Class 7: Machine Learning

Mia Fava (A17577873)

Today we are going to learn how to apply different machine learning methods, beginning with clustering.

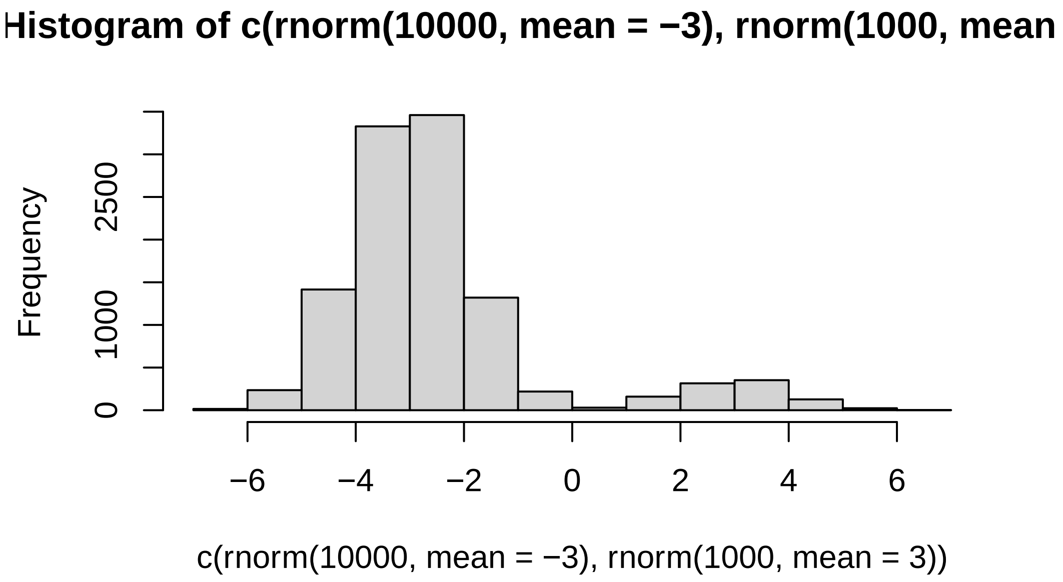
The goal here is to find groups/clusters in your input data.

First I will make up some data with clear groups. For this I will used the `rnorm()` function.

```r
rnorm(10)
```

```
 [1] -0.2676802  0.9789606  1.9138185  0.7371866  0.5999336  0.9330482
 [7]  1.0400379 -0.3929570 -2.3433195  0.5782594
```

```r
hist(c(rnorm(10000, mean = -3), rnorm(1000, mean=3)))
```



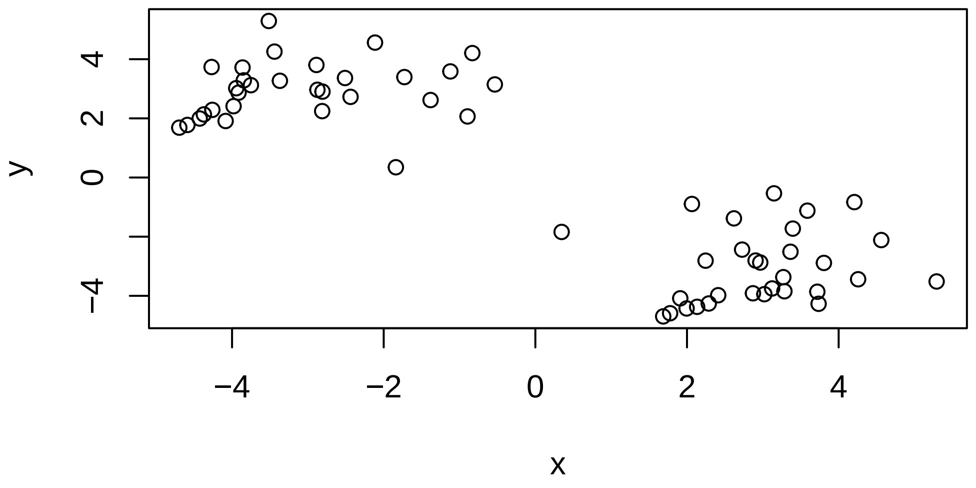
```
n <- 30
x <- c(rnorm(n, -3), rnorm(n, +3))
y <- rev(x)
y
```

```
 [1]  3.7356587  3.5873037  2.9045806  2.9661375  2.8697640  4.2568642
 [7]  2.1334817  3.2846679  2.2451662  1.9947550  3.2701151  2.7267248
[13]  3.8053719  2.4114859  0.3464704  2.2869272  3.1224123  1.6861259
[19]  4.2069363  1.9110946  3.1471211  5.2920667  2.6190060  3.3643625
[25]  1.7771430  4.5619053  3.0200020  3.7186749  2.0649012  3.3955716
[31] -1.7272029 -0.8946366 -3.8617240 -3.9448530 -2.1146269 -4.5886410
[37] -2.5104449 -1.3819212 -3.5153242 -0.5345297 -4.0846879 -0.8310902
[43] -4.6958851 -3.7511674 -4.2599577 -1.8395624 -3.9798800 -2.8874919
[49] -2.4371982 -3.3690522 -4.4263811 -2.8112717 -3.8449749 -4.3716569
[55] -3.4405579 -3.9139590 -2.8742847 -2.8074599 -1.1206745 -4.2696506
```

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

```
             x        y
[1,] -4.269651 3.735659
[2,] -1.120675 3.587304
[3,] -2.807460 2.904581
[4,] -2.874285 2.966138
[5,] -3.913959 2.869764
[6,] -3.440558 4.256864
```

```
plot(z)
```



Use `kmeans()` function setting k to2 and nstart = 20. >Q How mmany points are in each cluster

> Q What 'component' of your result object details? -cluster size -cluster assignment/member -cluster center?

> Q. Plot x colored by the kmean cluster assignment and add cluster centers as blue points

```
km <- kmeans(z, centers=2)
km
```

```

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

Cluster means:
          x         y
1 -3.036358  2.957093
2  2.957093 -3.036358

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] 73.69228 73.69228
 (between_SS / total_SS =  88.0 %)

Available components:

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

Reults in kmeans object `km`

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Cluster size?

```r
km$size
```

```
[1] 30 30
```

cluster assignment/member?

```r
km$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
```

-cluster center?

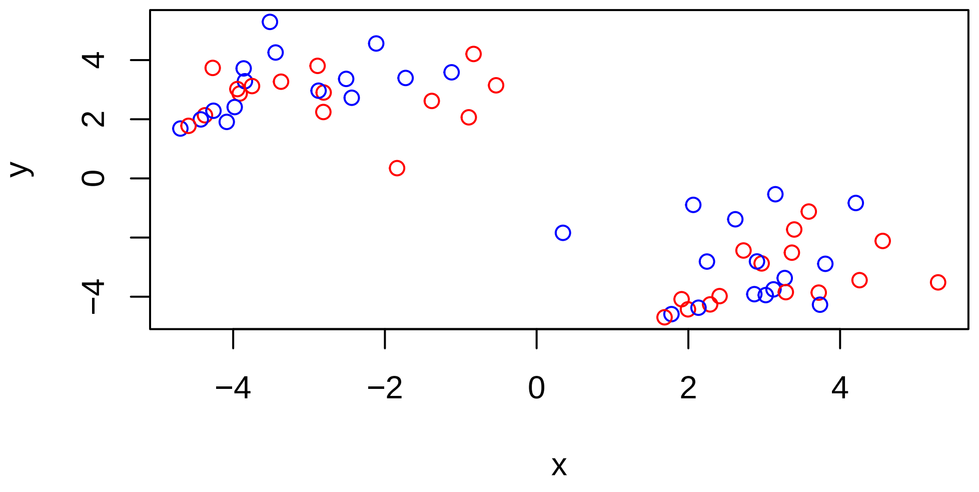
```r
km$centers
```

```
          x         y
1 -3.036358  2.957093
2  2.957093 -3.036358
```

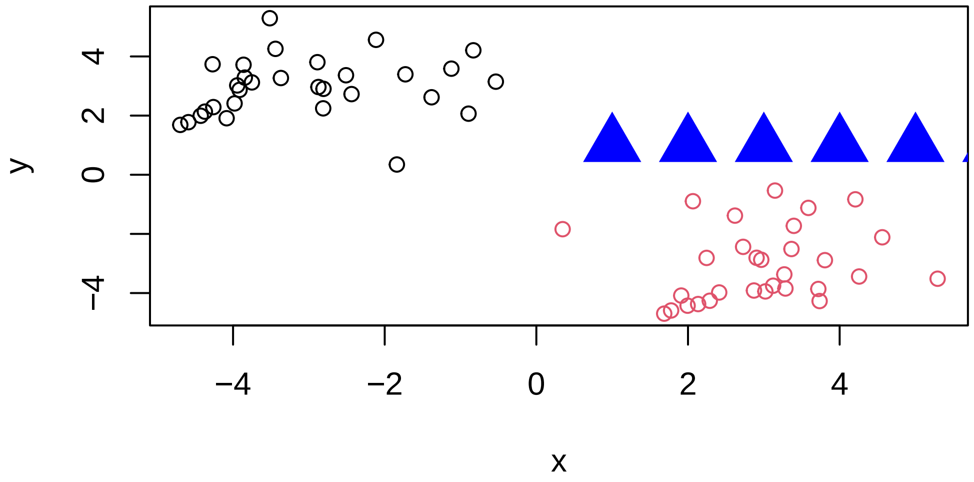
> Q Plot z colored by the kmeans cluster assignment and add cluster centers as blue points

R will recycle the shorter color version to be the same length as the longer

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



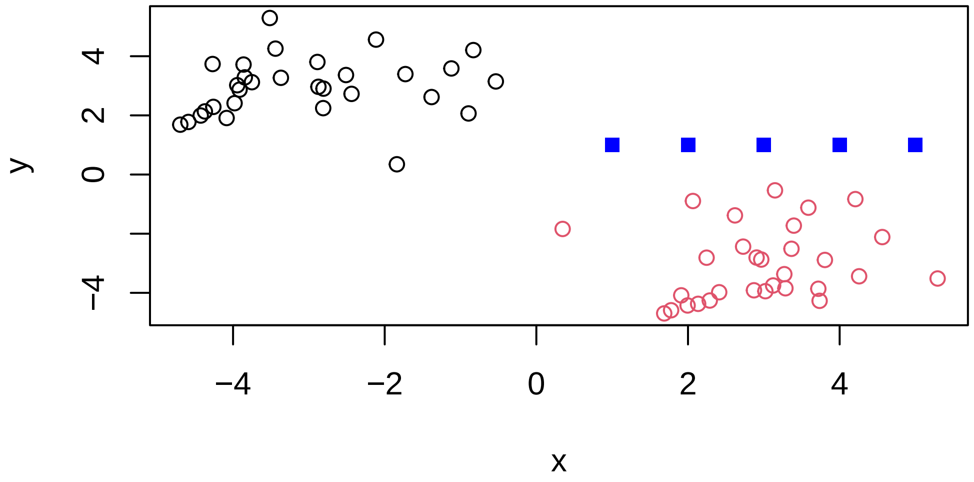
```
plot(z, col=km$cluster)
points(km$cluster, col="blue", pch=17, cex=3)
```



Q. Can you run kmean and ask for 4 clusters, please and plot the results like we have done above?

```
pm <- kmeans(z, centers=4)
```

```
plot(z, col=km$cluster)
points(km$cluster, col="blue", pch=15, cex=1)
```



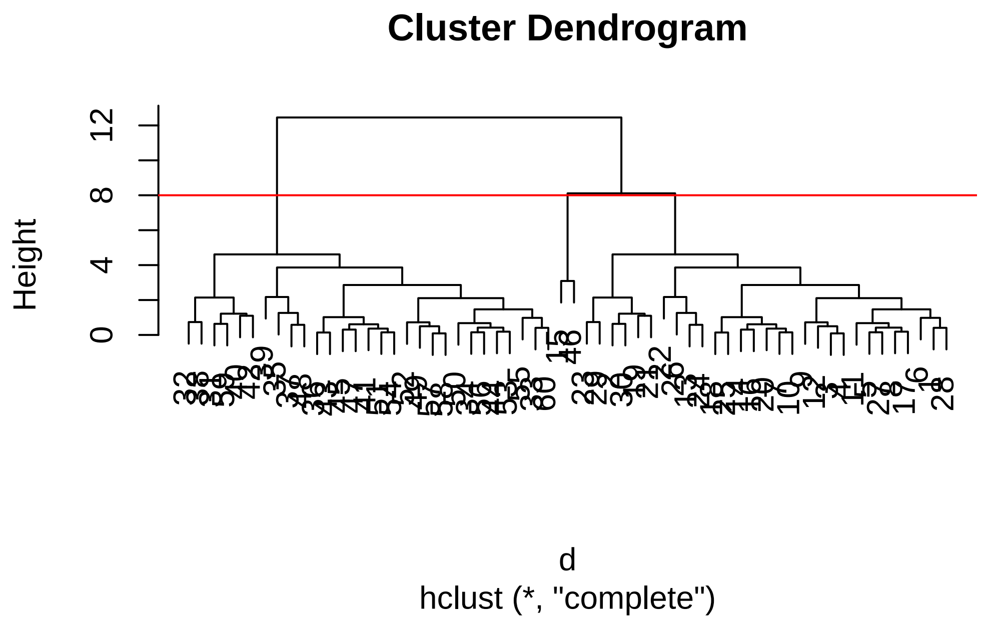
##Hierarchial Clustering

Lets take the same made-up datat `z` and see how hclust works

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

```
Call:
hclust(d = d)

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

```
plot(hc)
abline(h=8, col="red")
```



```
cutree(hc, h=8)
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[39] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

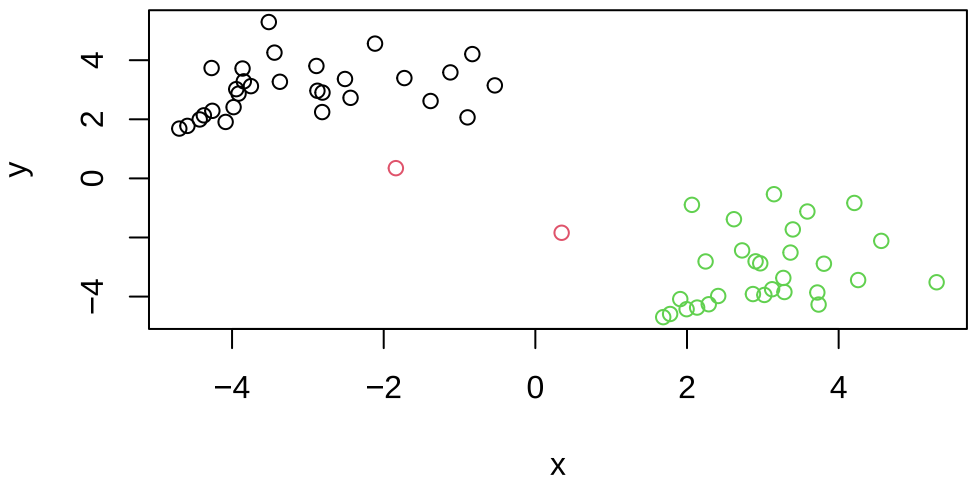
I can get my membership c=vector by "cutting the tree" with the `cutree()` function like so:

```
grps<-cutree(hc, h=8)
grps
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[39] 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

Can you plot `z` colored by our hclust results:

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



##START OF ON HANDS LAB

PCA of UK food data

Read the data from the UK on food consumpation in different parts of the UK

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

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```r
dim(x)
```

```
[1] 17  4
```

##Preview the first 6 rows

```r
head(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
```

```r
# Note how the minus indexing works
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
    Wales Scotland N.Ireland
105   103      103        66
245   227      242       267
685   803      750       586
147   160      122        93
193   235      184       209
156   175      147       139
```

```r
dim(x)
```

```
[1] 17  3
```

```r
x <- read.csv(url, row.names=1)
head(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
```

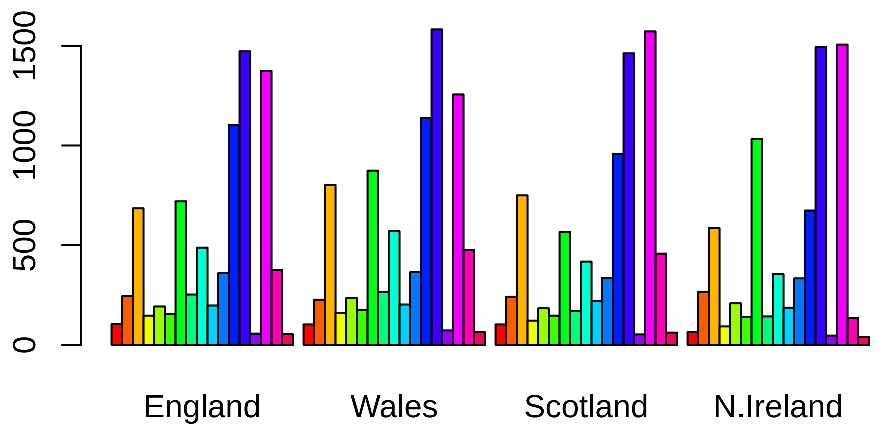
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

Personally I prefer the read.csv() function to be the easiest way to get the row names in comparison to the view() function.

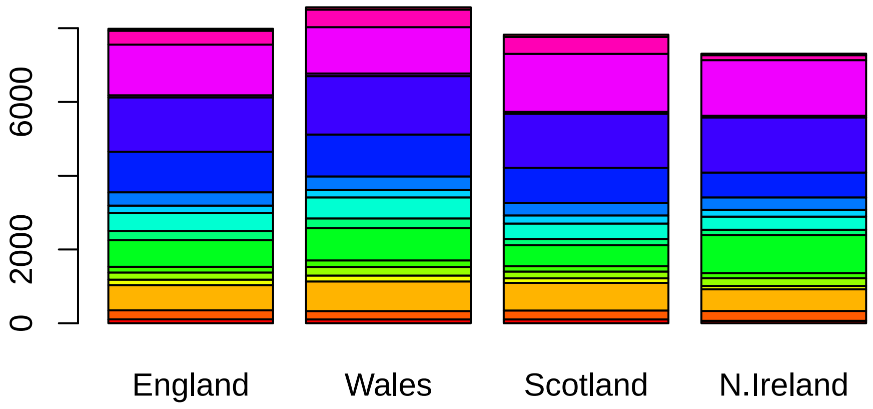
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

Changing the beside = from T to F would change the first plot to the second plot.

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



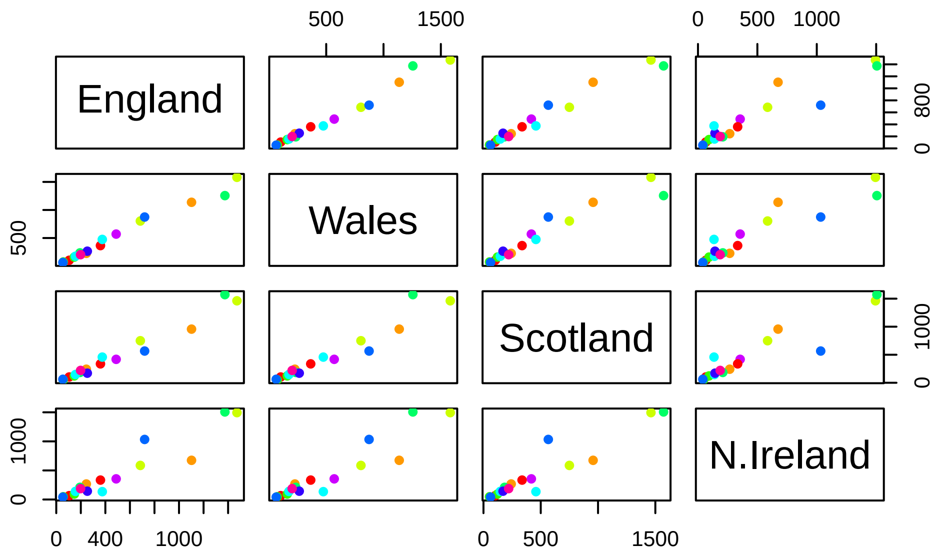
```r
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

This following code for the pairs plot is useful with small datas like the one provided. It is also showing the country names in a diagonal – in which the row is showing what country is on the y-axis; the x axis shows whatever is being compared. If the data isthe same for the two countries, the data will look like a straight diagonal line; if they are not similar/differ in data value, they will be more scattered and not as uniform across a diagonal line. It is hard to see structure and trend in this plotting, however we have to do this when we have big datasets with 1,000s or 10s of thousands of things we are measuring.

```
pairs(x, col=rainbow(10), pch=16)
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

The main differences between N. Ireland and other countries of the UK is that Irelands data values are seen to be more scattered, therefore the data values in each category of food are seen to be less similar to those of the other UK countries.

PCA to the rescue

See how PCA deals with this data set – main function in base R to do PCA is called `prcomp()`

```
# Use the prcomp() PCA function
pca <- prcomp( t(x) )
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-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 inside this `pca` object that we created from running `prcomp()`

```
attributes(pca)
```

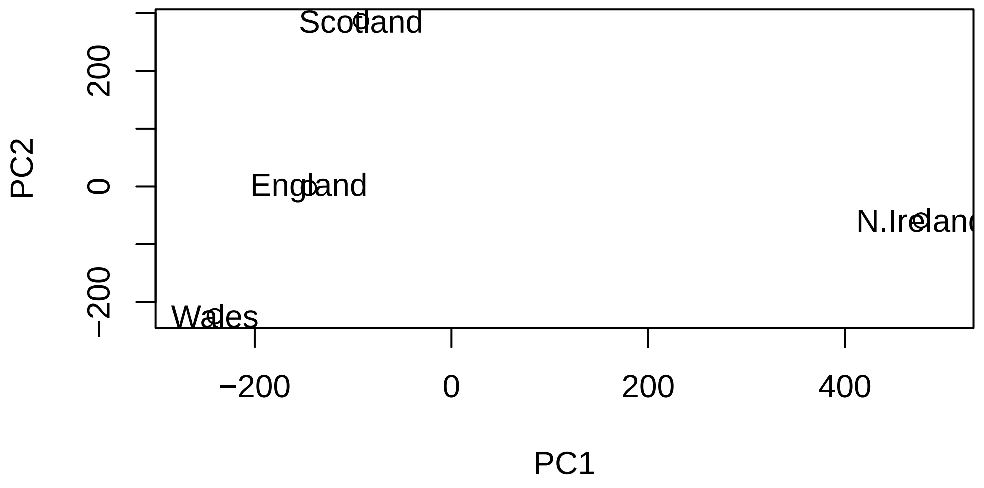
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

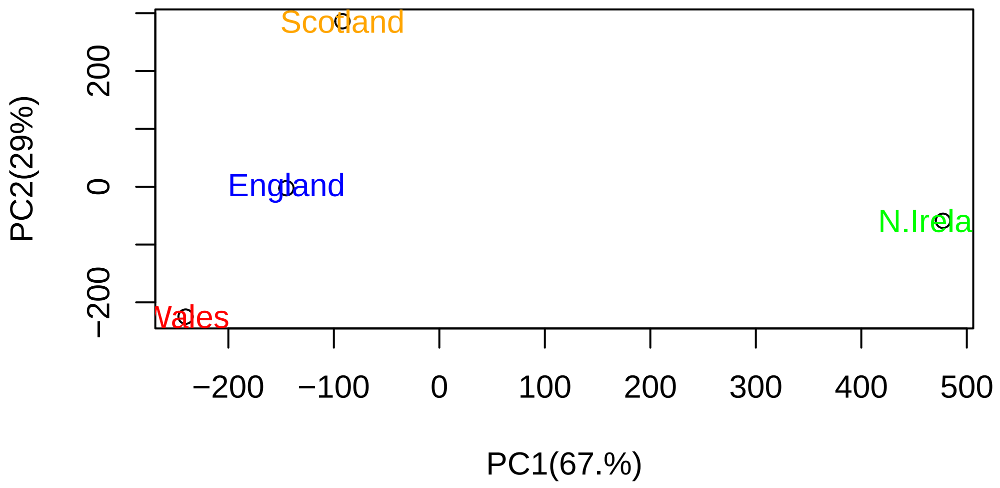
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```r
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```r
plot(pca$x[,1], pca$x[,2], xlab="PC1(67.%)", ylab="PC2(29%)")
text(pca$x[,1], pca$x[,2], colnames(x), col=c("blue", "red","orange", "green"))
```



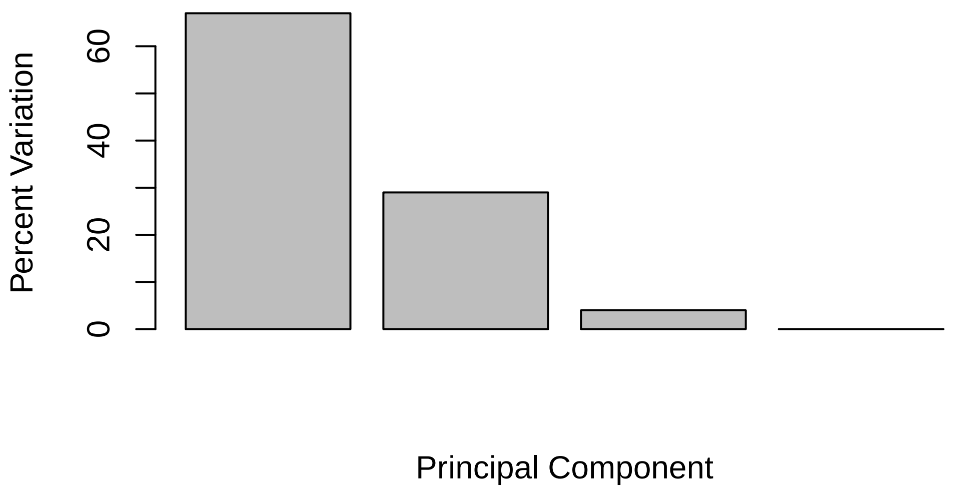
```r
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
[1] 67 29  4  0
```

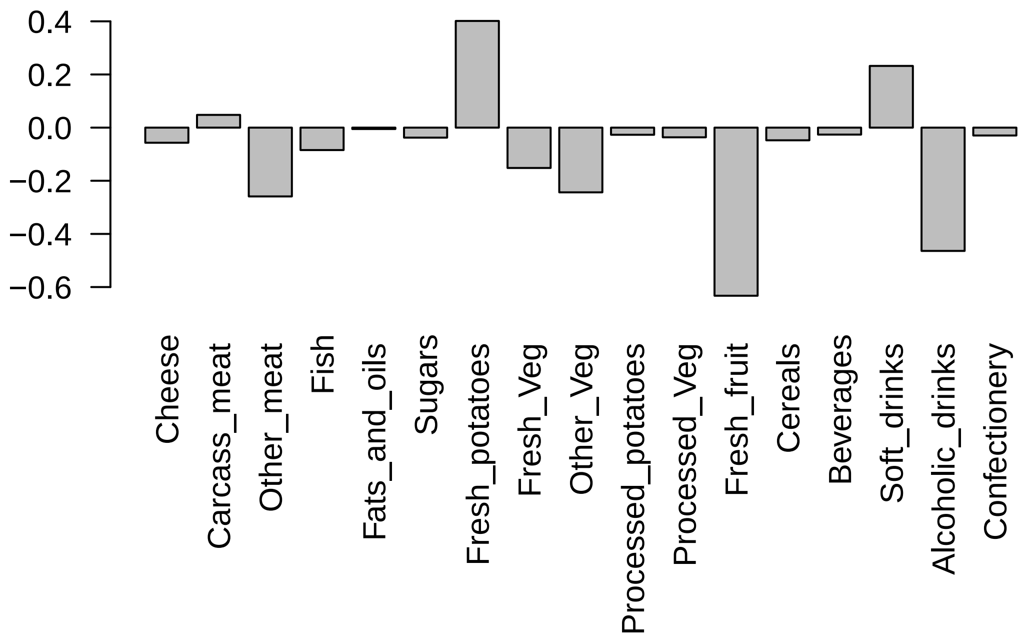
```r
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```r
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



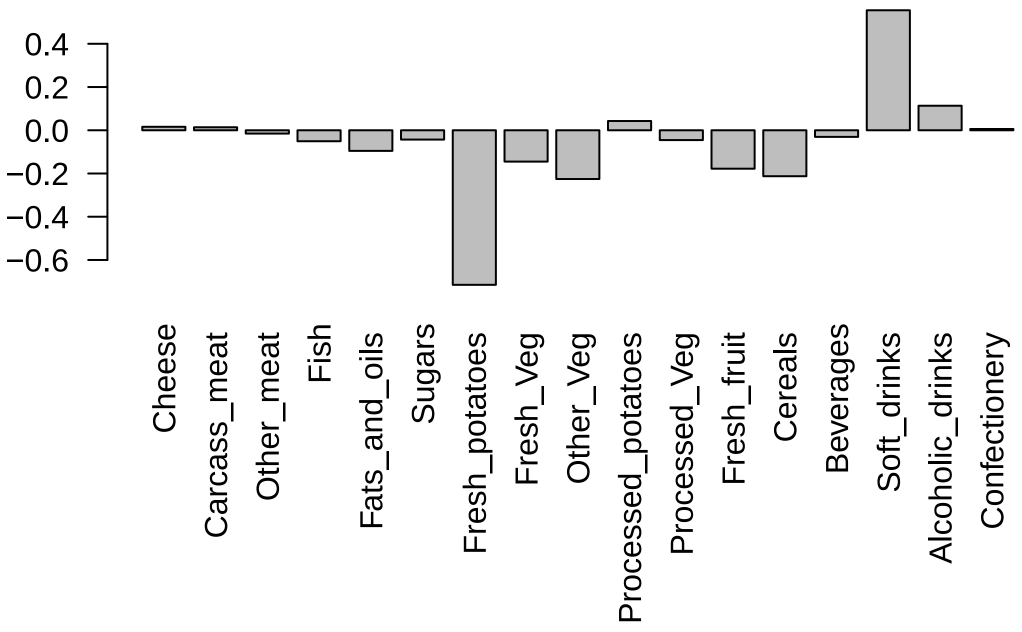
```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

Changed the pca*rotation*[, 1]*topca*rotation[,2]. The two groups that feature prominantely were the high negative loading score of the fresh potatoes in Scotland prodominately, and soft drinks are the positive loading score seen in N.Ireland, Wales, & England.

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery

Q10 :