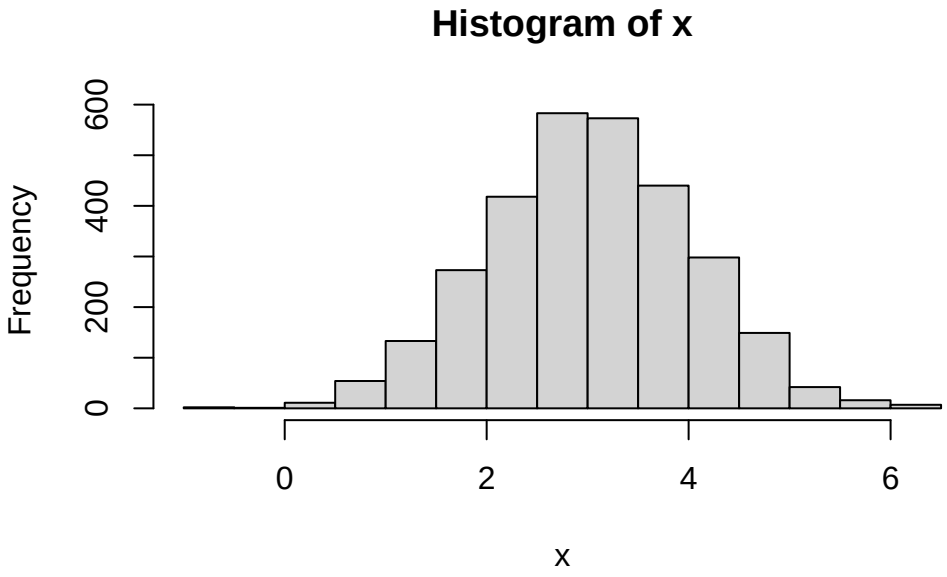
Class 7: Machine Learning

Sindy Chavez

K-means Clustering

Let's make up some data to cluster.

```
x <- rnorm(3000, 3)
# Random distribution of numbers, in a normal distribution
hist(x)
```




```
#This one is centered around 0
# Adding *, 3* results in the center of the histogram moving to 3, instead of 0 which is t
```

One set clustered around +3 and one around -3

```
rnorm(30, 3)
```

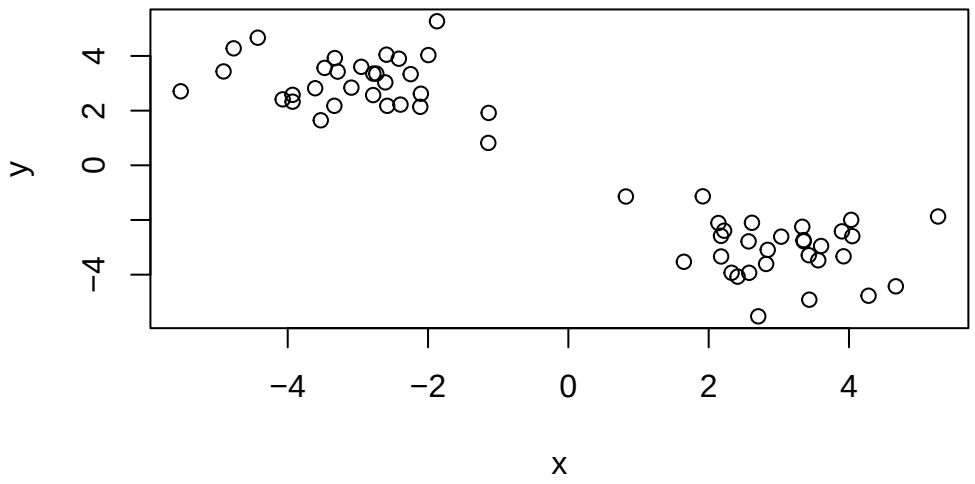
```
 [1] 3.659227 4.338327 3.019913 3.215759 4.285185 2.976799 2.826265 2.712500
 [9] 3.772274 4.436217 3.069992 3.208406 2.710089 3.736692 1.166280 3.831715
[17] 3.905674 2.061619 2.823966 2.086784 2.511651 3.709356 2.152233 3.141666
[25] 4.112923 3.088439 2.257716 2.825948 3.191279 3.592700
```

```
rnorm(30, -3)
```

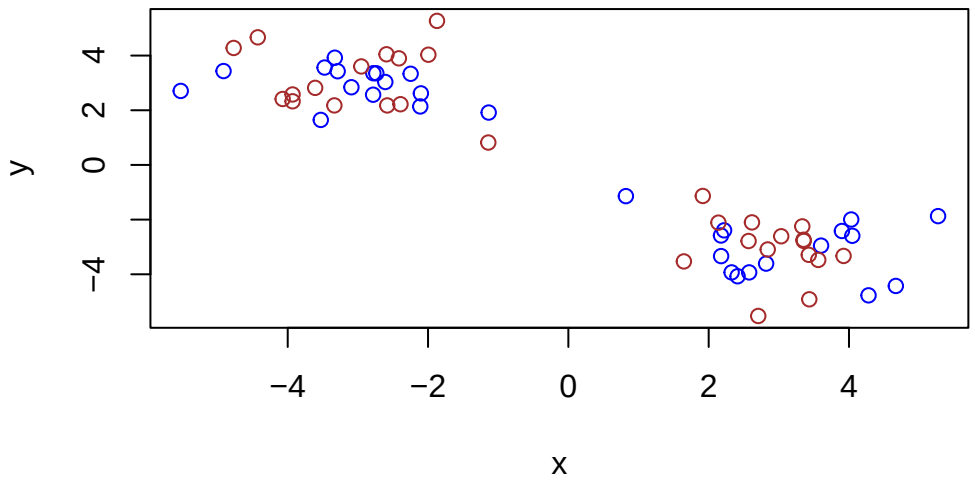
```
 [1] -4.670635 -3.680216 -1.440851 -2.213788 -3.593212 -1.988526 -3.881229
 [8] -5.237156 -3.032928 -3.114324 -2.084053 -1.550183 -2.392925 -1.960448
[15] -3.869964 -4.000878 -4.297271 -2.095683 -1.463641 -4.678704 -1.691476
[22] -2.305933 -2.573864 -4.986807 -3.739057 -2.634247 -4.056768 -2.960048
[29] -3.577801 -3.108046
```

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

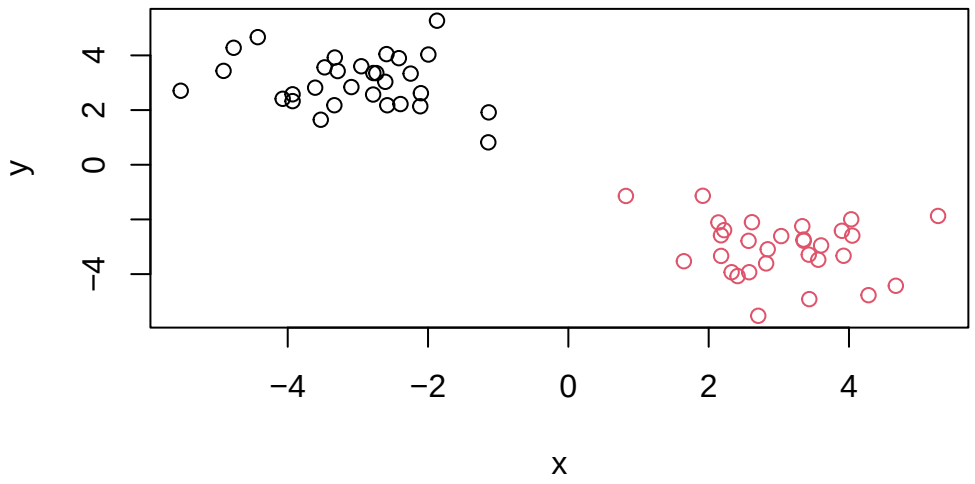
```
tmp <- c(rnorm(30, -3), rnorm(30,3))
x <- cbind(x=tmp, y=rev(tmp))
# rev - reverses the vector
# colors can also be numbers, 1 is black, 2 is red, etc
# cluster is indicated by 'cluster' so you can ask each to be colored differently
plot(x)
```




```
#plot(x, col=x$cluster)
plot(x, col=c("blue", "brown"))
```



```
# colors the dots alternating blue and brown, not very helpful

plot(x, col=c(rep(1,30), rep(2,30)))
```



Q. The function to do k-means clustering in base R is called 'kmeans()'. We give this our input data for clustering the the number of clusters we want 'centers'.

```
km <- kmeans(x, centers=4, nstart = 20)
km
```

```
K-means clustering with 4 clusters of sizes 17, 16, 14, 13

Cluster means:
          x         y
1  3.013121 -2.325821
2 -2.787617  2.351552
3 -3.362340  3.825423
4  3.073670 -4.010437

Clustering vector:
 [1] 2 2 3 3 2 2 2 3 2 3 2 2 2 3 3 2 3 2 3 2 2 2 3 3 3 3 3 2 2 4 1 1 1 1 1 1 4
```

```
[39] 4 1 4 4 1 4 4 4 4 1 4 1 1 1 4 1 1 4 1 4 1 1

Within cluster sum of squares by cluster:
[1] 22.18835 17.99011 21.93412 15.48890
 (between_SS / total_SS =  93.7 %)

Available components:

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

```
# kmeans() needs two arguments, x and centers
# nstart if centers is a number, how many random sets should be chosen, not sure what that
# clustering vector is which cluster the item is in, I'm not really sure what it means
# Always uses euclidean distance matrix(something about hypotenuses)
```

```
km$size
```

```
[1] 17 16 14 13
```

Q. What component of your result object

-

```
km$cluster
```

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

- cluster centers

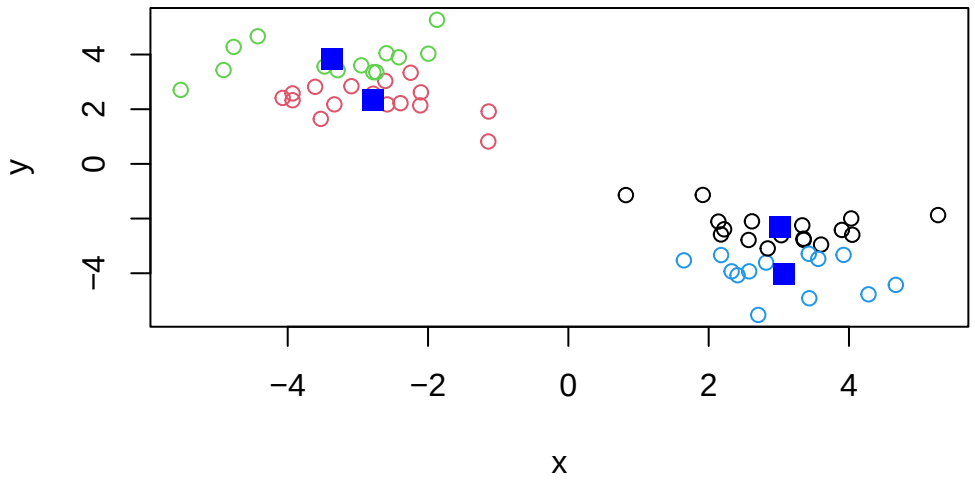
```
km
```

```
K-means clustering with 4 clusters of sizes 17, 16, 14, 13

Cluster means:
          x         y
1  3.013121 -2.325821
2 -2.787617  2.351552
3 -3.362340  3.825423
```

```
4  3.073670 -4.010437

Clustering vector:
 [1] 2 2 3 3 2 2 2 3 2 3 2 2 2 3 3 2 3 2 3 2 2 2 3 3 3 3 3 3 2 2 4 1 1 1 1 1 1 4
[39] 4 1 4 4 1 4 4 4 4 1 4 1 1 1 4 1 1 4 1 4 1 1

Within cluster sum of squares by cluster:
[1] 22.18835 17.99011 21.93412 15.48890
 (between_SS / total_SS =  93.7 %)

Available components:

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

More plot coloring

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=1.5)
```

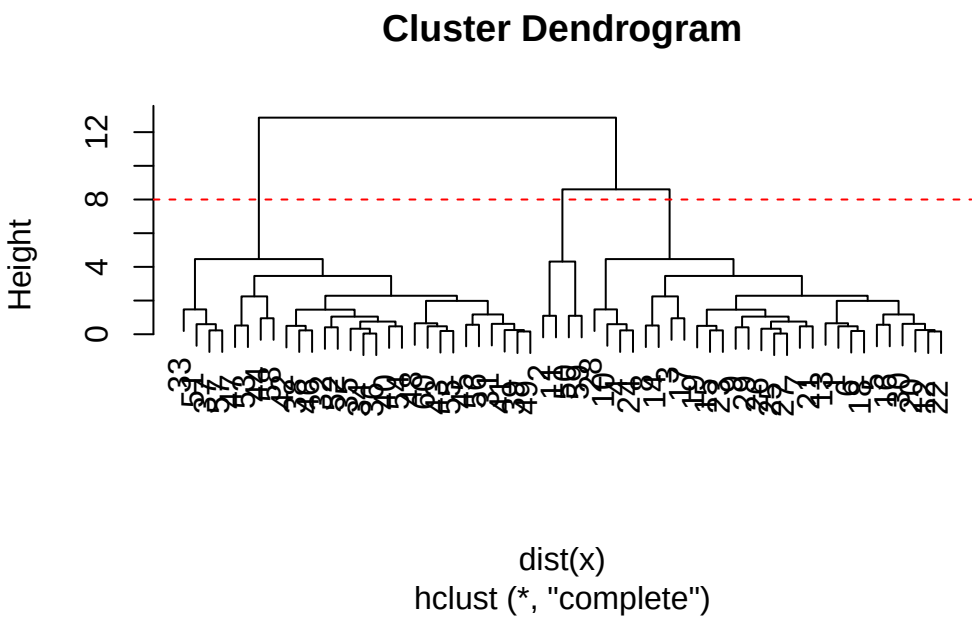


```
# pch made a square
# cex determines size of the points
```

The 'hclust()' function performs hierarchical clustering. The big advantage here is I don't need to tell it "k" the number of clusters To run 'hclust()'I need to provide a distance matrix as input (not the original data)

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

#only d (the dissimilarity structure) is a needed argument for this function
#function computes distance between the rows of data, only needs the argument x
#euclidean distance matrix
```

```
plot(hc)
abline(h=8, col="red", lty=2)
```



Pay attention the the lines going across in the dendrograms Shows structure in the data

To get my "main" result (cluster membership) I want to "cut" this tree to yield "branches"

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

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

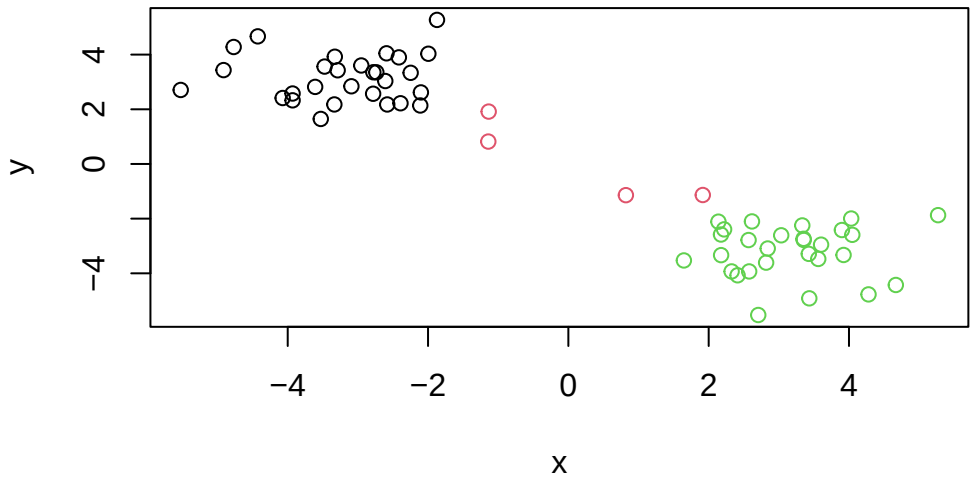
```
# Cuts a tree, e.g., as resulting from hclust, into several groups either by specifying th
```

More often we will use 'cutree()' with k=2 for example

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

# k=2 : give me a cut to yield 2 trees
```

Make a plot of our 'hclust()" results ie our data colored by color assignment!

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



Principal Component Analysis (PCA)

family of multivariate analysis methods pc1 will capture more variance than pc2

Read data for UK food trends from online

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

Check the data! Always check the data!

```
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?

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

```
dim(x)
```

```
[1] 17  4
```

rownames(x) <- x[,1] x <- x[,-1] head(x) - Don't use this, this just deletes the column, eventually you get an error - What we want to do is change the list of foods to be the title of the rows

```
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
```

```
dim(x)
```

```
[1] 17  4
```

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?

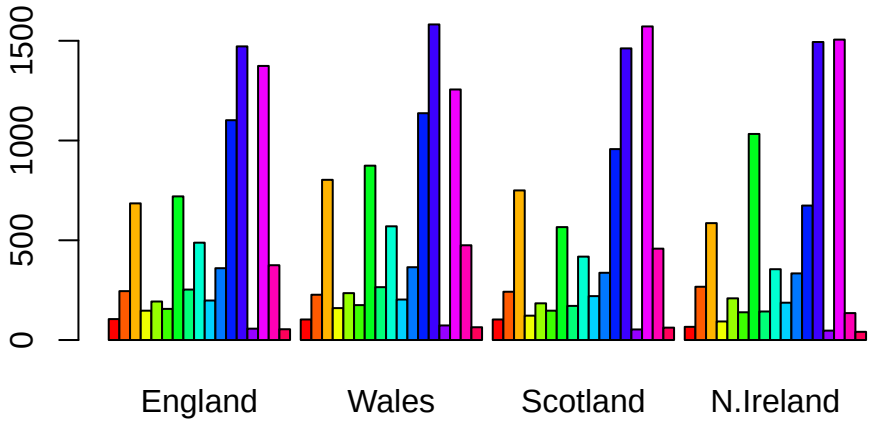
I prefer the second approach, 'row.names=1' because it explicitly states that the first row is just the name of row. The first approach just deletes a row, and if the code is run again, the first row will be deleted again, resulting in a loss of data.

Spotting major differences and trends

Explore the data, basically plot, then plot again.

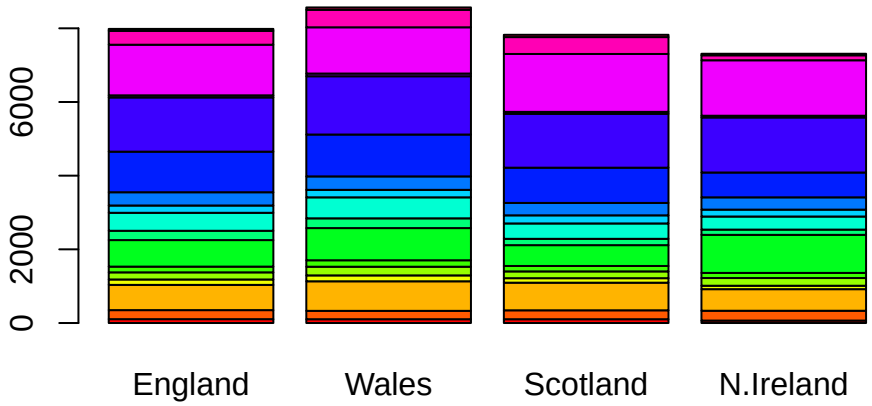
Q3: Changing what optional argument in the above barplot() function results in the following plot?

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



To change the plot above into a stacked bar plot:

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

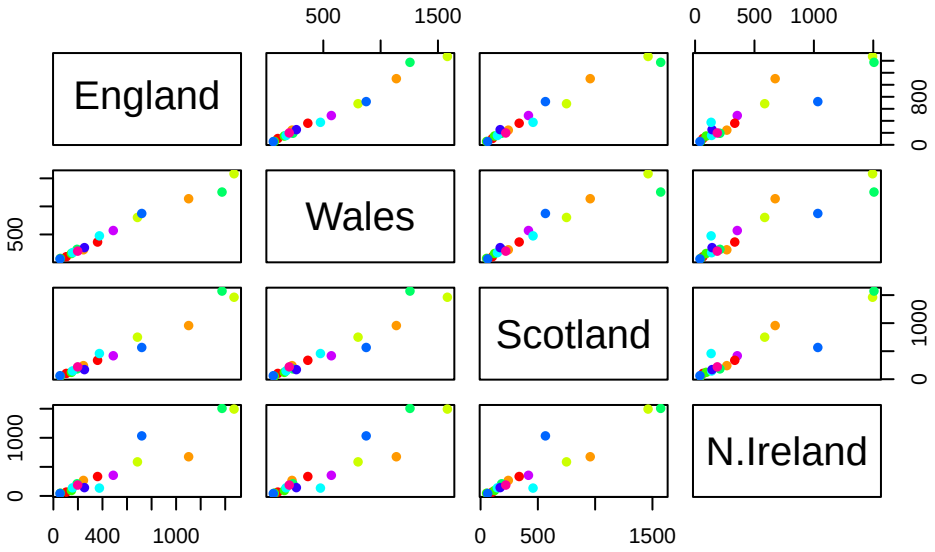


Instead of placing the bars beside each other (beside=T), they are placed on top of one another (beside=F)

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?

A "pairs" plot is somewhat useful, but there is a lot of repetition.

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




```
log2(20/10)
```

```
[1] 1
```

We use log2 because it has to do with doubling. (Doubling in biology is interesting?)

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

The blue dot, which is alcohol drinks I think, is different from the other countries of the UK.

PCA to the rescue

The main function in base R to do PCA is called 'prcomp()'. One issue with the 'prcomp()' function is that it expects the transpose of our data as input.

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

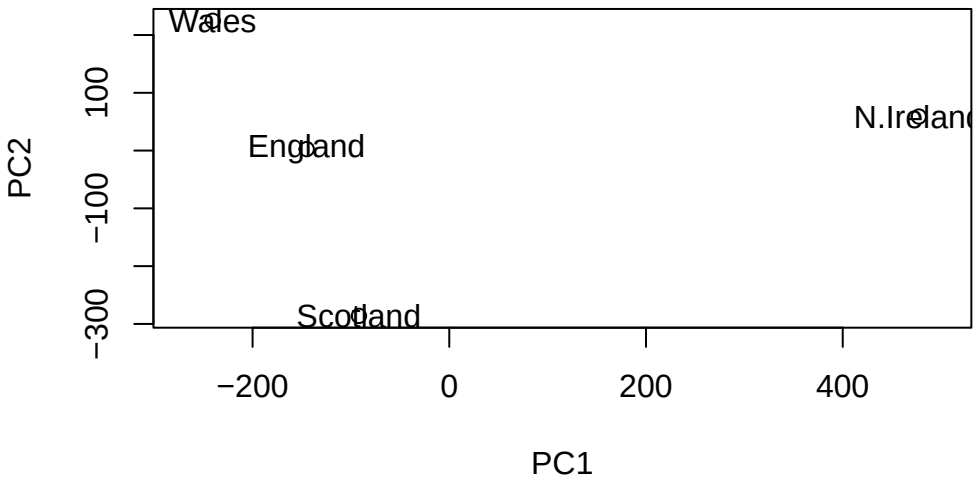
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Cumulative proportion, adds the PCs across the row, tells you how much of the data is captured.

The object returned by 'prcomp()' has our results that include a $Food component. This is our "scores" along the PCs (ie the plot of our data along the new PC axis).

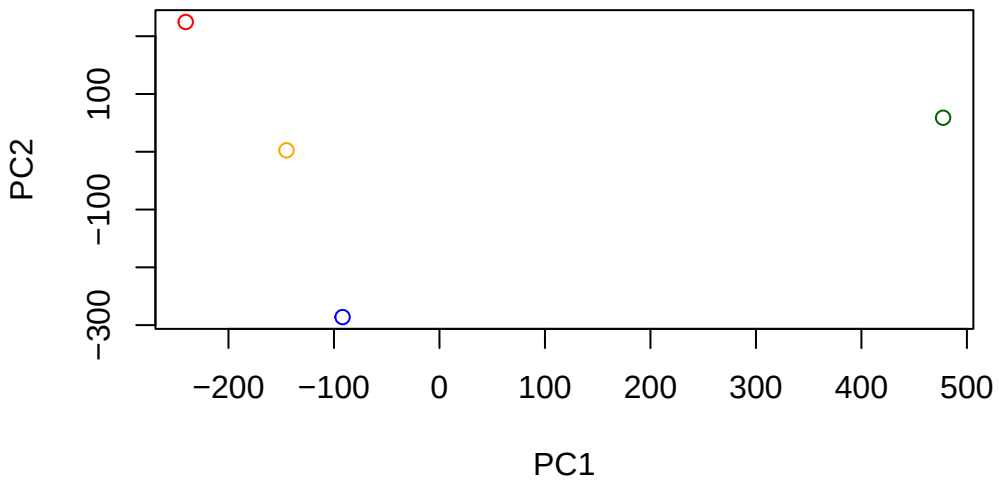
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
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",
     col=c("orange", "red", "blue", "darkgreen",
           pch=16))
```



Variation in the original data

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

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

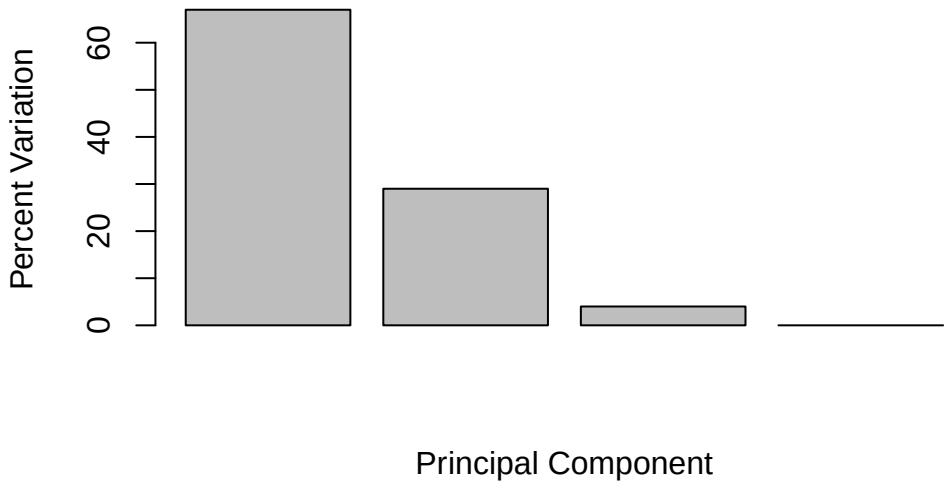
Coverage of the PCA (I think)

```
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
```

Bar graph of variance

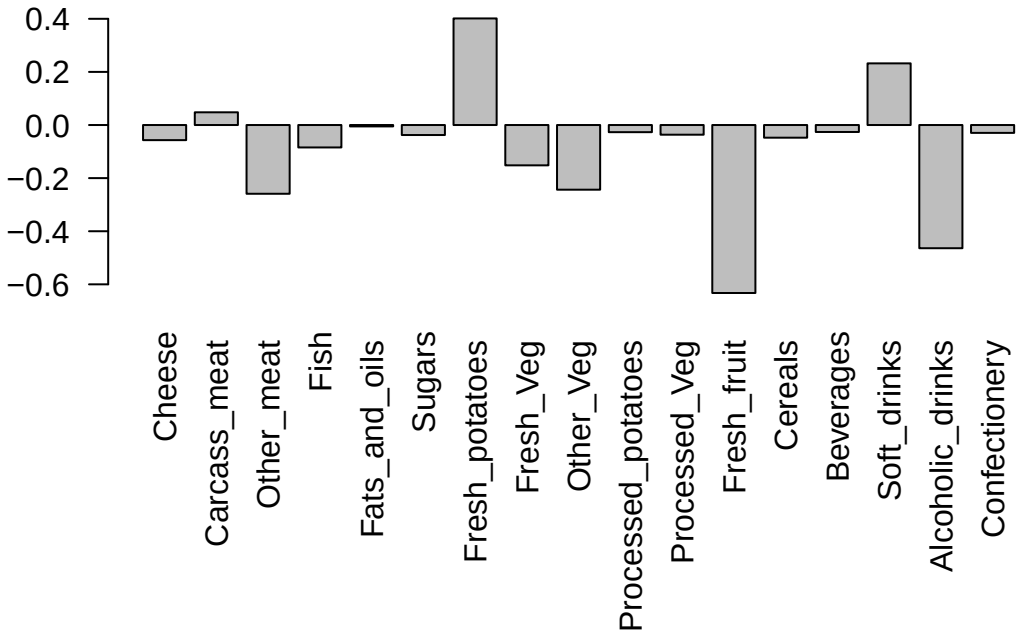
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging deeper (variable loadings)

Consider the influence of the original variables on the PCAs

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

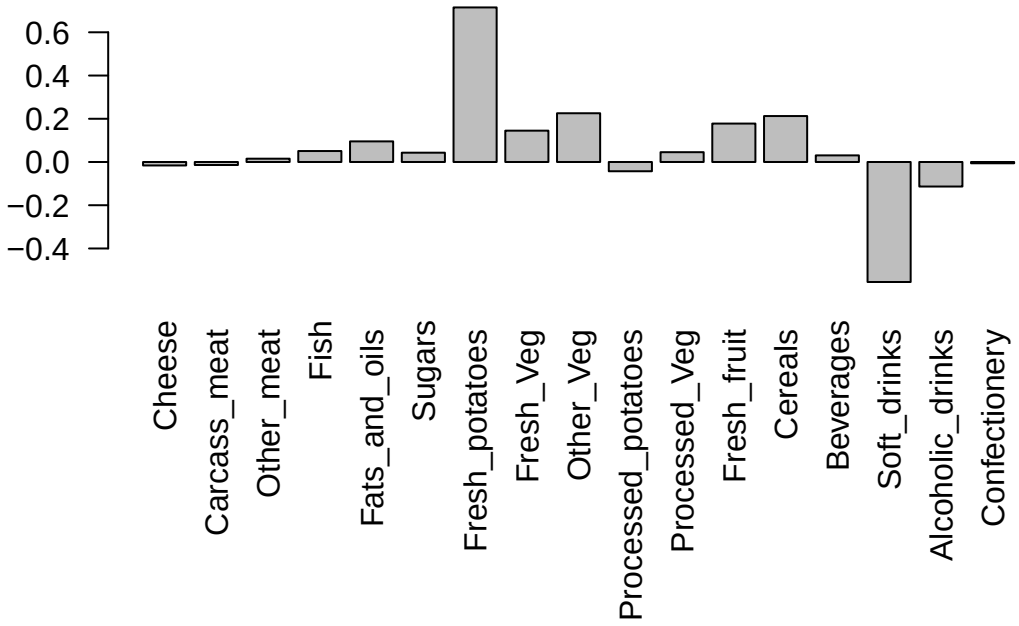


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

Fresh potatoes are very positive and soft drinks are very negative.

PC2 tells us the variance above and below the length of the variance set by PC1.

```
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