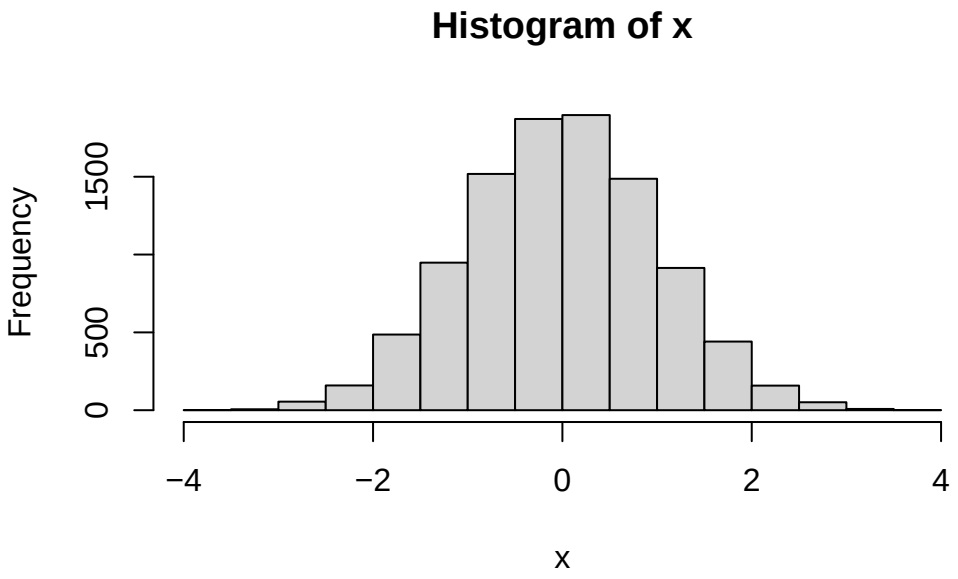
Class 7 Machine Learning 1

Ben Gersh

K-means clustering

First we will test how this method works in R with some made up data.

```
x<-rnorm(10000)
hist(x)
```



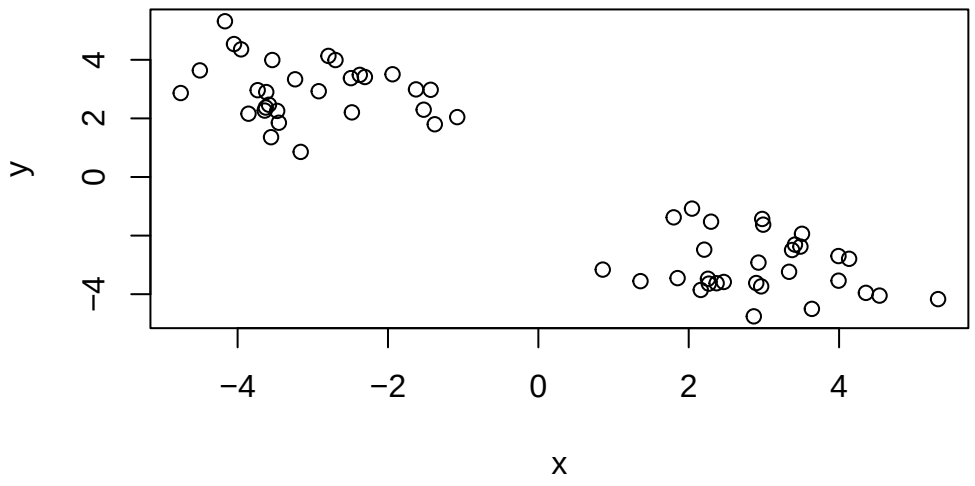
Let's make some numbers centered on -3

```r
tmp<-c(rnorm(30, -3), rnorm(30, +3))

x<-cbind(x=tmp, y=rev(tmp))
x
```

```
             x         y
 [1,] -3.855309  2.160279
 [2,] -3.451430  1.852003
 [3,] -4.047837  4.538008
 [4,] -1.432261  2.977360
 [5,] -3.639590  2.266956
 [6,] -1.078984  2.044053
 [7,] -2.479853  2.206502
 [8,] -1.525002  2.297041
 [9,] -4.756662  2.865906
[10,] -2.792145  4.133239
[11,] -2.492714  3.374704
[12,] -4.168556  5.316677
[13,] -3.235437  3.335023
[14,] -4.501821  3.638365
[15,] -3.473638  2.256040
[16,] -3.555219  1.358493
[17,] -3.538475  3.993305
[18,] -1.939469  3.506522
[19,] -3.583415  2.466181
[20,] -3.954007  4.356760
[21,] -2.698190  3.991774
[22,] -1.627266  2.990310
[23,] -3.619062  2.898345
[24,] -3.732871  2.963560
[25,] -3.625639  2.371220
[26,] -1.377493  1.799993
[27,] -2.374911  3.485846
[28,] -2.307267  3.413200
[29,] -2.921127  2.927661
[30,] -3.160828  0.856712
[31,]  0.856712 -3.160828
[32,]  2.927661 -2.921127
[33,]  3.413200 -2.307267
[34,]  3.485846 -2.374911
[35,]  1.799993 -1.377493
[36,]  2.371220 -3.625639
```

```
[37,]  2.963560 -3.732871
[38,]  2.898345 -3.619062
[39,]  2.990310 -1.627266
[40,]  3.991774 -2.698190
[41,]  4.356760 -3.954007
[42,]  2.466181 -3.583415
[43,]  3.506522 -1.939469
[44,]  3.993305 -3.538475
[45,]  1.358493 -3.555219
[46,]  2.256040 -3.473638
[47,]  3.638365 -4.501821
[48,]  3.335023 -3.235437
[49,]  5.316677 -4.168556
[50,]  3.374704 -2.492714
[51,]  4.133239 -2.792145
[52,]  2.865906 -4.756662
[53,]  2.297041 -1.525002
[54,]  2.206502 -2.479853
[55,]  2.044053 -1.078984
[56,]  2.266956 -3.639590
[57,]  2.977360 -1.432261
[58,]  4.538008 -4.047837
[59,]  1.852003 -3.451430
[60,]  2.160279 -3.855309
```

```
plot(x)
```



Now let's see how 'k(means)' works with this data...

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

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

Cluster means:
          x         y
1  2.954735 -3.031549
2 -3.031549  2.954735

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] 56.77522 56.77522
 (between_SS / total_SS =  90.4 %)

Available components:
```

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

```
km$centers
```

```
          x         y
1  2.954735 -3.031549
2 -3.031549  2.954735
```

Q. How many points are in each cluster

```
km$size
```

```
[1] 30 30
```

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

```
km$cluster
```

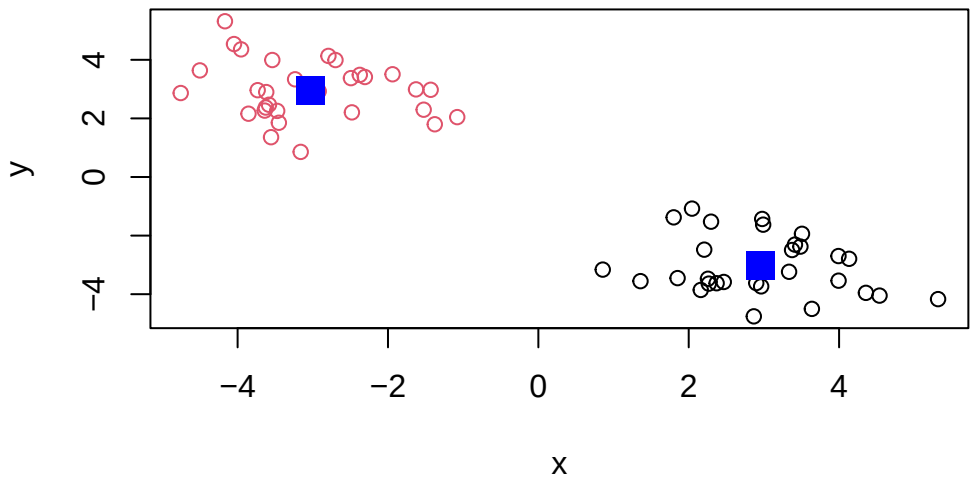
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
km$centers
```

```
          x         y
1  2.954735 -3.031549
2 -3.031549  2.954735
```

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)
```



Hierarchical Clustering

The 'hclust()' function in R performs hierarchical clustering.

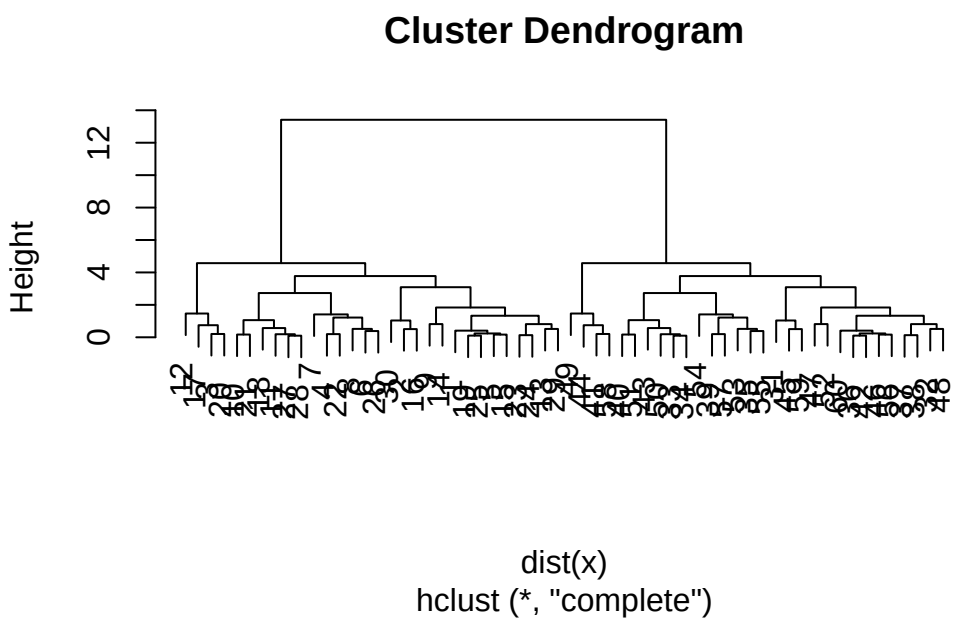
The 'hclus()' function requires an input distance matrix, which I can get from the 'dist()' function.

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

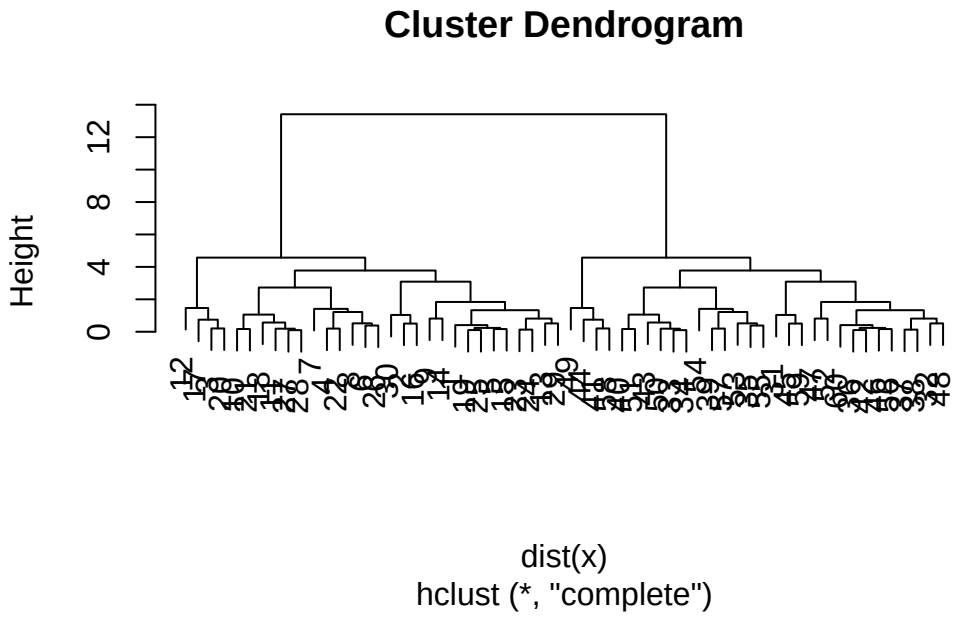
```
Call:
hclust(d = dist(x))

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

There is a plot () method for hclust objects...

```
plot(hc)
```




```
plot(hc)
```



Now to get my cluster membership vector I need to "cut" the tree to yield separate "branches" with the "leaves" on each branch being R clusters. To do this we use the 'cutree()' function.

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

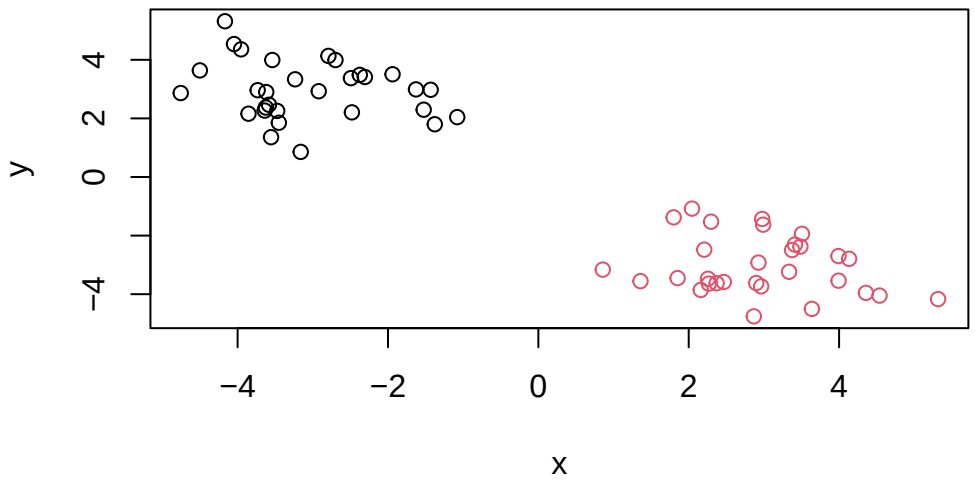
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Use 'cutree()' with a k=2.

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

A plot of our data colored by our hclust grps

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



#Principal Component Analysis (PCA)

```
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
```

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

```
dim(x)
```

```
[1] 17  5
```

> 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 17 rows and 5 columns

```
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
```

```
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
```

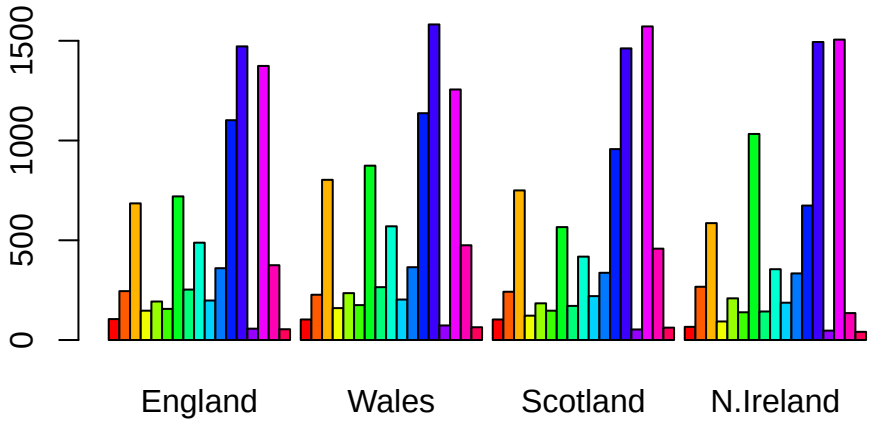
```
x <- read.csv(url, row.names=1)
head(x)
```

```
               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
```

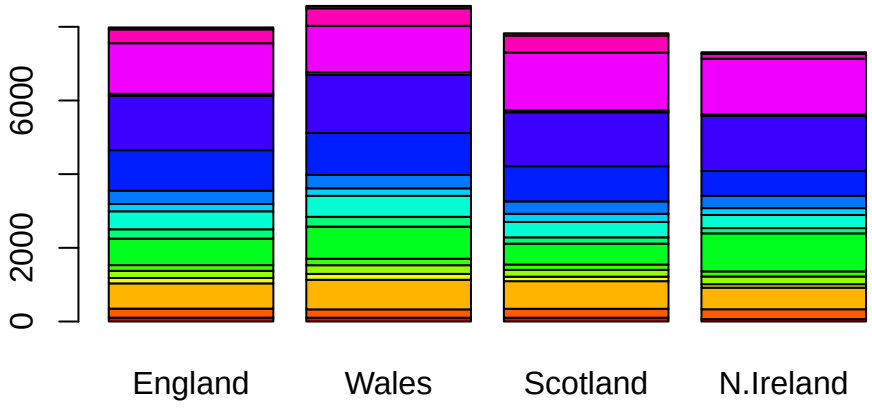
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I prefer the second approach to solving the row-names problem because you do not have to manually remove a column every time.

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




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

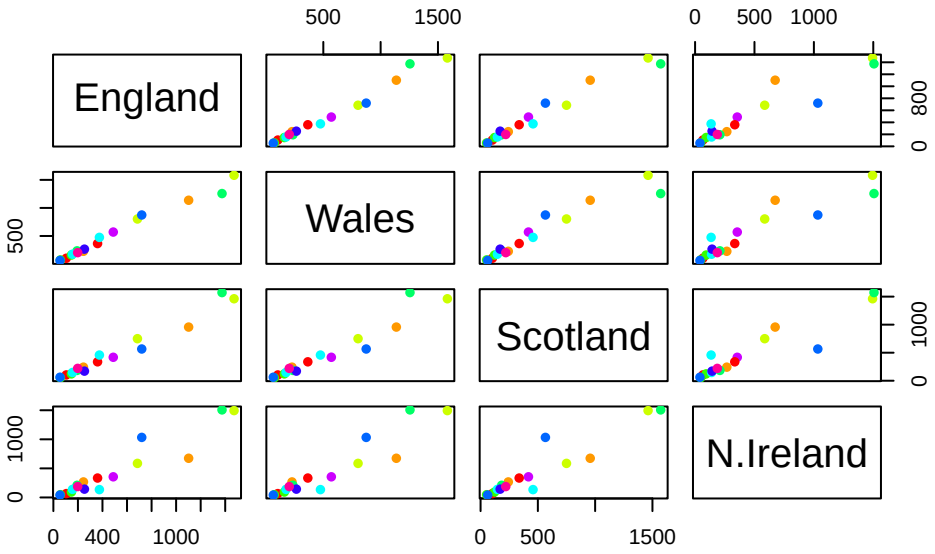


Q3. Changing what optional argument in the above barplot() function results in the following plot?

Changing beside=TRUE to beside=FALSE results in the above barplot

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)
```



This means that whichever country is on the bottom represents the x-axis and whichever country is on the left represents the y-axis. If a point lies on the diagonal, it means that the two countries do not have much variation in that aspect.

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

The main differences between N. Ireland and the other countries is that N. Ireland tends to eat far less fish and far less cheese than the other countries of the UK in terms of this data-set.

#PCA to the rescue

Principal Component Analysis (PCA for short) can be a big help in these cases where we have lots of things that are being measured in a dataset.

The main PCA function in base R is called 'prcomp()'

```
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
```

The above result shows that PCA captures 67% of the total variance in the original data in one PC and 96.5% in two PCs.

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

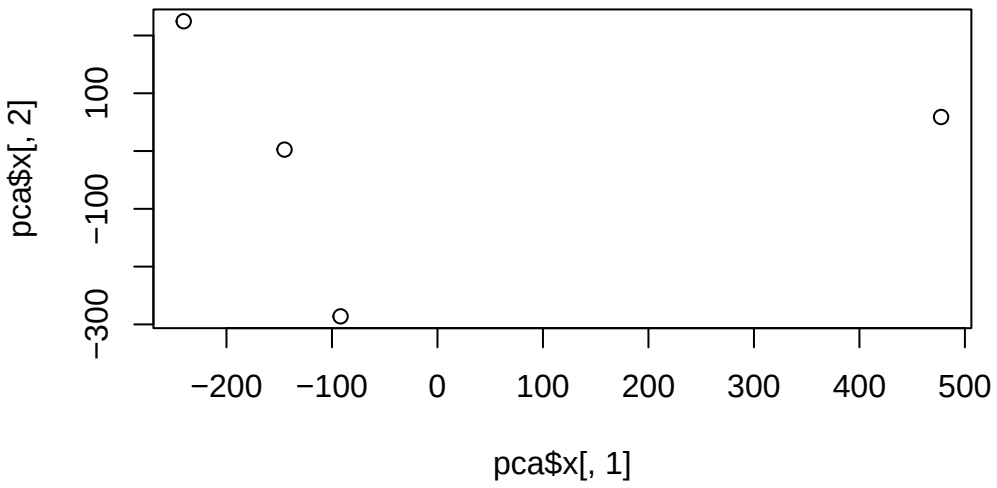
```
head(pca$x)
```

```
                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
```

Let's plot our main results

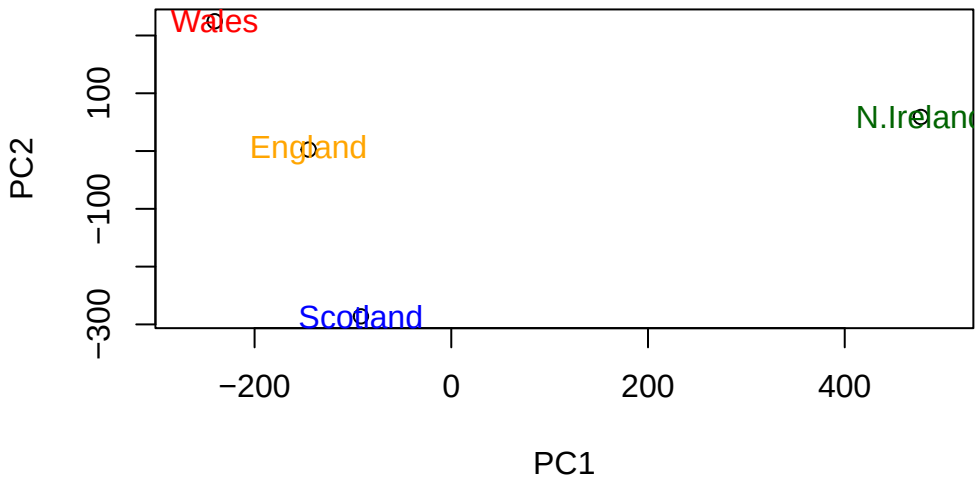
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(pca$x[,1], pca$x[,2])
```

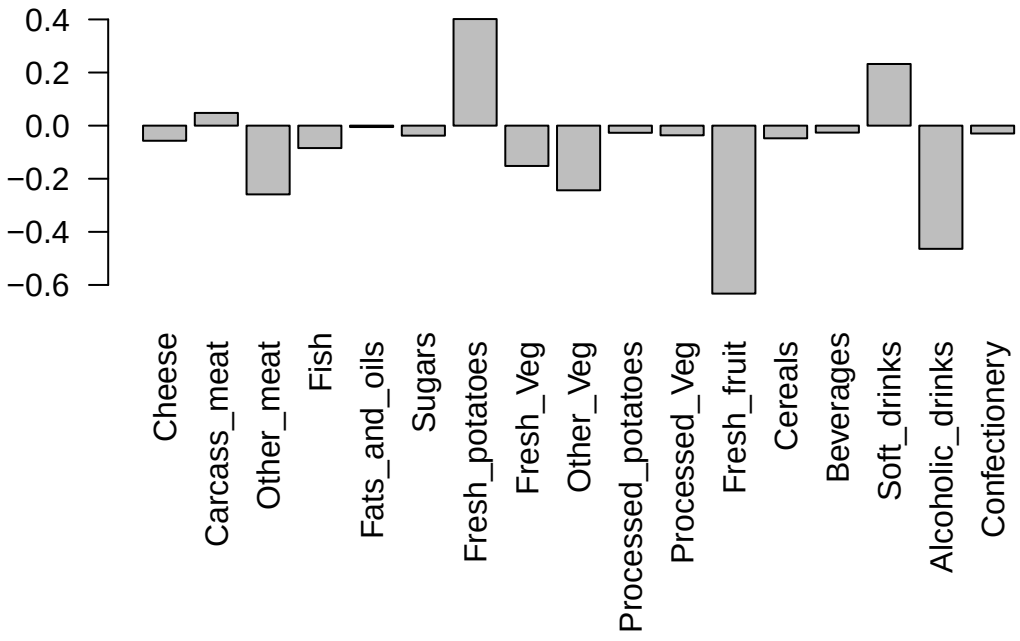


Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
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", "darkgreen")))
```

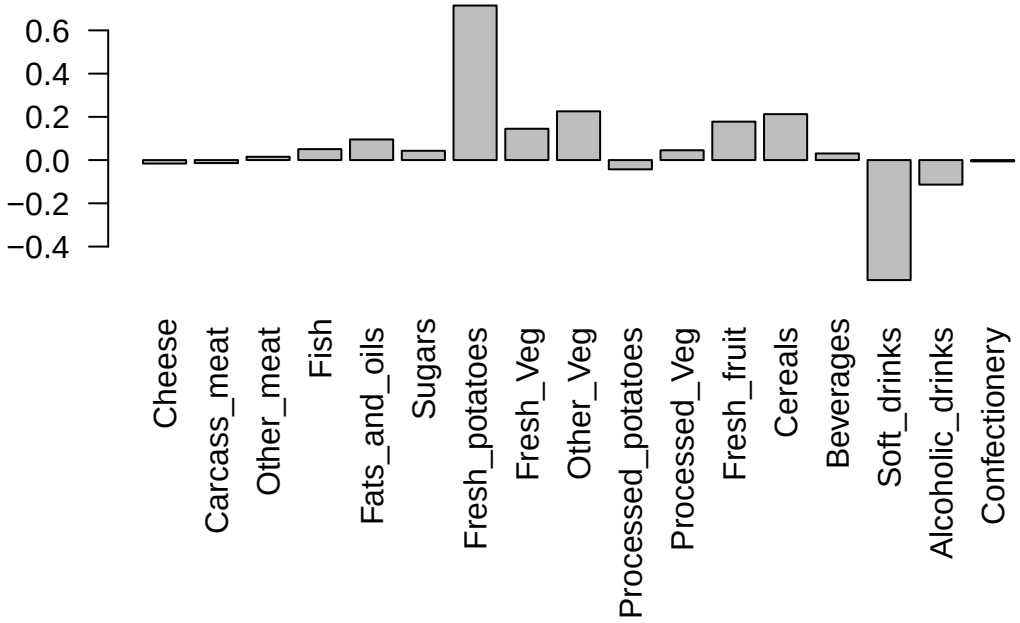



```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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 main food groups that feature prominently are fresh potatoes and soft drinks. PC2 mainly can inform us about what is "pushing" the data down or up.

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```

Q10. How many genes and samples are in this data set?

```
dim((rna.data))
```

```
[1] 100  10
```

There are 100 rows and 10 columns in this data set