hw4

shichenh

10/21/2017

Problem 1

let $\sigma = \sigma 1 = \sigma 2 = \sigma 3$.
Because $r_{12} = r_{13} = r_{23} = 0$, $\sigma_4^2 = \sigma_1^2 + \sigma_2^2 + \sigma_3^3 = 3\sigma^2$

$$\begin{aligned} r_{14} = cor(X_1, X_4) &= \frac{cov(X_1, X_4)}{\sigma_4 * \sigma_1} \\ &= \frac{cov(X_1, X_1 + X_2 + X_3)}{\sigma_4 * \sigma_1} \\ &= \frac{cov(X_1, X_1) + cov(X_1, X_2) + cov(X_1, X_3)}{\sigma_4 * \sigma_1} \\ &= \frac{\sigma_1^2 + 0 + 0}{\sigma_4 * \sigma_1} \\ &= \frac{\sigma^2 + 0 + 0}{\sqrt{3 * \sigma^2} * \sigma} \\ &= \frac{\sigma^2}{\sqrt{3} * \sigma^2} \\ &= \frac{1}{\sqrt{3}} = 0.577 \end{aligned}$$

Similary, $r_{14} = r_{24} = r_{34} = 0.577$

Problem 2

Core Idea

The central idea of the proof is to show for all $v \in X_1$ is orthogonal to z_1 and for all h, $X_h \subset X_{h-1}$. So bc $z_h \in X_{h-1} \subset X_1$, $z_h^T z_1 = 0$.

Notations

let h be the possible index of our algorithm.
let X_h be the model matrix after finding h components, $X_h[j]$ be the j^{th} column of matrix X_h.

Assumptions

In our proof, we assume w_h will never be 0. If it is zero, then z_h will be 0, the proof will be trivial.
Then bc $z_1 = X_0 w_1$, $z_1 \in span\{X_0\}$.
bc $X_1 = X_0 - z_1 * \frac{X_0^T z_1}{z_1^T z_1}$,

For $\forall j$,

$$X_1[j] = X_0[j] - z_1 * \frac{X_0[j]^T z_1}{z_1^T z_1}$$

We can see

1. $\forall j, X_1[j]$ is orthogonal to Z_1 because it is $X_0[j]$ substracts its projection onto Z_1, so therefore all vectors in X_1 are orthogonal Z_1.
2. $X_1 \subset X_0$ because $X_1[j]$ is a linear combination of $X_0[j]$ and Z_1.

With similar argument on

$$X_h[j] = X_{h-1}[j] - z_h * \frac{X_{h-1}[j]^T z_h}{z_h^T z_h}$$

we can see
$\forall h, X_h \subset X_{h-1}$ because $X_h[j]$ is a linear combination of $X_{h-1}[j]$ and Z_h. So $X_h \subset X_{h-1} ... \subset X_1$.
bc $z_h = X_{h-1} w_h$ and $w_h \neq 0$, $z_h \in X_{h-1} \subset X_1$, $z_h^T z_1 = 0$.

Problem 3

```
library(pls)
library(ISLR)
library(leaps)
library(glmnet)
library(dplyr)
library(tidyverse)
library(reshape)
```

```
prostate <- read.table("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data")
train <- prostate %>%
  filter(train) %>%
  select(-train)
test <- prostate %>%
  filter(!train) %>%
  select(-train)
```

```
cor.m <- cor(train %>% select(-lpsa))
cor.m
```

```
##             lcavol    lweight       age        lbph        svi          lcp
## lcavol  1.00000000 0.30023199 0.2863243  0.06316772  0.5929491  0.69204308
## lweight 0.30023199 1.00000000 0.3167235  0.43704154  0.1810545  0.15682859
## age     0.28632427 0.31672347 1.0000000  0.28734645  0.1289023  0.17295140
## lbph    0.06316772 0.43704154 0.2873464  1.00000000 -0.1391468 -0.08853456
## svi     0.59294913 0.18105448 0.1289023 -0.13914680  1.0000000  0.67124021
## lcp     0.69204308 0.15682859 0.1729514 -0.08853456  0.6712402  1.00000000
## gleason 0.42641407 0.02355821 0.3659151  0.03299215  0.3068754  0.47643684
## pgg45   0.48316136 0.07416632 0.2758057 -0.03040382  0.4813577  0.66253335
##            gleason       pgg45
## lcavol  0.42641407  0.48316136
## lweight 0.02355821  0.07416632
## age     0.36591512  0.27580573
## lbph    0.03299215 -0.03040382
```

```
## svi     0.30687537  0.48135774
## lcp     0.47643684  0.66253335
## gleason 1.00000000  0.75705650
## pgg45   0.75705650  1.00000000
```

```
train.x <- scale(train[,-c(9)])
test.x <- scale(test[,-c(9)])
seed <- 1
summary(train.x)
```

```
##      lcavol           lweight              age         
##  Min.   :-2.1411   Min.   :-2.62526   Min.   :-3.16524  
##  1st Qu.:-0.6641   1st Qu.:-0.62054   1st Qu.:-0.49935  
##  Median : 0.1242   Median :-0.05755   Median : 0.03382  
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000  
##  3rd Qu.: 0.8334   3rd Qu.: 0.54029   3rd Qu.: 0.56700  
##  Max.   : 2.0180   Max.   : 2.42189   Max.   : 1.89994  
##       lbph               svi               lcp             gleason       
##  Min.   :-0.99595   Min.   :-0.5331   Min.   :-0.8368   Min.   :-1.032  
##  1st Qu.:-0.99595   1st Qu.:-0.5331   1st Qu.:-0.8368   1st Qu.:-1.032  
##  Median :-0.08385   Median :-0.5331   Median :-0.4171   Median : 0.379  
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000  
##  3rd Qu.: 1.00848   3rd Qu.:-0.5331   3rd Qu.: 0.8631   3rd Qu.: 0.379  
##  Max.   : 1.54057   Max.   : 1.8480   Max.   : 2.0496   Max.   : 3.200  
##      pgg45        
##  Min.   :-0.8965  
##  1st Qu.:-0.8965  
##  Median :-0.3846  
##  Mean   : 0.0000  
##  3rd Qu.: 0.8099  
##  Max.   : 2.5163
```

Least Square

```
ols <- lm(train$lpsa~train.x)
ols.pred <- model.matrix(test$lpsa~test.x) %*% ols$coefficients
ols.mse <- mean((ols.pred - test$lpsa)^2)
ols.coef <- ols$coefficients
summary(ols)
```

```
##
## Call:
## lm(formula = train$lpsa ~ train.x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.64870 -0.34147 -0.05424  0.44941  1.48675 
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     2.45235    0.08702  28.182  < 2e-16 ***
## train.xlcavol   0.71641    0.13350   5.366 1.47e-06 ***
## train.xlweight  0.29264    0.10638   2.751  0.00792 ** 
## train.xage     -0.14255    0.10212  -1.396  0.16806
```

```
## train.xlbph     0.21201    0.10312   2.056  0.04431 *
## train.xsvi      0.30962    0.12539   2.469  0.01651 *
## train.xlcp     -0.28901    0.15480  -1.867  0.06697 .
## train.xgleason -0.02091    0.14258  -0.147  0.88389
## train.xpgg45    0.27735    0.15959   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

Best subset

```
set.seed(seed)
best.sub <- regsubsets(x=train.x, y=train$lpsa, method="exhaustive")

val.error <- rep(NA, 8)
for (i in 1:8) {
  coefi <- coef(best.sub, id=i)
  test.m <- model.matrix(test$lpsa~test.x)
  pred <- test.m[,1:length(coefi)] %*% coefi
  val.error[i] = mean((test$lpsa - pred)^2)
}

npred <- which.min(val.error)
best.sub.mse <- val.error[npred]
best.sub.mse
```

```
## [1] 0.4992376
```

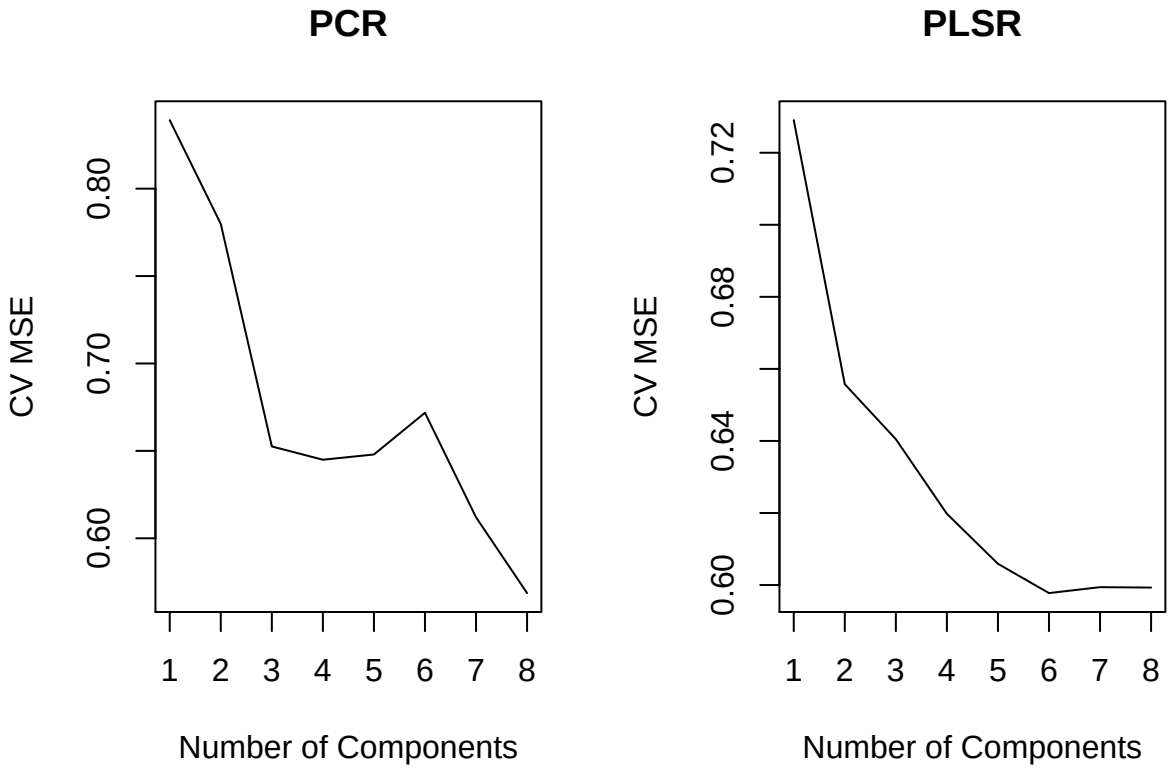
PCR and PLSR

```
set.seed(seed)
n = nrow(train)
# pcr
pcr.fit <- pcr(train$lpsa~train.x, validation = "CV", segments=10)
pcr.ncomp <- which.min(pcr.fit$validation$PRESS[1, ] / nrow(train))
pcr.pred <- predict(pcr.fit, ncomp = pcr.ncomp ,newdata = test.x)
pcr.mse <- mean((pcr.pred - test$lpsa)^2)
pcr.coef <- coef(pcr.fit, intercept = T)
# plsr
plsr.fit <- plsr(train$lpsa~train.x, validation = "CV", segments=10)
plsr.ncomp <- which.min(plsr.fit$validation$PRESS[1, ] / n)
plsr.pred <- predict(plsr.fit, ncomp = plsr.ncomp ,newdata = test.x)
plsr.mse <- mean((plsr.pred - test$lpsa)^2)
plsr.coef <- coef(plsr.fit, intercept = T)
```

```
par(mfrow=c(1,2))

plot(pcr.fit$validation$PRESS[1, ] / n, type="l", main="PCR",
xlab="Number of Components", ylab="CV MSE")
```

```
plot(plsr.fit$validation$PRESS[1, ] / n , type="l", main="PLSR",
xlab="Number of Components", ylab="CV MSE")
```



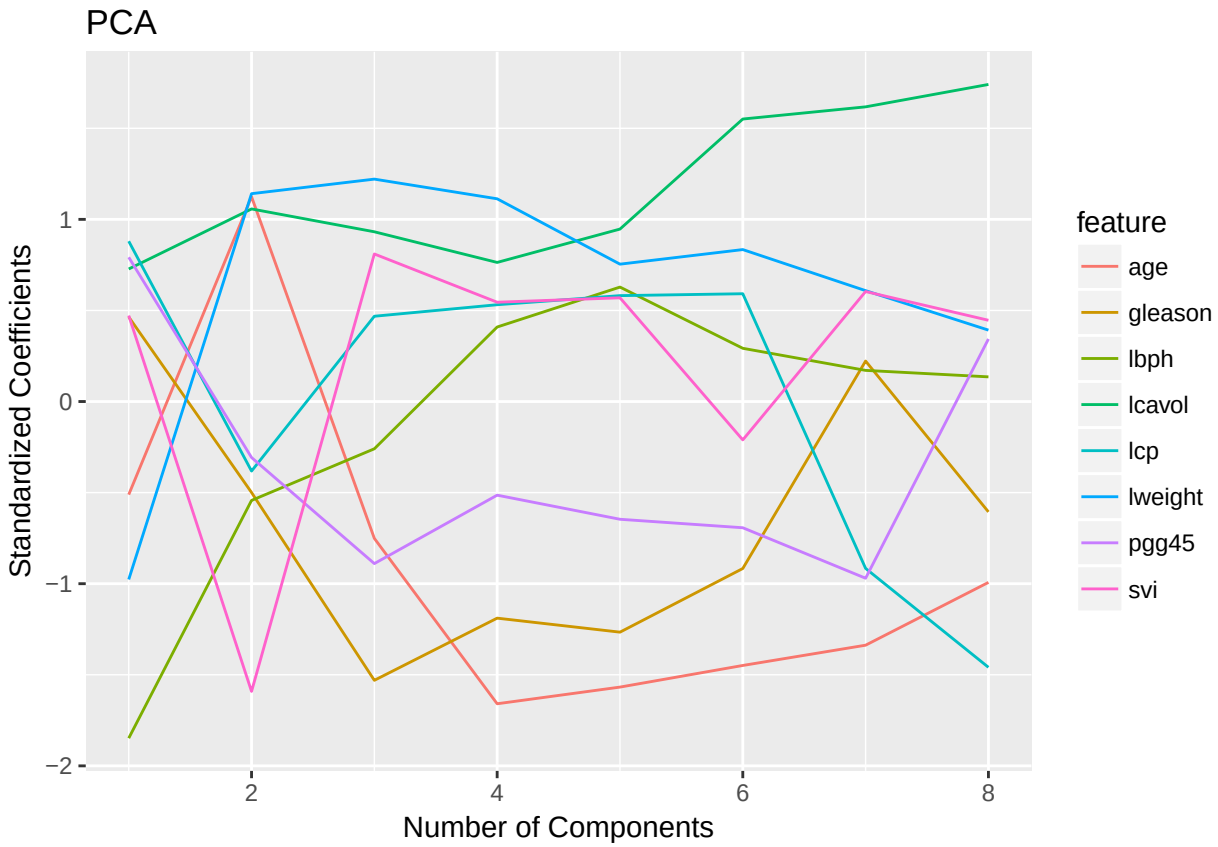

```
pcr.plot.df <- data.frame(scale(data.frame(pcr.fit$coefficients))) %>% rownames_to_column()
colnames(pcr.plot.df) <- c("feature", 1:8)
pcr.plot.df
```

```
##   feature          1          2          3          4          5
## 1  lcavol  0.7276599  1.0573491  0.9318079  0.7636897  0.9467490
## 2 lweight -0.9767553  1.1409964  1.2209161  1.1129636  0.7541277
## 3     age -0.5110459  1.1241328 -0.7513460 -1.6591716 -1.5676306
## 4    lbph -1.8480872 -0.5428600 -0.2594491  0.4090561  0.6282825
## 5     svi  0.4719900 -1.5913228  0.8100956  0.5451908  0.5694384
## 6     lcp  0.8799386 -0.3816960  0.4681217  0.5313884  0.5813855
## 7 gleason  0.4643593 -0.4993910 -1.5304767 -1.1891994 -1.2659649
## 8   pgg45  0.7919405 -0.3072085 -0.8896694 -0.5139176 -0.6463875
##            6          7          8
## 1  1.5507381  1.6177266  1.7406212
## 2  0.8341463  0.6086890  0.3920584
## 3 -1.4489466 -1.3375802 -0.9928703
## 4  0.2922690  0.1709565  0.1354511
## 5 -0.2105936  0.6039511  0.4460854
## 6  0.5917453 -0.9160205 -1.4589430
## 7 -0.9165147  0.2222579 -0.6057829
## 8 -0.6928439 -0.9699805  0.3433799
```

```
df2 <- melt(pcr.plot.df, id.vars = "feature")

ggplot(df2) +
```

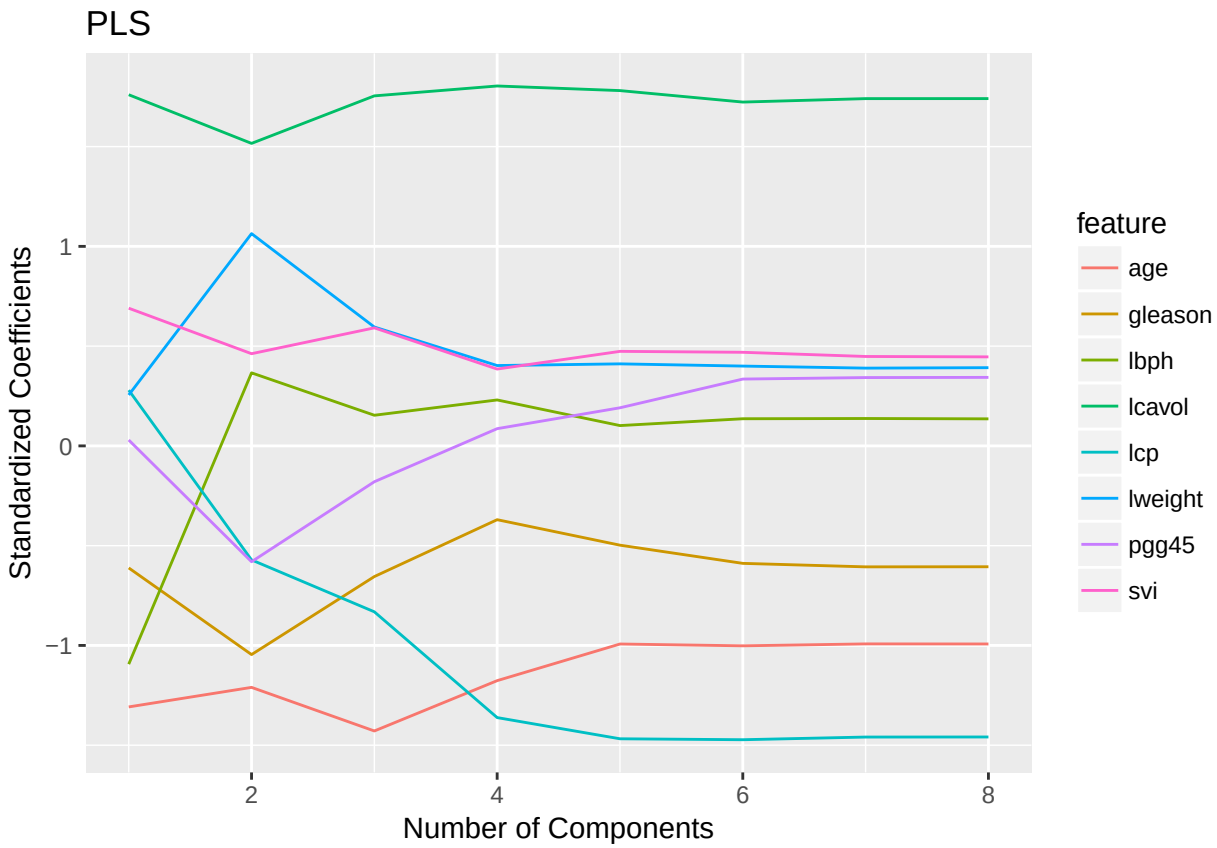
```
  geom_line(aes(x=as.numeric(variable), y=value, color=feature)) +
  xlab("Number of Components") +
  ylab("Standardized Coefficients") +
  labs(title="PCA")
```




```
plsr.plot.df <- data.frame(scale(data.frame(plsr.fit$coefficients))) %>% rownames_to_column()
colnames(plsr.plot.df) <- c("feature", 1:8)

plsr2 <- melt(plsr.plot.df, id.vars = "feature")

ggplot(plsr2) +
  geom_line(aes(x=as.numeric(variable), y=value, color=feature)) +
  xlab("Number of Components") +
  ylab("Standardized Coefficients") +
  labs(title="PLS")
```

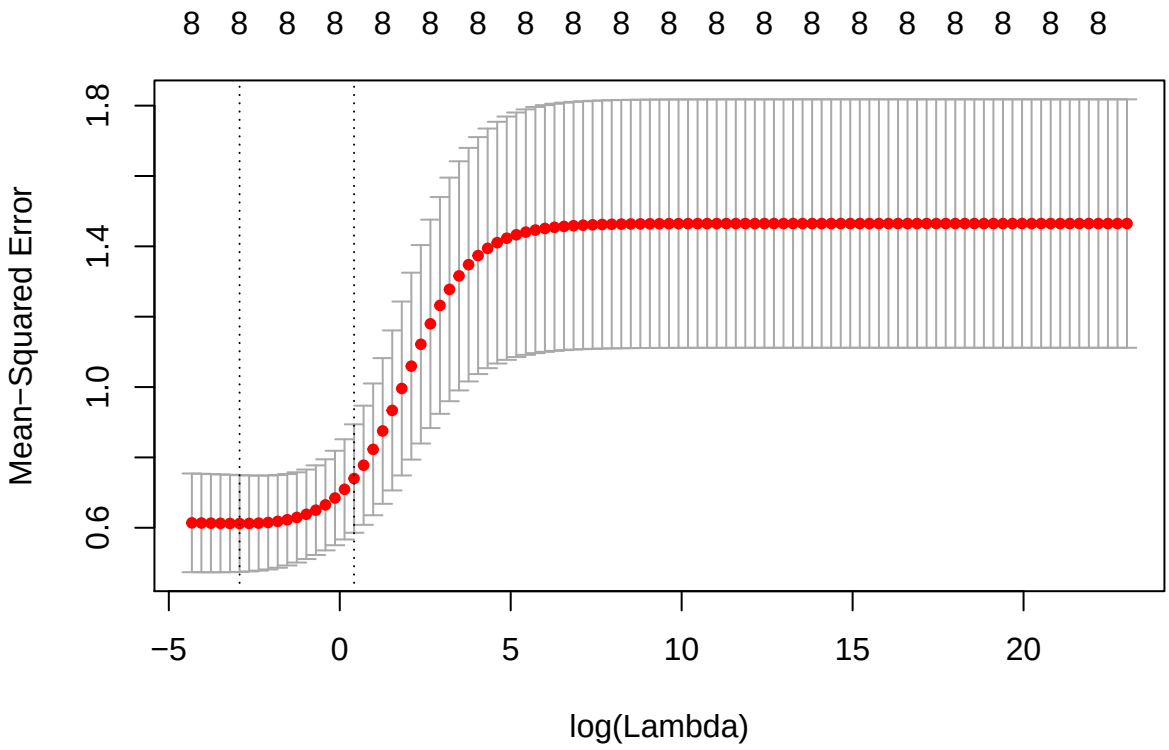


RR and Lasso

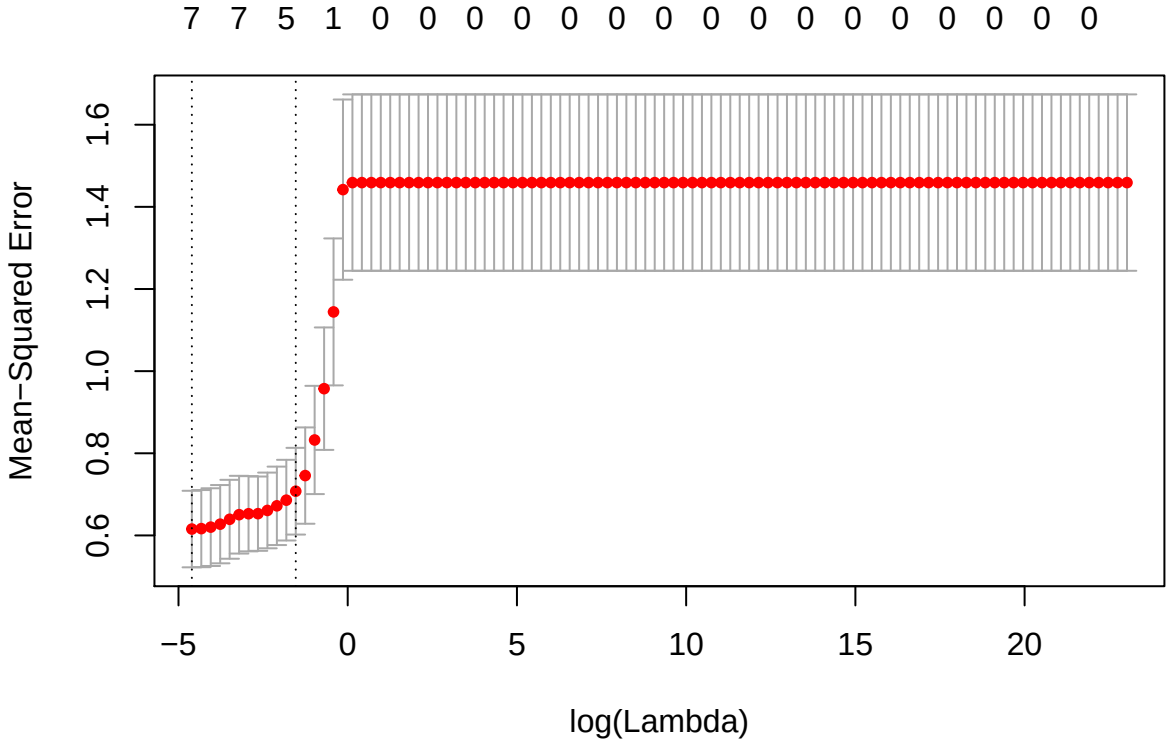
```
set.seed(seed)
grid <- 10^seq(10, -2, length=100)
## ridge
ridge.mod <- cv.glmnet(train.x, train$lpsa, lambda = grid, alpha = 0, nfolds = 10)
# using lambda.min before but I read that lambda.1se may be better because it overfits less
ridge.pred <- predict(ridge.mod, s = "lambda.1se", newx = test.x)
ridge.mse <- mean((ridge.pred - test$lpsa)^2)
ridge.coef <- coef.cv.glmnet(ridge.mod)

## lasso
lasso.mod <- cv.glmnet(train.x, train$lpsa, lambda = grid, alpha = 1, nfolds = 10)
lasso.pred <- predict(lasso.mod, s = "lambda.1se", newx = test.x)
lasso.mse <- mean((lasso.pred - test$lpsa)^2)
lasso.coef <- coef.cv.glmnet(lasso.mod)
```

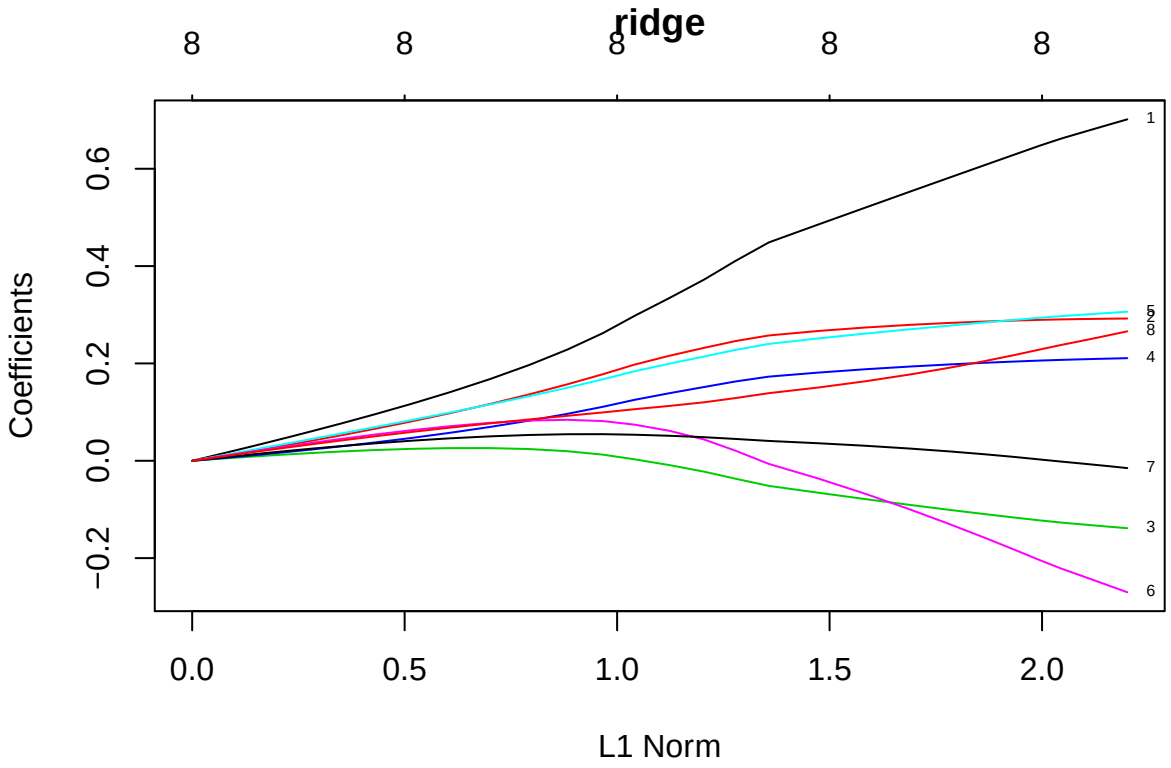
```
#par(mfrow=c(1,2))
plot.cv.glmnet(ridge.mod)
```



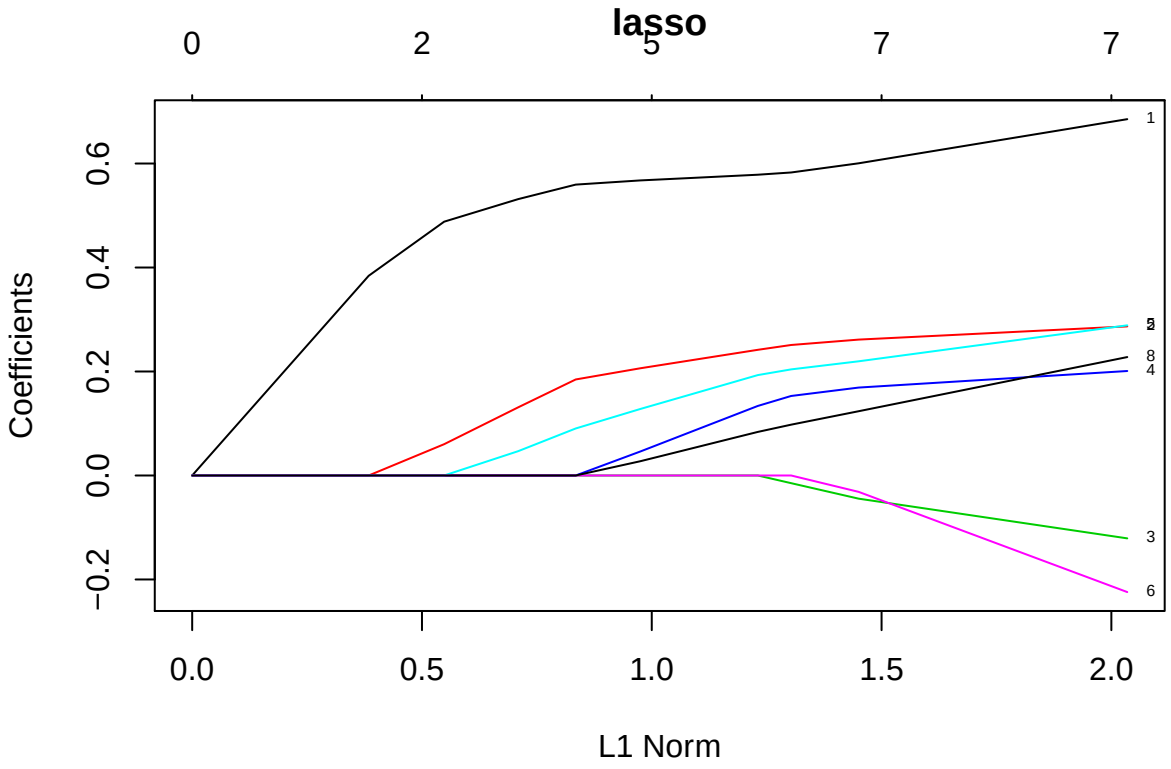

```
plot.cv.glmnet(lasso.mod)
```




```
plot(ridge.mod$glmnet.fit, "norm", label=TRUE, main = "ridge")
```




```
plot(lasso.mod$glmnet.fit, "norm", label=TRUE, main = "lasso")
```



Performance Table

```
coefs <- cbind(ols.coef, c(coef(best.sub, id=npred), rep(0, 7)), ridge.coef, lasso.coef, pcr.coef, plsr
mses <- c(ols.mse, best.sub.mse, ridge.mse, lasso.mse, pcr.mse, plsr.mse)
performance.table <- rbind(coefs, mses)
colnames(performance.table) <- c("LS", "Best Subset", "Ridge", "Lasso", "PCR", "PLS")
performance.table
```

```
## 10 x 6 sparse Matrix of class "dgCMatrix"
##                       LS Best Subset      Ridge      Lasso         PCR
## (Intercept)  2.45234509   2.4523451 2.45234509 2.45234509  2.45234509
## lcavol       0.71640701   0.8855136 0.26216841 0.55941589  0.71640701
## lweight      0.29264240   .         0.17776357 0.18466227  0.29264240
## age         -0.14254963   .         0.01263447 .          -0.14254963
## lbph         0.21200760   .         0.11085843 .           0.21200760
## svi          0.30961953   .         0.16729986 0.09040553  0.30961953
## lcp         -0.28900562   .         0.08159054 .          -0.28900562
## gleason     -0.02091352   .         0.05464270 .          -0.02091352
## pgg45        0.27734595   .         0.09931430 .           0.27734595
## mses         0.54919414   0.4992376 0.53016873 0.45500812  0.54919414
##                      PLS
## (Intercept)  2.45234509
## lcavol       0.71640701
## lweight      0.29264240
## age         -0.14254963
## lbph         0.21200760
## svi          0.30961953
## lcp         -0.28900562
## gleason     -0.02091352
## pgg45        0.27734595
## mses         0.54931525
```

My results in the table is different from the table in ESL. I am not sure what seed they use so I am unable to reproduce their results. However, my results are decently similar.

From the modeling perspective, my results changes when I use a different seed. But in general lasso and best subset seems to have better performances than others. This may suggest some features are either correlated or they have little predictive power in the models.