Jakob Lovato
Will Bliss
STAT 760
HW 3

1. Let $I_{p\times p}$ be a $p\times p$ identity matrix and $0_{p\times 1}$ be a $p\times 1$ vector of 0's. Then let

$$\hat{X} = \begin{pmatrix} X \\ \sqrt{\lambda}\, I_{p\times p} \end{pmatrix}$$

and

$$\hat{y} = \begin{pmatrix} y \\ 0_{p\times 1} \end{pmatrix}.$$

For the least squares method, we have

$$\hat{\beta} = (\underbrace{\hat{X}^T\hat{X}}_{1})^{-1}\underbrace{\hat{X}^T\hat{y}}_{2}.$$

Through evaluating equation 1, we get

$$\hat{X}^T\hat{X} = \begin{pmatrix} X^T & \sqrt{\lambda}\, I_{p\times p} \end{pmatrix}\begin{pmatrix} X \\ \sqrt{\lambda}\, I_{p\times p} \end{pmatrix}$$
$$= X^TX + \lambda I_{p\times p}.$$

For equation 2, we get

$$\hat{X}^T\hat{y} = \begin{pmatrix} X^T & \sqrt{\lambda}\, I_{p\times p} \end{pmatrix}\begin{pmatrix} y \\ 0_{p\times 1} \end{pmatrix}$$
$$= X^Ty.$$

Thus

$$\hat{\beta} = (X^TX + \lambda I_{p\times p})^{-1}X^Ty,$$

which is the equation for Ridge Regression estimates.

Stat 760 Homework 3

Will Bliss and Jakob Lovato

2/15/2022

```
data <- read.delim("/Users/jakoblovato/Desktop/Stat 760/HW 3/prostate.txt", header = T)
data <- data[,-1]
train <- data[which(data$train == TRUE),]
test <- data[which(data$train == FALSE),]
```

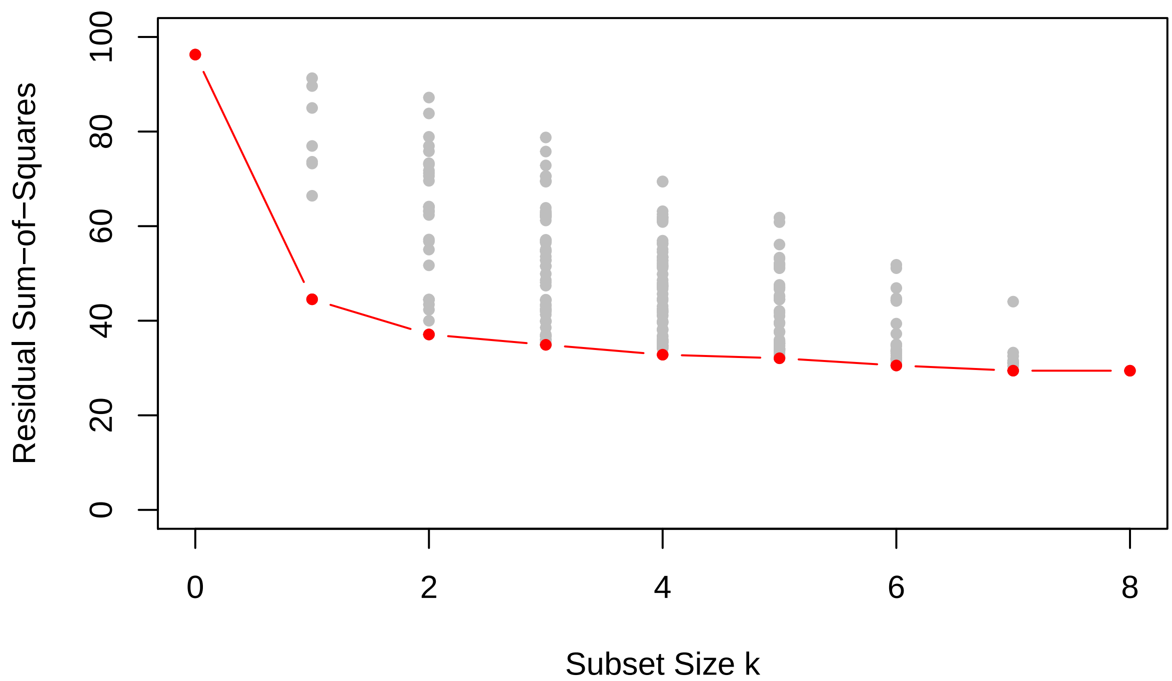
Question 2

```
RSS0 <- sum( (train$lpsa - mean(train$lpsa))^2 )

RSS1 <- c()
for(i in 1:(ncol(train) - 2)){
  betaZero <- matrix(1, ncol = 1, nrow(train))
  X <- as.matrix(cbind(betaZero, train[,i]))
  XTXInverse <- solve(t(X) %*% X)
  betas <- XTXInverse %*% t(X) %*% train$lpsa
  YHat <- X %*% betas
  RSS1 <- c(RSS1, sum((train$lpsa - YHat) ^ 2))
}

RSS2 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
    if(j > i){
      tempData <- train[,c(i,j)]
      betaZero <- matrix(1, ncol = 1, nrow(tempData))
      X <- as.matrix(cbind(betaZero, tempData))
      XTXInverse <- solve(t(X) %*% X)
      betas <- XTXInverse %*% t(X) %*% train$lpsa
      YHat <- X %*% betas
      RSS2 <- c(RSS2, sum((train$lpsa - YHat) ^ 2))
    }
  }
}

RSS3 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
    for (k in 1:(ncol(train)-2)) {
    if((k > j) & (j > i)){
      tempData <- train[,c(i,j,k)]
      betaZero <- matrix(1, ncol = 1, nrow(tempData))
```

```
      X <- as.matrix(cbind(betaZero, tempData))
      XTXInverse <- solve(t(X) %*% X)
      betas <- XTXInverse %*% t(X) %*% train$lpsa
      YHat <- X %*% betas
      RSS3 <- c(RSS3, sum((train$lpsa - YHat) ^ 2))
      }
    }
  }
}

RSS4 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
    for (k in 1:(ncol(train)-2)) {
      for (l in 1:(ncol(train)-2)) {
        if((l > k) & (k > j) & (j > i)){
          tempData <- train[,c(i,j,k,l)]
          betaZero <- matrix(1, ncol = 1, nrow(tempData))
          X <- as.matrix(cbind(betaZero, tempData))
          XTXInverse <- solve(t(X) %*% X)
          betas <- XTXInverse %*% t(X) %*% train$lpsa
          YHat <- X %*% betas
          RSS4 <- c(RSS4, sum((train$lpsa - YHat) ^ 2))
        }
      }
    }
  }
}

RSS5 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
    for (k in 1:(ncol(train)-2)) {
      for (l in 1:(ncol(train)-2)) {
        for (m in 1:(ncol(train)-2)) {
          if((m > l) & (l > k) & (k > j) & (j > i)){
            tempData <- train[,c(i,j,k,l,m)]
            betaZero <- matrix(1, ncol = 1, nrow(tempData))
            X <- as.matrix(cbind(betaZero, tempData))
            XTXInverse <- solve(t(X) %*% X)
            betas <- XTXInverse %*% t(X) %*% train$lpsa
            YHat <- X %*% betas
            RSS5 <- c(RSS5, sum((train$lpsa - YHat) ^ 2))
          }
        }
      }
    }
  }
}

RSS6 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
```

```r
    for (k in 1:(ncol(train)-2)) {
      for (l in 1:(ncol(train)-2)) {
        for (m in 1:(ncol(train)-2)) {
          for (n in 1:(ncol(train)-2)) {
            if((n > m) & (m > l) & (l > k) & (k > j) & (j > i)){
              tempData <- train[,c(i,j,k,l,m,n)]
              betaZero <- matrix(1, ncol = 1, nrow(tempData))
              X <- as.matrix(cbind(betaZero, tempData))
              XTXInverse <- solve(t(X) %*% X)
              betas <- XTXInverse %*% t(X) %*% train$lpsa
              YHat <- X %*% betas
              RSS6 <- c(RSS6, sum((train$lpsa - YHat) ^ 2))
            }
          }
        }
      }
    }
  }
}

RSS7 <- c()
for (i in 1:(ncol(train)-2)) {
  for (j in 1:(ncol(train)-2)) {
    for (k in 1:(ncol(train)-2)) {
      for (l in 1:(ncol(train)-2)) {
        for (m in 1:(ncol(train)-2)) {
          for (n in 1:(ncol(train)-2)) {
            for (o in 1:(ncol(train)-2)) {
              if((o > n) & (n > m) & (m > l) & (l > k) & (k > j) & (j > i)){
                tempData <- train[,c(i,j,k,l,m,n,o)]
                betaZero <- matrix(1, ncol = 1, nrow(tempData))
                X <- as.matrix(cbind(betaZero, tempData))
                XTXInverse <- solve(t(X) %*% X)
                betas <- XTXInverse %*% t(X) %*% train$lpsa
                YHat <- X %*% betas
                RSS7 <- c(RSS7, sum((train$lpsa - YHat) ^ 2))
              }
            }
          }
        }
      }
    }
  }
}

#RSS8
X <- as.matrix(cbind(betaZero, train[,1:8]))
XTXInverse <- solve(t(X) %*% X)
betas <- XTXInverse %*% t(X) %*% train$lpsa
YHat <- X %*% betas
RSS8 <- sum((train$lpsa - YHat) ^ 2)

minRSS <- c(min(RSS0), min(RSS1), min(RSS2), min(RSS3),
```

```r
            min(RSS4), min(RSS5), min(RSS6), min(RSS7), min(RSS8))

plot(0:8,minRSS, type = "b", col = "red", pch = 20, xlim = c(0, 8),
     ylim = c(0, 100), xlab = "Subset Size k", ylab = "Residual Sum-of-Squares")
points(rep(0, length(RSS0)), RSS0, pch = 20, col = 200)
points(rep(1, length(RSS1)), RSS1, pch = 20, col = 200)
points(rep(2, length(RSS2)), RSS2, pch = 20, col = 200)
points(rep(3, length(RSS3)), RSS3, pch = 20, col = 200)
points(rep(4, length(RSS4)), RSS4, pch = 20, col = 200)
points(rep(5, length(RSS5)), RSS5, pch = 20, col = 200)
points(rep(6, length(RSS6)), RSS6, pch = 20, col = 200)
points(rep(7, length(RSS7)), RSS7, pch = 20, col = 200)
points(rep(8, length(RSS8)), RSS8, pch = 20, col = 200)
points(0:8, minRSS, pch = 20, col = "red")
```



Question 3

```r
#Shuffle data
shuffled <- data[sample(nrow(data)),]
folds <- cut(seq(1,nrow(data)),breaks=10,labels=FALSE)

errors <- c()
foldSe <- c()
for(lambda in 0:8){
  foldErrors <- c()
  for(i in 1:10){
    indices <- which(folds == i)
    temp <- shuffled[-indices,]
    betaZero <- matrix(1, ncol = 1, nrow(temp))
    X <- as.matrix(cbind(betaZero, temp[,1:8]))
```

```r
    XTXLambdaInverse <- solve((t(X) %*% X) + diag(lambda, ncol(X)))
    betas <- XTXLambdaInverse %*% t(X) %*% temp$lpsa
    YHat <- X %*% betas

    betaZeroTest <- matrix(1, ncol = 1, nrow(shuffled[indices,]))
    XTest <- as.matrix(cbind(betaZeroTest, shuffled[indices,1:8]))
    YHatTest <- XTest %*% betas

    foldErrors <- c(foldErrors, mean((shuffled[indices,]$lpsa - YHatTest) ^ 2))
  }
  errors <- c(errors, mean(foldErrors))
  foldSe <- c(foldSe, sd(foldErrors)/sqrt(nrow(data)))
}

plot(0:8, errors, type = "b", col = "orange", pch = 19, xlab = "Lambda",
     ylab = "CV Error", ylim = c(0.5,0.6), main = "Ridge Regression")
for(i in 0:8){
  arrows(x0 = i, y0 = errors[i+1] - foldSe[i+1], x1 = i, y1 = errors[i+1] + foldSe[i+1],
         length=0.05, angle=90, code=3, col = "skyblue")
}
points(0:8, errors, col = "orange", pch = 19)
domain <- 0:8
abline(v = domain[which(errors == min(errors))], lty = 2, col = "purple")
abline(h = min(errors), lty = 2, col = "purple")
```