Jin Kweon_3032235207_HW4

Jin Kweon

10/9/2017

Problem 1 (10 points)

It is given in the problem that $r_{12} = r_{13} = r_{23} = 0$. So, for one example, $r_{12} = cor(X_1, X_2) = \frac{cov(X_1,X_2)}{sd(X_1)sd(X_2)}$ $= 0$. Since denominator cannot be zero (also, the problem never says standard deviation or variance of four predictors are zero), it implies that $cov(X_1, X_2) = 0$.

So, r_{13} and r_{23} also imply $cov(X_1, X_3) = 0$ and $cov(X_2, X_3) = 0$.

So, now, I need to prove $r_{14} = r_{24} = r_{34} = 0.577$. (or, closed to 0.577) I will start with proving $r_{14} = 0.577$.

$r_{14} = \frac{cov(X_1,X_4)}{sd(X_1)sd(X_4)} = \frac{cov(X_1,X_1+X_2+X_3)}{sd(X_1)\sqrt{var(X_4)}} = \frac{cov(X_1,X_1)\ +\ cov(X_1,X_2)\ +\ cov(X_1,X_3)}{sd(X_1)\sqrt{var(X_1\ +\ X_2\ +\ X_3)}} = \frac{cov(X_1,X_1)}{sd(X_1)\sqrt{var(X_1\ +\ X_2\ +\ X_3)}} =$ $\frac{var(X_1)}{sd(X_1)\sqrt{var(X_1)\ +\ var(X_2)\ +\ var(X_3)\ +\ 2cov(X_1,X_2)\ +\ 2cov(X_1,X_3)\ +\ 2cov(X_2,X_3)}} = \frac{var(X_1)}{sd(X_1)\sqrt{var(X_1)\ +\ var(X_2)\ +\ var(X_3)}}$ $= \frac{sd(X_1)}{\sqrt{var(X_1)\ +\ var(X_2)\ +\ var(X_3)}} = \frac{\sigma_1}{\sqrt{\sigma_1^2\ +\ \sigma_2^2\ +\ \sigma_3^2}} = \frac{\sigma_1}{\sqrt{3\sigma_1^2}} = \frac{1}{\sqrt{3}} \approx 0.577.$

Also, r_{24} and r_{34} can be proved in a similar way.

$r_{24} = \frac{cov(X_2,X_4)}{sd(X_2)sd(X_4)} = \frac{cov(X_2,X_1+X_2+X_3)}{sd(X_2)\sqrt{var(X_4)}} = \frac{var(X_2)}{sd(X_2)\sqrt{var(X_1)\ +\ var(X_2)\ +\ var(X_3)}} = \frac{sd(X_2)}{\sqrt{var(X_1)\ +\ var(X_2)\ +\ var(X_3)}}$ $= \frac{\sigma_2}{\sqrt{3\sigma_2^2}} = \frac{1}{\sqrt{3}} \approx 0.577.$

And, r_{34} will be eventually $\frac{\sigma_3}{\sqrt{3\sigma_3^2}} = \frac{1}{\sqrt{3}} \approx 0.577.$

The key point of this problem is that variance of X_1, X_2, and X_3 are the same.

Problem 2 (10 points)

As it says on the hint of the problem, it can definitely be proved by recursivity of PLS algorithm we learned in the class.

Here is the proof below:

I am going to pick i where 1 ≤ i ≤ n. And, what I need to do is prove i is orthogonal to any other PLS component.

1) $z_i^T z_{i+1} = z_i^T(\frac{X_i w_{i+1}}{w_{i+1}^T w_{i+1}}) = \frac{1}{w_{i+1}^T w_{i+1}} z_i^T(X_i w_{i+1})$. I only need to prove $z_i^T(X_i w_{i+1}) = 0$.

And, $z_i^T(X_i w_{i+1}) = z_i^T([x_{i-1} \ - \ z_i p_i^T]w_{i+1}) = z_i^T([x_{i-1} \ - \ z_i[\frac{x_{i-1}^T z_i}{z_i^T z_i}]^T]w_{i+1}) = (z_i^T x_{i-1} \ - \ z_i^T x_{i-1})w_{i+1} = 0$.

2) After, I will prove it recursively.

$z_i^T z_{i+2} = z_i^T(X_{i+1}w_{i+2})\frac{1}{w_{i+2}^T w_{i+2}} = z_i^T(X_i \ - \ z_{i+1}p_{i+1}^T)\frac{w_{i+2}}{w_{i+2}^T w_{i+2}} = (z_i^T X_i \ - \ z_i^T z_{i+1}p_{i+1}^T)\frac{w_{i+2}}{w_{i+2}^T w_{i+2}}$.

And, since $z_i^T z_{i+1} = 0$ as we proved in the last recursion proof, $(z_i^T X_i - z_i^T z_{i+1}p_{i+1}^T)\frac{w_{i+2}}{w_{i+2}^T w_{i+2}} = z_i^T X_i \frac{w_{i+2}}{w_{i+2}^T w_{i+2}}$.

So, I only need to prove $z_i^T X_i = 0$.

$z_i^T X_i = z_i^T(X_{i-1} \ - \ z_i p_i^T) = z_i^T(X_{i-1} \ - \ z_i[\frac{x_{i-1}^T z_i}{z_i^T z_i}]^T) = z_i^T X_{i-1} \ - \ z_i^T X_{i-1} = 0$.

So, $z_i^T z_{i+2} = 0$ is proved.

3) I will prove one more recursion.

$z_i^T z_{i+3} = z_i^T(X_{i+2}w_{i+3})\frac{1}{w_{i+3}^T w_{i+3}} = z_i^T(X_{i+1} \ - \ z_{i+2}p_{i+2}^T)w_{i+3}\frac{1}{w_{i+3}^T w_{i+3}}$.

I need to prove $z_i^T(X_{i+1} \ - \ z_{i+2}p_{i+2}^T) = z_i^T(X_{i+1} \ - \ z_{i+2}[\frac{x_{i+1}^T z_{i+2}}{z_{i+2}^T z_{i+2}}]^T) = z_i^T X_{i+1} \ - \ z_i^T z_{i+2}[\frac{x_{i+1}^T z_{i+2}}{z_{i+2}^T z_{i+2}}]^T$.

And, since we proved $z_i^T z_{i+2} = 0$, $z_i^T X_{i+1} \ - \ z_i^T z_{i+2}[\frac{x_{i+1}^T z_{i+2}}{z_{i+2}^T z_{i+2}}]^T = z_i^T X_{i+1}$.

So, I need to prove $z_i^T X_{i+1} = 0$.

$z_i^T X_{i+1} = (z_i^T X_i \ - \ z_i^T z_{i+1}p_{i+1}^T) = z_i^T X_i$, as $z_i^T z_{i+1} = 0$.

So, I need to prove $z_i^T X_i = 0$.

$z_i^T X_i = z_i^T(X_{i-1} \ - \ z_i p_i^T) = z_i^T(X_{i-1} \ - \ z_i[\frac{x_{i-1}^T z_i}{z_i^T z_i}]^T) = z_i^T X_{i-1} \ - \ z_i^T X_{i-1} = 0$.

I can keep proving this recursion.

Thus, $z_h^T z_l = 0$, for h ≠ l where 1 ≤ h ≤ n and 1 ≤ l ≤ n.

Problem 3 (100 points)

- lcavol: log cancer volume
- lweight: log prostate weight
- age: age of patient
- lbph: log of the amount of benign prostatic hyperplasia
- svi: seminal vesicle invasion
- lcp: log of capsular penetration
- gleason: Gleason score
- pgg45: percent of Gleason scores 4 or 5
- lpsa: log of prostate-specific antigen (response variable)

```
prostate <- prostate
```

```
training <- prostate %>% filter(train == "TRUE")
testing <- prostate %>% filter(train == "FALSE")
training <- training[,-10]
testing <- testing[,-10]

dim(training)
```

```
## [1] 67  9
```

```
dim(testing)
```

```
## [1] 30  9
```

```
sum(is.na(prostate)) #check NA
```

```
## [1] 0
```

lpsa is the response variable. The rest are the predictors. I will select training set and standardize training set only! After, I will get correlation matrix.

Correlations of predictors, and some preprocessing (10 pts)

```
trainingscale <- scale(training, T, T)

summary(trainingscale[,1:3]) #summary for lcavol, lweight, and age
```

```
##      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
```

```
summary(trainingscale[,4:6]) #summary for lbph, svi, lcp
```

```
##       lbph                svi                lcp
##  Min.   :-0.99595   Min.   :-0.5331   Min.   :-0.8368
##  1st Qu.:-0.99595   1st Qu.:-0.5331   1st Qu.:-0.8368
##  Median :-0.08385   Median :-0.5331   Median :-0.4171
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 1.00848   3rd Qu.:-0.5331   3rd Qu.: 0.8631
##  Max.   : 1.54057   Max.   : 1.8480   Max.   : 2.0496
```

```
summary(trainingscale[,7:8]) #summary for gleason and pgg45
```

```
##     gleason           pgg45
##  Min.   :-1.032   Min.   :-0.8965
##  1st Qu.:-1.032   1st Qu.:-0.8965
##  Median : 0.379   Median :-0.3846
##  Mean   : 0.000   Mean   : 0.0000
##  3rd Qu.: 0.379   3rd Qu.: 0.8099
##  Max.   : 3.200   Max.   : 2.5163
```

```
trainingscale_x <- trainingscale[,-9]

correlation <- cor(trainingscale_x)
correlation <- correlation[-1,-8]
round(correlation, 3)
```

```
##         lcavol lweight   age   lbph    svi    lcp gleason
## lweight  0.300   1.000 0.317  0.437  0.181  0.157   0.024
## age      0.286   0.317 1.000  0.287  0.129  0.173   0.366
## lbph     0.063   0.437 0.287  1.000 -0.139 -0.089   0.033
## svi      0.593   0.181 0.129 -0.139  1.000  0.671   0.307
## lcp      0.692   0.157 0.173 -0.089  0.671  1.000   0.476
## gleason  0.426   0.024 0.366  0.033  0.307  0.476   1.000
## pgg45    0.483   0.074 0.276 -0.030  0.481  0.663   0.757
```

Least Squares Model (10 pts)

```r
#response is not scaled, but predictors are.
trainxscale_only <- cbind(trainingscale_x, lpsa = training$lpsa)

ols <- lm(lpsa ~., data = as.data.frame(trainxscale_only))

table3.2 <- summary(ols)$coefficients[,-4]
colnames(table3.2) <- c("Coefficient", "Std.Error", "T value")

round(table3.2, 2)
```

```
##             Coefficient Std.Error T value
## (Intercept)        2.45      0.09   28.18
## lcavol             0.72      0.13    5.37
## lweight            0.29      0.11    2.75
## age               -0.14      0.10   -1.40
## lbph               0.21      0.10    2.06
## svi                0.31      0.13    2.47
## lcp               -0.29      0.15   -1.87
## gleason           -0.02      0.14   -0.15
## pgg45              0.28      0.16    1.74
```

Comment:

I agree with the points professor Sanchez made on the instruction. The first three coefficients (also, maybe the last one: pgg45) are slightly off.

And, actually, it should be t-test, not z-score, since we do not know actual standard deviation.

We can actually scale response variable as well.

1. When we did not scale response variable: When x variable goes up 1 unit, reponse variable changes coefficient of x (in y).

2. when we scale reponse variable: When x variable goes up 1 unit, response variable changes coefficient of x unit/quantile (in y).

Best Subset Regression (10 pts)

Good reference: http://rstudio-pubs-static.s3.amazonaws.com/2897_9220b21cfc0c43a396ff9abf122bb351.html

```r
subset <- regsubsets(lpsa ~., data = as.data.frame(trainxscale_only), nvmax = 8)
summary(subset)
```

```
## Subset selection object
## Call: regsubsets.formula(lpsa ~ ., data = as.data.frame(trainxscale_only),
##     nvmax = 8)
## 8 Variables  (and intercept)
##         Forced in Forced out
## lcavol      FALSE      FALSE
## lweight     FALSE      FALSE
## age         FALSE      FALSE
## lbph        FALSE      FALSE
## svi         FALSE      FALSE
## lcp         FALSE      FALSE
## gleason     FALSE      FALSE
## pgg45       FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          lcavol lweight age lbph svi lcp gleason pgg45
## 1  ( 1 ) "*"    " "     " " " "  " " " " " "     " "  
## 2  ( 1 ) "*"    "*"     " " " "  " " " " " "     " "  
## 3  ( 1 ) "*"    "*"     " " " "  "*" " " " "     " "  
## 4  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     " "  
## 5  ( 1 ) "*"    "*"     " " "*"  "*" " " " "     "*"  
## 6  ( 1 ) "*"    "*"     " " "*"  "*" "*" " "     "*"  
## 7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " "     "*"  
## 8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"  
```

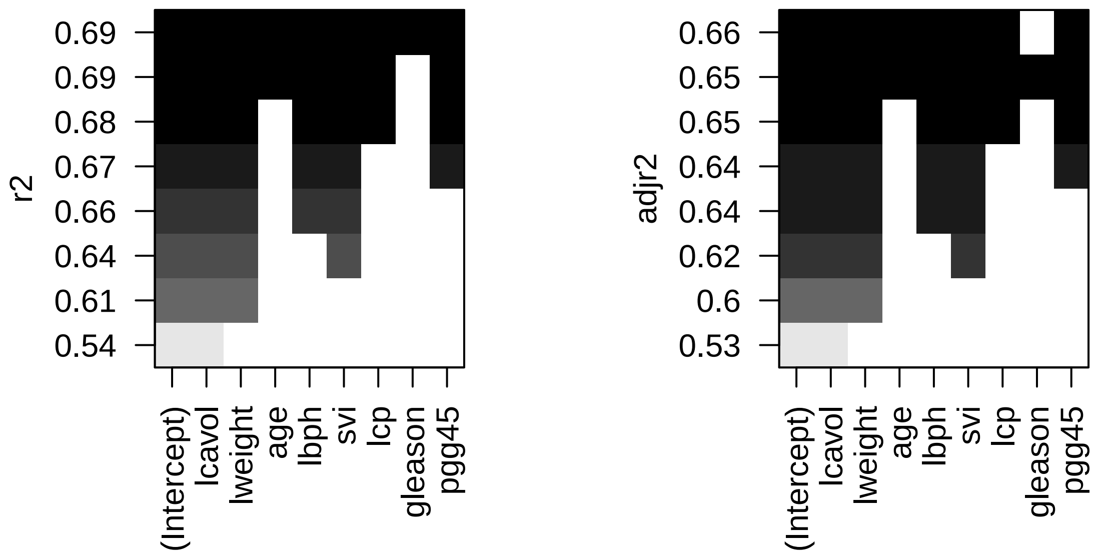
```r
summary(subset)$bic
```

```
## [1] -43.25728 -51.29578 -51.15720 -51.09467 -48.42976 -47.49961 -45.75833
## [8] -41.57849
```

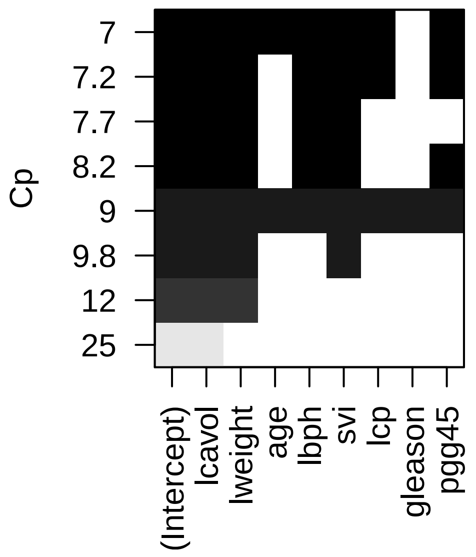
```r
paste("So, I keep the", which.min(summary(subset)$bic), "variables.")
```

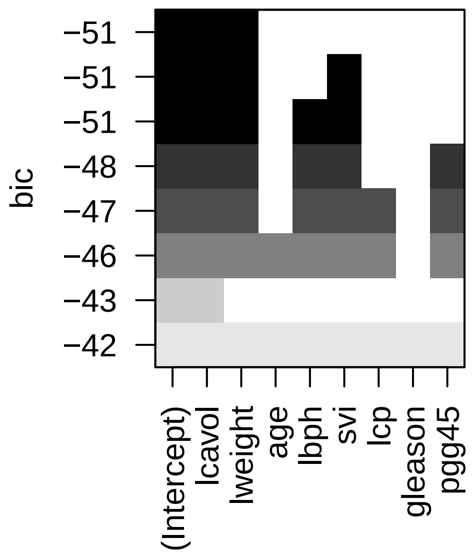
```
## [1] "So, I keep the 2 variables."
```

```r
par(mfrow = c(1,2))
plot(subset, scale = "r2")
plot(subset, scale = "adjr2")
```



```r
plot(subset, scale = "Cp")
plot(subset, scale = "bic")
```





```r
subsetcoef <- lm(lpsa ~ lcavol + lweight, data = as.data.frame(trainxscale_only))$coefficients

coef(subset, 2)
```

```
## (Intercept)      lcavol     lweight
##   2.4523451   0.7798589   0.3519101
```

Comment:

Using BIC, they tell me I should keep the best two variables. So, I output the minimum BIC for when each number of variables are kept. Actually, there are two steps.

First, since we have 8 variables, we need to find the minimum BIC when 1, 2, . . . , 8 variables are kept. So, I got -43.26 (miminimum BIC when 1 variable is kept), -51.30 (minimum BIC when 2 variables are kept), . . . , -41.58 (minimum BIC when 8 variables are kept). After that, I need to find how many variables to keep, by finding the mimimum from there. And, it is the second one.

Thus, the *BEST* two three variable model contains lcavol and lweight.

PCR and PLSR (40 pts)

Q. What is X in summary(plsrfunc)? Is it kind of each cumulative of eigenvalue / 8? ===> this is the cumulative variance of components!!! not the e-value!!!

Q. Why do I have 8 variables for coef() function for plsr even though my tuning parameter is 6? ===> So, y = zd = xbeta, and although you have 6 compoents beta still have 8 coefficients. d (coefficient of components) will have 6 coefficients, but beta (coefficient of design matrix - what we want!! and what we usually say coefficient!!) still have 8 coefficients.

So, lasso is the only one that sometimes has zero coefficient for design matrix!!

Q. What does coefplot() do compared to matplot? ==> coefplot shows the coefficient (d) plot for components, and matplot draws the coefficient (beta) plot for design matrix!!!

Use 10 fold cross validation.

```
set.seed(10)

#PLSR
plsrfunc <- plsr(formula = lpsa ~., data =as.data.frame(trainxscale_only), validation = "CV") #validati
summary(plsrfunc)
```

```
## Data:    X dimension: 67 8
##  Y dimension: 67 1
## Fit method: kernelpls
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.217   0.8546   0.8128   0.7945   0.7928   0.7853   0.7824
## adjCV        1.217   0.8518   0.8073   0.7891   0.7855   0.7787   0.7760
##        7 comps  8 comps
## CV      0.7833   0.7833
## adjCV   0.7768   0.7767
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       41.64    58.29    71.13    79.75    86.08    90.21    94.70
## lpsa    55.79    64.60    67.51    69.12    69.37    69.43    69.44
##       8 comps
## X      100.00
## lpsa    69.44
```

```
paste("Tuning parameter is", which.min(plsrfunc$validation$PRESS[1,]))
```

```
## [1] "Tuning parameter is 6"
```

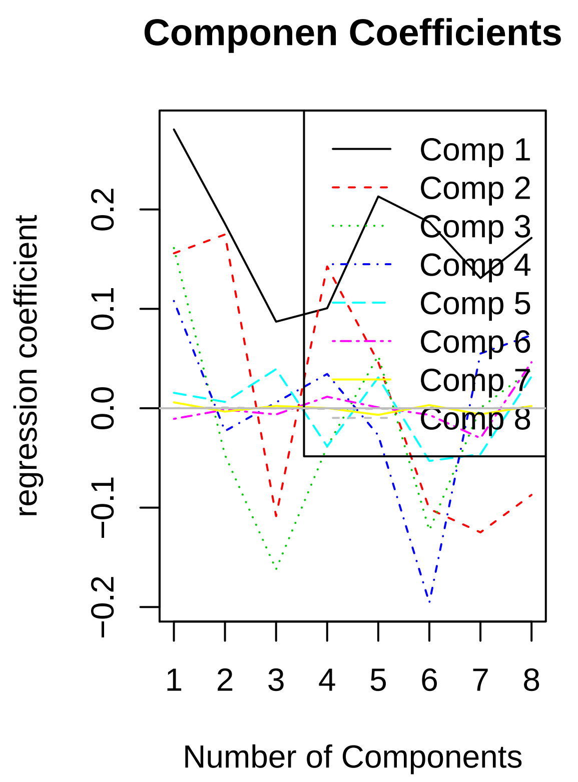
```
print("Associated coefficients of PLSR:")
```

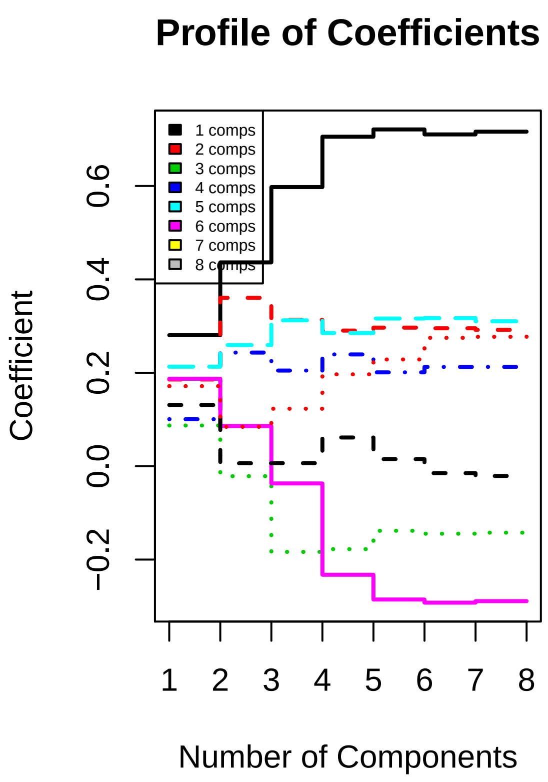
```
## [1] "Associated coefficients of PLSR:"
```

```
plsrfunc$coefficients[,,which.min(plsrfunc$validation$PRESS)]
```

```
##     lcavol    lweight        age       lbph        svi        lcp
##  0.7104094  0.2952801 -0.1446106  0.2124677  0.3169434 -0.2922292
##    gleason      pgg45
## -0.0149234  0.2748280
```

```
plscoef <- apply(plsrfunc$coefficients, 3, function(x) x)
par(mfrow= c(1,2))
coefplot(plsrfunc, comps = 1:8, separate = F, intercept = T, xlab = "Number of Components",
         main = "Componen Coefficients", legendpos = "topright")

matplot(t(plscoef), type= 's', lwd = 2, xlab = "Number of Components", main = "Profile of Coefficients"
legend("topleft", colnames(plscoef), col = seq_len(ncol(plscoef)), cex = 0.5, fill = seq_len(ncol(plsco
```

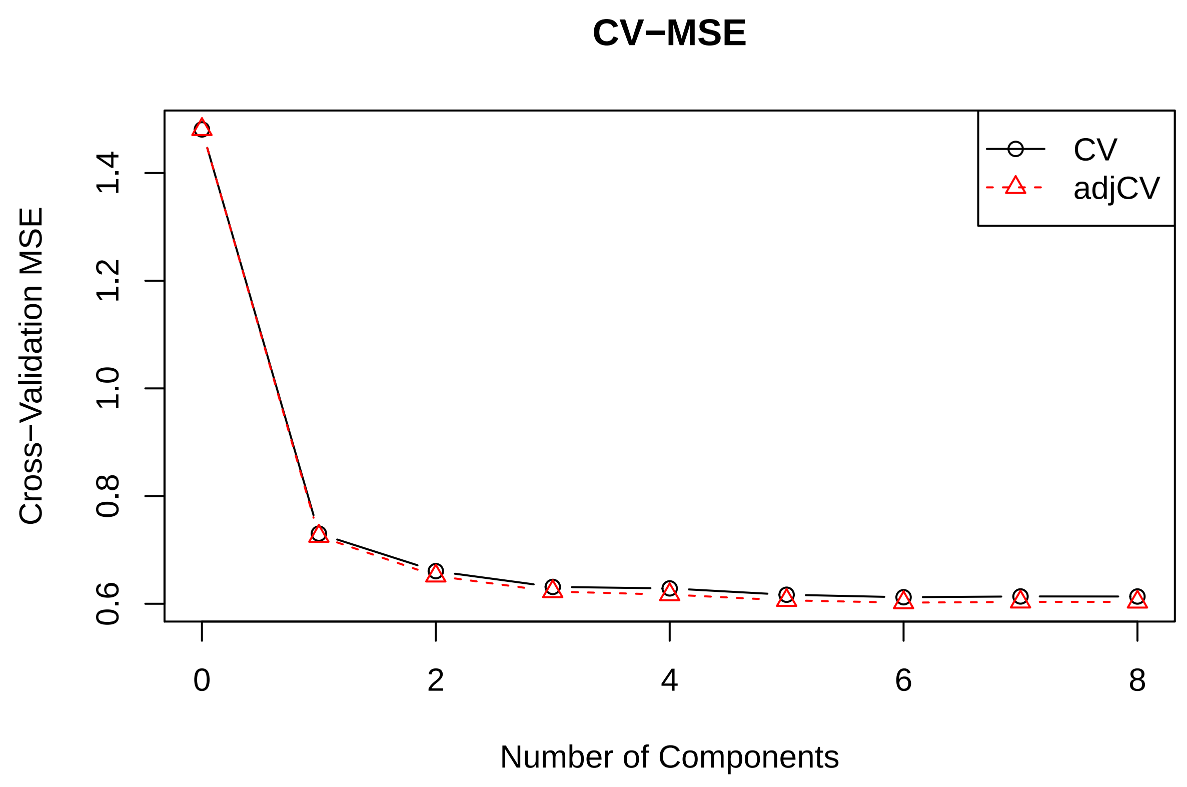




```
#matplot(plscoef, type= 'l', lwd = 2)
#matplot(t(plscoef), type= 'l', lwd = 2)


#RMSEP(plsrfunc) #This is what we have from summary
MSEP(plsrfunc) #Output MSE
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.481   0.7303   0.6606   0.6313   0.6285   0.6168   0.6121
## adjCV        1.481   0.7255   0.6517   0.6227   0.6170   0.6064   0.6021
##        7 comps  8 comps
## CV      0.6136   0.6135
## adjCV   0.6034   0.6033
```

```
par(mfrow = c(1,1))
validationplot(plsrfunc, val.type = "MSEP", ncomp = 1:8, type = "b",
               legendpos = "topright", xlab = "Number of Components",
               ylab = "Cross-Validation MSE", main = "CV-MSE")
```



```
# plot(plsrfunc$validation$PRESS[1, ] / nrow(trainxscale_only), type="l", main="PLSR",
#      xlab="Number of Components", ylab="CV MSE")


coef(plsrfunc, intercept = T)
```

```
## , , 8 comps
##
##                    lpsa
## (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
```

```
#PCR
pcrfunc <- pcr(formula = lpsa ~., data = as.data.frame(trainxscale_only), validation = "CV") #validatio
summary(pcrfunc)
```

```
## Data:    X dimension: 67 8
##  Y dimension: 67 1
## Fit method: svdpc
## Number of components considered: 8
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.217   0.9217   0.8875   0.8158   0.8109   0.8166   0.8362
## adjCV        1.217   0.9197   0.8863   0.8125   0.8074   0.8135   0.8324
##        7 comps  8 comps
## CV      0.7967   0.7521
## adjCV   0.7915   0.7474
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       42.83    63.24    76.20    83.92    89.61    94.32    97.82
## lpsa    45.18    50.84    59.58    61.00    61.17    62.08    66.36
##       8 comps
## X      100.00
## lpsa    69.44
```

```
paste("Tuning parameter is", which.min(pcrfunc$validation$PRESS[1,]))
```

```
## [1] "Tuning parameter is 8"
```

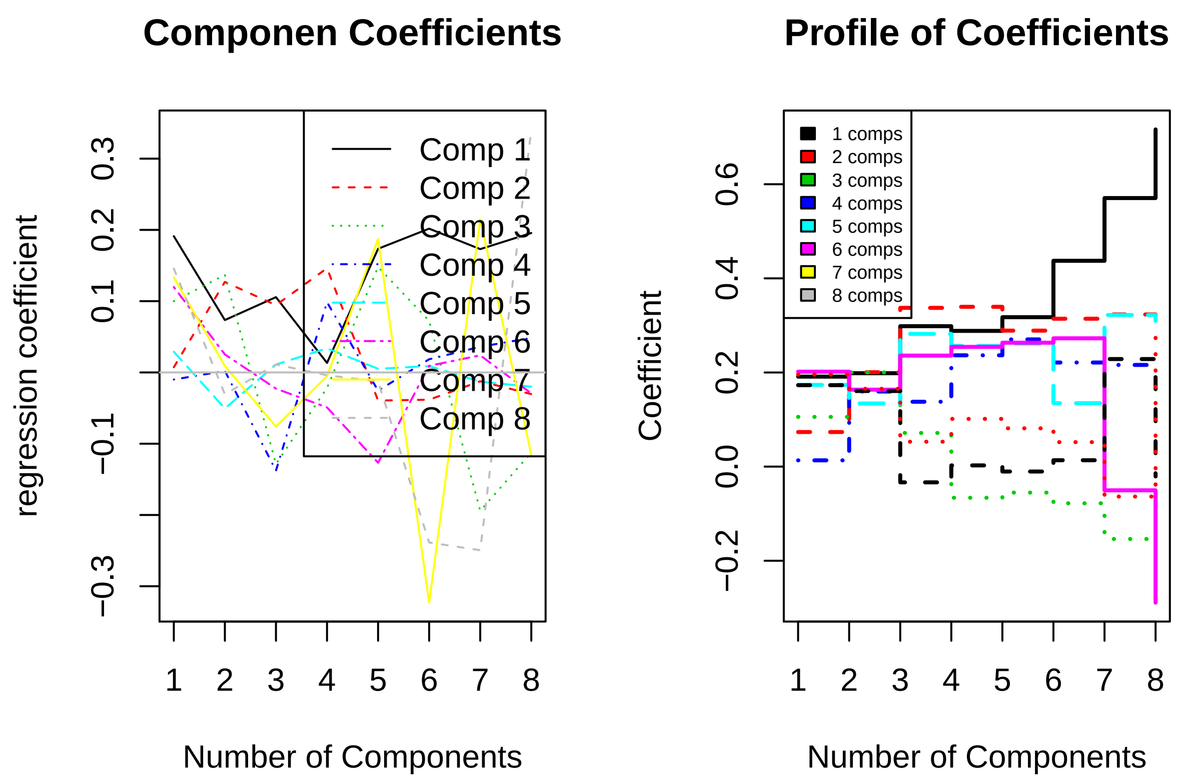
```
print("Associated coefficients of PCR:")
```

```
## [1] "Associated coefficients of PCR:"
```

```
pcrfunc$coefficients[,,which.min(pcrfunc$validation$PRESS)]
```

```
##      lcavol     lweight         age        lbph         svi         lcp
##  0.71640701  0.29264240 -0.14254963  0.21200760  0.30961953 -0.28900562
##     gleason       pgg45
## -0.02091352  0.27734595
```

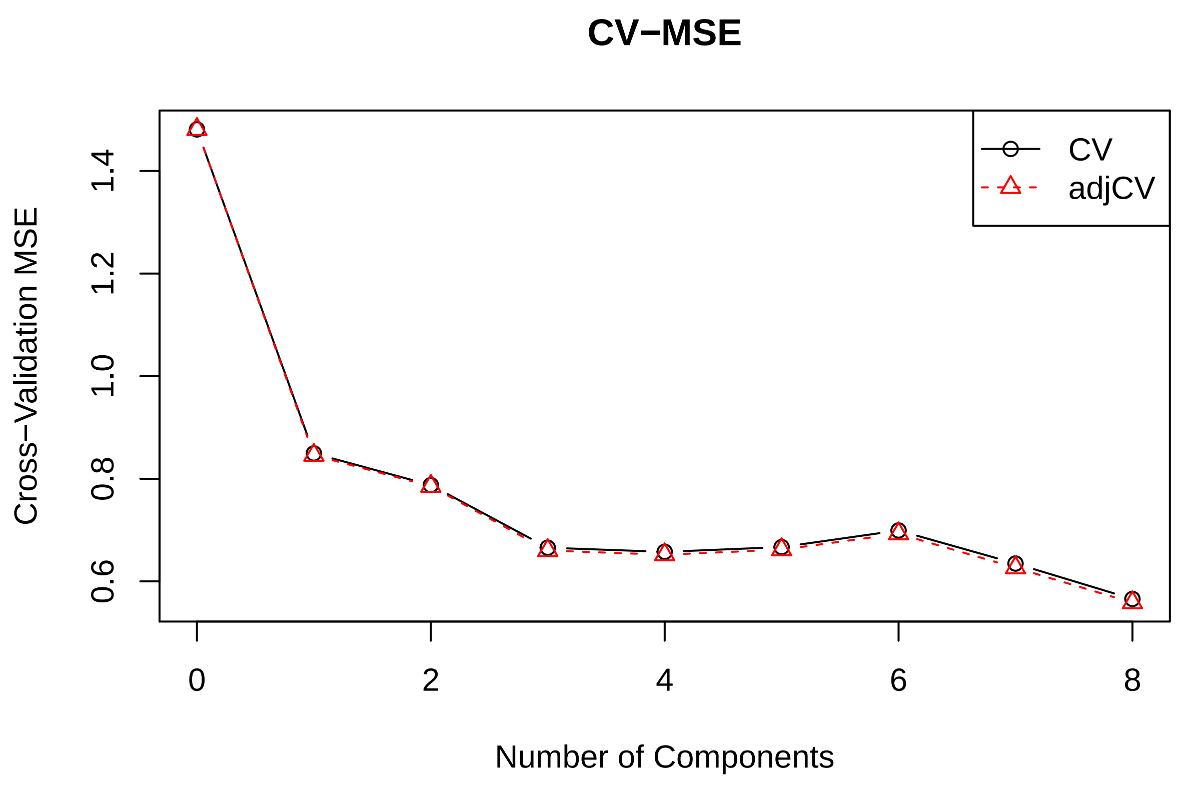
```
par(mfrow= c(1,2))
pcrcoef <- apply(pcrfunc$coefficients, 3, function(x) x)
coefplot(pcrfunc, comps = 1:8, separate = F, xlab = "Number of Components",
         main = "Componen Coefficients", legendpos = "topright")

matplot(t(pcrcoef), type= 's', lwd = 2, xlab = "Number of Components",
         main = "Profile of Coefficients", ylab = "Coefficient")
legend("topleft", colnames(pcrcoef), col = seq_len(ncol(pcrcoef)), cex = 0.6, fill = seq_len(ncol(pcrco
```



```r
#matplot(pcrcoef, type= 'l', lwd = 2)
#matplot(t(pcrcoef), type= 'l', lwd = 2)

#RMSEP(pcrfunc) #This is what we have from summary
MSEP(pcrfunc) #Output MSE
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.481   0.8495   0.7876   0.6656   0.6575   0.6668   0.6992
## adjCV        1.481   0.8459   0.7855   0.6601   0.6520   0.6618   0.6928
##        7 comps  8 comps
## CV      0.6347   0.5657
## adjCV   0.6265   0.5586
```

```r
par(mfrow = c(1,1))
validationplot(pcrfunc, val.type = "MSEP", ncomp = 1:8, type = "b",
               legendpos = "topright", xlab = "Number of Components",
               ylab = "Cross-Validation MSE", main = "CV-MSE")
```



```r
# plot(pcrfunc$validation$PRESS[1, ] / nrow(trainxscale_only), type="l", main="PCR",
#      xlab="Number of Components", ylab="CV MSE")

coef(pcrfunc, intercept = T)
```

```
## , , 8 comps
##
##                    lpsa
## (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
```

Comment:

Tuning parameter/Number of components are *6* and *8 (using all variables)* for PLSR and PCR respectively, since this has the smallest CV-RMSE (root square of MSE of prediction, which corresponds to the smallest MSE as well). I got these results based on 10-fold CV. They do 10-fold CV for each number of components, and we compare MSE of each number of component's.

Just for knowledge, plsr and pcr will have the same coefficients if we are using full coefficients (same with the OLS as well -> thus same MSE for all these three).

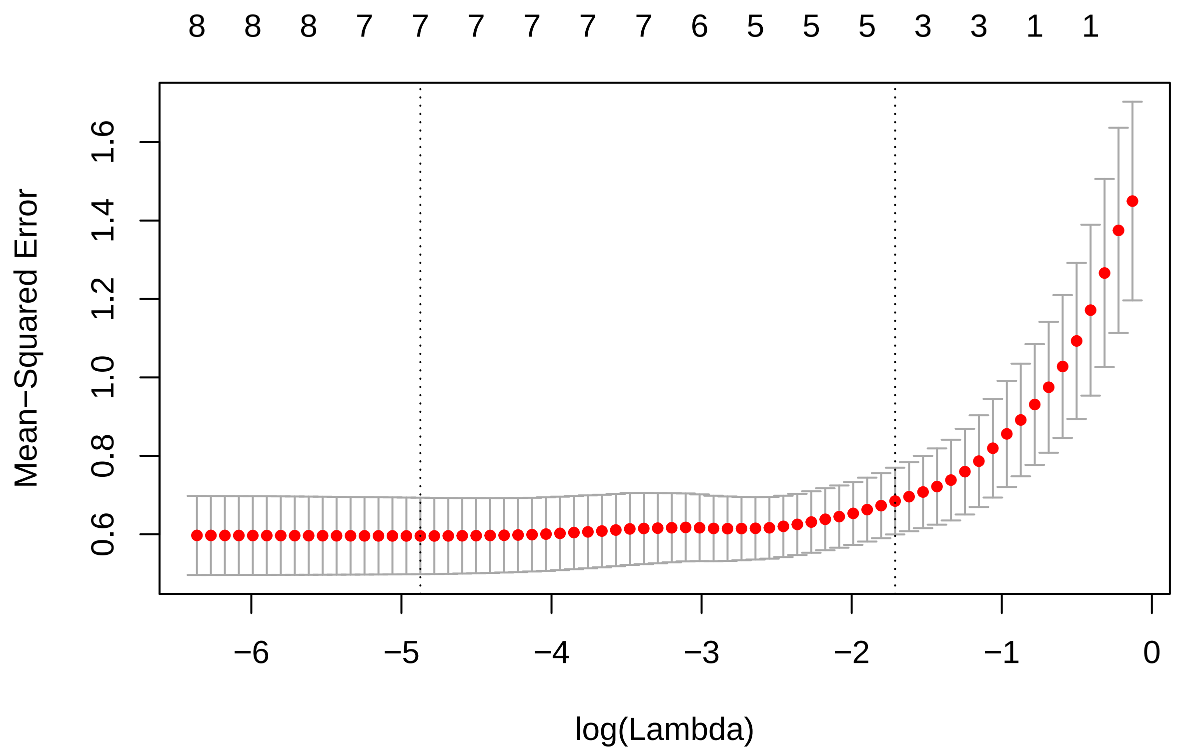
Q. Am I supposed to include intercept for my design matrix for ridge and lasso before using glmnet function? So, plsr and pcr are the only functions that do not include intercepts?

RR and Lasso (40 pts)

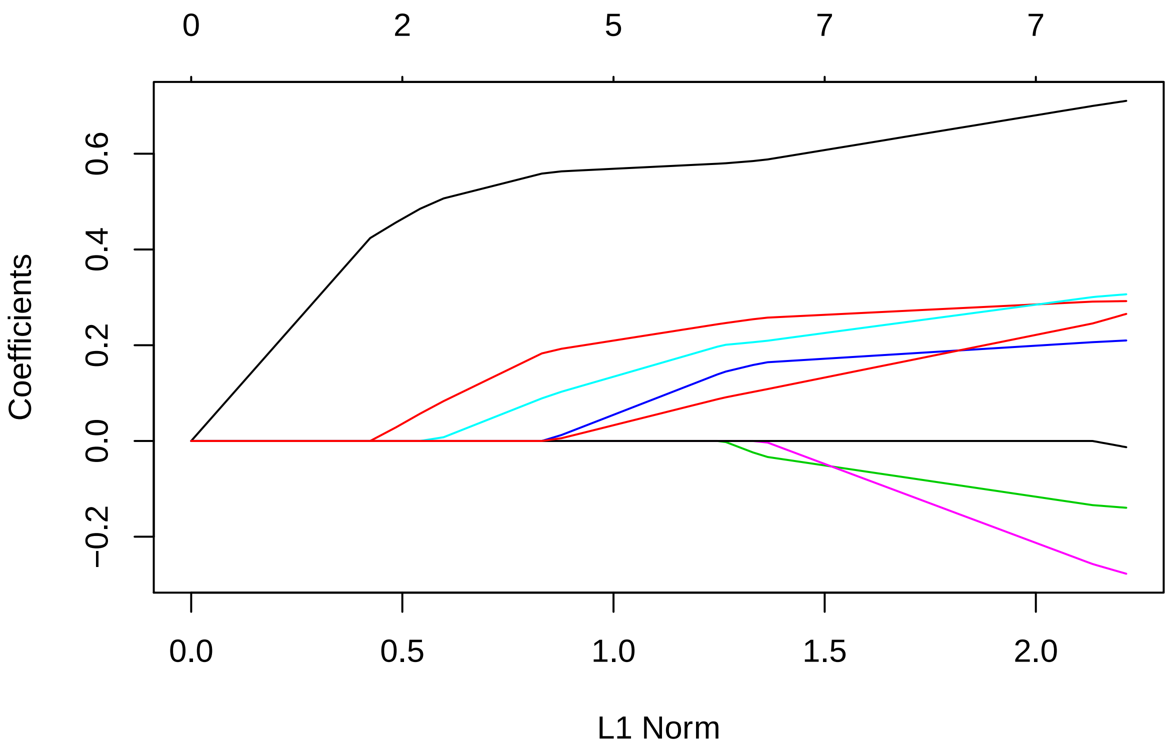
```
set.seed(10)
#Lasso
lasso <- cv.glmnet(trainxscale_only[,1:8], trainxscale_only[,9], nfolds = 10, alpha = 1) #validation

paste("Tuning parameter is", round(lasso$lambda.min, 4))
```

```
## [1] "Tuning parameter is 0.0076"
```

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



```
#Refit a model
lasso2 <- glmnet(trainxscale_only[,1:8], trainxscale_only[,9], alpha = 1,
                 lambda = lasso$lambda)
plot.glmnet(lasso2)
```



```
refit1 <- glmnet(trainxscale_only[,1:8], trainxscale_only[,9], alpha = 1, lambda = lasso$lambda.min)
refit1
```

```
##
## Call:  glmnet(x = trainxscale_only[, 1:8], y = trainxscale_only[, 9],      alpha = 1, lambda = lasso$
##
##      Df   %Dev   Lambda
## [1,]  7 0.6935 0.007644
```

```
coef(lasso, s = "lambda.min") #If i did not specify s, they give one standard away.
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  2.4523451
## lcavol       0.6918697
## lweight      0.2887031
## age         -0.1268621
## lbph         0.2033674
## svi          0.2940763
## lcp         -0.2389979
## gleason      .
## pgg45        0.2357199
```

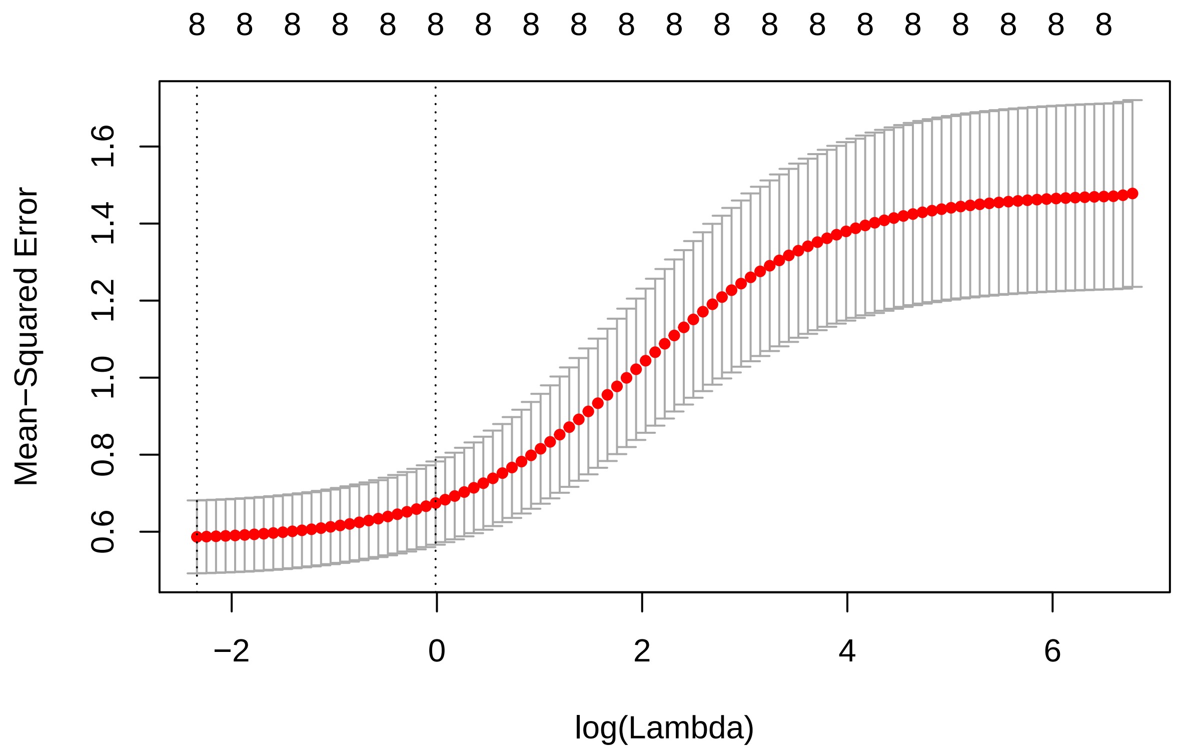
```
coef(refit1, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  2.4523451
## lcavol       0.6919179
## lweight      0.2887836
## age         -0.1269113
## lbph         0.2033150
## svi          0.2940639
## lcp         -0.2393184
## gleason      .
## pgg45        0.2359208
```

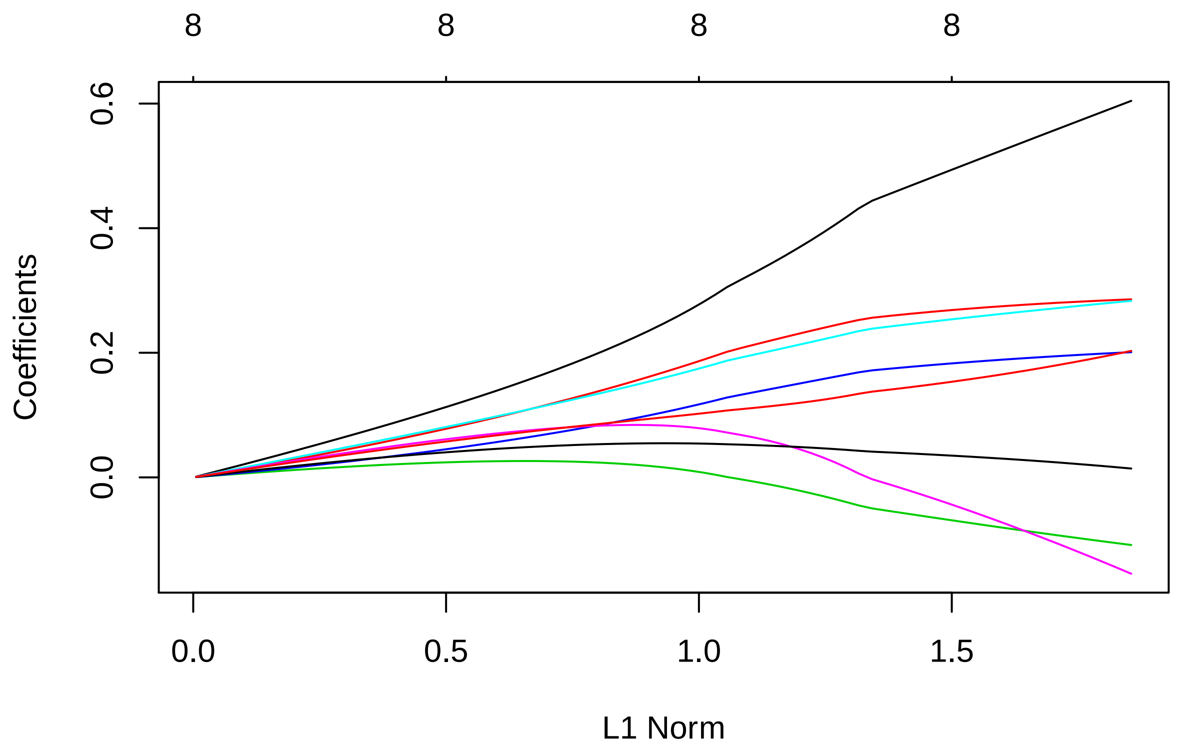
```
#Ridge
ridge <- cv.glmnet(trainxscale_only[,1:8], trainxscale_only[,9], nfolds = 10, alpha = 0) #validation

paste("Tuning paremeter is", round(ridge$lambda.min, 4))
```

```
## [1] "Tuning paremeter is 0.0965"
```

```
plot.cv.glmnet(ridge)
```



```
#Refit a model
ridge2 <- glmnet(trainxscale_only[,1:8], trainxscale_only[,9], alpha = 0,
                 lambda = ridge$lambda)
plot.glmnet(ridge2)
```



```r
refit2 <- glmnet(trainxscale_only[,1:8], trainxscale_only[,9], alpha = 0, lambda = ridge$lambda.min)
refit2
```

```
##
## Call:  glmnet(x = trainxscale_only[, 1:8], y = trainxscale_only[, 9],      alpha = 0, lambda = ridge
##
##      Df   %Dev  Lambda
## [1,]  8 0.6878 0.09646
```

```r
coef(ridge, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  2.45234509
## lcavol       0.60438317
## lweight      0.28576500
## age         -0.10858418
## lbph         0.20096586
## svi          0.28336365
## lcp         -0.15469409
## gleason      0.01414138
## pgg45        0.20305366
```

```r
coef(refit2, s = "lambda.min")
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  2.45234509
## lcavol       0.60438730
## lweight      0.28573832
## age         -0.10855978
## lbph         0.20098549
## svi          0.28347129
## lcp         -0.15474777
## gleason      0.01410086
## pgg45        0.20306133
```

Q. How to print out the coefficient with tuning for PCR and PLS with intercept?… using these coef(plsrfunc, intercept = T) and plsrfunc$coefficients[, , which.min(plsrfunc$validation$PRESS)], does not give me....$ ===> $plsrfunc$coefficients[„which.min(plsrfunc$validation$PRESS)] and append the intercept like this: coef(plsrfunc, intercept = T)[1]. ===> c(intercept = coef(plsrfunc, intercept = T)[1], plsrfunc$coefficients[, , which.min(plsrfunc$validation

Q. When I test MSE here, since training and testing are given, and also we already conducted 10-fold CV for tuning parameter, so we do not need to any CV. And, we can just use predict to get answers? ===> Right!!!

Model Selection (20 pts)

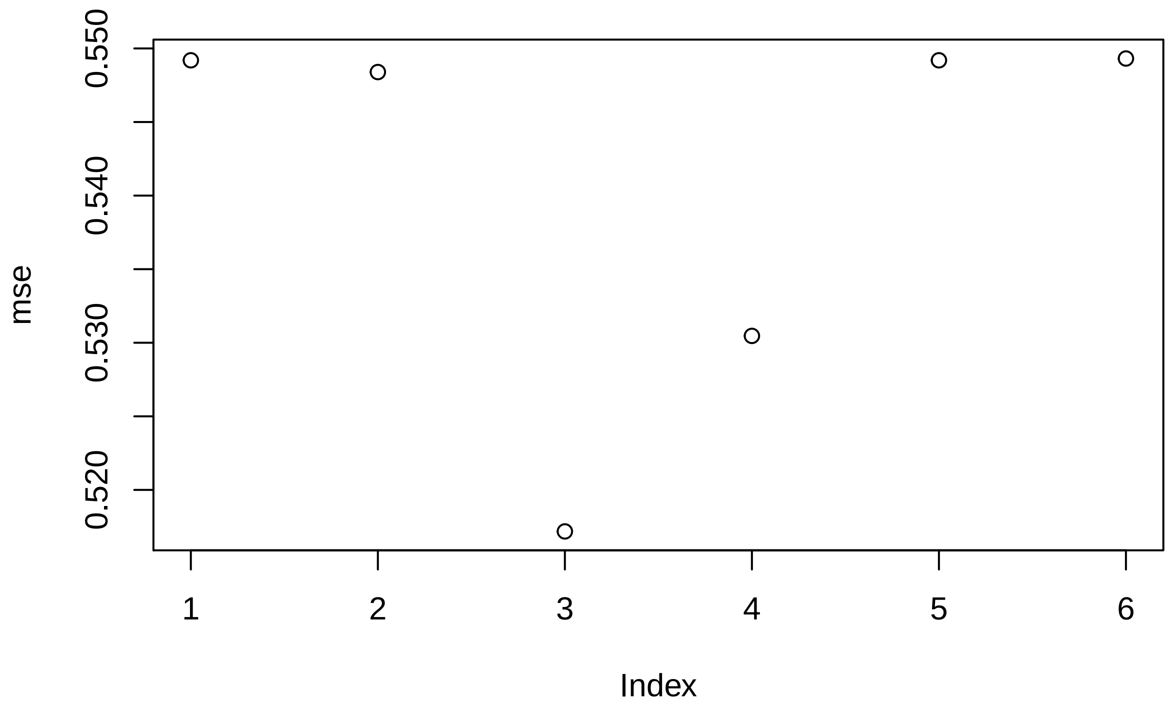
```r
subsetcoeffill <- matrix(0, 9, 1)
subsetcoeffill <- unname(rbind(as.matrix(subsetcoef), NA, NA, NA, NA, NA, NA)) #Fill NA for empty

Lasso <- coef(lasso, s = "lambda.min")[,1]
Lasso[c(8)] <- NA
Lasso <- unname(Lasso)


models <- data.frame(
      LS = as.vector(table3.2[,1]), "Best Subset" = subsetcoeffill,
      Ridge = as.vector(coef(ridge, s = "lambda.min")),
      Lasso = Lasso,
      #include an intercept with the minimum coefficients.
      PCR = unname(c(intercept = coef(pcrfunc, intercept = T)[1],
                     pcrfunc$coefficients[,,which.min(pcrfunc$validation$PRESS)])),
      PLS = unname(c(intercept = coef(plsrfunc, intercept = T)[1],
                     plsrfunc$coefficients[,,which.min(plsrfunc$validation$PRESS)]))
      )

#We scale x training and xtesting!!!
msefunc <- function(msefolder){
  x <- model.matrix(lm(lpsa ~.-1, data = as.data.frame(scale(training))))
  xint <- model.matrix(lm(lpsa ~., data = as.data.frame(scale(training))))

  ytest <- testing$lpsa
  xtest <- model.matrix(lm(lpsa ~., data = as.data.frame(scale(testing))))

  ytrain <- scale(training$lpsa)
  xtrain <- scale(xint)

  #OLS
  olsbeta <- table3.2[,1]
  mse[1] <- sum((ytest - (xtest %*% olsbeta))^2) / length(ytest)


  #Best Subset
  yhat <- xtest[,1:3] %*% subsetcoef
  #Have NA so calculate separately.
  mse[2] <- sum((ytest - yhat)^2) / length(ytest)

  #Ridge
  yhat <- predict(ridge, xtest[,-1], s = "lambda.min")
  mse[3] <- mean((ytest - yhat)^2)


  #Lasso
  yhat <- predict(lasso, xtest[,-1], s = "lambda.min")
  mse[4] <- mean((ytest - yhat)^2)


  #PCR
  yhat <- predict(pcrfunc, xtest[,-1], ncomp = unname(which.min(pcrfunc$validation$PRESS[1, ])))
  mse[5] <- mean((ytest - yhat)^2)


  #PLSR
  yhat <- predict(plsrfunc, xtest[,-1], ncomp = unname(which.min(plsrfunc$validation$PRESS[1, ])))
  mse[6] <- mean((ytest - yhat)^2)

  return(mse)
}

mse <- c()
mse <- msefunc(mse)

print("Here is the mse for 6 models:")
```

```
## [1] "Here is the mse for 6 models:"
```

```
mse
```

```
## [1] 0.5491941 0.5483947 0.5171794 0.5304655 0.5491941 0.5493153
```

```
plot(mse)
```



```r
models <- rbind(models, mse)
rownames(models) <- c("Intercept", "lcavol", "lweight",
                      "age", "lbph", "svi", "lcp", "gleason",
                      "pgg45", "Test Error")
models
```

```
##                     LS Best.Subset       Ridge      Lasso         PCR
## Intercept   2.45234509   2.4523451  2.45234509  2.4523451  2.45234509
## lcavol      0.71640701   0.7798589  0.60438317  0.6918697  0.71640701
## lweight     0.29264240   0.3519101  0.28576500  0.2887031  0.29264240
## age        -0.14254963          NA -0.10858418 -0.1268621 -0.14254963
## lbph        0.21200760          NA  0.20096586  0.2033674  0.21200760
## svi         0.30961953          NA  0.28336365  0.2940763  0.30961953
## lcp        -0.28900562          NA -0.15469409 -0.2389979 -0.28900562
## gleason    -0.02091352          NA  0.01414138         NA -0.02091352
## pgg45       0.27734595          NA  0.20305366  0.2357199  0.27734595
## Test Error  0.54919414   0.5483947  0.51717940  0.5304655  0.54919414
##                   PLS
## Intercept   2.4523451
## lcavol      0.7104094
## lweight     0.2952801
## age        -0.1446106
## lbph        0.2124677
## svi         0.3169434
## lcp        -0.2922292
## gleason    -0.0149234
## pgg45       0.2748280
## Test Error  0.5493153
```

Comment:

From the table I got, *RIDGE* (or best subset) is the best model, as it has the smallest MSE. They changed based on set.seed().