HW_5_ssingh478

R Markdown

Chapter 10, Exercise 1

Use the prostate data with lpsa as the response and the other variables as predictors. Implement the following variable selection methods to determine the “best” model.

```
library('faraway')

data(prostate,package='faraway')
lmod<-lm(lpsa~.,data=prostate)
lmod <- update(lmod, . ~ . -gleason)
```

(a) Backward elimination

First removing ‘gleason’ predictor–>lcp—>pgg45—>age—>lbph.

```
summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
##     pgg45, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73117 -0.38137 -0.01728  0.43364  1.63513
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.953926   0.829439   1.150  0.25319
## lcavol       0.591615   0.086001   6.879 8.07e-10 ***
## lweight      0.448292   0.167771   2.672  0.00897 **
## age         -0.019336   0.011066  -1.747  0.08402 .
## lbph         0.107671   0.058108   1.853  0.06720 .
## svi          0.757734   0.241282   3.140  0.00229 **
## lcp         -0.104482   0.090478  -1.155  0.25127
## pgg45        0.005318   0.003433   1.549  0.12488
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7048 on 89 degrees of freedom
## Multiple R-squared:  0.6544, Adjusted R-squared:  0.6273
## F-statistic: 24.08 on 7 and 89 DF,  p-value: < 2.2e-16
```

```
lmod<-update(lmod,.~.-gleason)

summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
##     pgg45, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.73117 -0.38137 -0.01728  0.43364  1.63513
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.953926   0.829439   1.150  0.25319
## lcavol       0.591615   0.086001   6.879 8.07e-10 ***
## lweight      0.448292   0.167771   2.672  0.00897 **
## age         -0.019336   0.011066  -1.747  0.08402 .
## lbph         0.107671   0.058108   1.853  0.06720 .
## svi          0.757734   0.241282   3.140  0.00229 **
## lcp         -0.104482   0.090478  -1.155  0.25127
## pgg45        0.005318   0.003433   1.549  0.12488
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7048 on 89 degrees of freedom
## Multiple R-squared:  0.6544, Adjusted R-squared:  0.6273
## F-statistic: 24.08 on 7 and 89 DF,  p-value: < 2.2e-16
```

```
lmod<-update(lmod,.~.-lcp)
summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + pgg45,
##     data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.77711 -0.41708  0.00002  0.40676  1.59681
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.980085   0.830665   1.180  0.24116
## lcavol       0.545770   0.076431   7.141 2.31e-10 ***
## lweight      0.449450   0.168078   2.674  0.00890 **
## age         -0.017470   0.010967  -1.593  0.11469
## lbph         0.105755   0.058191   1.817  0.07249 .
## svi          0.641666   0.219757   2.920  0.00442 **
## pgg45        0.003528   0.003068   1.150  0.25331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7061 on 90 degrees of freedom
## Multiple R-squared:  0.6493, Adjusted R-squared:  0.6259
## F-statistic: 27.77 on 6 and 90 DF,  p-value: < 2.2e-16
```

```
lmod<-update(lmod,.~.-pgg45)
summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83505 -0.39396  0.00414  0.46336  1.57888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.95100    0.83175   1.143 0.255882
## lcavol       0.56561    0.07459   7.583 2.77e-11 ***
## lweight      0.42369    0.16687   2.539 0.012814 *
## age         -0.01489    0.01075  -1.385 0.169528
## lbph         0.11184    0.05805   1.927 0.057160 .
## svi          0.72095    0.20902   3.449 0.000854 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7073 on 91 degrees of freedom
## Multiple R-squared:  0.6441, Adjusted R-squared:  0.6245
## F-statistic: 32.94 on 5 and 91 DF,  p-value: < 2.2e-16
```

```
lmod<-update(lmod,.~.-age)
summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + lbph + svi, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.82653 -0.42270  0.04362  0.47041  1.48530
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.14554    0.59747   0.244  0.80809
## lcavol       0.54960    0.07406   7.422 5.64e-11 ***
## lweight      0.39088    0.16600   2.355  0.02067 *
## lbph         0.09009    0.05617   1.604  0.11213
## svi          0.71174    0.20996   3.390  0.00103 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7108 on 92 degrees of freedom
## Multiple R-squared:  0.6366, Adjusted R-squared:  0.6208
## F-statistic: 40.29 on 4 and 92 DF,  p-value: < 2.2e-16
```

```
lmod<-update(lmod,.~.-lbph)
summary(lmod)
```

```
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + svi, data = prostate)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.72964 -0.45764  0.02812  0.46403  1.57013
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.26809    0.54350  -0.493  0.62298
## lcavol       0.55164    0.07467   7.388  6.3e-11 ***
## lweight      0.50854    0.15017   3.386  0.00104 **
## svi          0.66616    0.20978   3.176  0.00203 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7168 on 93 degrees of freedom
## Multiple R-squared:  0.6264, Adjusted R-squared:  0.6144
## F-statistic: 51.99 on 3 and 93 DF,  p-value: < 2.2e-16
```

(b) AIC:

AIC is minimized by a choice of four parameters, namely lcavol, lweight and svi as determined by the logical matrix.

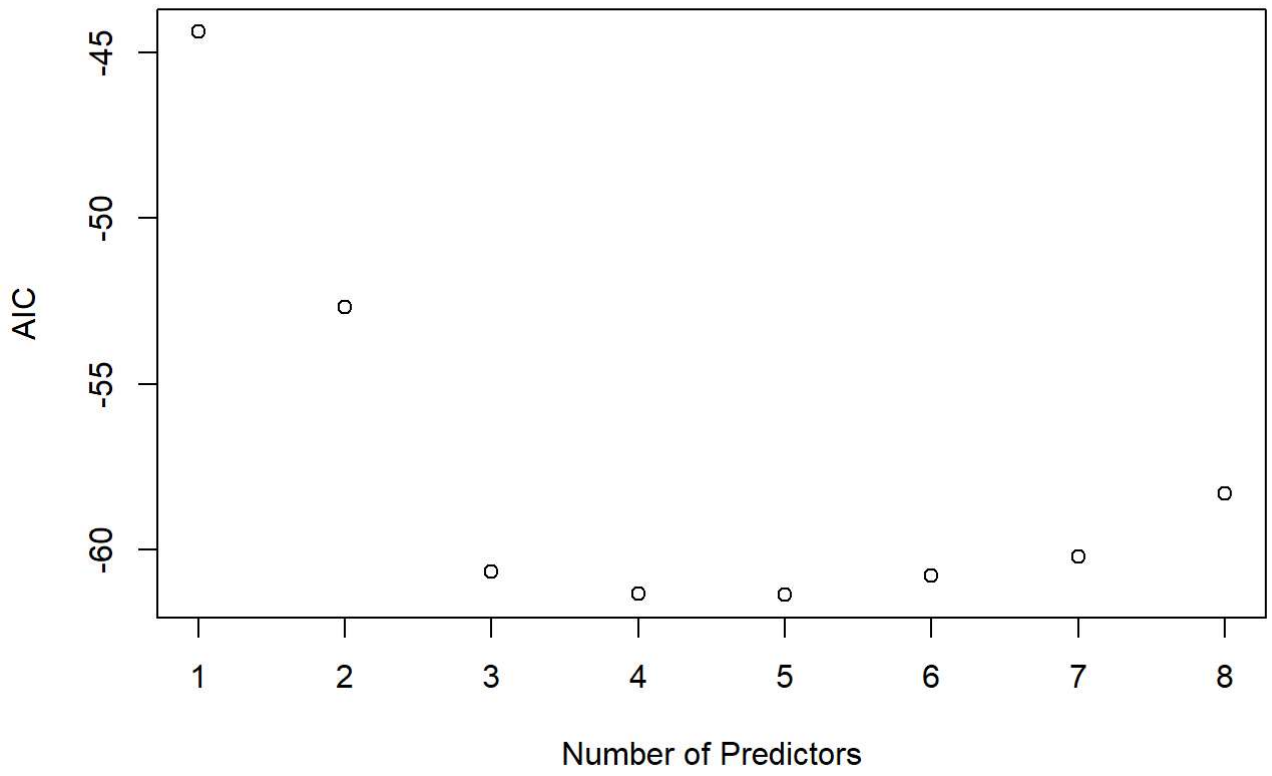
```
require(leaps)
```

```
## Loading required package: leaps
```

```
b<-regsubsets(lpsa~.,data=prostate)
rs<-summary(b)
rs$which
```

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

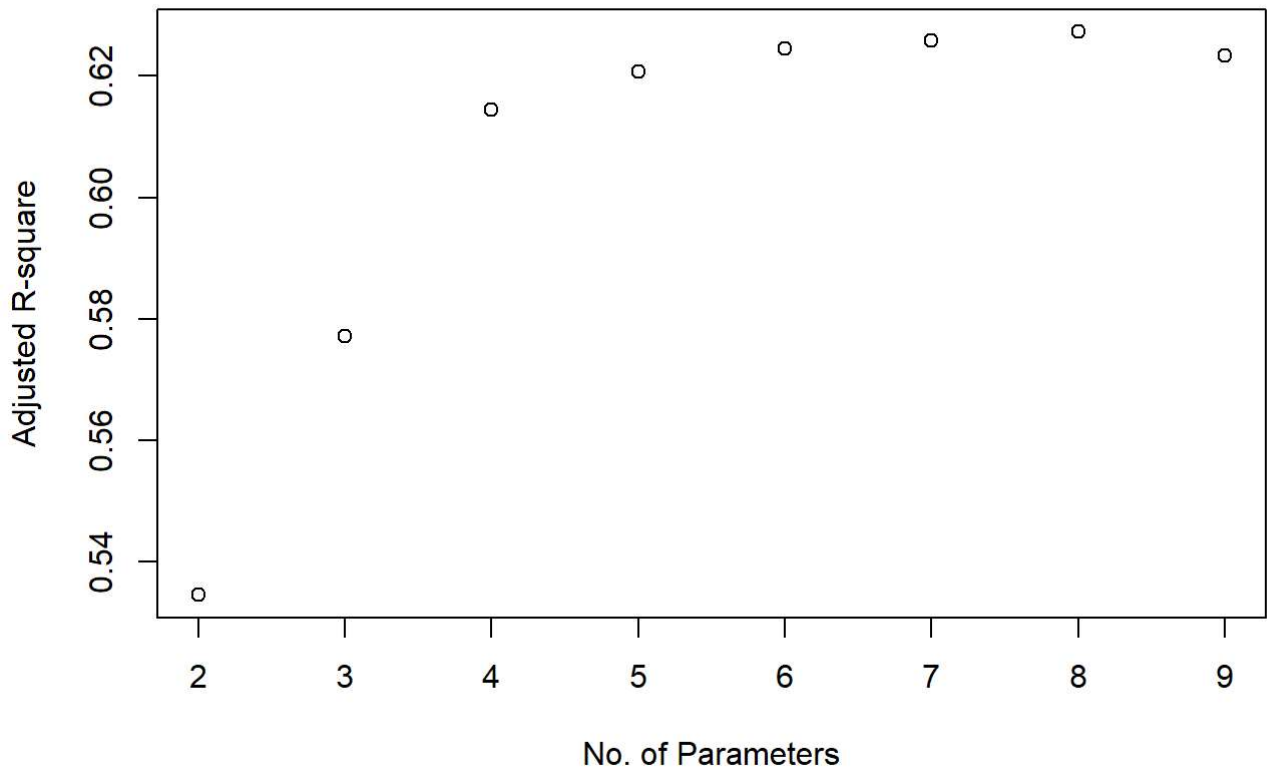
```
AIC <- nrow(prostate)*log(rs$rss/nrow(prostate)) + (2:9)*2
plot(AIC ~ I(1:8), ylab="AIC", xlab="Number of Predictors")
```



(c) Adjusted R2

The 7 parameter model with parameters, lcavol,lweight, age, lbph, svi and pgg45 as predictors is the clear winner as it is the smallest model that lies below the Cp=p line.

```
plot(2:9,rs$adjr2,xlab="No. of Parameters",ylab="Adjusted R-square")
```



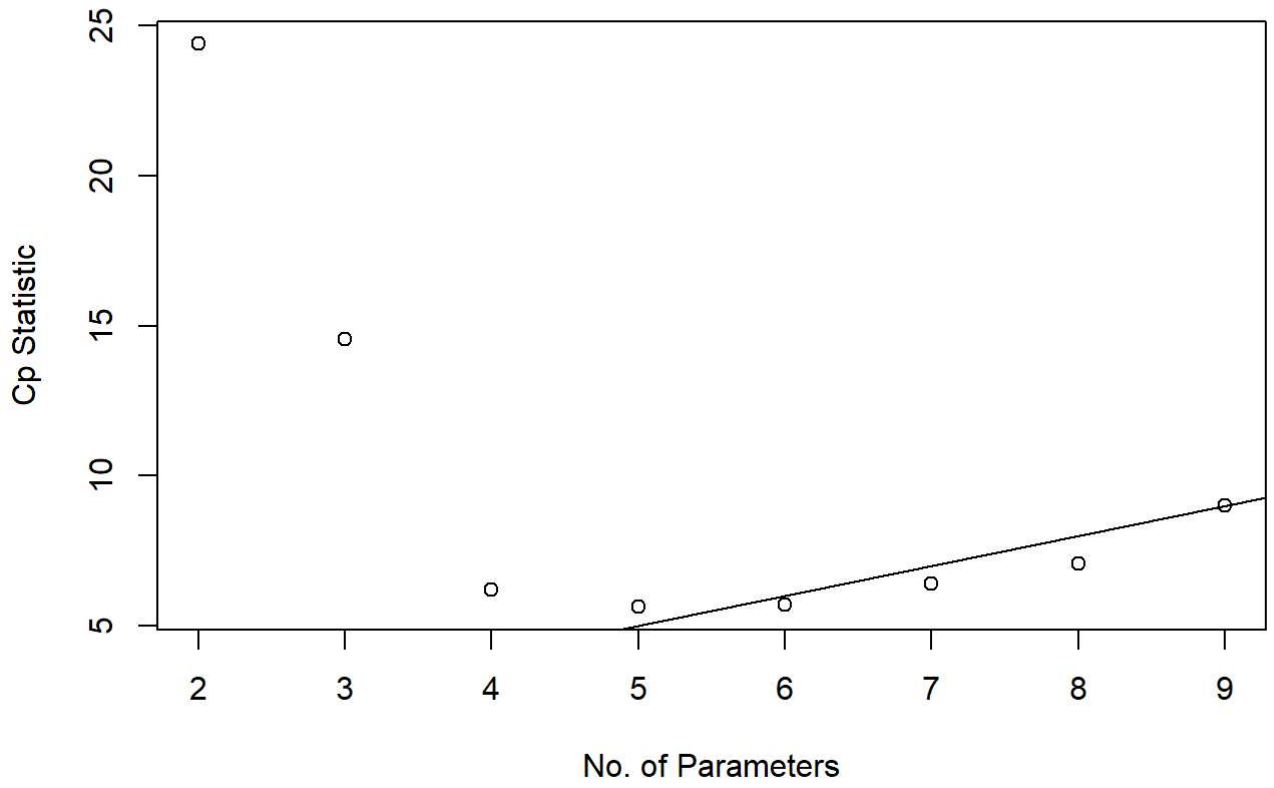

```
which.max(rs$adjr2)
```

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

(d) Mallows Cp

The 6 parameter model with parameters, lcavol,lweight, age, lbph and svi as predictors is the clear winner as it is the smallest model that lies below the Cp=p line.

```
plot(2:9,rs$cp,xlab="No. of Parameters",ylab="Cp Statistic")
abline(0,1)
```



Chapter 11: Exercise 4

Take the fat data, and use the percentage of body fat, siri, as the response and the other variables, except brozek and density as potential predictors. Remove every tenth observation from the data for use as a test sample. Use the remaining data as a training sample building the following models: (a) Linear regression with all predictors (b) Linear regression with variables selected using AIC (c) Principal component regression (d) Partial least squares (e) Ridge regression (f) LASSO regression Use the models you find to predict the response in the test sample. Make a report on the performances of the models

```
data(fat,package='faraway')
fat<-data.frame(fat)
ind <- seq(10, nrow(fat), by=10)
drops <- c("brozek","density")
train_data=fat[-ind,!(names(fat) %in% drops)]
#train_data=train_data[,-(!(names(train_data) %in% drops))]
test_data=fat[ind,!(names(fat) %in% drops)]
#test_data=test_data[,-(!(names(test_data) %in% drops))]
```

a. Linear regression with all predictors

```
lmod3<-lm(siri~.,data=test_data)
summary(lmod3)
```

```
##
## Call:
## lm(formula = siri ~ ., data = test_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91300 -0.33943  0.06558  0.28091  0.74021
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -150.28386   53.69480  -2.799   0.0208 *
## age           -0.01585    0.03507  -0.452   0.6619
## weight         0.03936    0.16746   0.235   0.8194
## height         2.16312    0.72677   2.976   0.0155 *
## adipos         2.21152    0.95912   2.306   0.0466 *
## free          -0.56551    0.04668 -12.115  7.1e-07 ***
## neck          -0.09268    0.17480  -0.530   0.6088
## chest          0.14160    0.11043   1.282   0.2318
## abdom          0.10218    0.07521   1.359   0.2074
## hip           -0.03646    0.13089  -0.279   0.7869
## thigh         -0.01546    0.13606  -0.114   0.9120
## knee          -0.14962    0.25385  -0.589   0.5701
## ankle          0.10486    0.37143   0.282   0.7841
## biceps         0.47382    0.16961   2.794   0.0209 *
## forearm       -0.17431    0.24492  -0.712   0.4947
## wrist          0.75940    0.51383   1.478   0.1736
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7632 on 9 degrees of freedom
## Multiple R-squared:  0.9953, Adjusted R-squared:  0.9876
## F-statistic: 128.2 on 15 and 9 DF,  p-value: 1.278e-08
```

```
rmse <- function(x,y) sqrt(mean((x-y)^2))
rmse(predict(lmod3), test_data$siri)
```

```
## [1] 0.4578904
```

(b)Linear regression with variables selected using AIC

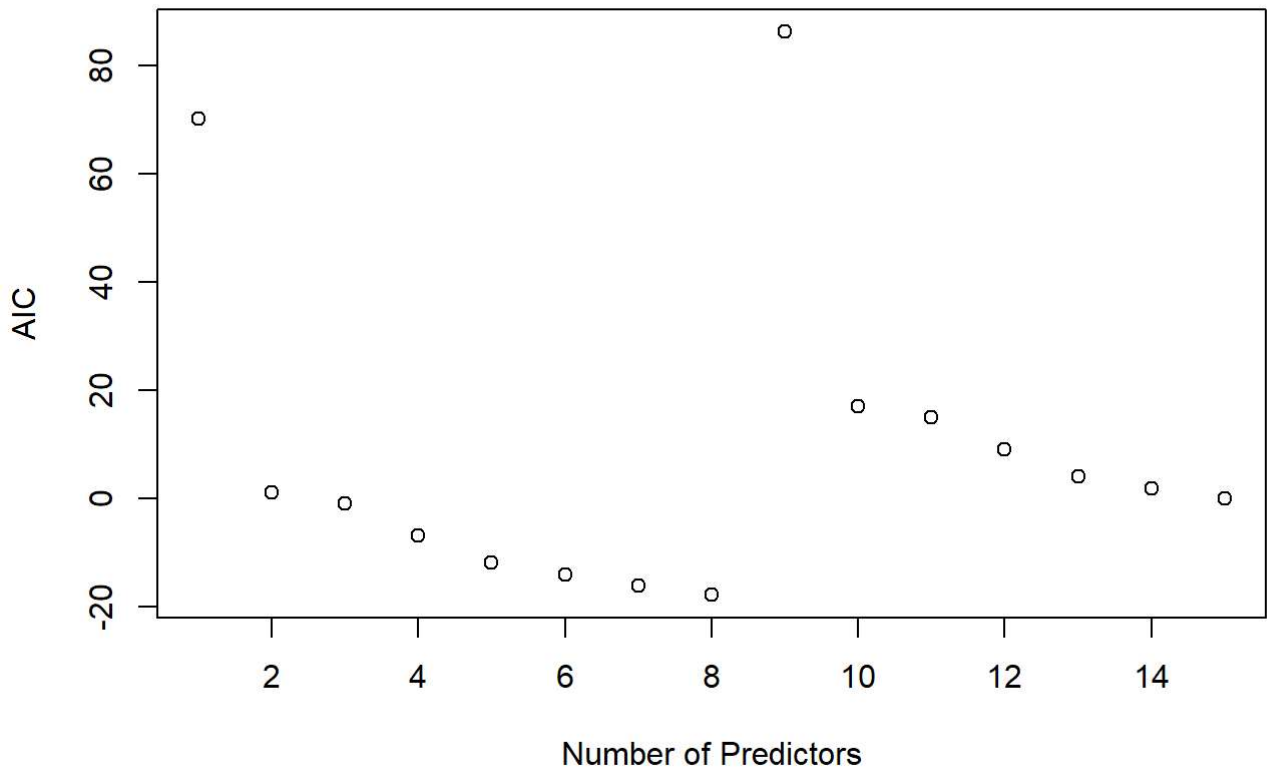
```
require(leaps)
b<-regsubsets(siri~.,data=test_data)
rs1<-summary(b)
rs1$which
```

```
##   (Intercept)   age weight height adipos  free  neck chest abdom   hip
## 1        TRUE FALSE  FALSE  FALSE   TRUE FALSE FALSE FALSE FALSE FALSE
## 2        TRUE FALSE   TRUE  FALSE  FALSE  TRUE FALSE FALSE FALSE FALSE
## 3        TRUE FALSE   TRUE  FALSE  FALSE  TRUE FALSE FALSE FALSE FALSE
## 4        TRUE FALSE  FALSE   TRUE   TRUE  TRUE FALSE FALSE FALSE FALSE
## 5        TRUE FALSE  FALSE   TRUE   TRUE  TRUE FALSE FALSE  TRUE FALSE
## 6        TRUE FALSE  FALSE   TRUE   TRUE  TRUE FALSE  TRUE  TRUE FALSE
## 7        TRUE FALSE  FALSE   TRUE   TRUE  TRUE FALSE  TRUE FALSE FALSE
## 8        TRUE FALSE  FALSE   TRUE   TRUE  TRUE FALSE  TRUE  TRUE FALSE
##   thigh  knee ankle biceps forearm wrist
## 1 FALSE FALSE FALSE  FALSE   FALSE FALSE
## 2 FALSE FALSE FALSE  FALSE   FALSE FALSE
## 3 FALSE FALSE FALSE  FALSE   FALSE  TRUE
## 4 FALSE FALSE FALSE   TRUE   FALSE FALSE
## 5 FALSE FALSE FALSE   TRUE   FALSE FALSE
## 6 FALSE FALSE FALSE   TRUE   FALSE FALSE
## 7 FALSE FALSE FALSE   TRUE    TRUE  TRUE
## 8 FALSE FALSE FALSE   TRUE    TRUE  TRUE
```

```
AIC <- nrow(test_data)*log(rs1$rss/nrow(test_data)) + (2:16)*2
```

```
## Warning in nrow(test_data) * log(rs1$rss/nrow(test_data)) + (2:16) * 2:
## longer object length is not a multiple of shorter object length
```

```
plot(AIC ~ I(1:15), ylab="AIC", xlab="Number of Predictors")
```




```
#No. of Parameters=8 :Intercept,height, adipos, free, chest, biceps, forearm, wrist

lmodAIC=lm(siri~height+adipos+free+chest+biceps+forearm+wrist,data=train_data)
rmse(predict(lmodAIC), test_data$siri)
```

```
## Warning in x - y: longer object length is not a multiple of shorter object
## length
```

```
## [1] 10.39814
```

c. Principal component regression

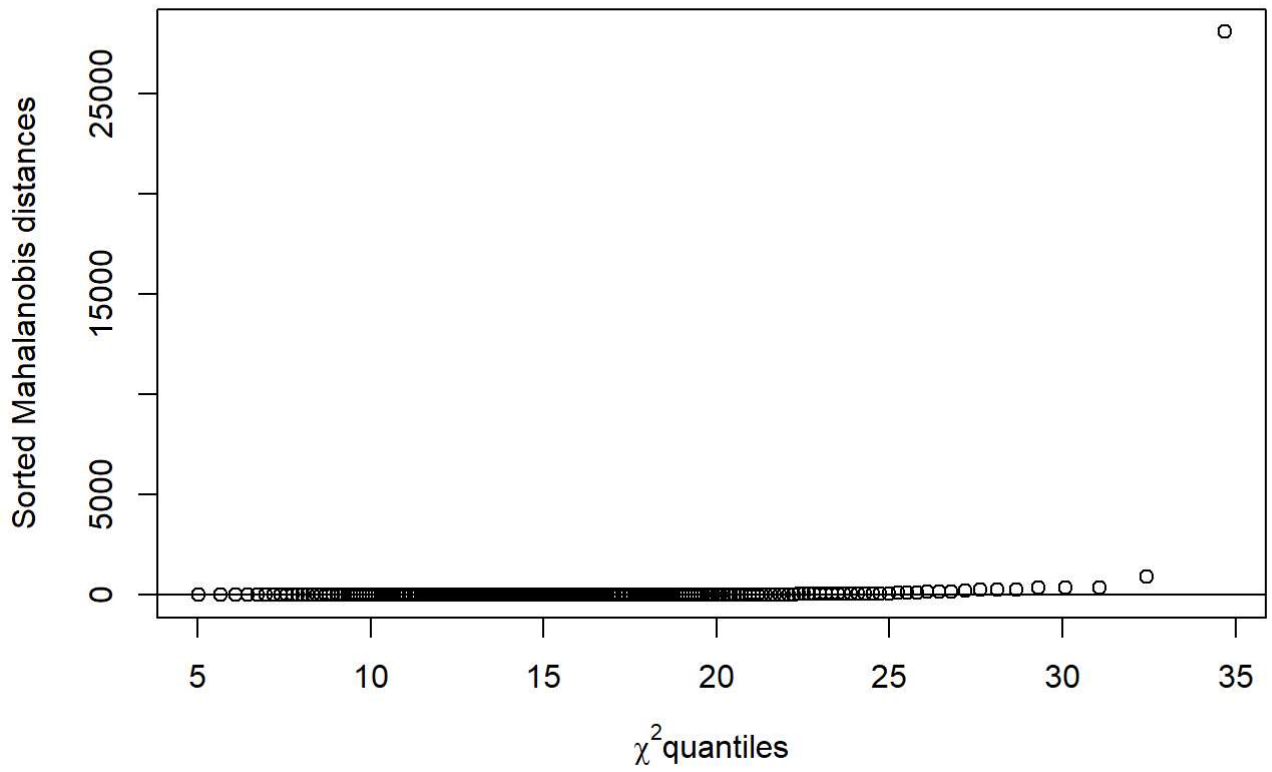
```
prfatc <- prcomp(train_data)
summary(prfatc)
```

```
## Importance of components:
##                             PC1     PC2      PC3     PC4    PC5     PC6
## Standard deviation     37.3563 15.5618 10.31319 3.78465 3.4687 2.59870
## Proportion of Variance  0.7766  0.1348  0.05919 0.00797 0.0067 0.00376
## Cumulative Proportion   0.7766  0.9114  0.97056 0.97853 0.9852 0.98898
##                             PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     2.21857 1.85400 1.59104 1.48755 1.32575 1.31106
## Proportion of Variance 0.00274 0.00191 0.00141 0.00123 0.00098 0.00096
## Cumulative Proportion  0.99172 0.99364 0.99505 0.99628 0.99725 0.99821
##                            PC13    PC14    PC15    PC16
## Standard deviation     1.14090 1.03327 0.77727 0.49028
## Proportion of Variance 0.00072 0.00059 0.00034 0.00013
## Cumulative Proportion  0.99894 0.99953 0.99987 1.00000
```

```
require(MASS)
```

```
## Loading required package: MASS
```

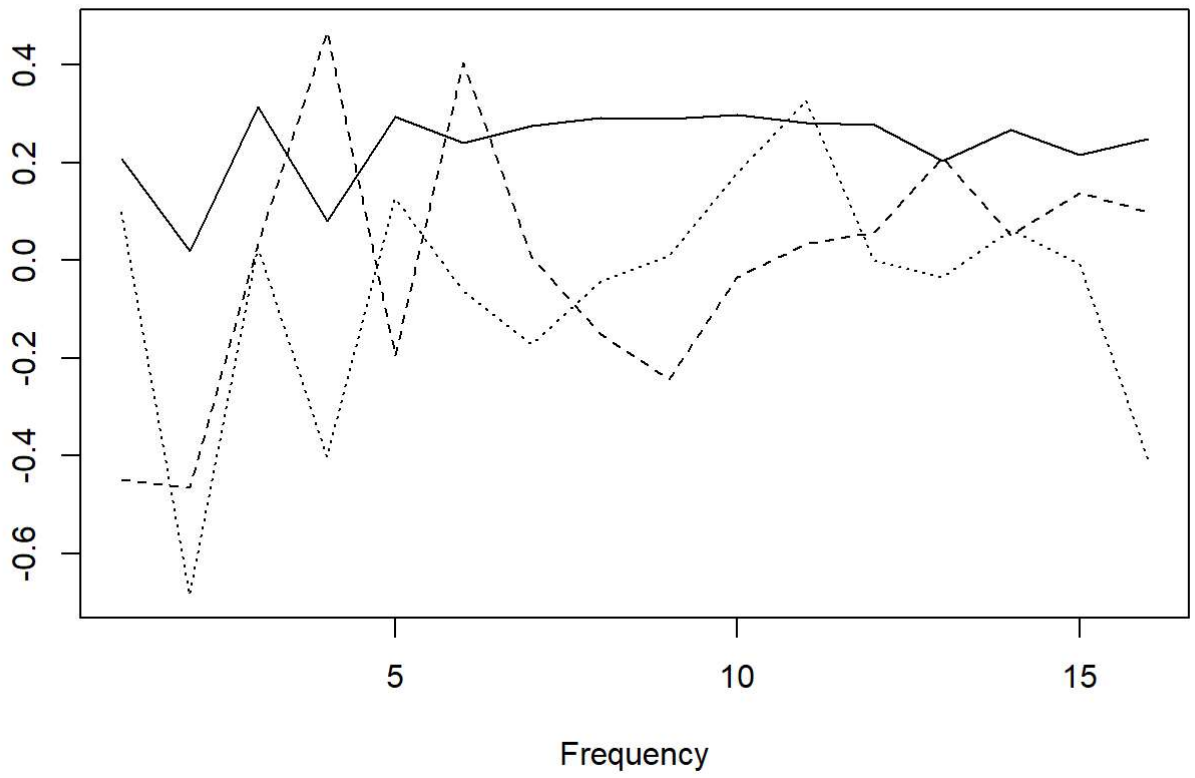
```
robfat <- cov.rob(train_data)
md <- mahalanobis(train_data, center=robfat$center, cov=robfat$cov)
n <- nrow(train_data);p <- ncol(train_data)
plot(qchisq(1:n/(n+1),p), sort(md), xlab=expression(paste(chi^2,"
quantiles")), ylab="Sorted Mahalanobis distances")
abline(0,1)
```




```
train_pca <- prcomp((train_data),scale=TRUE)
round(train_pca$sdev,3)
```

```
##  [1] 3.135 1.364 1.048 0.829 0.803 0.723 0.574 0.522 0.480 0.428 0.361
## [12] 0.280 0.234 0.198 0.179 0.077
```

```
matplot(1:16, train_pca$rot[,1:3], type="l", xlab="Frequency", ylab
="", col=1)
```




```
require(pls)
```

```
## Loading required package: pls
```

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

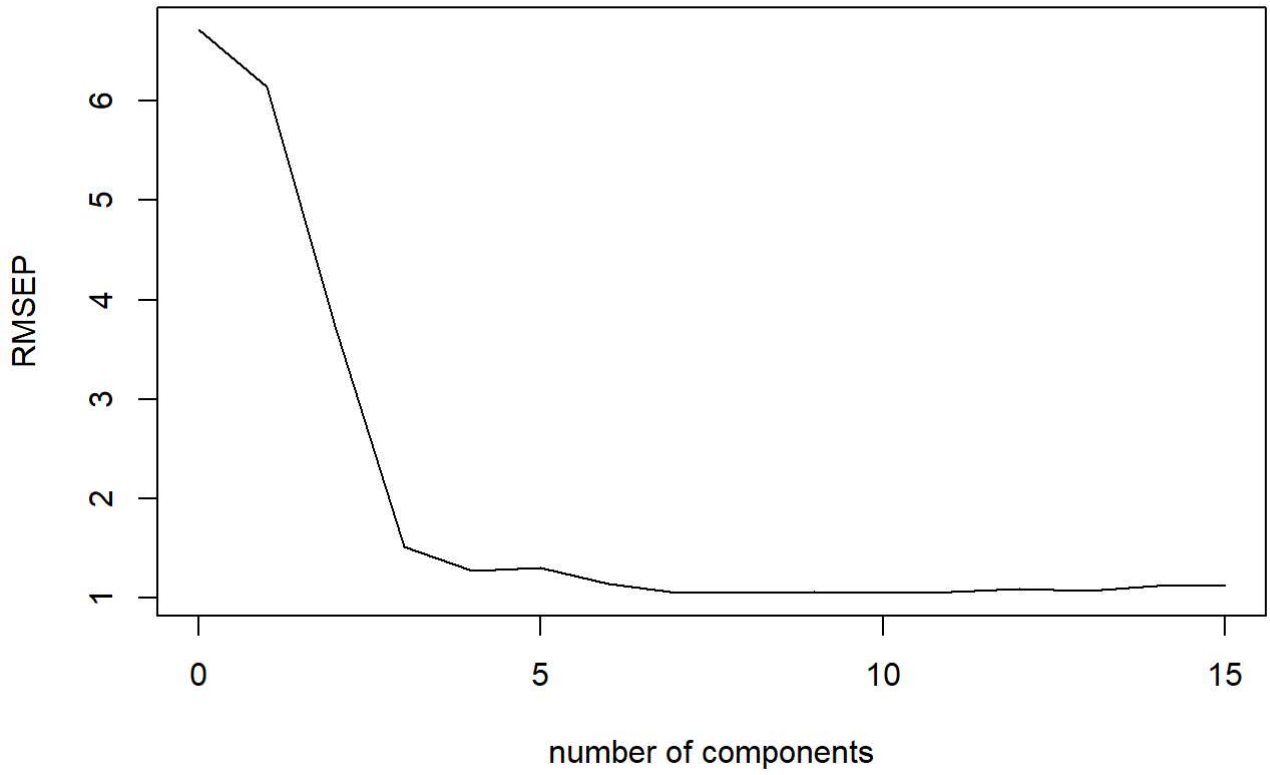
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pcrmod <- pcr(siri ~ ., data=(train_data), ncomp=15)
rmse <- function(x,y) sqrt(mean((x-y)^2))
rmse(predict(pcrmod, ncomp=15), test_data$siri)
```

```
## Warning in x - y: longer object length is not a multiple of shorter object
## length
```

```
## [1] 10.88871
```

```
pcrmse <- RMSEP(pcrmod, newdata=test_data)
plot(pcrmse,main="")
```




```
which.min(pcrmse$val)
```

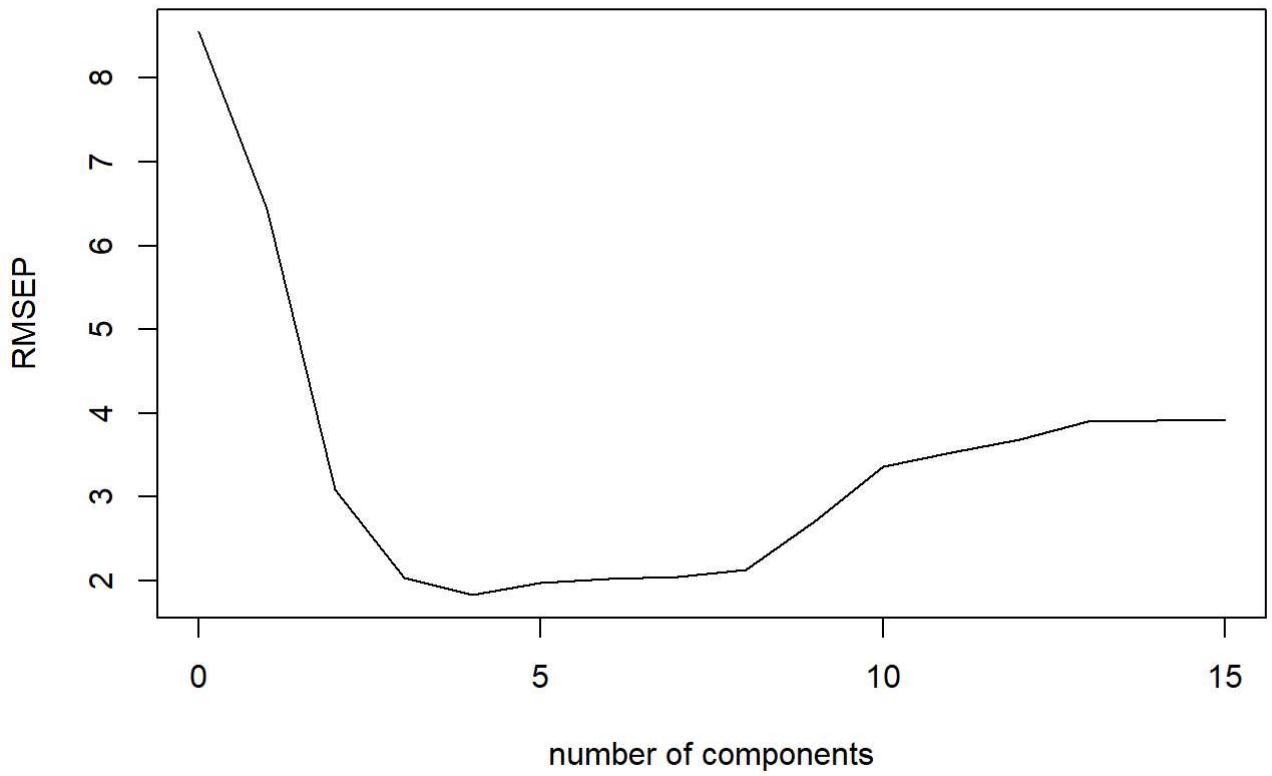
```
## [1] 8
```

```
#Optimum no. of components=11
pcrmse$val[11]
```

```
## [1] 1.049483
```

d)Partial Least Squares

```
set.seed(123)
plsmod <- plsr(siri ~ ., data=train_data, ncomp=15, validation
="CV")
plsCV <- RMSEP(plsmod, estimate="CV")
plot(plsCV,main="")
```




```
ypred <- predict(plsmod,ncomp=5)
rmse(ypred, train_data$siri)
```

```
## [1] 1.549977
```

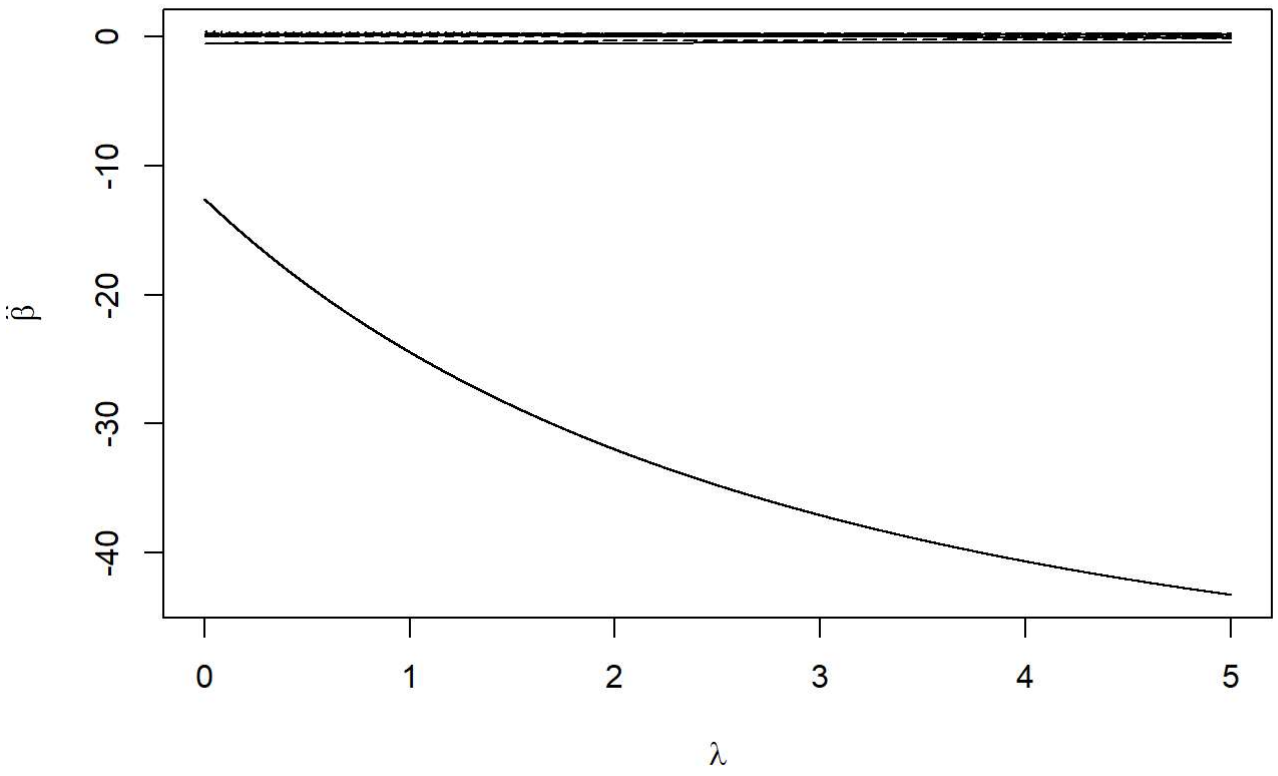
```
ytpred <- predict(plsmod, test_data, ncomp=5)
rmse(ytpred, test_data$siri)
```

```
## [1] 1.049793
```

e. Ridge Regression

```
require(MASS)
rgmod <- lm.ridge(siri ~ ., data.frame((train_data)), lambda = seq(0, 5, len=10000))

matplot(rgmod$lambda, coef(rgmod), type="l", xlab=expression(lambda)
,ylab=expression(hat(beta)),col=1)
```




```
which.min(rgmod$GCV)
```

```
## 0.04650465
##         94
```

```
ypred <- cbind(1,as.matrix(train_data[,-1])) %*% coef(rgmod)[94,]
rmse(ypred, train_data$siri)
```

```
## [1] 1.494422
```

```
#which is comparable to the above, but for the test sample we find:
ypred <- cbind(1,as.matrix(test_data[,-1])) %*% coef(rgmod)[94,]
rmse(ypred, test_data$siri)
```

```
## [1] 1.128102
```

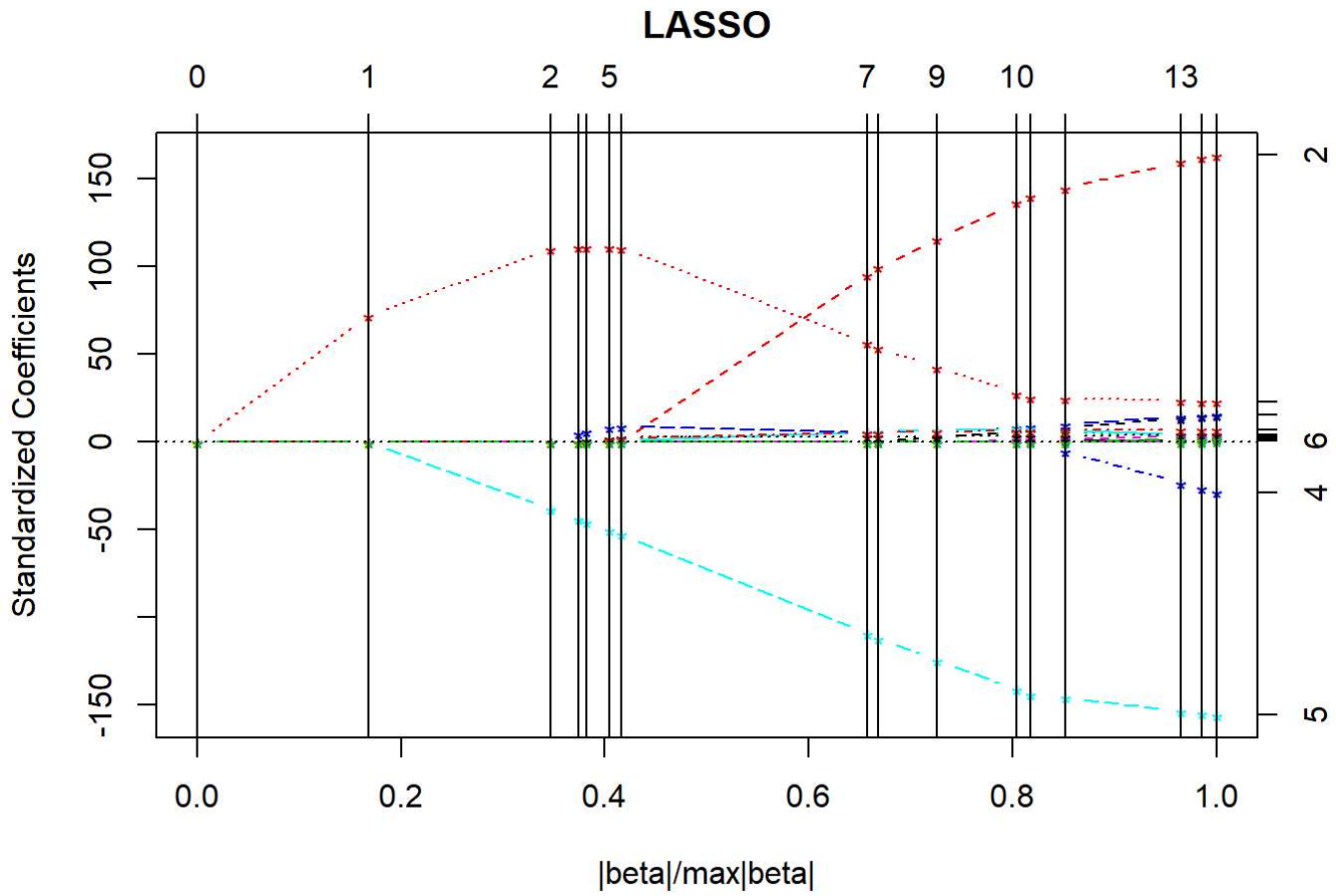
f. LASSO

```
require(lars)
```

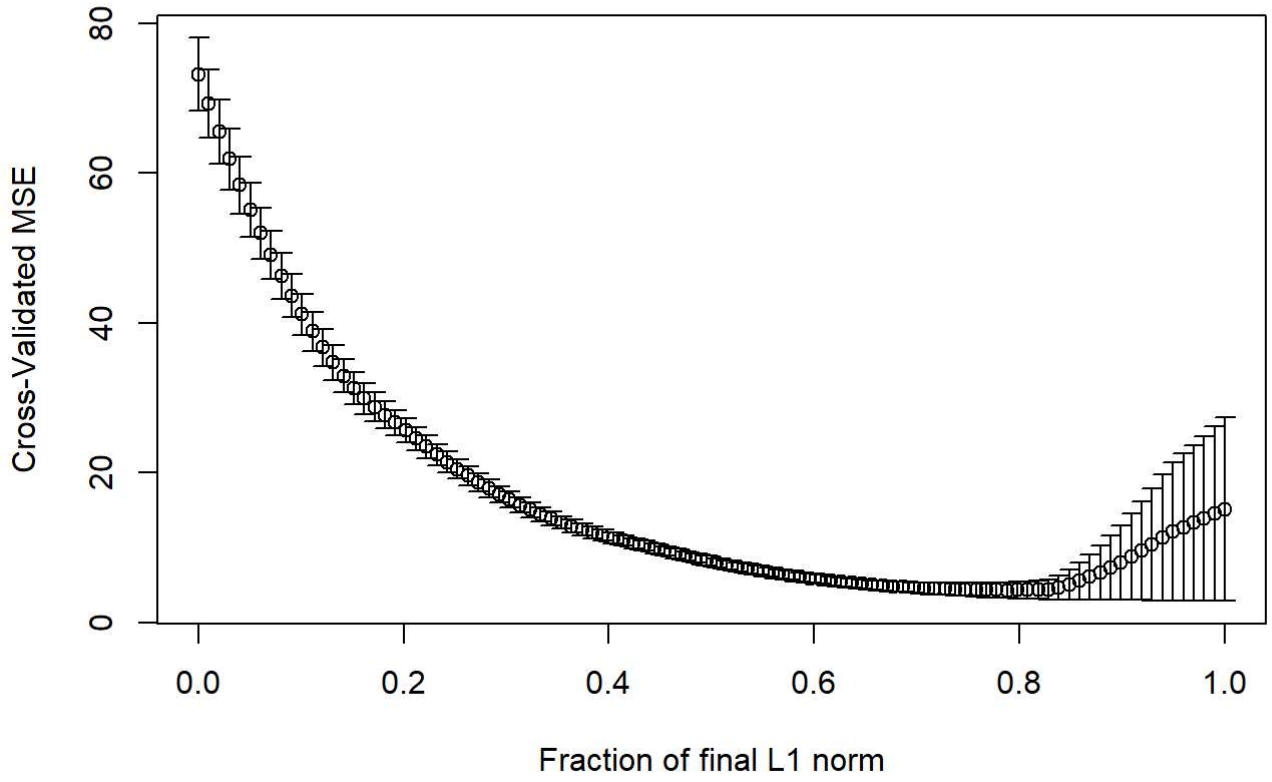
```
## Loading required package: lars
```

```
## Loaded lars 1.2
```

```
trainy <- train_data$siri
trainx <- as.matrix(train_data[,-1])
lassomod <- lars(trainx,trainy)
plot(lassomod)
```




```
#We now compute the crossvalidation choice of t:
set.seed(123)
cvout <- cv.lars(trainx,trainy)
```




```
cvout$index[which.min(cvout$cv)]
```

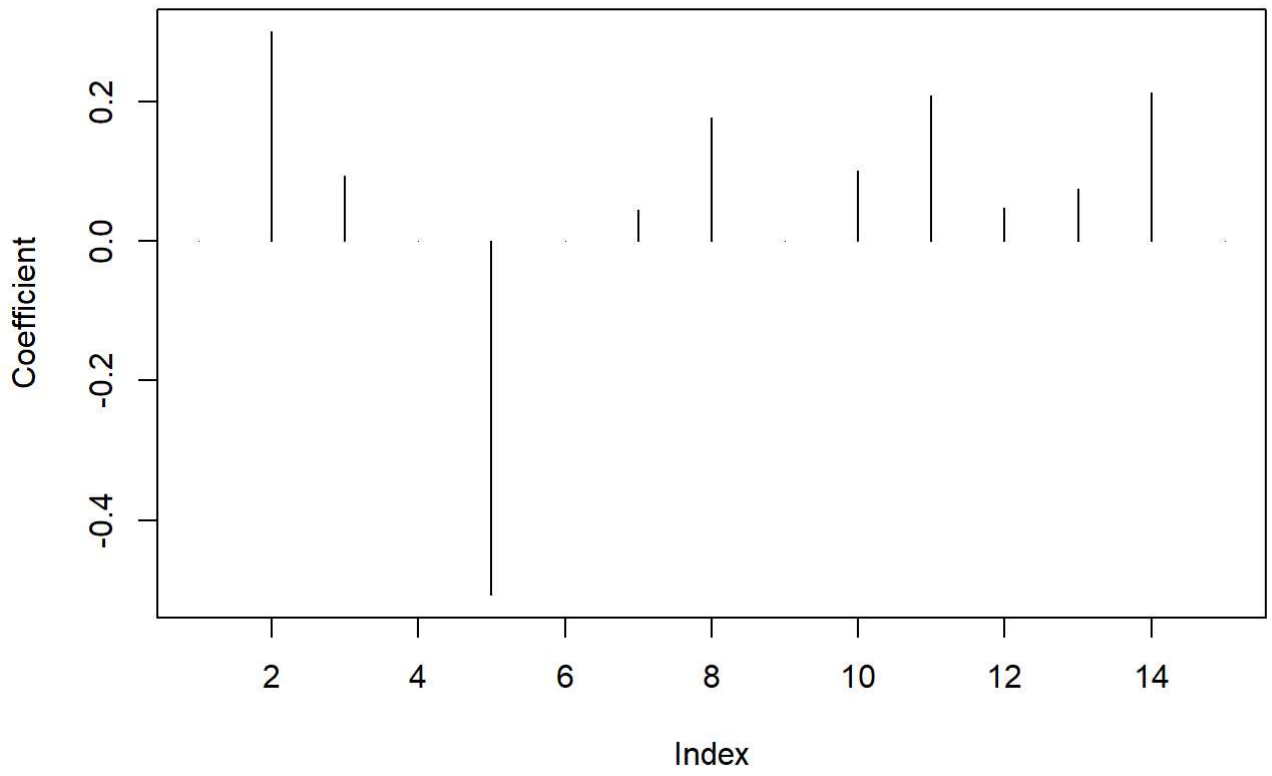
```
## [1] 0.7878788
```

```
#0.7979798
#For this choice of t, we compute the predicted values for the test data:

testx <- as.matrix(test_data[,-1])
predlars <- predict(lassomod,testx,s=0.7979798,mode="fraction")
#The RMSE may now be computed:
rmse(test_data$siri, predlars$fit)
```

```
## [1] 1.093538
```

```
predlars <- predict(lassomod, s=0.7979798, type="coef", mode="fraction")
plot(predlars$coef,type="h",ylab="Coefficient")
```




```
sum(predlars$coef != 0)
```

```
## [1] 10
```