Zhang.Chen

Chen Zhang

10/18/2018

# Bootstrap

set.seed(123); data=rnorm(200) The 90 percentile of data is estimated by quantile(data, 0.9, na.rm=TRUE) in R. We want to find the standard error of the estimated 90 percentile and 95 % confidence interval for the true 90 percentile of the population using bootstrap.

### a) Write an R function that calculates the 90 percentile of data to be used in boot function in b).

set.seed(123)  
data=rnorm(200)  
boot.fn = function(data,index){  
 x=data[index]  
 return(as.numeric(quantile(x,0.9,na.rm=T)))  
}  
boot.fn(data=data)

## [1] 1.209574

### b) Using boot function, find the standard error for the population 90 percentile and 95% CI from the data. Bootstrap size should be 1000.

library(boot)  
set.seed(123)  
bt.result = boot(data=rnorm(200), boot.fn, R=1000)  
sd=sd(bt.result$t);sd

## [1] 0.1409542

lower = round((bt.result$t0-1.96\*sd),digit=2)  
upper = round((bt.result$t0+1.96\*sd),digit=2)  
ci =c(lower,upper);ci

## [1] 0.93 1.49

The standard error is 0.1409542,and the 95% CI is0.93, 1.49.

# Regression

library(ISLR)( data(College) College data is statistics for a large number of US Colleges from the 1995 issue of US News and World Report. The data contains 777 observations on the 18 variables. The variable Grad.Rate is the graduation rate of each college. We want to build a prediction model for Grad.Rate from the data.

### 1)Partition the College data to 70% training set and 30% test set. Use set.seed(1)

library(ISLR)  
data.reg=College  
set.seed(1)  
train.col = sample(1:dim(data.reg)[1],dim(data.reg)\*0.7,replace = FALSE)  
test.col <- -train.col  
train.reg = data.reg[train.col,]  
test.reg = data.reg[test.col,]

### 2)From the linear model with all the variables in, find the best subset model for Grad.Rate for the training data using stepAIC. (may have to use glm function instead of lm).

library(MASS)  
step.glm = glm(Grad.Rate~., data = train.reg)  
summary(step.glm, cor = F)

##   
## Call:  
## glm(formula = Grad.Rate ~ ., data = train.reg)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -50.447 -6.505 -0.261 6.622 50.917   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 32.3536443 5.7343876 5.642 2.75e-08 \*\*\*  
## PrivateYes 2.5450215 1.9701603 1.292 0.197000   
## Apps 0.0009927 0.0005041 1.969 0.049438 \*   
## Accept -0.0005627 0.0009738 -0.578 0.563637   
## Enroll 0.0008930 0.0024924 0.358 0.720276   
## Top10perc 0.1464341 0.0838843 1.746 0.081454 .   
## Top25perc 0.1038193 0.0638749 1.625 0.104687   
## F.Undergrad -0.0002165 0.0004528 -0.478 0.632678   
## P.Undergrad -0.0015949 0.0005947 -2.682 0.007555 \*\*   
## Outstate 0.0010342 0.0002687 3.849 0.000133 \*\*\*  
## Room.Board 0.0019590 0.0006863 2.854 0.004484 \*\*   
## Books -0.0004690 0.0037203 -0.126 0.899727   
## Personal -0.0012161 0.0009156 -1.328 0.184695   
## PhD 0.0843978 0.0701055 1.204 0.229183   
## Terminal -0.0628465 0.0785994 -0.800 0.424316   
## S.F.Ratio 0.1137661 0.1912031 0.595 0.552099   
## perc.alumni 0.2410051 0.0581085 4.148 3.92e-05 \*\*\*  
## Expend -0.0005137 0.0001783 -2.882 0.004115 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 153.3709)  
##   
## Null deviance: 147917 on 542 degrees of freedom  
## Residual deviance: 80520 on 525 degrees of freedom  
## AIC: 4293.5  
##   
## Number of Fisher Scoring iterations: 2

reg.step = stepAIC(step.glm, trace =F)  
reg.step$anova

## Stepwise Model Path   
## Analysis of Deviance Table  
##   
## Initial Model:  
## Grad.Rate ~ Private + Apps + Accept + Enroll + Top10perc + Top25perc +   
## F.Undergrad + P.Undergrad + Outstate + Room.Board + Books +   
## Personal + PhD + Terminal + S.F.Ratio + perc.alumni + Expend  
##   
## Final Model:  
## Grad.Rate ~ Apps + Top10perc + Top25perc + P.Undergrad + Outstate +   
## Room.Board + Personal + perc.alumni + Expend  
##   
##   
## Step Df Deviance Resid. Df Resid. Dev AIC  
## 1 525 80519.73 4293.505  
## 2 - Books 1 2.437522 526 80522.17 4291.521  
## 3 - Enroll 1 19.559494 527 80541.73 4289.653  
## 4 - F.Undergrad 1 15.836981 528 80557.56 4287.760  
## 5 - S.F.Ratio 1 47.604292 529 80605.17 4286.081  
## 6 - Accept 1 86.530991 530 80691.70 4284.663  
## 7 - Terminal 1 132.589037 531 80824.29 4283.555  
## 8 - PhD 1 118.565399 532 80942.85 4282.351  
## 9 - Private 1 229.484254 533 81172.34 4281.888

### 3)What are the selected important variables?

According to the result, The selected important variables are Apps,Top10perc,Top25perc,P.Undergrad,Outstate,Room.Board,Personal,perc.alumni,Expend.

### 4)Use LASSO method to the data and find selected important variables.

library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-16

set.seed(1)  
lasso.x = model.matrix(Grad.Rate~.,data.reg)[,-1]  
lasso.y = data.reg$Grad.Rate  
cv.out=cv.glmnet(lasso.x[train.col,],lasso.y[train.col],alpha=1)  
bestlam=cv.out$lambda.min  
lasso.model=glmnet(lasso.x[train.col,],lasso.y[train.col],alpha=1,lambda =bestlam)  
lasso.coef=predict(cv.out,type="coefficients",s=bestlam)  
lasso.coef

## 18 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) 34.3691457731  
## PrivateYes 1.6909904998  
## Apps 0.0004836954  
## Accept .   
## Enroll .   
## Top10perc 0.1316799890  
## Top25perc 0.1084929948  
## F.Undergrad .   
## P.Undergrad -0.0015000107  
## Outstate 0.0009201558  
## Room.Board 0.0018087507  
## Books .   
## Personal -0.0011852412  
## PhD 0.0161264485  
## Terminal .   
## S.F.Ratio 0.0142402808  
## perc.alumni 0.2275751599  
## Expend -0.0002888006

Based on the result, we see the important variables are PrivateYes, Apps,Top10perc,Top25perc,P.Undergrad,Outstate,Room.Board,Personal,PhD,,S.F.Ratio,perc.alumi,Expend.

### 5)Find the predicted Grad.Rate for the three models, the best stepAIC model, the model with all the variables-in, and LASSO for the test data. Find and compare the test data R^2 value for the three models. Which one do you prefer?

#step AIC   
test\_y = data.reg$Grad.Rate[test.col]  
step.pred = predict(reg.step,newdata =test.reg)  
r.step = 1-sum((test\_y-step.pred)^2)/sum((test\_y-mean(test\_y))^2)  
  
#all variable-in  
all.pred = predict(step.glm, newdata = test.reg)  
r.all = 1-sum((test\_y-all.pred)^2)/sum((test\_y-mean(test\_y))^2)  
  
#Lasso  
lasso.pred = predict(lasso.model,s=bestlam,newx =lasso.x[test.col,])  
r.lasso = 1-sum((test\_y-lasso.pred)^2)/sum((test\_y-mean(test\_y))^2)

**Compare R square for three models**

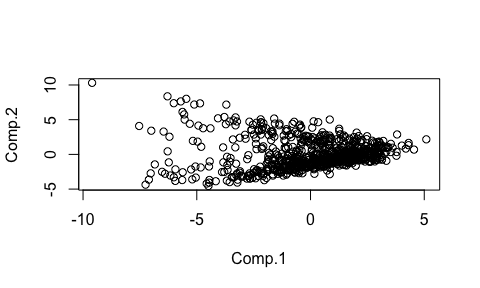
com.model = data.frame('Model'=c('Best stepAIC model','all the variable-in','Lasso'),'R square' = c(r.step,r.all,r.lasso))  
com.model

## Model R.square  
## 1 Best stepAIC model 0.4474706  
## 2 all the variable-in 0.4581183  
## 3 Lasso 0.4485480

Based on the result, we need to find the highest r square, I prefer the model with all variable in.

### 6)Find PCA for College data without Grad.Rate and plot the first principle component and the second principle component.

pc.cr <- princomp(data.reg[,c(2:17)],cor = TRUE)  
plot(pc.cr$score[,1:2])



### 7)How much variations are captured with these two components?

variation=cumsum(pc.cr$sdev[1:2]^2)/sum(pc.cr$sdev^2)  
variation

## Comp.1 Comp.2   
## 0.3235647 0.5938306

About 59.3830601% of variation is captured by first two principle component.

### 8)Run pcr for this data with Grad.Rate as response variable. How many components should you use? (Find the best number of components).

library(pls)

##   
## Attaching package: 'pls'

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

pcr.fit=pcr(Grad.Rate~., data=data.reg,scale=TRUE, validation="CV")  
press=pcr.fit$validation$PRESS   
bestcomp=which(press==min(press))   
bestcomp

## [1] 13

pcr.fit=pcr(Grad.Rate~., data=data.reg,scale=TRUE, validation="CV", ncomp=bestcomp)

Based on the result, 13 principle components should be used.

# Classification

library(ElemStatLearn);data(spam) The spam data is used to construct a personalized spam filter. It contains 4601 observations on 58 variables. The spam variable is response variable.

### 1)Partition the sample randomly to 70% training set and 30% test set. Use set.seed(123).

library(ElemStatLearn)  
spam0 <- ifelse(spam$spam == "spam","yes","no") #yes means is spam, no means email.  
data.class=data.frame(spam[,c(1:57)],spam = factor(spam0)) #delete the original spam variable  
set.seed(123)  
train.spam = sample(1:dim(data.class)[1],dim(data.class)\*0.7,replace = FALSE)  
test.spam = -train.spam  
train.data = data.class[train.spam,]  
test.data = data.class[test.spam,]  
class.test.y = data.class$spam

### 2)Find the logistic regression with all the variables in and find 10-fold cross-validated error on training data. (to Ignore all the warning messages and eliminate from the file use: suppressWarnings(SA.glm <- glm(spam~., data = spam[train,], family=binomial) )

source("cv.R")  
suppressWarnings(log.model <- glm(spam~., data=train.data, family="binomial"))  
cv.log = CV.logistic(data=train.data, glmfit=log.model,   
 yname="spam", K=10, seed=123)  
cv.log$error

## [1] 0.08354

the error rate is 0.08354.

### 3)Find the best logistic regression model using stepAIC on training data. (It may take over 5 minutes)

suppressWarnings(SA.glm2 <- stepAIC(SA.glm, trace = FALSE)) will do without warning messages.

source("cv.R")  
suppressWarnings(step.fit <- glm(spam~.,data=train.data,family="binomial"))  
cv.step <- cv.stepAIC.logistic(data=train.data, glmfit = step.fit, K=10, seed=123)  
cv.step$best.model

## Stepwise Model Path   
## Analysis of Deviance Table  
##   
## Initial Model:  
## spam ~ A.1 + A.2 + A.3 + A.4 + A.5 + A.6 + A.7 + A.8 + A.9 +   
## A.10 + A.11 + A.12 + A.13 + A.14 + A.15 + A.16 + A.17 + A.18 +   
## A.19 + A.20 + A.21 + A.22 + A.23 + A.24 + A.25 + A.26 + A.27 +   
## A.28 + A.29 + A.30 + A.31 + A.32 + A.33 + A.34 + A.35 + A.36 +   
## A.37 + A.38 + A.39 + A.40 + A.41 + A.42 + A.43 + A.44 + A.45 +   
## A.46 + A.47 + A.48 + A.49 + A.50 + A.51 + A.52 + A.53 + A.54 +   
## A.55 + A.56 + A.57  
##   
## Final Model:  
## spam ~ A.1 + A.2 + A.3 + A.4 + A.5 + A.6 + A.7 + A.8 + A.9 +   
## A.16 + A.17 + A.20 + A.21 + A.23 + A.24 + A.25 + A.26 + A.27 +   
## A.28 + A.29 + A.33 + A.34 + A.35 + A.36 + A.39 + A.41 + A.42 +   
## A.43 + A.44 + A.45 + A.46 + A.48 + A.49 + A.52 + A.53 + A.54 +   
## A.56 + A.57  
##   
##   
## Step Df Deviance Resid. Df Resid. Dev AIC  
## 1 2840 1137.551 1253.551  
## 2 - A.32 1 0.009971946 2841 1137.561 1251.561  
## 3 - A.19 1 0.013228769 2842 1137.575 1249.575  
## 4 - A.55 1 0.015384764 2843 1137.590 1247.590  
## 5 - A.14 1 0.059392228 2844 1137.649 1245.649  
## 6 - A.11 1 0.243508312 2845 1137.893 1243.893  
## 7 - A.40 1 0.307040678 2846 1138.200 1242.200  
## 8 - A.38 1 0.339236879 2847 1138.539 1240.539  
## 9 - A.50 1 0.380402526 2848 1138.920 1238.920  
## 10 - A.37 1 0.376700370 2849 1139.296 1237.296  
## 11 - A.18 1 0.401687539 2850 1139.698 1235.698  
## 12 - A.51 1 0.478652705 2851 1140.177 1234.177  
## 13 - A.13 1 0.554701502 2852 1140.731 1232.731  
## 14 - A.30 1 0.673743695 2853 1141.405 1231.405  
## 15 - A.10 1 0.745876177 2854 1142.151 1230.151  
## 16 - A.31 1 1.069975730 2855 1143.221 1229.221  
## 17 - A.15 1 1.164770890 2856 1144.386 1228.386  
## 18 - A.22 1 1.234592243 2857 1145.620 1227.620  
## 19 - A.47 1 1.597044188 2858 1147.217 1227.217  
## 20 - A.12 1 1.713979412 2859 1148.931 1226.931

According to the result, the best stepAIC model is spam ~ A.1 + A.2 + A.3 + A.4 + A.5 + A.6 + A.7 + A.8 + A.9 + A.16 + A.17 + A.20 + A.21 + A.23 + A.24 + A.25 + A.26 + A.27 + A.28 + A.29 + A.33 + A.34 + A.35 + A.36 + A.39 + A.41 + A.42 + A.43 + A.44 + A.45 + A.46 + A.48 + A.49 + A.52 + A.53 + A.54 + A.56 + A.57

### 4)What is the 10-fold cross-validated error for the stepAIC model on training data?

cv.step.error = round(cv.step$Error,digit=6)  
cv.step.error

## [1] 0.076087

the 10-fold cross-validated error for the stepAIC model on training data is 0.076087.

### 5)Apply the above models to the test data and get the prediction confusion matrix and prediction error rates.

#logistic model  
test.y = test.data$spam  
log.probs <- predict(log.model,newdata = test.data,type="response")  
le = levels(test.data$spam);le

## [1] "no" "yes"

log.test.pred = ifelse(log.probs>= 0.5,le[2],le[1])  
table(log.test.pred,test.y)

## test.y  
## log.test.pred no yes  
## no 802 66  
## yes 29 484

log.test.error <- mean(log.test.pred != test.y)  
round(log.test.error,digits = 6)

## [1] 0.068791

#stepAIC  
suppressWarnings(step.model <- glm(spam ~ A.1 + A.2 + A.3 + A.4 + A.5 + A.6 + A.7 + A.8 + A.9 + A.16 + A.17 + A.20 + A.21 + A.23 + A.24 + A.25 + A.26 + A.27 + A.28 + A.29 + A.33 + A.34 + A.35 + A.36 + A.39 + A.41 + A.42 + A.43 + A.44 + A.45 + A.46 + A.48 + A.49 + A.52 + A.53 + A.54 + A.56 + A.57,data = train.data,family="binomial"))  
step.pred.probs <- predict(step.model,newdata=test.data,type="response")  
le1 = levels(test.data$spam);le1

## [1] "no" "yes"

step.test.pred <- ifelse(step.pred.probs >= 0.5, le[2],le[1])  
table(step.test.pred,test.y)

## test.y  
## step.test.pred no yes  
## no 801 67  
## yes 30 483

step.test.error <- mean(step.test.pred != test.y)  
round(step.test.error,digit=6)

## [1] 0.070239

### 6)Plot ROC curves and find the area under ROC curve.

library(pROC)

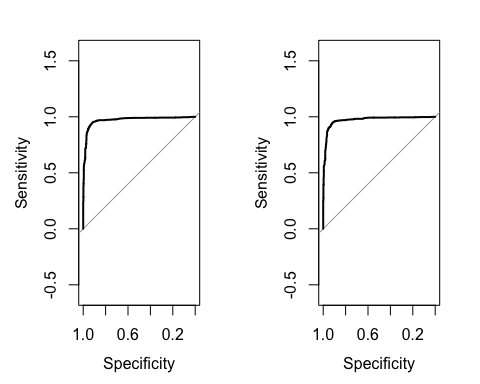
## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following object is masked from 'package:glmnet':  
##   
## auc

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

#logistic roc  
log.roc <- roc(response = test.y,  
 predictor = log.probs,  
 levels = rev(levels(test.y)))   
#stepAIC roc  
stepaic.roc <- roc(response = test.y,  
 predictor = step.pred.probs,  
 levels=rev(levels(test.y)))  
par(mfrow=c(1,2))  
plot(log.roc)  
plot(stepaic.roc)



models.auc <- data.frame('model'=c('logistic model','stepAIC for log'),'AUC'=c(log.roc$auc,stepaic.roc$auc))  
models.auc

## model AUC  
## 1 logistic model 0.9710480  
## 2 stepAIC for log 0.9720403

### 7)Compare two models in error rates. Which model do you prefer?

models.error <- data.frame('model'=c('Logistic Model','stepAIC of logistic'),'error train' = c(cv.log$error,cv.step$Error),'error test'=c(log.test.error,step.test.error))  
models.error

## model error.train error.test  
## 1 Logistic Model 0.08354000 0.06879073  
## 2 stepAIC of logistic 0.07608696 0.07023896

Based on the result, we can see that logistic model with all variable in has higher train error but lower test error. From the book we learnt, we care about test error, therefore, logistic model is perferred.

### 8)Find the best K on knn model using cv.knn on training data. (try K=1:200)

source("cv.R")  
set.seed(123)  
default.knn = NULL; knn.error=NULL  
  
for (i in 1:200) {  
 default.knn<- CV.knn(knn.data=train.data, knn.xname=c('A.1','A.2','A.3','A.4','A.5','A.6','A.7','A.8','A.9','A.10','A.11','A.12','A.13','A.14','A.15','A.16','A.17','A.18','A.19','A.20','A.21','A.22','A.23','A.24','A.25','A.26','A.27','A.28','A.29','A.30','A.31','A.32','A.33','A.34','A.35','A.36','A.37','A.38','A.39','A.40','A.41','A.42','A.43','A.44','A.45','A.46','A.47','A.48','A.49','A.50','A.51','A.52','A.53','A.54','A.55','A.56','A.57'), knn.yname="spam",knn.k.fold=10,knn.seed\_kfold=123,k=i)  
 knn.error[i] <- default.knn$knn\_error  
}  
min\_error <- min(knn.error)  
best.k <- which(knn.error == min\_error)  
print(best.k)

## [1] 13

### 9)What is the 10-fold cross-validated error rate for the knn for the best k on training data.

knn.bestk.error <- CV.knn(knn.data=train.data, knn.xname=c('A.1','A.2','A.3','A.4','A.5','A.6','A.7','A.8','A.9','A.10','A.11','A.12','A.13','A.14','A.15','A.16','A.17','A.18','A.19','A.20','A.21','A.22','A.23','A.24','A.25','A.26','A.27','A.28','A.29','A.30','A.31','A.32','A.33','A.34','A.35','A.36','A.37','A.38','A.39','A.40','A.41','A.42','A.43','A.44','A.45','A.46','A.47','A.48','A.49','A.50','A.51','A.52','A.53','A.54','A.55','A.56','A.57'), knn.yname="spam",knn.k.fold=10,knn.seed\_kfold=123,k=best.k)  
knn.bestk.error <- round(knn.bestk.error$knn\_error,digit=6)  
knn.bestk.error

## [1] 0.102795

the 10 fold cv error rate for the knn for the best k on training data is 0.102795.

### 10)Apply best knn model (best k selected from the training data) to the test data and get the error rate for the test data.

set.seed(123)  
knn.data <- scale(data.class[,c(-58)])  
  
knn.training\_data = scale(knn.data[train.spam,])  
knn.testing\_data = scale(knn.data[test.spam,])  
knn.training\_y <- data.class$spam[train.spam]  
knn.testing\_y <- data.class$spam[test.spam]  
knn.pred\_y = knn(knn.training\_data,knn.testing\_data,knn.training\_y,k=best.k)  
table(knn.pred\_y,knn.testing\_y)

## knn.testing\_y  
## knn.pred\_y no yes  
## no 783 95  
## yes 48 455

knn.test.error = mean(knn.pred\_y != knn.testing\_y)  
knn.test.error1=round(knn.test.error, digit=6)

Based on the result, the test error of knn is 0.103548.

### 11)Build the LDA and QDA model and find 10-fold cross-validate error rates on training data.

source("cv.R")  
#LDA  
lda.train<- CV.lda(lda\_data=train.data,ldamodel=spam~.,lda\_yname="spam",lda\_k=10, lda\_seed=123)  
lda.train.error <- round(lda.train$lda\_error,digit=6)  
lda.train.error

## [1] 0.111801

#QDA  
qda.train <- CV.qda(qda\_data=train.data, qdamodel=spam~.-A.32-A.41, qda\_yname="spam", qda\_k=10, qda\_seed=123)  
# I can't run qda model because A.32 and A.41 have exactly liner relationship, therefore, we need to exclude those two variables.  
qda.train.error <- round(qda.train$qda\_error,digit=6)  
qda.train.error

## [1] 0.165217

### 12)Apply the LDA and QDA models above to the test data. What are the error rates?

#lda  
lda.train.model <- lda(lda.train$call,data=train.data)  
lda\_pred <- predict(lda.train.model,test.data)  
lda\_pred\_y <- lda\_pred$class  
table(lda\_pred\_y, test.y)

## test.y  
## lda\_pred\_y no yes  
## no 797 125  
## yes 34 425

lda.test.error <- mean(lda\_pred\_y != test.y)  
round(lda.test.error,digit=6)

## [1] 0.115134

#QDA  
qda.train.model <- qda(qda.train$call,data=train.data)  
qda.pred <- predict(qda.train.model, test.data)  
qda\_pred\_y <- qda.pred$class  
table(qda\_pred\_y,test.y)

## test.y  
## qda\_pred\_y no yes  
## no 650 30  
## yes 181 520

qda.test.error <- mean(qda\_pred\_y != test.y)  
round(qda.test.error,digit=6)

## [1] 0.152788

Based on the result, the test error of lda is 0.115134, the test error of qda is 0.152788.

### 13)Run lasso on training data and get 10-fold cross-validate error rates on training data.

source("cv.R")  
lasso.train <- cv.Lasso(data=train.data,model=spam~.,yname="spam", Kfold=10,seed=123, alpha=1)  
lasso.train.error <- round(lasso.train$lasso.error,digit=6)  
lasso.train.error

## [1] 0.074534

### 14)Apply the lasso model (training data and the best lamda) to the test data. Get the error rates.

lasso.trainx = model.matrix(lasso.train$call,data=train.data)[,-1]  
lasso.trainy = data.class$spam[train.spam]  
lasso.testx <- scale(test.data[,-58])  
lasso.train.model = glmnet(lasso.trainx,lasso.trainy,alpha=1,family = "binomial",lambda =lasso.train$bestlam)  
lasso.prob = predict(lasso.train.model,s=lasso.train$bestlam,newx=lasso.testx,type="response")  
le3 <- levels(test.data$spam);le3

## [1] "no" "yes"

lasso.pred <- ifelse(lasso.prob >= 0.5,le3[2],le3[1])  
lasso.test.error <- mean(lasso.pred != test.y)  
lasso.test.error

## [1] 0.1035482

The error rate of lasso is 0.1035482.

### 15)Among the six models (glm with all variables in, stepAICglm, lasso, knn, lda, qda) which one do you prefer based on training data? (Must compare cross-validated errors on training data)

training.error <- data.frame('Model'=c('glm with all variable-in','stepAICglm','knn','lda','qda','lasso'),'train error'=c(cv.log$error,cv.step.error,knn.bestk.error,lda.train.error,qda.train.error,lasso.train.error))  
training.error

## Model train.error  
## 1 glm with all variable-in 0.083540  
## 2 stepAICglm 0.076087  
## 3 knn 0.102795  
## 4 lda 0.111801  
## 5 qda 0.165217  
## 6 lasso 0.074534

based on the result, we select the lowest training error which is lasso model with best lambda.

### 16)Among the six models (glm with all variables in, stepAICglm, lasso, knn, lda, qda) which one do you prefer based on predictions on test data? (Must use training data for modling and apply the fitted model to the test data.)

testing.error <- data.frame('Model'=c('glm with all variable-in','stepAICglm','knn','lda','qda','lasso'),'test error'=c(log.test.error,step.test.error,knn.test.error,lda.test.error,qda.test.error,lasso.test.error))  
testing.error

## Model test.error  
## 1 glm with all variable-in 0.06879073  
## 2 stepAICglm 0.07023896  
## 3 knn 0.10354815  
## 4 lda 0.11513396  
## 5 qda 0.15278783  
## 6 lasso 0.10354815

According to the result, we select the lowest test error, which is glm with all variable-in.