STAT40740 - Assignment 1 - Multivariate Analysis

Conor Ryan - 16340116

2023-04-04

```
# reading in required libraries
library(dplyr); library(tidyr); library(ggplot2); library(ggthemes);
library(ggplot2); library(knitr); library(kableExtra);
library(readxl); library(grid); library(e1071); library(GGally);
library(pls);

# Set working directory
setwd("C:/Users/conor/OneDrive/Documents/Data Analytics Masters/Stage 2/Trimester 2/STAT40740 - Multiva
```

Question 1

I load in the data from the Milk_MIR_Traits_data_2023 data set which contains mid-infrared (MIR) spectral data extracted from milk samples, and their associated protein and technological traits below. I randomly sample a number between 1 and n, where n is the number of rows in the data set and remove this observation. I have set the seed to be my student number in order to ensure my results are reproducible.

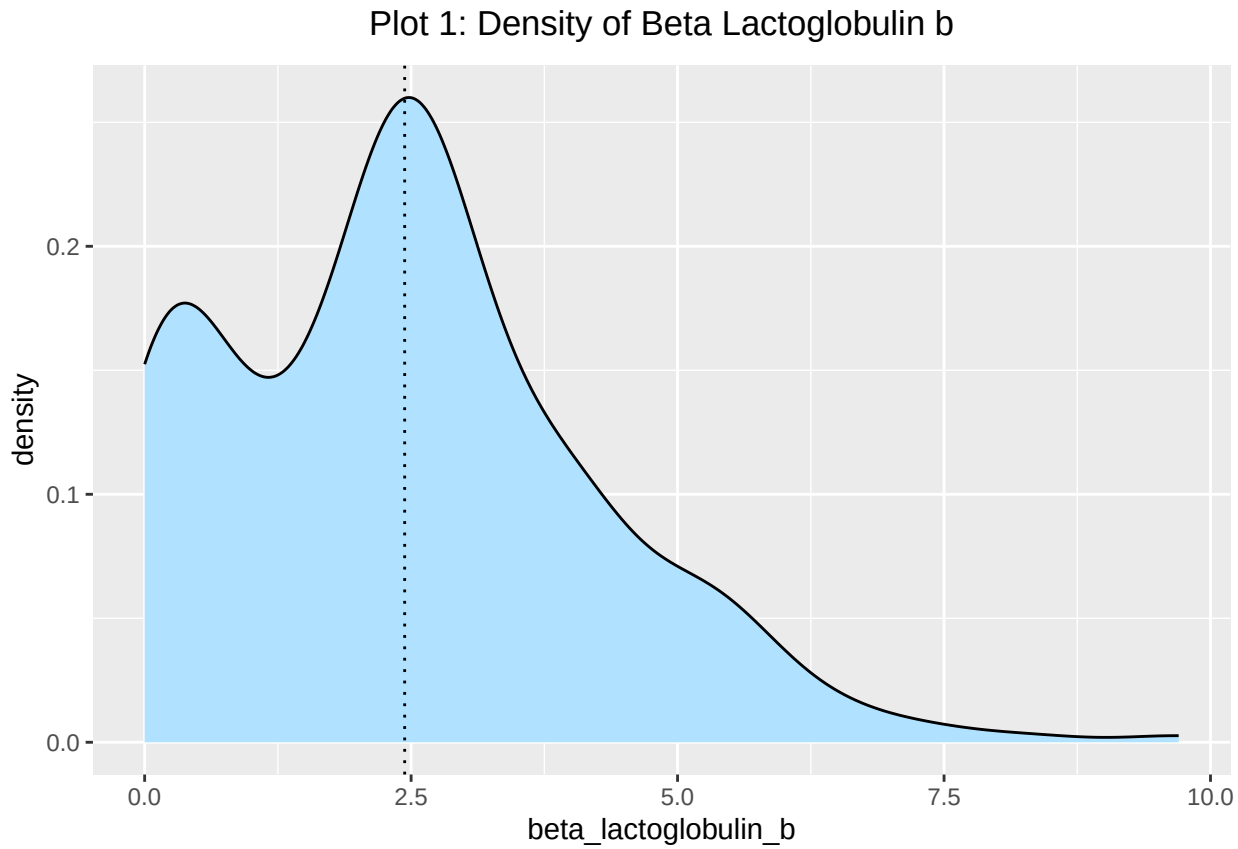
```
# clear workspace
rm(list = ls())

# Set seed to student number
set.seed(16340116)

# read in data file
data <- read.csv("Milk_MIR_Traits_data_2023.csv")

# correcting wavelength column names
colnames(data)[52:ncol(data)] <- gsub("X","",colnames(data)[52:ncol(data)])

# randomly generate number from 1:n and delete obs from data set
rand_obs <- sample(c(1:nrow(data)),size = 1)
data <- data[-rand_obs,]
```

Question 2

In question 2, I remove any observations in the data where the value for β Lactoglobulin b is missing or NA. I then produce separate plots to visualise both the protein trait β Lactoglobulin b and the MIR spectra.

```
# removing observations with missing Lactoglobulin b values
data <- data %>%
  filter(!is.na(beta_lactoglobulin_b))
```

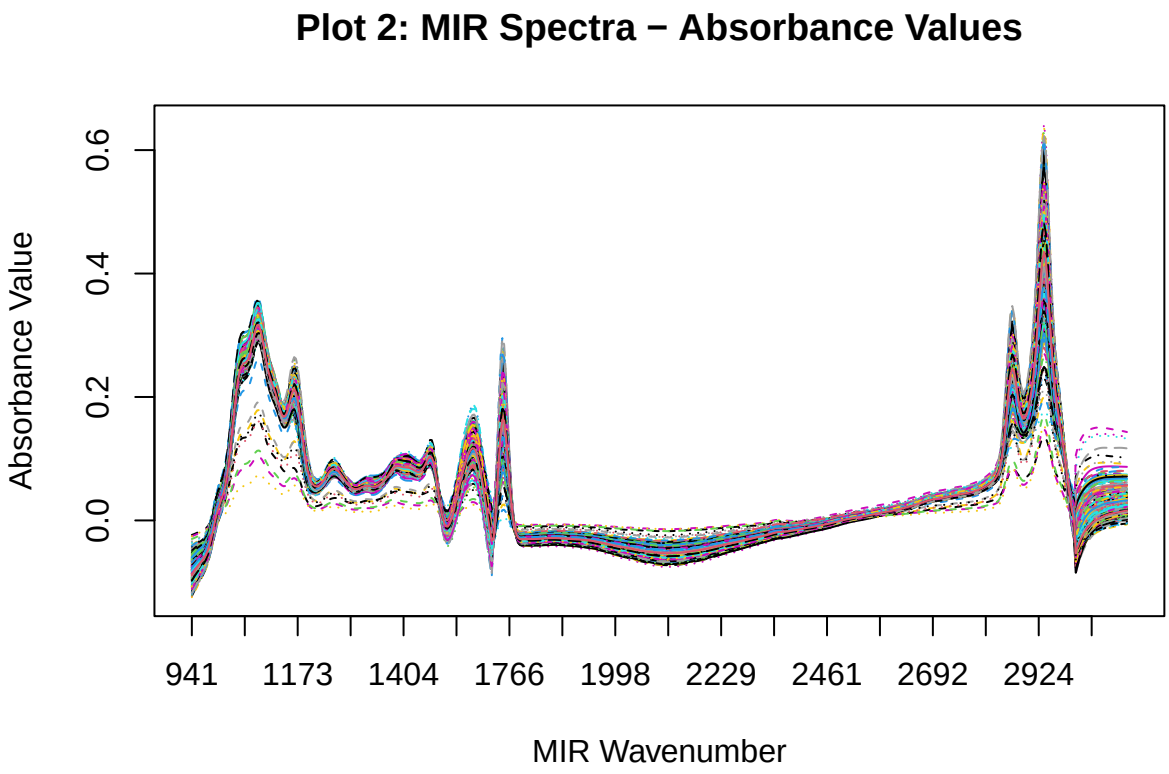
```
# plotting density of beta lactoglobulin protein
ggplot(data) +
  geom_density(aes(x = beta_lactoglobulin_b), fill = 'lightskyblue1') +
  geom_vline(xintercept = mean(data$beta_lactoglobulin_b), linetype='dotted') +
  labs(title = expression("Plot 1: Density of Beta Lactoglobulin b")) +
  theme(plot.title = element_text(hjust = 0.5))
```



I use a density plot to visualise the distribution of the β Lactoglobulin b protein trait. We can see from the plot above that the protein is multi-modal with peaks in density at around 0.25 and 2.44. We can see from the density plot that the distribution is non-symmetric and skewed to the right indicating there is more density in higher values of the protein. We also note that the distribution of β Lactoglobulin b is truncated at zero with 0.15 density of values being equal to zero. These zero values are assumed to be "missing at random" and will be imputed in a later question using PCA analysis.

I will now plot each of the MIR spectra for each observation in order to visualise the absorbance values at each wavelength.

```
# Plotting spectra
  matplot(t(data[,52:ncol(data)]), type = 'l', col = 1:nrow(data), xaxt='n',
        xlab = "MIR Wavenumber",
        ylab = "Absorbance Value",
        main = "Plot 2: MIR Spectra - Absorbance Values")
axis(side = 1,at = seq(1,ncol(data)-52,30),labels = colnames(data)[seq(52,ncol(data),30)])
```



I have plotted the MIR Spectra absorbance values for each of the 306 observations in the data set. Each observation has been plotted as a different colour line graph to help visualise the variance between values at each wavelength. We observe that for certain spectra values, there is a high variance in absorbance values between observations whereas in other spectra values, the absorbance values are tightly packed together. The spectra values which have a higher variance in absorbance values between observations will be more useful in clustering / partitioning the observations together.

Finally, I will remove any observations which have a β Lactoglobulin b value outside 3 standard deviations of the mean.

```
# Removing Lactoglobulin b values 3 sds from mean
data <- data %>%
  filter(abs(beta_lactoglobulin_b) < abs(3*sd(beta_lactoglobulin_b) + mean(data$beta_lactoglobulin_b)))
```

Question 3

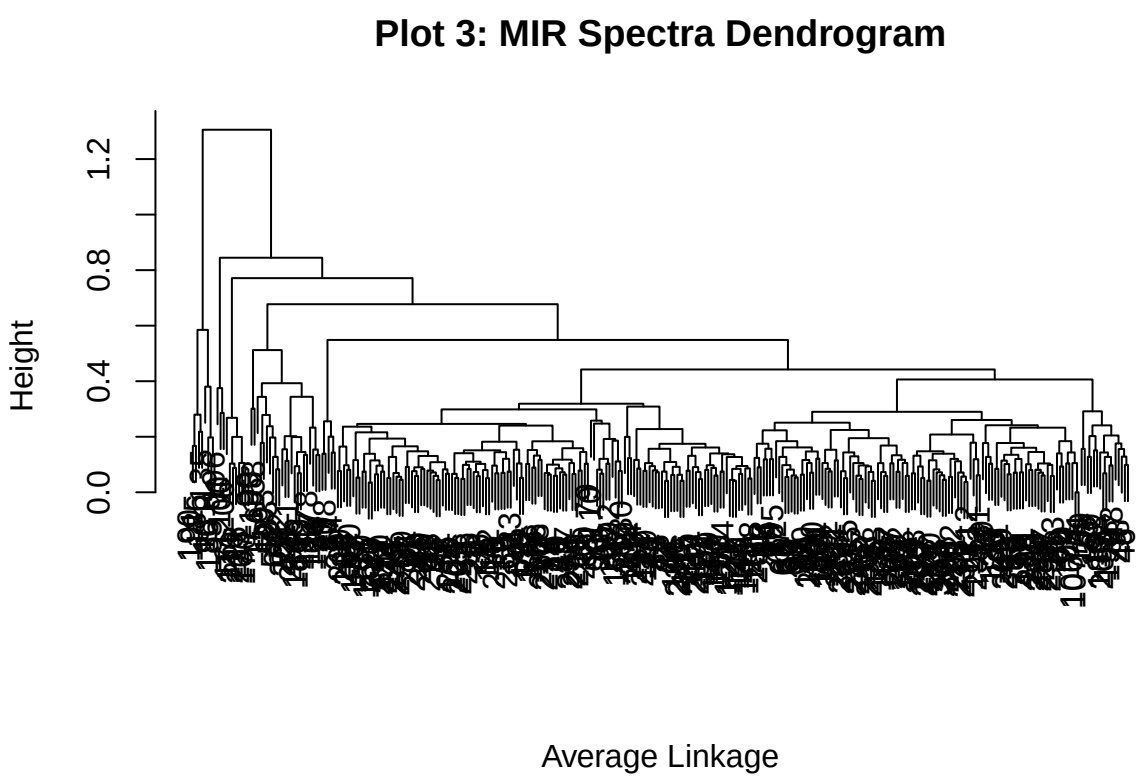
In question 3, I perform hierarchical and K-means clustering to the data to determine if there are any clusters of observations with similar MIR spectra. I subset the original data set to only include the numerical MIR spectra variables as these are the only variables I will use in each clustering method. I then perform hierarchical clustering on the data and plot the resulting dendrogram.

I have chosen not to standardise the spectra data for this task as I believe the spectra with higher variance will be more useful in identifying clusters and I do not wish to reduce their ability to partition the data. Similarly, spectra with lower variance will have a lessor effect on the clustering. I believe not standardising the data is a sensible approach here as there are no variables with drastically different values and all variables are absorbance values at different wavelengths of light (log10 of reciprocal of transmittance value) and were created using the same data generation process.

In terms of linakage method, I have used the average dissimilarity between groups to merge observations into clusters. I performed hierarchical clustering using the single linkage measure, however this led to significant "chaining" of observations to groups and as a result I ruled out using the "single" linkage method.

I have decided to use the "average" linkage method which calculates the average distance between groups and assigns observations to minimise this distance. I have used the euclidean distance as the metric to assess levels of disparity between observations / groups. I have plotted the associated dendrogram below.

```
# Extracting MIR spectra and creating new data frame
spectra_data <- data[,52:ncol(data)]

# Hierarchical clustering using euclidean distance measure and single linkage
hier_cl_average <- hclust(dist(spectra_data,method = "euclidean"),
                          method = "average")

# Plotting corresponding dendrogram
plot(hier_cl_average,
     xlab = "Average Linkage",
     sub = "",
     main = "Plot 3: MIR Spectra Dendrogram")
```



I will now cut the dendrogram horizontally to partition the spectra data into distinct clusters. I then compare this partition to the known breed of the cow which produced each sample of milk to see how well the hierarchical clustering aligns to the known "breed" covariate in the data.

I had previously partitoned the data into 7 clusters to match the distinct number of breeds present however, several of the clusters had very few observations and contained observations from many different breeds.

There was also a cluster containing all of the observations of certain breeds which indicates to me that these breeds produce similar milk absorbance values and so could be clustered together.

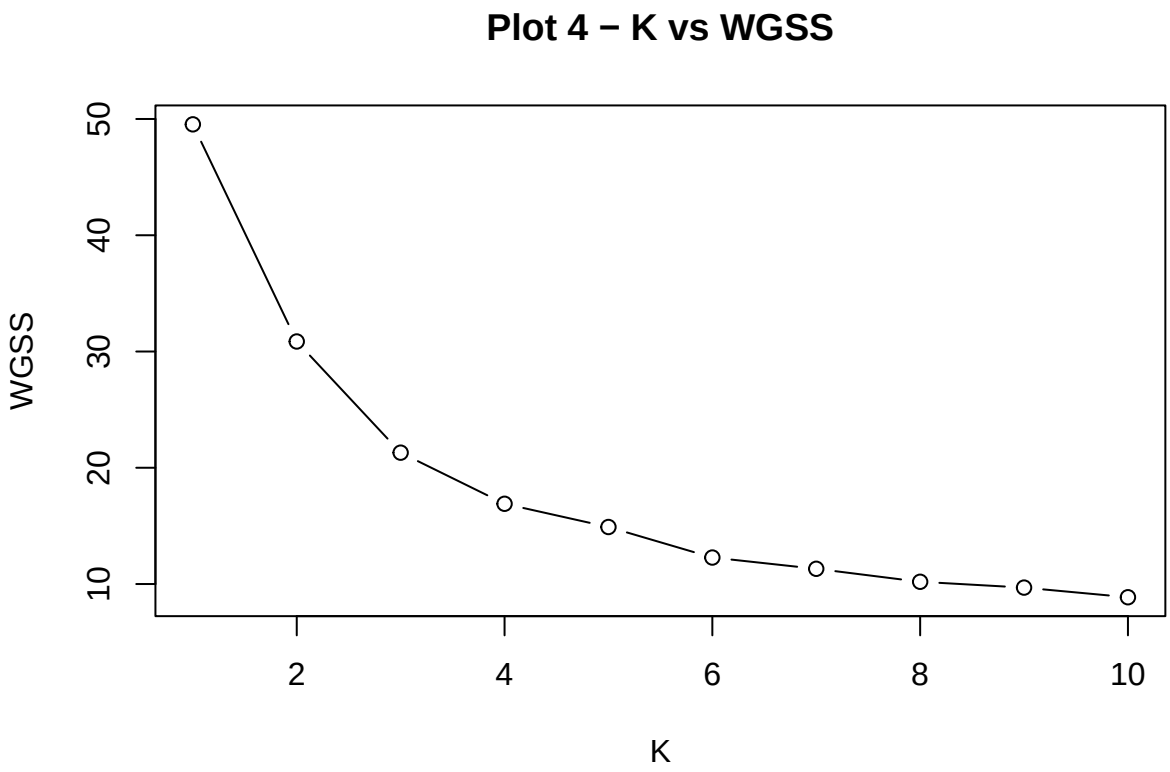
```
# cutting dendrogram to produce partitions / clusters
hier_cl_average_cut <- cutree(hier_cl_average,k = 3)

# comparing to known breed
table(data$Breed, hier_cl_average_cut)
```

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

We can see from the confusion table above that the hierarchical clusters do not correspond well to the different breeds. We can clearly see a dominant cluster (cluster 1) where 96% of the observations are partitioned into this cluster.

I will now perform K-means clustering on the MIR spectra data. I calculate and plot the within-group sum of squares (WGSS) for each k $\in [1, 10]$ below. There is no clear "elbow" observed in the plot however I believe a value of k = 3 is a sensible choice given the trade off in decreasing WGSS and increasing k.

```
# creating empty vector to store within-group sum of squares
wgss <- rep(0,10)

# calculating the wgss for the one cluster case
wgss[1] <- (nrow(spectra_data) - 1) * sum(apply(spectra_data, 2, var))

# calculating the wgss for k = 2,3,...,10
for (k in 2:10) {
  wgss[k] = sum(kmeans(spectra_data, centers = k)$withinss)
}

# plotting k versus wgss
plot(1:10, wgss, type = "b",
     xlab = "K",
     ylab = "WGSS",
     main = "Plot 4 - K vs WGSS")
```



I will now compare the hierarchical and K-means clustering solutions using the Rand index and ARI (adjusted Rand index).

```
# running K-means clustering
kmeans_cl <- kmeans(spectra_data, centers = 3)

# producing confusion matrix to compare two clustering methods
clust_comp_tab <- table(hier_cl_average_cut, kmeans_cl$cluster)
clust_comp_tab
```

```
##
## hier_cl_average_cut   1   2   3
##                   1 157  83  53
##                   2   0   0   8
##                   3   1   2   0
```

As we can see from the cross-tabulation above, there is quite a sizable difference between the K-means and hierarchical solutions. We observe much more spread clusters returned from the k-means solution than that of the hierarchical one. With the hierarchical solution the vast majority of observations are allocated to cluster 1 however in k-means these observations are dispersed across the 3 clusters.

```
# calculating Rand index and ARI for hierarchical vs k-means clustering
clust_comp_tab_res <- t(as.data.frame(classAgreement(clust_comp_tab),row.names = "Agreement"))
clust_comp_tab_res
```

```
##        Agreement
```

```
## diag  0.51644737
## kappa 0.01261628
## rand  0.42461351
## crand 0.03693318
```

I will now calculate the rand index and ARI to quantify the level of agreement between the two clustering solutions above. The Rand index between the two solutions is 0.425 however when we adjust for agreement by chance with the ARI we get an agreement of 0.037. This clearly suggests the level of agreement between the two methods is low.

Based on the above analyses it is difficult to ascertain any distinct structure in the MIR spectra data and I believe this is due to the high multi-collinearity present in the MIR spectra data. As each variable in the MIR spectra data is produced under the same data generating process with the only variance present being due to the different wavelengths of light being passed through the sample and the variability between the samples themselves it is not surprising the data is highly correlated.

Question 4

In question 4, I apply principal components analysis (PCA) to the spectral data. As in the clustering methods above,

I have decided against standardising the MIR spectra as I believe that the variance of each spectra is a measure of it's overall importance in the data set and this importance would be reduced if I was to use standard data. As all of the spectral data consist of absorbance values (all values lie between c.-0.12 and c.0.65) I don't believe there will be an issue regarding magnitudes of variables as mentioned above, each variable has been generated using the same process with the only difference being the wavelength of light used to measure the absorption levels at each wavelength.

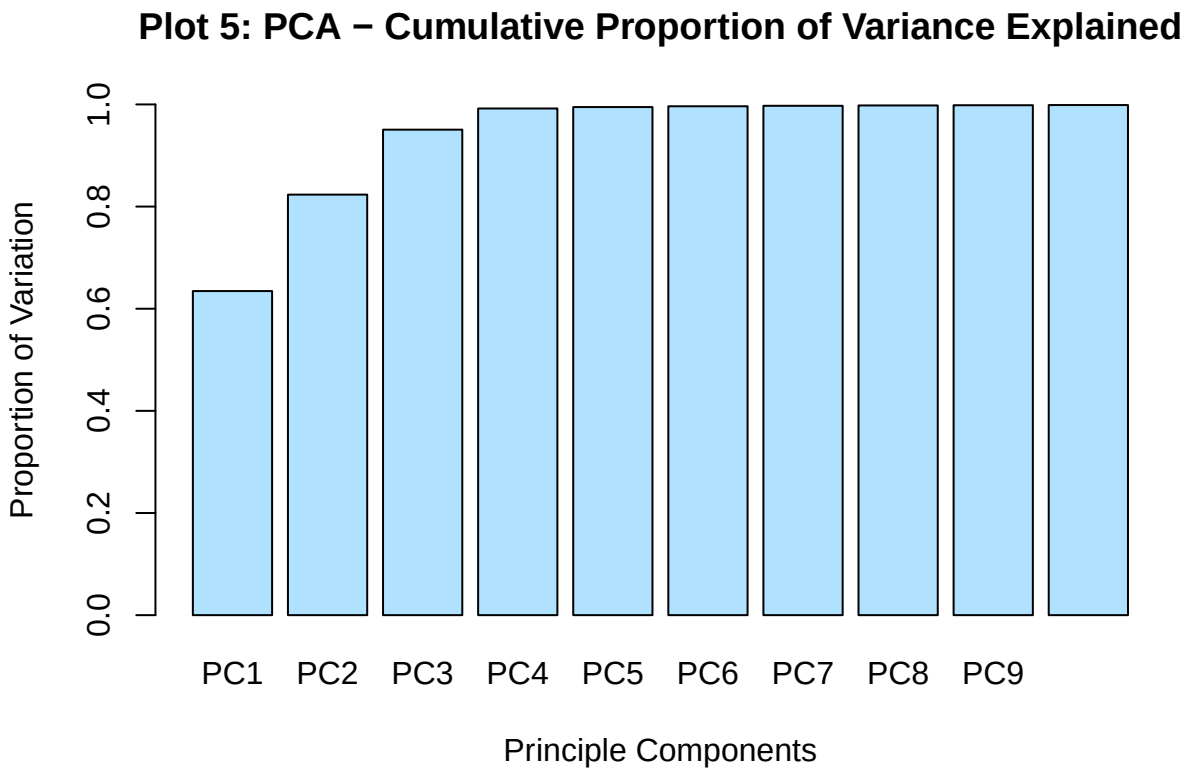
```r
# Applying PCA to the spectra data (only including top 10 components)
PCA_fit <- prcomp(spectra_data, rank. = 10)

# Printing summary of PCA fit
summary(PCA_fit)
```

```
## Importance of first k=10 (out of 304) components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     0.3221 0.1757 0.1442 0.08232 0.02136 0.01592 0.01256
## Proportion of Variance 0.6345 0.1889 0.1272 0.04144 0.00279 0.00155 0.00097
## Cumulative Proportion  0.6345 0.8234 0.9506 0.99204 0.99483 0.99638 0.99735
##                           PC8      PC9     PC10
## Standard deviation     0.01056 0.008316 0.007271
## Proportion of Variance 0.00068 0.000420 0.000320
## Cumulative Proportion  0.99803 0.998450 0.998770
```

I will now extract the cumulative proportion of the variance explained by the first 10 principal components and plot them.

```r
# Extracting cumulative proportion (for first 10 PCs only)
cum_prop <- cumsum(PCA_fit$sdev^2 / sum(PCA_fit$sdev^2))[1:10]

# Plotting cumulative proportion of variance explained by first 10 principle components
barplot(cum_prop[1:10],names.arg = paste0(rep("PC",10),1:10),ylim = c(0.0,1.0),
        col = "lightskyblue1",
```

```
        main = "Plot 5: PCA - Cumulative Proportion of Variance Explained",
        xlab = "Principle Components",
        ylab = "Proportion of Variation")
```



As we can see from the above plot of the cumulative proportion of variance explained by each of the first 10 principle components, (and summary table produced previously) 63% of the variance is explained by the first component, a further 19% explained by the second component and another 13% by the third. In total, 99.8% of the total variance is explained by the first ten principle components.

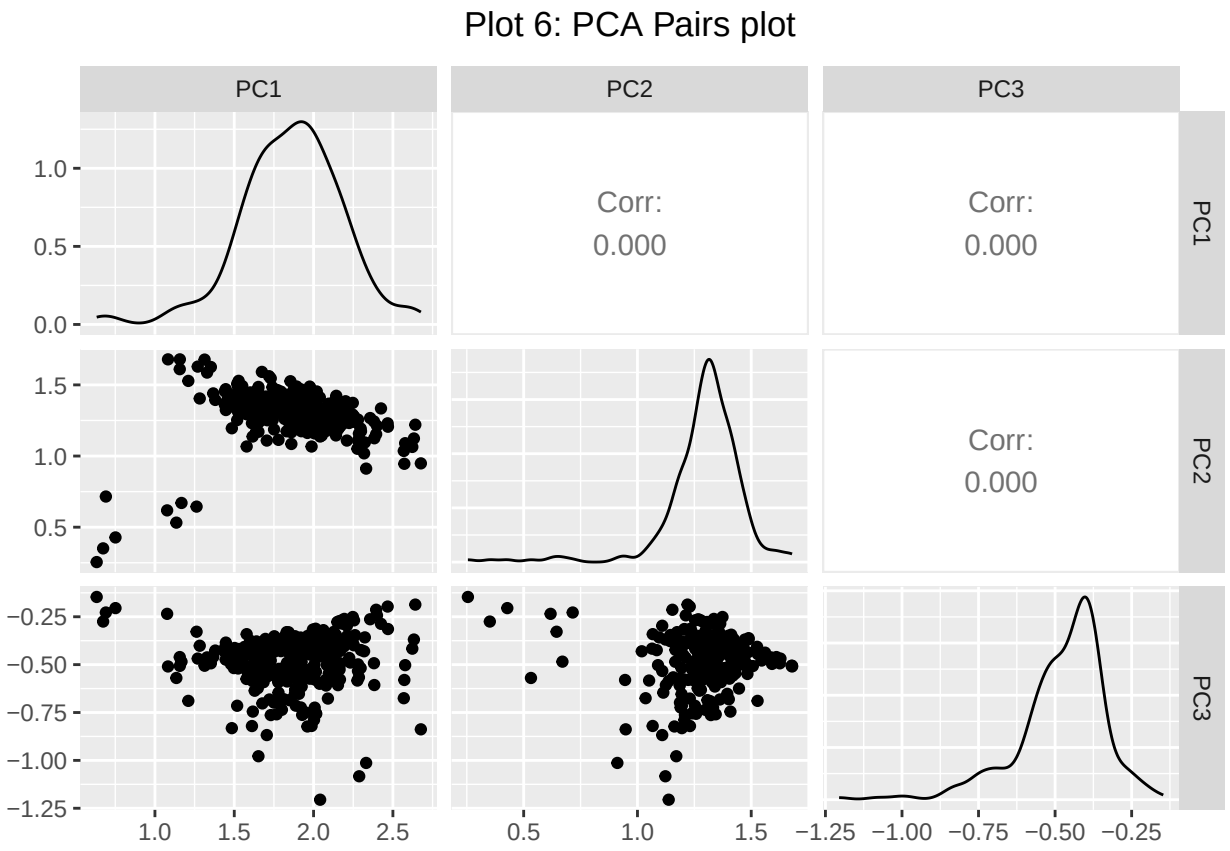
I believe the first 3 principle components can be used to sufficiently represent the data as over 95% of the variation within the spectra data is explained by these 3 variables. Only using the first 2 components can effectively account for 82% of variation, however I do not believe this is sufficiently high to warrant removal of the 3rd principle component. Similarly, adding the 4th component only explains a further 4% of variation where we have already accounted for 95% of the variation in the data and therefore I believe 3 components are sufficient.

Question 5

In question 5, I derive the principal component scores for the observations from first principles without the use of inbuilt functions such as predict. I then plot the principal component scores for the milk samples. I only plot the first 3 principle components as these reflect 95% of the variability in the data and so the remaining components are ignored.

```
# Extracting eigenvectors from prcomp
PCA_eigenvectors <- as.matrix(PCA_fit$rotation)
```

```r
# calculating manual scores (non-centered)
manual_scores <- as.matrix(spectra_data) %*% PCA_eigenvectors

# calculating manual scores (centered)
manual_scores_centered <- as.matrix(sweep(spectra_data,2,apply(spectra_data,2,mean),"-")) %*% PCA_eigen

# plotting top 3 principal component (non-centered) scores using pairs plot
ggpairs(as.data.frame(manual_scores[,1:3])) +
  labs(title = expression("Plot 6: PCA Pairs plot")) +
  theme(plot.title = element_text(hjust = 0.5))
```



As we can see in the above pairs plot of the first 3 principle components, each have zero correlation with each other which is by design of PCA. As expected, we observe the greatest variability in the distribution for PC1. For PC1 vs PC2, we can see the majority of data gathered at relatively high values for both components, with only a few observations at low values. For PC1 vs PC3 we observe more dispersion in values compared to PC1 vs PC2 with values gathering in a similar location for PC1. Finally, in PC2 vs PC3 we see less dispersion that the previous two pairs plots, with the majority of data gathering in relatively high values for the two components. We also observe the distribution of each of the components in the plots on the diagonal.

Question 6

Principal components regression (PCR) is a type of regression analysis which uses a mixture of multiple linear regression and principal components analysis (PCA). The goal of PCR is the same as in any other regression

analysis, which is to model the (linear) relationship between a target variable, Y and predictor variables $X_1, ..., X_p$. In contrast to multiple linear regression where the predictor variables are original or transformed original features of the data set, in PCR the predictor variables are the first k principal components obtained from principal components analysis.

The primary idea behind PCR is that by using the first k principle components where $k < N$, that explain most of the variability in the data, and assumedly are associated with the variability in the target variable Y we can mitigate problems such as overfitting and multicollinearity. The assumption underpinning PCR is that the principle components from PCA, which represent the directions of highest variance in the predictor variables, are associated with the target variable Y. This assumption may not always be true however, in many cases the principal components are a sufficient enough approximation to yield good results.

Principal components regression can be thought of as a 3 stage process as described below.

1. Principal components analysis (PCA) is performed on the data, X which contains N variables, to produce N principal components where each principal component or score (Z_q) is calculated by the below formula.

$$Zq = a_q^T x_i$$

where a_q is the eigenvector corresponding to the q^{th} highest eigenvalue (a_1 is the eigenvector corresponding to the largest eigenvalue of data X) and x_i is the i^{th} row of data set X.

2. The first k principal components $(Z_1, ..., Z_k)$ that explain most of the variance of X are kept where $(k < N)$. k is chosen and then validated using methods such as the bootstrap or the jackknife.

3. Finally, the target variable Y is regressed on the first k principle components $Z_1, ..., Z_k$ using ordinary least squares and since each Z_q are orthogonal by design of PCA, the regression can be expressed by the below equation.

$$\hat{y} = \alpha + \sum_{k=1}^{k} \hat{\beta}_k Z_k$$

This produces the least-squares line based on the first k principal components and results in a reduced regression compared to that obtained with multiple least squares.

When using implementing PCR we must choose an appropriate value for k, which is the number of principal components to regress our target variable Y against. An appropriate value of k can be determined and validated using methods such as the bootstrap and jackknife as mentioned above. We also need to determine whether or not to scale the data before implementing the model as the principal components will be greatly affected by variables with variances of larger magnitude.

The main advantages of using PCR over MLR is that by using the PCR we perform the regression using a reduced number of predictor variables which overcomes overfitting problems. Also, by using the principal components from PCA as our predictor variables, which are orthogonal by design, we greatly reduce issues with multicollinearity which occurs when predictor variables are highly correlated with one another.

There can also be a number of disadvantages with implementing PCR over MLR. If the primary assumption underpinning PCR which is that the principal components which represent most of the variability in the data, can describe also describe the observed variance in the target variable. As we remove principal components with low variance (low eigenvalues), we may remove information that while low in variance, might better describe the target variable and this can result in poor performance.

Another disadvantage in using PCR is that we use the principal components as explanatory variables in the model and as a result lose a lot of the interpretability associated with MLR. It is much more difficult to link the fitted coefficients back to the original features of the data set.

Question 7

In question 7, I use principal components regression (PCR) using the pls R package to predict the β Lactoglobulin b values from the spectra data for a test set which is one third the size of the original data set.

As in my prior analyses above I have decided not to scale or center the spectra data before performing the PCR regression. The reason for this is that the values in the spectra data are relatively similar in terms of their variance and were produced using the same data generating process with the only difference between values being the different wavelengths of light shone through the sample. In this way I believe by centering and scaling the data I would lose valuable information which could better explain the values of β Lactoglobulin b and therefore result in a poorer out of sample model fit.

I have also chosen to use the first 3 principal components in the PCR model as they collectively explain 95% of the variance in the spectra values. I believe using any more than 3 principal components could result in over-fitting and poorer out of sample performance.

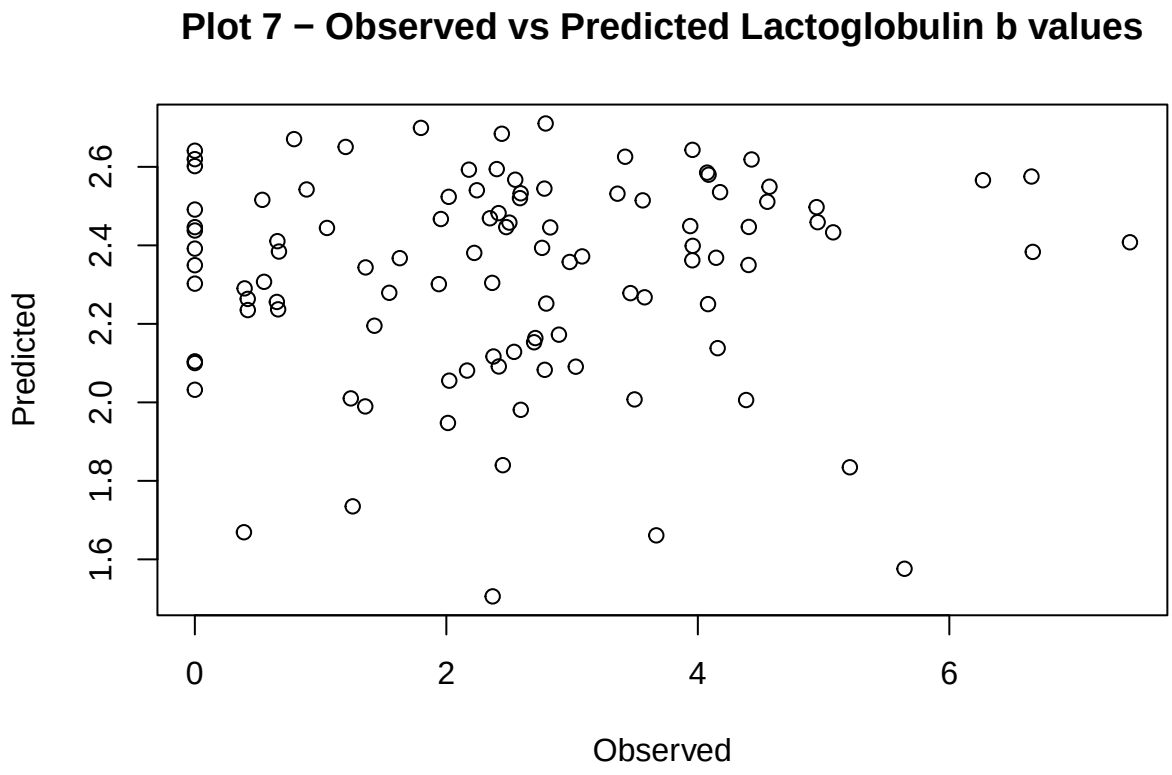
```
# setting seed for reproducibility
set.seed(101)

# creating dataset to use in model
pcr_data <- data[,13:ncol(data)] %>%
  select(-colnames(data)[14:51])

# splitting into training vs test data
train_obs <- sample(c(TRUE, FALSE), nrow(pcr_data), replace=TRUE, prob=c(0.67,0.33))
pcr_train <- pcr_data[train_obs,]
pcr_test <- pcr_data[!train_obs,]

# fitting PCR model to predict Lactoglobulin b values from spectra
pcr_fit <- pcr(beta_lactoglobulin_b~.,
               data = pcr_train,
               validation = "CV",
               ncomp = 3)

# printing summary of model
summary(pcr_fit)
```

```
## Data:    X dimension: 201 531
##  Y dimension: 201 1
## Fit method: svdpc
## Number of components considered: 3
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps
## CV           1.659    1.672    1.666    1.656
## adjCV        1.659    1.671    1.665    1.655
##
## TRAINING: % variance explained
##                         1 comps  2 comps  3 comps
## X                     63.16295   81.346   94.839
## beta_lactoglobulin_b   0.01006    1.699    2.398
```

I have outputted the summary of the model fit on the training dataset above. The training dataset consists

of 201 rows (c. one third) of the original spectra data. We observe that roughly 95% of the variance in X is explained by the three principal components however only 2.4% of the variance of the β Lactoglobulin b values are explained by the principal components. This indicates that the principal components will not perform very well in predicting the β Lactoglobulin b values in the test set which I carry out below.

```
# predicting Lactoglobulin b values with PCR model
pcr_pred <- predict(pcr_fit,pcr_test,ncomp = 3)

# calculating RMSE for predicted vs observed values
pcr_pred_rmse <- sqrt(mean((pcr_pred - pcr_test$beta_lactoglobulin_b)^2))

# plotting predicted Lactoglobulin b values against fitted ones
plot(pcr_test$beta_lactoglobulin_b,pcr_pred,
     main = "Plot 7 - Observed vs Predicted Lactoglobulin b values",
     xlab = "Observed",
     ylab = "Predicted")
```



I have used the PCR model fit on the training data set to predict values for the β Lactoglobulin b protein. I have calculated the root mean squared error (RMSE) of the model which represents the average distance between the predicted values of Lactoglobulin b and the observed ones. The RMSE for the PCR model is 1.7205838 which indicates that on average, the predicted values are 1.7205838 away from the observed Lactoglobulin b values.

I have also produced a scatter plot of the predicted Lactoglobulin b values against the observed values and we observe that the predicted values are only between c.1.5 and 2.7 however we observe values of Lactoglobulin b as low as 0 and as high as c7.4 in the test data set. The high RMSE and dispersion in the plot indicate a poor model fit and subsequent poor out of sample model performance.

Question 8

In question 8, I use principal components analysis on the seven milk proteins data to impute values of β Lactoglobulin b that are equal to 0.

```
# subsetting data to include only 7 milk proteins
X <- data %>%
  select(kappa_casein, alpha_s2_casein, alpha_s1_casein, beta_casein,
         alpha_lactalbumin, beta_lactoglobulin_a, beta_lactoglobulin_b)

# creating function to approximate matrices using PCA
fit_prcomp <- function(X, M = 3){
  prcomp_dob <- prcomp(X, center = FALSE, scale = FALSE)
  prcomp_dob$x[, 1:M, drop=FALSE] %*% t(prcomp_dob$rotation[, 1:M, drop = FALSE])
}

# step 1 - Replace 0 values with column means of non-missing values (call Xhat)

# create dataframe which is a copy of X only with NA values in place of 0
Xna <- X
# create vector of indices where Xna is 0 (i.e. values we are imputing)
index_na <- Xna$beta_lactoglobulin_b == 0
# converting zero values to NA
Xna$beta_lactoglobulin_b[index_na] <- NA
# creating a dataframe Xhat which contains mean of lactoglobulin values in
# place of zero / NA values
Xhat <- Xna
Xhat$beta_lactoglobulin_b[index_na] <- mean(Xna$beta_lactoglobulin_b,
                                            na.rm = TRUE)
# calculating column means
xbar <- colMeans(Xna, na.rm = TRUE)

# step 2 - setting up initial conditions

# setting threshold which will be used to stop loop
thresh <- 1e-7
# initial value for relative error
rel_err <- 1
# setting initial iteration number
iter <- 0
# index of values which are missing (values we will impute)
ismiss <- is.na(Xna)
# mean of squared non-missing elements
mss0 <- mean(Xna[!ismiss]^2)
# mean squared error of non-missing elements of prior version of xhat
mssold <- mean((scale(Xna, xbar,FALSE)[!ismiss])^2)

# step 3 - iteration to implement algorithm 12.1 from ISLR v2

while (rel_err > thresh) {
  # increase iteration by 1
  iter <- iter + 1
  # step 2(a) - approximating X using function from step 1
    Xapp <- fit_prcomp(Xhat, M = 3)
```

```
  # step 2(b) - setting value for each missing obs to approximation Xapp
  Xhat[ismiss] <- Xapp[ismiss]
  # step 2(c) - computing the objective
  mss <- mean(((Xna[!ismiss] - Xapp[!ismiss])^2))
  rel_err <- (mssold-mss) / mss0
  mssold <- mss

  cat("Iter:",iter,"MSS:",mss,"Rel. Err",rel_err,"\n")
}
```

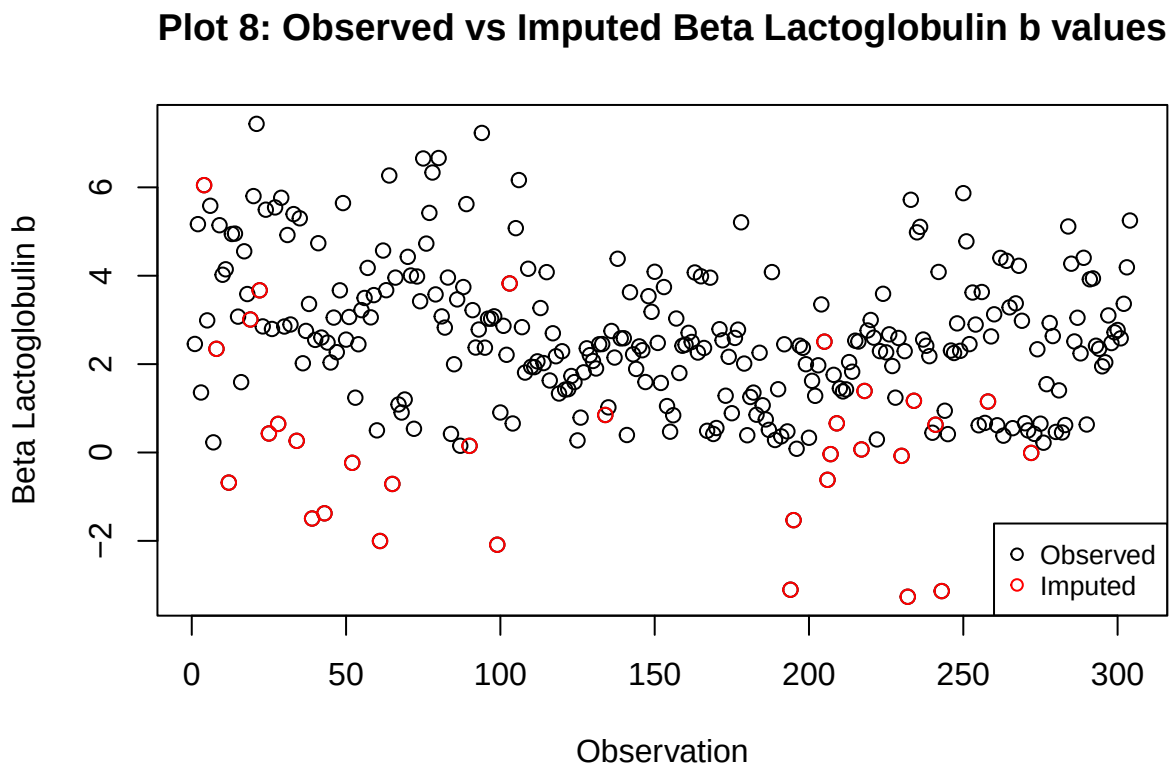
```
## Iter: 1 MSS: 0.4092991 Rel. Err 0.0646578
## Iter: 2 MSS: 0.4017937 Rel. Err 0.0001186334
## Iter: 3 MSS: 0.3970004 Rel. Err 7.57634e-05
## Iter: 4 MSS: 0.3940252 Rel. Err 4.702787e-05
## Iter: 5 MSS: 0.3921473 Rel. Err 2.968261e-05
## Iter: 6 MSS: 0.3909259 Rel. Err 1.930569e-05
## Iter: 7 MSS: 0.3901067 Rel. Err 1.294896e-05
## Iter: 8 MSS: 0.3895418 Rel. Err 8.92874e-06
## Iter: 9 MSS: 0.3891429 Rel. Err 6.304691e-06
## Iter: 10 MSS: 0.3888555 Rel. Err 4.542308e-06
## Iter: 11 MSS: 0.3886449 Rel. Err 3.328667e-06
## Iter: 12 MSS: 0.3884884 Rel. Err 2.474636e-06
## Iter: 13 MSS: 0.3883706 Rel. Err 1.862356e-06
## Iter: 14 MSS: 0.388281 Rel. Err 1.416284e-06
## Iter: 15 MSS: 0.3882122 Rel. Err 1.086758e-06
## Iter: 16 MSS: 0.388159 Rel. Err 8.403804e-07
## Iter: 17 MSS: 0.3881176 Rel. Err 6.542323e-07
## Iter: 18 MSS: 0.3880852 Rel. Err 5.122993e-07
## Iter: 19 MSS: 0.3880597 Rel. Err 4.032101e-07
## Iter: 20 MSS: 0.3880396 Rel. Err 3.187735e-07
## Iter: 21 MSS: 0.3880236 Rel. Err 2.530127e-07
## Iter: 22 MSS: 0.3880108 Rel. Err 2.015164e-07
## Iter: 23 MSS: 0.3880006 Rel. Err 1.60995e-07
## Iter: 24 MSS: 0.3879925 Rel. Err 1.289725e-07
## Iter: 25 MSS: 0.3879859 Rel. Err 1.035695e-07
## Iter: 26 MSS: 0.3879806 Rel. Err 8.334932e-08
```

I have imputed values for β Lactoglobulin b which were previously equal to zero using the principal components analysis algorithm as described in section 12.3 in An Introduction to Statistical Learning with Applications in R by James et al. (2021). I used the function 'prcomp' to fit principal component analysis to the seven milk proteins data. The observations which we are required to impute are then set to the principal component values as per step 2(b) and the mean square error (objective) is calculated. This process is repeated iteratively until the mean square error falls below a threshold.

We can see from the ouput above, it took the 26 iterations for the algorithm to cease (objective to fall below the threshold) and for the final imputed values of β Lactoglobulin b to be estimated. I have plotted the observed and imputed values of the β Lactoglobulin b protein below for comparison.

```
plot(Xhat$beta_lactoglobulin_b,
     xlab = "Observation",
     ylab = "Beta Lactoglobulin b",
     main = "Plot 8: Observed vs Imputed Beta Lactoglobulin b values")
points(x = as.data.frame(1:nrow(Xhat))[index_na,],
```

```
       y = Xhat$beta_lactoglobulin_b[index_na], col = 'red')
legend("bottomright", legend = c("Observed","Imputed"),
       col = c("black","red"), pch = 1,cex = 0.8)
```



As we can see from the plot above, the imputed values for the β Lactoglobulin b protein are much different to that of the observed values. We can see that some of the imputed values are actually negative and in the context of the milk proteins data set negative g/L values do not make sense. In this way, the imputed values do not correspond well to the original observed data.

Question 9

In question 9, I use PCR to predict β Lactoglobulin b values from the MIR spectra for a test set where we have three distinct training sets which each contains the following:

(a). all observations with an observed, non-zero value of β Lactoglobulin b. (b). all observations but where 0 values of β Lactoglobulin b are imputed using the observed mean. (c). all observations but where 0 values of β Lactoglobulin b are imputed using principal components analysis.

As in question 7 above I have decided not to center or scale the data and I have also decided to use 3 principal components in each regression.

```
# setting seed for reproducibility
set.seed(101)

# creating dataset to use in model
pcr_data <- data[,13:ncol(data)] %>%
```

```
  select(-colnames(data)[14:51])

# creating column to contain imputed values
pcr_data$beta_lactoglobulin_b_impute <- Xhat$beta_lactoglobulin_b

# number of iterations to run
iterations <- 100

# creating empty data frame to store rmse values for each scenario
res_rmse <- as.data.frame(matrix(nrow = iterations,ncol = 3))
colnames(res_rmse) <- c("Scenario_A","Scenario_B","Scenario_C")

# for loop
for (i in 1:iterations) {

# splitting into training vs test data
train_obs <- sample(c(TRUE, FALSE), nrow(pcr_data), replace=TRUE, prob=c(0.67,0.33))
pcr_train <- pcr_data[train_obs,]
pcr_test <- pcr_data[!train_obs,]

# index of zero values in training set
pcr_train_zero <- pcr_train$beta_lactoglobulin_b == 0
# index of zero values in test set
pcr_test_zero <- pcr_test$beta_lactoglobulin_b == 0
# subsetting test set to remove observations that have zero lactoglobulin values
pcr_test <- pcr_test[!pcr_test_zero,]

#-------------------------------------------------------------------------------

# performing prediction for scenario (a)
pcr_train_a <- pcr_train[!pcr_train_zero,] %>%
  select(-beta_lactoglobulin_b_impute)

# fitting PCR model for scenario (a)
pcr_fit_a <- pcr(beta_lactoglobulin_b~.,
                 data = pcr_train_a,
                 validation = "CV",
                 ncomp = 3)

# predicting Lactoglobulin b values for scenario (a)
pcr_pred_a <- predict(pcr_fit_a,pcr_test,ncomp = 3)

# calculating RMSE for predicted vs observed values for scenario (a)
pcr_pred_a_rmse <- sqrt(mean((pcr_pred_a - pcr_test$beta_lactoglobulin_b)^2))
res_rmse$Scenario_A[i] <- pcr_pred_a_rmse

#-------------------------------------------------------------------------------

# performing prediction for scenario (b)
pcr_train_b <- pcr_train %>%
  select(-beta_lactoglobulin_b_impute)

# imputing zero values of lactoglobulin b with observed mean (less zero values)
```

```
pcr_train_b$beta_lactoglobulin_b[pcr_train_zero] <- mean(pcr_train_b$beta_lactoglobulin_b[!pcr_train_ze

# fitting PCR model for scenario (b)
pcr_fit_b <- pcr(beta_lactoglobulin_b~.,
                 data = pcr_train_b,
                 validation = "CV",
                 ncomp = 3)


# predicting Lactoglobulin b values for scenario (b)
pcr_pred_b <- predict(pcr_fit_b,pcr_test,ncomp = 3)

# calculating RMSE for predicted vs observed values for scenario (b)
pcr_pred_b_rmse <- sqrt(mean((pcr_pred_b - pcr_test$beta_lactoglobulin_b)^2))
res_rmse$Scenario_B[i] <- pcr_pred_b_rmse

#------------------------------------------------------------------------------

# performing prediction for scenario (c)
pcr_train_c <- pcr_train
# imputing zero values of lactoglobulin b with imputed values from q8
pcr_train_c$beta_lactoglobulin_b[pcr_train_zero] <- pcr_train_c$beta_lactoglobulin_b_impute[pcr_train_z
# dropping imputed column
pcr_train_c <- pcr_train_c %>%
  select(-beta_lactoglobulin_b_impute)

# fitting PCR model for scenario (c)
pcr_fit_c <- pcr(beta_lactoglobulin_b~.,
                 data = pcr_train_c,
                 validation = "CV",
                 ncomp = 3)

# predicting Lactoglobulin b values for scenario (c)
pcr_pred_c <- predict(pcr_fit_c,pcr_test,ncomp = 3)

# calculating RMSE for predicted vs observed values for scenario (c)
pcr_pred_c_rmse <- sqrt(mean((pcr_pred_c - pcr_test$beta_lactoglobulin_b)^2))
res_rmse$Scenario_C[i] <- pcr_pred_c_rmse
}

# printing RMSE results
colMeans(res_rmse)
```

```
## Scenario_A Scenario_B Scenario_C
##   1.554562   1.552468   1.575097
```

I have predicted the β Lactoglobulin b values from the MIR specrta above for each of the scenarios outlined in the question. I have fit each of the models 100 times to ensure results are statistically robust. I have calculated the root mean square error of each of the scenarios for each iteration and store these in a dataframe.

We can see from the average RMSE above that the best performing model on average is scenario B which is where the zero-values for the Lactoglobulin b protein are imputed using the observed mean in the training set. The next best performing is the scenario A model where the training set drops any of the zero values for the Lactoglobulin b protein and the worst performing model is scenario C where the zero values were

imputed using PCA. I believe one reason for the poor performance of the PCA imputed values is due to some of the imputed values being negative and this will negatively effect the out of sample performance of the model.