Class 7: Machine Learning I

Danika

In this class we will explore clustering and dimensionality reduction

K-means

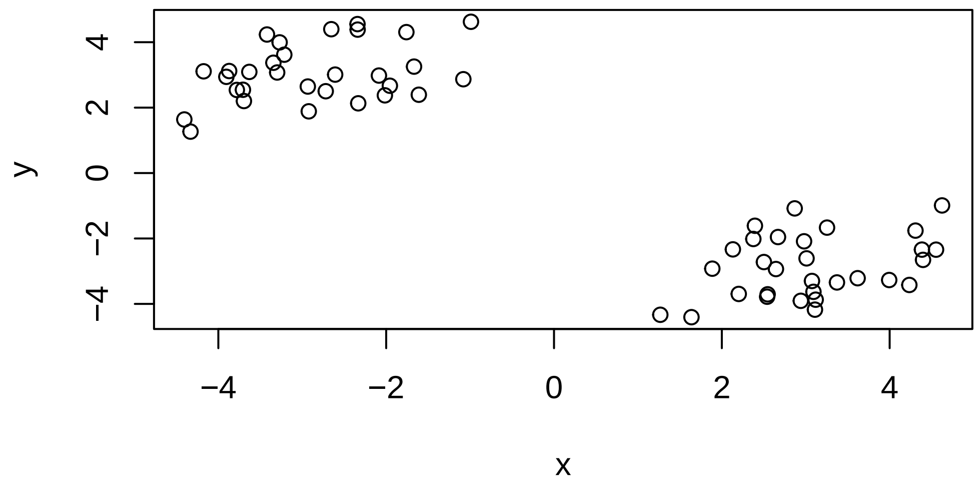
Make up some input data where we know what the answer should be.

```
tmp <- c(rnorm(30,-3), rnorm(30,+3))
x <- cbind(x=tmp,y=rev(tmp))
head(x)
```

```
             x        y
[1,] -3.298975 3.074100
[2,] -4.331778 1.266882
[3,] -1.759133 4.307632
[4,] -2.720366 2.501287
[5,] -4.175537 3.109561
[6,] -3.269835 3.994418
```

Quick plot of x to see the two groups at -3, +3 and +3, -3

```
plot(x)
```



Use the ‘kmeans()’ function setting k to 2 and nstart=20

```r
km <- kmeans(x, centers=2, nstart=20)
km
```

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

Cluster means:
          x         y
1  3.057634 -2.868953
2 -2.868953  3.057634

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

Within cluster sum of squares by cluster:
[1] 49.06566 49.06566
 (between_SS / total_SS =  91.5 %)

Available components:
```

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

Q. How many points are in each cluster?

```r
km$size
```

```
[1] 30 30
```

Q. What ‘component’ of your result object detail? - cluster assignment/ membership? - cluster center?

```r
km$cluster
```

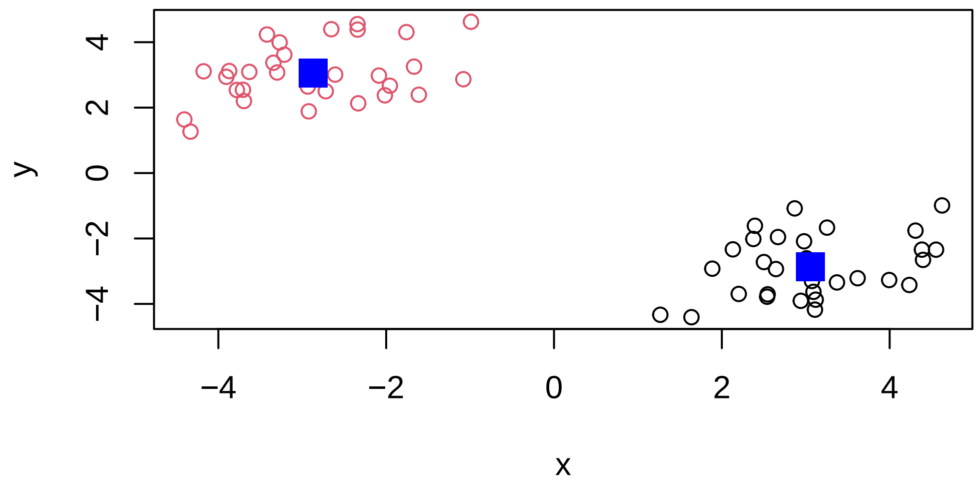
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```r
km$centers
```

```
          x         y
1  3.057634 -2.868953
2 -2.868953  3.057634
```

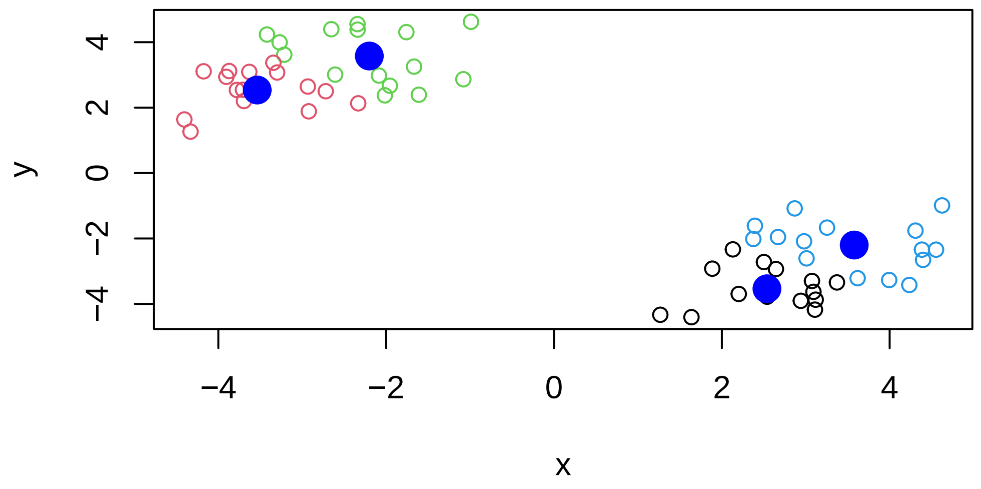
Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```r
plot(x, col=km$cluster,)
points(km$centers, col="blue", pch=15, cex=2)
```



Play with kmeans and ask for different number clusters

```r
km <- kmeans(x, centers=4, nstart=20)
plot(x, col=km$cluster,)
points(km$centers, col="blue", pch=16, cex=2)
```



Hierarchical Clustering

This is another very useful and widely employed clustering methods which has the advantage over kmeans in that it can help reveal the something of the true grouping in your data.

The 'hclust()' function wants a distance matrix as input. We can get this from the 'dist()' function.

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

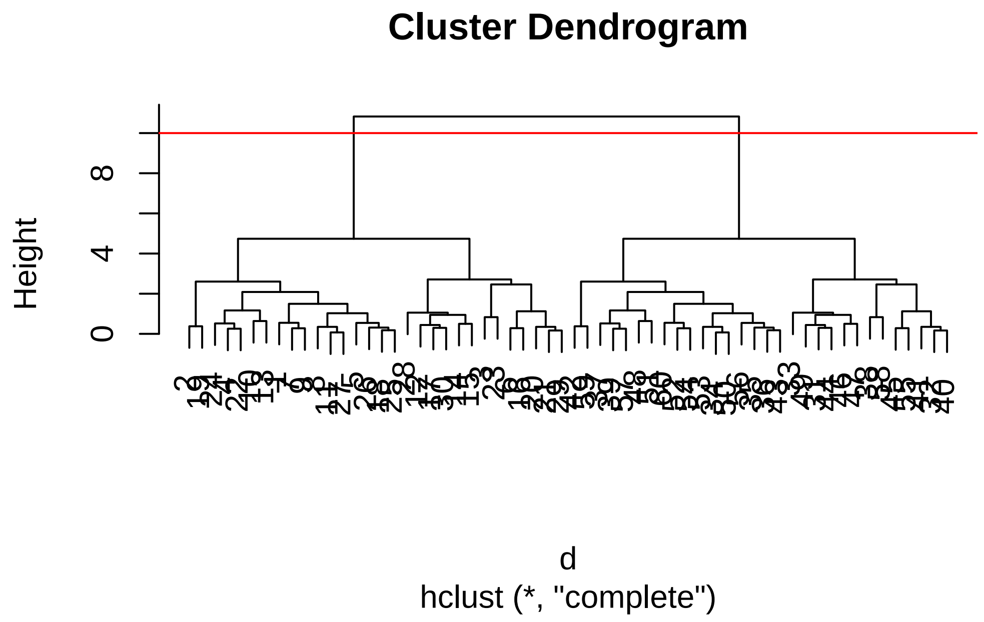
```

Call:
hclust(d = d)

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

There is a plot method for hclust results:

```
plot(hc)
abline(h=10, col="red")
```



It is often helpful to use the 'k=' argument to cutree rather than the 'h=' height of cutting with 'cutree()'. This will cut the tree to yield the number of cluster you want.

```
cutree(hc,k=4)
```

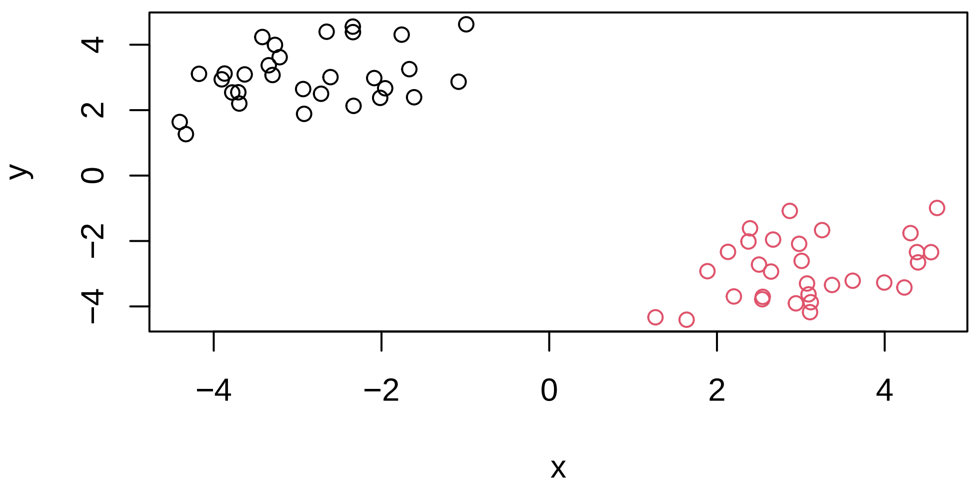
```
 [1] 1 1 2 1 1 2 1 1 1 1 1 2 1 2 2 2 2 1 1 2 2 1 2 1 1 1 1 2 2 2 3 3 3 4 4 4 4 3
[39] 4 3 3 4 4 3 3 3 3 4 3 4 4 4 4 4 3 4 4 3 4 4
```

To get my cluster membership vector, I need to "cut" my tree to yield sub-tree or branches with all the members of a given cluster residing on the same cut branch. The function to do this is called 'cutree()'

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

```
 [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
```

```r
plot (x, col=grps)
```



Principal Component Analysis (PCA)

The base R function for PCA is called ‘prcomp()’

```r
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

> 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  5
```

```r
nrow(x)
```

```
[1] 17
```

```r
ncol(x)
```

```
[1] 5
```

There are 17 rows and 5 columns. They functions that could be used are 'dim()', 'nrow()', and 'ncol()'

```r
head(x)
```

```
               X England Wales Scotland N.Ireland
1         Cheese     105   103      103        66
2  Carcass_meat      245   227      242       267
3    Other_meat      685   803      750       586
4           Fish     147   160      122        93
5 Fats_and_oils      193   235      184       209
6         Sugars     156   175      147       139
```

Note how the minus indexing works

```r
rownames(x) <- x[,1]
x <- x[,-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
```

```r
dim(x)
```

```
[1] 17  4
```

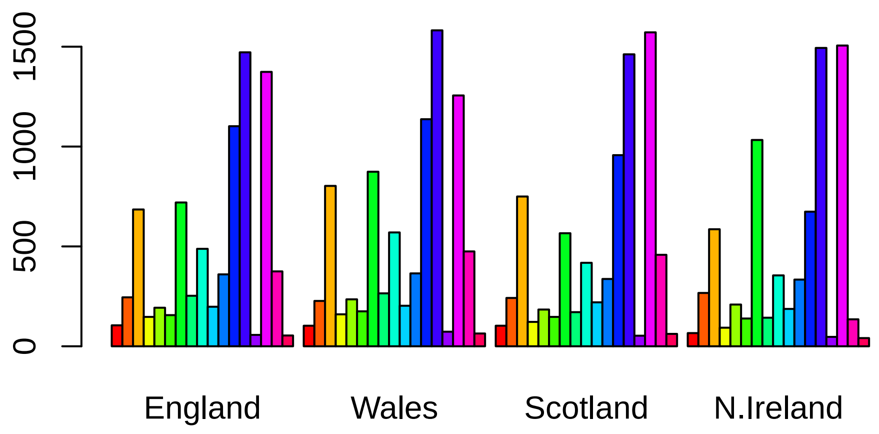
```r
x <- read.csv(url, row.names=1)
head(x)
```

```
               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?

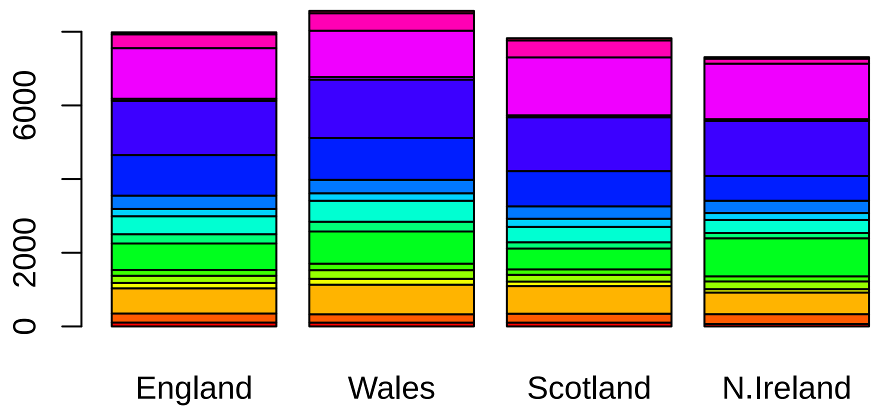
I prefer the second approach using 'x <- read.csv(url, row.names=1)' because it is one less line of code and less defining of variables. The second approach is more robust than the first one because if you continue running the code, it keeps on eliminating the first column and treat it as the row names.

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



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

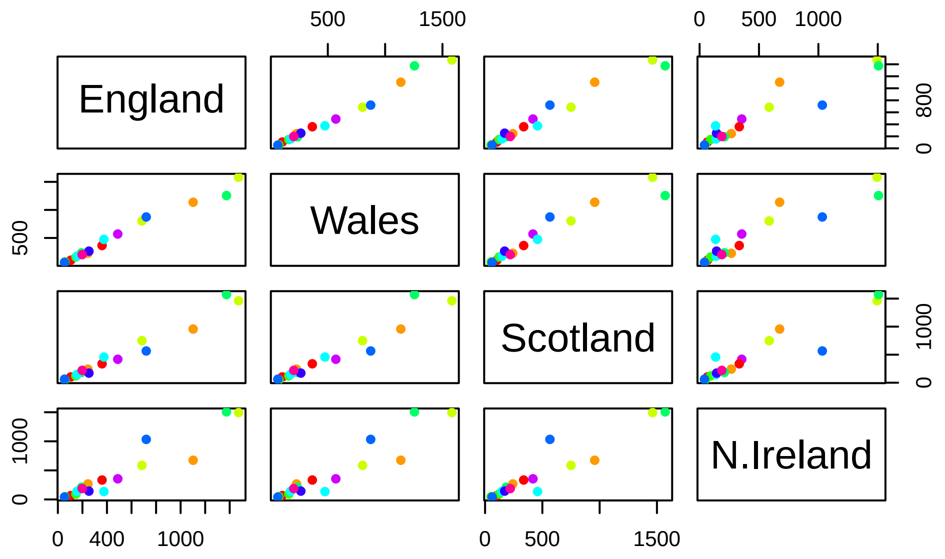
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



The optional argument that can be changed to change the plot it to replace the 'beside=T' to 'beside=F'

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(10), pch=16)
```



The 'pairs()' code produces a matrix of scatterplots. X is the input data. 'col=rainbow(10)' produces the rainbow color. 'pch=16' produces the shape and size of the points. The resulting figure compares the data from 2 countries. The closer the point is to the middle diagonal, the more similar the data from that specific category is and the farther, the less similar.

> 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 the two are the dark blue and orange points.

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 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

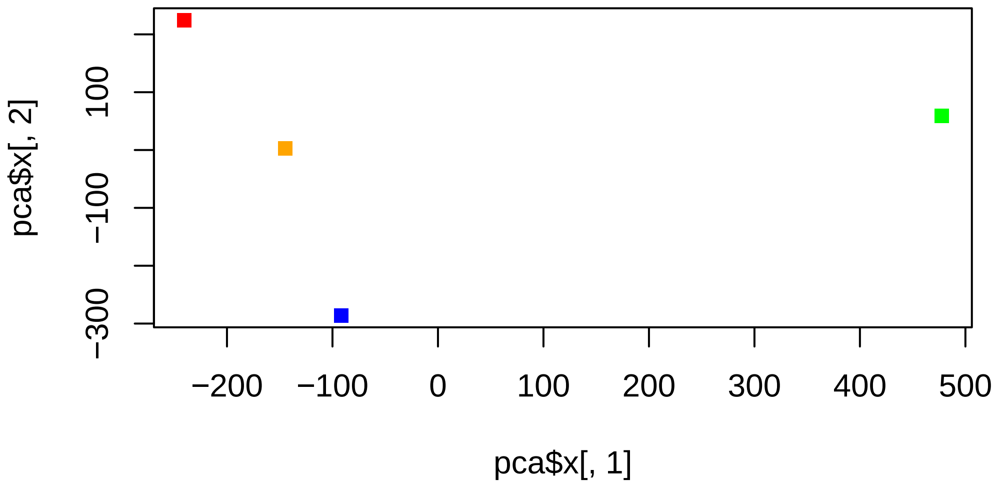
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

A "PCA plot" (a.k.a "Score plot", PC1vsPC2 plot, etc.)

```
pca$x
```

```
                 PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

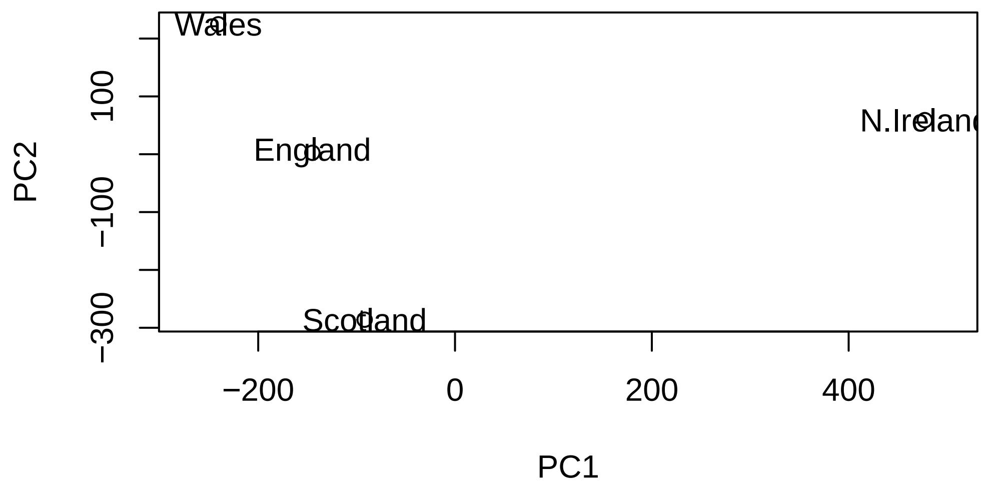
```
plot(pca$x[,1], pca$x[,2],
     col=c("orange", "red", "blue", "green"), pch=15)
```



This code summarizes the similarity between the countries from the data set.

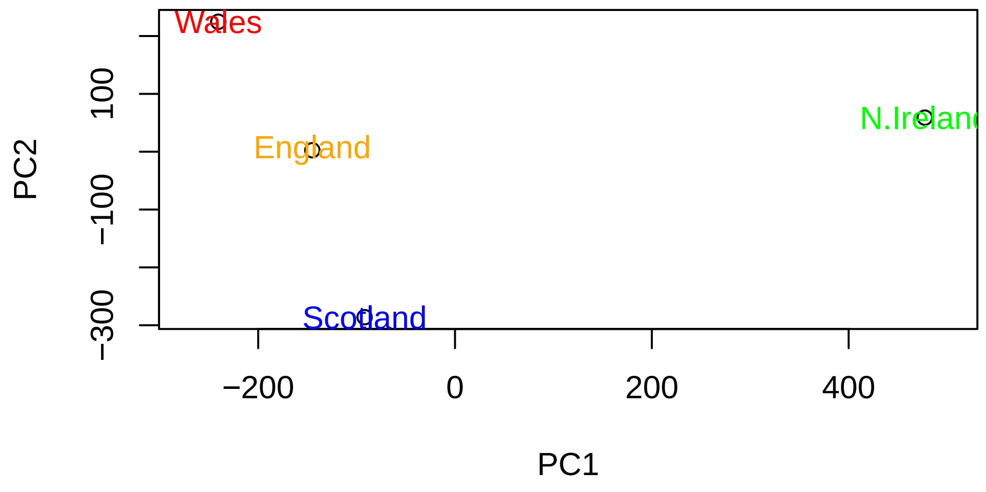
Plot PC1 vs PC2

```
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.

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "green"), pch=15)
```



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

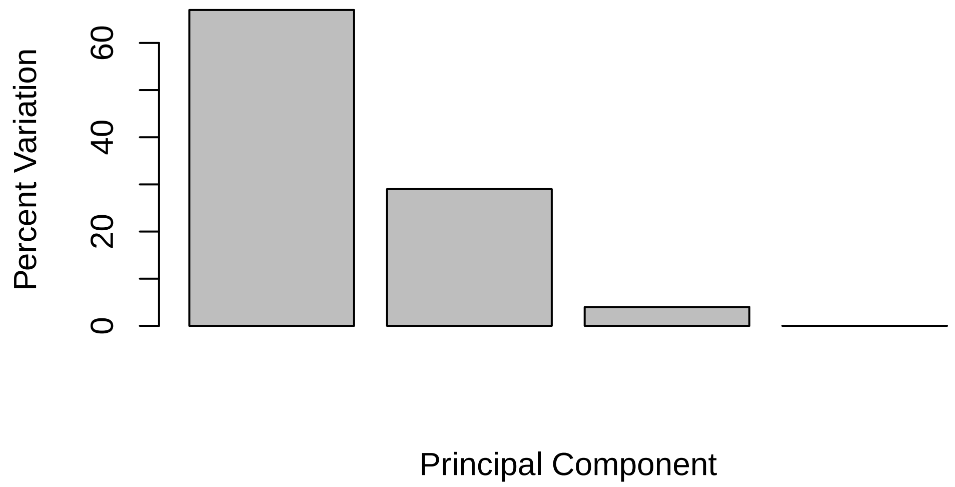
```
[1] 67 29  4  0
```

or the second row here…

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

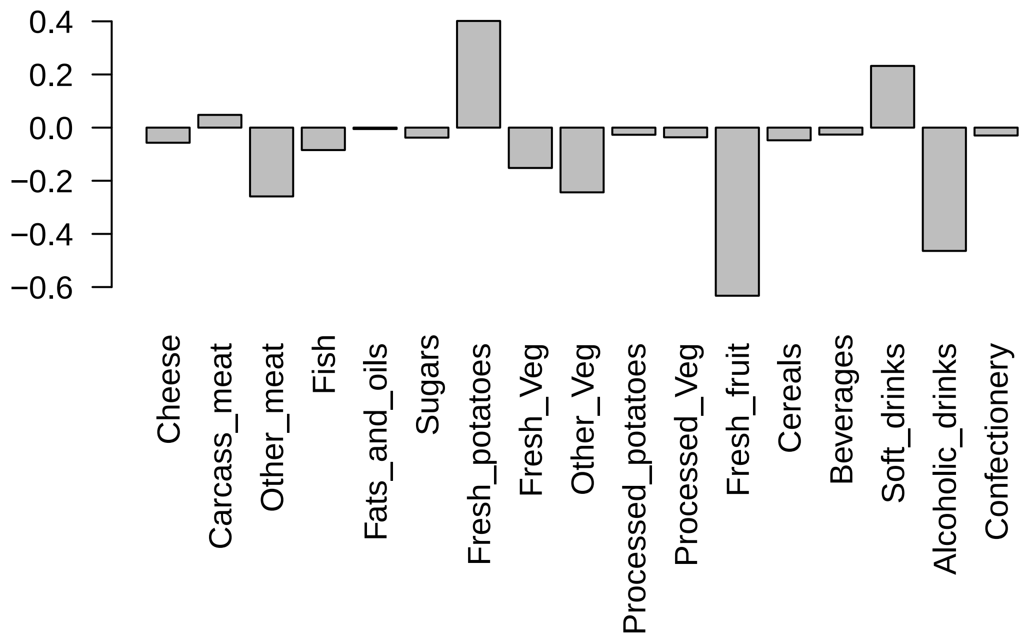
```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



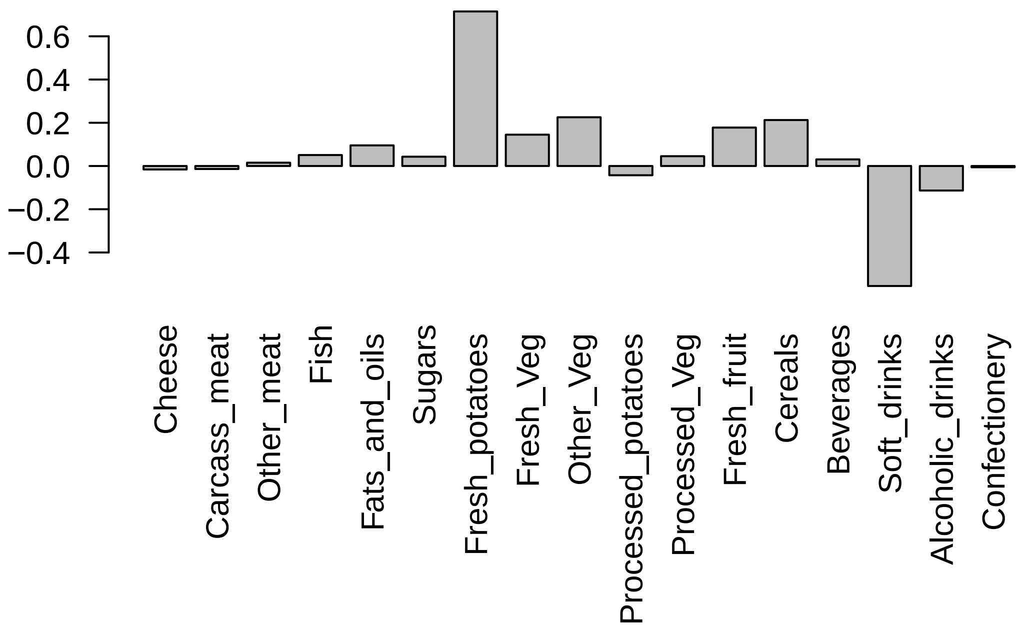
Lets focus on PC1 as it accounts for $>$ 90% of variance

```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?

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


0.6
0.4
0.2
0.0
−0.2
−0.4
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