STAT40150 - Assignment 2

Jamie Kennedy - 17372983

Question 1

The first thing to do is load in the data and delete a random row from the data.

```r
# First load in the data
data = read.csv("Grain_data.csv",header=TRUE,na.strings = '..')
# set the seed to student number
set.seed(17372983)
# check how many rows are in the data
rows = nrow(data)
# generate a random row
random = sample(1:rows,1)
# delete that random row
data = data[-c(random),]
```

Question 2

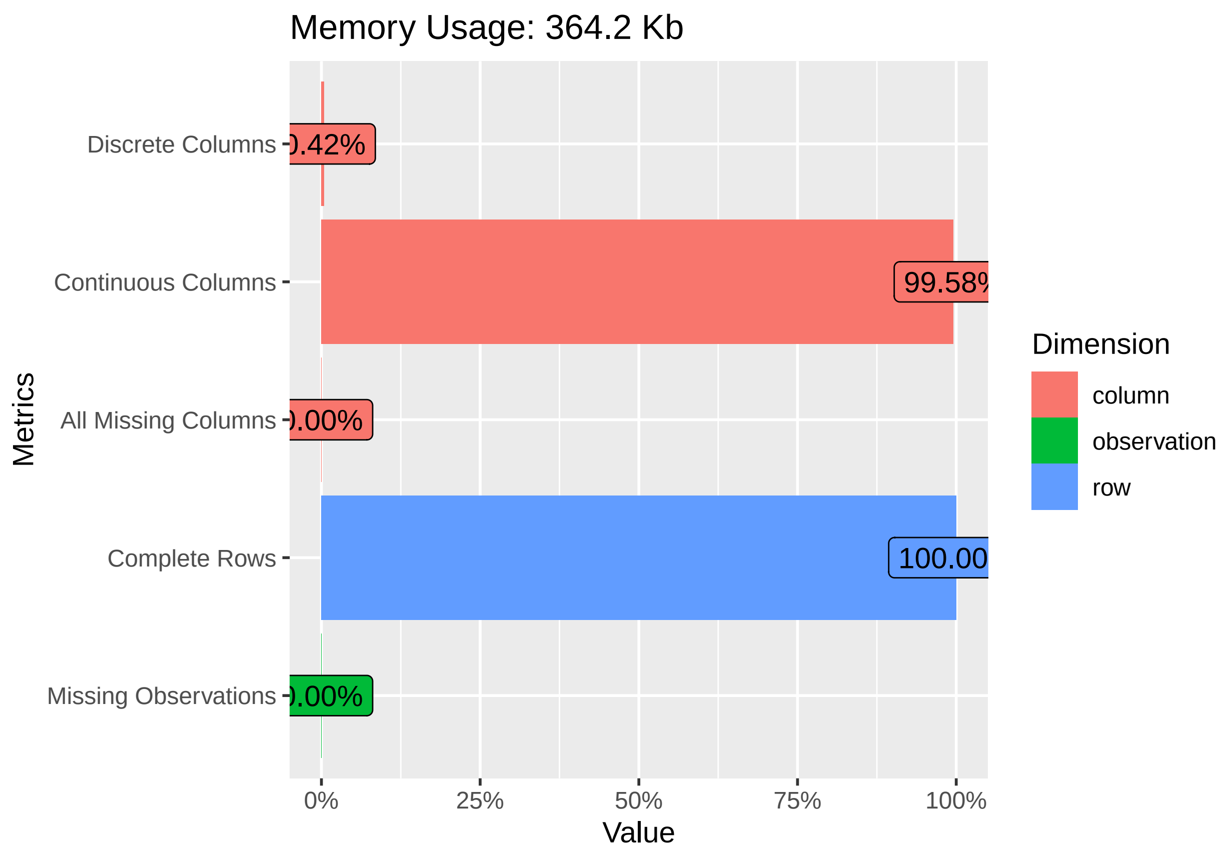
We want to visualize the spectra and the traits. I begin by just getting an overview of the data

```
### Question 2
# Let's visualize the data to get a better understanding of it. This library can give
# some pretty nice visualizations
library(DataExplorer)

# I just want to get a general idea of the data
introduce(data)
```

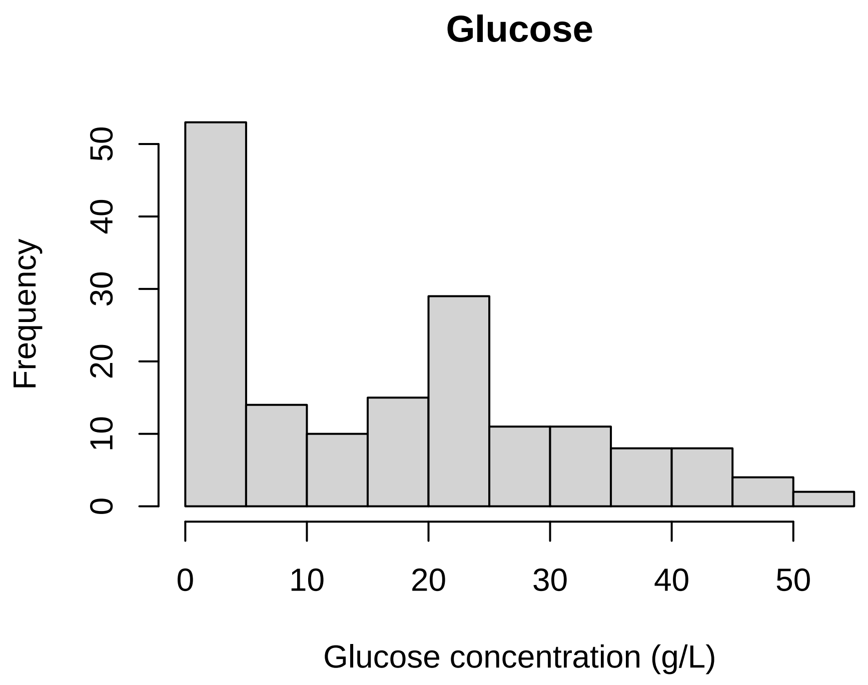
```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1  165     238                1                237                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                    0           165              39270       372976
```

```
plot_intro(data)
```

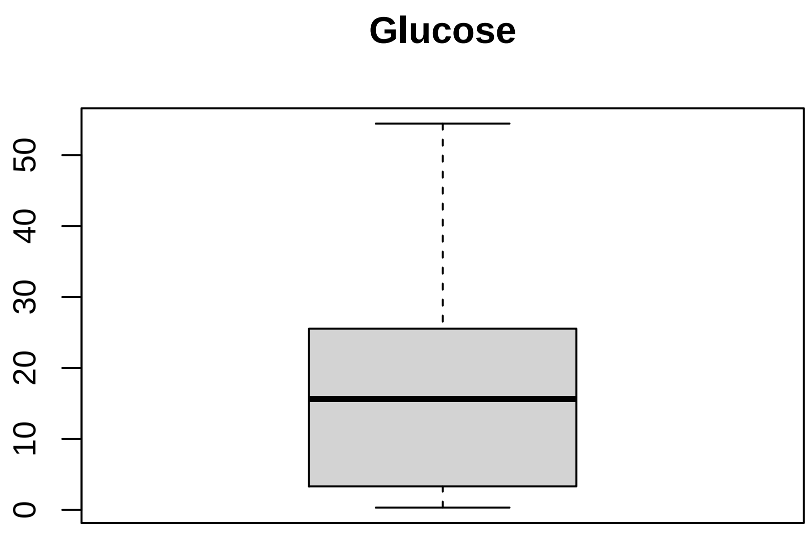


It is possible use histograms and boxplots to learn about the distribution of the data for the Glucose and Ethanol traits. We focus on Glucose first.

```
hist(data$Glucose,main = "Glucose",xlab="Glucose concentration (g/L)")
```



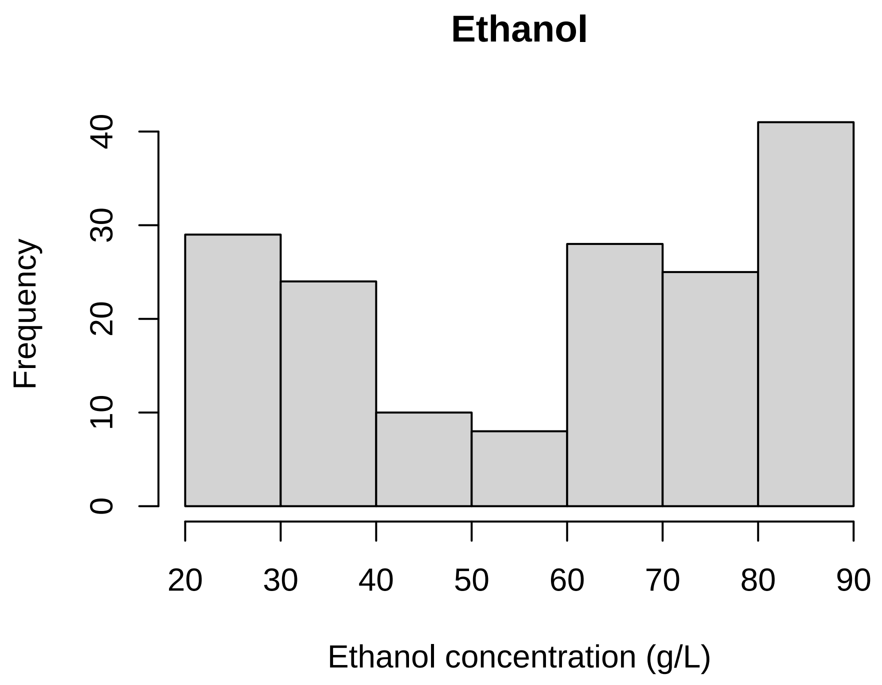
```
boxplot(data$Glucose,main = "Glucose")
```



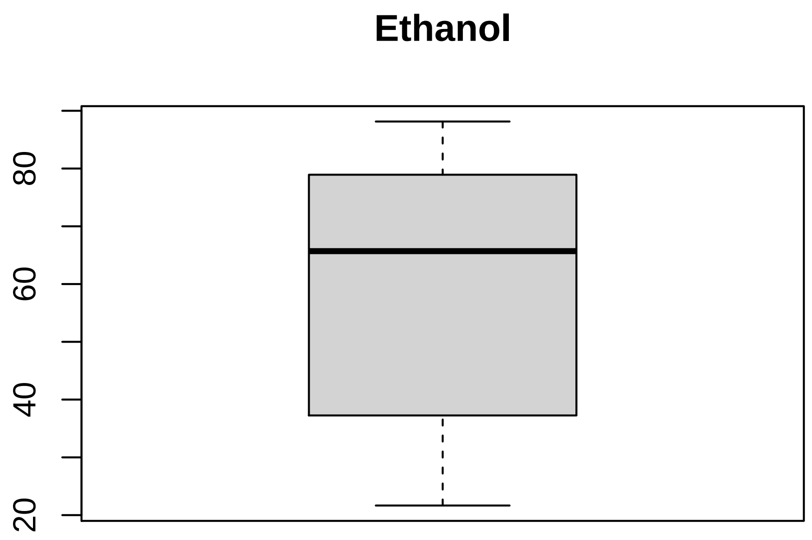
We see that the Glucose trait has a right-skewed distribution which means the mean is greater than the median which in turn is greater than the mode. We also learn from the boxplot that there aren't any outliers.

Now lets focus on Ethanol.

```r
hist(data$Ethanol,main = "Ethanol",xlab="Ethanol concentration (g/L)")
```

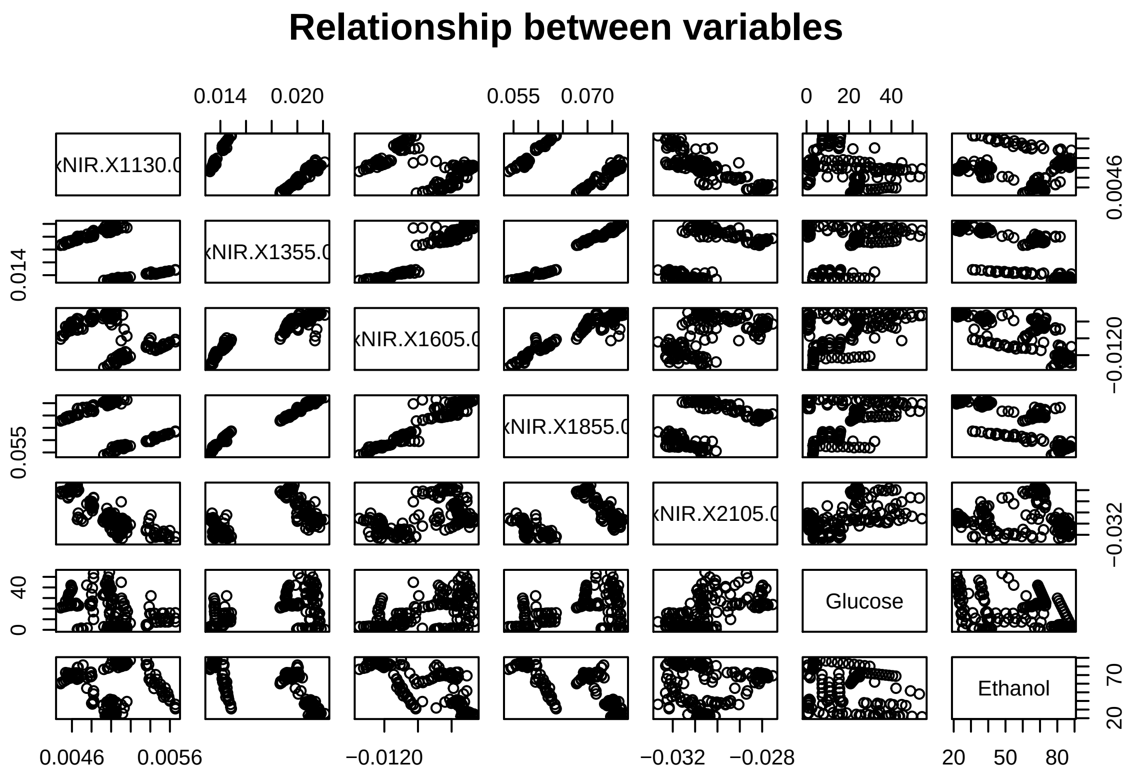


```r
boxplot(data$Ethanol,main = "Ethanol")
```



The histogram of Ethanol values tells us that both high and low values are common but it is less common to see values in between the two. Again, the boxplot tells us that there are no outliers.
Next I want to learn more about the relationships between the different variables. We take a small sample of the NIR variables and look at a pairs plot which compares these variables to the Ethanol and Glucose traits.

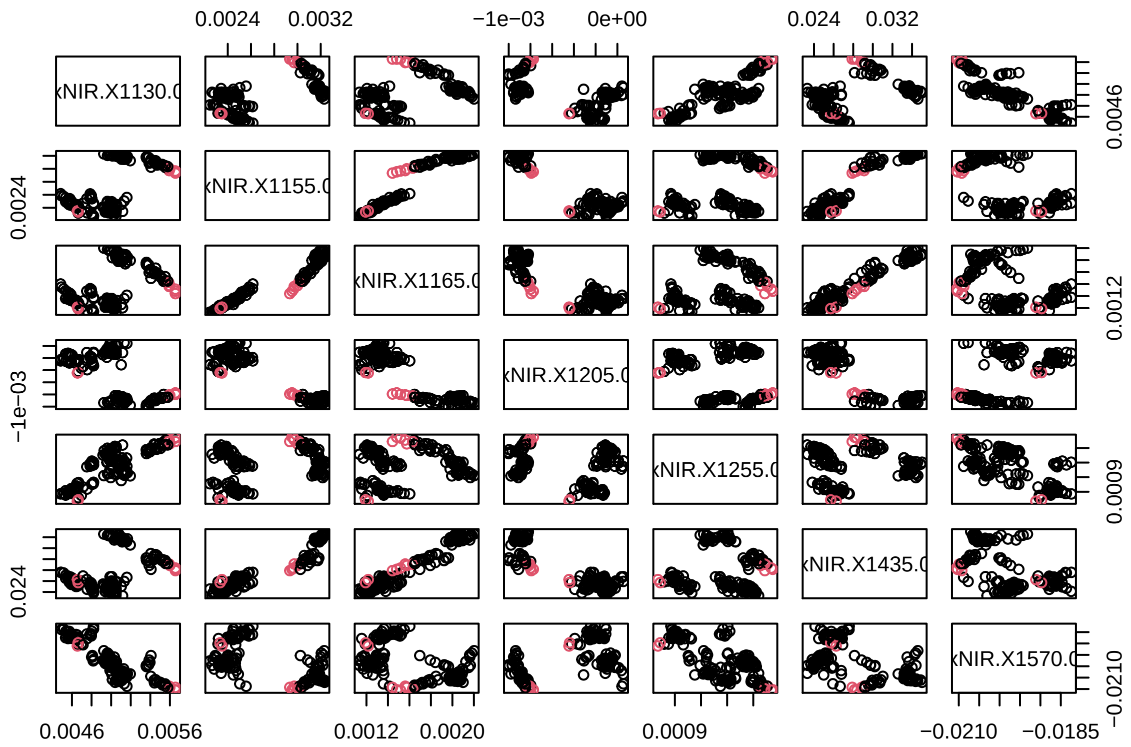
```
# want to see how ethanol and glucose relate to spectra variables
set2 <- c(5,50,100,150,200,237,238)  # just want to look at a sample of variables from data
pairs(data[,set2],cex.labels=1,main="Relationship between variables")
```



This tells us that the variables relating to NIR spectra have quite strong relationships with eachother. For example, NIR.X1605 and NIR.X1855 have a strong positive correlation. The relationships between the NIR spectra variables and the traits are less clear due to the less obvious patterns. One pattern we do see is that for varying levels of Ethanol, the majority of the NIR spectra variables will either take quite a high value or quite a low value i.e. the values of observations for these variables are split into two distinct groups.
From the pairs plot there doesn't seem to be many outliers in the variables relating to the NIR spectra but we will investigate further by looking at a sample of these variables and seeing if there are any observations with a value that differs greatly from the mean of that variable. Originally, this code checked if any observations were more than 3 standard deviations from the mean but none were so then we look at observations which are more than 2 standard deviations from the mean.

```
# Look to see if outliers within NIR spectra variables by looking at values more
# than 2 s.d's from mean
set <- c(5,10,12,20,30,66,93)  # get sample of variables from NIR spectra data
samp = data[,set]
mu = apply(samp,2,mean)
sds = apply(samp,2,sd)
lower = mu - 2*sds    # tried 3 s.d's but no samples changed colour
```

```
upper = mu + 2*sds
ind = NULL
for(i in 1:7)
{
  ind = c(ind, which((samp[,i] <lower[i]) | (samp[,i] >upper[i])))
}
n = nrow(samp)
cols = rep(1,n)
cols[ind] = 2
pairs(data[,set], cex.labels= 1, col=cols)
```

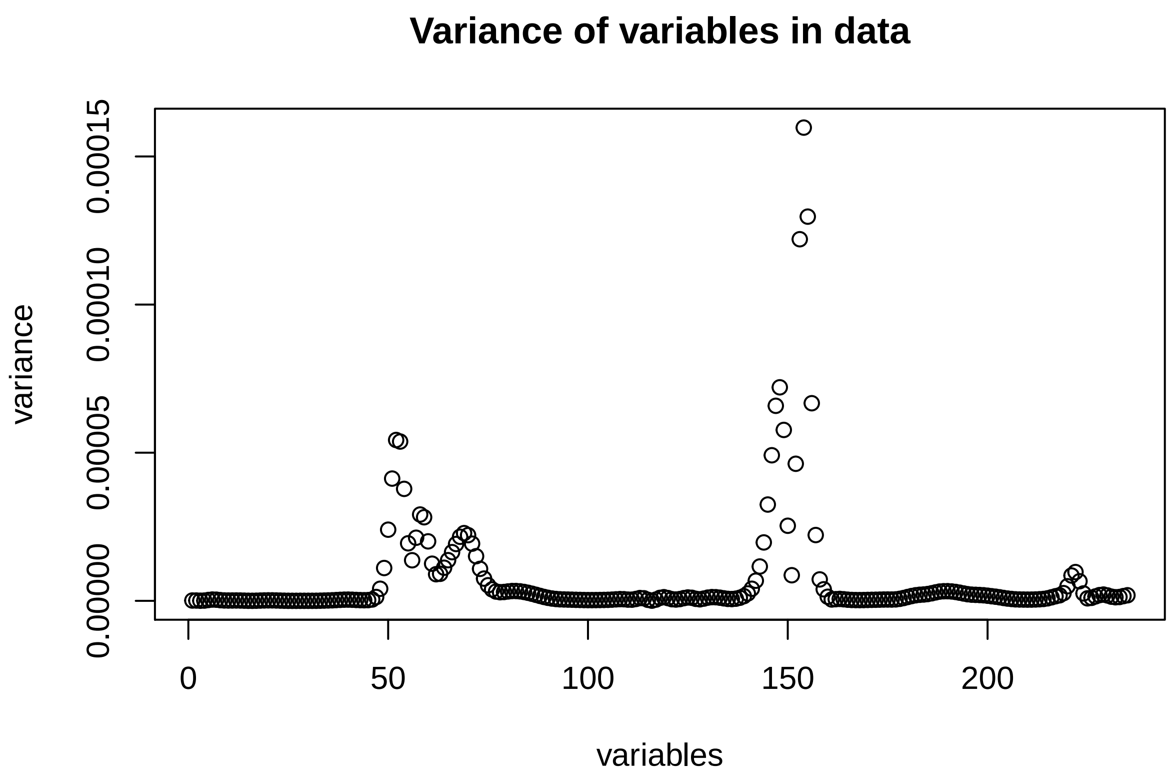


From this plot we can see that there are very few observations with a value more than 2 standard deviations from the mean and even when this does occur, the extreme observations don't differ from the majority of the observations by too much. Thus, none of these extreme observations are removed from the data.

Question 3

We want to apply PCA to the spectral data. The first thing to do is get the data and check if it needs to be scaled by looking at the variance of the different variables. We find that the variance is very low across the data so there is no need to scale the data. We know this because the plot below shows the highest variance is around 0.00015 which is very small.

```
### Question 3
# need spectra data
vars = data[,2:236]
# look to see if we need to scale data
check=apply(vars, 2, var)
plot(check,main="Variance of variables in data",ylab="variance",xlab="variables")
```



```
# Don't scale as variance is really low across the data
```

Next we want to actually apply PCA to the data. The reason for this is because our data has a large number of variables and we want to see if we can account for most of the information in the data with a much smaller number of variables.

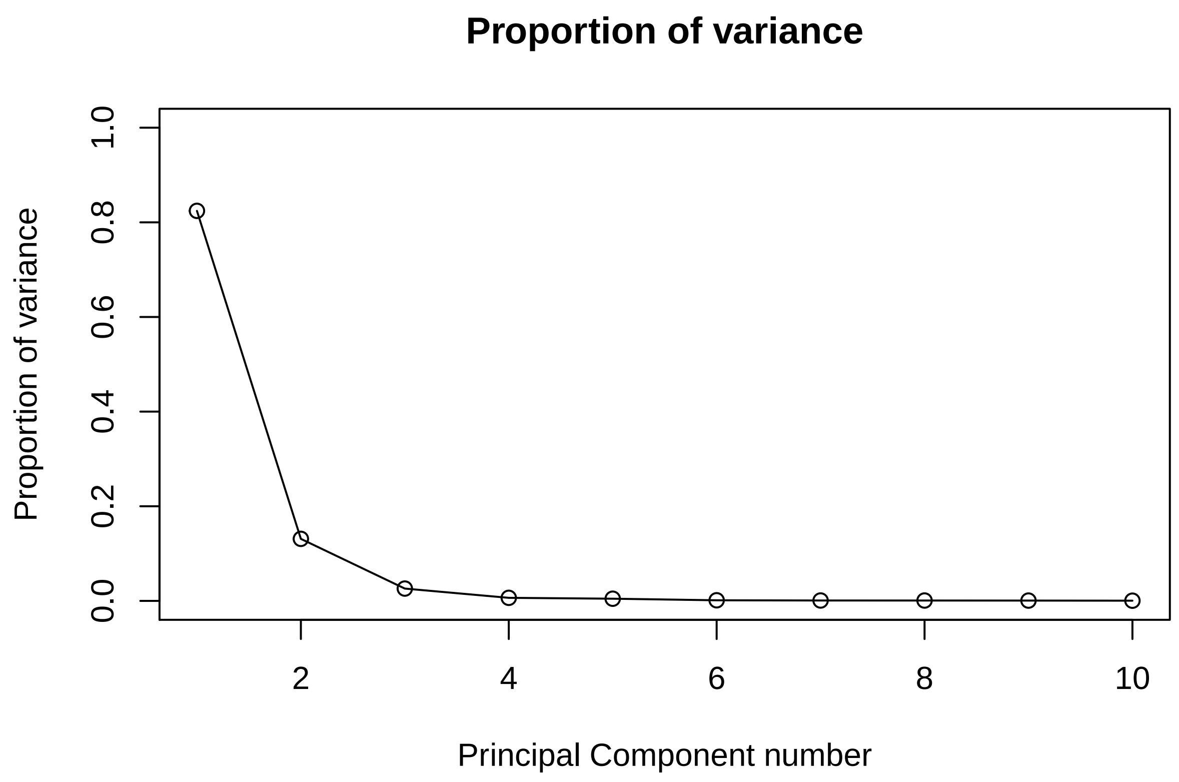
```
fit = prcomp(vars)
```

Once we have applied PCA we want to look at the variance of our principal components to decide how many principal components are needed to capture most of the information in our data. We will look at both proportion of variance and cumulative proportion of variance for each principal component. All of the information we need to do this is stored in the importance section of our summary of PCA.

```r
x = summary(fit)$importance # we want importance as it contains proportion of variance and
#cumulative proportion of variance in its second row.
#look at plot of proportion of variance explained by each principal component
x[2,1:10]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
## 0.82427 0.13110 0.02606 0.00645 0.00458 0.00129 0.00084 0.00079 0.00064 0.00041
```

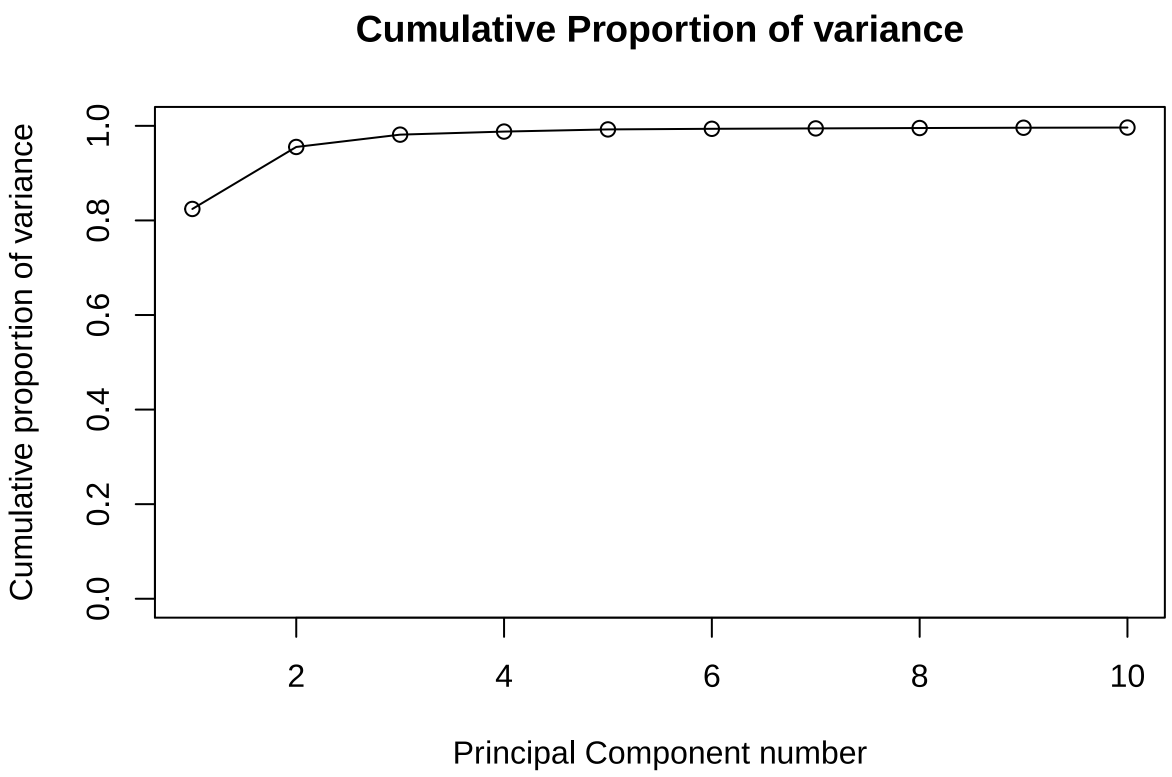
```r
plot(x[2,1:10],type = "o",ylim = c(0,1), ylab="Proportion of variance",
     xlab="Principal Component number", main="Proportion of variance")
```



```r
# look at plot of cumulative proportion of variance
x[3,1:10]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
## 0.82427 0.95537 0.98143 0.98787 0.99246 0.99375 0.99459 0.99538 0.99601 0.99642
```

```r
plot(x[3,1:10],type = "o",ylim = c(0,1), ylab="Cumulative proportion of variance",
     xlab="Principal Component number", main="Cumulative Proportion of variance")
```

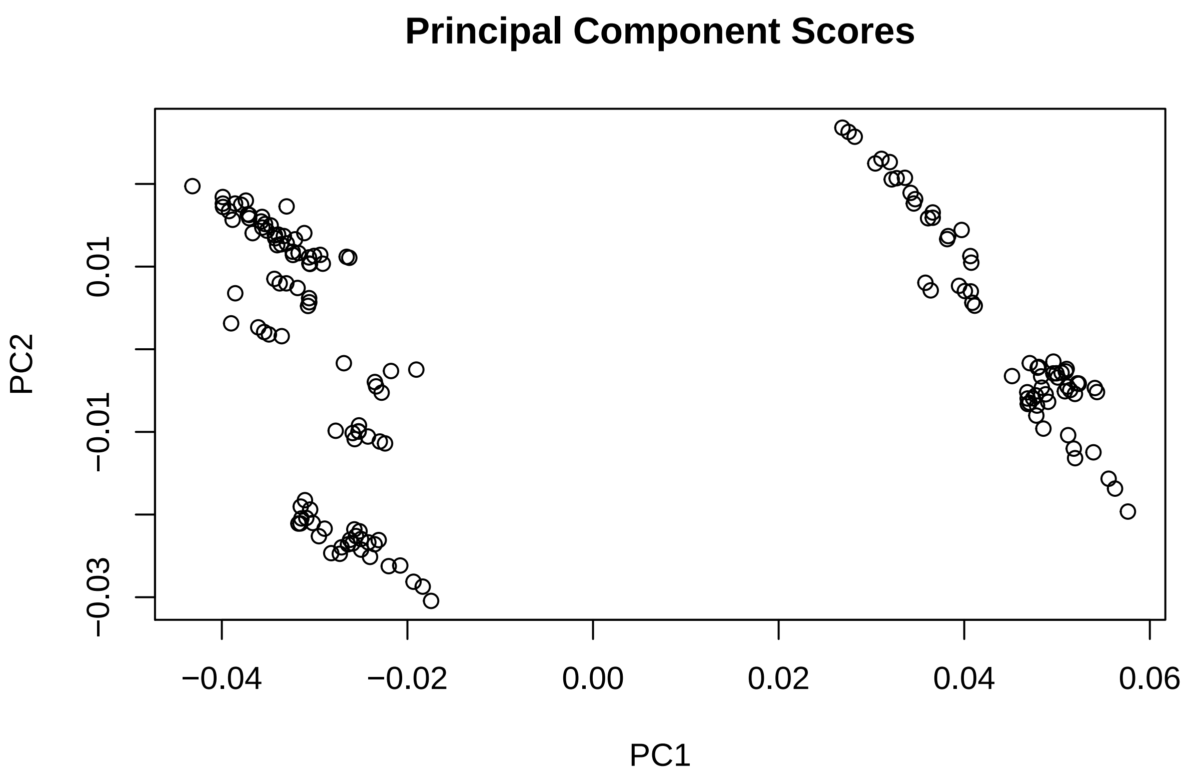


From these plots we can see that the majority of the variance is accounted for by the first two PC's and any PC's after this account for a very small proportion of variance so we decide to proceed with two PC's. In general, we look for the elbow in the scree plot to determine how many principal components to use.

Question 4

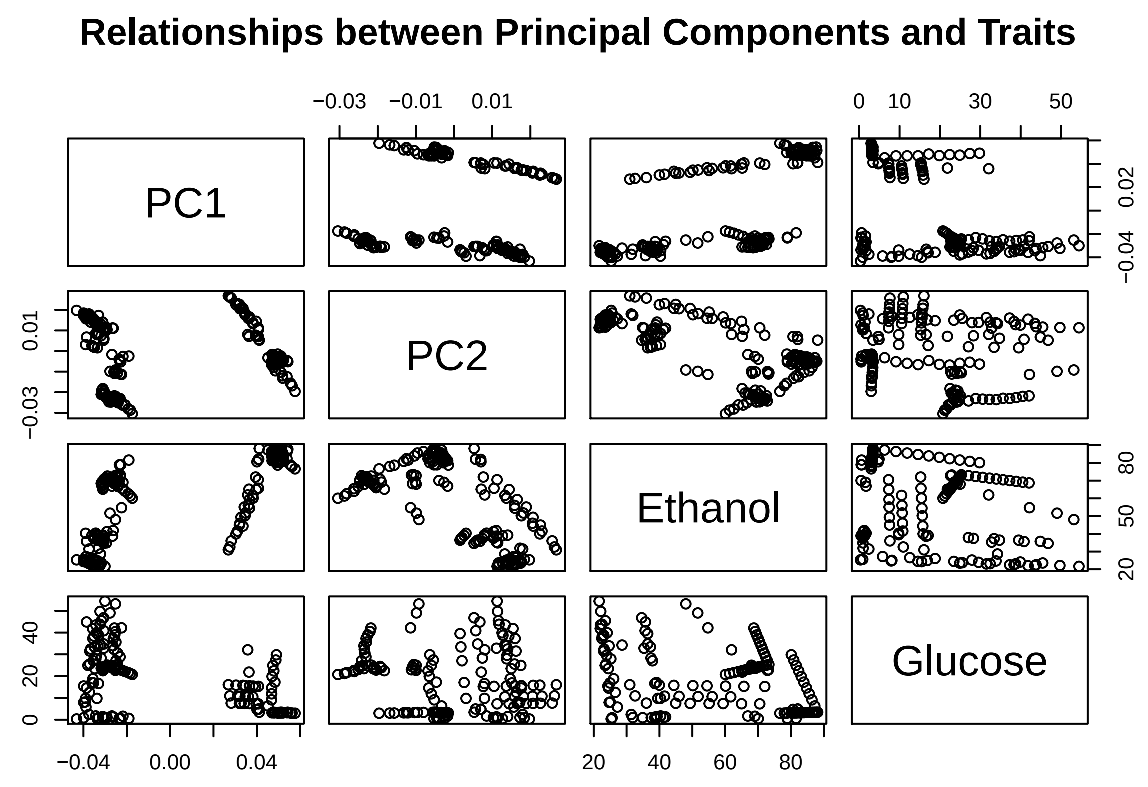
Next we want to plot the principal component scores for each grain mash.

```r
# We want to calculate the scores
scores = predict(fit)
# plot the scores
plot(scores[,1:2],main="Principal Component Scores")
```



It is clear the PC1 has two distinct groups where one group takes high values and the other takes low values while PC2 takes a more varied range of values. We also want to see if any structure that we observe is related to the Glucose and/or Ethanol traits. To see this we look at pairs plot.

```r
# create a dataframe with variables of interest to make it easier to plot
library(tidyverse)
df <- data.frame(scores[,1:2],data$Ethanol,data$Glucose)
dat = df %>% rename(Ethanol=data.Ethanol, Glucose=data.Glucose) # rename some of the columns
pairs(dat,main="Relationships between Principal Components and Traits")
```

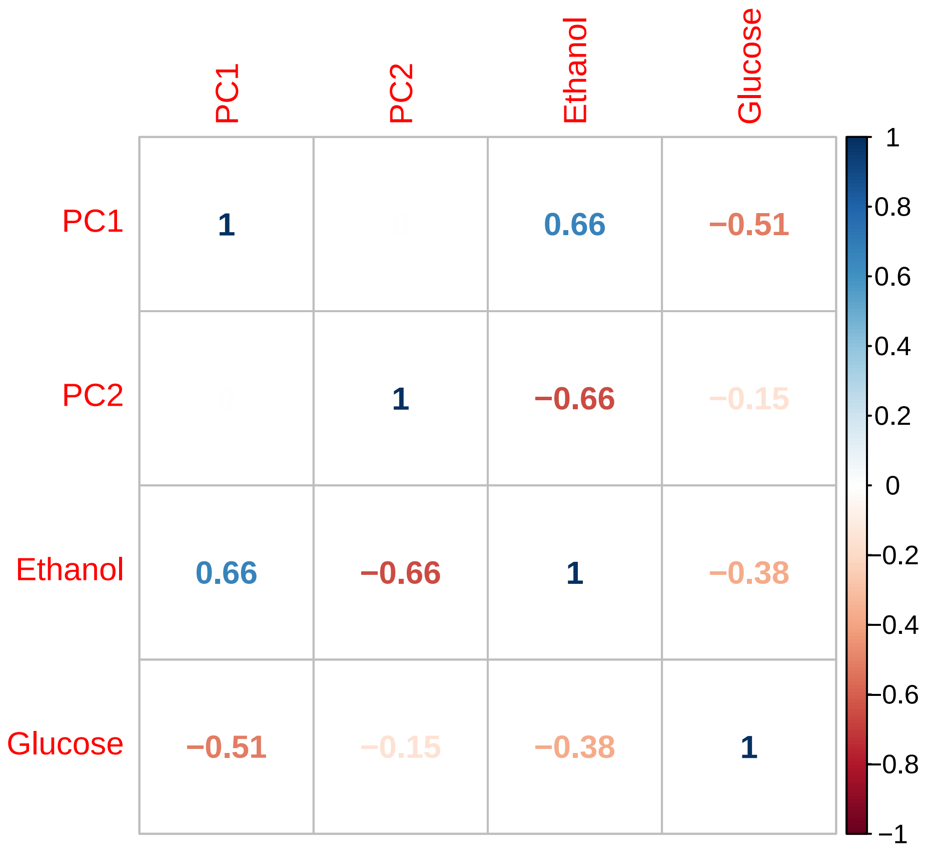


It is difficult to tell if any observed structure is related to the ethanol/glucose traits so to learn a bit more about the relationship with the two traits we can look at a correlation plot.

```
library(corrplot)
cor(dat)
```

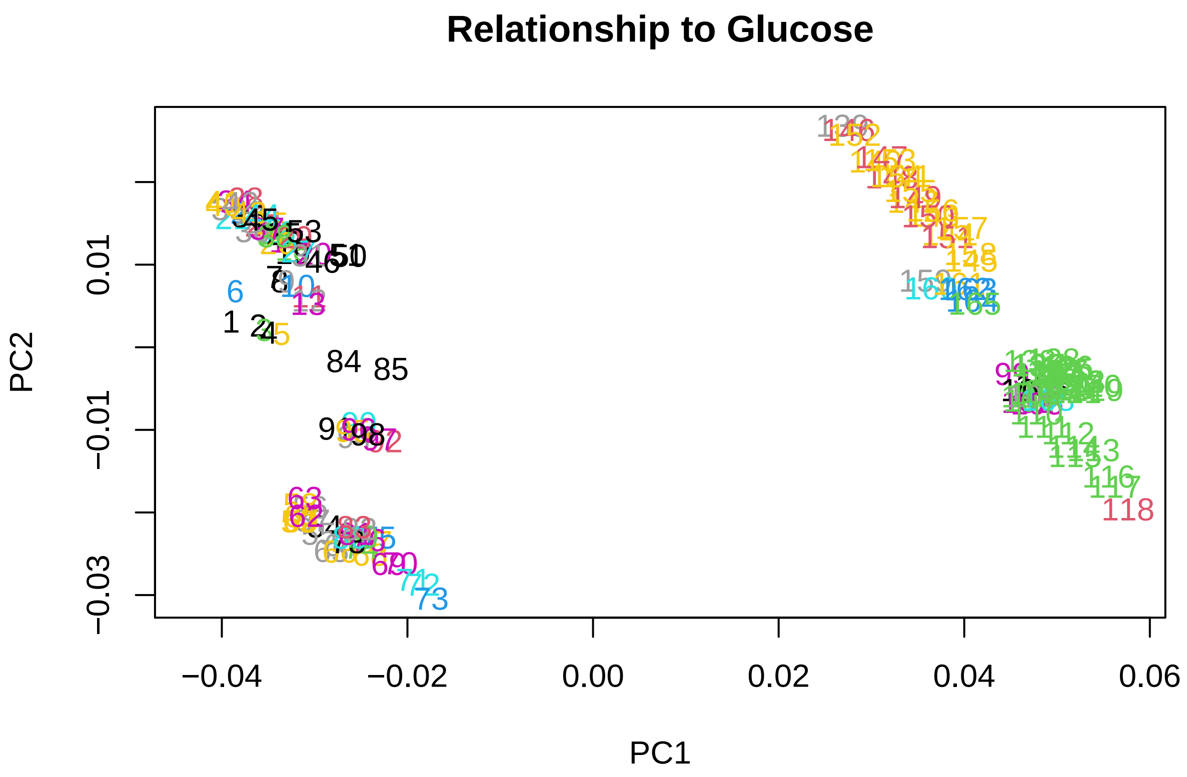
```
##                   PC1           PC2    Ethanol    Glucose
## PC1      1.000000e+00  4.494384e-15  0.6622067 -0.5111875
## PC2      4.494384e-15  1.000000e+00 -0.6588828 -0.1507668
## Ethanol  6.622067e-01 -6.588828e-01  1.0000000 -0.3799647
## Glucose -5.111875e-01 -1.507668e-01 -0.3799647  1.0000000
```

```
corrplot(cor(dat),method="number")
```



This tells us that Ethanol has quite a strong positive correlation with PC1 and strong negative correlation with PC2. It aslo tells us that Glucose has quite a strong negative correlation with PC1 but a weaker correlation with PC2. To try to understand the relationships a bit more we can colour the observations by their ethanol/glucose values. We begin by looking at Glucose values.

```
# try to colour the observations based on values of glucose
plot(scores[,1], scores[,2], type="n", xlab="PC1", ylab="PC2",main="Relationship to Glucose")
text(scores[,1], scores[,2], labels=1:165, col=as.integer(data[,237]))
```



We learn that the there is a relationship between Glucose value and the PC's as there are some clear groupings in the plot. In particular the green and yellow groups on the right hand side of the plot show that an observation with a glucose value equal to the glucose value of observations in those groups will be more likely to have a higher value for PC1. The plot also tells us that observations in the yellow group have a higher value for PC2 than observations in the green group. We see that observation 146 is contained in the yellow group so the glucose value of observations in that group is around

```
data[146,237]
```

```
## [1] 10.925
```

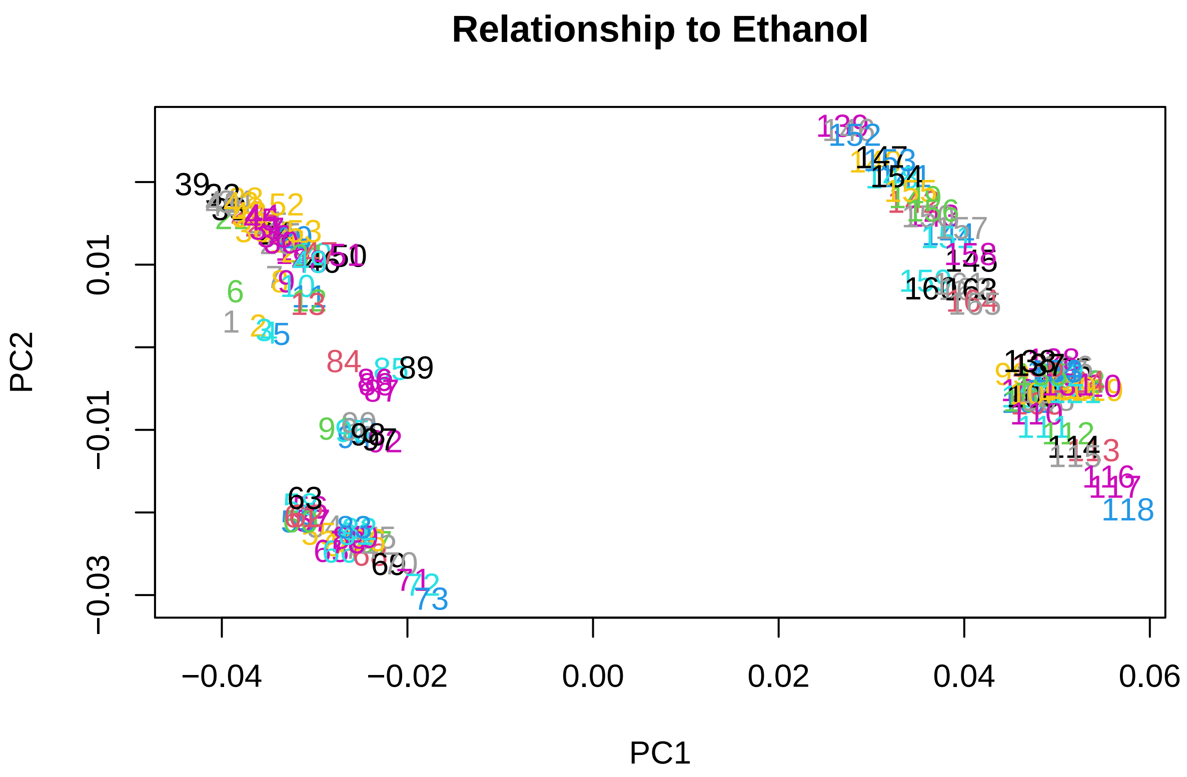
Similarly, observation 116 is contained in the green group so the glucose value of observations in that group is around

```
data[116,237]
```

```
## [1] 3.05
```

We can take the same approach for Ethanol values.

```
plot(scores[,1], scores[,2], type="n", xlab="PC1", ylab="PC2",main="Relationship to Ethanol")
text(scores[,1], scores[,2], labels=1:165, col=as.integer(data[,238]))
```

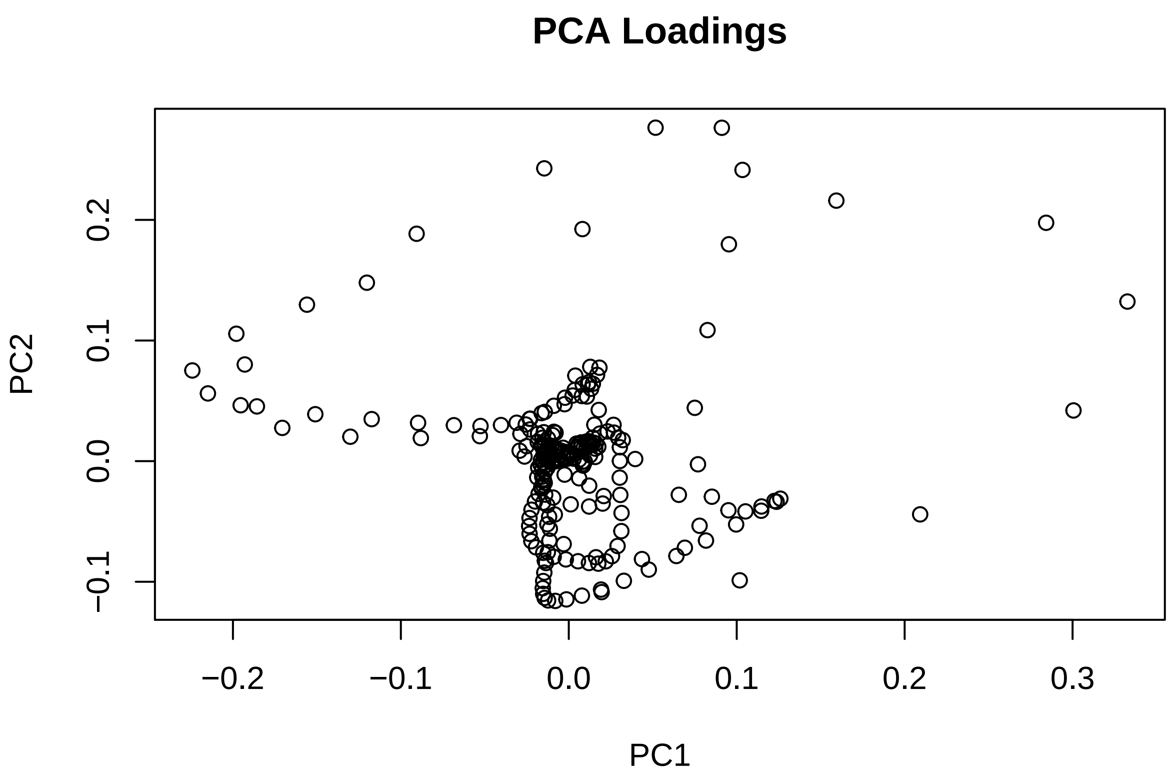


In this case, we do not see any clear groupings in the plot so we can't infer any further relationship between the PC's and Ethanol.

Question 5

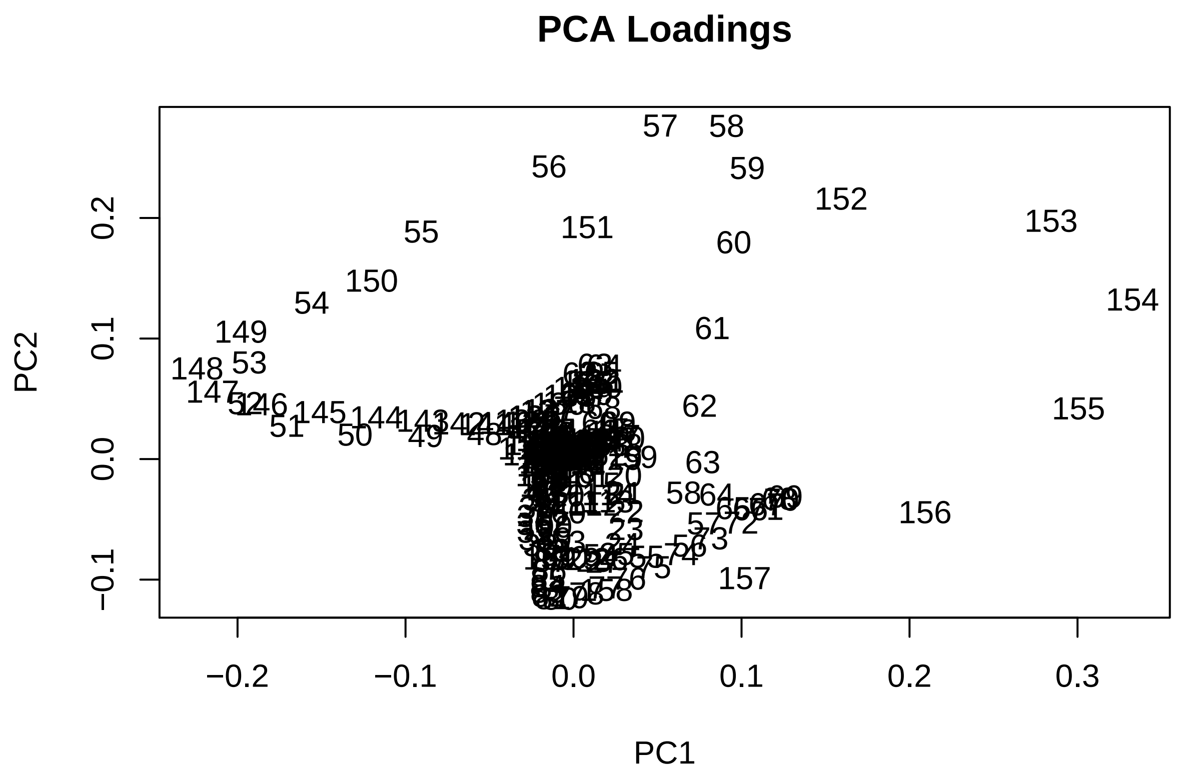
We want to look at the loadings for the two principal components we found earlier.

```
# want to extract loadings
loadings = fit$rotation
# plot the loadings for each of the three PC's
plot(loadings[,1:2],main="PCA Loadings")
```



To learn more about the loadings we can plot the variable number of each loading.

```
# Now want to label loadings by variable number to see the effect of certain variables.
# Compare loadings for PC1 and PC2
plot(loadings[,1:2],main="PCA Loadings",type="n")
text(loadings[,1:2], labels=1:165)
```



Loadings are interpreted as the coefficients of the linear combination of the initial variables from which the principal components are constructed. When interpreting loadings we must consider both sign and magnitude. For PC1 the majority of variables have zero loading so those variables have little affect on PC1. Some variables have quite high loading values for PC1 such as variables 153-156 and some variables have quite low loading values for PC1 such as variables 147,148,149,51,52 and 53. So the variables listed will have the biggest affect on the value of PC1. For PC2 there are more non-zero loadings than PC1 although the majority are still between -0.1 and 0.1. The variables which will have the biggest affect on PC2 are variables 56,57,58 and 59 as these have the greatest magnitude of loading.

Question 6

Principal component regression is a regression analysis technique used for computing regressions when the explanatory variables are highly correlated or even collinear. It uses principal component analysis to find principal components which are uncorrelated and represent the majority of the variation in the data. It then uses these principal components as the explanatory variables in a linear regression model that is fit using least squares. This is so useful because often a small number of PC's are enough to explain most of the variability in the data. Least squares estimates are unbiased when multicollinearity occurs but variances are large. Obviously high variance in estimates is not ideal so principal component regression can provide a way to combat this by adding bias to the regression estimates which reduces the standard errors and give more reliable results.

By using principal components as the explanatory variables, it is hoped that only a small number of principal components will be needed to explain most of the variability within the data. The choice of the number of principal components to be used is an essential choice in principal component regression. In PCA, a scree plot is used to determine the optimal number of principal components but in principal component regression it is possible to use cross validation to determine the optimal number of principal components. Another important choice to be made when performing principal component regression is whether the data should be standardized or not.The data should be standardized unless all variables are measured on the same scale. This is because principal component regression uses PCA which is heavily influenced by standardizing.

In multiple linear regression we have an NxP matrix X where N is number of observations and P is number of variables, an Nx1 vector Y and our model is $Y = XB$. This model struggles when the columns in X are strongly correlated. In PCR we replace the NxP matrix with a smaller NxQ matrix of scores that summarizes the original matrix X where Q is the number of principal components. To estimate the model we regress Y on the first q columns of the scores where q is the choice for number of principal components i.e. find coeffcients $B = L(T^\top T)^{-1}T^\top Y$ where T are scores and L are loadings. This model can then be used to predict future observations. In PCR we make an assumption that the directions in which the original variables show the most variation are the directions which are associated with the response variable. If this assumption holds then fitting a least squares model to the PC's will give better results than fitting a least squares model to the original explanatory variables because we mitigate overfitting by fitting less coefficients while most of the information relating to the response is still contained in the PC's.

Principal component regression is widely used as it has a number of key advantages. As we have already seen a major advantage is that PCR can perform regression when the explanatory variables are highly correlated or even collinear. On top of this, by replacing the original variables with fewer principal components it means you can run PCR when there are more variables than observations. Another advantage is that PCR is intuitive: you replace the original variables with principal components, drop the components that do not explain much variance, and regress the response onto the remaining components. The principal components that are dropped give insight into which linear combinations of variables are responsible for the collinearities.

However, the method does have a number of disadvantages. Since PCR uses PCA to find the principal components and then use these as explanatory variables, its results will be heavily influenced by whether or not the data has been standardized in the same way that PCA is heavily affected by standardizing. Also, after implementing PCA on the data the original variables are replaced by principal components and these principal components are not as readable and interpretable as the original variables. Another disadvantage is that it can be difficult to select the number of principal components so it is possible that information may be missed if the incorrect number of principal components is selected.

Question 7

To perform the prediction of glucose levels for a test set we must first split the data into a training and testing set. Two-thirds of the data is randomly assigned to the training set and the rest of the data is assigned to the test set. This is performed with the following code

```
library(pls)
# we want NIR data and also glucose trait
gluc = data[,2:237]
# we want to split data into training and test set where test set is 1/3 of data
N <- nrow(gluc)
# we want to train on 2/3's of data
keep <- sample(1:N, size=2/3*N)
# and test on 1/3 of the data
test <- setdiff(1:N, keep)
training <- gluc[keep,]
testing <- gluc[test,]
```

Then we want to train the PCR model on the training data. When training the model we have to make decisions regarding the type of cross-validation used and also the number of components we want in the model. We use LOO cross validation as it provides a reliable and unbiased estimate of model performance. Then we specify the number of components as 10 as it is likely the optimal number of components will be smaller than this. This is because we will be performing PCA on the spectral data when performing PCR and in earlier questions we saw that the spectral data only requires a small number of principal components.

```
gluc1 <- pcr(Glucose ~., ncomp = 10, data = training, validation = "LOO")
```

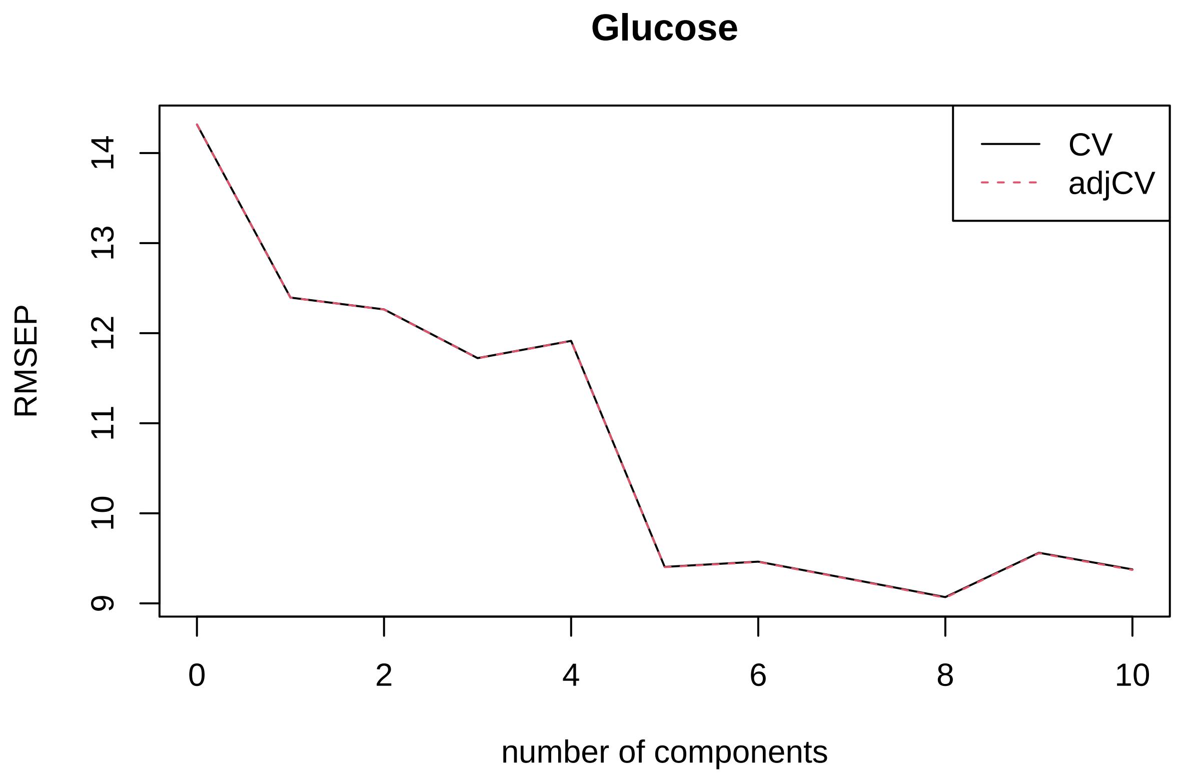
Then once the model has been fit we can get an overview of the fit and validation results with the summary method

```
summary(gluc1)
```

```
## Data:    X dimension: 110 235
##  Y dimension: 110 1
## Fit method: svdpc
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 110 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           14.32    12.40    12.26    11.72    11.91    9.405    9.463
## adjCV        14.32    12.39    12.26    11.72    11.91    9.402    9.460
##        7 comps  8 comps  9 comps  10 comps
## CV       9.267    9.070    9.562     9.377
## adjCV    9.264    9.063    9.557     9.372
##
## TRAINING: % variance explained
##          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X          81.60    95.21    98.17    98.82    99.22    99.37    99.46    99.54
## Glucose    26.13    28.67    36.02    36.31    61.60    62.31    63.58    64.94
##          9 comps  10 comps
## X          99.60     99.65
## Glucose    65.15     66.69
```

To decide how many PC's to proceed with we can look at the results from cross validation for varying numbers of PC's. The cross validation returns a value for root mean squared error so by plotting the root mean squared error we can see which number of PC's is optimal.

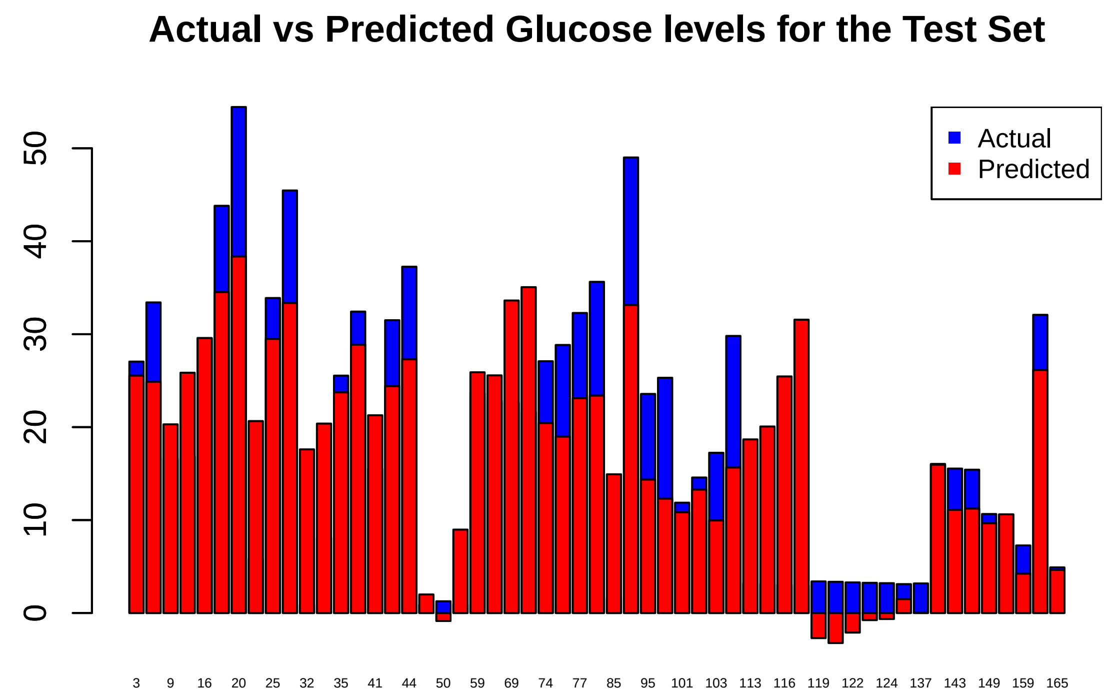
```r
par(mfrow=c(1,1))
plot(RMSEP(gluc1), legendpos = "topright")
```



This plot tells us that five PC's is optimal. Once the number of components has been chosen we can predict the glucose levels for a test set of the NIR data provided and look at the output as follows:

```r
pred.gluc = predict(gluc1, ncomp = 5, newdata = testing)

# change it to a vector for plotting
predtest = vector(length = 55)      # Define an empty vector
for (i in 1:55){
  predtest[i] = pred.gluc[i,1,1]    # Store the predicted value to the vector
}
# visualize output by comparing to actual values
# Now create a barplot of the actual values
barplot(testing[,236], col='blue', main = "Actual vs Predicted Glucose levels for the Test Set",
        names.arg = rownames(testing), cex.names=0.4)
# Add a barplot of the predicted values to the prior plot
barplot(predtest, col='red', add=T)
# Create the legend
legend("topright", legend=(c("Actual","Predicted")), pch=c(15,15), cex = 0.8,
       col=c("blue","red"))
```



This plot shows that the predicted values are not completely accurate. This could be due to a trade-off that we make when using principal component regression. We may decide it is acceptable to sacrifice a bit of accuracy for the ability to reduce the dimensions of the data and make computation less heavy. We can understand the error in the model a bit more by looking at the root mean square error

```
# Because we know the true response values for these samples, we can calculate the test set RMSEP
RMSEP(gluc1, newdata = testing)
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 comps
##      14.179       12.142       12.069       11.684       11.544        9.826
##     6 comps      7 comps      8 comps      9 comps     10 comps
##       9.670        9.247        9.466        9.324        8.922
```

For five components we get 9.826, which is quite close to the cross-validated estimate of 9.405 above.

Question 8

We want to predict the glucose levels for a test set of the NIR data provided using principal component regression. To do this we train a model on the training set and then apply it to the test set. To perform principal component regression we perform PCA on the predictor variables of the training set to generate principal components which we can then use as explanatory variables in a least squares regression.Once we have found the principal components we can calculate the scores for these principal components. In question 7 we saw that five principal components account for the majority of the variation in the data so we proceed with the first five columns of scores and use these to fit the least squares model.We then perform LOO cross validation to test the robustness of the model. Now it is possible to use this model to perform predictions. The scores of the test set can be calculated and these are multiplied by the coefficients of the model. it is important to remeber to add the intercept of the model as well. The resulting values from these calculations are the predictions. I wrote the following function to carry out this analysis.

```
# we can take the training and testing sets from Q7 and also test and keep.

# define function to be equivalent to pcr function( might take a coupe of seconds to
# run because of cross validation)
func <- function(data){
  # get predictor variables for training set
  preds_train = data[keep,1:235]
  # get predictor variables for testing set
  preds_test = data[test,1:235]
  # perform PCA on predictor variables for training set
  fit.pcr = prcomp(preds_train)
  # We want to calculate the scores
  scores = predict(fit.pcr)
  # create linear model with Glucose as response and first five columns of scores as predictors
  model = lm(training[,236] ~ scores[,1] + scores[,2] + scores[,3] + scores[,4] + scores[,5])

  # This does LOO CV
  # PC will store output of cross validation for each PC
  PC <- vector(mode="logical",length=5)
  # run loop for each PC
  for (i in 1:5){
    N = nrow(training)
    err <- vector(mode="logical",length=N)
    # run loop for each observation
    for (r in 1:N){
      # LOO so want to predict one value for each iteration(this value is r)
      test.cv <- r
      # training data is the remaining data
      train.cv <- setdiff(1:N, test.cv)
      # get response variable for new training set
      resp <- training[train.cv,236]
      # get predictor variables for cross validation training and testing set
      preds_train.cv = training[train.cv,1:235]
      preds_test.cv = training[test.cv,1:235]
      # perform pca on training set
      fit.pcr.cv = prcomp(preds_train.cv)
      # get scores
      scores.cv = predict(fit.pcr.cv)
      # fit model
      model.cv = lm(training[train.cv,236] ~ scores.cv[,1] + scores.cv[,2] + scores.cv[,3]
                    + scores.cv[,4] + scores.cv[,5])
      # only want coefficients that correspond to number of PC's for this iteration of loop
      ind= i+1
      coefficients.cv = as.matrix(model.cv$coefficients[2:ind])
      test.pca.cv = predict(fit.pcr.cv, preds_test.cv)
      # remember to add back intercept from model
      prediction.cv = (test.pca.cv[,1:i] %*% coefficients.cv)+model.cv$coefficients[1]

      # find RMSE
      err[r] <- sqrt(mean((as.matrix(prediction.cv)-training[test.cv,236])^2))
    }
    # get average RMSE
    PC[i] = mean(err)
  }

  # do prediction
  # extract coefficients from model
  coefficients = as.matrix(model$coefficients[2:6])
  # get scores for test set
  test.pca = predict(fit.pcr, preds_test)
  # multiply coefficients times scores and add back intercept of model to get predictions
  preds_final = (test.pca[,1:5] %*% coefficients)+model$coefficients[1]

  # check how results compare
  # get mse
  mse = mean((as.matrix(preds_final)-as.matrix(pred.gluc))^2)
  # get rmse
  rmse = sqrt(mean((as.matrix(preds_final)-as.matrix(pred.gluc))^2))

  # define outputs of function
  objects <- list("predictions"=preds_final, "MSE"=mse, "RMSE"=rmse,
                  "validation for different numbers of PC's"=PC)
  return(objects)
}
```

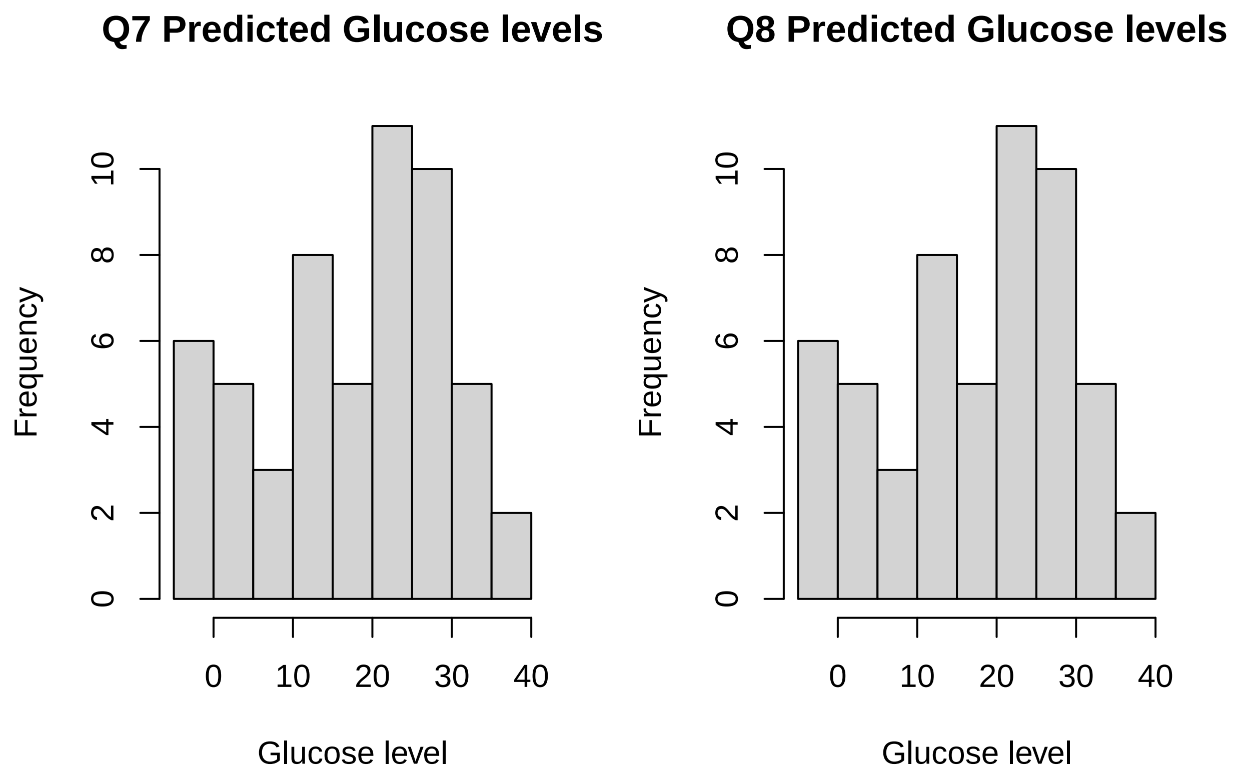
Now that the function has been created we can run the function on our data and look at the results.

```
# run this function to train model
myresults = func(data=gluc)
myresults
```

```
## $predictions
##             [,1]
## 3   25.54673073
## 4   24.88786111
## 9   20.29868262
## 14  25.85295377
## 16  29.58802741
## 18  34.52275271
## 20  38.35889073
## 21  20.64937192
## 25  29.49515050
## 27  33.35947432
## 32  17.60568567
## 33  20.38140240
## 35  23.73627014
## 36  28.87175503
## 41  21.27217334
## 43  24.40959670
## 44  27.29191662
## 48   1.99448216
## 50  -0.85516999
## 52   8.97458607
## 59  25.90025616
## 62  25.56597689
## 69  33.62165864
## 71  35.05718833
## 74  20.43989959
## 75  18.97803921
## 77  23.10799457
## 79  23.39645104
## 85  14.92367065
## 91  33.13836074
## 95  14.35694694
## 98  12.30885295
## 101 10.83248303
## 102 13.27814767
## 103  9.95706532
## 108 15.65383508
## 113 18.67921285
## 114 20.06518943
## 116 25.44956369
## 118 31.56057049
## 119 -2.69928544
## 120 -3.22944973
## 122 -2.10306116
## 123 -0.76207244
## 124 -0.64460539
## 127  1.49741201
## 137  0.02961778
## 139 15.95870543
## 143 11.10741115
## 144 11.23786442
## 149  9.65974874
## 150 10.61745980
## 159  4.23359594
## 160 26.15227747
## 165  4.64740613
##
## $MSE
## [1] 8.068641e-27
##
## $RMSE
## [1] 8.982562e-14
##
## $`validation for different numbers of PC's`
## [1] 9.183356 9.429525 9.216702 9.338440 7.289976
```

The value for mean square error and root mean square error are both pretty much zero which tells us that the two sets of predictions are nearly identical. To visualize the agreement between the two sets of predictions we can look at the following histograms.

```r
# plot results
par(mfrow=c(1,2))
hist(pred.gluc,xlab="Glucose level", main="Q7 Predicted Glucose levels")
hist(myresults$predictions,xlab="Glucose level", main="Q8 Predicted Glucose levels")
```



It is clear that the two sets of predictions are nearly identical so I have successfully re-created the code in question 7 .