sx2337_datamining_hw1_coding

Shun Xie

Contents

Q3

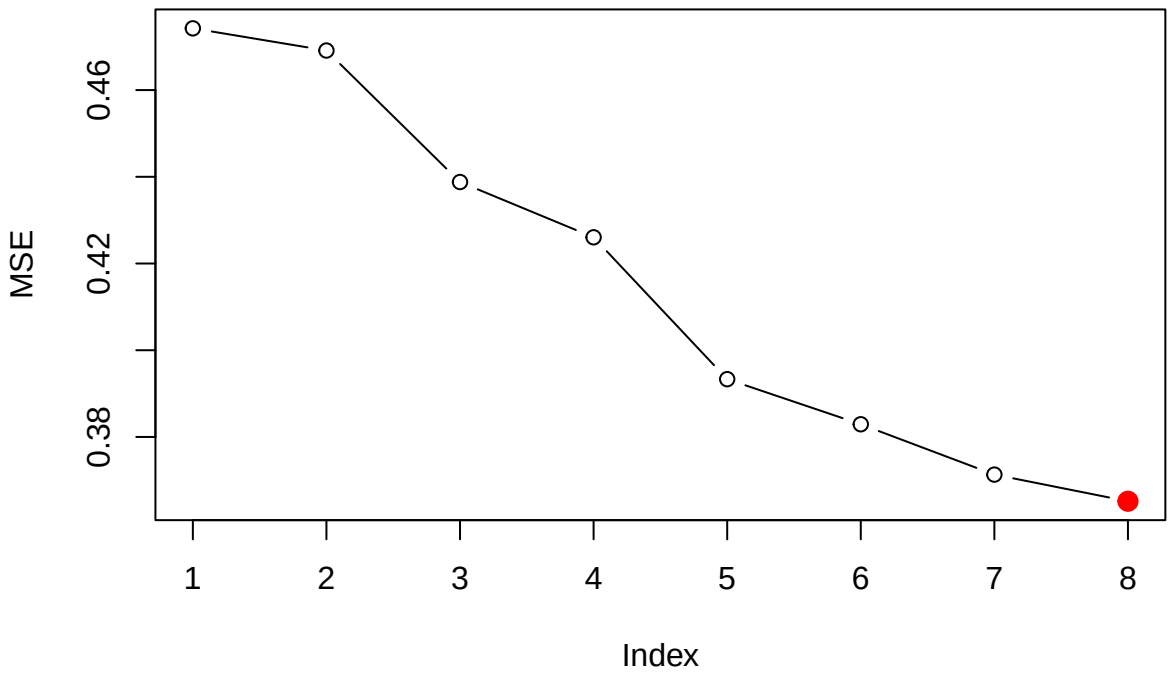
Part 1: Best-subset linear regression with k chosen by 5-fold cross-validation

```
library(leaps)
library(boot)
library(genridge)
library(dplyr)
library(forcats)
library(ISLR)
library(glmnet)
library(caret)
library(stats)
library(pls)
#get data
data(prostate)
```

```
#Preprocessing
prostate_tmp <- prostate %>%
    mutate(train = fct_relevel(factor(train), "TRUE", "FALSE"),
           gleason = fct_relevel(factor(gleason), "6", "7"),
           svi = factor(svi))

#shuffle the data
set.seed(2337)
prostate_shuffle <- prostate[sample(nrow(prostate_tmp)), ]

#get the training and test sets
train_data <- prostate[1:67, ]
test_data <- prostate[68:length(prostate$lcavol), ]
```

```
#get training data x design matrix and y response value
x <- model.matrix(lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,train_data)[,-1]
y <- train_data$lpsa

#get test data x design matrix and y response value
x_test <- model.matrix(lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,test_data)[,-
y_test <- train_data$lpsa


# define the predict function
predict.regsubsets = function(object,newdata,id,...){
      form = as.formula(object$call[[2]]) #extract the formula of regsubset
      mat = model.matrix(form,newdata)    #get the model matrix
      coefi = coef(object,id=id)          #get the coefficient with i number of predictors
      xvars = names(coefi)                #get the name of predictors
      mat[,xvars]%*%coefi               #make prediction using new data
}
```

```r
#number of subsets
k=5

## Manually conduct 5 fold cross validation to get MSE
#Get the randomized 5 folds
set.seed(2337)
folds = sample(1:k, nrow(train_data), replace = TRUE)

#initialization to store matrix
cv_error = matrix(NA, k, 8, dimnames = list(NULL, paste(1:8)))

#For all 5-fold data
for(j in 1:k){
    #perform the best subset on the train dataset except for the jth field
    best_fit = regsubsets(lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45, data = tra
    #loop over at most 8 parameters.
    for(i in 1:8){
        #predict the values of the current fold from regsubset with i predictors
        pred = predict(best_fit, train_data[folds==j,], id=i)
        #calculate the MSE
        cv_error[j,i] = mean((train_data$lpsa[folds==j]-pred)^2)
    }
}


#get the mean for each fold (apply on columns)
MSE = apply(cv_error, 2, mean)

#find the number of predictor with minimum mse value
optimal_size = which.min(MSE)

#plot cv error, with optimal point
plot(MSE, type='b')
points(optimal_size, MSE[optimal_size][1], col = "red", cex = 2, pch = 20)
```




```
#final model with all predictors
lmodel1 <- lm(lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason + pgg45,train_data)
#get summary
summary1 <- summary(lmodel1)
summary1
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
##     gleason + pgg45, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.23148 -0.19471  0.00214  0.29164  1.22895
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.658773   1.419166  -0.464   0.6442
## lcavol       0.437818   0.089939   4.868 9.04e-06 ***
## lweight      0.574631   0.223983   2.566   0.0129 *
## age         -0.025391   0.011507  -2.207   0.0313 *
## lbph         0.158509   0.064590   2.454   0.0171 *
## svi         -0.301447   0.378989  -0.795   0.4296
## lcp         -0.107738   0.095686  -1.126   0.2648
## gleason      0.243641   0.164984   1.477   0.1452
## pgg45        0.003096   0.004621   0.670   0.5055
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5763 on 58 degrees of freedom
## Multiple R-squared:  0.5806, Adjusted R-squared:  0.5227
## F-statistic: 10.04 on 8 and 58 DF,  p-value: 1.191e-08
```

```
#Get coefficient
coef1 = summary1$coefficients[,1]

#get test mse and sd
y_pred1 <- predict(lmodel1, newdata = test_data)
mse1 = mean((test_data$lpsa-y_pred1)^2)
sd1 = sd((test_data$lpsa-y_pred1)^2)
print(sprintf("test mse: %.3f", mse1))
```

```
## [1] "test mse: 2.204"
```

```
print(sprintf("test sd: %.3f", sd1))
```

```
## [1] "test sd: 2.374"
```

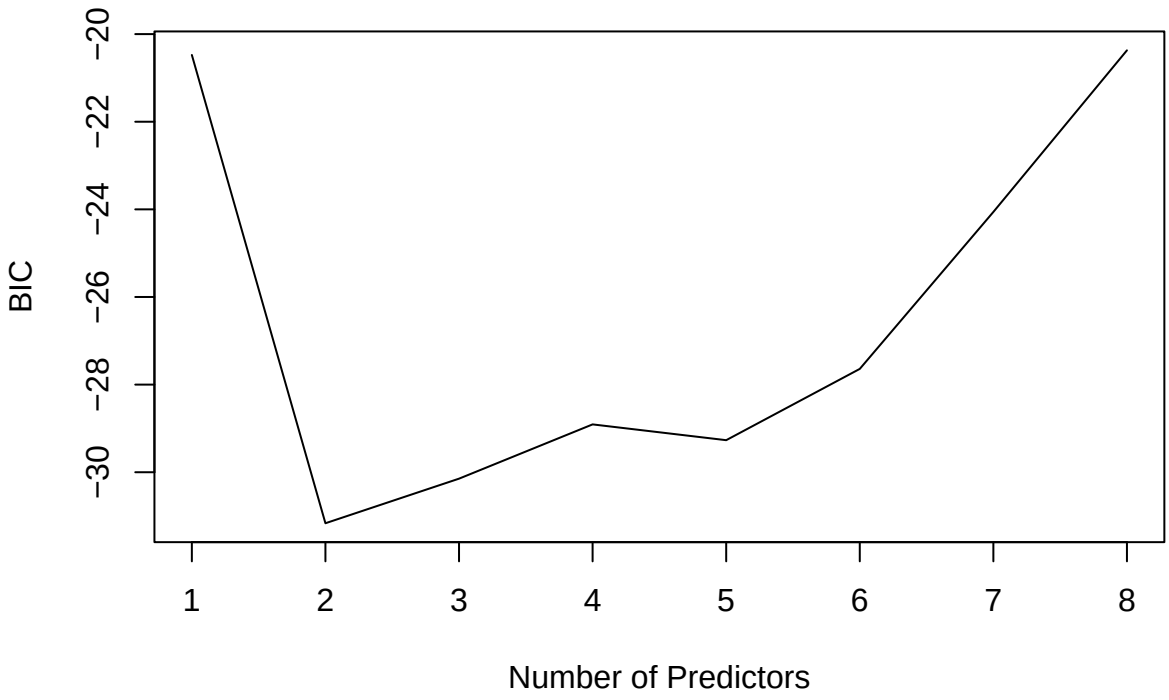
Thus, the test MSE is 2.204 and standard deviation of 2.374 with a model chosen by cv with 8 predictors (full model).

Part 2: best-subset linear regression with k chosen by BIC

```
# Perform best-subset linear regression with 5-fold cross-validation
cv_fit <- regsubsets(
  x=x, y=y,
  data = train_data,
  nvmax = length(train_data),
  really.big = TRUE,  # Required for BIC
  criterion = "bic"  # Use BIC for model selection
)


#Plot the predicted error using bic
summary(cv_fit)
```

```
## Subset selection object
## 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 ) "*"    "*"     "*" "*"  "*" "*" "*"     "*"
```

```
plot(summary(cv_fit)$bic, xlab = "Number of Predictors", ylab = "BIC", type = "l")
```




```
#final model
lmodel2 <- lm(lpsa ~ lcavol + lweight,train_data)
#Get summary
summary2 <- summary(lmodel2)
summary2
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.39287 -0.39542  0.07754  0.41172  1.36318
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.13072    0.66225  -1.707 0.092599 .
## lcavol       0.40021    0.07482   5.349 1.27e-06 ***
## lweight      0.75587    0.18937   3.991 0.000172 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.611 on 64 degrees of freedom
## Multiple R-squared:  0.4797, Adjusted R-squared:  0.4635
## F-statistic: 29.51 on 2 and 64 DF,  p-value: 8.304e-10
```

```r
#get coefficient
coef2 = append(summary2$coefficients[,1],rep(0,6))

#get test mse and sd
y_pred2 <- predict(lmodel2, newdata = test_data)
mse2 <- mean((test_data$lpsa-y_pred2)^2)
sd2 <- sd((test_data$lpsa-y_pred2)^2)
print(sprintf("test mse: %.3f", mse2))
```

```
## [1] "test mse: 1.536"
```

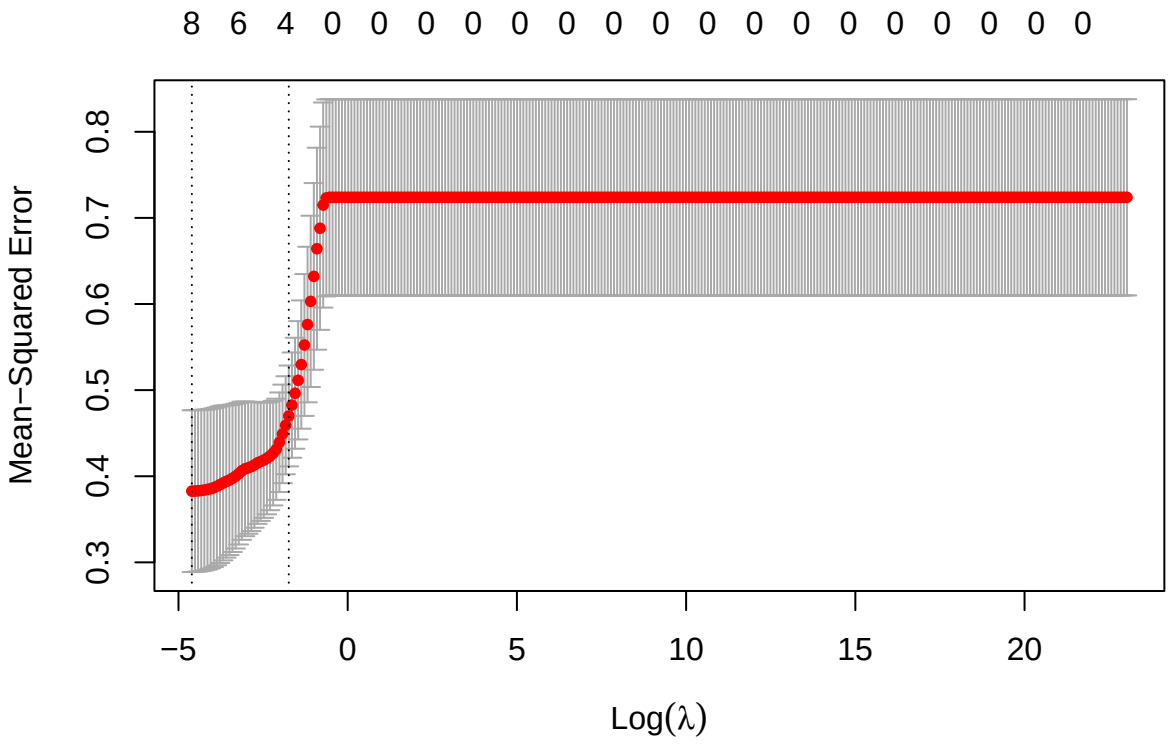
```r
print(sprintf("test sd: %.3f", sd2))
```

```
## [1] "test sd: 1.808"
```

Thus, the test MSE is 1.536 and standard deviation of 1.808 with a model chosen by cv with 2 predictors.

Part 3: lasso regression with λ chosen by 5-fold cross-validation

```
# Set up a range of lambda values to try
lambda_seq <- 10^seq(10, -2, length = 300)

#Lasso regression with 5-fold cross-validation
set.seed(2337)
lasso_cv <- cv.glmnet(
  x = x,
  y = y,
  alpha = 1,       #lasso for 1
  nfolds = 5,
  lambda =lambda_seq
)

#Get all MSE value for lasso
cv_mse <- lasso_cv$cvm

#plot cross-validation estimates of MSE
plot(lasso_cv)
```




```
#Get best lambda
cv_optimal_lambda <- lasso_cv$lambda.min
print(sprintf("Optimal Lambda: %.3f", cv_optimal_lambda))
```

```
## [1] "Optimal Lambda: 0.010"
```

```r
#fit the final Lasso model using the optimal lambda on the training data
final_model_cvlasso <- glmnet(
  x = x,
  y = y,
  alpha = 1,
  lambda = cv_optimal_lambda
)

#Get test mse and sd
y_pred3 <- predict(final_model_cvlasso, newx = x_test)
mse3 <- mean((test_data$lpsa-y_pred3)^2)
sd3 <- sd((test_data$lpsa-y_pred3)^2)

#get coefficient
coef3 = predict(final_model_cvlasso, s='lambda.min', type='coefficients')[,1]

print(sprintf("test mse: %.3f", mse3))
```

```
## [1] "test mse: 2.118"
```

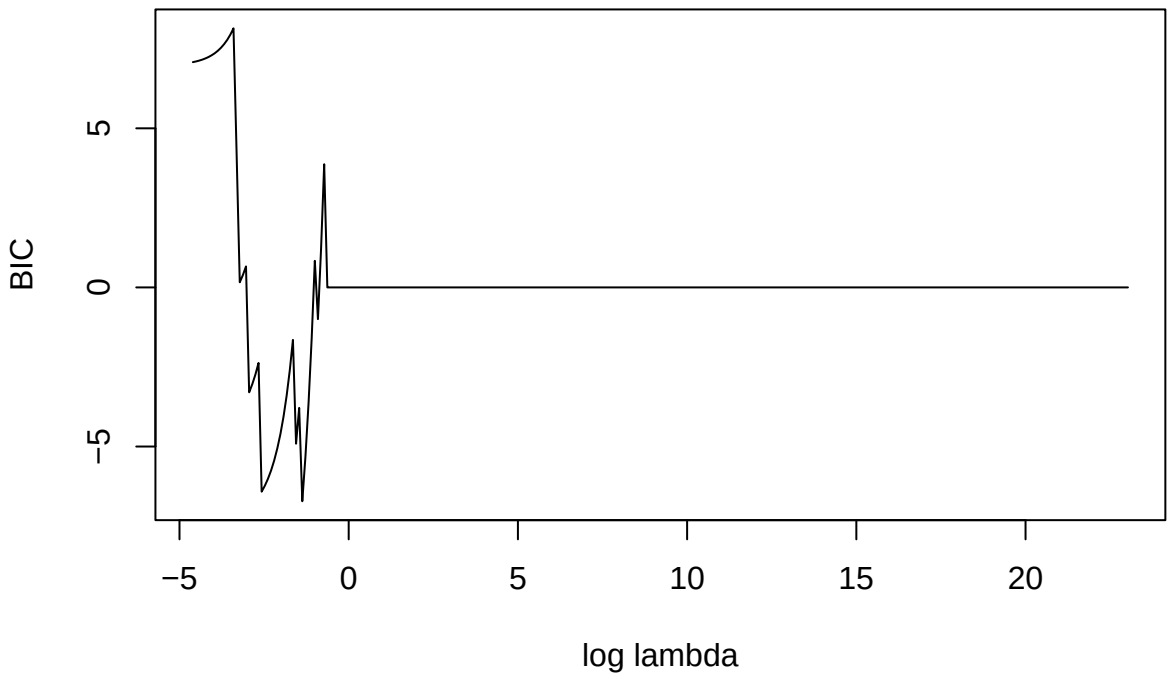
```r
print(sprintf("test sd: %.3f", sd3))
```

```
## [1] "test sd: 2.270"
```

The model chosen by cv has λ=0.010. It includes all 8 predictors. The test MSE is 2.118 and standard deviation is 2.270

Part 4: lasso regression with λ chosen by BIC.

```
# Set up a range of lambda values to try
lambda_seq <- 10^seq(10, -2, length = 300)

#initialization for bic vector with all 0
bic_values <- rep(0, length(lambda_seq))

#iterate all values to get bic
for (i in seq_along(lambda_seq)) {
    lasso_model <- glmnet(
    x = x,
    y = y,
    alpha = 1,
    lambda = lambda_seq[i]
  )
  tLL <- lasso_model$nulldev - deviance(lasso_model)
  k <- lasso_model$df
  n <- lasso_model$nobs

  #Get BIC
  bic_values[i] <- log(n)*k - tLL
}


#plot cross-validation estimates of MSE
plot(log(lambda_seq),bic_values,type = "l", xlab = "log lambda", ylab = "BIC")
```




```
#Get best lambda
bic_optimal_lambda <- lambda_seq[which.min(bic_values)]
print(sprintf("Optimal Lambda: %.3f", bic_optimal_lambda))
```

```
## [1] "Optimal Lambda: 0.254"
```

```
#fit the final Lasso model using the optimal lambda on the training data
final_model_biclasso <- glmnet(
  x = x,
  y = y,
  alpha = 1,
  lambda = bic_optimal_lambda
)

#Get test mse and sd
y_pred4 <- predict(final_model_biclasso, newx = x_test)
mse4 <- mean((test_data$lpsa-y_pred4)^2)
sd4 <- sd((test_data$lpsa-y_pred4)^2)

#Get coefficient
coef4 = coef(final_model_biclasso)[,1]

print(sprintf("test mse: %.3f", mse4))
```

```
## [1] "test mse: 2.508"
```

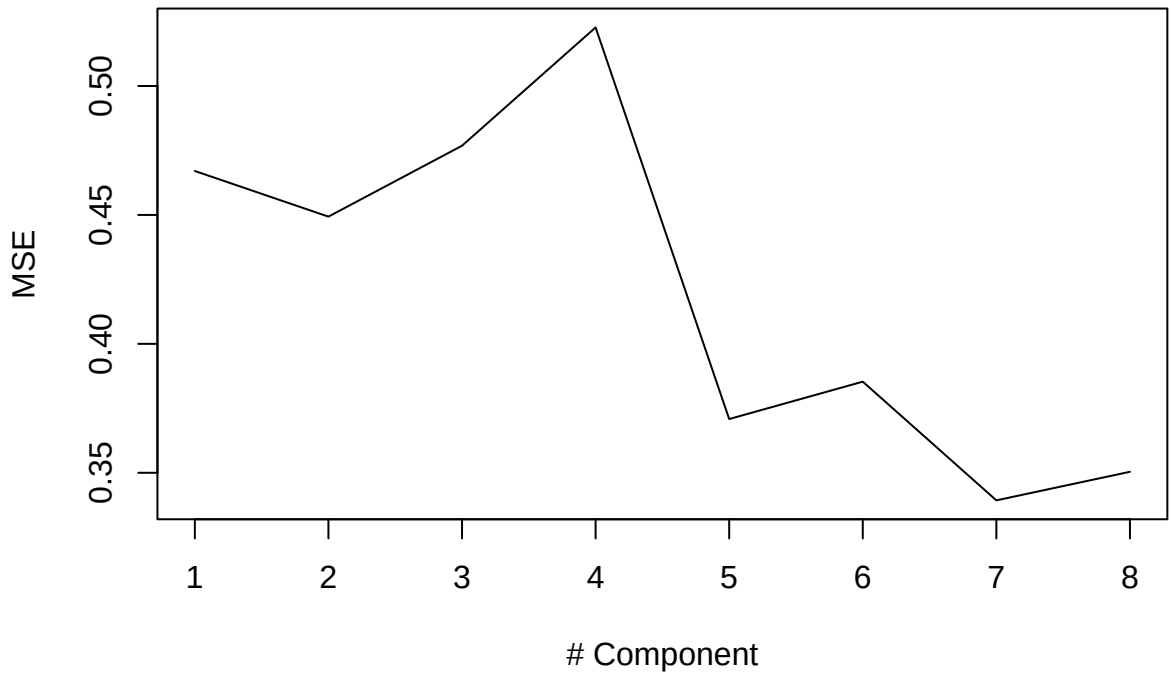
```
print(sprintf("test sd: %.3f", sd4))
```

```
## [1] "test sd: 2.426"
```

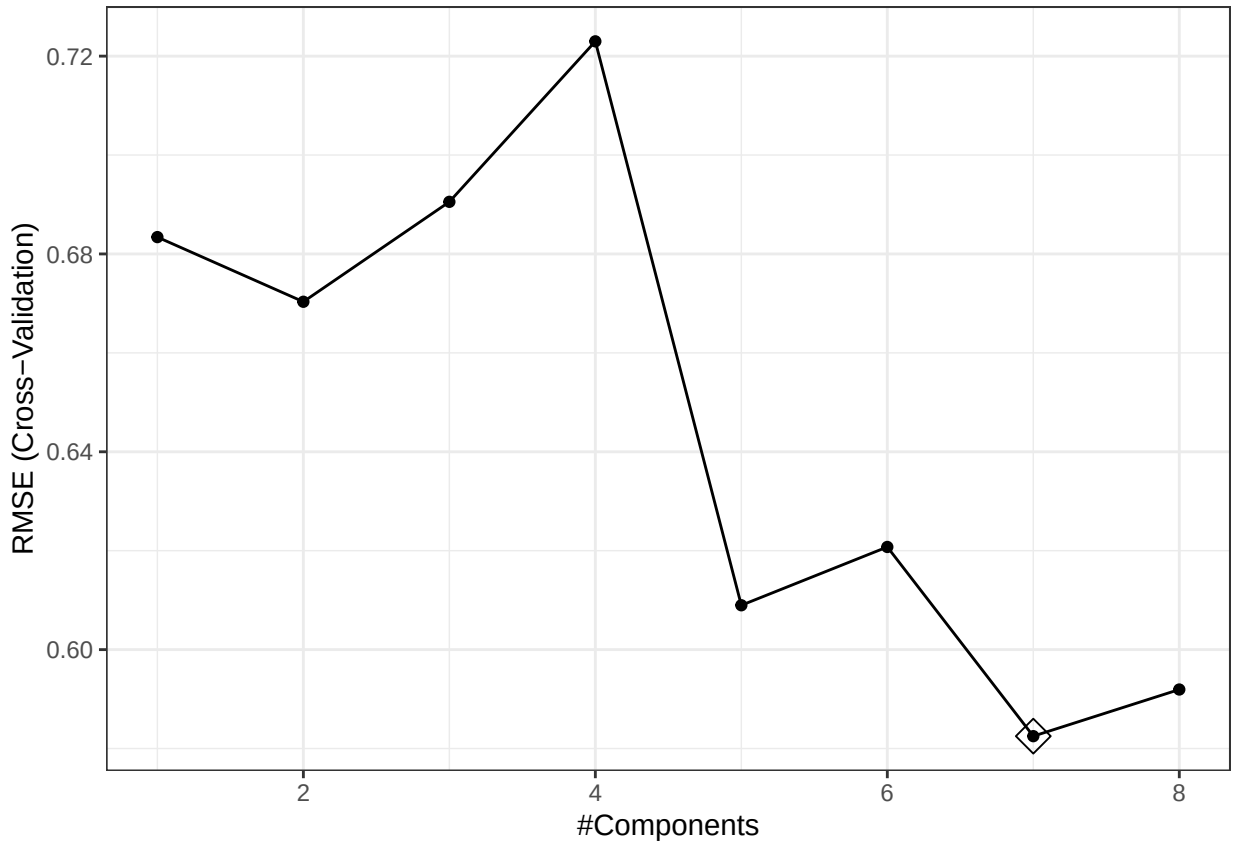
The model chosen by bic has λ=0.254. It only includes 2 predictors, lcavol and lweight. The test MSE is 2.508 and standard deviation is 2.426.

Part 5: Principle component regression with q chosen by 5-fold cross-validation

```
#train control
ctrl1 <- trainControl(method="cv",
                      repeats=5,
                      selectionFunction = 'best')

set.seed(2337)

#train the pcr and use cv
pcr_fit <- train(
    x=x,
    y=y,
    method='pcr',
    tuneGrid = data.frame(ncomp=1:8),
    trControl = ctrl1,
    scale = TRUE)

plot(pcr_fit$results$RMSE**2,type = "l", xlab = "# Component", ylab = "MSE")
```




```
ggplot(pcr_fit,highlight=TRUE)+theme_bw()
```




```
#Get test mse and sd
y_pred5 <- predict(pcr_fit, newx = x_test)
mse5 <- mean((test_data$lpsa-y_pred5)^2)
sd5 <- sd((test_data$lpsa-y_pred5)^2)

#get coefficients
coef5 = coef(pcr_fit$finalModel)[1:8]
print(sprintf("test mse: %.3f", mse5))
```

```
## [1] "test mse: 3.680"
```

```
print(sprintf("test sd: %.3f", sd5))
```

```
## [1] "test sd: 3.239"
```

Thus, the test MSE is 3.680 and standard deviation is 3.239. The final model has 7 predictors.

Discussion

```
#Get output table from coeff
output <- rbind(bs_cv = coef1,
                    bs_bic = coef2,
                    lasso_cv = coef3,
                    lasso_bic = coef4,
                    pcr_cv = coef5)
```

```
## Warning in rbind(bs_cv = coef1, bs_bic = coef2, lasso_cv = coef3, lasso_bic =
## coef4, : number of columns of result is not a multiple of vector length (arg 5)
```

```
#Get all test mse and sd
output_msesd <- data.frame(Test_MSE = c(mse1, mse2,mse3, mse4,mse5), Test_SD =  c(sd1, sd2, sd3,sd4,sd5)

#Specify row name
rownames(output) <- c("bs_cv","bs_bic","lasso_cv","lasso_bic","pcr_cv")

#output table
output %>% knitr::kable(digits=3,col.names = c("intercept", "lcavol", "lweight", "age", "lbph", "svi",
```

	intercept	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45
bs_cv	-0.659	0.438	0.575	-0.025	0.159	-0.301	-0.108	0.244	0.003
bs_bic	-1.131	0.400	0.756	0.000	0.000	0.000	0.000	0.000	0.000
lasso_cv	-0.695	0.412	0.560	-0.022	0.145	-0.253	-0.078	0.229	0.002
lasso_bic	0.819	0.203	0.258	0.000	0.000	0.000	0.000	0.000	0.000
pcr_cv	0.446	0.239	-0.184	0.232	-0.071	-0.118	0.206	0.054	0.446

```
output_msesd %>% knitr::kable(digits=4,col.names = c("Test MSE", "Test MSE SD"))
```

Test MSE	Test MSE SD
2.2041	2.3741
1.5364	1.8085
2.1182	2.2704
2.5080	2.4261
3.6800	3.2390

It can be seen that model chosen using BIC has a lower number of parameters included in the final model. (0.000 means that the predictor is not included in the model) For example, both best subset method and lasso using BIC criterion choose a model with 2 predictors. On the other hand, both methods using cv choose a full model with 8 predictors. PCR also can perform variable selection by dimension reduction. It includes 7 predictors at the end. Thus, it can show that all of lasso, best subset and pcr can perform variable selection. Comparing to cross validation error, which aims to achieve the minimum error, BIC has a better ability to select variables. Hence, the model selected by BIC contains less number of predictors in our analysis.

Model chosen by BIC seems to perform better on test data for best subset method in our case. The final result shows that best subset method using BIC has the best performance on test data, with test MSE 1.536.