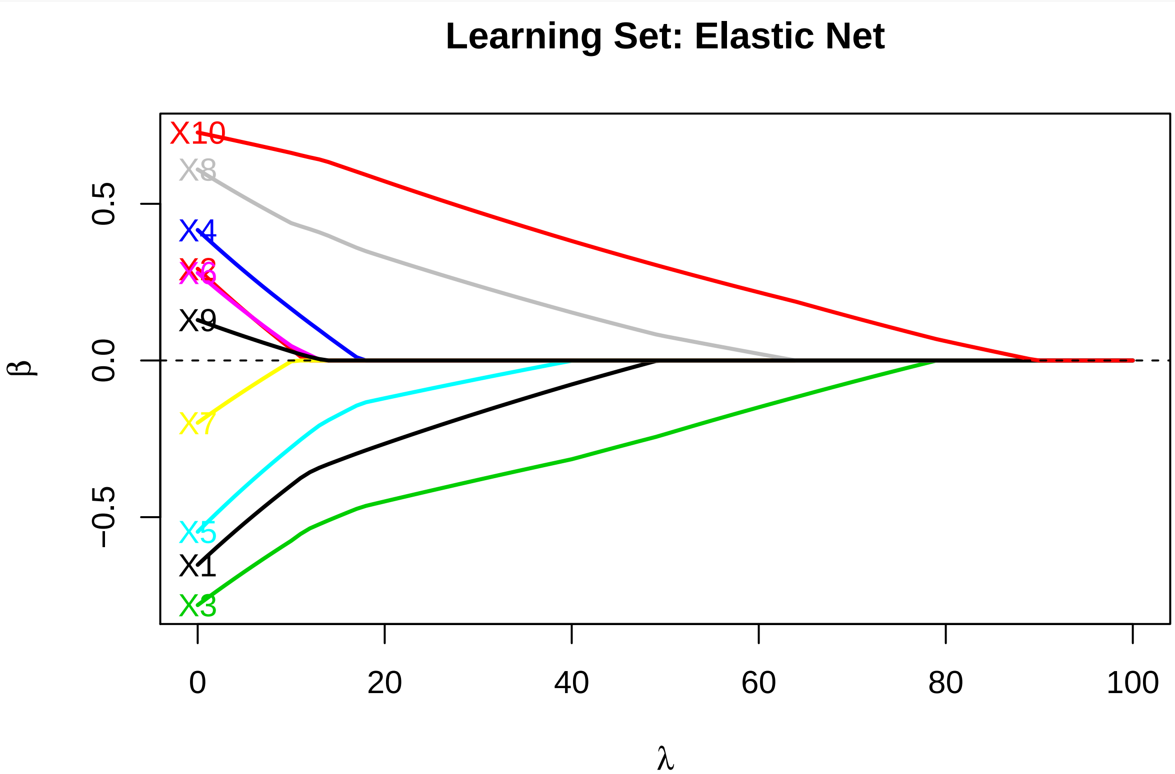

```
myPlotBeta(lambda, ridge$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
           labels=colnames(Lset[,1:10]), main="Learning Set: Ridge")
```



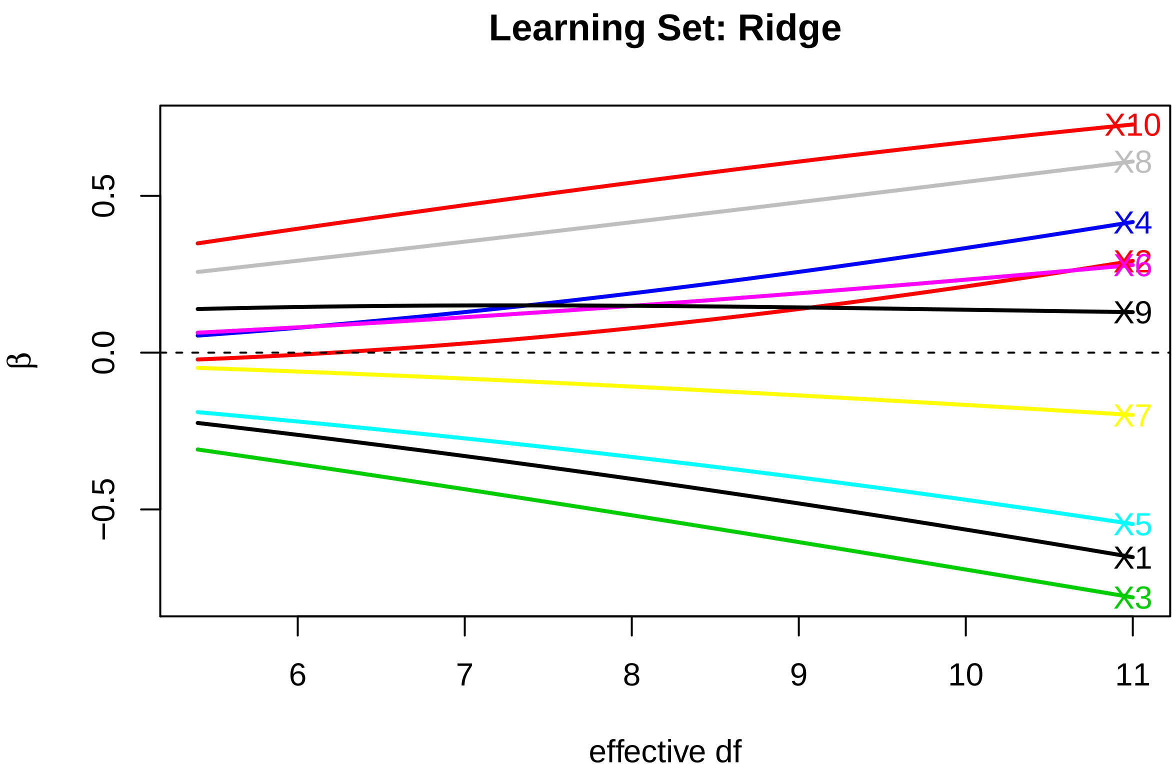

```
myPlotBeta(lambda, lasso$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
           labels=colnames(Lset[,1:10]), main="Learning Set: LASSO")
```



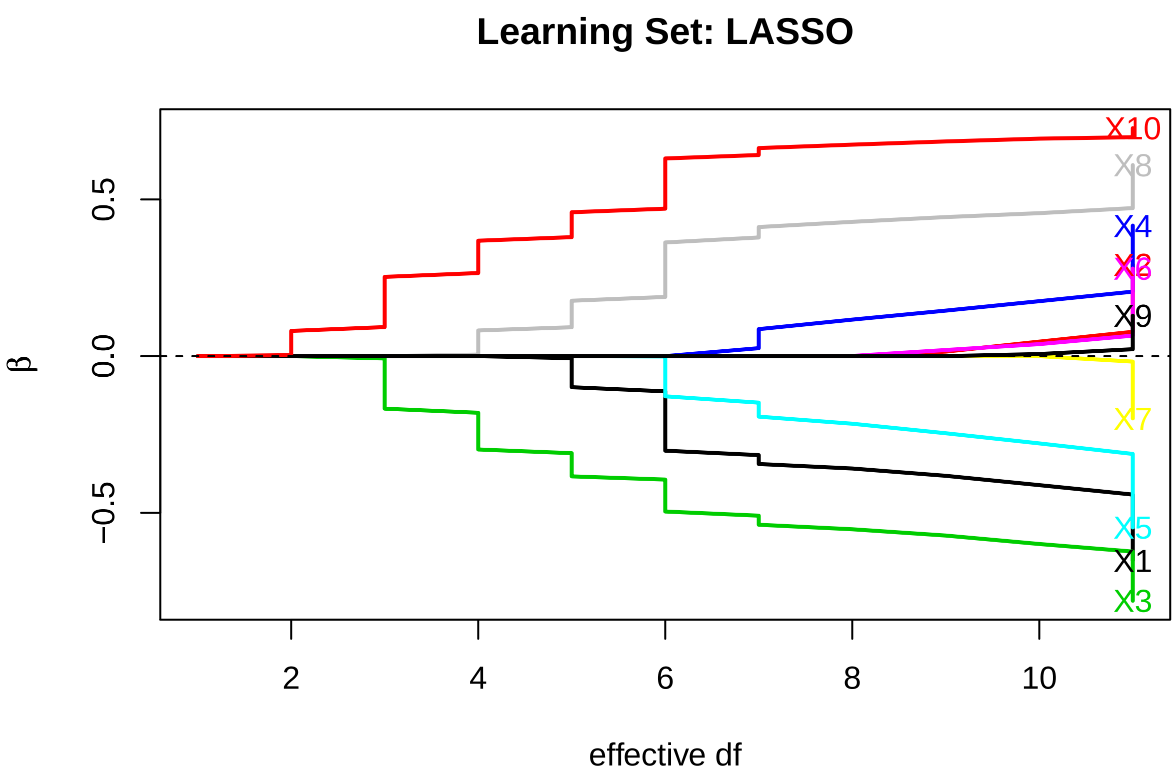

```r
myPlotBeta(lambda, enet$beta.hat[,-1],type="l",lwd=2, ylab=expression(hat(beta)),
           labels=colnames(Lset[,1:10]), main="Learning Set: Elastic Net")
```



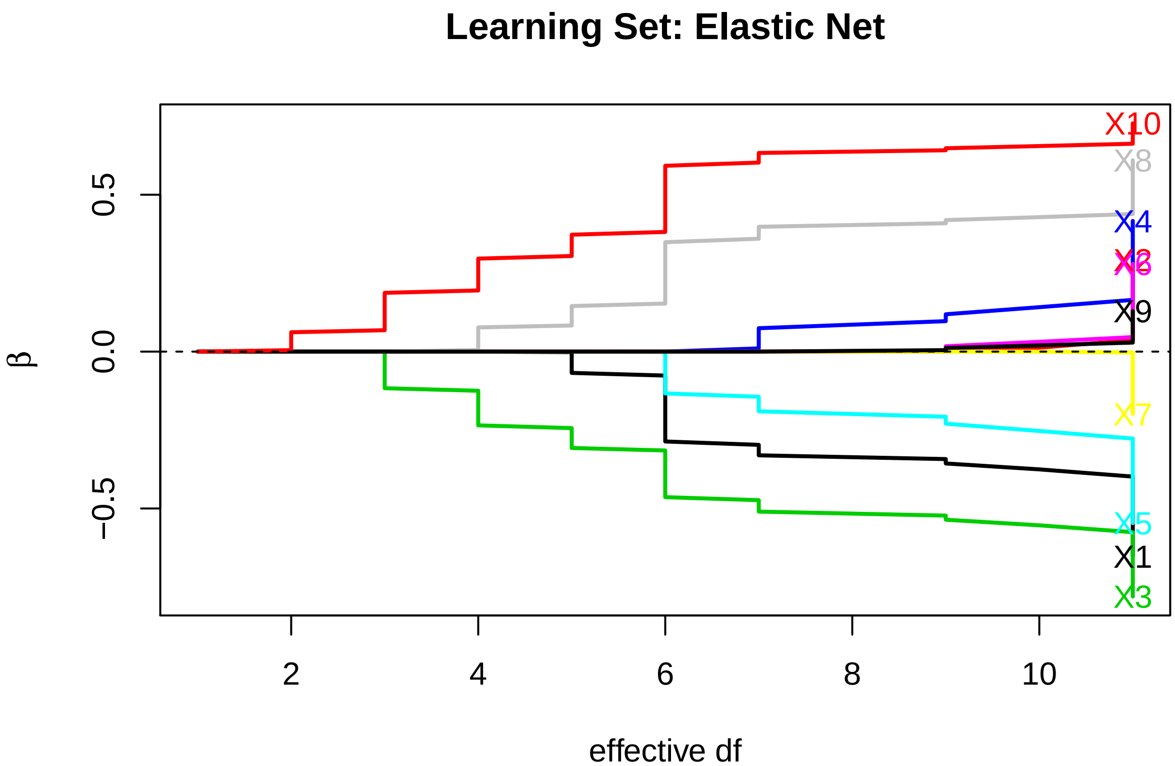

```r
# Beta vs. effective df
myPlotBeta(ridge$df,ridge$beta.hat[,-1],type="l",lwd=2,xlab="effective df",
           ylab=expression(hat(beta)),labels=colnames(Lset[,1:10]),main="Learning Set: Ridge")
```




```
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")
```



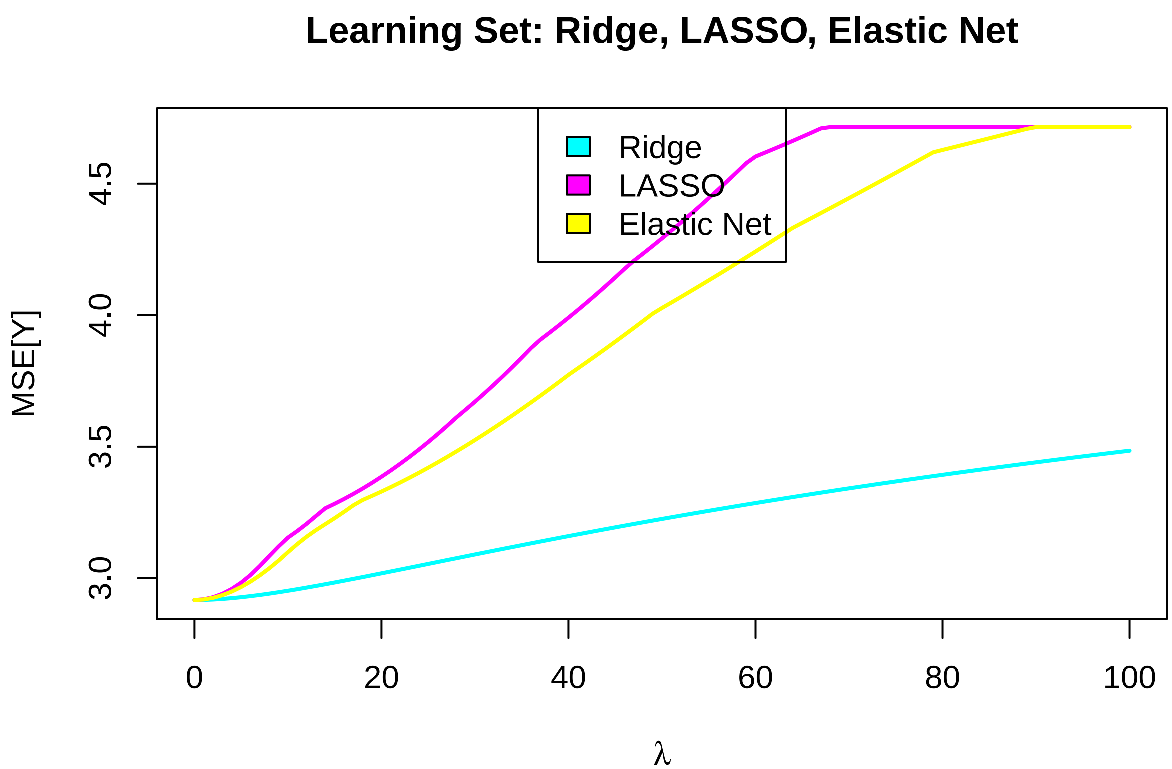
```
myPlotBeta(enet$df,enet$beta.hat[,-1],type="l",lwd=2,xlab="effective df",
          ylab=expression(hat(beta)),labels=colnames(Lset[,1:10]),main="Learning Set: Elastic Net")
```




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

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

summary(mse.Lset)
```

```
##       Ridge            LASSO          Elastic Net
##  Min.   :2.917   Min.   :2.917   Min.   :2.917
##  1st Qu.:3.054   1st Qu.:3.518   1st Qu.:3.420
##  Median :3.225   Median :4.292   Median :4.027
##  Mean   :3.210   Mean   :4.102   Mean   :3.957
##  3rd Qu.:3.368   3rd Qu.:4.715   3rd Qu.:4.541
##  Max.   :3.485   Max.   :4.715   Max.   :4.715
```

```
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.

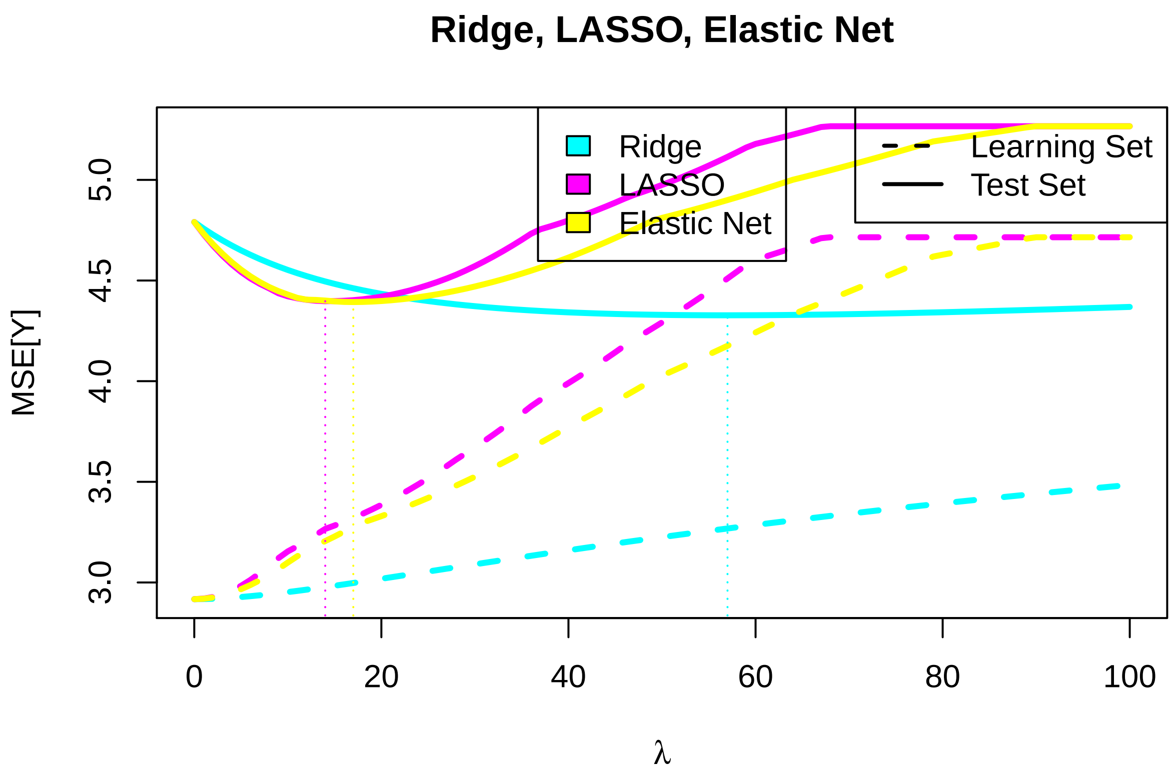
Solution:

```
############################
# 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)
```

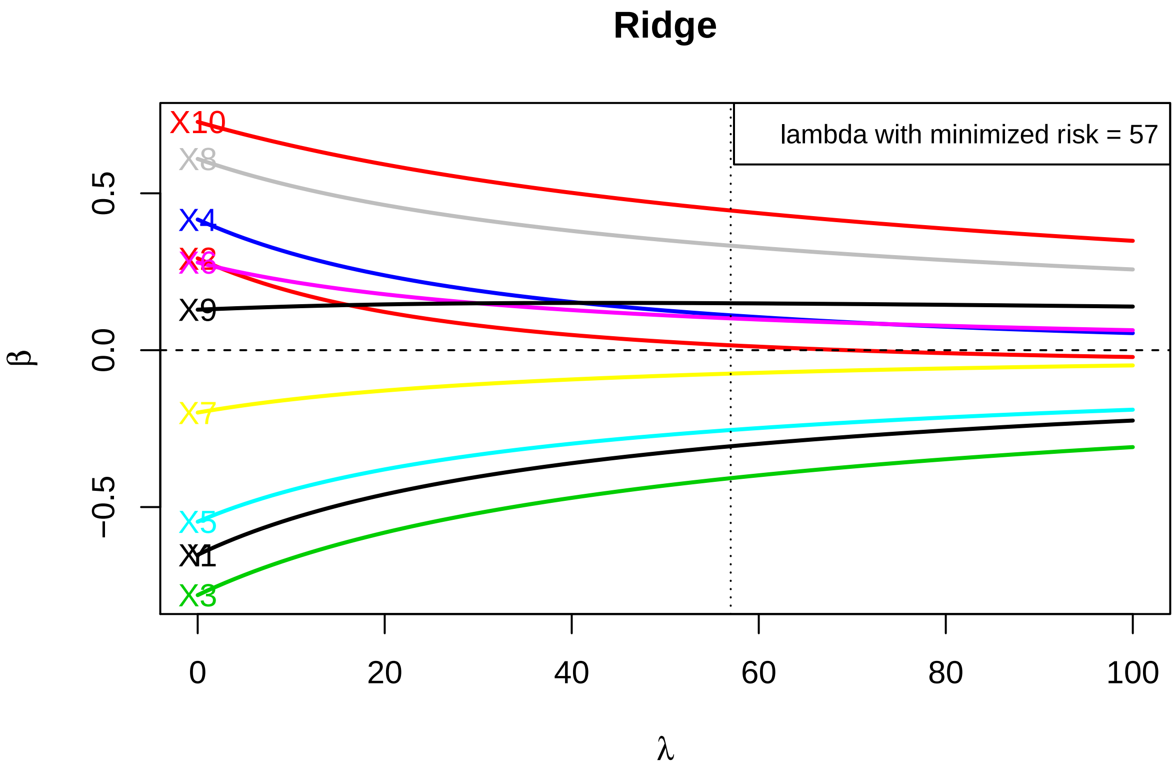
```
##      Ridge           LASSO        Elastic Net
##  Min.   :4.327   Min.   :4.398   Min.   :4.393
##  1st Qu.:4.334   1st Qu.:4.572   1st Qu.:4.494
##  Median :4.351   Median :4.975   Median :4.811
##  Mean   :4.396   Mean   :4.915   Mean   :4.819
##  3rd Qu.:4.398   3rd Qu.:5.266   3rd Qu.:5.137
##  Max.   :4.791   Max.   :5.266   Max.   :5.266
```

```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)
```

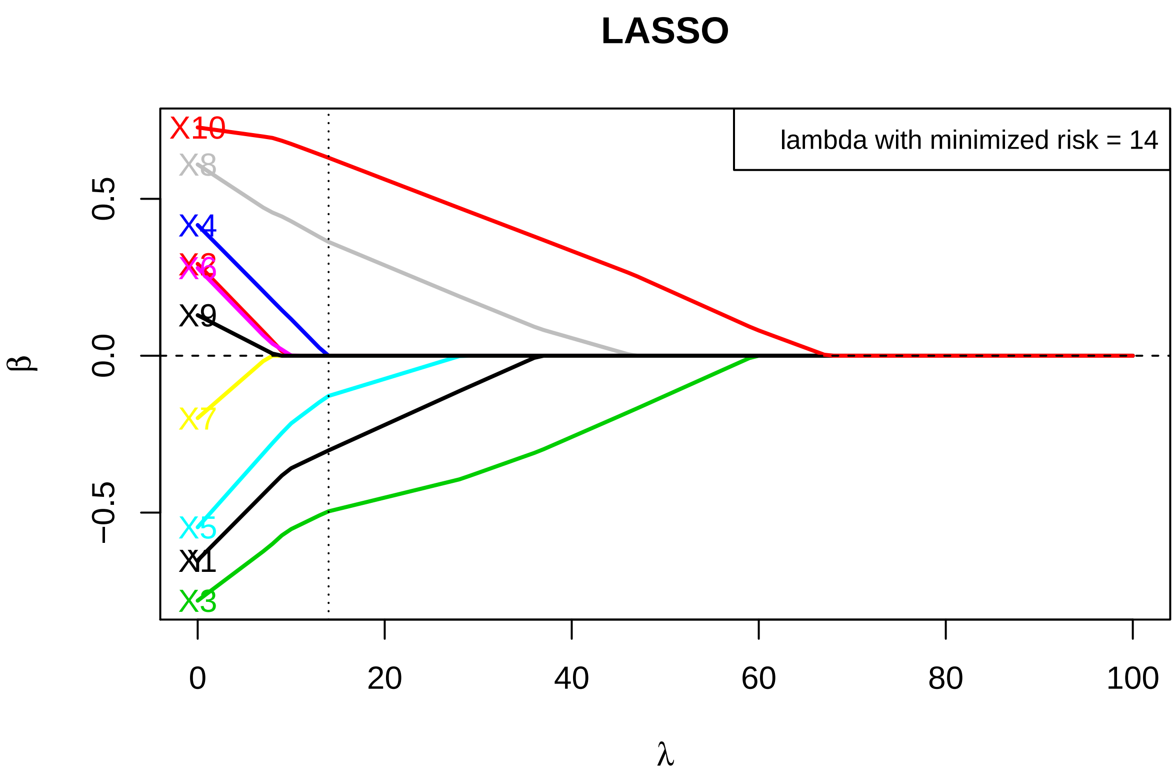



```r
##########################
# Risk v Lambda
##########################
```

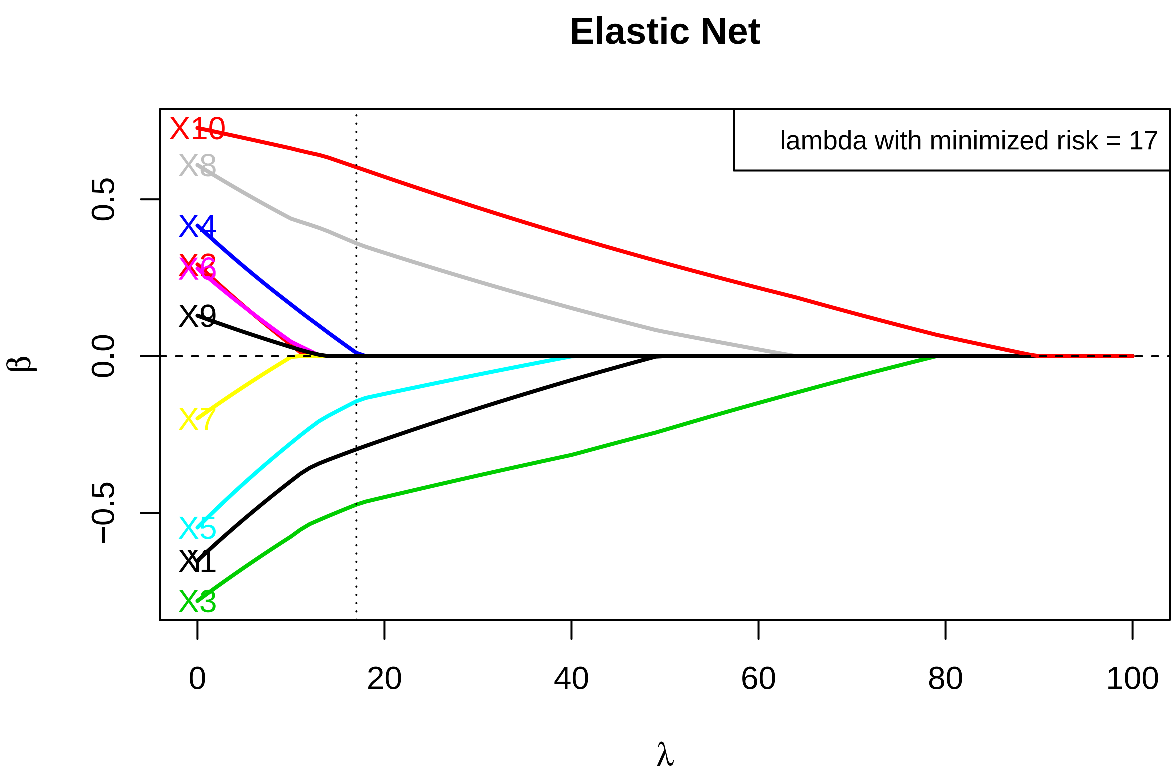
```r
myPlotBeta(lambda,ridge$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="Ridge")
abline(v = lambda[l1],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[l1], sep = "")))
```



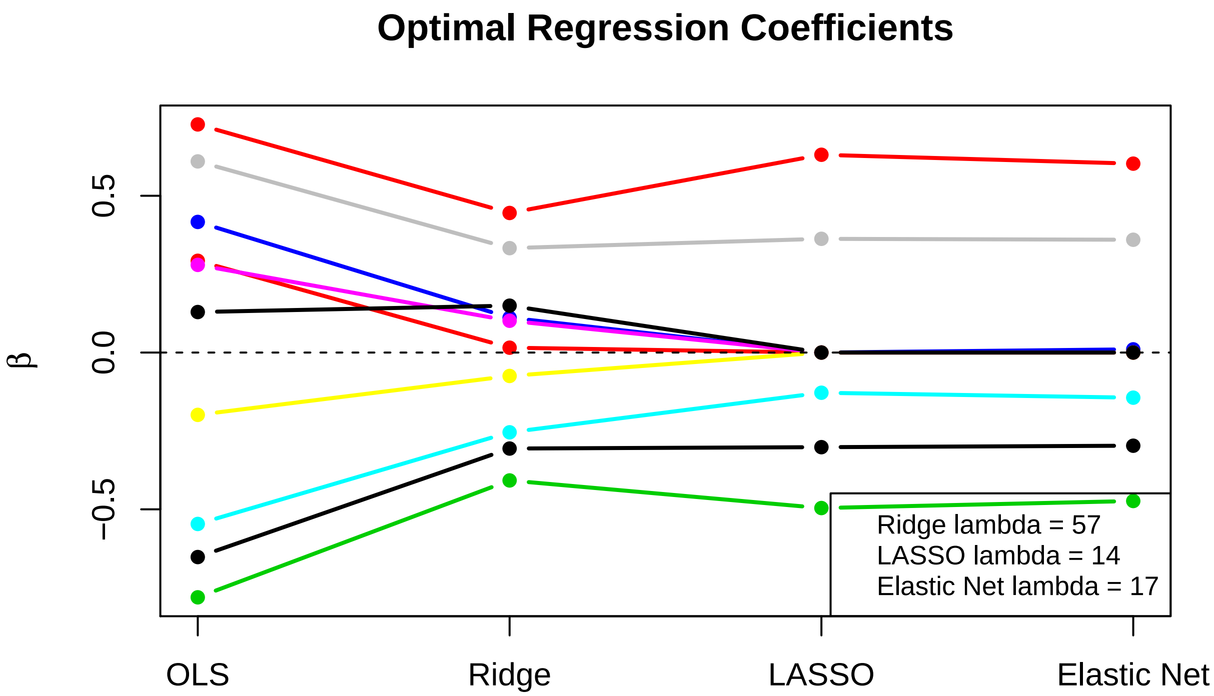

```r
myPlotBeta(lambda, lasso$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="LASSO")
abline(v = lambda[l2],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[l2], sep = "")))
```



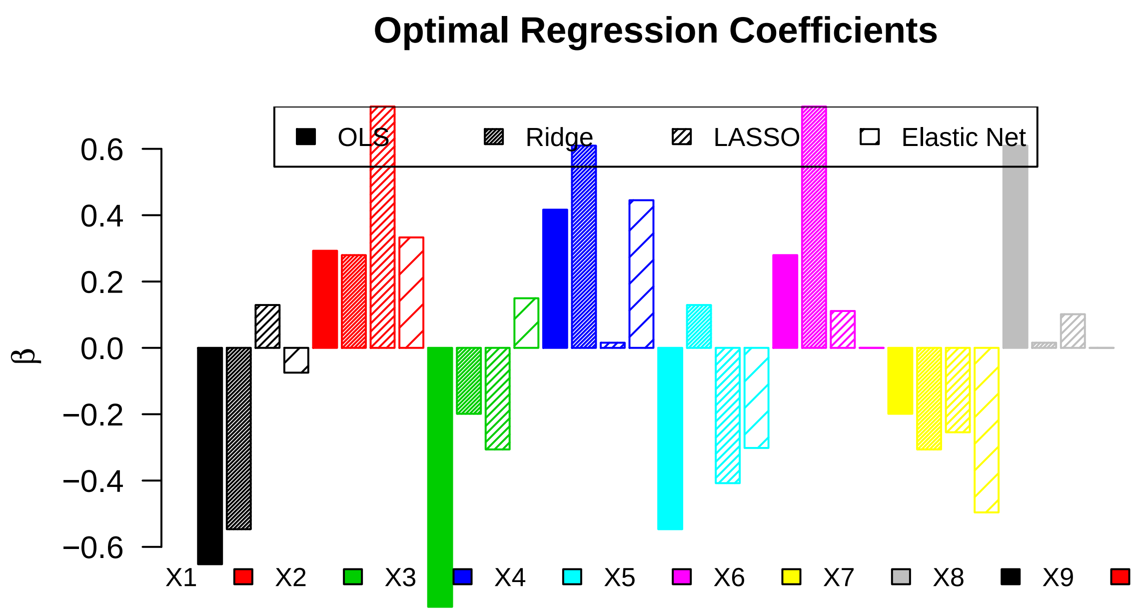

```r
myPlotBeta(lambda, enet$beta.hat[,-1],labels=colnames(Lset),right=FALSE,
           ylab=expression(hat(beta)),main="Elastic Net")
abline(v = lambda[13],col=1,lty=3)
legend("topright", pt.cex = 1, cex=.8,
       c(paste("lambda with minimized risk = ", lambda[13], sep = "")))
```



```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[l1,-1]),
            "LASSO" = c(ridge$beta.hat[l2,-1]),
            "Elastic Net" = c(ridge$beta.hat[l2,-1]))
rownames(df) <- c("X1", "X2", "X3", "X4", "X5", "X6", "X7",
                  "X8", "X9", "X10")
df
```

```
##            OLS       Ridge      LASSO Elastic.Net
## X1  -0.6522109 -0.30610622 -0.5028801  -0.5028801
## X2   0.2927177  0.01555183  0.1574876   0.1574876
## X3  -0.7806131 -0.40784698 -0.6276316  -0.6276316
## X4   0.4166336  0.11111076  0.2773685   0.2773685
## X5  -0.5466858 -0.25439491 -0.4164833  -0.4164833
## X6   0.2795932  0.10143037  0.2006043   0.2006043
## X7  -0.1987538 -0.07466878 -0.1444085  -0.1444085
## X8   0.6096349  0.33301186  0.4970527   0.4970527
## X9   0.1292092  0.14957152  0.1424199   0.1424199
## X10  0.7276034  0.44516036  0.6265951   0.6265951
```

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, $\text{Bias}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = E[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] - \beta = (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n \beta - \beta = \Big((\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n - 1\Big)\beta$.

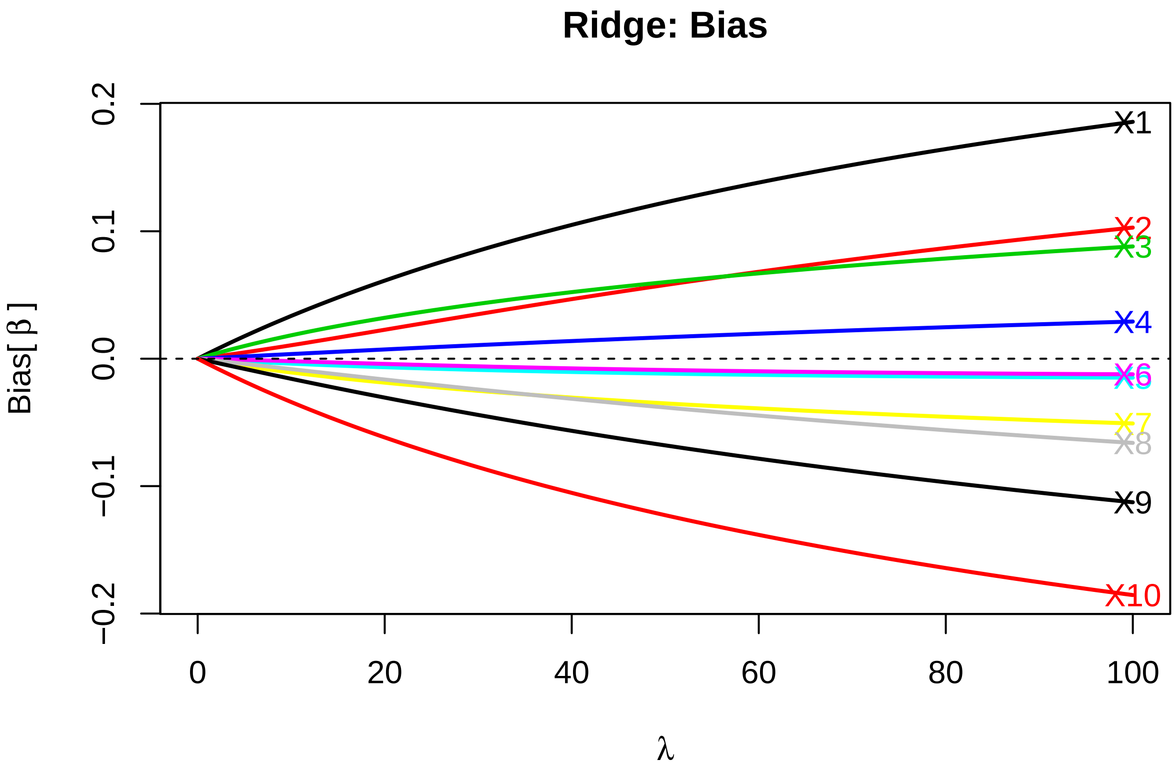
According to equation (22) on the 'Regularized Regression' lecture slides, the covariance matrix of the ridge regression estimator, $\text{Cov}[\hat{\beta}_n^{\text{ridge}}|\mathbf{X}_n] = \sigma^2 (\mathbf{X}_n^\top \mathbf{X}_n + \lambda \mathbf{I}_J)^{-1}\mathbf{X}_n^\top \mathbf{X}_n (\mathbf{X}_n^\top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

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

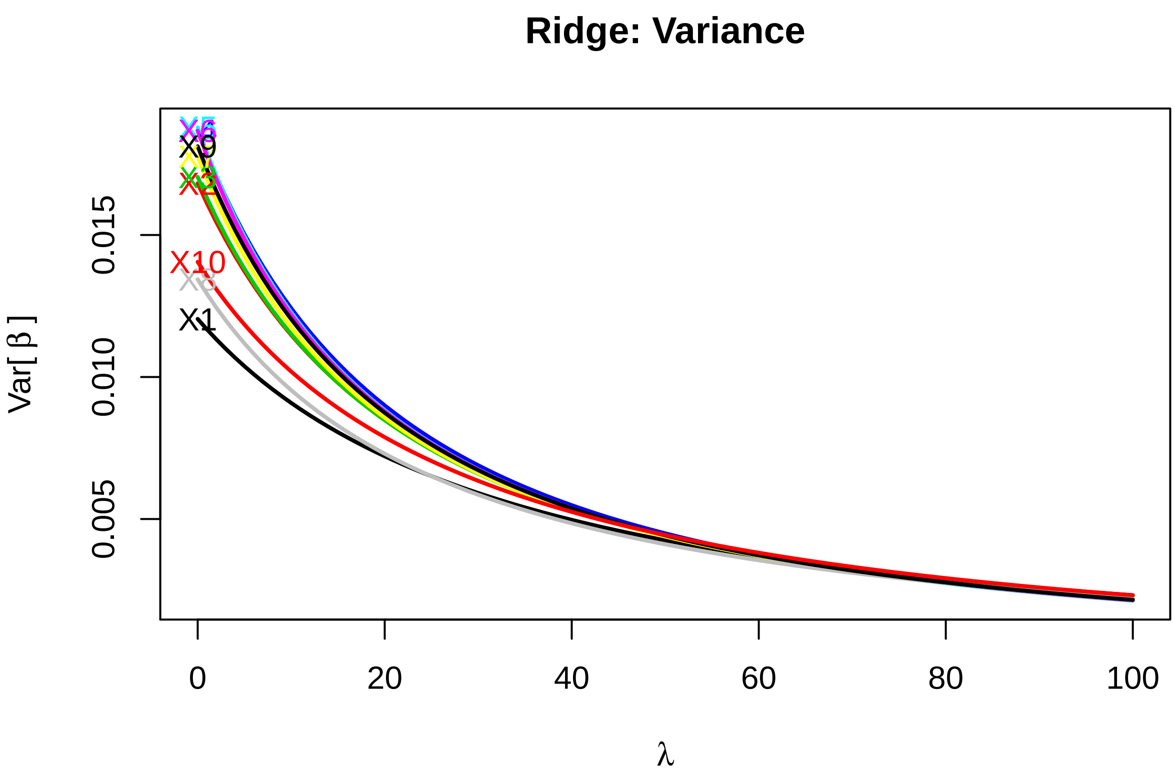
According to the slides, and we can see here, that the ridge estimator is biased. As the shrinkage parameter increases, the bias tends to increase while variance tends to decrease. This is because we become more data-adaptive and less smooth as we increase the shrinkage parameter, highlighting the bias-variance 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
  }

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

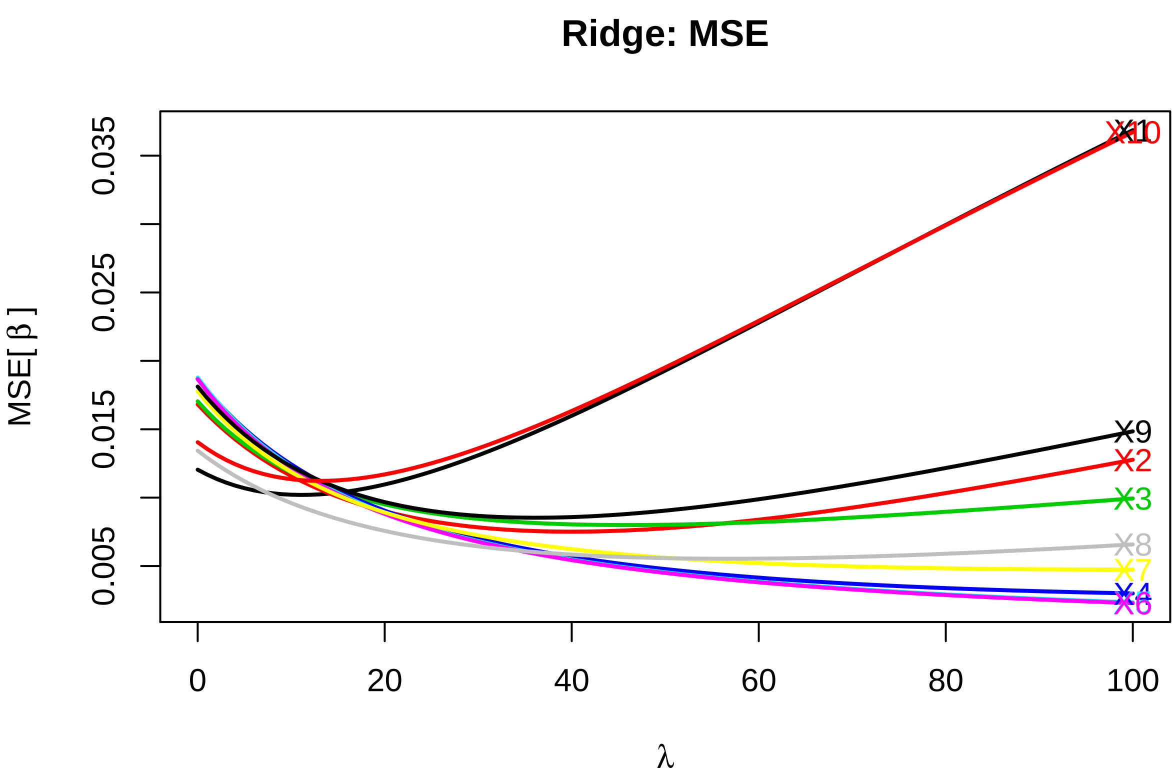
```r
# Plots
###########################

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")
```



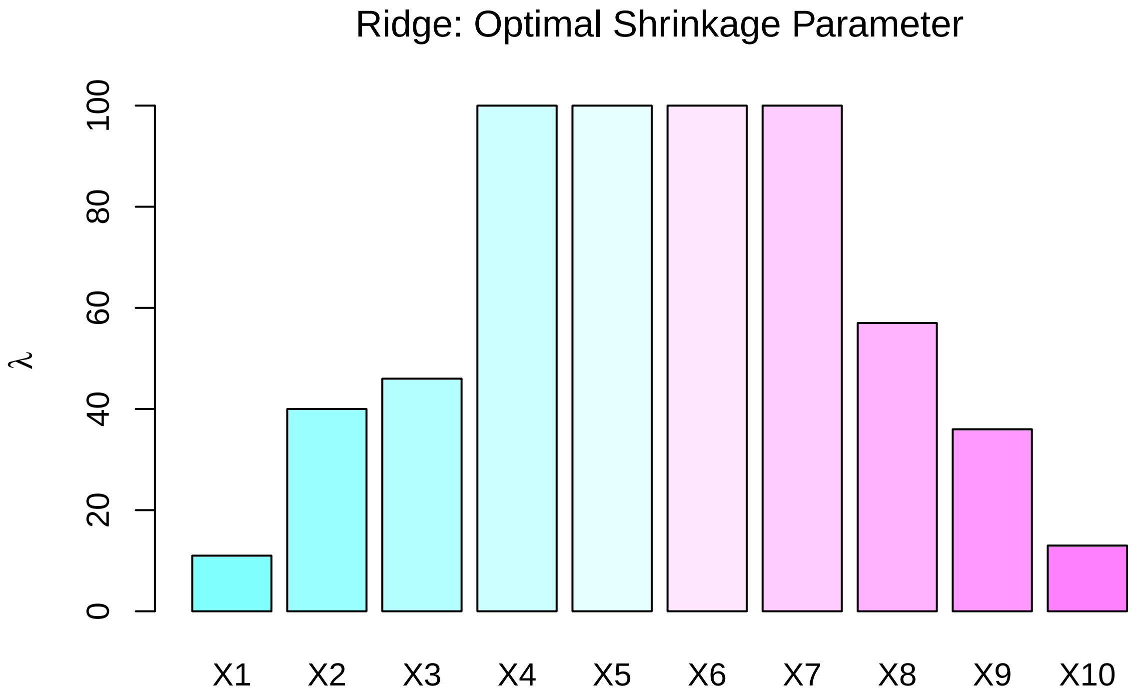

```r
# Variance

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



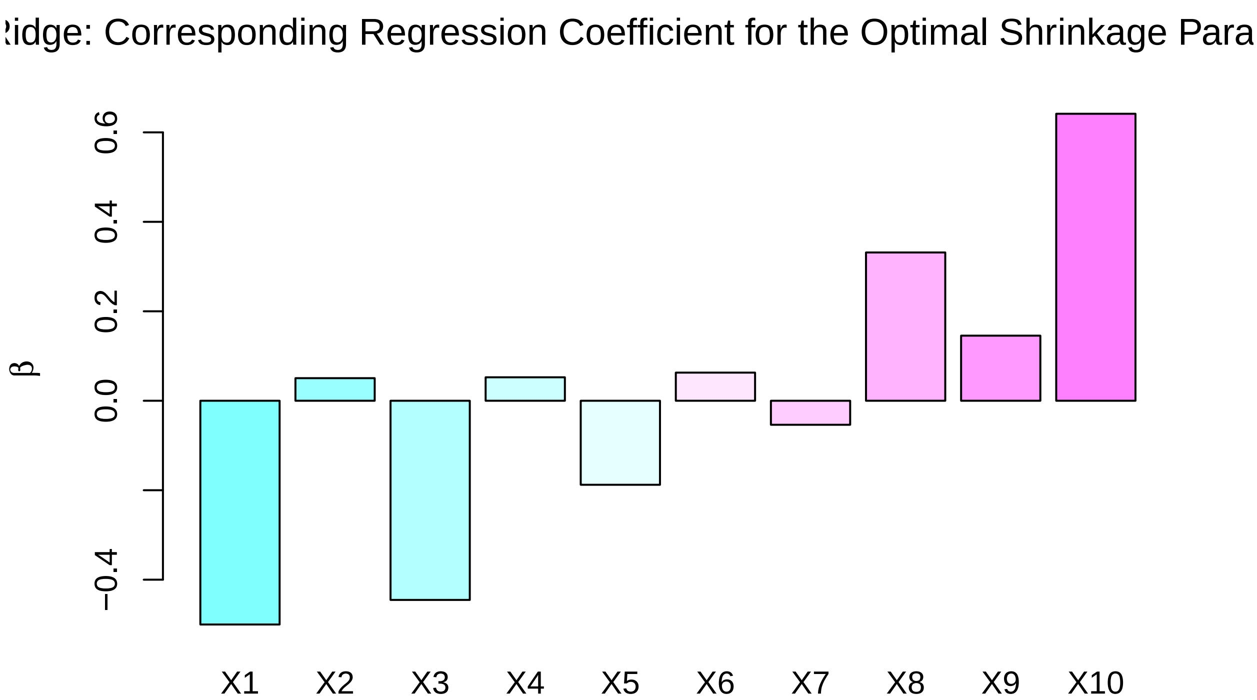

```r
# MSE

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



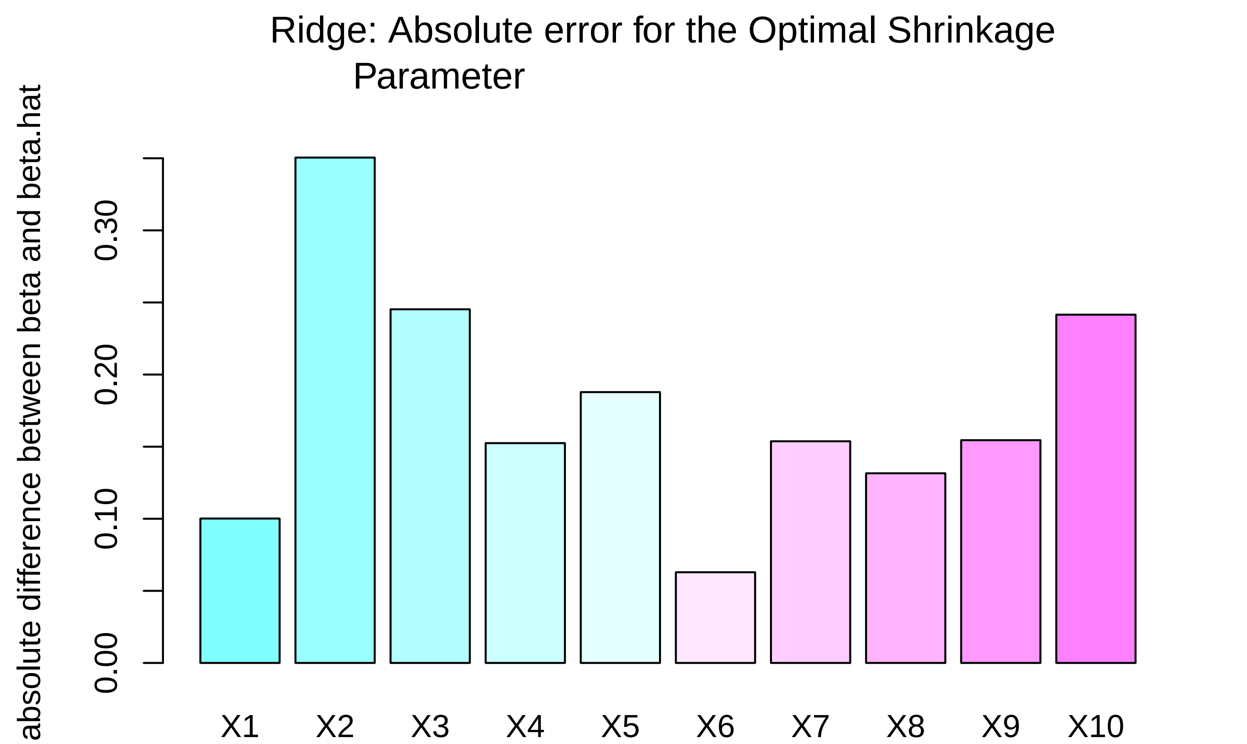
```
############################
# Optimal lambda with beta
############################

index <- apply(ridge_perf$mse, 2, which.min)
lambda_optimal <- lambda[index]

beta_hat <- rep(NA,J)
beta <- rep(NA,J)
abs_error <- rep(NA,J)
for(j in 1:J){
   beta_hat[j] <- ridge_perf$beta.hat[index[j],j]
   beta[j] <- Beta[j]
   abs_error[j] <- abs(beta_hat[j] - beta[j])
 }

df <- data.frame(lambda_optimal, beta_hat, 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
```




```
barplot(df$beta_hat, main = expression("Ridge: Corresponding Regression Coefficient for the Optimal Shri
```




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




```
df
```

```
##     lambda_optimal    beta_hat beta  abs_error
## X1              11 -0.50015644 -0.4 0.10015644
## X2              40  0.05046947 -0.3 0.35046947
```

```
## X3                46 -0.44524538 -0.2 0.24524538
## X4               100  0.05243988 -0.1 0.15243988
## X5               100 -0.18781601  0.0 0.18781601
## X6               100  0.06289929  0.0 0.06289929
## X7               100 -0.05372434  0.1 0.15372434
## X8                57  0.33151644  0.2 0.13151644
## X9                36  0.14551544  0.3 0.15448456
## X10               13  0.64153257  0.4 0.24153257
```

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 $B = 1000$ times (Step 3). So, we obtain 1000 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.

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

$\hat{\text{Cov}}[\hat{\beta}_n^{\text{LASSO}}|\mathbf{X}_n]$.

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

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).

Collaborators & Resources

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*

Tommy Carpenito