# Class 07: ML 1

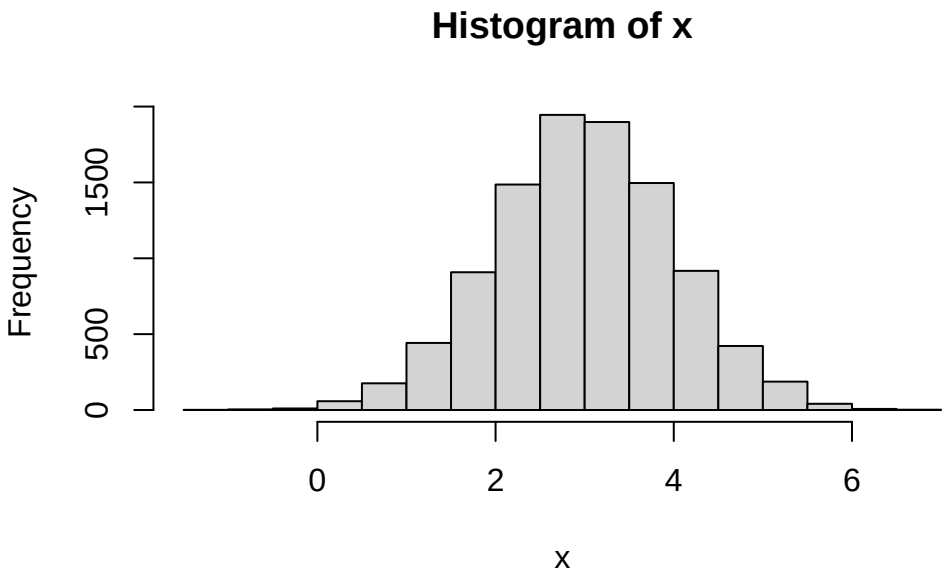
James Woolley (A16440072)

#clustering

Today we will start by exploring clustering methods such as K-means using the `kmeans()` function.

We can begin with made up data so we know what the answer should be.

```
x <- rnorm(10000, mean=3)
hist(x)
```



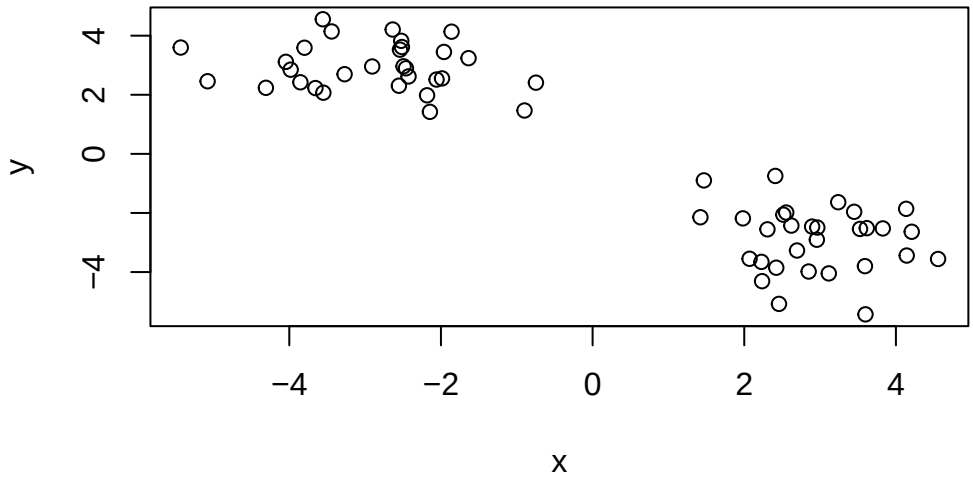

60 points

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

```
           x         y
[1,] 2.457858 -5.077181
[2,] 2.515566 -2.057764
[3,] 3.591910 -3.799588
[4,] 3.449359 -1.960568
[5,] 4.556085 -3.557897
[6,] 2.235217 -4.308188
```

We can take a look with base R plot

```
plot(x)
```




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

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

```
Cluster means:
          x         y
1  2.935222 -2.883363
2 -2.883363  2.935222

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 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

Within cluster sum of squares by cluster:
[1] 53.96118 53.96118
 (between_SS / total_SS =  90.4 %)

Available components:

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

Q1. How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q2. Cluster membership

```
k$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 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
```

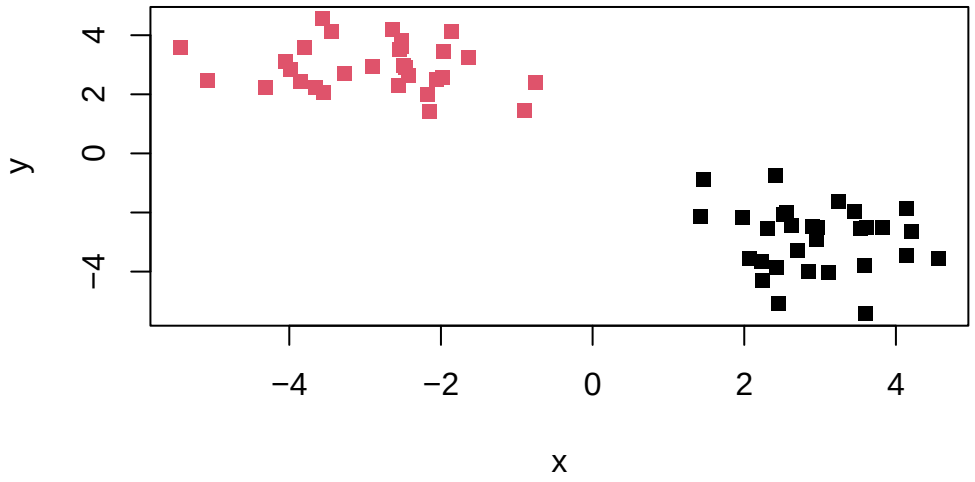
Q3. Cluster centers?

```
k$centers
```

```
          x         y
1  2.935222 -2.883363
2 -2.883363  2.935222
```

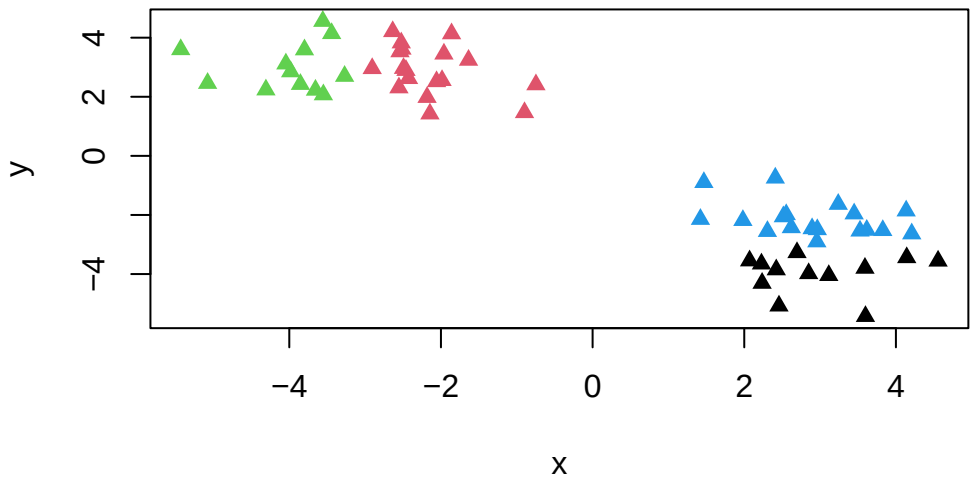
Q4. Plot the restults

```r
plot(x, col=k$cluster, pch=15)
```




Q5. CLuster the data again with kmeans() into 4 groups and plot the results.

```r
k4 <- kmeans(x,centers=4,nstart=20)
plot(x, col=k4$cluster, pch=17)
```



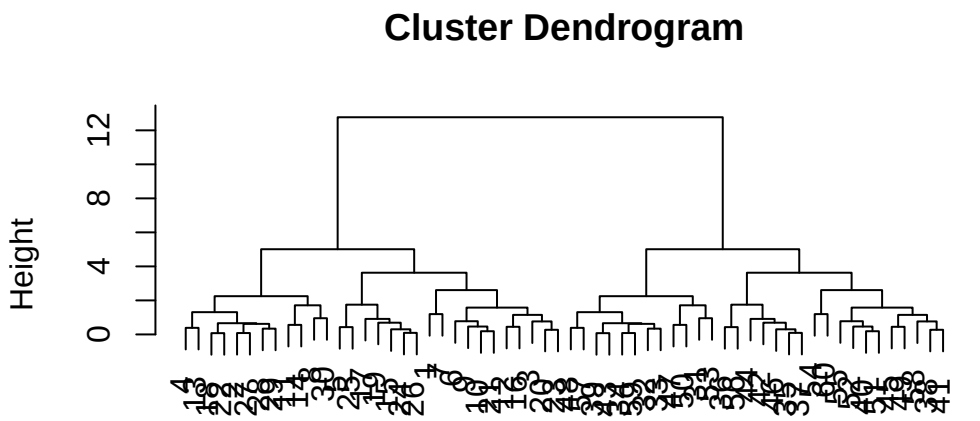

`kmeans` will always spit out what you tell it, even if it makes no sense (above). So if you don't know how many groups you SHOULD have, how do you determine how many centers data has?

kmeans is very popular because of how fast it is, but it

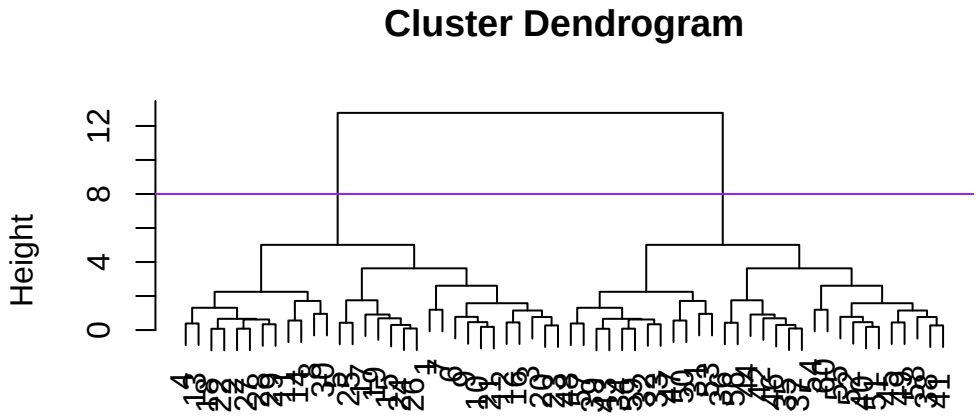
## Hierarchical Clustering

The way to use hclustering in base R is through the `hclust()` function. You have to pass it in a distance matrix. It won't work with ur input data.

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




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



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

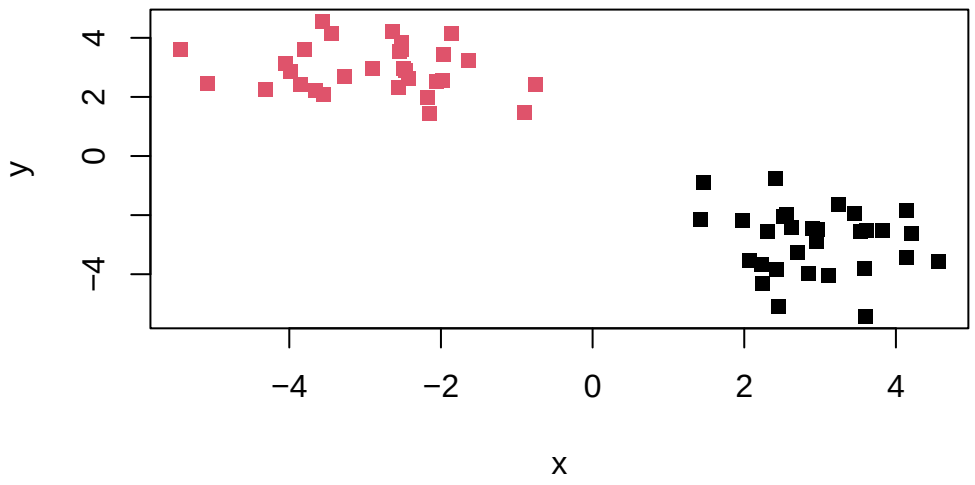
To find the clusters from a `hcluster`

```
table(grps)
```

```
grps
 1  2
30 30
```

Q6. Plot the hclust results

```
plot (x, col=grps, pch=15)
```




## PCA of food consumed in the UK

Consumption of 17 different types of food measured and averaged in England, Scotland, Wales, and N Ireland.

We can use PCA to help us analyse the data, but we can begin with conventional analysis.

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
rownames(x) <- x[,1] #set correct row and col names
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
```

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

You can use the `nrow()` and `ncol()` functions to answer this question.

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 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?

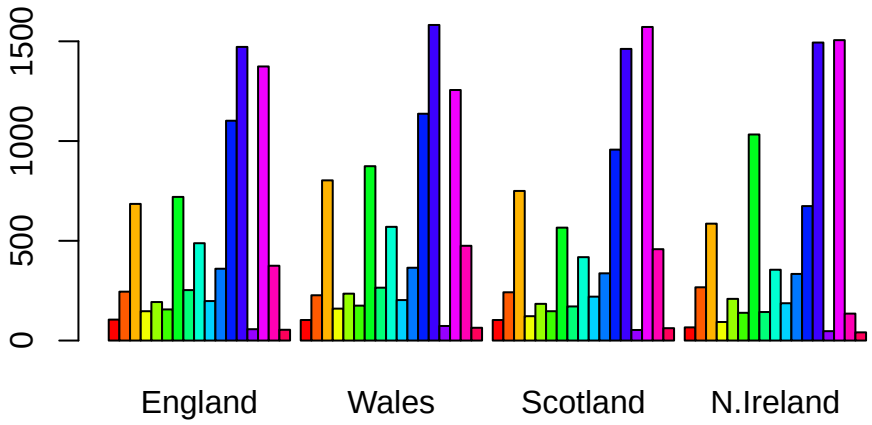
Using `x <- read.csv(url, row.names=1) head(x)`is better than the other method, because it will fix the row-names column without removing columns. If you keep removing columns, you will lose data. This method sets the correct column instead of subtracting to arrive at the correct column.

```
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
```

We can visualize the data in a bar plot!

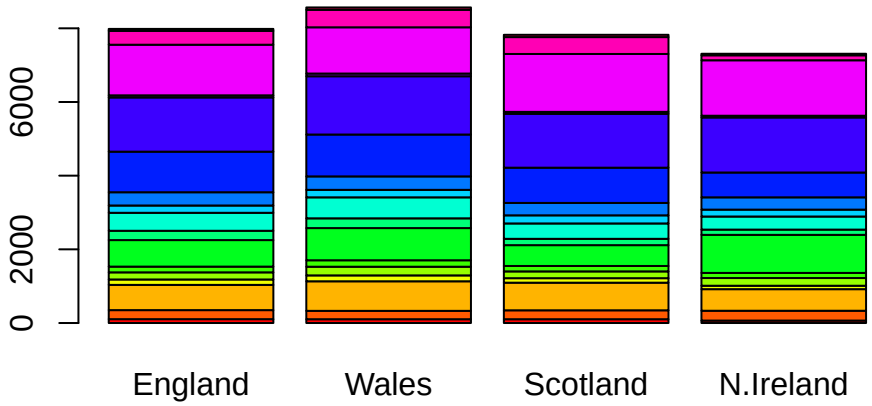
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




But that's not very helpful, so we want to try to put the data together in a way that makes it easier to understand.

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

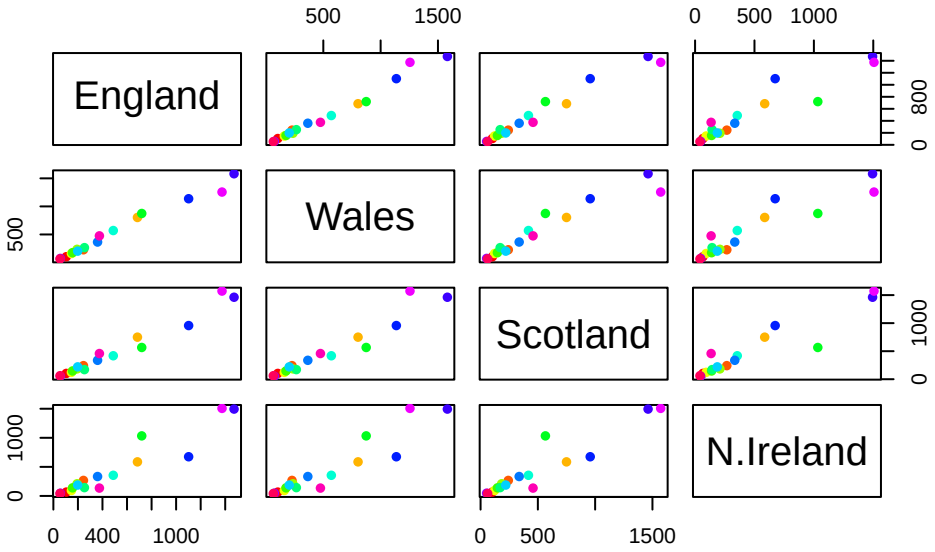
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```




This stacks the data by country instead of producing many tiny little bars.

> 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(17), pch=16)
```




The closer a point is to the diagonal, the more similar the consumption for that point is between the two different countries.

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

Northern Ireland consmes more of the blue point and less of the green point than the other countries, and you know because it

## Principal component analysis (PCA)

PCA helps visualise data like this even better. THe main function in base R to use PCA is `prcomp()`. IN this case we want to first take the transpose of the original data `x` so the columns and rows are switched.

```
head(t(x))
```

```
           Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
England       105           245         685  147            193    156
Wales         103           227         803  160            235    175
Scotland      103           242         750  122            184    147
N.Ireland      66           267         586   93            209    139
```

```
          Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
England              720        253        488                 198
Wales                874        265        570                 203
Scotland             566        171        418                 220
N.Ireland           1033        143        355                 187
          Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England             360         1102     1472         57         1374
Wales               365         1137     1582         73         1256
Scotland            337          957     1462         53         1572
N.Ireland           334          674     1494         47         1506
          Alcoholic_drinks  Confectionery
England                375             54
Wales                  475             64
Scotland               458             62
N.Ireland              135             41
```

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

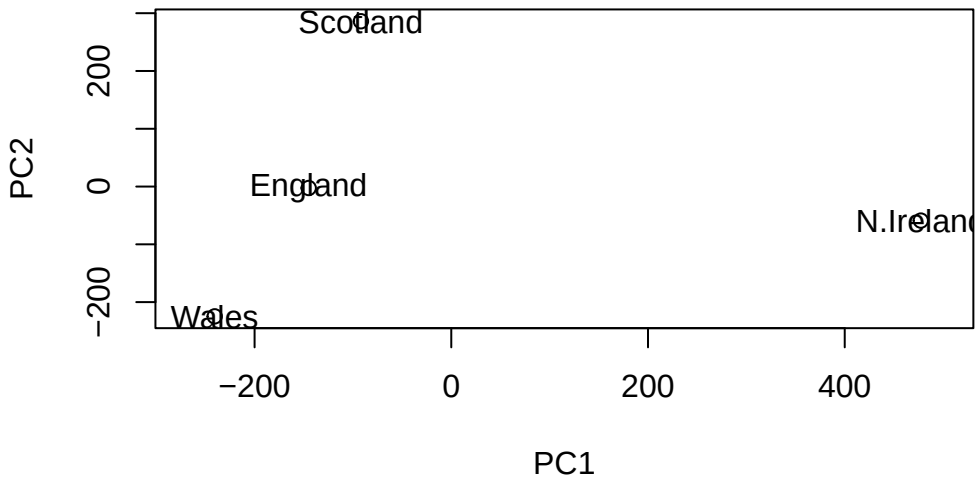
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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
```

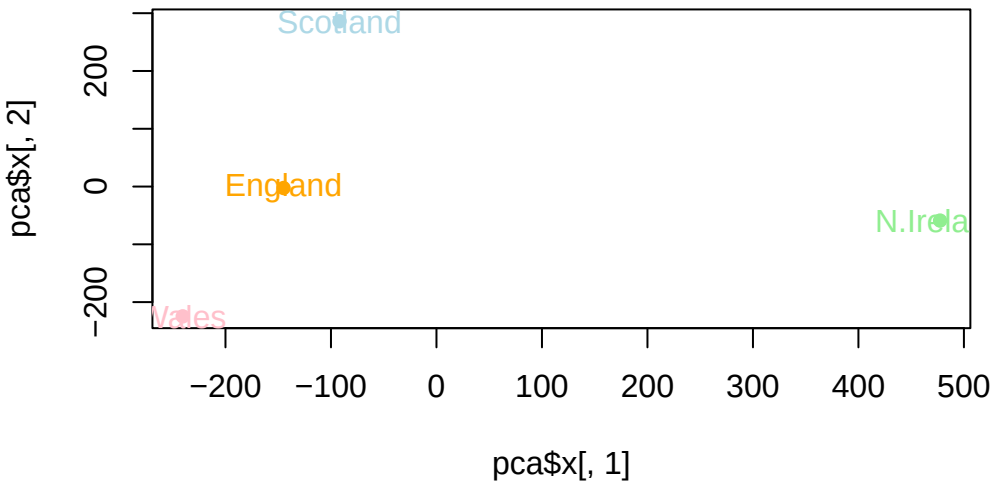
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

```
plot(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.

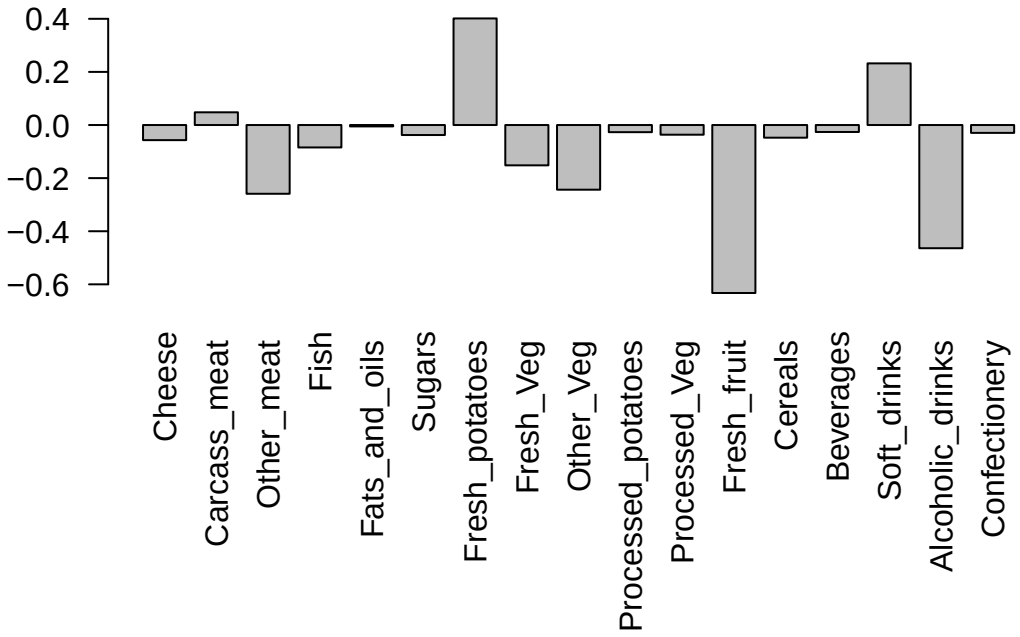
```
plot(pca$x[,1], pca$x[,2], col=c("orange", "pink", "lightblue", "lightgreen"), pch=16)
text(pca$x[,1], pca$x[,2], colnames(x),col=c("orange", "pink", "lightblue", "lightgreen"))
```




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

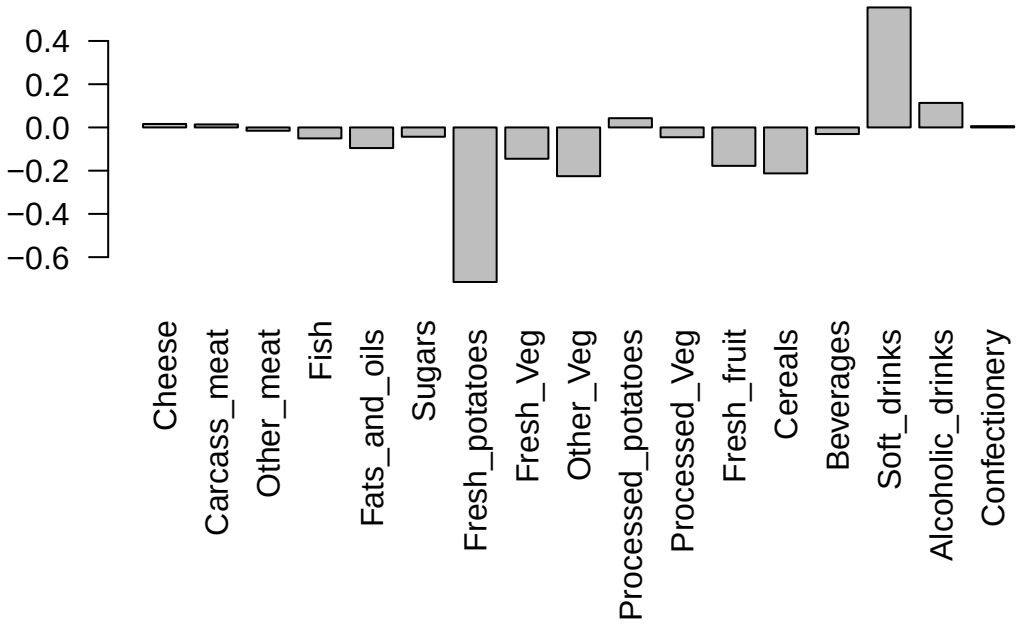
This will help create a PC1 plot.

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




TO make a PC2 plot, use

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




this shows that the biggest difference is Fresh Potatoes and Soft Drinks. This shows us variance that PC 1 didn't account for.