Student Name: Sophie Stelmach
Student ID: 400172067

**Stats 790**
Assignment 2

---

1. a) Derivation of linear regression coefficients.

**Naive Linear Algebra**

Given $Y = X\beta + \epsilon$, we want to solve for $\beta$ by minimizing the sum of squared residuals $\sum \epsilon_i^2 = \epsilon^T \epsilon$.

$$\begin{aligned}
\epsilon^T \epsilon &= (Y - X\beta)^T (Y - X\beta) \\
&= (Y^T - \beta^T X^T)(Y - X\beta) \\
&= Y^T Y - \beta^T X^T Y - Y^T X\beta + \beta^T X^T X\beta
\end{aligned}$$

We take the derivative of this equation with respect to $\beta$:

$$\begin{aligned}
&= -X^T Y - Y^T X + X^T X\beta \\
&= -2X(Y - X\beta)
\end{aligned}$$

Setting this equal to 0 we get:

$$\begin{aligned}
0 &= -2X^T (Y - X\beta) \\
0 &= X^T Y - X^T X\beta \\
X^T X\beta &= X^T Y \\
(X^T X)^{-1} X^T X\beta &= (X^T X)^{-1} X^T Y \\
\beta &= (X^T X)^{-1} X^T Y
\end{aligned}$$

**QR Decomposition**

We want $A = QR$ where Q is an orthogonal matrix ($Q^T Q = I$), and R is upper triangular. Let $X = QR$.

Starting from the result from earlier:

$$\begin{aligned}
X^T X\beta &= X^T Y \\
(QR)^T (QR)\beta &= (QR)^T Y \\
R^T Q^T QR\beta &= R^T Q^T Y \\
R^T R\beta &= R^T Q^T Y \\
R\beta &= Q^T Y
\end{aligned}$$

Following which we backwards solve for $\beta$.

**SVD**

We will let $X = UDV^T$, where $D$ is a diagonal matrix, $V$ is orthogonal. We then know the following:

$$
\begin{aligned}
Y &= X\beta \\
Y &= UDV^T\beta \\
(UDV^T)^{-1}Y &= (UDV^T)^{-1}UDV^T\beta \\
VD^{-1}U^TY &= VD^{-1}U^T(UDV^T)\beta \\
VD^{-1}U^TY &= \beta
\end{aligned}
$$

Which we can solve for in R.

**Cholesky Decomposition**

We want $A = LL^T$ where $A$ is a symmetric and positive definite matrix and $L$ is a lower triangular matrix.

$$
\begin{aligned}
X^TY &= X^TX\beta \\
X^TY &= LL^T\beta \\
X^TY &= L(L^T\beta) \\
X^TY &= L^T\beta
\end{aligned}
$$

Following which you backwards solve the upper triangular matrix $L^T$.

b) Implementation in R.

First, we load packages needed.

```
library(dplyr)
library(magrittr)
library(readr)
library(ggplot2)
library(ISLR2)
library(leaps)
library(glmnet)
library(pls)
library(matlib)
library(microbenchmark)
set.seed(5)
```

Then, we can create functions for solving linear regression using naive linear algebra, Cholesky decomposition, and QR decomposition.

```
# naive linear algebra;

linalg <- function(X1,y1) {
  xtr <- t(X1)
  b_la <- (inv(xtr %*% X1) %*% xtr %*% y1)
```

```
  yhat_la <- X1 %*% b_la
}

# Cholesky decomposition;

cholesky <- function(X1,y1){
  xtr <- t(X1)
  L <- chol(xtr %*% X1)
  cp <- crossprod(X1, y1)
  a <- xtr %*% X1 %*% cp

  b_cd <- backsolve(L, a, transpose = T)
  yhat_cd <- X1 %*% b_cd
}

# QR decomposition;

qrd <- function(X1,y1){
  qrdecomp <- qr(X1)
  qr_q <- qr.Q(qrdecomp)
  qr_r <- qr.R(qrdecomp)
  qr_qty <- crossprod(qr_q,y1)

  b_qrd <- backsolve(qr_r, qr_qty)

  yhat_qrd <- X1 %*% b_qrd
}
```

Next, we will write a function to run each of the above functions and time them.

```
# create function to run each function above for various n, p

solve_lr <- function(n, p){
  X <- matrix(rnorm(n*p),ncol=p) #matrix x, p*n x p
  y <- matrix(rnorm(n),nrow=n) #vector y, 1 x n
  list(X=X,y=y)

  qrd(X,y)
  cholesky(X,y)
  linalg(X,y)

  timing <- microbenchmark(qrd(X,y),
                           cholesky(X,y),
                           linalg(X,y))
  summ <- summary(timing)
  results <- c(summ$mean,n,p)
  df <<- rbind(df,results)
  #return(df)
}
```

Then, we can create a data frame of average times for each method of solving and use various values of $n$ and $p$.

```
ns <- round(10^seq(2, 5, by = 0.25))
ps <- round(2^seq(2, 5, by = 0.25))
```

```
df <- data.frame()

for (i in seq_along(ns)) {
  solve_lr(ns[i], p=10)
}

for (i in seq_along(ps)) {
  solve_lr(n=100, ps[i])
}

names(df) = c("QRD","CD","LA","n","p")
```

We can create plots to show the log-linear relationship between $n$ and time and $p$ and time.

```
    df %>% filter(p==10) %>%
  ggplot() +
  geom_point(aes(x = log(QRD), y = log(n)), color = "pink") +
  geom_point(aes(x = log(CD), y = log(n)), color = "blue") +
  geom_point(aes(x = log(LA), y = log(n)), color = "purple")+
  xlab("Time")+
  ylab("n")+
  theme_bw()

df %>% filter(n==100) %>%
  ggplot() +
  geom_point(aes(x = log(QRD), y = log(p)), color = "pink") +
  geom_point(aes(x = log(CD), y = log(p)), color = "blue") +
  geom_point(aes(x = log(LA), y = log(p)), color = "purple")+
  xlab("Time")+
  ylab("p")+
  theme_bw()
```

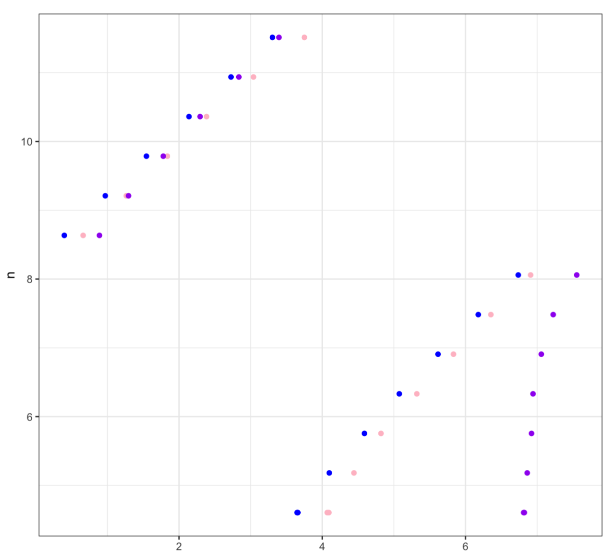



Figure 1: Plot of log(n) against log(time) for Cholesky decomposition(blue), QR decomposition (pink) and linear algebra solving (purple) for p=10.

Lastly, we will create log-log models of average time using the value of $n$.

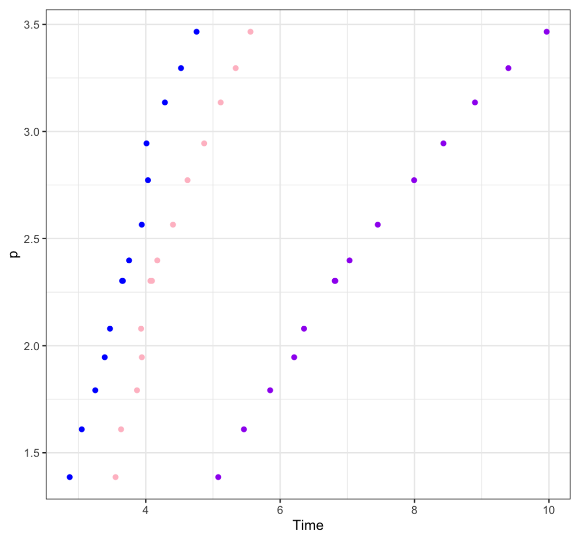



Figure 2: Plot of log(p) against log(time) for Cholesky decomposition(blue), QR decomposition (pink) and linear algebra solving (purple) for n=100.

```
#create log log model of average times to n

df2 <- df %>% filter(p==10)

lmqr <- lm(log(QRD) ~ log(n), data= df2)
summary(lmqr)

Call:
lm(formula = log(QRD) ~ log(n), data = df2)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9526 -1.0021 -0.1323  1.1035  3.0860

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.7491     1.6985   3.974  0.00185 **
log(n)       -0.3631     0.2089  -1.738  0.10769
---
\Signif. codes:  0  ***   0.001   **   0.01   *   0.05   .   0.1       1

Residual standard error: 1.765 on 12 degrees of freedom
Multiple R-squared:  0.2012,  Adjusted R-squared:  0.1346
F-statistic: 3.022 on 1 and 12 DF,  p-value: 0.1077

lmcd <- lm(log(CD) ~ log(n), data= df2)
summary(lmcd)

Call:
lm(formula = log(CD) ~ log(n), data = df2)

Residuals:
    Min      1Q  Median      3Q     Max
-2.9258 -1.1234 -0.1202  0.9922  3.2052
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.4314     1.7370   3.702  0.00302 **
log(n)       -0.3599     0.2136  -1.685  0.11786
---
\Signif. codes:  0  ***   0.001   **   0.01   *   0.05   .   0.1       1

Residual standard error: 1.805 on 12 degrees of freedom
Multiple R-squared:  0.1913,  Adjusted R-squared:  0.1239
F-statistic: 2.838 on 1 and 12 DF,  p-value: 0.1179

lmla <- lm(log(LA) ~ log(n), data= df2)
summary(lmla)

Call:
lm(formula = log(LA) ~ log(n), data = df2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3274 -0.7810  0.0206  1.2441  2.8536

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.4564     1.7239   6.646 2.38e-05 ***
log(n)       -0.8385     0.2120  -3.955  0.00191 **
---
\Signif. codes:  0  ***   0.001   **   0.01   *   0.05   .   0.1       1

Residual standard error: 1.791 on 12 degrees of freedom
Multiple R-squared:  0.5659,  Adjusted R-squared:  0.5298
F-statistic: 15.65 on 1 and 12 DF,  p-value: 0.001909
```

The linear algebra method is the only one that has the $log(n)$ term significant in the model. All models have a coefficient value for $log(n)$ between $-1$ and 0, so the models display a relatively linear relationship. R-squared is much higher for linear algebra than for the other models, indicating a stronger linear relationship.

2. First, we will import necessary packages.

```
library(dplyr)
library(magrittr)
library(readr)
library(ggplot2)
library(ISLR2)
library(leaps)
library(glmnet)
library(pls)
```

Next, the data is imported and scaled as described in the data information.

```
data <- read.table('prostatedata.txt')

train <- data %>% filter(train == TRUE)
test <- data %>% filter(train == FALSE)

Xmat <- model.matrix(lpsa ~ ., data)[, -10]
preds <- data[,1:8]
p <- ncol(Xmat)
y <- data$lpsa
xscaled <- scale(preds,TRUE,TRUE)
```

For data augmentation, we append $\sqrt{\lambda}I$ to X (where $\lambda = 0.01$ to get $X = \left(\begin{smallmatrix} X \\ \sqrt{\lambda}I \end{smallmatrix}\right)$, and $p$ zeros to $Y$.

```
# append 0's to y vector and sqrt(lambda)*I to x
os <- matrix(0, (dim(xaug)[1]-length(y)), 1)
yaug <- rbind(y, os)

aug <- sqrt(0.01) * diag(8)
xaug <- rbind(xscaled, aug)

ridge_aug <- glmnet(xaug, yaug, alpha = 0)
```

For the native implementation of ridge regression, we use the scaled matrix of predictors and the response column.

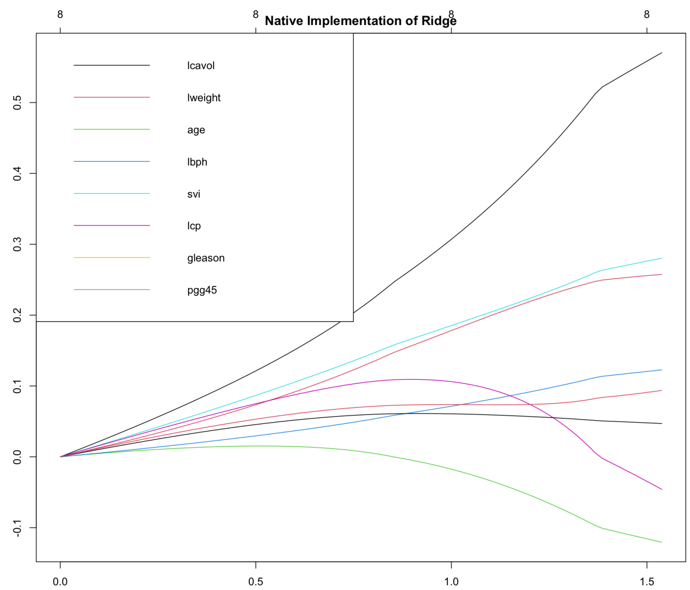
```
#native implementation of ridge regression
ridge_nat <- glmnet(xscaled, y, family = gaussian, alpha = 0)
```

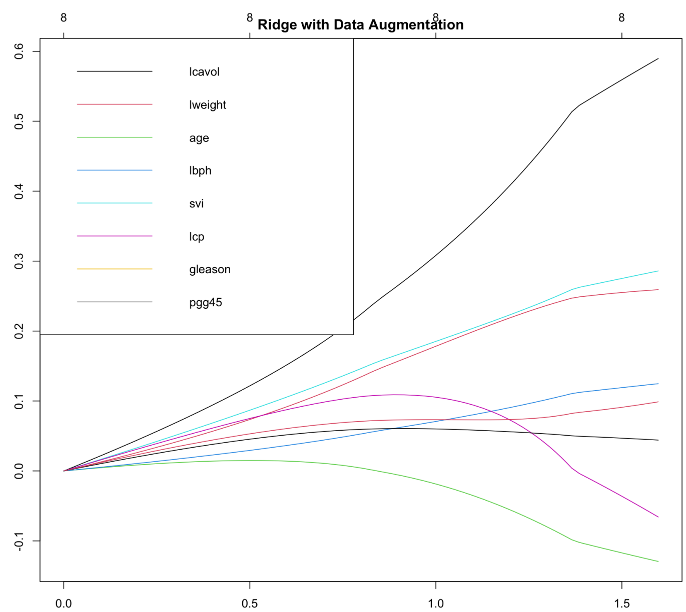
Next, we will examine the plots of both ridge models to compare. As we can see below, the plots look pretty much identical.

```
plot(ridge_aug, main = "Ridge with Data Augmentation")
legend("topleft", legend = rownames(coef(ridge_aug))[-1], lty = 1, col = 1:p)

plot(ridge_nat, main = "Native Implementation of Ridge")
legend("topleft", legend = rownames(coef(ridge_nat))[-1], lty = 1, col = 1:p)
```

Looking at timing for each method, the data augmentation method is much faster than native ridge implementation.

```
library(microbenchmark)

ridgetime <- microbenchmark(
  ridge_nat = glmnet(xscaled, y, family = gaussian, alpha = 0),
```




```
  ridge_aug = glmnet(xaug, yaug, alpha = 0)
)

ridgetime

Unit: microseconds
      expr       min         lq       mean     median         uq
 ridge_nat 79390.596 81461.8340 86378.4503 83297.3220 91084.4315
 ridge_aug   490.483   516.3745   587.1573   557.0055   653.6015
        max neval
 101190.542   100
    823.649   100
```

3. Ex. 3.6

From Equation 3.41 and 3.43 in ESL, we have:

$$\hat{\beta}^{ridge} = argmin_{\beta} \left[ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

$$RSS(\lambda) = (Y - X\beta)^T (Y - X\beta) + \lambda \beta^T \beta$$

We can begin by taking the product of the Gaussian distributions:

$$\begin{aligned} P(\beta|D) &= N(X^T\beta, \sigma^2 I) * N(0, \tau I) \\ &= \frac{1}{\sigma\sqrt{2\pi}} exp\left(\frac{-(Y - X^T\beta)^2}{2\sigma^2}\right) * \frac{1}{\tau\sqrt{2\pi}} exp\left(\frac{-(Y-0)^2}{2\tau}\right) \end{aligned}$$

We can take the log of the likelihood:

$$\begin{aligned} &= log\left(\frac{1}{\sigma\sqrt{2\pi}} exp\left(\frac{-(Y - X^T\beta)^2}{2\sigma^2}\right)\right) + log\left(\frac{1}{\tau\sqrt{2\pi}} exp\left(\frac{-(Y-0)^2}{2\tau}\right)\right) \\ &= \frac{-(Y - X^T\beta)^2}{2\sigma^2} - \frac{Y^2}{2\tau} \\ &= \frac{-(Y - X^T\beta)^T(Y - X^T\beta)}{2\sigma^2} - \frac{\beta^T\beta}{2\tau} \end{aligned}$$

Then,

$$\begin{aligned} &argmin_{\beta} \left[ \frac{-(Y - X^T\beta)^T(Y - X^T\beta)}{2\sigma^2} - \frac{\beta^T\beta}{2\tau} \right] \\ &= RSS + \lambda\beta^T\beta \\ &= RSS + \lambda \sum \beta_j^2 \end{aligned}$$

Where $\lambda = 2\sigma^2/\tau$. Thus, this is the ridge estimate.

4. Ex. 3.19

From Equation 3.44 in ESL we have:

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$

Using SVD and setting $X = UDV^T$, this becomes,

$$\begin{aligned}
&= ((UDV^T)^T (UDV^T) + \lambda I)^{-1} (UDV^T)^T Y \\
&= (VD^T U^T + \lambda I)^{-1} (VD^T U^T) Y \\
&= (VDU^T UDV^T + \lambda I)^{-1} (VDU^T) Y \\
&= D^2 + \lambda I)^{-1} (VDU^T) Y
\end{aligned}$$

$$\begin{aligned}
||\hat{\beta}^{ridge}||^2 &= (D^2 + \lambda I)^{-1} (VDU^T) Y)^2 \\
&= Y^T (VDU^T)^T (D^2 + \lambda I)^{-2} (VDU^T) Y \\
&= Y^T (UD)(D^2 + \lambda I)^{-2} (DU^T) Y \\
&= \sum_{j=1}^{p} y u_j d_j \frac{1}{(dj + \lambda)^2} d_j u_j y \\
&= \sum_{j=1}^{p} \frac{y u_j d_j d_j u_j y}{(dj + \lambda)^2}
\end{aligned}$$

Which increases as $\lambda$ goes to 0.

**Lasso**

We can look at table 3.4 to see the estimators in the case of orthonormal columns of X. For lasso, this is $sign(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$. AS $\lambda$ goees to zero, $(|\hat{\beta}_j| - \lambda)$ will increase since $\lambda$ is getting smaller, and so the entire estimator will increase.

This can be seen for the ridge estimator as well, $\hat{\beta}_j/1 + \lambda$, so as $\lambda$ goes to 0 the estimator increases.

5. Ex. 3.28

The lasso solution is given by Equation 3.52 in ESL:

$$\hat{\beta}^{lasso} = argmin_{\beta}\left[1/2\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right]$$

Subject to $\sum_{j=1}^{p}|\beta_j| <= t$.

Let $X' = [X_j X_j^*]$ and $\beta^{*\prime} = [\beta_j \beta_j^*]$so that $X\beta^* = X\beta_j + X\beta_J^*$.

Then we have

$$\begin{aligned}
\hat{\beta}^{lasso} &= argmin_{\beta}\left[1/2\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right] \\
&= argmin_{\beta}\left[1/2\sum_{i=1}^{N}(y_i - \beta_0 - x\beta_j - x\beta_j^*)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right] \\
&= argmin_{\beta}\left[1/2\sum_{i=1}^{N}(y_i - \beta_0 - x(\beta_j + \beta_j^*))^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right] \\
&= argmin_{\beta}\left[1/2\sum_{i=1}^{N}(y_i - \beta_0 - x(\beta_j + \beta_j^*))^2 + \lambda(|\beta_j| + |\beta_j^*|)\right]
\end{aligned}$$

Subject to $\sum_{j=1}^{p}|\beta_j| <= t$ i.e., $(|\beta_j| + |\beta_j^*|) <= t$. and $(|\beta_j| + a) <= t$.

6. Ex. 3.30

Let $X = \binom{X}{\lambda I}$, and $Y = \binom{y}{0}$.

We want to show how this can turn into the lasso problem.

$$min||y - X\beta||^2 + \lambda[\alpha||\beta||_2^2 + (1-\alpha)||\beta||_1]$$

$$\begin{aligned}
&min||y - X\beta||^2 \\
&= ||y - X\beta||_2^2 + ||0 - \lambda\beta||_2^2 \\
&= ||y - X\beta||_2^2 + ||\lambda\beta||_2^2 \\
&= ||y - X\beta||_2^2 + \lambda||\beta||_2^2
\end{aligned}$$

Setting $\alpha$ to 1 we get:

$$\begin{aligned}
&= ||y - X\beta||_2^2 + \lambda||\beta||_2^2 + \lambda||\beta||_2^2 \\
&= ||y - X\beta||_2^2 + \lambda||\beta||_2^2
\end{aligned}$$

Which is equal to $\hat{\beta}^{lasso}$.