Milk Spectra Analysis - Multivariate Analysis Task

Clementine Surya

```
# load necessary packages
library(tidyr)
library(factoextra) # to plot the dendogram
library(clValid) # to compute internal measure when doing hierarchical clustering
library(pls) # for PCR predict
library(Metrics)
```

Background of the Assignment

This assignment is based on the milk spectra data set (Milk_MIR_Traits_data_2023.csv). The initial columns in the dataset contain details (i.e. covariates) of the cows which produced the milk samples and the protein and technological traits of the milk samples measured in the laboratory. The data in the final 531 columns are the MIR spectra, with the first row of these columns detailing the wavenumber (measured in cm^{-1}). The spectral values in the dataset are the absorbance values (the log10 of the reciprocal of the transmittance value). The water region has been removed.

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.

```
df <- read.csv("Milk_MIR_Traits_data_2023.csv", sep =";") # load the data set
set.seed(22200226) # set seed to my student number
delete_obs <- sample(1:nrow(df), 1) # randomly generate a number between 1 and n (where n is the number of r
ows in the dataset)
df <- df[-delete_obs, ] # delete the observation from the dataset
```

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

First, I remove any observations which contains any missing value in β Lactoglobulin B column.

```
na_values <- sum(is.na(df$beta_lactoglobulin_b)) # count the NA's observations in β Lactoglobulin B column
cat(paste0("The number NA's observations: ", na_values, "\n"))
```

```
## The number NA's observations: 124
```

```
df2 <- df[!is.na(df$beta_lactoglobulin_b), ] # remove NA's observations
```

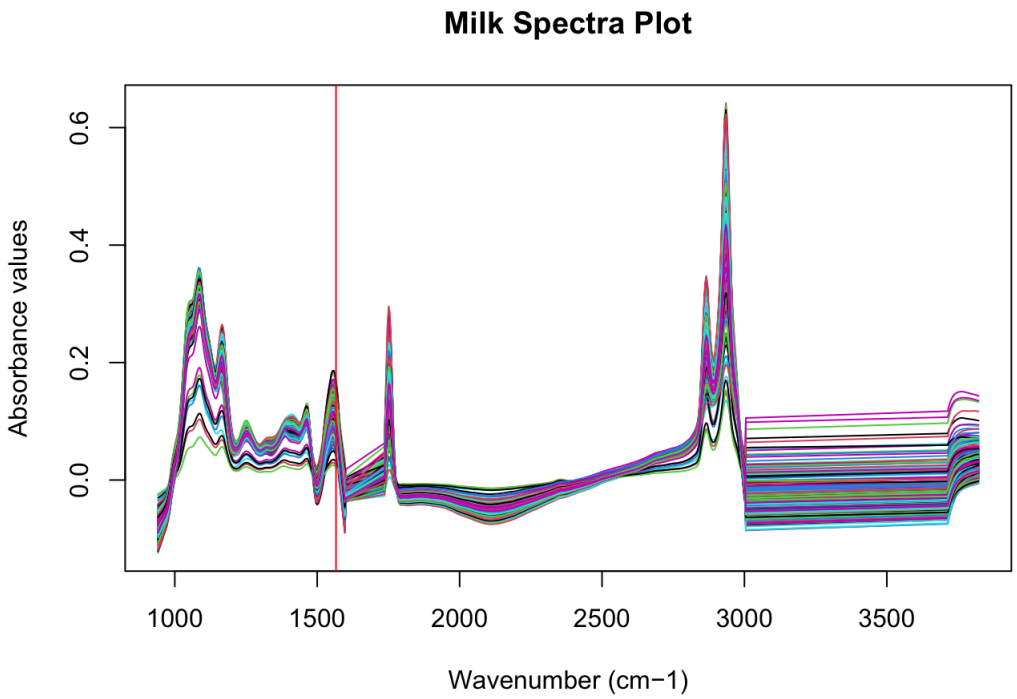
Then, I visualize the milk spectra using below code:

```
MIR_spectra <-  df2[ , (ncol(df2)-530) : ncol(df2)] # subset the 531 MIR spectra
df3 <- cbind(df2$beta_lactoglobulin_b, MIR_spectra) # combine β Lactoglobulin B and MIR_spectra column
names(df3)[names(df3) == "df2$beta_lactoglobulin_b"] <- "beta_lactoglobulin_b" # rename the column
cor_matrix <- cor(df3) # compute correlation matrix
wavelength_with_highest_correlation <- names(which.max(cor_matrix["beta_lactoglobulin_b", -1])) # compute th
e wavelength that gives highest correlation with β Lactoglobulin B protein
cat(paste0("Wavenumber with the highest correlation with β Lactoglobulin B protein: ", wavelength_with_highe
st_correlation, "\n"))
```

```
## Wavenumber with the highest correlation with β Lactoglobulin B protein: X1566
```

```
correlation_coefficient <- cor(df3$beta_lactoglobulin_b, df3$X1566) # compute the correlation value of the p
rotein and the X1566 spectra

# visualise the milk spectra
wavelength <- as.numeric(gsub("X", "", colnames(MIR_spectra)))
# plot the spectra over the wavelengths
matplot(t(MIR_spectra),
        x = wavelength,
        type = "l",
        lty = 1,
        xlab="Wavenumber (cm-1)",
        ylab="Absorbance values",
        main = "Milk Spectra Plot")
# add a vertical line at wavelength 1566
abline(v=1566, col="red")
```



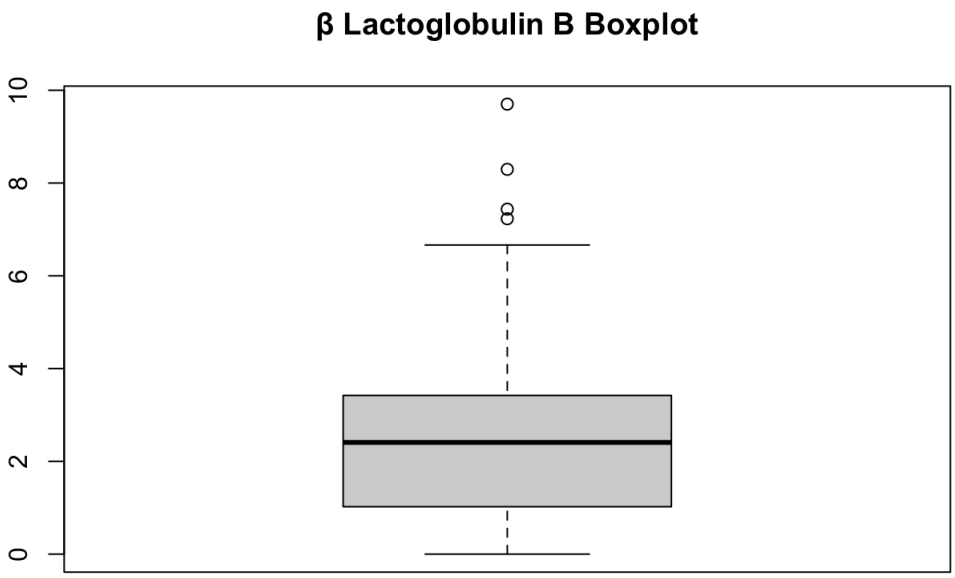
The milk spectra plot above shows the absorption of light by different wavelengths of milk. The x-axis shows the wavelengths measured in cm^{-1} , and the y-axis shows the absorbance values (the log10 of the reciprocal of the transmittance value). The plot shows several peaks and valleys, indicating the presence of different components in milk.

One of the prominent peaks in the plot is around 2800 cm^{-1} . Another peak at around 2700 cm^{-1} and 1750 cm^{-1} .The milk spectra plot provides valuable information about the composition of milk and can be used to analyze the quality of milk.

Furthermore, based on the correlation analysis of the milk spectra, we found that the concentration of β Lactoglobulin B protein is highest at wavenumber 1566 cm^{-1} , visualized by the red line for reference.

Now, I visualise β Lactoglobulin B protein:

```
boxplot(df3$beta_lactoglobulin_b,
        main = "β Lactoglobulin B Boxplot") # visualise the β Lactoglobulin B using boxplot
```




```
summary(df3$beta_lactoglobulin_b) # compute the summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.029   2.408   2.440   3.410   9.702
```

In β Lactoglobulin B distribution, the median is 2.408 and the mean is 2.440. The distribution is rightly skewed, where the mean is greater than the median. The spread of the middle distribution is between 1.029 and 3.410 (IQR range of 2.381). Furthermore, the it is ranging from 0 to 9.702 (range value of 9.702). There are several outliers located above the maximum value in this distribution.

At last, we remove any observations with β Lactoglobulin B outside of 3 standard deviations from the mean of the trait.

```
mean_blb <- mean(df2$beta_lactoglobulin_b) # calculate mean of β Lactoglobulin B
sd_blb <- sd(df2$beta_lactoglobulin_b) # calculate standard deviation of β Lactoglobulin B

# create new data set with only the observations within 3 standard deviations from the mean of β Lactoglobul
in B
df4 <- df2[df2$beta_lactoglobulin_b >= mean_blb - 3*sd_blb &
                  df2$beta_lactoglobulin_b <= mean_blb + 3*sd_blb, ]
```

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.

First, I standardize the milk spectra.

```
MIR_spectra_new <-  df4[ , (ncol(df4)-530) : ncol(df4)] # subset only 531 MIR spectra from df4
stdspectra <- scale(MIR_spectra_new) # standardize the milk spectra
```

Hierarchical Clustering

Here, I do hierarchical clustering to the milk spectra using average, single, and complete linkage and choose the best one. I use cophenetic function to choose the best linkage. A higher cophenetic correlation coefficient indicates that the linkage method better preserves the original pairwise distances, and therefore may be a better choice for clustering the data.

```
cl.average = hclust(dist(stdspectra), method="average")
cl.single = hclust(dist(stdspectra), method="single")
cl.complete = hclust(dist(stdspectra), method="complete")

# choosing the dendogram by the highest correlation of cophenetic and distance
dist <- dist(stdspectra)
average_coph <- as.numeric(cophenetic(cl.average)) # highest correlation
cor(average_coph, dist)
```

```
## [1] 0.8515918
```

```
single_coph <- as.numeric(cophenetic(cl.single))
cor(single_coph, dist)
```

```
## [1] 0.8089581
```

```
complete_coph <- as.numeric(cophenetic(cl.complete))
cor(complete_coph, dist)
```

```
## [1] 0.6139278
```

As we can see from the result above, average linkage appears to have the highest correlation. Thus, I am going to use average linkage method to do the analysis.

Now, I choose number of cluster using internal measures and obtain 3 clusters as the optimal number using below method:

```
internal_measure <- clValid(stdspectra, nClust = 3:4,
                    clMethods = "agnes",
                    validation = "internal",
                    metric = "euclidean",
                    method = "average")
summary(internal_measure)
```

```
##
## Clustering Methods:
##  agnes
##
## Cluster sizes:
##  3 4
##
## Validation Measures:
##                             3       4
##
## agnes Connectivity   7.9758 15.9599
##       Dunn           0.2293  0.0983
##       Silhouette     0.4758  0.3615
##
## Optimal Scores:
##
##              Score  Method Clusters
## Connectivity 7.9758 agnes  3
## Dunn         0.2293 agnes  3
## Silhouette   0.4758 agnes  3
```

Based on the internal measure that I conducted above, I cut the dendogram at a level that results in three distinct clusters.

```
hcl = cutree(cl.average, k = 3)
table(hcl)
```

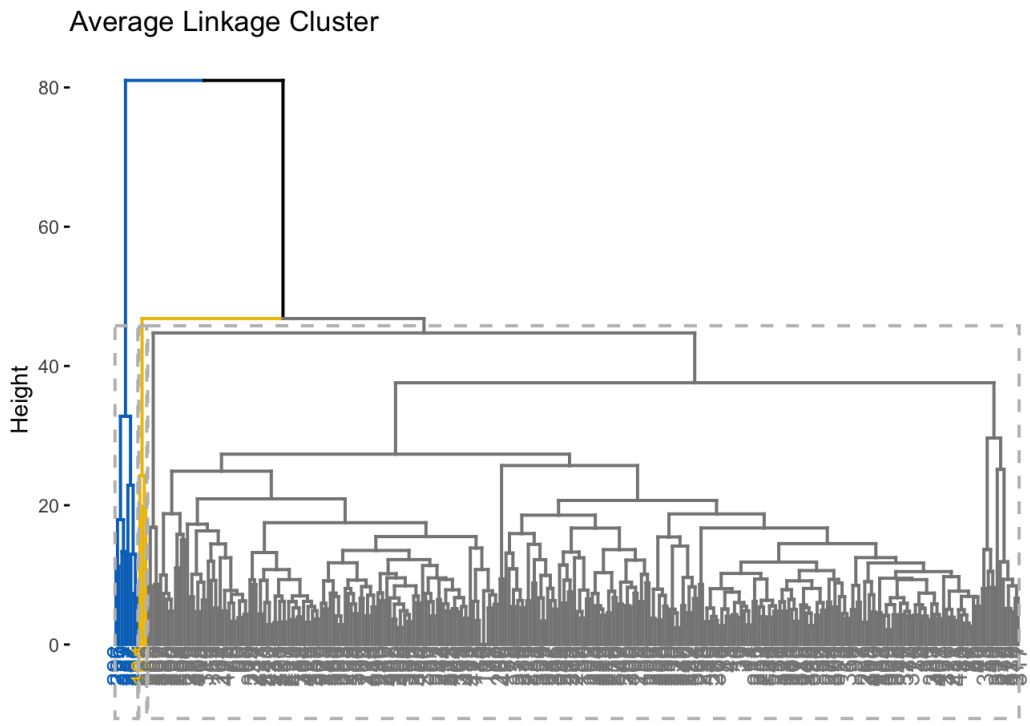
```
## hcl
##   1   2   3
## 293   8   3
```

```
table(hcl, df4[,1])
```

```
##
## hcl FRX- Hol Fri HOX-  JE JEX-  MO  NR
##   1   15     186    3  57   22   1   9
##   2    0       6    0   1    1   0   0
##   3    1       1    0   0    1   0   0
```

At last, I show the number of milk spectra in each clusters and present the dendogram below.

```
fviz_dend(cl.average, k = 3, k_colors = "jco", rect = T,
          main = "Average Linkage Cluster")
```



Summary:
To plot the dendrogram, I utilized the average linkage method because it provides the highest cophenetic correlation. The cophenetic correlation is a measure of the correlation between the pairwise distances of observations in the original data and the distances between them in the dendrogram. A higher cophenetic correlation indicates that the dendrogram better preserves the pairwise distances among observations in the original data.

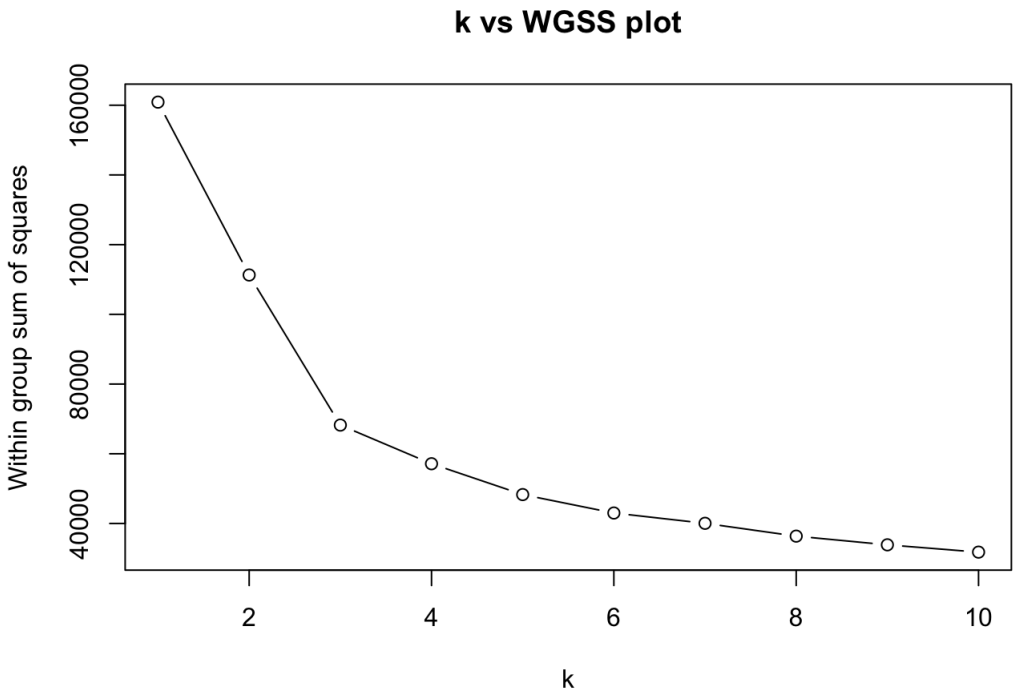
I identified three distinct clusters that correspond to my chosen number of clusters. To determine the optimal number of clusters, I utilized internal measure with AGNES method. Internal measures, such as Silhouette and Dunn index combine measures of compactness and separation of the clusters, while Connectivity indicates the degree of connectedness of the clusters.

From the result of Hierarchical Clustering:
Cluster 1 contains 293 observations, cluster 2 contains 8 observations, and cluster 3 contains 3 observations. These results suggest that the majority of observations belong to cluster 1.

KMeans Clustering
Now, I do k-means clustering to determine if there are clusters of similar MIR spectra in the data.

```
df5 <- cbind(df2$Breed, stdspectra) # combine breed and MIR_spectra columns
names(df5)[names(df5) == "df2$Breed"] <- "breed" # rename the column

# First, run the k-means clustering algorithm over the range k = 1 to 10 clusters and record the within grou
p sum of squares (WGSS) for each value of k using below code
WGSS = rep(0,10)
n = nrow(stdspectra)
WGSS[1] = (n-1) * sum(apply(stdspectra, 2, var))
for(k in 2:10){
  WGSS[k] = sum(kmeans(stdspectra, centers = k)$withinss)
}
plot(1:10, WGSS, type="b", xlab="k", ylab="Within group sum of squares", main = "k vs WGSS plot") # plot wit
hin group sum of squares for each k
```




```
k = 3 # optimal number of cluster
pcl = kmeans(stdspectra, center=k)
table(pcl$cluster) # present the pcl clustering
```

```
##
##   1   2   3
## 129 167   8
```

Based on the results of K-means clustering method, the optimal number of clusters is found to be 3. The clustering method has produced 129 observations in cluster 1, 167 observations in cluster 2, and 8 observations in cluster 3.

On the other hand, the hierarchical clustering method using average linkage has also produced an optimal number of 3 clusters. However, the distribution of the observations among the clusters is different from that of K-means clustering. Specifically, hierarchical clustering method has assigned 293 observations to cluster 1, 8 observations to cluster 2, and 3 observations to cluster 3.

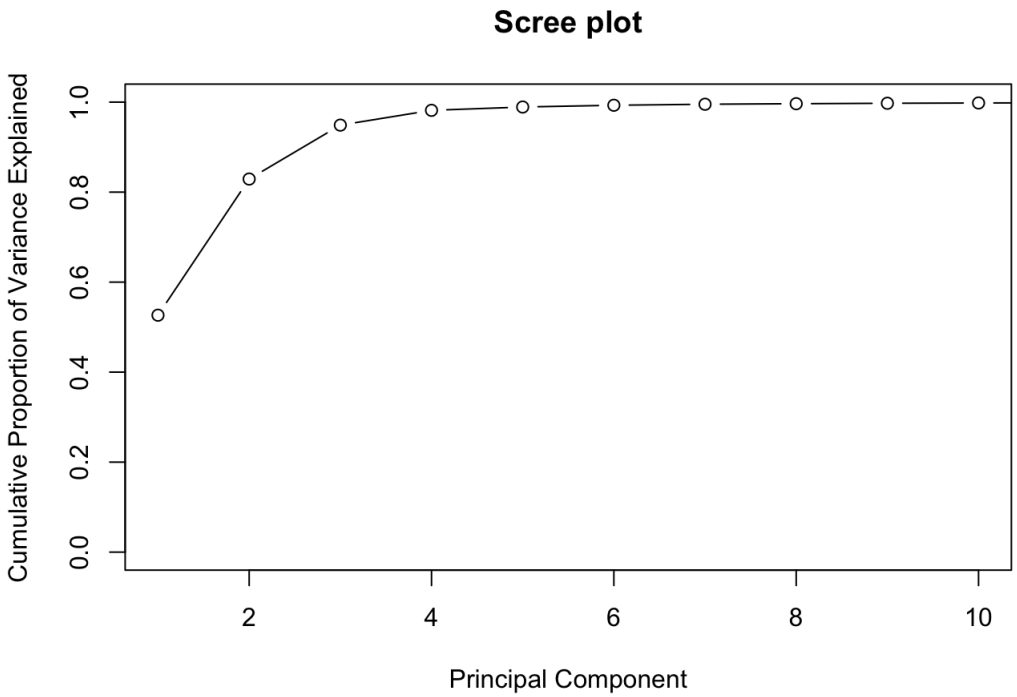
The difference in the positioning of the clusters between the two methods can be attributed to the fact that they use different algorithms for clustering. K-means clustering is a partition-based clustering algorithm, which aims to minimize the sum of squared distances between the data points and their assigned cluster centers. On the other hand, our hierarchical clustering model using average linkage merges clusters based on the average distance between their data points.

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.

Here, I apply principal components analysis to the spectral data.

```
pca <- prcomp(stdspectra) # apply principal component analysis on the standardized spectral data

# plot the cumulative proportion of the variance explained by the first 10 principal components
mir.pca.eigen <- eigen(cov(stdspectra))
mir.pca.ve <- mir.pca.eigen$values/sum(mir.pca.eigen$values)
plot(cumsum(mir.pca.ve),
     xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     xlim = c(1,10),
     ylim = c(0,1),
     type = 'b',
     main = 'Scree plot')
```

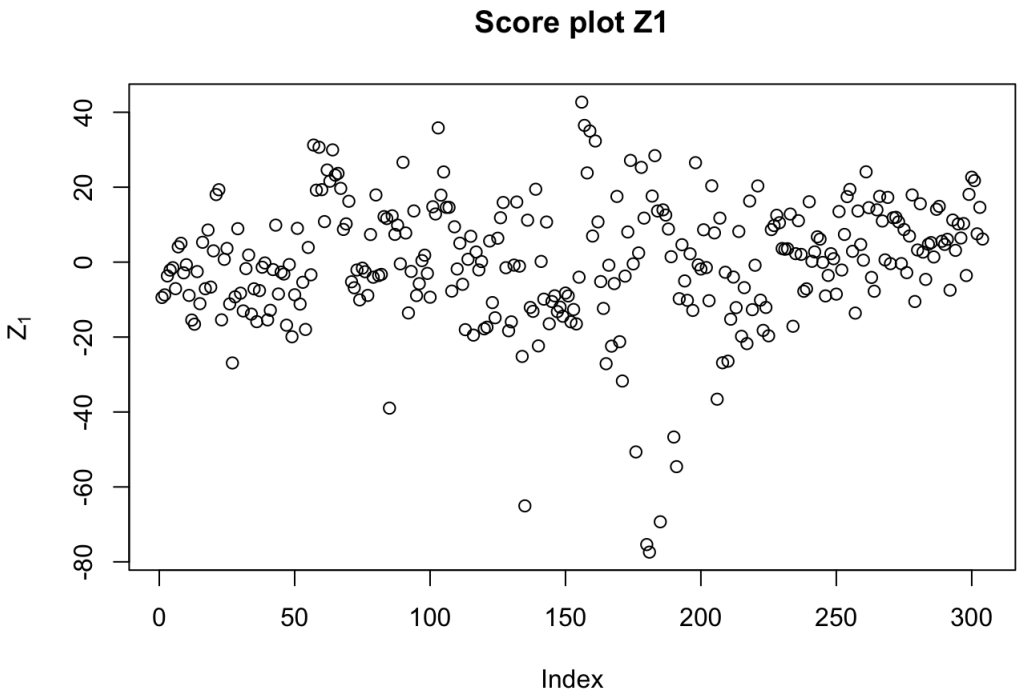


The first PC accounts for 52.6% of the variation in the data. The second principal component accounts for an additional 30.2% of the variation. The second principal component accounts for another additional 12% of the variation.

The elbow appears to occur at the third principal component. This means that the first three components the first three principal components appear to be the most informative, as they collectively explain approximately 95% of the total variance in the data, while the remaining components contribute less to the overall variance.

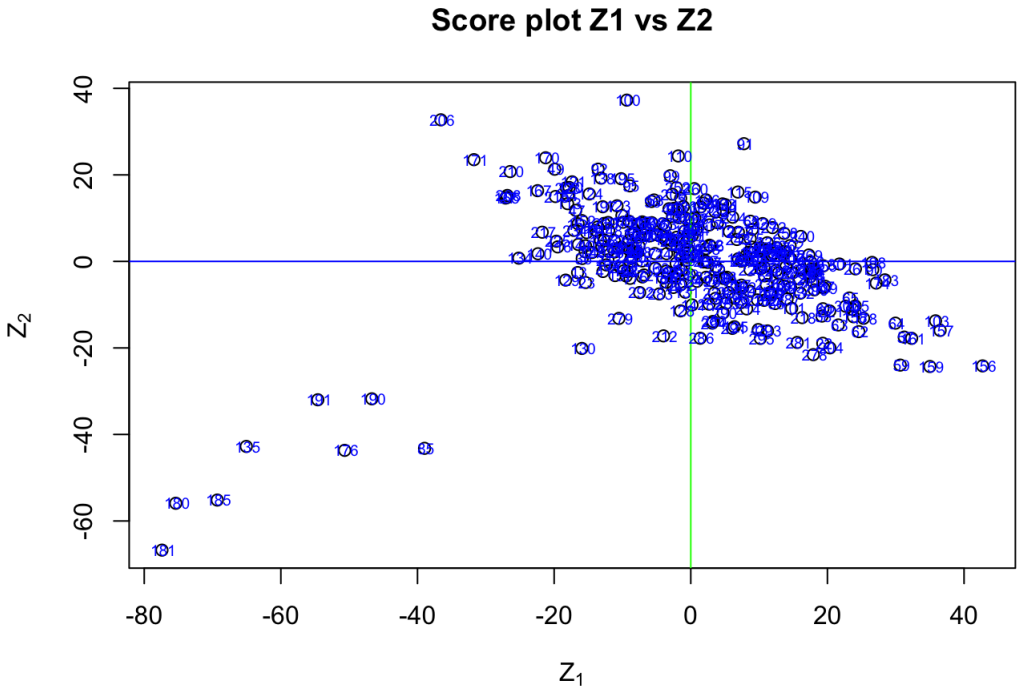
5. Derive the principal component scores for the milk samples from first principles (i.e., you should not use an inbuilt function such as predict(. . .)). Plot the principal component scores for the milk samples. Comment on any structure you observe.
Here, I derive the principal component scores for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

```
# select the first two principal components for plotting purpose
pca.loading <- mir.pca.eigen$vectors[,1:2]
# rename columns
colnames(pca.loading) <- c('PC1', 'PC2')
rownames(pca.loading) <- colnames(stdspectra)
pca.scores <- data.frame(stdspectra %*% pca.loading)
rownames(pca.scores) <- seq(1, nrow(pca.scores))
# derive the principal component scores for the milk samples from first principle
pc_1 <- pca.scores[,1]
# plot the first principal component scores for the milk samples
plot(pc_1, main = 'Score plot Z1', ylab = expression('Z'[1]))
```




```
# Plot first vs principical component scores
plot(pca.scores,
     xlab = expression('Z'[1]),
     ylab = expression('Z'[2]),
     main = 'Score plot Z1 vs Z2')
abline(h = 0, col = "blue")
abline(v = 0, col = "green")

# Plot the scores as points
text(pca.scores[,1]+0.2,
      pca.scores[,2],
      rownames(pca.scores),
      col="blue", cex=0.6)
```



The first two score vectors, Z1 and Z2, explain the greatest variation in the data. We look at the {Z1,Z2} scatter plot of the scores.

Points close the average appear at the origin of the score plot. An observation that is at the mean value for all k-variables will have a score vector Zi = [0,0,…,0].

Scores further out are either outliers. Original observations in X that are similar to each other will be similar in the score plot, while observations much further apart are dissimilar.

6. Here is the synopsis of the PCR method that contains (i) the method's purpose, (ii) general description of how the method works, (iii) choices that need to be made when using the method and (iv) the advantages and disadvantages of the method.

i. PCR Method's Purpose

PCR, which is short for Principal Component Regression, is a regression analysis method that relies on Principal Component Analysis (PCA). Its main purpose is to estimate the regression coefficients of a linear regression model, especially for high-dimensional data, where the number of predictor variables is significantly higher than the number of observations.

In such datasets, the traditional linear regression method can face challenges such as multicollinearity, overfitting, and model instability. When these challenges occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, principal components regression reduces the standard errors. It is hoped that the net effect will be to give more reliable estimates.

PCR addresses these challenges by reducing the number of predictors in the model while retaining most of the information contained in the original features. By reducing the number of predictors used in the model, PCR can improve the model's performance and reduce the risk of overfitting. Additionally, by eliminating the effects of multicollinearity, PCR can help improve the stability and interpretability of the model.

ii. How PCR works

Consider a linear regression problem with the model form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \epsilon$$

If p is too large, it may not be practical to use all the predictors in a regression model. In such cases, Principal Component Analysis (PCA) can be used to reduce the number of predictors by transforming them into d principal components, where d is less than p. These principal components are linear combinations of the original predictors, and can be used to fit a regression model. Typically, the principal components with higher variances are selected for regression, which are based on eigenvectors corresponding to higher eigenvalues of the sample variance-covariance matrix of the explanatory variables.

The steps involved are as follows:

1. Standardize the explanatory variables: The first step is to standardize the explanatory variables to ensure that they are on the same scale. This is important because the principal components are sensitive to the scale of the variables.
2. Conduct principal component analysis (PCA): The second step is to perform PCA on the standardized explanatory variables to obtain the principal components. The number of principal components to retain is usually determined by examining the proportion of variance explained by each component and selecting those that explain a significant amount of the total variability in the data.
3. Regress the response variable on the principal components: The final step is to regress the response variable on the selected principal components. This is done using standard regression techniques, such as least squares regression.

The primary idea behind PCR is that a smaller number of principal components can capture most of the variability in the data and the relationship with the target variable. Therefore, instead of using all the original features for regression, a subset of the principal components is used.

iii. Choices that need to be made when using the method

There are a few choices that need to be made when using Principal Component Regression (PCR):

1. Standardization: It is recommended to standardizing each predictor prior to generating the principal components. This standardization ensures that all variables are on the same scale. In the absence of standardization, the high-variance variables will tend to play a larger role in the principal components obtained, and the scale on which the variables are measured will ultimately have an effect on the final PCR model.
2. Number of Principal Components: One of the most critical choices when using PCR is determining the number of principal components to retain for the regression model. This choice depends on the dataset and is often determined using cross-validation. Generally, a smaller number of principal components is preferred to reduce overfitting and improve model performance, but it is essential to ensure that enough variance in the data is captured.
3. Regression Method: Once the principal components are determined, the next choice is selecting a regression method to fit the model. Typically, ordinary least squares (OLS) regression is used to fit the model, but other regression methods such as ridge regression or lasso regression can also be used.
4. Interpretation: Another consideration when using PCR is the interpretation of the resulting model. Since the principal components are linear combinations of the original features, it may be challenging to interpret the relationship between the principal components and the target variable. Therefore, it is essential to carefully consider the practical implications of the resulting model and the feasibility of implementing it in real-world scenarios.

iv. Advantages and Disadvantages of the method

Advantages:

- PCR fits a linear regression model on a reduced set of k principal components instead of all the original features. This reduction in the number of predictors helps to avoid overfitting and results in better model performance compared to a standard linear regression model that uses all the original features.
- PCR can also remove multicollinearity in the data by eliminating principal components that are associated with small eigenvalues.

Disadvantage:

- Since principal components are used in the model, these predictors lose their 'explainability' compared to the original features.
- PCA does not consider the target variable when determining principal components. Hence, there is no guarantee that the principal components with the largest variance will be the best ones for predicting the target. There is a chance that the directions with high predictive power but low variance are omitted, resulting in poor model performance.

7. In this part, I 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.

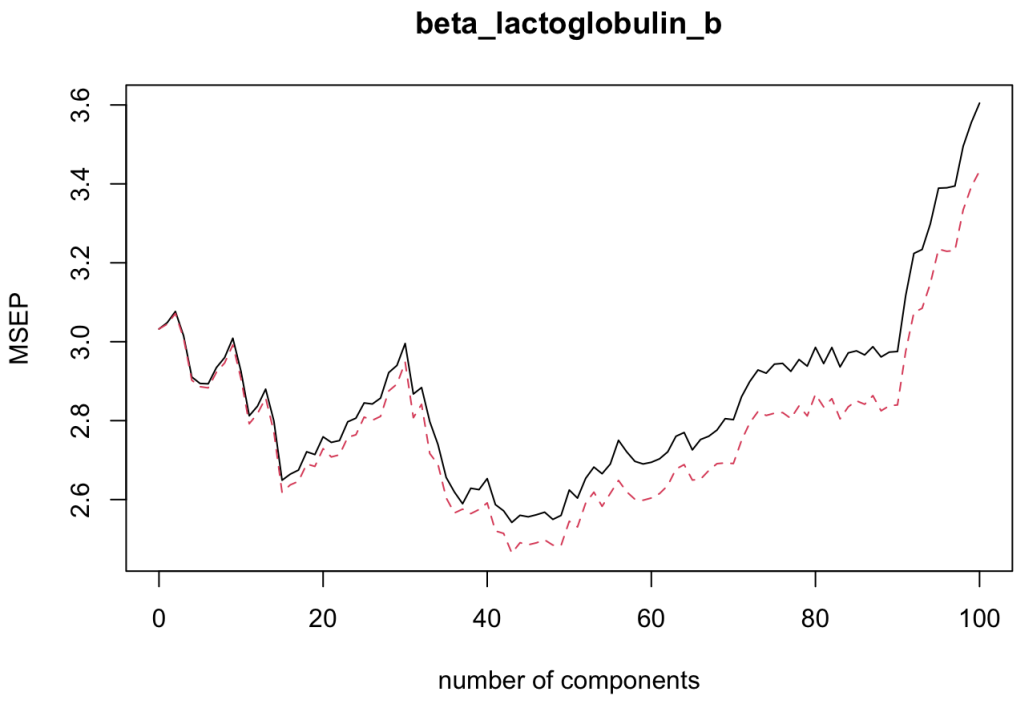
```
set.seed(22200226)

# split the data into a training set (2/3) and a test set (1/3)
n <- nrow(df4)
train_idx <- sample(seq_len(n), size = round(n*2/3))
train <- df4[train_idx, ]
test <- df4[-train_idx, ]

# define the predictor and response variables
x_train <- train[ , (ncol(train)-530) : ncol(train)] # spectra for the train set
y_train <- data.frame(train$beta_lactoglobulin_b) # β Lactoglobulin B for train set
train_data <- cbind(y_train, x_train) # merge y_train and x_train
colnames(train_data)[1] <- "beta_lactoglobulin_b" # rename the first column
x_test <- test[ , (ncol(test)-530) : ncol(test)]  # spectra for the test set
y_test <- data.frame(test$beta_lactoglobulin_b) # β Lactoglobulin B for test set
test_data <- cbind(y_test, x_test) # merge x_train and y_train
colnames(test_data)[1] <- "beta_lactoglobulin_b"

# perform PCR on the training set
pcr_model <- pcr(beta_lactoglobulin_b ~., data = train_data, scale = TRUE, validation = "CV")

# display the MSEP validation plot
pcr_model$ncomp <- 100
validationplot(pcr_model, val.type = "MSEP")
```



```
pcr_model_cv <- RMSEP(pcr_model)$val[1,,] # use the RMSEP function to pull out the CV and adjCV
min_cv <- which.min(pcr_model_cv) - 1 # get the number of component that gives lowest CV. Here, we have to s
ubtract 1, since the model with no principal components is included in the output of RMSEP(pcr_model) as wel
l
cat(paste0("The optimal number of components: ", min_cv, "\n"))
```

```
## The optimal number of components: 43
```

```
# predict the test set using optimal number of components = 43
pcr_pred <- predict(pcr_model, x_test, ncomp=43)

# compute RMSE from the prediction
rmse_val <- rmse(actual = test$beta_lactoglobulin_b, predicted = as.numeric(pcr_pred))
rmse_val <- round(rmse_val,4)

cat(paste0("RMSE: ", rmse_val, "\n"))
```

```
## RMSE: 1.5814
```

I obtain RMSE of 1.5814 with the PCR method that we compute above to predict the levels of β Lactoglobulin B based on the spectral data in the test set.

8. Using the algorithm from the book "An Introduction to Statistical Learning with Applications in R by James et al. (2021) in section 12.3, I impute the β Lactoglobulin B values that are 0 using matrix completion method that uses principal components analysis on the seven milk proteins data. I put the Algorithm in the appendix:

First, I create protein dataframe:

```
protein <- cbind(df4$alpha_s1_casein, df4$alpha_s2_casein, df4$beta_casein, df4$kappa_casein, df4$alpha_lact
albumin, df4$beta_lactoglobulin_a, df4$beta_lactoglobulin_b) # combine all the protein

colnames(protein) <- c("alpha_s1_casein", "alpha_s2_casein","beta_casein","kappa_casein","alpha_lactalbumi
n","beta_lactoglobulin_a","beta_lactoglobulin_b") # rename the columns

protein <- data.frame(protein) # create protein data frame
protein["beta_lactoglobulin_b"][protein["beta_lactoglobulin_b"] == 0] <- NA
Xna <- data.matrix(scale(protein)) # change 0 value in beta lactoglobulin b to NA
```

We first write a function that takes in a matrix, and returns an approximation to the matrix using the prcomp() function:

```
fit.pca <- function(X, M = 1) {
  pcaob <- prcomp(X)
  with(pcaob,
       x[, 1:M, drop = FALSE] %*%
         t(rotation[, 1:M, drop = FALSE])
  )
}
```

To conduct Step 1 of the algorithm, we initialize Xhat — this is $\tilde{X}$ in Algorithm — by replacing the missing values with the column means of the non-missing entries.

```
# initialize Xhat by replacing the missing values with the column means of the non-missing entries
Xhat <- data.frame(Xna)
xbar <- colMeans(Xna, na.rm = TRUE)
Xhat$beta_lactoglobulin_b[is.na(Xhat$beta_lactoglobulin_b)] <- mean(Xhat$beta_lactoglobulin_b, na.rm = TRUE)
Xhat <- data.matrix(Xhat)
```

Here, we set ourselves up to measure the progress of our iterations:

```
thresh <- 1e-7
rel_err <- 1
iter <- 0
ismiss <- is.na(Xna) # ismiss is a new logical matrix with the same dimensions as Xna
mssold <- mean((scale(Xna, xbar, FALSE)[!ismiss])^2) # store the mean squared error of the non-missing eleme
nts of the old version of Xhat
mss0 <- mean(Xna[!ismiss]^2) # store the mean of the squared non-missing elements
```

```
# iterate until the relative error falls below thresh = 1e-7
while(rel_err > thresh) {
  iter<-iter+1
  # Step 2(a): approximate Xhat using fit.pca(); we call this Xapp
  Xapp <- fit.pca(Xhat, M = 1)
  # Step 2(b): use Xapp to update the estimates for elements in Xhat that are missing in Xna
  Xhat[ismiss] <- Xapp[ismiss]
  # Step 2(c): compute the relative error
  mss <- mean(((Xna - Xapp)[!ismiss])^2) # store the mean squared error of the non-missing elements of the c
urrent version of Xhat in mss
  rel_err <- (mssold - mss) / mss0
  mssold <- mss
  cat("Iter:", iter, "MSS:", mss,
      "Rel. Err:", rel_err, "\n")
}
```

```
## Iter: 1 MSS: 0.468226 Rel. Err: 0.530205
## Iter: 2 MSS: 0.4681813 Rel. Err: 4.48198e-05
## Iter: 3 MSS: 0.468181 Rel. Err: 3.242358e-07
## Iter: 4 MSS: 0.468181 Rel. Err: 2.42732e-09
```

We see that after 4 iterations, the relative error has fallen below thresh = 1e-7, and so the algorithm terminates. When this happens, the mean squared error of the non-missing elements equals 0.468.

9. Using PCR, predict the β Lactoglobulin B values from the MIR spectra for a test set where the training set contains:
(a) all records with an observed, non-zero value of β Lactoglobulin B.
(b) all records but where 0 values of β Lactoglobulin B are imputed using the observed mean.
(c) all records but where 0 values of β Lactoglobulin B values are imputed using principal components analysis.
Comment on what you observe.

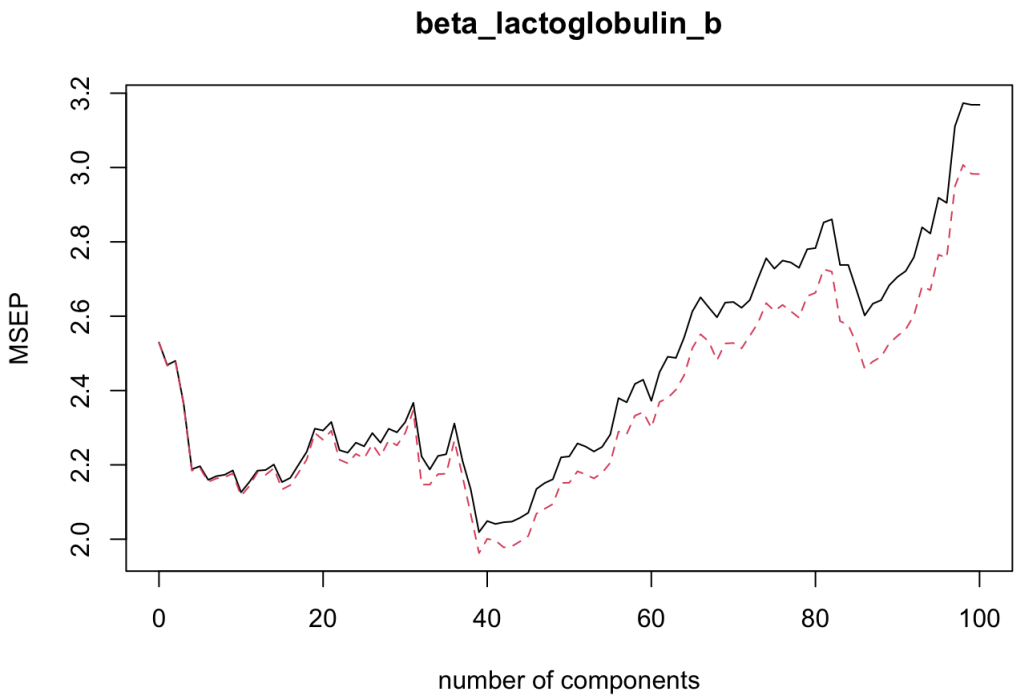
PCR for dataset (a)
When performing PCR using dataset (a), I obtain RMSE 1.35.

```
set.seed(22200226)
# Create dataset for the all records with an observed, non zero value of β Lactoglobulin B
dat_a <-  data.frame(cbind(df4$beta_lactoglobulin_b,  df4[ , (ncol(df4)-530) : ncol(df4)]))
names(dat_a)[names(dat_a) == "df4.beta_lactoglobulin_b"] <- "beta_lactoglobulin_b"
dat_a["beta_lactoglobulin_b"][dat_a["beta_lactoglobulin_b"] == 0] <- NA # change 0 value to NA
dat_a <- dat_a %>%
  drop_na() # remove observations that contains NA

# Split the data into a training set (2/3) and a test set (1/3)
n <- nrow(dat_a)
train_idx_a <- sample(seq_len(n), size = round(n*2/3))
train_a <- data.frame(dat_a[train_idx_a, ])
test_a <- data.frame(dat_a[-train_idx_a, ])

# Define the predictor and response variables
x_train_a <- data.frame(train_a[ , (ncol(train_a)-530) : ncol(train_a)]) # spectra for the train set
y_train_a <- data.frame(train_a$beta_lactoglobulin_b) # β Lactoglobulin B for train set
train_data_a <- cbind(y_train_a, x_train_a) # combine y_train and x_train
colnames(train_data_a)[1] <- "beta_lactoglobulin_b" # rename the protein column
x_test_a <- data.frame(test_a[ , (ncol(test_a)-530) : ncol(test_a)])  # spectra for the test set
y_test_a <- data.frame(test_a$beta_lactoglobulin_b) # β Lactoglobulin B for test set
test_data_a <- cbind(y_test_a, x_test_a) # combine y_test and x_test
colnames(test_data_a)[1] <- "beta_lactoglobulin_b" # rename the protein column

# Perform PCR on the training set
pcr_model_a <- pcr(beta_lactoglobulin_b ~., data = train_data_a, scale =TRUE, validation = "CV")
# choose the optimal number of component based on validation plot
pcr_model_a$ncomp <- 100
validationplot(pcr_model_a, val.type = "MSEP")
```




```
pcr_model_cv_a <- RMSEP(pcr_model_a)$val[1,,] # use the RMSEP function to pull out the CV and adjCV
min_cv_a <- which.min(pcr_model_cv_a) - 1 # get the number of component that gives lowest CV
cat(paste0("The optimal number of components: ", min_cv_a, "\n"))
```

```
## The optimal number of components: 39
```

```
pcr_pred_a <- predict(pcr_model_a, x_test_a, ncomp=39) # predict the model
rmse_val_a <- rmse(actual = test_a$beta_lactoglobulin_b, predicted = as.numeric(pcr_pred_a)) # compute rmse
cat(paste0("RMSE: ", round(rmse_val_a,4), "\n"))
```

```
## RMSE: 1.3517
```

PCR for dataset (b)

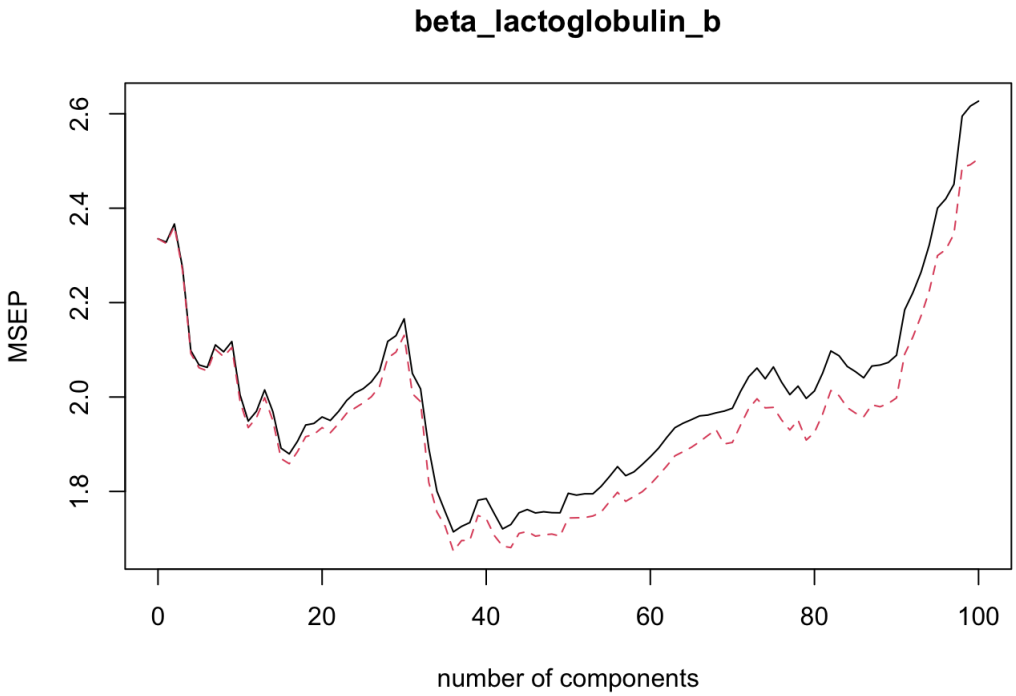
When performing PCR using dataset (b), I obtain RMSE 1.38.

```
set.seed(22200226)
# dataset for (b)
dat_b <- data.frame(cbind(df4$beta_lactoglobulin_b,  df4[ , (ncol(df4)-530) : ncol(df4)]))
names(dat_b)[names(dat_b) == "df4.beta_lactoglobulin_b"] <- "beta_lactoglobulin_b"
dat_b["beta_lactoglobulin_b"][dat_b["beta_lactoglobulin_b"] == 0] <- NA # change 0 value to NA
dat_b$beta_lactoglobulin_b[is.na(dat_b$beta_lactoglobulin_b)] <- mean(dat_b$beta_lactoglobulin_b, na.rm = TR
UE)

# Split the data into a training set (2/3) and a test set (1/3)
n <- nrow(dat_b)
train_idx_b <- sample(seq_len(n), size = round(n*2/3))
train_b <- data.frame(dat_b[train_idx_b, ])
test_b <- data.frame(dat_b[-train_idx_b, ])

# Define the predictor and response variables
x_train_b <- data.frame(train_b[ , (ncol(train_b)-530) : ncol(train_b)]) # spectra for the train set
y_train_b <- data.frame(train_b$beta_lactoglobulin_b) # β Lactoglobulin B for train set
train_data_b <- cbind(y_train_b, x_train_b) # combine y_train and x_train
colnames(train_data_b)[1] <- "beta_lactoglobulin_b" # rename protein column
x_test_b <- data.frame(test_b[ , (ncol(test_b)-530) : ncol(test_b)])  # spectra for the test set
y_test_b <- data.frame(test_b$beta_lactoglobulin_b) # β Lactoglobulin B for test set
test_data_b <- cbind(y_test_b, x_test_b) # combine y_test and x_test
colnames(test_data_b)[1] <- "beta_lactoglobulin_b" # rename protein column

# Perform PCR on the training set
pcr_model_b <- pcr(beta_lactoglobulin_b ~., data = train_data_b, scale =TRUE, validation = "CV")
# choose the optimal number of component based on validation plot
pcr_model_b$ncomp <- 100
validationplot(pcr_model_b, val.type = "MSEP")
```




```
pcr_model_cv_b <- RMSEP(pcr_model_b)$val[1,,] # use the RMSEP function to pull out the CV and adjCV
min_cv_b <- which.min(pcr_model_cv_b) - 1 # get the number of component that gives lowest CV
cat(paste0("The optimal number of components: ", min_cv_b, "\n"))
```

```
## The optimal number of components: 36
```

```
pcr_pred_b <- predict(pcr_model_b, x_test_b, ncomp= 36) # predict the model
rmse_val_b <- rmse(actual = test_b$beta_lactoglobulin_b, predicted = as.numeric(pcr_pred_b)) # compute rmse
cat(paste0("RMSE: ", round(rmse_val_b,4), "\n"))
```

```
## RMSE: 1.3822
```

PCR for dataset (c)

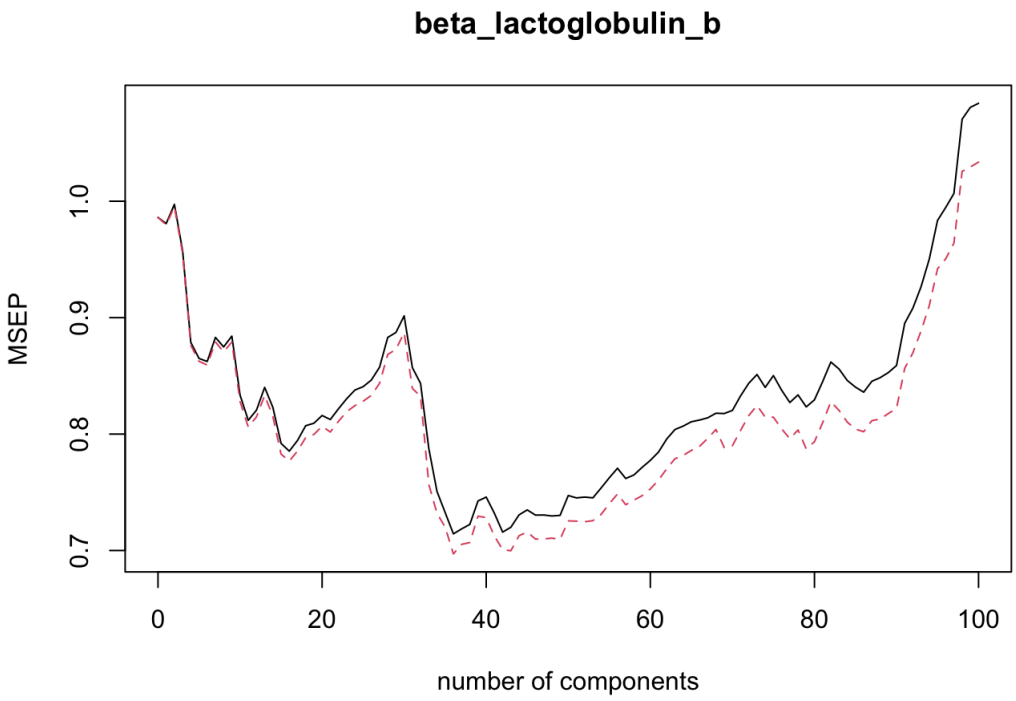
When performing PCR using dataset (c), I obtain RMSE 0.90.

```
set.seed(22200226)
# dataset for (c)
Xhat_df <- data.frame(Xhat)
dat_c <- data.frame(cbind(Xhat_df$beta_lactoglobulin_b,  df4[ , (ncol(df4)-530) : ncol(df4)]))
names(dat_c)[names(dat_c) == "Xhat_df.beta_lactoglobulin_b"] <- "beta_lactoglobulin_b"

# Split the data into a training set (2/3) and a test set (1/3)
n <- nrow(dat_c)
train_idx_c <- sample(seq_len(n), size = round(n*2/3))
train_c <- data.frame(dat_c[train_idx_c, ])
test_c <- data.frame(dat_c[-train_idx_c, ])

# Define the predictor and response variables
x_train_c <- data.frame(train_c[ , (ncol(train_c)-530) : ncol(train_c)]) # spectra for the train set
y_train_c <- data.frame(train_c$beta_lactoglobulin_b) # β Lactoglobulin B for train set
train_data_c <- cbind(y_train_c, x_train_c) # combine y_train and x_train
colnames(train_data_c)[1] <- "beta_lactoglobulin_b" # rename protein column
x_test_c <- data.frame(test_c[ , (ncol(test_c)-530) : ncol(test_c)])  # spectra for the test set
y_test_c <- data.frame(test_c$beta_lactoglobulin_b) # β Lactoglobulin B for test set
test_data_c <- cbind(y_test_c, x_test_c) # combine y_test and x_test
colnames(test_data_c)[1] <- "beta_lactoglobulin_b" # rename protein column

# Perform PCR on the training set
pcr_model_c <- pcr(beta_lactoglobulin_b ~., data = train_data_c, scale =TRUE, validation = "CV")
# choose the optimal number of component based on validation plot
pcr_model_c$ncomp <- 100
validationplot(pcr_model_c, val.type = "MSEP")
```




```
pcr_model_cv_c <- RMSEP(pcr_model_c)$val[1,,] # use the RMSEP function to pull out the CV and adjCV
min_cv_c <- which.min(pcr_model_cv_c) - 1 # get the number of component that gives lowest CV
cat(paste0("The optimal number of components: ", min_cv_c, "\n"))
```

```
## The optimal number of components: 36
```

```
pcr_pred_c <- predict(pcr_model_c, x_test_c, ncomp=36) # predict the model
rmse_val_c <- rmse(actual = test_c$beta_lactoglobulin_b, predicted = as.numeric(pcr_pred_c)) # compute rmse
cat(paste0("RMSE: ", round(rmse_val_c,4), "\n"))
```

```
## RMSE: 0.9008
```

To sum up, the best model goes to dataset (c) with all records of zero values in β Lactoglobulin B values being imputed using principal components analysis. It provides the lowest RMSE which is 0.90 compared to dataset (a) with RMSE of 1.35 and dataset (b) with RMSE of 1.38, they have similar RMSE results.

The reason why PCA method is a better approach for handling missing values because it can preserve the most essential patterns and variability in the data while addressing the issue of missing data. This is particularly suitable for the milk spectra dataset, where most of the missingness can be caused by random factors, such as instrument errors or sample damage. As a result, this method outperforms the other two approaches, resulting in better outcomes.

References

Djafar, N. M. (2021, July 25). Hierarchical Clustering Using R Studio. Medium. https://mutmainnahdj.medium.com/hierarchical-clustering-using-r-studio-875b0a4dd0ac\ (https://mutmainnahdj.medium.com/hierarchical-clustering-using-r-studio-875b0a4dd0ac\)

How to choose the appropriate clustering algorithms for your data? - Unsupervised Machine Learning - Easy Guides - Wiki - STHDA. (n.d.). Www.sthda.com. Retrieved March 29, 2023, from http://www.sthda.com/english/wiki/wiki.php?id_contents=7932# (http://www.sthda.com/english/wiki/wiki.php?id_contents=7932#):~:text=The%20internal%20measures%20included%20in

Interpretation and Visualization. (2017, May 1). Www.geo.fu-Berlin.de. https://www.geo.fu-berlin.de/en/v/soga/Geodata-analysis/Principal-Component-Analysis/principal-components-basics/Interpretation-and-visualization/index.html\ (https://www.geo.fu-berlin.de/en/v/soga/Geodata-analysis/Principal-Component-Analysis/principal-components-basics/Interpretation-and-visualization/index.html\)

Leung, K. (2022). Principal Component Regression — Clearly Explained and Implemented. [online] Medium. Available at: https://towardsdatascience.com/principal-component-regression-clearly-explained-and-implemented-608471530a2f\ (https://towardsdatascience.com/principal-component-regression-clearly-explained-and-implemented-608471530a2f\)

Alice, M. (2016, July 21). Performing Principal Components Regression (PCR) in R | R-bloggers. https://www.r-bloggers.com/2016/07/performing-principal-components-regression-pcr-in-r/ (https://www.r-bloggers.com/2016/07/performing-principal-components-regression-pcr-in-r/)

kjytay. (2018, October 15). Obtaining the number of components from cross validation of principal components regression. Statistical Odds & Ends. https://statisticaloddsandends.wordpress.com/2018/10/15/obtaining-the-number-of-components-from-cross-validation-of-principal-components-regression/ (https://statisticaloddsandends.wordpress.com/2018/10/15/obtaining-the-number-of-components-from-cross-validation-of-principal-components-regression/)

James et al. (2023). An Introduction to Statistical Learning with Applications in R.