BIMM 143 Class 7: Machine Learning

Mariam Benny (A17103451)

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

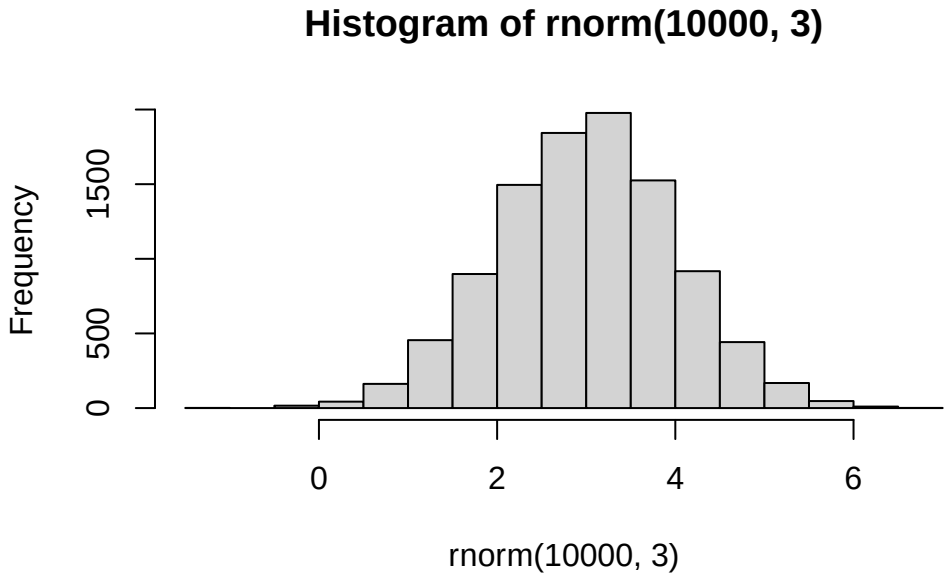
The goal here is to find groups/clusters in your input data.

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

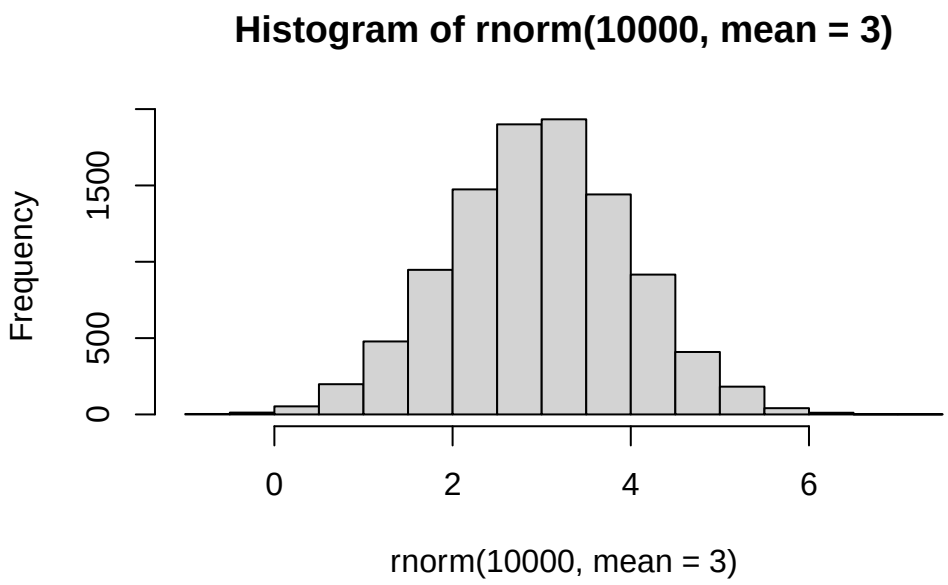
```
rnorm(10)
```

```
 [1]  1.34466379  0.71568862  0.03932867 -1.14968484  0.04798151  1.01636551
 [7] -0.21999185 -1.62972252 -1.12449640 -2.52195314
```

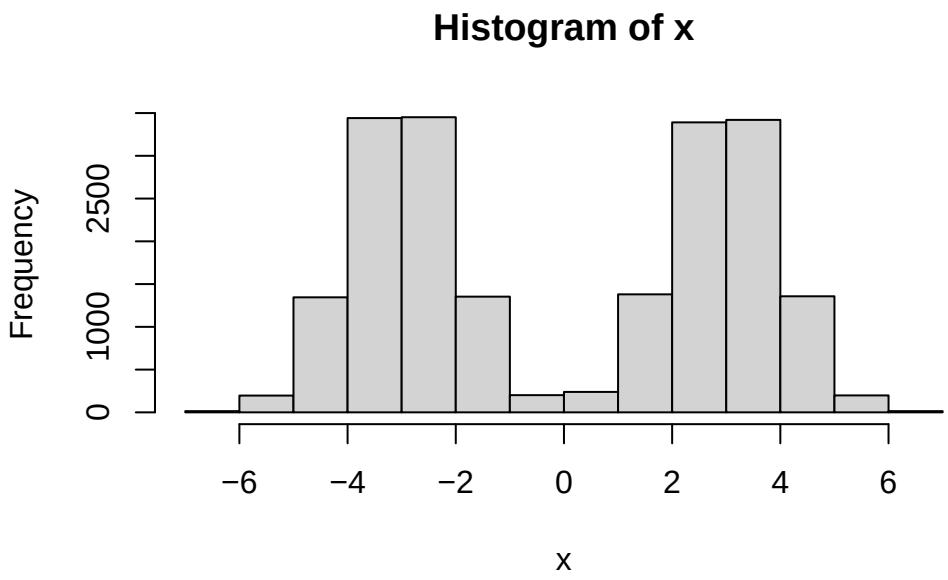
```
hist(rnorm(10000,3))
```



```r
hist( rnorm(10000, mean=3))
```



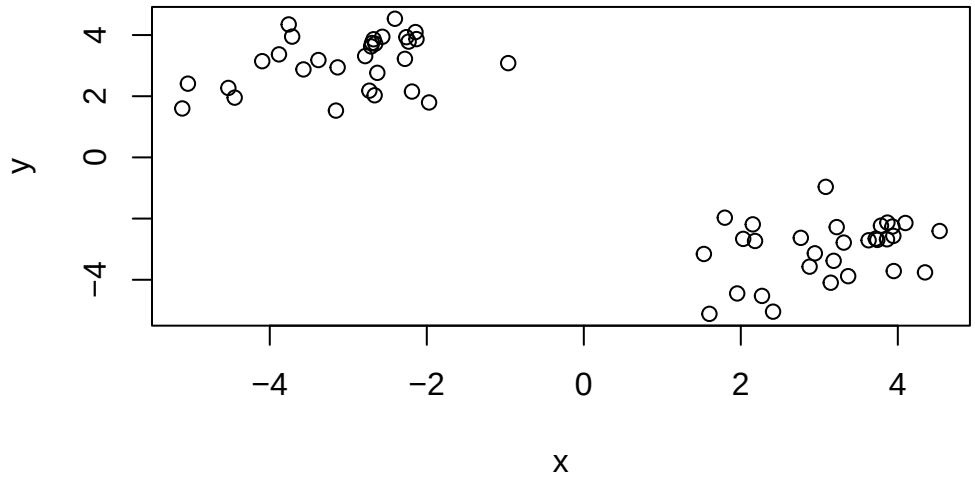

```r
n <- 10000
x <- c(rnorm(n, -3), rnorm(n, +3))
hist(x)
```




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

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

```
              x        y
[1,] -2.1441953 4.096536
[2,] -2.7318202 2.181272
[3,] -2.7851424 3.311453
[4,] -3.7153071 3.948279
[5,] -0.9630529 3.081757
[6,] -1.9708771 1.795964
```

```
plot(z)
```



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

Inspect/print the results

> Q. How many points are in each cluster?

There are thirty points

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

Size is km$size and it is 30 30

Cluster assignment is km$cluster is [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 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 [51] 1 1 1 1 1 1 1 1 1 1

Cluster center is km$cluster which is: x y 1 3.006377 -3.019749 2 -3.019749 3.006377

> Q Plot x colored by the kmeans 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 -3.017498  3.107239
2  3.107239 -3.017498

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 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] 48.50241 48.50241
 (between_SS / total_SS =  92.1 %)

Available components:

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

Results in kmeans object `km`

```
km$size
```

```
[1] 30 30
```

```
km$cluster
```

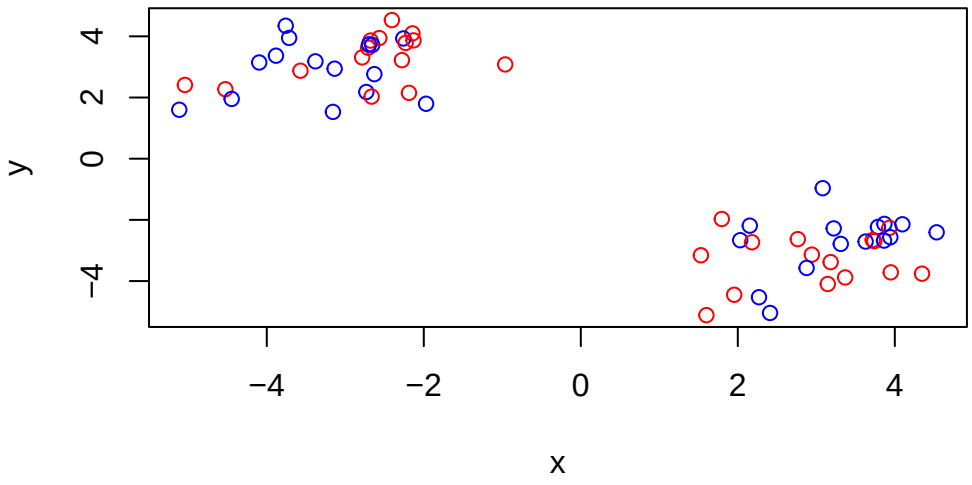
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$center
```

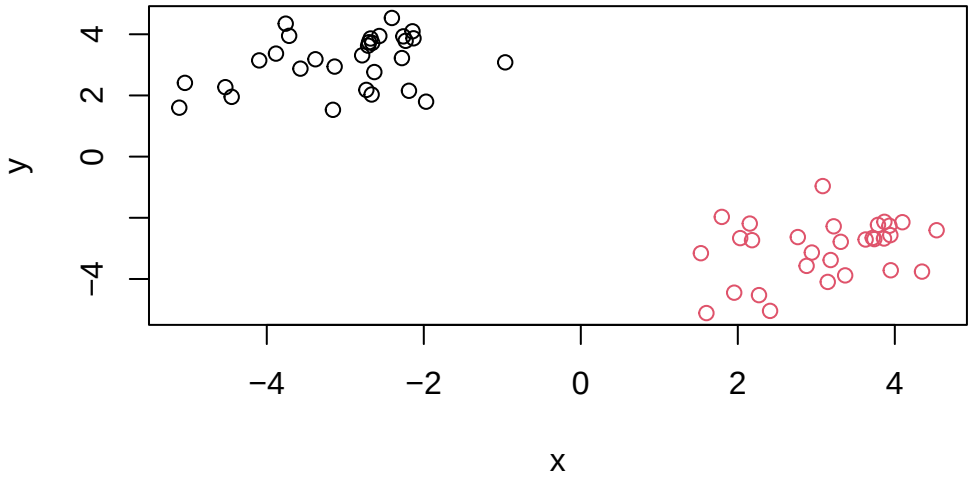
```
          x         y
1 -3.017498  3.107239
2  3.107239 -3.017498
```

R will re-cycle the shorter color vector to be the same length as the longer (number of data points is) in z

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

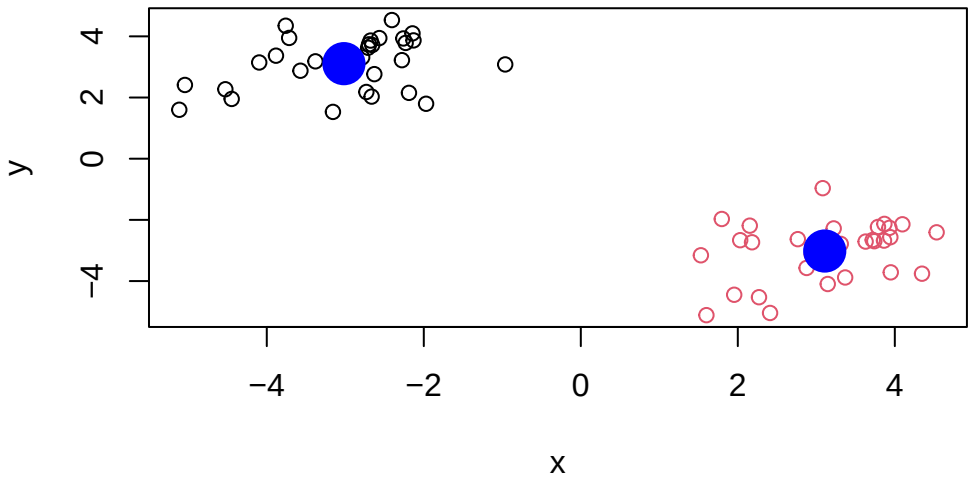



```
plot(z, col=c(km$cluster))
```



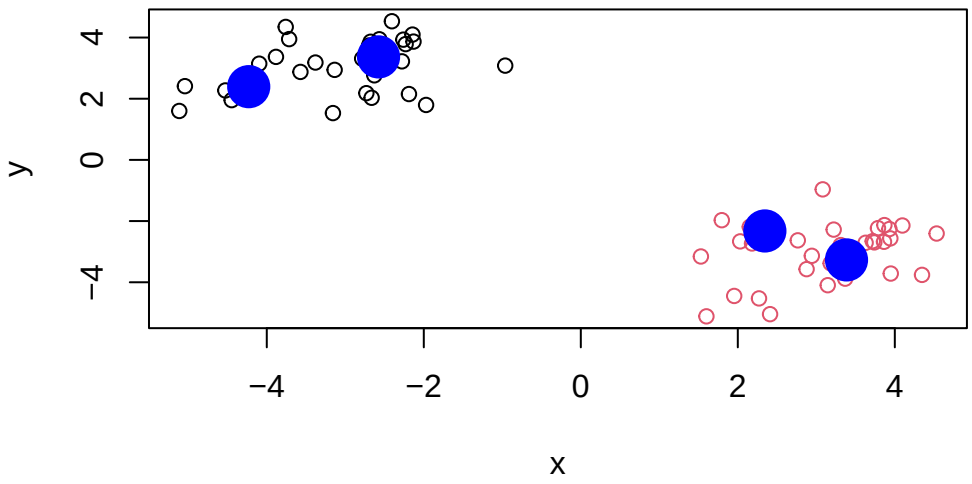
We can use the `points()` function to add new points to an existing plot... like the cluster centers

```
plot(z, col=c(km$cluster))
points(km$centers, col="blue", pch=16, cex=3)
```



Q. Can you run `kmeans()` and ask for 4 clusters and plot the results like we have done above?

```
km4 <- kmeans(z, center=4)
plot(z, col=c(km$cluster))
points(km4$centers, col="blue", pch=16, cex=3)
```



Hierarchichal Clustering

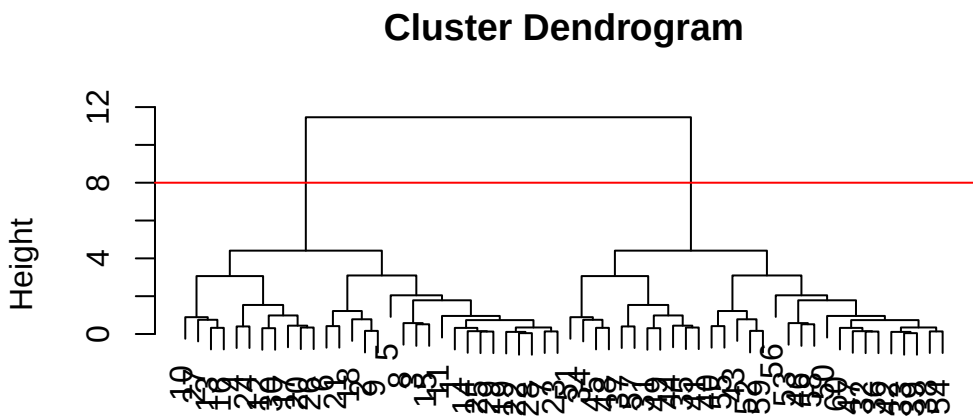
Let's take our same data **z** and see how hclust works.

First we need a distance matrix of our data to be clustered.

```
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")
```



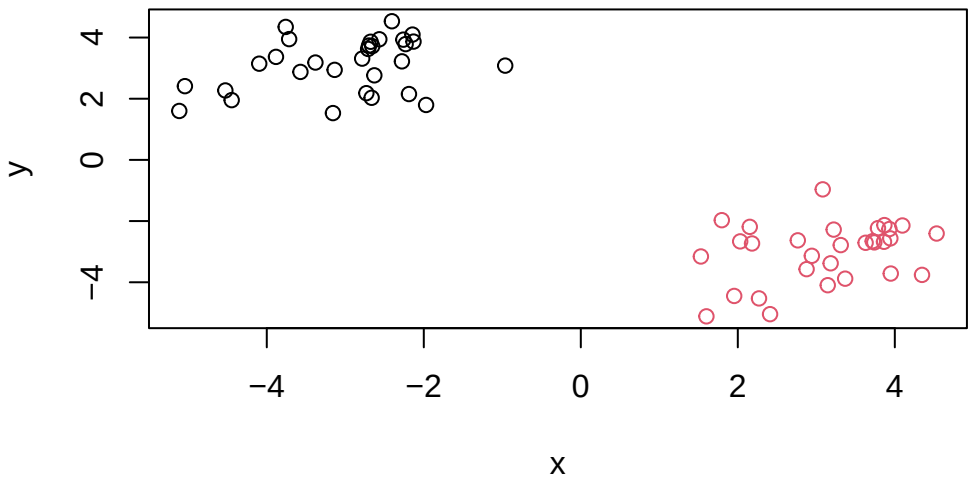
I can get my cluster membership vector by "cutting the tree" with the `cutree()` function like so:

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

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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
```

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

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



PCA data of UK food

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

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

There are 5 rows and 17 columns.

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

```
dim(x)
```

```
[1] 17  5
```

```
# 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
```

```
dim(x)
```

```
[1] 17  4
```

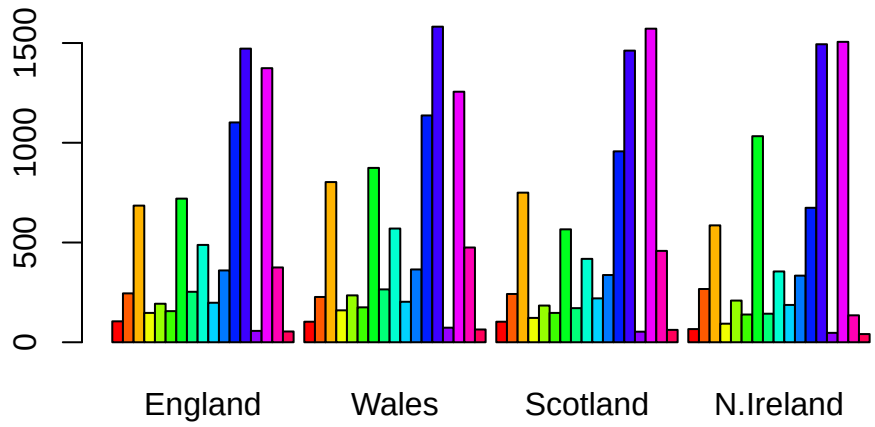
```
x <- read.csv(url, row.names=1)
head(x)
```

```
               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?

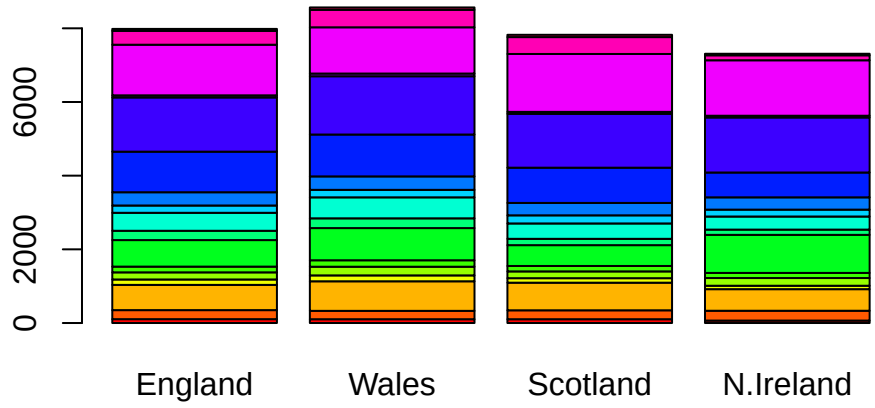
I prefer the second method of code because it can be used in different circumstances which means it is a more versatile/robust code. Also, when I ran the first method multiple times, it messed up some data in R.

```
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?

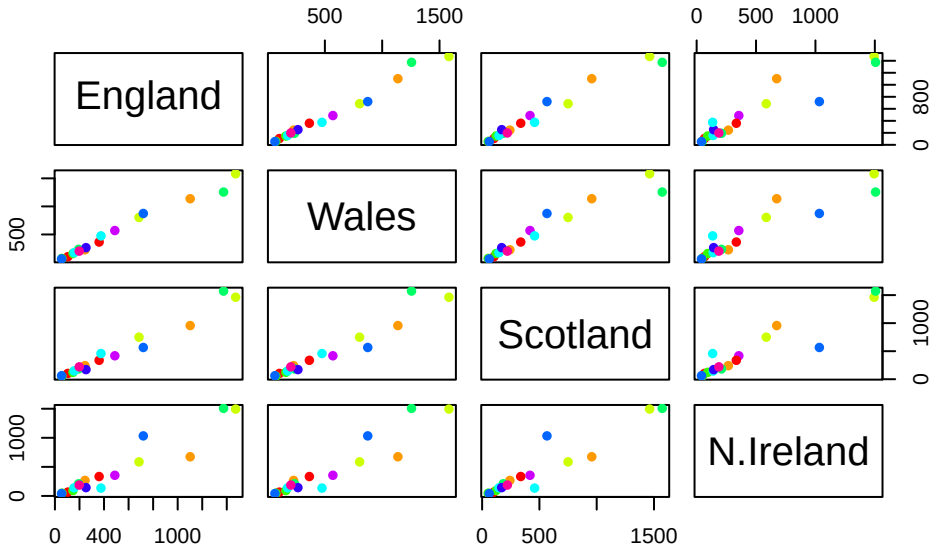
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

The `pairs()` function in R creates a scatterplot matrix to visualize pairwise relationships between variables in a certain dataset. In the scatterplot , the diagonal represents each variable plotted against itself. Points on the diagonal are not meaningful for analysis, as they only show identical variables being plotted together.

```
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?

It's hard to see structure and trends in this small data set using this plot, so I cannot tell the main differences between N. Ireland and the other countries.

PCA to the Rescue

The main function is `prcomp()` in base R to do PCA

```
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's inside this `pca` object that we created from running `prcomp()`

```
attributes(pca)
```

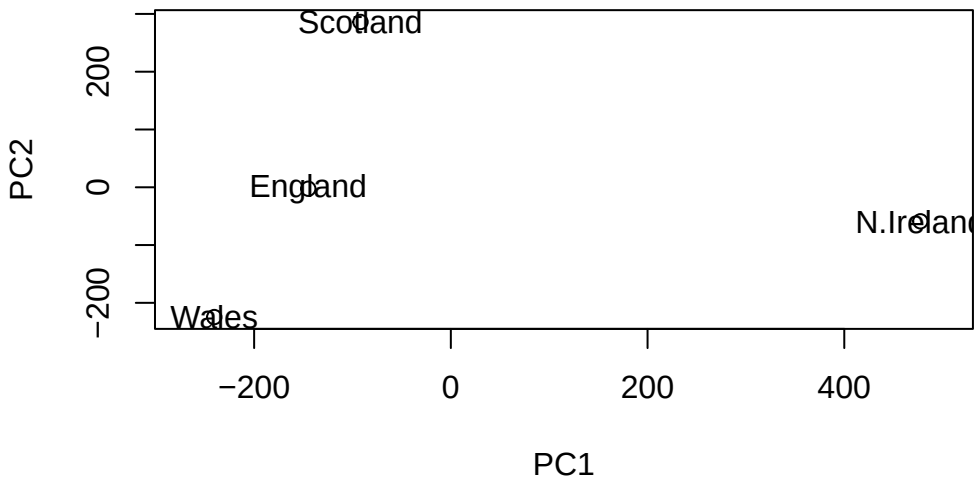
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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
```

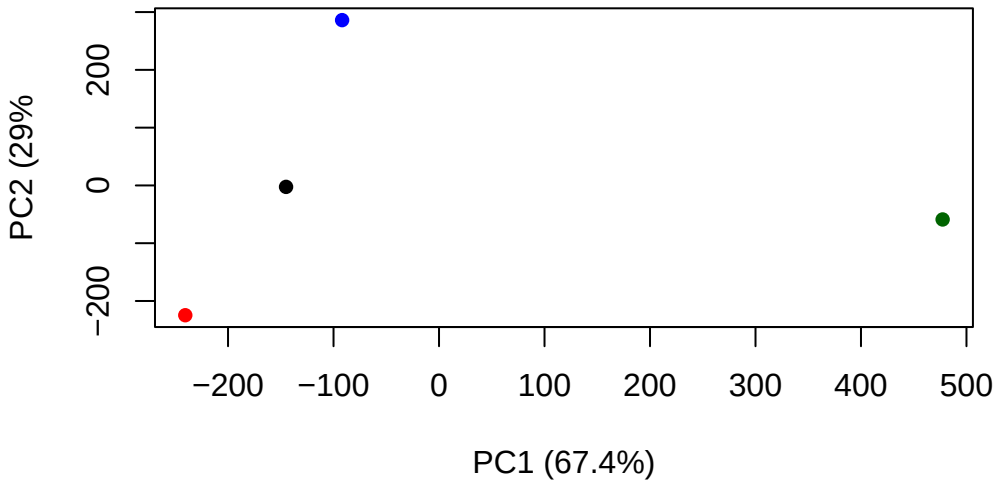
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points

```
# 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))
```

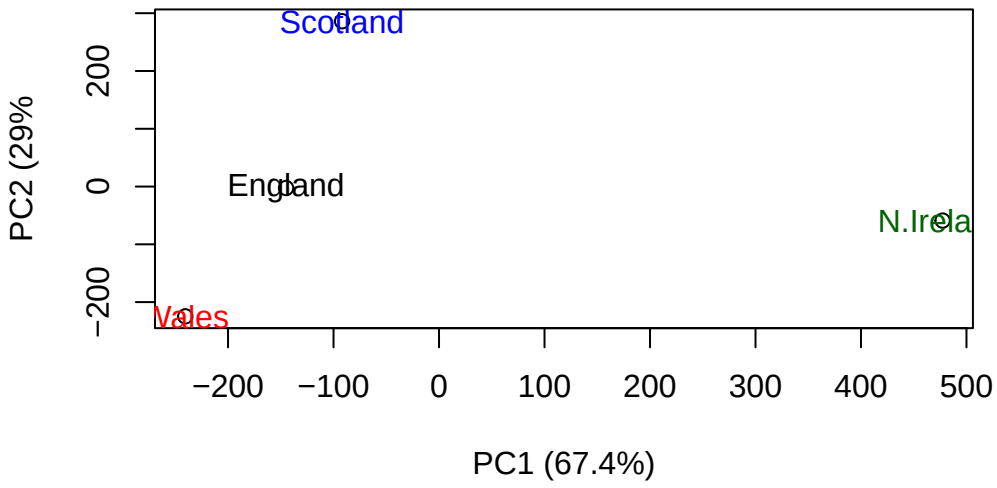


Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), pch=16, xlab="PC1 (67
```




```
plot(pca$x[,1], pca$x[,2], xlab="PC1 (67.4%)", ylab="PC2 (29%")
text(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), colnames(x))
```



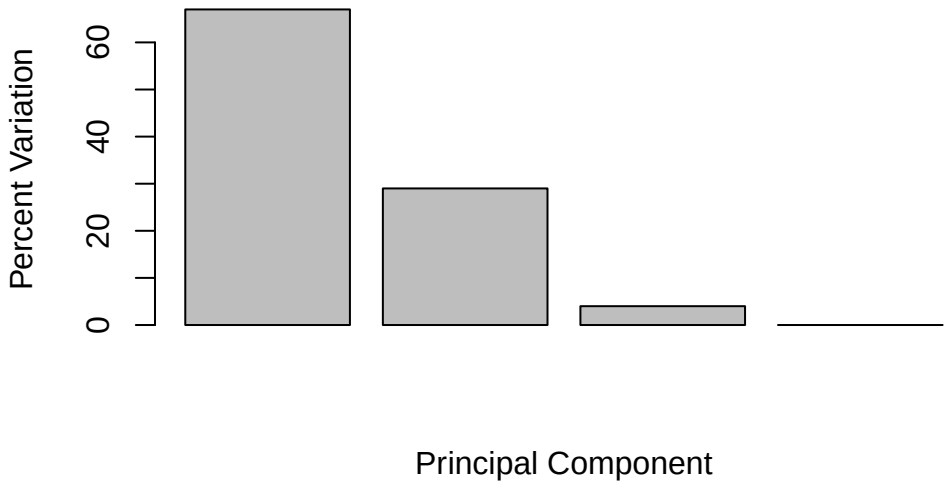

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 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
```

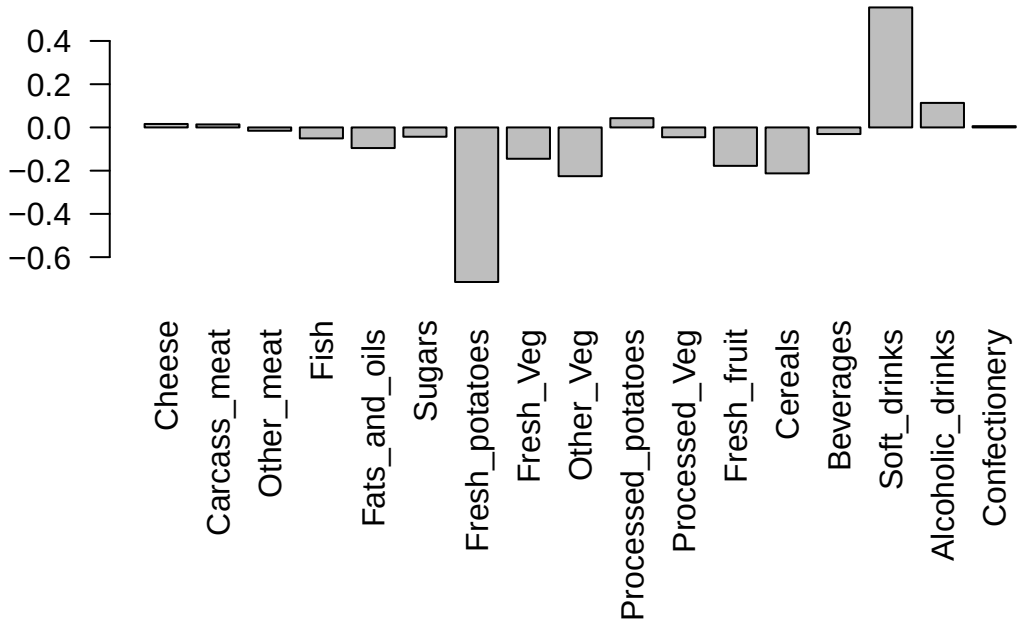
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



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

PC2 tells us that fresh potatoes and soft drinks are the two food groups feature prominently. It tells us the second most degree of variation among the countries.

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




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