PredictiveModelComparison

Chase Enzweiler

10/12/2017

```
# packages
library(ElemStatLearn)
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.3.2
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.2
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.3.2
```

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

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

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.2
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 3.3.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
##
## Attaching package: 'caret'
```

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

```
library(ggplot2)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.2
```

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

```
## Warning: package 'Matrix' was built under R version 3.3.2
```

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

```
## Loaded glmnet 2.0-13
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.3.2
```

load the prostate data

```
# prostate data
data <- prostate

# training data
training <- data[data$train == TRUE,-10]

# test data
test <- data[data$train == FALSE, -10]
```

Obtain a matrix of correlation of the predictors

```
# matrix of correlations of predictors, -9 removes lpsa
cor_mat <- cor(training[,-9])
```

Standardize the predictors and confirm summary statistics

```
# standardize the predictors

stand_train <- as.data.frame(scale(training))

# standardized training predictors

stand_train_pred <- stand_train[,-9]

# confirm summary staatistics

summary(stand_train_pred)
```

```
##      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
```

Ordinary Least Squares

Now fit an ordinary least squares regression by regressing lpsa on the rest of the predictors

```
# fit ols model
ols_fit <- lm(training$lpsa ~ ., data = stand_train_pred)

# create a table of the estimates and standard errors

ols_table <- tidy(ols_fit)[,1:3]

ols_table$estimate <- round(ols_table$estimate, digits = 3)
ols_table$std.error <- round(ols_table$std.error, digits = 3)

ols_table
```

```
##          term estimate std.error
## 1 (Intercept)    2.452     0.087
## 2      lcavol    0.716     0.134
## 3     lweight    0.293     0.106
## 4         age   -0.143     0.102
## 5        lbph    0.212     0.103
## 6         svi    0.310     0.125
## 7         lcp   -0.289     0.155
## 8     gleason   -0.021     0.143
## 9       pgg45    0.277     0.160
```

Best Subset Regression

Now find the best subset regression

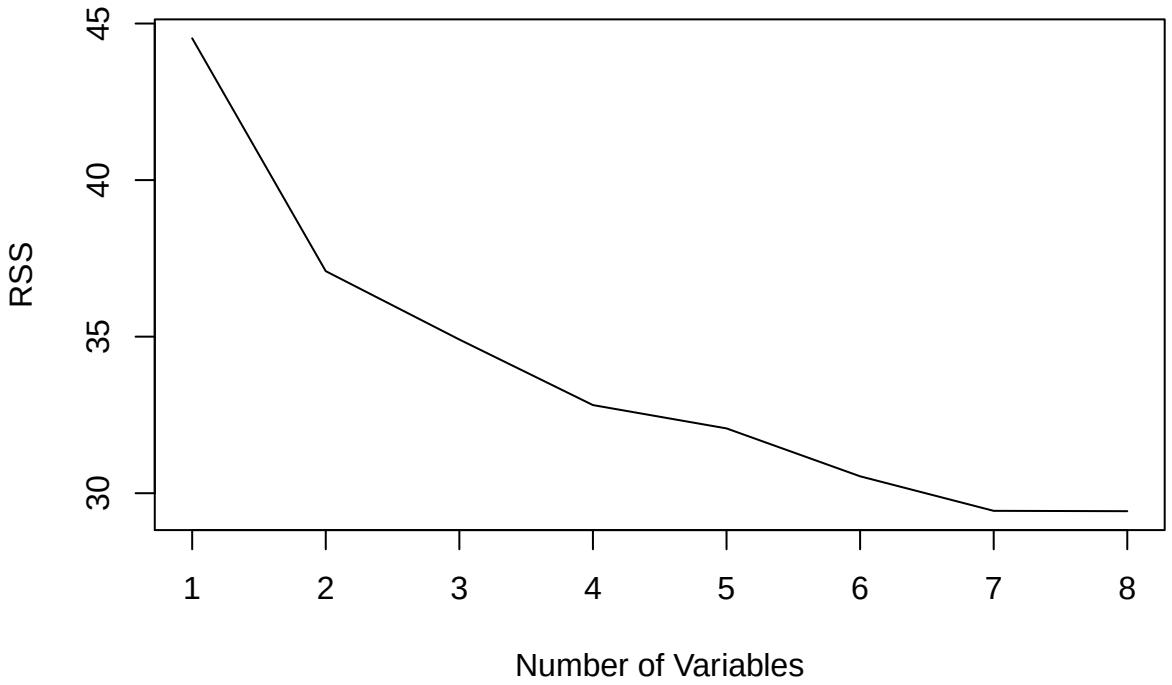
```
# best subset regression using regsubsets() exhaustive

subset_reg <- regsubsets(training$lpsa ~ ., data = stand_train_pred)

subset_reg_summ  <- summary(subset_reg)

subset_reg_summ
```

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

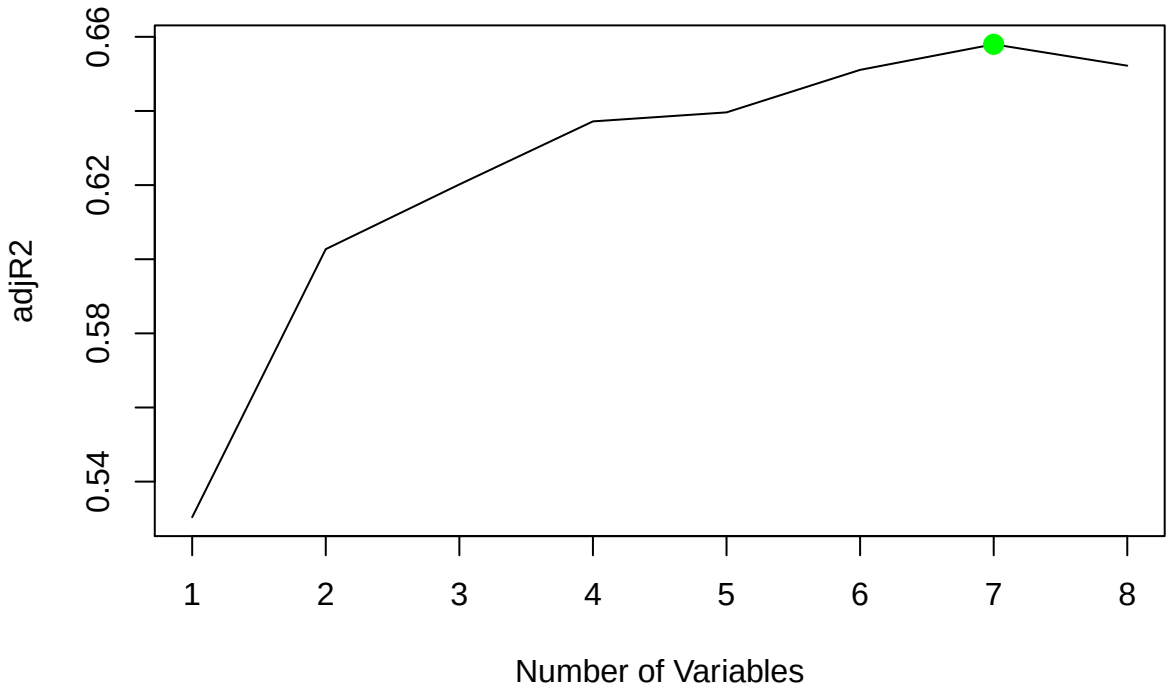
Now to choose the best overall model, we can plot the RSS, adjusted R^2, C_p, and BIC

```
# subset model selection plots

##### code from introduction to statistical learning #####

# fix this with creating plots from ggplot
# remove cross validation part when submitting to homework, just reference the BIC for subset model sel

# create the plots

#par(mfrow = c(2,2))

# can do with ggplot also

# RSS
plot(subset_reg_summ$rss, xlab = "Number of Variables", ylab = "RSS", type = "l")
```

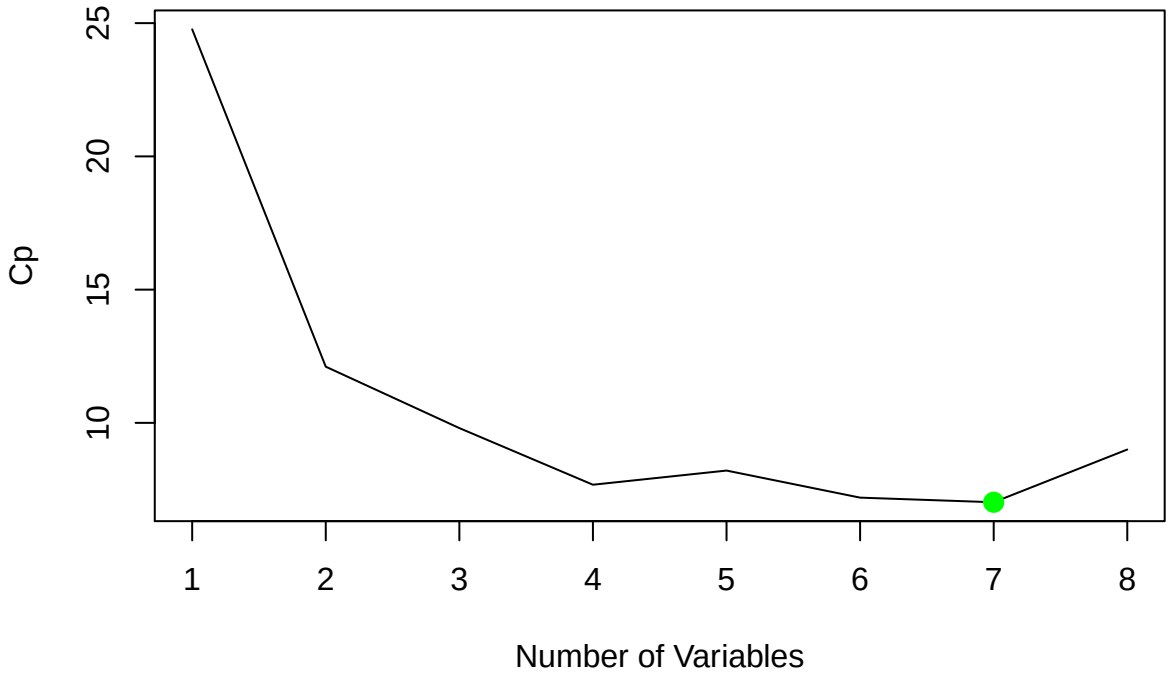



```
# adj R squared

plot(subset_reg_summ$adjr2, xlab = "Number of Variables", ylab = "adjR2", type = "l")
```

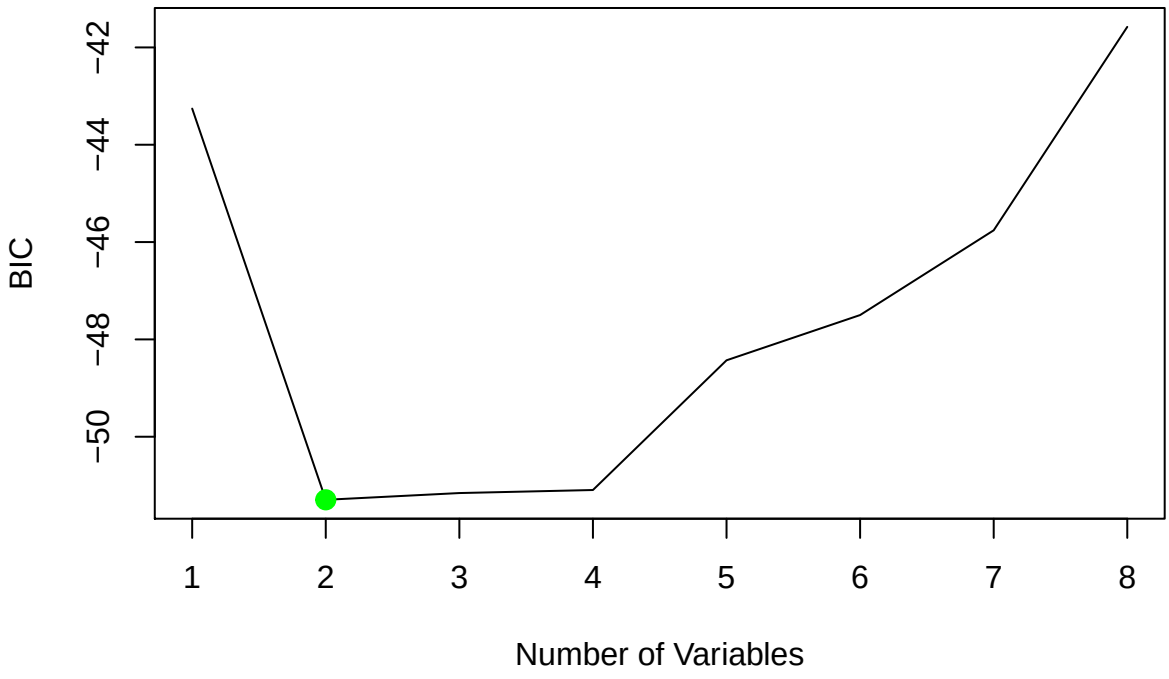
```r
# location of max of adjR2
adjR2_max <- which.max(subset_reg_summ$adjr2)

points(adjR2_max, subset_reg_summ$adjr2[adjR2_max], col = "green", cex = 2, pch = 20)
```




```r
# Cp

plot(subset_reg_summ$cp, xlab = "Number of Variables", ylab = "Cp", type = "l")
#location of min
cp_min <- which.min(subset_reg_summ$cp)
#point for min
points(cp_min, subset_reg_summ$cp[cp_min], col = "green", cex = 2, pch = 20 )
```




```
# BIC

plot(subset_reg_summ$bic, xlab = "Number of Variables", ylab = "BIC", type = "l")

# location of min of BIC
bic_min <- which.min(subset_reg_summ$bic)

#plot min

points(bic_min, subset_reg_summ$bic[bic_min], col = "green", cex = 2, pch = 20)
```



We

can see that the BIC is lowest for the 2 variable model. With subset regression we can also check the test error of each model by validation set approach or cross validation. In the book Elements of Statistical Learning they choose 2 variable model based on the BIC, however i will use cross validation

```
set.seed(1)

# Code written in reference to Introduction to Statistical Learning #
# Code similiar to Lab at end of chapter 6

# Create folds
folds <- sample(1:10 ,size = nrow(training), replace = TRUE )

# matrix of the test errors
test_error <- matrix(NA, ncol = 8, nrow = 10)

# best subset regression

for(i in 1 : 10){

  # vector to hold model test mse
  MSE <- rep(NA, 8)

  # fit each model to training folds
  best_subreg <- regsubsets( training[folds != i, ]$lpsa ~ ., data = stand_train_pred[folds != i, ])

  for (j in 1:8){

    # creates a design matrix for the test folds
    test_design_mat <- model.matrix(training[folds == i, ]$lpsa ~ ., data = stand_train_pred[folds == i

    # coefficients of the fit model fitted with # folds != i
    sub_coef <- coef(best_subreg, j)

    # fitted test values
    # basically X %*% B
    fitted <- test_design_mat[, names(sub_coef)] %*% sub_coef

    # stores mse for each model
    MSE[j] <- mean((training[folds == i,]$lpsa - fitted)^2)

  }

  # stores test mse for each model for each k fold
  test_error[i,] <- MSE
}

# calculates the mean of test mse for each model
avg_test_MSE <- apply(test_error, 2, mean)

# find the model with the lowest mean test mse

which.min(avg_test_MSE)
```

```
## [1] 7
```

From 10 fold cross validation, our optimal best subset regression model is the model with 7 variables.

```
# our best subset regression optimal model
coef(subset_reg, 7)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##   2.4523451   0.7131604   0.2951154  -0.1461421   0.2113905   0.3115400
##         lcp       pgg45
##  -0.2877348   0.2621042
```

PCR and PLSR

Fit a principal component regression model with the training data using ten fold cross validation

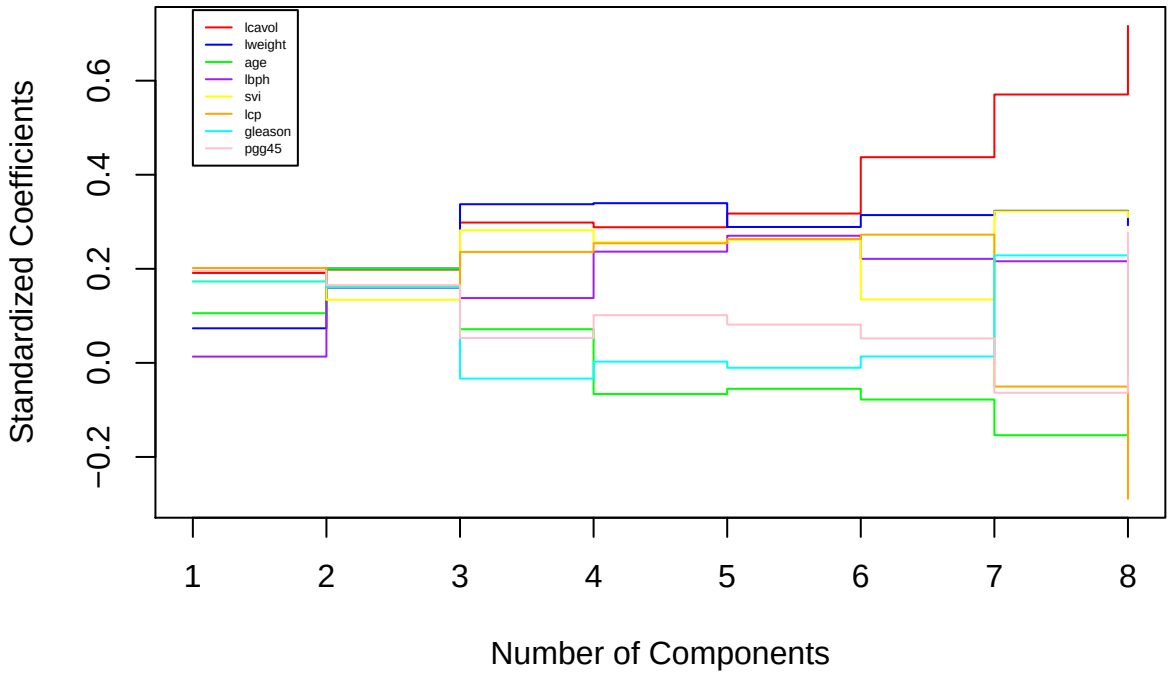
```
# principal component regression

set.seed(1)

# validation = "CV" performs 10 fold cv for each number of components

pcr_fit <- pcr(training$lpsa ~ ., data = stand_train_pred, scale = FALSE, validation = "CV" )

summary(pcr_fit)
```

```
## 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   0.9161   0.8831   0.8078   0.8031   0.8049   0.8196
## adjCV        1.217   0.9143   0.8828   0.8046   0.7999   0.8038   0.8154
##        7 comps  8 comps
## CV      0.7823   0.7541
## adjCV   0.7788   0.7492
##
## TRAINING: % variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                42.83    63.24    76.20    83.92    89.61    94.32
## training$lpsa    45.18    50.84    59.58    61.00    61.17    62.08
##                7 comps  8 comps
## X                97.82   100.00
## training$lpsa    66.36    69.44
```

plot the standardized coefficients against the components

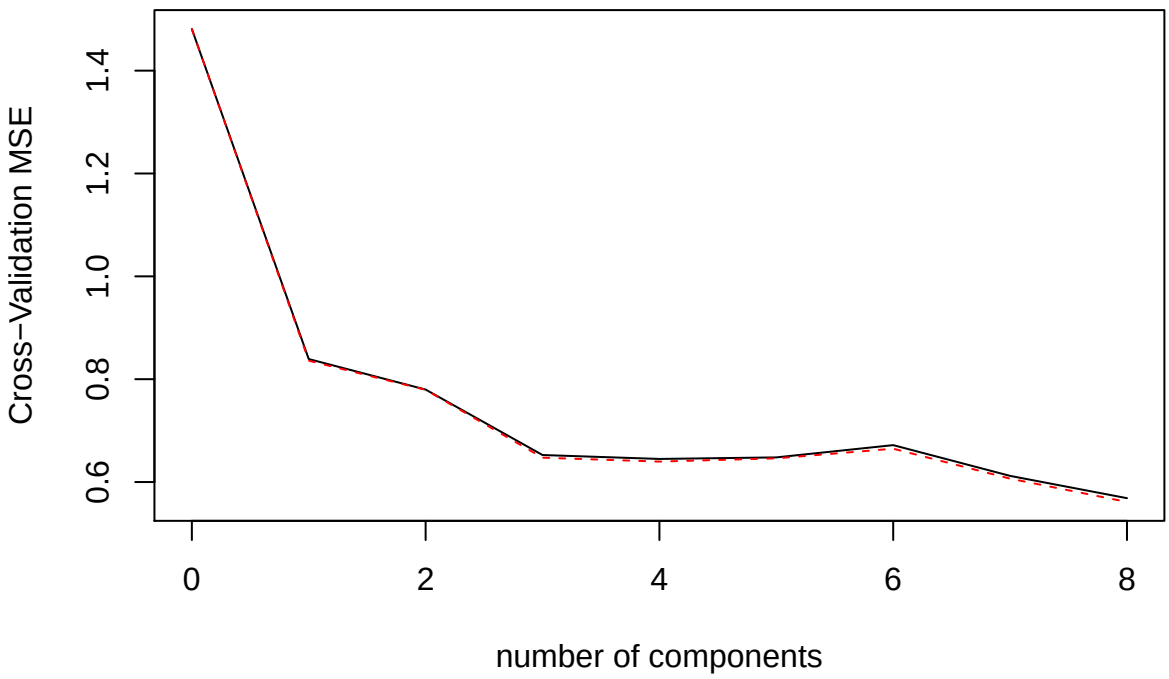
```
# matrix of the coefficients

pcr_coef_matrix <- matrix(pcr_fit$coefficients, nrow = 8, ncol = 8)

rownames(pcr_coef_matrix) <- colnames(stand_train_pred)
```

```r
pcr_coef_matrix <- t(pcr_coef_matrix)

# profiles of coefficients plot

matplot(c(1,2,3,4,5,6,7,8), pcr_coef_matrix[,1:8], type = "s", lty = 1, col = c("red", "blue", "green",

legend(1,.75, colnames(pcr_coef_matrix), cex = .45 , lty=c(1,1), lwd=c(1,1), col=c("red", "blue", "green
```



Now we can plot the cross validation MSE against the number of components to determine the optimal number of components to have in our model.

```r
# plot cross validation mse against the number of components

validationplot(pcr_fit, val.type = "MSEP", ylab = "Cross-Validation MSE")
```



From our validation plot we see that our optimal principal component regression model includes 8 components which is the same as our ordinary least squares model.

```
# optimal principal component regression model

# coefficients of optimal model
pcr_fit$coefficients[, , 8]
```

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

```
# number of components used in optimal model
pcr_fit$ncomp
```

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

Now we fit a model to the training data using Partial Least Squares Regression and using 10 fold cross-validation to find an optimal model.

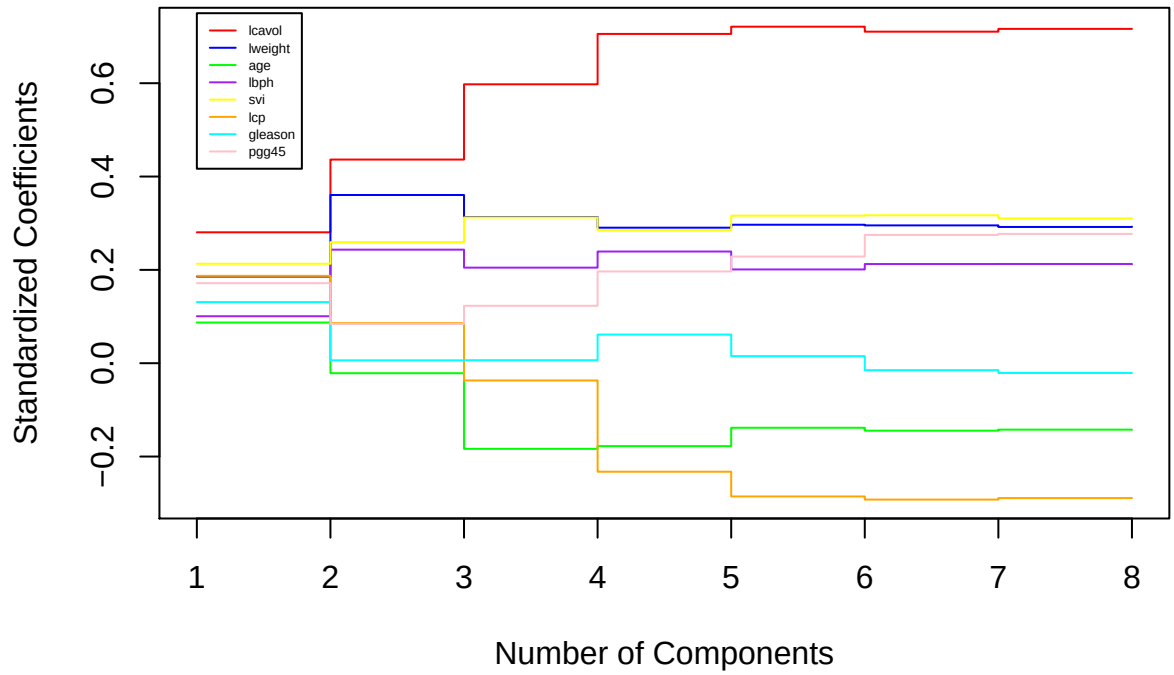
```
# partial least squares regression
set.seed(1)

plsr_fit <- plsr(training$lpsa ~ ., data = stand_train_pred, scale = FALSE, validation = "CV")

summary(plsr_fit)
```

```
## Data:    X dimension: 67 8
##  Y dimension: 67 1
## Fit method: kernelpls
## Number of components considered: 8
##
```

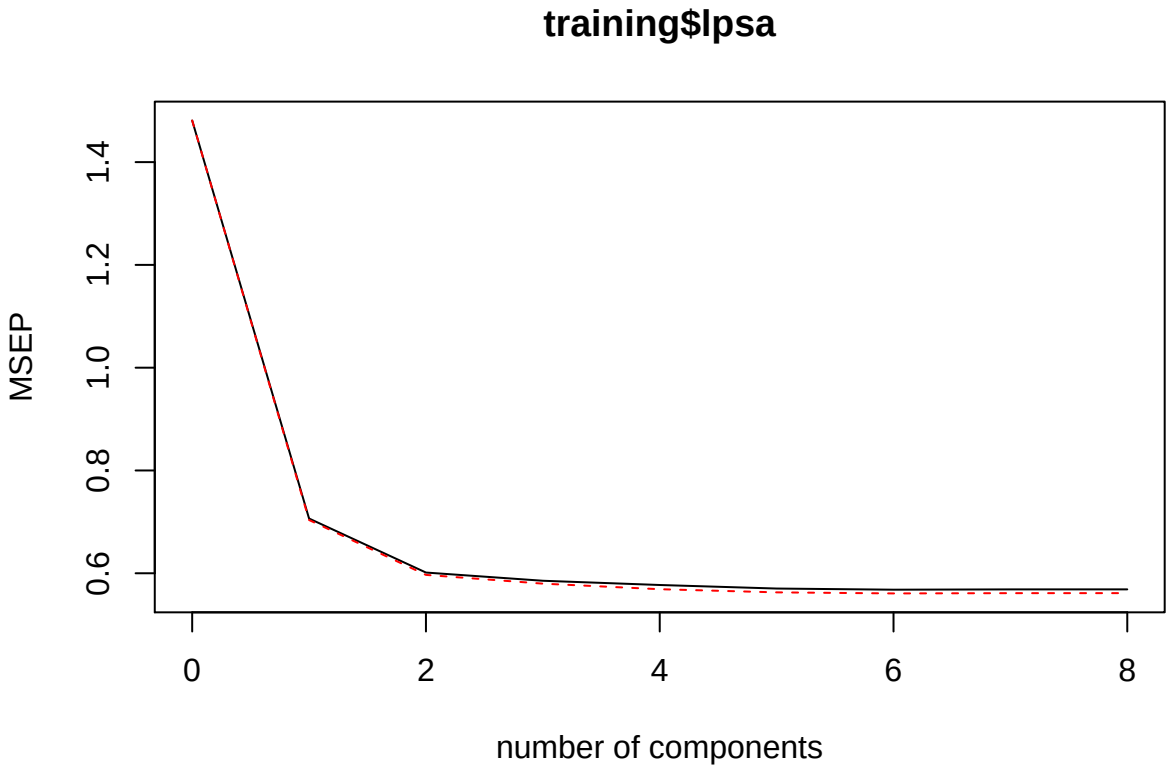
```
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.217   0.8406   0.7755   0.7651   0.7597   0.7552   0.7537
## adjCV        1.217   0.8388   0.7726   0.7615   0.7543   0.7502   0.7489
##        7 comps  8 comps
## CV      0.7541   0.7541
## adjCV   0.7492   0.7492
##
## TRAINING: % variance explained
##                 1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                 41.64    58.29    71.13    79.75    86.08    90.21
## training$lpsa     55.79    64.60    67.51    69.12    69.37    69.43
##                 7 comps  8 comps
## X                 94.70   100.00
## training$lpsa     69.44    69.44
```

now we can plot the standardized coefficients against the number of components and plot the Cross validation MSE against the number of components

```
# plot of profile of coefficients

#matrix of coefficients

plsr_coef_matrix <- matrix(plsr_fit$coefficients, ncol = 8, nrow = 8)

rownames(plsr_coef_matrix) <- colnames(stand_train_pred)

plsr_coef_matrix <- t(plsr_coef_matrix)

# time to plot similiar to pcr
matplot(c(1,2,3,4,5,6,7,8), plsr_coef_matrix[,1:8], type = "s", lty = 1, col = c("red", "blue", "green"

legend(1,.75, colnames(plsr_coef_matrix), cex = .45 , lty=c(1,1), lwd=c(1,1), col=c("red", "blue", "gre
```




```
# now plot the MSE against the components
validationplot(plsr_fit, val.type = "MSEP")
```



From looking at our plot and mainly observing our root mse from our summary of our partial least squares function we see that our cross validation rmse is the lowest when the model uses 6 components and therefore our optimal partial least squares model is the model that utilizes 6 components.

```
# our optimal partial least squares model
```

```
plsr_fit$coefficients[, , 6]
```

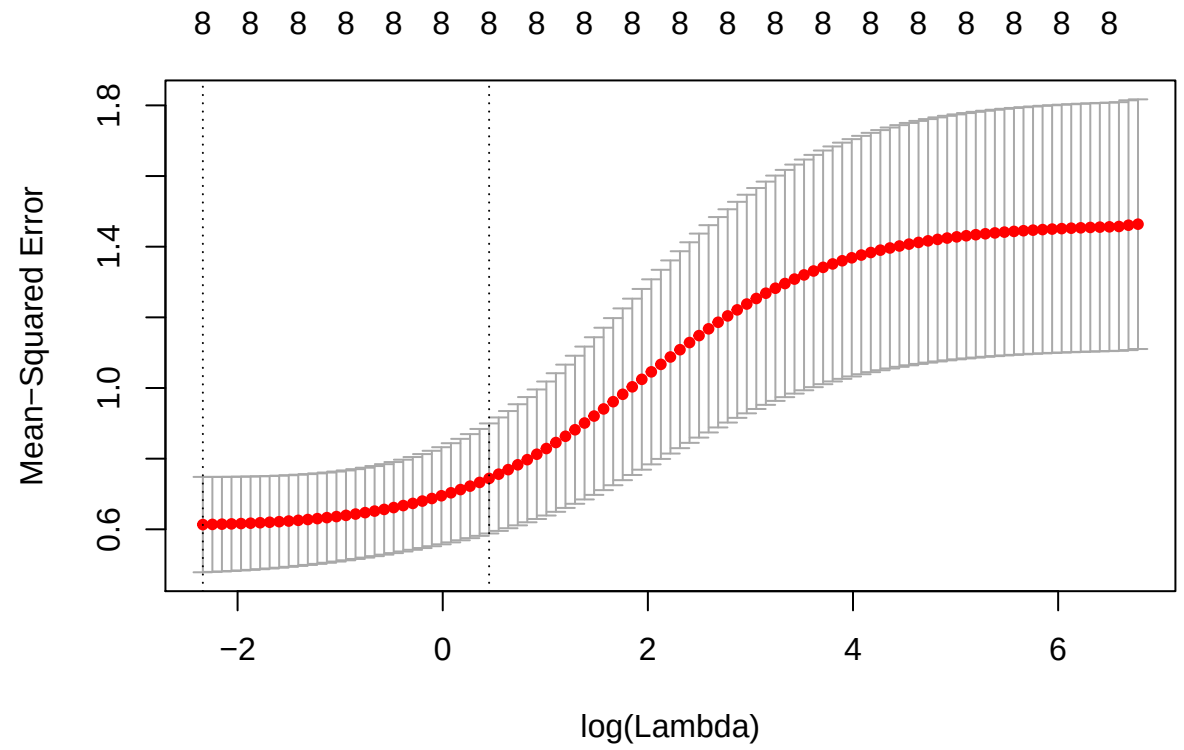
```
##     lcavol    lweight        age       lbph        svi        lcp
##  0.7104094  0.2952801 -0.1446106  0.2124677  0.3169434 -0.2922292
##    gleason      pgg45
## -0.0149234  0.2748280
```

Above are the coefficients of our partial least squares optimal model that uses 6 components.

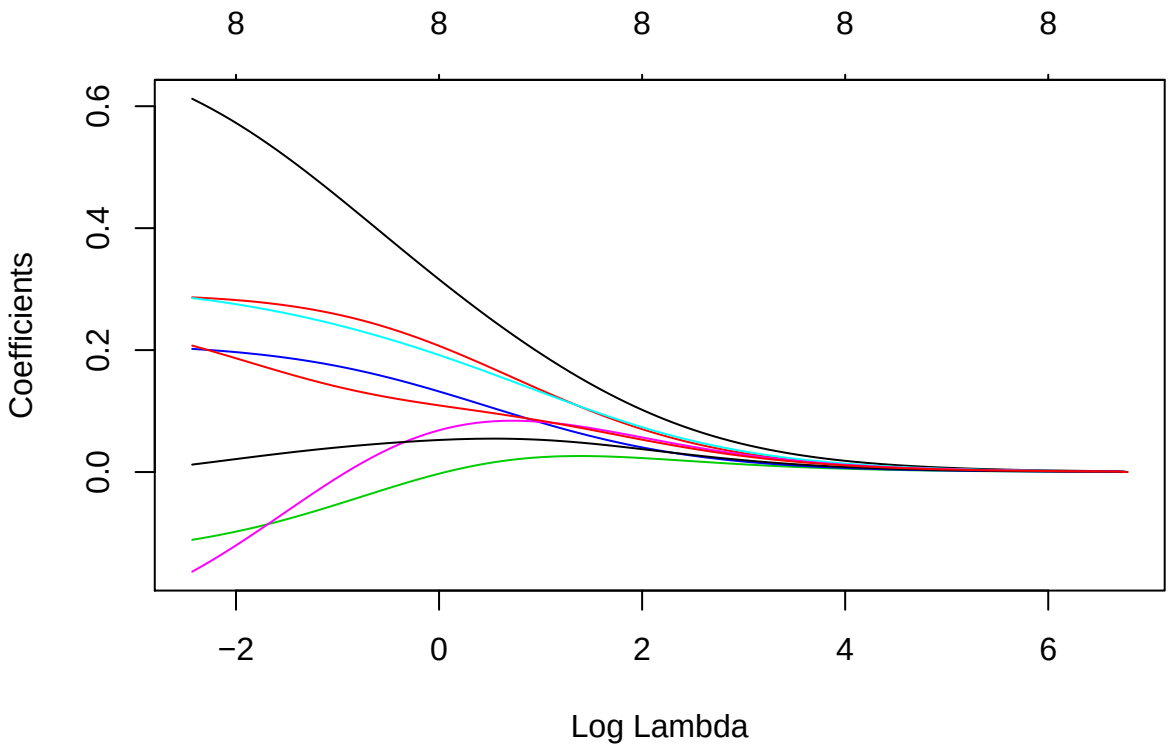
Ridge Regression and Lasso

Now we can fit a Ridge Regression model using cv.glmnet from package "glmnet".

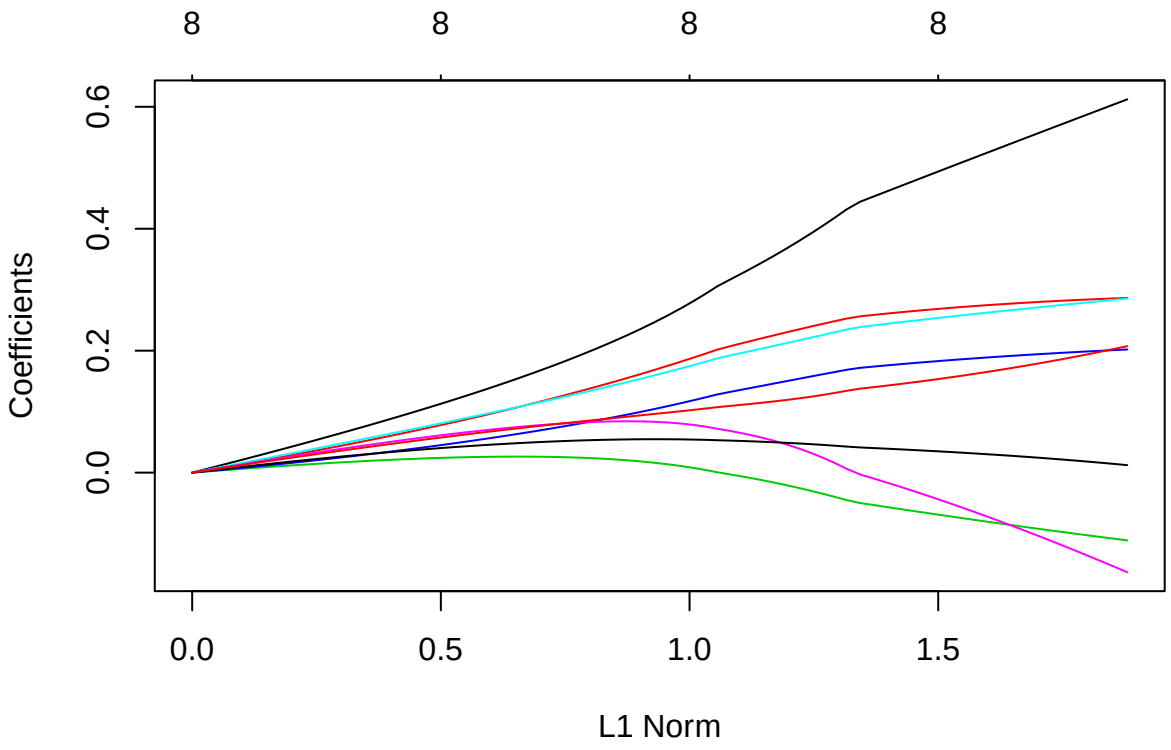
```
# fit a ridge regression model
set.seed(1)

# create standardized design matrix to run in glmnet

stand_train_pred_mat <- as.matrix(stand_train_pred)

# 10 fold cross validation to find best lambda

cv_rr <- cv.glmnet(stand_train_pred_mat, training$lpsa, alpha = 0)

# lambda with smallest cross validation error

rr_lamda_best <- cv_rr$lambda.min

# fit the ridge regression model with the best lambda tuning parameter

rr_fit <- glmnet(stand_train_pred_mat, training$lpsa, alpha = 0, lambda = rr_lamda_best)

# plot of Cross Validation Mean squared error

plot(cv_rr)
```




```
# plot of profiles of coefficients using lambda and l1 norm

plot(glmnet(stand_train_pred_mat, training$lpsa, alpha = 0), xvar = "lambda")
```

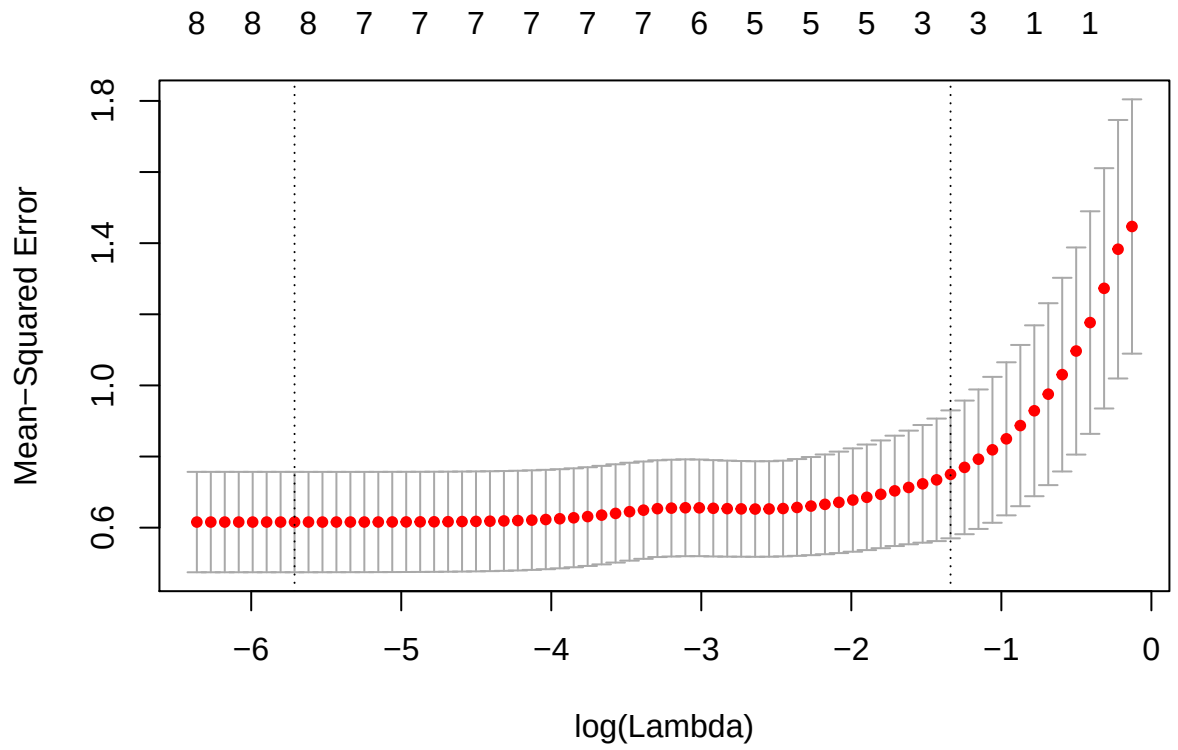



```
plot(glmnet(stand_train_pred_mat, training$lpsa, alpha = 0))
```

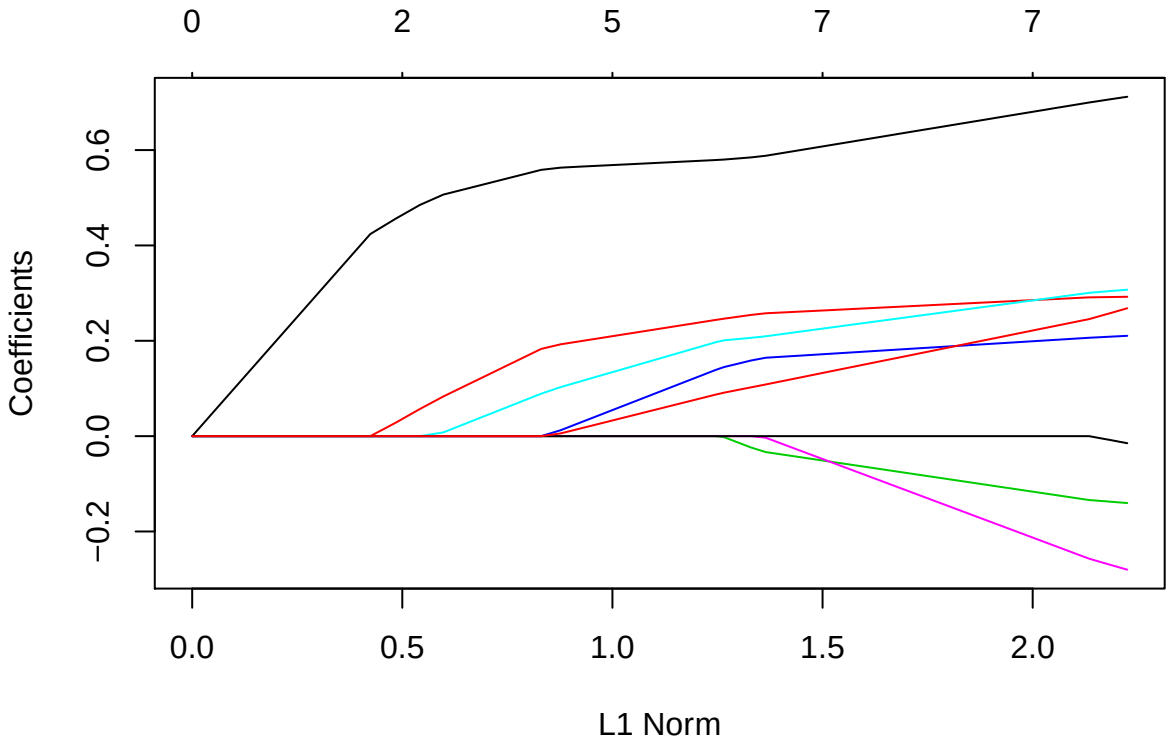


Now the Lasso

```
# the Lasso
set.seed(1)
# find the lambda with the best cv mse for a tuning parameter

cv_lasso <- cv.glmnet(stand_train_pred_mat, training$lpsa, alpha = 1)

# best tuning parameter

lasso_lambda_best <- cv_lasso$lambda.min

# fit the lasso model with best best tuning paramter

lasso_fit <- glmnet(stand_train_pred_mat, training$lpsa, alpha = 1, lambda = lasso_lambda_best)

# plot of cross validation mse

plot(cv_lasso)
```




```
# plot of the profile of coefficients # use without best lambda

plot(glmnet(stand_train_pred_mat, training$lpsa, alpha = 1))
```



Therefore our optimal Ridges Regression model with tuning paramter is

```
#optimal ridge regression
```

```
coef(rr_fit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  2.45234509
## lcavol       0.60438730
## lweight      0.28573832
## age         -0.10855978
## lbph         0.20098549
## svi          0.28347129
## lcp         -0.15474777
## gleason      0.01410086
## pgg45        0.20306133
```

```
# best tuning parameter

rr_lamda_best
```

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

Our optimal model for the Lasso with tuning paramter is

```
# optimal lasso model

coef(lasso_fit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  2.452345085
## lcavol       0.705015407
## lweight      0.291655879
## age         -0.136684230
## lbph         0.208045668
## svi          0.303287346
## lcp         -0.267204216
## gleason     -0.006769133
## pgg45        0.255746826
```

```
# best tuning parameter

lasso_lambda_best
```

```
## [1] 0.003308928
```

Model Selection

The final stage of the predictive modeling cycle consists of selecting the best model among the candidates obtained in the model assessment stages. That is, you should have six candidate models: OLS, best subset, PCR, PLSR, RR, and Lasso. The next step is to determine the best model using the test set. This means computing test MSE for each candidate model, and selecting the one with the smallest test MSE Use our test set to get predictions of lpsa and calculate the test mse. First find the test mse for our ordinary least squares model

```
# compute test MSE for ols model
```

```
# standardize the test data
stand_test <- as.data.frame(scale(test))

# number of observations in the test data
n <- dim(test)[1]

# predicted response values
lm_predicted <- predict(ols_fit, newdata = stand_test)

# test mse
lm_test_mse <- sum((test$lpsa - lm_predicted)^2) / n

lm_test_mse
```

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

find the test mse for the subset regression model of 7 variables

```
# compute test mse for subset regression model

# prediction for our ols with 7
subset_predicted <- predict(lm(training$lpsa ~. -gleason, data = stand_train_pred), newdata = stand_tes

# test mse

subset_test_mse <- sum((test$lpsa - subset_predicted)^2) / n

subset_test_mse
```

```
## [1] 0.5459418
```

test mse for the principal component regression

```
# pcr test mse

pcr_predicted <- predict(pcr_fit, newdata = stand_test, ncomp = 8)

# test mse

pcr_test_mse <- sum((test$lpsa - pcr_predicted)^2) / n

pcr_test_mse
```

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

calculate the test mse for the partial least squares regression model

```
# plsr test mse

plsr_predicted <- predict(plsr_fit, newdata = stand_test, ncomp = 6 )

# test mse

plsr_test_mse <- mean((test$lpsa - plsr_predicted)^2)

plsr_test_mse
```

```
## [1] 0.5493153
```

Calculate the test mse for the ridge regression model

```
# ridge regression test mse

rr_predicted <- predict(rr_fit, newx = as.matrix(stand_test)[,-9], s = rr_lamda_best)

# test mse

rr_test_mse <- mean((test$lpsa - rr_predicted)^2)

rr_test_mse
```

```
## [1] 0.517176
```

Calculate the test mse for the lasso model

```
# test mse for lasso

lasso_predicted <- predict(lasso_fit, s = lasso_lambda_best, newx = as.matrix(stand_test)[,-9])

# test mse

lasso_test_mse <- mean((test$lpsa - lasso_predicted)^2)

lasso_test_mse
```

```
## [1] 0.5401353
```

	LS	Best Subset	Ridge	Lasso	PCR	PLSR
Intercept	2.4523	2.4523	2.4523	2.4523	0.0000	0.0000
lcavol	0.7164	0.7132	0.6044	0.7050	0.7164	0.7104
lweight	0.2926	0.2951	0.2857	0.2917	0.2926	0.2953
age	-0.1425	-0.1461	-0.1086	-0.1367	-0.1425	-0.1446
lbph	0.2120	0.2114	0.2010	0.2080	0.2120	0.2125
svi	0.3096	0.3115	0.2835	0.3033	0.3096	0.3169
lcp	-0.2890	-0.2877	-0.1547	-0.2672	-0.2890	-0.2922
gleason	-0.0209	0.0000	0.0141	-0.0068	-0.0209	-0.0149
pgg45	0.2773	0.2621	0.2031	0.2557	0.2773	0.2748
test mse	0.5492	0.5459	0.5172	0.5401	0.5492	0.5493