Class 07: Machine Learning 1

Tomi Lane Timmins

In this class, we will explore clustering and dimensionality reduction methods.

K-means

(K is number of clusters - number of groups that you want. In k clustering, we provide k.)

Make up some input data where we know what the answer should be.

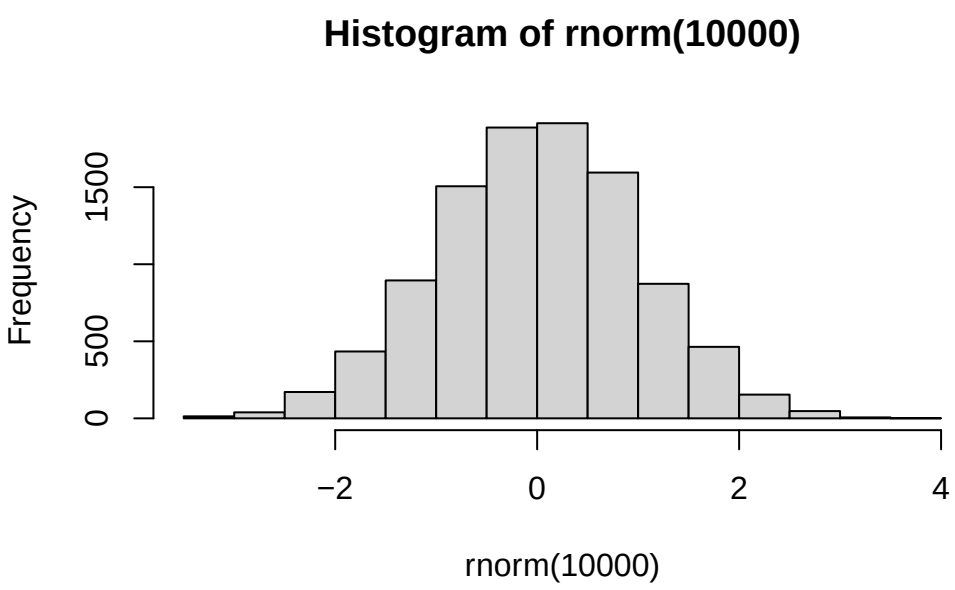
SIDENOTE: rnorm() - if you input 10, it will give you 10 points, all centered around 0. (only input is n and n = # observations we want)

```
rnorm(10)
```

```
 [1] -0.1415884 -0.9047940 -0.2118333 -0.6950032  1.5461874  1.9897543
 [7] -1.5422812 -0.2438224  0.6129740  0.1370930
```

If we make a histogram of rnorm(),it gives a standard distribution

```
hist(rnorm(10000))
```



You can change where it centers (one of the inputs is mean (default 0):

```
rnorm(30, -3)
```

```
 [1] -2.4490336 -1.1600266 -3.5182309 -2.5158638 -2.5609998 -3.7318190
 [7] -3.0018916 -2.6457103 -4.9464354 -1.4899516 -0.9822045 -2.8485292
[13] -4.9385672 -3.7944959 -2.3402990 -2.0806426 -4.8534524 -2.8845203
[19] -1.5942467 -3.8552493 -3.5271218 -2.1193055 -4.0335303 -3.9167149
[25] -3.4680152 -4.0088608 -2.3823760 -2.4075079 -2.3025920 -3.0501546
```

Back to making up data:

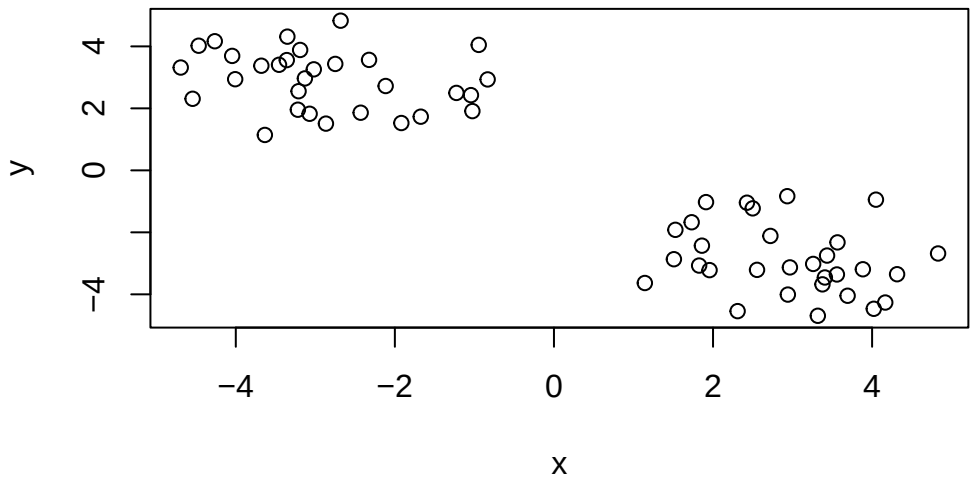
```
tmp <- c( rnorm(30, -3), rnorm(30, 3) )
# cbind binds together two columns:  the rev(dataset) (reverses the direction of the data
x <- cbind(x = tmp, y = rev(tmp))
head(x)
```

```
            x        y
[1,] -3.351428 4.314054
[2,] -1.918525 1.526745
[3,] -2.429774 1.860270
```

```
[4,] -0.945756 4.048156
[5,] -4.045553 3.692147
[6,] -3.191026 3.884028
```

Quick plot of x to see the code:

```
plot(x)
```



The plot above can clearly be seen as two groups.

Use the 'kmeans()' function setting k to 2 and nstart = 20. nstart specifies how many iterations.

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

km
```

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

Cluster means:
          x         y
1 -2.872607  2.922649
```

```
2  2.922649 -2.872607

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 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

Within cluster sum of squares by cluster:
[1] 63.24255 63.24255
 (between_SS / total_SS =  88.8 %)

Available components:

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

Q. How many points are in each cluster?

(check the help page to see which component will help you)

```
km$size
```

```
[1] 30 30
```

What 'component' of your reulst object details: - cluster assignment/membership - cluster center

```
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 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
```

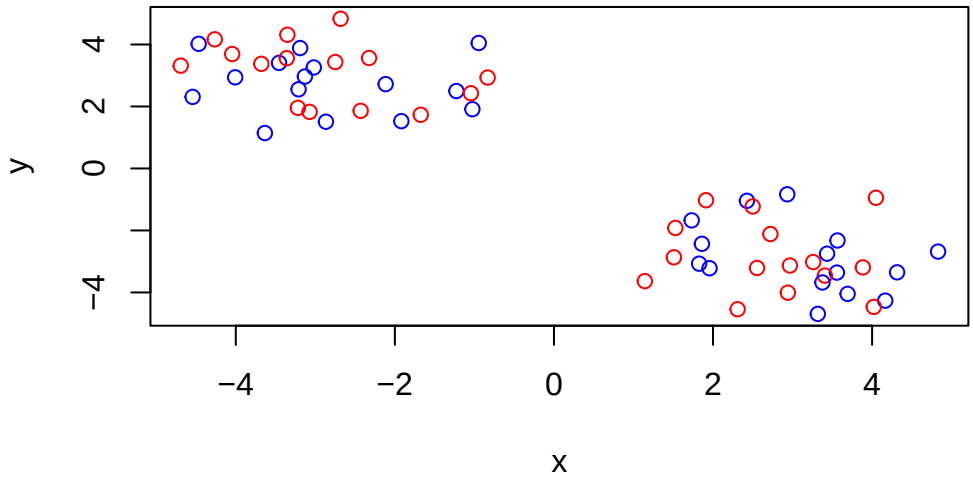
```
km$centers
```

```
          x         y
1 -2.872607  2.922649
2  2.922649 -2.872607
```

Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points.

You can color by 'col = "blue" ' or can color by 'col = 1' because colors are numbered.

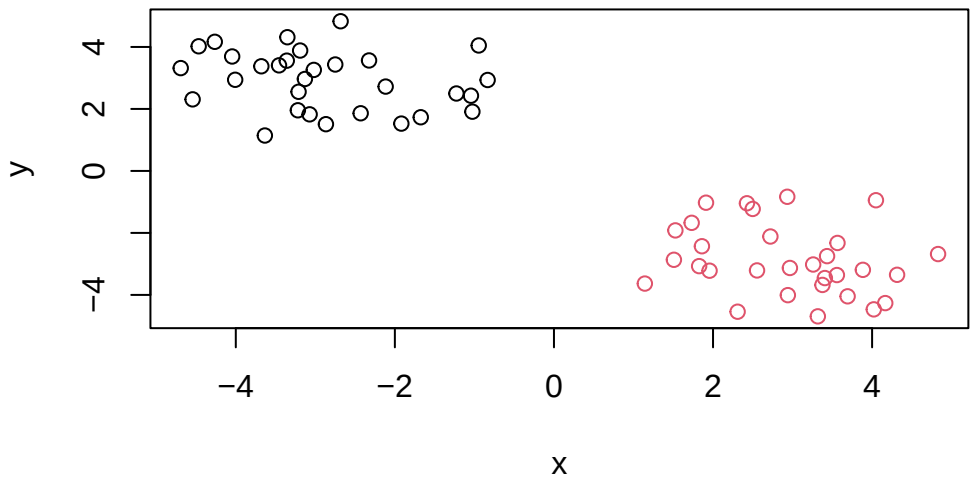
```
plot(x, col= c("red", "blue"))
```



This input alternates the color by each data point and is not what is desired.

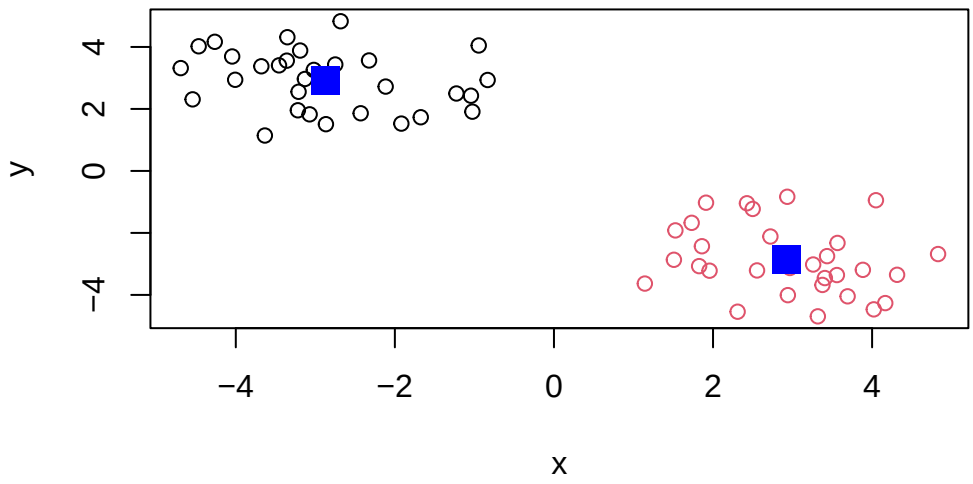
Instead, use the assignments of each variable to each cluster to code the color:

```
plot(x, col = km$cluster)
```



Then add cluster centers as blue points: (pch and cex are not necessary, just make the points bigger)

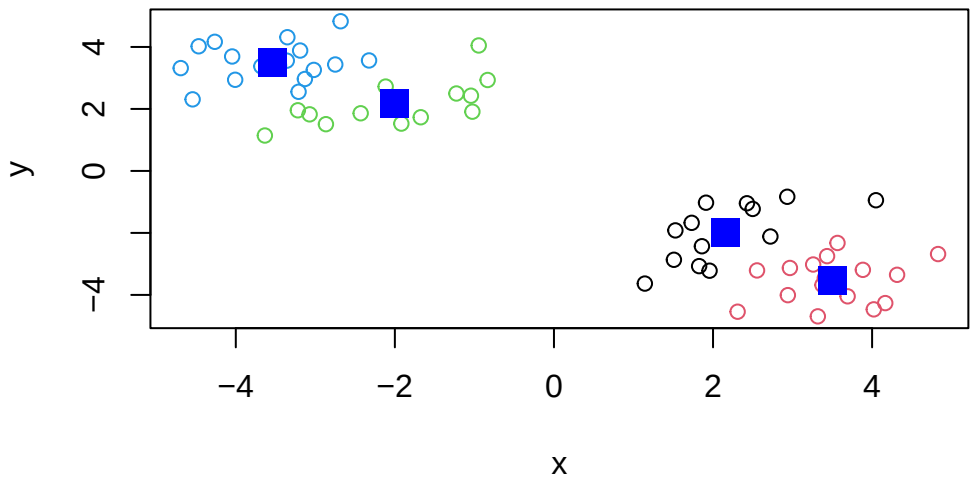
```
plot(x, col = km$cluster)

points(km$centers, col = "blue", pch =15, cex=2)
```



Play with kmeans and ask for different number of clusters:

R will provide you with the number of clusters you ask for, even if there are not that amount of clusters.

```
km2 <- kmeans(x, centers = 4, nstart = 20)

plot(x, col = km2$cluster)

points(km2$centers, col = "blue", pch =15, cex=2)
```



Hierarchical Clustering:

This is another very useful and widely employed clustering method which has the advantage over kmeans in that it can help reveal something of the true grouping in your data.

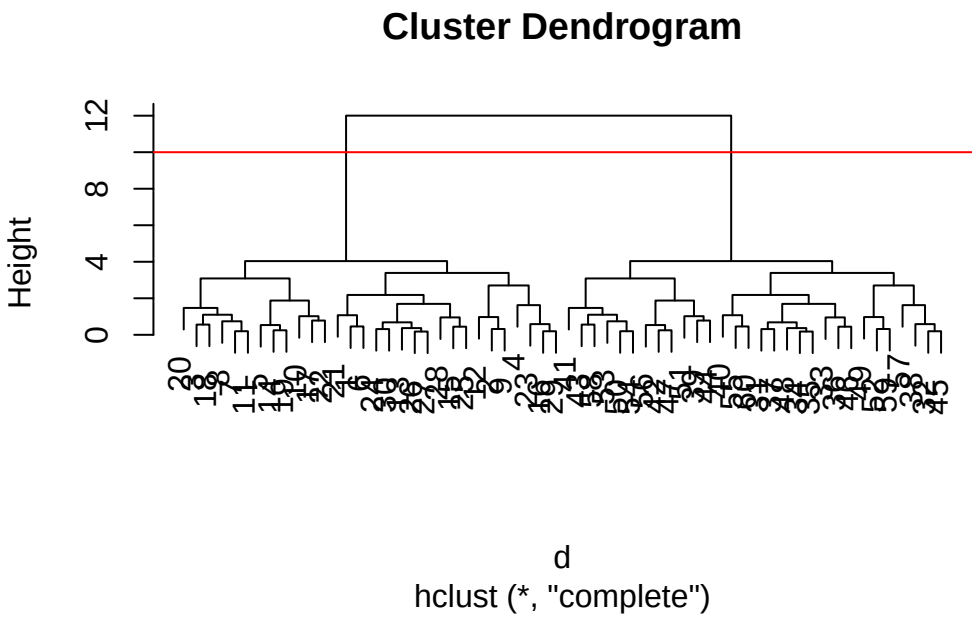
The 'hclust()' function wants a distance matrix as input. We can get this from the 'dist()' function.

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

```
Call:
hclust(d = d)

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

There is a plot method for hclust results.

```r
plot(hc)
#abline() adds a solid line at specified height
abline(h=10, col="red")
```



To get my cluster membership vector, I need to "cut" my tree to yield sub-trees or branches with all the members of a given cluster residing on the same cut branch. The function to do this is called 'cutree()' (note this is spelled with one t).

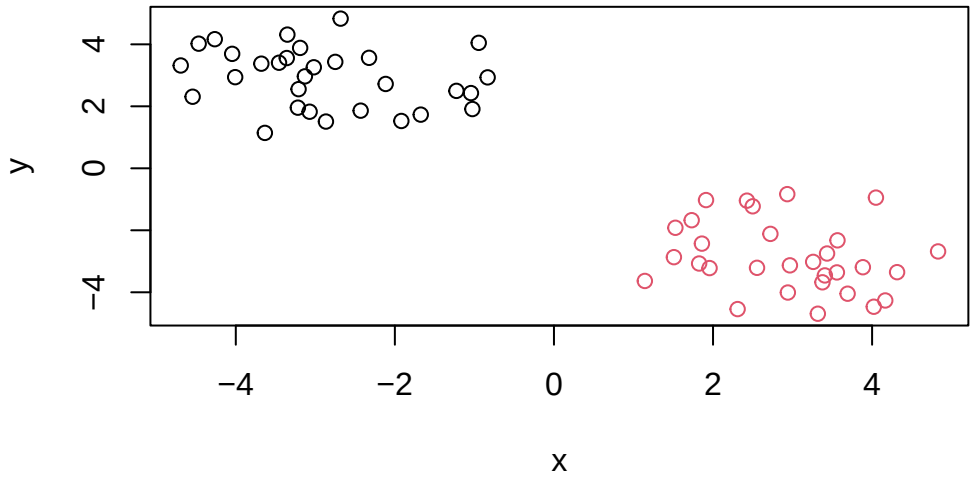
For cutree(), you can input the height you would like to cut the dendogram (h=10) or the number of clusters you would like (k=2).

It is often helpful to use the 'k=' argument rather than the 'h=' because it can be hard to distinguish the height you want for a specific number of clusters.

```r
# save grps to plot results
grps <- cutree(hc, h =10)
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
```

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



Principal Component Analysis (PCA)

Objective is to take complex datasets and make them easier to visualize and understand.

The base R function for PCA is called 'prcomp()'.

PCA 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?

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

There are 17 rows and 5 columns in this dataset. (also could have used 'dim()' function)

Preview the first 6 columns to check for issues with importing:

```
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
```

We were expecting 4 columns (one for each country), not 5! Now we have to change the rownames from classification as a column:

```
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)
```

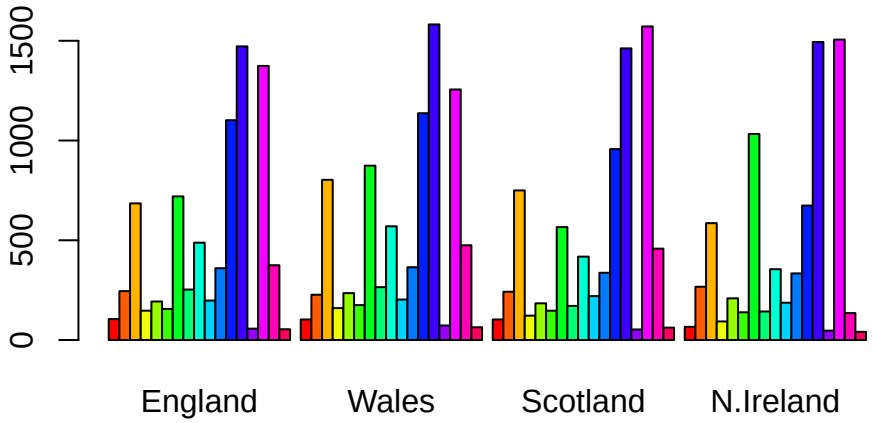
```
[1] 17  4
```

It is now fixed!There are only 4 columns now. (could also have written: 'x <- read.csv(url, row.names = 1)')

> 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 'read.csv' solution as it requires less code and leaves less room for error. The first solution could remove more rows than intended. If used multiple times, we can lose data.

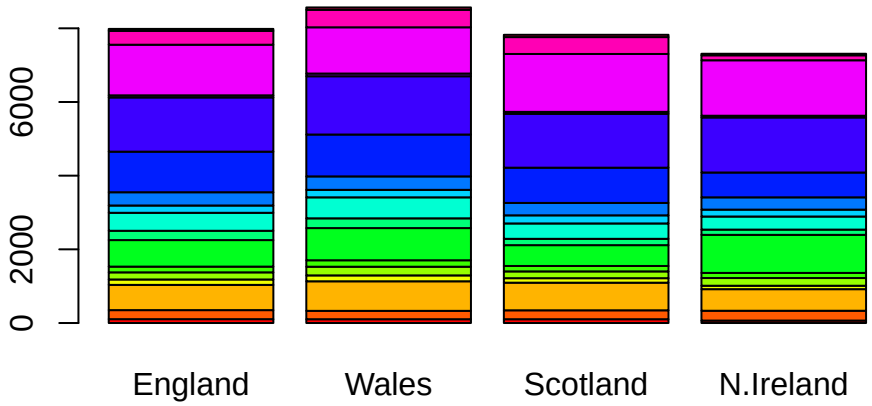
```
barplot(as.matrix(x), beside = T, col = rainbow(nrow(x)))
```



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

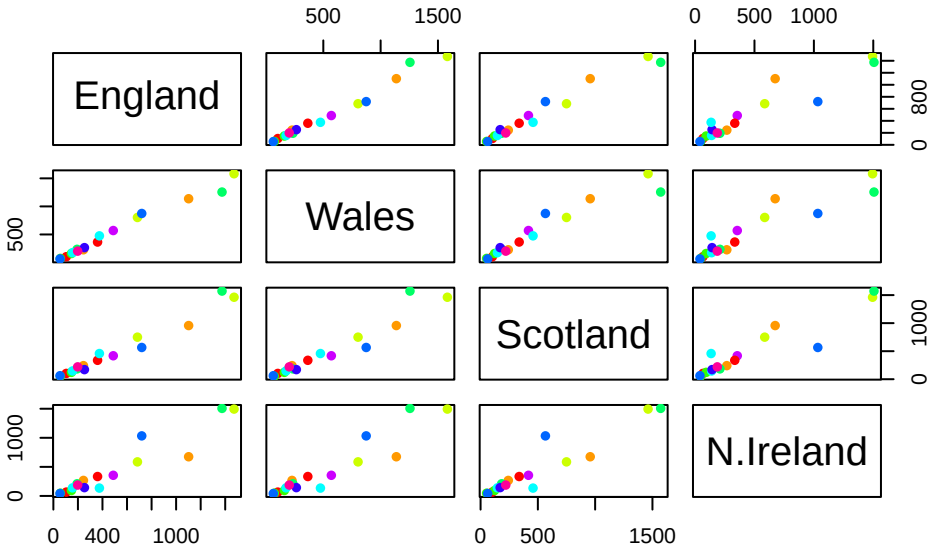
We could change 'beside'. to 'FALSE' to get stacked bars instead of juxtaposed bars.

```
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 code above produces a matrix of scatterplots. 'col' and 'pch' are just cosmetic compenents. But the diagonal (which lists each of the countries) specifies the two countries being compared in each plot in the matrix.

When a given point (type of food) lies on the diagonal, the two countries consume the same amount of that given food. More variance from the diagonal means that the two countries consume different amounts of the types of food. One can see that Northern Ireland has more variance in their consumption than the other 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 consumes more of the "dark blue" datapt, less of the "orange" datapt, and less of the "light blue" data pt. They consume more soft drinks, and less fresh fruit and alcohol.

Note: 't' is transverse of x (countries go from columns to rows)

```
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
```

Note: 'proportion of variance' is variance captured with that particular PCA, 'cumulative proportion' is the variance captured with each PCA up to that point. For example, PCA1 and PCA2 capture 96% of the variation.

A "PCA plot" (a.k.a. "Score Plot", PC1vsPC2 plot, etc)

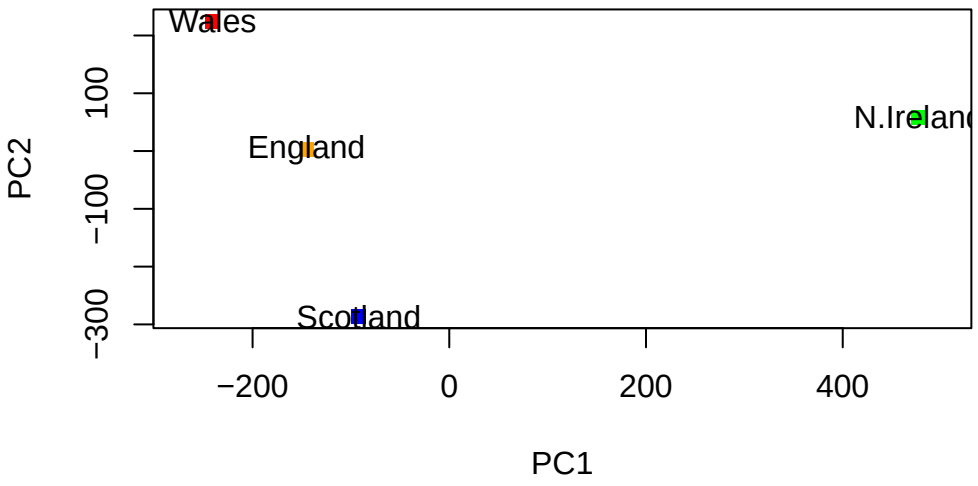
```
pca$x
```

```
                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
```

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

(answered Q7 + Q8 in same plot)

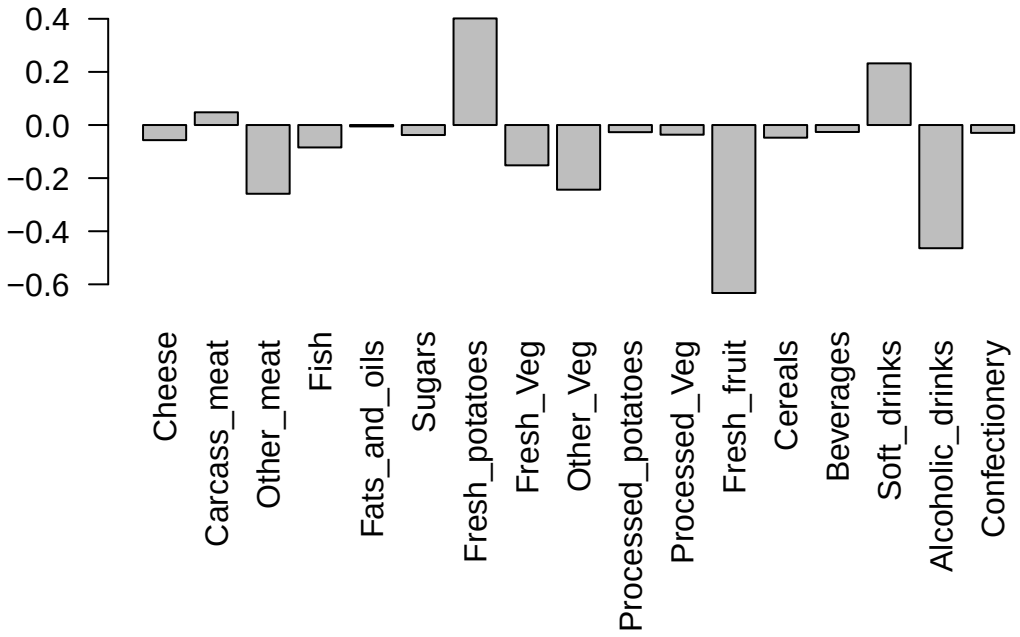
```
plot(pca$x[,1], pca$x[,2],
     col = c( "orange", "red", "blue", "green"), pch = 15, xlab = "PC1", ylab = "PC2", xli

text(pca$x[,1], pca$x[,2], colnames(x))
```



From this plot, we can see that Ireland is farther from the other countries along the PC1 axis, which means there is more variance.

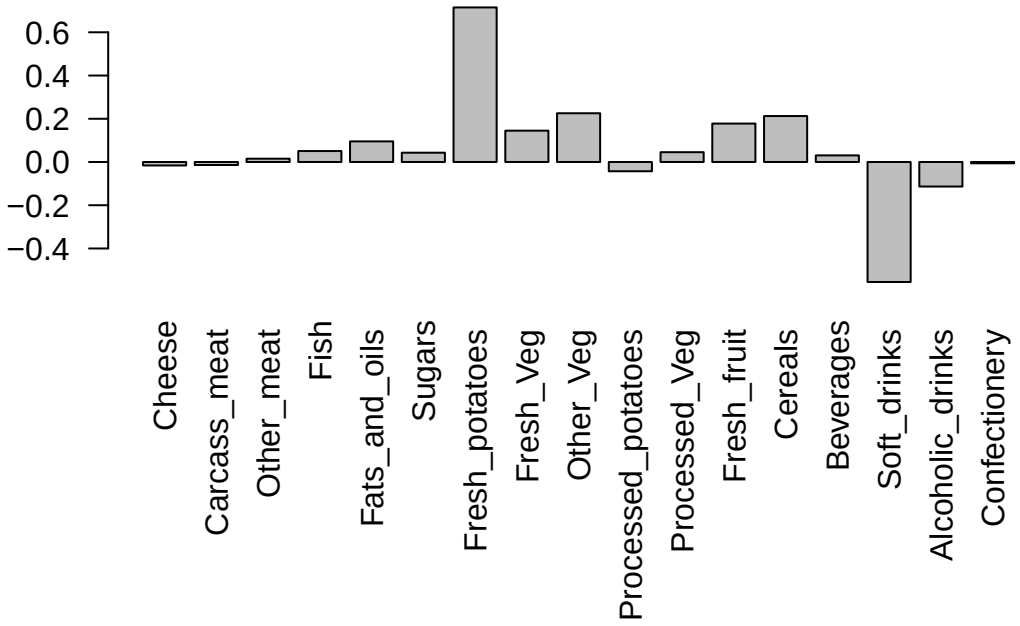
Loadings plot for PC1:

```
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 )
```



Soft drinks and fresh potatoes feature most prominently in the loadings plot for PC2.