# Class 08

Nadia Haghani

10/21/2021

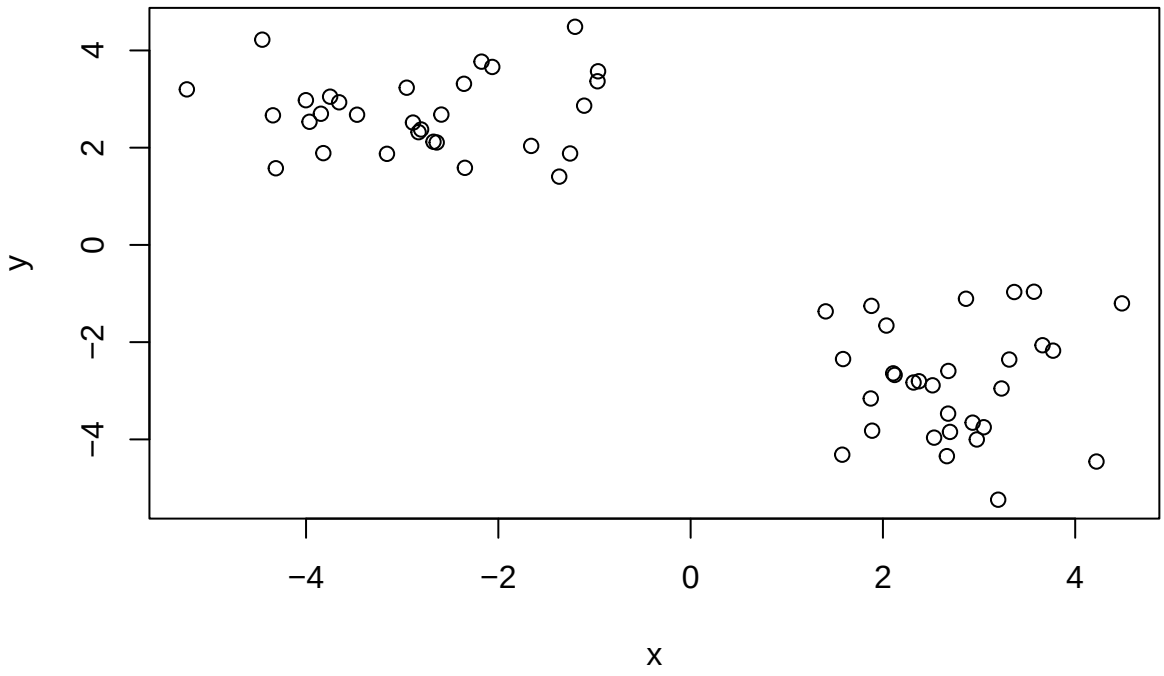
First up is clustering methods

## Kmeans clustering

The function in base R to do Kmeans clustering is called 'kmeans()'.

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

```
tmp <- c(rnorm(30, -3), rnorm(30, 3)) #rnorm gives random data around ( , __)
x <- cbind(x=rev(tmp), y=tmp) #cbind binds columns, rbind binds rows
plot(x)
```



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

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1  2.719356 -2.828879
## 2 -2.828879  2.719356
##
## 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 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
##
## Within cluster sum of squares by cluster:
## [1] 57.42523 57.42523
##  (between_SS / total_SS =  88.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

How many points 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 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
```

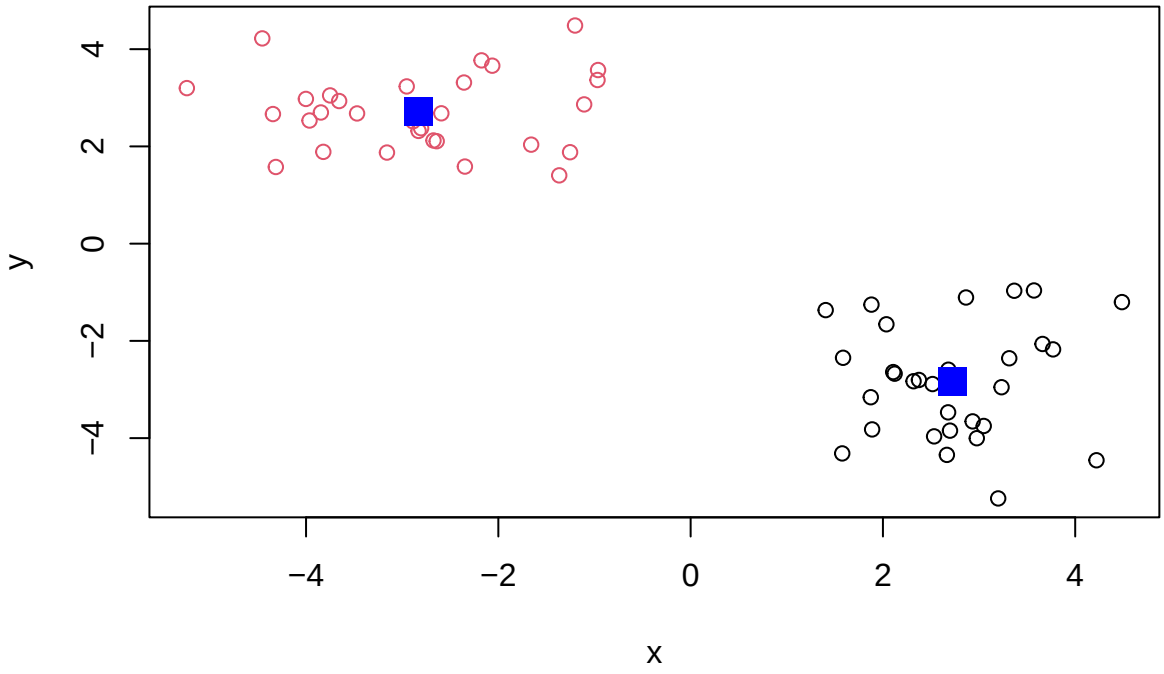
Q. What 'component of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1  2.719356 -2.828879
## 2 -2.828879  2.719356
```

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

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




## hclust - Hierarchical Clustering

A big limitation with k-means is that we have to tell it K (the number of clusters we want)

Analyze this same dataset with hclust()

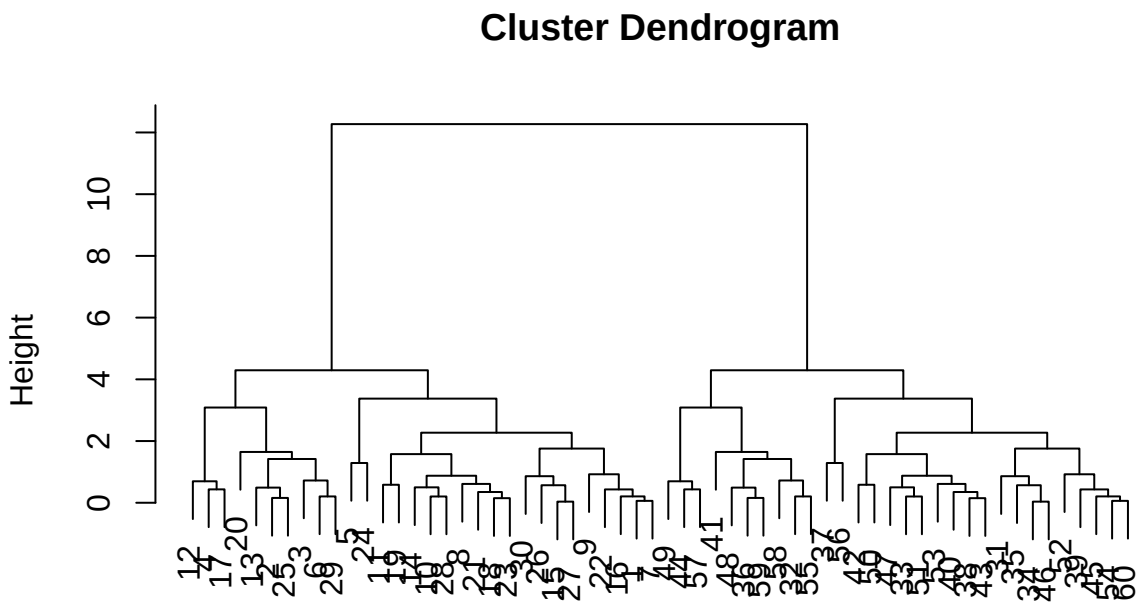
Demonstrate use of dist(), hclust(), plot() and cutree() functions to do clustering, generate dendrograms and return cluster assignments

```
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 memebership vector, we have to do a wee bit more work. We have to "cut" the tree where we think it makes sense. For this we use the 'cutree()' function.

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

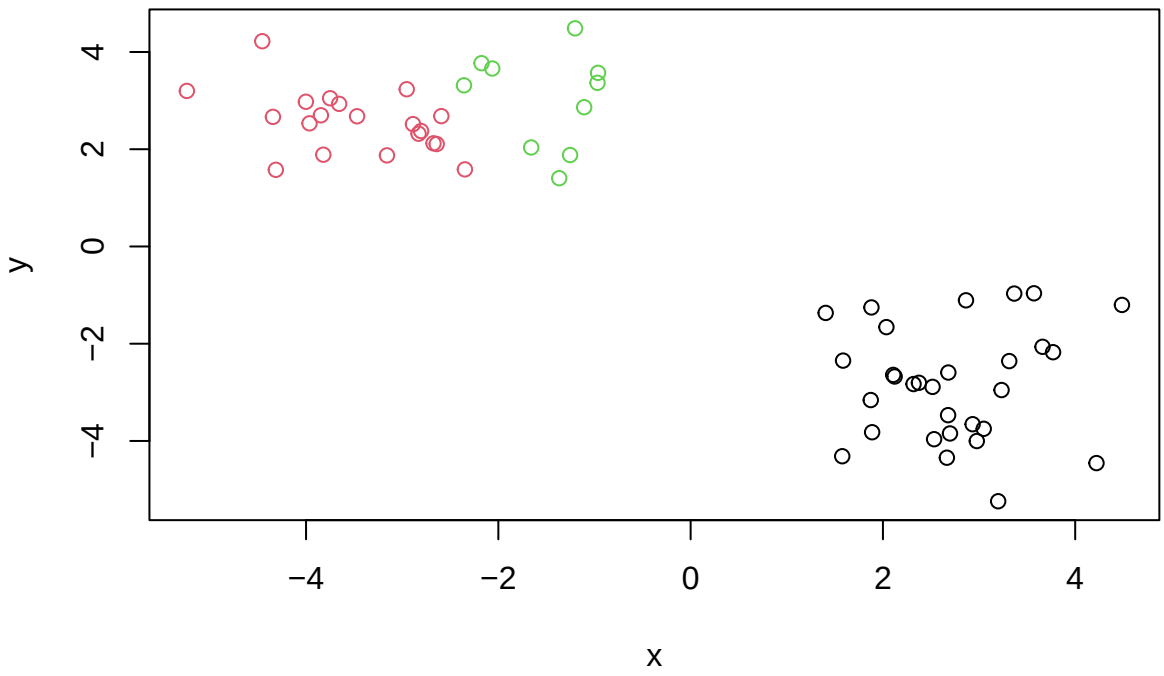
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call 'cutree()' setting k = numer of grps/clusters you want.

```
grps <- cutree(hc, k=3)
```

Make our results plot

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




# 1. PCA of UK food data

Read URL

```
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?

```
nrow(x)
```

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

```
ncol(x)
```

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

So, there are 17 rows and 5 columns in our dataframe. We can also use the 'dim()' function:

```
dim(x)
```

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

Check your data to make sure nothing odd happened during its upload

```
View(x)
```

The first column should set to row-names. We can do this in two ways

The first way is setting the rownames to the first column in x.

```
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 problem is that this code overrides other code, so a better way to write this code is to correct row-names while reading the data file.

```
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
```

Now, we can check the dimensions again.

```
dim(x)
```

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

We see there are 17 rows and 4 columns now.

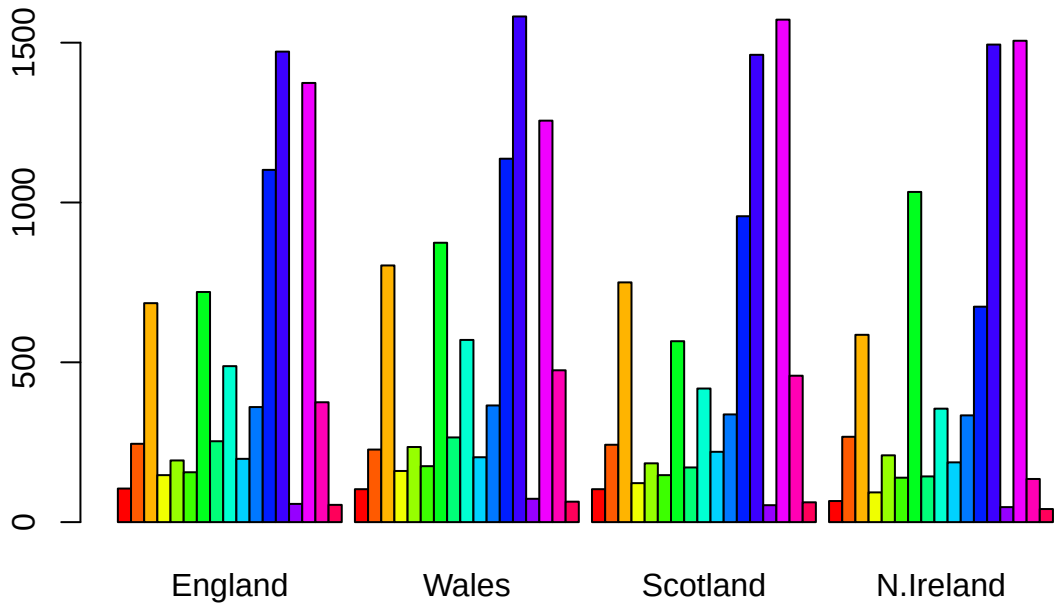
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The second approach is more robust than the other because it will make sure while reading the dataframe, we set the first column to the row names.

## Spotting differences and trends

Plot our data in a bar plot.

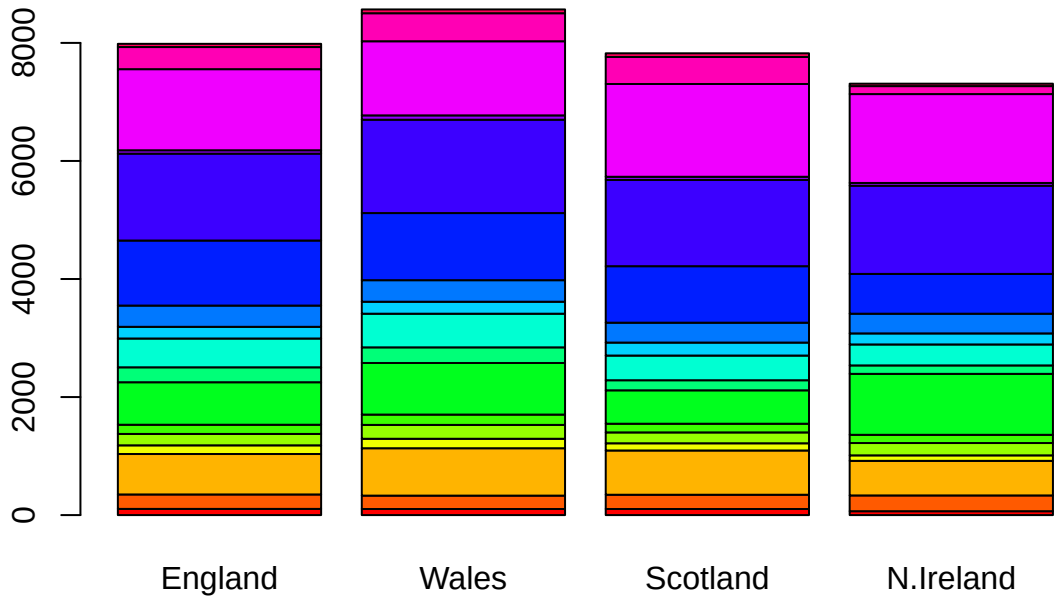
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




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

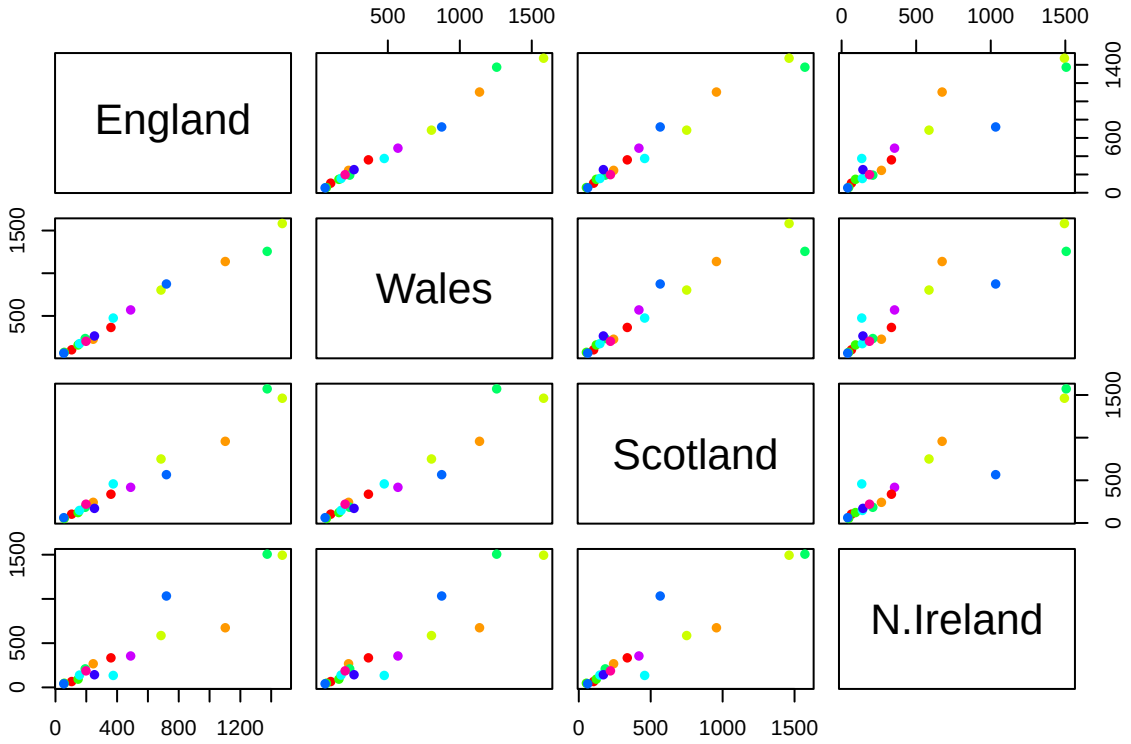
Take out 'beside = T'. If 'beside = T' the columns are portrayed as bars next to eachother.

```
barplot(as.matrix(x), 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(10), pch=16)
```




The pairs plot takes x as an input and generates a matrix of scatterplots. Color is rainbow, and pch is a graphical parameter.

The graphs compare a single country to every other country.

If a given point lies on the diagonal for a given plot, then the value would be the same for both countries. Northern Ireland seems to be the most dissimilar from other countries, as told by the points in the 4th row being the furthest from the diagonal line.

> 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!

Use the prcomp() PCA function

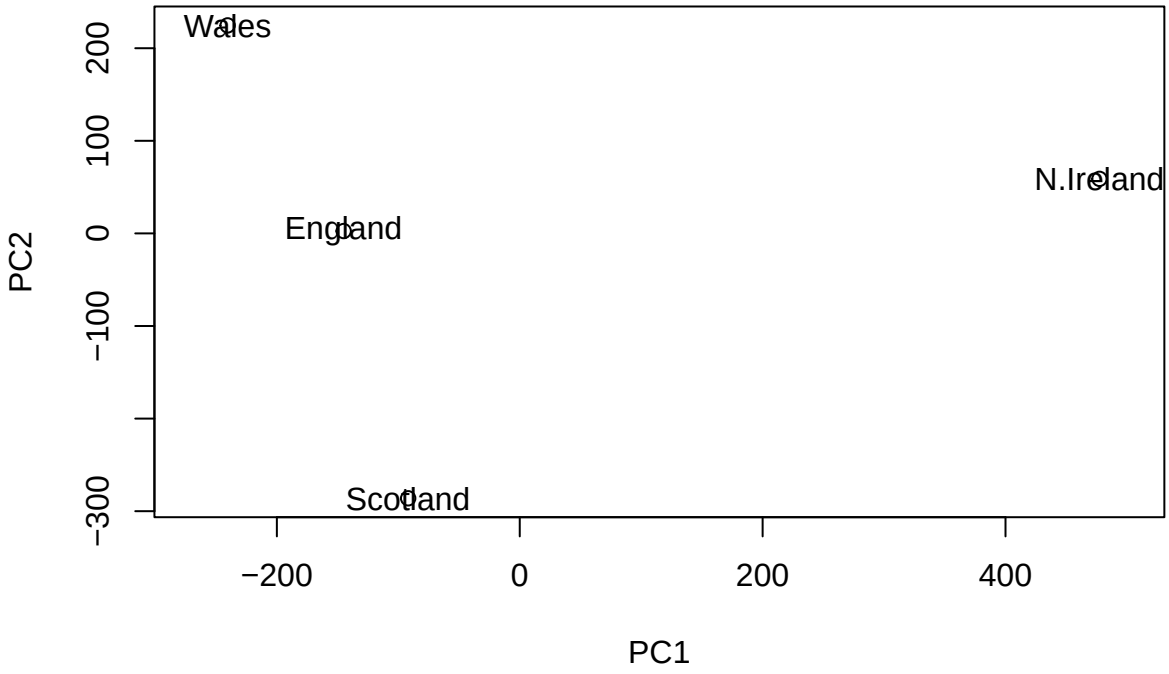
```
pca <- prcomp(t(x))
summary(pca)
```

```
## 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.

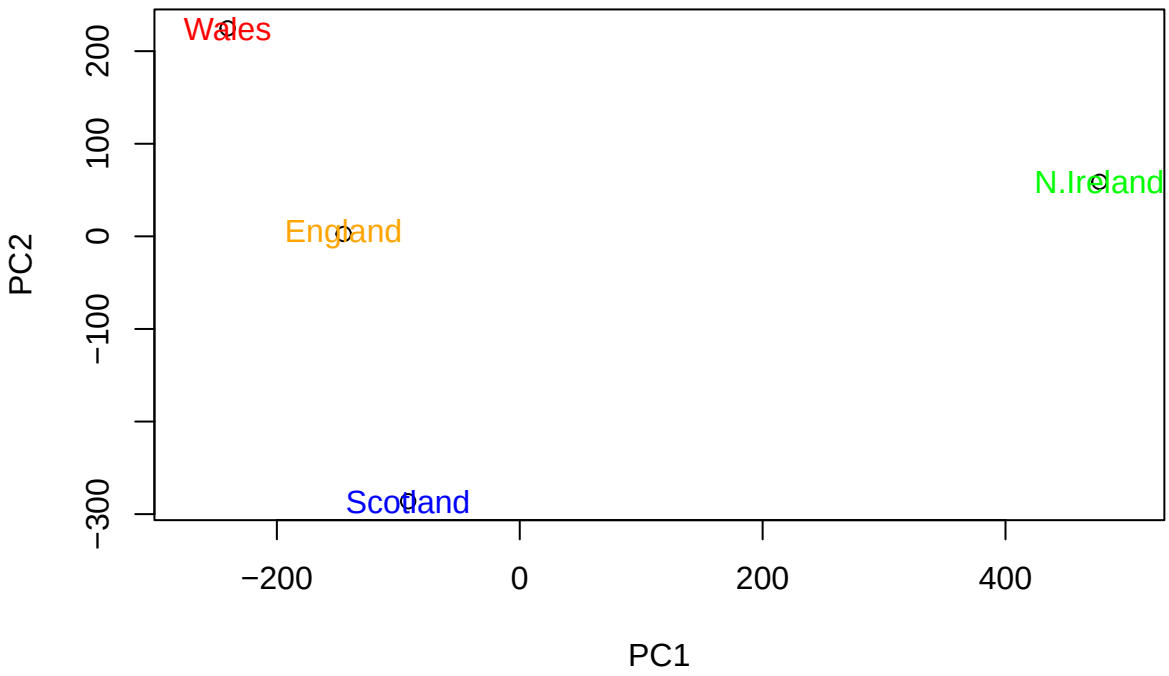
Plot PC1 vs PC2 Add text labels over the data points

```
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"))
```




We can use the square of pca$sdev , which stands for “standard deviation”, to calculate how much variation in the original data each PC accounts for.

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

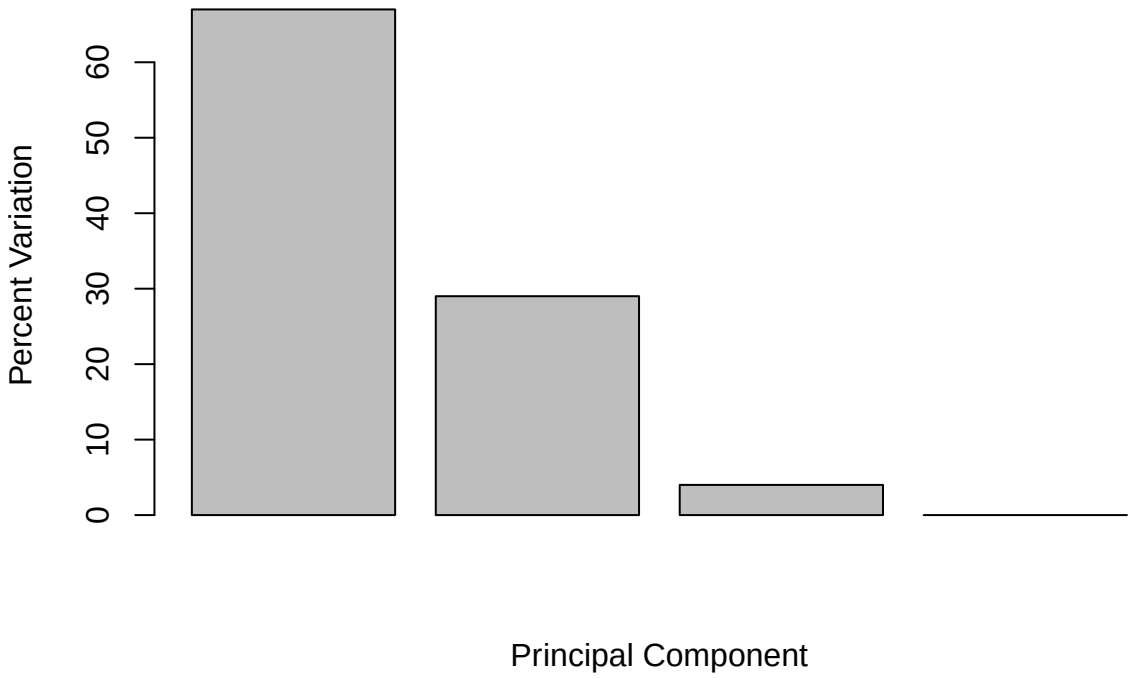
```
## [1] 67 29  4  0
```

```
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
```

Now, summarize the eigenvalues in a barplot.

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

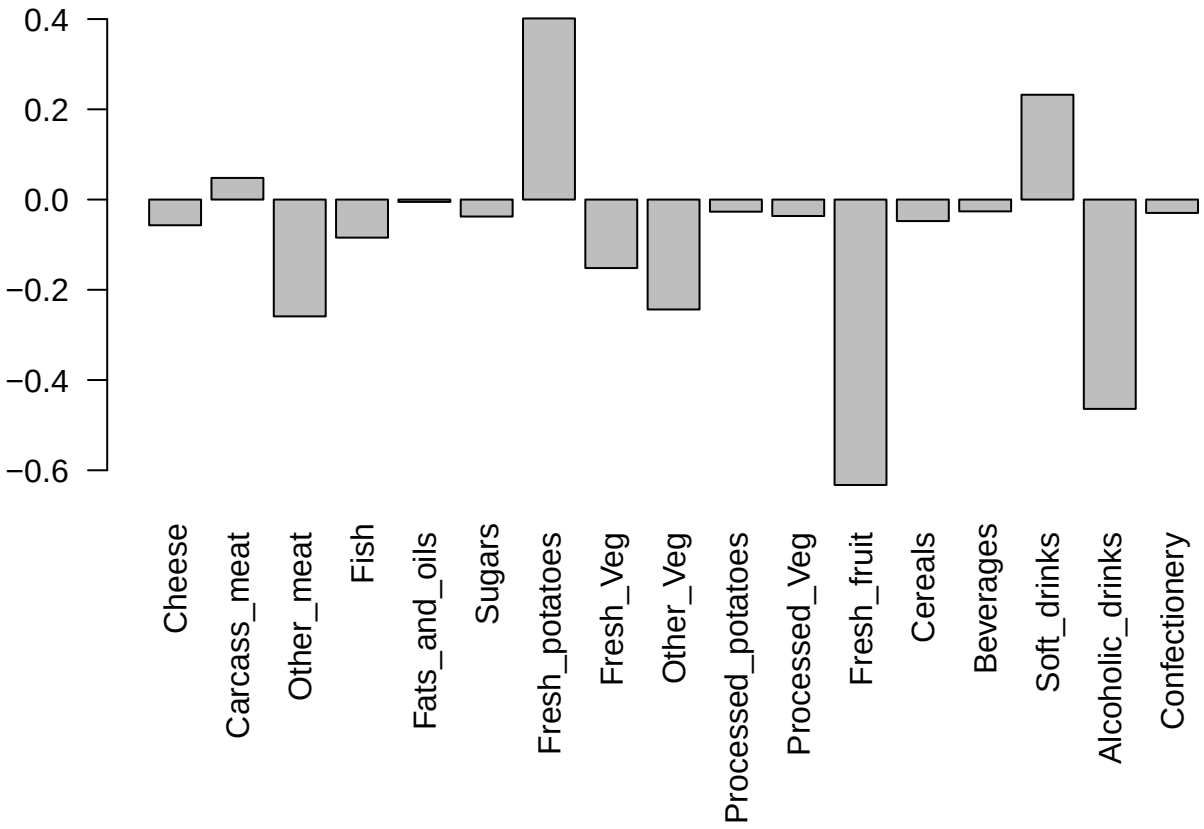



## Digging deeper (variable loadings)

Look at the influence of each of the original variables on the principle components.

Focus on PC1

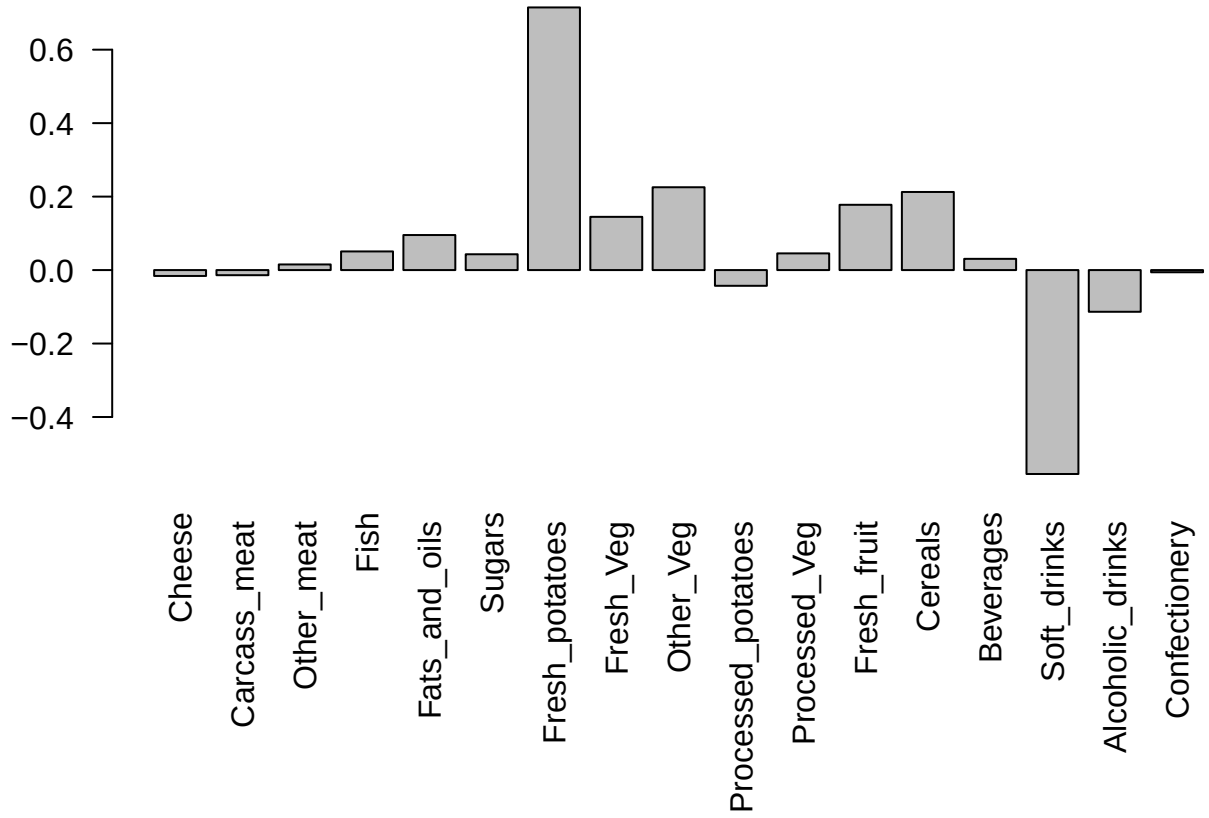
```
par(mar=c(10,3,0.35,0))
barplot(pca$rotation[,1],las=2)
```




We see that foods with the largest push Northern Ireland to the right positive side of the plot. We also see that foods with negative scores push other countries ot the left side of the plot.

> 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)
```



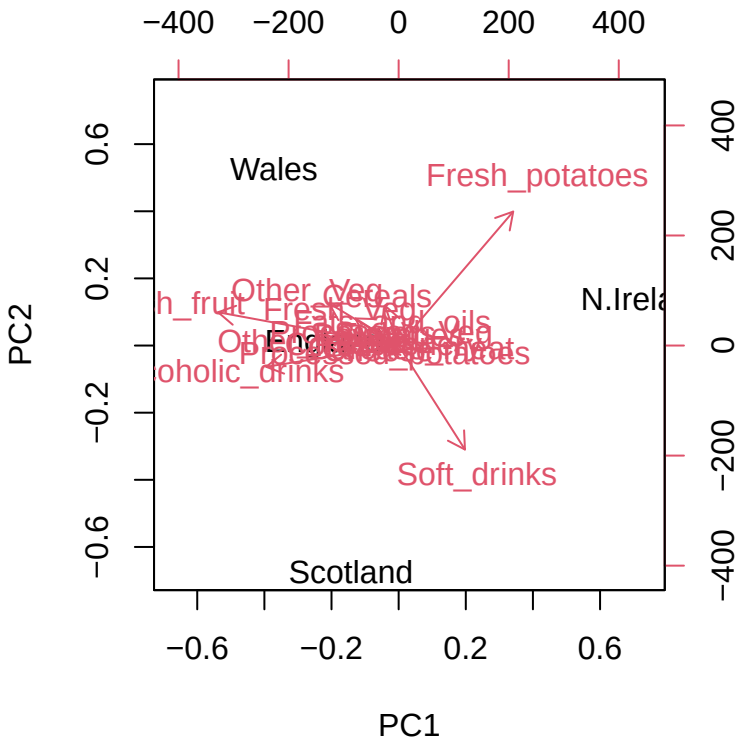

What two food groups feature prominantely and what does PC2 maninly tell us about?

Fresh potatoes and soft drinks feature prominently.

PC2 mainly tells us that fresh potatoes push N. Ireland to the right side of the plot while soft drinks push other countries to the left side of the plot.

## Bioplots

```
biplot(pca)
```




**2. PCA of RNA-seq data**