Machine Learning 1

Kai Movellan

10/21/2021

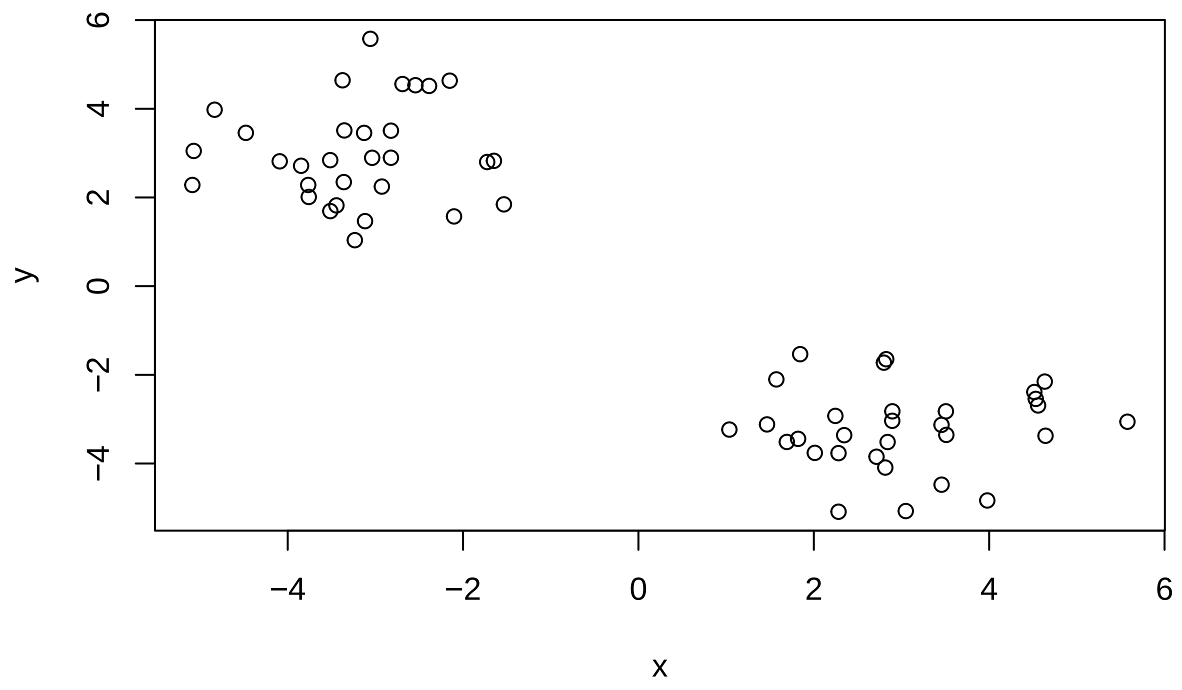
First up is clustering methods

K means clustering

The function in base R to do Kmeans is called `kmeans()`

First make up some data where we know the answer should be:

```r
#rnorm makes a random set of data close to -3
tmp <- c(rnorm(30,-3), rnorm(30,3))
#tmp
#hist(tmp)
x<- cbind(x=tmp,y=rev(tmp))
#x
plot(x)
```



Q. Can we use kmeans() to cluster this data setting k 2 and nstart 20?

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.214294  2.993471
## 2  2.993471 -3.214294
##
## 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] 61.86102 61.86102
##  (between_SS / total_SS =  90.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
#clustering vector: says for all the data points which group it belongs to
```

Q. How many points are in each cluster?

```
km$size
```

```
## [1] 30 30
```

Q. What 'component' of your result object details cluster assignment/ membership?

```
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
```

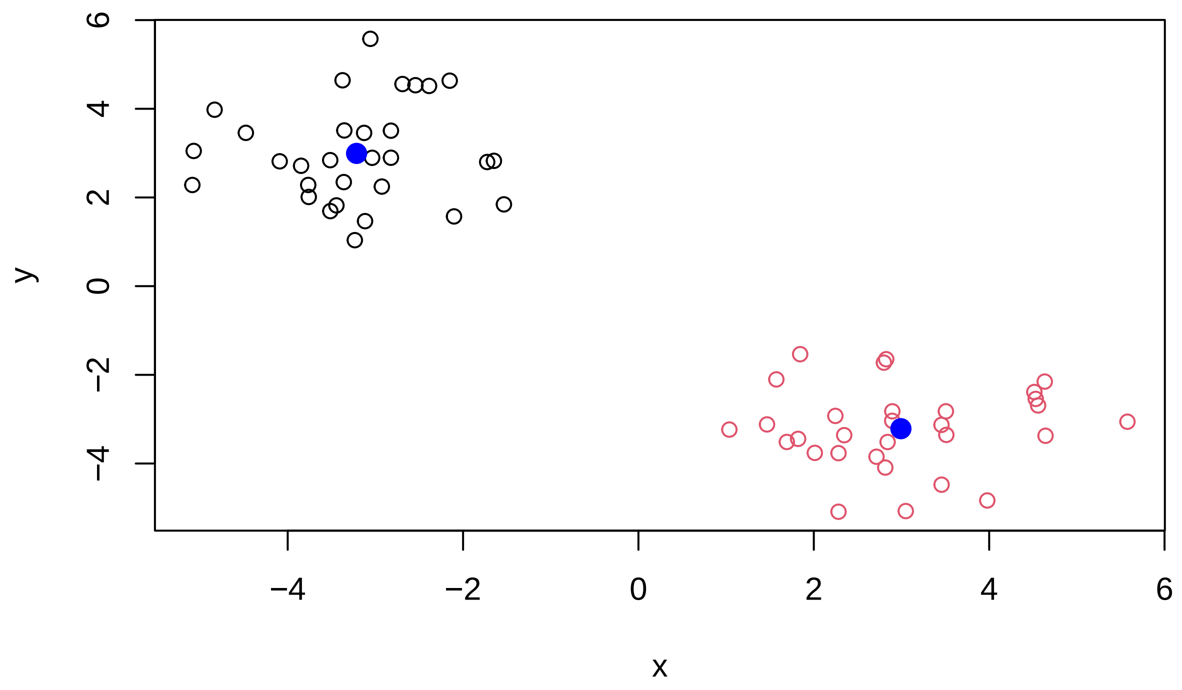
Q. What 'component' of your result object details cluster centers?

```
km$centers
```

```
##           x         y
## 1 -3.214294  2.993471
## 2  2.993471 -3.214294
```

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

```
plot(x,col=km$cluster)
points(km$centers, col="blue", pch=20, cex=2)
```



Hierarchical Clustering

analysis on a set of dissimilarities and methods for analyzing it.

Analuyze this dara with hclust()

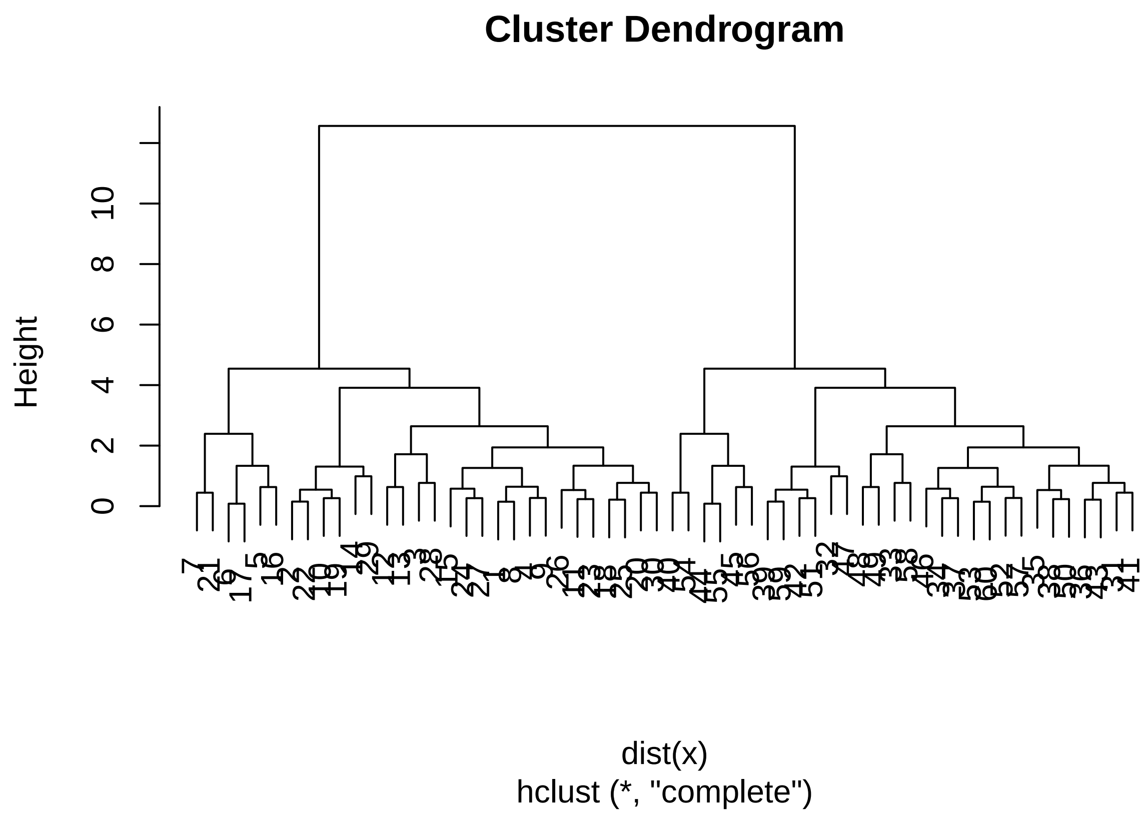
Demonstrate the use of dist(), hclust(), plot(), and cutree() functions to do clustering Generate aenarograms and return cluster assignment/ membership vector…

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

```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

There is a plot method for hclust result objects. Let's see it

```
plot(hc)
```



To get our cluster membership vector we have to do a wee bit more work. We have to “cut” the tree with what we think makes sense. For this we use `cutree()` function

```
cutree(hc,h=6)
```

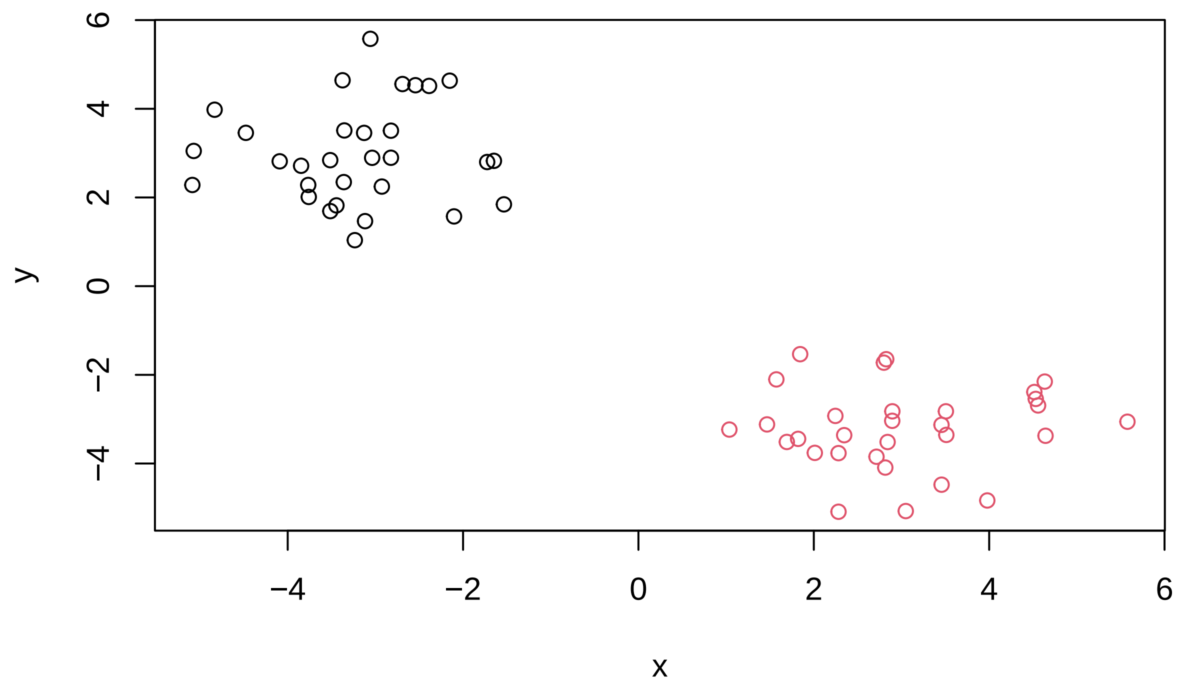
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call `cutree()` setting k=the number of grps/clusters you want

```
grps<-cutree(hc, k=2)
```

Make our result plot

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



#Principal Component Analysis (PCA)

PCA of UK food data

Read data from website and try a few visualizations

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

```
# Complete the following code to find out how many rows and columns are in x?
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

```
dim(x)
```

```
## [1] 17  5
```

Pants! this should be 17x4

```
# Note how the minus indexing works
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
```

The dimmensions are 17x4

> 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?

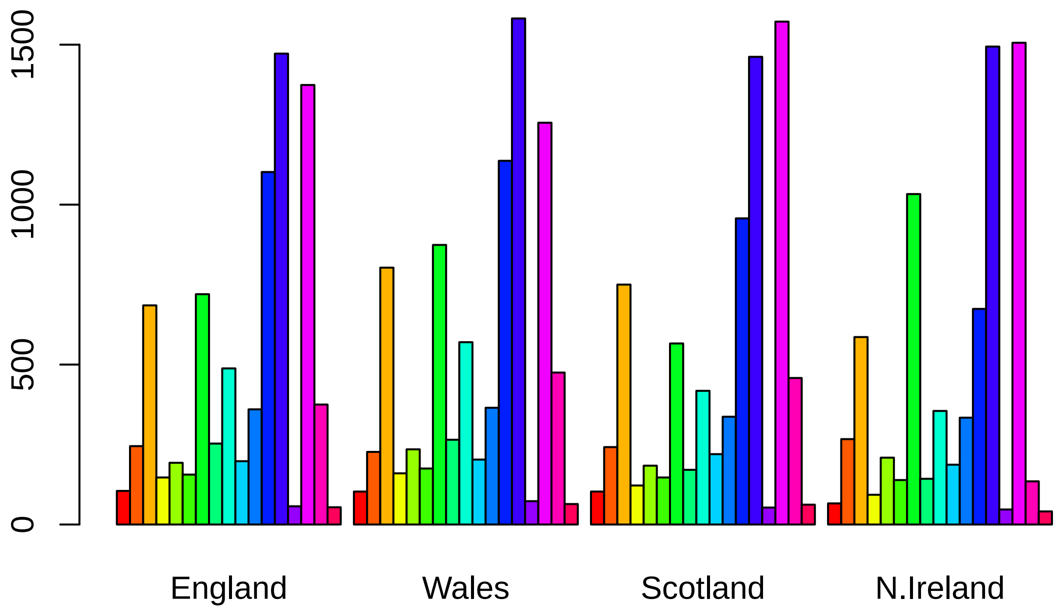
```
x <- read.csv(url, row.names=1)
dim(x)
```

```
## [1] 17  4
```

The minus indexing removes a collumn every time the code is ran

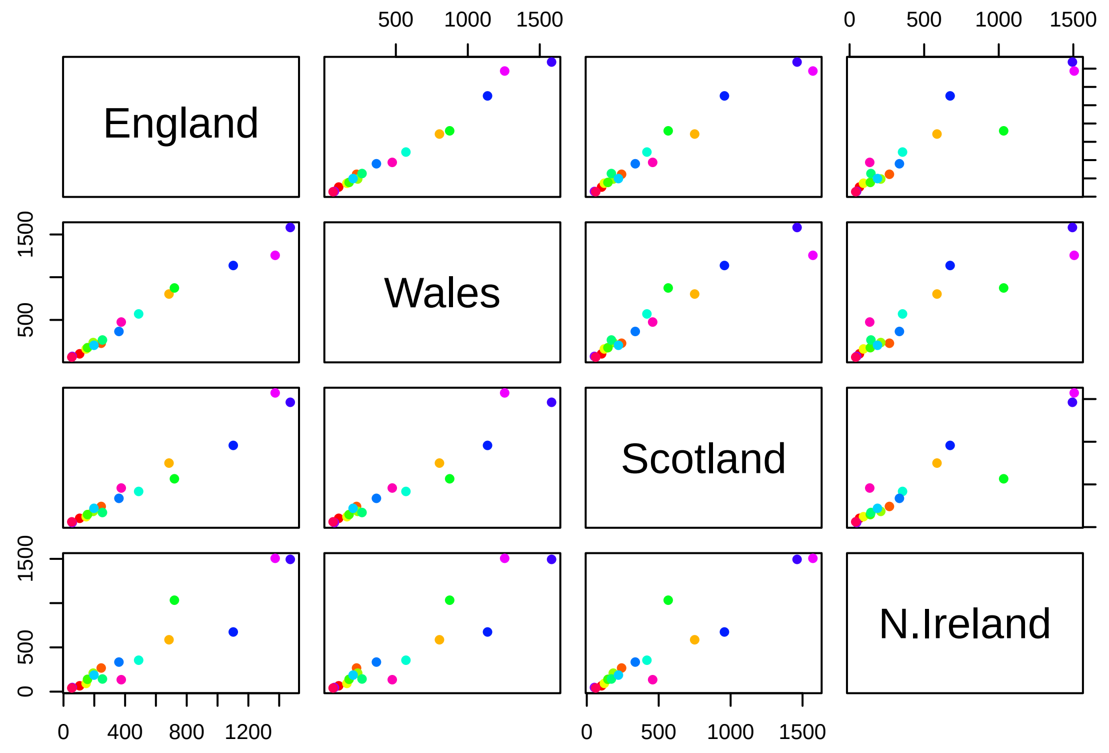
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

```
cols<-rainbow(nrow(x))
barplot(as.matrix(x),col=cols,beside=TRUE)
```



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?

```r
#A plot of all possible pairs of countries
pairs(x,col=cols,pch=16)
```



If a point lies on the diagonal the data is similar. When the points aren't on the diagonal line the data is different

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

PCA to the rescue!! The main base R PCA function is called `prcomp()`and we will need to give it the transpose of the input data!

```r
#t(x)
# 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
```

```
attributes(pca)
```

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

```
pca$sdev
```

```
## [1] 3.241502e+02 2.127478e+02 7.387622e+01 4.188568e-14
```

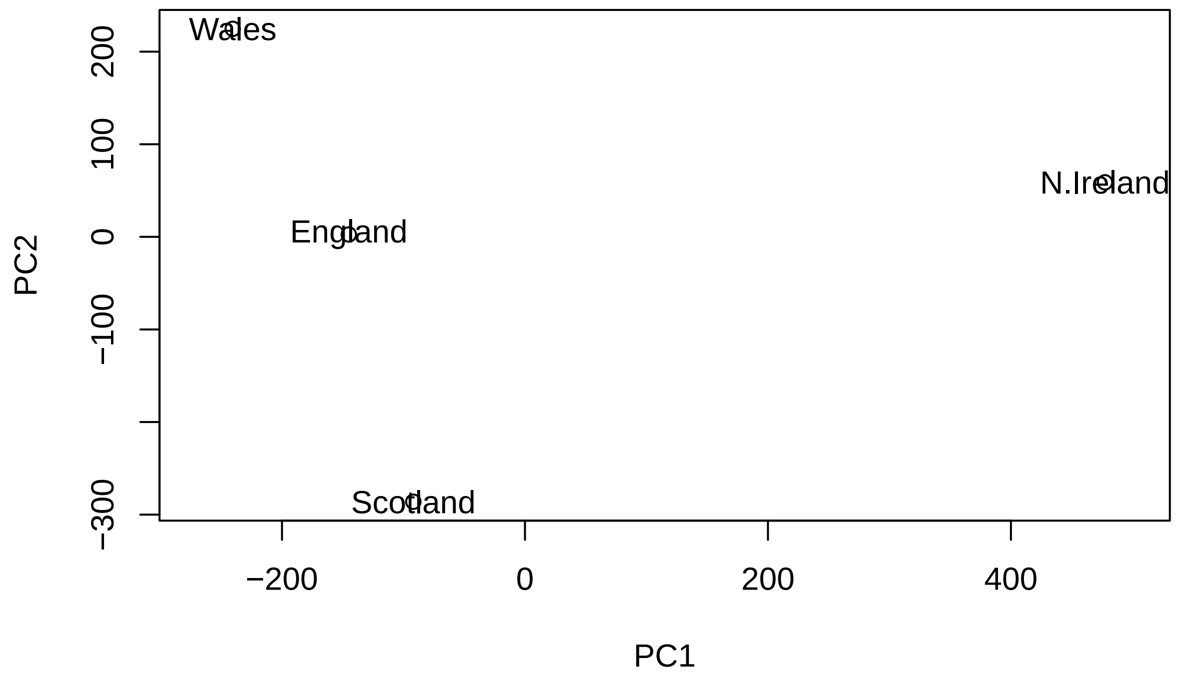
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points

To make our new PCA plot (PCA score plot) we access `pca$x`

```
pca$x
```

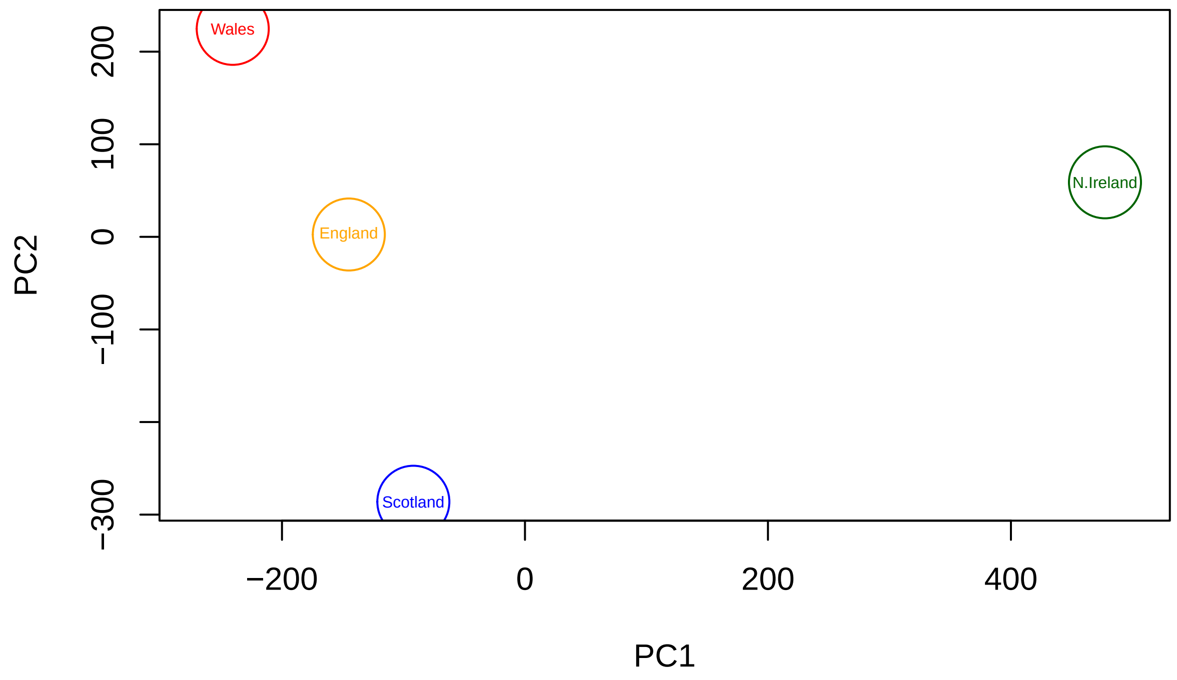
```
##                  PC1         PC2         PC3           PC4
## England   -144.99315    2.532999 -105.768945  2.842865e-14
## Wales     -240.52915  224.646925   56.475555  7.804382e-13
## Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
## N.Ireland  477.39164   58.901862    4.877895 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

```
# Plot 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.

```
country_cols <- c("orange","red","blue", "dark green")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500),pch=1, cex=5,col=country_cols)
text(pca$x[,1], pca$x[,2], colnames(x), col=country_cols, cex=0.5)
```



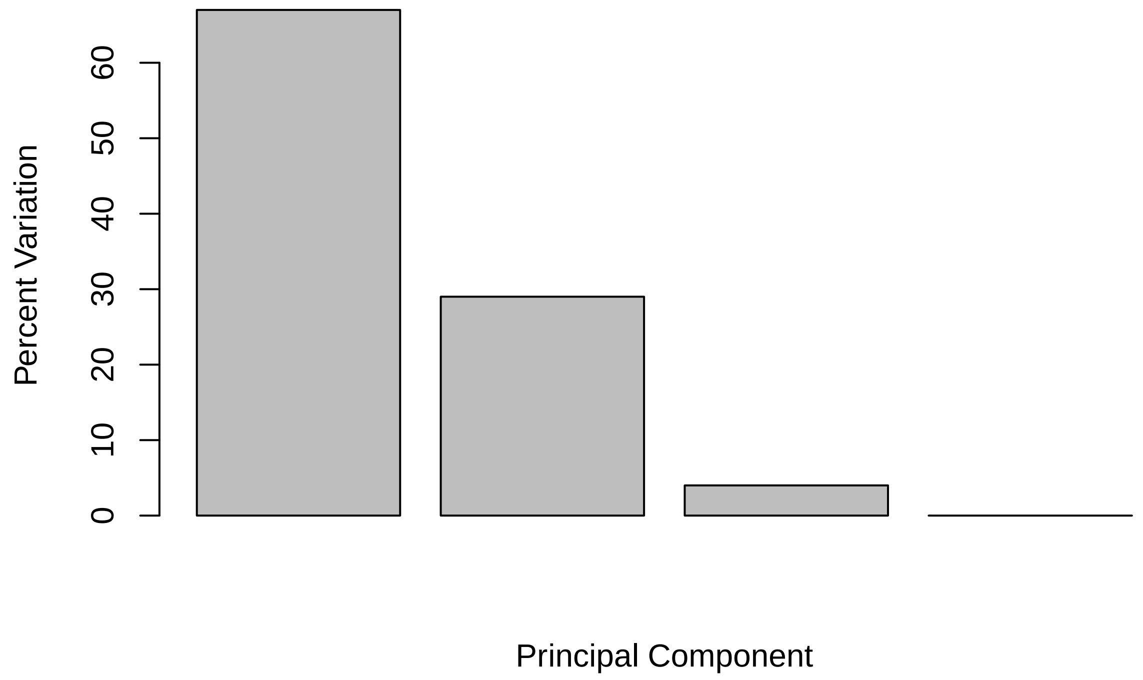
```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

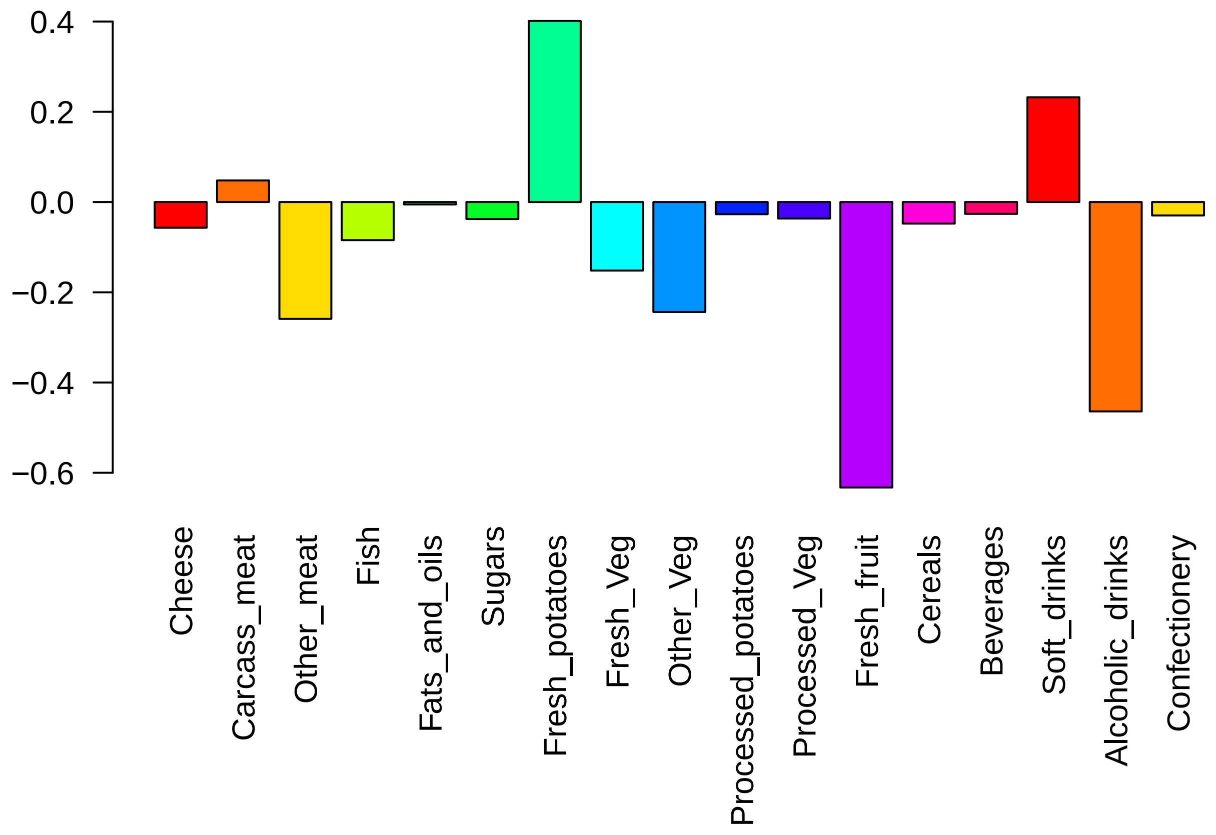
```
## or the second row here...
z <- summary(pca)
z$importance
```

```
##                              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")
```

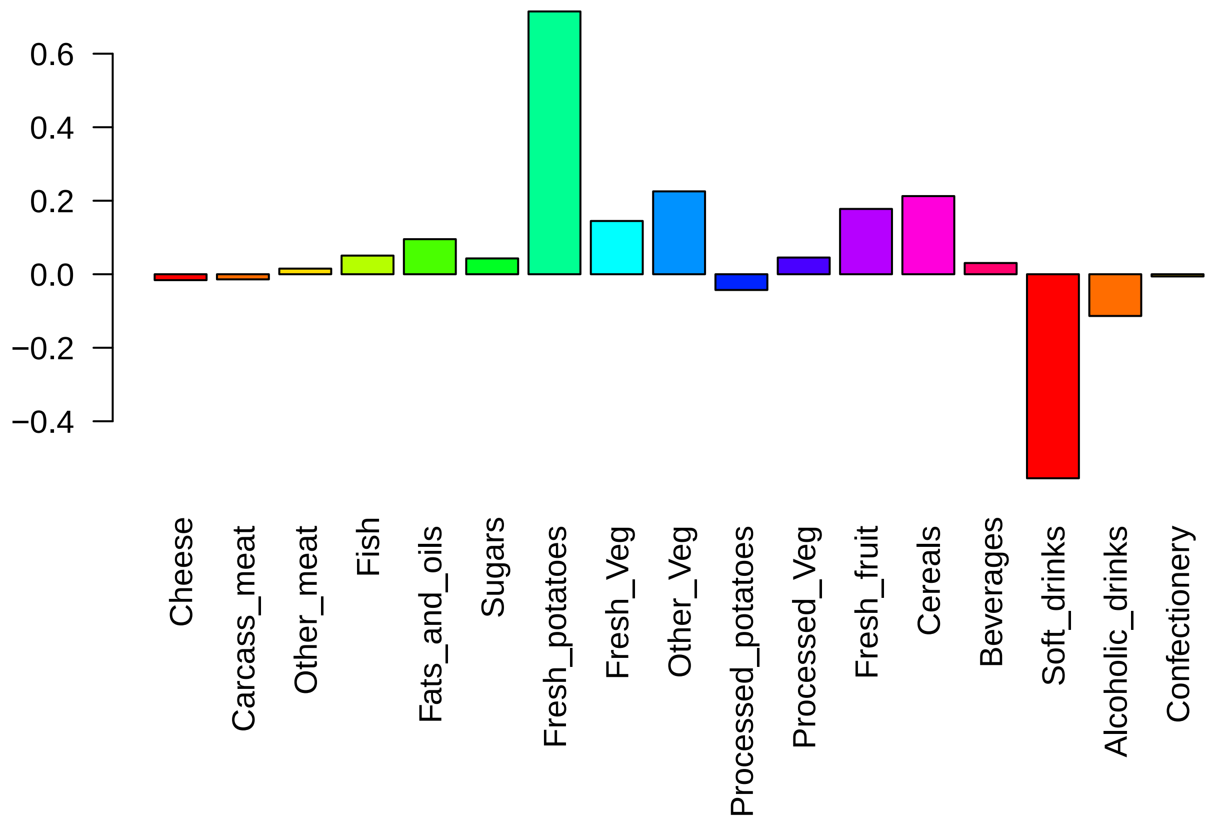


```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2, col=rainbow(14) )
```



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

```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2, col=rainbow(14))
```


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