Assignment

STAT40150 – Multivariate Analysis

Meghashree Madhava Rao Ramachandrahosur (21200301)

Load the necessary libraries

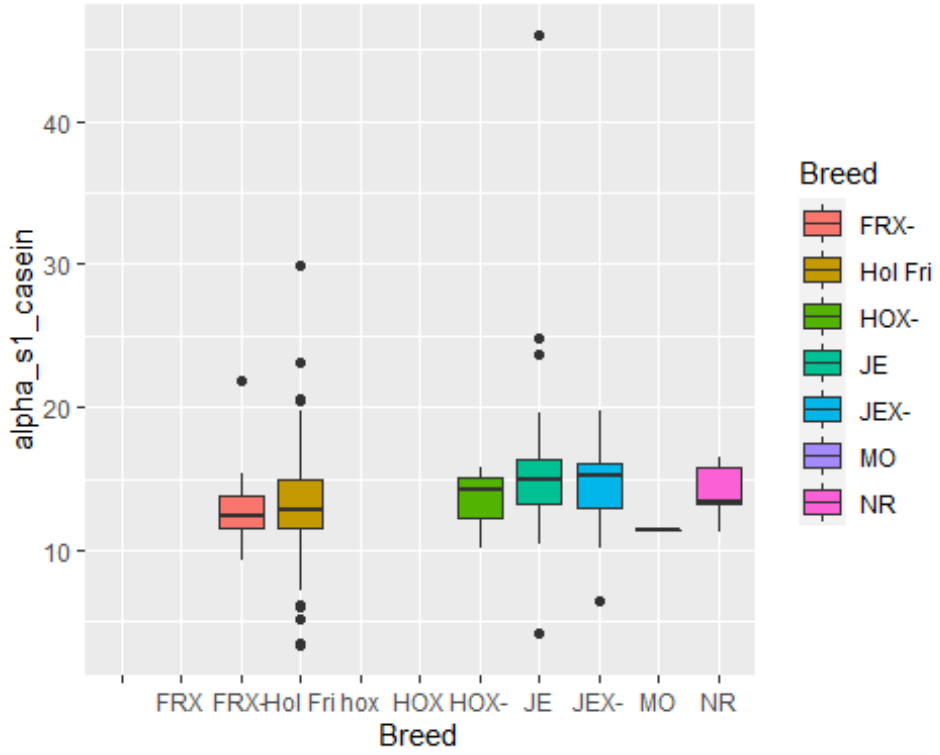
```
# set the working directory path
setwd("C:/Users/DELL/OneDrive/Documents/SEM-2/Multivariative")

# load the required libraries
library(ggplot2)
library(ggthemes)
library(tidyr)
library(e1071)
library(pls)
library(plyr)

# Libraries to plot k-means cluster
library(ggpubr)
library(factoextra)
```

Question 1

```
# Set the seed value
set.seed(21200301)

# import the data set
df <- read.csv("Milk_MIR_Traits_data.csv", header = TRUE, check.names = F)
colnames(df)[1] <- "Breed"
n <- nrow(df)

del_row <- sample(1:n, 1, replace=F)

# delete a random row
df <- df[-c(del_row),]
```

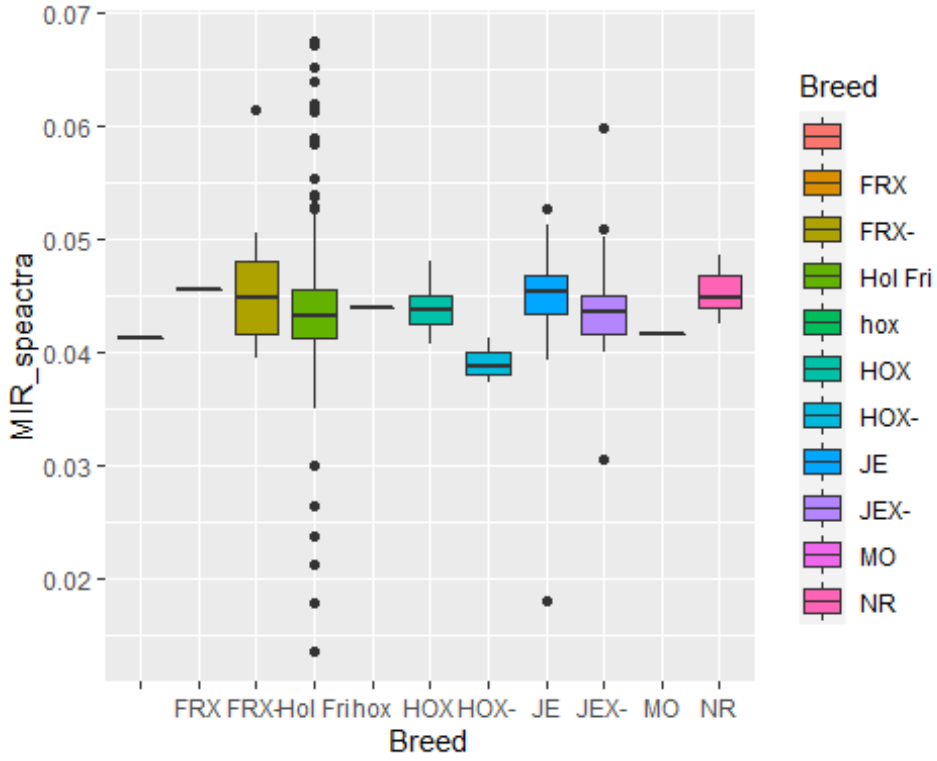
Question 2

```
# Plot for protein trait aplha s1 casein

df$alpha_s1_casein <- unlist(df$alpha_s1_casein)
p<-ggplot(df, aes(x=Breed, y=alpha_s1_casein, fill=Breed)) +
  geom_boxplot()
p
```

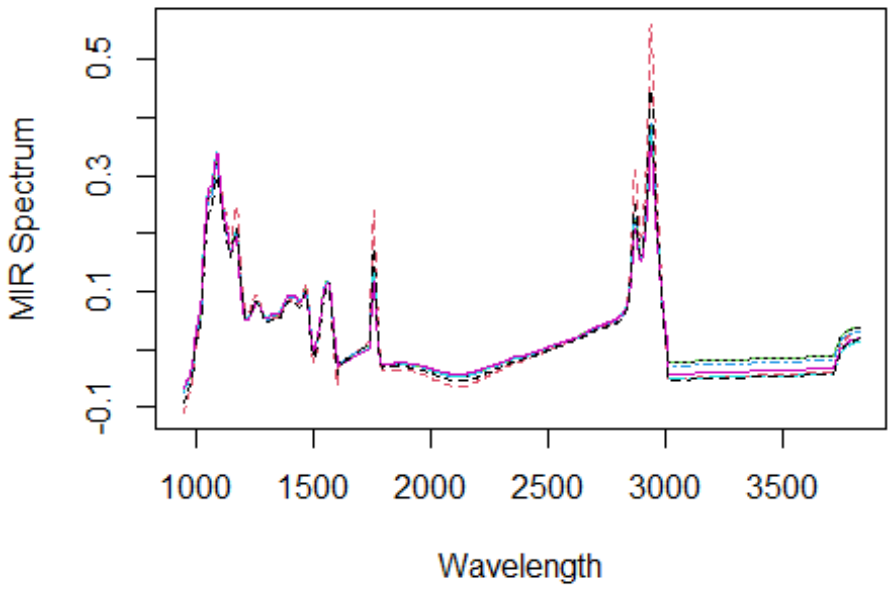



```
which(colnames(df)=="X941")

## integer(0)

MIR_speactra <- rowSums(df[,52:ncol(df)], na.rm = T)/531
df1 <- cbind(df,MIR_speactra)

# plot of MIR specta
p<-ggplot(df, aes(x=Breed, y=MIR_speactra, fill=Breed)) +
  geom_boxplot()
p
```

The above is a plot of Protein train α_{s1} – casein against the breed. We can observe many outliers with respect to the breed "Hol Fri" and a large value for 1 observation of breed type "JE". Even with the outliers, the distribution of the protein for mostly all breeds are not skewed.




```
t <- table(df$Breed)
df2 <- df[df$Breed=="FRX-",]
dff <- colMeans(df2[,c(52:582)], na.rm = T)

for (name in rownames(t)[2:7]) {
  df2 <- df[df$Breed==name,]
  mn <- colMeans(df2[,c(52:582)], na.rm = T)
  dff <- rbind(dff, mn)
}
dff <- cbind(dff, "Breed"= rownames(t))
dff <- data.frame(dff, check.names = F)
y <- as.numeric(colnames(dff)[1:531])

matplot(y=t(dff[,1:531]),x=y, type = 'l',xlab = "Wavelength", ylab = "MIR Spe
ctrum")
```

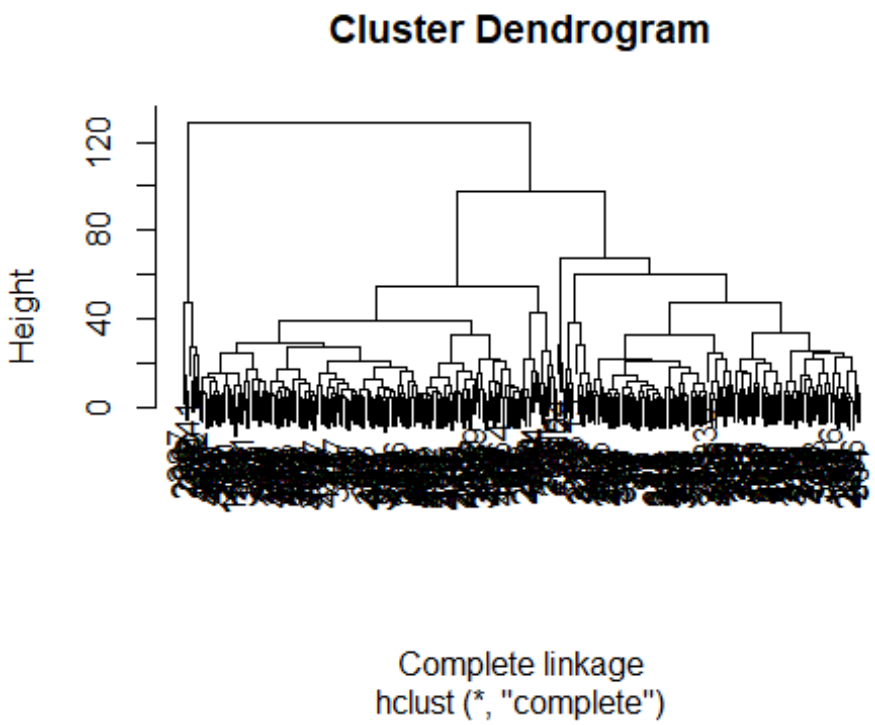


The MIR spectrum of mostly all breeds are the same with slightly higher variations after the wavenumber of 3000 cm^{-1}. 'HolFri' breed has high variation in data for both the traits and MIR spectrum

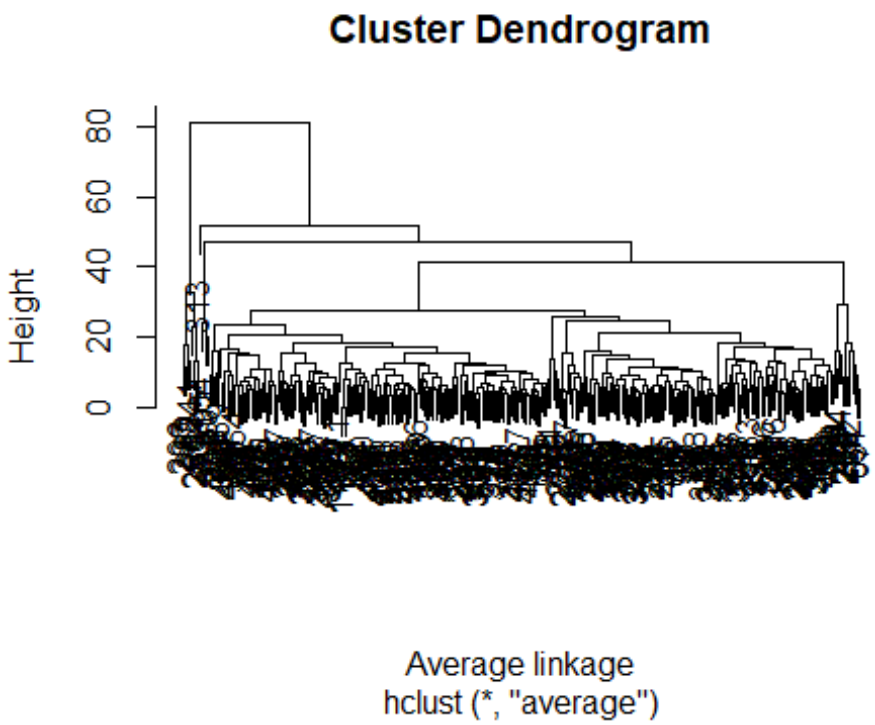
```
# Remove points from data with > 3 sd from mean
df$alpha_s1_casein <- lapply(df$alpha_s1_casein, function(x) replace(x,abs(sc
ale(x))>3,NA))

df <- df[!is.na(df$alpha_s1_casein),]
```

Question 3

```
x = df1[1:30,52:582]

dist.eucl = dist(x, method="euclidean")

MI <- data.frame(scale(df[,52:582]), check.names = F)
cl.complete = hclust(dist(MI),method="complete")
plot(cl.complete, xlab="Complete linkage")
```



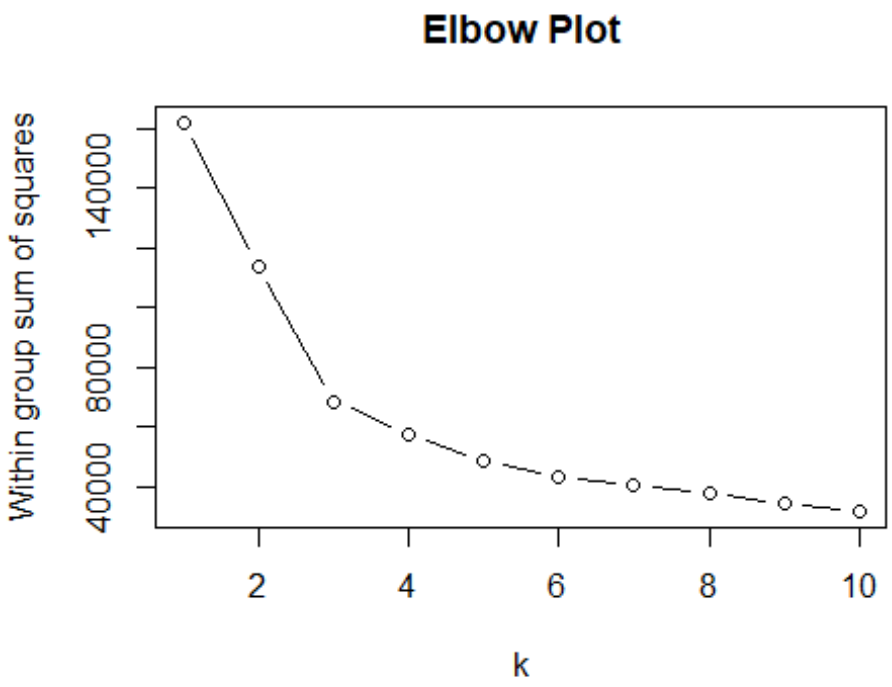

```
hcl = cutree(cl.complete, k = 4)

table(hcl)

## hcl
##   1   2   3   4
## 134 161   8   3

table(hcl, df[,1])

##
## hcl FRX- Hol Fri HOX- JE JEX- MO NR
##   1    9      95    1 18    7  1  3
##   2    6      92    2 40   15  0  6
##   3    0       6    0  1    1  0  0
##   4    1       1    0  0    1  0  0

cl.avg = hclust(dist(MI),method="average")
plot(cl.avg, xlab="Average linkage")
```



In the above plots we visualize both complete and average linkage in hierarchical clustering. K-means is more widely used method of clustering. We can observe very different clusters for both the hierarchical clustering forms. The complete linkage indicates the presence of 3 clusters in the data whereas the average linkage shows a 4-cluster model. Hence the tree is cut at k = 4, such that we obtain 4 cluster solution.

```
hcl = cutree(cl.avg, k = 4)

table(hcl)

## hcl
##   1   2   3   4
## 294   8   3   1

table(hcl, df[,1])

##
## hcl FRX- Hol Fri HOX-  JE JEX-  MO  NR
##   1   15     186    3  58   22   1   9
##   2    0       6    0   1    1   0   0
##   3    1       1    0   0    1   0   0
##   4    0       1    0   0    0   0   0

# K-means
WGSS = rep(0,10)
n = nrow(MI)
WGSS[1] = (n-1) * sum(apply(MI, 2, var))
```

```
for(k in 2:10)
{
  WGSS[k] = sum(kmeans(MI, centers = k)$withinss)
}

plot(1:10, WGSS, type="b", xlab="k", ylab="Within group sum of squares", main
= "Elbow Plot")
```

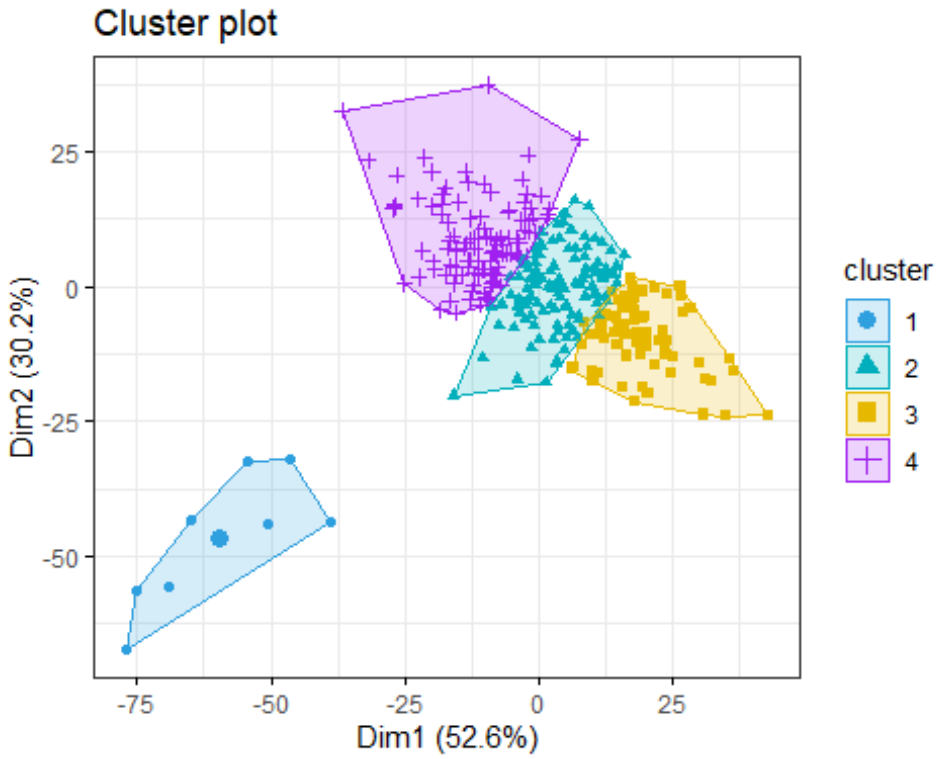


Before employing K-means clustering method, we plot the 'Elbow Plot' , a plot of Within group sum of squares versus the k value. On carefully looking at it, I have decided to continue assuming that there are 4 good groups in the MIR spectral data.

The function 'classAgreement' is used to see the agreement between the two clustering solutions obtained from different methods. We get an approval of 56%, indicating a good cluster set.

```
# K-means algorithm
k = 4
km = kmeans(MI, center=k)
table(km$cluster)

##
##   1   2   3   4
##   8 118  67 113
```

```
# Comparing
tab <- table(hcl,km$cluster)
classAgreement(tab)$rand

## [1] 0.5590496

# Plot the clusters
fviz_cluster(km, data = MI, palette = c("#2E9FDF", "#00AFBB", "#E7B800", "pur
ple", "red"), geom = "point", ellipse.type = "convex", ggtheme = theme_bw())
```



The K-means cluster is now obtained and plotted above with MIR data. It has a clear cut 4 huge cluster sets. The 'HolFri' with 'JE' breed forms a bigger cluster. This is an indication to obtain data collection from other breed to incorporate variability.

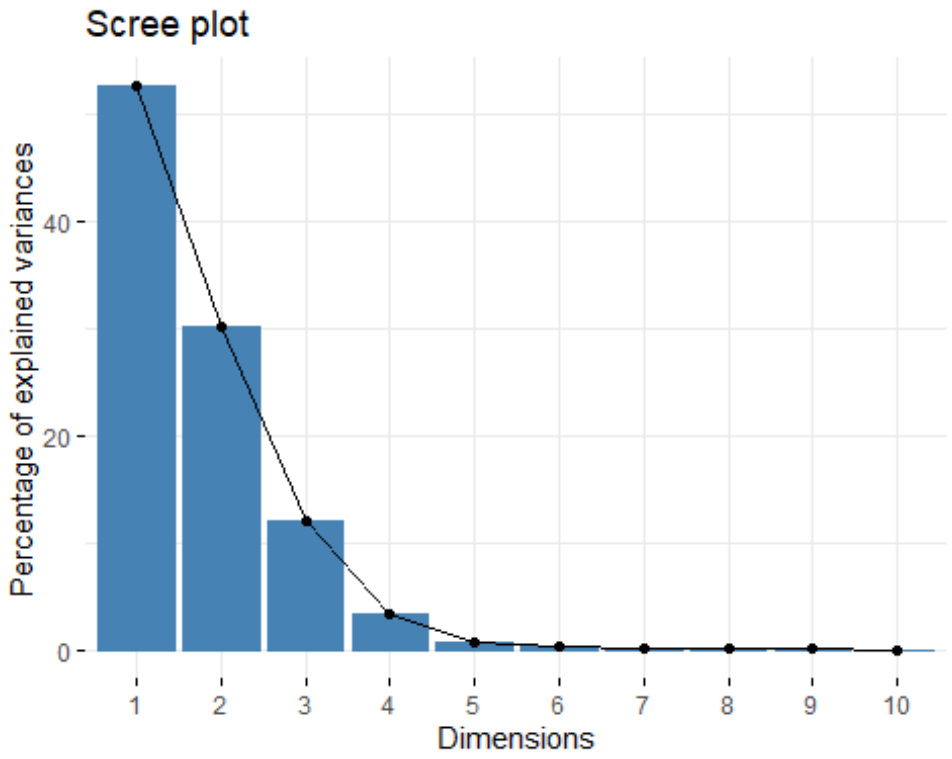
Question 4

The MIR data is now fit using prcomp() function because the observations are lesser in number when compared to the variables taken into consideration.

```
# PCA of the spectra
fit = prcomp(df[,52:582], scale. = T)
summary(fit)

## Importance of components:
##                           PC1      PC2     PC3      PC4     PC5      PC6      P
C7
```

```
## Standard deviation     16.7171 12.6664 7.9914 4.19587 1.9685 1.43713 1.089
07
## Proportion of Variance  0.5263  0.3021 0.1203 0.03316 0.0073 0.00389 0.002
23
## Cumulative Proportion   0.5263  0.8284 0.9487 0.98185 0.9891 0.99304 0.995
27
##                            PC8    PC9    PC10    PC11    PC12    PC13    P
C14
## Standard deviation     0.79274 0.7297 0.58319 0.42457 0.33248 0.31508 0.29
573
## Proportion of Variance 0.00118 0.0010 0.00064 0.00034 0.00021 0.00019 0.00
016
## Cumulative Proportion  0.99646 0.9975 0.99810 0.99844 0.99865 0.99884 0.99
900
```

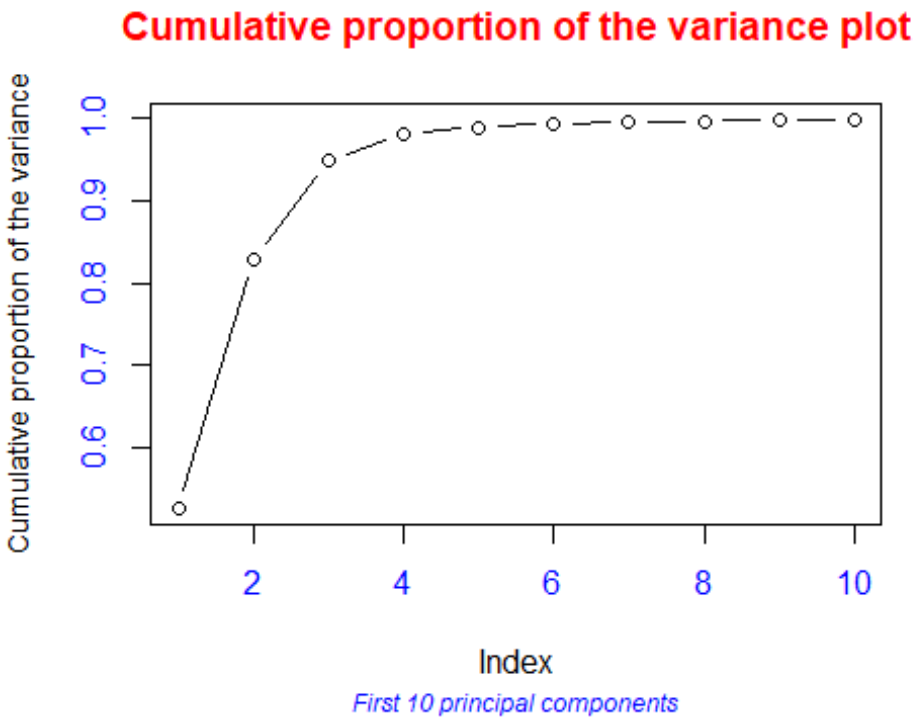
```
# Scree plot
fviz_eig(fit)
```



We find the PC scores. We can see that the first component accounts for more than half of variability of Y . The second and third component together with the first component account for almost **95% variation** in the Y by the data indicating a good fit of data with only the **3** components. This can also be verified with a scree plot, which is a plot of variances versus the components. We can observe a steep drop in variance from one to two and two to three, after which the line almost flatlines .

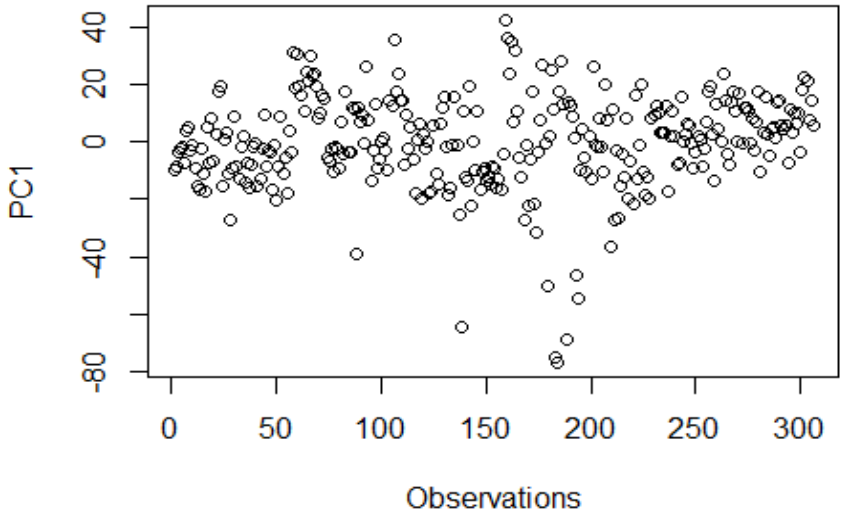
```
# cumulative proportion of the variance plot
plot((cumsum(fit$sdev^2 / sum(fit$sdev^2)))[1:10], type="b", main = "", ylab
```

```
= "", col.axis = "blue")
title(main = "Cumulative proportion of the variance plot", ylab = "Cumulative
proportion of the variance", sub = "First 10 principal components", col.main=
"red", cex.lab = 0.9, cex.sub = 0.75, font.sub = 3, col.sub = "blue")
```

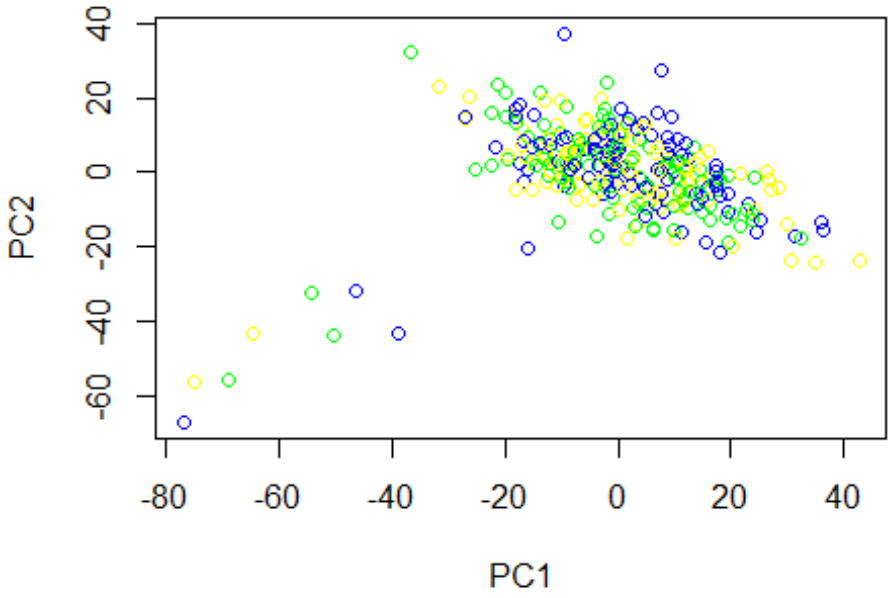


Question 5

```
# The eigenvectors in R point in the negative direction by default, so we mul
tiply by -1 to reverse the signs.

S <- cov(scale(df[,52:582], center = T))
S.eigen <- eigen(S)

# fit$sdev^2 is equal to S.eigen$values

# newfit <- predict(fit, newdata = df[,52:582])

y <- data.matrix(scale(df[,52:582])) %*% fit$rotation

# check
# all(y, newfit)

# Plot of principle component scores
plot(y[,1], ylab = "PC1", xlab = "Observations")
```




```
plot(y[,1], y[,2], type="p", xlab="PC1", ylab="PC2",col=c('blue','green', 'ye
llow'))
```



We can obtain the prediction values by using the first principles. Here we find the eigen vectors of the covariance matrix of the MIR data first, followed by multiplying it with the rotational matrix.

This method can be done in various way and compared to the actual result which was obtained using predict or the x coefficient in the output function.

In the above plot, even though we do not find any particular group with respect to the individual animal, we can clearly see a huge group/pattern that the dairy animals mostly have a PC1 score between [-20,20].

Question 6

The principal component regression (PCR) 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. These PCs are then used to build the linear regression model. The number of principal components, to incorporate in the model, is chosen by cross-validation (cv).

Partial Least Squares (PLS) regression, which identifies new principal components that not only summarizes the original predictors, but also that are related to the outcome. These components are then used to fit the regression model. So, compared to PCR, PLS uses a dimension reduction strategy that is supervised by the outcome.

Method description :

In PCR, we approximate the predictors vector matrix by the first a principal component (PCs), usually obtained from the singular value decomposition follow by regressing Y on the scores, which leads to regression coefficients. Whereas, In PLSR, the components, called Latent Variables (LVs) in this context, are obtained iteratively. One starts with the SVD of the cross product matrix S = X>Y, thereby including information on variation in both X and Y , and on the correlation between them. The left and right vectors are the weights of X and Y. The X scores are obtained and then regressed against the same vector to calculate loadings. This process is iterative and lastly, regression coefficients.

Advantages and Disadvantages :

PCA is an unsupervised transformation, which indicates that no knowledge about the targets is utilized. As a result, in some datasets where the target is strongly correlated with low variance directions, PCR may perform poorly. Indeed, PCA's dimensionality reduction projects the data into a lower-dimensional space where the projected data's variance is greedily maximized along each axis. The directions with the lowest variance will be discarded, and the final regressor will not be able to leverage them, despite having the highest predictive power on the target. PCA, as a dimension reduction tool, is used without taking into account the correlation between the dependent and independent variables, whereas PLS is used with the correlation in mind.

Typically, PLS needs fewer latent variables/components than PCR. PLS is more prone to over fitting because it fits more components than PCA. The main drawback of PCR is that the decision about how many principal components to keep does not depend on the response variable. PLSR is an example of supervised learning. Even with a lot of advantages that PLSR has over PCR, we are

at higher risk of overlooking 'real' correlations and sensitivity to the relative scaling of the descriptor variables with PLS.

Even with the disadvantages of overfitting the data with Partial Least Squares Regression, we are accounting for correlation between X and Y in the data. This seems to be realistic with large data sets with mostly, correlated independent variables.

Question 7

We now use plsr() to fit the data and predict the protein trait, alpha-s1 casein. The data is split into 70-30 set for training and testing purposes. 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.

```
dat <- data.frame(df['alpha_s1_casein'],df[,52:582], check.names = F)

split1 <- sample(c(rep(0, 0.7 * nrow(dat)), rep(1, 0.3 * nrow(dat))))

table(split1)
```

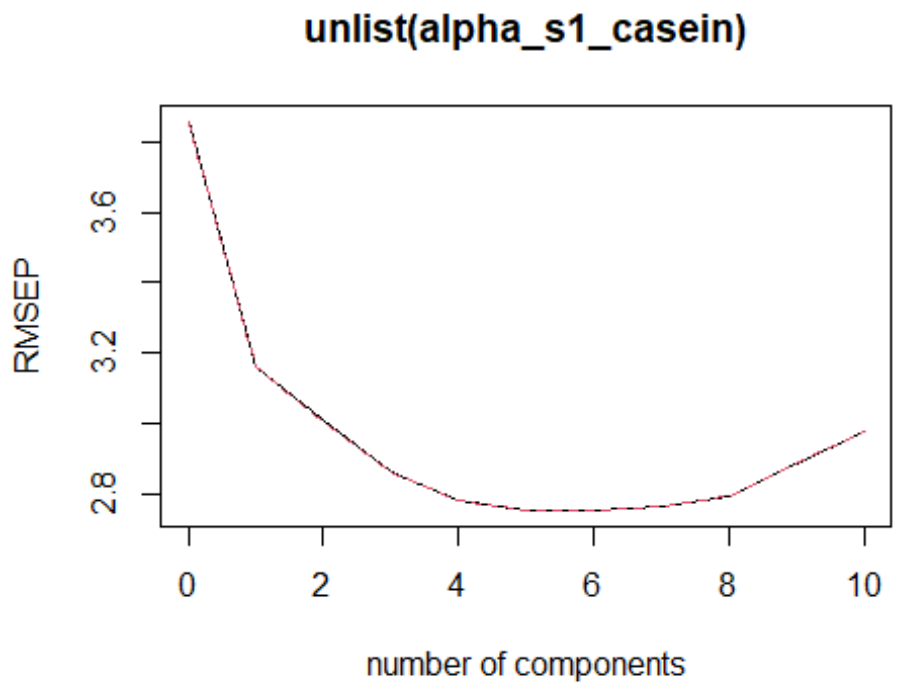
```
## split1
##   0   1
## 214  91
```

```
train <- dat[split1 == 0, ]
test <- dat[split1 == 1, ]

plsfit <- plsr(unlist(alpha_s1_casein) ~ . , ncomp = 10, data = train, valida
tion = "LOO", na.action = na.exclude, scale = T)

summary(plsfit)
```

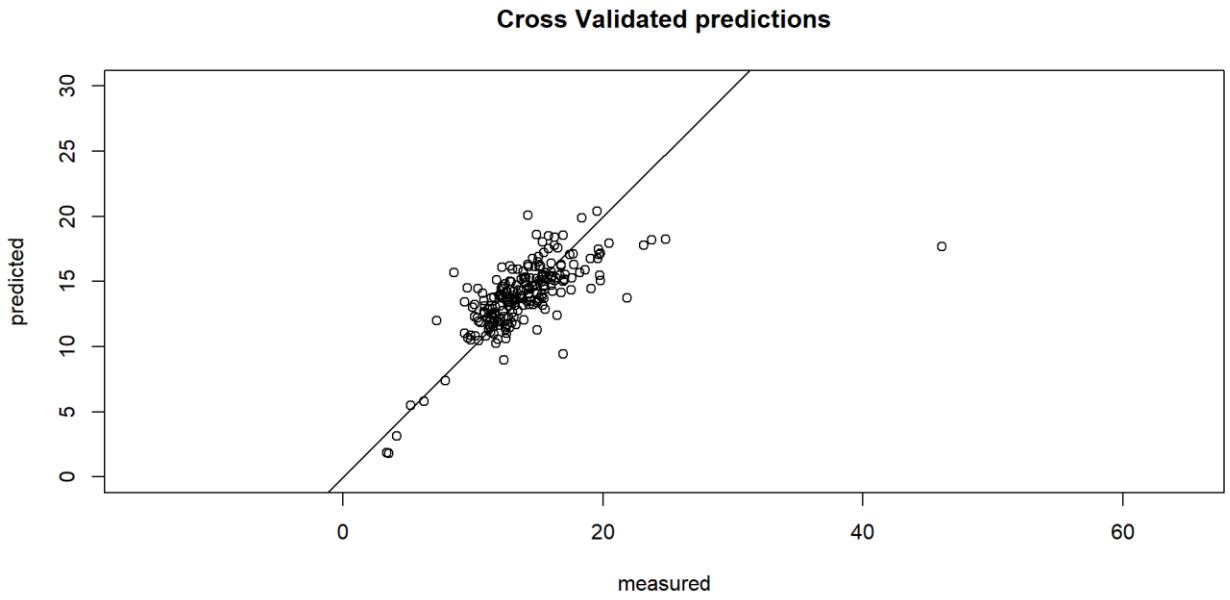
```
## Data:    X dimension: 215 531
##  Y dimension: 215 1
## Fit method: kernelpls
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 215 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           3.858    3.159    3.008    2.861    2.781    2.754    2.753
## adjCV        3.858    3.159    3.008    2.861    2.781    2.754    2.753
##        7 comps  8 comps  9 comps  10 comps
## CV       2.763    2.792    2.887     2.976
## adjCV    2.763    2.791    2.886     2.975
##
## TRAINING: % variance explained
##                          1 comps  2 comps  3 comps  4 comps  5 comps  6 co
mps
```

```
## X                          52.06    72.57    90.35    98.28    98.75    99
.16
## unlist(alpha_s1_casein)    33.79    41.81    47.14    50.26    52.60    53
.92
##                          7 comps  8 comps  9 comps  10 comps
## X                          99.51    99.63    99.69     99.75
## unlist(alpha_s1_casein)    54.66    55.96    58.67     61.04
```

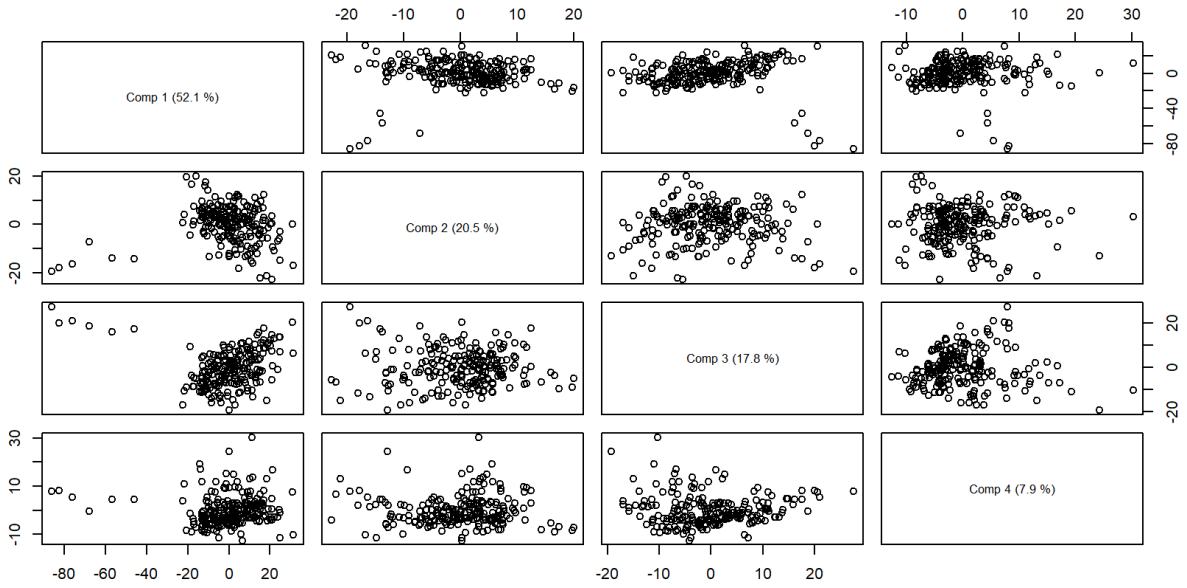
```
# Plot of estimated RMSEPs as functions of the number of components

plot(RMSEP(plsfit)) #Only 3 components are enough
```




```
# Plot of prediction scores
dev.off()

plot(plsfit, ncomp = 3, asp = 1, line = TRUE, ylim = c(0,30), main = "Cross V
alidated predictions")
```




```
plot(plsfit, plottype = "scores", comps = 1:4)
```




```
explvar(plsfit) # we can observe a large group and a small grouping
```

```
##      Comp 1      Comp 2      Comp 3      Comp 4      Comp 5      Comp 6
## 52.05586886 20.50964535 17.78757695  7.92539908  0.47065498  0.41172117
##      Comp 7      Comp 8      Comp 9     Comp 10
##  0.34623312  0.12588306  0.05957168  0.05544736

plspred <- predict(plsfit, ncomp = 2:4, newdata = test, scale = T, type = "sc
ores", na.action = na.exclude)
```

```
head(plspred)
```

```
##       Comp 2     Comp 3     Comp 4
## 3   6.005332 -2.1743375  0.1960397
## 8   2.398132  7.6606797  4.1162348
## 9   3.749946 -0.4415478  0.8489910
## 13  7.172798  4.9726569 -4.6250405
## 25 -1.174633  1.0293486  4.1064491
## 27 -1.307184  1.7146113  6.6822426
```

```
# calculation of the RMSEP for the test set
RMSEP(plsfit, newdata = test)
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 com
ps
##       3.076        2.812        2.656        2.581        2.443        2.4
66
##     6 comps      7 comps      8 comps      9 comps     10 comps
##       2.420        2.429        2.376        2.402        2.465
```

```
# For 3 components, we get 2.8, which is quite close to the cross-validated e
stimate above (2.6).
```

By observing the RMSEP curve plot, we can conclude that only 3 components are enough . This shows the cross-validated predictions with three components versus measured values . We have chosen an aspect ratio of 1, and to draw a target line. The points follow the target line quite nicely, and there is no indication of a curvature or other anomalies.

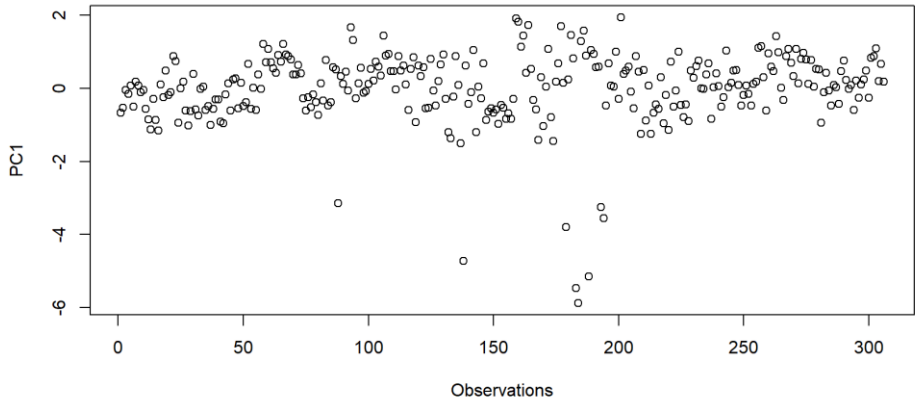
The pair-wise plot is obtained to observe any outliers or anomalies in the data. We can observe a clear group in the data along with the smaller cluster. The explvar() function is used to extract the explained variances. For 3 components, we get 2.8, which is quite close to the cross-validated estimate above (2.6).

Question 8

```
# PC1 is given by multiplying the covriance matrix with the eigen vector asso
ciated with the largest eigen value

# PC Score is given by left eigen vector multiplied by the singular value and
loadings are given by right eigen vector

# The components, called Latent Variables (LVs) in this context, are obtained
iteratively.

dat$alpha_s1_casein <- unlist(dat$alpha_s1_casein)
dat <- dat[complete.cases(dat),]
e <- dat[,2:532]
f <- unlist(dat$alpha_s1_casein)
S <- t(e) %*% as.matrix(f)
```

```r
S.svd <- svd(S)
w <- S.svd$u
t <- as.matrix(e) %*% S.svd$u[,1] # X-scores
t <- scale(t)

u <-  as.matrix(f) %*% S.svd$u[,1] # Y-scores

# obtaining X and Y loadings
p <- t(e) %*% t
q <- t(f) %*% t

# next iteration
en <- e - t %*% t(p)
fn <- f - t %*% t(q)

# Deflated data matrices
en <- data.frame(en)
fn <- data.frame(fn)

W <- data.frame(w)
T_ <- data.frame(t)
P <- data.frame(p)
Q <- data.frame(q)


# Obtaining 5 PC with iterative method since we already know the
number of Pc's required
for (i in 1:4) {

  S <- t(en[,i]) %*% fn[,i]

  S.svd <- svd(S)
  w <- S.svd$u
  t <- as.matrix(en[,i]) %*% S.svd$u[,1]
  t <- scale(t)

  # u <-  as.matrix(f) %*% S.svd$u[,1]

  # obtaining X and Y loadings
  p <- t(en[,i]) %*% t
  q <- t(fn[,i]) %*% t

  # next iteration
  en <- cbind(en, (en[,i] - t %*% t(p)))
  fn <- cbind(fn, (fn[,i] - t %*% t(q)))

  W <- cbind(W, w)
  T_ <- cbind(T_, t)
```

```
  P <- cbind(P, p)
  Q <- cbind(Q, q)
}
plot(T_[,1], ylab = "PC1", xlab = "Observations")
```



The above is a scaled plot od first principle scores is a comparison to the previously predicted values whose range was between [-20,20]. This being a scaled version, most of the values are between the range [-2,2].