**Alex Riles**

**DSCI 425 – Supervised Learning**

**Assignment 5 – PCR and PLS ( points)**

**Problem 1 – Near infrared (NIR) Spectra & Gasoline octane**

In this regression problem your goal is to model gasoline octane as a function of NIR spectra values. The NIR spectra were measured using diffuse reflectance as log(1/R) from 900 nm to 1700 nm in 2 nm intervals, giving 401 wavelengths, i.e. p = 401. We will use 50 of the original 60 observations to train the model and compare the predictive performance of different methods when predicting back the 10 test cases. The data frame is gasoline and comes with the pls package. (NIR on Wiki: <http://en.wikipedia.org/wiki/Near-infrared_spectroscopy> )

Some preliminary plots to get a sense of the nature of these data:

> library(pls)  
> data(gasoline)

Attaching package: ‘pls’

1. Construct some plots of these data using the following code as your guide:

gasoline.x = gasoline$NIR

dim(gasoline.x)

[1] 60 401

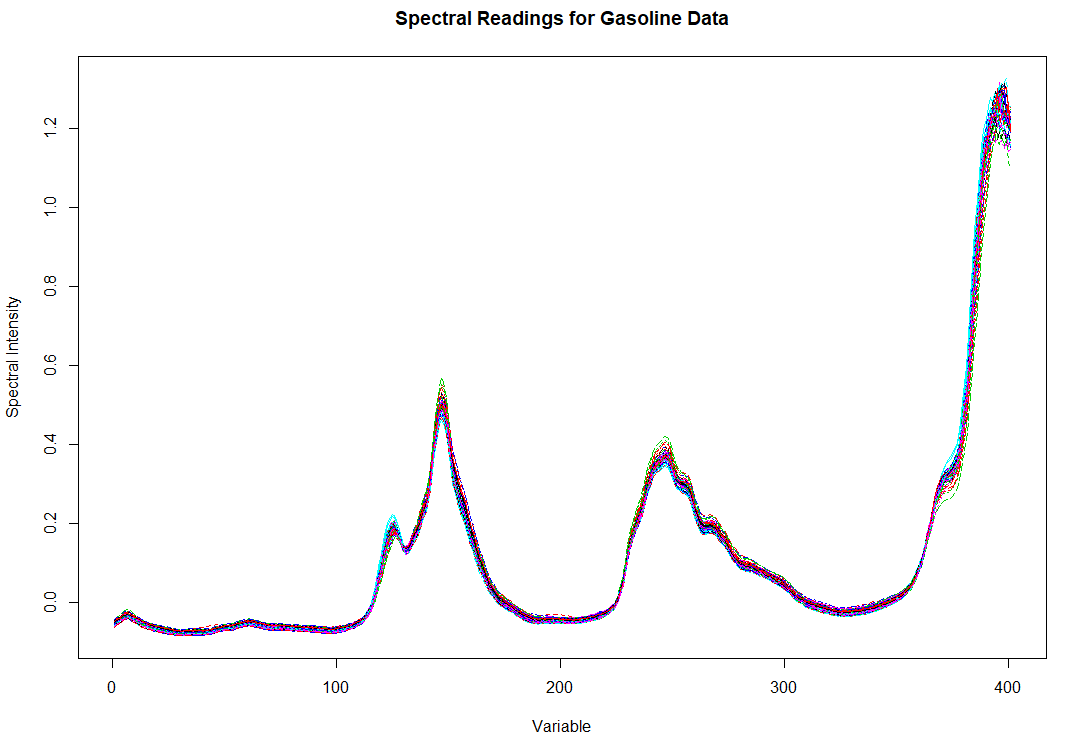
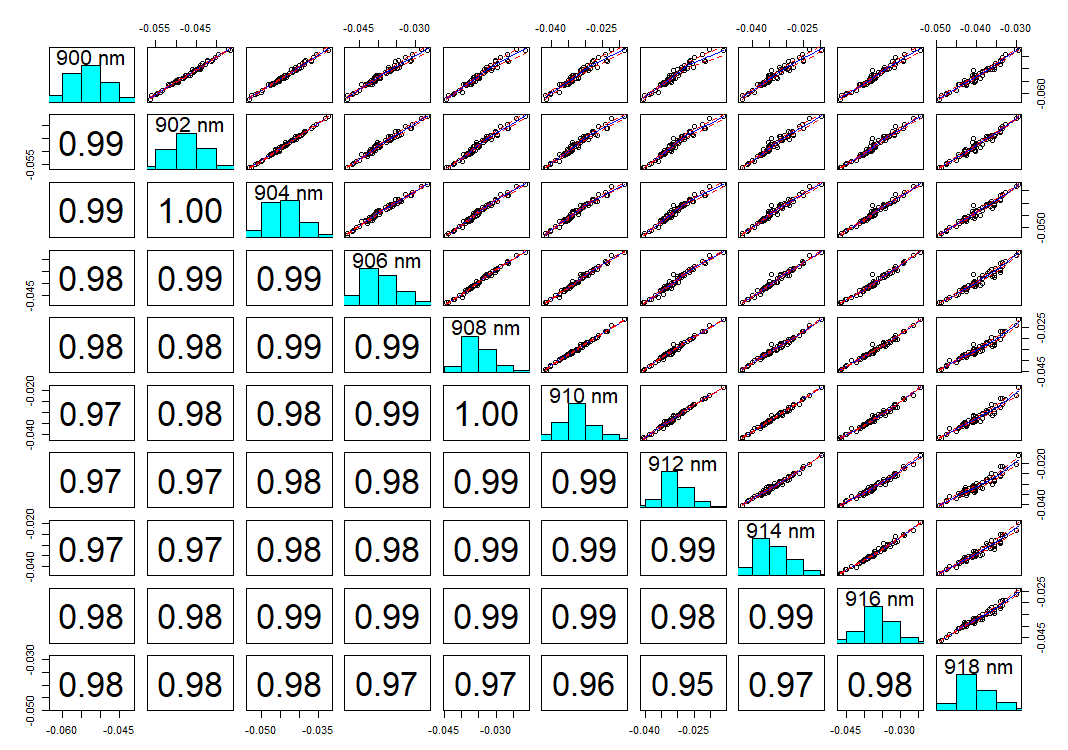
matplot(t(gasoline.x),type="l",xlab="Variable",ylab="Spectral Intensity")

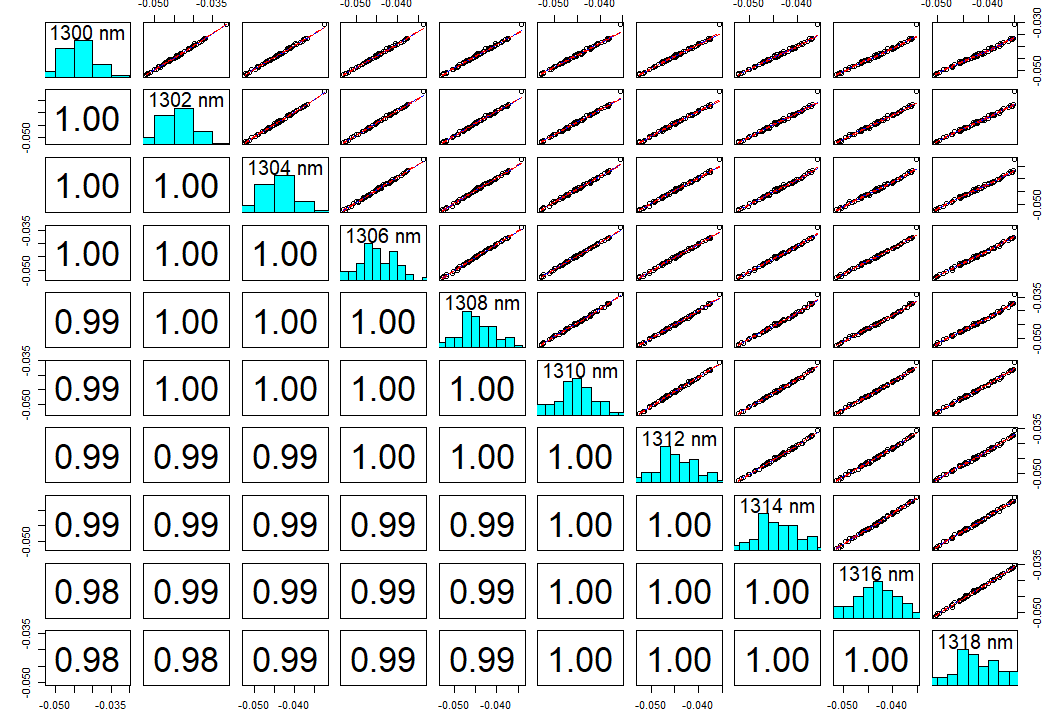
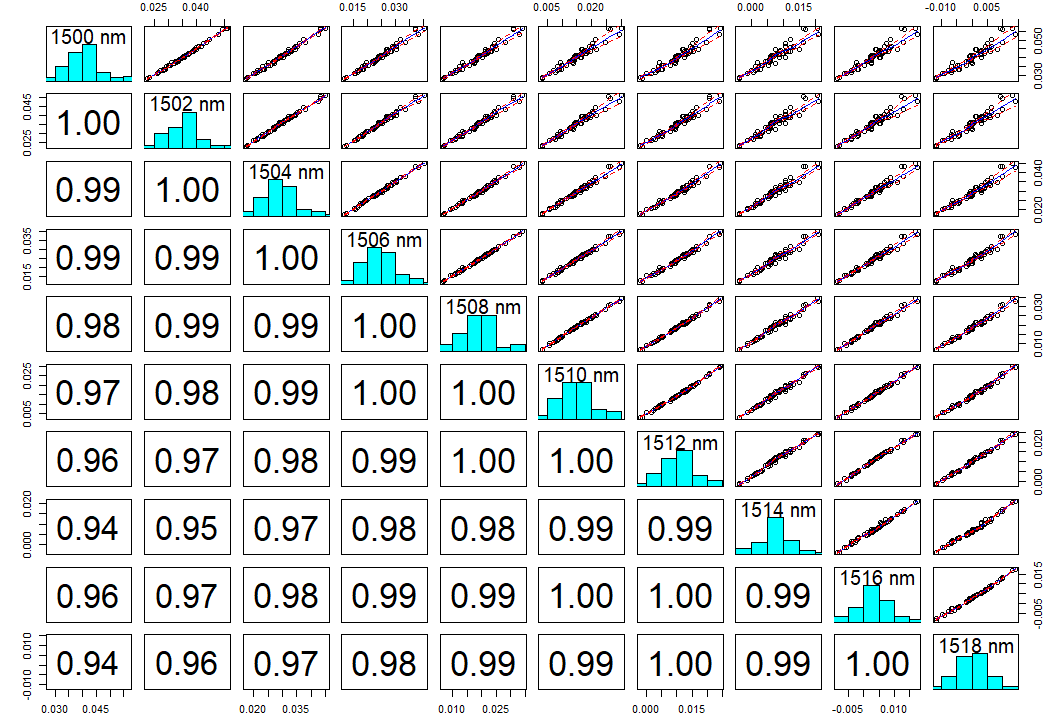
title(main="Spectral Readings for Gasoline Data")

pairs.plus(gasoline.x[,1:10])

pairs.plus(gasoline.x[,201:210])

pairs.plus(gasoline.x[,301:310])

In two or three sentences summarize what important features you see when examining these plots. (5 pts.)

Some of the important features that I have noticed it that the variables in the three correlation plots above have very high to almost perfect collinearity. The distribution in the data could add some issues to the data, so transforming the data could be necessary to get a fit.

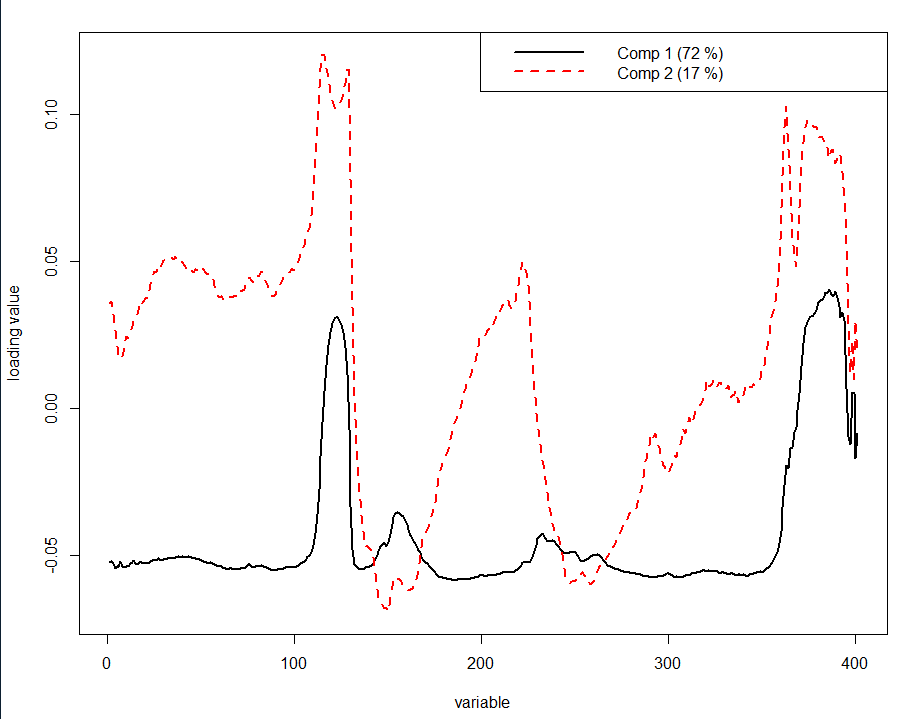
1. Now use the pcr() function as shown in yarn example in your notes to fit a PCR model for these data. What is the “optimal” # of components to use in your model? (4 pts.)

oct.pcr=pcr(octane~scale(NIR),data=gasoline,ncomp=40,validation=”CV”)

summary(oct.pcr)

The optimal number of components to use in the model is 6.

1. Examine the loadings on the components you used in your model. Use this plot to determine which NIR spectra load heavy on the 1st two principal components. Summarize your findings. (3 pts.)

loadingplot(oct.pcr,comps=1:2,lty=1:2,lwd=2,legendpos=”topright”)  
  


For the first principle component the variable ranges that have high loading are about 110 to 130 and 360 to 390. For the second component it was 110 to 130, around 360, and 380 to almost 400.

1. Form a training subset of the gasoline data as follows:

gasoline.train = gasoline[1:50,]

gasoline.test = gasoline[51:60,]

attributes(gasoline.train)

$names

[1] "octane" "NIR"

$row.names

[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" "14" "15" "16" "17" "18" "19" "20"

[21] "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40"

[41] "41" "42" "43" "44" "45" "46" "47" "48" "49" "50"

$class

[1] "data.frame"

dim(gasoline.train$NIR)

[1] 50 401

Using the optimal number of components chosen above, fit the model to these training data and predict the octane of the test cases using their NIR spectra. What is the RMSEP using the training/test set approach? (4 pts.)

Assuming you have already built a model called mymodel fit to the training data set do the following to obtain the predicted octanes for the observations in the test set.

oct.train = pcr(octane~scale(NIR),data=gasoline.train,ncomp=??) ypred = predict(oct.train,ncomp=??,newdata=gasoline.test)  
 yact = gasoline.test$octane

sqrt(mean((ypred – yact)^2)) 🡨 RMSEP

Note: ?? = the number of components you think should be used.

RMSEP = root mean squared error for prediction, i.e. square root of the mean PSE.

The RMSEP that was found with 6 principle components was .1721793.

1. Now use the plsr() function as shown in yarn example in your notes to fit a PLS model for these data. What is the “optimal” # of components to use in your model? (4 pts.)

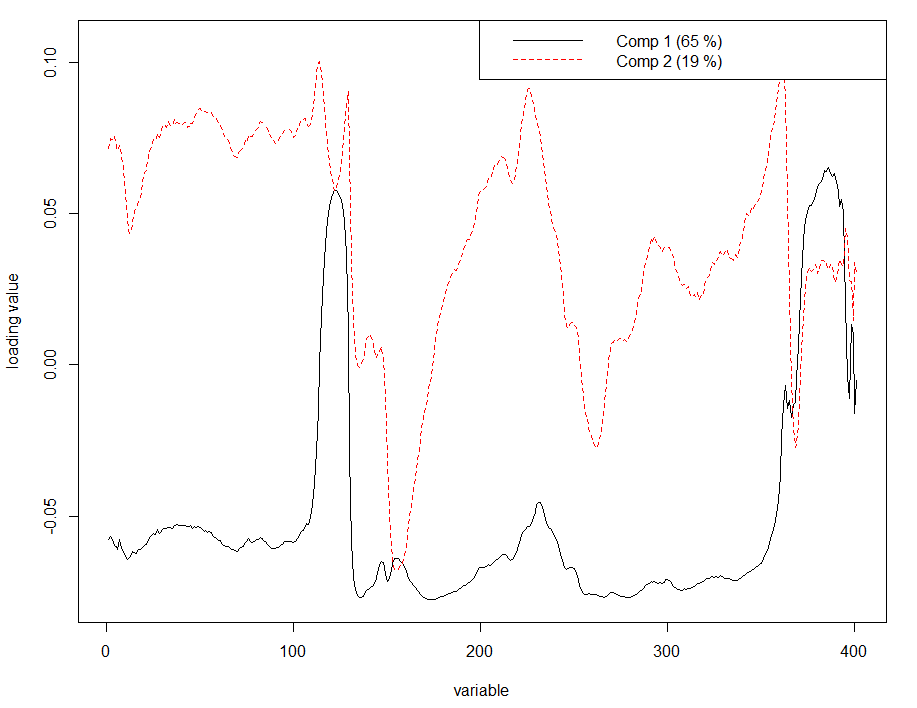
oct.pls = plsr(octane~scale(NIR),data=gasoline,ncomp=40,validation=”CV”)

summary(oct.pls)

The optimal number of components for the model are 5.

1. Examine the loadings on the components you used in your model. Use this plot to determine which NIR spectra load heavy on the 1st two principal components. Summarize your findings. (3 pts.)

loadingplot(oct.pls,comps=1:2,legendpos=”topright”)



For the first component the variables that have high loadings are similar to the variables with high loadings in the pcr model. The pattern is also similar to the pcr model. The second component does have different variables with high loading but the pattern in the loading are similar to the pcr model.

1. Form a training subset of the gasoline data as follows:

gasoline.train = gasoline[1:50,]

gasoline.test = gasoline[51:60,]

attributes(gasoline.train)

$names

[1] "octane" "NIR"

$row.names

[1] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10" "11" "12" "13" "14" "15" "16" "17" "18" "19" "20"

[21] "21" "22" "23" "24" "25" "26" "27" "28" "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40"

[41] "41" "42" "43" "44" "45" "46" "47" "48" "49" "50"

$class

[1] "data.frame"

dim(gasoline.train$NIR)

[1] 50 401

Using the optimal number of components chosen above, fit the model to these training data and predict the octane of the test cases using their NIR spectra. What is the RMSEP using a the training/test set approach? (4 pts.)

Assuming you have already built a training model called mymodel do the following to obtain the predicted octanes for the observations in the test set.

ypred = predict(*mymodel*,ncomp=??,newdata=gasoline.test)  
 yact = gasoline.test$octane

sqrt(mean((ypred-yact)^2)) 🡨 RMSEP

Note: ?? = the number of components you think should be used.

RMSEP = root mean squared error for prediction, i.e. square root of the mean PSE.

The RMSEP that was found with 5 components in .1665508.

1. Estimate the RMSEP using Monte Carlo Cross-Validation (MCCV) using

for both PLS and PCR (8 pts.)

The code for the function pls.cv is shown below. It takes the X’s, the response y, and the number of components to use in the PLS fit as the required arguments. Note the function computes RMSEP for each MC sample. Copy and paste this code into R. You can modify this code to do the same for PCR. Include the code you wrote for your pcr.cv() function. You will want to change the number of components (ncomp) to those you found to be optimal above.

pls.cv = function(X,y,ncomp=***2***,p=.667,B=100) {

n = length(y)

X = scale(X)

data = data.frame(X,y)

cv <- rep(0,B)

for (i in 1:B) {

ss <- floor(n\*p)

sam <- sample(1:n,ss,replace=F)

fit2 <- plsr(y~.,ncomp=ncomp,data=data[sam,])

ynew <- predict(fit2,ncomp=ncomp,newdata=data[-sam,])

cv[i] <- sqrt(mean((y[-sam]-ynew)^2,na.rm=T))

}

cv

}

The RMSEP using MCCV for PLS was .2150344.

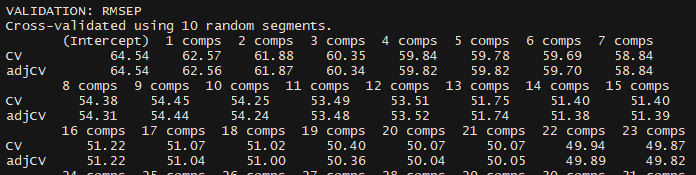
The RMSEP using MCCV for PCR was .2178565.

**Problem 2 – CRYSTAL MELTING POINT DATA**

In their 2005 paper “*General Boiling Point Prediction Based on a Diverse Compound Data Set and Artificial Neural Networks*”, Karthikeyan, Glen, and Bender examine methods for the prediction of melting points using a number of 2D and 3D descriptors that capture molecular physicochemical and other graph-based properties. The melting point is a fundamental physicochemical property of a molecule that is controlled by both single-molecule properties and intermolecular interactions due to packing in the solid state. Thus, it is difficult to predict, and previously only melting point models for clearly defined and smaller compound sets have been developed. The data frame QSAR.melt contains data for 4401 compounds that can be used to develop a model for melting point. There data are contained in the file **QSAR Melting Points (subset).csv** on the course website.

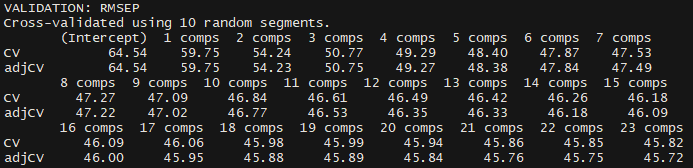
Goal: Use PCR and PLS develop models and compare their ability to predict melting   
 point.

1. Develop an “optimal” PCR model. Justify your choice using cross-validation. (10 pts.)



For the optimal component that I will be using for this model will be 14 due to the adjcv going up instead of down.

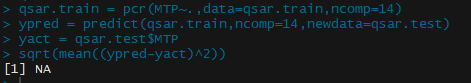
1. Develop an “optimal” PLS model. Justify your choice using cross-validation. (10 pts.)



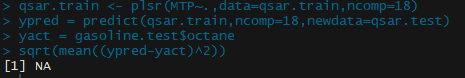
For the optimal component that I will be using for this model will be 18 due to the adjcv going up instead of down.

1. Use your “optimal” model for each method (PCR and PLS) to predict the melting point   
    of the test cases. Which modeling method performs the best when predicting the test   
    cases? (10 pts.)

PCR

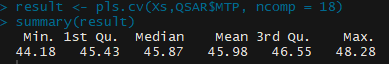


PLS

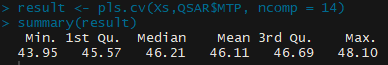


1. Use the pcr.cv and pls.cv functions to assess the predictive performance of these two   
    methods. Summarize the results. (10 pts.)

PLS



PCR



BONUS:

1. Use cross-validation to estimate the prediction error using the .632 Bootstrap for both PCR and   
    PLS using the number of components you chose as optimal. You can modify the code for the   
    bootstrap estimate of the prediction error for OLS MLR to these new situations. (10 pts.)

Some R code to get you started

Form Training and Test Datasets  
> QSAR.melt = read.table(file.choose(),header=T,sep=”,”)  
> set.seed(1)  
> QSAR.melt = QSAR.melt[,-1] # remove Case column which is an ID  
> train = sample(nrow(QSAR.melt),3900)  
> test = -(train)

> X = QSAR.melt[,-1] # grab all the predictors, Y = MTP is the 1st column  
> Xs = scale(X) # scale the predictors  
> QSAR = data.frame(MTP=QSAR.melt$MTP,Xs)  
> qsar.train = QSAR[train,]  
> qsar.test = QSAR[test,]

Fit PCR and PLS models to training data, you will need to decide how many components to have in your “final” model.  
> qsar.pcr = pcr(MTP~.,ncomp=40,validation=”CV”,data=qsar.train)  
> qsar.pls = plsr(MTP~.,ncomp=40,validation=”CV”,data=qsar.train)

Make predictions for test cases and compare to actual test case melting points (*y*)  
> ypred = predict(qsar.pcr,ncomp=??,newdata=qsar.test)  
> ytest = qsar.test$MTP  
> plot(ytest,ypred,xlab=”Actual Test MTP”,ylab=”Predicted Test MTP”)

> Rsq.pred = 1 – (sum((ypred-ytest)^2)/sum((ytest – mean(ytest))^2))

**Revised pcr.cv and pls.cv functions for THIS PROBLEM only.**

Because there are likely to be missing values on some of the in the test set, we will get missing values for the predicted response, thus producing a missing value for squared error for prediction. To fix that problem for these data, use line of code highlighted in bold in step where the **cv[i]** is calculated.

pls.cv = function(X,y,ncomp=***2***,p=.667,B=100) {

n = length(y)

X = scale(X)

data = data.frame(X,y)

cv <- rep(0,B)

for (i in 1:B) {

ss <- floor(n\*p)

sam <- sample(1:n,ss,replace=F)

fit2 <- plsr(y~.,ncomp=ncomp,data=data[sam,])

ynew <- predict(fit2,ncomp=ncomp,newdata=data[-sam,])

**cv[i] <- sqrt(mean((y[-sam]-ynew)^2,na.rm=T))**

}

cv

}

pcr.cv = function(X,y,ncomp=***2***,p=.667,B=100) {

n = length(y)

X = scale(X)

data = data.frame(X,y)

cv <- rep(0,B)

for (i in 1:B) {

ss <- floor(n\*p)

sam <- sample(1:n,ss,replace=F)

fit2 <- pcr(y~.,ncomp=ncomp,data=data[sam,])

ynew <- predict(fit2,ncomp=ncomp,newdata=data[-sam,])

**cv[i] <- sqrt(mean((y[-sam]-ynew)^2,na.rm=T))**

}

cv

}