HW2

20989977 Zhang Mingtao

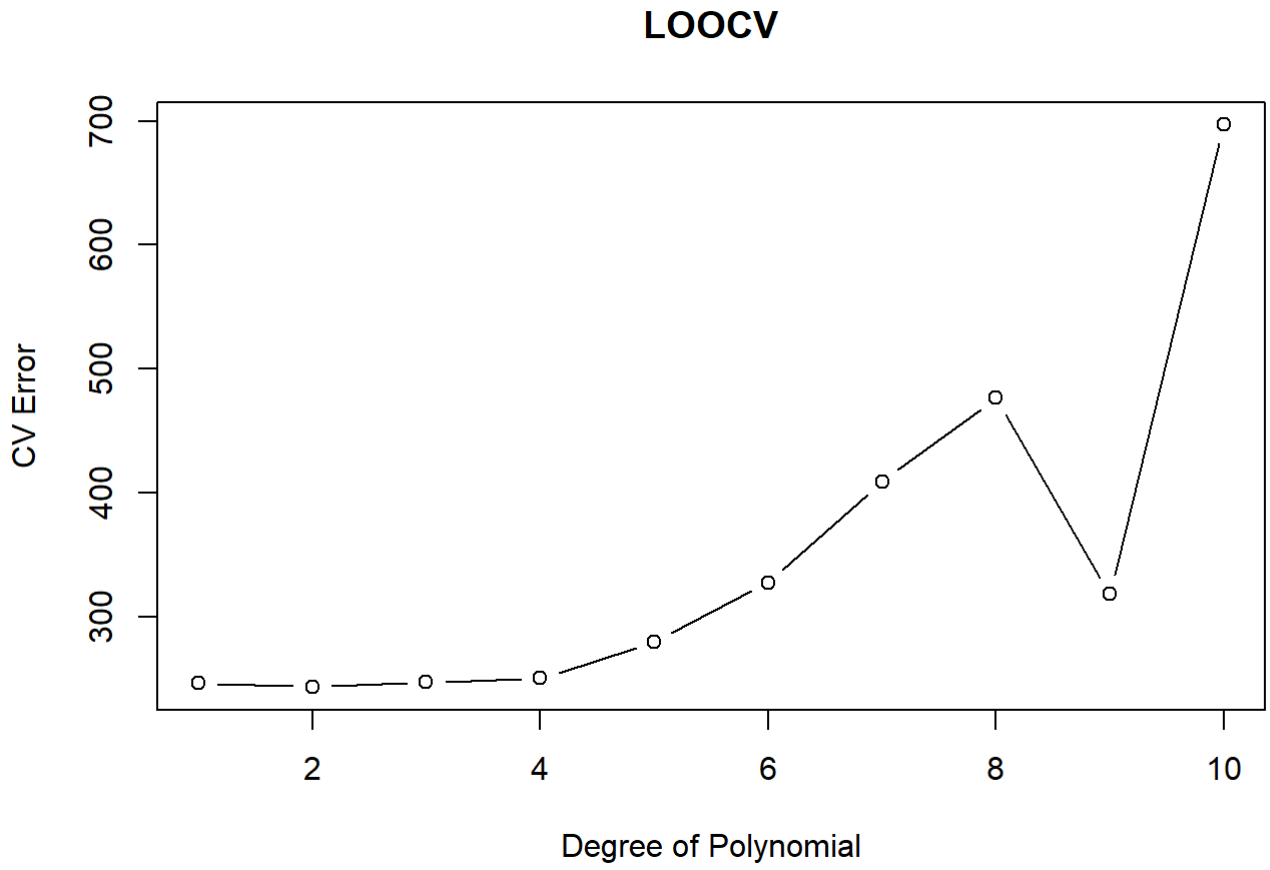
2023/10/12

1.

```
#(1)
library(boot)
#1
car_df=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//cars.csv")
cv.error=rep(0,10)
for (i in 1:10){
  set.seed(1)
  glm.fit=glm(dist~poly(speed,i),data=car_df)
  cv.error[i]=cv.glm(car_df,glm.fit)$delta[1]
}
cv.error
```

```
##  [1] 246.4054 243.0292 246.8288 250.0914 279.6864 327.5014 408.9479 476.4366
##  [9] 318.1917 696.9015
```

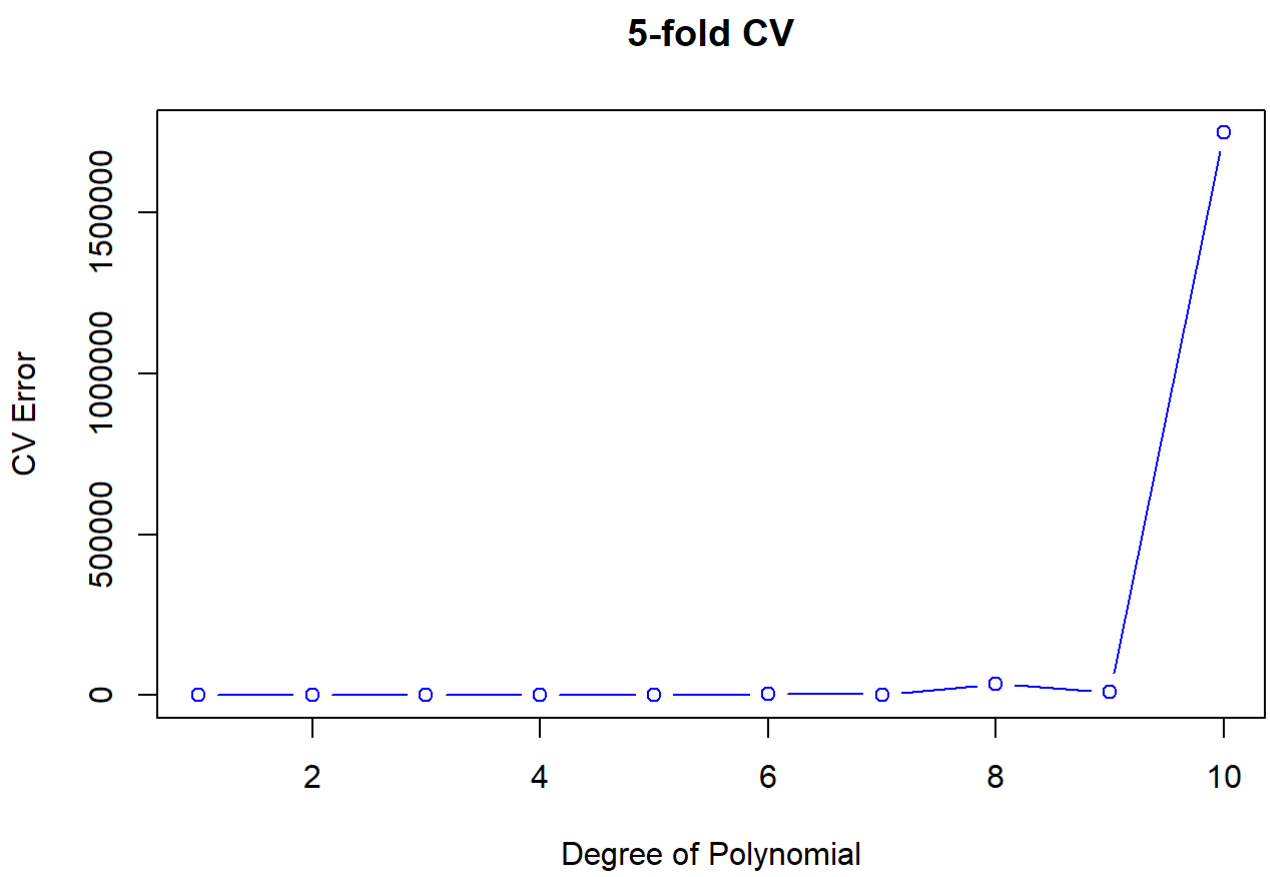
```
plot(1:10,cv.error,type='b',xlab="Degree of Polynomial",ylab="CV Error",main="LOOCV")
```



```
#We can choose the i=2 to minimize error and model complexity.

#2
cv.error.5=rep(0,10)
for (i in 1:10){
  set.seed(1)
  glm.fit2=glm(dist~poly(speed,i),data=car_df)
  cv.error.5[i]=cv.glm(car_df,glm.fit2,K=5)$delta[1]     ## K=10 means 10-fold cross validation
}
cv.error.5
```

```
##  [1]     238.8155     226.9948     319.2785     273.8090     321.6122
##  [6]    4811.9276    2105.4446   34708.1047    9409.7427 1748017.0426
```

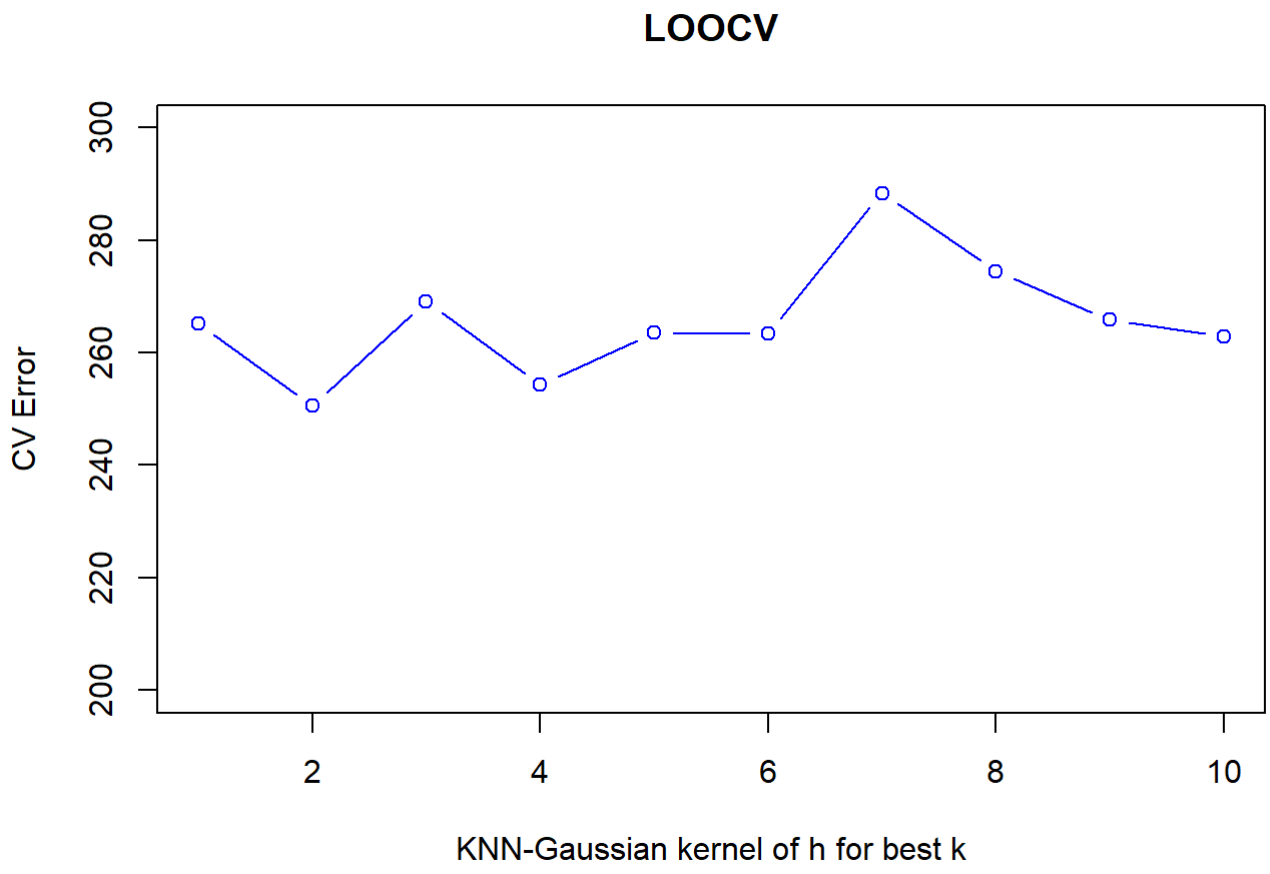
```
plot(1:length(cv.error.5),cv.error.5,type='b',col='blue',xlab='Degree of Polynomial',ylab='CV E
rror',main='5-fold CV')
#We can choose the i=2 to minimize error and model complexity.

#3
library(class)
library(kknn)
```



```
# trainx=car_df[,1]
# trainy=car_df[,2]

#留一法交叉验证选择最佳带宽
# loo_result=train.kknn(dist~.,car_df,kernel="gaussian",scale=T,kmax=10,tune=TRUE,kcv=nrow(car_df))
error_LOO=rep(0,10)
set.seed(1)
for (i in 1:10){
  tentmodel=train.kknn(dist~.,car_df,kernel = "gaussian",distance=i,scale=T,kmax=10)
  tenterror=cv.kknn(tentmodel,car_df,kcv=nrow(car_df))[[2]][2]
  # tenterror=cv.kknn(dist~.,car_df,kernel = "gaussian",distance=i,scale=T,,kcv=nrow(car_df))[[2]][2]
  error_LOO[i]=tenterror
}
error_LOO
```

```
##  [1] 265.2218 250.5974 269.1518 254.2604 263.5870 263.3651 288.4120 274.4839
##  [9] 265.8238 262.8714
```

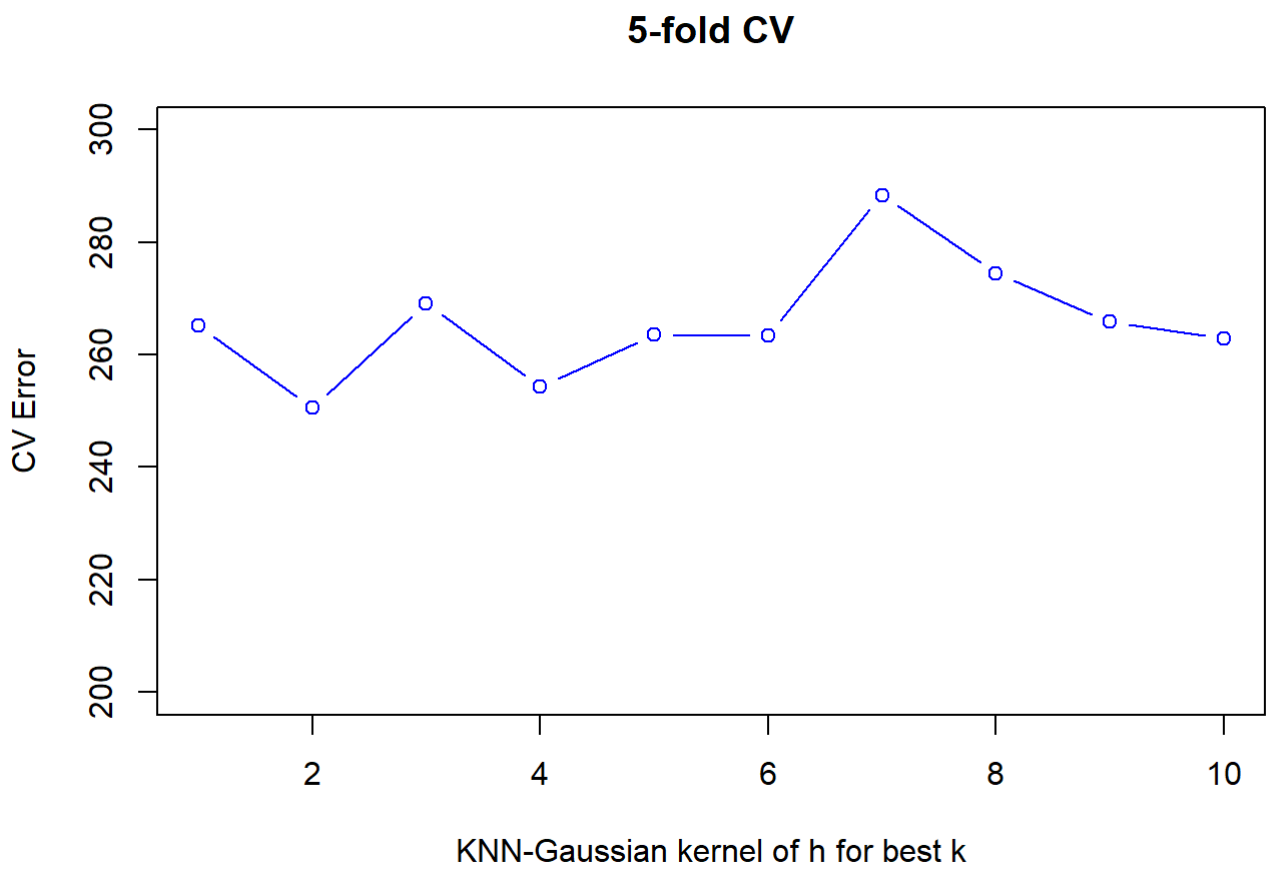
```
plot(1:length(error_LOO),error_LOO,type='b',col='blue',xlab='KNN-Gaussian kernel of h for best k',ylab='CV Error',main='LOOCV',ylim=c(200,300))
```



```
#h=2


# 使用5折交叉验证选择最佳带宽
# kf_result=train.kknn(dist~.,car_df,kernel="gaussian",scale=T,kmax=10,tune=TRUE,kcv=5)
error_K5=rep(0,5)
set.seed(1)
for (i in 1:10){
  tentmodel=train.kknn(dist~.,car_df,kernel = "gaussian",distance=i,scale=T,kmax=10)
  tenterror=cv.kknn(tentmodel,car_df,kcv=5)[[2]][2]
  # tenterror=cv.kknn(dist~.,car_df,kernel = "gaussian",distance=i,scale=T,,kcv=nrow(car_df))
[[2]][2]
  error_K5[i]=tenterror
}
error_K5
```

```
##  [1] 294.2271 229.9909 348.4192 304.3537 272.1031 319.1875 249.7708 254.5507
##  [9] 275.4869 273.9370
```

```
plot(1:length(error_K5),error_LOO,type='b',col='blue',xlab='KNN-Gaussian kernel of h for best
k',ylab='CV Error',main='5-fold CV',ylim=c(200,300))
```



```
#h=2

#4
# Taking the minimun error(MSE) and the simpler model into consideration:
# polynomial regression: i=2
min(cv.error)
```

```
## [1] 243.0292
```

```
min(cv.error.5)
```

```
## [1] 226.9948
```

```
# KNN-Gaussian kernel: h=2 kbest
min(error_LOO)
```

```
## [1] 250.5974
```

```
min(error_K5)
```

```
## [1] 229.9909
```

```
# And the 5-fold CV is faster and preciser than LOOCV.
```

2.

```
#(2)
titanic_df=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//titanic.csv")
titanic_df=na.omit(titanic_df)

#因子型处理+舍弃共线性变量(female/Pclass=2)
titanic_df$Sex[which(titanic_df$Sex== "female")]=1
titanic_df$Sex[which(titanic_df$Sex== "male")]=0
titanic_df$Sex=as.factor(titanic_df$Sex)

titanic_df$Pclass[which(titanic_df$Pclass==1)]=11
titanic_df$Pclass[which(titanic_df$Pclass==2)]=22
titanic_df$Pclass=as.factor(titanic_df$Pclass)

#建模
glmod1=glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=titanic_df,family=binomial)
summary(glmod1)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Fare, family = binomial,
##     data = titanic_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8192  -0.6436  -0.3845   0.6282   2.4471
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.970314   0.276945  -3.504 0.000459 ***
## Pclass11     2.525790   0.334601   7.549 4.40e-14 ***
## Pclass22     1.217522   0.246563   4.938 7.89e-07 ***
## Sex1         2.616107   0.215413  12.145  < 2e-16 ***
## Age         -0.044147   0.008262  -5.343 9.13e-08 ***
## SibSp       -0.393449   0.123221  -3.193 0.001408 **
## Fare         0.001750   0.002455   0.713 0.475990
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 636.03  on 707  degrees of freedom
## AIC: 650.03
##
## Number of Fisher Scoring iterations: 5
```

```
coef(glmod1)
```

```
##  (Intercept)     Pclass11     Pclass22         Sex1          Age        SibSp
## -0.970313611  2.525789746  1.217522120  2.616107108 -0.044146772 -0.393448732
##         Fare
##  0.001750058
```

```
confint(glmod1,level=0.95)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %       97.5 %
## (Intercept) -1.51866506 -0.431345669
## Pclass11     1.87617451  3.191602285
## Pclass22     0.73784428  1.705974784
## Sex1         2.20328715  3.048940312
## Age         -0.06071739 -0.028280932
## SibSp       -0.64335524 -0.158928341
## Fare        -0.00281513  0.007146204
```

```
# coefficients for Sex/male: -2.616107108
# coefficients for pclass/3rd: -1.217522120
# 95% confidence intervals for Sex/male: [-3.04894031,-2.203287151]
# 95% confidence intervals for pclass/3rd: [-1.70597478,-0.737844276]


#或者将因子型变量视为数值型变量不做处理建模：
titanic_df2=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//titanic.csv")
titanic_df2=na.omit(titanic_df2)
titanic_df2$Sex[which(titanic_df2$Sex== "female")]=1
titanic_df2$Sex[which(titanic_df2$Sex== "male")]=0
titanic_df2$Sex=as.integer(titanic_df2$Sex)

glmod2=glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=titanic_df2,family=binomial)
summary(glmod2)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Fare, family = binomial,
##     data = titanic_df2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8136  -0.6538  -0.3834   0.6333   2.4488
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.776757   0.546233   5.083 3.71e-07 ***
## Pclass      -1.254155   0.161315  -7.775 7.57e-15 ***
## Sex          2.613101   0.214887  12.160  < 2e-16 ***
## Age         -0.043923   0.008176  -5.372 7.77e-08 ***
## SibSp       -0.392416   0.123149  -3.187  0.00144 **
## Fare         0.001870   0.002387   0.784  0.43331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 636.07  on 708  degrees of freedom
## AIC: 648.07
##
## Number of Fisher Scoring iterations: 5
```

```
coef(glmod2)
```

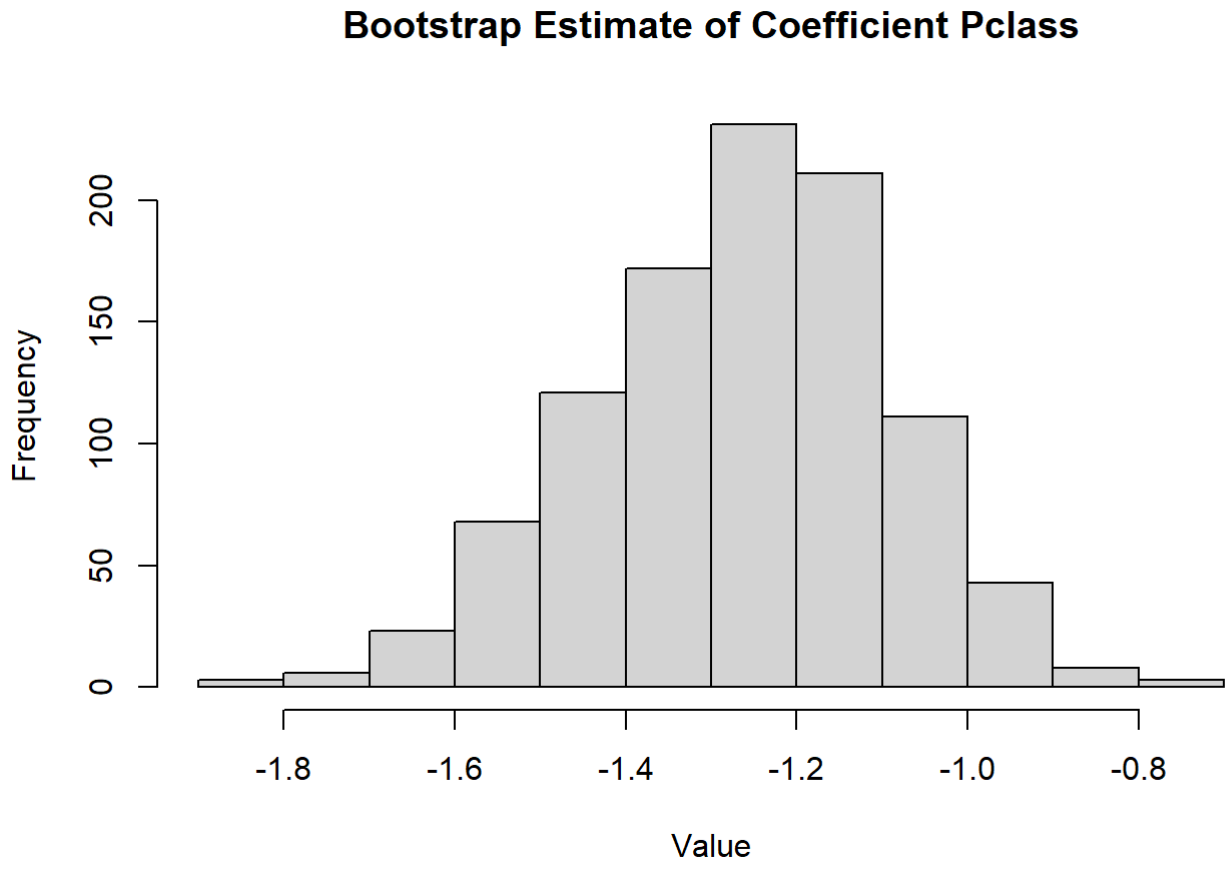
```
##  (Intercept)       Pclass          Sex          Age        SibSp         Fare
##  2.776756629 -1.254155279  2.613101036 -0.043923317 -0.392416329  0.001870447
```

```
confint(glmod2,level=0.95)
```

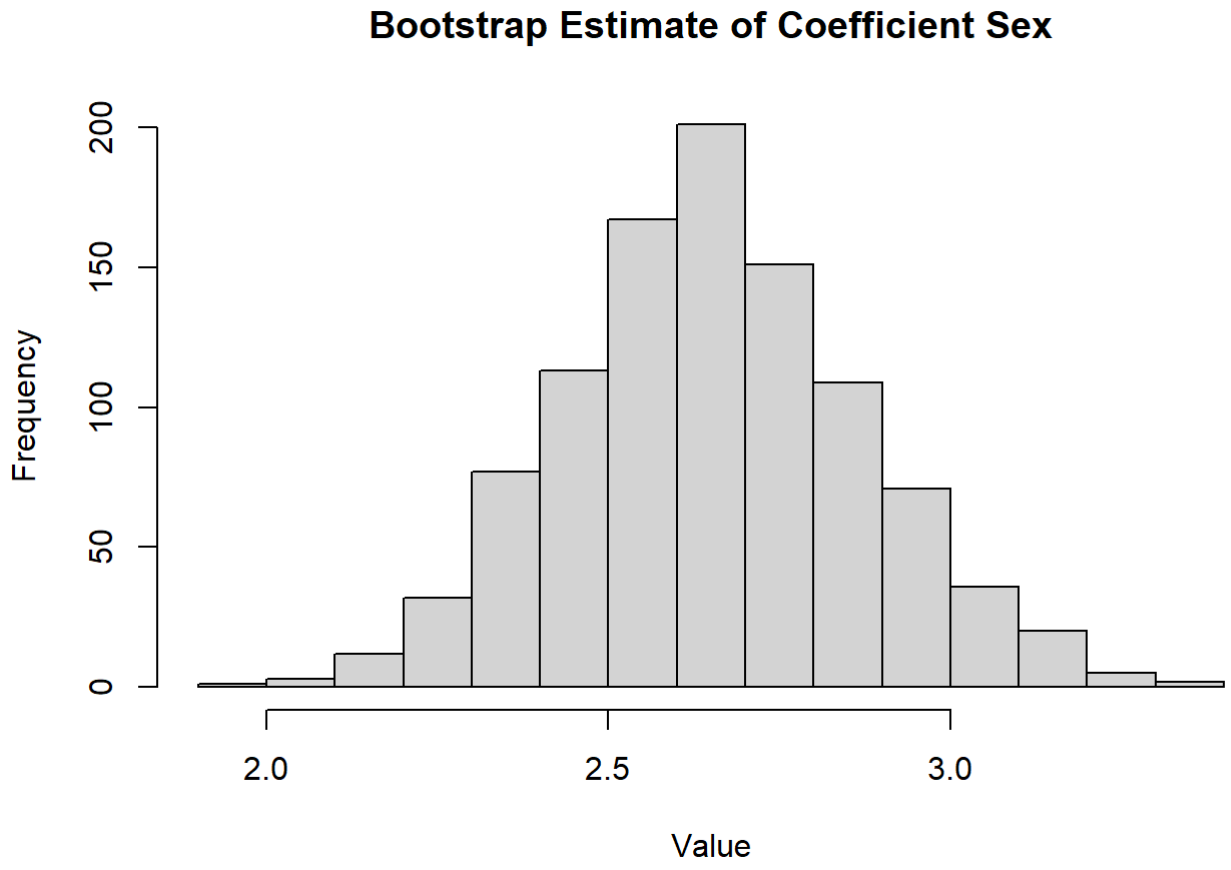
```
## Waiting for profiling to be done...
```

```
##                    2.5 %       97.5 %
## (Intercept)  1.716283188  3.862508989
## Pclass      -1.575659790 -0.941778047
## Sex          2.201391972  3.045015754
## Age         -0.060312776 -0.028212668
## SibSp       -0.642252358 -0.158094322
## Fare        -0.002555242  0.007111004
```

```
# coefficients for Sex: 2.613101036
# coefficients for pclass: -1.254155279
# 95% confidence intervals for Sex: [2.201391972,3.045015754]
# 95% confidence intervals for Pclass: [-1.575659790,-0.941778047]

#此时pclass差别相对大些，因为第一个模型舍弃了Pclass=2的因子变量
#为统一起见后续几问均采用titanic_df2数据，全部视作数值型变量处理

#2
n=nrow(titanic_df2)

coef_fun=function(df,index){
  df=df[index,]
  fit=glm(Survived~Pclass+Sex+Age+SibSp+Fare,data=df,family=binomial)
  return(coef(fit,level=0.95))
}

# coef_fun(titanic_df2,sample(n,n,replace=T))

set.seed(1) # 设置随机数种子
boot.res=boot(titanic_df2,coef_fun,R=1000)

hist(boot.res$t[,2],xlab="Value",main="Bootstrap Estimate of Coefficient Pclass")
```




```
hist(boot.res$t[,3],xlab="Value",main="Bootstrap Estimate of Coefficient Sex")
```



```
quantile(boot.res$t[,3], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##     2.5%    97.5%
## 2.247117 3.104147
```

```
# 95% confidence intervals for Sex: [2.247117,3.104147]
quantile(boot.res$t[,2], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##       2.5%      97.5%
## -1.6088958 -0.9433257
```

```
# 95% confidence intervals for Pclass: [-1.6088958,-0.9433257]

# The results of the 95% confidence intervals are about the same as that in #1.

#3
# Take Parch into consideration:
glmod3=glm(Survived~Pclass+Sex+Age+SibSp+Fare+Parch,data=titanic_df2,family=binomial)
summary(glmod3)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Fare +
##     Parch, family = binomial, data = titanic_df2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7953  -0.6476  -0.3847   0.6271   2.4433
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.754158   0.548818   5.018 5.21e-07 ***
## Pclass      -1.242249   0.163191  -7.612 2.69e-14 ***
## Sex          2.634845   0.219609  11.998  < 2e-16 ***
## Age         -0.043953   0.008179  -5.374 7.70e-08 ***
## SibSp       -0.375755   0.127361  -2.950  0.00317 **
## Fare         0.002160   0.002493   0.866  0.38627
## Parch       -0.061937   0.122925  -0.504  0.61436
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 635.81  on 707  degrees of freedom
## AIC: 649.81
##
## Number of Fisher Scoring iterations: 5
```

```
coef(glmod3)
```

```
##  (Intercept)       Pclass          Sex          Age        SibSp         Fare
##  2.754158272 -1.242248625  2.634844835 -0.043952596 -0.375754871  0.002160034
##        Parch
## -0.061937366
```

```
confint(glmod3,level=0.95)
```

```
## Waiting for profiling to be done...
```

```
##                    2.5 %       97.5 %
## (Intercept)  1.687178447  3.844026242
## Pclass      -1.567042098 -0.925558640
## Sex          2.214253357  3.076432044
## Age         -0.060347810 -0.028235954
## SibSp       -0.633493997 -0.132852389
## Fare        -0.002412263  0.007697787
## Parch       -0.309930305  0.175802874
```

```
# It seems that Parch is not a significant variable.

#4
train=titanic_df2[-1,c(2,3,5,6,7,10)]
test=titanic_df2[1,c(2,3,5,6,7,10)]

glmod4=glm(Survived~., data=train,family=binomial)
summary(glmod4)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8128  -0.6546  -0.3831   0.6349   2.4477
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.774662   0.546098   5.081 3.76e-07 ***
## Pclass      -1.252742   0.161300  -7.767 8.07e-15 ***
## Sex          2.610786   0.214897  12.149  < 2e-16 ***
## Age         -0.043908   0.008173  -5.372 7.78e-08 ***
## SibSp       -0.391615   0.123094  -3.181  0.00147 **
## Fare         0.001869   0.002386   0.783  0.43351
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 963.47  on 712  degrees of freedom
## Residual deviance: 635.88  on 707  degrees of freedom
## AIC: 647.88
##
## Number of Fisher Scoring iterations: 5
```

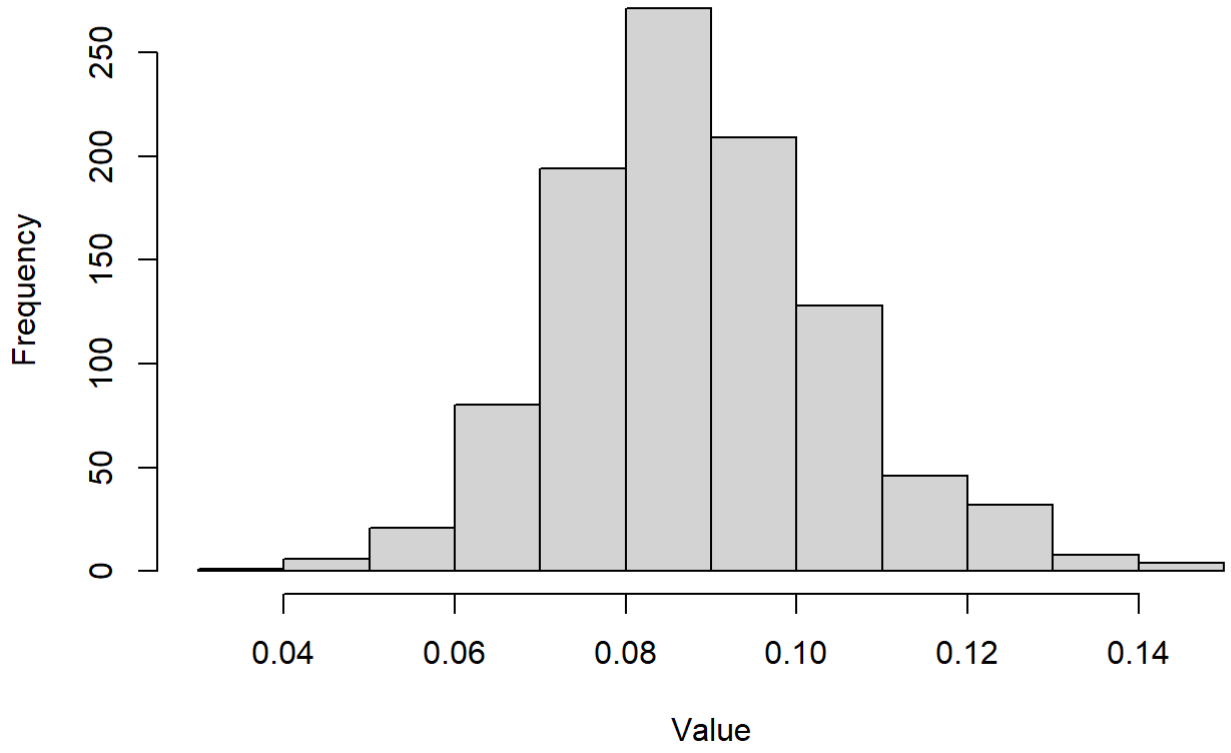
```
pred1=predict(glmod4,test,interval="prediction",type="response")
pred1 #判断能活(=1)的概率为0.08886329
```

```
##          1
## 0.08886329
```

```
as.numeric(ifelse(pred1>0.5,1,0)) #(Survived=0)
```

```
## [1] 0
```

```
#####  Bootstrap  #####
n=nrow(train)

prop_fun=function(df,index){
  df=df[index,]
  fit=glm(Survived~., data=df,family=binomial)
  pred1=predict(fit,test,interval="prediction",type="response")
  return(pred1)
}

set.seed(1) # 设置随机数种子
boot.res2=boot(train,prop_fun,R=1000)

hist(boot.res2$t,xlab="Value",main="Bootstrap Estimate of Survive Probability")
```




```
quantile(boot.res2$t, probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##       2.5%      97.5% 
## 0.05940475 0.12233676
```

```
# 95% confidence intervals for Survive Probability: [0.05940475,0.12233676]

#5
library(MASS)
qda.fit=qda(Survived~.,data=train)
qda.fit  #Prior probabilities of groups先验概率
```

```
## Call:
## qda(Survived ~ ., data = train)
##
## Prior probabilities of groups:
##         0         1 
## 0.5932679 0.4067321 
##
## Group means:
##     Pclass       Sex      Age     SibSp     Fare
## 0 2.484634 0.1513002 30.64657 0.5248227 23.00261
## 1 1.872414 0.6793103 28.34369 0.4931034 51.84321
```

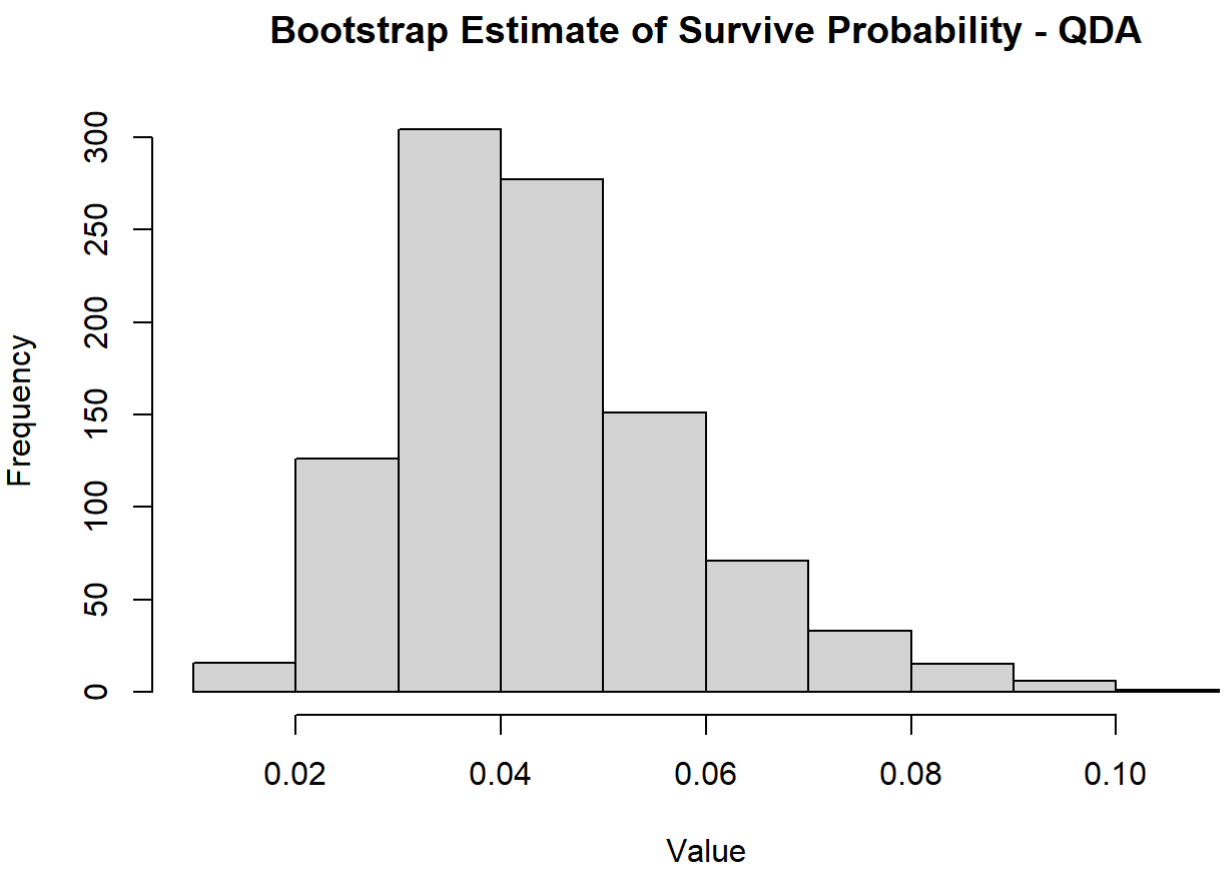
```
qda.pred=predict(qda.fit,test)
qda.pred$class  #预测的所属类的结果;后验概率为qda.pred$posterior
```

```
## [1] 0
## Levels: 0 1
```

```
qda.pred$posterior[2] #判断能活(=1)的概率为0.04359962
```

```
## [1] 0.04359962
```

```
#####  Bootstrap  #####
n=nrow(train)

qda_prop_fun=function(df,index){
  df=df[index,]
  fit=qda(Survived~.,data=df)
  qda.pred=predict(fit,test,interval="prediction",type="response")
  return(qda.pred$posterior[2])
}

set.seed(1) # 设置随机数种子
boot.res3=boot(train,qda_prop_fun,R=1000)

hist(boot.res3$t,xlab="Value",main="Bootstrap Estimate of Survive Probability - QDA")
```



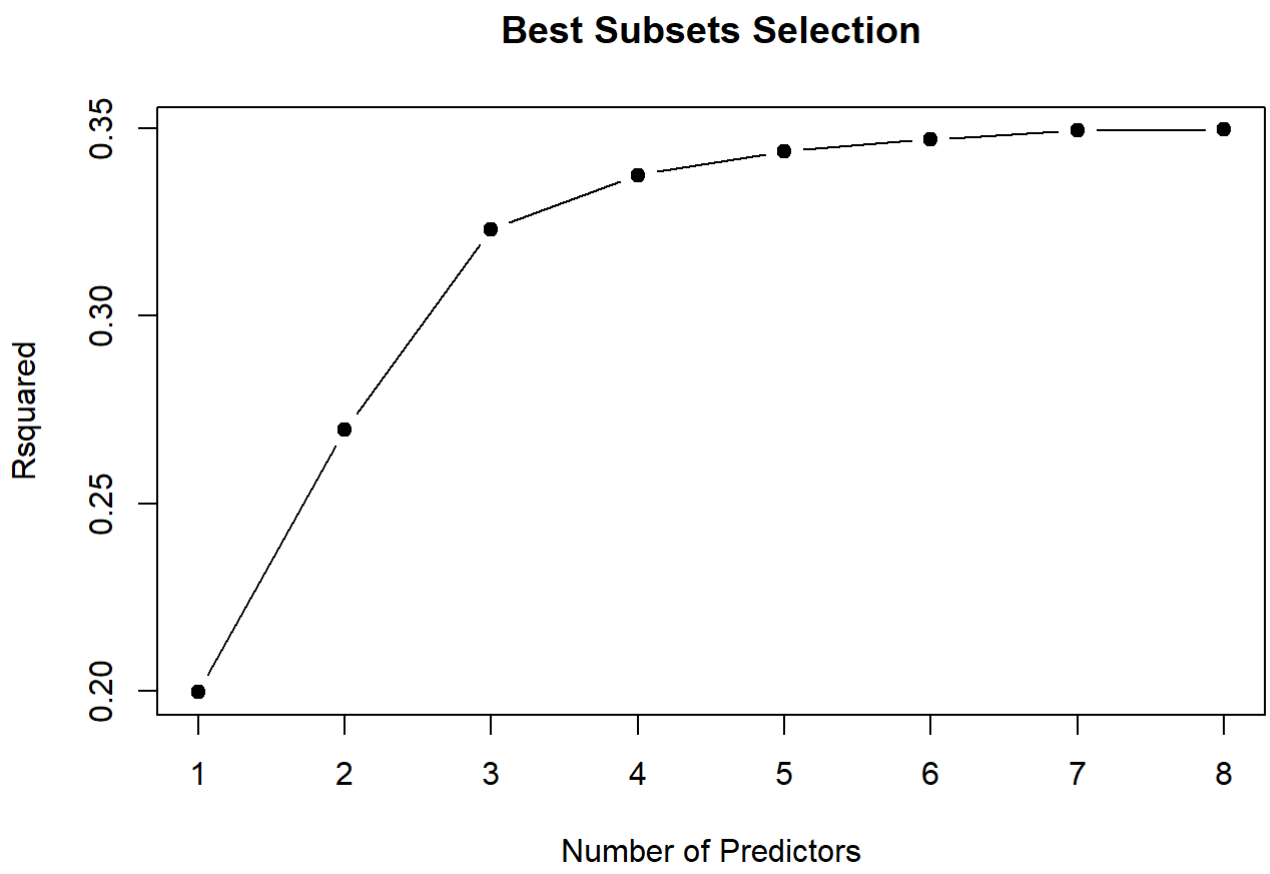

```
quantile(boot.res3$t, probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##       2.5%      97.5%
## 0.02116826 0.07719099
```

```
# 95% confidence intervals for Survive Probability (QDA): [0.02116826,0.07719099 ]

# The probability of QDA model is likely smaller than that of logistic regression model.
```

3.

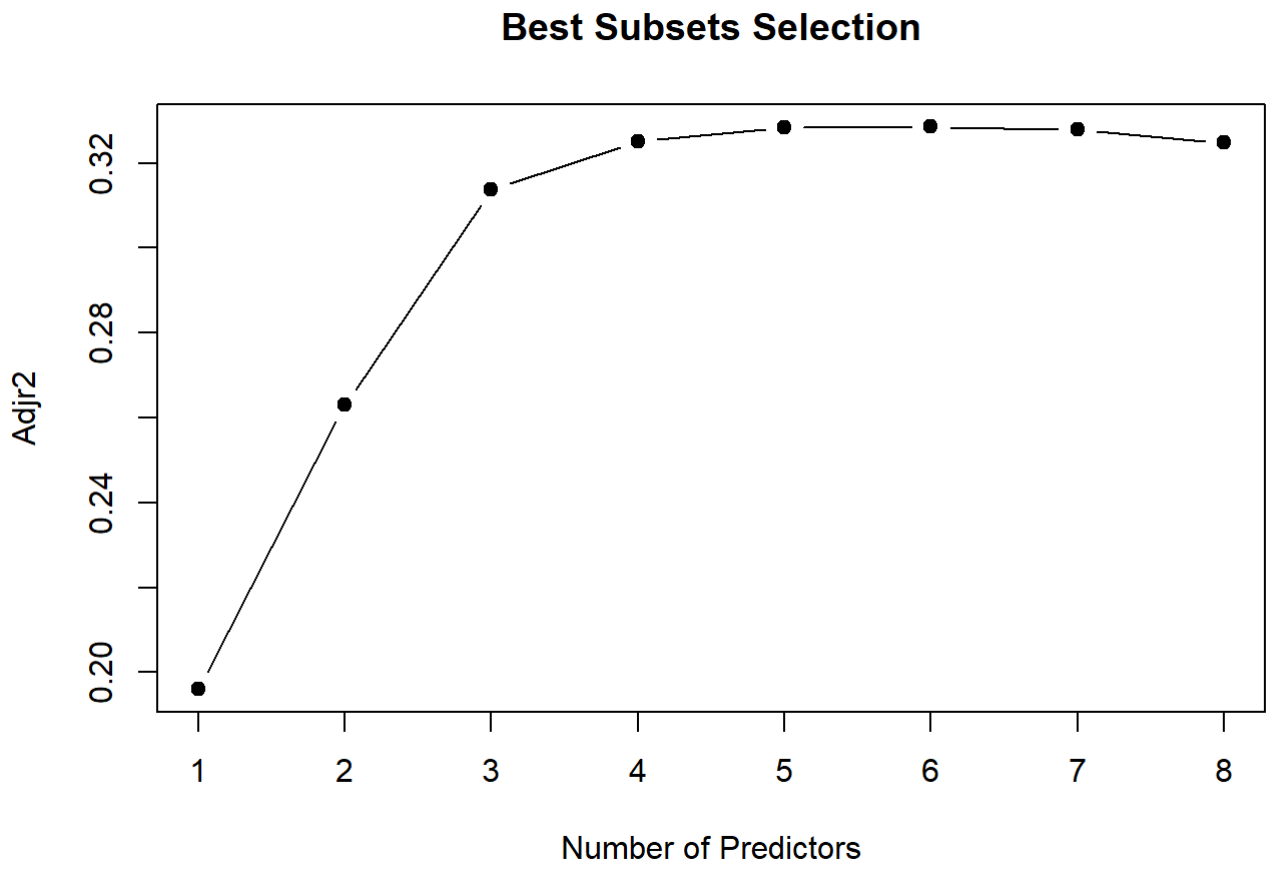
```
#(3)
library(leaps)
#1
GPAdf=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//FirstYearGPA.csv")

regfit.full1=regsubsets(GPA~.,data=GPAdf,method="exhaustive",nbest=1,nvmax=8)
full.sum1=summary(regfit.full1)
plot(as.numeric(row.names(full.sum1$which)),full.sum1$rsq,pch=19,col="black",type="b",xlab="Number of Predictors",ylab="Rsquared",main="Best Subsets Selection")
```




```
full.sum1$adjr2
```

```
## [1] 0.1960203 0.2629543 0.3136440 0.3251153 0.3283423 0.3285134 0.3278000
## [8] 0.3247430
```

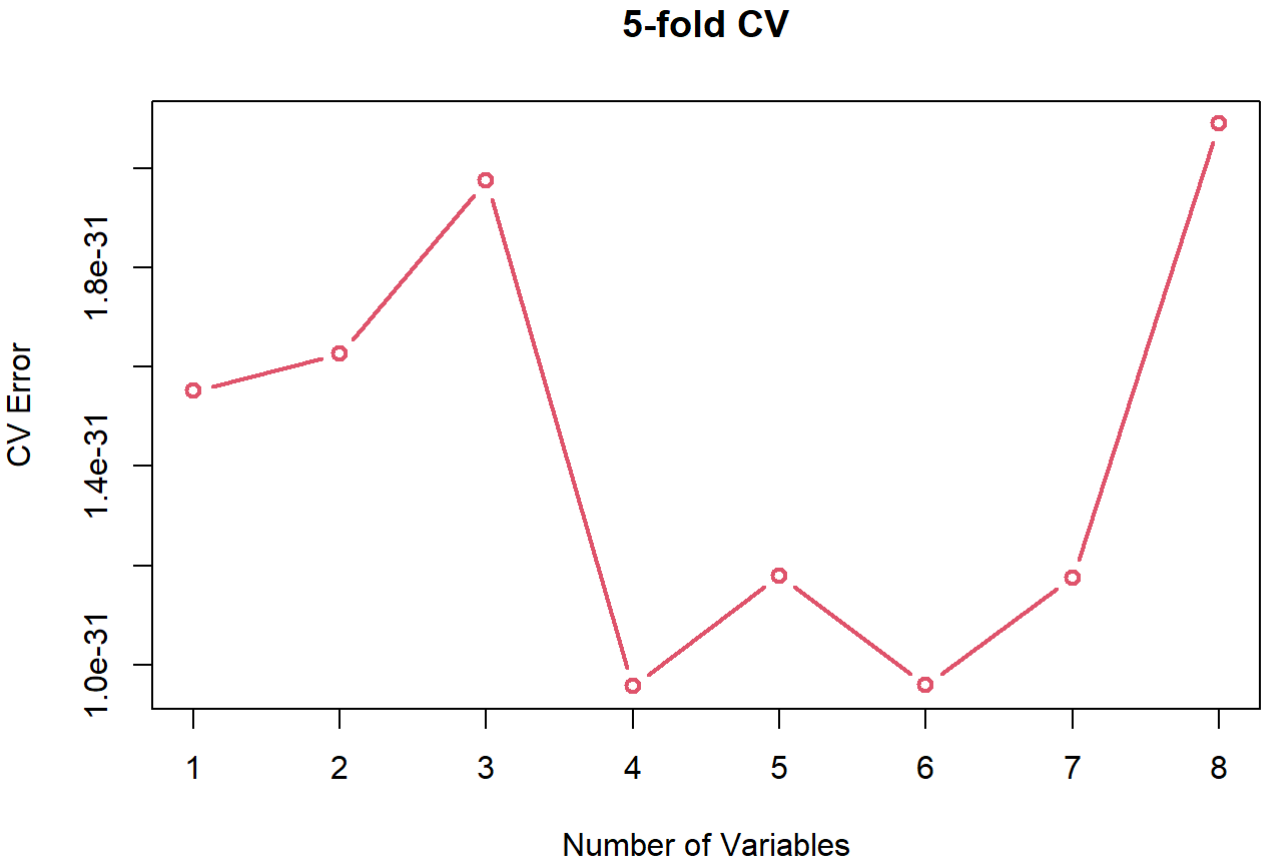
```
plot(as.numeric(row.names(full.sum1$which)),full.sum1$adjr2,pch=19,col="black",type="b",xlab="Number of Predictors",ylab="Adjr2",main="Best Subsets Selection")
```




```
# 4 number of predictors is the best model using adjusted R-square, considering the model complexity.

#2
library(boot)
CV5.err=rep(0,8)
set.seed(1)
for(p in 1:8){
  x=which(summary(regfit.full1)$which[p,])
  x=as.numeric(x)
  x=x[-1]-1
  dfname=c("GPA",names(GPAdf)[x])
  newGPAdf=GPAdf[,dfname]
  glm.fit=glm(GPA~.,data=newGPAdf)
  cv.err=cv.glm(newGPAdf,glm.fit,K=5)
  CV5.err[p]=cv.err$delta[1] #残差平方和
}
CV5.err
```

```
## [1] 1.551156e-31 1.625450e-31 1.974404e-31 9.568090e-32 1.179689e-31
## [6] 9.590603e-32 1.175187e-31 2.089221e-31
```

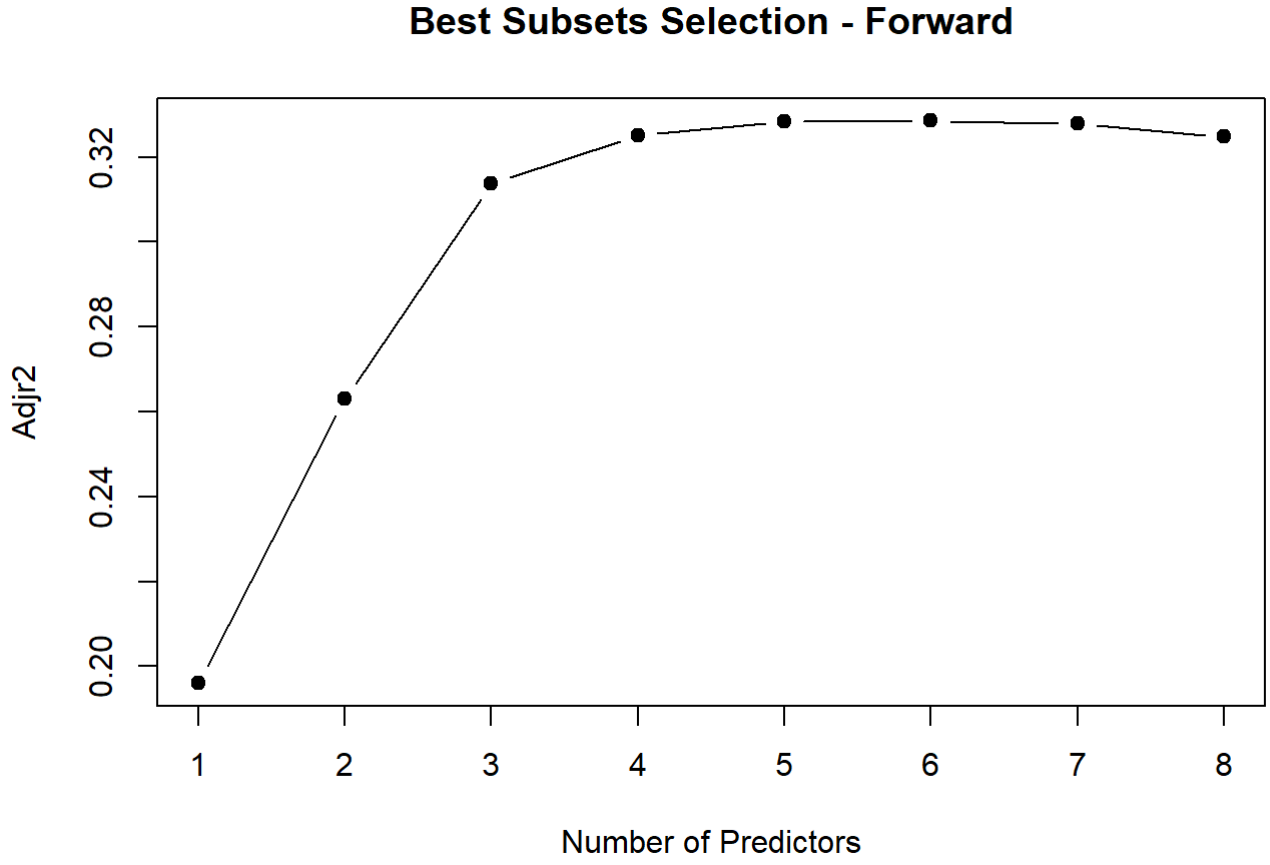
```
plot(1:8,CV5.err,type="b",lwd=2,col=2,xlab="Number of Variables",ylab="CV Error",main="5-fold CV")
```




```
# We can also choose 4 number of predictors to be the best model using 5-fold CV, considering the model complexity.

#3
########## Forward Subset Selection
regfit.fwd=regsubsets(GPA~.,data=GPAdf,method="forward",nvmax=8)
fwd.sum=summary(regfit.fwd)

fwd.sum$adjr2
```

```
## [1] 0.1960203 0.2629543 0.3136440 0.3251153 0.3283423 0.3285134 0.3278000
## [8] 0.3247430
```

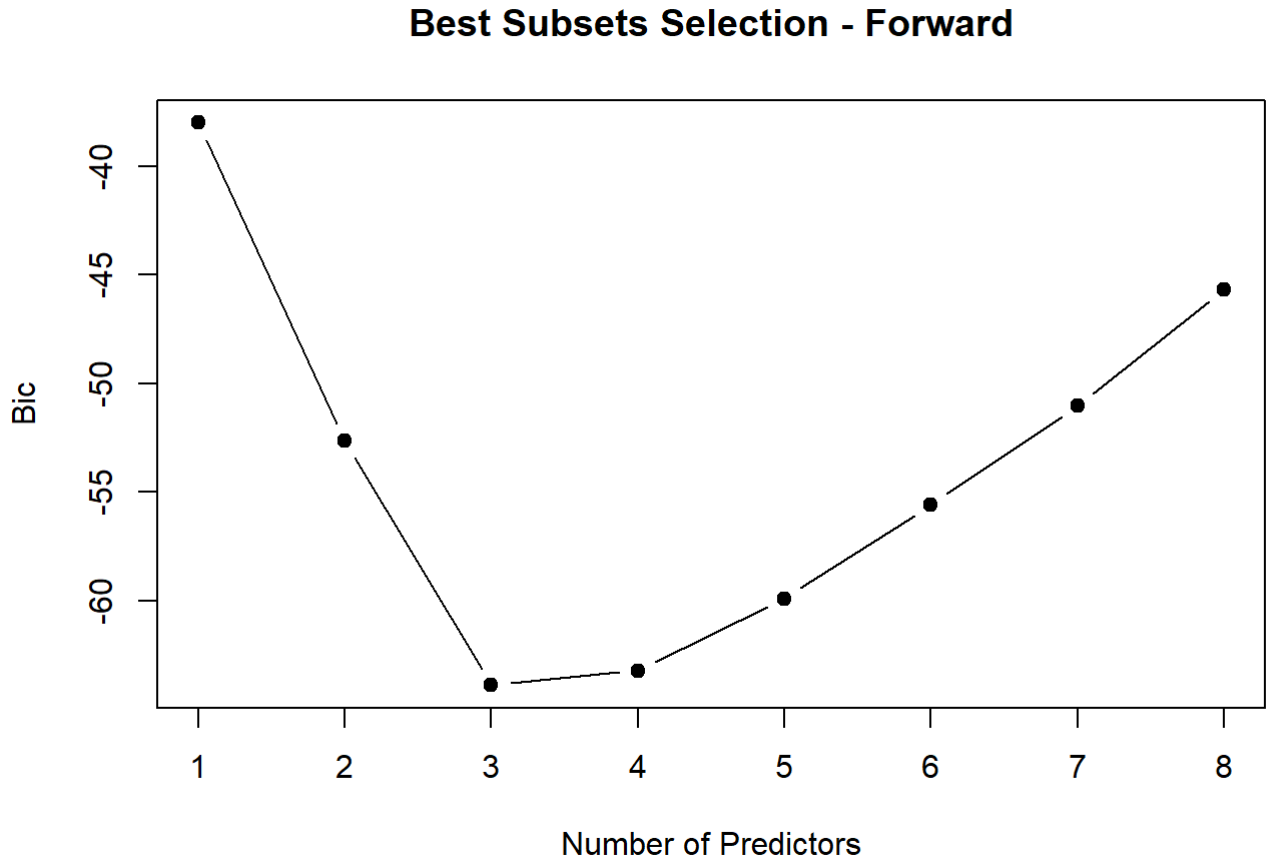
```
plot(as.numeric(row.names(fwd.sum$which)),fwd.sum$adjr2,pch=19,col="black",type="b",xlab="Number of Predictors",ylab="Adjr2",main="Best Subsets Selection - Forward")
```




```
# The result is the same as the exhaustive method.

fwd.sum$bic
```

```
## [1] -38.01046 -52.66931 -63.90100 -63.22406 -59.91041 -55.60773 -51.02158
## [8] -45.67917
```

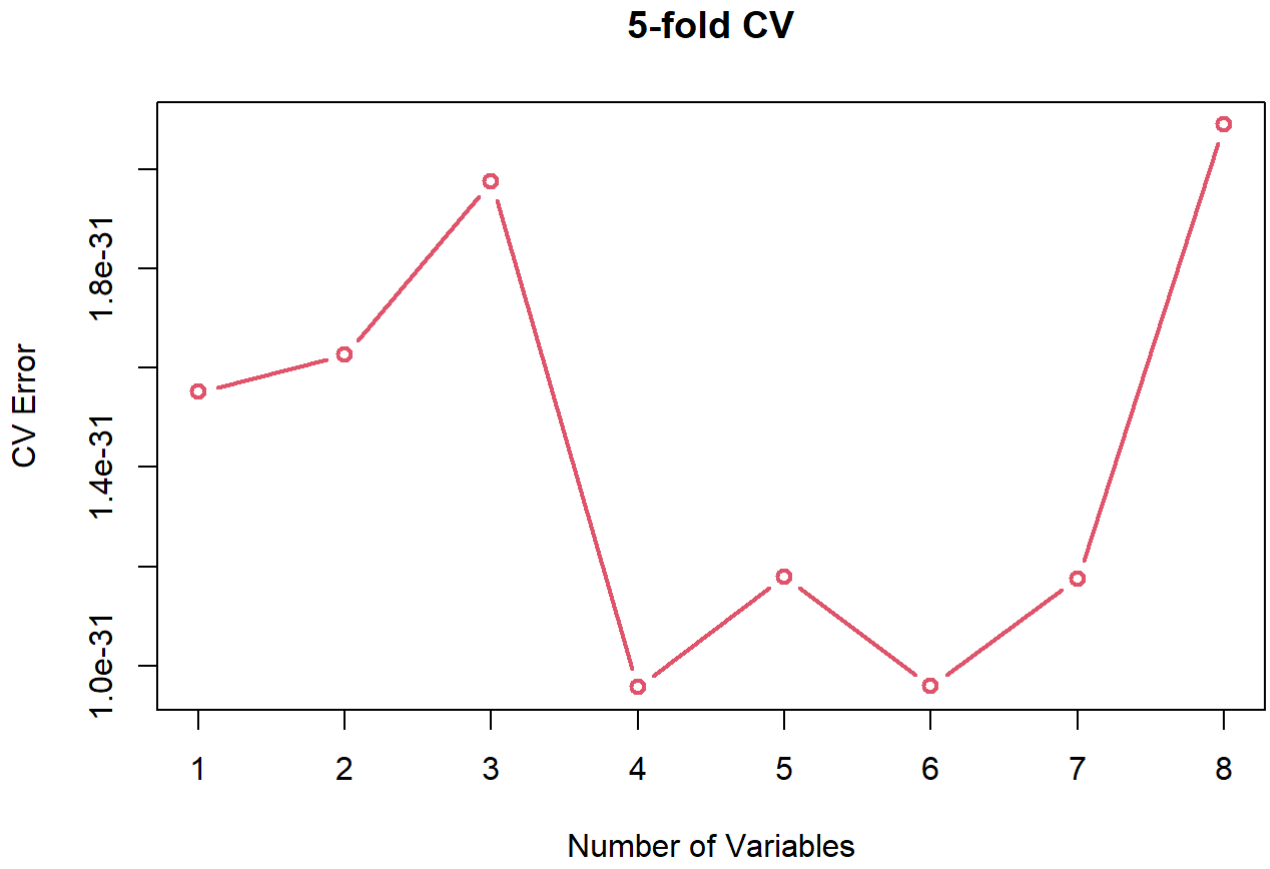
```
plot(as.numeric(row.names(fwd.sum$which)),fwd.sum$bic,pch=19,col="black",type="b",xlab="Number of Predictors",ylab="Bic",main="Best Subsets Selection - Forward")
```



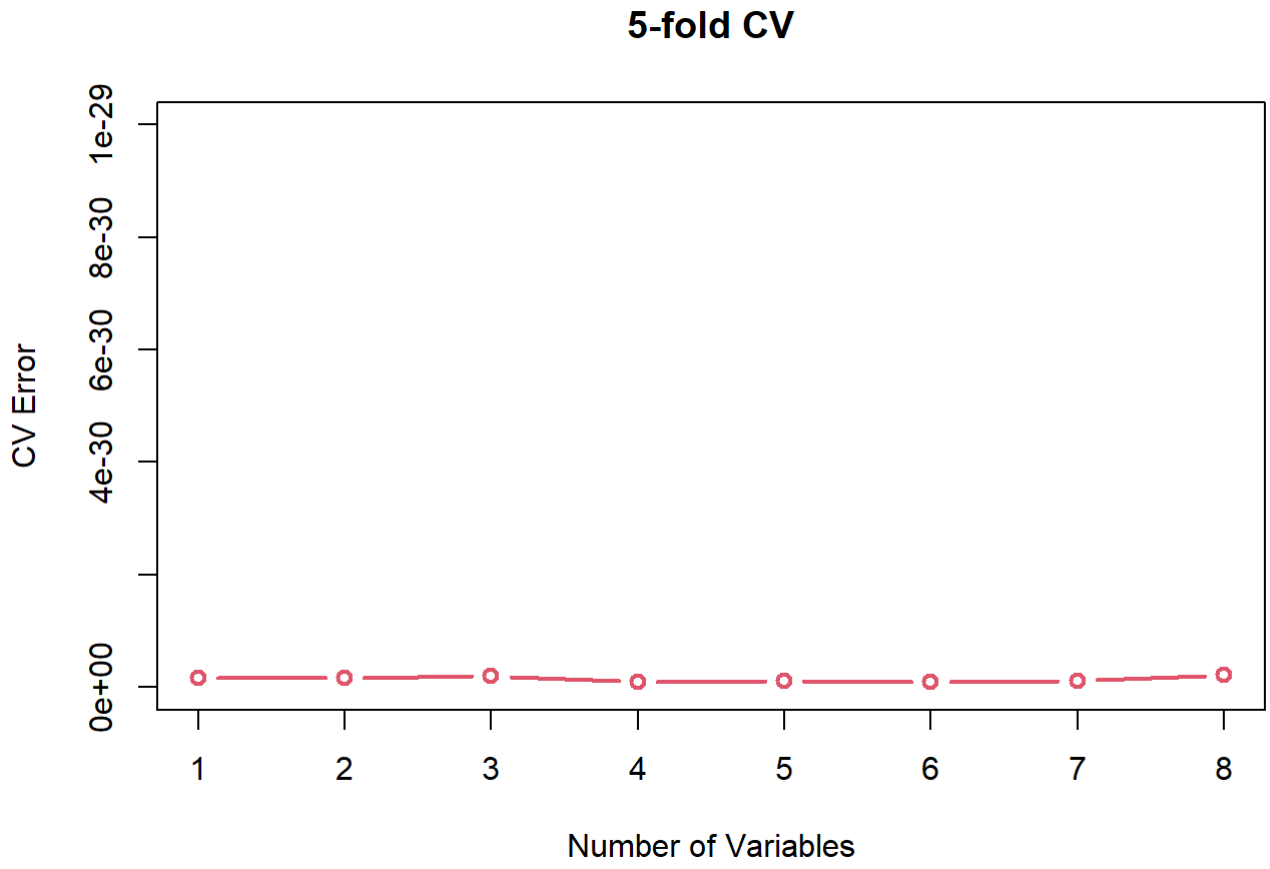

```
# 3 number of predictors is the best model using BIC, considering the model complexity.

#4
CV5.err2=rep(0,8)
set.seed(1)
for(p in 1:8){
  x=which(summary(regfit.fwd)$which[p,])
  x=as.numeric(x)
  x=x[-1]-1
  dfname=c("GPA",names(GPAdf)[x])
  newGPAdf=GPAdf[,dfname]
  glm.fit=glm(GPA~.,data=newGPAdf)
  cv.err=cv.glm(newGPAdf,glm.fit,K=5)
  CV5.err2[p]=cv.err$delta[1] #残差平方和
}
CV5.err2
```

```
## [1] 1.551156e-31 1.625450e-31 1.974404e-31 9.568090e-32 1.179689e-31
## [6] 9.590603e-32 1.175187e-31 2.089221e-31
```

```
plot(1:8,CV5.err2,type="b",lwd=2,col=2,xlab="Number of Variables",ylab="CV Error",main="5-fold CV")
```




```
# The forward model is the same as the exhaustive model.
# We can choose 4 number of predictors to be the best model using 5-fold CV, considering the mo
del complexity.

# 由于参数设置相同，且变量相互之间可能没有高度相关性，Forward和exhaustive的最优模型相同；
# 根据adjr2和bic分别可选4或3个变量进行建模；
# 而交叉验证的CV由于尺度过小，以及受种子随机性影响较大，在允许范围内可不纳入考量范围优先级中，
放大尺度如下所示：
plot(1:8,CV5.err2,type="b",lwd=2,col=2,xlab="Number of Variables",ylab="CV Error",main="5-fold
CV",ylim=c(0,1e-29))
```




```
#5
GPAdf2=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//FirstYe
arGPA.csv")
GPAdf2$GPA=ifelse(GPAdf2$GPA>=3,1,0)

library(MASS)
library(caret)
```

```
## 载入需要的程辑包：ggplot2
```

```
## 载入需要的程辑包：lattice
```

```
##
## 载入程辑包：'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
##
## 载入程辑包：'caret'
```

```
## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```
GPAdf2$GPA=as.factor(GPAdf2$GPA) # 分类问题
set.seed(1)

############ 循环实现forward子集选择，交叉验证由train函数实现 ############

selected_vars=c()
best_performance=0

for (feature in colnames(GPAdf2)[-1]){
  # 添加
  selected_vars=c(selected_vars,feature)

  # 训练LDA模型
  lda_model=lda(GPA~.,data=GPAdf2[,c("GPA",selected_vars)])

  # 使用交叉验证评估模型性能
  cv_results=train(GPA~.,data=GPAdf2[,c("GPA",selected_vars)],method="glm",trControl=trainContr
ol(method ="cv",number=5))

  # 如果模型性能得分更好，则更新最佳模型和变量子集
  if(cv_results$results$Accuracy>best_performance){
    best_performance=cv_results$results$Accuracy
    best_lda_model=lda_model
    best_selected_vars=selected_vars
  }else{
    # 如果模型性能没有提升，则从变量子集中移除最后添加的特征
    selected_vars=selected_vars[-length(selected_vars)]
  }
}

best_selected_vars # "HSGPA"、"SATV"、"SATM"、"HU"
```

```
## [1] "HSGPA" "SATV"  "SATM"  "HU"
```

```
best_performance   # 0.7040334
```

```
## [1] 0.7040334
```

```
# 选取"HSGPA"、"SATV"、"SATM"、"HU"这四个变量构建GLM模型最优

#6
library(MASS)
library(caret)

set.seed(1)

############ 循环实现forward子集选择，交叉验证由train函数实现 ############

selected_vars=c()
best_performance=0

for (feature in colnames(GPAdf2)[-1]){
  # 添加
  selected_vars=c(selected_vars,feature)

  # 训练LDA模型
  lda_model=lda(GPA~.,data=GPAdf2[,c("GPA",selected_vars)])

  # 使用交叉验证评估模型性能
  cv_results=train(GPA~.,data=GPAdf2[,c("GPA",selected_vars)],method="lda",trControl=trainContr
ol(method ="cv",number=5))

  # 如果模型性能得分更好，则更新最佳模型和变量子集
  if(cv_results$results$Accuracy>best_performance){
    best_performance=cv_results$results$Accuracy
    best_lda_model=lda_model
    best_selected_vars=selected_vars
  }else{
    # 如果模型性能没有提升，则从变量子集中移除最后添加的特征
    selected_vars=selected_vars[-length(selected_vars)]
  }
}

best_selected_vars # "HSGPA"、"SATV"、"SATM"、"HU"
```

```
## [1] "HSGPA" "SATV"  "SATM"  "HU"
```

```
best_performance   # 0.7039323
```

```
## [1] 0.7039323
```

```
# 选取"HSGPA"、"SATV"、"SATM"、"HU"这四个变量构建LDA模型最优
```

4.

```
#(4)
train=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//diabetes_train.csv")
test=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//diabetes_test.csv")
trainx=train[,-1]
trainy=train[,1]
testx=test[,-1]
testy=test[,1]

#1
library(glmnet)
```

```
## 载入需要的程辑包：Matrix
```

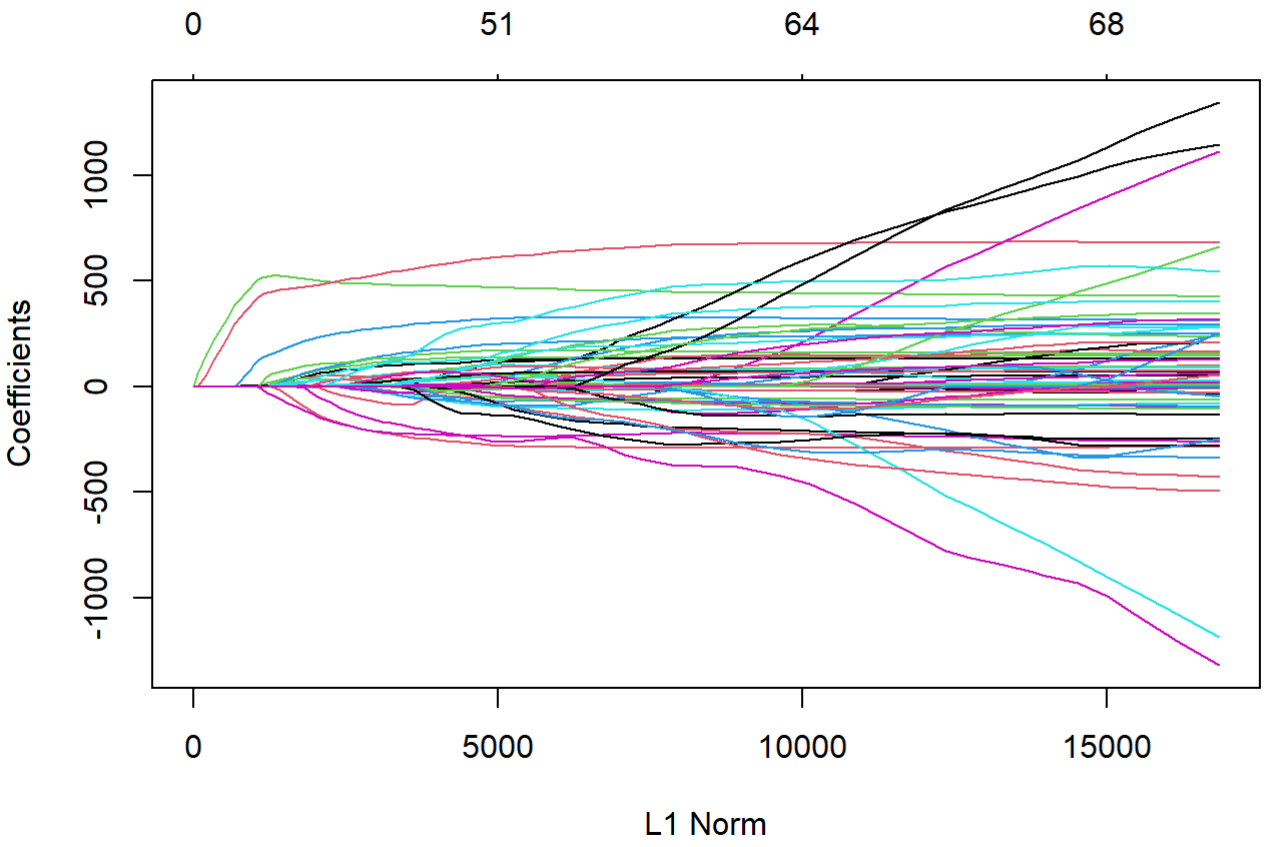
```
## Loaded glmnet 4.1-3
```

```
grid=10^seq(4,-2,length=100)
lasso.mod=glmnet(trainx,trainy,alpha=1,lambda=grid)
names(lasso.mod)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"    "dev.ratio"
##  [7] "nulldev"   "npasses"   "jerr"      "offset"    "call"      "nobs"
```

```
# coef(lasso.mod)
plot(lasso.mod)
```

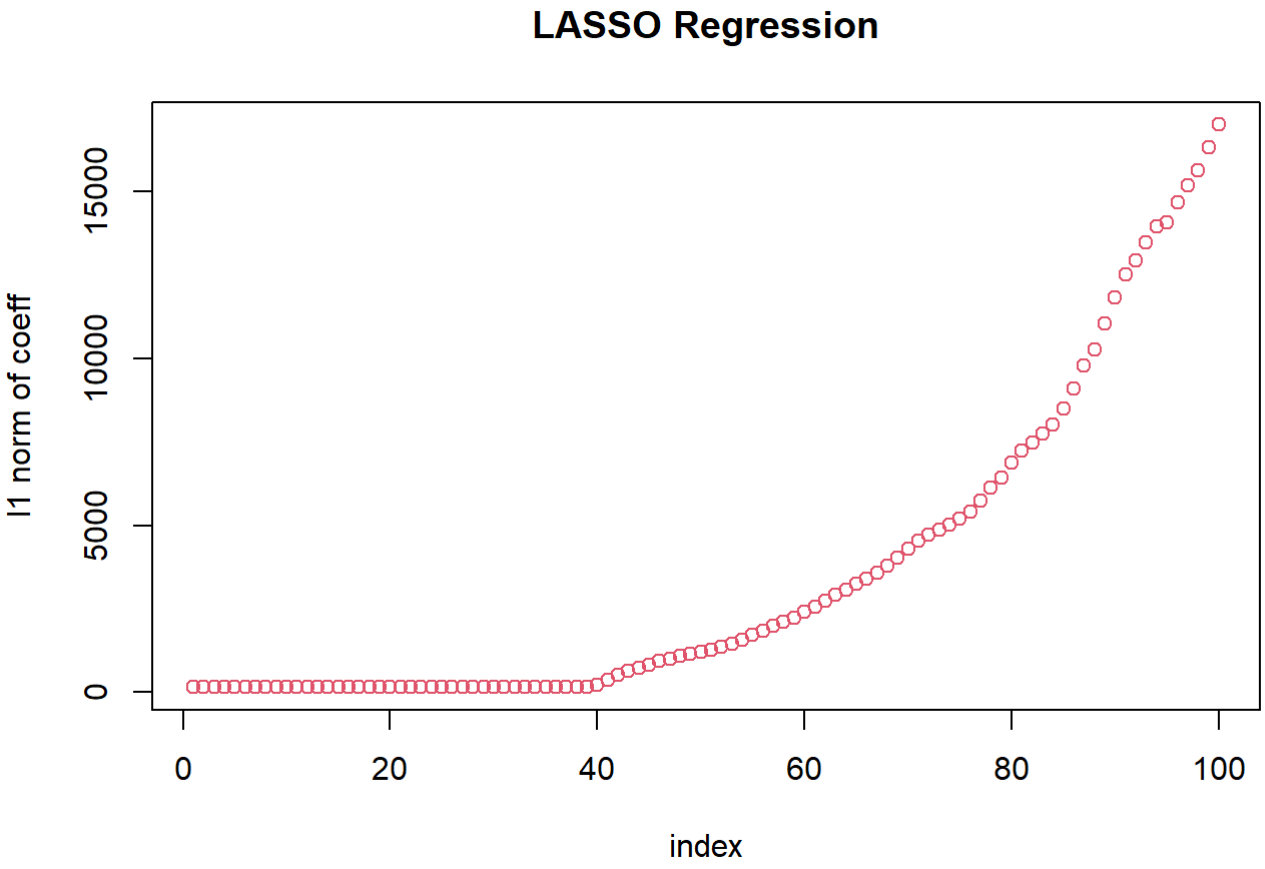
```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```



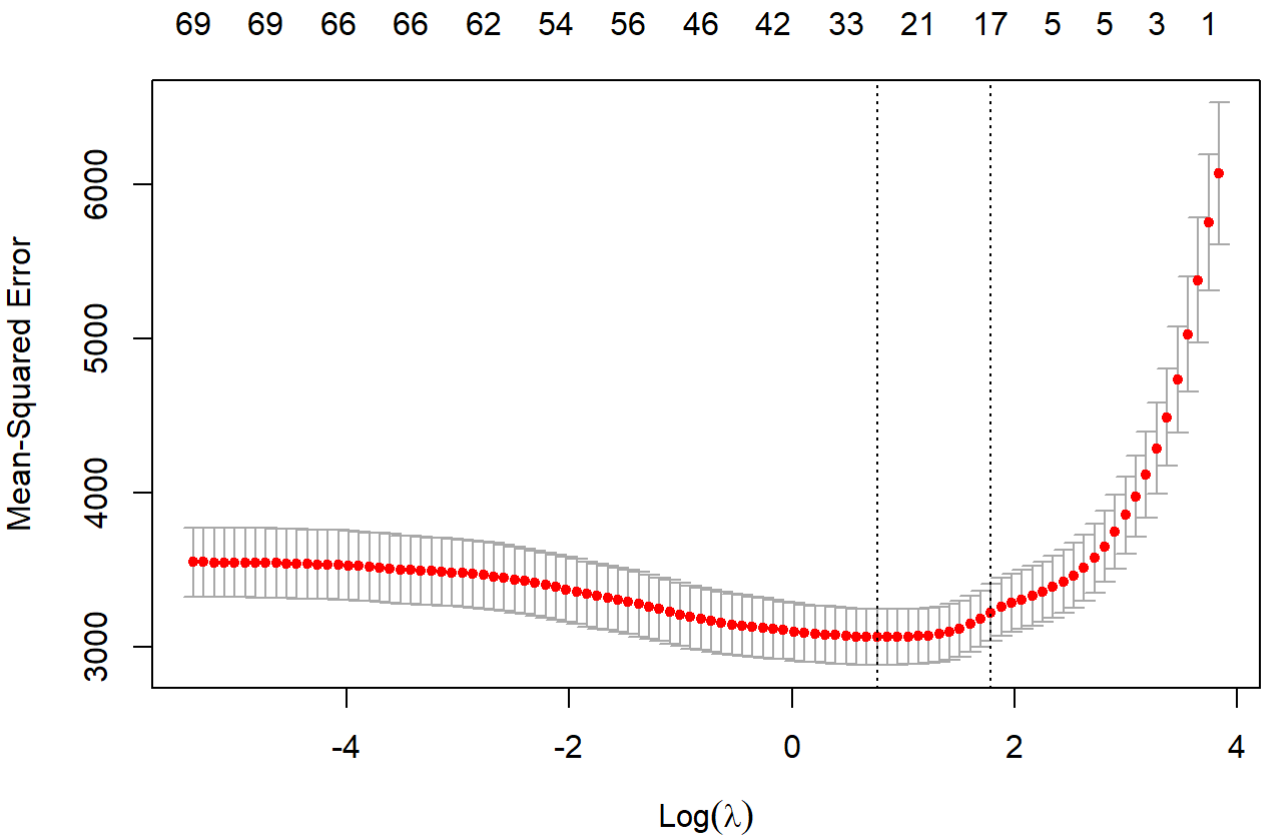

```
coefficients=coef(lasso.mod)
l1_norm=apply(abs(coefficients),2,sum)
l1_norm
```

```
##         s0         s1         s2         s3         s4         s5         s6
##   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889
##         s7         s8         s9        s10        s11        s12        s13
##   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889
##        s14        s15        s16        s17        s18        s19        s20
##   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889
##        s21        s22        s23        s24        s25        s26        s27
##   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889
##        s28        s29        s30        s31        s32        s33        s34
##   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889   152.8889
##        s35        s36        s37        s38        s39        s40        s41
##   152.8889   152.8889   152.8889   152.8889   213.9801   363.4119   506.2635
##        s42        s43        s44        s45        s46        s47        s48
##   630.5073   738.5683   832.5542   928.0557  1011.7738  1084.5931  1147.9277
##        s49        s50        s51        s52        s53        s54        s55
##  1203.0340  1276.1132  1352.2698  1434.6654  1571.1851  1718.4214  1846.3992
##        s56        s57        s58        s59        s60        s61        s62
##  1972.7853  2104.7862  2231.6613  2392.3353  2557.1361  2725.1784  2905.4508
##        s63        s64        s65        s66        s67        s68        s69
##  3081.2927  3241.2597  3394.3265  3568.8515  3772.7930  4037.1247  4294.6731
##        s70        s71        s72        s73        s74        s75        s76
##  4524.8964  4707.2280  4866.5671  5020.2854  5207.2163  5409.7108  5745.2624
##        s77        s78        s79        s80        s81        s82        s83
##  6125.0881  6416.1819  6883.6656  7228.2591  7481.3180  7729.7101  8006.5675
##        s84        s85        s86        s87        s88        s89        s90
##  8487.5556  9081.6339  9786.8507 10270.3534 11027.6568 11815.9543 12512.8121
##        s91        s92        s93        s94        s95        s96        s97
## 12915.3045 13463.6631 13934.6165 14078.9686 14681.4525 15188.6670 15635.4721
##        s98        s99
## 16313.2013 16995.1262
```

```
# sqrt(sum(coef(ridge.mod)[-1,]^2)) ## l2 norm of the coefficients
plot(l1_norm,xlab="index",ylab="l1 norm of coeff",type='b',col=2,main="LASSO Regression")   ## plot l1 norm
```




```
#LASSO模型对于某些特征会将其系数收缩至0，从而实现特征选择的效果
#对于某些自变量,当l1范数较大时,系数仍然保持较大;而对于另一些自变量,当l1范数较小时,系数趋向于0

#2
set.seed(1)
#数据框需转化为矩阵
cv.out=cv.glmnet(as.matrix(trainx),trainy,alpha=1,nfolds=10) ## CV errors by fitting LASSO on t
rain dataset
plot(cv.out)    ## plot mean squared error w.r.t. values of lambda
```




```
bestlam=cv.out$lambda.min
bestlam  # 2.14688
```

```
## [1] 2.14688
```

```
out=glmnet(trainx,trainy,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:11,]   # use full trainning dataset

lasso.coef
```

```
## (Intercept)         age         sex         bmi         map          tc
##    150.7879      0.0000   -181.0402    490.2316    257.8220      0.0000
##         ldl         hdl         tch         ltg         glu
##      0.0000   -181.9006      0.0000    506.0437     94.9178
```

```
lasso.coef[lasso.coef!=0]
```

```
## (Intercept)         sex         bmi         map         hdl         ltg
##    150.7879   -181.0402    490.2316    257.8220   -181.9006    506.0437
##         glu
##     94.9178
```

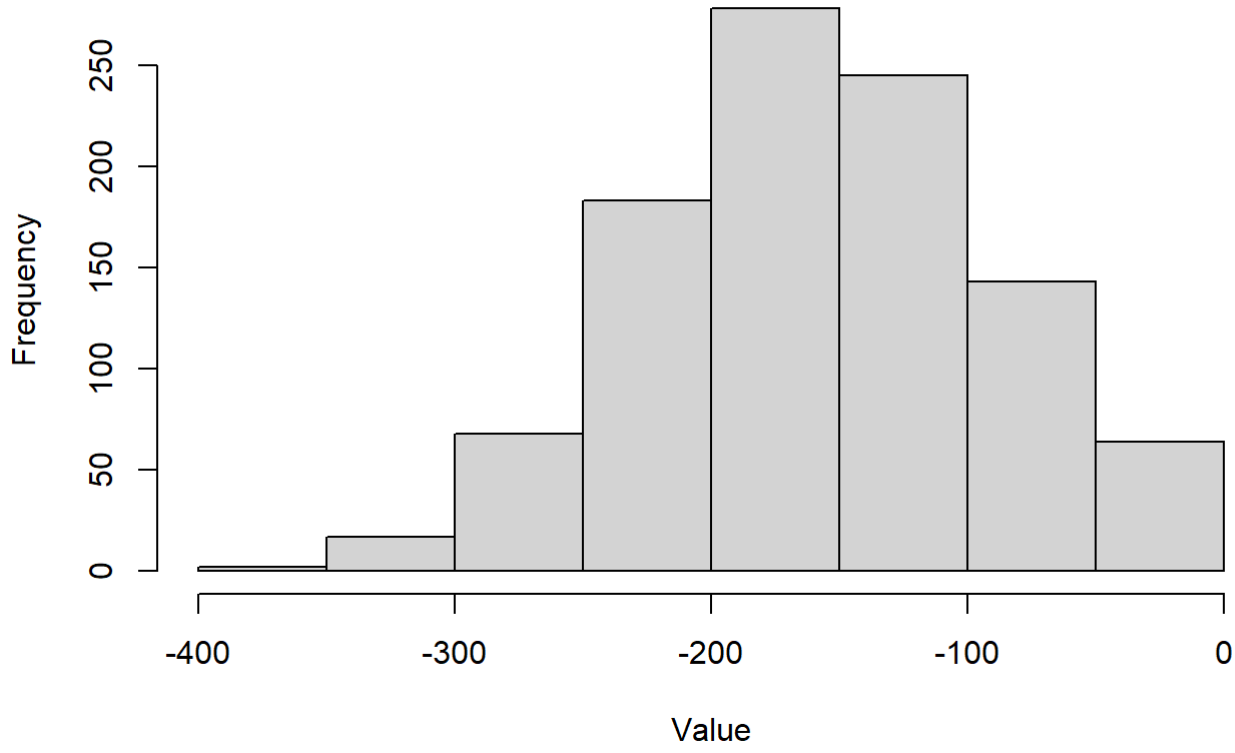
```
# 6 variables are included in the model.
# They are sex、bmi、map、hdl、ltg、glu.

#3
lasso.pred=predict(lasso.mod,s=bestlam,newx=as.matrix(testx))   ## 基于最优λ值预测
mean((lasso.pred-testy)^2)
```

```
## [1] 2970.938
```

```
#4
###### Bootstrap抽样 ######
# coefficients=coef(lasso.mod,s=bestlam)

# 定义函数
n=nrow(trainx)
bootstrap=function(df=train,index){
  df=df[index,]
  x=df[,-1]
  y=df[,1]
  fit=glmnet(x,y,alpha=1,lambda=grid)
  coefficients=coef(fit,s=bestlam)[1:11,]
  return(coefficients)
}

boot.res=boot(train,bootstrap,R=1000)
boot.res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = train, statistic = bootstrap, R = 1000)
##
##
## Bootstrap Statistics :
##       original       bias    std. error
## t1*   150.7879  -0.05432553    2.762085
## t2*     0.0000  10.06513667   29.536896
## t3*  -181.0402  22.65468685   67.382444
## t4*   490.2316   6.29078531   79.561828
## t5*   257.8220  -7.63453690   73.534046
## t6*     0.0000 -16.33615993   32.790278
## t7*     0.0000  -0.38257678    9.417730
## t8*  -181.9006  30.91589114   73.594207
## t9*     0.0000   7.68923908   26.687720
## t10*  506.0437   7.56958823   79.927490
## t11*   94.9178  -1.96596774   67.921704
```

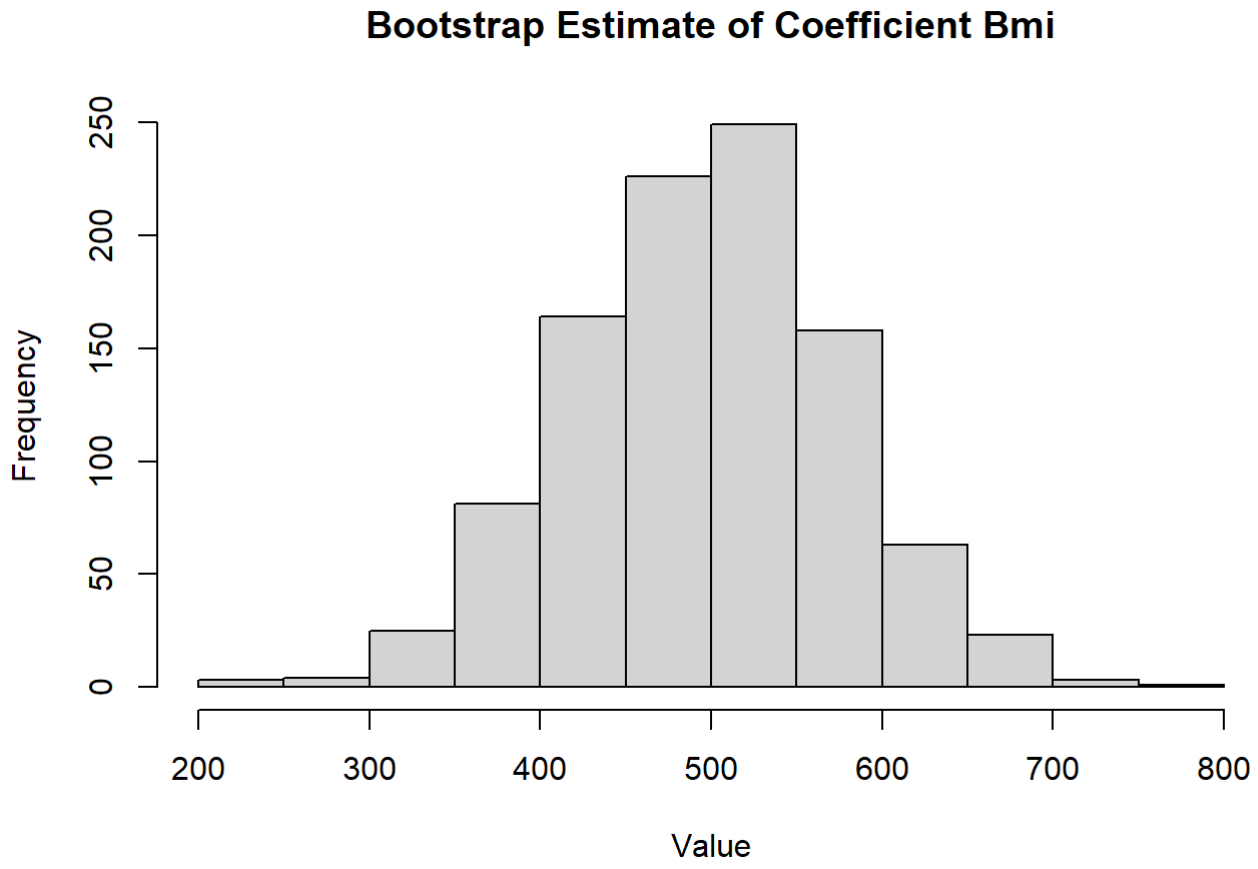
```
# boot.res$t[,3]:sex
hist(boot.res$t[,3],xlab="Value",main="Bootstrap Estimate of Coefficient Sex")
```




```
quantile(boot.res$t[,3], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##       2.5%       97.5%
## -292.40024  -26.10737
```

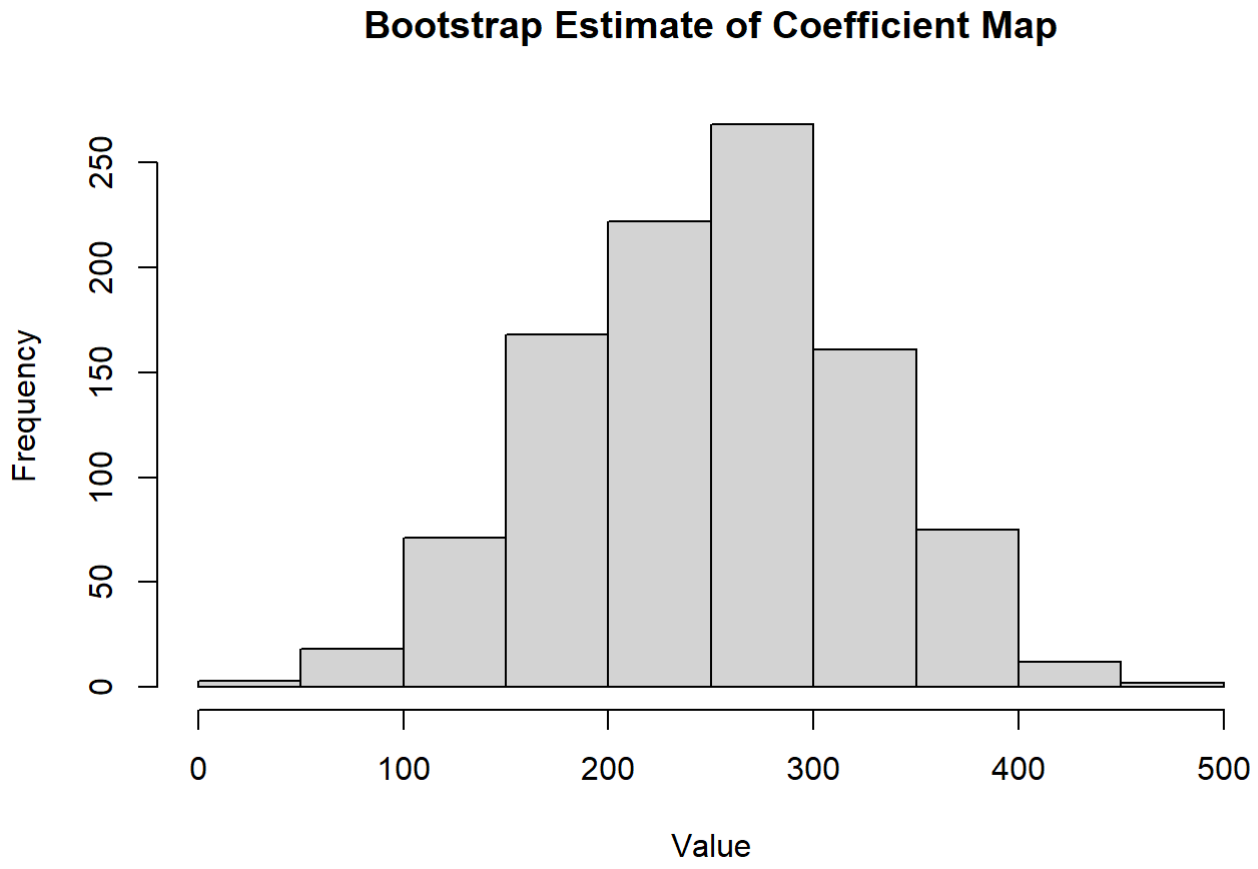
```
# boot.res$t[,4]:bmi
hist(boot.res$t[,4],xlab="Value",main="Bootstrap Estimate of Coefficient Bmi")
```




```
quantile(boot.res$t[,4], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##     2.5%    97.5%
## 341.0104 652.6520
```

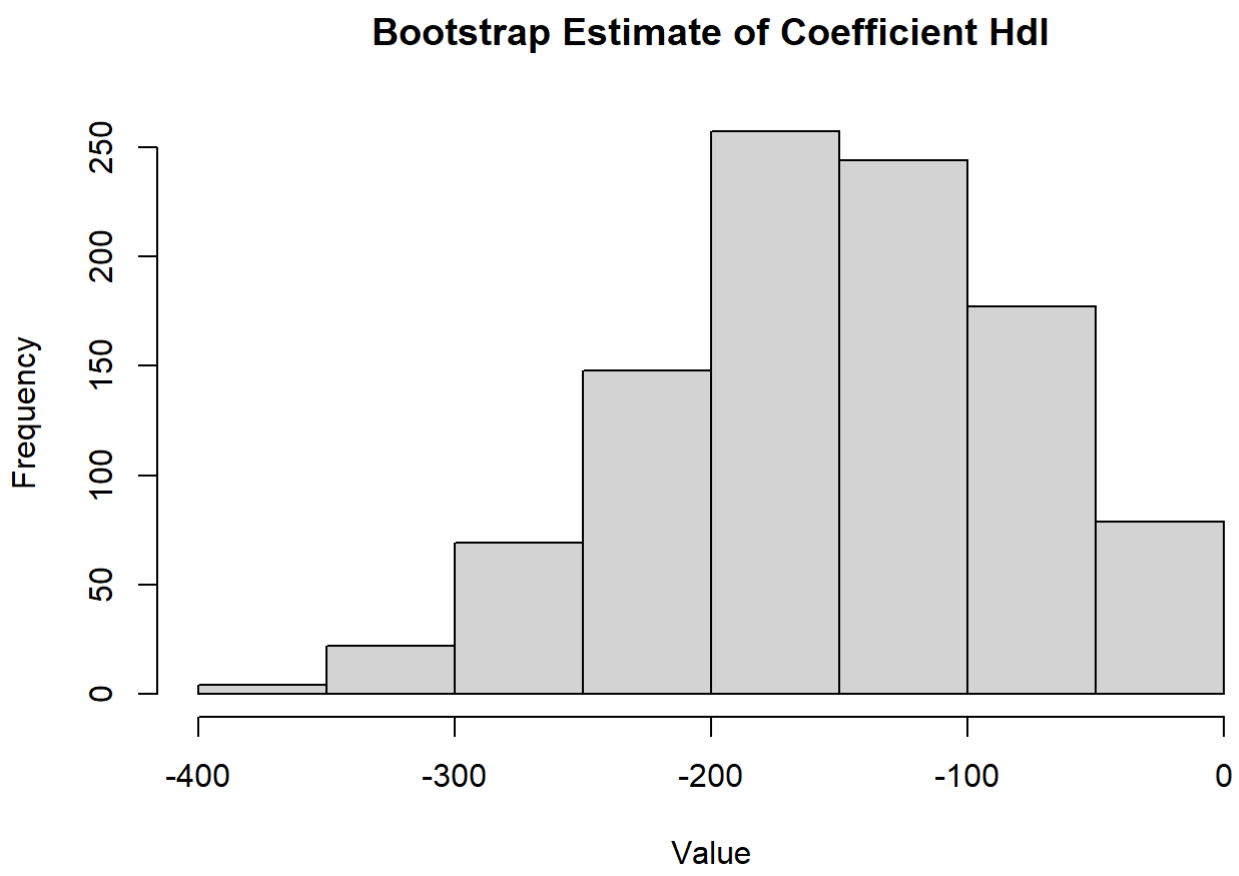
```
# boot.res$t[,5]:map
hist(boot.res$t[,5],xlab="Value",main="Bootstrap Estimate of Coefficient Map")
```




```
quantile(boot.res$t[,5], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##     2.5%    97.5%
## 106.9494 390.7264
```

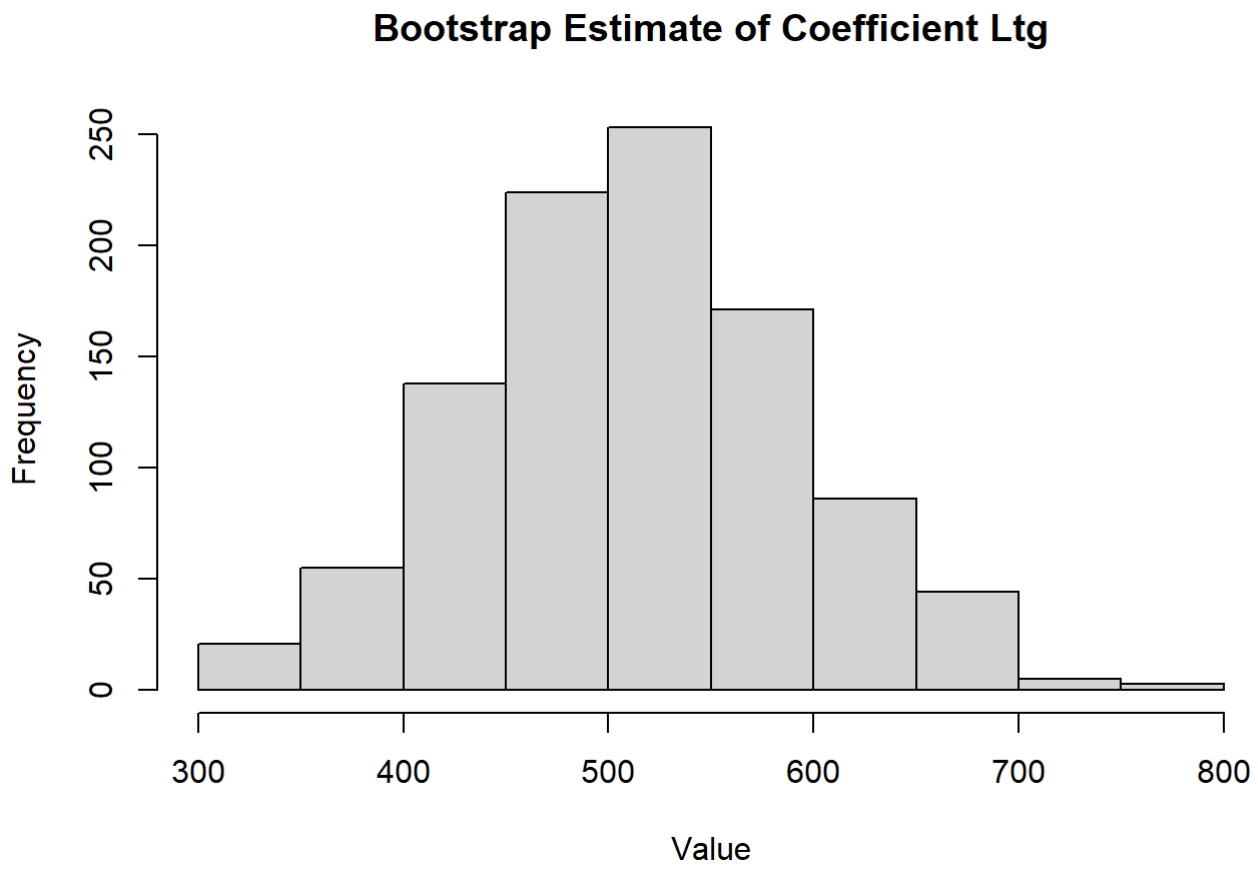
```
# boot.res$t[,8]:hdl
hist(boot.res$t[,8],xlab="Value",main="Bootstrap Estimate of Coefficient Hdl")
```




```
quantile(boot.res$t[,8], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##         2.5%        97.5%
## -301.084412   -3.557045
```

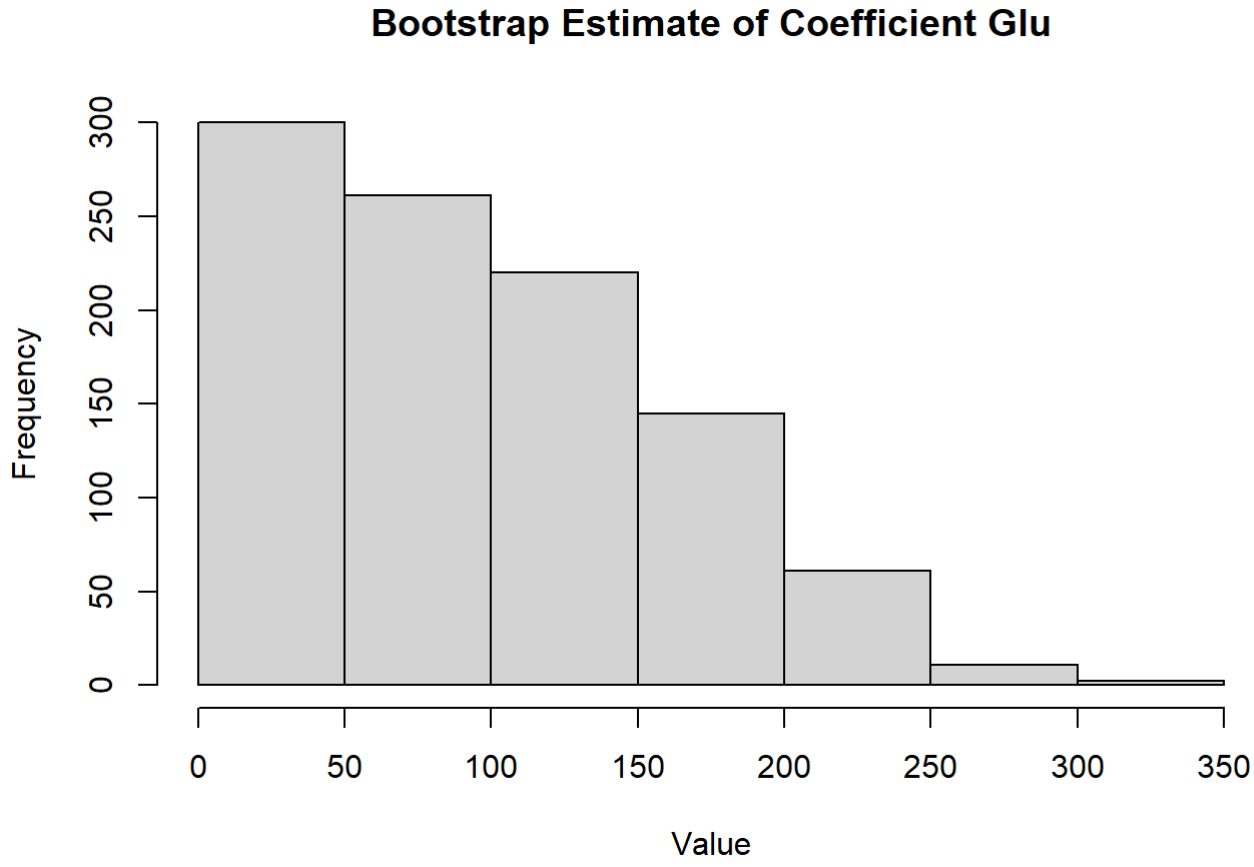
```
# boot.res$t[,10]:ltg
hist(boot.res$t[,10],xlab="Value",main="Bootstrap Estimate of Coefficient Ltg")
```




```
quantile(boot.res$t[,10], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##     2.5%    97.5%
## 358.8953 674.1647
```

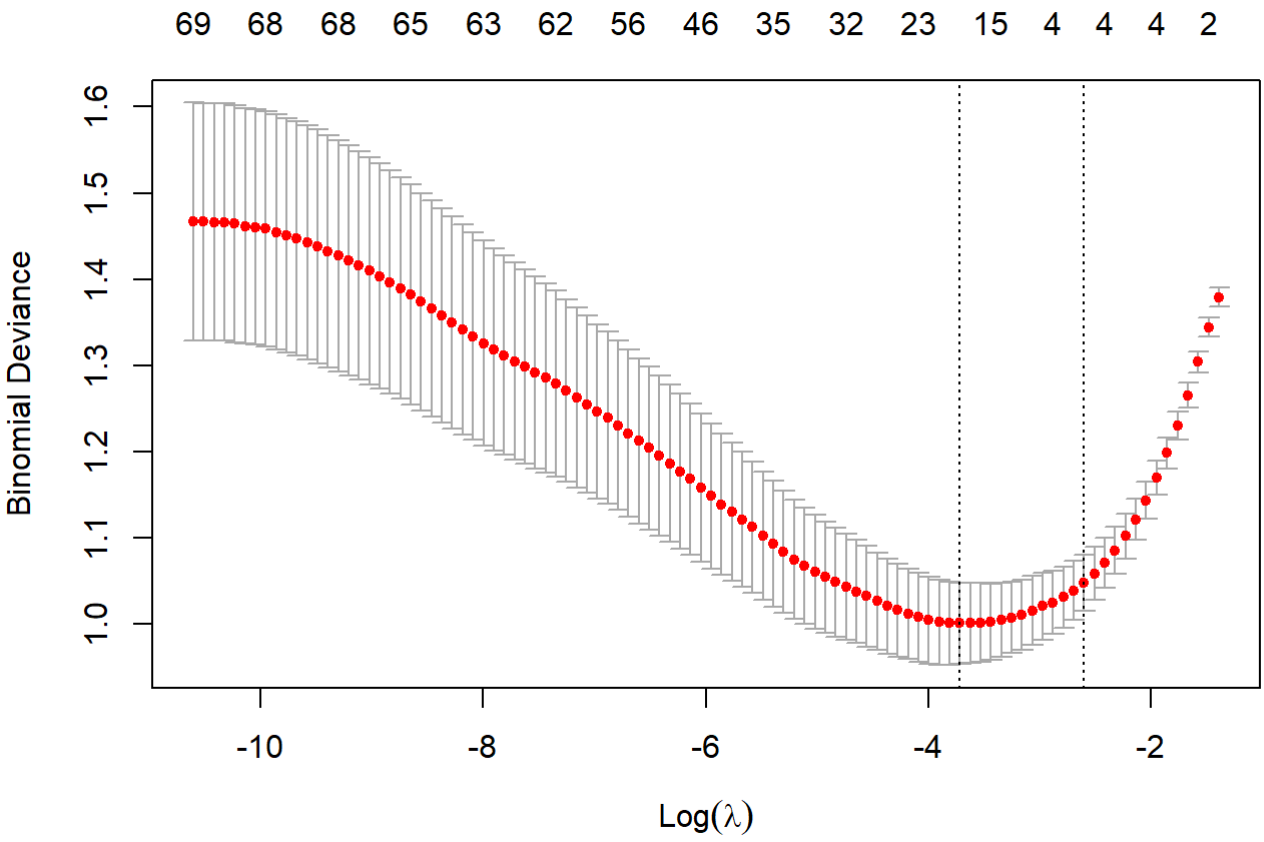
```
# boot.res$t[,11]:glu
hist(boot.res$t[,11],xlab="Value",main="Bootstrap Estimate of Coefficient Glu")
```




```
quantile(boot.res$t[,11], probs = c(0.025, 0.975), na.rm = TRUE)
```

```
##      2.5%     97.5%
##   0.0000 231.3093
```

```
#5
train=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//diabetes_train.csv")
test=read.csv("C://Users//张铭韬//Desktop//学业//港科大//MSDM5054机器学习//作业//hw2//diabetes_test.csv")
trainx=train[,-1]
trainy=train[,1]
testx=test[,-1]
testy=test[,1]

trainy=ifelse(trainy>150,0,1)
testy=ifelse(testy>150,0,1)

trainy=as.factor(trainy)
testy=as.factor(testy)

set.seed(1)

lasso.mod2=glmnet(trainx,trainy,alpha=1,lambda=grid,family="binomial")

cv.out2=cv.glmnet(as.matrix(trainx),trainy,alpha=1,nfolds=10,family="binomial") ## CV errors by fitting LASSO on train dataset
plot(cv.out2)    ## plot mean squared error w.r.t. values of lambda
```




```
bestlam2=cv.out2$lambda.min
bestlam2  # 0.02429548
```

```
## [1] 0.02429548
```

```
probs=predict(lasso.mod2,s=bestlam2,newx=as.matrix(testx),type="response")

# predictions=ifelse(probs>0.5, "1", "0")
# table(predictions,testy)

lasso.coef=predict(lasso.mod2,type="coefficients",s=bestlam2)[1:11,]

lasso.coef
```

```
## (Intercept)         age         sex         bmi         map          tc
##   0.2781011   0.0000000   3.6419170 -10.9672027  -9.5181434   0.0000000
##         ldl         hdl         tch         ltg         glu
##   0.0000000   2.1460772   0.0000000 -12.6299717  -1.2377505
```

```
lasso.coef[lasso.coef!=0]
```

```
## (Intercept)         sex         bmi         map         hdl         ltg
##   0.2781011   3.6419170 -10.9672027  -9.5181434   2.1460772 -12.6299717
##         glu
##  -1.2377505
```

```
# 6 variables are included in the model.
# They are sex、bmi、map、hdl、ltg、glu, the same as that in the above model.
```