P9120 - hw 1 sj2921 Shan Jiang

1. To prove the Bridge(β) function is convex, we need to use the first and second order characterization of convex function:

First we seperate the Bridge function into two parts:

Let $F(\beta) = (y - X\beta)^T(y - X\beta)$ be the first part,

$$G(\beta) = \lambda \sum_{j=1}^{p} |\beta_j|^q = \sum_{j=1}^{p} |\beta_j|_q^q$$

Expand $F(\beta) = y^Ty - y^TX\beta - \beta^TX^Ty + \beta^TX^TX\beta$

$$\Rightarrow \quad \frac{\partial F(\beta)}{\partial \beta} = -X^Ty - X^Ty + 2X^TX\beta = -2X^Ty + 2X^TX\beta$$

$$\Rightarrow \quad \frac{\partial F(\beta)}{\partial \beta^2} = 2X^TX \geq 0 \Rightarrow \text{It's a positive semi-definite matrix.} \Rightarrow F(\beta) \text{ is convex.}$$

By definition: $F(\beta)$ is convex since second-order $\nabla^2 F(\beta) \geq 0$ for all $\beta \in dom(f) \Leftrightarrow F(\beta)$ is convex.

Then we turn to $G(\beta)$, for any vector β_1, β_2.

$\|t\beta_1 + (1-t)\beta_2\|_q \leq t\|\beta_1\|_q + (1-t)\|\beta_2\|_q \quad (\forall t \in [0,1])$ [Minkovski Inequality]

By definition $|\beta|_q$ is convex if $q \geq 1$

So $G(\beta) = \lambda|\beta|_q^q$ is then rewritten as.

$\lambda\|t\beta_1 + (1-t)\beta_2\|_q^q \leq \lambda t\|\beta_1\|_q^q + \lambda(1-t)\|\beta_2\|_q^q$, thus, we derive that

$G(\beta)$ is also convex if $\lambda > 0$ and $q \geq 1$, since ($m^q \geq n^q$ for $m, n > 0$ and $q \geq 1$)

Thus, $Bridge_\lambda(\beta) = H(\beta) + G(\beta)$ is a convex function when $\lambda > 0$ and $q \geq 1$

Besides, for $q \neq 1$, the equal sign cannot be attained as below shown:

$$\lambda\|t\beta_1 + (1-t)\beta_2\|_q^q < \lambda t\|\beta_1\|_q^q + \lambda(1-t)\|\beta_2\|_q^q$$

So, we conclude that $Bridge_\lambda(\beta) = H(\beta) + G(\beta)$ is strictly convex when $\lambda > 0$, $q > 1$

To sum up: Bridge(β) is convex when $\lambda > 0$, $p \geq 1$.

Bridge(β) is strictly convex when $\lambda > 0$, $p > 1$.

(b) From Question (a), we know that:

① The bridge function is a convex function in β, and the second-order derivative proofs to be positive, So the minimum over the domain (f) can be attained. thus, the bridge function has at least one solution, there is minimizer for the function.

② However, to prove its uniqueness, we need to prove: if the is another β_m, Bridge(β_m) = Bridge(β_n) in which m, n allows $\arg\min$ Bridge(β), By definition of strictly convex:

For $t \in (0,1)$,

$$\text{Bridge}(t\beta_m + (1-t)\beta_n) < t\,\text{Bridge}(\beta_m) + (1-t)\,\text{Bridge}(\beta_n)$$
$$= t\,\text{Bridge}(\beta_m) + \text{Bridge}(\beta_n) - t\,\text{Bridge}(\beta_n)$$
$$= \text{Bridge}(\beta_n)$$

Since β_n is set to attain the min value, the above function can not hold, so the minimizer should be unique.

For $\sum_{j=1}^{p} |\hat{\beta}_j(\lambda)|^q \le t_0$, we are assuming $\tilde{\beta}(\lambda)$ is the minimizer for $F(x) = (y - X\beta)^T(y - X\beta)$

So, for Bridge minimizer $\hat{\beta}(\lambda) \in \text{dom}(f)$:

$$(y - X\tilde{\beta}(\lambda))^T(y - X\tilde{\beta}(\lambda)) + \|\tilde{\beta}(\lambda)\|_q^q \le (y - X\hat{\beta}(\lambda))^T(y - X\hat{\beta}(\lambda)) + \|\hat{\beta}(\lambda)\|_q^q$$

(Assume $\|\tilde{\beta}(\lambda)\|_q^q \le \|\hat{\beta}(\lambda)\|_q^q$)

Then the equation suggests that $\hat{\beta}(\lambda)$ could not be the minimizer of Bridge(β) as there is another $\tilde{\beta}(\lambda)$ appears to have lower value.

So, the part of $\sum_{j=1}^{p} |\hat{\beta}_j(\lambda)|^q \le t_0$.

(c) The Bridge function is proved to be convex;

Suppose there're 2 minimizers $\hat{\beta}_1(\lambda)$ and $\hat{\beta}_2(\lambda)$.

for $\forall\, t \in (0,1)$ M denotes the min value obtained by $\hat{\beta}_1(\lambda)$ and $\hat{\beta}_2(\lambda)$ under the LASSO regression.

$$\|y - X[t\hat{\beta}_1(\lambda) + (1-t)\hat{\beta}_2(\lambda)]\|_2^2 + \lambda\|t\hat{\beta}_1(\lambda) + (1-t)\hat{\beta}_2(\lambda)\| < tM + (1-t)\cdot M = M.$$

where the equal sign cannot be achieved as the function is strict convex.

this is contradict to the Assumption M is the minimum value as a lower value was attained.

Thus, two minimizers should be equal and the panelty function should take the same value.

$$S(\lambda) \triangleq \sum_{j=1}^{p} |\hat{\beta}_j(\lambda)|^q$$

While for proving $S(\lambda) \le t_0$, let $\hat{\beta}(\lambda)$ be the minimizer of Bridge function and $\tilde{\beta}(\lambda)$ be the minimizer of $F(\beta) = (y - X\beta)^T(y - X\beta)$

Suppose $\|\hat{\beta}(\lambda)\| > \|\tilde{\beta}(\lambda)\|$ holds,

$$\text{Bridge}_\lambda(\tilde{\beta}) = (y - X\tilde{\beta}(\lambda))^T(y - X\tilde{\beta}(\lambda)) + \|\tilde{\beta}(\lambda)\|$$
$$< (y - X\hat{\beta}(\lambda))^T(y - X\hat{\beta}(\lambda)) + \|\hat{\beta}(\lambda)\|$$

Cause $\hat{\beta}(\lambda)$ cannot be the unique minimizer, so there's some contradiction.

$$\Rightarrow S(\lambda) \triangleq \sum_{j=1}^{p} |\hat{\beta}_j(\lambda)|^q \le t_0$$

(d) Since $G(\beta)$ is a convex function,

so, $G(\beta)$ has at least one subgradient at

every point in relint dom G, also G is differentiable $\nabla G(a)$ is subgradient of G at a.

we can define vector h as a subgradient at any β, and it satisfies that:

$$g(\beta') \geq g(\beta) + h^T(g(\beta') - g(\beta)) \quad \text{for } \forall\, \beta \in \text{dom}(\beta)$$

hence: $g(\beta') \leq g(\beta) \Rightarrow h^T(g(\beta') - g(\beta)) \leq 0$, $\partial g(\beta)$ denotes the subdifferential that includes all possible subgradients of g at β.

① To prove Condition in (d) $\Rightarrow$ Condidition in (C)

The lagrangian form of P_2:

$$L(\beta, r) = (y - X\beta)^T(y - X\beta) + r(\|\beta\|_q^q - s(\lambda))$$

$\Rightarrow$ the lagrangian condition: $0 \in -2X^Ty + 2X^TX\beta + r\partial G(\beta)$

② To prove Condition in (C) $\Rightarrow$ Condition in (d).

Due to the constraints specified, we checked:

for $(y - X\beta)^T(y - X\beta)$ subject to $\sum_{j=1}^{p} |\beta_j|^q \leq s(\lambda)$ ... (1)

a. $\|\beta\|_q^q - s(\lambda) \leq 0$ holds as β can be all zeros.

b. When $\lambda = r$, $\beta = \beta(r)$, the complementary condition is satisfied at solution point β.

To conclude, to minimize Bridge(β) in (c) is equvalent to that minimizing $(y - X\beta)^T(y - X\beta)$ with $\sum_{j=1}^{p} |\beta_j|^q \leq s(\lambda)$.

P9120 Homework #2, #3, #4
UNI: sj2921

2. We perform best subset, forward stepwise, and backward stepwise selection on a single data set. For each approach, we obtain p + 1 models, containing 0, 1, 2, . . . , p predictors. Explain your answers:

   (a) Which of the three models with k predictors has the smallest training RSS?

   The smallest training RSS will be the model selected with the best subset approach, as this model have considered all models with predictor number ranging from 0 to P, so final model will be chosen with k parameters for best subset, the other two methods would not have such small RSS.

   (b) Which of the three models with k predictors has the smallest test RSS?

   The model with best subset approach will select a best model based on **training** error**, but it does not mean a necessary** smallest **test error**, so for any given data, the smallest test RSS can appear in either three models: best-subset model, forward stepwise and backward stepwise approach.

   (c) True or False:

   i. The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the (k+1)-variable model identified by forward stepwise selection.

      True, by definition, the forward stepwise adds variable once at a time and the K+1 variable should include all elements in K variable model, so the K variable model is a subset of (K+1) variable model.

   ii. The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the (k + 1) variable model identified by backward stepwise selection.

      True, also by definition, the backward algorithm reduces one variable at a time based on the original K+1 model, so the K-variable model should be a subset of the (K+1) variable model.

   iii. The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the (k + 1) variable model identified by forward stepwise selection.

      False, there is no direct connection between the forward selection algorithm and backward selection algorithm, so we have no rationale to infer from the K-variable model in backward method to K+1 model in forward stepwise method.

   iv. The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the (k+1)-variable model identified by backward stepwise selection.

      False, there is no direct connection between the forward selection algorithm and backward selection algorithm, so we have no rationale to infer from the K-variable model in forward stepwise method to K+1 model in backward method.

   v. The predictors in the k-variable model identified by best subset are a subset of the predictors in the (k + 1)-variable model identified by best subset selection.

No, since for each given N choose K, we can select more than 1 different models, and then pick up the best from the best subset method, so the K-variable model is not necessarily the subset of the (K+1)-variable model.

3. Derive the entries in Table 3.4, the explicit forms for estimators in the orthogonal case.

TABLE 3.4. *Estimators of β_j in the case of orthonormal columns of* $\mathbf{X}$. *M and λ are constants chosen by the corresponding techniques;* sign *denotes the sign of its argument (± 1), and x_+ denotes "positive part" of x. Below the table, estimators are shown by broken red lines. The 45° line in gray shows the unrestricted estimate for reference.*

Estimator	Formula
Best subset (size M)	$\hat{\beta}_j \cdot I(\lvert\hat{\beta}_j\rvert \geq \lvert\hat{\beta}_{(M)}\rvert)$
Ridge	$\hat{\beta}_j/(1+\lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)(\lvert\hat{\beta}_j\rvert - \lambda)_+$

1. From the *Least Square method*, we can derive the formula:

$$\hat{\beta}^{ls} = (\mathbf{X}^{\mathrm{T}}\,\mathbf{X})^{-1}\,\mathbf{X}^{\mathrm{T}}\,\mathrm{y} = \mathbf{X}^{\mathrm{T}}\,\mathrm{y}$$

2. For *Best-subset*, we get the equation:

$$\hat{\beta}^{ls} = \hat{\beta}^{BS}(M = P) = \mathbf{X}^{\mathrm{T}}\,\mathrm{y}$$

where the coefficients are identical even if we take $M \leq p$ since the design matrix is orthogonal;

3. For *ridge regression* estimates:

$$\beta^{ridge} = (\mathbf{X}^{\mathrm{T}}\,\mathbf{X} + \lambda\,\boldsymbol{I})^{-1}\,\mathbf{X}^{\mathrm{T}}\,\mathrm{y} = (\mathbf{I} + \lambda\,\mathbf{I})^{-1}\mathbf{X}^{T}\,y = \hat{\beta}^{ls}\,/\,(1+\lambda)$$

4. For *lasso regression*, we get the following:

$$L(\beta) = (\,y - \mathbf{X}\beta)^{\mathrm{T}}\,(y - \mathbf{X}\beta) + \lambda\,|\beta|$$

The **first order derivative** is:

$$\frac{\partial L(\beta)}{\partial(\beta)} = -\mathbf{X}^{\mathbf{T}}\mathrm{y} + \mathbf{X}^{\mathrm{T}}\,\mathbf{X}\,\beta + \lambda\,sign(\beta)$$

Setting the gradient as 0, we get that the solution with respect to β is:

$$\beta^{Lasso} = \mathbf{I}(\,\mathbf{X}^{\mathbf{T}}y - \lambda\,sign(\beta)) = sign(\beta)\,(\mathbf{X}^{\mathbf{T}}y - \lambda)$$ Q.E.D.

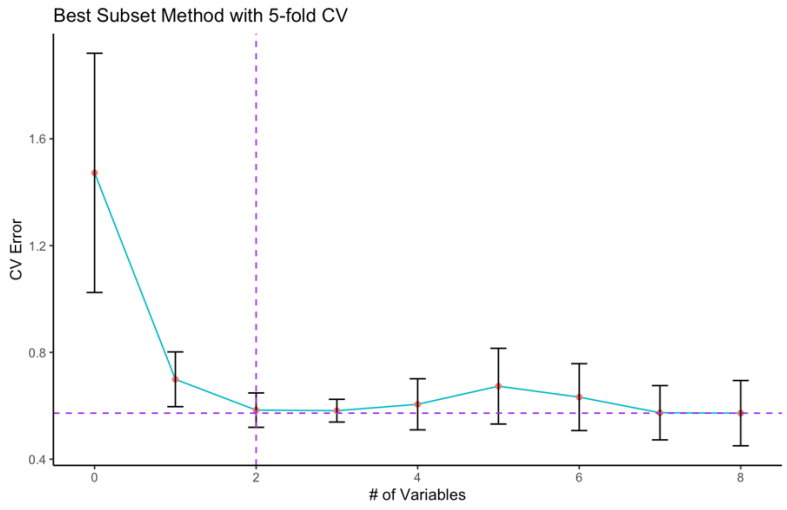
4. Table summary:

Table 1. Estimated coefficients and test error results, for different subset and shrinkage methods applied to the prostate data.

Term	Best Subset	LASSO with 5-fold CV	LASSO with BIC	PCR with 5-fold CV	LS
(Intercept)	2.4773	2.4686	2.4551	2.455	2.465
lcavol	0.7397	0.5259	0.0871	0.286	0.68
lweight	0.3163	0.1622		0.3391	0.263
age				0.0562	-0.141
lbph				0.1015	0.21
svi		0.0602		0.2614	0.305
lcp				0.2187	-0.288
gleason				-0.016	-0.021
pgg45				0.0617	0.267
Test Error	0.4713	0.5151	0.9472	0.5513	0.521
Std Error	0.0434	0.1762	0.3591	0.1328	0.179

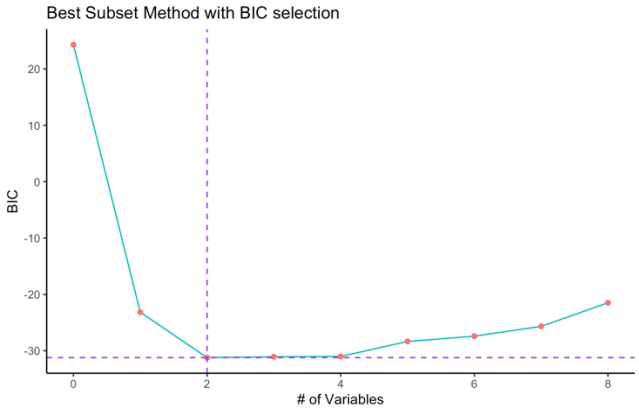
Figure output:

(a). The best-subset method produced with 5-fold cross-validation shown below just chose two variables in the model, and has a relatively small test error with small std.dev.



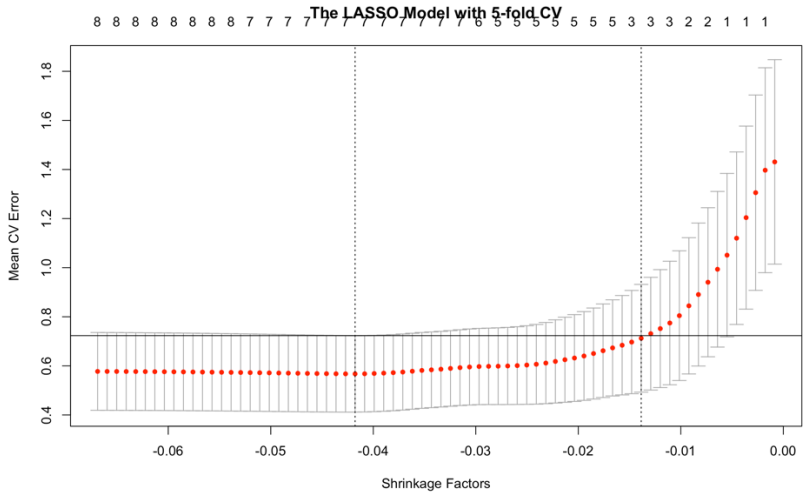
(b) **Best-subset with BIC**

This method selects variables “lcavol” and “lweight” into the model, obtained the lowest BIC value, compared with Best-subset with 5-fold CV, these two have the same model selected and the test error and the standard error of test error are the lowest.



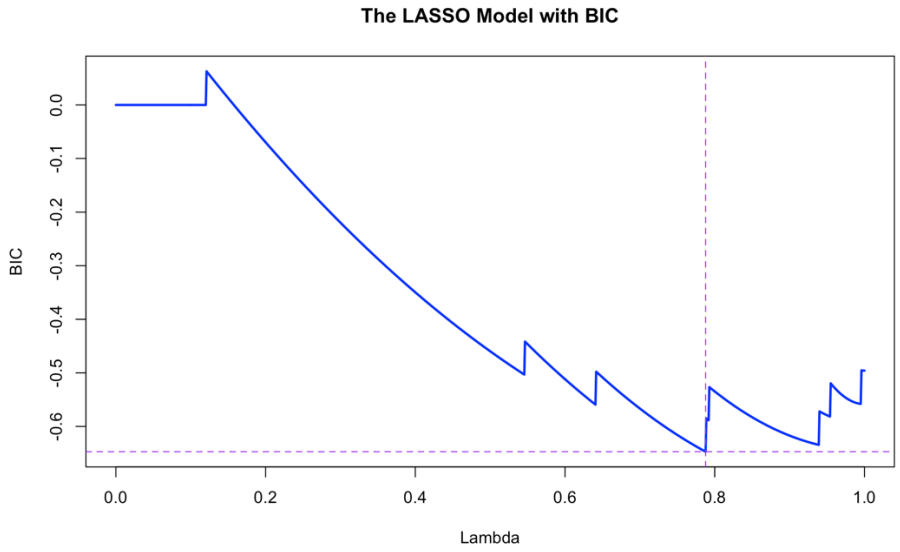
(c). **LASSO with 5-fold CV**

In the LASSO with 5-fold CV, we select from multiple lambdas, and found the best one is λ = 0.250016 provides the best CV error under the one-standard deviation rule. This method selects three variables “lcavol”, “lweight” and “svi”.



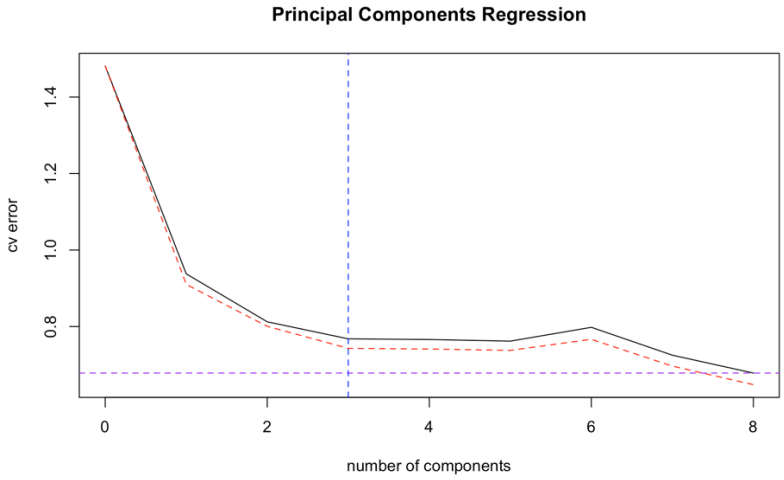
d. LASSO with BIC

We find $\lambda = 0.7877$ provides the lowest BIC value. This method selects only variable "lcavol" into the model. LASSO with BIC criteria for model fitting has the largest test error among all the models.



e. **PCR** with 5-fold cross-validation

Principle component regression presents three factors provide the best CV error under the one-standard deviation rule. The fitted coefficients in model PCR shown below has selected model with all 8 variables.



Data Mining Homework 1

Shan Jiang

09/27/2019

Contents

Data Preparation

```
library(tidyverse)
library(leaps)
library(lattice)
library(caret)
library(glmnet)
library(pls)
require(ggplot2)
```

Prostate data:

Predictors (columns 1–8)

- lcavol
- lweight
- age
- lbph
- svi
- lcp
- gleason
- pgg45

outcome (column 9): lpsa

train/test indicator (column 10)

This last column indicates which 67 observations were used as the `training set` and which 30 as the test set, as described on page 48 in the book.

The features must first be scaled to have mean zero and variance 96 (=n) before the analyses in Tables 3.1 and beyond.

That is, if x is the 96 by 8 matrix of features, we compute xp <- scale(x,TRUE,TRUE)

Import data and Cleaning

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

```
## [1] 97
```

```
## sampling segments
library(bestglm)
k = 5
set.seed(1818)
folds = sample(1:k, nrow(df), replace = TRUE)

## Set up the train and test data
testdata = modeldf0 %>%
  filter(train == FALSE) %>%
  select(-train)

traindata = modeldf0 %>%
  filter(train == TRUE) %>%
  select(-train)

# standardization of predictors
prostate = df %>%
  dplyr::select(-X)

trainst <- traindata

for(i in 1:8) {
  trainst[,i] <- trainst[,i] - mean(prostate[,i]);
  trainst[,i] <- trainst[,i]/sd(prostate[,i]);
}

testst <- testdata
```

```
for(i in 1:8) {
  testst[,i] <- testst[,i] - mean(prostate[,i]);
  testst[,i] <- testst[,i]/sd(prostate[,i]);
}
```

Data Analysis

For the Analysis below, all the data have been standardized already.

(a) Best-subset linear regression with k chosen by 5-fold cross-validation.

```
set.seed(21)
## Fit model: with k = 5
res.bestcv = bestglm(trainst,
                     IC= "CV",
                     CVArgs = list(Method="HTF", K=5, REP=1))

## Extract the cv.error and sd. for cv.error
res.bestcv$Subsets$CV
```

```
## [1] 1.4723738 0.6992092 0.5837649 0.5817422 0.6055277 0.6734276 0.6324876
## [8] 0.5739700 0.5724480
```

```
res.bestcv$Subsets$sdCV
```

```
## [1] 0.44804318 0.10253172 0.06436491 0.04261650 0.09563564 0.14180478
## [7] 0.12532319 0.10165354 0.12227593
```

Correlation Coefficients

```
## Show top model
res.bestcv
```

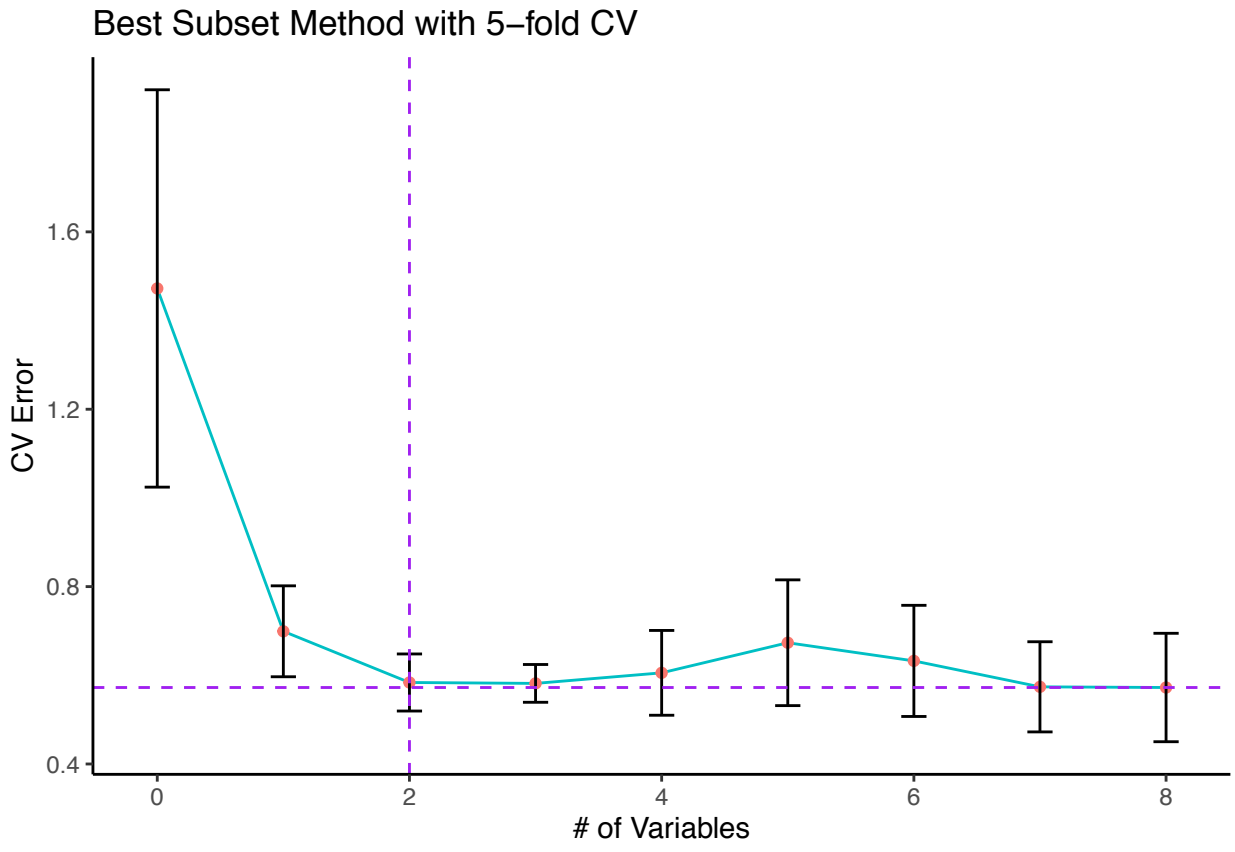
```
## CV(K = 5, REP = 1)
## BICq equivalent for q in (0.0176493852011195, 0.512566675362627)
## Best Model:
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.4773573 0.09304738 26.624687 2.475214e-36
## lcavol      0.7397137 0.09318316  7.938277 4.141615e-11
## lweight     0.3163282 0.08830716  3.582135 6.576173e-04
```

```
coef(res.bestcv$BestModel)
```

```
## (Intercept)      lcavol     lweight
##   2.4773573   0.7397137   0.3163282
```

graph of the CV error

```
v = c(0:8)
error1 = res.bestcv$Subsets$CV
sd1 = res.bestcv$Subsets$sdCV
fit1 = data.frame(cbind(v, error1, sd1))
ggplot(fit1, aes(x = c(0:8),
                 y = error1
                       )) +
  geom_line(aes(col = '#E69F00')) +
  geom_point(aes(col = '#999999'))+
  geom_errorbar(aes(ymin = error1 - sd1,
                    ymax = error1 + sd1),
                width = 0.2,
                position = position_dodge(0.05)) +
   labs(title="Best Subset Method with 5-fold CV",
      x="# of Variables", y = "CV Error")+
   theme_classic() +
   theme(legend.position="none") +
    geom_hline(yintercept= min(error1),
               linetype="dashed", color = "purple") +
    geom_vline(xintercept= 2,
               linetype="dashed", color = "purple")
```



Fitting the trained model on the test data: Test Error

```r
## Yhat: predicted value on the testdata
yh <- as.matrix(predict
                (res.bestcv$BestModel,newx = as.matrix(testst[,1:8]) ))

### test error
te1 = sum(lm(lpsa~lcavol + lweight, testst)$residuals^2)/30
te1
```

```
## [1] 0.4713028
```

```r
## estimating Std Error
sd1 = sum(sqrt(mean(((yh[1:30] - testst[,9])^2))))/nrow(testst)
sd1
```

```
## [1] 0.04339654
```

We see that cross-validation selects an 1-variable model, with CV error experienced a large plunge from No variable was fitted to just 1 variable fit in the model.

(b) Best-subset linear regression with k chosen by BIC.

```r
set.seed(1)
reg.bestbic = bestglm(trainst,
                      IC="BIC")

## Model summary coefficients
summary(reg.bestbic)
```

```
## Fitting algorithm:  BIC-leaps
## Best Model:
##            df deviance
## Null Model 64 37.09185
## Full Model 66 96.28145
##
##  likelihood-ratio test - GLM
##
## data:  H0: Null Model vs. H1: Best Fit BIC-leaps
## X = 59.19, df = 2, p-value = 1.403e-13
```

- When the BIC criteria is chosen, No. of variables equal to 2 are selected in the model for `lcavol` and `svi`, with an intercept.
- From the Left plot of Rss: we find it always decrease with the increse of No. of variables, while for adjusted square, it starts to drop after No. of variables exceeds `which.max(reg.summary1$adjr2)`.

Correlation Coefficients

```r
reg.bestbic$BestModel$coefficients
```

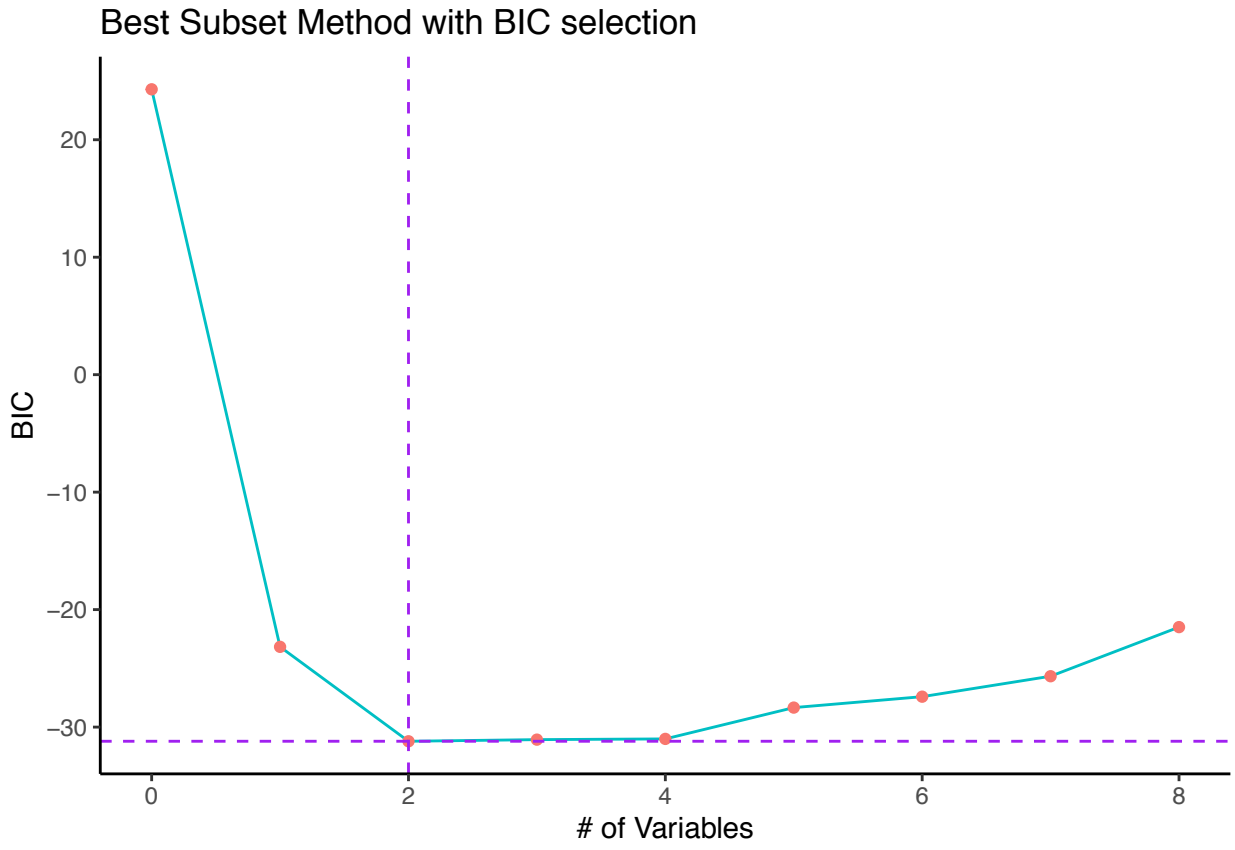
```
## (Intercept)      lcavol     lweight
##   2.4773573   0.7397137   0.3163282
```

- Based on BIC criteria, we should choose a model with 2 variables: `lcavol` and `svi`.

Plots

```
error2 = reg.bestbic$Subsets$BIC
fit2 = data.frame(cbind(v, error2))

ggplot(fit2, aes(x = c(0:8),
                 y = error2
                       )) +
  geom_line(aes( col = '#E69F00')) +
  geom_point(aes(col = '#999999')) +
   labs(title="Best Subset Method with BIC selection",
      x="# of Variables", y = "BIC")+
   theme_classic() +
   theme(legend.position="none") +
    geom_hline(yintercept= min(error2),
               linetype="dashed", color = "purple") +
    geom_vline(xintercept= 2,
               linetype="dashed", color = "purple")
```




```
### BIC:
BIC(lm(lpsa~lcavol+lweight, testst))

## [1] 76.17347

### test error
## Yhat: predicted value on the testdata
```

```
yh2 <- as.matrix(predict
                 (reg.bestbic$BestModel,newx = as.matrix(testst[,1:8]) ))

### test error
te2 = sum(lm(lpsa~lcavol + lweight, testst)$residuals^2)/30
te2
```

```
## [1] 0.4713028
```

```
## estimating Std Error
sd2 = sum(sqrt(mean(((yh2[1:30] - testst[,9])^2))))/nrow(testst)
sd2
```

```
## [1] 0.04339654
```

- While for BIC, it starts to increase after No. of variables exceeds `which.min(reg.summary1$bic)`, so it is consistent with the conclusion that the chosen best model is with 2 variables.
- Also the CV and BIC fitted best-subset models are the same, share the same model parameters, test error, and SD of test error.

(c) Lasso regression with λ chosen by 5-fold cross-validation.

```
# Duplicate part of Figure 3.7 (the lasso regularization results)
## use glmnet in package glmnet
library(glmnet)
# use 5-fold cross-validation to choose best lambda
set.seed(18)

cv.out = cv.glmnet(x = as.matrix(trainst[,1:8]),
                   y= as.numeric(trainst[,9]),
                   nfolds = 5,
                   alpha = 1, # alpha = 1 => lasso
                   standardize = F) ## has standardized

# the best lambda chosen by 5-fold cross-validation
lambda.5fold = cv.out$lambda.1se
```

select the suitable parameter λ

```
# get the optimal value of lambda:
# apply Lasso with chosen lambda
fitlasso = glmnet(x =  as.matrix(trainst[,1:8]),
                  y=   as.numeric(trainst[,9]),
                  alpha = 1,
                  lambda = lambda.5fold,
                  standardize= F,
                  thresh = 1e-12)
```

Fitted coefficients

If we try to refit with this opimal value of lambda, and we get the coefficients est.

```
## note that they are not the same as Table 3.3 on page 63 due to the chosen labmda and fitting algorit
# Intercept
fitlasso$a0
```

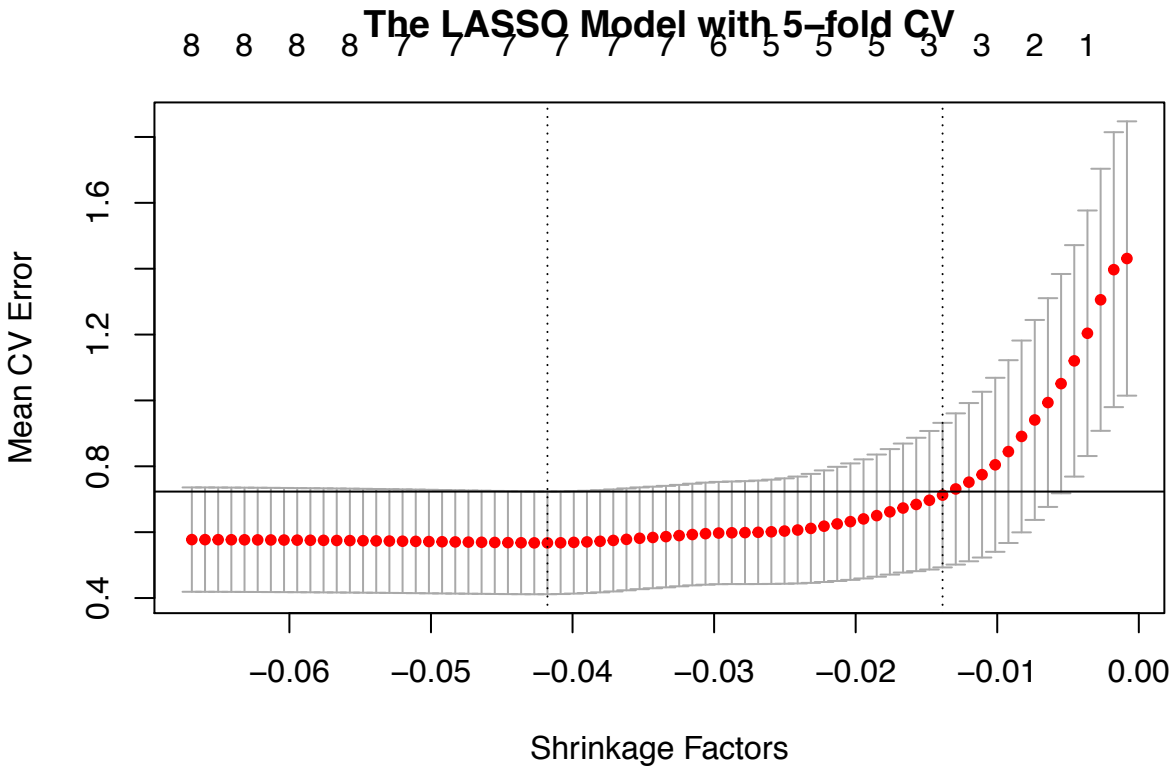
```
##       s0
## 2.468639
```

```
# beta
fitlasso$beta
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                 s0
## lcavol  0.52589103
## lweight 0.16216336
## age     .
## lbph    .
## svi     0.06015292
## lcp     .
## gleason .
## pgg45   .
```

Graph of the Lasso Parameters

```
plot(cv.out,
       xlab="Shrinkage Factors",
       main = "The LASSO Model with 5-fold CV",
       ylab="Mean CV Error", 0.01)
abline(h=cv.out$cvup[which.min(cv.out$cvm)])
```



ModelEvaluation

```
# estimating mean prediction error
test.lasso = predict(fitlasso,
                     newx=as.matrix(teststst[,1:8]))

# mean (absolute) prediction error
mean(abs(testdata[,9]-test.lasso))

## [1] 0.5150874

# mean (squared) prediction error
mean((testdata[,9]-test.lasso)^2)

## [1] 0.5040247

# standard error of mean (squared) prediction error
sd((testdata[,9]-test.lasso)^2)/sqrt(30)

## [1] 0.1761895
```

The model fit with 4 variables with 5-fold cv in LASSO. This method select variables “lcavol”, “lweight”, “lbph” and “svi” into the model.

From this graph above, we found that the lowest MSE appears to be minimum when No. of log(shrinkage factor) falls, $\lambda = 0.2500116$ provides the best CV error using one-standard deviation rule.

(d) Lasso regression with λ chosen by BIC

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 $\lambda = 1010$ to $\lambda = 10^{-2}$, essentially covering the full range of scenarios from the null model containing only the intercept, to the least squares fit.

Method 1: using the `ic.glmnet` package;

```
set.seed(1)
library(HDeconometrics)

## == LASSO == ##
# (1) Fit lasso model on training data
lasso_bic = ic.glmnet(
                    x = as.matrix(trainst[,1:8]),
                    y = as.numeric(trainst[,9]),
                    crit = "bic")

## Draw plot of coefficients
plot(lasso_bic$glmnet,"lambda", ylim=c(-2,2))
plot(lasso_bic)

## BIC selected model coefficients
c = lasso_bic$coefficients
c

## Tuning parameter
lasso_bic$lambda


# (2) Fit lasso model on test data
## == Forecasting == ##
pred.lasso = predict(lasso_bic, newdata = testst[,1:8])
plot(testst[,9], type="l")
lines(pred.lasso, col=2)
error = sqrt(mean((testst[,9]-pred.lasso)^2))
```

Method 2: using the `glmnet` package with BIC self-calculation

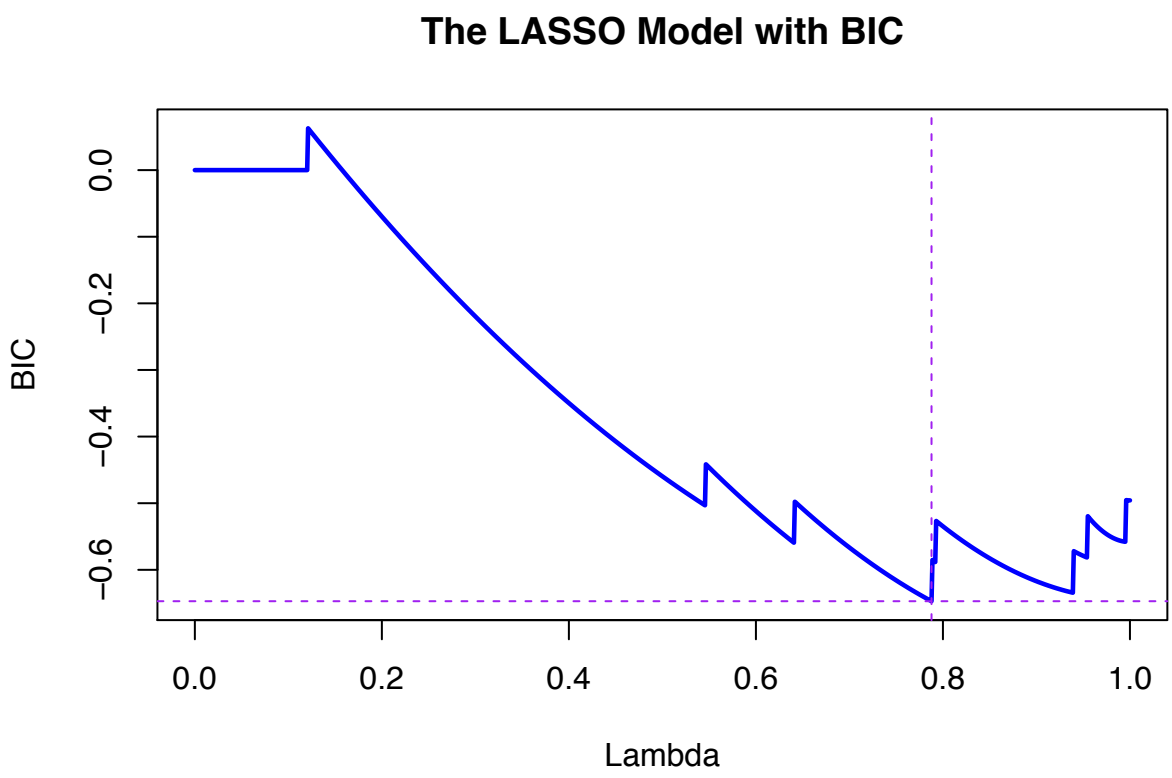
```
set.seed(221)
## == LASSO == ##
# Fit lasso model on training data
lassobic_fit <- glmnet(
              x = as.matrix(trainst[,1:8]),
               y = as.numeric(trainst[,9]),
               lambda = seq(0, 1, length.out = 1000))

bic <- (- lassobic_fit$nulldev +
          deviance(lassobic_fit) +
          log(lassobic_fit$nobs)* lassobic_fit$df)/lassobic_fit$nobs
```

```r
## Draw plot of coefficients
plot(seq(0,1,length.out = 1000), bic, type="l",
     col = "blue",
     main = "The LASSO Model with BIC",
     lwd= 2.3,
     xlab = "Lambda", ylab = "BIC")

abline(h = min(bic),
       lty = 2,
       family="Symbol",
       col = "purple")

abline(v = seq(0,1,length.out = 1000)[min(which(bic==min(bic)))],
       lty =2, col = "purple")
```



choosing tuning paramter

```r
### Best lambda chosen
bestlam = seq(0,1,length.out = 1000)[min(which(bic == min(bic)))]
bestlam
```

```
## [1] 0.7877878
```

```
### Coefficients Estimates
bic_est = coef(glmnet(as.matrix(trainst[,1:8]), trainst[,9],
          lambda = bestlam))
bic_est
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                      s0
## (Intercept) 2.45504238
## lcavol      0.08705551
## lweight     .
## age         .
## lbph        .
## svi         .
## lcp         .
## gleason     .
## pgg45       .
```

```
##### Now we can calculate the forecast: Test error
test_error4 = sum((model.matrix(~., (testst[,1:8])) %*%
        coef(glmnet(as.matrix(trainst[,1:8]), trainst[,9],
            lambda = bestlam)) - testst[,9])^2)/30
test_error4
```

```
## [1] 0.9472603
```

```
# standard error of mean (squared) prediction error
sd4 = sd((model.matrix(~., (testst[,1:8])) %*%
        coef(glmnet(as.matrix(trainst[,1:8]), trainst[,9],
            lambda = bestlam)) - testst[,9])^2)/sqrt(30)
sd4
```

```
## [1] 0.3590992
```

- The Second plot shows the BIC curve and the selected model, the lowest lambda value is specified after the 1000 iteration in the model, and it was specified as `bestlam`.
- Then we fit the model to the testdata and calculated the test error.

(e) Principle component regression with q chosen by 5-fold cross-validation

```
# REG_CHOICE: 1 (does principal component regression),
set.seed(8)

#### Principal Components Regression
pcr.fit = pcr(lpsa~.,
              data = trainst,
              scale = F,
              validation = "CV",
              ## Segments for 5-fold randomised cross-validation:
              segments = 5)

## Find the best number of components,
## regenerating part of Figure 3.7 on page 62
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 5 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           1.217   0.9684   0.9013   0.8763   0.8753   0.8728   0.8933
## adjCV        1.217   0.9540   0.8947   0.8620   0.8610   0.8589   0.8756
##        7 comps  8 comps
## CV      0.8516   0.8237
## adjCV   0.8347   0.8051
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       42.34    63.46    76.26    83.72    89.79    94.53    97.87
## lpsa    47.23    53.13    59.83    61.17    61.89    62.66    66.60
##       8 comps
## X      100.00
## lpsa    69.44
```

```
validationplot(pcr.fit,
               cex = 1,
               main = "Principal Components Regression",
               val.type="MSEP",
               xlab = "number of components",
               ylab = "cv error" )
```

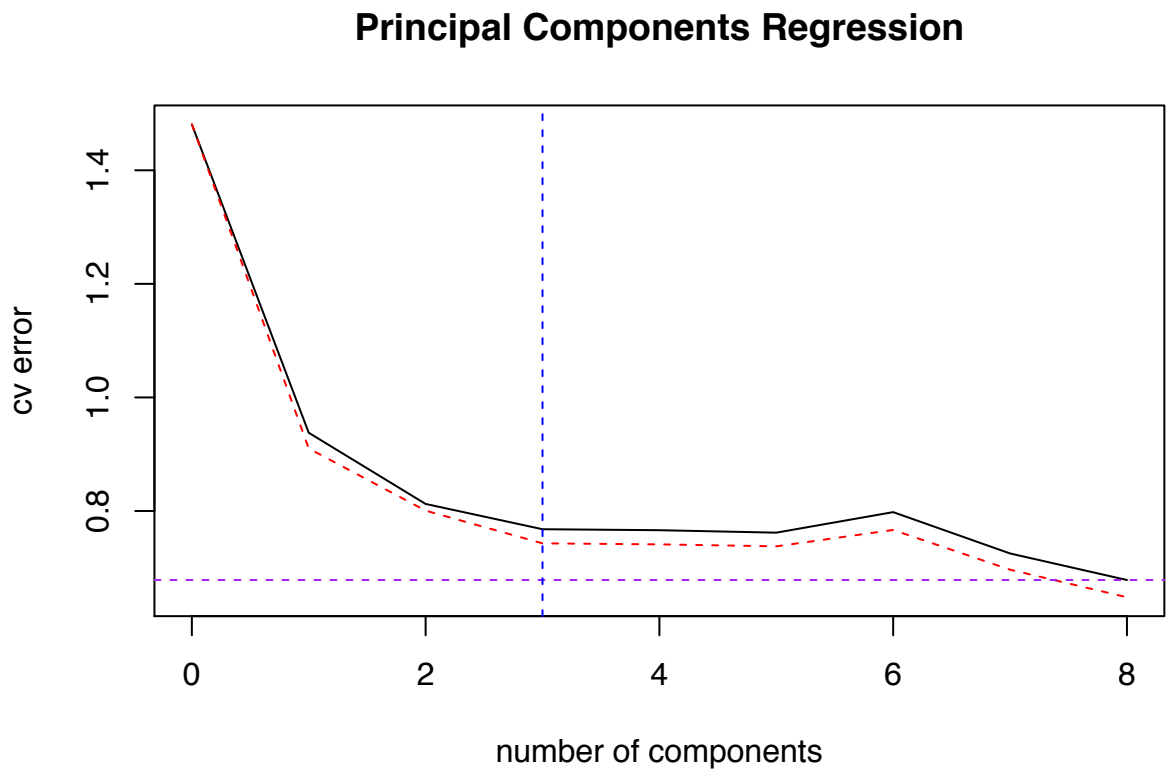
```
itemp = which.min(pcr.fit$validation$PRESS)     # 8
itemp.mean = pcr.fit$validation$PRESS[itemp]/67
```

```
mean((pcr.fit$validation$pred[,,itemp]- trainst[,9])^2)
```

```
## [1] 0.6784573
```

```
itemp.sd = sd((pcr.fit$validation$pred[,,itemp]-
                 trainst[,9])^2)/sqrt(67)
```

```
abline(h= itemp.mean, lty=2, col="purple")
k.pcr = min((1:pcr.fit$validation$ncomp)[pcr.fit$validation$PRESS/67 < itemp.mean + itemp.sd])  # the c
abline(v = k.pcr, lty=2, col="blue")
```



Coefficients summary with intercept

```
### Coefficients summary with intercept
B <- coef(pcr.fit, ncomp = k.pcr, intercept = TRUE)
B
```

```
## , , 3 comps
##
##                    lpsa
## (Intercept)  2.45502159
## lcavol       0.28666128
## lweight      0.33910369
## age          0.05628529
## lbph         0.10152838
## svi          0.26148505
## lcp          0.21868062
## gleason     -0.01605594
## pgg45        0.06170971
```

fitted model with chosen number of components

```
# estimating mean prediction error
test.pcr = predict(pcr.fit, as.matrix(testst[,1:8]),
                   ncomp = k.pcr)
```

```r
# mean (absolute) prediction error
te5 = mean(abs(testst[,9]- test.pcr))
te5
```

```
## [1] 0.5513026
```

```r
# mean (squared) prediction error
mean((testst[,9]-test.pcr)^2)
```

```
## [1] 0.4956846
```

```r
# standard error of mean (squared) prediction error
se5 = sd((testst[,9]-test.pcr)^2)/sqrt(30)
se5
```

```
## [1] 0.1328556
```

5-fold cross-validation for principle component regression presents three factors provide the best CV error under the one-standard deviation rule. The fitted coefficients in model PCR shown above has selected model with all 8 variables, test error = 0.5513026 , se(test.error) = 0.1328556