LASSO/Ridge/Elastic Net

Kevin Benac

19/10/2017

```
library(MASS)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```
library(ggplot2)
set.seed(1)
```

```
## Ridge regression
myRidge <- function(x,y,x.new=NULL,intercept=TRUE,scale=FALSE,lambda=0)
  {
    n <- nrow(x)
    J <- ncol(x)
    df <- mse <- rep(NA,length(lambda))
    if(intercept)
      beta.hat <- matrix(NA,length(lambda),J+1)
    else
      beta.hat <- matrix(NA,length(lambda),J)
    var <- matrix(NA,length(lambda),J)
    cov <- array(NA,c(length(lambda),J,J))
    y.hat <- e <- matrix(NA,length(lambda),n)
    y.new <- NULL

    xx <- scale(x,center=TRUE,scale=scale)
    beta0.hat <- mean(y)

    for(l in 1:length(lambda))
      {
        a <- solve(crossprod(xx)+lambda[l]*diag(J))
        if(intercept)
          {
            df[l] <- sum(diag(xx%*%a%*%t(xx)))+1
            beta.hat[l,] <- c(beta0.hat,a%*%crossprod(xx,y))
            y.hat[l,] <- beta0.hat + xx%*%beta.hat[l,-1]
          }
        else
          {
            df[l] <- sum(diag(xx%*%a%*%t(xx)))
            beta.hat[l,] <- a%*%crossprod(xx,y)
            y.hat[l,] <- xx%*%beta.hat[l,]
          }
```

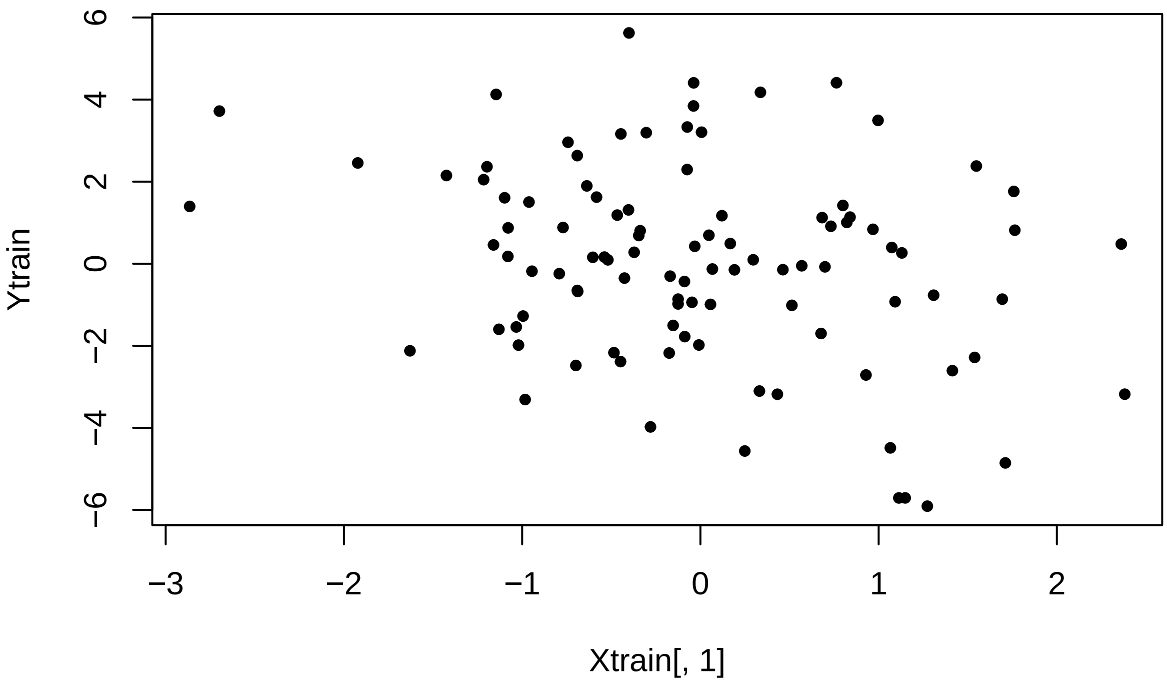
```r
        e[l,] <- y-y.hat[l,]
        mse[l] <- mean(e[l,]^2)
        cov[l,,] <- (mse[l]*n/(n-df[l]))*a%*%crossprod(xx)%*%a
        var[l,]  <- diag(cov[l,,])
      }

    if(!is.null(x.new))
      {
        xx.new <- scale(x.new,center=TRUE,scale=scale)
        if(intercept)
          xx.new <- cbind(1,xx.new)
        y.new <- beta.hat%*%t(xx.new)
      }

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

## 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
  }

## 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=FALSE,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)))}) + intercept
    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*intercept

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

  }

## Plots of regression coefficients, bias, variance, and MSE vs. shrinkage parameter or degrees of free
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)
  }
```

##a) Simulation model.

```
sigma = 2
rho = 0.5
J = 10
n = 100

beta = c(-J/2 + (1:(J/2)), (0:(J/2 -1)))/J

Gamma = toeplitz(rho^c(0:9))
```

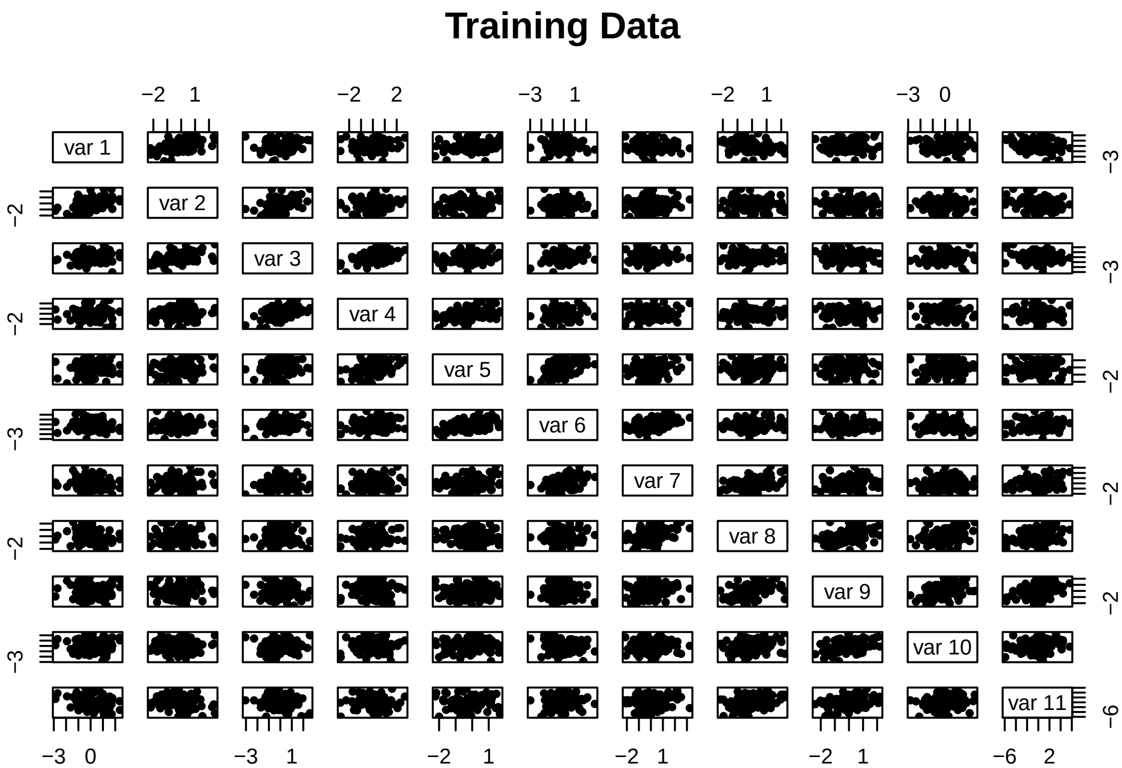
```r
Xtrain <- mvrnorm(n, mu = rep(0,J), Sigma= Gamma)

Ytrain = Xtrain%*%beta + sigma*rnorm(n)

Xtest <- mvrnorm(1000, mu = rep(0,J), Sigma= Gamma)

Ytest = Xtest%*%beta + sigma*rnorm(1000)

plot(Xtrain[,1],Ytrain, pch = 20)
```




```r
corr <- c()
for(i in 1:J){
  corr[i]<- cor(Ytrain, Xtrain[,i])
}


corr
```

```
##  [1] -0.281452173 -0.239881584 -0.187438575 -0.028063697  0.005361358
##  [6]  0.101699252  0.187535325  0.314692572  0.385426832  0.258685071
```

```r
pairs(cbind(Xtrain, Ytrain),main="Training Data", pch = 20)
```

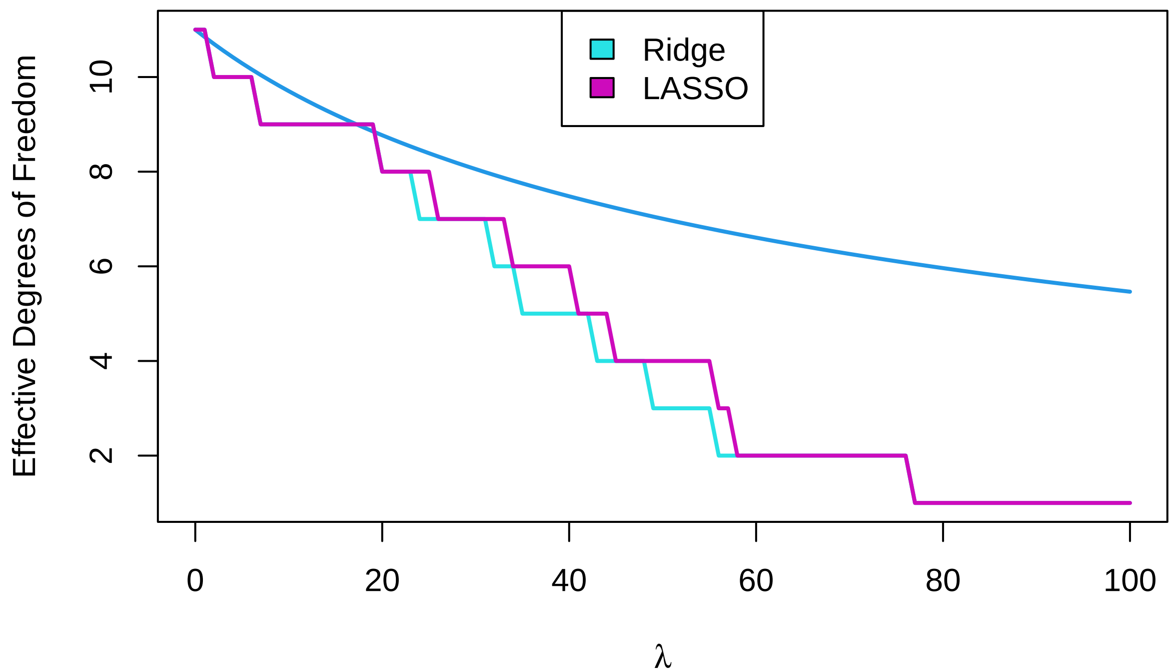


What numerical and graphical summaries for the learning set?

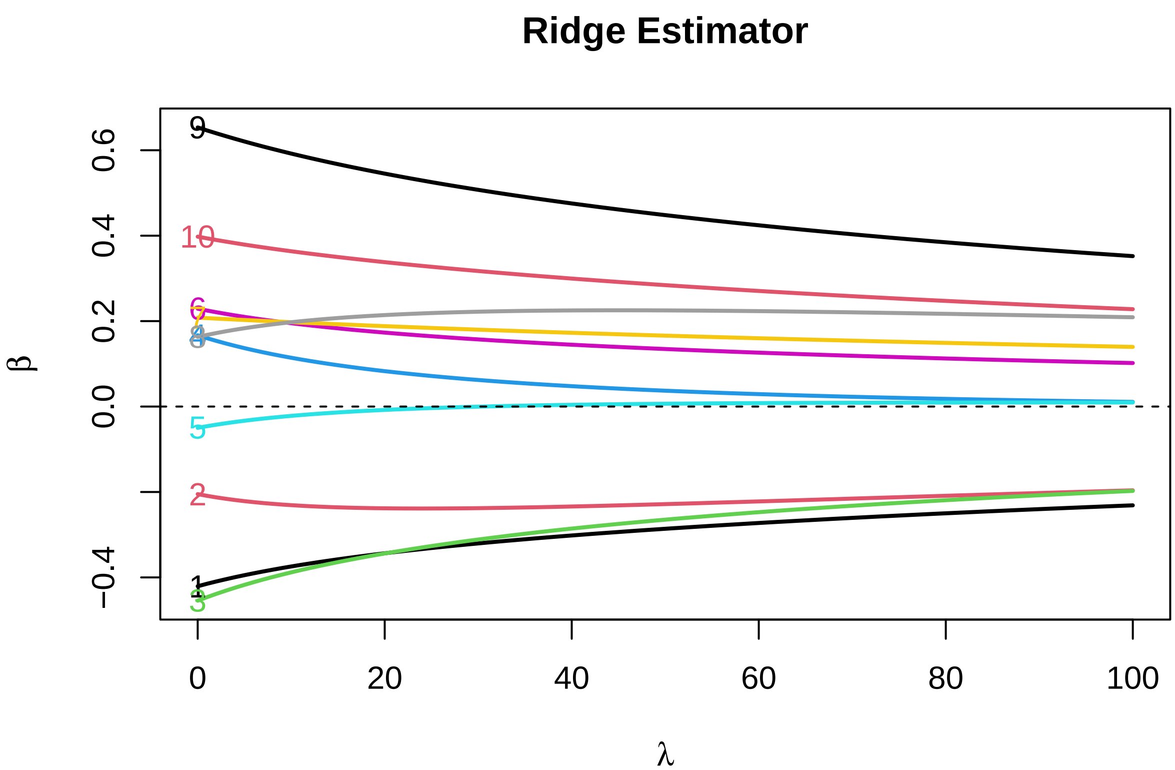
b) Elastic net regression on learning set.

```
Lambda <- 0:100

#Xtr = apply(Xtrain, FUN = "scale", MARGIN =2) #not necessary since glmnet automatically standardize co

#fitridge<- glmnet(x= Xtrain, y= Ytrain, family= "gaussian", lambda = Lambda*(1/nrow(Xtrain)), alpha= 0
#fitlasso<- glmnet(x= Xtr, y= Ytrain, family= "gaussian", lambda = Lambda*(1/(2*nrow(Xtrain))), alpha=
#fitenet<- glmnet(x= Xtrain, y= Ytrain, family= "gaussian", lambda = Lambda*(3/(2*nrow(Xtrain))), alpha

fitridge<- myGlmnet(x = Xtrain, y = Ytrain, x.new=Xtest, intercept=TRUE,
                    scale=TRUE, alpha=0, lambda=Lambda*(1/nrow(Xtrain)))
fitlasso <- myGlmnet(x = Xtrain, y = Ytrain, x.new=Xtest,intercept=TRUE,
                     scale=TRUE,alpha=1,lambda= Lambda*(1/(2*nrow(Xtrain))))
fitenet <- myGlmnet(x = Xtrain, y = Ytrain, x.new=Xtest,intercept=TRUE,
                    scale=TRUE,alpha=1/3,lambda= Lambda*(3/(2*nrow(Xtrain))))







#svdX<- svd(Xtr)$d
#df.r <- function(l){
#sum({svdX^2}/{(l+ svdX)^2}) +1
#}
```

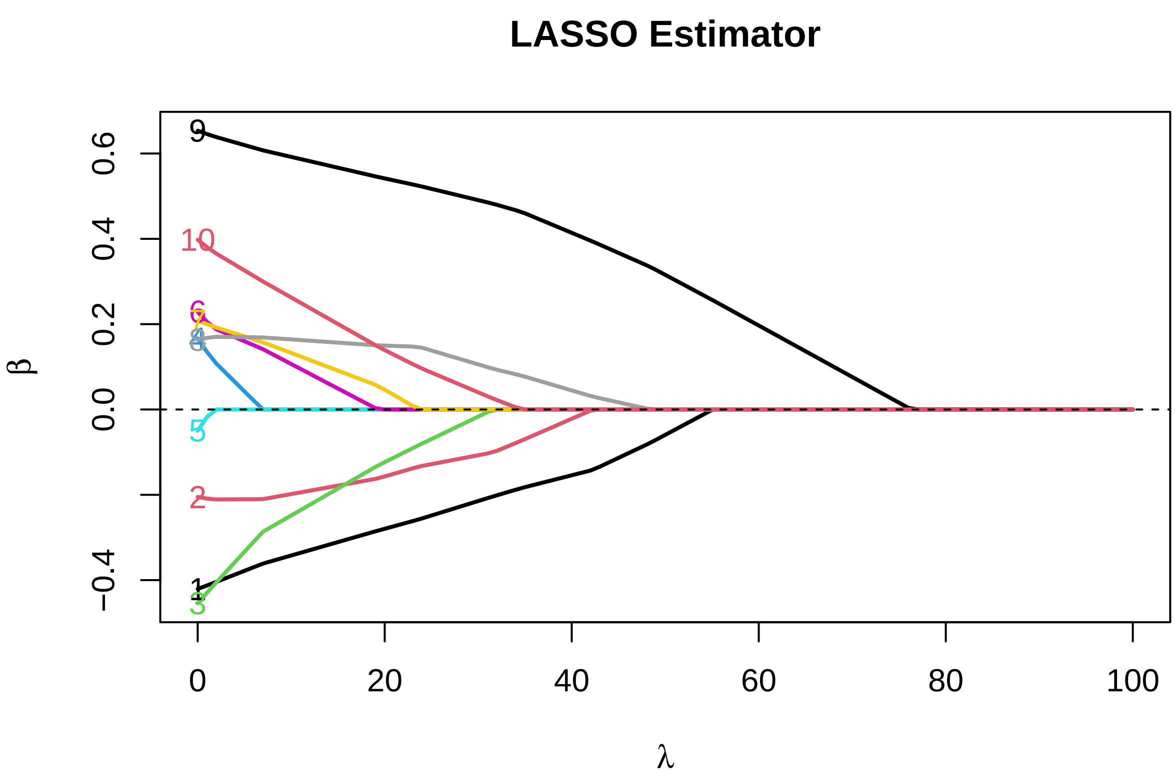
```r
#df.ridge = sapply(Lambda, FUN = "df.r")
#df.lasso = 1+ fitlasso$df
#df.enet = 1+fitenet$df

#plot(Lambda, df.ridge, type="l", ylab = "Effective Degrees of Freedom"); lines(x=Lambda, y=df.lasso, c

matplot(Lambda,cbind(fitridge$df,fitlasso$df, fitenet$df),type="l",lwd=2,
        lty=1,col=4:6,xlab=expression(lambda), ylab = "Effective Degrees of Freedom");
legend("top",c("Ridge","LASSO"),fill=5:6)
```




```r
myPlotBeta(Lambda,fitridge$beta.hat[,-1],type="l",lwd=2,
           ylab=expression(hat(beta)),labels=1:10, xlab= expression(lambda),
           main = "Ridge Estimator")
```

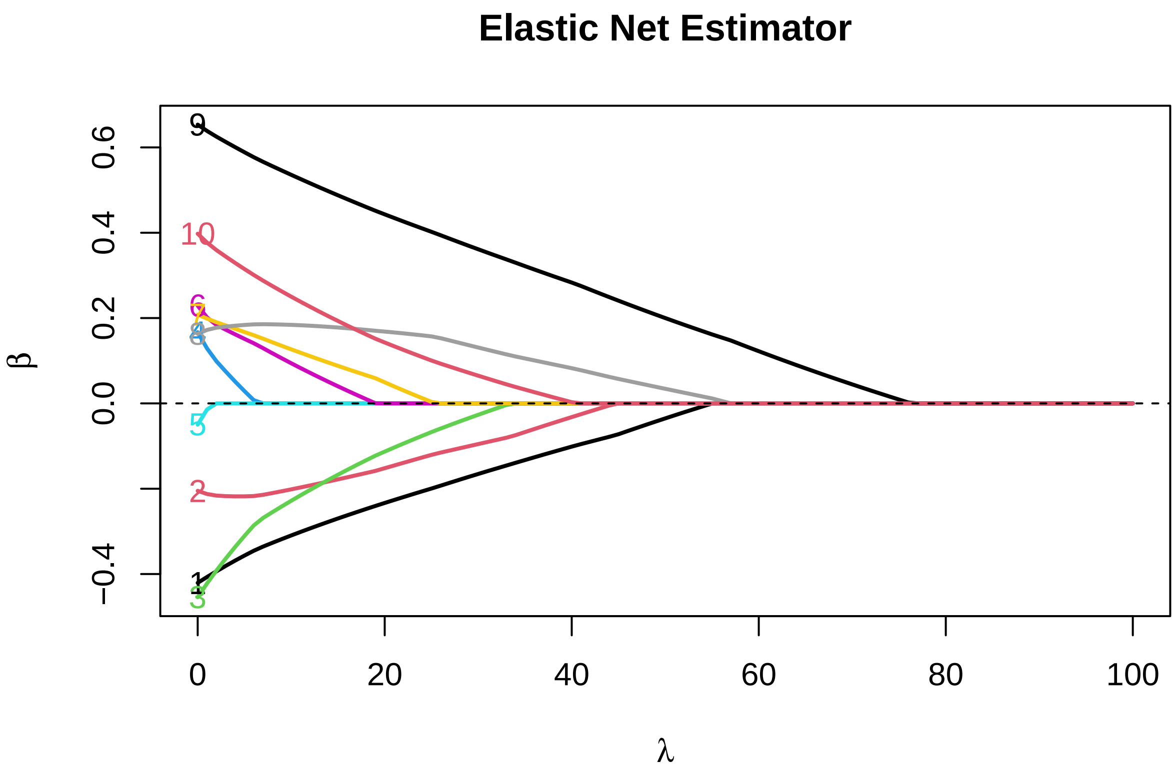



```
myPlotBeta(Lambda,fitlasso$beta.hat[,-1],type="l",lwd=2,
           ylab=expression(hat(beta)),labels=1:10,
           xlab= expression(lambda), main = "LASSO Estimator")
```



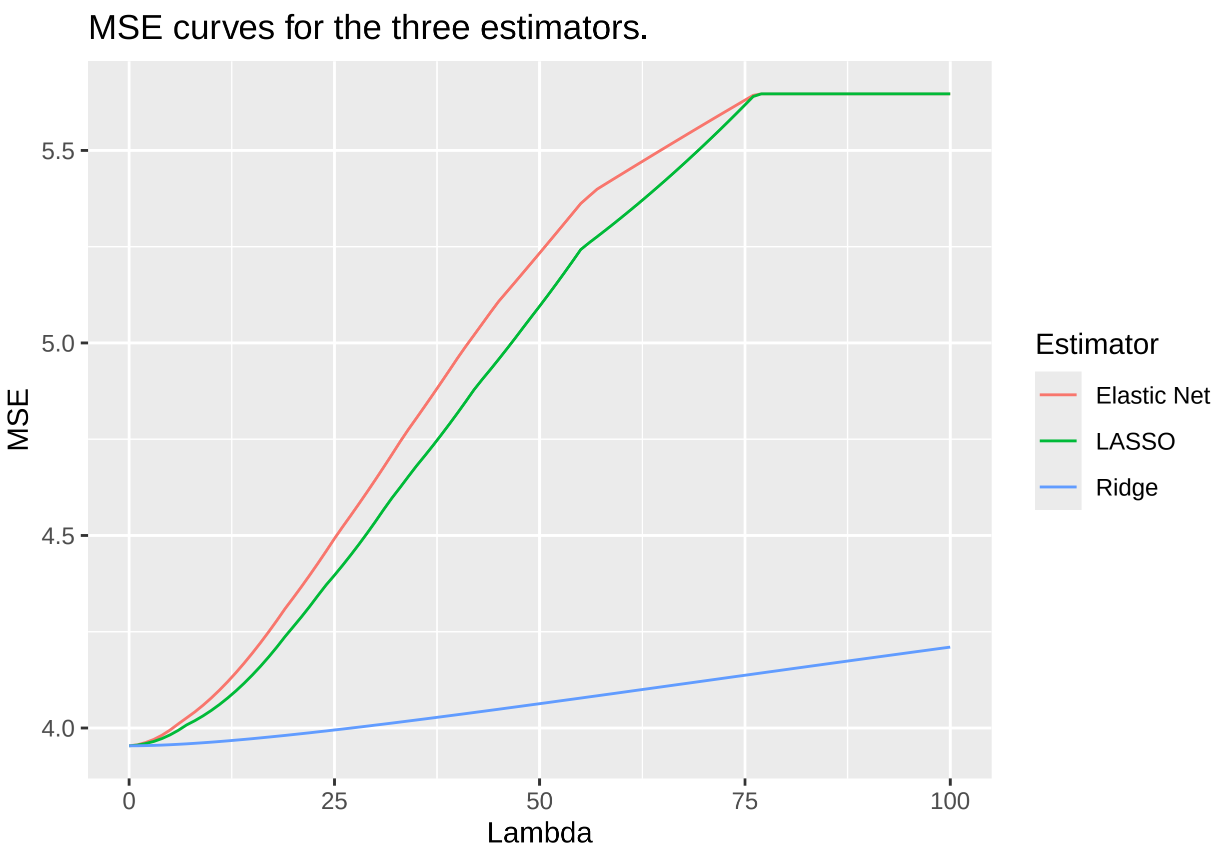

```
myPlotBeta(Lambda,fitenet$beta.hat[,-1],type="l",lwd=2,
           ylab=expression(hat(beta)),labels=1:10,
```

```r
        xlab= expression(lambda), main ="Elastic Net Estimator")
```




```r
MSE<- matrix(rep(NA, 3*(length(Lambda))), ncol= 3)
colnames(MSE) = c("Ridge", "LASSO", "Elastic Net")


#MSE[,1]<- colMeans((Ytrain- Xtr%*%bridge$beta)^2)
#MSE[,2]<- colMeans((Ytrain- Xtr%*%blasso$beta)^2)
#MSE[,3]<- colMeans((Ytrain- Xtr%*%benet$beta)^2)


#yridge_hat<- predict(fitridge, newx = Xtrain, s = Lambda*(1/nrow(Xtrain)))
#ylasso_hat<- predict(fitlasso, newx = Xtrain, s = Lambda*(1/(2*nrow(Xtrain))))
#yenet_hat<- predict(fitenet, newx = Xtrain, s = Lambda*(3/(2*nrow(Xtrain))))

#yridge_hat<- scale(Xtrain %*% fitridge$beta, center = -fitridge$a0, scale = F)
#ylasso_hat<- scale(Xtrain %*% fitlasso$beta, center = -fitlasso$a0, scale = F)
#yenet_hat<- scale(Xtrain %*% fitenet$beta, center = -fitenet$a0, scale = F)

#for(j in 1:length(Lambda)){
#  MSE[j,1]<- mean((Ytrain - yridge_hat[,j])^2)
#  MSE[j,2]<- mean((Ytrain - ylasso_hat[,j])^2)
#  MSE[j,3]<- mean((Ytrain - yenet_hat[,j])^2)
#}

#e_ridge <- scale(t(yridge_hat),center= as.numeric(Ytrain), scale=FALSE)
#e_lasso <- scale(t(ylasso_hat),center= as.numeric(Ytrain),scale=FALSE)
#e_net <- scale(t(yenet_hat),center= as.numeric(Ytrain),scale=FALSE)
```

```r
#MSE[,1]<- rowMeans(e_ridge^2)
#MSE[,2]<- rowMeans(e_lasso^2)
#MSE[,3]<- rowMeans(e_net^2)



#matplot( x=Lambda, y=MSE, type= 'l', xlab = "lambda")

MSE[,1]<- fitridge$mse
MSE[,2]<- fitlasso$mse
MSE[,3]<- fitenet$mse

plotdf<- data.frame(MSE = c(MSE[,1], MSE[,2], MSE[,3]), Lambda= rep(Lambda, 3),
                    Estimator = c(rep("Ridge",length(Lambda)),
                                  rep("LASSO", length(Lambda)), rep("Elastic Net", length(Lambda)) ) )

ggplot(plotdf, aes(x = Lambda, y = MSE, color= Estimator)) + geom_line()+
  ggtitle("MSE curves for the three estimators.")
```




```r
lambda_ridge = Lambda[which.min(MSE[,1])]
lambda_LASSO = Lambda[which.min(MSE[,2])]
lambda_enet = Lambda[which.min(MSE[,3])]

lambda_ridge
```

```
## [1] 0
```

```r
lambda_LASSO
```

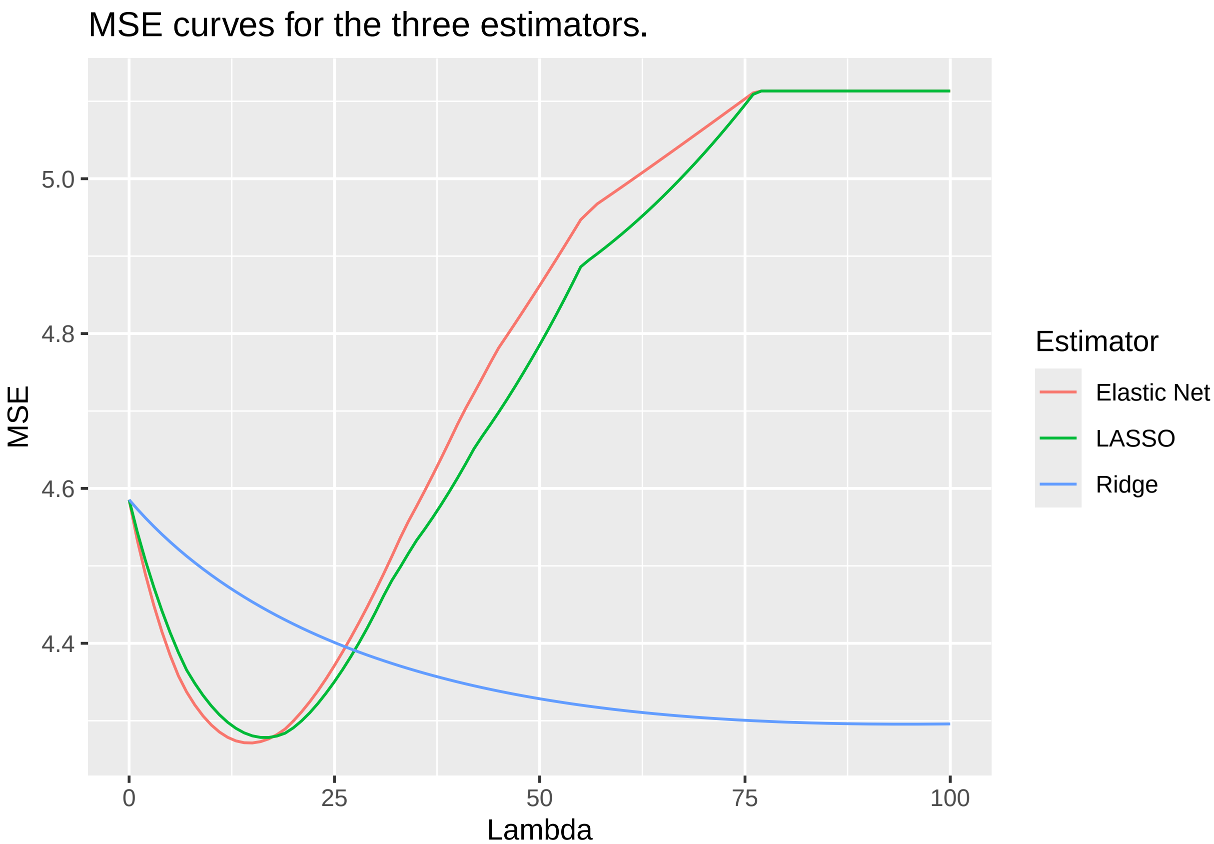
```
## [1] 0
```

```r
lambda_enet
```

```
## [1] 0
```

Performance assessment on test set

```r
#yridge_hat<- predict(fitridge, newx = Xtest, s = Lambda*(1/nrow(Xtest)))
#ylasso_hat<- predict(fitlasso, newx = Xtest, s = Lambda*(1/(2*nrow(Xtest))))
#yenet_hat<- predict(fitenet, newx = Xtest, s = Lambda*(3/(2*nrow(Xtest))))



#e_ridge <- scale(t(yridge_hat),center= as.numeric(Ytest), scale=FALSE)
#e_lasso <- scale(t(ylasso_hat),center= as.numeric(Ytest),scale=FALSE)
#e_net <- scale(t(yenet_hat),center= as.numeric(Ytest),scale=FALSE)

#MSE[,1]<- rowMeans(e_ridge^2)
#MSE[,2]<- rowMeans(e_lasso^2)
#MSE[,3]<- rowMeans(e_net^2)


MSE[,1] <- rowMeans(scale(fitridge$y.new,center=Ytest,scale=FALSE)^2)
MSE[,2] <- rowMeans(scale(fitlasso$y.new,center=Ytest,scale=FALSE)^2)
MSE[,3] <- rowMeans(scale(fitenet$y.new,center=Ytest,scale=FALSE)^2)
lambda_ridge <- which.min(MSE[,1])
lambda_LASSO <- which.min(MSE[,2])
lambda_enet <- which.min(MSE[,3])

plotdf<- data.frame(MSE = c(MSE[,1], MSE[,2], MSE[,3]), Lambda= rep(Lambda, 3),
                    Estimator = c(rep("Ridge",length(Lambda)),
                                  rep("LASSO", length(Lambda)), rep("Elastic Net", length(Lambda)) ) )

ggplot(plotdf, aes(x = Lambda, y = MSE, color= Estimator)) + geom_line()+
ggtitle("MSE curves for the three estimators.")
```




```r
lambda_ridge = Lambda[which.min(MSE[,1])]
lambda_LASSO = Lambda[which.min(MSE[,2])]
lambda_enet = Lambda[which.min(MSE[,3])]

lambda_ridge
```

```
## [1] 94
```

```r
lambda_LASSO
```

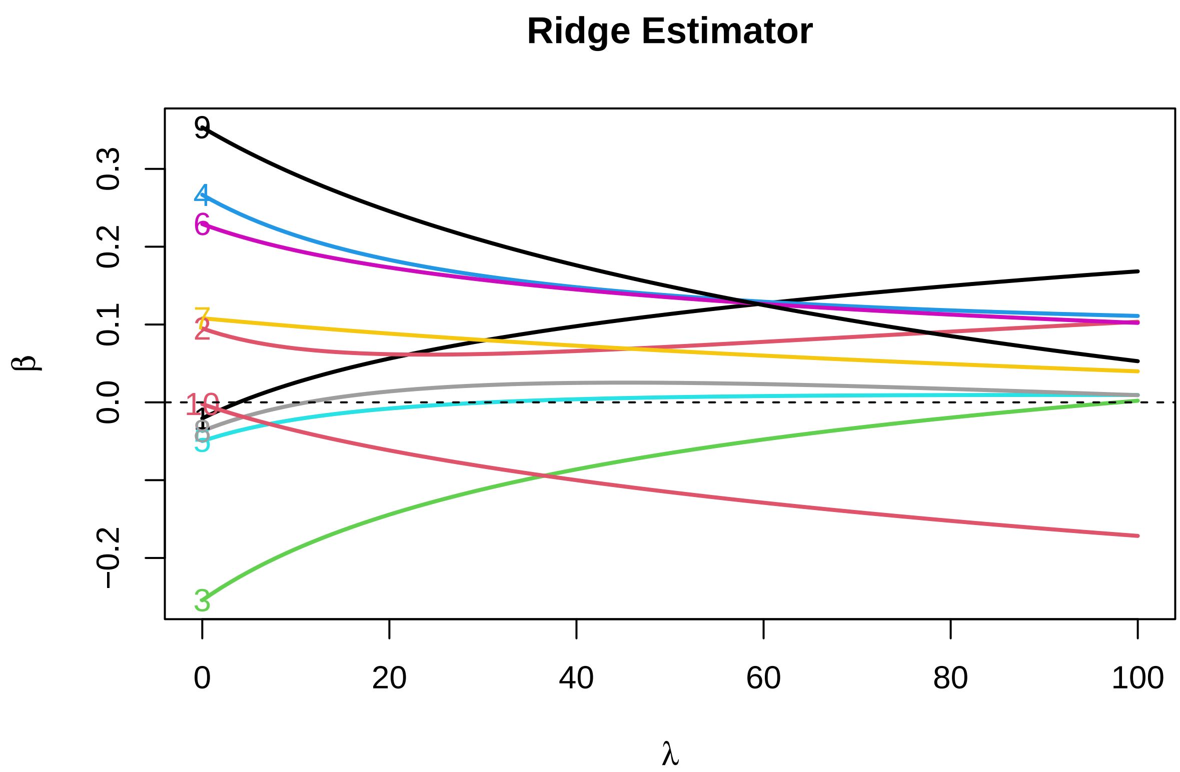
```
## [1] 17
```

```r
lambda_enet
```

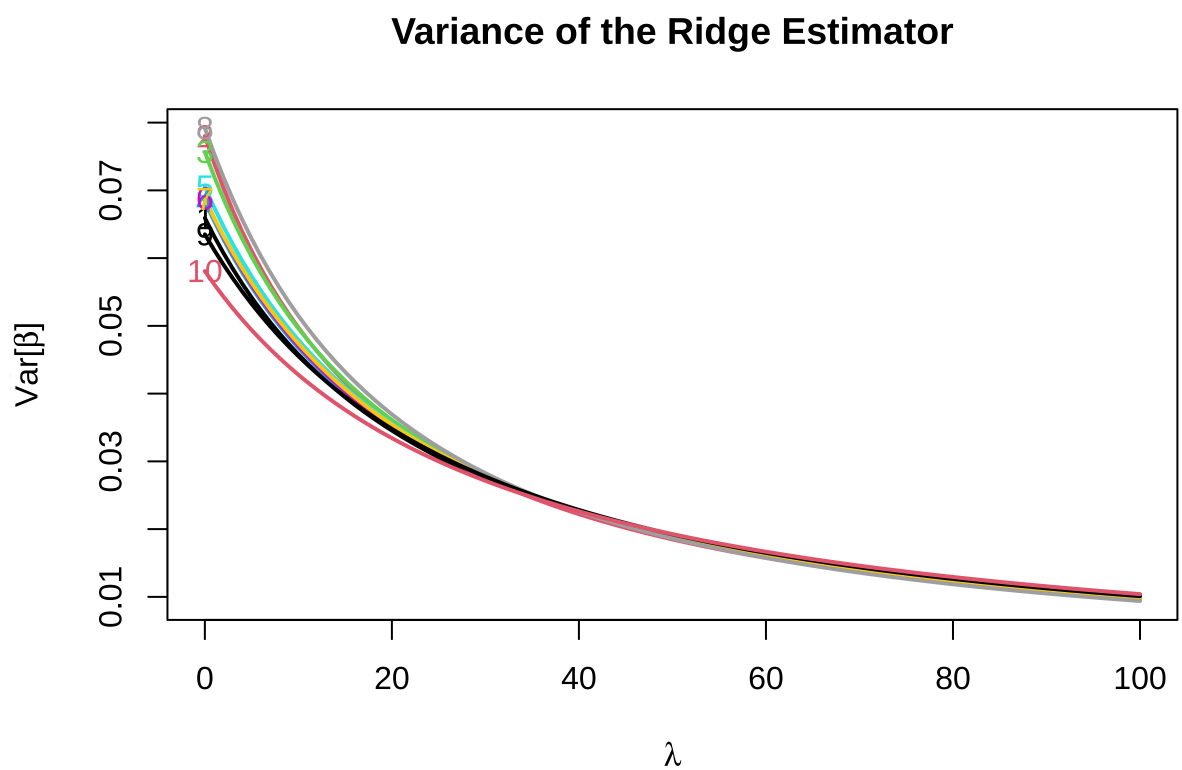
```
## [1] 15
```

Ridge regression: Bias, variance, and mean squared error of estimated regression coeffi- cients

```r
fitridge2<- myRidge(x = Xtrain, y = Ytrain, x.new=Xtest, intercept=TRUE,
                    scale=TRUE, lambda=Lambda)
biasRidge <- scale(fitridge2$beta.hat[,-1], center = beta, scale= F)
myPlotBeta(Lambda,biasRidge,ylab=expression(paste(hat(beta))), labels=1:10,
           xlab = expression(lambda), main = "Ridge Estimator")
```

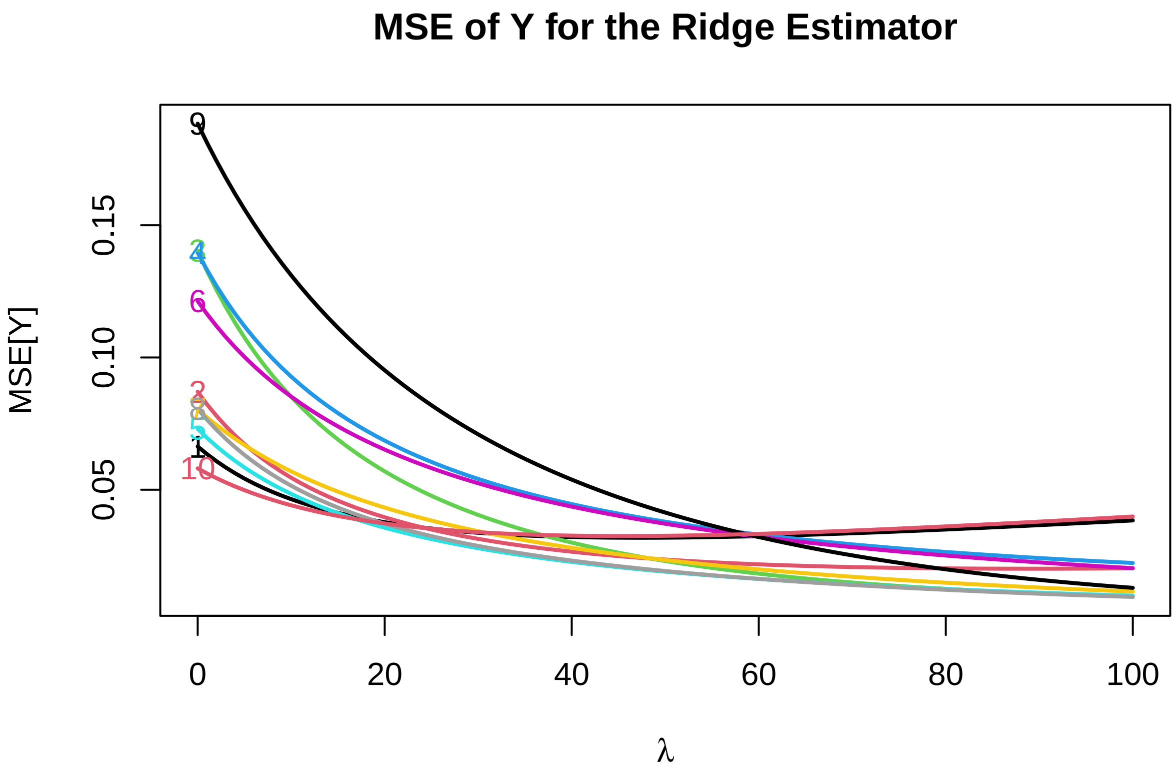



```
myPlotBeta(Lambda,fitridge2$var,ylab=expression(paste(hat(Var),"[",hat(beta),"]",sep="")),
          labels=1:10, xlab = expression(lambda), main = "Variance of the Ridge Estimator")
```




```
biasRidge <- scale(fitridge2$beta.hat[,-1], center = beta, scale= F)
varRidge<- fitridge2$var
```

```
MSERridge <- biasRidge^2 + varRidge
myPlotBeta(Lambda,MSERridge ,ylab=expression(paste("MSE","[",Y,"]",sep="")),
           labels=1:10, xlab = expression(lambda), main = "MSE of Y for the Ridge Estimator")
```

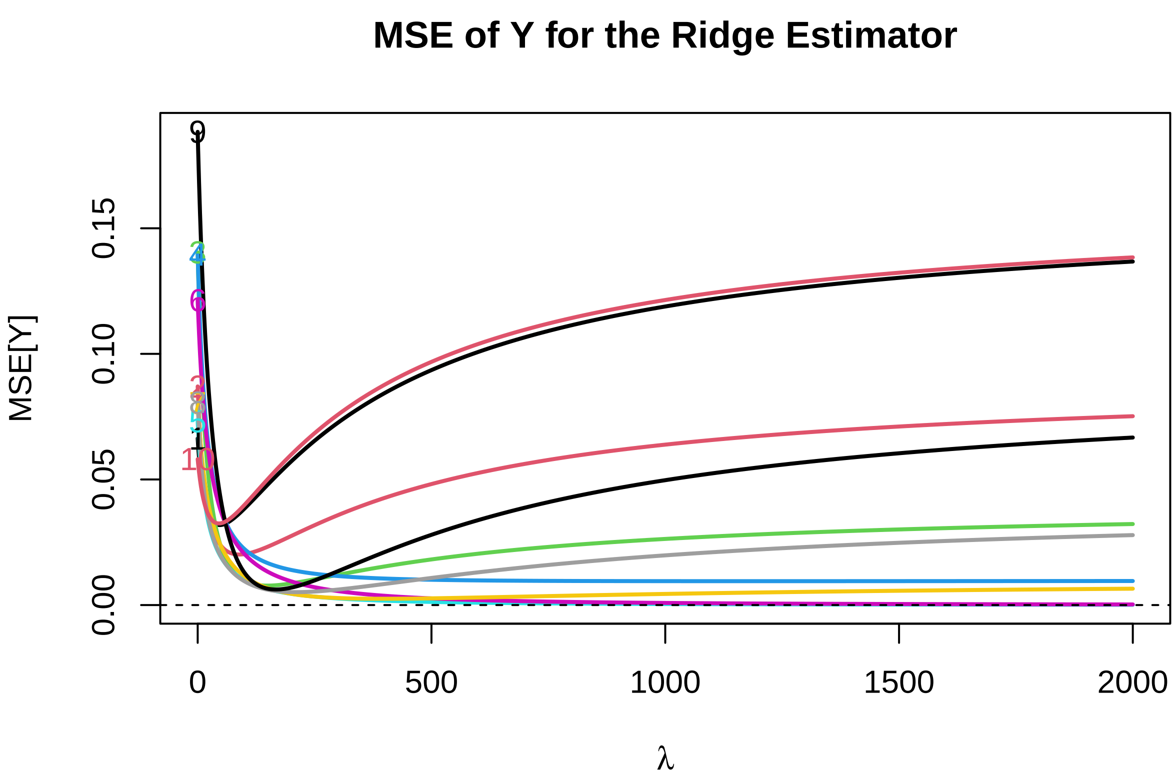



```
apply(MSERridge, FUN = "which.min", MARGIN = 2)
```

```
##  [1]  48  90 101 101 101 101 101 101 101  46
```

We observe that for 7 of the coefficients, the grid of λ we chose does not enable us to tell what value of the shrinkage parameter minimizes the MSE. Therefore, we are going to consider a larger grid, say $\lambda = (1 : 2000)$.

```
fitridge2<- myRidge(x = Xtrain, y = Ytrain, x.new=Xtest, intercept=TRUE,scale=TRUE, lambda=(0:2000))

biasRidge <- scale(fitridge2$beta.hat[,-1], center = beta, scale= F)
varRidge<- fitridge2$var

MSERridge <- biasRidge^2 + fitridge2$var
myPlotBeta(0:2000,MSERridge ,ylab=expression(paste("MSE","[",Y,"]",sep="")),
           labels=1:10, xlab = expression(lambda), main = "MSE of Y for the Ridge Estimator")
```




```
apply(MSERridge, FUN = "which.min", MARGIN = 2)-1
```

```
## [1]   47   89  149 1187 2000 2000  396  211  168   45
```

We note that for two of the coefficients, the optimal λ seem to be bigger than 2000, which is completely normal since the true value of β is zero so in theory, the optimal λ for these coefficients should be ∞. For the other coefficients, we see that the optimal value of λ is bigger when β is smaller, which completely makes sense.

LASSO regression: Bias, variance, and mean squared error of estimated regression coeffi- cients

```
summaryLASSO <- function(nsim, lambda= Lambda){
  Beta <- array(rep(NA, nsim*length(lambda)*J), dim = c(nsim, length(lambda), J))
  eLasso<- array(rep(NA, nsim*length(lambda)*J), dim = c(nsim, length(lambda), J))
  for(i in 1:sim){
    Y = Xtrain%*%beta + sigma*rnorm(n)
    fitlasso<- glmnet(x= Xtrain, y= Y, family= "gaussian", lambda = Lambda, alpha= 1)
    Beta[i,,]<- fitlasso$beta
    ylasso_hat<- predict(fitlasso, newx = XTest, s = Lambda)
    e_lasso[i,,] <- scale(t(ylasso_hat),center= as.numeric(Y), scale=FALSE)
  }

}
```