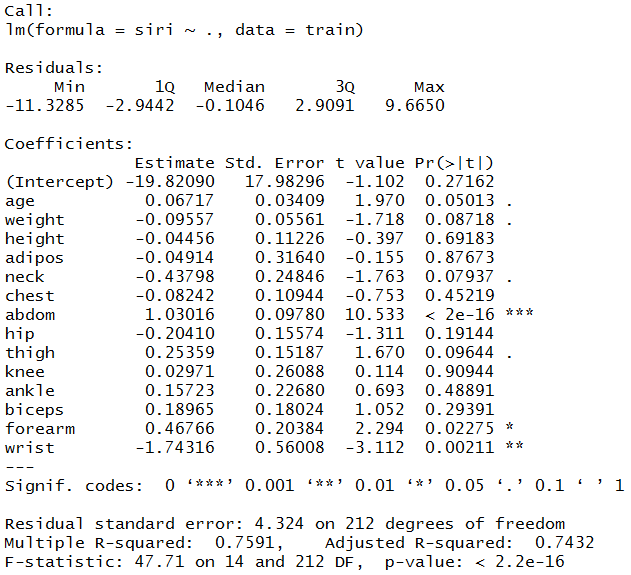
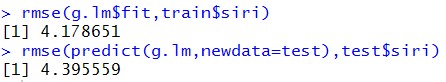
**STATS 500 - Homework 8**

Take the fat data, and use the percentage of body fat as the response and the other variables 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:

1. **Linear regression with all predictors**

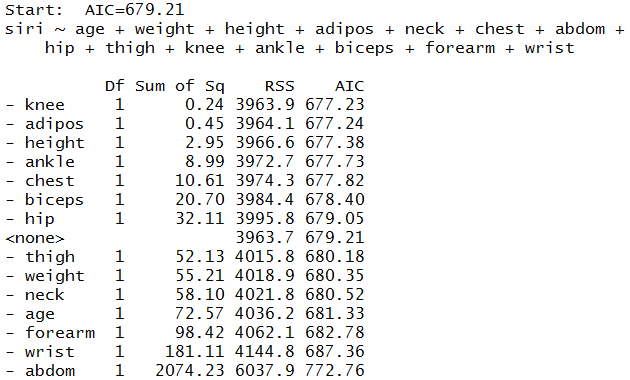


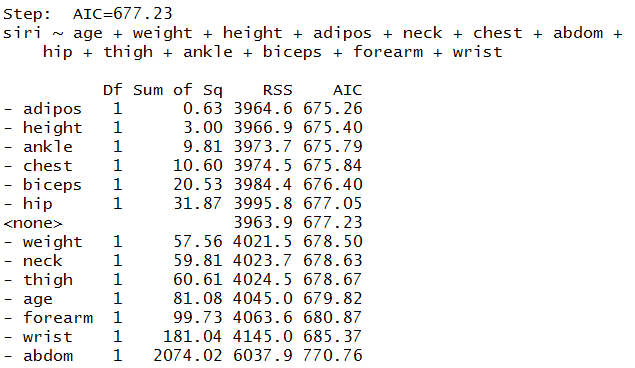
Calculate the RMSE(Root Mean Squared Error)

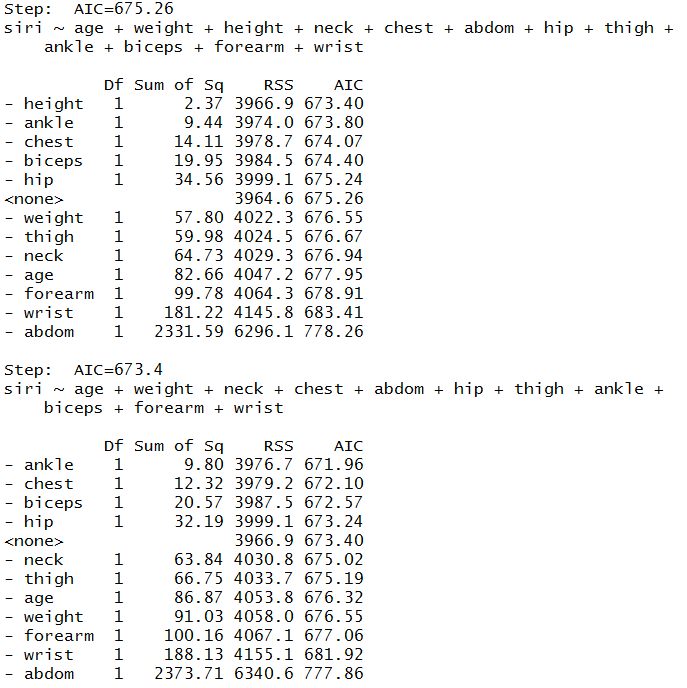


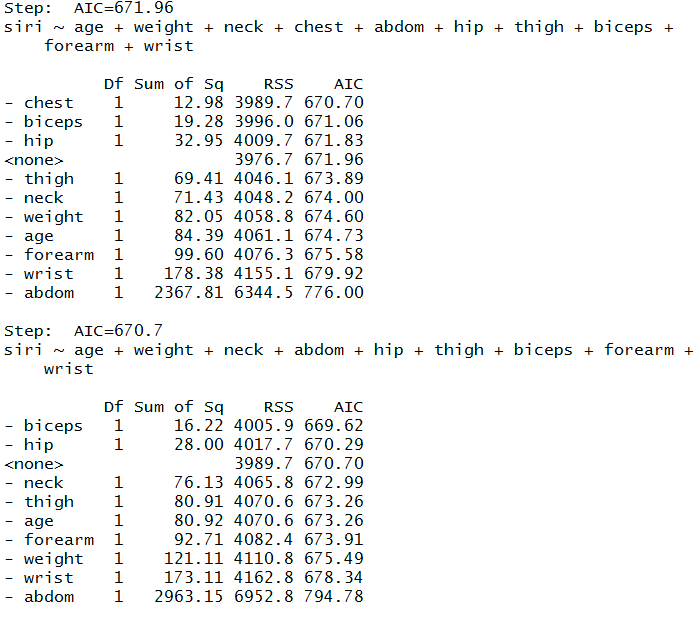
The performance of the linear regression with all predictors is satisfying with a relative large adjusted R2.

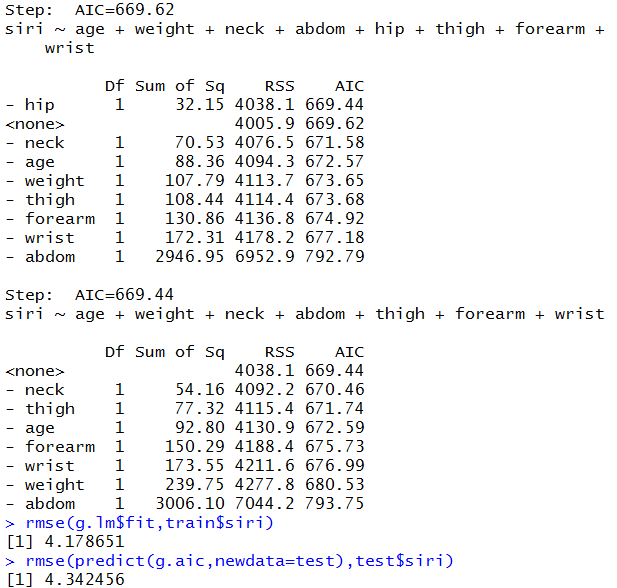
1. **Linear regression with variables selected using AIC**





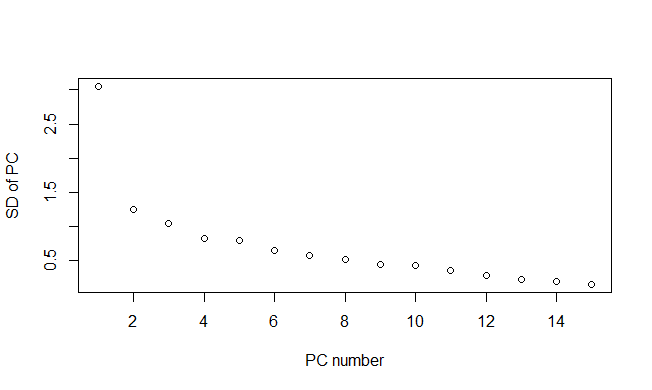
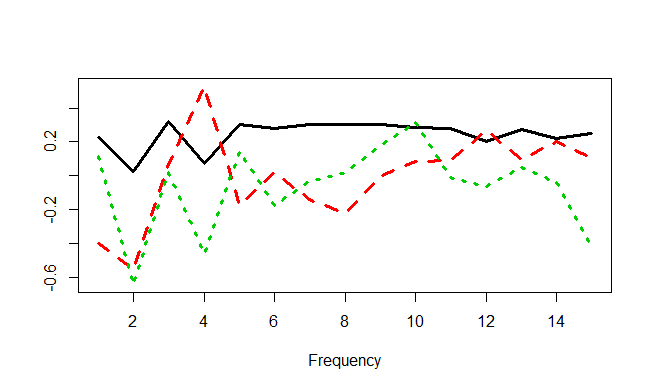


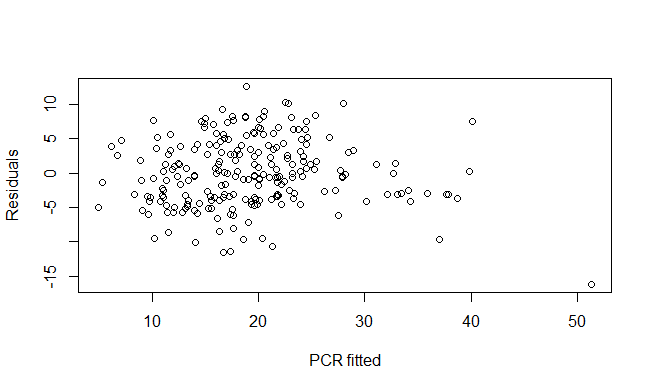
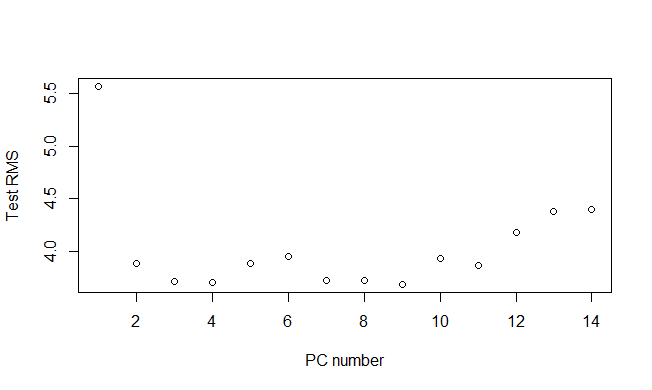




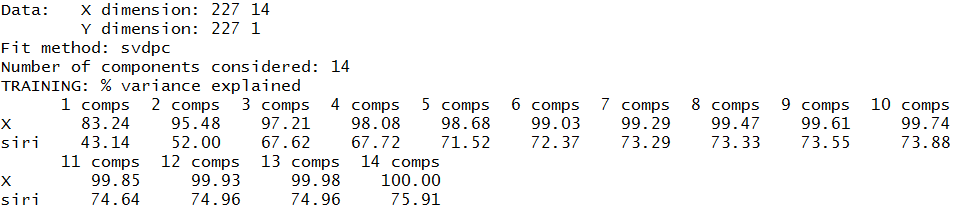
The performance of AIC model is slightly better than that of linear regression full model, with a larger adjusted R2 and a lower RMSE.

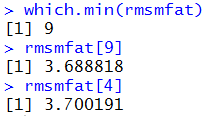
1. **Principal component regression**

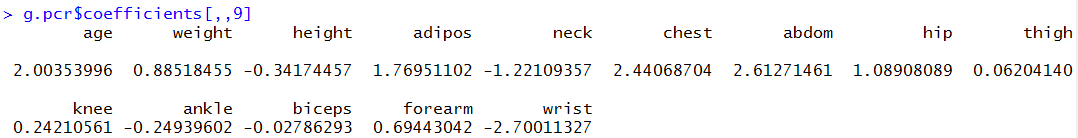




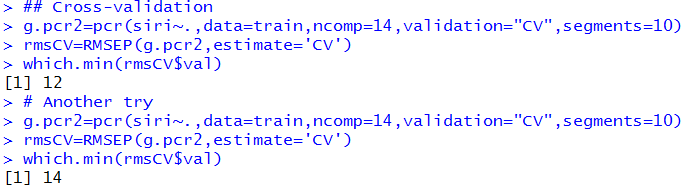
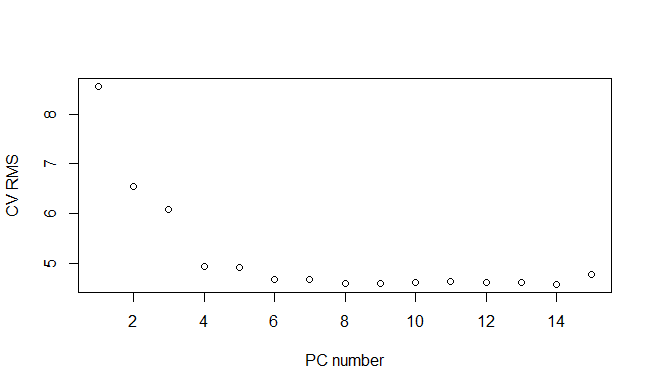
The residuals are randomly distributed.



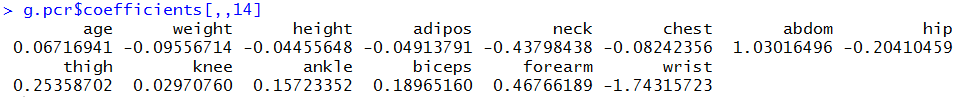




Cross-validation

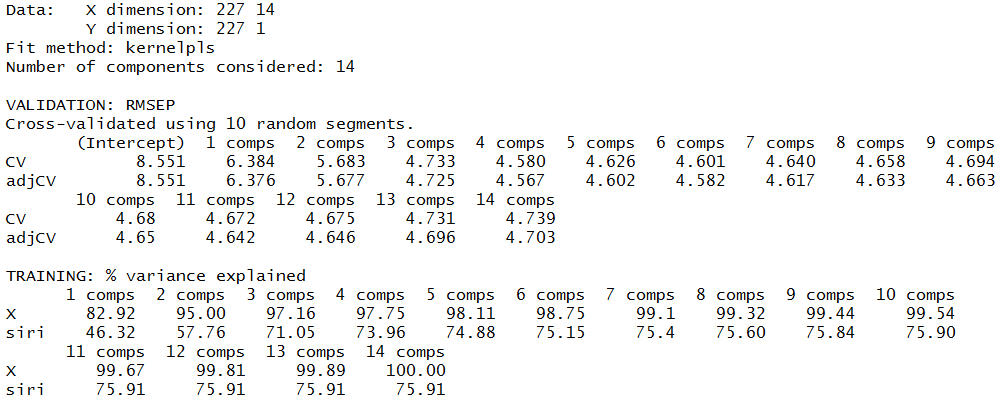
But if set.seed(123) before that the result is stable 14. That means that with 13 principle components we will get the best cross-validation.

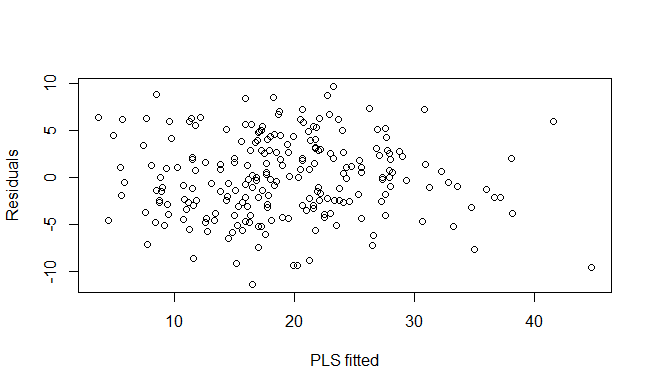
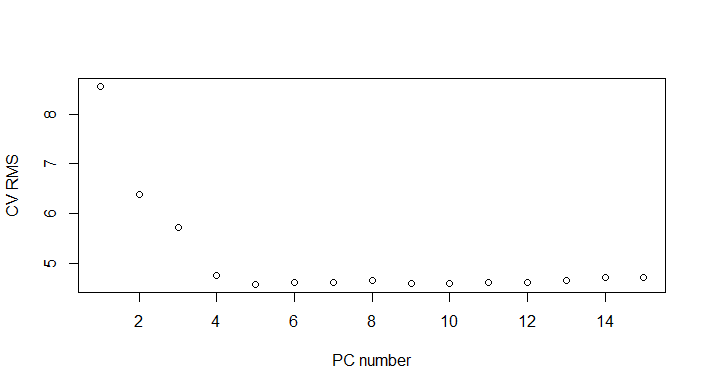


CV tends to overestimate the real test RMSE.

The performance of Principle Component regression is better than the former two with a smaller root mean squared error if we choose 9 principle components as suggested.

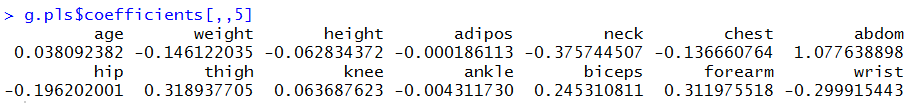
1. **Partial least squares**

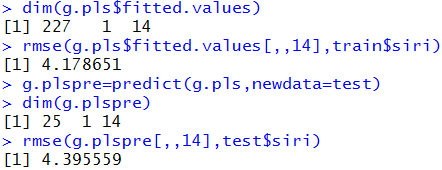




The distribution of residuals are almost the same with that of the PCR method. Random.

C:\Users\Heathtasia\AppData\Roaming\Tencent\Users\327471682\QQ\WinTemp\RichOle\OK9K%}M}OV$@)OMSOF_TXTN.png

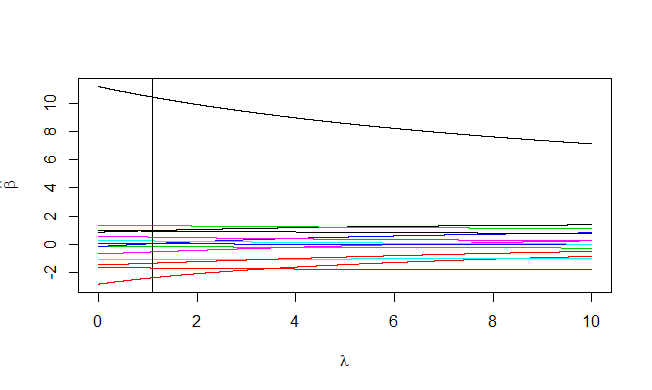


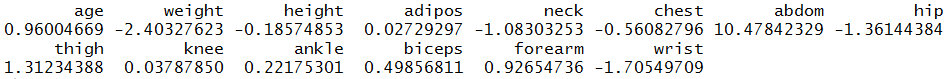


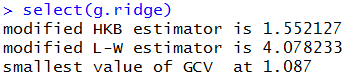
The partial least squares model gives a similar performance to the linear regression full model.

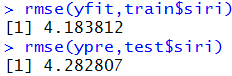
The PLS and PCR methods give almost the same result.

1. **Ridge regression**









The ridge regression here gives a good and stable performance.

In summary, all of the five methods give a fairly good performance. PCR with 9 components gives the smallest RMSE and smallest difference between train and test set.

**R-code used:**

library(faraway)

data(fat)

index <- seq(10, 250, by=10)

## Extract data and remove ‘‘brozek’’, ‘‘density’’ and ‘‘free’’

train <- fat[-index, -c(1, 3, 8)]

test <- fat[index, -c(1, 3, 8)]

## Linear model

g.lm=lm(siri~.,data=train)

summary(g.lm)

## Root mean squared error

rmse=function(x,y){sqrt(mean((x-y)^2))}

rmse(g.lm$fit,train$siri)

## Prediction

rmse(predict(g.lm,newdata=test),test$siri)

## AIC

g.aic=step(g.lm)

rmse(g.lm$fit,train$siri)

rmse(predict(g.aic,newdata=test),test$siri)

## Principal components regression

library(stats)

set.seed(123)

g.pca=prcomp(train[,1:15],scale=TRUE)

## Square root of the eigenvalues

round(g.pca$sdev,3)

matplot(1:15,g.pca$rot[,1:3],type="l",xlab="Frequency",ylab="",lwd=3)

## Make a scree plot (to choose number of PCs k)

plot(1:15,g.pca$sdev[1:15],tpe="l",xlab="PC number",ylab="SD of PC")

## Fit all PCRs at once and calculate test RMSE for each k

## Cross-validation

library(pls)

set.seed(123)

g.pcr=pcr(siri~.,data=train,ncomp=14,scale=TRUE)

rmsmfat=NULL

for(k in 1:14){

pv=predict(g.pcr,newdata=test,ncomp=k)

rmsmfat[k]=rmse(pv,test$siri)

}

plot(1:14,rmsmfat,xlab="PC number",ylab="Test RMS")

## Scree plot suggestion

which.min(rmsmfat)

rmsmfat[9]

rmsmfat[4]

## Cross-validation

set.seed(123)

g.pcr2=pcr(siri~.,data=train,ncomp=14,validation="CV",segments=10)

rmsCV=RMSEP(g.pcr2,estimate='CV')

which.min(rmsCV$val)

# Another try

set.seed(123)

g.pcr2=pcr(siri~.,data=train,ncomp=14,validation="CV",segments=10)

rmsCV=RMSEP(g.pcr2,estimate='CV')

which.min(rmsCV$val)

## Plot the RMSE; k=0 is the model with intercept only

plot(rmsCV$val,xlab="PC number",ylab="CV RMS")

## Get the test error

yfit= predict(g.pcr2,newdata=test,ncomp=14)

rmse(test$siri,yfit)

plot(g.pcr$fitted.values[,,9],g.pcr$residuals[,,9],xlab="PCR fitted",ylab="Residuals")

g.pcr$coefficients[,,9]

## Partial least squares

set.seed(123)

g.pls=plsr(siri~.,data=train,ncomp=14,validation="CV")

## Plot RMSE estimated by CV

pls\_rmsCV=RMSEP(g.pls,estimate='CV')

plot(pls\_rmsCV$val,xlab="PC number",ylab="CV RMS")

which.min(pls\_rmsCV$val)

##RMSE on the training data

dim(g.pls$fitted.values)

rmse(g.pls$fitted.values[,,14],train$siri)

##RMSE on the test data

g.plspre=predict(g.pls,newdata=test)

dim(g.plspre)

rmse(g.plspre[,,14],test$siri)

plot(g.pls$fitted.values[,,14],g.pls$residuals[,,14],xlab="PLS fitted",ylab="Residuals")

g.pls$coefficients[,,5]

library(MASS)

g.ridge=lm.ridge(siri~.,lambda=seq(0,10,1e-3),data=train)

matplot(g.ridge$lambda,t(g.ridge$coef),type="l",lty=1,xlab=expression(lambda),ylab=expression(hat(beta)))

## Select an appropriate lambda

select(g.ridge)

abline(v=1.087)

yfit=g.ridge$ym+scale(train[,-1],center=g.ridge$xm,scale=g.ridge$scales) %\*% g.ridge$coef[,which.min(g.ridge$GCV)]

## RMSE on training data

rmse(yfit,train$siri)

ypre=g.ridge$ym+scale(test[,-1],center=g.ridge$xm,scale=g.ridge$scales)%\*%g.ridge$coef[,which.min(g.ridge$GCV)]

rmse(ypre,test$siri)

g.ridge$coef[,which.min(g.ridge$GCV)]