Class 7: Clustering

Jacqueline Cheung(A17085191)

2025-01-28

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some dats(where we know there are clear groups) that we can use to test out different clustering methods.

We can use th `rnorm()` function to help us here:

```
hist(rnorm(n=3000, mean=3))
```
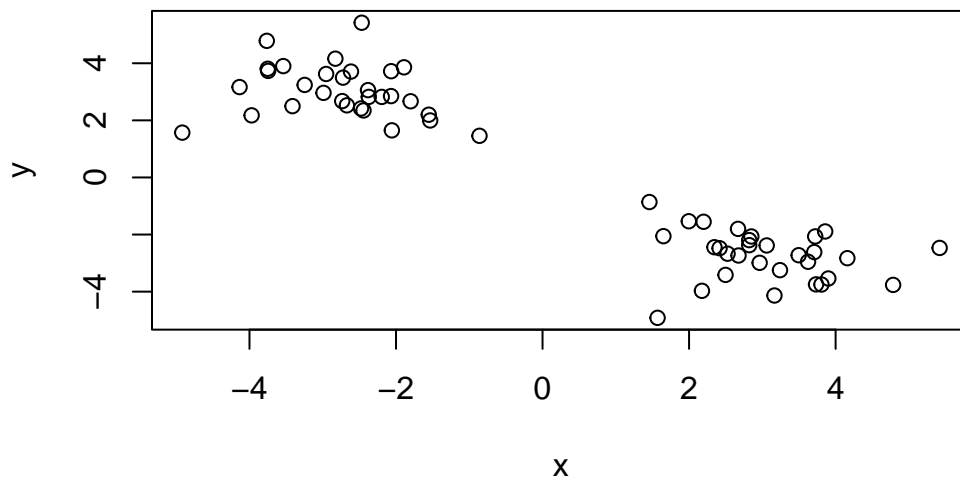
Histogram of rnorm(n = 3000, mean = 3)
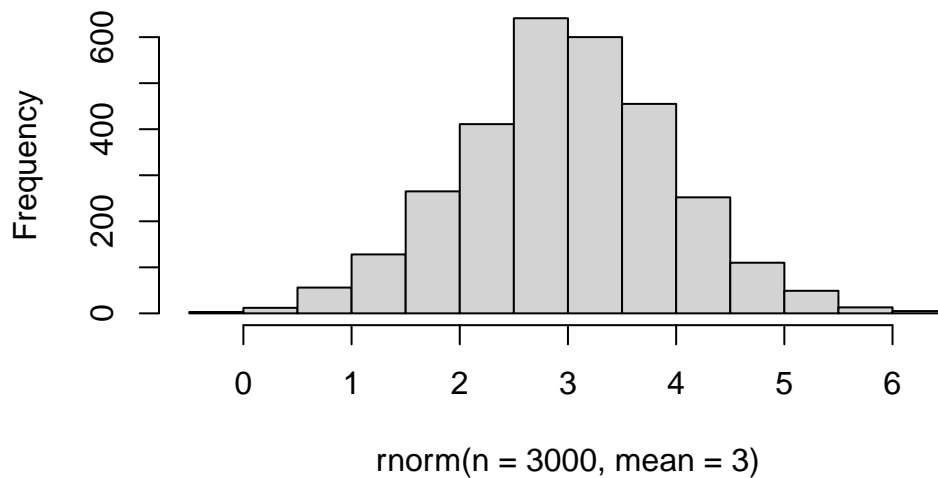
Make data `z` with two "clusters"

```r
x <- c( rnorm(30, mean=-3),
        rnorm(30, mean=+3))

z <- cbind(x=x, y=rev(x))
head(z)
```

```
            x        y
[1,] -2.989781 2.963345
[2,] -2.614604 3.708536
[3,] -3.538934 3.900708
[4,] -3.763845 4.785432
[5,] -2.670584 2.525520
[6,] -1.533754 1.996294
```

```r
plot(z)
```


How big is z

```r
nrow(z)
```

```
[1] 60
```

```r
ncol(z)
```

```
[1] 2
```

K-means clustering

The main function in "base" R for K-means clustering is called `kmeans()`

```r
k <- kmeans(z, centers = 2)
k
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x         y
1 -2.738798  3.043900
2  3.043900 -2.738798

Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 47.87285 47.87285
 (between_SS / total_SS =  91.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```r
attributes(k)
```

```
$names
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

$class
[1] "kmeans"
```

3

Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What component of our results tells us about the cluster membership?(i.e. Which point lies in which cluster?)

```
k$cluster
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

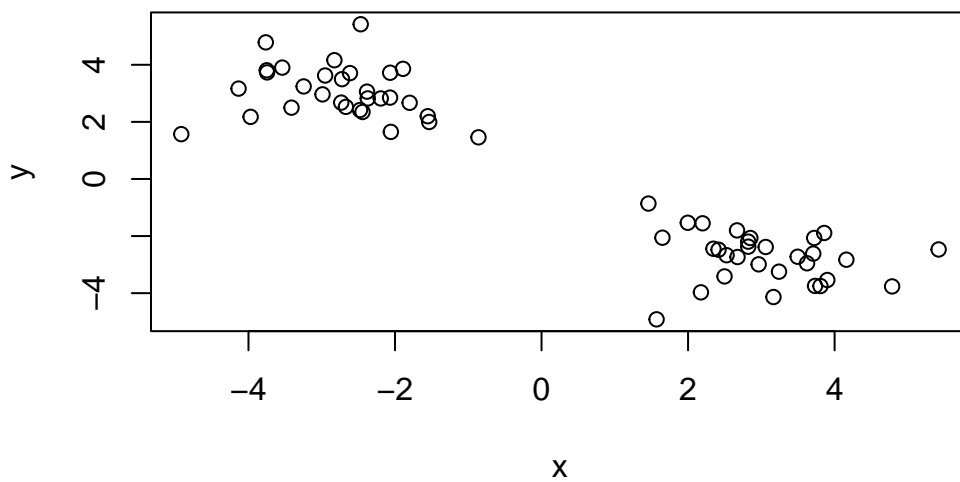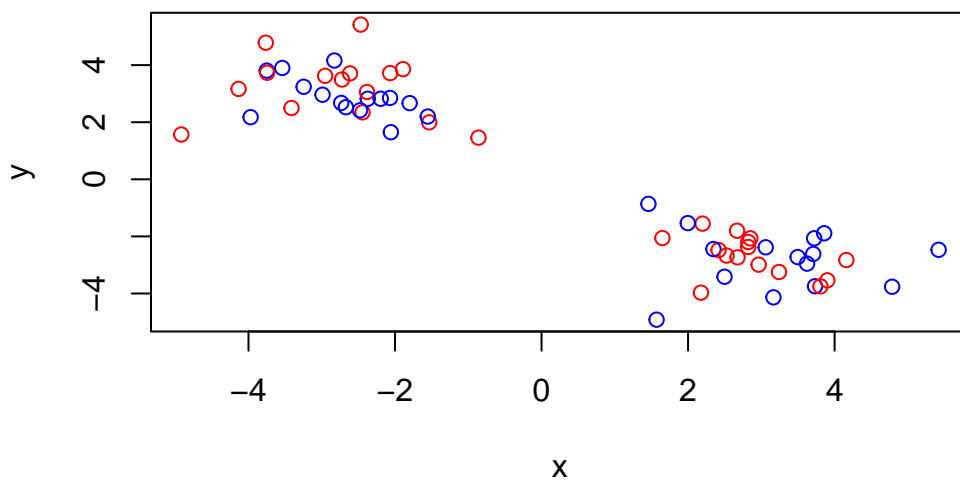Q. What is the center of each cluster

```
k$centers
```

```
          x         y
1 -2.738798  3.043900
2  3.043900 -2.738798
```

Q. Put this result info together and make a little "base R" plot of out clustering result. Also add the cluster center points to this plot
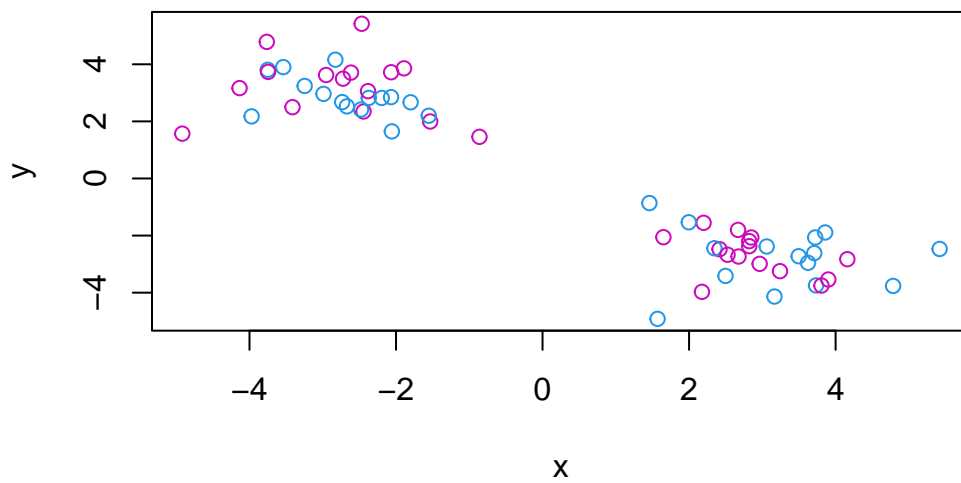
```
plot(z)
```

```r
plot(z, col=c("blue", "red"))
```
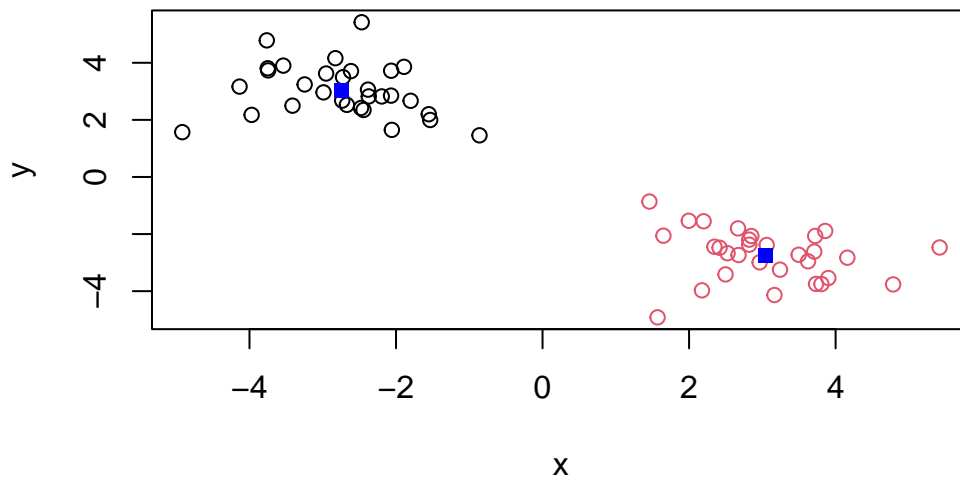

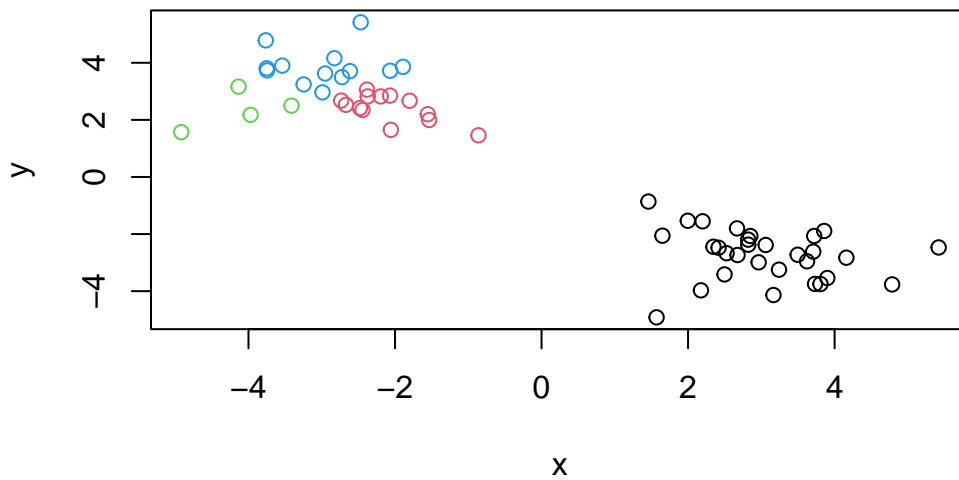You can color by number

```
plot(z, col=c(4,6))
```


Plot colored by cluster membership:

```
plot(z, col=k$cluster)
points(k$centers, col="blue", pch=15)
```

Q. Run kmeans on our input `z` and define 4 clusters making the same result vizual-
ization plot as above(plot of z colored by cluster membership)

```
k4 <- kmeans(z, centers = 4)
plot(z, col=k4$cluster)
```

```
#to measure how well the clustering was
k4$tot.withinss
```

```
[1] 66.05542
```

Hierarchical Clustering

The main function in base R for this is called `hclust()`. It will take as input a distance matrix(key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data)

```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```
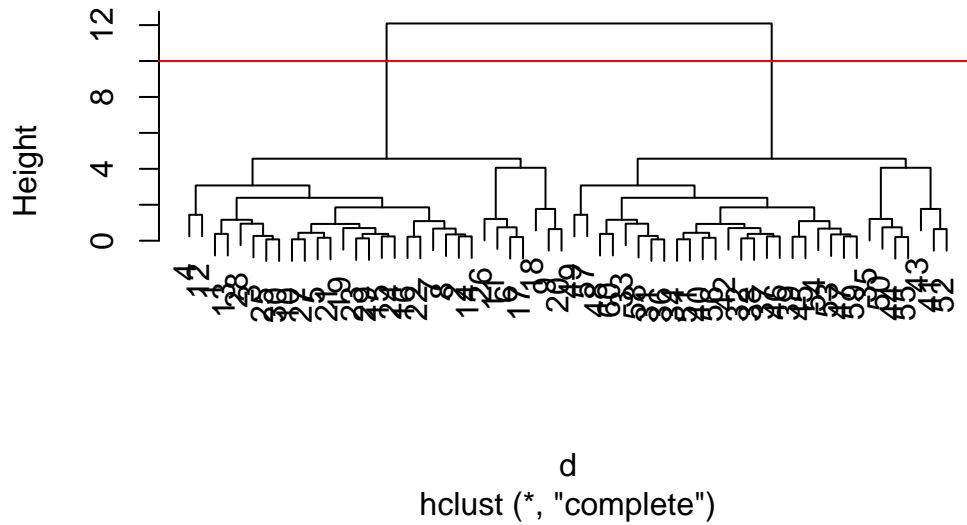
```
plot(hc)
abline(h=10, col = "red")
```
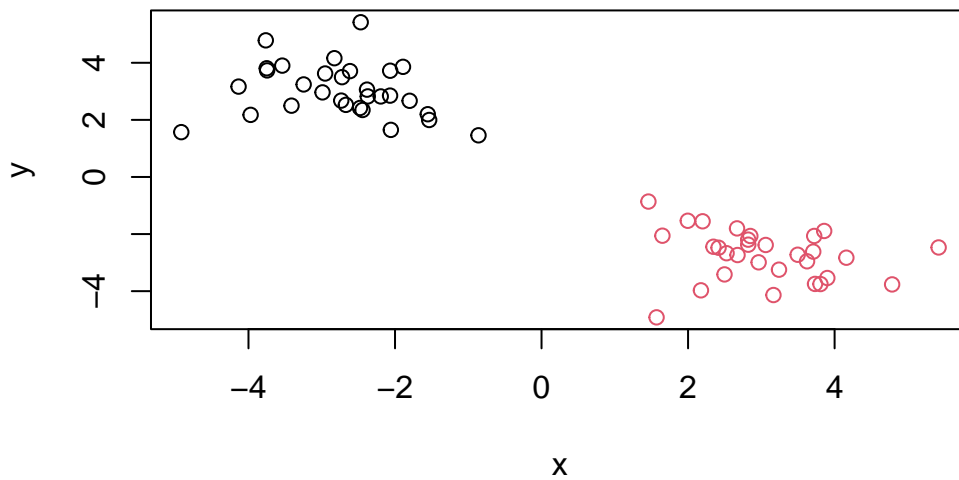
Cluster Dendrogram

d
hclust (*, "complete")

Once I inspect the "tree" I can "cut" the tree to yield my groupings or clusters. The function to this is called `cutree()`

```
grps <- cutree(hc, h=10)
```

```
plot(z, col=grps)
```

Hands on with Principal Component Analysis(PCA)

Let's examine some silly 17-dimensional data detailing food consumption in the UK (England, Scotland, Wales and N. Ireland). Are these countries eating habits different or similar and how so?

Data Import

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
x
```

	England	Wales	Scotland	N.Ireland
Cheese	105	103	103	66
Carcass_meat	245	227	242	267
Other_meat	685	803	750	586
Fish	147	160	122	93
Fats_and_oils	193	235	184	209
Sugars	156	175	147	139
Fresh_potatoes	720	874	566	1033
Fresh_Veg	253	265	171	143
Other_Veg	488	570	418	355

```
Processed_potatoes      198    203      220       187
Processed_Veg           360    365      337       334
Fresh_fruit            1102   1137      957       674
Cereals                1472   1582     1462      1494
Beverages                57     73       53        47
Soft_drinks            1374   1256     1572      1506
Alcoholic_drinks        375    475      458       135
Confectionery            54     64       62        41
```
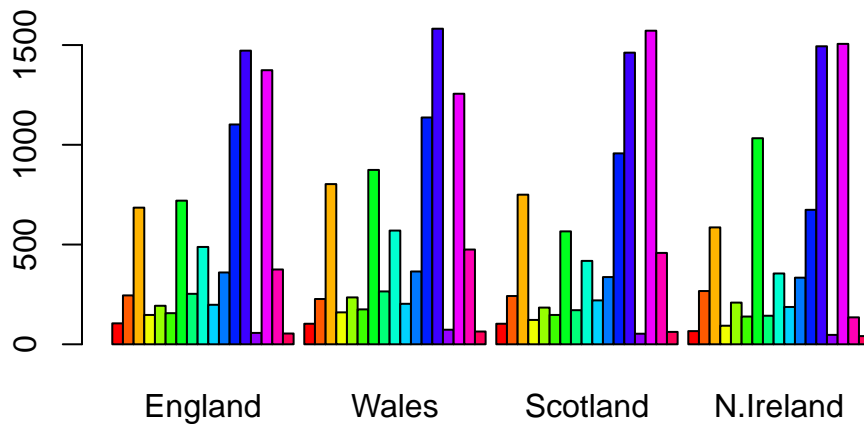
Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?
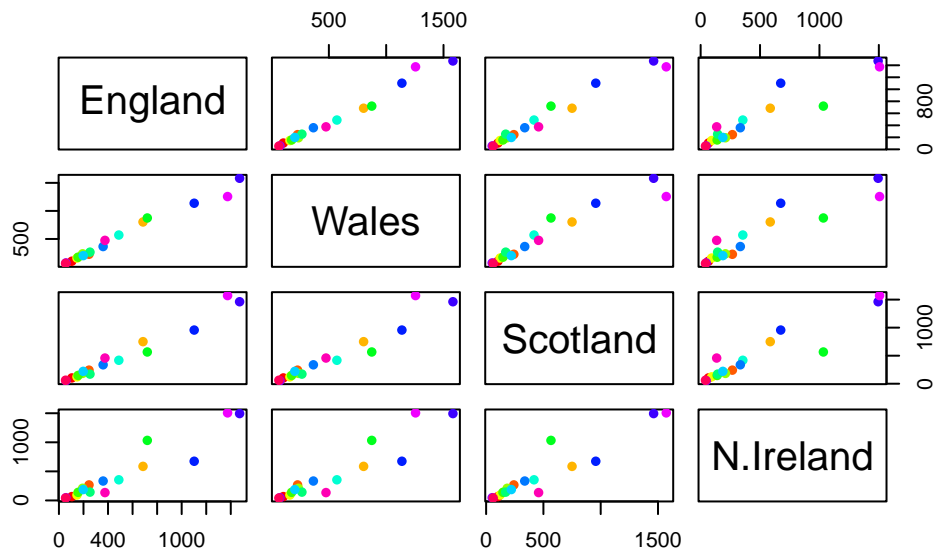
```
dim(x)
```

```
[1] 17   4
```

```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```


Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

```
pairs(x, col=rainbow(nrow(x)), pch=16)
```


Looking at these types of "pairwise plots" can be helpful but it does not scale well and kind of sucks! There must be a better way...

PCA to the rescue!

The main function for PCA in base R is called `prcomp()`. This function wants the transpose of our input data - i.e. the important foods in as columns and the countries as rows.

```
pca <- prcomp(t(x))
summary(pca)
```

```
Importance of components:
                          PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's see what is in our PCA result object `pca$x`

```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"     "x"

$class
[1] "prcomp"
```

The `pca$x` result is where we will focus first as this details how the countries are realted to each other in terms of our new "axis" (aka)

```
head(pca$x)
```

```
                 PC1          PC2         PC3           PC4
England    -144.99315   -2.532999 105.768945 -4.894696e-14
Wales      -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland    -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland   477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1], pca$x[,2], col=c("orange","red","blue","green"), xlab ="PC1", ylab="PC2")
```
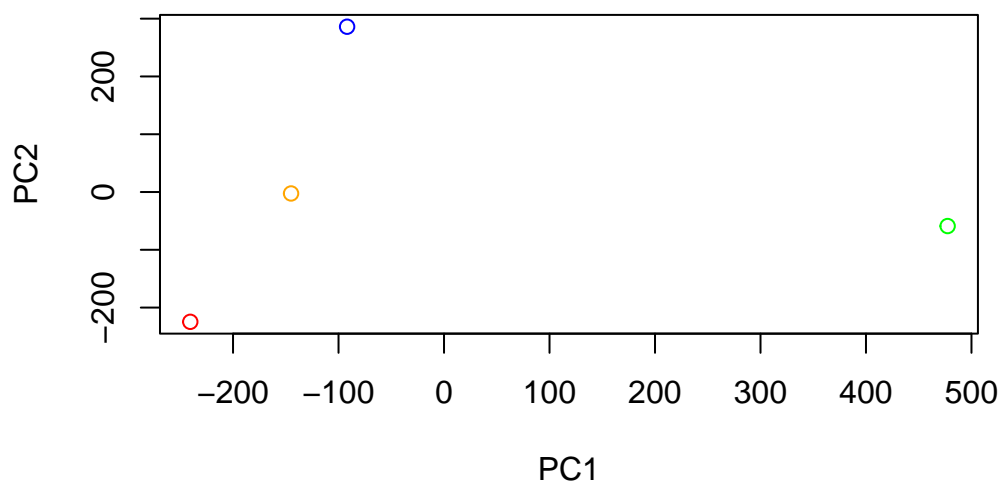
We can look at the so-called PC "loadings" result objects to see how the original foods contribute to our new PCs (i.e. How the origina variables contribute to our new better PC variables)

```
pca$rotation[,1]
```

```
        Cheese       Carcass_meat           Other_meat                 Fish
   -0.056955380         0.047927628         -0.258916658         -0.084414983
  Fats_and_oils               Sugars       Fresh_potatoes            Fresh_Veg
   -0.005193623        -0.037620983          0.401402060         -0.151849942
      Other_Veg  Processed_potatoes        Processed_Veg          Fresh_fruit
   -0.243593729        -0.026886233         -0.036488269         -0.632640898
        Cereals            Beverages          Soft_drinks     Alcoholic_drinks
   -0.047702858        -0.026187756          0.232244140         -0.463968168
  Confectionery
   -0.029650201
```

Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```
pca$rotation[,2]
```

```
        Cheese       Carcass_meat           Other_meat                 Fish
    0.016012850         0.013915823         -0.015331138         -0.050754947
  Fats_and_oils               Sugars       Fresh_potatoes            Fresh_Veg
   -0.095388656        -0.043021699         -0.715017078         -0.144900268
      Other_Veg  Processed_potatoes        Processed_Veg          Fresh_fruit
   -0.225450923         0.042850761         -0.045451802         -0.177740743
        Cereals            Beverages          Soft_drinks     Alcoholic_drinks
   -0.212599678        -0.030560542          0.555124311          0.113536523
  Confectionery
    0.005949921
```