

STATS790 A2

Hanwen Ju

February 2023

1 Question 1(a)

Derive/show how to compute linear regression coefficients (for general choice of y and X) using the following four methods: naive linear algebra; QR decomposition; SVD; and Cholesky decomposition.

Consider residual sum-of-squares of the linear regression model:

$$\mathbf{RSS}(\beta) = (y - \mathbf{X}\beta)^T(y - \mathbf{X}\beta) \quad (1)$$

We want to minimize it by differentiating with respect to β :

$$\frac{\partial \mathbf{RSS}}{\partial \beta} = -2\mathbf{X}^T(y - \mathbf{X}\beta) \quad (2)$$

So set it to 0 to obtain the normal equation:

$$\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^Ty \quad (3)$$

We can derive the linear regression coefficients by using the following 4 methods.

- Naive linear algebra

$$\begin{aligned} \mathbf{X}^T\mathbf{X}\beta &= \mathbf{X}^Ty \\ \hat{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^Ty \end{aligned} \quad (4)$$

- QR decomposition

By the definition of QR decomposition. For a real $n \times p$ matrix A , where $n \geq p$, matrix A can be decomposed into two matrices, A and Q :

$$A = QR \quad (5)$$

where $Q^TQ = I_p$ and R is an $p \times p$ upper triangular matrix. Hence, we can decompose the matrix X into QR to get:

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \beta &= \mathbf{X}^T y \\
(QR)^T (QR) \beta &= (QR)^T y \\
R^T Q^T Q R \beta &= R^T Q^T y \\
R^T R \beta &= R^T Q^T y \\
R \beta &= Q^T y \\
\beta &= R^{-1} Q^T y
\end{aligned} \tag{6}$$

- SVD decomposition

By definition of SVD decomposition. A $n \times p$ matrix A can be decomposed into the following form:

$$A = U \Sigma V^T \tag{7}$$

where U and V are $n \times p$ and $p \times p$ orthogonal matrices, and D is a $p \times p$ diagonal matrix contain singular value of A . And $U^T U = I$ and $V V^T = I$. So we can decompose X to obtain:

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \beta &= \mathbf{X}^T y \\
(U \Sigma V^T)^T (U \Sigma V^T) \beta &= (U \Sigma V^T)^T y \\
V \Sigma^T U^T U \Sigma V^T \beta &= (U \Sigma V^T)^T y \\
V \Sigma^T \Sigma V^T \beta &= V \Sigma^T U^T y \\
\Sigma V^T \beta &= U^T y \\
V^T \beta &= \Sigma^{-1} U^T y \\
V V^T \beta &= V \Sigma^{-1} U^T y \\
\beta &= V \Sigma^{-1} U^T y
\end{aligned} \tag{8}$$

where $V \Sigma^{-1} U^T$ is the pseudo-inverse of X .

- Cholesky decomposition

By definition of Cholesky decomposition. If the matrix A is positive definite and symmetric, then there exists a lower triangular matrix, L , such that

$$A = L L^T \tag{9}$$

We know $X^T X$ is symmetric. If it is also positive definite(full rank), then we can decompose it as $\mathbf{X}^T \mathbf{X} = L L^T$ to obtain:

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \beta &= \mathbf{X}^T y \\
L L^T \beta &= X^T y \\
L(L^T \beta) &= X^T y
\end{aligned} \tag{10}$$

Let $c = L^T \beta$, we first solve $Lc = X^T y$ to get:

$$c = L^{-1} X^T y \quad (11)$$

Then solve $L^T \beta = c$ to get the regression coefficient:

$$\beta = (L^T)^{-1} c \quad (12)$$

2 Question 2

Suppose augmenting X and y into \mathbf{B} and \mathbf{y}^* :

$$\mathbf{B} = \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda} \mathbf{I} \end{bmatrix} \quad \mathbf{y}^* = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \quad (13)$$

We can solve it using standard linear regression by QR decomposition from Q1. Let:

$$\mathbf{B} = \mathbf{QR} \quad (14)$$

The normal equation will be:

$$\begin{aligned} \mathbf{B}^T \mathbf{B} \beta &= \mathbf{B}^T \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \\ \mathbf{B}^T \mathbf{B} \beta &= \mathbf{B}^T \mathbf{y}^* \\ (\mathbf{QR})^T (\mathbf{QR}) \beta &= (\mathbf{QR})^T \mathbf{y}^* \end{aligned} \quad (15)$$

Thus, β will be:

$$\beta^{ridge} = \mathbf{R}^{-1} \mathbf{Q}^T \mathbf{y}^* \quad (16)$$

3 ESL 3.6

Considering the posterior distribution of β^{ridge} :

$$\mathbf{P}(\beta | \mathbf{y}, \mathbf{X}) = \frac{\mathbf{P}(\mathbf{y} | \beta, \mathbf{X}) \mathbf{P}(\beta)}{\mathbf{P}(\mathbf{y} | \mathbf{X})} \propto \mathbf{P}(\mathbf{y} | \beta, \mathbf{X}) \mathbf{P}(\beta) \quad (17)$$

If we choose a Gaussian prior $\beta \sim \mathcal{N}(0, \tau \mathbf{I})$ and $y \sim \mathcal{N}(\mathbf{X}\beta, \sigma^2 \mathbf{I})$, we see that:

$$\begin{aligned} \mathbf{P}(\beta | \mathbf{y}, \mathbf{X}) &\propto \mathbf{P}(\mathbf{y} | \beta, \mathbf{X}) \mathbf{P}(\beta) \\ &\propto \frac{1}{(2\pi)^{\frac{p}{2}} |\sigma|} \exp \left(-\frac{(y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta)}{2\sigma^2} \right) \frac{1}{(2\pi)^{\frac{p}{2}} |\tau|^{\frac{1}{2}}} \exp \left(-\frac{\beta^T \beta}{2\tau} \right) \\ &\propto \exp \left(-\frac{(y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta)}{2\sigma^2} \right) \exp \left(-\frac{\beta^T \beta}{2\tau} \right) \end{aligned} \quad (18)$$

We can take the log to **MAP** and ignore the constant:

$$\begin{aligned}
\arg \max_{\beta} (\log \mathbf{P}(\beta | \mathbf{y}, \mathbf{X})) &= \arg \max_{\beta} \left(\log \left(\exp \left(-\frac{(y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta)}{2\sigma^2} \right) \right) + \log \exp \left(-\frac{\beta^T \beta}{2\tau} \right) \right) \\
&= \arg \max_{\beta} \left(-\frac{(y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta)}{2\sigma^2} - \frac{\beta^T \beta}{2\tau} \right) \\
&= \arg \max_{\beta} \left(-\frac{1}{\sigma^2} \left((y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta) + \frac{\sigma^2}{\tau} \beta^T \beta \right) \right) \\
&= \arg \min_{\beta} \left((y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta) + \frac{\sigma^2}{\tau} \beta^T \beta \right)
\end{aligned} \tag{19}$$

which is the same for the ridge regression in matrix form. Here $\lambda = \frac{\sigma^2}{\tau}$, we can also derive the β :

$$\beta^{ridge} = (\mathbf{X}^T \mathbf{X} + \frac{\sigma^2}{\tau} \mathbf{I})^{-1} \mathbf{X}^T y \tag{20}$$

4 ESL 3.19

By using SVD decomposition to solve ridge regression, we can obtain:

$$\begin{aligned}
\beta^{ridge} &= (X^T X + \lambda I)^{-1} X^T y \\
&= ((V \Sigma^T U^T)(U \Sigma V^T) + \lambda V V^T)^{-1} V \Sigma U^T y \\
&= (V \Sigma^2 V^T + \lambda V V^T)^{-1} V \Sigma U^T y \\
&= ((V \Sigma^2 + \lambda V) V^T)^{-1} V \Sigma U^T y \\
&= (V^T)^{-1} (\Sigma^2 + \lambda I)^{-1} V^{-1} V \Sigma U^T y \\
&= V (\Sigma^2 + \lambda I)^{-1} \Sigma U^T y
\end{aligned} \tag{21}$$

Then, consider the L2 norm of β^{ridge} :

$$\begin{aligned}
\|\beta^{ridge}\| &= \sqrt{\beta^{ridge T} \beta^{ridge}} \\
&= \sqrt{y^T U \Sigma (\Sigma^2 + \lambda I)^{-1} V^T V (\Sigma^2 + \lambda I)^{-1} \Sigma U^T y} \\
&= \sqrt{y^T U \Sigma (\Sigma^2 + \lambda I)^{-2} \Sigma U^T y} \\
&= \sqrt{(U^T y)^T \Sigma (\Sigma^2 + \lambda I)^{-2} \Sigma U^T y} \\
&= \sqrt{(U^T y)^T \frac{\Sigma^2}{(\Sigma^2 + \lambda I)^2} U^T y} \\
&= \sqrt{\sum_{j=1}^p (U^T y)_j^2 \frac{d_j^2}{(d_j^2 + \lambda)^2}}
\end{aligned} \tag{22}$$

Since $\lambda \geq 0$, it's easy to see $0 \leq \frac{d_j^2}{(d_j^2 + \lambda)^2} \leq 1$. As $\lambda \rightarrow 0$, the fraction $\frac{d_j^2}{(d_j^2 + \lambda)^2} \rightarrow 1$. And $\|\beta^{ridge}\|$ also increases. This also holds for lasso regression. Consider the constraint optimization of lasso regression:

$$\begin{aligned} \beta^{lasso} = \arg \min_{\beta} & \left(\sum_{i=1}^N \left(y_i - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right) \\ \text{s.t. } & \sum_j^p |\beta_j| \leq t \end{aligned} \quad (23)$$

When λ decrease, t will increase. The feasible region for $\sum_j^p |\beta_j|$ will also increase. So smaller λ will allow the model to select a larger β . Another way to explain this is to consider the other format, which is equivalent to the above minimization problem of the lasso:

$$\beta^{lasso} = \arg \min_{\beta} \left(\sum_{i=1}^N \left(y_i - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \right) \quad (24)$$

If we have a larger complexity parameter λ , the shrinkage will be larger. This will shrink the coefficient toward 0 and vice versa. Hence, we can conclude $\|\beta\|$ increases as its tuning parameter approach 0.

5 ESL 3.28

Suppose t is fixed. The original lasso regression can be written as:

$$\beta^{lasso} = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}^{ori} \beta^{ori}\|_2^2 + \lambda \|\beta^{ori}\|_1 \quad (25)$$

If we add $X_j^* = X_j$, we will obtain

$$\beta^{new} = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}^{new} \beta^{new}\|_2^2 + \lambda \sum_{k=0}^p |\beta_k^{ori}| + \lambda |\beta_j^*| \quad (26)$$

Let $\beta_j^c = \beta_j + \beta_j^*$, we can rewrite the above equation in the following way:

$$\beta^{lasso} = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}^{new} \beta^{new}\|_2^2 + \lambda \sum_{k \neq j}^p |\beta_k| + \lambda |\beta_j^c| + (\lambda |\beta_j| + \lambda |\beta_j^*| - \lambda |\beta_j^c|) \quad (27)$$

By triangle inequality:

$$|\beta_j| + |\beta_j^*| \geq |\beta_j + \beta_j^*| = |\beta_j^c| \quad (28)$$

The later term in equation(27) $\lambda |\beta_j| + \lambda |\beta_j^*| - \lambda |\beta_j^c|$ has to be positive. Also, $\beta_j = \beta_j^*$. Given $\beta_j^{lasso} = a$, the term $\lambda |\beta_j| + \lambda |\beta_j^*| - \lambda |\beta_j^c|$ becomes 0. This implies $\beta_j = \beta_j^* = \frac{a}{2}$ give the optimal solution.

6 ESL 3.30

We want to form a lasso problem in the space of augmented space. Assume we are augmenting \mathbf{X} and \mathbf{y} into \mathbf{B} and \mathbf{y}^* in the general form.

$$\mathbf{B} = \begin{bmatrix} \mathbf{X} \\ \eta \mathbf{I} \end{bmatrix} \quad \mathbf{y}^* = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \quad (29)$$

We can form the minimization problem as:

$$\begin{aligned} \beta &= \arg \min_{\beta} \|\mathbf{y}^* - \mathbf{B}\beta\|_2^2 \\ &= \arg \min_{\beta} \left\| \begin{bmatrix} \mathbf{y} - \mathbf{X}\beta \\ 0 - \eta\beta \end{bmatrix} \right\|_2^2 \\ &= \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \|0 - \eta\beta\|_2^2 \\ &= \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \eta^2 \|\beta\|_2^2 \end{aligned} \quad (30)$$

Clearly, this is the ridge regression (L2 norm). We can also form a lasso regression in the augmented space. And replace the minimization with the above ridge regression since $\|\mathbf{y}^* - \mathbf{B}\beta\|_2^2 = \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \eta^2 \|\beta\|_2^2$.

$$\begin{aligned} \beta^{lasso} &= \arg \min_{\beta} \|\mathbf{y}^* - \mathbf{B}\beta\|_2^2 + \delta \|\beta\|_1 \\ &= \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \eta^2 \|\beta\|_2^2 + \delta \|\beta\|_1 \end{aligned} \quad (31)$$

Consider the elastic net:

$$\begin{aligned} &\arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda[\alpha \|\beta\|_2^2 + (1 - \alpha) \|\beta\|_1] \\ &= \arg \min_{\beta} \|y - X\beta\|_2^2 + \lambda\alpha \|\beta\|_2^2 + \lambda(1 - \alpha) \|\beta\|_1 \end{aligned} \quad (32)$$

Let $\eta = \sqrt{\lambda\alpha}$ and $\delta = \lambda(1 - \alpha)$. Then we successfully convert the elastic net to a lasso problem. We can solve the lasso by:

$$\beta^{lasso} = \arg \min_{\beta} \|\mathbf{y}^* - \mathbf{B}\beta\|_2^2 + \lambda(1 - \alpha) \|\beta\|_1 \quad (33)$$

STATS790 A2

Hanwen Ju

2023-02-12

Question 1(b)

#Naive Linear Algebra

```
beta.la <- function(X,y){  
  beta <- solve(t(X) %*% X) %*% t(X) %*% y  
  
  return(beta)  
}
```

#QR decomposition

```
beta.qr <- function(X,y){  
  QR <- qr(X)  
  Q <- qr.Q(QR)  
  R <- qr.R(QR)  
  
  beta <- solve(R) %*% t(Q) %*% y  
  
  return(beta)  
}
```

#SVD decomposition

```
beta.svd <- function(X,y){  
  svd <- svd(X)  
  u <- svd$u  
  d <- diag(svd$d)  
  v <- svd$v  
  
  beta <- v %*% solve(d) %*% t(u) %*% y  
  
  return(beta)  
}
```

```
library(microbenchmark)  
#my_lm <- function(X,y) rnorm(ncol(X)) ## trivial: for testing  
simfun <- function(n, p) {  
  y <- rnorm(n)  
  X <- matrix(rnorm(p*n), ncol = p)
```

```

X <- as.matrix(cbind(rep(1,n),X))
list(X = X, y = y)
}
set.seed(101)
s <- simfun(100, 10)
## a test to make sure we get the same coefficients -- FAILS in this case
## (also see the 'check' argument of microbenchmark())

#stopifnot(all.equal(lm.fit(s$X, s$y)$coefficients,
#                      beta.la(s$X, s$y)))
## consider setting the `times` argument to a value less than the default of 100
## if the individual fits are too slow
m <- microbenchmark(
  beta.qr(s$X, s$y),
  beta.la(s$X, s$y),
  beta.svd(s$X, s$y))
results <- summary(m) ## store the components you need from this
results

##           expr      min       lq      mean   median      uq      max
## 1 beta.qr(s$X, s$y) 193.098 205.9535 256.17646 211.2170 219.6425 4003.978
## 2 beta.la(s$X, s$y)  57.853  64.5320  96.40849  67.2575  70.8060 2830.153
## 3 beta.svd(s$X, s$y) 168.795 197.5005 436.40762 205.2970 211.7730 18703.816
##   neval
## 1    100
## 2    100
## 3    100

nvec <- round(10^seq(2, 5, by = 0.25))

#store the result:

ren.la <- c()
ren.qr <- c()
ren.svd <- c()

#looping over values of n
for (i in 1:length(nvec)) {

  s <- simfun(nvec[i], p = 10)
  m <- microbenchmark(
    beta.la(s$X, s$y),
    beta.qr(s$X, s$y),
    beta.svd(s$X, s$y))
  re <- summary(m, unit = 'ms')
  ren.la <- append(ren.la, re$mean[1])
  ren.qr <- append(ren.qr, re$mean[2])
  ren.svd <- append(ren.svd, re$mean[3])

}

ren.df <- data.frame(Naive = ren.la, QR = ren.qr, SVD = ren.svd)

```



```

#pvec <- round(10^seq(1, 2, by = 0.25))

pvec <- c(5,25,50,100,200,400)
rep.la <- c()
rep.qr <- c()
rep.svd <- c()

#looping over values of p

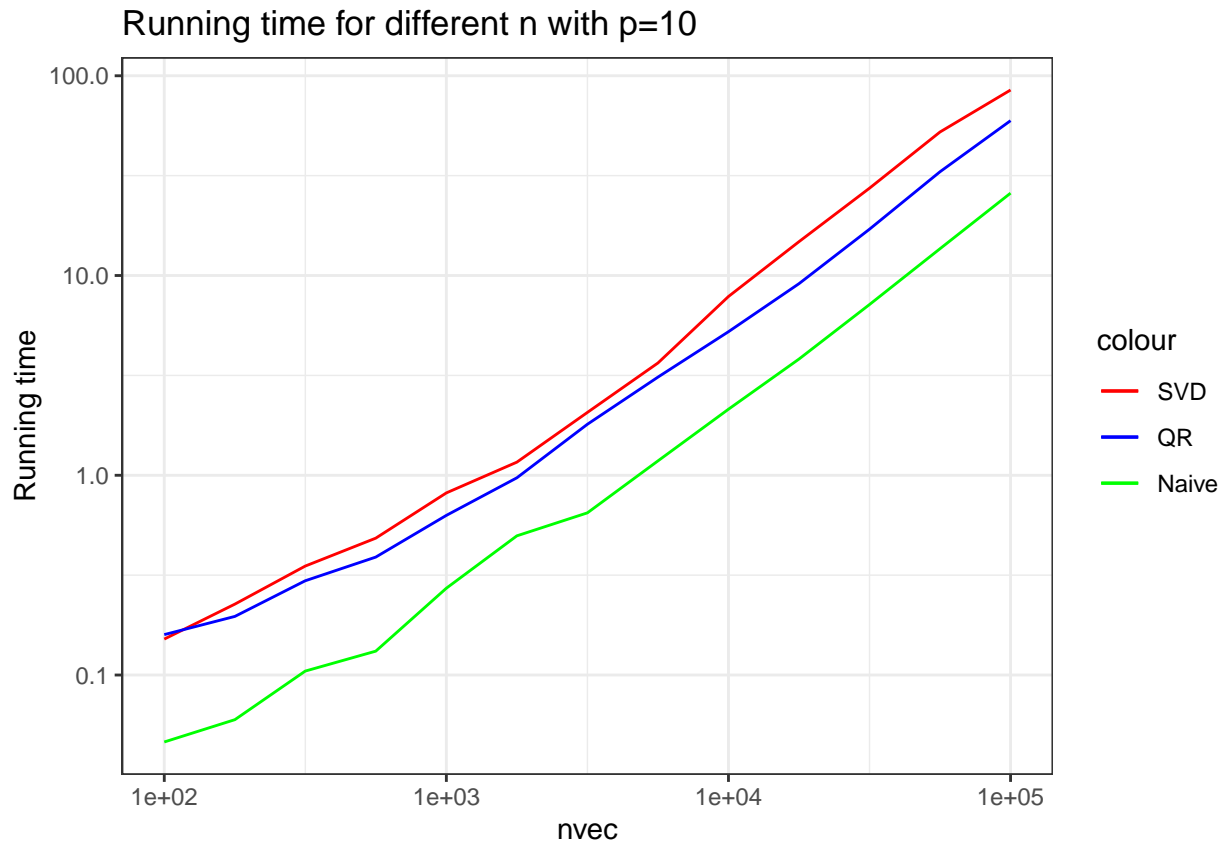
for (i in 1:length(pvec)) {
  s <- simfun(500, p = pvec[i])
  m <- microbenchmark(
    beta.la(s$X, s$y),
    beta.qr(s$X, s$y),
    beta.svd(s$X, s$y))
  re <- summary(m,unit = 'ms')
  rep.la <- c(rep.la, re$mean[1])
  rep.qr <- c(rep.qr, re$mean[2])
  rep.svd <- c(rep.svd, re$mean[3])
}

rep.df <- data.frame(Naive = rep.la, QR = rep.qr, SVD = rep.svd)

library(ggplot2)

#plot Running time for different n with p=10
(ggplot()
+ geom_line(aes(x= nvec,y = ren.la,colour = "red"))
+ geom_line(aes(x= nvec,y = ren.qr,colour = "blue"))
+ geom_line(aes(x= nvec,y = ren.svd,colour = "green"))
+ scale_color_manual(labels = c("SVD", "QR","Naive"), values = c("red", "blue","green"))
+ scale_x_continuous(trans='log10')
+ scale_y_continuous(trans='log10')
+ ggtitle("Running time for different n with p=10")
+ ylab("Running time")
+ theme_bw())

```

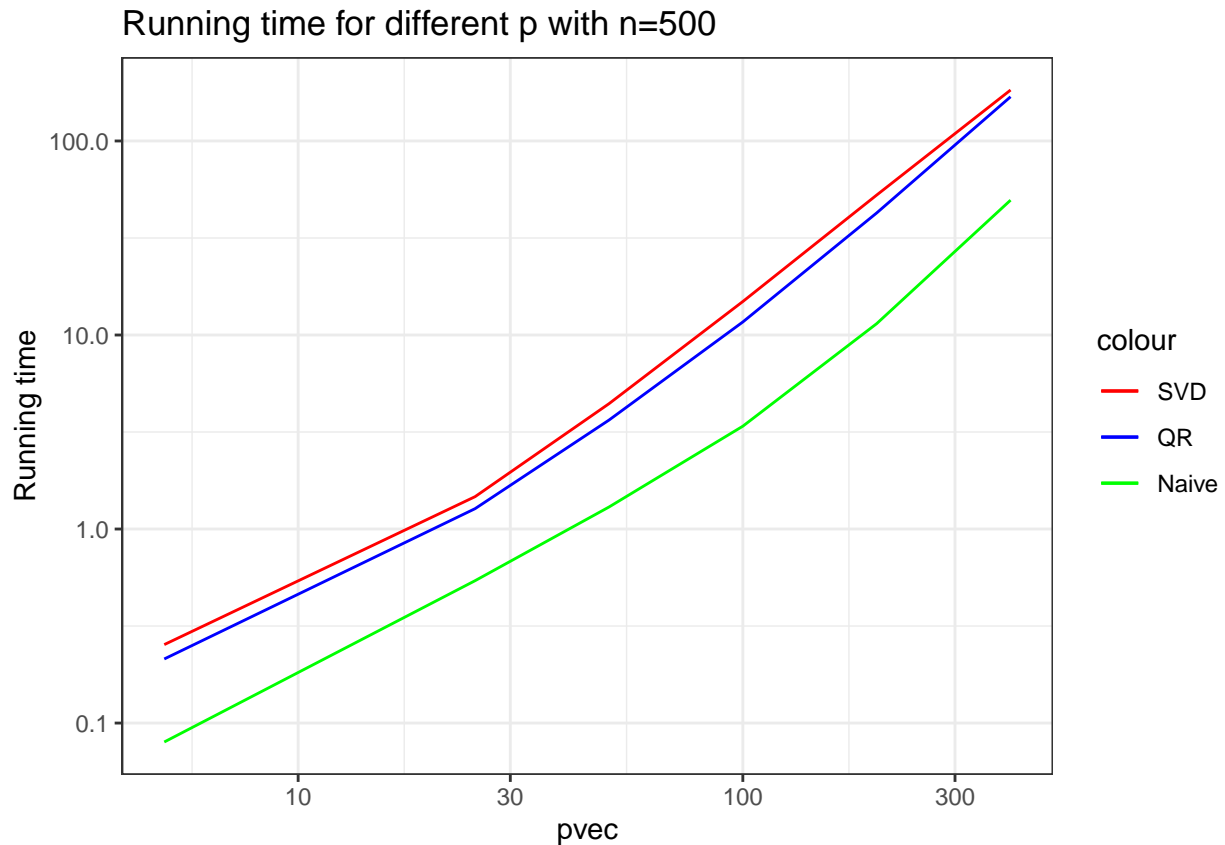


```
#plot Running time for different n with p=10
(ggplot()

+ geom_line(aes(x= pvec,y = rep.svd,colour = "green"))
+ geom_line(aes(x= pvec,y = rep.la,colour = "red"))

+ geom_line(aes(x= pvec,y = rep.qr,colour = "blue"))

+ scale_color_manual(labels = c("SVD", "QR","Naive"), values = c("red", "blue","green"))
+ scale_x_continuous(trans='log10')
+ scale_y_continuous(trans='log10')
+ ggtitle("Running time for different p with n=500")
+ ylab("Running time")
+ theme_bw())
```



Question 2

```
#Ridge regression by data augmentation(QR decomposition)

lmridge <- function(X,y,lambda){

  X <- unname(X)
  n <- dim(X)[1]
  p <- dim(X)[2]

  #augmenting X
  B <- as.matrix(rbind(as.matrix(X),sqrt(lambda)*diag(p)))

  B <- as.matrix(cbind(c(rep(1,n),rep(0,p)),B))

  #augmentin y
  y <- as.matrix(c(y,rep(0,p)))

  #QR decomposition
  QR <- qr(B)
  Q <- qr.Q(QR)
  R <- qr.R(QR)
```

```

beta <- solve(R) %*% t(Q) %*% as.matrix(y)

return(beta)
}

# read data
df <- read.table("https://hastie.su.domains/ElemStatLearn/datasets/prostate.data")
head(df)

##      lcavol  lweight age      lbph svi      lcp gleason pgg45      lpsa
## 1 -0.5798185 2.769459 50 -1.386294 0 -1.386294      6      0 -0.4307829
## 2 -0.9942523 3.319626 58 -1.386294 0 -1.386294      6      0 -0.1625189
## 3 -0.5108256 2.691243 74 -1.386294 0 -1.386294      7     20 -0.1625189
## 4 -1.2039728 3.282789 58 -1.386294 0 -1.386294      6      0 -0.1625189
## 5  0.7514161 3.432373 62 -1.386294 0 -1.386294      6      0  0.3715636
## 6 -1.0498221 3.228826 50 -1.386294 0 -1.386294      6      0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE

train <- df[df$train==TRUE,]
train <- scale(train,TRUE,TRUE)
train_x <- train[,1:8]
train_y <- train[,9]

test <- df[df$train==FALSE,]
test <- scale(test,TRUE,TRUE)
test_x <- test[,1:8]
test_y <- test[,9]

#ridge regression by data augmenting(QR)

beta.ridge <- lmridge(train_x,train_y,2)

pred.ridge <- as.matrix(cbind(rep(1,nrow(test_x)),test_x)) %*% beta.ridge

#RSS error
sum((test_y - pred.ridge)^2)

## [1] 13.9301

#ridge regression by glmnet
fit <- glmnet::glmnet(
  x = train_x,
  y = train_y,
  family = "gaussian",
  alpha = 0, ## ridge penalty
  lambda = 2,
  standardize = FALSE,

```

```
)

fit_pred <- predict(
  fit,
  s = 2,
  newx = test_x
)

#RSS Error
sum((test_y - fit_pred)^2)
```

```
## [1] 16.38894
```

```
m1 <- microbenchmark(
  lmridge(train_x,train_y,2),
  fit <- glmnet::glmnet(
    x = train_x,
    y = train_y,
    family = "gaussian",
    alpha = 0, ## ridge penalty
    lambda = 2,
    standardize = FALSE,
    intercept = FALSE,
  )
)
```

```
results <- summary(m1)
```

```
results
```

```
##
```

```
## 1
```

```
## 2 fit <- glmnet::glmnet(x = train_x, y = train_y, family = "gaussian", alpha = 0, lambda = 2, s
```

```
##      min      lq      mean    median      uq      max neval
```

```
## 1 142.978 157.1285 214.2857 170.664 209.2495 3173.200 100
```

```
## 2 993.041 1010.4560 1033.6195 1021.433 1042.4510 1477.928 100
```