NIR Spectra & Gasoline Octane

Samuel Andrews & Mikolaj Wieczorek

3/5/2020

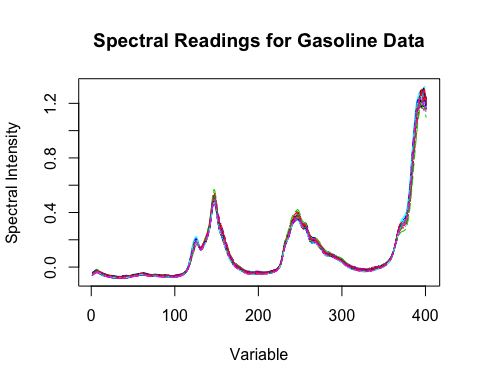
setwd("~/OneDrive - MNSCU/myGithub/Supervised\_Learning/Multiple\_Linear\_Regression/MachineLearning-SupervisedLearning/PrincipalComponentRegression")  
data(gasoline)

This data consits of n=30 subjects and 403 columns that represent genetic marker intensity measurments for 403 different genes.

gasoline.x = gasoline$NIR  
dim(gasoline.x)

## [1] 60 401

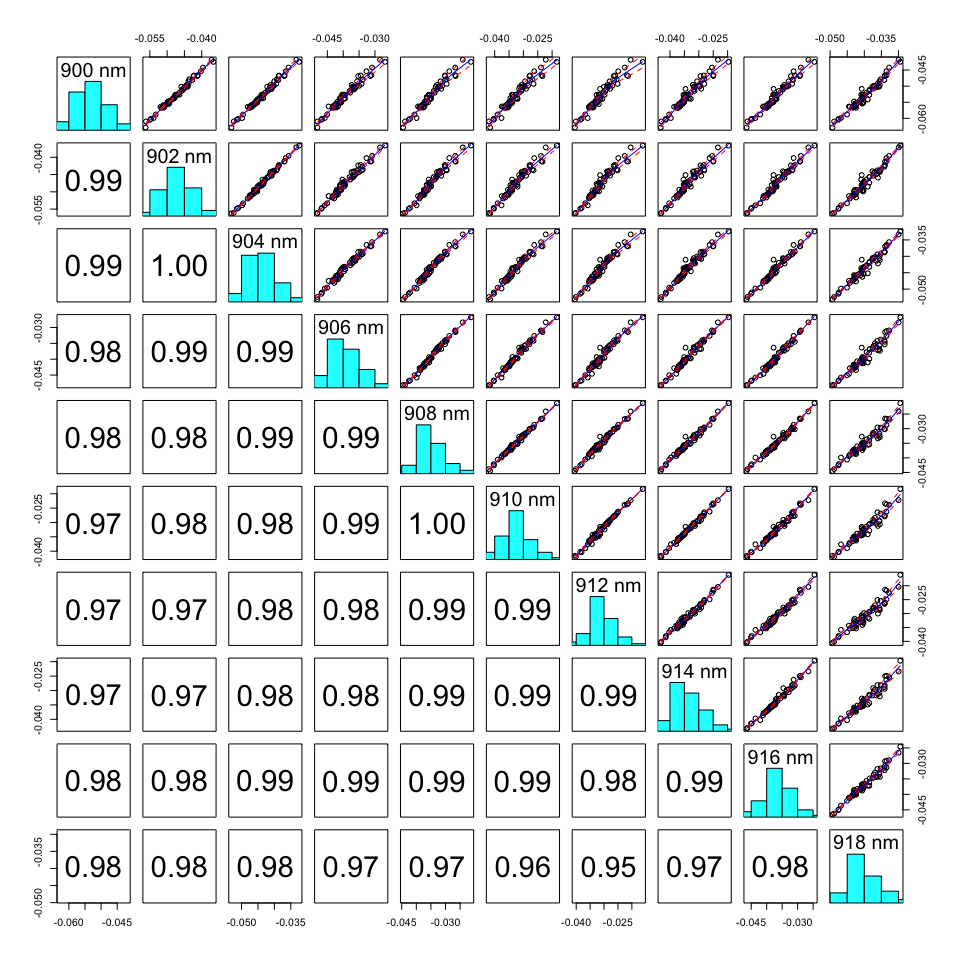
matplot(t(gasoline.x),type="l",xlab="Variable",ylab="Spectral Intensity")  
title(main="Spectral Readings for Gasoline Data")



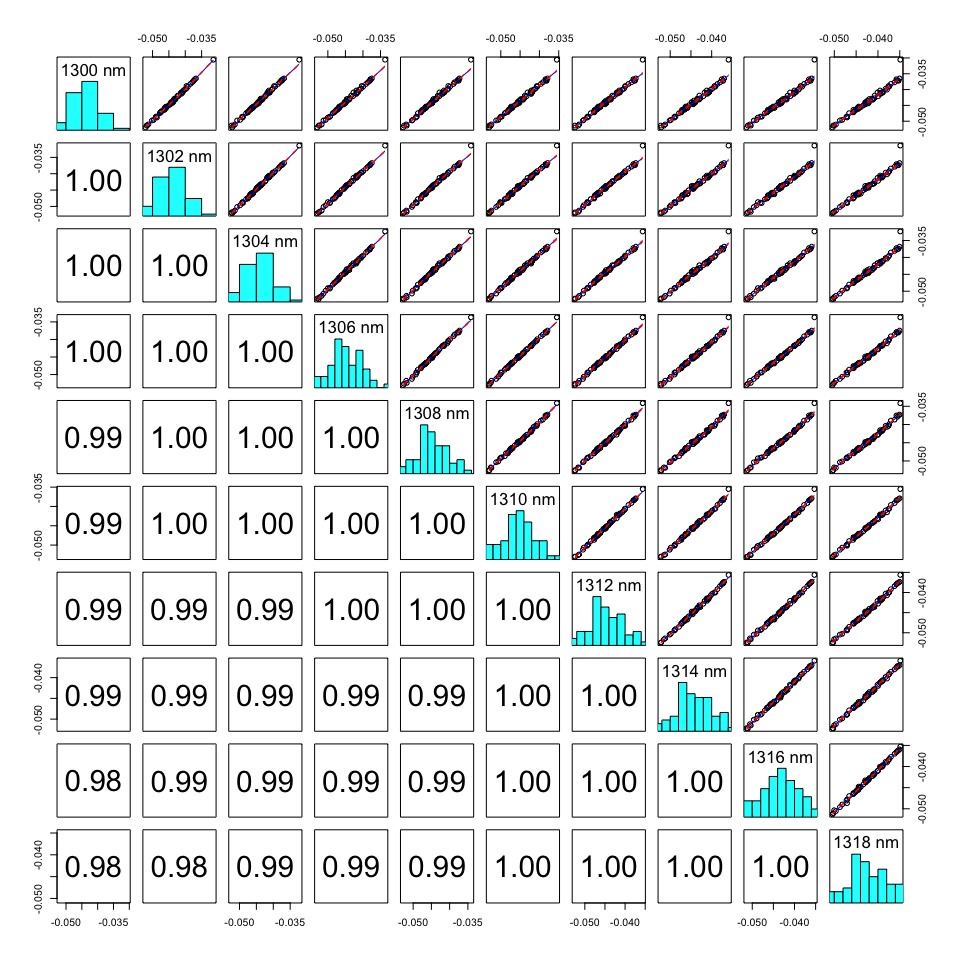
The graph above shows how the value of spectral intensity changes based on the variables in the data.

### Correlation structure

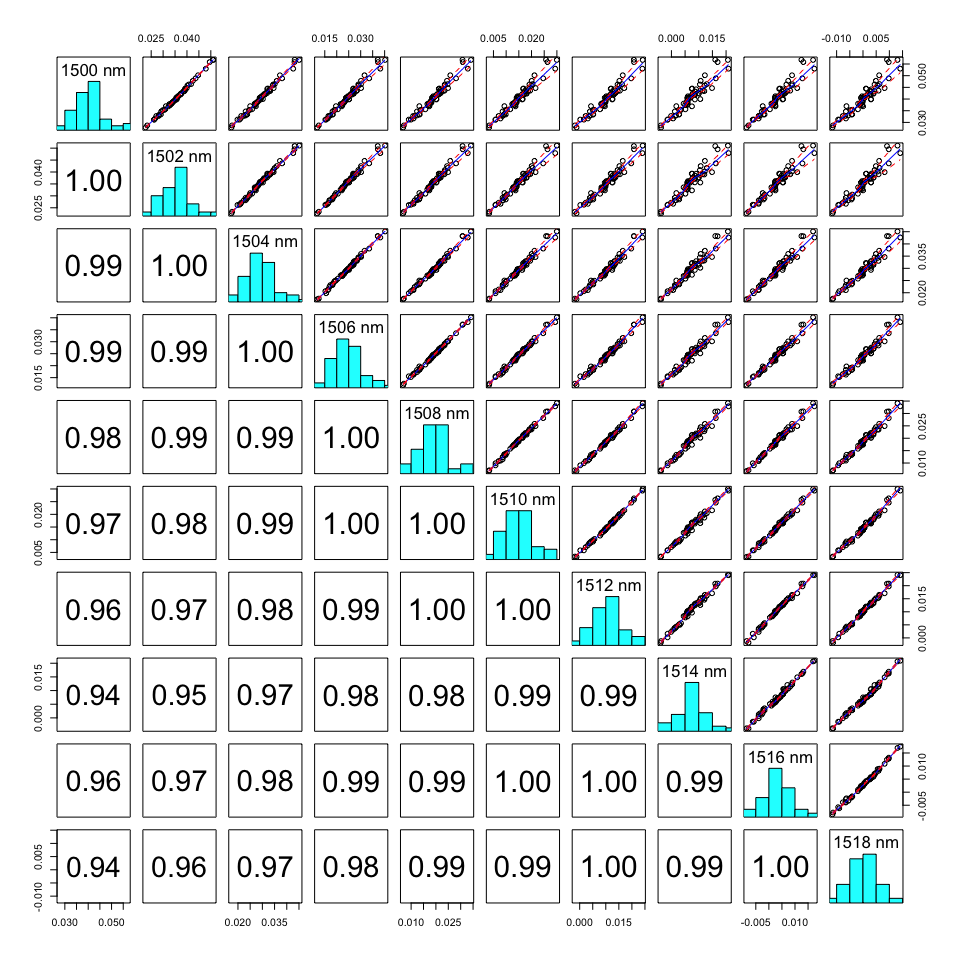
pairs.plus(gasoline.x[,1:10])



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

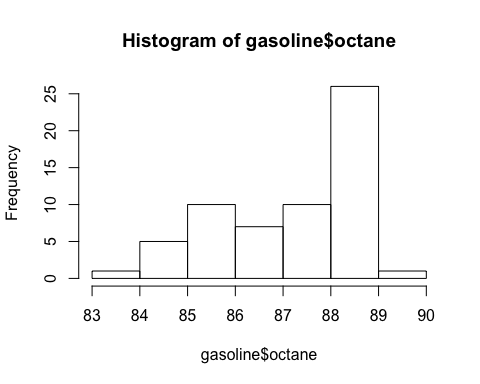


pairs.plus(gasoline.x[,301:310])

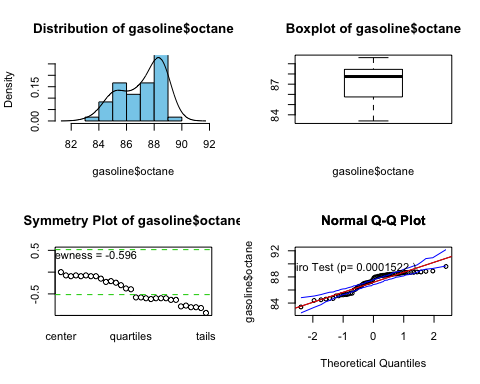


As the variables are very higly correlated (> .94), our principal components will help reduce the dimentionality of the data. This is depicted by the scatterplot matrices generated above with the pairs.plus() function. Those are very higly correlated with each other and we are going to use princiapl components and partial least squares in predicting our response as they will make sure that we are not redundant in fitting the model.

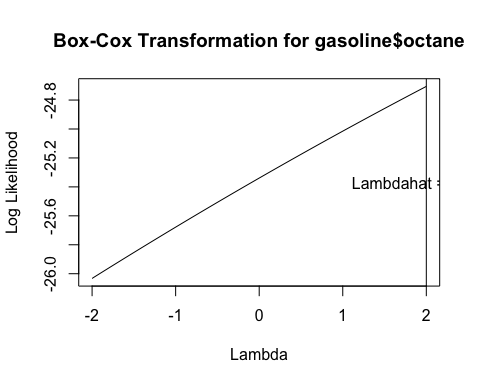
#Explore the response variable  
par(mfrow=c(1,1))  
hist(gasoline$octane)



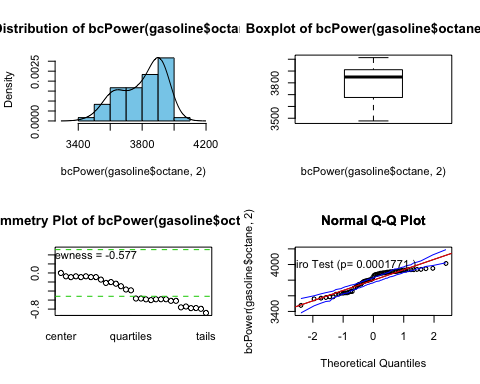
Statplot(gasoline$octane)



BCtran(gasoline$octane)



Statplot(bcPower(gasoline$octane, 2))



There is not much skewness prevailing. When aplpying a suggested optimal lambda=2 Box-Cox transformation, the skewness hasn’t changed that much to apply the transformation on the response (octane).

## PART B

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)

## Data: X dimension: 60 401   
## Y dimension: 60 1  
## Fit method: svdpc  
## Number of components considered: 40  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 1.543 1.484 1.412 0.3918 0.2649 0.2226 0.2074  
## adjCV 1.543 1.482 1.408 0.3289 0.2640 0.2216 0.2030  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 0.2089 0.2156 0.2116 0.2124 0.2163 0.2095 0.2103  
## adjCV 0.2076 0.2166 0.2088 0.2097 0.2139 0.2067 0.2074  
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
## CV 0.2119 0.2208 0.2155 0.2269 0.2298 0.2356  
## adjCV 0.2091 0.2187 0.2125 0.2242 0.2259 0.2300  
## 20 comps 21 comps 22 comps 23 comps 24 comps 25 comps  
## CV 0.2393 0.2417 0.2514 0.2572 0.2434 0.2493  
## adjCV 0.2340 0.2361 0.2451 0.2517 0.2374 0.2433  
## 26 comps 27 comps 28 comps 29 comps 30 comps 31 comps  
## CV 0.2527 0.2669 0.2690 0.2512 0.2533 0.2609  
## adjCV 0.2460 0.2594 0.2615 0.2448 0.2479 0.2561  
## 32 comps 33 comps 34 comps 35 comps 36 comps 37 comps  
## CV 0.2608 0.2464 0.2614 0.2467 0.2524 0.2267  
## adjCV 0.2534 0.2401 0.2528 0.2368 0.2426 0.2194  
## 38 comps 39 comps 40 comps  
## CV 0.2302 0.2410 0.2407  
## adjCV 0.2229 0.2329 0.2330  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 71.725 88.57 93.74 97.51 98.28 98.67 99.01  
## octane 8.856 22.69 96.39 97.40 98.18 98.51 98.51  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
## X 99.20 99.36 99.48 99.57 99.64 99.70 99.74  
## octane 98.57 98.79 98.79 98.81 98.88 98.88 98.88  
## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
## X 99.78 99.81 99.83 99.85 99.86 99.88  
## octane 98.88 98.93 98.93 99.00 99.05 99.08  
## 21 comps 22 comps 23 comps 24 comps 25 comps 26 comps  
## X 99.89 99.90 99.91 99.92 99.93 99.94  
## octane 99.10 99.12 99.13 99.21 99.24 99.28  
## 27 comps 28 comps 29 comps 30 comps 31 comps 32 comps  
## X 99.94 99.95 99.95 99.95 99.96 99.96  
## octane 99.30 99.33 99.36 99.37 99.38 99.47  
## 33 comps 34 comps 35 comps 36 comps 37 comps 38 comps  
## X 99.96 99.97 99.97 99.97 99.98 99.98  
## octane 99.51 99.55 99.60 99.61 99.62 99.62  
## 39 comps 40 comps  
## X 99.98 99.98  
## octane 99.64 99.64

Based on the CV RMSEP, it is optimal to retain 6 principal components (19.72%).

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] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"  
## [29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"  
## [43] "43" "44" "45" "46" "47" "48" "49" "50"  
##   
## $class  
## [1] "data.frame"

dim(gasoline.train$NIR)

## [1] 50 401

### Check optimal # of component predictions

Let’s explore the optimal number of components further with our oct.pcr model, where we are considering within ncomp=40.

#10 components  
ypred = predict(oct.pcr, ncomp = 11, newdata = gasoline.test)  
paste("RMSEP for testing:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing: 0.0180920986810605"

#9 components  
ypred = predict(oct.pcr, ncomp = 9, newdata = gasoline.test)  
paste("RMSEP for testing, 9PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 9PCs: 0.0256621310600451"

#7 components  
ypred = predict(oct.pcr, ncomp = 7, newdata = gasoline.test)  
paste("RMSEP for testing, 7PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 7PCs: 0.000898561170103562"

#6 components  
ypred = predict(oct.pcr, ncomp = 6, newdata = gasoline.test)  
paste("RMSEP for testing, 6PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 6PCs: 0.000887087398619712"

#5 components  
ypred = predict(oct.pcr, ncomp = 5, newdata = gasoline.test)  
paste("RMSEP for testing, 5PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 5PCs: 0.0274881902636182"

#4 components  
ypred = predict(oct.pcr, ncomp = 4, newdata = gasoline.test)  
paste("RMSEP for testing, 4PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

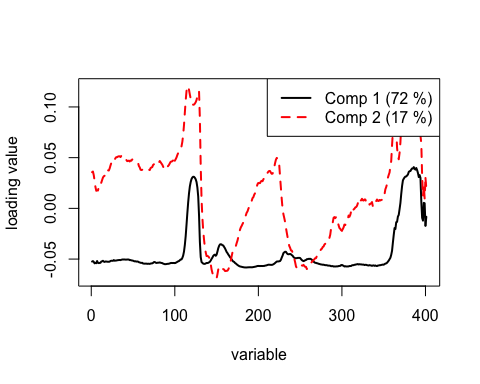
## [1] "RMSEP for testing, 4PCs: 0.0945304325000407"

Optimal number of components to retain is 6, indeed, 6.

## PART C

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")



Each pc comnponent takes pieces of variables. The loadings plot above depicts which component takes in which ranges of pieces of variables. PC1 contains pieces of variables 110-120 and 350-400. PC2 is similar to PC1 but it puts more emphasis on variables 110-120 and 350-400 than PC1. PC2 also contains a lot information about some of the 170-230 variables.

## PART D

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.)

We picked ncomp=6.

oct.train = pcr(octane~scale(NIR),data=gasoline.train,ncomp=6)

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

Let’s see how the model oct.train, that was fir with ncomp=6, predicts on the test set with ncomp=6.

ypred = predict(oct.train,ncomp=6,newdata=gasoline.test)  
yact = gasoline.test$octane  
paste("RMSEP:", sqrt(mean((ypred-yact)^2)))

## [1] "RMSEP: 0.172179304321268"

# Partial Least Squares Regression (PLS)

## PART E

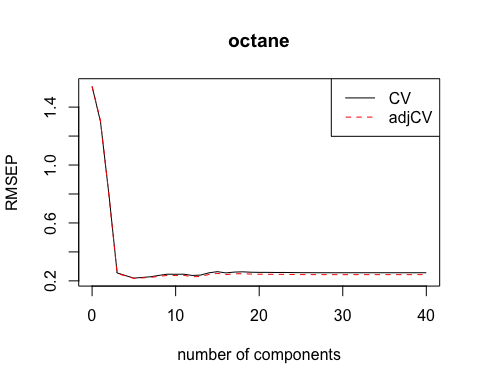
Now use the plsr() function as shown in yarn example in your notes to fit a PLS model for these data. What is the “optimal” # of components to use in your model? (4 pts.)

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

## Data: X dimension: 60 401   
## Y dimension: 60 1  
## Fit method: kernelpls  
## Number of components considered: 40  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 1.543 1.300 0.8034 0.2548 0.2365 0.2189 0.2237  
## adjCV 1.543 1.307 0.7981 0.2508 0.2348 0.2170 0.2203  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 0.2282 0.2378 0.2460 0.2455 0.2459 0.2373 0.2401  
## adjCV 0.2240 0.2311 0.2374 0.2365 0.2379 0.2283 0.2305  
## 14 comps 15 comps 16 comps 17 comps 18 comps 19 comps  
## CV 0.2557 0.2635 0.2550 0.2605 0.2624 0.2597  
## adjCV 0.2444 0.2519 0.2433 0.2483 0.2497 0.2469  
## 20 comps 21 comps 22 comps 23 comps 24 comps 25 comps  
## CV 0.2587 0.2586 0.2579 0.2580 0.2572 0.2565  
## adjCV 0.2457 0.2456 0.2448 0.2449 0.2441 0.2433  
## 26 comps 27 comps 28 comps 29 comps 30 comps 31 comps  
## CV 0.2564 0.2561 0.2561 0.2559 0.2560 0.2560  
## adjCV 0.2432 0.2429 0.2430 0.2428 0.2429 0.2429  
## 32 comps 33 comps 34 comps 35 comps 36 comps 37 comps  
## CV 0.2560 0.2560 0.2560 0.2560 0.2560 0.2560  
## adjCV 0.2429 0.2429 0.2429 0.2429 0.2429 0.2429  
## 38 comps 39 comps 40 comps  
## CV 0.2560 0.2560 0.2560  
## adjCV 0.2429 0.2429 0.2429  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 64.97 83.51 93.72 96.33 98.21 98.58 98.81  
## octane 30.54 79.79 97.73 98.27 98.67 98.90 99.05  
## 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps  
## X 98.89 99.02 99.19 99.44 99.49 99.55 99.60  
## octane 99.29 99.44 99.53 99.57 99.69 99.76 99.83  
## 15 comps 16 comps 17 comps 18 comps 19 comps 20 comps  
## X 99.69 99.72 99.76 99.78 99.81 99.83  
## octane 99.86 99.90 99.93 99.95 99.97 99.99  
## 21 comps 22 comps 23 comps 24 comps 25 comps 26 comps  
## X 99.85 99.87 99.88 99.89 99.9 99.91  
## octane 99.99 99.99 100.00 100.00 100.0 100.00  
## 27 comps 28 comps 29 comps 30 comps 31 comps 32 comps  
## X 99.91 99.92 99.93 99.93 99.94 99.94  
## octane 100.00 100.00 100.00 100.00 100.00 100.00  
## 33 comps 34 comps 35 comps 36 comps 37 comps 38 comps  
## X 99.95 99.96 99.96 99.96 99.96 99.97  
## octane 100.00 100.00 100.00 100.00 100.00 100.00  
## 39 comps 40 comps  
## X 99.97 99.97  
## octane 100.00 100.00

Based on the CV RMSEP, it is optimal to retain 5 principal components (20.56%).

plot(RMSEP(oct.pls), legendpos = "topright",)



It looks like the optimal number of components is somewhere between 4-6.

Let’s ecplore choosing the optimal number of components further.

#4 components  
ypred = predict(oct.pls, ncomp = 4, newdata = gasoline.test)  
paste("RMSEP for testing, 4PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 4PCs: 0.141324145004918"

#5 components  
ypred = predict(oct.pls, ncomp = 5, newdata = gasoline.test)  
paste("RMSEP for testing, 5PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 5PCs: 0.00365425120471343"

#6 components  
ypred = predict(oct.pls, ncomp = 6, newdata = gasoline.test)  
paste("RMSEP for testing, 6PCs:", sqrt(mean(ypred-gasoline.test$octane)^2))

## [1] "RMSEP for testing, 6PCs: 0.00699294540983573"

Based on the above using 5 components yields lowest RMSEP.

We picked ncomp=5.

oct.train.pls = plsr(octane~scale(NIR),data=gasoline.train,ncomp=5)

Let’s see how the model oct.train.pls, that was fit with ncomp=5, predicts on the test set with ncomp=5.

ypred = predict(oct.train.pls,ncomp=5,newdata=gasoline.test)  
yact = gasoline.test$octane  
paste("RMSEP:", sqrt(mean((ypred-yact)^2)))

## [1] "RMSEP: 0.443601936176341"

However, ncomp=4 is also worth exploring further in Monte Carlo Cross-Validation (PART H) as we are getting RMSEP=.44.

oct.train.pls = plsr(octane~scale(NIR),data=gasoline.train,ncomp=4)

Let’s see how the model oct.train.pls, that was fit with ncomp=4, predicts on the test set with ncomp=4.

ypred = predict(oct.train.pls,ncomp=4,newdata=gasoline.test)  
yact = gasoline.test$octane  
paste("RMSEP:", sqrt(mean((ypred-yact)^2)))

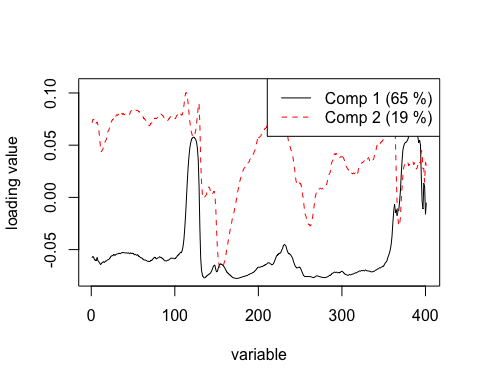
## [1] "RMSEP: 0.182541871464115"

ncomp=4, yileds RMSEP=.18.

## PART F

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")



The two components explain about 88% of total variation (PC1=.65, PC2=.19). Here, PC1 is pretty similar to the loadings when we used pcr, with a bit more empahsis on the variables 360-400. Also it contains more informaiton about pieces of variables between 110-120. PC2, here, puts a lot emphasis on variables 170-230 as well as 350-360; it doesn’t emphasize variables >350 as much as PC1 on this figure (or PC2 from loadings of pcr).

## PART G

mymodel = plsr(octane~scale(NIR),data=gasoline.train,ncomp=4)

ypred = predict(mymodel,ncomp=4,newdata=gasoline.test)  
yact = gasoline.test$octane  
paste("RMSEP:",sqrt(mean((ypred-yact)^2)))

## [1] "RMSEP: 0.182541871464115"

## PART H

Estimate the RMSEP using Monte Carlo Cross-Validation (MCCV) using p=.80 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. 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=5,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=6,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,])  
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))  
}  
#check = return(data.frame(RMSEP=cv))  
cv  
}

### PCR

The final optimal number of components we chose for our Principal Component Regression model is ncomp=6.

final.pcr.cv = pcr.cv(gasoline$NIR, gasoline$octane, ncomp = 6, p=.80, B=1000)  
mean(final.pcr.cv)

## [1] 0.2155044

### PLS

The final optimal number of components we chose for our Partial Least Squares Regression model is ncomp=5.

We ran the Monte Carlo Cross-Validation multiple times with ncomp=4 and ncomp=5 and the best results with least variance were when we used 5 components to fir out plsr model.

final.pls.cv = pls.cv(gasoline$NIR, gasoline$octane, ncomp = 5 , p=.80, B=1000)  
mean(final.pls.cv)

## [1] 0.2144094

df = data.frame(PCR.RMSEP = mean(final.pcr.cv), PLS.RMSEP = mean(final.pls.cv))  
df

## PCR.RMSEP PLS.RMSEP  
## 1 0.2155044 0.2144094

Comparing our final PCR and PLS models, PLS performed slightly better.