Multivariate Analysis

2023-03-05

1. Load the data set into R. Use the set.seed function in R to set the seed to your student number. Randomly generate a number between 1 and n (where n is the number of rows in the dataset), and delete that observation/row from the dataset. Ensure that you include the code used in this step in the R code you submit with your assignment so that your work can be reproduced.

Loading packages

```
library(dplyr)
library(tidyverse)
library(ggplot2)
library(ggcorrplot)
library(GGally)
library(egg)
library(class) # for knn()
library(MASS) # for lda() and qda()
library(sparcl) # for ColorDendrogram() for coluring dendogram dendogram
library(cluster) # for silhouette()
library(e1071) # for ClassAgreement
library(mclust)
library(pls)
library(caret)
options(warn = -1) # to ignore the warnings generated during pairs plot using ggplot
```

Loading the DataSet

```
# Load the dataset
df <- read.csv("Milk_MIR_Traits_data_2023.csv")
dim(df)

## [1] 431 582
```

```
# Set the seed to your student number
set.seed(22200315) # replace 1234 with your student number

# Generate a random number between 1 and n and delete the corresponding observation/r
ow
n <- nrow(df)
random_row <- sample(n, 1)
df <- df[-random_row, ]
```

1.The MIR data contains 431 recorded observations of various traits of milk samples with a total of 582 columns.

2. The initial 51 columns consist information about the breed, date of sampling, protein and technological traits.

3.The last 531 columns of this data contains the MIR Spectra readings on 531 unique wave-lengths.

2. The milk protein beta Lactoglobulin B is used in the production of protein drinks. Remove from the dataset any record/observation which has a missing/NA value for Beta Lactoglobulin B. Then, visualise the spectra and the protein trait beta Lactoglobulin B using (separate) suitable plots. Comment on the plots. Remove any observations with beta Lactoglobulin B outside of 3 standard deviations from the mean of the trait.

```
#Remove rows with any missing value or Beta Lactoglobulin B
df <- df %>% drop_na(beta_lactoglobulin_b)
```

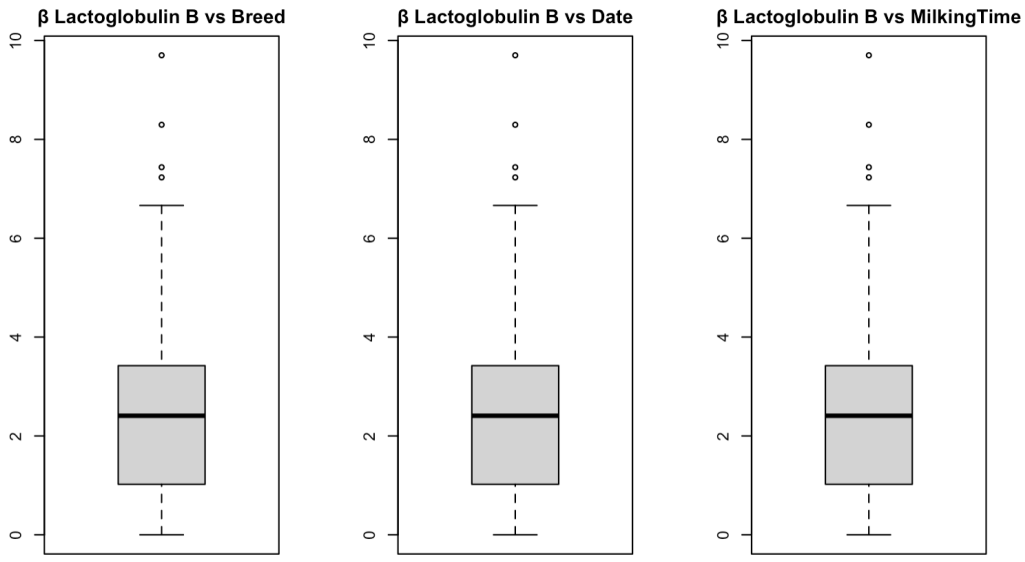
Analysing the protein and Beta Lactoglobulin B trait of the data set

```
sum(is.na(df$beta_lactoglobulin_b))

## [1] 0

#Removing any missing values for Beta Lactoglobulin B from the dataset


df1 <- df %>% drop_na(c("beta_lactoglobulin_b"))
dim(df1)
```

```
## [1] 306 582
```

There were 124 missing values for Beta Lactoglobulin B in the data set. Removing the missing values to get a dataset of dimension 306 x 582.

```
#Creating a data frame with beta lactoglobulin_b values
beta_lb <- data.frame(df[,13])

#Visualization of the milk protein Beta Lactoglobulin B using a Box Plot


par(mfrow = c(1,3))
boxplot(beta_lb, df1['Breed'], main = 'β Lactoglobulin B vs Breed')
boxplot(beta_lb, df1['Date_of_sampling'], main = 'β Lactoglobulin B vs Date')
boxplot(beta_lb, df1['Milking_Time'], main = 'β Lactoglobulin B vs MilkingTime')
```



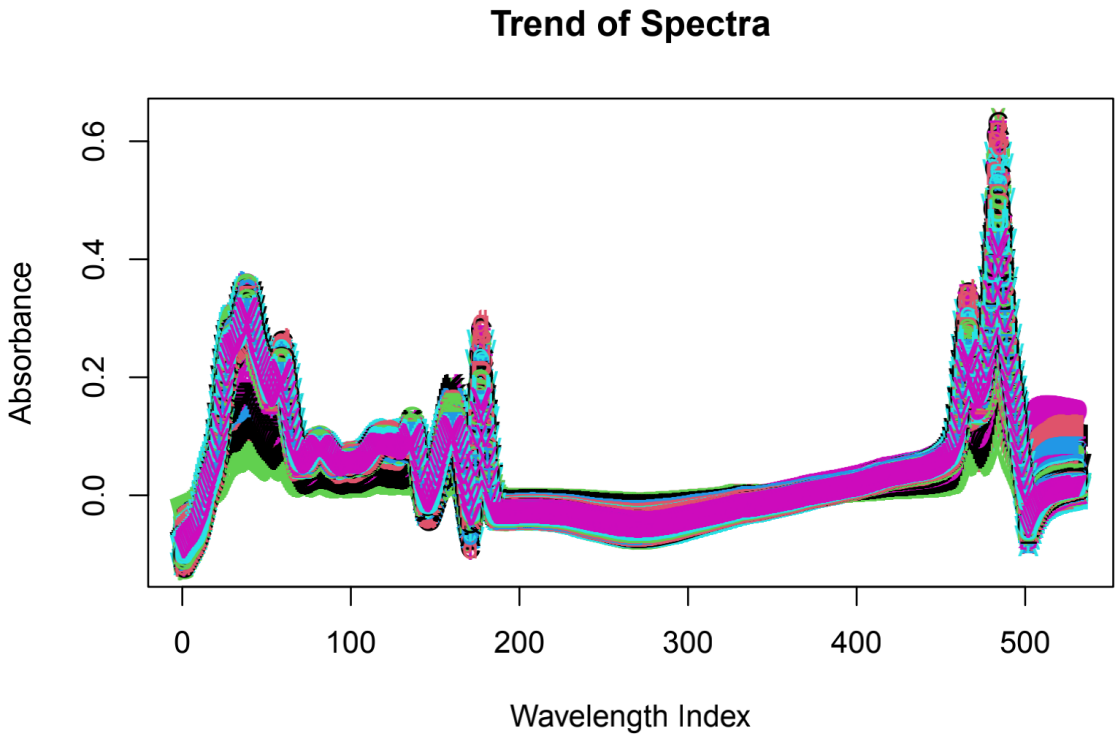

```
# Selecting Data pertaining to the Spectra
Spectra_Data <- df[,(ncol(df)-530):ncol(df)]
milk_spectra <- df1[,-c(1:51)]


# Checking for Missing values
paste("Number of Missing values:",sum(is.na(milk_spectra)))
```

```
## [1] "Number of Missing values: 0"
```

Visualizing the Spectra

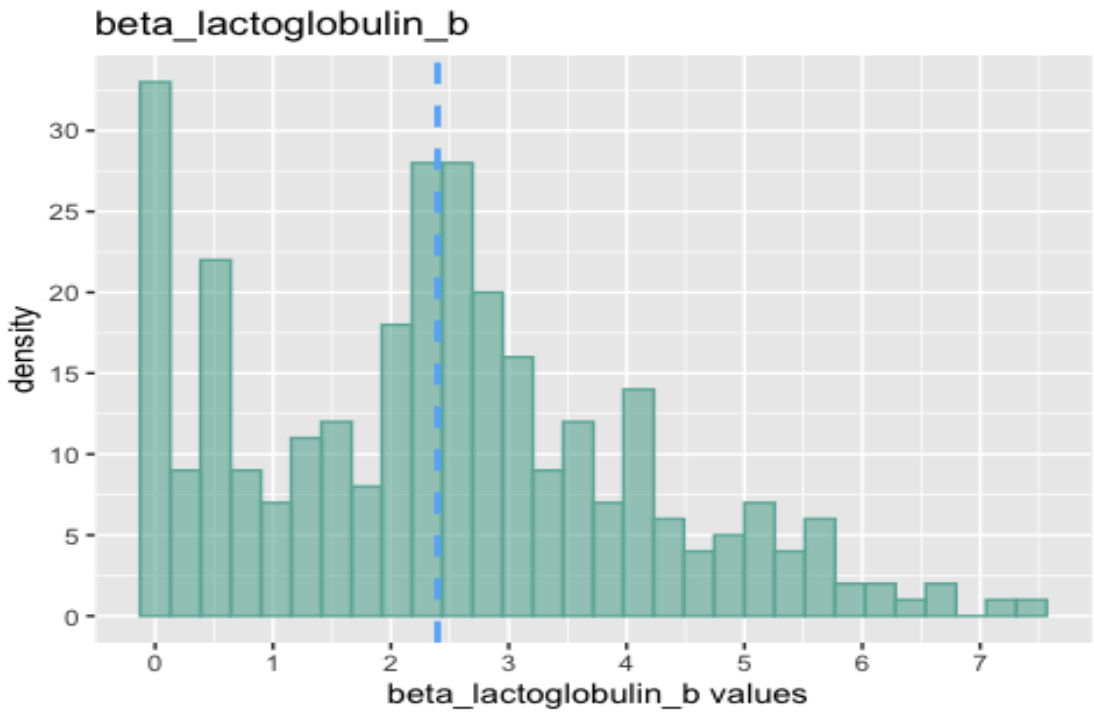
This graph represents the MIR_spectra for the 87 observation in our data set.

```
matplot(t(milk_spectra),xlab = "Wavelength Index",ylab = "Absorbance",
        main="Trend of Spectra", xlim= c(0,500), ylim = c(0,1))
```



The graph plots the absorbance values of this sample at each of the 531 recorded wavelengths. Each wavelength has the potential to correspond to the different traits of the sample such as the Heat stability,milk fat content, protein content ,processability etc, the absorbance values at these wavelengths indicate the strength of these traits. From the plot obtained above, we can see that the wavelengths are overlapping to a great extent, this indicates that there is high correlation among those wavelengths. We can also see that the maximum absorbance in the spectra is around 0.64 and the minimum absorbance is approximately −0.1.

```
# Remove observations with Beta Lactoglobulin B outside of 3 standard deviations from th
e mean of the trait

mean_beta <- mean(df1$beta_lactoglobulin_b)
sd_beta <- sd(df1$beta_lactoglobulin_b)
df1 <- df1 %>% filter(beta_lactoglobulin_b >= (mean_beta - 3*sd_beta) & beta_lactoglobul
in_b <= (mean_beta + 3*sd_beta))

summary(df1$beta_lactoglobulin_b)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.003   2.388   2.397   3.367   7.436

ggplot()+geom_histogram(aes(x=df1$beta_lactoglobulin_b),color="#69b3a7",
                        fill="#69b3a2",alpha=0.6)+
xlab(" beta_lactoglobulin_b values")+      ggtitle("beta_lactoglobulin_b")+scale_x_continuou
s(n.breaks = 10) +
scale_y_continuous(n.breaks = 10)+stat_bin(bins = 30)+
geom_density()+
  geom_vline(aes(xintercept=mean(df1$beta_lactoglobulin_b)),
             color="#69b3f9", linetype="dashed", size=1) #mean line
```



From the plot and summary statistics of data:

- Data is slightly right skewed.
- Average Beta_Lactoglobulin_B found in milk samples was around 2.397.
- The mean and median for this data are close with slight deviation.
- 2 unusual values (outliers) were recorded for this variable.
- The values range from 0 to 7.23.

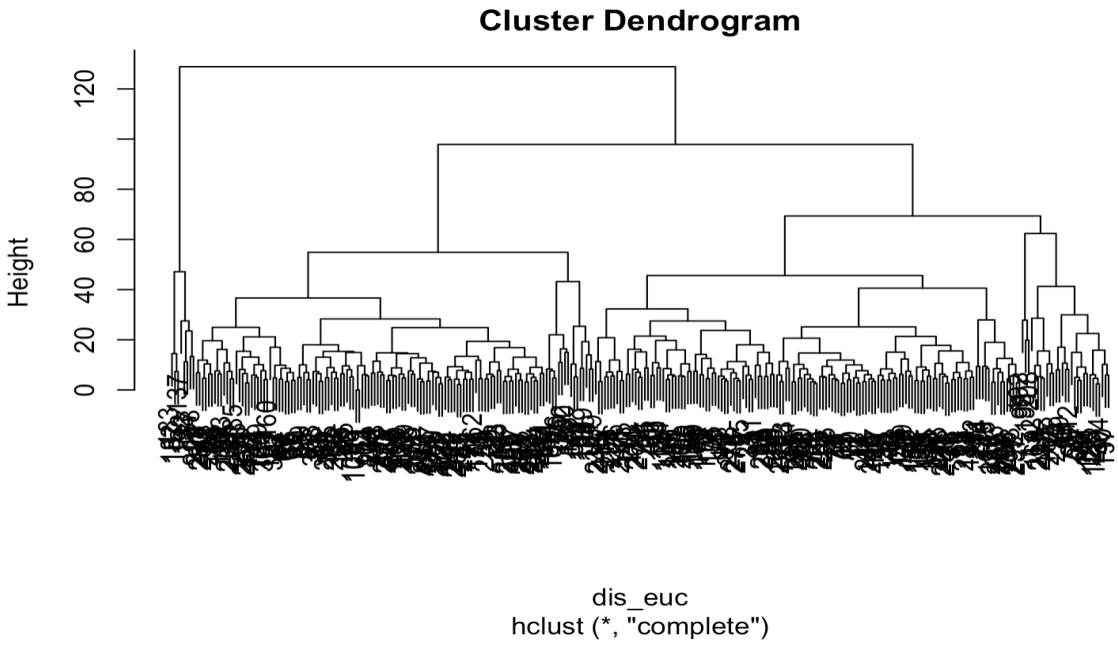
3. Use hierarchical clustering and k-means clustering to determine if there are clusters of similar MIR spectra in the data. Motivate any decisions you make. Compare the hierarchical clustering and k-means clustering solutions. Comment on/explore any clustering structure you uncover, considering the data generating context.

Hierarchical Clustering

```
milk_spectra <- scale(milk_spectra) #scaling data to bring the variable in same range
dim(milk_spectra)

## [1] 306 531
```

Visualizing clustering structures through hierarchical clustering technique

```
# creating dissimilarity matrix
dis_euc <- dist(milk_spectra, method = 'euclidean')
# clustering the milk spectra data using the complete linkage method
cl_comp <- hclust(dis_euc, method = 'complete')
plot(cl_comp)
# Cutting dendogram to create groups
hcl = cutree(cl_comp, k = 3)
table(hcl)
## hcl
##   1   2   3
## 168 130   8
```



Analysis of Hierarchical Clustering

1.To identify the distinct groups in a milk spectra data set, I utilized hierarchical clustering techniques. I used the Euclidean distance measure to determine the dissimilarity between two data points and complete linkage to calculate the dissimilarity between joined clusters. The results revealed the presence of two main groups in the MIR spectra data for milk

samples, with further subgroups detected within these groups, possibly related to different breeds of cows in the samples.

2.The first group was found to have two subgroups, while the second group had five subgroups. To further investigate these subgroups, I performed clustering on the original data without eliminating unusual observations previously identified in the protein and technological traits.

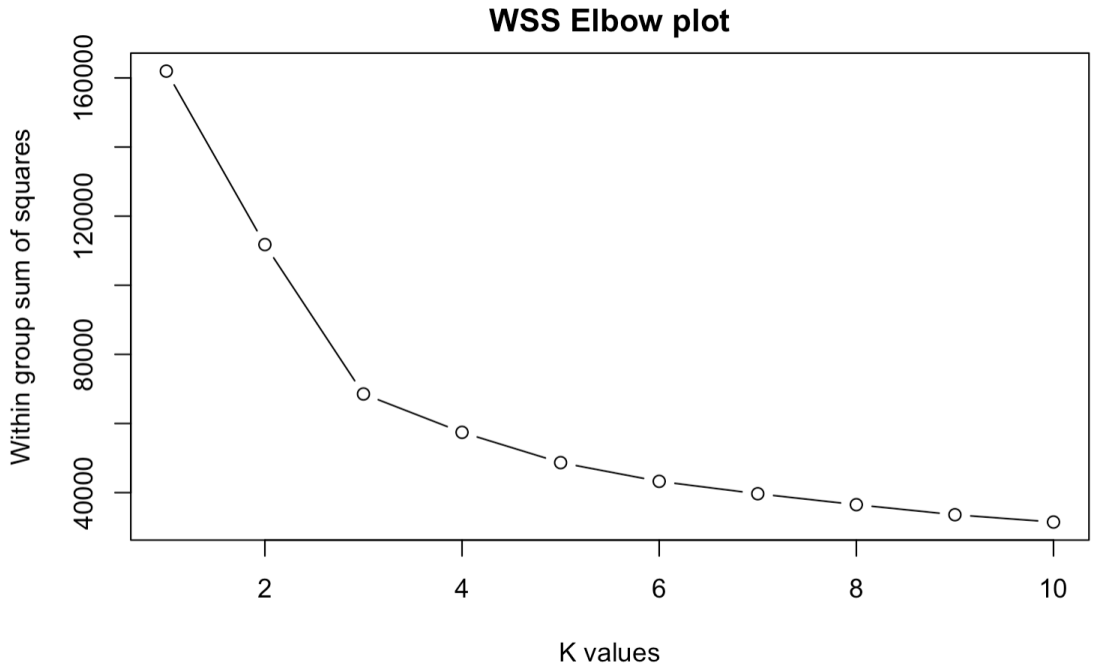
3.Based on these findings, I concluded that there are three significant groups in the data. I chose not to use single linkage due to the possibility of chaining issues, and both average and complete linkage produced similar results for the data.Checking this further through kmeans clustering algorithm:

Kmeans clustering

Now that I have a basic understanding of how the milk spectrum data is grouped, I can use the kmeans algorithm to investigate how tightly the clusters are formed within the dataset.

Visualizing clustering structures through K-means clustering technique

```
# Function to fit kmeans clustering model and computing within sum of squares for different values of K for optimal solution.


set.seed(22200315)
WGSS = rep(0,10)
n = nrow(milk_spectra)
WGSS[1] = (n-1) * sum(apply(milk_spectra, 2, var))
for(k in 2:10)
{
WGSS[k] = sum(kmeans(milk_spectra, centers = k,nstart = 50)$withinss)
}
plot(1:10, WGSS, type="b", xlab="K values",
     ylab="Within group sum of squares",
     main = "WSS Elbow plot")
```



From the table obtained above for K means clustering, we have divided the MIR Spectra in the data in three clusters using WGSS elbow method.

```
#The k versus WGSS plot suggests k = 3 is a good clustering solution.
#Hence:

k=3
set.seed(22200315)
# using nstart = 50 to avoid local optimum
Spectra_k = kmeans(milk_spectra, center=k, nstart = 50)
table(Spectra_k$cluster)

##
##   1   2   3
##   8 131 167
```

Analysis of K means Clustering

Our k-means clustering suggests that our data is divided into 3 clusters:

Cluster 1 : 8 observations

Cluster 2 : 131 observations

Cluster 3 : 167 observations

4. Apply principal components analysis to the spectral data, motivating any decisions you make in the process. Plot the cumulative proportion of the variance explained by the first 10 principal components. How many principal components do you think are required to represent the spectral data? Explain your answer.

PCA

First we must make the decision of whether we need to standardize the data or not.
If the variance related to the variables are different then we shall standardize the data.
Variables with larger variances will end up having more effect on the PCA.

Upon finding the variance we see there is a difference in the variance of the variables

```
sample(apply(Spectra_Data,2,var),10)
```

```
##        X2546         X976        X1797        X1466        X2245        X1469
## 7.032836e-06 1.327575e-04 2.661900e-05 1.604586e-04 5.175886e-05 1.361504e-0
4
##        X1230         X984        X1431        X3737
## 5.726451e-05 1.148230e-04 1.045229e-04 8.321901e-04
```

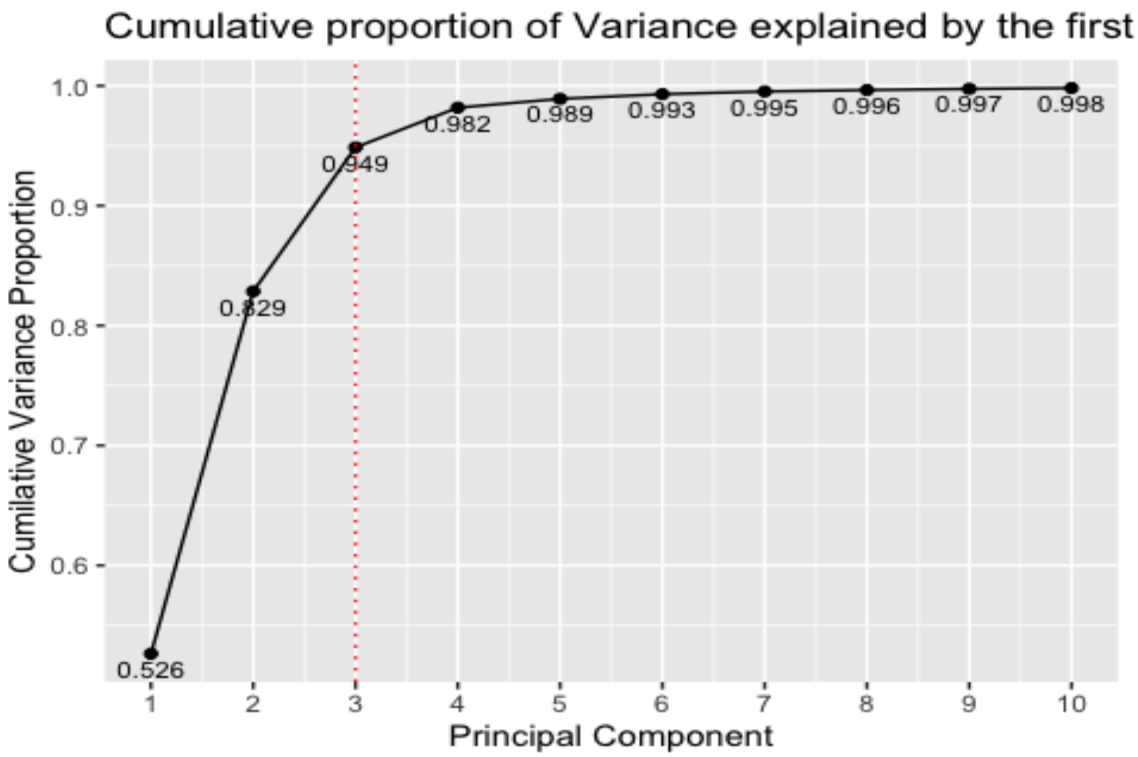
Although the variation values are extremely small in all variables, it is to be noted that there is a difference of 10 to 100 times. I consider this sufficient to standardize the data.

```
# using the scale() function to scale our data
Spectra_Data_std <- scale(Spectra_Data,center = T,scale = T)

Spectra_PCA <- prcomp(Spectra_Data_std,scale. = T,center = T)
PCA_sum <- summary(Spectra_PCA)


# Displaying the first 10 Principle components
PCA_10 <-as.data.frame(t(PCA_sum$importance[,1:10]))
PCA_10
```

```
##       Standard deviation Proportion of Variance Cumulative Proportion
## PC1          16.7181847                0.52636               0.52636
## PC2          12.6686666                0.30225               0.82861
## PC3           7.9829132                0.12001               0.94862
## PC4           4.1967406                0.03317               0.98179
## PC5           1.9732226                0.00733               0.98913
## PC6           1.4380204                0.00389               0.99302
## PC7           1.0935541                0.00225               0.99527
## PC8           0.7925339                0.00118               0.99646
## PC9           0.7284310                0.00100               0.99745
## PC10          0.5838457                0.00064               0.99810
```

Visualizing Cumulative Proportion plot

```
ggplot(data = PCA_10,aes(x=1:10,y= `Cumulative Proportion`,label = round(`Cumulative Proportion`,3)))+
  geom_point()+
  geom_line()+
  geom_text(hjust =0.5,vjust=1.4,size=3)+
  scale_x_continuous(n.breaks =10)+
  xlab("Principal Component")+
  ylab("Cumilative Variance Proportion")+
  ggtitle("Cumulative proportion of Variance explained by the first 10 Principal Components")+
  geom_vline(xintercept = 3,linetype="dotted", col='red')
```



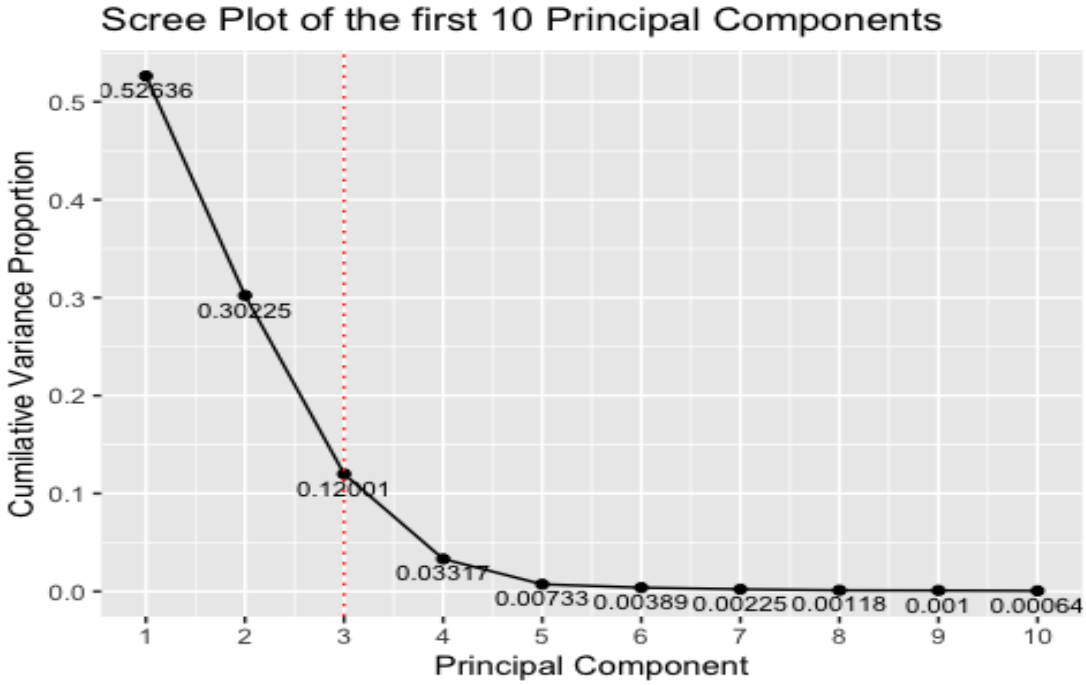
I use the following method to choose the number of Principal Components required to represent the data:

Scree Plot

A scree plot is a graphical representation of how much variance is accounted for by each principal component. We use this plot to identify the point at which subsequent components explain very little or practically none of the variance. Since principal components are arranged in order of increasing variance explained, we only plot the first 10 components. By the 10th principal component, the amount of variance explained is less than 0.006% of the total variance in the data. Therefore, exploring additional components is not worth the computational resources.

```
ggplot(data = PCA_10,aes(x=1:10,y= `Proportion of Variance`,label = `Proportion of Varianc
e`))+
  geom_point()+
  geom_line()+
  geom_text(hjust =0.5,vjust=1.4,size=3)+
  scale_x_continuous(n.breaks =10)+
```

```
xlab("Principal Component")+
ylab("Cumilative Variance Proportion")+
ggtitle("Scree Plot of the first 10 Principal Components")+
geom_vline(xintercept = 3,linetype="dotted", col="red")
```



After analyzing the data, we notice a drop odd in value at the third component, indicating that we should select the first three components. When conducting principal component analysis, we observe that by using the first four principal components, we can preserve 94.85% of the information or variation that exists in the data. Therefore, we will use the first three principal components to represent the data.

5. Derive the principal component scores for the milk samples from first principles (i.e., you should not use an inbuilt function such as predict(. . .)). Plot the principal component scores for the milk samples.Comment on any structure you observe.

```
S =cov(scale(Spectra_Data))
eig = eigen(S)
e_vec = eig$vectors
scores <- as.matrix(scale(Spectra_Data,center=T))%*%e_vec
scores[1:6,1:3]
```

```
##           [,1]       [,2]      [,3]
## [1,] -9.530658 -3.3815641 -1.316620
## [2,] -8.839395  0.6618523 -5.000652
## [3,] -3.809277  8.2118701 -2.526234
## [4,] -2.243791 -0.4064360  1.143293
## [5,] -1.619035 14.6118053 -5.606203
## [6,] -7.166971 -3.5751139 -1.822382
```

Looking at the principal component scores for the milk sample provided above, we can observe that there are certain patterns or structures present in the data.

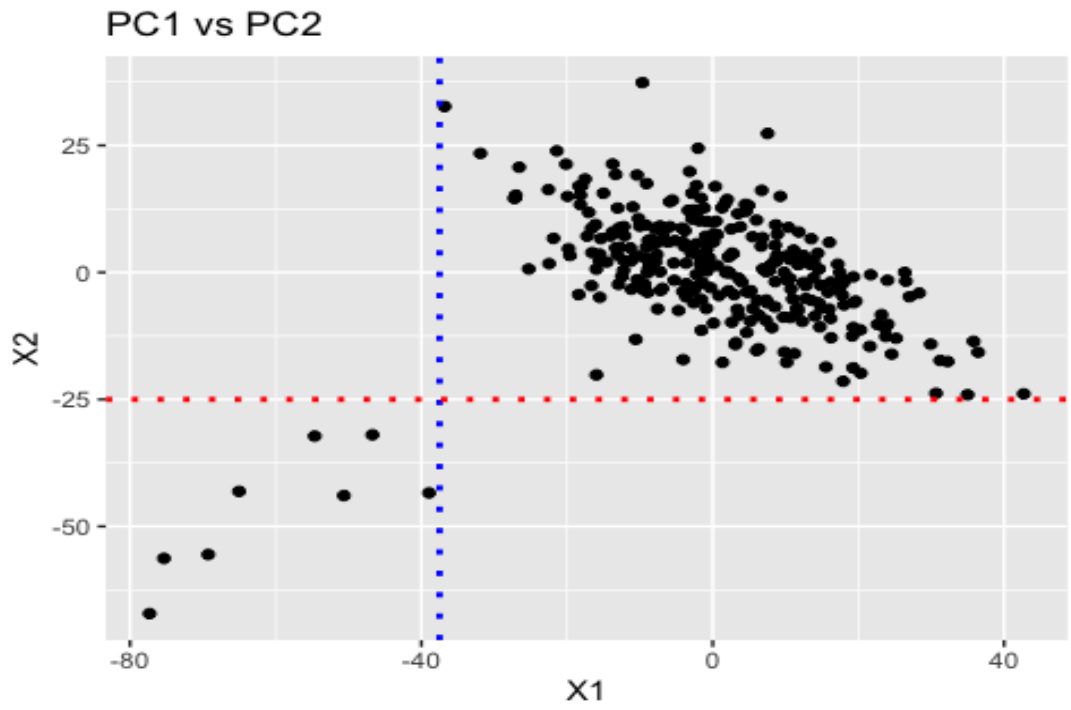
PC1 appears to have negative values for all the samples, with the first sample having the most negative value. This suggests that PC1 is capturing some common factor that is present in all the samples, but is particularly strong in the first sample.

PC2, on the other hand, has a wide range of values, both positive and negative, with the fifth sample having the highest positive value. This suggests that PC2 is capturing some factor that is present in some samples, but not others.

PC3 has a smaller range of values compared to PC2, and also have both positive and negative values. This suggests that these principal components are capturing more subtle patterns or structures in the data, which are present in some samples, but not others.

Overall, the patterns or structures present in the principal component scores indicate that there are underlying factors or variables that are affecting the composition of the milk samples in different ways, and that these factors can be captured and analyzed using principal component analysis.

```
ggplot(data = data.frame(scores),aes(x=X1,y=X2))+geom_point()+
  ggtitle("PC1 vs PC2")+
  geom_vline(xintercept = -37.5,linetype="dotted",col="blue",lwd=1)+
  geom_hline(yintercept = -25,linetype="dotted",col="red",lwd=1)
```



By plotting PC1 vs PC2 scores we can see a structure emerging. It seems as though the data gets divides into 2 clusters:
Cluster 1 with a PC1 score >=−37.5 and PC2 score >=−25 in the first quadrant. Cluster 2 with a PC1 score <=−37.5 and PC2 score <=−25 in the third quadrant.

6. In your own words, write a maximum 1 page synopsis of the PCR method. Your synopsis should (i) explain the method's purpose, (ii) provide a general description of how the method works, (iii) detail any choices that need to be made when using the method and (iv) outline the advantages and disadvantages of the method

The Principal Component Regression (PCR) is a widely used statistical method that first applies Principal Component Analysis on the data set to summarize the original predictor variables into few new variables also known as principal components (PCs), which are a linear combination of the original data to build the linear regression model. With high−dimensional data, where the number of predictor variables (p) is significantly greater than the number of samples, the primary goal of PCR is to address the problem of overfitting. (n).

There are several steps involved in the general procedure for performing PCR.Once the n principal components are obtained, a linear regression model is fitted using these

components as predictors. The regression coefficients are estimated using ordinary least squares (OLS) or partial least squares (PLS) regression, depending on the choice of the algorithm. The resulting model can then be used to predict the outcome variable for new samples.

As a result of the growth in very large datasets in areas such as image analysis or Web data analysis, significant methodological advances in data analysis, which frequently have their origins in PCA, have been made. One of the key advantages of PCR is its ability to handle high-dimensional data with relatively few samples, which is commonly encountered in a variety of scientific domains such as biology, chemistry, and finance. Furthermore, PCR can aid in the discovery of significant predictor variables and shed light on the underlying structure of the data.

However, there are a few PCR limitations that should be considered while deploying the technique. Firstly, identifying the right number of principal components is essential and might ask for some knowledge or trial and error. Second, PCR relies on the unavoidable assumption that the relationship between the predictor factors and the result variable is linear. Third, there is a risk that PCR will exhibit multicollinearity, which can result in estimations of the regression coefficients that are unstable. Additionally, PCR is suitable when the data set contains highly correlated predictors.

In conclusion, PCR is an effective technique for deciphering high-dimensional data and making predictions based on a more condensed collection of major components. To assure the validity of the results, it is vital to carefully analyze the method's constraints and underlying presumptions.

7. Use the function pcr in the pls R package to use PCR to predict the β Lactoglobulin B levels from the spectra for a test set, where the test set is one third of the data. Motivate any decisions you make.

PCR to predict β Lactoglobulin B levels

```
set.seed(22200315) # for reproducibility
# Manipulating data to get beta Lacto globulin B and Spectra Data in the same data frame
Data_pcr <- df1[,c(13,52:582)]
```

```
#Splitting data into test and train
test_index <- createDataPartition(Data_pcr$beta_lactoglobulin_b,
                                  times = 1, p=0.33, list = F)
train_data <- Data_pcr[-test_index,]
test_data  <- Data_pcr[test_index,]
```

Setting scale=TRUE has the effect of standardizing each predictor, prior to generating the principal components, so that the scale on which each variable is measured will not have an effect. Setting validation="CV" causes pcr() to compute the ten-fold cross-validation error for each possible value of the number of principal components used. The resulting fit can be examined using summary().

Fitting the model

```
#Fitting the model
pls_fit <- pcr(beta_lactoglobulin_b~.,data=train_data,scale=TRUE,
               validation = "CV")
```

The CV score is provided for each possible number of components, ranging from 0 onwards. Note that pcr() reports the root mean squared error; in order to obtain the usual MSE, we must square this quantity.

Cumulative variation

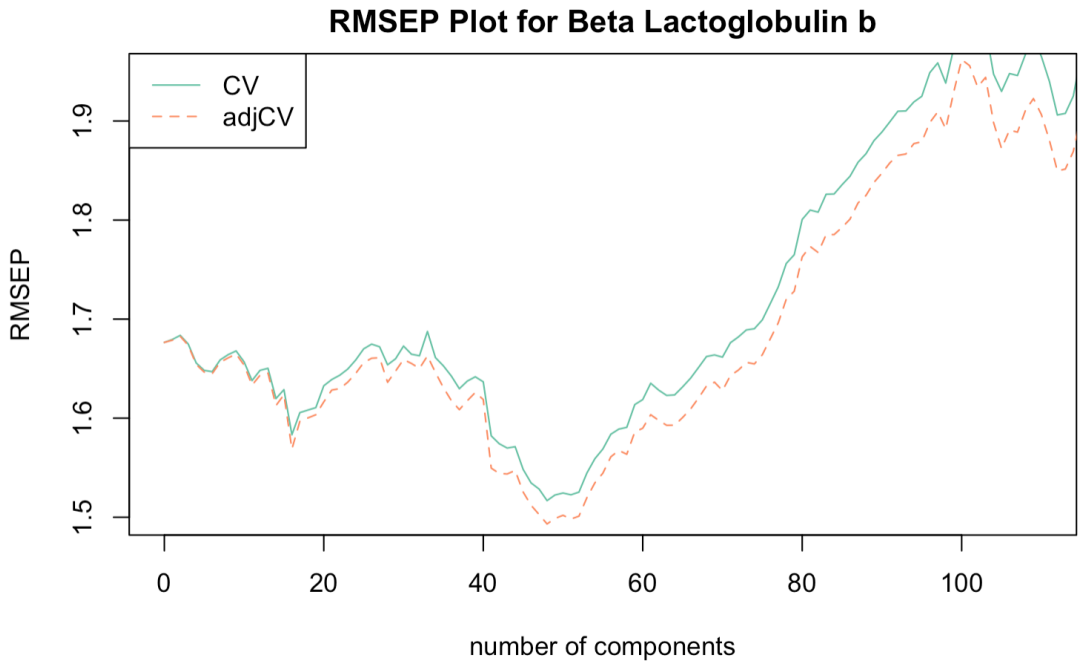
The optimum number of clusters for this model is 4 as that is when the lowest cross validation error occurs. We can confirm this decision by cumulative variation captured by the first 10 components. We can see that the value is stable after Comp 4 making ncomp=4 the ideal number of components for our model.

```
cumsum(explvar(pls_fit)[1:10])

##   Comp 1   Comp 2   Comp 3   Comp 4   Comp 5   Comp 6   Comp 7   Comp 8
## 53.03539 83.23290 94.96069 98.35918 98.99464 99.36433 99.57737 99.68242
##   Comp 9  Comp 10
## 99.77473 99.83325
```

We can also plot the cross-validation scores using the validationplot() function. Using val.type="MSEP" will cause the cross-validation MSE to be plotted.

Validation Plot

```
#Validation Plot
validationplot(pls_fit, val.type = "RMSEP",legendpos="topleft",
               main="RMSEP Plot for Beta Lactoglobulin b",
               ylim = c(1.50,1.95),xlim=c(0,110))
```



The validation results here are root mean squared error of prediction (RMSEP). There are two cross-validation estimates: CV is the ordinary CV estimate, and adjCV is a bias-corrected CV estimate.

Analysis of the PCR model

1.We see that the smallest cross-validation error occurs when M = 4 components are used. From the plot we also see that the cross-validation error is roughly the same when only one component is included in the model. This suggests that a model that uses just a small number of components might suffice.

2. The summary() function also provides the percentage of variance explained in the predictors and in the response using different numbers of components For instance,

setting M = 1 only captures 50.34% of all the variance, or information, in the predictors. In contrast, using M = 4 increases the value to 97.32%. If we were to use all M = p = 179 components, this would increase to 100%.

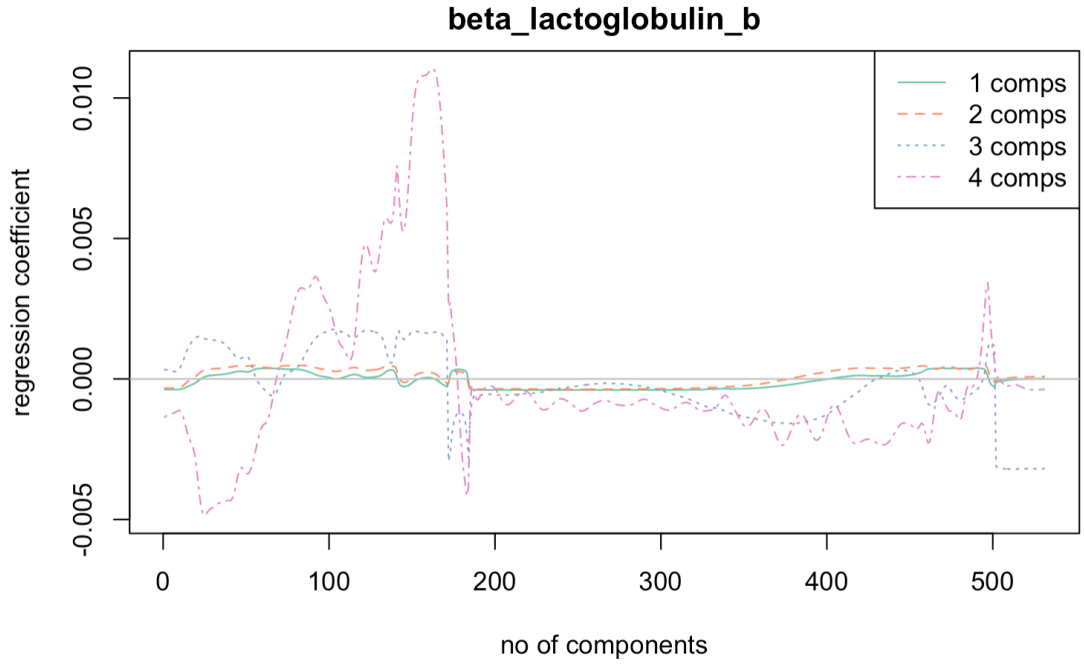
Evaluating performance using RMSEP of the model

```
RMSEP_comparison<-data.frame(x1<-(RMSEP(pls_fit)$val)[seq(2,20,2)],x2<-(RMSEP(pls_fit,newdata = test_data)$val)[seq(2,20,2)])

names(RMSEP_comparison)<-c("Cross validated estimate","Test RMSEP")

RMSEP_comparison[4,]
```

```
##   Cross validated estimate Test RMSEP
## 4                 1.673524   1.599431
```

Since the test value is close to the estimated value indicating that our prediction is good. We also observe that the RMSEP value is close to the adj CV value of ncomp=4 for the pls model.

```
plot(pls_fit, plottype = "coef", ncomp=1:4, legendpos = "topright", xlab = "no of components")
```

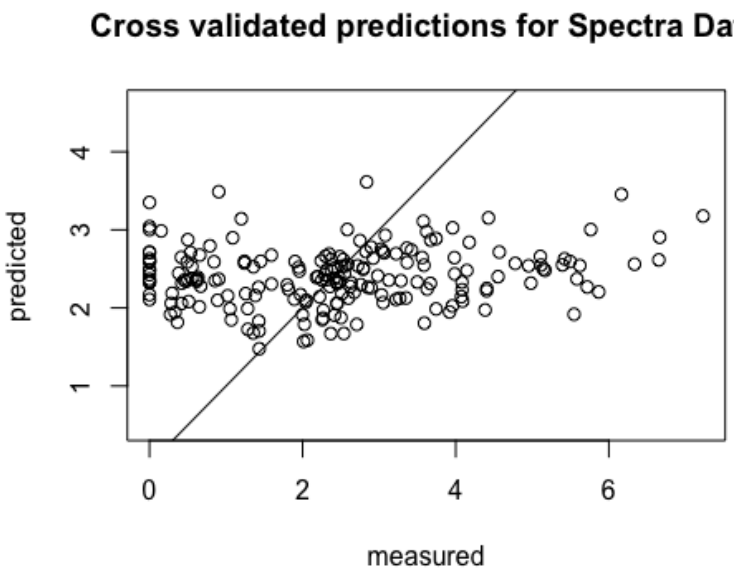


The graph above visualizes the regression coefficients. This allows simultaneous plotting of the regression vectors for several different numbers of components at once.

```
#Prediction values
test_pred <- predict(pls_fit, newdata = test_data) #to generate prediction values
head(test_pred[1:5])

## [1] 2.336365 2.383103 2.384759 2.423498 2.429562
```

Prediction

```
plot(pls_fit, ncomp = 4, asp = 1, line = TRUE,main="Cross validated predictions for Spectra Data")
```



From the plot above we can see that the predicted values do not accurately follow the observed data points. Alternatively, we can use MSE to assess the model performance.

8. Write your own code to impute the β Lactoglobulin B values that are 0 using principal components analysis on the seven milk proteins data. You must use the function prcomp or eigen in your solution. Comment on the results you obtain.

```
protein <- df1[c(7:13)]
colSums(protein==0)
```

```
##         kappa_casein      alpha_s2_casein      alpha_s1_casein
##                    0                    0                    0
##          beta_casein    alpha_lactalbumin beta_lactoglobulin_a
##                    0                    0                    0
## beta_lactoglobulin_b
##                   32
```

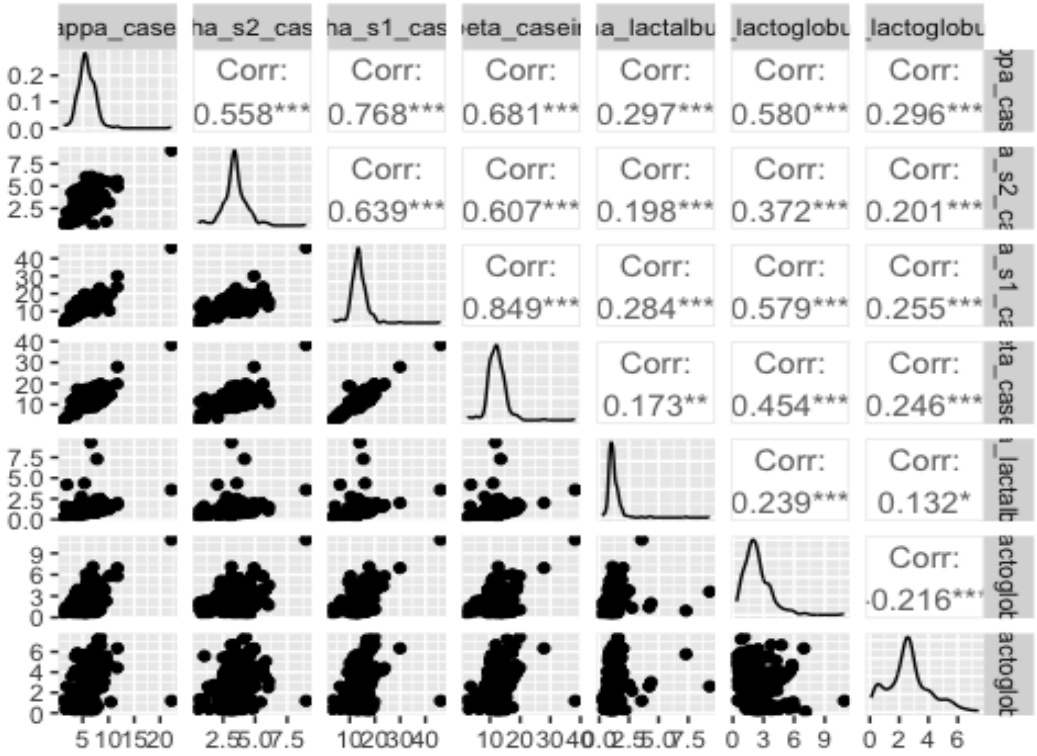
```
pca<-prcomp (protein,scale=TRUE)
```

Using Statistical Learning method, we use the prcomp function to extract the principal components of the complete data, and then use these components to predict the missing values using linear regression.

```r
#Extracting principal components
pcom<-pca$x
#Extracting loadings
loadings<- pca$rotation

# split data into complete and incomplete subsets
complete_rows <- which(rowSums((protein!=0))==7)
incomplete_rows <- which(rowSums((protein!=0))<7)
complete_data <- protein[complete_rows, ]
incomplete_data <- protein[incomplete_rows, ]
# predict missing values using linear regression on principal components
predicted_data <- incomplete_data

for (i in 1:ncol(incomplete_data))
{
 y <- complete_data[, i]
 x <- pcom[complete_rows, ]
 x_pred <- pcom[incomplete_rows, ]
 loadings_i <- loadings[, i]
 model <- lm(y ~ x %*% loadings_i)
 predicted_data[, i] <- predict(model, newdata = list(x = x_pred))
}

# combine predicted and observed data to form completed matrix
completed_data <- protein
completed_data[incomplete_rows, ] <- predicted_data
colSums(completed_data==0)

##         kappa_casein       alpha_s2_casein       alpha_s1_casein
##                    0                     0                     0
##          beta_casein     alpha_lactalbumin beta_lactoglobulin_a
##                    0                     0                     0
```

```
## beta_lactoglobulin_b
##                   0
```

```
ggpairs(completed_data[, colnames(completed_data)], progress = FALSE)
```



We can see that the correlation of the protein traits mainly positive with still a lot of skewness. This does not increase the model accuracy to much extent.

9. Using PCR, predict the β Lactoglobulin B values from the MIR spectra for a test set where the training set contains:

Using the same Spectra Dataset from Q7, I shall solve the questions below:

(a) all records with an observed, non−zero value of β Lactoglobulin B.

```
set.seed(22200315)
data_nonzero<- Data_pcr[Data_pcr$beta_lactoglobulin_b!=0,]
test_index1 <- createDataPartition(data_nonzero$beta_lactoglobulin_b,
                                   times = 1, p=0.33, list = F)
train1 <- data_nonzero[-test_index1,]
```

```
test1  <- data_nonzero[test_index1,]

#Fitting the model
fit1 <- pcr(beta_lactoglobulin_b~., data= train1, scale= TRUE, validation = "CV")
x_test1 <- test1[,-1]
y_test1<- test1[,1]

pred1 <- predict(fit1, x_test1, ncomp=60)
mse1=mean((pred1-y_test1)^2)
rmse1=sqrt(mse1)
```

The model's performance is evaluated using two metrics – MSE and RMSE, which are 2.076 and 1.4408 respectively. However, the model is not considered highly accurate because a significant amount of data (32 rows) is excluded due to the filtering of the dataset where the beta lactoglobulin B values are zero, resulting in a loss of valuable information.

(b) all records but where 0 values of β Lactoglobulin B are imputed using the observed mean.

```
set.seed(22200315)
Data_mean<-Data_pcr
Data_mean$beta_lactoglobulin_b[(Data_mean$beta_lactoglobulin_b)==0]<-
mean(Data_pcr$beta_lactoglobulin_b,na.rm=TRUE)

test_index2 <- createDataPartition(Data_mean$beta_lactoglobulin_b,
                                   times = 1, p=0.33, list = F)
train2 <- Data_mean[-test_index2,]
test2 <- Data_mean[test_index2,]

#Fitting the model
fit2 <- pcr(beta_lactoglobulin_b~., data= train2, scale= TRUE, validation = "CV")
x_test2 <- test2[,-1]
```

```
y_test2<- test2[,1]

pred2 <- predict(fit2, x_test2, ncomp=60)
mse2=mean((pred2-y_test2)^2)
rmse2=sqrt(mse2)
mse2
```

```
## [1] 1.79891
```

```
rmse2
```

```
## [1] 1.341234
```

The model's performance metrics MSE and RMSE are 1.798 and 1.341 respectively. The MSE value is comparatively lower which signifies that this model is more accurate than the first one and the predicted values are much closer to the observed values.

(c) all records but where 0 values of β Lactoglobulin B values are imputed using principal components analysis

```
set.seed(22200315)
pca2<- prcomp(Data_pcr[,'beta_lactoglobulin_b'],scale=TRUE)

complete_rows <- which(rowSums((Data_pcr!=0))==532)
incomplete_rows <- which(rowSums((Data_pcr!=0))<532)
complete_data <- Data_pcr[complete_rows, ]
incomplete_data <- Data_pcr[incomplete_rows, ]

# predict missing values using linear regression on principal components

predicted_data <- incomplete_data
for (i in 1:ncol(incomplete_data)) {
  y <- complete_data[, i]
  x <- pcom[complete_rows, ]
  x_pred <- pcom[incomplete_rows, ]
```

```
  loadings_i <- loadings
  model <- lm(y ~ x %*% loadings_i)
  predicted_data[, i] <- predict(model, newdata = list(x = x_pred))
}

# combine predicted and observed data to form completed matrix
df_pca <- Data_pcr
df_pca[incomplete_rows, ] <- predicted_data

test_index3 <- createDataPartition(df_pca$beta_lactoglobulin_b,
                                   times = 1, p=0.33, list = F)
train3 <- df_pca[-test_index3,]
test3 <- df_pca[test_index3,]

#Fitting the model
fit3 <- pcr(beta_lactoglobulin_b~., data= train3, scale= TRUE, validation = "CV")
x_test3 <- test3[,-1]
y_test3<- test3[,1]

pred3 <- predict(fit3, x_test3, ncomp=60)
mse3=mean((pred3-y_test3)^2)
rmse3=sqrt(mse3)
mse3
```

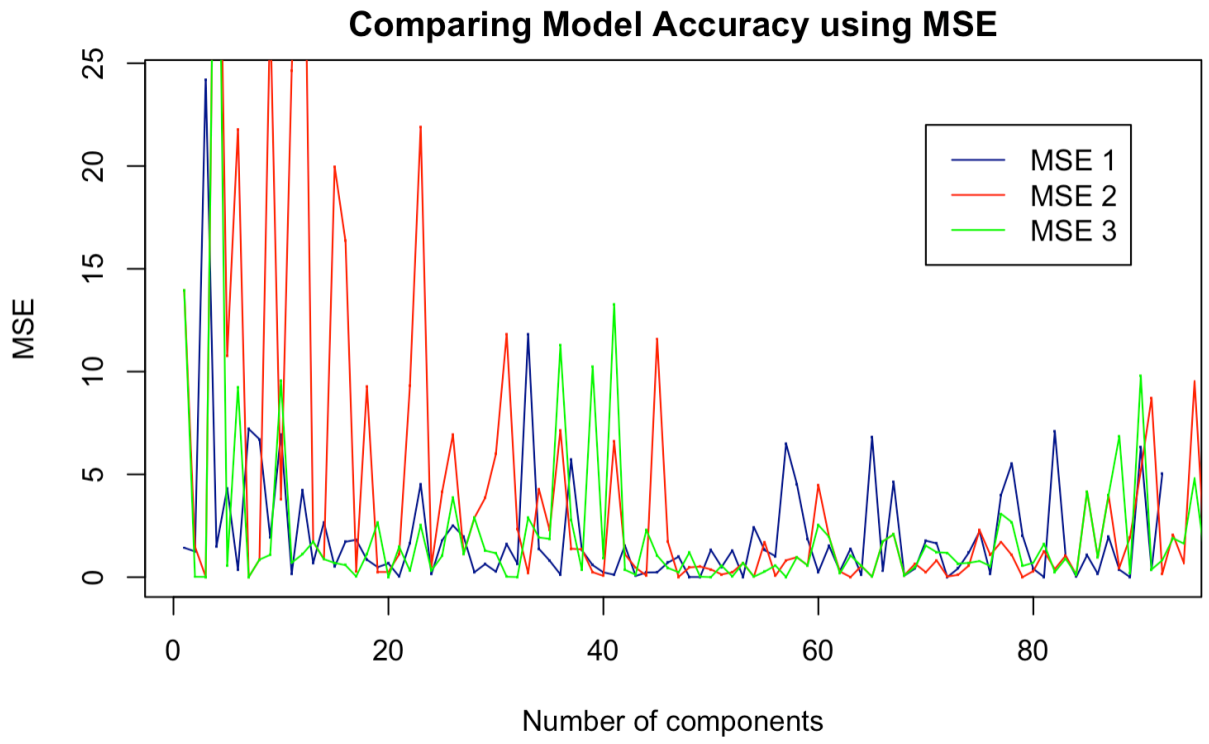
```
## [1] 2.267925
```

```
rmse3
```

```
## [1] 1.505963
```

The model's performance metrics MSE and RMSE are 2.267 and 1.505 respectively. The model is not extremely accurate.

```
mse_1=((pred1-y_test1)^2)
mse_2=((pred2-y_test2)^2)
mse_3=((pred3-y_test3)^2)

#Plotting and comparing MSE of above three models
plot(mse_1, type="o", col="darkblue", pch=".", xlab="Number of components",
     ylab="MSE", lty=1,
     main='Comparing Model Accuracy using MSE')
points(mse_2, col="red", pch=".") #Adding MSE2 to same plot
lines(mse_2, col="red",lty=1)
points(mse_3, col="green",pch=".") #Adding MSE3 to the same plot
lines(mse_3, col="green", lty=1)

#add a legend in top left corner of chart at (x, y) coordinates = (1, 19)
legend(70,22,legend=c("MSE 1","MSE 2","MSE 3"), col=c("darkblue","red","green"),
lty=c(1,1,1), ncol=1)
```



This graph plots the range of variation in Mean squared error for all three methods. The maximum variation is observed in the second model making it the best amongst the three with least mean squared error.