HW2

Shiyu Wang

February 8, 2017

Question 1

- Shiyu Wang - Computer Science - 1st semester
- Xingyan Deng - Computer Science - 1st semester
- Qian Shen - Computer science - 2nd semester

Question 2

"Classification and prediction of clinical Alzheimer's diagnosis based on plasma signaling proteins" talks about using a molecular test to classify and predict Alzheimer's disease. According to the paper, Alzheimer's disease affects one in eight people after reaching 65 years of age. And it usually requires a set of psychological test, image and exclusion of neurological disorders. For these reasons, this paper introduces molecular biomarker in blood plasma and MCI to classify and identify Alzheimer's as early as possible.

They collected totally 259 plasma samples with 120 known signaling proteins are separated into training and test set for supervised classification. The data were analyzed with a shrunken centroid algorithm called predictive analysis of microarrays. It identified 18 predictors that may be related to the Alzheimer's and nondemented control. Also, unsupervised clustering based on predictive signaling proteins lead to good separation. The output indicates that a highly specific plasma biomarker phenotype can characterize Alzheimer's disease years before a clinical diagnosis can be made. Moreover, a series of investigation shows the biological relevance of the 18 predictors for Alzheimer's disease.

Positive Aspects:

The first positive aspect caught my attention is the reason to do this research. It provides the current approach and digital data of predict and measure Alzheimer's disease, which are inefficient and inaccurate. However, one in eight people by the time are meeting this disease. Thus, this paper introduces a new method to classify and predict Alzheimer's disease by analyzing the plasma to test the 18 predictive signaling proteins.

Another positive aspect is the process of the experiment. It firstly gave a bold hypothesis that the pathological process leading to Alzheimer's would cause characteristic changes in the concentrations of signaling. After that, it started to try to narrow the factors based on experiment of 259 plasma samples. By a series of analysis, including significance analysis of microarrays, predictive analysis of microarrays and unsupervised clustering, it indicates there are 18 predictors of signaling proteins can be used to classify and predict Alzheimer's disease and NDC. Moreover, it doesn't stop after the result of an experiment, but also tries

to understand the potential biological relevance of the 18 signaling proteins that characterize Alzheimer's. During this process, they used several functional annotation tools and PubMed to guarantee the accuracy of the analysis.

Negative Aspects:

From my point of view, this research and experiment is great and interesting. If we have to say some negative aspects, one could be the hypothesis. According to the paper, they hypothesized the changes in the concentrations of signaling proteins in blood and generating of a detectable disease-specific molecular phenotype just because the brain controls many body functions via the release of signaling proteins. This hypothesis is not rigorous and too arbitrary. I agreed we need to identify Alzheimer's as early as possible, but it is also important to make sure the prediction is accurate and responsible for the potential patients.

Another negative aspect can be the dataset. There are only 259 archived plasma samples used to experiment. And we still have to separate them into training and test set. I believe this number is not enough for a biological experiment. However, they just simply announced they found some secreted signaling proteins differ considerably between subjects with Alzheimer's disease and NDC subjects from the results.

Possible Extensions: If I have this opportunity, I'd like to extend the sample size of their experiment to verify their result is accurate. In the paper, they have already found these 18 signaling proteins, so we can find more archived plasma or volunteers with alternative methods to test and verify the result is accurate and can be used to predict Alzheimer's.

Question 3

Number: 024

URL: http://cs229.stanford.edu/proj2015/024_report.pdf

The twenty-fourth project is "Matching Handwriting with Its Author" This project is interesting, special and useful in the handwritten digit aspect. As I know, many projects focused on handwritten digit recognition, but matching handwriting with its author is also very important in reality. As the report stated, it can be widely used for bank check authentication, forensic or police investigation and document verification.

For the experiment, they chose to use an app called INKredible to manually collect handwriting samples on the tablet screen using stylus/finger. 300 handwriting samples were collected among 3 different authors and scaled to four different resolutions (4X4, 8X8, 16X16 and 64X64). After that, 3 preprocessing techniques are applied, including

- Conversion of image to B/W – each pixel will either be '1' (for white) or '0' (for black).
- Handwriting size normalization.

- Background removal After all these preparations, they start applying Naïve Bayes algorithm and SVM to train and test the samples. By comparing the results, SVM performs better in 8X8 and 64X64 samples with large training samples, otherwise Naïve Bayes is more accurate.

Positive Aspects:

The first positive aspect I'd like to mention is the "sample generating" of this project. They prudentially consider the problem of this project before the experiment that it is very time consuming to manually create handwritten samples. Therefore, they researched and found the way to automatically generate many artificial handwriting samples based on some original authentic samples written by human. Although the automatically generated handwriting may be not always natural-looking, it is worth to think and try to experiment.

Another interesting point is the research finds out the size-normalization and centering can improve SVM algorithm's accuracy, but hinder the Naïve Bayes. The report doesn't mention the reason for them to compare normalization, but we can see this is an important factor from the experiment diagram.

Negative Aspects:

After reading the project, I feel it is too simple. I understand it is not easy to implement these algorithms, but implementing them doesn't solve real world problems. We could extend the project to research the reason that normalization is important for these algorithms. And add more implemented algorithms to test the hypothesis about it.

Another negative part is the dataset size. I understand it is time consuming to collect them, but comparing with 60000 samples in MNIST, 300 samples are really not enough to guarantee the accuracy of the result. According to the Fig. 11 of the report, the best algorithm at different number of raining samples and image resolutions is very unstable. If we increase the training samples and test samples, the result probably becomes totally different.

Possible Extensions: As I mentioned in "Negative Aspects", the first thing I'd like to extend is to find the reason that normalization affects algorithms so much. It will be another interesting research in this project. Moreover, I strongly suggest to increase handwriting samples for this project. Beside manually collect samples, it will be helpful if we can find a better way to generate more samples to make the result more accurate.

Question 4

Setting up

```
# Set up work directory
#-----------------------------------------------------------------
#setwd('D:/Dropbox/CS6140/hw2')
setwd('C:/Users/Shiyu/Desktop')
```

```r
library(leaps)
library(MASS)
library(lars)

## Loaded lars 1.2
```

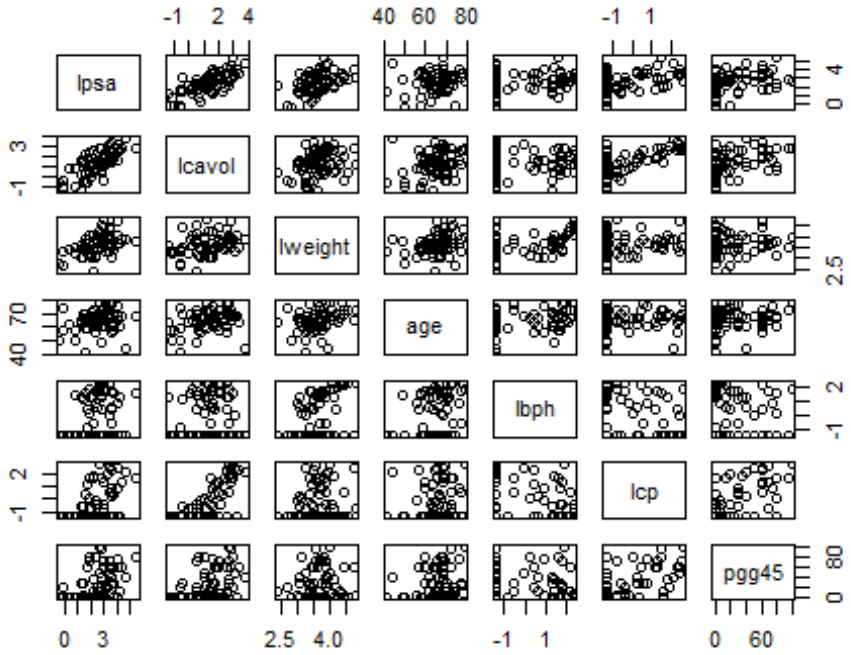
Part A

```r
# Read data
#---------------------------------------------------------------------
data <- read.table('prostate.txt', header=TRUE, row.names=1, sep='\t',
stringsAsFactors = FALSE)

# factor the two categorical variables
# data$svi <- factor(data$svi)
# data$gleason <- factor(data$gleason)

# Select the training sets and validation sets
#---------------------------------------------------------------------
train <- data[data$train, -10]
valid <- data[!data$train, -10]
```

Part B

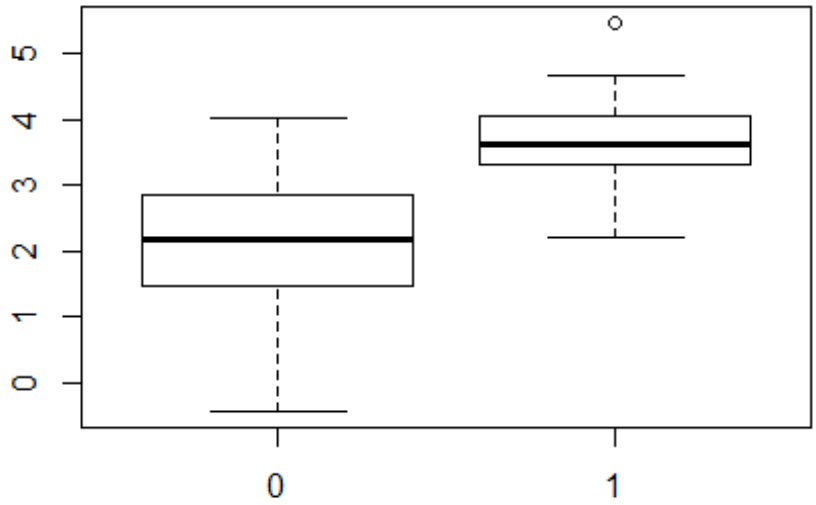
```r
#### one-variable summary statistics ###
summary(train)

##      lcavol           lweight           age              lbph
##  Min.   :-1.3471   Min.   :2.375   Min.   :41.00   Min.   :-1.38629
##  1st Qu.: 0.4883   1st Qu.:3.330   1st Qu.:61.00   1st Qu.:-1.38629
##  Median : 1.4679   Median :3.599   Median :65.00   Median :-0.05129
##  Mean   : 1.3135   Mean   :3.626   Mean   :64.75   Mean   : 0.07144
##  3rd Qu.: 2.3491   3rd Qu.:3.884   3rd Qu.:69.00   3rd Qu.: 1.54751
##  Max.   : 3.8210   Max.   :4.780   Max.   :79.00   Max.   : 2.32630
##       svi              lcp              gleason          pgg45
##  Min.   :0.0000   Min.   :-1.3863   Min.   :6.000   Min.   :  0.00
##  1st Qu.:0.0000   1st Qu.:-1.3863   1st Qu.:6.000   1st Qu.:  0.00
##  Median :0.0000   Median :-0.7985   Median :7.000   Median : 15.00
##  Mean   :0.2239   Mean   :-0.2142   Mean   :6.731   Mean   : 26.27
##  3rd Qu.:0.0000   3rd Qu.: 0.9948   3rd Qu.:7.000   3rd Qu.: 50.00
##  Max.   :1.0000   Max.   : 2.6568   Max.   :9.000   Max.   :100.00
##       lpsa
##  Min.   :-0.4308
##  1st Qu.: 1.6673
##  Median : 2.5688
##  Mean   : 2.4523
##  3rd Qu.: 3.3652
##  Max.   : 5.4775
```

```
it analyzed the general data of each column, including man, min, mean, etc.

### two-variable summary statistics ###
pairs(lpsa ~ lcavol + lweight + age + lbph + lcp + pgg45, data = train)
```

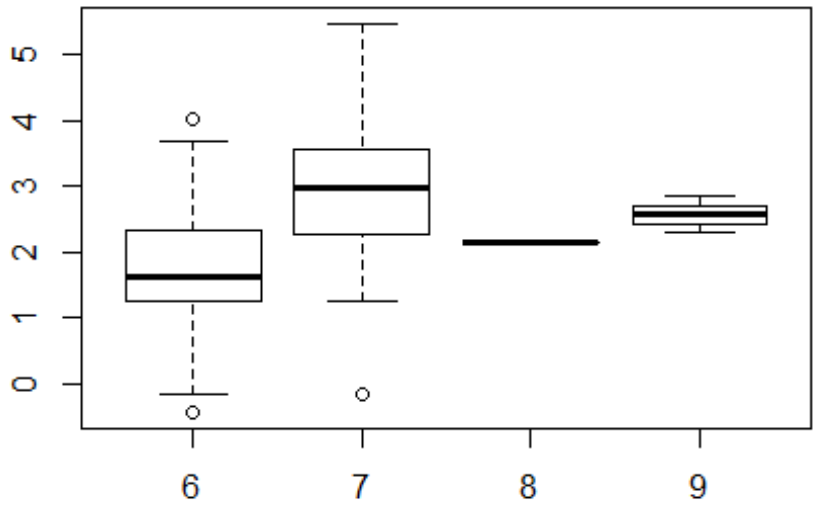



```
From the chart
By simply compare lpsa and oher variables, it tells that:
- lspa increases with lcavol
- Lcp and lweight also increse with lspa, but it is not quite strong
- A even less strong but possible positive association is with lbph
Generally, This suggests that not all variabels may be needed
since some are correlated and may provide similar information about ;psa.

### Correlation coefficients ###
cor(data[,c(1,2,3,4,6,8,9)])

##            lcavol   lweight       age         lbph          lcp      pgg45
## lcavol  1.0000000 0.2805214 0.2249999  0.027349703  0.675310484 0.43365225
## lweight 0.2805214 1.0000000 0.3479691  0.442264399  0.164537142 0.10735379
## age     0.2249999 0.3479691 1.0000000  0.350185896  0.127667752 0.27611245
## lbph    0.0273497 0.4422644 0.3501859  1.000000000 -0.006999431 0.07846002
## lcp     0.6753105 0.1645371 0.1276678 -0.006999431  1.000000000 0.63152825
## pgg45   0.4336522 0.1073538 0.2761124  0.078460018  0.631528246 1.00000000
## lpsa    0.7344603 0.4333194 0.1695928  0.179809404  0.548813175 0.42231586
##              lpsa
## lcavol  0.7344603
## lweight 0.4333194
```

```
## age     0.1695928
## lbph    0.1798094
## lcp     0.5488132
## pgg45   0.4223159
## lpsa    1.0000000

The result
- lcavol has the highest correlation coefficients (0.7344603) suggesting a
moderate positive linear association
- age and lbph have very low correlations (0.1695928, 0.1798094)
- lweight, lcp and pgg45 have small positive correlations (0.4333194,
0.5488132, 0.4223159)
- This suggests that the most important linear variables are lcavol with lcp,
lweight and pgg45 possibly
- being important as well

### Categorical variables ###
boxplot(lpsa ~ svi, data = train)
```



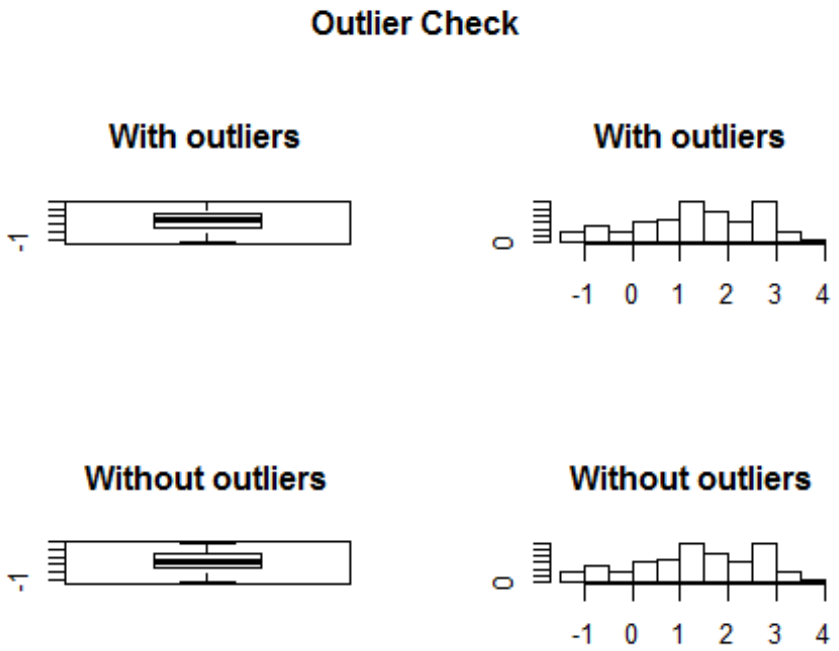

```
boxplot(lpsa ~ gleason, data = train)
```



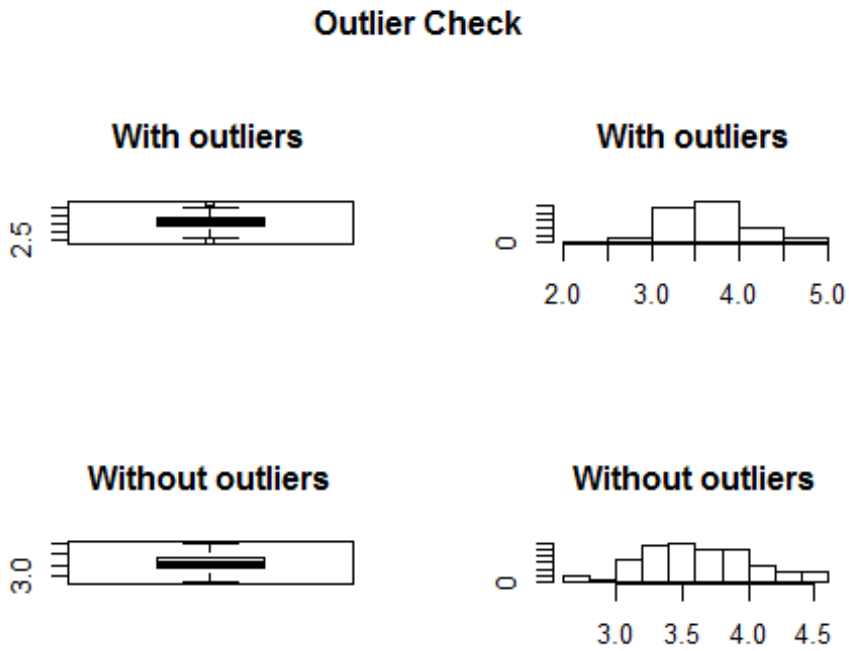

```
From the chart
Both of these boxplots show lpsa varies significantly over the levels of svi
and gleason.
Therefore We need to consider including these variabels in our model.

### Missing values ###
is.na(train)

##    lcavol lweight   age  lbph   svi   lcp gleason pgg45  lpsa
## 1   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 2   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 3   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 4   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 5   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 6   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 8   FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 11  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 12  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 13  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 14  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 16  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 17  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 18  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 19  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 20  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 21  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 23  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
```

```
## 24  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 27  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 29  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 30  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 31  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 33  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 35  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 37  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 38  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 39  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 40  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 41  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 43  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 45  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 46  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 47  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 51  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 52  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 56  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 58  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 59  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 60  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 61  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 63  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 67  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 68  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 69  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 70  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 71  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 72  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 75  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 76  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 77  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 78  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 79  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 81  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 82  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 83  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 85  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 86  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 87  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 88  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 89  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 90  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 91  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 92  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 93  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 94  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
## 96  FALSE   FALSE FALSE FALSE FALSE FALSE   FALSE FALSE FALSE
```

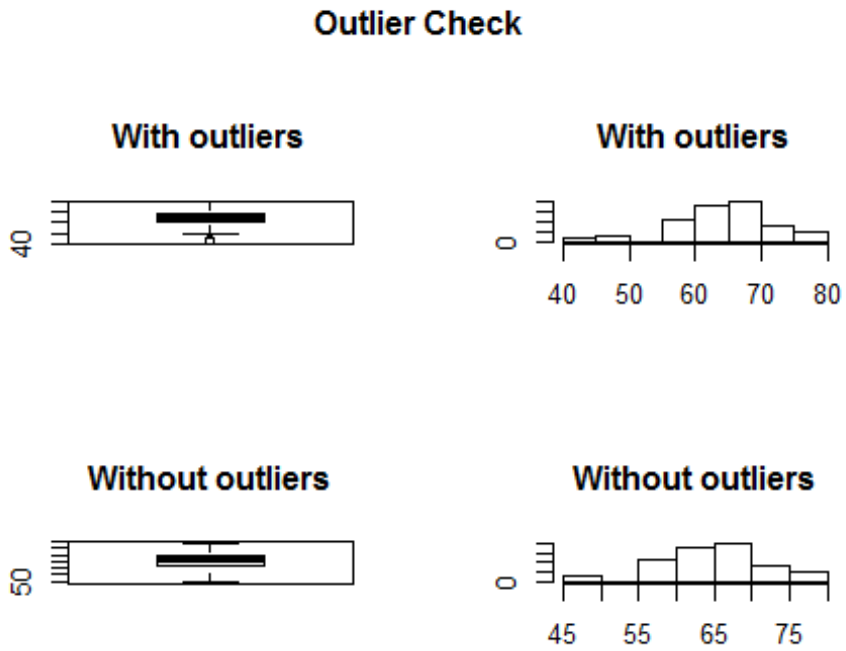
```
The result shows no missing values in training set.
After dealing with outliers, we will have some missing values.
So if we need to calculate mean or other group vale, we need to exclude NA
value

### Outliers
# To detect the outliers I use the command boxplot.stats()$out
# which use the Tukey's method to identify the outliers ranged above and
below the 1.5*IQR.
outlier <- function(dt, var) {
    var_name <- eval(substitute(var),eval(dt))
    na1 <- sum(is.na(var_name))
    m1 <- mean(var_name, na.rm = T)
    par(mfrow=c(2, 2), oma=c(0,0,3,0))
    boxplot(var_name, main="With outliers")
    hist(var_name, main="With outliers", xlab=NA, ylab=NA)
    outlier <- boxplot.stats(var_name)$out
    mo <- mean(outlier)
    var_name <- ifelse(var_name %in% outlier, NA, var_name)
    boxplot(var_name, main="Without outliers")
    hist(var_name, main="Without outliers", xlab=NA, ylab=NA)
    title("Outlier Check", outer=TRUE)
    na2 <- sum(is.na(var_name))
    cat("Outliers identified:", na2 - na1, "\n")
    cat("Propotion (%) of outliers:", round((na2 - na1) /
sum(!is.na(var_name))*100, 1), "\n")
    cat("Mean of the outliers:", round(mo, 2), "\n")
    m2 <- mean(var_name, na.rm = T)
    cat("Mean without removing outliers:", round(m1, 2), "\n")
    cat("Mean if we remove outliers:", round(m2, 2), "\n")
    # dt[as.character(substitute(var))] <- invisible(var_name)
    # assign(as.character(as.list(match.call())$dt), dt, envir = .GlobalEnv)
    # cat("Outliers successfully removed", "\n")
    # return(invisible(dt))
}

outlier(train, lcavol)
```



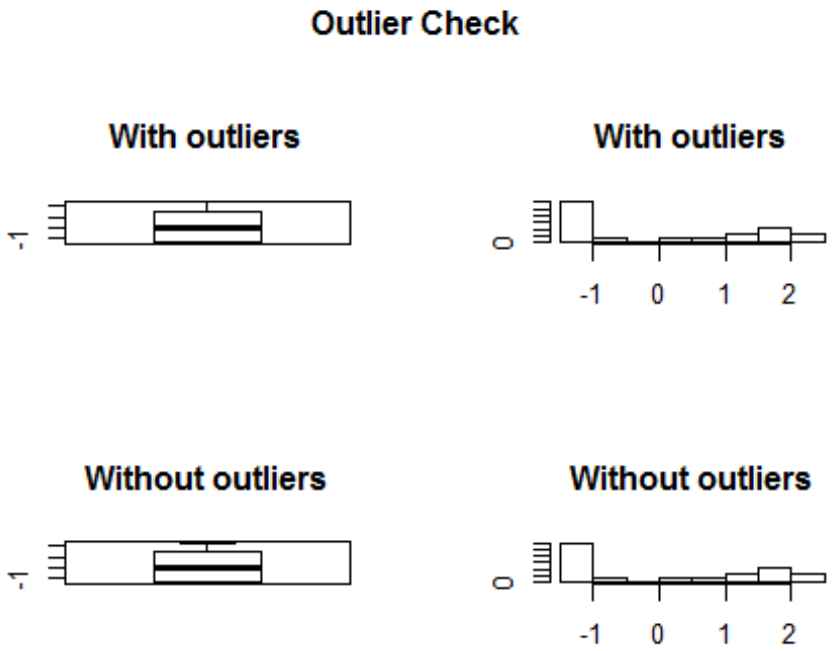

```
## Outliers identified: 0
## Propotion (%) of outliers: 0
## Mean of the outliers: NaN
## Mean without removing outliers: 1.31
## Mean if we remove outliers: 1.31

outlier(train, lweight)
```



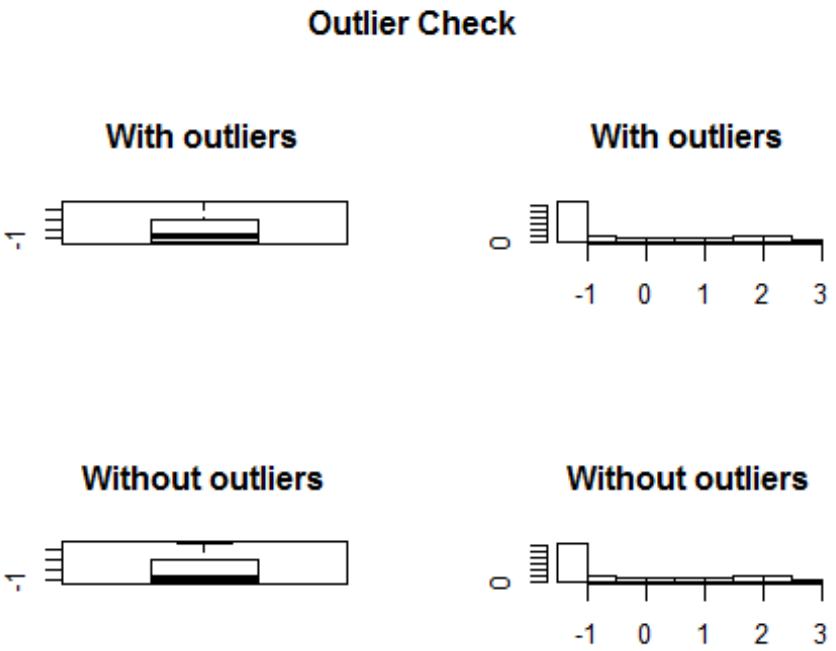

```
## Outliers identified: 3
## Propotion (%) of outliers: 4.7
## Mean of the outliers: 3.96
## Mean without removing outliers: 3.63
## Mean if we remove outliers: 3.61

outlier(train, age)
```



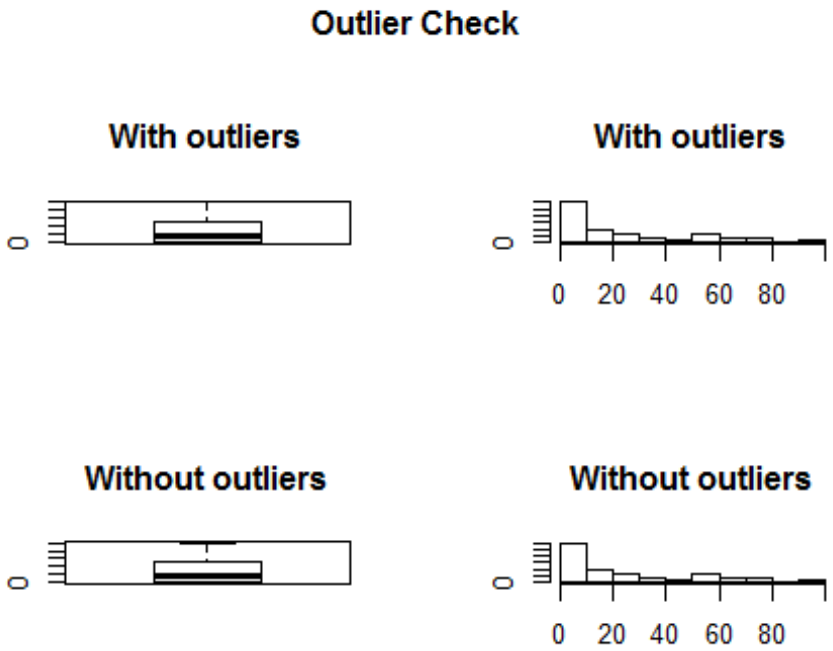

```
## Outliers identified: 2
## Propotion (%) of outliers: 3.1
## Mean of the outliers: 42.5
## Mean without removing outliers: 64.75
## Mean if we remove outliers: 65.43

outlier(train, lbph)
```



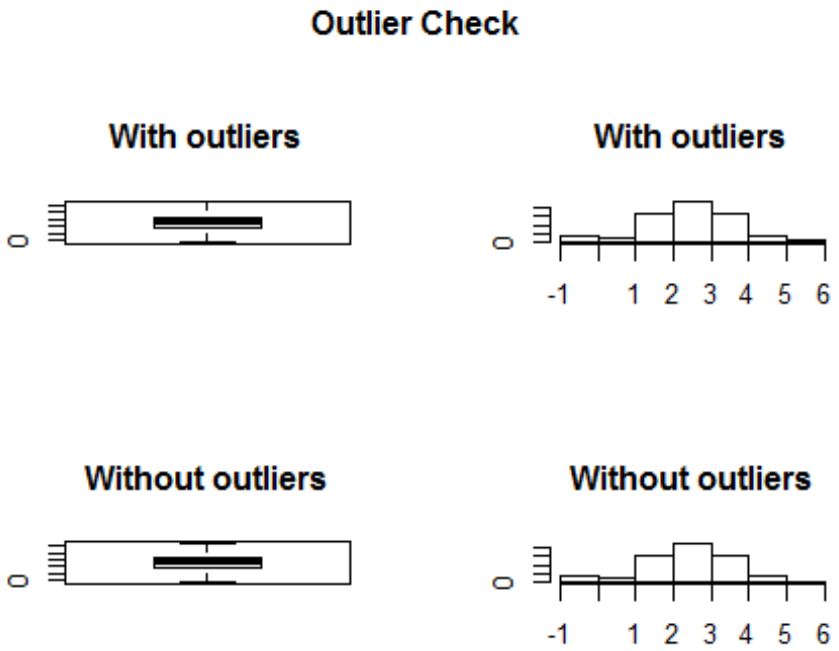

```
## Outliers identified: 0
## Propotion (%) of outliers: 0
## Mean of the outliers: NaN
## Mean without removing outliers: 0.07
## Mean if we remove outliers: 0.07

outlier(train, lcp)
```




```
## Outliers identified: 0
## Propotion (%) of outliers: 0
## Mean of the outliers: NaN
## Mean without removing outliers: -0.21
## Mean if we remove outliers: -0.21

outlier(train, pgg45)
```




```
## Outliers identified: 0
## Propotion (%) of outliers: 0
## Mean of the outliers: NaN
## Mean without removing outliers: 26.27
## Mean if we remove outliers: 26.27

outlier(train, lpsa)
```




```
## Outliers identified: 0
## Propotion (%) of outliers: 0
## Mean of the outliers: NaN
## Mean without removing outliers: 2.45
## Mean if we remove outliers: 2.45
```

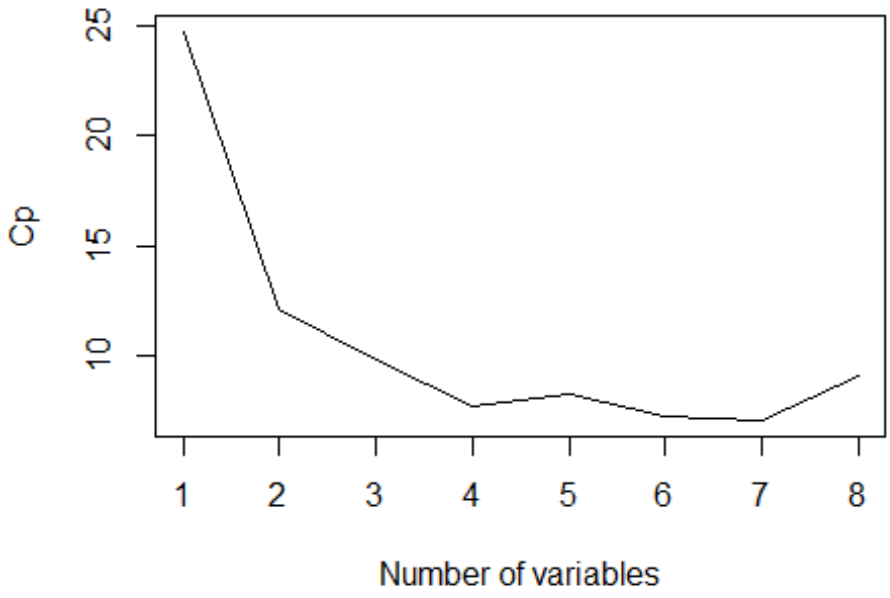
Part D

```
fit2 <- regsubsets(lpsa ~ ., data = train, intercept = TRUE, method =
"exhaustive")
regs <-summary(fit2)
regs

## Subset selection object
## Call: regsubsets.formula(lpsa ~ ., data = train, intercept = TRUE,
##     method = "exhaustive")
## 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
```

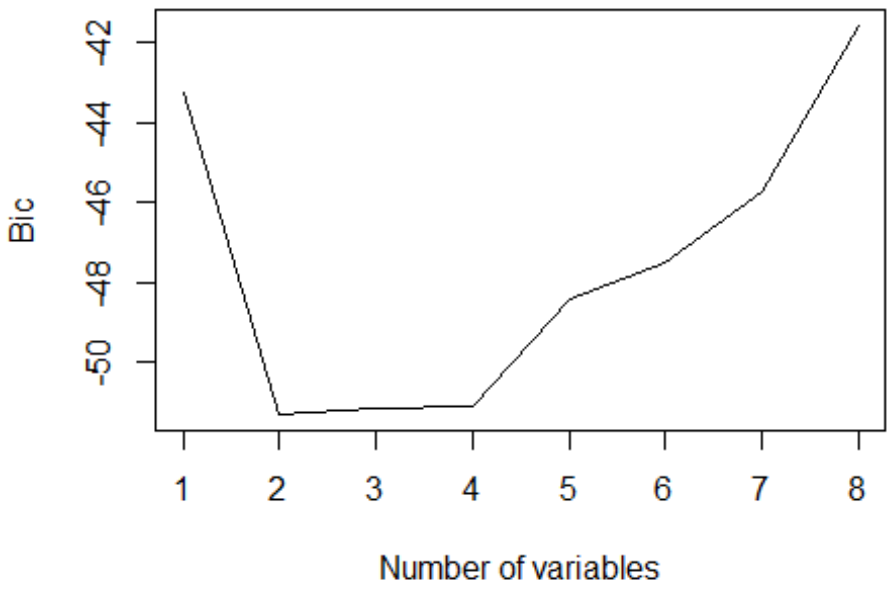
```
##          lcavol lweight age lbph svi lcp gleason pgg45
## 1  ( 1 ) "*"    " "     " " " "  " " " " " " "     " "
## 2  ( 1 ) "*"    "*"     " " " "  " " " " " " "     " "
## 3  ( 1 ) "*"    "*"     " " " "  "*" " " " " "     " "
## 4  ( 1 ) "*"    "*"     " " "*"  "*" " " " " "     " "
## 5  ( 1 ) "*"    "*"     " " "*"  "*" " " " " "     "*"
## 6  ( 1 ) "*"    "*"     " " "*"  "*" "*" " " "     "*"
## 7  ( 1 ) "*"    "*"     "*" "*"  "*" "*" " " "     "*"
## 8  ( 1 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

This summary suggests lcavol + lweight

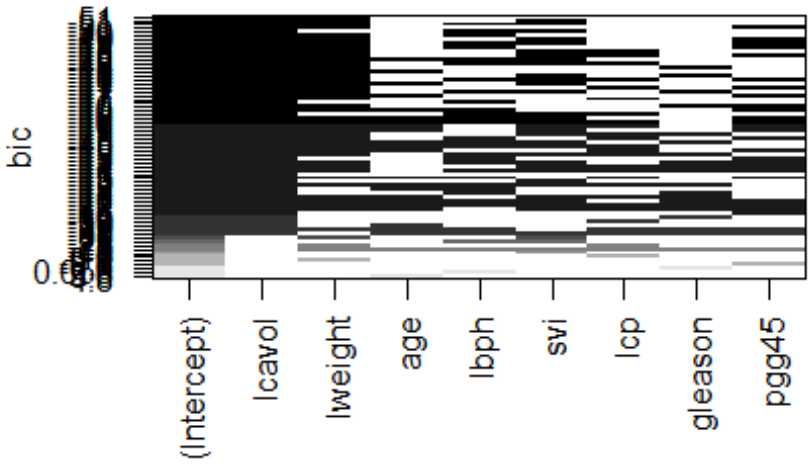
```
plot(regs$cp, xlab="Number of variables", ylab='Cp', type="l")
```




```
plot(regs$bic, xlab="Number of variables", ylab='Bic', type="l")
```



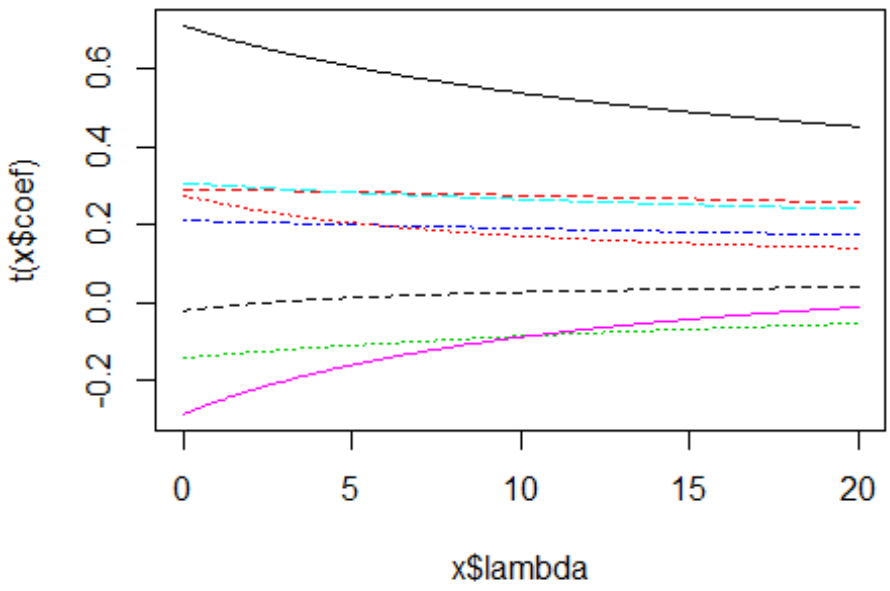

```
Cp and BIC are the criterion for model selection among finite set of models,
the smaller the value, the better the model.
BIC penalizes more than Cp regarding n.
Therefore, 2 variables may be a good choice

leaps.plot = regsubsets(lpsa~.,data=train, nbest=10)
plot(leaps.plot,scale="bic")
```



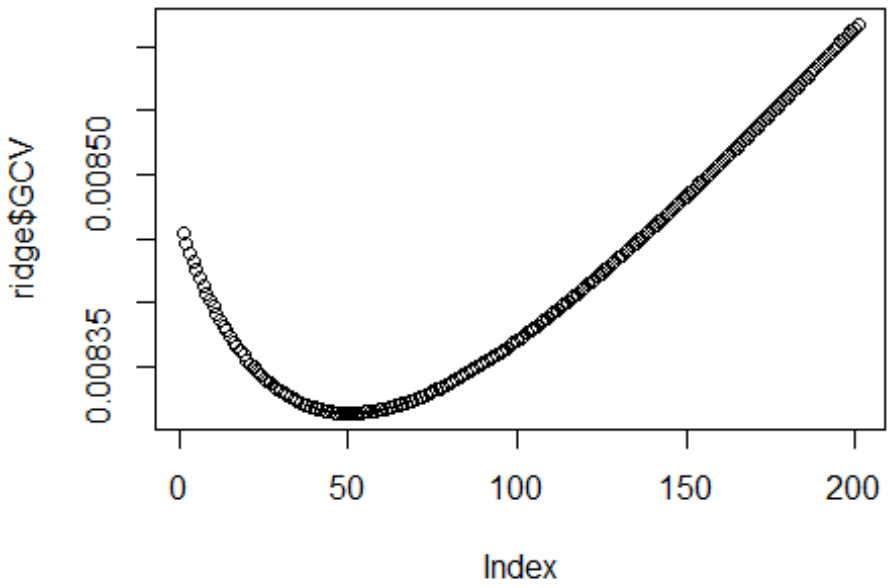

```
this chart also suggests lcavol + lweight

coef(fit2, which.min(regs$bic))

## (Intercept)      lcavol     lweight
##  -1.0494396   0.6276074   0.7383751
```

Part E

```
#####################################
## Ridge regression
#####################################
ridge <- lm.ridge(lpsa ~ .,data=train, lambda = seq(0,20,0.1))
plot(ridge)
```



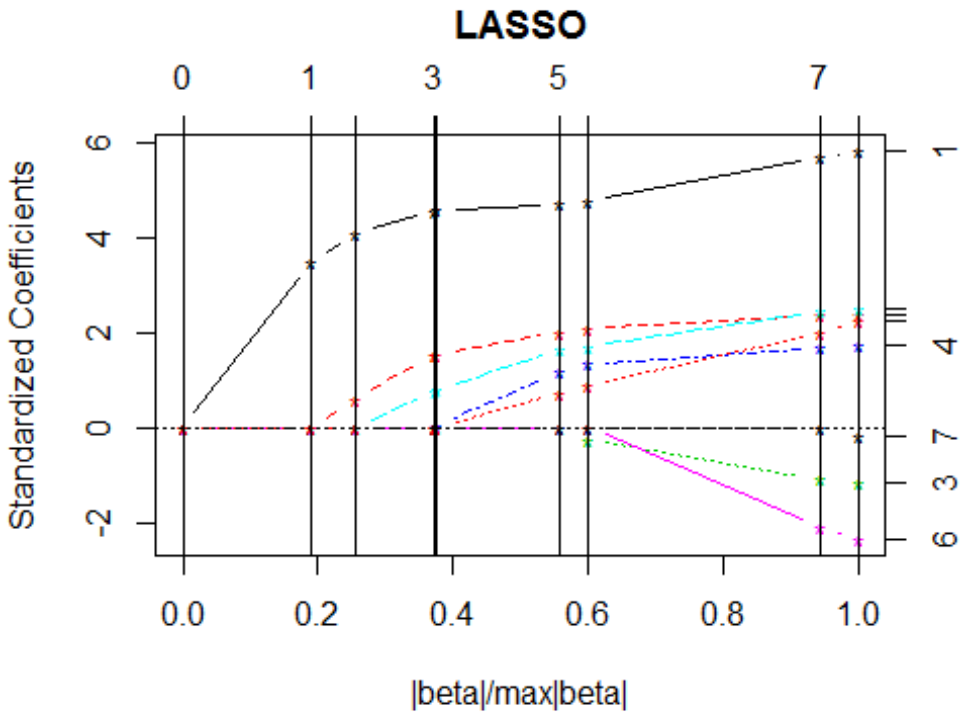

```r
# select parameter by minimum GCV
plot(ridge$GCV)
```



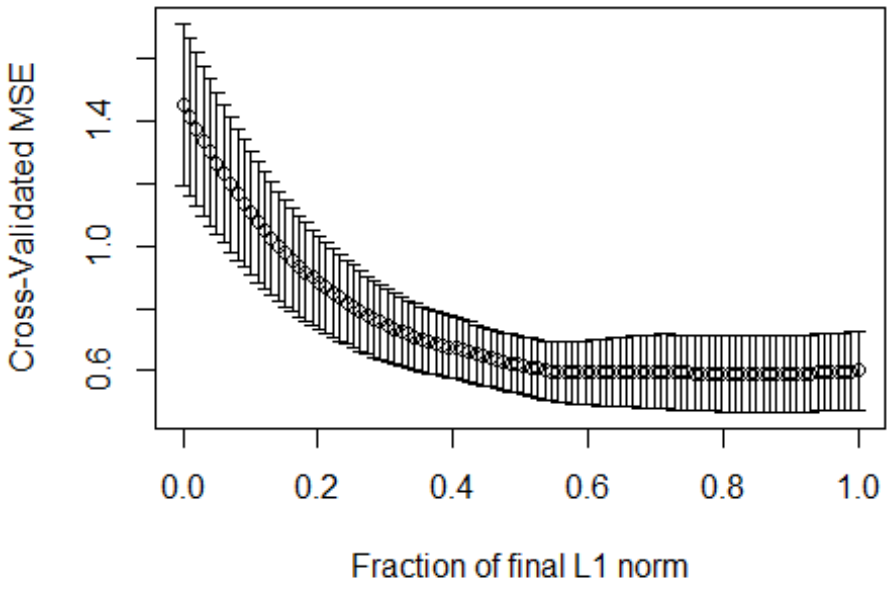
```
ridge$coef[,which.min(ridge$GCV)]
```

```
##      lcavol     lweight         age        lbph         svi         lcp
##  0.60774607  0.28434026 -0.11036319  0.20050020  0.28330007 -0.16236691
##     gleason       pgg45
##  0.01204058  0.20593909
```

```
#####################################
## Lasso
#####################################
lasso <- lars(as.matrix(train[,c(1:8)]),train$lpsa)
plot(lasso)
```




```
# Cross-validation
r <- cv.lars(as.matrix(train[,c(1:8)]), train$lpsa)
```




```
# bestfraction <- r$fraction[which.min(r$cv)]
bestfraction <- r$index[which.min(r$cv)]
bestfraction
```

```
## [1] 0.8484848
```

```
# Observe coefficients
coef.lasso <-
predict(lasso,as.matrix(valid[,c(1:8)]),s=bestfraction,type="coefficient",mod
e="fraction")
coef.lasso
```

```
## $s
## [1] 0.8484848
##
## $fraction
## [1] 0.8484848
##
## $mode
## [1] "fraction"
##
## $coefficients
##       lcavol      lweight          age         lbph          svi
##  0.537797527  0.591046233 -0.014093496  0.132921994  0.654627967
##          lcp      gleason        pgg45
## -0.132637724  0.000000000  0.007065319
```

Part F

```
####################################
## Best subset selection Prediction
####################################
bestsubset <- lm(lpsa ~ .,data= train[,c(1,2,9)])
summary(bestsubset)

##
## Call:
## lm(formula = lpsa ~ ., data = train[, c(1, 2, 9)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.58852 -0.44174  0.01304  0.52613  1.93127
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.04944    0.72904  -1.439 0.154885
## lcavol       0.62761    0.07906   7.938 4.14e-11 ***
## lweight      0.73838    0.20613   3.582 0.000658 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7613 on 64 degrees of freedom
## Multiple R-squared:  0.6148, Adjusted R-squared:  0.6027
## F-statistic: 51.06 on 2 and 64 DF,  p-value: 5.54e-14

pred.bestsubset <- predict(bestsubset,valid[,c(1,2,9)])
summary((pred.bestsubset - valid$lpsa)^2)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000024 0.020600 0.176100 0.492500 0.531100 3.460000

####################################
## Ridge regression Prediction
####################################
pred.ridge = scale(valid[,c(1:8)],center = F)%*%
ridge$coef[,which.min(ridge$GCV)] + ridge$ym
summary((pred.ridge - valid$lpsa)^2)

##        V1
##  Min.   :0.006074
##  1st Qu.:0.395358
##  Median :1.116145
##  Mean   :1.359318
##  3rd Qu.:1.926647
##  Max.   :5.505241
```

```
####################################
## Lasso Prediction
####################################
# Prediction
pred.lasso <-
predict(lasso,as.matrix(valid[,c(1:8)]),s=bestfraction,type="fit",mode="fract
ion")$fit
summary((pred.lasso - valid$lpsa)^2)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000429 0.040110 0.144600 0.489100 0.451500 3.786000

Compare with these models, the Ridge regression has the strongest prediction
power. mean = 1.359318
```

Part G

```
ridge$coef[,which.min(ridge$GCV)]

##      lcavol     lweight         age        lbph         svi         lcp
##  0.60774607  0.28434026 -0.11036319  0.20050020  0.28330007 -0.16236691
##     gleason       pgg45
##  0.01204058  0.20593909
```

Based on this model, we can say that:

- One unit increase in lcavol will result in an expected increase of 0.61 units in lpsa, holding all other variables constant.
- One unit increase in lweight will result in an expected increase of 0.28 units in lpsa, holding all other variables constant.
- One unit increase in age will result in an expected increase of -0.11 units in lpsa, holding all other variables constant.
- One unit increase in lbph will result in an expected increase of 0.20 units in lpsa, holding all other variables constant.
- One unit increase in lcp will result in an expected increase of -0.16 units in lpsa, holding all other variables constant.
- Svi is a binary variable with a value of one of either one or zero.