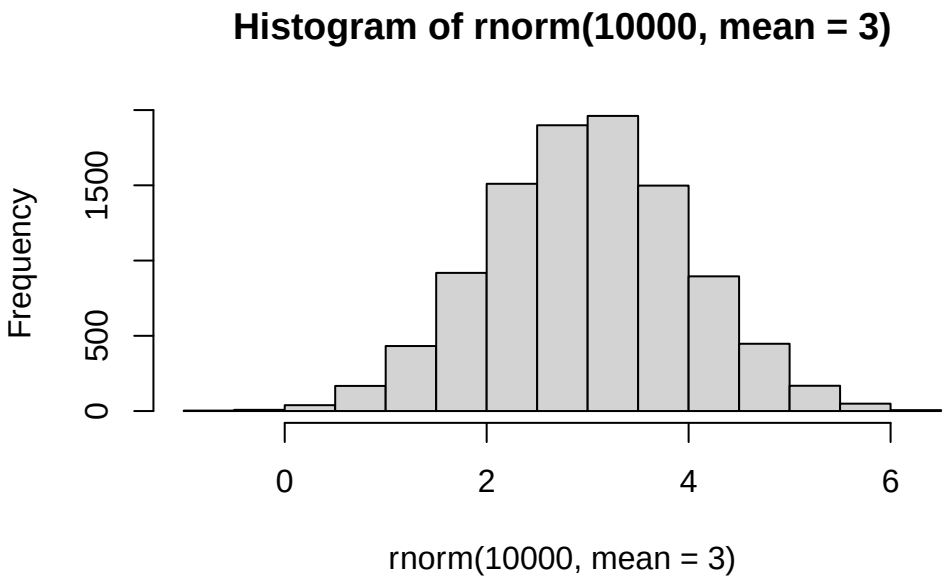
Class 7: Machine Learning 1

Clustering

We will start with k-means clustering, one of the most prevalent clustering methods.

Let's make some data up:

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

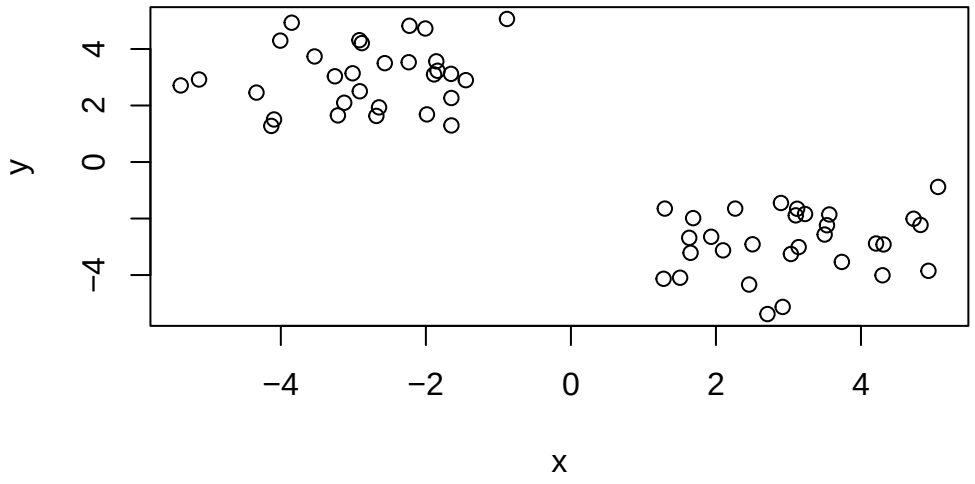



```
tmp <- c( rnorm(30,3), rnorm(30, -3))
x <- cbind(x=tmp, y=rev(tmp))
x
```

```
              x          y
 [1,]  2.0978239 -3.1261159
 [2,]  2.7104422 -5.3805752
 [3,]  1.6317415 -2.6823867
 [4,]  2.2653507 -1.6483708
 [5,]  2.4576641 -4.3359793
 [6,]  4.7256270 -2.0075553
 [7,]  1.2949811 -1.6479462
 [8,]  4.2967605 -4.0080012
 [9,]  2.9203820 -5.1281062
[10,]  3.1007889 -1.8872448
[11,]  4.3111749 -2.9166175
[12,]  3.5632519 -1.8560507
[13,]  3.5312181 -2.2366538
[14,]  1.6500115 -3.2130888
[15,]  3.7361563 -3.5349289
[16,]  5.0630973 -0.8817049
[17,]  3.2289478 -1.8393146
[18,]  4.8209727 -2.2267220
[19,]  1.2779773 -4.1313886
[20,]  4.2078244 -2.8835194
[21,]  3.0341181 -3.2546142
[22,]  2.8965371 -1.4490940
[23,]  3.1405122 -3.0102488
[24,]  1.9342950 -2.6457737
[25,]  1.6853333 -1.9853351
[26,]  3.4990496 -2.5665318
[27,]  4.9304724 -3.8500325
[28,]  1.5058555 -4.0939609
[29,]  2.5045929 -2.9111124
[30,]  3.1200117 -1.6531378
[31,] -1.6531378  3.1200117
[32,] -2.9111124  2.5045929
[33,] -4.0939609  1.5058555
[34,] -3.8500325  4.9304724
[35,] -2.5665318  3.4990496
[36,] -1.9853351  1.6853333
[37,] -2.6457737  1.9342950
[38,] -3.0102488  3.1405122
[39,] -1.4490940  2.8965371
[40,] -3.2546142  3.0341181
[41,] -2.8835194  4.2078244
[42,] -4.1313886  1.2779773
```

```
[43,] -2.2267220  4.8209727
[44,] -1.8393146  3.2289478
[45,] -0.8817049  5.0630973
[46,] -3.5349289  3.7361563
[47,] -3.2130888  1.6500115
[48,] -2.2366538  3.5312181
[49,] -1.8560507  3.5632519
[50,] -2.9166175  4.3111749
[51,] -1.8872448  3.1007889
[52,] -5.1281062  2.9203820
[53,] -4.0080012  4.2967605
[54,] -1.6479462  1.2949811
[55,] -2.0075553  4.7256270
[56,] -4.3359793  2.4576641
[57,] -1.6483708  2.2653507
[58,] -2.6823867  1.6317415
[59,] -5.3805752  2.7104422
[60,] -3.1261159  2.0978239
```

```
plot(x)
```



The main function in R for K-means clustering is called `kmeans()`

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

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

Cluster means:
          x         y
1 -2.833070  3.038099
2  3.038099 -2.833070

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 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] 72.90485 72.90485
 (between_SS / total_SS =  87.6 %)

Available components:

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

Q1. How many points are in each cluster?

```r
k$size
```

```
[1] 30 30
```

Q2. What is the clustering result, i.e. membership vector?

```r
k$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 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
```

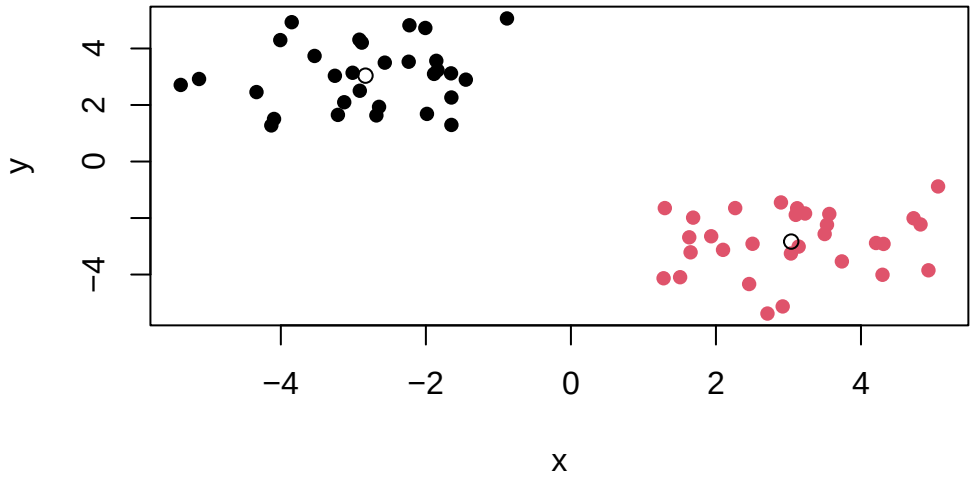
Q3. What is the cluster center?

```r
k$centers
```

```
          x         y
1 -2.833070  3.038099
2  3.038099 -2.833070
```

Q4. Make a plot of our data colored by clustering results with optionally the cluster centers shown.

```
plot(x, col=k$cluster, pch=16)
points(k$centers)
```



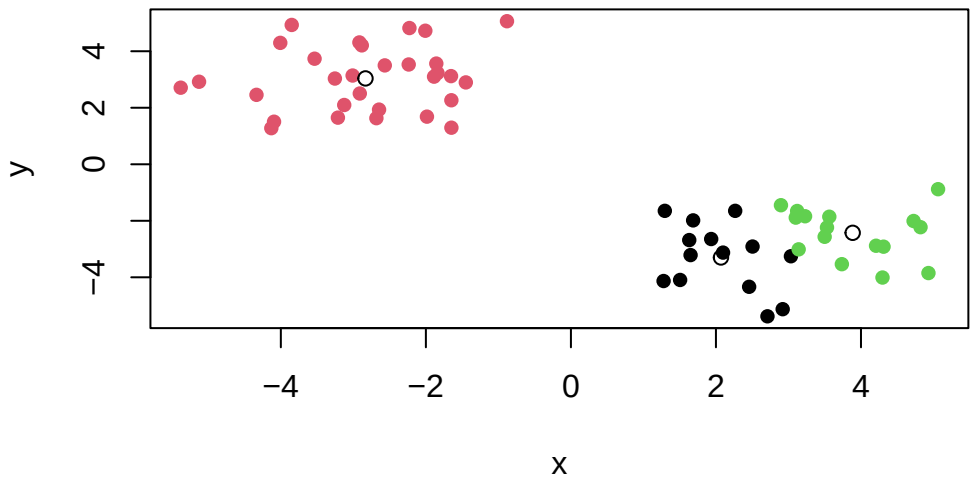
Q5. Run kmeans again but cluster into 3 groups, then plot those results like we did above.

```
k3 <- kmeans(x, centers=3, nstart=20)
k3
```

```
K-means clustering with 3 clusters of sizes 14, 30, 16

Cluster means:
          x         y
1  2.069326 -3.298911
2 -2.833070  3.038099
```

```
3  3.885775 -2.425460

Clustering vector:
 [1] 1 1 1 1 1 3 1 3 1 3 3 3 3 1 3 3 3 3 1 3 1 3 3 1 1 3 3 1 1 3 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] 22.77361 72.90485 19.79863
 (between_SS / total_SS =  90.2 %)

Available components:

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

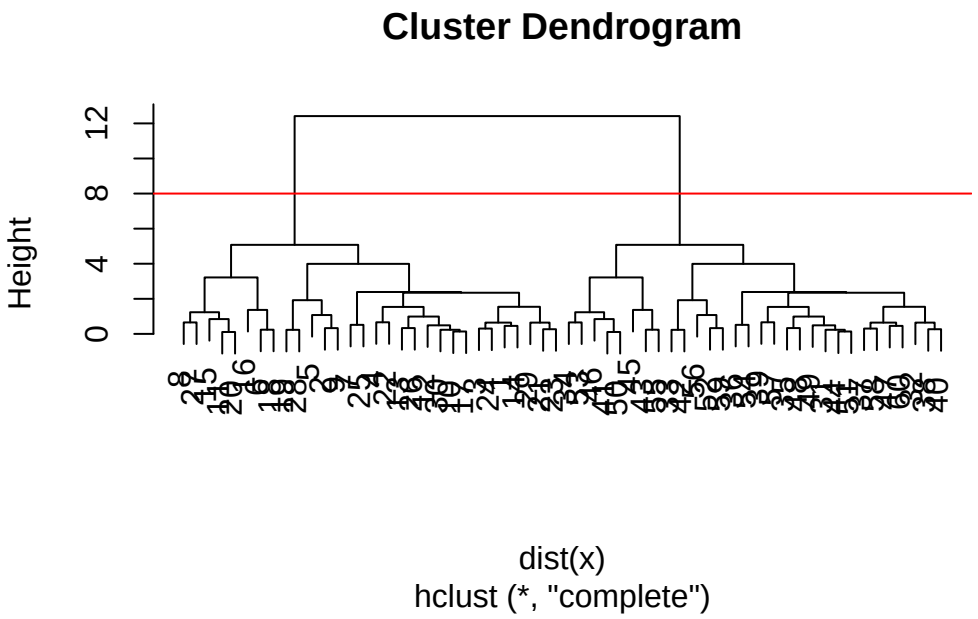
```r
plot(x, col=k3$cluster, pch=16)
points(k3$centers)
```




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

```
Call:
hclust(d = dist(x))

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

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



The cluster dendrogram shows hierarchical clustering: we start at the bottom, with each point as their own cluster. The number of clusters then drops in half as points next to each other, and so on until there is only 1 cluster at the top of the dendrogram.

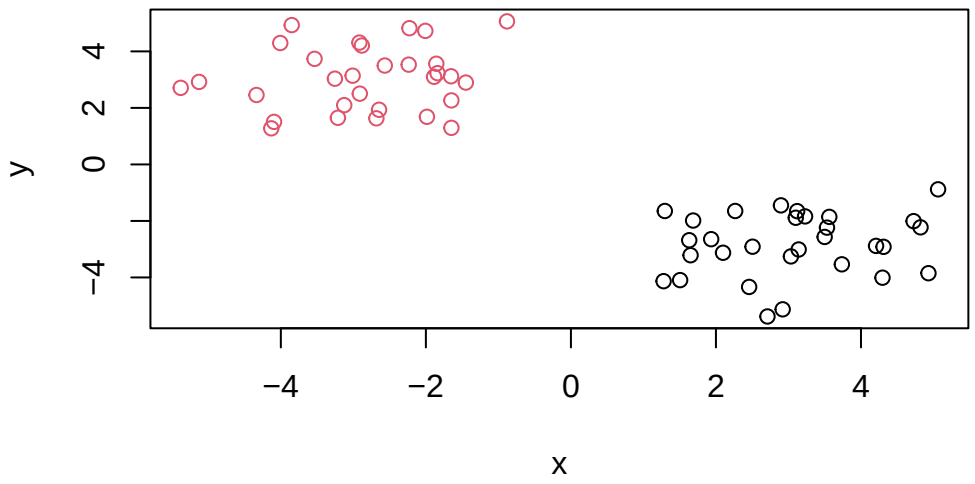
The function to get our clusters/groups from a hclust object is called `cutree()`

```
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 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. Plot our hclust results in terms of our data colored by cluster membership.

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



Principal Component Analysis (PCA)

The first principal component follows a "best fit" through the data points. The data also has maximum variance for PC1.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
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
7      Fresh_potatoes      720   874      566      1033
8           Fresh_Veg      253   265      171       143
9           Other_Veg      488   570      418       355
10 Processed_potatoes      198   203      220       187
11      Processed_Veg      360   365      337       334
12        Fresh_fruit     1102  1137      957       674
13            Cereals     1472  1582     1462      1494
14          Beverages       57    73       53        47
15        Soft_drinks     1374  1256     1572      1506
16   Alcoholic_drinks      375   475      458       135
17      Confectionery       54    64       62        41
```

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

```
dim(x)
```

```
[1] 17  5
```

Preview the first 6 rows

```
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
x <- read.csv(url)
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
```

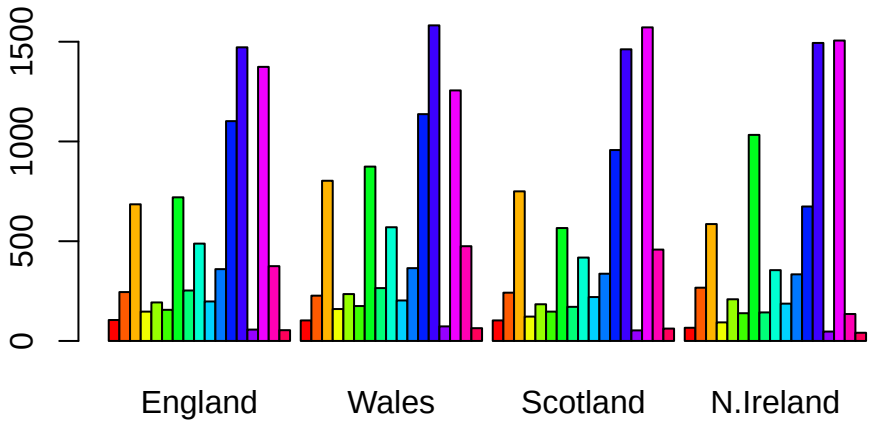
```
dim(x)
```

```
[1] 17  4
```

> 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 approach to setting rownames=1 is preferable. In the original method, running the code multiple times without adding an additional "read.csv(url)" step will result in offsetting the column names by 1 each time and losing column titles.

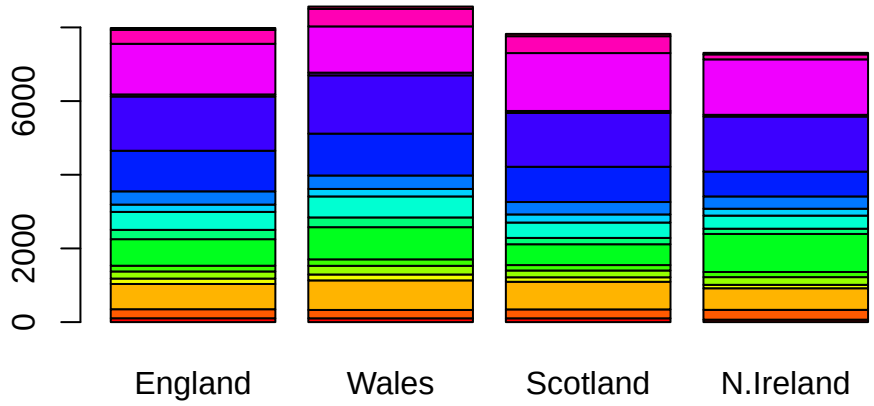
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

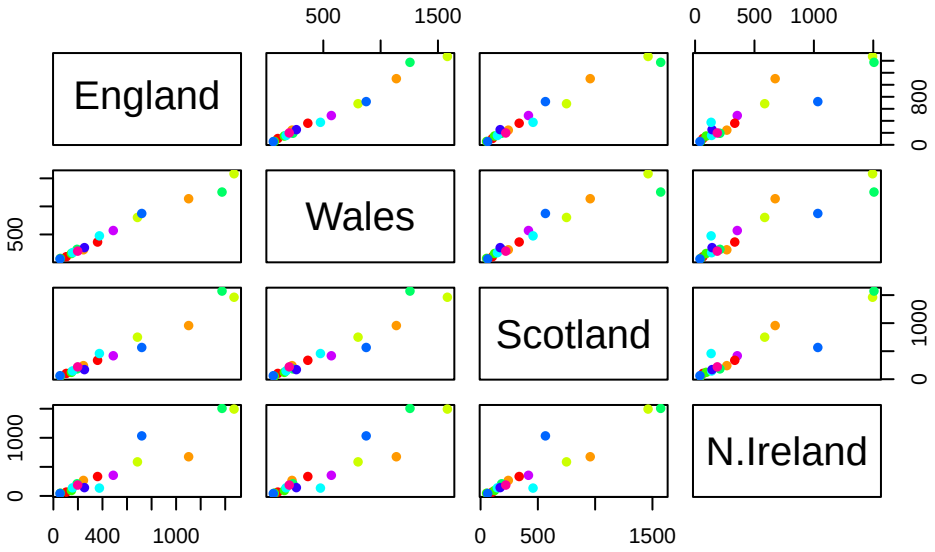
Changing the "beside" argument to FALSE or leaving it out will change the barplot. Changing beside to FALSE or removing it will portray the columns as stacked bars.

```
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?

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



The x-axis of each plot is equal to whatever region is listed in that column (England in column 1, Wales in column 2, etc).

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

The correlation for the plotted points are not fully linear for N. Ireland like they are for the other countries. Dots that are not in the diagonal line represent that it is more variant than the trends in the other countries. N. Ireland had lower levels of fresh fruit consumption than the other countries, and higher levels of potato consumption.

```
# Use the prcomp() PCA function
pca <- prcomp( t(x) )
summary(pca)
```

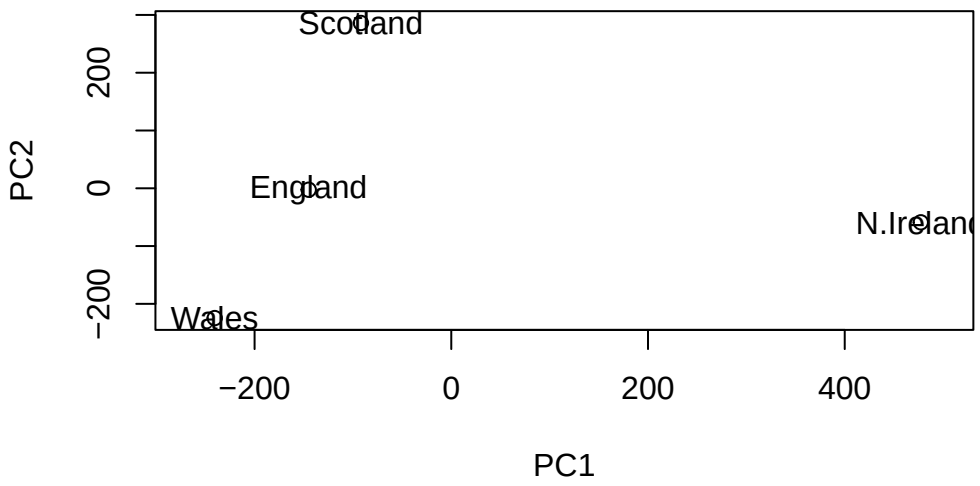
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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

```
pca$x
```

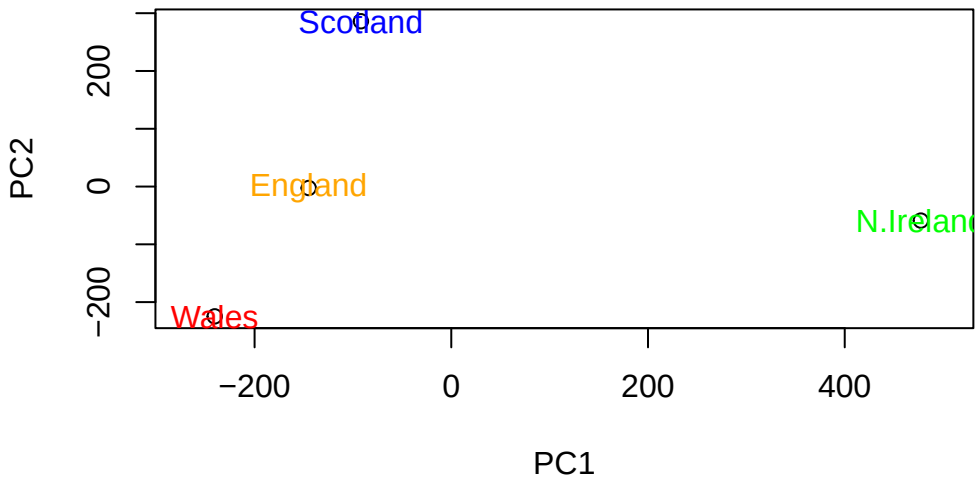
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

```
# 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"))
```



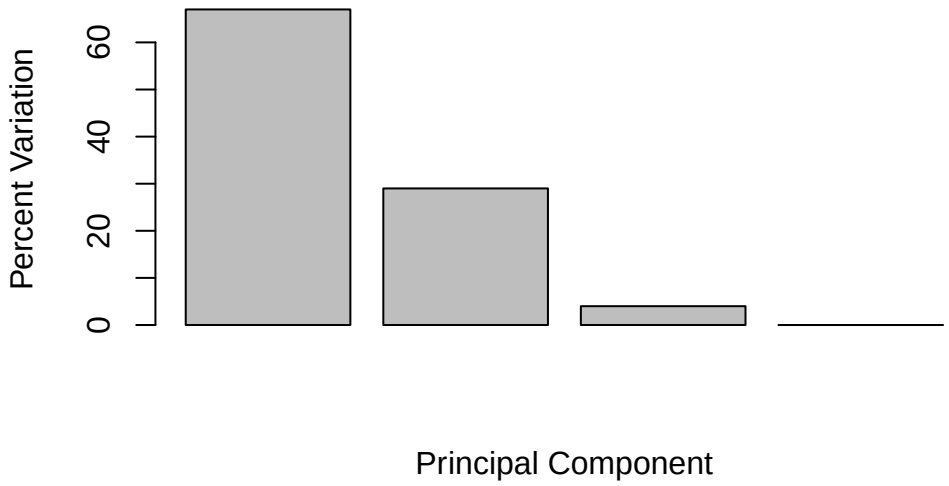

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

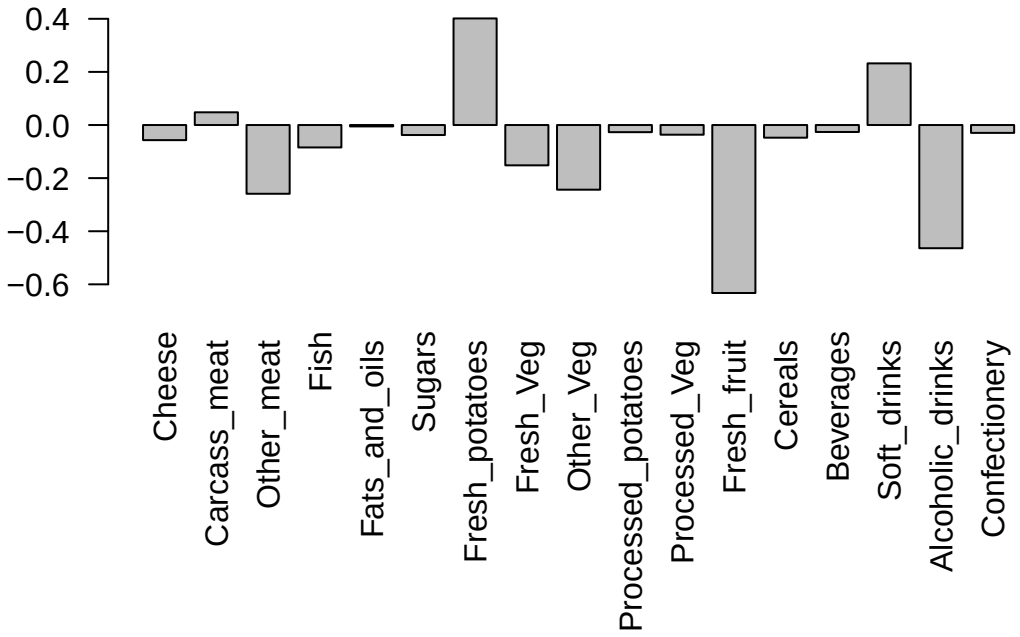
```
## or the second row here...
z <- summary(pca)
z$importance
```

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

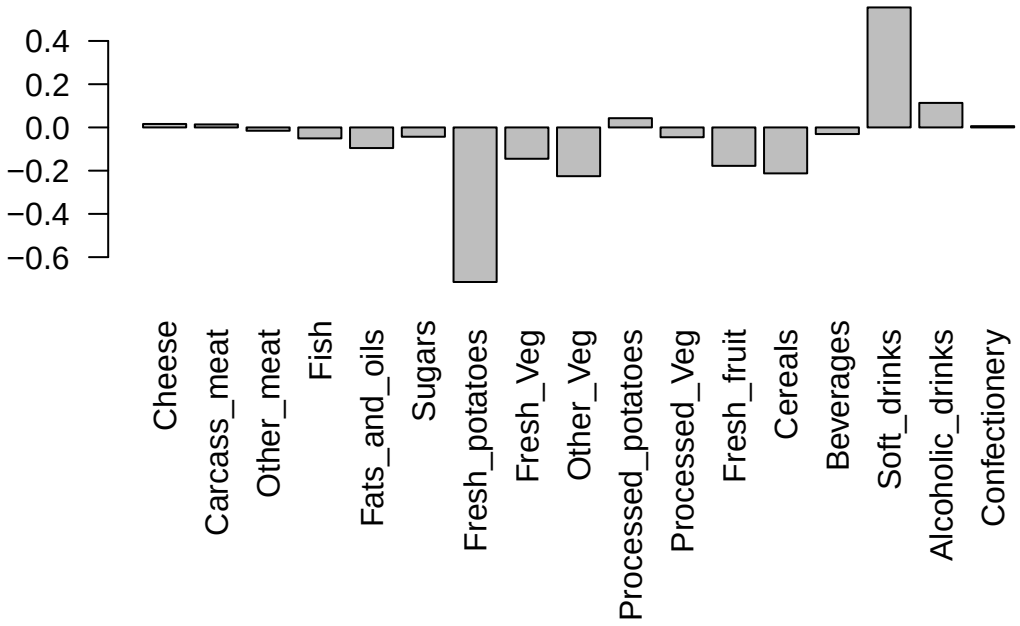



```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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 )
```



The two categories that feature most prominently are fresh potatoes and soft drinks. Wales has less consumption of soft drinks, and Scotland has less consumption of fresh potatoes. These two factors cause the most variance along the PC2 axis, which mainly shows the differences between Scotland, England, and Wales.