HW2_Solutions

Sara Taheri

2/7/2017

Problem 2

Classification and Prediction of Clinical Alzheimer's diagnosis based on plasma signaling proteins

The paper discusses the diagnosis of Alzheimer's disease from molecular measurements, before symptoms appear. The authors hypothesized that the pathological process which leads to this disease, would cause charasteristic changes in the concentration of signaling proteins in the blood that can makes the disease to be detectable. To prove it, first they collected 259 plasma samples (i.e., observations) from people with presymptomatic to late-stage Alzheimer's disease and measured the abundance of 120 known signaling proteins (i.e., features) in these samples.

The authors divided the samples into two equal groups of trainig set and test set. The training set was used to predict discovery and supervised classification. The test set was used to test the model on blinded samples. The samples from the test set were not used for any part of the predictor discovery process.

The authors applied statistical hypothesis testing step (called SAM) on the training data to reduce the features to 19 proteins with highly significant differences in expression between Alzheimer's disease (AD) and nondemented control (NDC) samples. Then they applied an unsupervised clustring algorithm to the training set based on the similarity in abundance of that 19 proteins and the algorithm returned two clusters that one of them contains mostly Alzheimers cases and the other contains mostly NDC subjects. Finally, the authors used a supervised classification algorithm (called PAM) to classify the training set with different number of predictors. They used 10-fold cross validation to identify an optimal number of predictive proteins.

It is obvious that the classification error of the training set is smaller than when we use cross-validation. By comparing the results of percentage of error in applying PAM to training and cross-validation (figure 1-c) they identified 18 predictors to have the best results and minimum error among other number of predictors. With this predictor, they were able to predict AD and NDC samples with 95% positive agreement and 83% negative agreement. Then the authors applied their method on the test sets which classified them with a very high accuracy (90% positive agreement and 88% negative agreement). In addition, they applied the method to the combination of training and test set which again yeild to a good separation of all Alzheimer's samples and non-Alzheimer's sample.

The positive aspects of their work is that they separate training and validation set. They analyzed their result on the validation set 10 times (because they used 10-fold cross validation). They also select part of their data as test sets and checked the quality of their method on test set which was never used to create the model which is helpful. The other positive aspect of their research is that they waited 2-6 years and saw the results of their predictions on the patiensts who participated in their study and they found out that their predictions were very close to what happened in the future.This proves that a highly specific plasma biomarker phenotype can characterize AD years before a clinical diagnosis can be made.

One of the negative aspects of this work is that the sample size is relatively small to make general conclusions from. The approach can be extended by collecting more data to build a better model. We can use other methods such as deep learning to extend the work.

An integrated approach to prognosis using protein microarrays and nonparametric methods

The goal of this manuscript is to develp a model that predicts the probability of a patient dying within first 15 weeks of initiating the treatment (which is the kidney dialysis). They collected their data from 208 patients who died in this period, and 260 patients who survived for at least 15 weeks. They found that standard data mining methods such as hierarchial clustering, k-means clustering, nearest neighbor methods, principle component analysis, decision trees and adaptive boosting all failed to distinguish those who died from those who survived. The authors decided to use logistic regression methods to build their method because their goal was to develop a continous predictor of early mortality and not classifying samples. They started by the best subset selection method (exhaustive search) to select the best variables for use in logistic regression. Exhaustive search chose four clinical variables (age, diastolic blood pressure, serum albumin, method of vascular access) and three molecular markers (Ang, IL-12, VCAM-1). There were no reason to assume linearity in what they wanted to predict, so they also developed additive non-parametric models (by calculating log-odds of death by using a spline function) to find non-linear relationships between the molcular markers, clinical variables, and patient outcome. Therefore, the authors used non-linear transformations of the predictors, and developed two logistic regressions, one for clinical predictors, and the other for molecular predictors. The authors combined these models into a unified model with smoothed estimates of probability density function and their final method is able to predict a continous value.

Positive aspects of their work is that the authors selected their features very well. They talked about the methods they used which resulted in failure and they said why those methods were not appropriate. They didn't rely only on linear model and developed non-linear models. For the analysis part, the authors were careful to avoid dependencies among features and the response variable.

Negative aspects of their work is that the study did not include a validation set, and this is a significant drawback. The authors did not mention why the parametric models need to be transformed to non-parametric model. The authors claim that their methods is generalizable to determining the prognosis of other diseases, but they didn't provide any other information and it is not clear how and to which disease they can generalize their model.

Possible extensions of this work is to sample more patients. The samples for this paper were collected from ArMORR in united states, however it can be helpful to investigate other samples from other countries. It is useful to have a large validation set from a bigger population of patients and test the model on them.

Problem 3

No solution provided

Problem 4

a)

I saved the data from the website on my computer and then read it from my computer. You can directly get it from the website.

```
# read the data
#prostate <- read.csv("/Users/saratahe ri/Desktop/prostate/prostate.csv")
prostate <-
  read.table("/Users/ovitek/Dropbox/CS-76140 - Spring 2017/Homeworks/HW2/prostate.data",
  sep="\t", header=TRUE)
# get the training set
prostate_train <- prostate[prostate$train == TRUE,]
head(prostate_train)
```

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

```
prostate_train <- prostate_train[,-1] # remove the first column from training set
prostate_train <- prostate_train[,-10] # remove the last column (named train) from training set
# get the validation set
prostate_validation_set <- prostate[prostate$train == FALSE,]
prostate_validation_set <- prostate_validation_set[,-1] # remove the first column from validation set
prostate_validation_set <- prostate_validation_set[,-10] # remove the last column from validation set
```

---

b)

One variable summary statistics:

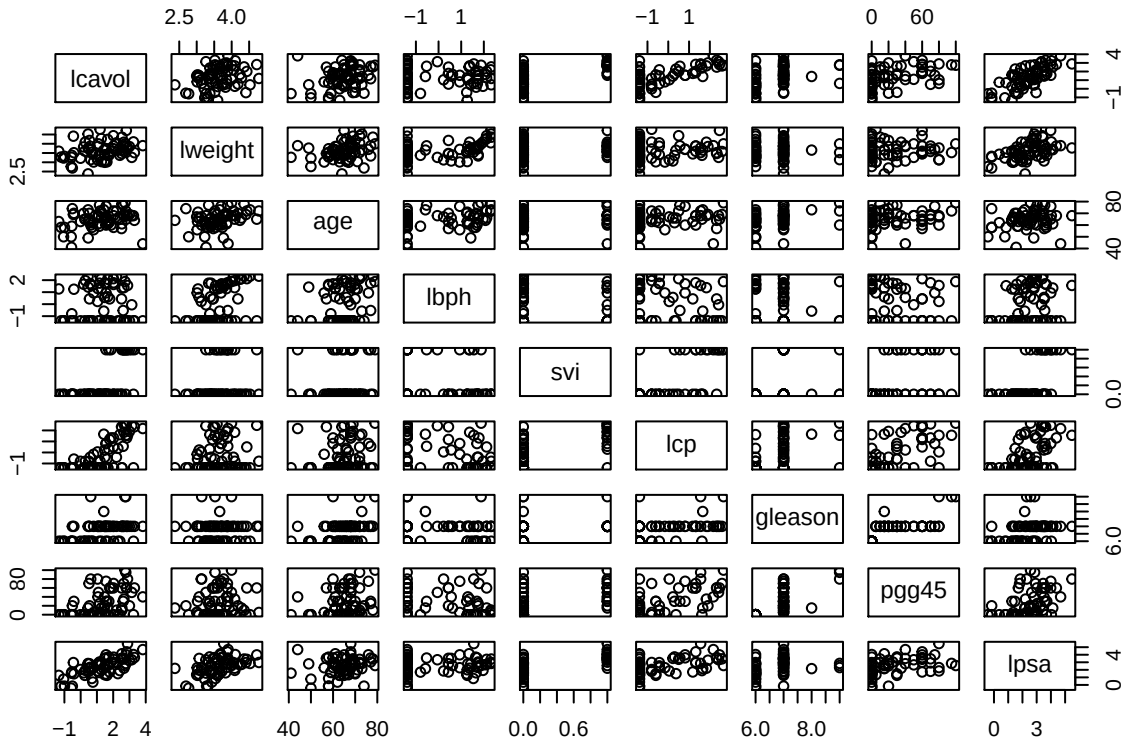
```
summary(prostate_train)
```

```
##      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
```

Two variable summary statistics:

```
pairs(prostate_train) # scatter plot matrix
```




```
round(cor(prostate_train[,-10]), digits = 2)
```

```
##         lcavol lweight  age  lbph   svi   lcp gleason pgg45 lpsa
## lcavol    1.00    0.30 0.29  0.06  0.59  0.69    0.43  0.48 0.73
## lweight   0.30    1.00 0.32  0.44  0.18  0.16    0.02  0.07 0.49
## age       0.29    0.32 1.00  0.29  0.13  0.17    0.37  0.28 0.23
## lbph      0.06    0.44 0.29  1.00 -0.14 -0.09    0.03 -0.03 0.26
## svi       0.59    0.18 0.13 -0.14  1.00  0.67    0.31  0.48 0.56
## lcp       0.69    0.16 0.17 -0.09  0.67  1.00    0.48  0.66 0.49
## gleason   0.43    0.02 0.37  0.03  0.31  0.48    1.00  0.76 0.34
## pgg45     0.48    0.07 0.28 -0.03  0.48  0.66    0.76  1.00 0.45
## lpsa      0.73    0.49 0.23  0.26  0.56  0.49    0.34  0.45 1.00
```

As we can see, lcavol is highly correlated with lpsa. Later on we will see that lcavol will be selected by best subset selection method as one of the predictors in linear regression. We can see that gleason has a high correlation with lpsa, but later on, we will see that best subset selection never selects it, unless we want to include all 8 predictors in our model.

Looking for missing values

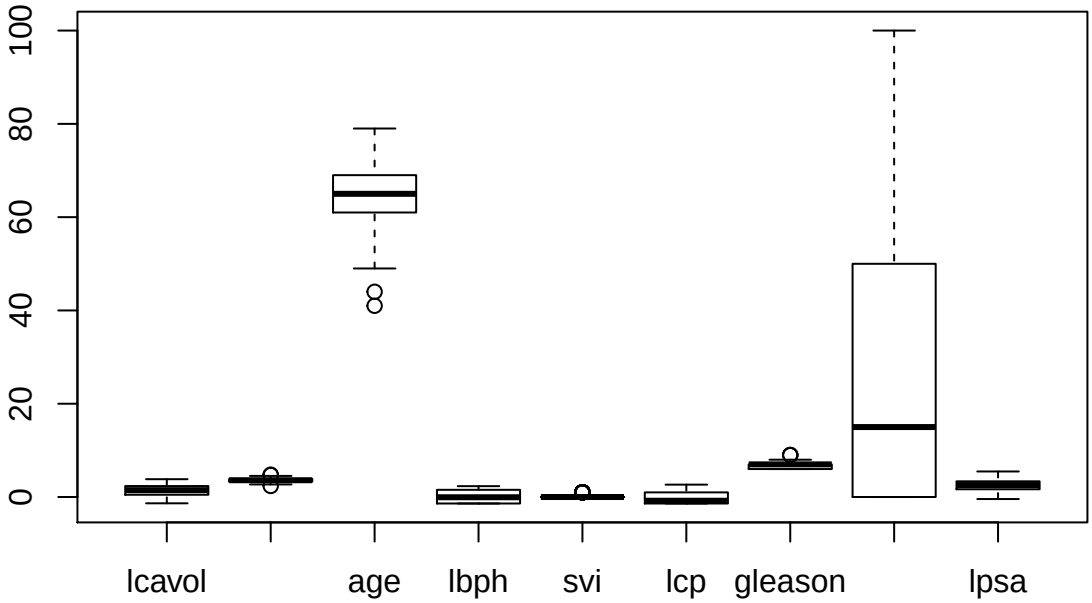
```
any(is.na(prostate))
```

```
## [1] FALSE
```

Fortunately, there are no missing values in our data set.

Looking for outliers

```
boxplot(prostate_train)
```



It is not obvious whether we can consider the points outside the boxplot area for each feature as outliers or not. Therefore, we do not remove any observations.

---

c)

Fit linear regression with all predictors:

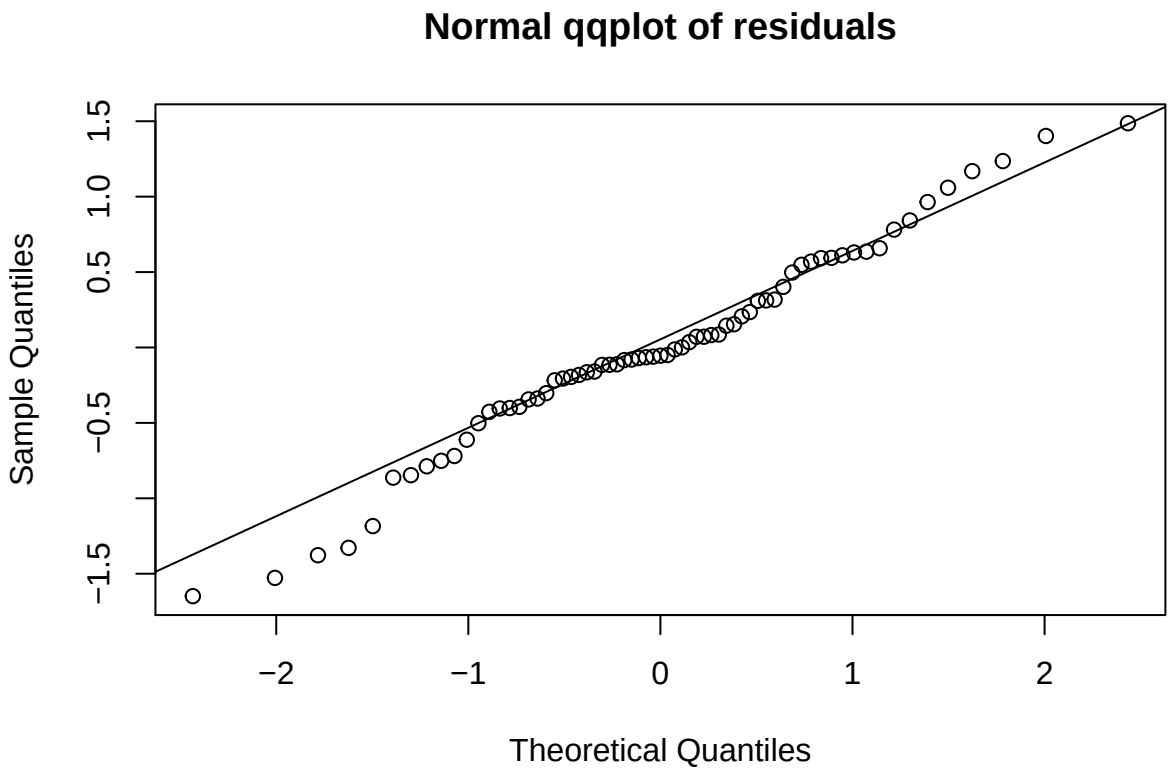
```
lm_prostate_train <- lm(lpsa~.,data = prostate_train)
summary(lm_prostate_train)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostate_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64870 -0.34147 -0.05424  0.44941  1.48675
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.429170   1.553588   0.276  0.78334
```

```
## lcavol       0.576543   0.107438   5.366 1.47e-06 ***
## lweight      0.614020   0.223216   2.751  0.00792 **
## age         -0.019001   0.013612  -1.396  0.16806
## lbph         0.144848   0.070457   2.056  0.04431 *
## svi          0.737209   0.298555   2.469  0.01651 *
## lcp         -0.206324   0.110516  -1.867  0.06697 .
## gleason     -0.029503   0.201136  -0.147  0.88389
## pgg45        0.009465   0.005447   1.738  0.08755 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7123 on 58 degrees of freedom
## Multiple R-squared:  0.6944, Adjusted R-squared:  0.6522
## F-statistic: 16.47 on 8 and 58 DF,  p-value: 2.042e-12
```

Draw the quantile quantile plot of the residuals. It indicates only minor departures from Normality.

```
#quantile-quantile plot
qqnorm(lm_prostate_train$residuals, main = "Normal qqplot of residuals")
qqline(lm_prostate_train$residuals)
```



---

d)

The **regsubsets()** function (part of the **leaps** library) performs all subset selection (best subset selection) by identifying the best model that contains a given number of predictors, where *best* is quantified using RSS.

```
library(leaps)
regfit.full <- regsubsets(lpsa~., data = prostate_train)
reg.summary <- summary(regfit.full)
reg.summary
```

```
## Subset selection object
## Call: regsubsets.formula(lpsa ~ ., data = prostate_train)
## 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 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

In the output, an asterisk indicates that a given variable is included in the corresponding model. For instance, this output indicates that the best one-variable model contains only lcavol and the best two-variable model contains only lcavol and lweight and so on. Now we have to find the best model among these 8 models. We know that the model containing all of the predictors will always have the smallest RSS and the largest R^2, since these quantities are related to the training error. Instead,we wish to choose a model with a low test error and the training error can be a poor estimate of the test error. Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

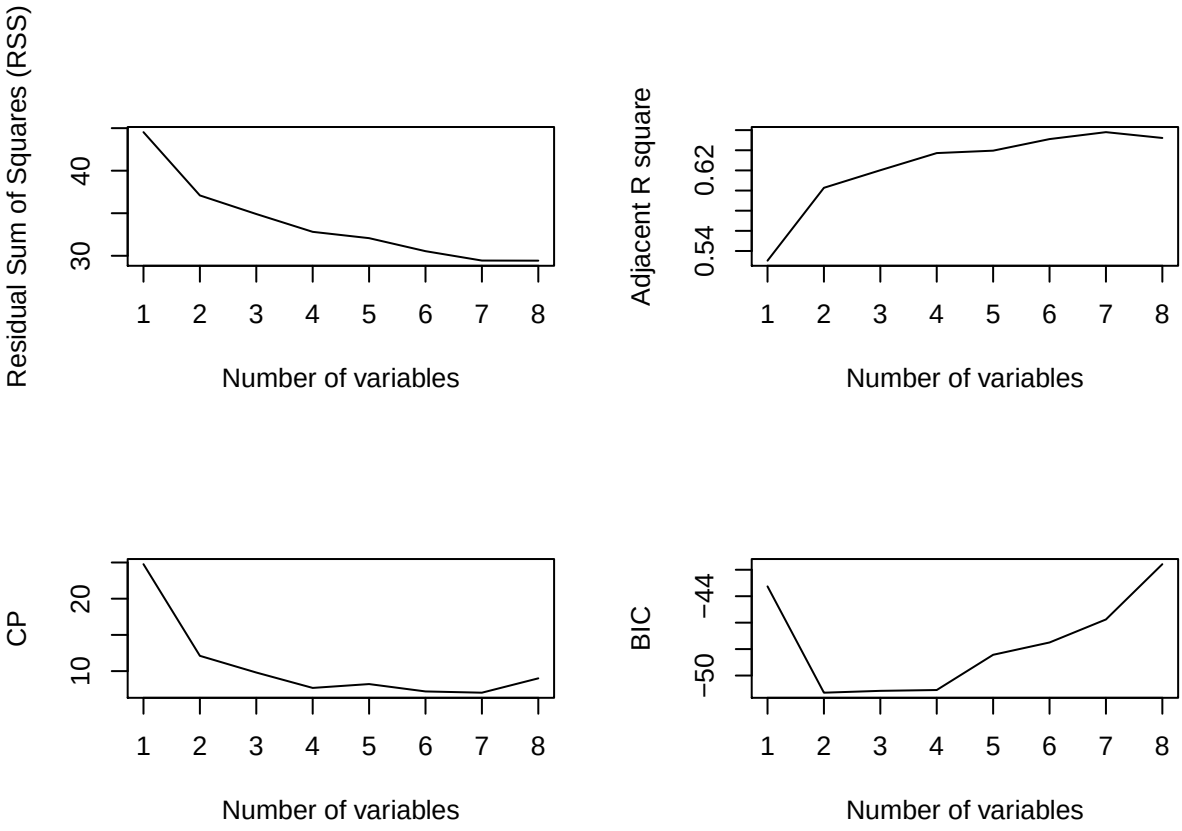
In order to select the best model with respect to test error, we need to estimate this test error. There are two common approaches:

1. We can indirectly estimate test error by adding a penalty to the training error to account for the bias due to overfitting.
2. We can directly estimate the test error, using either a validation set approach or a cross-validation approach

Here we will use the first approach.

A number of techniques for adjusting the training error for the model size are available. These approaches can be used to select among a set of models with different numbers of variables. We now consider four such approaches: Cp, Akaike information criterion (AIC), Bayesian information criterion (BIC), and adjusted R2. Let's plot them:

```
par(mfrow = c(2,2))
plot(reg.summary$rss, xlab = "Number of variables", ylab = "Residual Sum of Squares (RSS)", type = "l")
plot(reg.summary$adjr2, xlab = "Number of variables", ylab = "Adjacent R square", type = "l")
plot(reg.summary$cp, xlab = "Number of variables", ylab = "CP", type = "l")
plot(reg.summary$bic, xlab = "Number of variables", ylab = "BIC", type = "l")
```



By looking at the plots, it seems that choosing 2 predictors is fine. We can also see it like this:

```
which.min(reg.summary$bic)
```

```
## [1] 2
```

Therefore, it is reasonable to include 2 predictors in our linear regression and these two predictors based on best subset selection that we did above are lcavol and lweight. Note that if you look at the RSS plot you can see that as the number of predictors increases the RSS decreases. I mentioned the reason and why we shouldn't rely on RSS in previous paragraph. Now let's get the coefficients for the linear model based on 2 predictors.

```
# linear model based on 2 predictors
subset_select_2predictor <- lm(lpsa ~ lcavol + lweight, data = prostate_train)
# coefficients of the predictors
coef(regfit.full, 2)
```

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

The linear model is : $lpsa = -1.0494396 + 0.6276074 \times lcavol + 0.738375 \times lweight + error,\ error \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$

---

e)

Ridge regression:

First, I will plot the ridge regression coefficients for different values of λ :

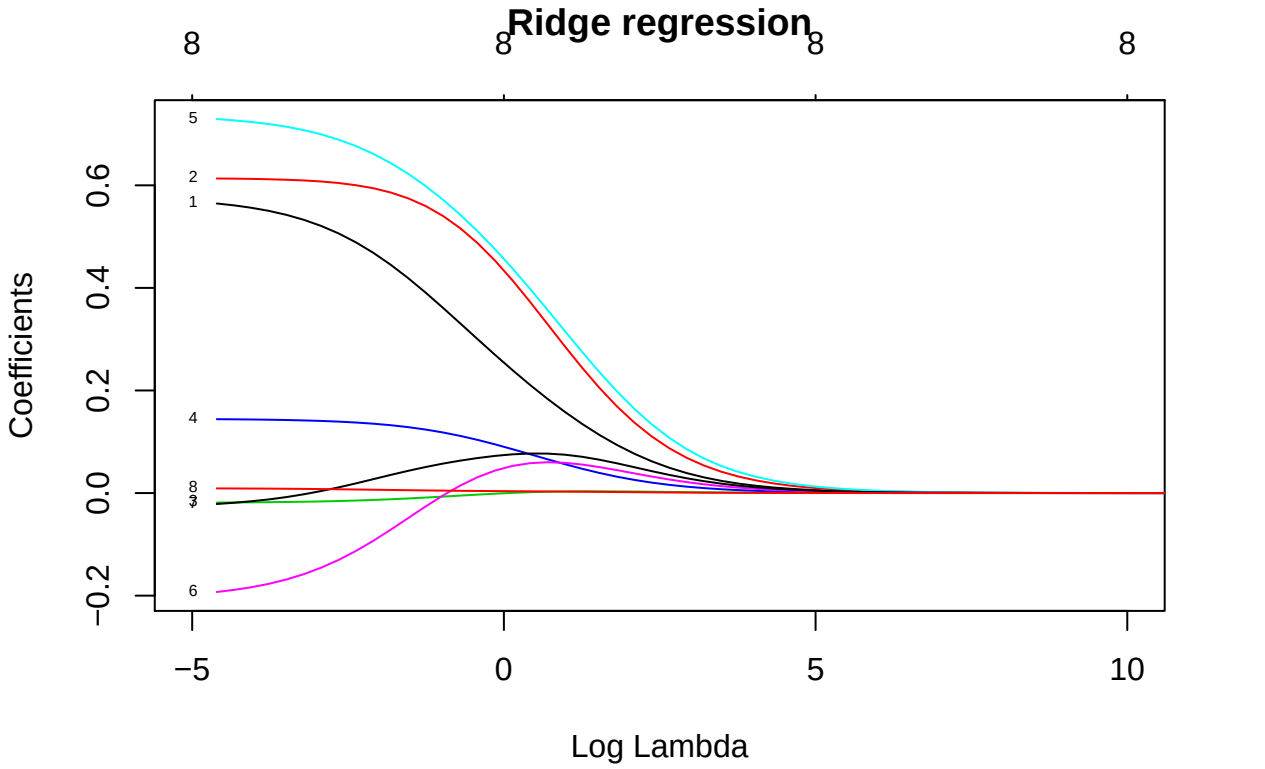
```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-2
```

```
x <- model.matrix(lpsa~.,prostate_train)[,-1]
y <- prostate_train$lpsa
grid = 10^seq(10,-2, length = 100)
ridge.mode = glmnet(x, y, alpha=0, lambda = grid)
plot(ridge.mode, main = "Ridge regression",label = TRUE, xvar = "lambda", xlim = c(-5,10))
```



By default the **glmnet()** function performs ridge regression for an automatically selected range of λ values. However, here we have chosen to implement the function over a grid of values ranging from $\lambda = 10^{10}$ to $\lambda = 10^{-2}$, essentially covering the full range of scenarios from the null model containing only the intercept, to the least squares fit.

Note that in order to evaluate the performance of a statistical learning method on a given data set, we need some way to measure how well its predictions actually match the observed data. That is, we need to quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation. In the regression setting, the most commonly-used measure is the mean squared error (MSE), given by,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

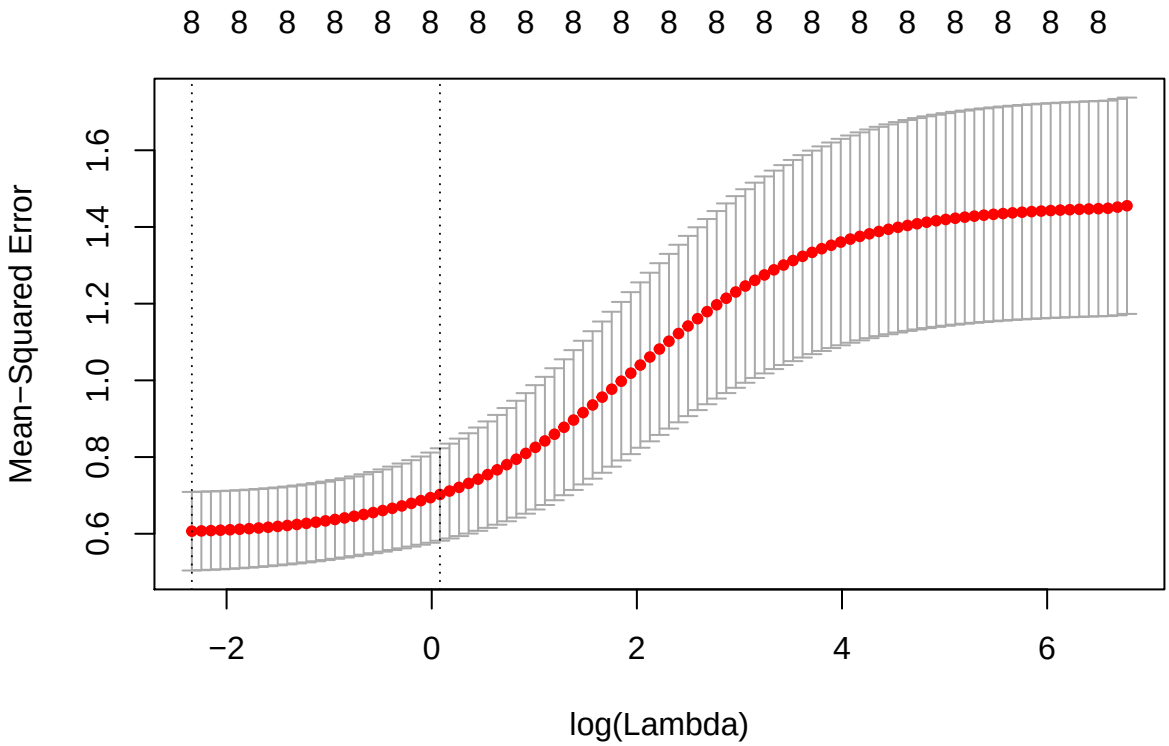
where $\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the ith observation. y_i is the real value of the observation. Basically, you will find a model based on the training set and you will evaluate your model on the validation set by computing MSE on the validation set. Here y_i's are the real values of lpsa in our validation set and $\hat{f}(x_i)$'s are the estimated value of lpsa that are computed based on the model we obtained from the training set. The MSE will be small if the predicted responses are very close to the true responses, and will be large if for some of the observations, the predicted and true responses differ substantially.

In general, instead of arbitrarily choosing λ, it would be better to use cross-validation to choose the tuning

parameter λ. It involves randomly dividing the available set of observations (here our observations are prostate_train) into two parts, a *training set* and a *validation set* or *hold-out* set (I am talking about dividing prostate_train into two part). The model is fit on the training set, and the fitted model is used to predict the response for the observations in the validation set. The resulting validation set error rate typically used MSE in the case of a quantitative response to provide and estimate the test error rate. We can do this using the built-in cross-validation function, **cv.glmnet()**. By default, the function performs ten-fold cross-validation, though this can be changed using the argument **folds**.

Now I will plot the MSE vs log(lambda):

```
cv.out <- cv.glmnet(x,y, alpha = 0)
plot(cv.out)
```



If you look at the top of this plot, you can see that regardless of range of λ the model always selects 8 predictors. This is because the ridge doesn't do variable selection. You can see that a range of $\log(\lambda)$ is specified between two vertical dashed lines which shows a range of $\log(\lambda)$ that the MSE is almost the same and it is significantly less than the MSE for other ranges of $\log(\lambda)$. The model chooses the λ with least MSE. Here we find the best λ:

```
bestlam.ridge = cv.out$lambda.min
bestlam.ridge
```

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

```
log(bestlam.ridge)
```

```
## [1] -2.338658
```

The value of λ that results in the smallest cross-validation error is 0.0964. Now we fit a ridge regression model on the training set using $\lambda = 0.0964$

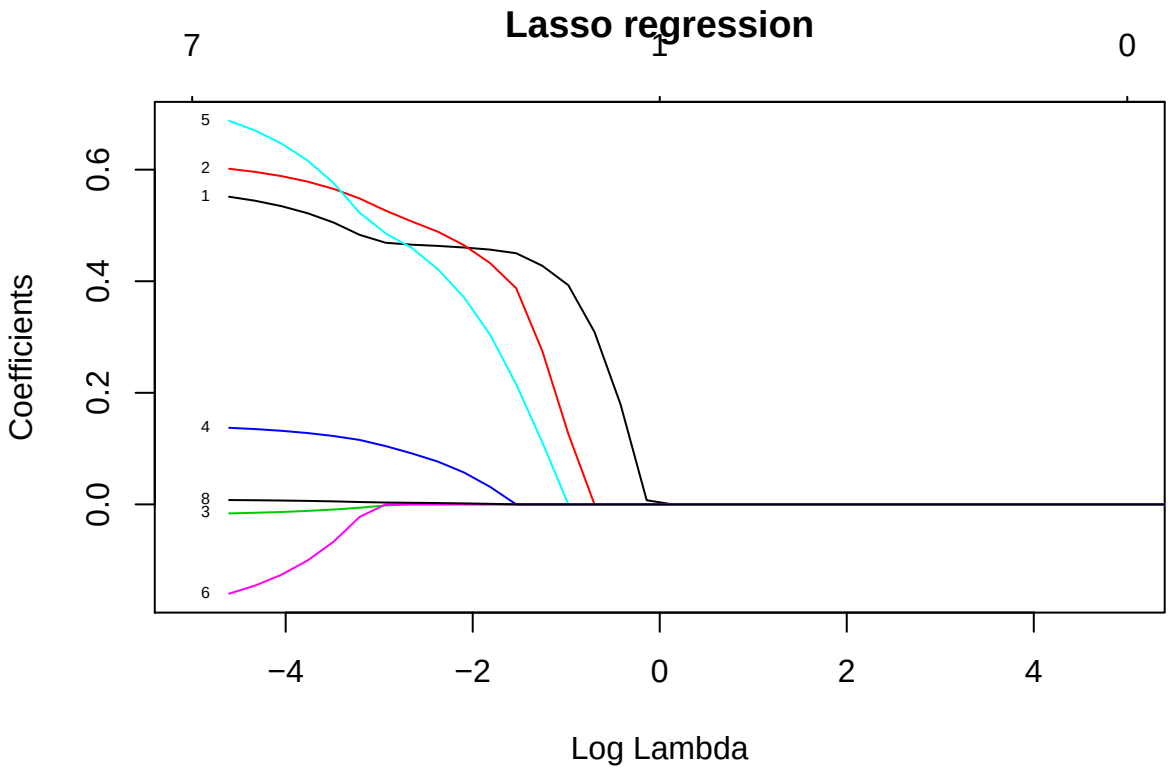
```
ridge.mode <- glmnet(x, y, alpha=0, lambda = bestlam.ridge)
predict(ridge.mode, s = bestlam.ridge, type = "coefficients")[1:9,]
```

```
##  (Intercept)       lcavol      lweight          age         lbph
##  0.075874913  0.486393030  0.599533912 -0.014470377  0.137317539
##          svi          lcp      gleason        pgg45
##  0.674949305 -0.110476104  0.019892200  0.006930003
```

By looking at the coefficients we can see that coefficients of *gleason* and *pgg45* and *age* are very close to zero but none of the coefficients is exactly zero. We end up with a model with 8 predictors by ridge method.

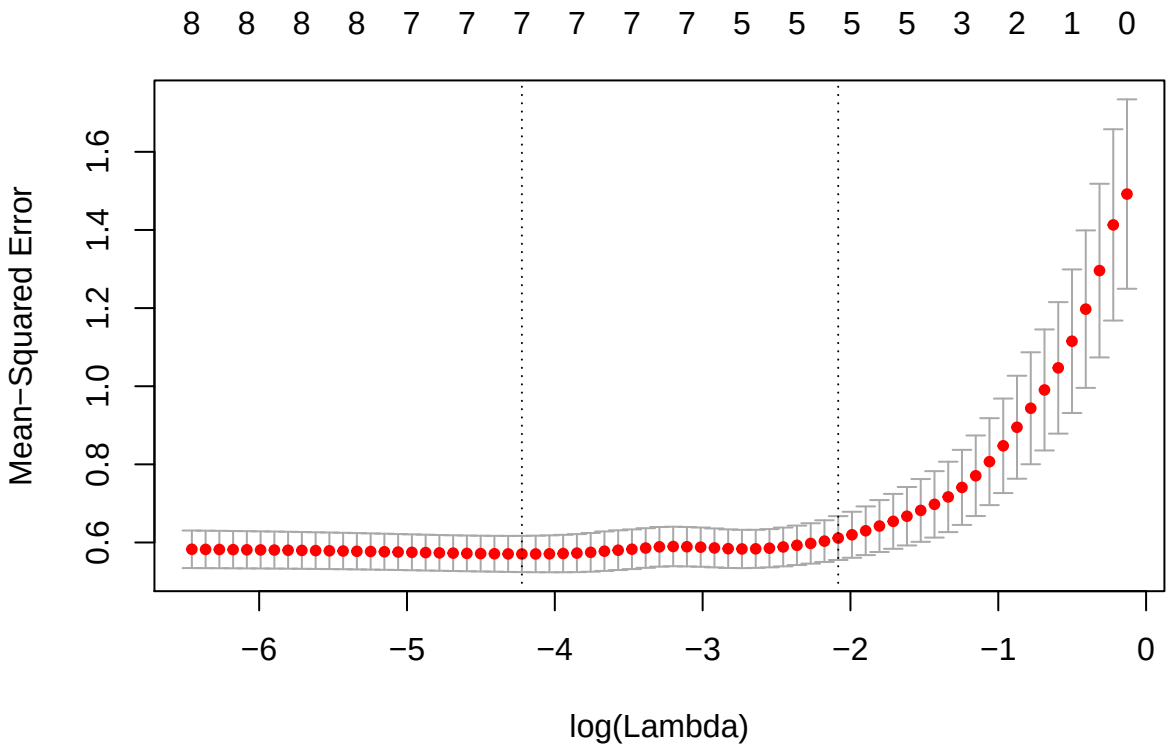
For the lasso regression, again, first I will plot the coefficients for different values of λ :

```
lasso.mod <- glmnet(x,y, alpha = 1, lambda = grid)
plot(lasso.mod, main = "Lasso regression", label = TRUE, xvar = "lambda", xlim = c(-5,5))
```



Now I will perform cross-validation and compute the best λ :

```
cv.out <- cv.glmnet(x,y,alpha = 1)
plot(cv.out)
```



If you look at the top of this plot, you can see that for different values of λ the model selects different number of predictors. This is because the lasso does the variable selection (note that ridge does not). You can see that a range of $\log(\lambda)$ is specified between two vertical dashed lines which shows a range of $\log(\lambda)$ that the MSE is almost the same and it is significantly less than the MSE for other ranges of λ. For this range of $\log(\lambda)$ the number of predictors selected by lasso is one of 7,6,5 or 3. The model chooses the λ with least MSE which contains 7 predictors. Let's find the best λ:

```
bestlam.lasso <- cv.out$lambda.min
bestlam.lasso
```

```
## [1] 0.01466061
```

```
#best log(lambda)
log(bestlam.lasso)
```

```
## [1] -4.222591
```

I fit a lasso regression on the training set with $\lambda = 0.011$ to get the coefficient of my linear model:

```
lasso.mode <- glmnet(x, y, alpha=1, lambda = bestlam.lasso)
predict(lasso.mode, s = bestlam.lasso, type = "coefficients")[1:9,]
```

```
##  (Intercept)       lcavol      lweight          age         lbph
##  0.154541678  0.541058145  0.593668833 -0.014565214  0.133870559
##          svi          lcp      gleason        pgg45
##  0.661837580 -0.139025510  0.000000000  0.007230984
```

You can see that the coefficient of *gleason* is zero, which means that we should not include it in our model and the coefficient of *pgg45* and *age* are very close to zero. We end up with a model with 7 (8-1) predictors by lasso method.

There are various special algorithms for LASSO that you can read about them in the chapter 13 section 4 of KM textbook.

---

f)

To evaluate the performance of the models that we found in previous sections, I will calculate the mean square error for all the models:

```
real_value_lpsa <- prostate_validation_set$lpsa
prostate_validation_set$lpsa <- NULL
newx <- data.matrix(prostate_validation_set)
# predict the values of lpsa based on the linear regression model on all predictors
all_variables_prediction <- predict(lm_prostate_train, newdata = prostate_validation_set)
# predict values of lpsa based on 2 predictors selected in all subset selection
all_subset_selection_prediction <- predict(subset_select_2predictor,
                                           newdata = prostate_validation_set)
# predict values of lpsa based on lasso regression
lasso_prediction <- predict(lasso.mode, newx = newx)
# predict values of lpsa based on ridge regression
ridge_prediction <- predict(ridge.mode, newx = newx)
```

Now that we have all the predicted values of lpsa based on our 4 models, we can compare the real value of lpsa in validation set with the predicted values to find out which model performs best. I will choose to do it by computing the value of MSE for each model.

```
error_all_variables_prediction <-
  sum((all_variables_prediction-real_value_lpsa)^2)/nrow(prostate_validation_set)
error_all_variables_prediction
```

```
## [1] 0.521274
```

```
error_all_subset_selection_prediction <-
  sum((all_subset_selection_prediction - real_value_lpsa)^2)/nrow(prostate_validation_set)
error_all_subset_selection_prediction
```

```
## [1] 0.4924823
```

```
error_lasso_prediction <-
  sum((lasso_prediction - real_value_lpsa)^2)/nrow(prostate_validation_set)
error_lasso_prediction
```

```
## [1] 0.4912834
```

```
error_ridge_prediction <-
  sum((ridge_prediction - real_value_lpsa)^2)/nrow(prostate_validation_set)
error_ridge_prediction
```

```
## [1] 0.4932241
```

It seems that the model chosen by all subset selection which contains 2 predictors is the best model among other models.

Just to dig more into variable selection and compare different methods that we have learned so far, I will want to compare ridge, lasso and linear regression as follows:

We know that ridge includes all 8 predictors in the model, but with different coefficients assigned to them than what linear regression will assign to. We already applied both models to validation set and compared their MSE and we saw that the MSE for linear model including all predictors is 0.521 and the MSE for Ridge is 0.493.

We know that lasso does the subset selection and if you look at the plot on page 12 you can see that the suggested number of predictors depending on which value of λ to pick is 7 or 6 or 5 or 3. We chose 7 predictors (all the predictors except gleson) and build our model based on those 7 predictors and then applied that

model on the validation set and calculated the MSE. The MSE was 0.511 . Let's apply linear regression on these 7 predictors selected by Lasso and then validate it on the validation set and compare it's MSE with the MSE of Lasso:

```
lm_7predictor_of_lasso <- lm(lpsa~lcavol+lweight+age+lbph+svi+lcp+pgg45, data = prostate_train)
lm_7predictor_of_lasso_prediction <- predict(lm_7predictor_of_lasso, newdata = prostate_validation_set)
error_lm_7predictor_of_lasso_prediction <-
  sum((lm_7predictor_of_lasso_prediction - real_value_lpsa)^2)/nrow(prostate_validation_set)
error_lm_7predictor_of_lasso_prediction
```

```
## [1] 0.5165135
```

The MSE is 0.516 . This suggests that with 7 predictors lasso is doing better than simple linear regression exactly with the 7 predictors that lasso selects.

You can do this kind of comparison yourself. Pick a value of λ from the plot on page 12 that results in 5 predictors (or 6 or 3, whatever you like!). Fit a lasso regression on the training set with that λ and find the coefficients for the model and find out which predictors get the zero coefficient and are not included in the model. you should find 3 predictors with zero coefficients! Fit your model on the validation set and find it's MSE. Now move to simple linear regression and apply linear regression on those 5 predictors that lasso founds. Then fit your linear model on the validation set and find it's MSE. Compare this two MSE with each other and see which one does a better job.

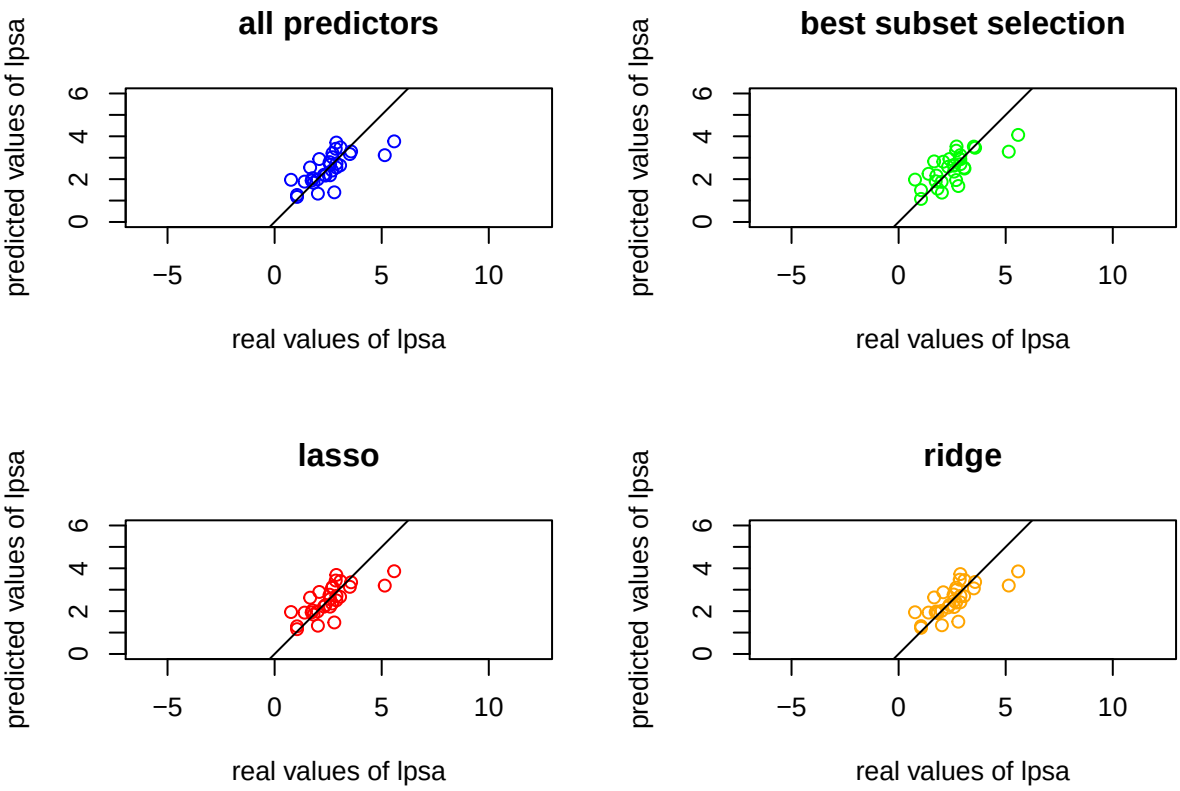
---

g)

It is not surprising that the model selected by best subset selection is the best model, because in order to reach this model in the algorithm of best subset selection, we consider all the possible models with all the possible number of predictors and among all the 2^p models we choose the best one.

We can compare the obserevd values for each model (the values that each model predict for the value of lpsa) with the actual value of lpsa by drawing the below plots. Note that the closer the points are to the line $y = x$, the better the model is.

```
par(mfrow = c(2,2))
plot(x = real_value_lpsa, y = all_variables_prediction, xlab =  "real values of lpsa",
     ylab = "predicted values of lpsa", main = "all predictors", col = "blue", xlim=c(0,6), ylim=c(0,6)
abline(a = 0, b = 1)
plot(x = real_value_lpsa, y = all_subset_selection_prediction, xlab =  "real values of lpsa",
     ylab = "predicted values of lpsa", main = "best subset selection", col = "green", xlim=c(0,6), yli
abline(a = 0, b = 1)
plot(x = real_value_lpsa, y = lasso_prediction, xlab =  "real values of lpsa",
     ylab = "predicted values of lpsa", main = "lasso", col = "red", xlim=c(0,6), ylim=c(0,6), asp=1)
abline(a = 0, b = 1)
plot(x = real_value_lpsa, y = ridge_prediction, xlab =  "real values of lpsa",
     ylab = "predicted values of lpsa", main = "ridge", col = "orange", xlim=c(0,6), ylim=c(0,6), asp=1
abline(a = 0, b = 1)
```



Selecting the best bodel may not be very obvious by looking at the plots.