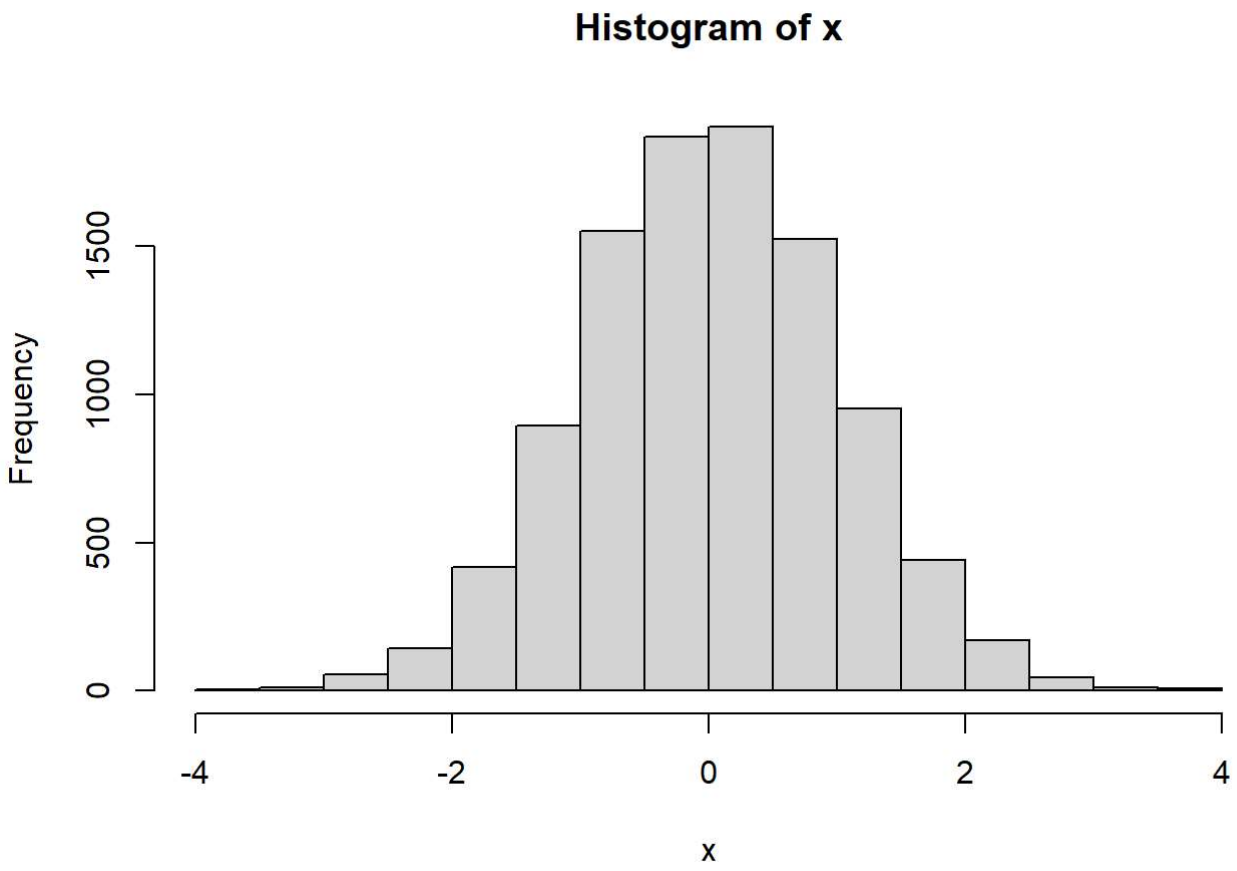
Class 7 machine learning 1

AUTHOR
Joe Kesler

K-means clustering

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

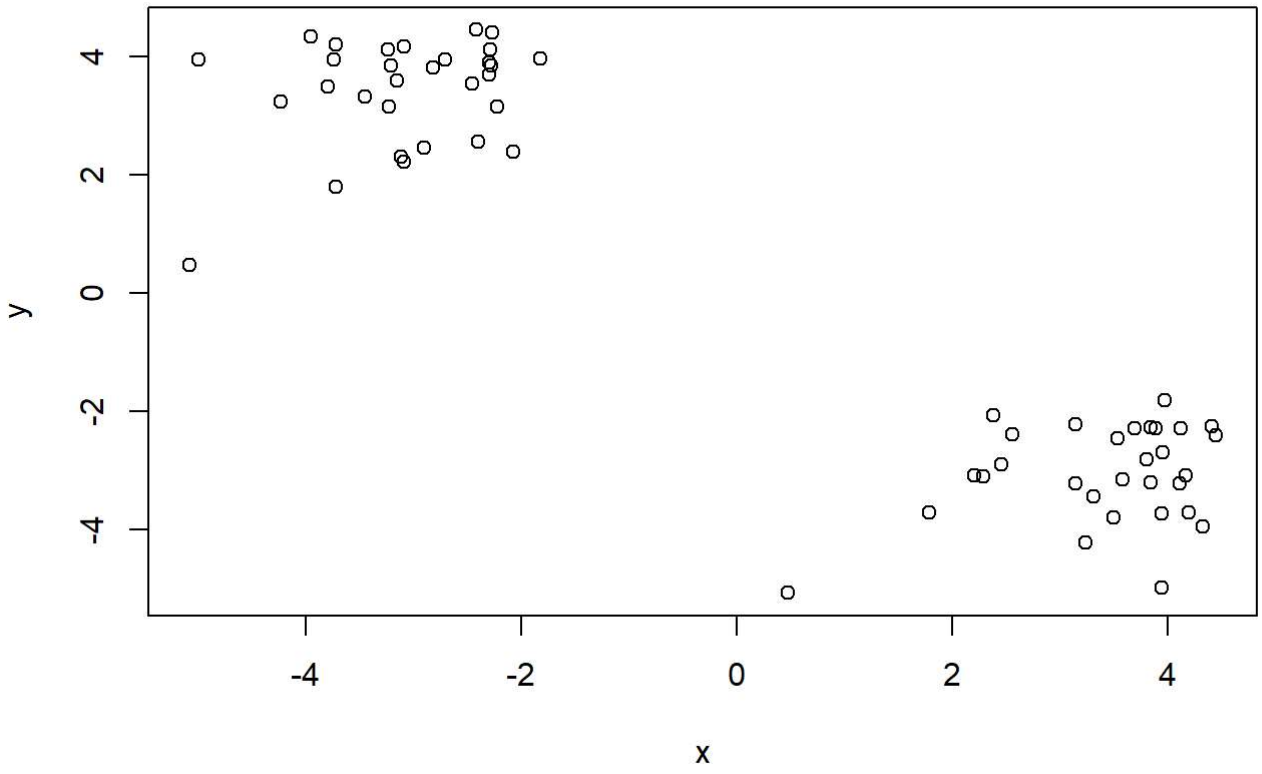
```
x <- rnorm(10000)
hist(x)
```



Let's make some numbers centered on -3 and +3

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

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



Now lets see how kmeans works.

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

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

Cluster means:
          x         y
1  3.406777 -3.067912
2 -3.067912  3.406777

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 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] 43.71661 43.71661
 (between_SS / total_SS =  93.5 %)

Available components:

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

> Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

> Q. What component of your result object details cluster assignment/membership?

```
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 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
```

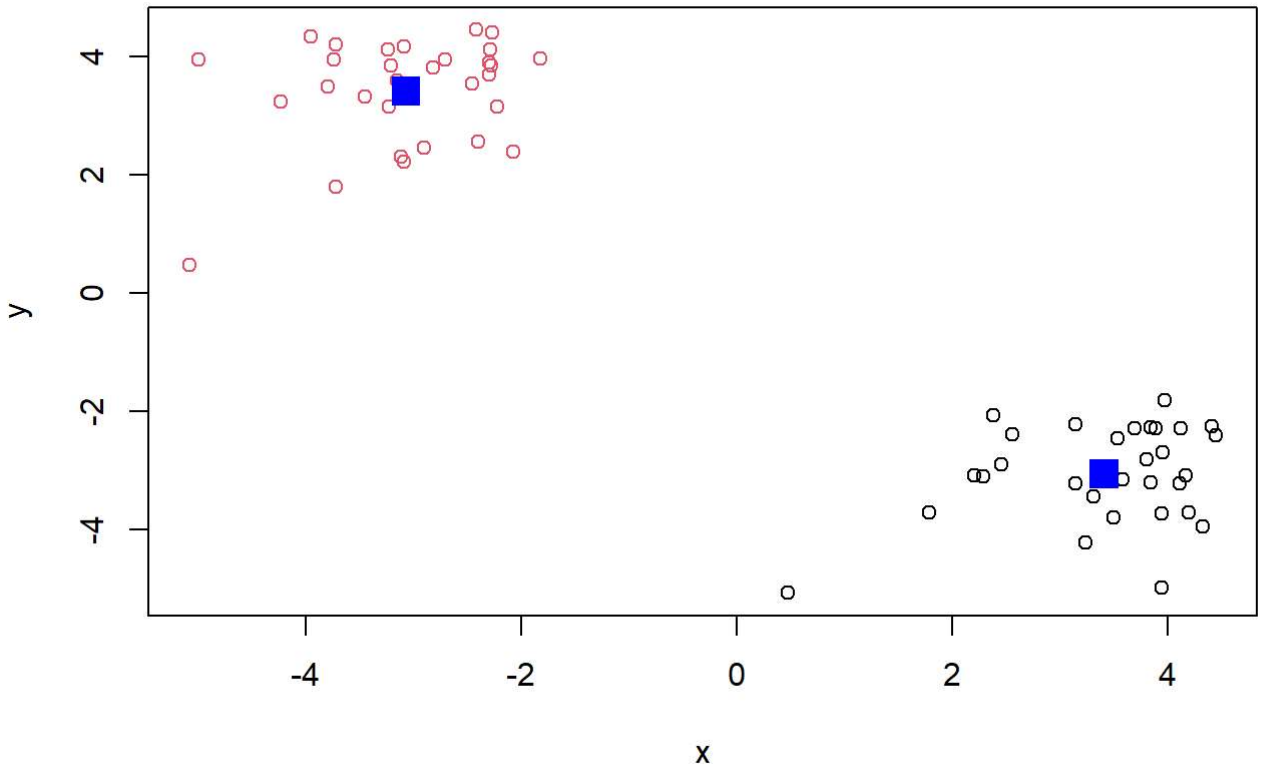
> what about the cluster centers?

```
km$centers
```

```
          x         y
1  3.406777 -3.067912
2 -3.067912  3.406777
```

> 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

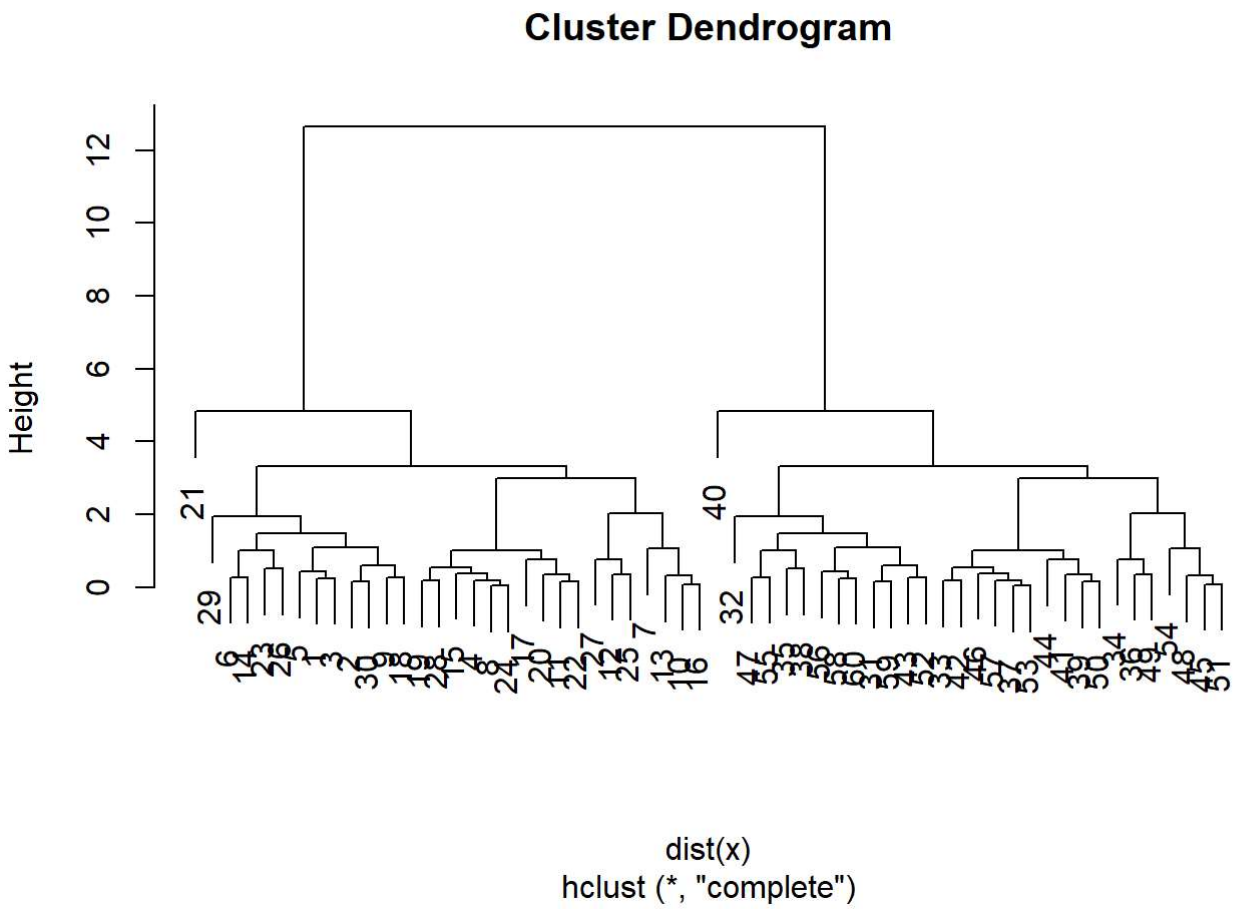
The hclust() function in R performs hierarchical clustering

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

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

There is a plot hc method for hclust objects...

```
plot(hc)
```



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

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

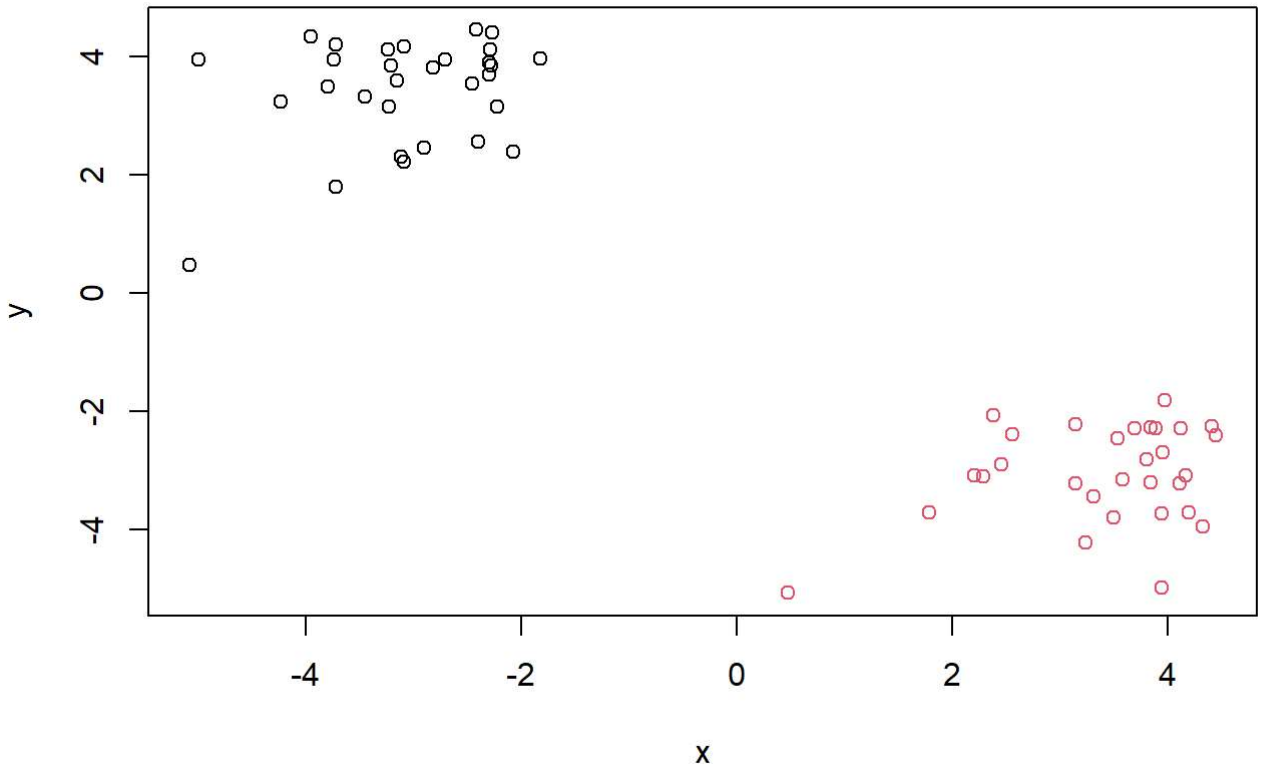
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

use cutree() with k = 2

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

A plot of our data colored by our hclust grps

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



PCA = Principal Component Analysis

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

> Q1. How many rows and columns in your new data frame named x?

```
dim(x)
```

```
[1] 17  5
```

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
```

Uh oh

```
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
```

much better Lets check the dimensions again

```
dim(x)
```

```
[1] 17  4
```

Ok, so there are 17 rows and FOUR columns. Here's another way to do what we just did:

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

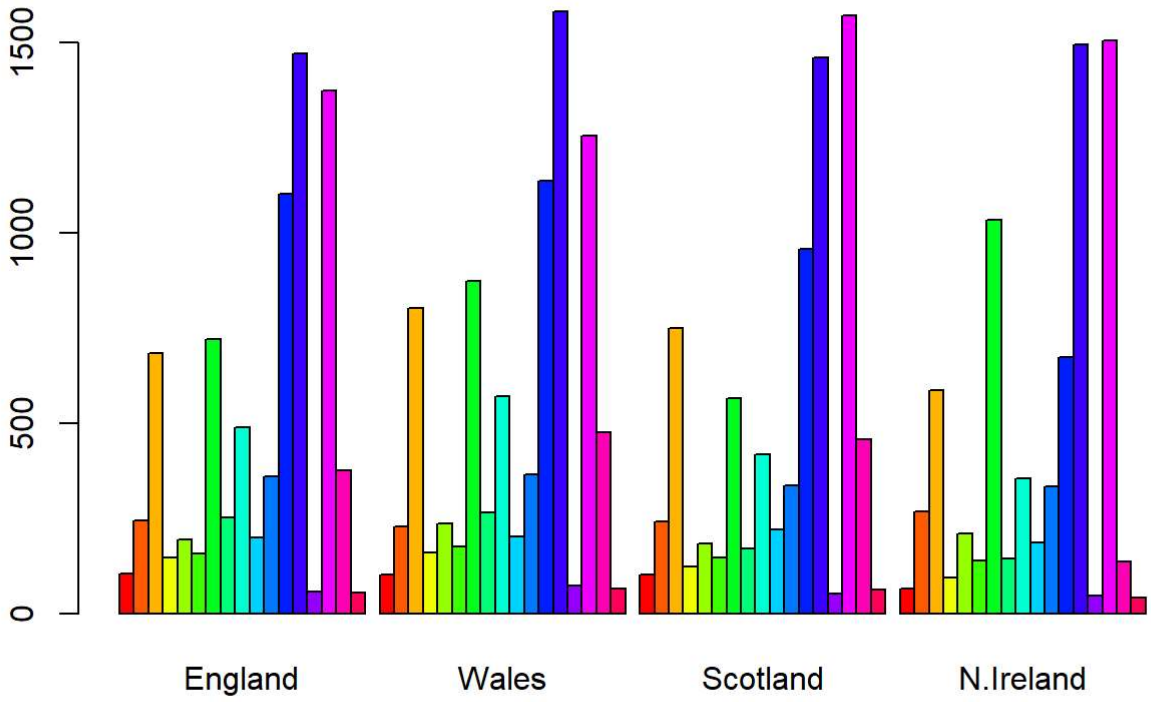
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            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 like the second approach better to solving the rownames problem because it tackles the problem more directly by reassigning the rownames rather than eliminating a column. If you repeatedly run it this first way, it will repeatedly remove columns.

Lets plot our data to analyze trends!

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

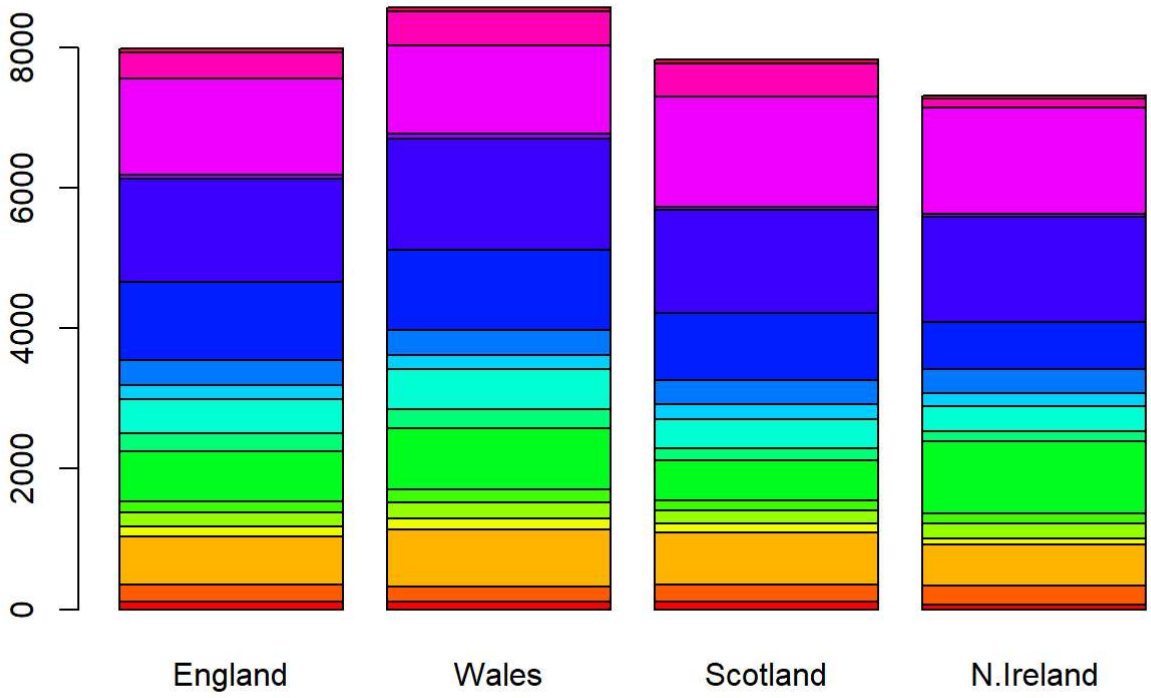


This is hard to get any info from. Lets try to replot better.

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

Changing beside = F will make the following plot:

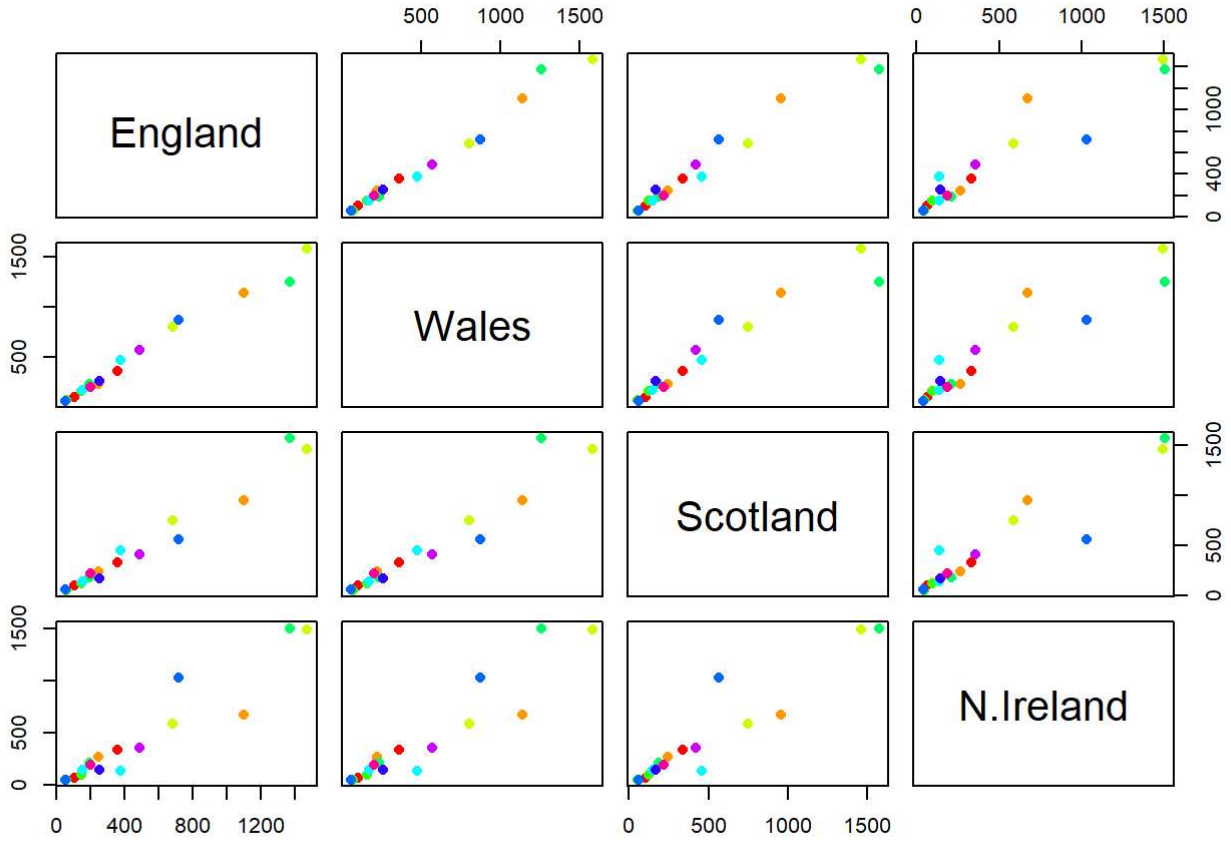
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



It's still unreadable though.

> 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)
```



Ok how do I read this and what is it showing me? Basically it is a plot matrix. All pairs are plotted against each other. It is still quite hard to read.

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

For england and whales, it looks the same as they are all a straight line. Scotland looks pretty similar to england and whales too. But N.Ireland looks super different, it isn't in a straight line at all when compared to the rest. This still isn't that helpful though. How can we make this more interpretable?

PCA to the rescue. 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 results show that PCA captures 67% of the total variance in the original data in one PC and 96.5% in two PC's

96.5% in two PCs.

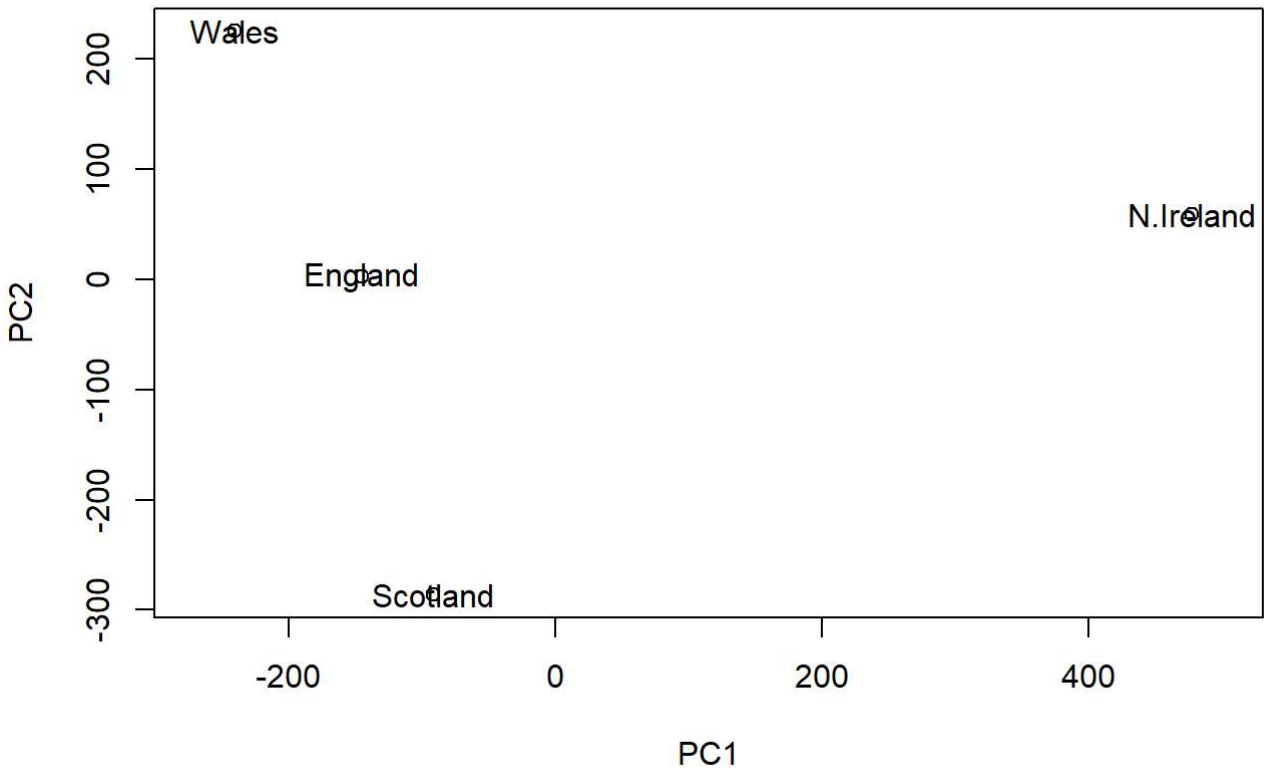
```
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

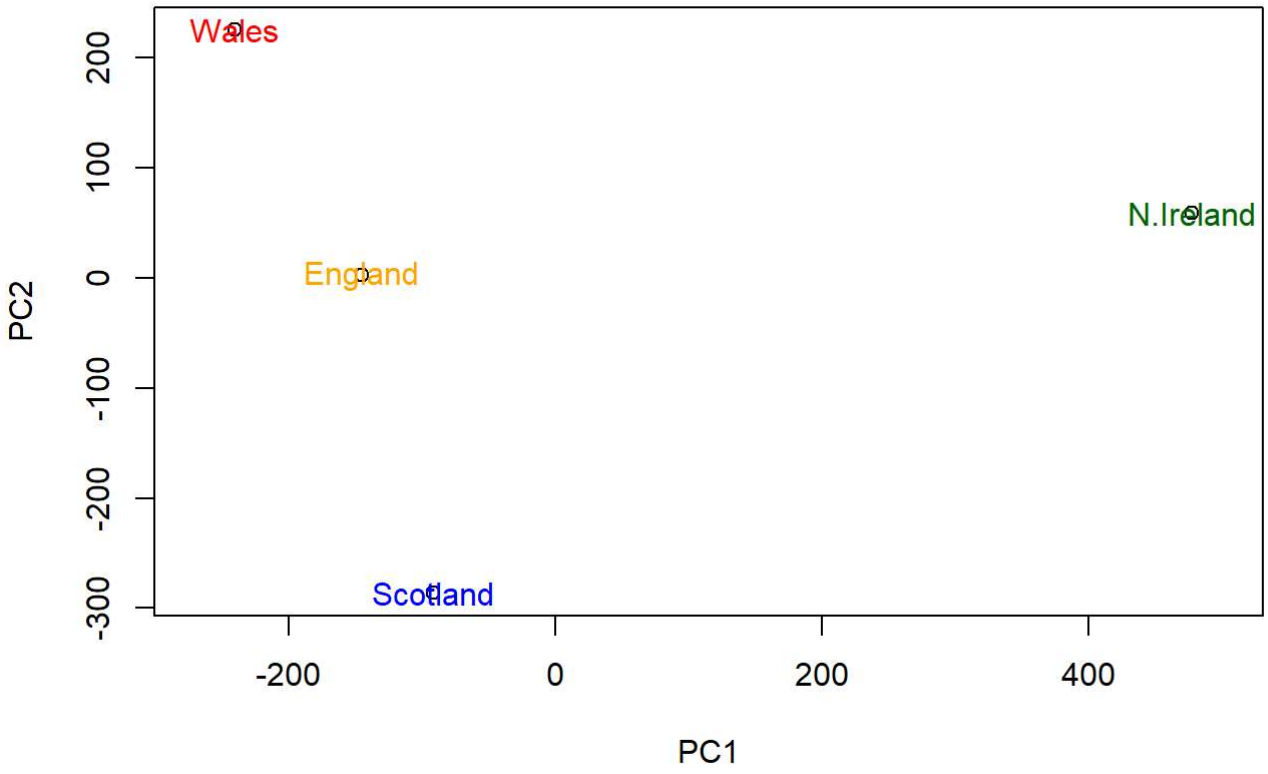
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points

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

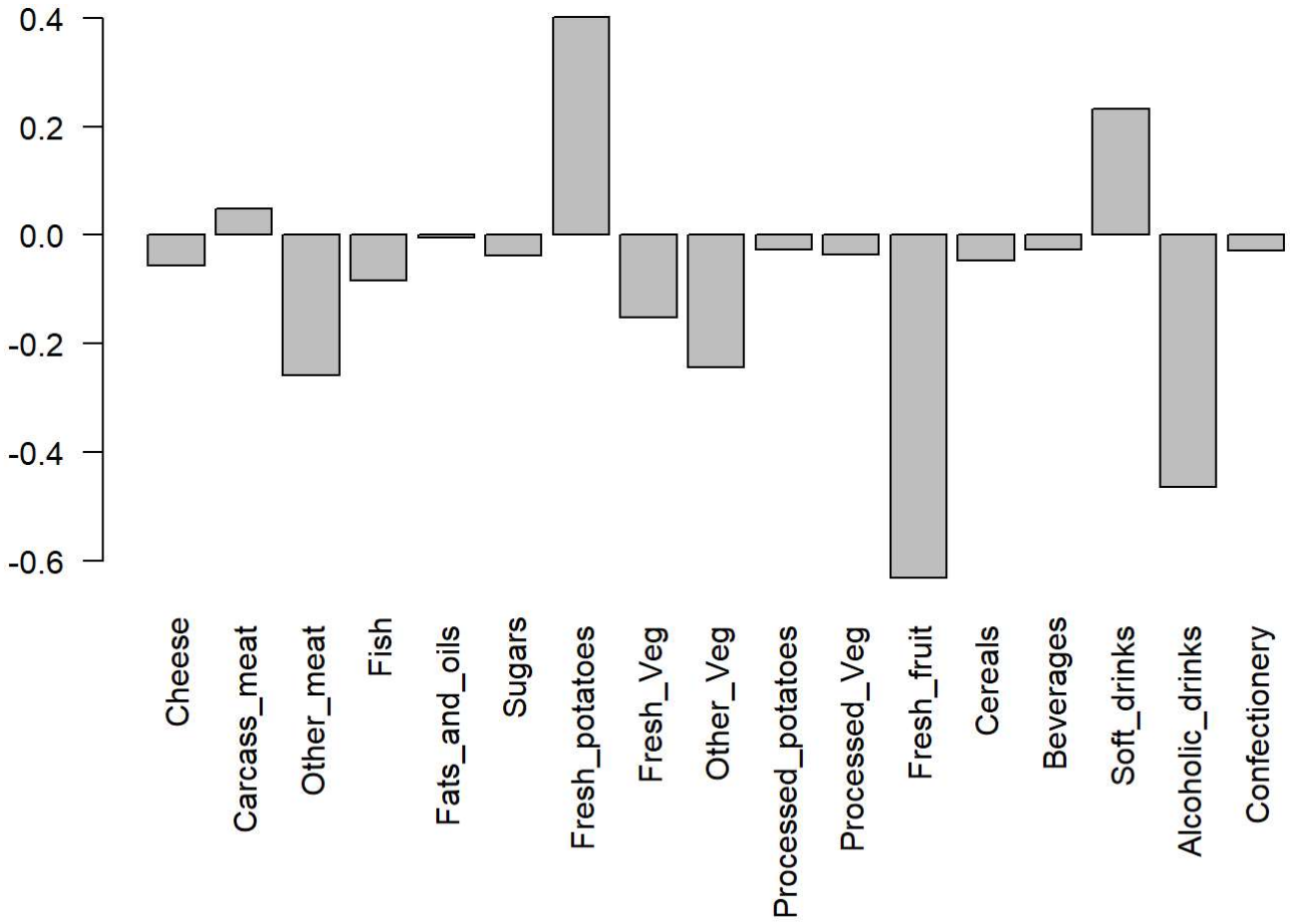
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col=(c("orange","red","blue","darkgreen")))
```



N.Ireland is clearly different from the others.

Digging Deeper

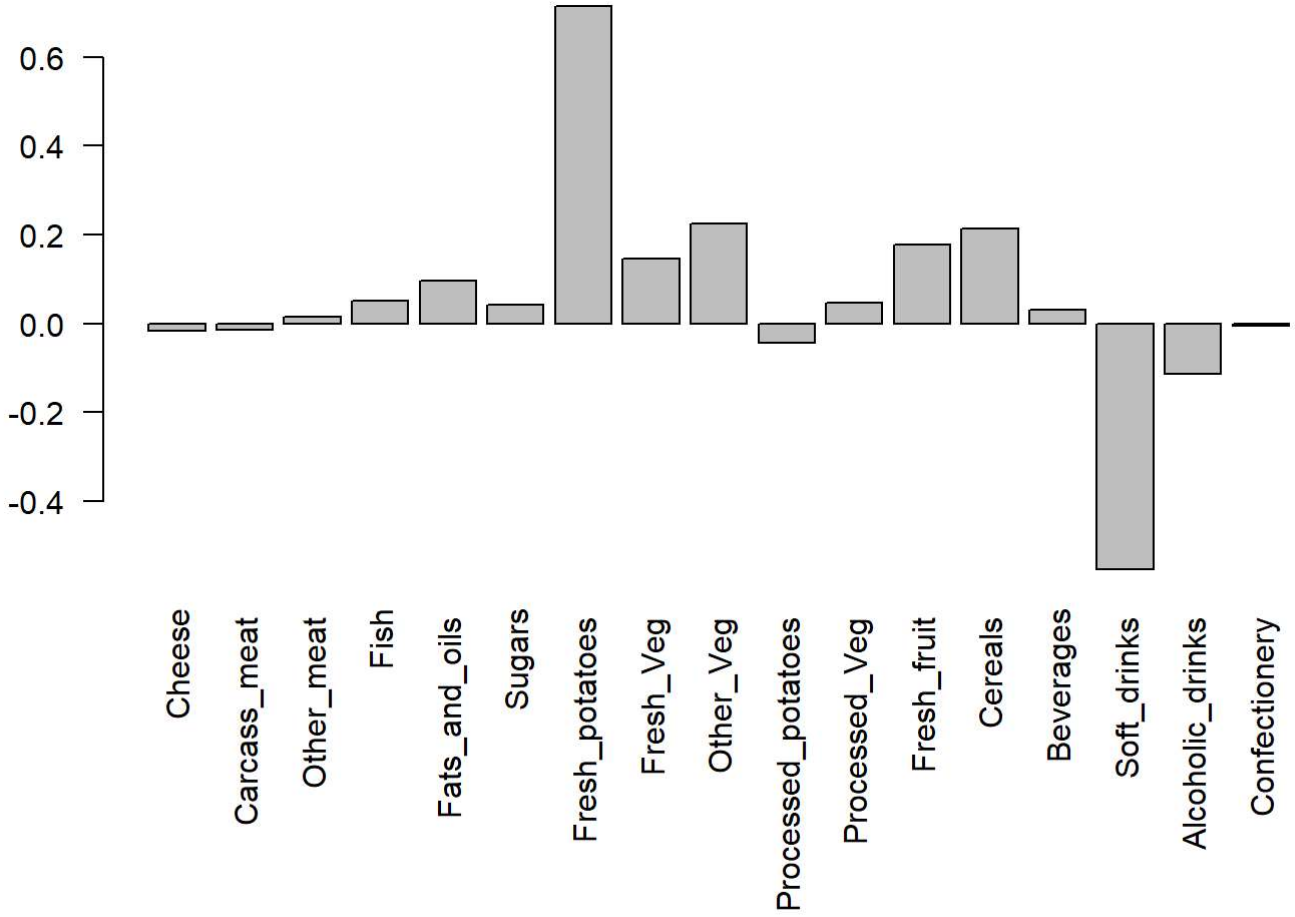
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



Here, we can see what pushed the countries to different sides of the plot. The irish love potatoes but despise alcoholic drinks and fresh fruit.

> 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 )
```



This PC2 plot above shows what pushes the countries on the plot up or down, while the PC1 plot showed us what pushed countries to the left or right.For example, we can say scotland likes soft drinks a lot more than the other countries from this plot.

PCA of RNA seq data

```
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?

```
nrow(rna.data)
```

```
[1] 100
```

```
ncol(rna.data)
```

```
[1] 10
```

There are 100 genes and 10 samples.

```
pca <- prcomp(t(rna.data), scale=TRUE)
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```