154HW4

Jiyoon Clover Jeong

10/11/2017

```r
library(ElemStatLearn)
library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

```r
library(leaps)
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'caret'

## The following object is masked from 'package:pls':
##
##     R2
```

Problem 1 (10 pts)

$$r_{1,4} = \frac{cov(X1, X4)}{SD(X1)SD(X4)}$$

and cov(X1, X4) = cov(X1, X1 + X2 + X3) = cov(X1, X1) + cov(X1, X2) + cov(X1, X3) = var(X1) + 0 + 0 = var(X1) = σ_1^2

var(X4) = var(X1 + X2 + X3) = var(X1) + var(X2) + var(X3) = $\sigma_1^2 + \sigma_1^2 + \sigma_1^2 = 3\sigma_1^2$

Therefore,

$$r_{1,4} = \frac{\sigma_1^2}{\sqrt{\sigma_1^2 * 3\sigma_1^2}} = 1/\sqrt{3} = 0.5773503$$

$$r_{2,4} = \frac{cov(X2, X4)}{SD(X2)SD(X4)}$$

and cov(X2, X4) = cov(X2, X1 + X2 + X3) = cov(X2, X1) + cov(X2, X2) + cov(X2, X3) = var(X2) = σ_1^2

var(X4) = var(X1 + X2 + X3) = var(X1) + var(X2) + var(X3) = $\sigma_1^2 + \sigma_1^2 + \sigma_1^2 = 3\sigma_1^2$

Therefore,

$$r_{2,4} = \frac{\sigma_1^2}{\sqrt{\sigma_1^2 * 3\sigma_1^2}} = 1/\sqrt{3} = 0.5773503$$

$$r_{3,4} = \frac{cov(X3, X4)}{SD(X3)SD(X4)}$$

and cov(X3, X4) = cov(X3, X1 + X2 + X3) = cov(X3, X1) + cov(X3, X2) + cov(X3, X3) = var(X3) = σ_1^2

var(X4) = var(X1 + X2 + X3) = var(X1) + var(X2) + var(X3) = $\sigma_1^2 + \sigma_1^2 + \sigma_1^2 = 3\sigma_1^2$

Therefore,

$$r_{3,4} = \frac{\sigma_1^2}{\sqrt{\sigma_1^2 * 3\sigma_1^2}} = 1/\sqrt{3} = 0.5773503$$

Problem 2

Show that any two components

$$z_h^T z_l = 0$$

for

$$(h \neq l)$$

are indeed orthogonal.

** Base Case (i+1) **

$z_i^T z_{i+1} = z_i^T(\frac{X_i w_{i+1}}{w_{i+1}^T w_{i+1}}) = \frac{1}{w_{i+1}^T w_{i+1}} z_i^T(X_i w_{i+1})$. ——–> $z_i^T(X_i w_{i+1}) = 0$.

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

** Recursion (i+2) **

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

$z_i^T z_{i+1} = 0$ as we proved in the Base case.

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

—-> Show $z_i^T X_i = 0$.

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

——–> $z_i^T z_{i+2} = 0$

Proof finished.

Problem 3

```
length(names(prostate))
```

```
## [1] 10
```

```r
names(prostate)
```

```
##  [1] "lcavol"  "lweight" "age"     "lbph"    "svi"     "lcp"     "gleason"
##  [8] "pgg45"   "lpsa"    "train"
```

```r
train <- prostate[prostate$train == "TRUE", -10 ]

test <- prostate[prostate$train == "FALSE", -10 ]
```

Correlations of predictors, and some preprocessing (10 pts)

```r
cor(train[,-9])
```

```
##             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
```

```r
train_stan <- scale(train[,-9])
```

```r
summary(train_stan)
```

```
##      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 Squares Model (10 pts)

```r
train_stan <- as.data.frame(cbind(train_stan, lpsa = train$lpsa))

lsfit <- lm(lpsa ~., data = train_stan)
summary(lsfit)$coefficients[,1:3]
```

```
##                Estimate Std. Error    t value
## (Intercept)  2.45234509 0.08701959 28.1815274
## lcavol       0.71640701 0.13350135  5.3662905
## lweight      0.29264240 0.10638488  2.7507894
## age         -0.14254963 0.10211957 -1.3959090
## lbph         0.21200760 0.10312428  2.0558456
## svi          0.30961953 0.12538985  2.4692552
## lcp         -0.28900562 0.15480404 -1.8669126
## gleason     -0.02091352 0.14257805 -0.1466812
## pgg45        0.27734595 0.15959237  1.7378397
```

Best Subset Regression (10 pts)

```r
subset <- regsubsets(lpsa ~., data = train_stan)
coef(subset, 1:3)
```

```
## [[1]]
## (Intercept)      lcavol
##   2.4523451   0.8855136
##
## [[2]]
## (Intercept)      lcavol     lweight
##   2.4523451   0.7798589   0.3519101
##
## [[3]]
## (Intercept)      lcavol     lweight         svi
##   2.4523451   0.6461297   0.3511573   0.2259134
```

```r
summary(subset)
```

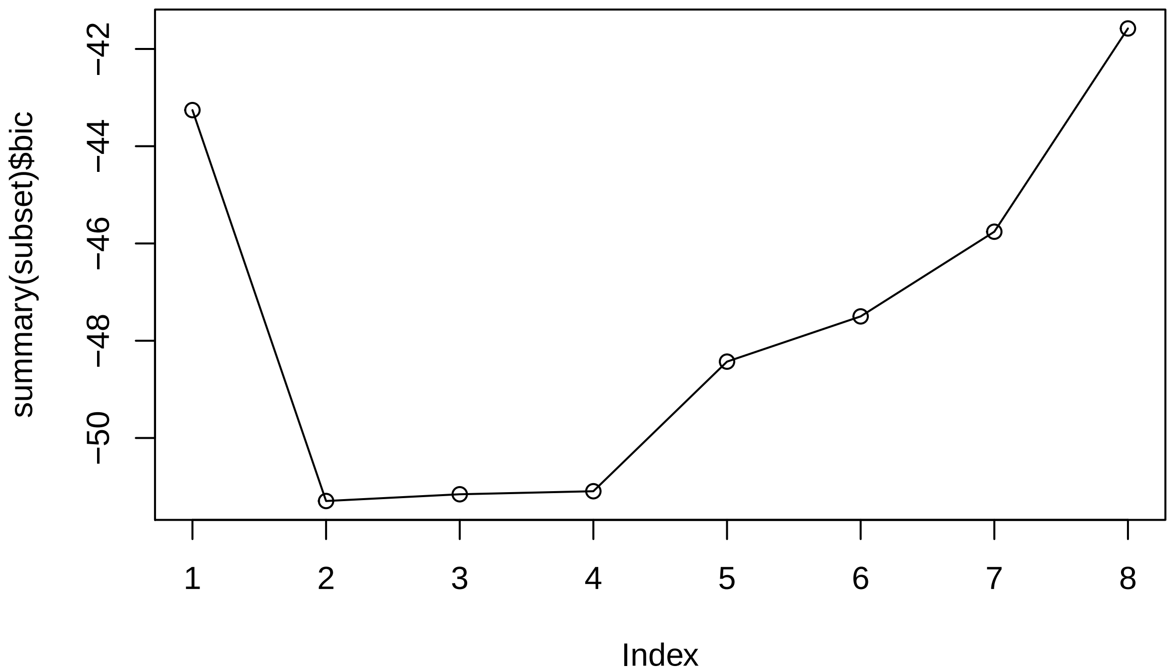
```
## Subset selection object
## Call: regsubsets.formula(lpsa ~ ., data = train_stan)
## 8 Variables  (and intercept)
##         Forced in Forced out
## lcavol      FALSE      FALSE
## lweight     FALSE      FALSE
## age         FALSE      FALSE
## lbph        FALSE      FALSE
## svi         FALSE      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
# Best number of subset



minsubset <- which.min(summary(subset)$bic)
cat("The best number of subset is ", minsubset, "\n")
```

```
## The best number of subset is  2
```

```r
plot(summary(subset)$bic, type = "o")
```



PCR and PLSR (40 pts)

```r
set.seed(10)

pcr_model <- pcr(lpsa~., data = train_stan, validation = "CV")

# how to change number of folds in validation?
```

```r
which.min(pcr_model$validation$PRESS)
```

```
## [1] 8
```

```r
summary(pcr_model)
```

```
## 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   0.9172   0.8975   0.8372   0.8323   0.8485   0.8467
## adjCV        1.217   0.9141   0.8949   0.8330   0.8269   0.8431   0.8419
##        7 comps  8 comps
## CV      0.8066   0.7833
## adjCV   0.8012   0.7767
##
## 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
```

```r
# Model fits with the smallest CV-MSE of
cat("PCR Tuning parameter : ", which.min(pcr_model$validation$PRESS))
```

```
## PCR Tuning parameter :  8
```

```r
print("Associated coefficients : ")
```

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

```r
pcr_model$coefficients[, , which.min(pcr_model$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
```

```r
plsr_model <- plsr(lpsa~., data = train_stan, validation = "CV")
summary(plsr_model)
```

```
## 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   0.8475   0.7885   0.7709   0.7616   0.7530   0.7524
## adjCV        1.217   0.8453   0.7844   0.7667   0.7559   0.7482   0.7476
```

```
##        7 comps  8 comps
## CV      0.7520   0.7521
## adjCV   0.7473   0.7474
##
## 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
```

```r
plsr_model
```

```
## Partial least squares regression , fitted with the kernel algorithm.
## Cross-validated using 10 random segments.
## Call:
## plsr(formula = lpsa ~ ., data = train_stan, validation = "CV")
```

```r
# Model fits with the smallest CV-MSE of

# Q  how to get CV table from summary???

# CV MSE :  MSEP(pcr_model)

cat("Plsr Tuning parameter : ", which.min(plsr_model$validation$PRESS))
```

```
## Plsr Tuning parameter :  7
```

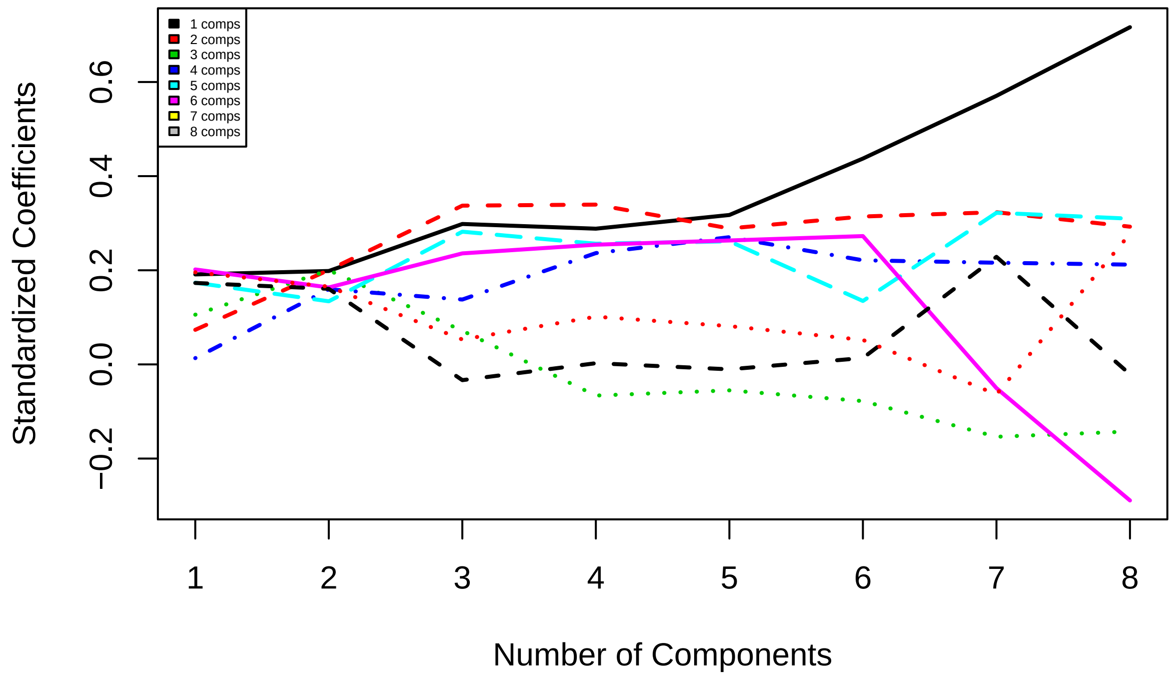
```r
print("Associated coefficients : ")
```

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

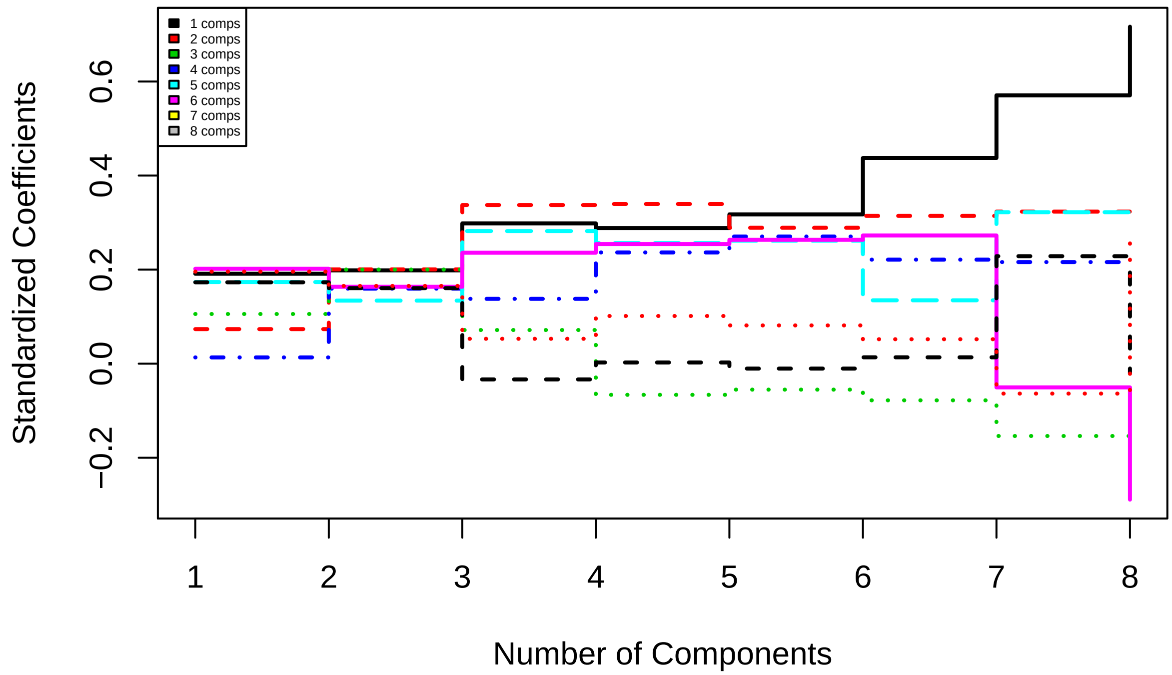
```r
plsr_model$coefficients[, , which.min(plsr_model$validation$PRESS)]
```

```
##      lcavol     lweight         age        lbph         svi         lcp
##  0.71636186  0.29192059 -0.14233890  0.21259719  0.31026935 -0.28905136
##     gleason       pgg45
## -0.02101011  0.27706347
```

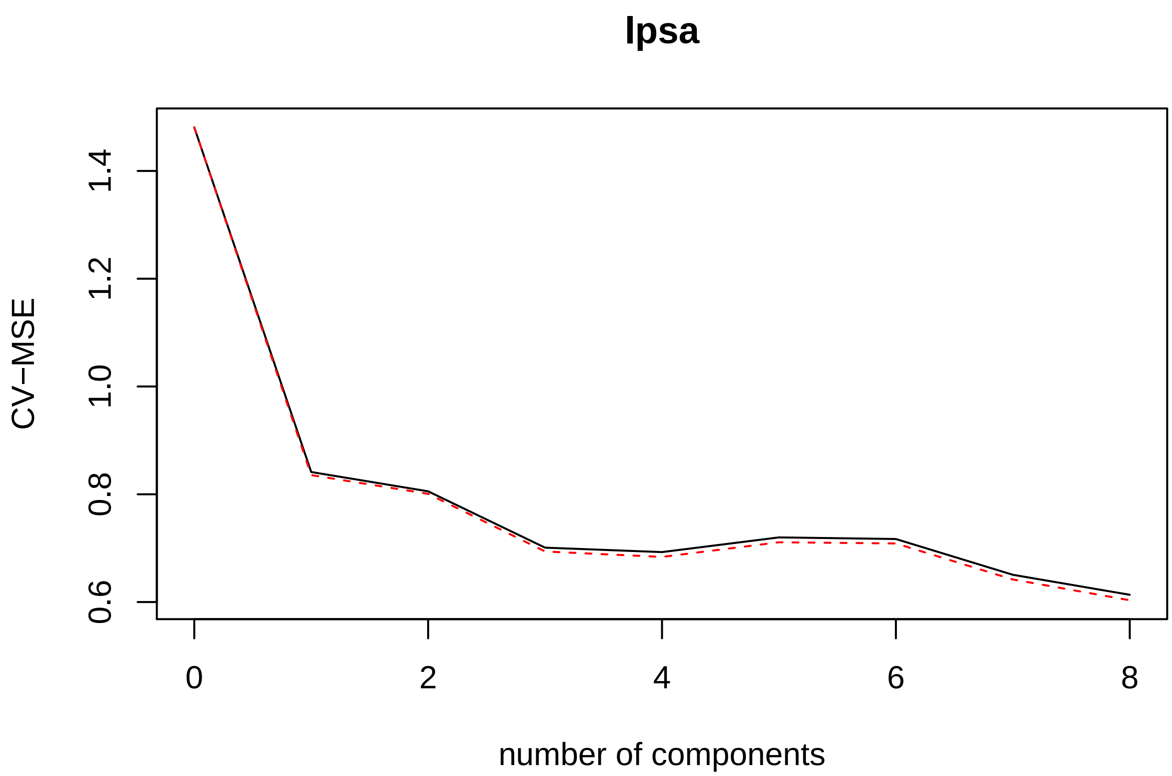
```r
# fit = glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa)
# plot(fit)


# Standardized coefficients??

#  Q   is this plot correct  --- >standardized coefficients ???


# coefplot(pcr_model, ncomp = 1:8, legendpos = "bottomright", xlab = "number of components")
# plot(pcr_model, plottype = "coefficients", ncomp = 1:8)

pcr_coefs = apply(pcr_model$coefficients, 3, function(x) x)

matplot(t(pcr_coefs), type= 'l', lwd = 2, xlab = "Number of Components", ylab = "Standardized Coefficie
legend("topleft", colnames(pcr_coefs),col=seq_len(ncol(pcr_coefs)),cex=0.4,fill=seq_len(ncol(pcr_coefs)
```



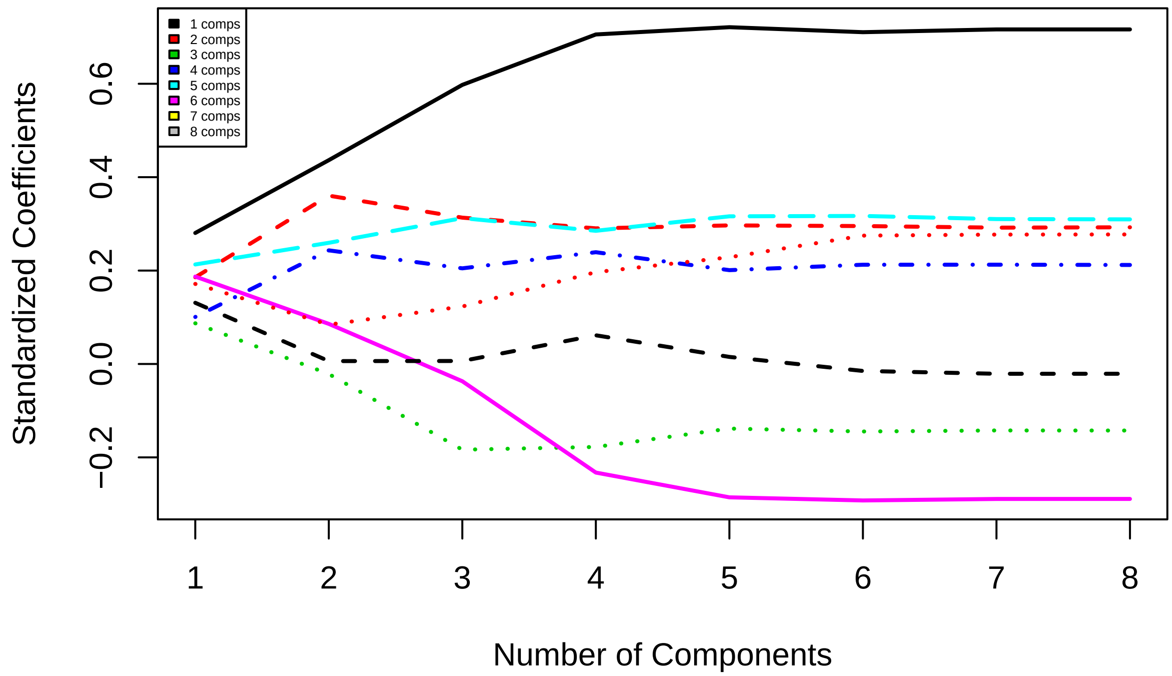

```
matplot(t(pcr_coefs), type= 's', lwd = 2, xlab = "Number of Components", ylab = "Standardized Coefficie
legend("topleft", colnames(pcr_coefs),col=seq_len(ncol(pcr_coefs)),cex=0.4,fill=seq_len(ncol(pcr_coefs))
```



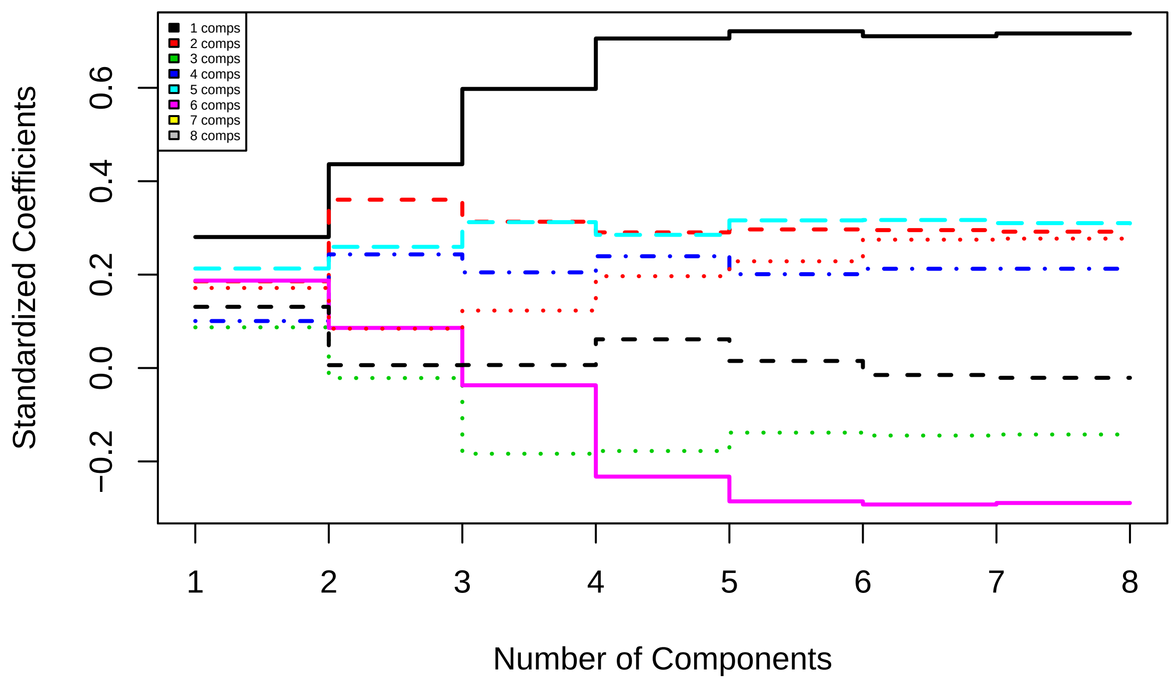

```
validationplot(pcr_model, val.type = "MSEP", ylab = "CV-MSE")
```



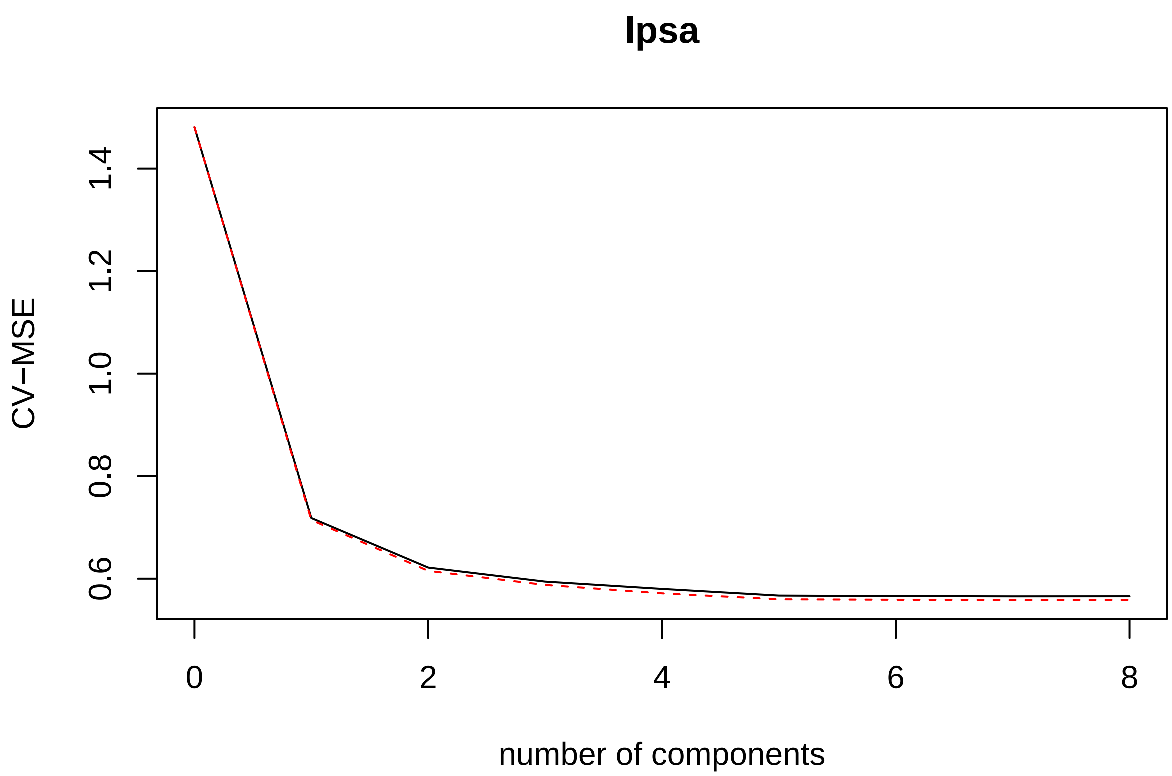

```
#plot(pcr_model$validation$PRESS[1,], type = "o", main = "PCR")


# coefplot(plsr_model, ncomp = 1:8, legendpos = "bottomright")
# plot(plsr_model, plottype = "coefficients", ncomp = 1:8)


plsr_coefs = apply(plsr_model$coefficients, 3, function(x) x)

matplot(t(plsr_coefs), type= 'l', lwd = 2, xlab = "Number of Components", ylab = "Standardized Coeffici
legend("topleft", colnames(plsr_coefs),col=seq_len(ncol(plsr_coefs)),cex=0.4,fill=seq_len(ncol(plsr_coef
```




```
matplot(t(plsr_coefs), type= 's', lwd = 2, xlab = "Number of Components", ylab = "Standardized Coeffici
legend("topleft", colnames(plsr_coefs),col=seq_len(ncol(plsr_coefs)),cex=0.4,fill=seq_len(ncol(plsr_coe
```




```
validationplot(plsr_model, val.type = "MSEP", ylab = "CV-MSE")
```



RR and Lasso (40 pts)

```
set.seed(10)

# Fitting the model (Ridge: Alpha = 0)
ridgecv <- cv.glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0)

ridgecv
```

```
## $lambda
##  [1] 878.88041366 800.80310001 729.66195971 664.84080224 605.77817775
##  [6] 551.96251403 502.92768556 458.24897610 417.53940005 380.44635054
## [11] 346.64854531 315.85324395 287.79371228 262.22691208 238.93139593
## [16] 217.70538924 198.36504248 180.74283883 164.68614317 150.05588011
## [21] 136.72532929 124.57902787 113.51177039 103.42769756  94.23946596
## [26]  85.86749154  78.23926025  71.28870001  64.95560838  59.18513116
## [31]  53.92728723  49.13653566  44.77138126  40.79401514  37.16998727
## [36]  33.86790804  30.85917644  28.11773225  25.61983042  23.34383529
## [41]  21.27003330  19.38046217  17.65875532  16.09000017  14.66060890
## [46]  13.35820081  12.17149507  11.09021299  10.10498904   9.20728966
## [51]   8.38933942   7.64405363   6.96497698   6.34622762   5.78244625
## [56]   5.26874966   4.80068846   4.37420853   3.98561590   3.63154477
## [61]   3.30892834   3.01497227   2.74713044   2.50308294   2.28071595
## [66]   2.07810343   1.89349045   1.72527798   1.57200904   1.43235609
## [71]   1.30510952   1.18916719   1.08352485   0.98726749   0.89956137
## [76]   0.81964682   0.74683165   0.68048519   0.62003276   0.56495076
## [81]   0.51476209   0.46903204   0.42736453   0.38939864   0.35480554
## [86]   0.32328559   0.29456579   0.26839738   0.24455370   0.22282822
```

```
## [91]   0.20303277   0.18499590   0.16856138   0.15358685   0.13994262
## [96]   0.12751050   0.11618282   0.10586146   0.09645702
##
## $cvm
##  [1] 1.4624171 1.4588464 1.4557350 1.4547654 1.4539582 1.4530739 1.4521052
##  [8] 1.4510443 1.4498826 1.4486110 1.4472192 1.4456963 1.4440305 1.4422090
## [15] 1.4402178 1.4380421 1.4356658 1.4330714 1.4302405 1.4271531 1.4237880
## [22] 1.4201226 1.4161330 1.4117939 1.4070786 1.4019592 1.3964066 1.3903908
## [29] 1.3838807 1.3768449 1.3692513 1.3610681 1.3522638 1.3428079 1.3326713
## [36] 1.3218273 1.3102518 1.2979243 1.2848292 1.2709557 1.2562998 1.2408642
## [43] 1.2246599 1.2077064 1.1900326 1.1716772 1.1526892 1.1331273 1.1130601
## [50] 1.0925653 1.0717288 1.0506434 1.0294073 1.0081221 0.9868914 0.9658180
## [57] 0.9450023 0.9245401 0.9045207 0.8850256 0.8661268 0.8478860 0.8303540
## [64] 0.8135704 0.7975639 0.7823526 0.7679446 0.7543392 0.7415273 0.7294934
## [71] 0.7182160 0.7076692 0.6978235 0.6886473 0.6801064 0.6721697 0.6647990
## [78] 0.6579699 0.6516401 0.6457817 0.6403657 0.6353647 0.6307637 0.6265293
## [85] 0.6226447 0.6190893 0.6158408 0.6128857 0.6102119 0.6078035 0.6056559
## [92] 0.6037363 0.6020391 0.6005525 0.5992583 0.5981385 0.5971935 0.5964014
## [99] 0.5957481
##
## $cvsd
##  [1] 0.25120463 0.25152114 0.24997254 0.24970794 0.24957030 0.24941941
##  [7] 0.24925403 0.24907278 0.24887416 0.24865655 0.24841816 0.24815706
## [13] 0.24787115 0.24755812 0.24721549 0.24684055 0.24643039 0.24598182
## [19] 0.24549140 0.24495545 0.24436995 0.24373062 0.24303284 0.24227166
## [25] 0.24144179 0.24053760 0.23955310 0.23848194 0.23731744 0.23605255
## [31] 0.23467994 0.23319195 0.23158069 0.22983804 0.22795578 0.22592557
## [37] 0.22373914 0.22138835 0.21886533 0.21616264 0.21327343 0.21019165
## [43] 0.20691223 0.20343136 0.19974667 0.19585749 0.19176518 0.18747332
## [49] 0.18298797 0.17831793 0.17347494 0.16847383 0.16333271 0.15807299
## [55] 0.15271947 0.14730021 0.14184646 0.13639241 0.13097493 0.12563308
## [61] 0.12040765 0.11534052 0.11047385 0.10584916 0.10150632 0.09748226
## [67] 0.09380970 0.09051572 0.08762043 0.08513578 0.08306471 0.08140088
## [73] 0.08012883 0.07922498 0.07865857 0.07839414 0.07839664 0.07862499
## [79] 0.07904075 0.07960777 0.08029215 0.08106522 0.08190291 0.08277949
## [85] 0.08367818 0.08458392 0.08548206 0.08636460 0.08722548 0.08805939
## [91] 0.08887153 0.08964818 0.09038762 0.09109286 0.09176492 0.09240508
## [97] 0.09301170 0.09358723 0.09413297
##
## $cvup
##  [1] 1.7136218 1.7103675 1.7057076 1.7044733 1.7035285 1.7024933 1.7013592
##  [8] 1.7001171 1.6987568 1.6972675 1.6956373 1.6938534 1.6919017 1.6897671
## [15] 1.6874333 1.6848827 1.6820962 1.6790532 1.6757319 1.6721085 1.6681579
## [22] 1.6638532 1.6591659 1.6540656 1.6485204 1.6424968 1.6359597 1.6288727
## [29] 1.6211982 1.6128974 1.6039312 1.5942600 1.5838445 1.5726459 1.5606271
## [36] 1.5477529 1.5339909 1.5193127 1.5036945 1.4871184 1.4695732 1.4510559
## [43] 1.4315722 1.4111377 1.3897793 1.3675347 1.3444544 1.3206006 1.2960481
## [50] 1.2708832 1.2452038 1.2191172 1.1927400 1.1661951 1.1396109 1.1131182
## [57] 1.0868487 1.0609325 1.0354957 1.0106587 0.9865345 0.9632265 0.9408278
## [64] 0.9194195 0.8990702 0.8798349 0.8617543 0.8448549 0.8291477 0.8146292
## [71] 0.8012807 0.7890700 0.7779524 0.7678723 0.7587650 0.7505639 0.7431956
## [78] 0.7365949 0.7306809 0.7253895 0.7206578 0.7164300 0.7126666 0.7093088
## [85] 0.7063229 0.7036732 0.7013228 0.6992503 0.6974374 0.6958629 0.6945275
## [92] 0.6933845 0.6924268 0.6916454 0.6910232 0.6905436 0.6902052 0.6899887
```

```
## [99] 0.6898811
##
## $cvlo
##  [1] 1.2112125 1.2073252 1.2057625 1.2050575 1.2043879 1.2036545 1.2028512
##  [8] 1.2019715 1.2010085 1.1999544 1.1988010 1.1975392 1.1961594 1.1946509
## [15] 1.1930024 1.1912016 1.1892354 1.1870896 1.1847491 1.1821976 1.1794180
## [22] 1.1763920 1.1731002 1.1695222 1.1656368 1.1614216 1.1568535 1.1519088
## [29] 1.1465633 1.1407923 1.1345713 1.1278761 1.1206831 1.1129698 1.1047156
## [36] 1.0959017 1.0865126 1.0765360 1.0659638 1.0547931 1.0430263 1.0306726
## [43] 1.0177477 1.0042750 0.9902859 0.9758198 0.9609240 0.9456540 0.9300721
## [50] 0.9142474 0.8982539 0.8821696 0.8660746 0.8500491 0.8341719 0.8185178
## [57] 0.8031558 0.7881476 0.7735458 0.7593926 0.7457192 0.7325455 0.7198801
## [64] 0.7077212 0.6960576 0.6848703 0.6741349 0.6638234 0.6539069 0.6443576
## [71] 0.6351513 0.6262683 0.6176947 0.6094223 0.6014479 0.5937756 0.5864024
## [78] 0.5793449 0.5725994 0.5661740 0.5600735 0.5542995 0.5488608 0.5437498
## [85] 0.5389665 0.5345053 0.5303587 0.5265211 0.5229865 0.5197442 0.5167844
## [92] 0.5140882 0.5116515 0.5094596 0.5074933 0.5057334 0.5041818 0.5028142
## [99] 0.5016152
##
## $nzero
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71
##   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
## s72 s73 s74 s75 s76 s77 s78 s79 s80 s81 s82 s83 s84 s85 s86 s87 s88 s89
##   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8   8
## s90 s91 s92 s93 s94 s95 s96 s97 s98
##   8   8   8   8   8   8   8   8   8
##
## $name
##                  mse
## "Mean-Squared Error"
##
## $glmnet.fit
##
## Call:  glmnet(x = as.matrix(train_stan[, -9]), y = train_stan$lpsa,      alpha = 0)
##
##        Df      %Dev    Lambda
##   [1,]  8 3.558e-36 878.90000
##   [2,]  8 5.236e-03 800.80000
##   [3,]  8 5.743e-03 729.70000
##   [4,]  8 6.298e-03 664.80000
##   [5,]  8 6.906e-03 605.80000
##   [6,]  8 7.573e-03 552.00000
##   [7,]  8 8.303e-03 502.90000
##   [8,]  8 9.102e-03 458.20000
##   [9,]  8 9.978e-03 417.50000
##  [10,]  8 1.094e-02 380.40000
##  [11,]  8 1.199e-02 346.60000
##  [12,]  8 1.313e-02 315.90000
```

```
## [13,] 8 1.439e-02 287.80000
## [14,] 8 1.576e-02 262.20000
## [15,] 8 1.726e-02 238.90000
## [16,] 8 1.890e-02 217.70000
## [17,] 8 2.070e-02 198.40000
## [18,] 8 2.265e-02 180.70000
## [19,] 8 2.479e-02 164.70000
## [20,] 8 2.711e-02 150.10000
## [21,] 8 2.965e-02 136.70000
## [22,] 8 3.242e-02 124.60000
## [23,] 8 3.543e-02 113.50000
## [24,] 8 3.870e-02 103.40000
## [25,] 8 4.226e-02  94.24000
## [26,] 8 4.612e-02  85.87000
## [27,] 8 5.031e-02  78.24000
## [28,] 8 5.485e-02  71.29000
## [29,] 8 5.976e-02  64.96000
## [30,] 8 6.507e-02  59.19000
## [31,] 8 7.081e-02  53.93000
## [32,] 8 7.699e-02  49.14000
## [33,] 8 8.364e-02  44.77000
## [34,] 8 9.078e-02  40.79000
## [35,] 8 9.845e-02  37.17000
## [36,] 8 1.066e-01  33.87000
## [37,] 8 1.154e-01  30.86000
## [38,] 8 1.247e-01  28.12000
## [39,] 8 1.346e-01  25.62000
## [40,] 8 1.451e-01  23.34000
## [41,] 8 1.562e-01  21.27000
## [42,] 8 1.679e-01  19.38000
## [43,] 8 1.802e-01  17.66000
## [44,] 8 1.931e-01  16.09000
## [45,] 8 2.065e-01  14.66000
## [46,] 8 2.205e-01  13.36000
## [47,] 8 2.349e-01  12.17000
## [48,] 8 2.498e-01  11.09000
## [49,] 8 2.651e-01  10.10000
## [50,] 8 2.808e-01   9.20700
## [51,] 8 2.967e-01   8.38900
## [52,] 8 3.128e-01   7.64400
## [53,] 8 3.291e-01   6.96500
## [54,] 8 3.455e-01   6.34600
## [55,] 8 3.618e-01   5.78200
## [56,] 8 3.781e-01   5.26900
## [57,] 8 3.942e-01   4.80100
## [58,] 8 4.100e-01   4.37400
## [59,] 8 4.256e-01   3.98600
## [60,] 8 4.408e-01   3.63200
## [61,] 8 4.556e-01   3.30900
## [62,] 8 4.699e-01   3.01500
## [63,] 8 4.838e-01   2.74700
## [64,] 8 4.971e-01   2.50300
## [65,] 8 5.099e-01   2.28100
## [66,] 8 5.221e-01   2.07800
```

```
##  [67,]  8 5.337e-01   1.89300
##  [68,]  8 5.447e-01   1.72500
##  [69,]  8 5.552e-01   1.57200
##  [70,]  8 5.651e-01   1.43200
##  [71,]  8 5.745e-01   1.30500
##  [72,]  8 5.833e-01   1.18900
##  [73,]  8 5.917e-01   1.08400
##  [74,]  8 5.995e-01   0.98730
##  [75,]  8 6.068e-01   0.89960
##  [76,]  8 6.137e-01   0.81960
##  [77,]  8 6.202e-01   0.74680
##  [78,]  8 6.263e-01   0.68050
##  [79,]  8 6.319e-01   0.62000
##  [80,]  8 6.372e-01   0.56500
##  [81,]  8 6.422e-01   0.51480
##  [82,]  8 6.468e-01   0.46900
##  [83,]  8 6.511e-01   0.42740
##  [84,]  8 6.551e-01   0.38940
##  [85,]  8 6.589e-01   0.35480
##  [86,]  8 6.623e-01   0.32330
##  [87,]  8 6.655e-01   0.29460
##  [88,]  8 6.685e-01   0.26840
##  [89,]  8 6.712e-01   0.24460
##  [90,]  8 6.737e-01   0.22280
##  [91,]  8 6.759e-01   0.20300
##  [92,]  8 6.780e-01   0.18500
##  [93,]  8 6.799e-01   0.16860
##  [94,]  8 6.816e-01   0.15360
##  [95,]  8 6.831e-01   0.13990
##  [96,]  8 6.845e-01   0.12750
##  [97,]  8 6.858e-01   0.11620
##  [98,]  8 6.869e-01   0.10590
##  [99,]  8 6.878e-01   0.09646
## [100,]  8 6.887e-01   0.08789
##
## $lambda.min
## [1] 0.09645702
##
## $lambda.1se
## [1] 0.9872675
##
## attr(,"class")
## [1] "cv.glmnet"
```

```r
coef(ridgecv, 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
opt_lambda <- ridgecv$lambda.min
opt_lambda
```

```
## [1] 0.09645702
```

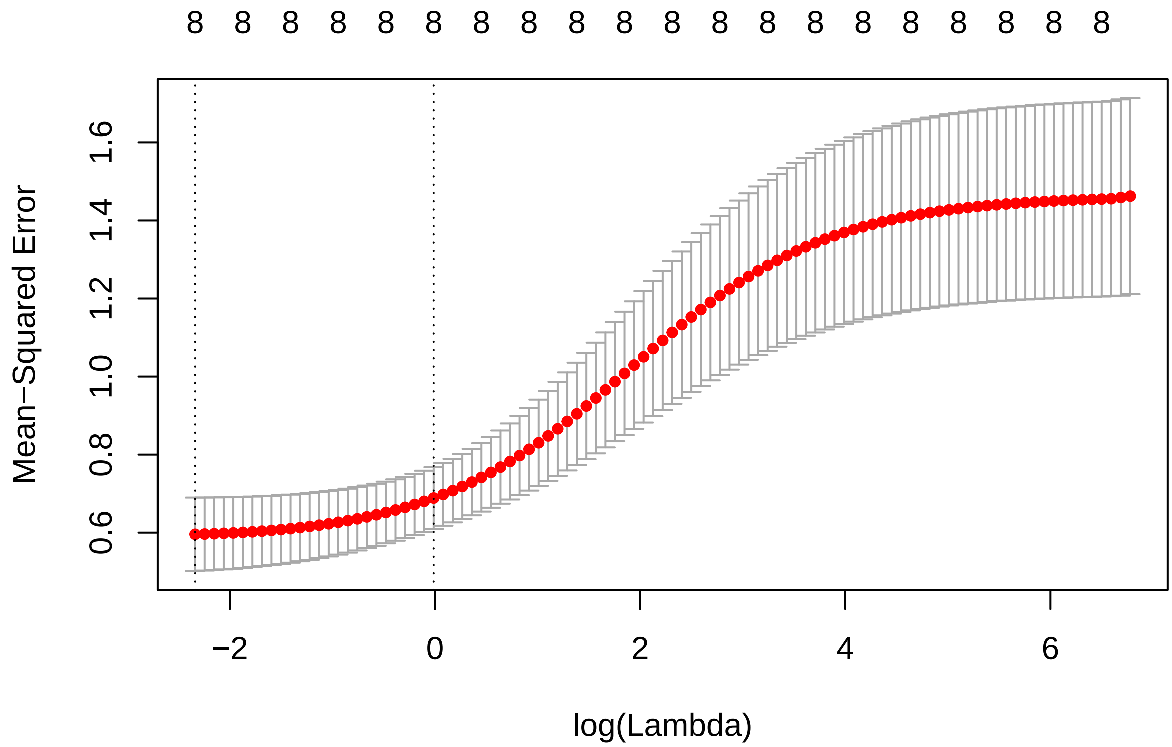
```r
# Q  lambda.min.ratio = opt_lambda


ridge <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0, lambda.min.ratio = opt_lambda)

ridge <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0, lambda = opt_lambda)



# Q is summary enough???
summary(ridge)
```

```
##           Length Class     Mode
## a0        1      -none-    numeric
## beta      8      dgCMatrix S4
## df        1      -none-    numeric
## dim       2      -none-    numeric
## lambda    1      -none-    numeric
## dev.ratio 1      -none-    numeric
## nulldev   1      -none-    numeric
## npasses   1      -none-    numeric
## jerr      1      -none-    numeric
## offset    1      -none-    logical
## call      5      -none-    call
## nobs      1      -none-    numeric
```

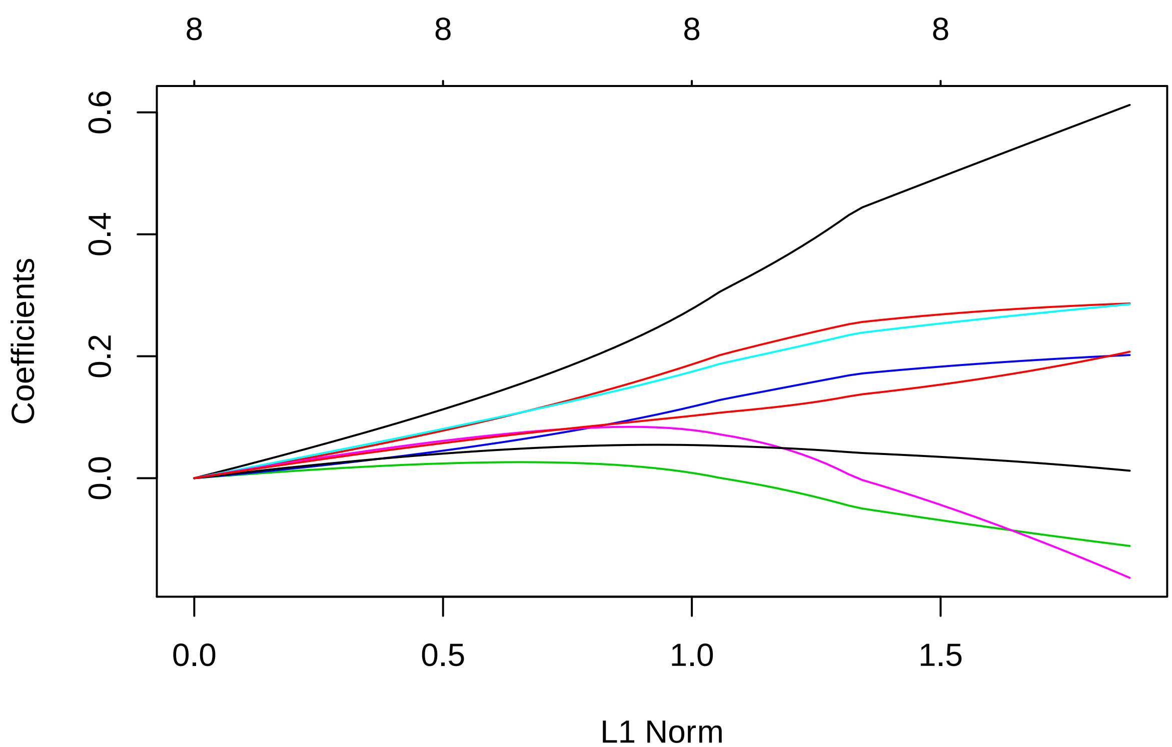
```r
# Q :  ridge.coef gives the best model's coefficients????


coef(ridgecv, s = ridgecv$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
plot.cv.glmnet(ridgecv)
```




```
#   Q :    ???? straight?

ridge <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0)


plot.glmnet(ridge)
```



```r
set.seed(10)
# Fitting the model (Lasso: Alpha = 1)


lassocv <- cv.glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 1)

lassocv
```

```
## $lambda
##  [1] 0.878880414 0.800803100 0.729661960 0.664840802 0.605778178
##  [6] 0.551962514 0.502927686 0.458248976 0.417539400 0.380446351
## [11] 0.346648545 0.315853244 0.287793712 0.262226912 0.238931396
## [16] 0.217705389 0.198365042 0.180742839 0.164686143 0.150055880
## [21] 0.136725329 0.124579028 0.113511770 0.103427698 0.094239466
## [26] 0.085867492 0.078239260 0.071288700 0.064955608 0.059185131
## [31] 0.053927287 0.049136536 0.044771381 0.040794015 0.037169987
## [36] 0.033867908 0.030859176 0.028117732 0.025619830 0.023343835
## [41] 0.021270033 0.019380462 0.017658755 0.016090000 0.014660609
## [46] 0.013358201 0.012171495 0.011090213 0.010104989 0.009207290
## [51] 0.008389339 0.007644054 0.006964977 0.006346228 0.005782446
## [56] 0.005268750 0.004800688 0.004374209 0.003985616 0.003631545
## [61] 0.003308928 0.003014972 0.002747130 0.002503083 0.002280716
## [66] 0.002078103 0.001893490 0.001725278
##
## $cvm
##  [1] 1.4495864 1.3748827 1.2661084 1.1715141 1.0929790 1.0277767 0.9748502
##  [8] 0.9308870 0.8914314 0.8559751 0.8194052 0.7864824 0.7597421 0.7381668
## [15] 0.7217217 0.7078173 0.6958802 0.6846987 0.6730528 0.6629171 0.6532195
## [22] 0.6450963 0.6382981 0.6312325 0.6252097 0.6201932 0.6165538 0.6152285
## [29] 0.6146022 0.6143005 0.6148388 0.6166678 0.6174565 0.6166245 0.6156366
## [36] 0.6148322 0.6136893 0.6108610 0.6082566 0.6061759 0.6043631 0.6023808
## [43] 0.6004946 0.5991178 0.5981771 0.5974701 0.5969291 0.5965204 0.5962029
## [50] 0.5959843 0.5958515 0.5957964 0.5958097 0.5958886 0.5959800 0.5960695
## [57] 0.5961649 0.5962521 0.5963552 0.5964501 0.5965507 0.5966405 0.5967238
## [64] 0.5968069 0.5968945 0.5969681 0.5970432 0.5971159
##
## $cvsd
##  [1] 0.25327067 0.26159165 0.23977068 0.21797620 0.19887861 0.18207992
##  [7] 0.16685341 0.15395216 0.14365053 0.13530767 0.12572314 0.11688194
## [13] 0.10931720 0.10291853 0.09721438 0.09218828 0.08813678 0.08510700
## [19] 0.08298359 0.08147538 0.08009646 0.07929736 0.07896562 0.07827778
## [25] 0.07802900 0.07818849 0.07872737 0.07964203 0.08081182 0.08208932
## [31] 0.08351785 0.08503489 0.08663478 0.08822121 0.08960356 0.09084344
## [37] 0.09172827 0.09222408 0.09257572 0.09296872 0.09334075 0.09355016
## [43] 0.09369661 0.09401804 0.09447775 0.09496337 0.09543752 0.09589600
## [49] 0.09633315 0.09675071 0.09714485 0.09751823 0.09786233 0.09817762
## [55] 0.09847383 0.09874705 0.09900509 0.09924459 0.09946556 0.09966971
## [61] 0.09986063 0.10003448 0.10019260 0.10034093 0.10047097 0.10059496
## [67] 0.10070746 0.10081329
##
## $cvup
##  [1] 1.7028571 1.6364744 1.5058791 1.3894903 1.2918576 1.2098566 1.1417036
##  [8] 1.0848391 1.0350820 0.9912827 0.9451284 0.9033644 0.8690593 0.8410853
## [15] 0.8189361 0.8000056 0.7840170 0.7698057 0.7560364 0.7443924 0.7333159
```

```
## [22] 0.7243936 0.7172637 0.7095103 0.7032387 0.6983817 0.6952812 0.6948705
## [29] 0.6954140 0.6963899 0.6983566 0.7017027 0.7040912 0.7048457 0.7052401
## [36] 0.7056756 0.7054176 0.7030851 0.7008324 0.6991446 0.6977039 0.6959310
## [43] 0.6941912 0.6931358 0.6926548 0.6924335 0.6923667 0.6924164 0.6925361
## [50] 0.6927350 0.6929963 0.6933146 0.6936720 0.6940662 0.6944538 0.6948165
## [57] 0.6951700 0.6954967 0.6958207 0.6961198 0.6964113 0.6966750 0.6969164
## [64] 0.6971478 0.6973654 0.6975631 0.6977507 0.6979292
##
## $cvlo
##  [1] 1.1963158 1.1132911 1.0263377 0.9535379 0.8941004 0.8456968 0.8079968
##  [8] 0.7769348 0.7477809 0.7206674 0.6936821 0.6696005 0.6504249 0.6352482
## [15] 0.6245073 0.6156291 0.6077434 0.5995917 0.5900692 0.5814417 0.5731230
## [22] 0.5657989 0.5593324 0.5529547 0.5471807 0.5420048 0.5378264 0.5355865
## [29] 0.5337903 0.5322112 0.5313209 0.5316329 0.5308217 0.5284033 0.5260330
## [36] 0.5239888 0.5219611 0.5186369 0.5156809 0.5132071 0.5110224 0.5088307
## [43] 0.5067979 0.5050997 0.5036993 0.5025068 0.5014916 0.5006244 0.4998698
## [50] 0.4992336 0.4987067 0.4982782 0.4979474 0.4977109 0.4975062 0.4973224
## [57] 0.4971599 0.4970075 0.4968896 0.4967804 0.4966901 0.4966060 0.4965311
## [64] 0.4964660 0.4964235 0.4963732 0.4963358 0.4963026
##
## $nzero
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   1   1   1   1   1   1   1   2   2   3   3   3   3   3   3   5   5
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   5   5   5   5   5   5   5   5   5   5   5   6   6   6   7   7   7   7
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67
##   7   7   8   8   8   8   8   8   8   8   8   8   8   8
##
## $name
##                  mse
## "Mean-Squared Error"
##
## $glmnet.fit
##
## Call:  glmnet(x = as.matrix(train_stan[, -9]), y = train_stan$lpsa,      alpha = 1)
##
##       Df    %Dev   Lambda
##  [1,]  0 0.00000 0.878900
##  [2,]  1 0.09126 0.800800
##  [3,]  1 0.16700 0.729700
##  [4,]  1 0.22990 0.664800
##  [5,]  1 0.28220 0.605800
##  [6,]  1 0.32550 0.552000
##  [7,]  1 0.36150 0.502900
##  [8,]  1 0.39140 0.458200
##  [9,]  2 0.42810 0.417500
## [10,]  2 0.45980 0.380400
## [11,]  3 0.48770 0.346600
## [12,]  3 0.51310 0.315900
## [13,]  3 0.53420 0.287800
## [14,]  3 0.55180 0.262200
## [15,]  3 0.56630 0.238900
```

```
## [16,]  3 0.57840 0.217700
## [17,]  5 0.59170 0.198400
## [18,]  5 0.60450 0.180700
## [19,]  5 0.61510 0.164700
## [20,]  5 0.62390 0.150100
## [21,]  5 0.63120 0.136700
## [22,]  5 0.63720 0.124600
## [23,]  5 0.64230 0.113500
## [24,]  5 0.64650 0.103400
## [25,]  5 0.64990 0.094240
## [26,]  5 0.65280 0.085870
## [27,]  5 0.65520 0.078240
## [28,]  5 0.65720 0.071290
## [29,]  5 0.65890 0.064960
## [30,]  6 0.66050 0.059190
## [31,]  6 0.66320 0.053930
## [32,]  6 0.66530 0.049140
## [33,]  7 0.66760 0.044770
## [34,]  7 0.67210 0.040790
## [35,]  7 0.67590 0.037170
## [36,]  7 0.67900 0.033870
## [37,]  7 0.68160 0.030860
## [38,]  7 0.68370 0.028120
## [39,]  7 0.68550 0.025620
## [40,]  7 0.68700 0.023340
## [41,]  7 0.68820 0.021270
## [42,]  7 0.68930 0.019380
## [43,]  7 0.69010 0.017660
## [44,]  7 0.69080 0.016090
## [45,]  7 0.69140 0.014660
## [46,]  7 0.69190 0.013360
## [47,]  7 0.69230 0.012170
## [48,]  7 0.69260 0.011090
## [49,]  7 0.69290 0.010100
## [50,]  7 0.69310 0.009207
## [51,]  7 0.69330 0.008389
## [52,]  7 0.69350 0.007644
## [53,]  7 0.69360 0.006965
## [54,]  7 0.69370 0.006346
## [55,]  7 0.69380 0.005782
## [56,]  7 0.69390 0.005269
## [57,]  8 0.69390 0.004801
## [58,]  8 0.69400 0.004374
## [59,]  8 0.69410 0.003986
## [60,]  8 0.69410 0.003632
## [61,]  8 0.69420 0.003309
## [62,]  8 0.69420 0.003015
## [63,]  8 0.69420 0.002747
## [64,]  8 0.69430 0.002503
## [65,]  8 0.69430 0.002281
## [66,]  8 0.69430 0.002078
## [67,]  8 0.69430 0.001893
## [68,]  8 0.69430 0.001725
## [69,]  8 0.69430 0.001572
```

```
## [70,]  8 0.69430 0.001432
## [71,]  8 0.69430 0.001305
##
## $lambda.min
## [1] 0.007644054
##
## $lambda.1se
## [1] 0.1807428
##
## attr(,"class")
## [1] "cv.glmnet"
```

```
opt_lambda <- lassocv$lambda.min
opt_lambda
```

```
## [1] 0.007644054
```

```
lasso <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 1, lambda.min.ratio = opt_lambda)

lasso <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 1, lambda= opt_lambda)



lasso
```

```
##
## Call:  glmnet(x = as.matrix(train_stan[, -9]), y = train_stan$lpsa,      alpha = 1, lambda = opt_lam
##
##      Df   %Dev   Lambda
## [1,]  7 0.6935 0.007644
```

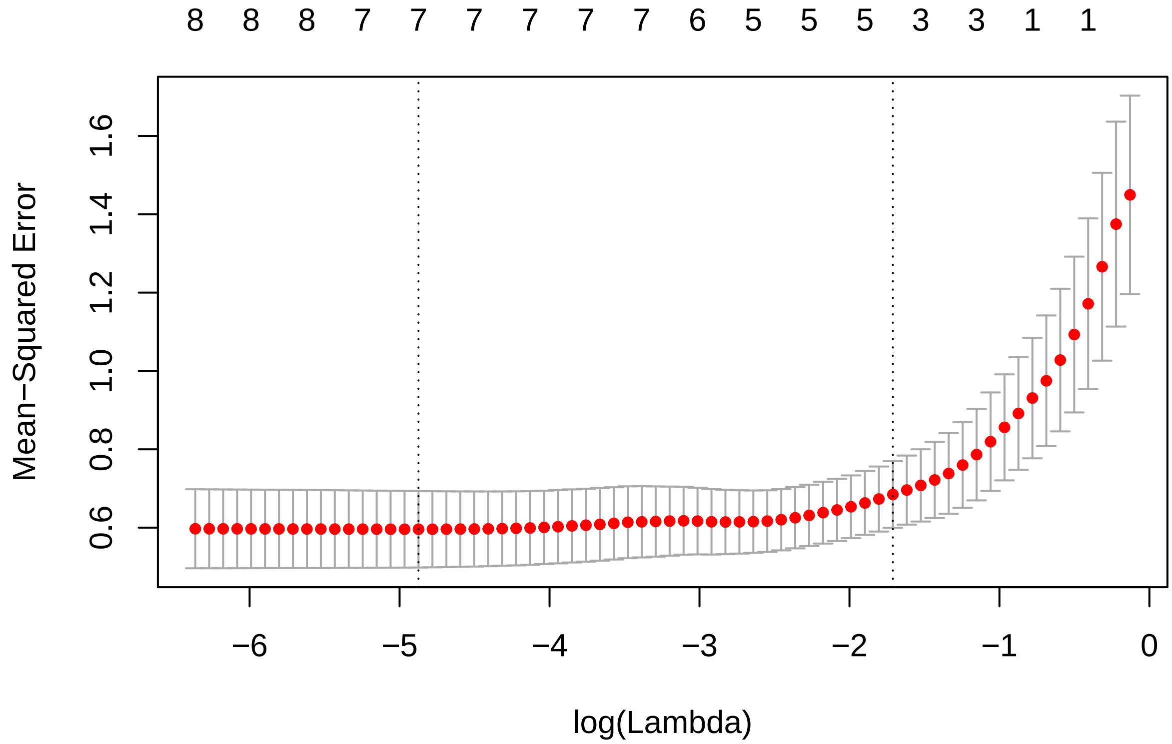
```
coef(lassocv, s = "lambda.min")
```

```
## 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
```

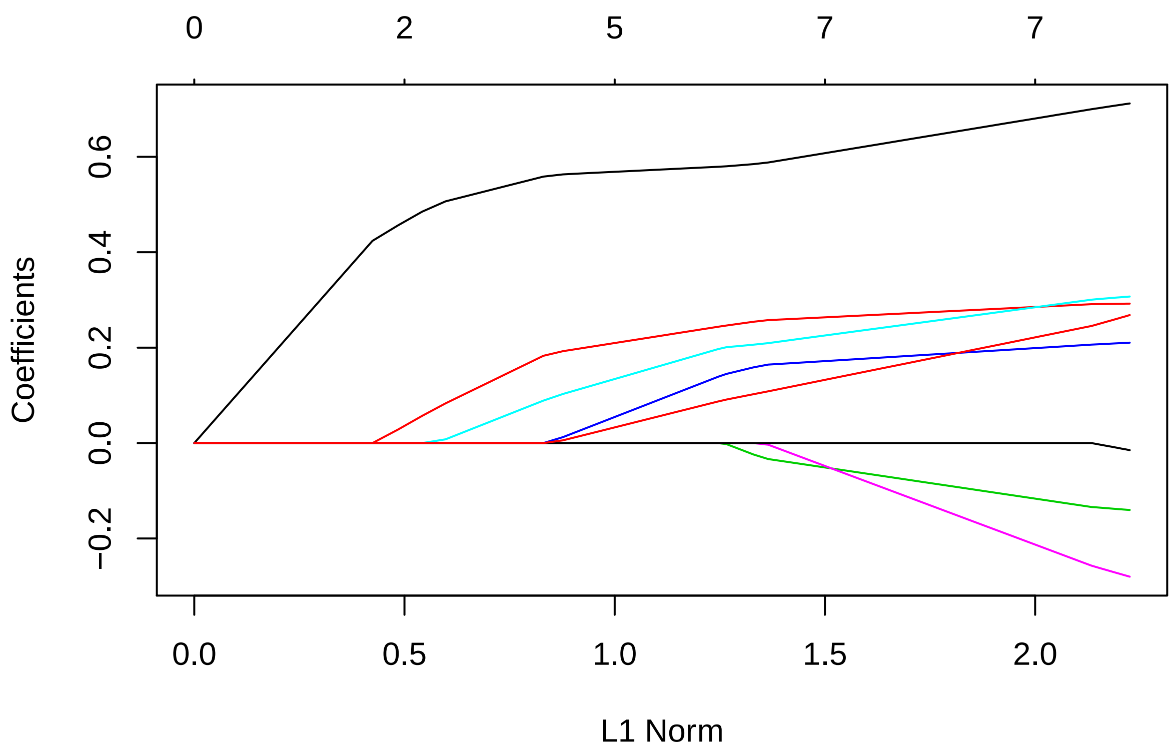
```
coef(lasso, s = "lambda.min")
```

```
## 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
```

```r
plot.cv.glmnet(lassocv)
```




```r
lasso <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 1)


plot.glmnet(lasso)
```



Model Selection (20 pts)

```
subset.coef <- coef(subset, minsubset)

#  c(coef(subset, minsubset), rep(NA,6))

ridge.coef <- (coef(ridgecv, s = ridgecv$lambda.1se))[1:9]
# Q :   s = labmda.min?
# or coef(ridgecv)


lasso.coef <- (coef(lassocv, s = lassocv$lambda.1se))[1:9]




# don't recycle
# Q PCR, PLS intercept coefficient needed


table <- cbind(LS = lsfit$coefficients,
               Best_Subset = c(subset.coef, rep(0, 6)),
              Ridge = ridge.coef , Lasso = lasso.coef, PCR =  coef(pcr_model, intercept = T), PLS = coe

table
```

```
##                      LS Best_Subset        Ridge      Lasso         PCR
## (Intercept)  2.45234509   2.4523451  2.452345085 2.45234509  2.45234509
## lcavol       0.71640701   0.7798589  0.317607465 0.56536414  0.71640701
## lweight      0.29264240   0.3519101  0.207749630 0.19936439  0.29264240
## age         -0.14254963   0.0000000 -0.003245594 0.00000000 -0.14254963
## lbph         0.21200760   0.0000000  0.132737031 0.02917609  0.21200760
## svi          0.30961953   0.0000000  0.192527220 0.11532140  0.30961953
## lcp         -0.28900562   0.0000000  0.067982376 0.00000000 -0.28900562
## gleason     -0.02091352   0.0000000  0.052456497 0.00000000 -0.02091352
## pgg45        0.27734595   0.0000000  0.109563765 0.01647872  0.27734595
##                     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
```

```
y <- test[,9]
X <- as.data.frame(scale(test[,-9]))

LSmse <- mean((predict(lsfit, X) - y)^2)
# why so huge?
LSmse
```

```
## [1] 0.5491941
```

```r
# Q : what's wrong??
#subsetmse <- mean(summary(subset)$rss^2)
#subset coefficients are the same as the OLS regression coefficients with the selected variable
subset_lm <- lm(lpsa ~ lcavol + lweight, data = train_stan)
subsetmse <- mean((predict(subset_lm, X) - y)^2)
subsetmse
```

```
## [1] 0.5483947
```

```r
#yhat <- as.matrix(cbind(1,test[,c(1,2)])) %*% as.matrix(subset_coef)
#  predict(subset_lm, test)  give the same results so it should be correct.
#head(yhat)



#ridgecv <- cv.glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0)
#opt_lambda <- ridgecv$lambda.min

# Q  lambda.min.ratio = opt_lambda
#ridge <- glmnet(as.matrix(train_stan[,-9]), train_stan$lpsa , alpha = 0, lambda.min.ratio = opt_lambda



#ridgemse <- min(ridgecv$cvm)
ridgemse <- mean((predict(ridgecv, as.matrix(X), s = "lambda.min") - y)^2)

ridgemse
```

```
## [1] 0.5171794
```

```r
#lassomse <- min(lassocv$cvm)
lassomse <- mean((predict(lassocv, as.matrix(X), s = "lambda.min") - y)^2)

lassomse
```

```
## [1] 0.5304655
```

```r
#pcrmse <- mean(pcr_model$residuals^2)
pcrmse <- mean((predict(pcr_model, as.matrix(X), s = "lambda.min") - y)^2)
pcrmse
```

```
## [1] 0.5735587
```

```r
#plsmse <- mean(plsr_model$residuals^2)
plsmse <- mean((predict(plsr_model, as.matrix(X), s = "lambda.min") - y)^2)
plsmse
```

```
## [1] 0.5416171
```

```r
Test_Error <- c(LSmse, subsetmse, ridgemse, lassomse, pcrmse, plsmse )

Test_Error
```

```
## [1] 0.5491941 0.5483947 0.5171794 0.5304655 0.5735587 0.5416171
```

```r
table <- rbind(table, Test_Error)

table
```

```
##                      LS Best_Subset        Ridge      Lasso         PCR
## (Intercept)  2.45234509   2.4523451  2.452345085 2.45234509  2.45234509
## lcavol       0.71640701   0.7798589  0.317607465 0.56536414  0.71640701
## lweight      0.29264240   0.3519101  0.207749630 0.19936439  0.29264240
## age         -0.14254963   0.0000000 -0.003245594 0.00000000 -0.14254963
## lbph         0.21200760   0.0000000  0.132737031 0.02917609  0.21200760
## svi          0.30961953   0.0000000  0.192527220 0.11532140  0.30961953
## lcp         -0.28900562   0.0000000  0.067982376 0.00000000 -0.28900562
## gleason     -0.02091352   0.0000000  0.052456497 0.00000000 -0.02091352
## pgg45        0.27734595   0.0000000  0.109563765 0.01647872  0.27734595
## Test_Error   0.54919414   0.5483947  0.517179403 0.53046551  0.57355874
##                     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
## Test_Error   0.54161713
```

```
cat("Best model is ", colnames(table)[which.min(Test_Error)])
```

```
## Best model is  Ridge
```