BIOST 527 Homework 2

Benjamin Stan

April 26, 2021

Question 1

(a)

The ridge regression optimization with an intercept is as follows:

$$\text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda||\beta||^2\right\}$$

This expression can equivalently be written

$$\begin{aligned}
||\vec{y}-\beta_0\vec{1}-X\beta||^2+\lambda||\beta||^2 &= ||(P+P^{\perp})(\vec{y}-\beta_0\vec{1}-X\beta)||^2+\lambda||\beta||^2\\
&= ||P(\vec{y}-\beta_0\vec{1}-X\beta)+P^{\perp}(\vec{y}-\beta_0\vec{1}-X\beta)||^2+\lambda||\beta||^2\\
&= ||P(\vec{y}-\beta_0\vec{1}-X\beta)||^2+||P^{\perp}(\vec{y}-\beta_0\vec{1}-X\beta)||^2\\
&+ 2(\vec{y}-\beta_0\vec{1}-X\beta)P^{\perp}P(\vec{y}-\beta_0\vec{1}-X\beta)+\lambda||\beta||^2\\
&\text{Note that} P^{\perp}P=(I-P)P=P-PP=0\\
&= ||P(\vec{y}-\beta_0\vec{1}-X\beta)||^2+||P^{\perp}(\vec{y}-\beta_0\vec{1}-X\beta)||^2+\lambda||\beta||^2\\
&= ||\frac{1}{N}\vec{1}\vec{1}^T\vec{y}-\frac{\beta_0}{N}\vec{1}\vec{1}^T\vec{1}+\frac{1}{N}\vec{1}\vec{1}^TX\beta||^2\\
&+||(I-\frac{1}{N}\vec{1}\vec{1}^T)\vec{y}-(I-\frac{1}{N}\vec{1}\vec{1}^T)\beta_0\vec{1}+(I-\frac{1}{N}\vec{1}\vec{1}^T)X\beta||^2+\lambda||\beta||^2\\
&\text{Note that}\frac{1}{N}\vec{1}\vec{1}^TX\beta=\vec{1}\big[\bar{x_1},\ldots,\bar{x_p}\big]\beta=\vec{1}\bar{x}^T\beta\\
&= ||\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta||^2+||\vec{y}-\bar{y}\vec{1}-(\beta_0\vec{1}-\beta_0\vec{1})-(X\beta-\vec{1}\bar{x}^T\beta)||^2+\lambda||\beta||^2\\
&= ||\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta||^2+||(\vec{y}-\bar{y}\vec{1})-(X-\vec{1}\bar{x}^T)\beta||^2+\lambda||\beta||^2
\end{aligned}$$

Consider the first term

$$\begin{aligned}
||\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta||^2 &= (\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta)^T(\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta)\\
&= (\bar{y}\vec{1}^T-\beta_0\vec{1}^T-\beta^T\bar{x}\vec{1}^T)(\bar{y}\vec{1}-\beta_0\vec{1}-\vec{1}\bar{x}^T\beta)\\
&= \bar{y}^2N-\bar{y}\beta_0N-\bar{y}N\bar{x}^T\beta-\bar{y}\beta_0N+\beta_0^2N+\beta_0N\bar{x}^T\beta-\bar{y}\beta^T\bar{x}N+\beta_0\beta^T\bar{x}N+N\beta^T\bar{x}\bar{x}^T\beta\\
&= \bar{y}^2N-2\bar{y}\beta_0N-2\bar{y}N\bar{x}^T\beta+\beta_0^2N+2\beta_0N\bar{x}^T\beta+N\beta^T\bar{x}\bar{x}^T\beta
\end{aligned}$$

Derive with respect to β_0 and set equal to zero

$$\begin{aligned}\frac{\partial}{\partial\beta_0} &= -2\bar{y}N + 2\beta_0 N + 2N\bar{x}^T\beta = 0\\ 2\beta_0 N &= 2\bar{y}N - 2N\bar{x}^T\beta\\ \hat{\beta}_0 &= \bar{y} - \bar{x}^T\beta\end{aligned}$$

(b)

We want to show that the $\hat{\beta}$ that minimizes the objective from part (a) also minimizes

$$\text{minimize}_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}(\widetilde{y}_i - \widetilde{x}_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

Note that the expression in part (a) can be equivalently written

$$\text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda||\beta||^2\right\}$$

Add and subtract the mean of the columns of X, $\bar{x}_j$ from each x_{ij}

$$\begin{aligned}&\text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j + \bar{x}_j)\beta_j\right)^2 + \lambda||\beta||^2\right\}\\ &= \text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p}\bar{x}_j\beta_j - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j\right)^2 + \lambda||\beta||^2\right\}\\ &= \text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}\left(y_i - (\beta_0 + \sum_{j=1}^{p}\bar{x}_j\beta_j) - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j\right)^2 + \lambda||\beta||^2\right\}\end{aligned}$$

From the solution to part (a), we know that the argmin_{β_0} yields $\bar{y} = \beta_0 + \sum_{j=1}^{p}\bar{x}_j\beta_j$. Thus the expression evaluates to

$$\text{minimize}_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}\left(y_i - \bar{y} - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j\right)^2 + \lambda||\beta||^2\right\}$$

This matches the desired form

$$\text{minimize}_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}(\widetilde{y}_i - \widetilde{x}_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

with $\widetilde{y}_i = y_i - \bar{y}$ and $\widetilde{x}_{ij} = x_{ij} - \bar{x}_j$ for all N observations in X.

(c)

In this scenario we have a function, which we will call `FUN`, that solves the following

$$\text{minimize}_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}(y_i - x_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

We would like to use this function to solve for β_0 and β in

$$\text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

In order to solve for $\hat{\beta}$, we can use the relationship that we established in part (b) to plug in centered data. Thus, taking features x and y and centering them to $\widetilde{x}_{ij} = x_{ij} - \bar{x}_j$ for all N observations in X and $\widetilde{y}_i = y_i - \bar{y}$, the function `FUN` will solve

$$\text{argmin}_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}(\widetilde{y}_i - \widetilde{x}_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

With this output for $\hat{\beta}$ along with the original data, we can solve $\hat{\beta}_0$ as

$$\hat{\beta}_0 = \bar{y} - \bar{x}^T\beta$$

Where $\bar{x}$ is the vector with the mean of each column in X. With these two quantities, we have solved

$$\text{minimize}_{\beta\in\mathbb{R}^p,\beta_0\in\mathbb{R}}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda||\beta||^2\right\}$$

Question 2

(a)

We are seeking a closed-form to the single coordinate update

$$\hat{\beta}_j = \text{argmin}_{\beta_j}\left\{\frac{1}{2}\sum_{i=1}^{N}(y_i - x_i^T\beta)^2 + \lambda||\beta||_1\right\}$$

where coordinates $\beta_1, \ldots, \beta_{j-1}, \beta_{j+1}, \ldots, \beta_p$ are fixed. This can be equivalently written as

$$f(\hat{\beta}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{k\neq j}^{p} x_{ik}\beta_k - x_{ij}\beta_j\right)^2 + \lambda\sum_{k\neq j}^{p}|\beta_k| + \lambda|\beta_j|$$

where values for β_k for $k \neq j$ are fixed at values $\hat{\beta}_k(\lambda)$. To minimize with respect to β_j, take the derivative and set equal to zero.

$$\frac{\partial}{\partial\beta_j} = \sum_{i=1}^{N}\left(y_i - \sum_{k\neq j}^{p} x_{ik}\hat{\beta}_k - x_{ij}\beta_j\right)\big(-x_{ij}\big) + \lambda\frac{\partial}{\partial\beta_j}\big(|\beta_j|\big) = 0$$

This simplifies to

$$\beta_j \sum_{i=1}^{N} \left(x_{ij}^2\right) - x_{ij} \sum_{i=1}^{N} \left(y_i - \hat{y}_i^{(-j)}\right) + \lambda \frac{\partial}{\partial \beta_j} \left(|\beta_j|\right) = 0$$

where $\hat{y}^{(-j)}$ is the fitted value of $\hat{y}$ excluding the jth product term, $x_{ij}\beta_j$. For simplicity, we will define the following terms:

$$\begin{aligned}
\psi_j &= \sum_{i=1}^{N} x_{ij}^2 \\
\delta_j &= x_{ij} \sum_{i=1}^{N} \left(y_i - \hat{y}_i^{(-j)}\right) \\
&= x_{ij} \sum_{i=1}^{N} (y_i - \hat{y}_i + x_{ij}\beta_j)
\end{aligned}$$

The expression can now be written

$$\psi_j \beta_j + \delta_j + \lambda \frac{\partial}{\partial \beta_j} \left(|\beta_j|\right) = 0$$

We will now consider the conditions that results from the derivative of $|\beta_j|$:

$$0 = \begin{cases} \psi_j \beta_j - \delta_j + \lambda, \text{ if } \beta_j > 0 \\ [-\delta_j - \lambda, -\delta_j + \lambda], \text{ if } \beta_j = 0 \\ \psi_j \beta_j - \delta_j - \lambda, \text{ if } \beta_j < 0 \end{cases}$$

Rearranging this to condition on λ and δ_j:

$$\hat{\beta}_j = \begin{cases} \frac{\delta_j - \lambda}{\psi_j}, \text{ if } \delta_j > \lambda \\ 0, \text{ if } -\lambda \leq \delta_j \leq \lambda \\ \frac{\delta_j + \lambda}{\psi_j}, \text{ if } \delta_j < -\lambda \end{cases}$$

(b)

The function UpdateCoefficient is created in the following code:

```
##############
# Question 2b
##############

## Define soft threshold function
soft_threshold = function(rho,lam) {
  if(rho > lam) {
     return(rho-lam)
  } else if(rho < -lam) {
    return(rho+lam)
  } else {
    return(0)
  }
}

## Define UpdateCoefficient function
UpdateCoefficient <- function(x,y,lambda,beta,j){
  y_hat <- x %*% beta
  #alternatively, y_hat <- x[,-j] %*% beta[-j]
  delta <- t(x[,j]) %*% ((y-y_hat)+beta[j]*x[,j])
  #alternatively, delta <- t(x[,j]) %*% (y-y_hat)
  beta_j <- 1/sum(x[,j]^2)*soft_threshold(delta,lambda)
  beta[j] <- beta_j
  return(beta)
}
```

(c)

In order to use the single coordinate update to perform coordinate descent and solve a lasso optimization problem, perform the single coordinate update for each coefficient, 1 to p, in the initial beta vector. One iteration corresponds to fitting each coefficient in beta one time. Perform this optimization until either a maximum number of iterations limit is hit or until the changes in the coefficients from one iteration to the next are below a predesignated threshold.

(d)

The code for the function `MyLasso` is below. Note that the equation for the fitted intercept term, $\hat{\beta_0}$, is taken from Question 1.

$$\hat{\beta_0} = \bar{y} - \bar{X}^T\beta$$

```
##############
# Question 2d
##############

## Define function to find L2 norm
norm_vec <- function(x) sqrt(sum(x^2))

## Define MyLasso
MyLasso <- function(x, y, lambda, maxiter=200, thr=1e-3){
  ## Initialize beta
  beta <- rep(0,ncol(x))
  beta.iter <- matrix(nrow=ncol(x), ncol=0)
  num_iter <- 0
  beta_diff <- 1
  ## Establish number of iterations and threshold criteria
  while ((num_iter <= maxiter) && (beta_diff >= thr)) {
    beta_init <- beta
    for (it in 1:length(beta)) {
      beta <- UpdateCoefficient(x,y,lambda,beta,it)
    }
    ## Add results of iteration to storing objects
    beta.iter <- cbind(beta.iter,beta)
    num_iter <- num_iter+1
    beta_diff <- norm_vec(beta-beta_init)/norm_vec(beta_init)
  }
  colnames(beta.iter) <- paste0(rep("iter",ncol(beta.iter)),
                                as.character(1:ncol(beta.iter)))
  ## Compute intercept
  beta.intercept <- mean(y)-t(apply(x,2,mean)) %*% beta
  return(list(beta,beta.intercept,beta.iter))
}
```

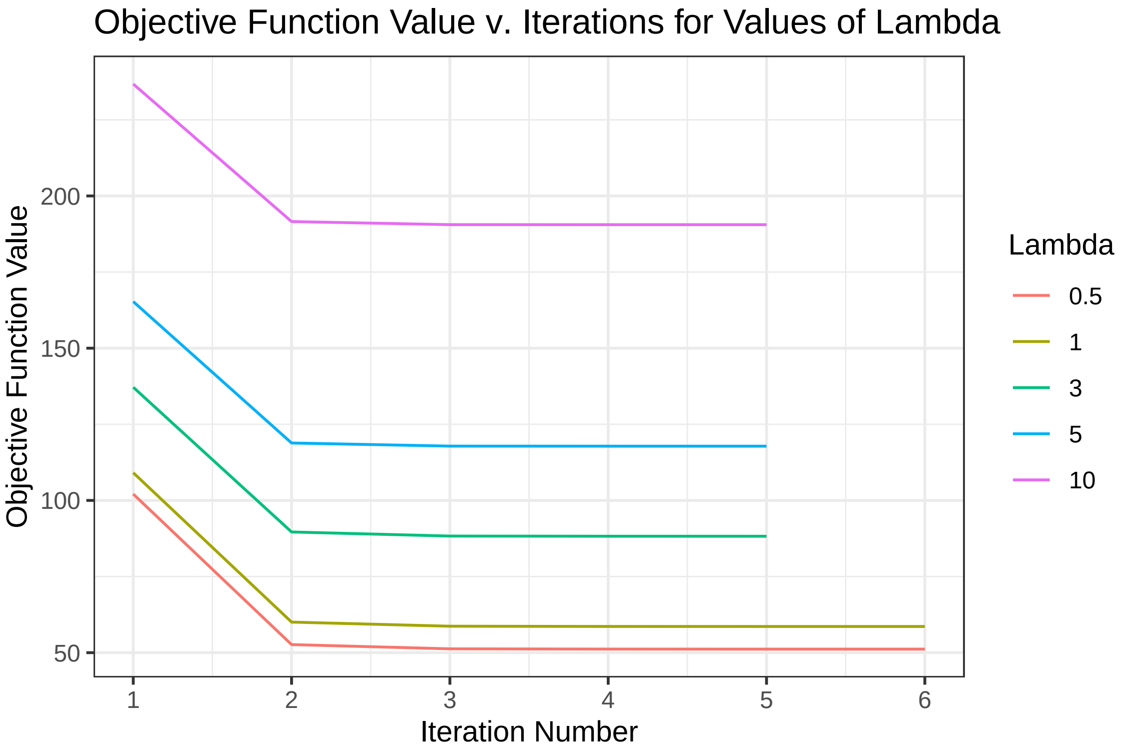
(e)

The table below shows the coefficients of $\hat{\beta}$ obtained with different values of λ.

	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 3$	$\lambda = 5$	$\lambda = 10$
$\hat{\beta}_0$	0.075	0.075	0.074	0.073	0.059
$\hat{\beta}_1$	3.029	3.022	2.998	2.974	2.920
$\hat{\beta}_2$	3.002	2.990	2.945	2.903	2.840
$\hat{\beta}_3$	2.802	2.796	2.768	2.738	2.685
$\hat{\beta}_4$	3.049	3.041	3.013	2.986	2.932
$\hat{\beta}_5$	2.942	2.937	2.918	2.903	2.870
$\hat{\beta}_6$	-0.005	0	0	0	0
$\hat{\beta}_7$	-0.007	0	0	0	0
$\hat{\beta}_8$	0	0	0	0	0
$\hat{\beta}_9$	-0.120	-0.114	-0.096	-0.079	-0.034
$\hat{\beta}_{10}$	0.102	0.096	0.070	0.046	0
$\hat{\beta}_{11}$	-0.123	-0.107	-0.051	0	0
$\hat{\beta}_{12}$	-0.007	0	0	0	0
$\hat{\beta}_{13}$	-0.147	-0.142	-0.121	-0.100	-0.043
$\hat{\beta}_{14}$	0.015	0.008	0	0	0
$\hat{\beta}_{15}$	0	0	0	0	0
$\hat{\beta}_{16}$	0.100	0.089	0.049	0.008	0
$\hat{\beta}_{17}$	0.114	0.107	0.080	0.053	0
$\hat{\beta}_{18}$	0.125	0.114	0.071	0.029	0
$\hat{\beta}_{19}$	-0.125	-0.122	-0.120	-0.117	-0.099
$\hat{\beta}_{20}$	0.016	0.008	0	0	0

(f)

The plot showing the value of the objective function versus number of iterations for varying values of λ is below. From the graph, it appears that the greatest decrease in the objective function occurs between iterations 1 and 2 (Note that the initialized value of β , which was all 0's, is not included). All optimization procedures considered here, for λ from 0.5 to 10, complete in 5 to 6 cycles with the objective function monotone decreasing. As the value of λ increases, the minimum attained by the objective function increases.



(g)

The table below shows the coefficients of $\hat{\beta}$ obtained with least squares and `MyLasso` with $\lambda = 10^{-5}$. The values are generally comparable across coefficients, with differences on the magnitude of 0.01 or 0.001. This confirms the desired output of `MyLasso`.

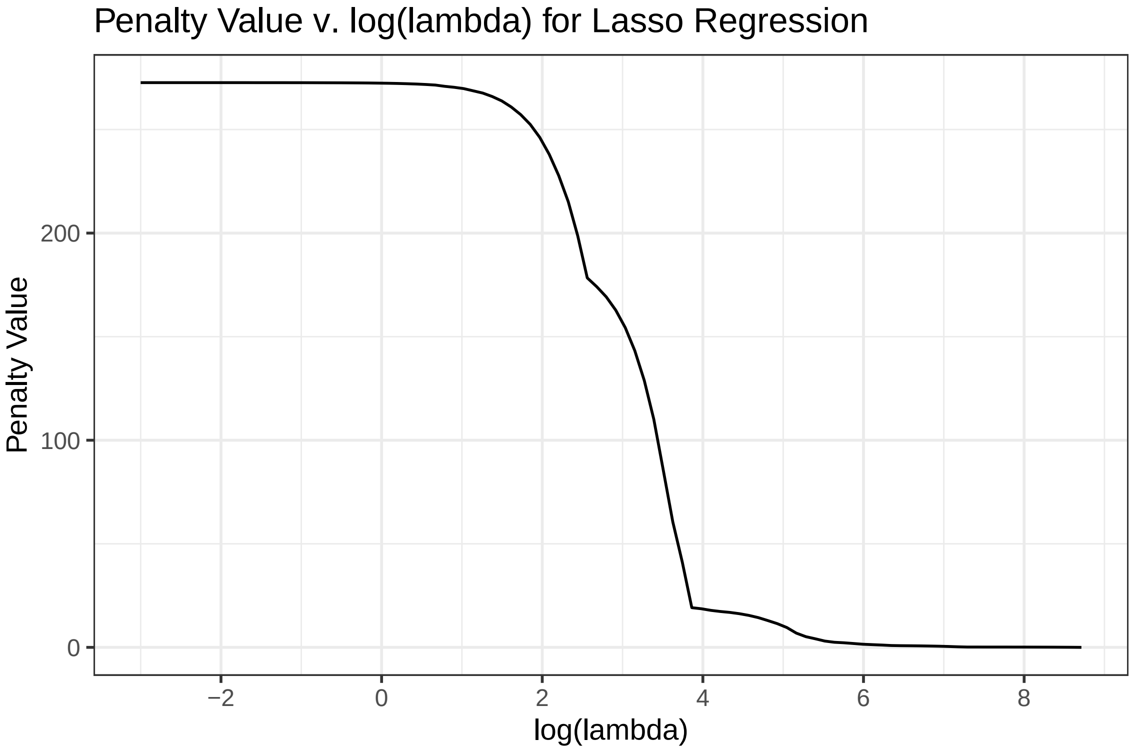
	Least Squares	MyLasso ($\lambda = 10^{-5}$)
$\hat{\beta}_0$	0.091	0.076
$\hat{\beta}_1$	3.040	3.038
$\hat{\beta}_2$	3.040	3.019
$\hat{\beta}_3$	2.809	2.811
$\hat{\beta}_4$	3.076	3.058
$\hat{\beta}_5$	2.964	2.949
$\hat{\beta}_6$	-0.015	-0.017
$\hat{\beta}_7$	-0.021	-0.018
$\hat{\beta}_8$	-0.019	-0.008
$\hat{\beta}_9$	-0.121	-0.128
$\hat{\beta}_{10}$	0.104	0.108
$\hat{\beta}_{11}$	-0.135	-0.142
$\hat{\beta}_{12}$	-0.015	-0.020
$\hat{\beta}_{13}$	-0.138	-0.153
$\hat{\beta}_{14}$	0.018	0.023
$\hat{\beta}_{15}$	-0.009	-0.003
$\hat{\beta}_{16}$	0.104	0.111
$\hat{\beta}_{17}$	0.123	0.121

	Least Squares	MyLasso ($\lambda = 10^{-5}$)
$\hat{\beta}_{18}$	0.132	0.137
$\hat{\beta}_{19}$	-0.122	-0.130
$\hat{\beta}_{20}$	0.027	0.026

Question 3

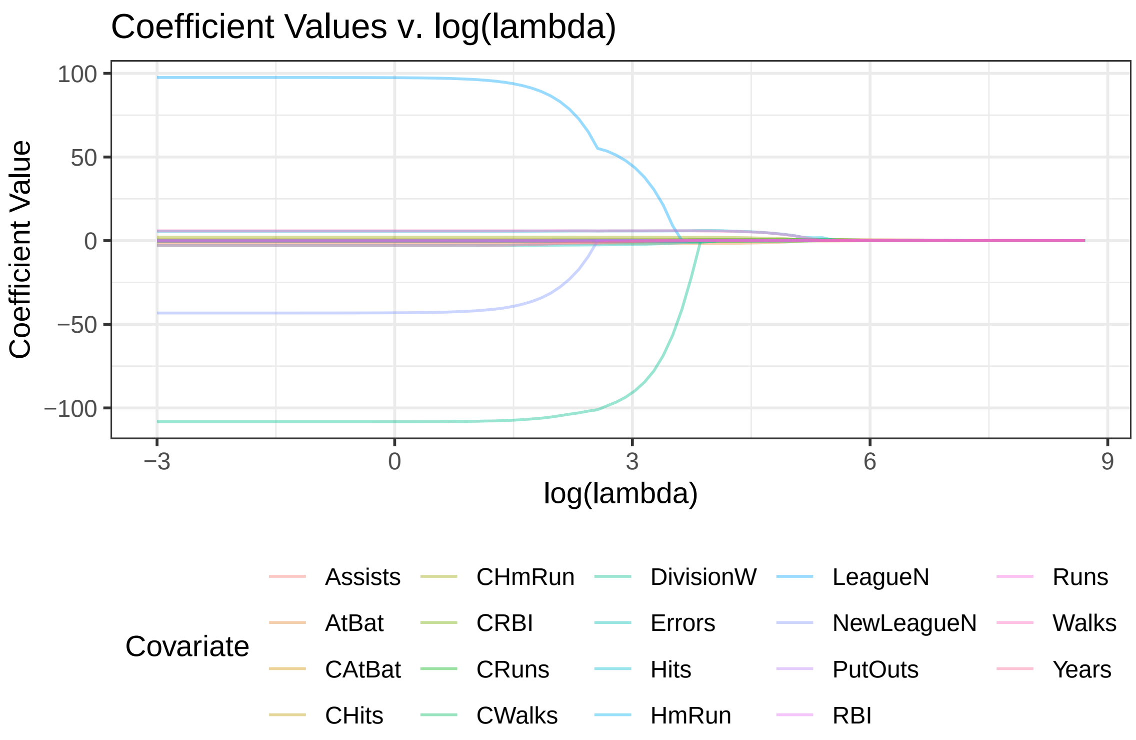
(a)

The plot below shows the value of the penalty term $||\hat{\beta}||_1$ for varying values of λ. The penalty value remains high, above 250, for values of λ between 0.001 and 1, at which point it begins to decrease sharply with increasing λ. The rate of decrease slows around $\lambda = 10^4$ and approaches zero for increasing values thereafter. As the penalty is the sum of the absolute value of coefficient values, this suggests that the coefficients converge towards zero.

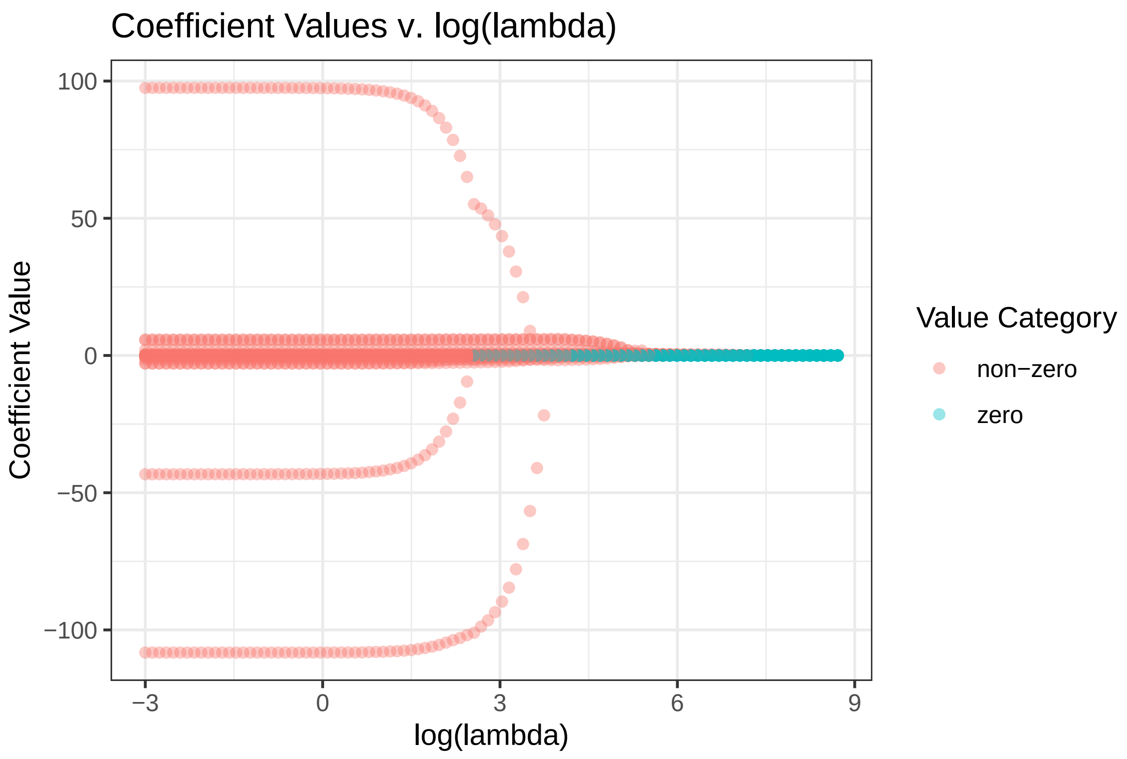


(b)

The graph showing lasso coefficients for each value of λ is below. The graph shows that as the value of λ increases, the coefficients shrink towards zero. This is even the case for coefficients which are initially large in magnitude such as LeagueN, DivisionW, and NewLeagueN.



Below is a similar plot showing individual coefficient values, color coded by whether the coefficients equal zero. As can be seen by the plot, there are several values of λ that results in sparse solutions, with some, but not all, coefficients being equal to zero. These generally occur for λ values between 10^3 and 10^6.



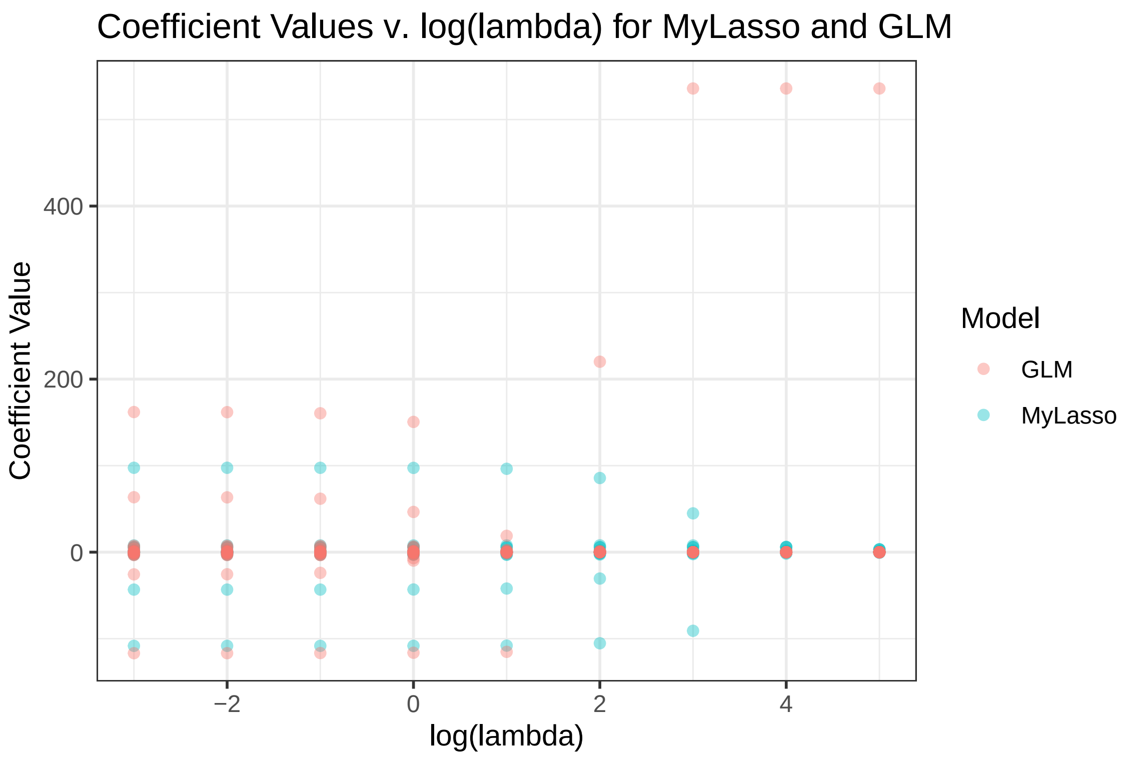
Question 4

(a)

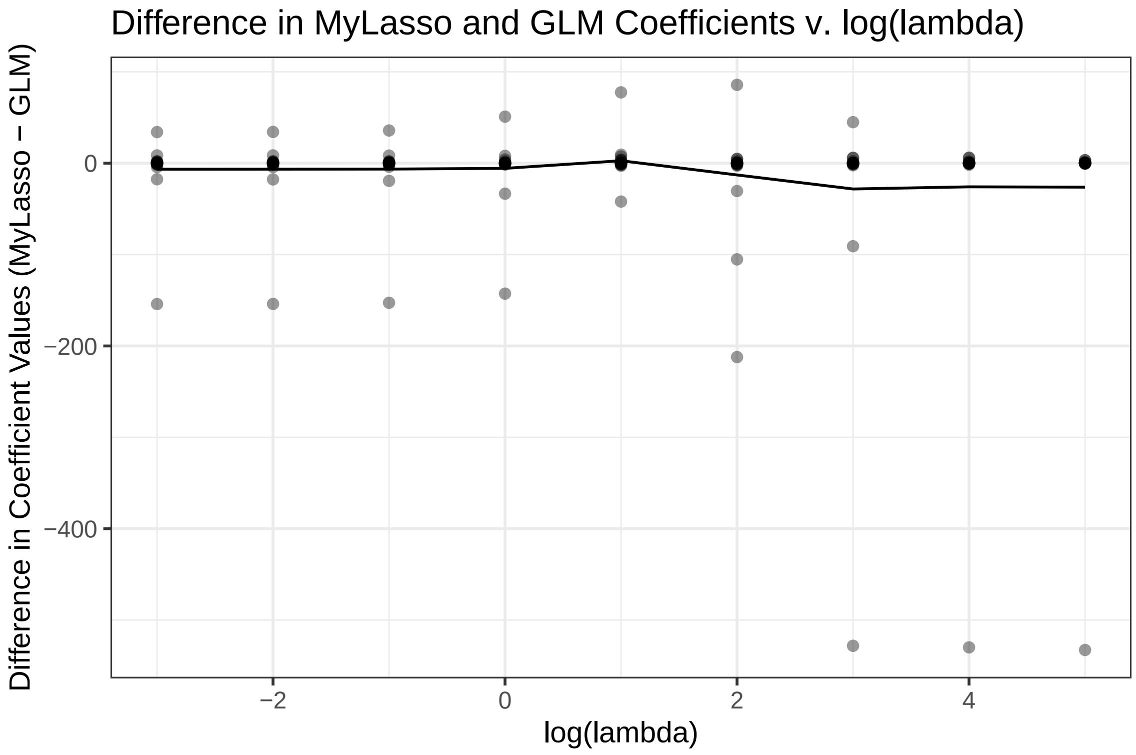
Below is a table comparing the $\hat{\beta}$ coefficients for `MyLasso` and the `glmnet` functions at several values of λ. The coefficients from the two models do not appear to be the same, although the relative ranking of coefficient values is similar. It is also worth noting that the coefficients of `glmnet` converge towards zero more rapidly than those of `MyLasso` as the λ value increases.

	glmnet	MyLasso	glmnet	MyLasso	glmnet	MyLasso
	$\lambda = 0.5$	$\lambda = 0.5$	$\lambda = 10$	$\lambda = 10$	$\lambda = 100$	$\lambda = 100$
(Intercept)	156.086	7.977	-1.465	7.797	220.104	7.952
AtBat	-1.962	-1.443	0	-1.444	0	-1.452
Hits	7.162	5.746	2.011	5.747	1.136	5.755
HmRun	2.458	-0.104	0	-0.104	0	-0.102
Runs	-1.732	-0.536	0	-0.534	0	-0.515
RBI	-0.342	0.070	0	0.070	0	0.076
Walks	5.889	5.914	2.249	5.913	1.183	5.901
Years	-5.194	-1.663	0	-1.650	0	-1.521
CAtBat	-0.117	-0.070	0	-0.070	0	-0.071
CHits	0.031	0.542	0	0.542	0	0.541
CHmRun	0	2.088	0.042	2.088	0	2.088
CRuns	1.339	0.368	0.220	0.368	0.110	0.372
CRBI	0.701	-0.256	0.404	-0.256	0.315	-0.254
CWalks	-0.776	-0.656	0	-0.656	0	-0.654
LeagueN	54.353	97.120	18.877	96.108	0	85.585
DivisionW	-115.891	-106.306	-115.191	-106.153	0	-104.70
PutOuts	0.283	0.284	0.236	0.285	0.003	0.285
Assists	0.329	0.243	0	0.243	0	0.244
Errors	-3.082	-2.779	-0.783	-2.772	0	-2.709
NewLeagueN	-16.521	-41.608	0	-40.505	0	-30.055

For a wider range of values of λ, a dot plot of the coefficient values for `MyLasso` and glmnet are shown below. From the plots, it is apparent that the coefficient values are not the same between the two methods.

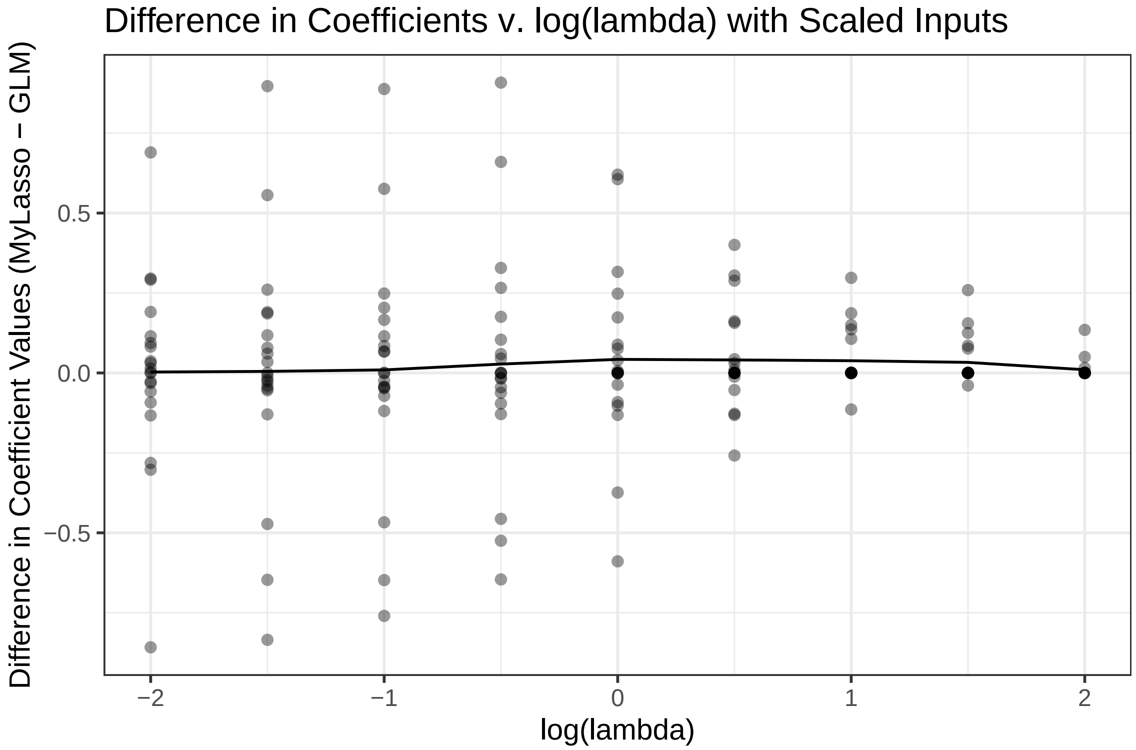


Below is a plot of the difference between coefficients in the two models, with a line indicating the mean of the difference at each value of λ.



(b)

The primary difference between the two models appears to involve standardization. The documentation from `glmnet` suggests that it standardizes the x and y inputs by default. By setting the parameter `standardize=FALSE` in the function and inputting a standardized x and y, it is possible to dramatically shrink the difference in values between the two models. The same plot showing difference between the models is below.



Question 5

The goal is to show that the ridge regression estimator is the mean (and mode) of the posterior distribution under a Gaussian prior $\beta \sim N(0, \tau I)$, and Gaussian sampling model $y \sim N(X\beta, \sigma^2 I)$. The posterior distribution is given by

$$P(\beta|y, X) = \frac{P(y|\beta, X)P(\beta)}{P(y|X)} \propto P(y|\beta, X)P(\beta)$$

This evaluates to

$$P(\beta|y, X) \propto (2\pi)^{-p/2}\sigma^{-p} \exp\bigg(-\frac{(y - X\beta)^T(y - X\beta)}{2\sigma^2}\bigg)(2\pi)^{-p/2}\tau^{-p/2} \exp\bigg(-\frac{\beta^T\beta}{2\tau}\bigg)$$

Taking the log of this distribution in order to find the maximum likelihood estimate of β:

$$log\big(P(\beta|y, X)\big) = -\frac{(y - X\beta)^T(y - X\beta)}{2\sigma^2} - \frac{\beta^T\beta}{2\tau} + C$$

Where C is a constant term not dependent on β. Deriving with respect to β and setting equal to 0,

$$\begin{aligned}\frac{\partial}{\partial \beta} log\big(P(\beta|y, X)\big) &= \frac{X^T y - X^T X\beta}{\sigma^2} - \frac{\beta}{\tau} = 0 \\ X^T y &= X^T X \beta + \frac{\sigma^2}{\tau}\beta \\ \hat{\beta} &= \left(X^T X + \frac{\sigma^2}{\tau} I\right)^{-1} X^T y\end{aligned}$$

This matches the form of the coefficients for ridge regression with

$$\lambda = \frac{\sigma^2}{\tau}$$

As the normal distribution is the conjugate prior of the normal distribution, we know that $P(\beta|y, x)$ is also normal. As a property of this distribution, the mean equals the mode. Using μ_β and Σ_β to denote the mean and covariance of this normal posterior distribution, the exponential term in the posterior will be

$$\exp\bigg(- \frac{1}{2}(\beta - \mu_\beta)^T \Sigma_\beta^{-1}(\beta - \mu_\beta)\bigg) = \exp\bigg(- \frac{1}{2}(\beta^T \Sigma_\beta^{-1} \beta - 2\beta^T \Sigma_\beta^{-1}\mu_\beta + \mu_\beta^T \Sigma_\beta^{-1} \mu_\beta^T)\bigg)$$

Equating the quadratic term with that in the posterior above yields

$$\begin{aligned}\beta^T \Sigma_\beta^{-1} \beta &= \frac{1}{\sigma^2}\beta^T X^T X \beta + \frac{1}{\tau}\beta^T \beta \\ \Sigma_\beta^{-1} &= \frac{1}{\sigma^2} X^T X + \frac{1}{\tau} I\end{aligned}$$

Equating the linear term with that in the posterior above yields

$$\begin{aligned}2\beta^T \Sigma_\beta^{-1} \mu_\beta &= \frac{2}{\sigma^2}\beta^T X^T y \\ \mu_\beta &= \frac{1}{\sigma^2}\Sigma_\beta X^T y \\ &= \frac{1}{\sigma^2}\left(\frac{1}{\sigma^2} X^T X + \frac{1}{\tau} I\right)^{-1} X^T y \\ &= \left(X^T X + \frac{\sigma^2}{\tau} I\right)^{-1} X^T y \\ &= \hat{\beta}\end{aligned}$$

Thus we have proved that the mean and mode of the posterior distribution is the ridge regression estimator.

Question 6

In this problem, the observations in the kth class are drawn from a $N(\mu_k, \Sigma_k)$ distribution. The density of $N(\mu_k, \Sigma_k)$ is

$$f_k(x) = (2\pi)^{-p/2} \det(\Sigma_k)^{-1/2} \exp\bigg(- \frac{1}{2}\big(x - \mu_k\big)^T \Sigma_k^{-1} \big(x - \mu_k\big)\bigg)$$

The decision boundary between class 1 and class 2 would be defined

$$\begin{aligned}\text{Decision boundary} &= \left\{x : \frac{f_1(x)\pi_1}{f_1(x)\pi_1 + f_2(x)\pi_2} = \frac{f_2(x)\pi_2}{f_1(x)\pi_1 + f_2(x)\pi_2}\right\} \\ &= \left\{x : f_1(x)\pi_1 = f_2(x)\pi_2\right\} \\ &= \left\{x : log\big(f_1(x)\big) + log\big(\pi_1\big) = log\big(f_2(x)\big) + log\big(\pi_2\big)\right\}\end{aligned}$$

Plugging in the density function for $f_k(x)$:

$$\begin{aligned}\text{Decision boundary} &= \bigg\{x : log\big(\pi_1\big) + log\big((2\pi)^{-p/2}\big) + log\big(\det(\Sigma_1)^{-1/2}\big) - \frac{1}{2}\big(x-\mu_1\big)^T\Sigma_1^{-1}\big(x-\mu_1\big) \\ &= log\big(\pi_2\big) + log\big((2\pi)^{-p/2}\big) + log\big(\det(\Sigma_2)^{-1/2}\big) - \frac{1}{2}\big(x-\mu_2\big)^T\Sigma_2^{-1}\big(x-\mu_2\big)\bigg\} \\ &= \bigg\{x : log\big(\pi_1\big) - log\big(\pi_2\big) + log\big(\det(\Sigma_1)^{-1/2}\big) - log\big(\det(\Sigma_2)^{-1/2}\big) \\ &- \frac{1}{2}\big(x-\mu_1\big)^T\Sigma_1^{-1}\big(x-\mu_1\big) + \frac{1}{2}\big(x-\mu_2\big)^T\Sigma_2^{-1}\big(x-\mu_2\big) = 0\bigg\} \\ &= \bigg\{x : log\bigg(\frac{\pi_1}{\pi_2}\bigg) - \frac{1}{2}log\bigg(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\bigg) - \frac{1}{2}\big(x-\mu_1\big)^T\Sigma_1^{-1}\big(x-\mu_1\big) + \frac{1}{2}\big(x-\mu_2\big)^T\Sigma_2^{-1}\big(x-\mu_2\big) = 0\bigg\} \\ &= \bigg\{x : log\bigg(\frac{\pi_1}{\pi_2}\bigg) - \frac{1}{2}log\bigg(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\bigg) - \frac{1}{2}\big(x^T\Sigma_1^{-1}x - 2\mu_1^T\Sigma_1^{-1}x + \mu_1^T\Sigma_1^{-1}\mu_1\big) \\ &+ \frac{1}{2}\big(x^T\Sigma_2^{-1}x - 2\mu_2^T\Sigma_2^{-1}x + \mu_2^T\Sigma_2^{-1}\mu_2\big) = 0\bigg\} \\ &= \bigg\{x : log\bigg(\frac{\pi_1}{\pi_2}\bigg) - \frac{1}{2}log\bigg(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\bigg) - \frac{1}{2}x^T\Sigma_1^{-1}x + \mu_1^T\Sigma_1^{-1}x - \frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1 \\ &+ \frac{1}{2}x^T\Sigma_2^{-1}x - \mu_2^T\Sigma_2^{-1}x + \frac{1}{2}\mu_2^T\Sigma_2^{-1}\mu_2 = 0\bigg\} \\ &= \bigg\{x : \frac{1}{2}x^T\bigg(\Sigma_2^{-1} - \Sigma_1^{-1}\bigg)x + \bigg(\mu_1^T\Sigma_1^{-1} - \mu_2^T\Sigma_2^{-1}\bigg)x \\ &+ \bigg(log\bigg(\frac{\pi_1}{\pi_2}\bigg) - \frac{1}{2}log\bigg(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\bigg) - \frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma_2^{-1}\mu_2\bigg) = 0\bigg\}\end{aligned}$$

This corresponds to a quadratic decision boundary with

$$\begin{aligned}\text{Quadratic term} &: \frac{1}{2}\bigg(\Sigma_2^{-1} - \Sigma_1^{-1}\bigg) \\ \text{Linear term} &: \bigg(\mu_1^T\Sigma_1^{-1} - \mu_2^T\Sigma_2^{-1}\bigg) \\ \text{Constant term} &: \bigg(log\bigg(\frac{\pi_1}{\pi_2}\bigg) - \frac{1}{2}log\bigg(\frac{\det(\Sigma_1)}{\det(\Sigma_2)}\bigg) - \frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1 + \frac{1}{2}\mu_2^T\Sigma_2^{-1}\mu_2\bigg)\end{aligned}$$

Question 7

(a)

For the non-scaled model, the residual sum of squares is defined

$$\sum_{i=1}^{N}\left(y_i - \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2}\right)^2$$

For the scaled model, the RSS is defined

$$\sum_{i=1}^{N}\left(y_i - \widetilde{\beta}_0 + \widetilde{\beta}_1(100 * X_{i1}) + \widetilde{\beta}_2 X_{i2}\right)^2$$

where $\widetilde{\beta}_0$, $\widetilde{\beta}_1$, and $\widetilde{\beta}_2$ are coefficients corresponding to the scaled data, $\widetilde{X}$. From part (b) below, we know these coefficients, $\widetilde{\beta}$, are

$$\widetilde{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 * \frac{1}{100} \\ \hat{\beta}_2 \end{pmatrix}$$

Plugging these values into the RSS for the scaled model:

$$\sum_{i=1}^{N}\left(y_i - \hat{\beta}_0 + \frac{1}{100}\hat{\beta}_1(100 * X_{i1}) + \hat{\beta}_2 X_{i2}\right)^2 = \sum_{i=1}^{N}\left(y_i - \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2}\right)^2$$

Thus, the RSS for the scaled and the non-scaled models are the same.

(b)

We will define the scaling matrix D as

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We can then define the scaled X matrix as

$$\widetilde{X} = XD$$

Then considering the objective function for $\widetilde{X}$ matrix and corresponding coefficient matrix, $\widetilde{\beta}$,

$$\begin{aligned} ||\vec{y} - \widetilde{X}\widetilde{\beta}||^2 &= ||\vec{y} - XD\widetilde{\beta}||^2 \\ &= (\vec{y} - XD\widetilde{\beta})^T(\vec{y} - XD\widetilde{\beta}) \\ &= (\vec{y}^T - \widetilde{\beta}^T D^T X^T)(\vec{y} - XD\widetilde{\beta}) \\ &= \vec{y}^T\vec{y} - \widetilde{\beta}^T D^T X^T \vec{y} - \vec{y}^T XD\widetilde{\beta} + \widetilde{\beta}^T D^T X^T XD\widetilde{\beta} \\ &= \vec{y}^T\vec{y} - 2\widetilde{\beta}^T D^T X^T \vec{y} + \widetilde{\beta}^T D^T X^T XD\widetilde{\beta} \end{aligned}$$

Taking the derivative with respect to $\widetilde{\beta}$, $\frac{\partial}{\partial\widetilde{\beta}}$, and setting equal to zero, we have

$$\begin{aligned} 0 &= -2D^T X^T \vec{y} + 2D^T X^T XD\widetilde{\beta} \\ D^T X^T \vec{y} &= D^T X^T XD\widetilde{\beta} \\ X^T \vec{y} &= X^T XD\widetilde{\beta} \\ D\widetilde{\beta} &= (X^T X)^{-1} X^T \vec{y} \end{aligned}$$

Since the right hand side of this equation evaluates to $\hat{\beta}$, the coefficients of the non-scaled X, we have

$$\widetilde{\beta} = D^{-1}\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 * \frac{1}{100} \\ \hat{\beta}_2 \end{pmatrix}$$

(c)

The two sets of fitted values are shown below.

- For Model 1 (Non-scaled): $\hat{Y} = X\beta$
- For Model 2 (Scaled): $\widetilde{Y} = \widetilde{X}\widetilde{\beta} = XDD^{-1}\beta = X\beta$

Thus, the two fitted values are equivalent.

(d)

This problem compares the fitted values from the following models for a given value of λ:

- Ridge regression with non-scaled data and corresponding coefficients: $\hat{Y} = X\beta_\lambda$
- Ridge regression with scaled data and corresponding coefficients: $\widetilde{Y} = \widetilde{X}\widetilde{\beta}_\lambda$

These quantities can be evaluated to

$$\begin{aligned}
\hat{Y} &= X\beta_\lambda \\
&= X(X^TX + \lambda I)^{-1}X^T\vec{y} \\
&= \left(1 + \frac{1}{\lambda}XX^T\right)\vec{y} \\
\widetilde{Y} &= \widetilde{X}\widetilde{\beta}_\lambda \\
&= \widetilde{X}(\widetilde{X}^T\widetilde{X} + \lambda I)^{-1}\widetilde{X}^T\vec{y} \\
&= XD(DX^TXD + \lambda I)^{-1}DX^T\vec{y} \\
&= XD\left(D^{-1}X^{-1}(X^T)^{-1}D^{-1} + \frac{1}{\lambda}I\right)DX^T\vec{y} \\
&= \left((X^T)^{-1}D^{-1} + \frac{1}{\lambda}XD\right)DX^T\vec{y} \\
&= \left(1 + \frac{1}{\lambda}XDDX^T\right)\vec{y}
\end{aligned}$$

As these two equations for fitted values are not equal, we can conclude that the two regressions would not yield the same fitted values for a given λ. This is due to the addition of the penalty term, which does not allow for the coefficients to scale in a manner that negates the scaling in the data, as was the case in the simple least squares regression.

(e)

The code below performs a simulation to verify the results of parts (a) through (d) above. We will prove part (a) once we have proven that the results for coefficients (b) and fitted values (c) hold.

The code below generates the X data, as well as the scaled version , $\widetilde{X}$. It then computes the corresponding coefficient values.

```r
##############
# Question 7e
##############

## Define data
set.seed(44)
x1 <- rnorm(100,10,1)
x2 <- rnorm(100,0,1)
X <- cbind("ones"=rep(1,100),"x1"=x1,"x2"=x2)
X_scaled <- cbind("ones"=rep(1,100),"x1"=x1*100,"x2"=x2)
y <- runif(100,0,10)

## B
beta <- solve(t(X) %*% X) %*% t(X) %*% y
beta_scaled <- solve(t(X_scaled) %*% X_scaled) %*% t(X_scaled) %*% y
print(beta)
print(beta_scaled)
```

The coefficients for non-scaled data, $\hat{\beta}$ are below.

$$\hat{\beta} = \begin{pmatrix} 9.23022 \\ -0.41633 \\ -0.04217 \end{pmatrix}$$

The coefficients for scaled data, $\widetilde{\beta}$ are below. The results match the relationship established in part (b).

$$\widetilde{\beta} = \begin{pmatrix} 9.23022 \\ -0.00416 \\ -0.04217 \end{pmatrix}$$

To show that the fitted values are the same, the following code computes the values and compares them. Note that rounding was needed to a very small digit (10th decimal place) due to digit carrying in R. The equality in fitted values proves the conclusion reached in part (c).

```r
## C
y_hat <- X %*% beta
y_hat_scaled <- X_scaled %*% beta_scaled
identical(round(y_hat,10),round(y_hat_scaled,10)) # TRUE; rounding needed to show equality
```

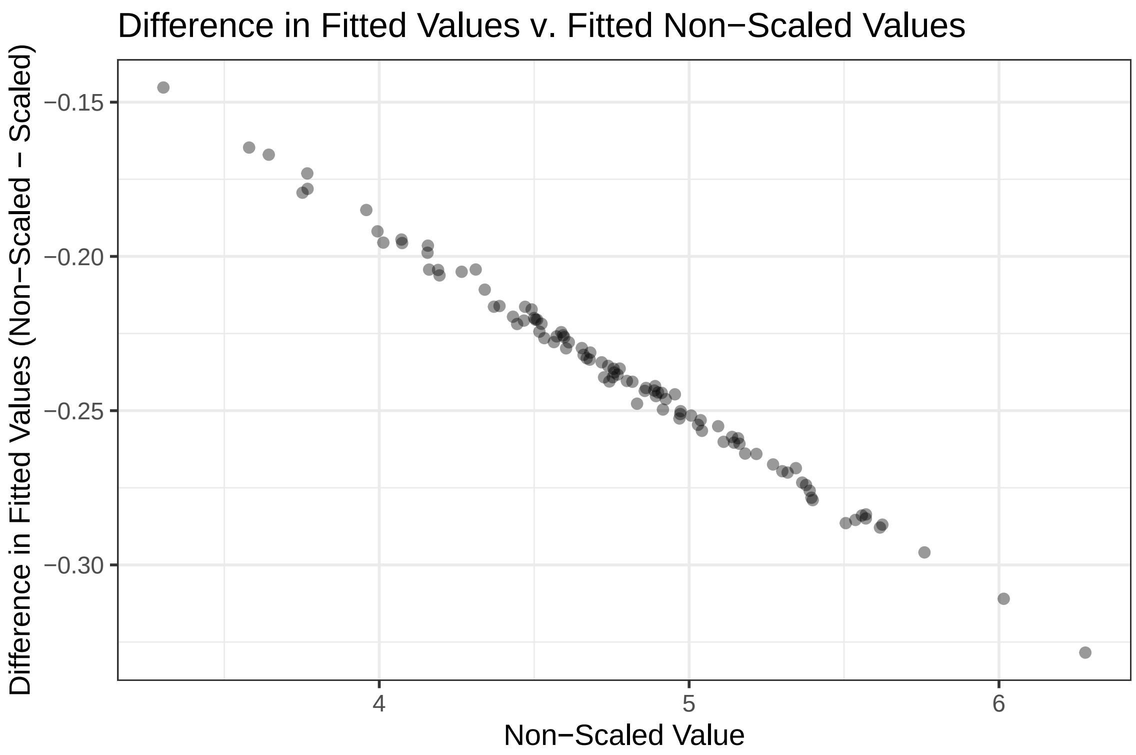
Now, to compare the sum of residual squares, the following code computes the RSS for both models. The output for both is 4104663, confirming the results in part (a).

```r
## A
sum((y-y_hat)^2)
sum((y-y_hat_scaled)^2)
```

Lastly, to compute the fitted values in ridge regression for the scaled and non-scaled data, the fitted values were computed using the corresponding data and coefficients for a λ value of 500. The difference in fitted values (non-scaled minus scaled) is plotted against the non-scaled fitted values in the chart below. As the plot shows, the fitted values in the scaled mode differ by 0.16 to 0.32 from the non-scaled model. This is consistent with the conclusion reached in part (d) of non-equal fitted values.

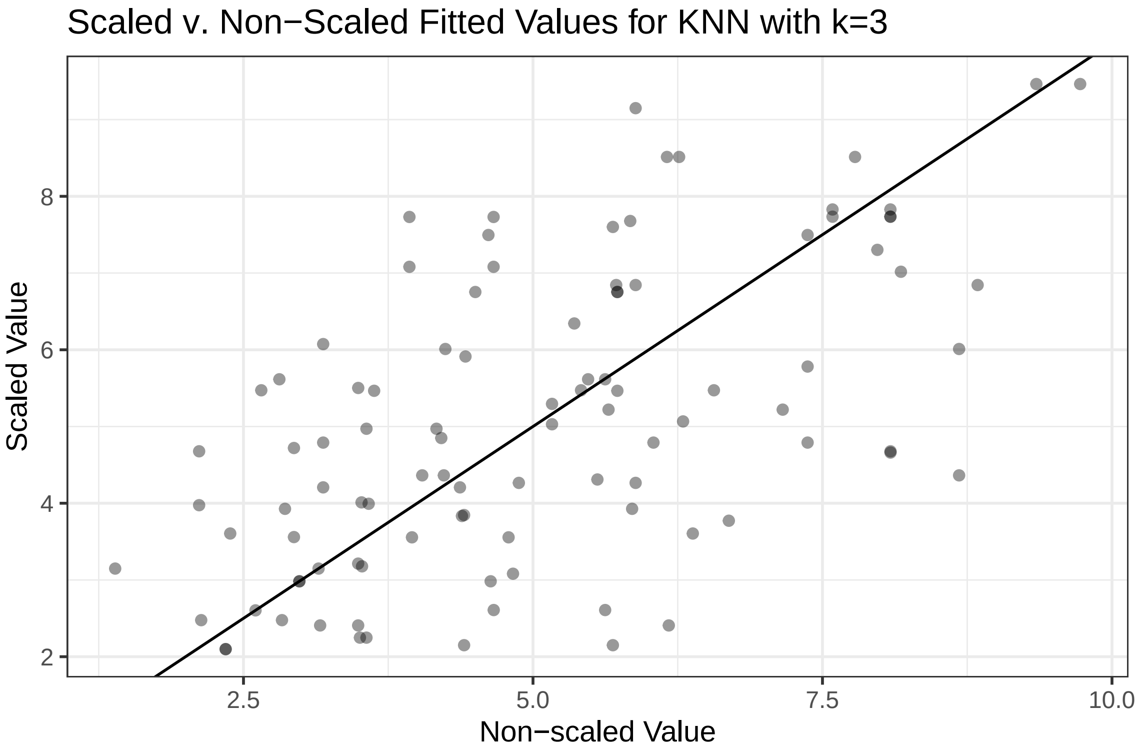
```
## D
lambda = 500
y_hat_ridge <- X %*% solve(t(X) %*% X + lambda * diag(3)) %*% t(X) %*% y
y_hat_ridge_scaled <- X_scaled %*% solve(t(X_scaled) %*% X_scaled + lambda * diag(3)) %*% t(X_scaled) %*
scaled_comp_df <- data.frame(regular = y_hat_ridge, scaled = y_hat_ridge_scaled)
scaled_comp_df$difference <- scaled_comp_df$regular - scaled_comp_df$scaled

g7 <- ggplot() +
  geom_point(data=scaled_comp_df, aes(x=regular, y=difference), alpha = 0.4) +
  labs(x='Non-Scaled Value', y='Difference in Fitted Values (Non-Scaled - Scaled)') +
  ggtitle("Difference in Fitted Values v. Fitted Non-Scaled Values") +
  theme_bw()
print(g7)
```



(f)

The KNN fitted values for scaled and non-scaled data are shown below. The plot shows the fitted values plotted against each other. If the points were to lie on a diagonal line through the origin, that would indicate that the values are equal. As can be seen in the plot, the fitted values are not equal for the scaled and non-scaled inputs with a k of 3.



(g)

We will consider a ridge regression to predict Y using just X_1 and tuning parameter $\lambda > 0$ compared to a regression that predicts Y using $\widetilde{X_1}$ and tuning parameter $\widetilde{\lambda} > 0$. New definitions for X, $\widetilde{X}$ are as follows:

$$\vec{X} = \vec{X_1}$$
$$\widetilde{X} = 100\vec{X_1} = 100\vec{X}$$

The ridge coefficients for the two regressions are as follows:

$$\begin{aligned}\beta_\lambda &= (\vec{X}^T\vec{X} + \lambda I)^{-1}\vec{X}^T\vec{y} \\ \widetilde{\beta}_{\widetilde{\lambda}} &= (\widetilde{X}^T\widetilde{X} + \widetilde{\lambda}I)^{-1}\widetilde{X}^T\vec{y} \\ &= (100\vec{X}^T * 100\vec{X} + \widetilde{\lambda}I)^{-1}100\vec{X}^T\vec{y} \\ &= (10000\vec{X}^T\vec{X} + \widetilde{\lambda}I)^{-1}100\vec{X}^T\vec{y}\end{aligned}$$

Consider the fitted values for non-scaled data, $\hat{Y}$:

$$
\begin{aligned}
\hat{Y} &= \vec{X}\beta_\lambda \\
&= \vec{X}(\vec{X}^T\vec{X} + \lambda I)^{-1}\vec{X}^T\vec{y} \\
&= \vec{X}\left(\vec{X}^{-1}(\vec{X}^T)^{-1} + \frac{1}{\lambda}I\right)\vec{X}^T\vec{y} \\
&= \left((\vec{X}^T)^{-1} + \frac{1}{\lambda}\vec{X}\right)\vec{X}^T\vec{y} \\
&= \left(1 + \frac{1}{\lambda}X\vec{X}^T\right)\vec{y}
\end{aligned}
$$

Now consider the fitted values for scaled data, $\widetilde{Y}$:

$$
\begin{aligned}
\widetilde{Y} &= \widetilde{X}\widetilde{\beta}_{\widetilde{\lambda}} \\
&= 100\vec{X}(10000\vec{X}^T\vec{X} + \widetilde{\lambda}I)^{-1}100\vec{X}^T\vec{y} \\
&= \vec{X}\left(100\vec{X}^T\vec{X} + \frac{\widetilde{\lambda}I}{100}\right)^{-1}100\vec{X}^T\vec{y} \\
&= \vec{X}\left(\frac{1}{100\vec{X}^T\vec{X}} + \frac{100}{\widetilde{\lambda}}I\right)100\vec{X}^T\vec{y} \\
&= \left(\frac{1}{100\vec{X}^T\vec{X}}\vec{X} + \frac{100}{\widetilde{\lambda}}\vec{X}\right)100\vec{X}^T\vec{y} \\
&= \left(1 + \frac{10000}{\widetilde{\lambda}}\vec{X}\vec{X}^T\right)\vec{y}
\end{aligned}
$$

Now, set these two quantities for $\hat{Y}$ and $\widetilde{Y}$ equal to each other and solve for $\widetilde{\lambda}$:

$$
\begin{aligned}
\left(1 + \frac{1}{\lambda}\vec{X}\vec{X}^T\right)\vec{y} &= \left(1 + \frac{10000}{\widetilde{\lambda}}\vec{X}\vec{X}^T\right)\vec{y} \\
\frac{1}{\lambda}\vec{X}\vec{X}^T &= \frac{10000}{\widetilde{\lambda}}\vec{X}\vec{X}^T \\
\widetilde{\lambda} &= 10000\lambda
\end{aligned}
$$

Thus, a value of $\widetilde{\lambda} = 10000\lambda$ will lead to equal fitted values $\hat{Y}$ and $\widetilde{Y}$ in ridge regression.

Appendix:

```
## Load libraries and set working directory
setwd("/Users/bstan/Documents/UW/Courses/BIOST 527")
rm(list = ls())
library(tidyverse)
library(tidyr)
library(tinytex)
library(class)
library(ISLR)
library(glmnet)

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

```
# Question 2b
##############

## Define soft threshold function
soft_threshold = function(rho,lam) {
  if(rho > lam) {
     return(rho-lam)
  } else if(rho < -lam) {
    return(rho+lam)
  } else {
    return(0)
  }
}

## Define UpdateCoefficient function
UpdateCoefficient <- function(x,y,lambda,beta,j){
  y_hat <- x %*% beta
  #alternatively, y_hat <- x[,-j] %*% beta[-j]
  delta <- t(x[,j]) %*% ((y-y_hat)+beta[j]*x[,j])
  #alternatively, delta <- t(x[,j]) %*% (y-y_hat)
  beta_j <- 1/sum(x[,j]^2)*soft_threshold(delta,lambda)
  beta[j] <- beta_j
  return(beta)
}

##############
# Question 2d
##############

## Define function to find L2 norm
norm_vec <- function(x) sqrt(sum(x^2))

## Define MyLasso
MyLasso <- function(x, y, lambda, maxiter=200, thr=1e-3){
  ## Initialize beta
  beta <- rep(0,ncol(x))
  beta.iter <- matrix(nrow=ncol(x), ncol=0)
  num_iter <- 0
  beta_diff <- 1
  ## Establish number of iterations and threshold criteria
  while ((num_iter <= maxiter) && (beta_diff >= thr)) {
    beta_init <- beta
    for (it in 1:length(beta)) {
      beta <- UpdateCoefficient(x,y,lambda,beta,it)
    }
    ## Add results of iteration to storing objects
    beta.iter <- cbind(beta.iter,beta)
    num_iter <- num_iter+1
    beta_diff <- norm_vec(beta-beta_init)/norm_vec(beta_init)
  }
  colnames(beta.iter) <- paste0(rep("iter",ncol(beta.iter)),
                                as.character(1:ncol(beta.iter)))
  ## Compute intercept
```

```r
  beta.intercept <- mean(y)-t(apply(x,2,mean)) %*% beta
  return(list(beta,beta.intercept,beta.iter))
}

##############
# Question 2e
##############

## Plug in values
set.seed(28)
n <- 100
p <- 20
x <- matrix(rnorm(n*p), ncol=p)
beta <- c(rep(3,5),rep(0,15))
y <- x %*% beta + rnorm(n)

## Evaluate MyLasso for a given value
samp_lambda <- 10
MyLasso_results <- MyLasso(x,y,samp_lambda)
MyLasso_results[2]
round(MyLasso_results[[1]],3)

##############
# Question 2f
##############
## Create empty lists
iter <- 0
lambda_list <- NULL
iter_list <- NULL
obj_values_list <- NULL
## Loop through values of lambda, adding lambda, number of iterations,
## and objective function value to respective lists
for (lamb in c(0.5,1,3,5,10)) {
  iter <- iter + 1
  obj_values <- NULL
  MyLasso_betas <- MyLasso(x,y,lamb)[[3]]
  ## Loop through the betas in each iteration
  for (col_i in 1:ncol(MyLasso_betas)) {
    obj_value_i <- 0.5*sum((y-x%*%MyLasso_betas[,col_i])^2)+lamb*sum(MyLasso_betas[,col_i])
    obj_values <- c(obj_values,obj_value_i)
  }
  lambda_list <- c(lambda_list,rep(lamb,ncol(MyLasso_betas)))
  iter_list <- c(iter_list,1:ncol(MyLasso_betas))
  obj_values_list <- c(obj_values_list,obj_values)
}
## Create data frame with all results
obj_function_df <- data.frame(lambda = as.factor(lambda_list),
                              iteration = iter_list,
                              obj_function_value = obj_values_list)
## Plot results
g1 <- ggplot() +
  geom_line(data=obj_function_df, aes(x=iteration, y=obj_function_value, color=lambda)) +
  labs(x='Iteration Number', y='Objective Function Value', color='Lambda') +
  ggtitle("Objective Function Value v. Iterations for Values of Lambda") +
  scale_x_continuous(breaks=c(0,1,2,3,4,5,6)) +
  theme_bw()
print(g1)

##############
# Question 2g
##############
## Run LS
ls <- lm(y~x)
summary(ls)$coefficients

## Run MyLasso for small lambda
samp_lambda <- 1e-5
MyLasso_results <- MyLasso(x,y,samp_lambda,maxiter = 3000,thr=1e-20)
MyLasso_results[2]
round(MyLasso_results[[1]],3)

##############
# Question 3a
##############

## Load data and remove NAs
#sum(is.na(Hitters))
Hitters=na.omit(Hitters)
#sum(is.na(Hitters))
Hitters <- data.frame(Hitters)

## Create x and y
x <- model.matrix(Salary~.,Hitters)[,-1]
y <- Hitters$Salary

## Run MyLasso and compute sum of betas for each lambda
lambda_list <- 10^seq(8.713,-3,length=100)
penalty_list <- NULL
for (lamb in lambda_list) {
  MyLasso_output <- MyLasso(x,y,lamb)[-3]
  penalty_i <- sum(abs(MyLasso_output[[1]]))
  penalty_list <- c(penalty_list,sum(penalty_i))
}

## Plot results
g2 <- ggplot() +
  geom_line(aes(x=log10(lambda_list), y=penalty_list)) +
  labs(x='log(lambda)', y='Penalty Value') +
  ggtitle("Penalty Value v. log(lambda) for Lasso Regression") +
  scale_x_continuous(breaks=c(-2,-0,2,4,6,8)) +
  theme_bw()
print(g2)

##############
# Question 3b
##############
```

```
## Iterate through lambdas, calculating beta values
lambda_list_plot <- NULL
betas_list <- NULL
names_list <- NULL
lambda_list <- 10^seq(8.713,-3,length=100)
for (lamb in lambda_list) {
  MyLasso_output <- MyLasso(x,y,lamb)[-3]
  betas_list <- c(betas_list,MyLasso_output[[1]])
  lambda_list_plot <- c(lambda_list_plot,rep(lamb,length(MyLasso_output[[1]])))
  names_list <- c(names_list,colnames(x))
}

## Create data frame with results
betas_df <- data.frame(lambda = lambda_list_plot,betas = betas_list,coef_name=names_list)
betas_df$value_category <- ifelse(betas_df$betas==0,"zero","non-zero")

## Plot results
g3b <- ggplot() +
  geom_line(data=betas_df, aes(x=log10(lambda), y=betas, color=coef_name), alpha = 0.4) +
  labs(x='log(lambda)', y='Coefficient Value', color='Covariate') +
  ggtitle("Coefficient Values v. log(lambda)") +
  theme_bw() +
  theme(legend.position = "bottom")
print(g3b)

## Plot coefficient values as dots with color indicating whether or not they equal zero
g3a <- ggplot() +
  geom_point(data=betas_df, aes(x=log10(lambda), y=betas, color=value_category), alpha = 0.4) +
  labs(x='log(lambda)', y='Coefficient Value', color='Value Category') +
  ggtitle("Coefficient Values v. log(lambda)") +
  theme_bw()
print(g3a)

##############
# Question 4a
##############

## Redefine x and y
x <- model.matrix(Salary~.,Hitters)[,-1]
y <- Hitters$Salary

samp_lambda <- 100 #10 #100
round(coef(glmnet(x,y,alpha=1,lambda=samp_lambda,standardize=TRUE)),3)
MyLasso_results <- MyLasso(x,y,samp_lambda,thr=1e-10)
MyLasso_results[2]
round(MyLasso_results[[1]],3)

## Create data frame to plot values of coefficients for MyLasso and glmnet
lambda_list <- 10^seq(5,-3,length=9)
model_list <- NULL
betas_list <- NULL
lambda_list_comp <- NULL
for (lamb in lambda_list) {
  MyLasso_output <- MyLasso(x,y,lamb)[-3]
  lasso_glm <- glmnet(x,y,alpha=1,lambda=lamb,standardize=TRUE)
  betas_list <- c(betas_list,MyLasso_output[[2]][1],MyLasso_output[[1]])
  model_list <- c(model_list,rep("MyLasso",length(MyLasso_output[[1]])+1))
  betas_list <- c(betas_list,coef(lasso_glm)[,1])
  model_list <- c(model_list,rep("GLM",length(MyLasso_output[[1]])+1))
  lambda_list_comp <- c(lambda_list_comp,rep(lamb,(length(MyLasso_output[[1]])+1)*2))
}
coef_comp_df <- data.frame("lambda" = lambda_list_comp,
                           "model" = model_list,
                           "beta" = betas_list)

## Plot coefficient values
g4 <- ggplot() +
  geom_point(data=coef_comp_df, aes(x=log10(lambda), y=beta, color=model), alpha = 0.4) +
  labs(x='log(lambda)', y='Coefficient Value', color='Model') +
  ggtitle("Coefficient Values v. log(lambda) for MyLasso and GLM") +
  theme_bw()
print(g4)

## Create data frame compute differences between MyLasso and glmnet
lambda_list_comp <- NULL
betas_list_mylasso <- NULL
betas_list_glm <- NULL
for (lamb in lambda_list) {
  MyLasso_output <- MyLasso(x,y,lamb)[-3]
  lasso_glm <- glmnet(x,y,alpha=1,lambda=lamb,standardize=TRUE)
  betas_list_mylasso <- c(betas_list_mylasso,MyLasso_output[[2]][1],MyLasso_output[[1]])
  betas_list_glm <- c(betas_list_glm,coef(lasso_glm)[,1])
  lambda_list_comp <- c(lambda_list_comp,rep(lamb,length(MyLasso_output[[1]])+1))
}
coef_comp_df <- data.frame("lambda" = lambda_list_comp,
                           "glm" = betas_list_glm,
                           "MyLasso" = betas_list_mylasso)
coef_comp_df$difference <- coef_comp_df$MyLasso - coef_comp_df$glm

## Plot differences
g5 <- ggplot() +
  geom_point(data=coef_comp_df, aes(x=log10(lambda), y=difference), alpha = 0.4) +
  labs(x='log(lambda)', y='Difference in Coefficient Values (MyLasso - GLM)') +
  ggtitle("Difference in MyLasso and GLM Coefficients v. log(lambda)") +
  theme_bw() +
  stat_summary(data=coef_comp_df,aes(x=log10(lambda), y=difference, group=1),
               fun=mean,
               geom="line",
               group=1)
print(g5)

##############
# Question 4b
##############

## Compute coefficient differences
```

```
lambda_list <- 10^seq(2,-2,length=9)
lambda_list_comp <- NULL
betas_list_mylasso <- NULL
betas_list_glm <- NULL
for (lamb in lambda_list) {
  ## Input scaled values
  MyLasso_output <- MyLasso(scale(x),scale(y),lamb,maxiter=3000, thr=1e-10)[-3]
  lasso_glm <- glmnet(scale(x),scale(y),alpha=1,lambda=lamb,standardize=FALSE)
  betas_list_mylasso <- c(betas_list_mylasso,MyLasso_output[[2]][1],MyLasso_output[[1]])
  betas_list_glm <- c(betas_list_glm,coef(lasso_glm)[,1])
  lambda_list_comp <- c(lambda_list_comp,rep(lamb,length(MyLasso_output[[1]])+1))
}
coef_comp_df <- data.frame("lambda" = lambda_list_comp,
                           "glm" = betas_list_glm,
                           "MyLasso" = betas_list_mylasso)
coef_comp_df$difference <- coef_comp_df$MyLasso - coef_comp_df$glm

## Plot coefficient differences
g6 <- ggplot() +
  geom_point(data=coef_comp_df, aes(x=log10(lambda), y=difference), alpha = 0.4) +
  labs(x='log(lambda)', y='Difference in Coefficient Values (MyLasso - GLM)') +
  ggtitle("Difference in Coefficients v. log(lambda) with Scaled Inputs") +
  theme_bw() +
  stat_summary(data=coef_comp_df,aes(x=log10(lambda), y=difference, group=1),
               fun=mean,
               geom="line",
               group=1)
print(g6)

##############
# Question 7e
##############

## Define data
set.seed(44)
x1 <- rnorm(100,10,1)
x2 <- rnorm(100,0,1)
X <- cbind("ones"=rep(1,100),"x1"=x1,"x2"=x2)
X_scaled <- cbind("ones"=rep(1,100),"x1"=x1*100,"x2"=x2)
y <- runif(100,0,10)

## B
beta <- solve(t(X) %*% X) %*% t(X) %*% y
beta_scaled <- solve(t(X_scaled) %*% X_scaled) %*% t(X_scaled) %*% y
print(beta)
print(beta_scaled)

## C
y_hat <- X %*% beta
y_hat_scaled <- X_scaled %*% beta_scaled
identical(round(y_hat,10),round(y_hat_scaled,10)) # TRUE; rounding needed to show equality

## A
```

```
sum((y-y_hat)^2)
sum((y-y_hat_scaled)^2)

## D
lambda = 500
y_hat_ridge <- X %*% solve(t(X) %*% X + lambda * diag(3)) %*% t(X) %*% y
y_hat_ridge_scaled <- X_scaled %*% solve(t(X_scaled) %*% X_scaled + lambda * diag(3)) %*% t(X_scaled) %*
scaled_comp_df <- data.frame(regular = y_hat_ridge, scaled = y_hat_ridge_scaled)
scaled_comp_df$difference <- scaled_comp_df$regular - scaled_comp_df$scaled

g7 <- ggplot() +
  geom_point(data=scaled_comp_df, aes(x=regular, y=difference), alpha = 0.4) +
  labs(x='Non-Scaled Value', y='Difference in Fitted Values (Non-Scaled - Scaled)') +
  ggtitle("Difference in Fitted Values v. Fitted Non-Scaled Values") +
  theme_bw()
print(g7)

##############
# Question 7f
##############

## Compute scaled and non-scaled fitted values with KNN and creates dataframe
pred_vals_non_scaled <- FNN::knn.reg(train=X,
                               test=X,
                               y=y,k=3)$pred
pred_vals_scaled <- FNN::knn.reg(train=X_scaled,
                               test=X_scaled,
                               y=y,k=3)$pred
pred_vals_df <- data.frame(nonscaled = pred_vals_non_scaled,
                           scaled = pred_vals_scaled)

## Plot values to see if they fit on a diagonal line
g8 <- ggplot() +
  geom_point(data=pred_vals_df, aes(x=nonscaled, y=scaled), alpha = 0.4) +
  labs(x='Non-scaled Value', y='Scaled Value') +
  ggtitle("Scaled v. Non-Scaled Fitted Values for KNN with k=3") +
  theme_bw()
print(g8+geom_abline(slope = 1,intercept = 0))
```