Class 7: Machine Learning

Tyler Sy (PID: A16651122)

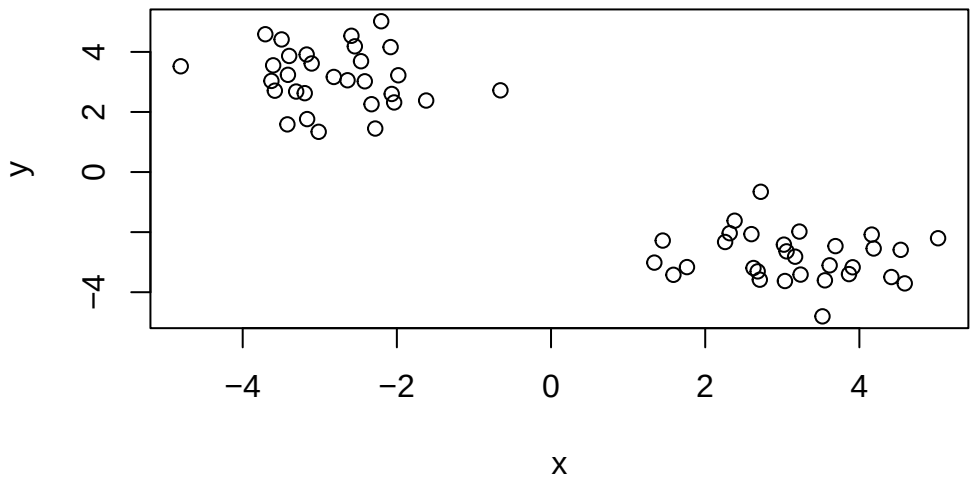
First up kmeans()

Demo of using kmeans() function in base R. First make up some data with a known structure.

```
tmp <- c(rnorm(30, mean = -3), rnorm(30, mean = 3))
tmp
```

```
 [1] -3.6048212 -3.3064579 -3.1702295 -4.8040287 -1.9813723 -3.1052244
 [7] -2.2031373 -2.2806638 -2.0352280 -3.7066133 -2.4650138 -3.1642207
[13] -3.4953618 -2.8181282 -0.6563769 -2.6399572 -3.0145158 -2.5453771
[19] -1.6198640 -2.0668551 -2.3286143 -3.5832448 -2.4161378 -3.4132781
[25] -3.6276797 -3.4201344 -2.0814357 -3.3971109 -3.1952386 -2.5894616
[31]  4.5364119  2.6264207  3.8656337  4.1600720  1.5876187  3.0342465
[37]  3.2372023  3.0211158  2.7084021  2.2583015  2.5997044  2.3813470
[43]  4.1854745  1.3398880  3.0555506  2.7205812  3.1662187  4.4150636
[49]  1.7636023  3.6896833  4.5883890  2.3177353  1.4490747  5.0209616
[55]  3.6142101  3.2219629  3.5212633  3.9129110  2.6804622  3.5526973
```

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



Now we have some made up data in `x` let's see how kmeans works with this data

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

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

Cluster means:
          x         y
1  3.141074 -2.824526
2 -2.824526  3.141074

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] 45.65137 45.65137
 (between_SS / total_SS =  92.1 %)

Available components:
```

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

Q. How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. How do we get to the cluster membership/assignment?

```
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
```

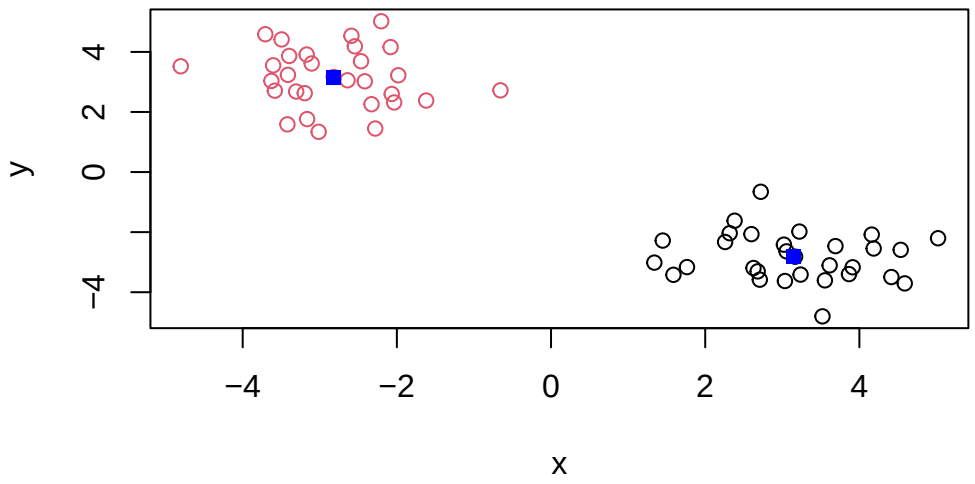
Q. What about cluster centers?

```
k$centers
```

```
          x         y
1  3.141074 -2.824526
2 -2.824526  3.141074
```

Now that we got to the main results, let's use them to plot our data with the kmeans result.

```
plot(x, col = k$cluster)
points(k$centers, col = 'blue', pch = 15)
```



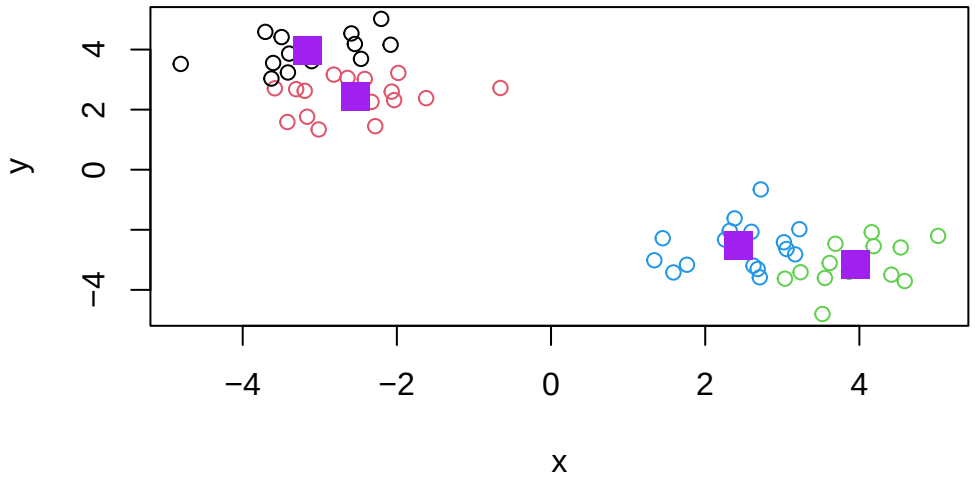
Cluster the data into 4 groups now

```
k_4 <- kmeans (x, centers = 4)
k_4
```

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

Cluster means:
          x         y
1 -3.157769  3.952444
2 -2.532938  2.431124
3  3.952444 -3.157769
4  2.431124 -2.532938

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

Within cluster sum of squares by cluster:
[1] 11.01077 14.44454 11.01077 14.44454
 (between_SS / total_SS =  95.6 %)
```

```
Available components:

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

```r
plot(x, col=k_4$cluster)
points(k_4$centers, col="purple", pch=15, cex=2)
```



Now for Hierarchical Clustering

We will cluster the same data `x` with the `hclust()`. In this case `hclust()` requires a distance matrix as input.

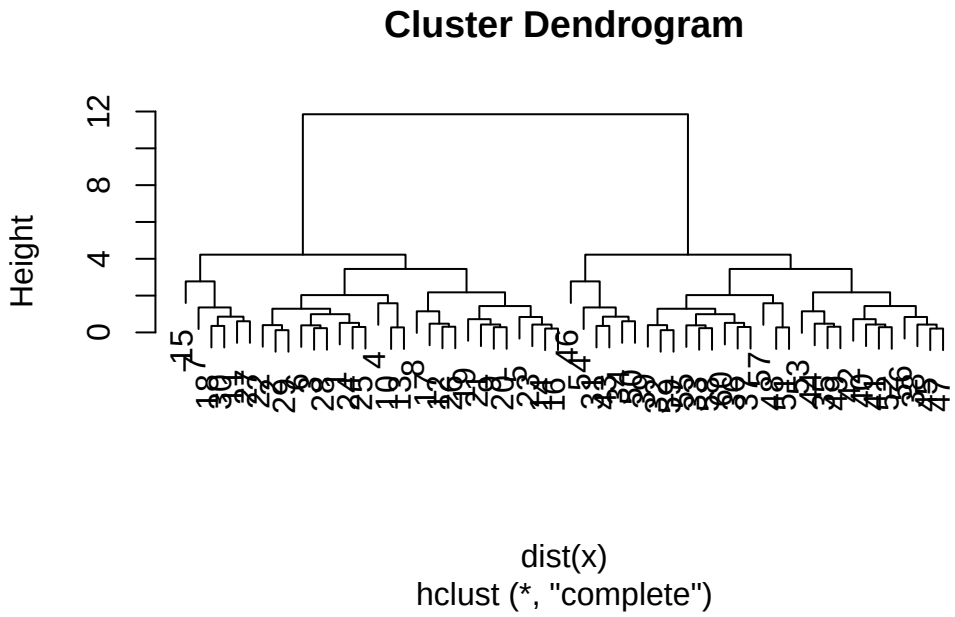
```r
hc <- hclust(dist(x))
hc
```

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

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

Let's plot our hclust result

```
plot(hc)
```



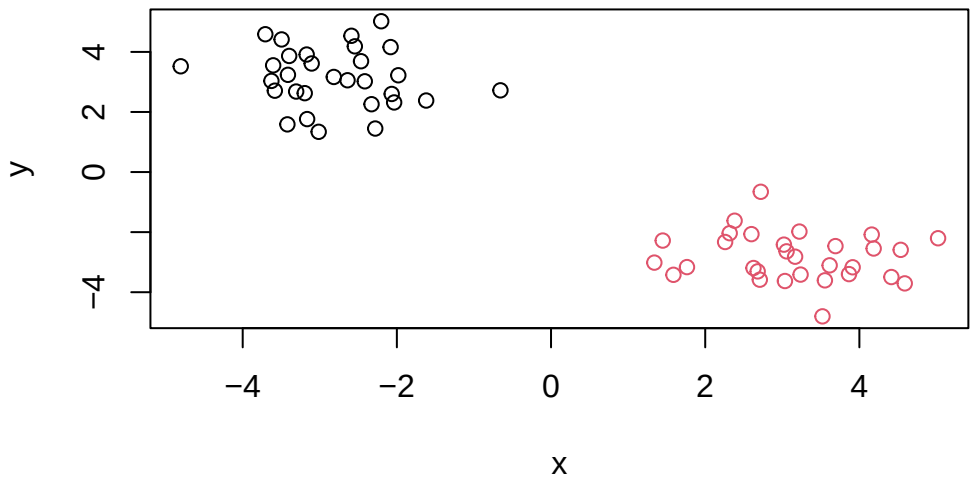
To get our cluster membership vector, we need to "cut" the tree with the `cutree()`.

```
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 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
```

Now plot our data with the hclust() results.

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



Principal Component Analysis (PCA)

```
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?

```
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
```

```
dim(x)
```

```
[1] 17  5
```

There are 17 columns and 5 rows.

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
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 dimensions have been changed.

```
dim(x)
```

```
[1] 17  4
```

The proper amount of rows and columns are 17 and 4 respectively.

Alternative method

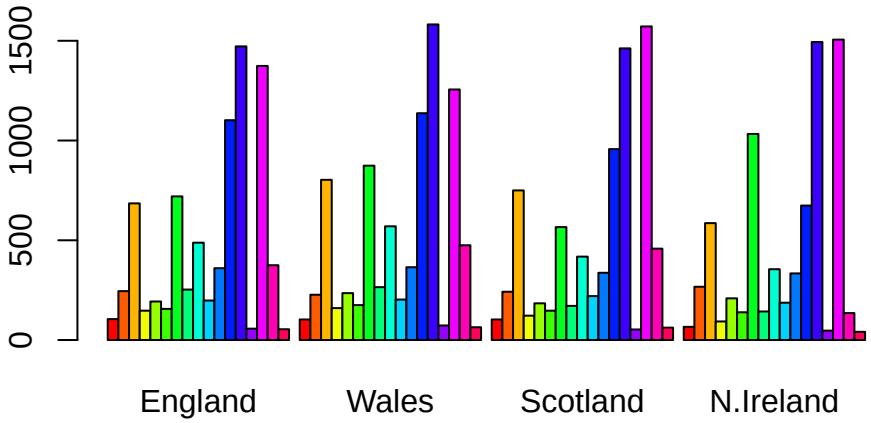
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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?

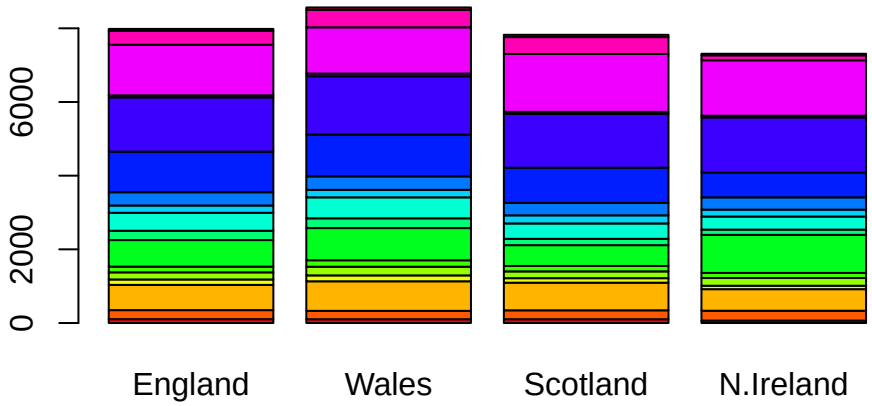
I prefer the second option since the first option results in an error if it is run too many times. Thus, the second option is more robust since it can be more consistently ran.

```
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)))
```

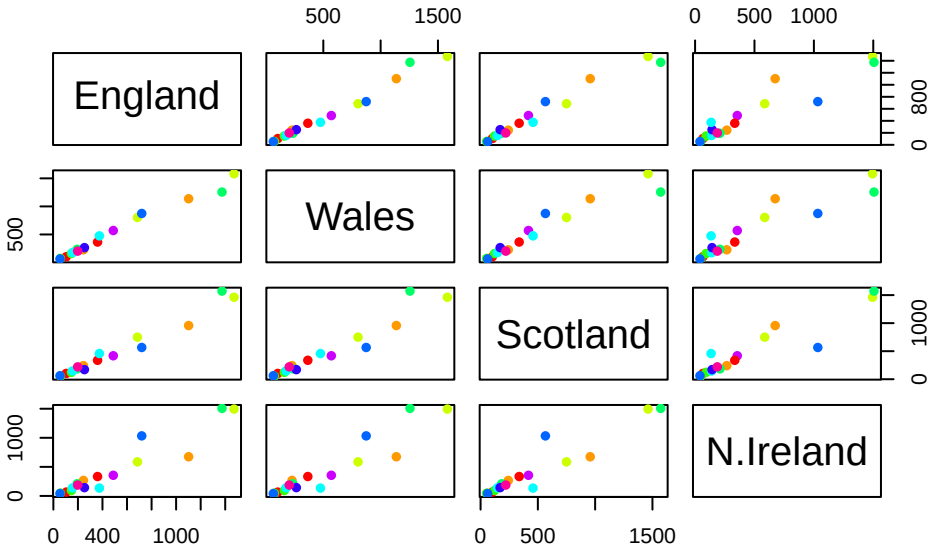


Changing `beside = T` to `beside = F` or simply removing the argument `beside = T` results in the plot above.

> 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 following code determines the pairs between matching values for those that are in the same countries against one another. The y-axis is a certain country in that row. As a result, the graphs represent each pair of countries plotted against each other. If a point is on a diagonal, there isn't a difference in that category for the countries that are being compared.

```
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?

The data points in each of the graphs for N. Ireland are not as diagonal. This suggests a greater difference in food consumption relative to the other countries of the UK. The points that differ the most are the orange and blue points.

PCA to the Rescue

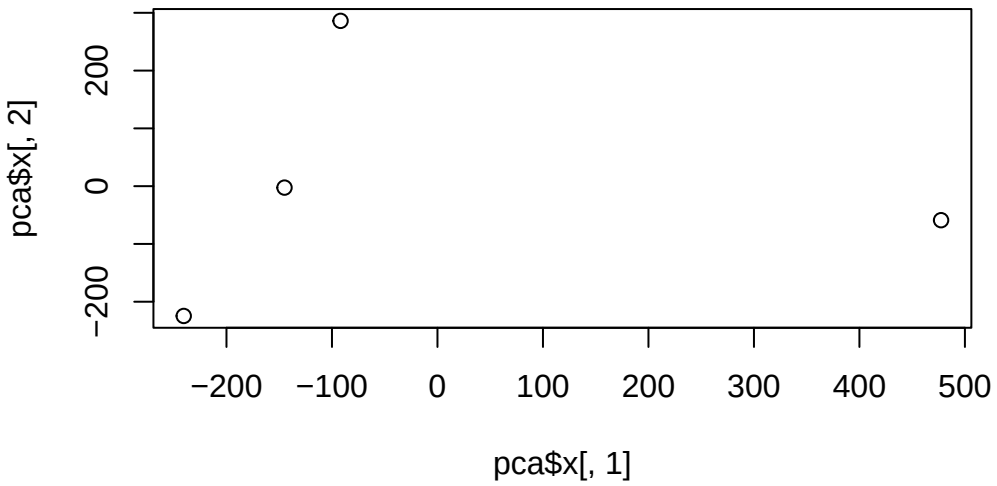
```
# Use the prcomp() PCA function
pca <- prcomp(t(x))
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 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
```

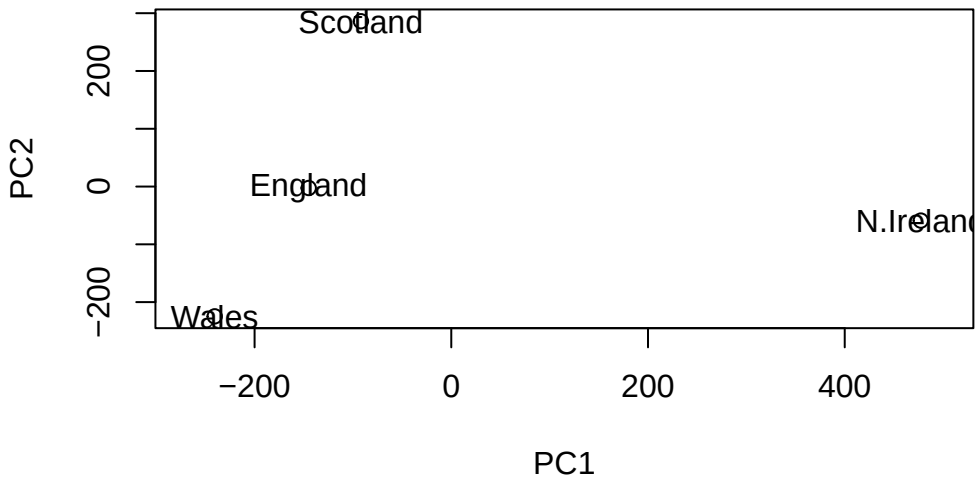
PC1 captures 67.44% and PC2 captures 29.05% for a total of 96.5% variance captured in the first 2 PCs.

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

```
# Plot of the first 2 PCs
plot(pca$x [,1], pca$x[,2])
```

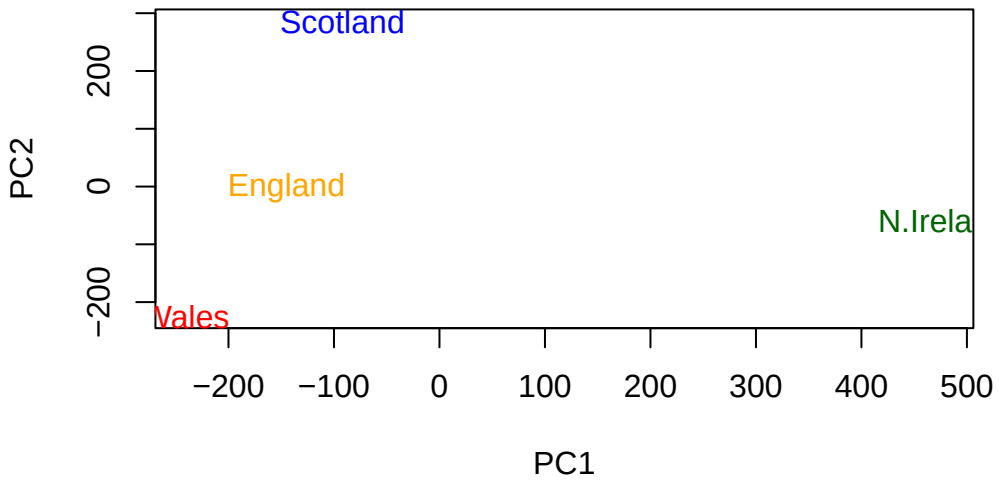



```
# Plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2", xlim = c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



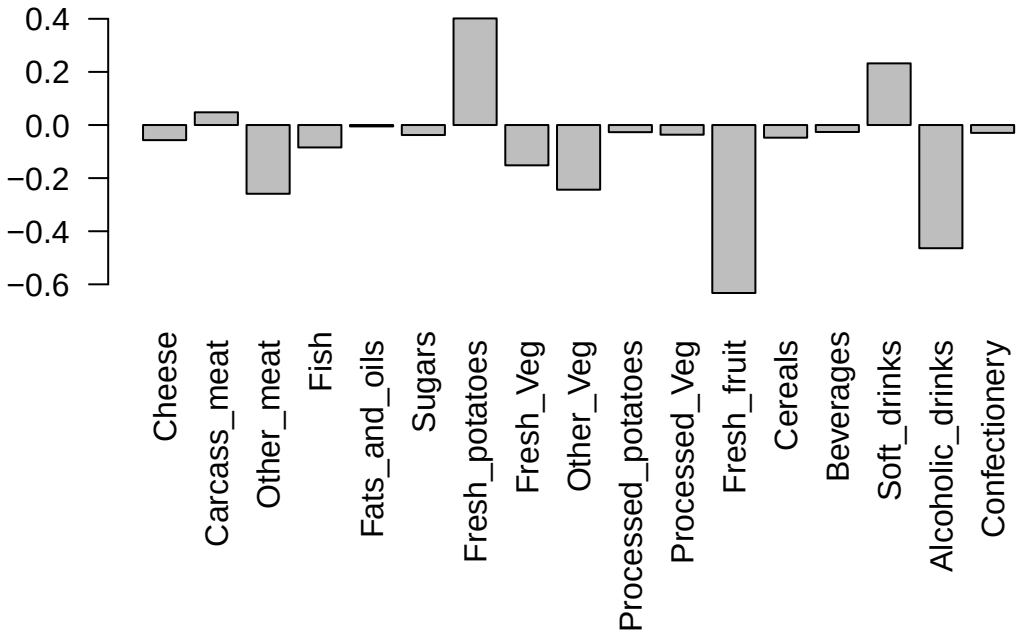
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
my_colors <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], pch = -10, xlab = "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(x), col = my_colors)
```



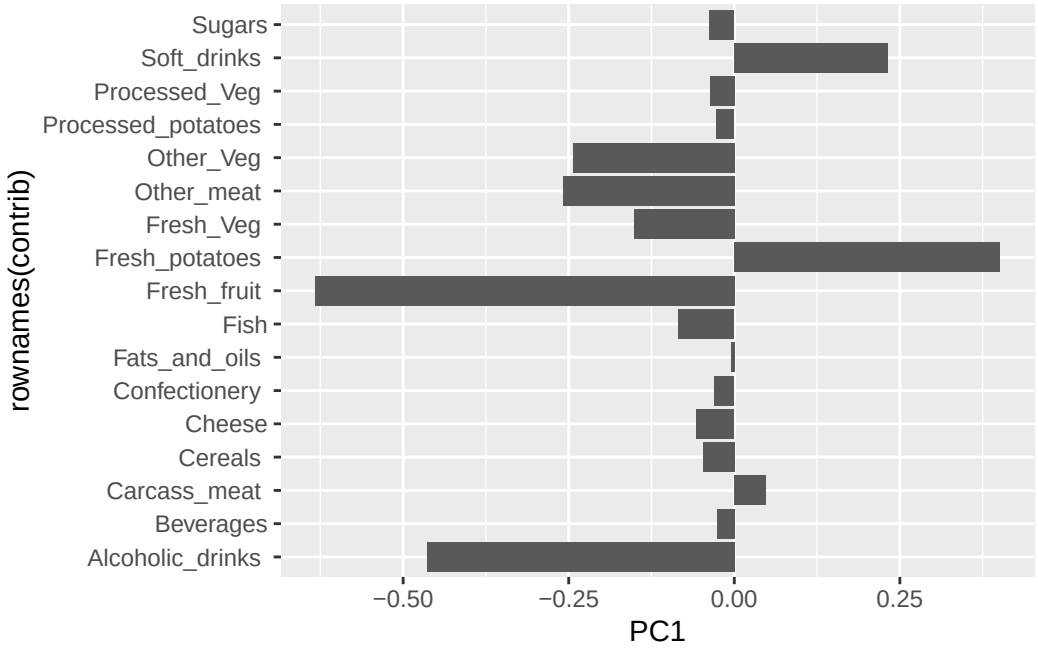
Digging Deeper

```
## 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 )
```



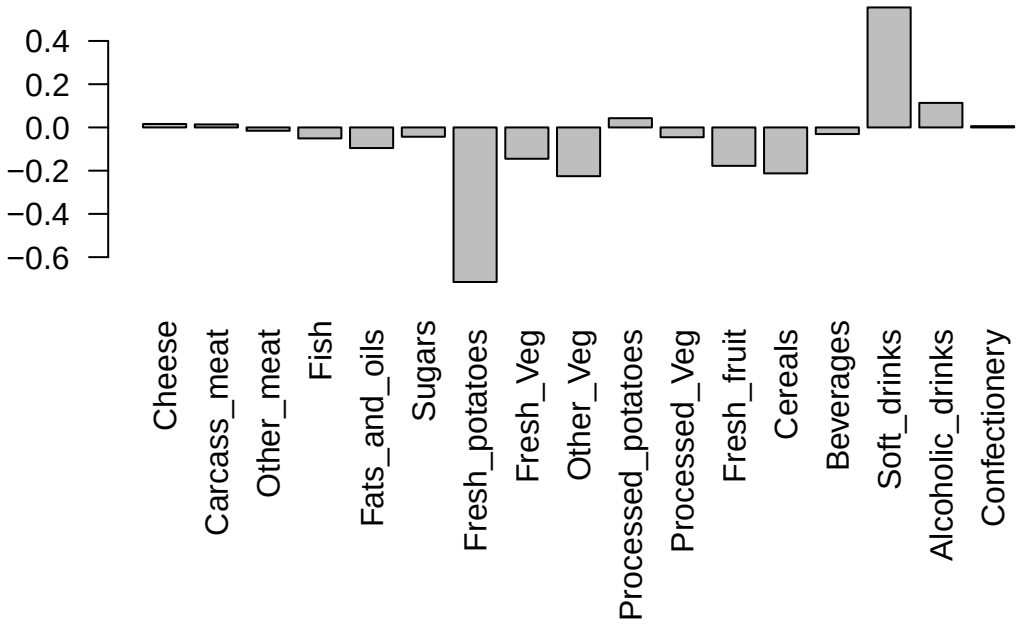
Plot along PC1.

```
library(ggplot2)
contrib <- as.data.frame((pca$rotation))
ggplot(contrib) +
  aes(PC1, rownames(contrib)) + geom_col()
```

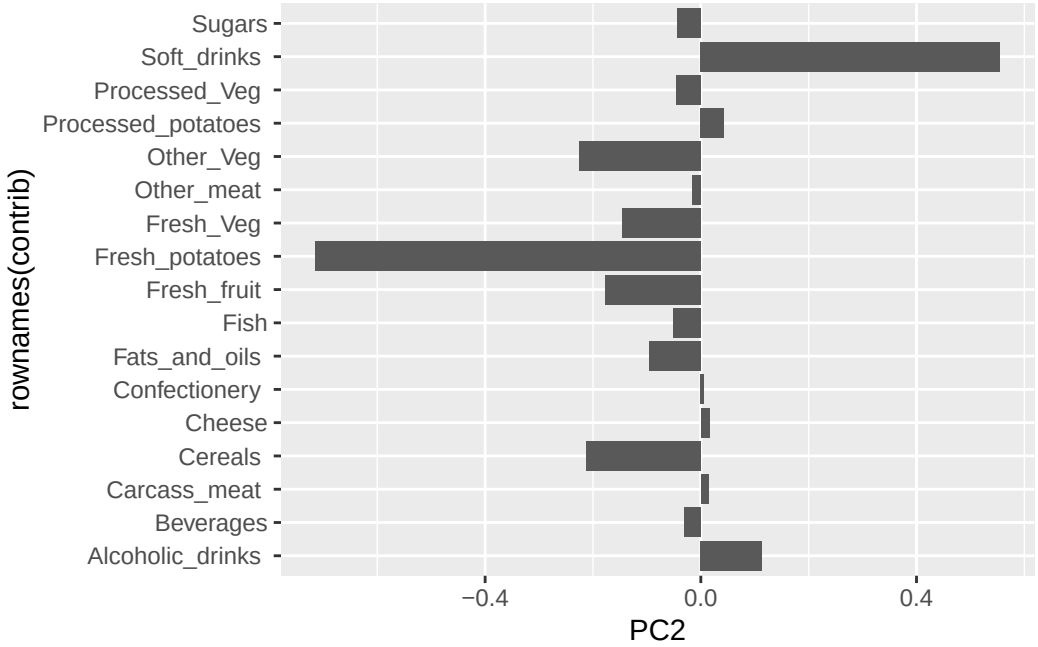


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

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




```
ggplot(contrib) +
  aes(PC2, rownames(contrib)) + geom_col()
```



Soft drinks and fresh potatoes are the most prominent food groups and PC2 mainly tells us that the variance is due to these two food groups - soft drinks and fresh potatoes.