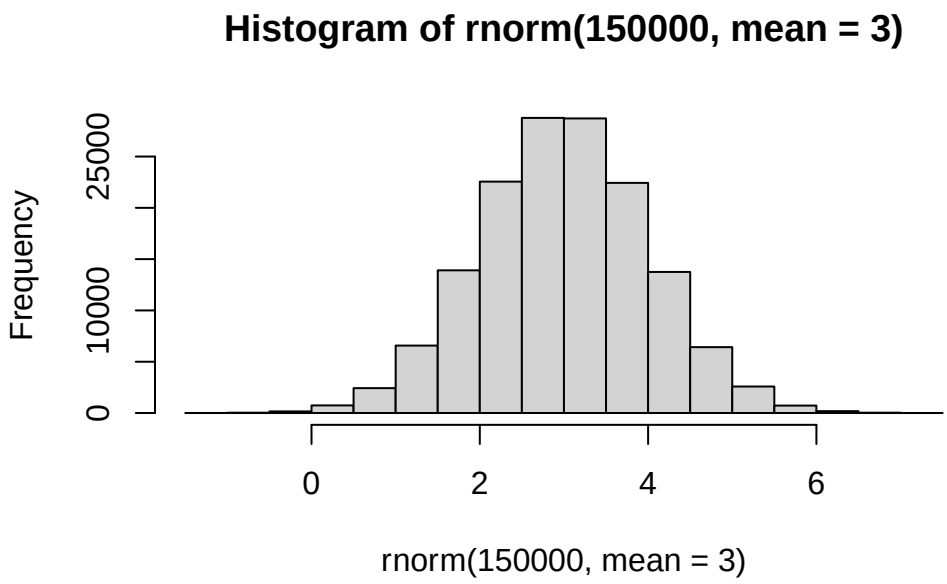
lab7

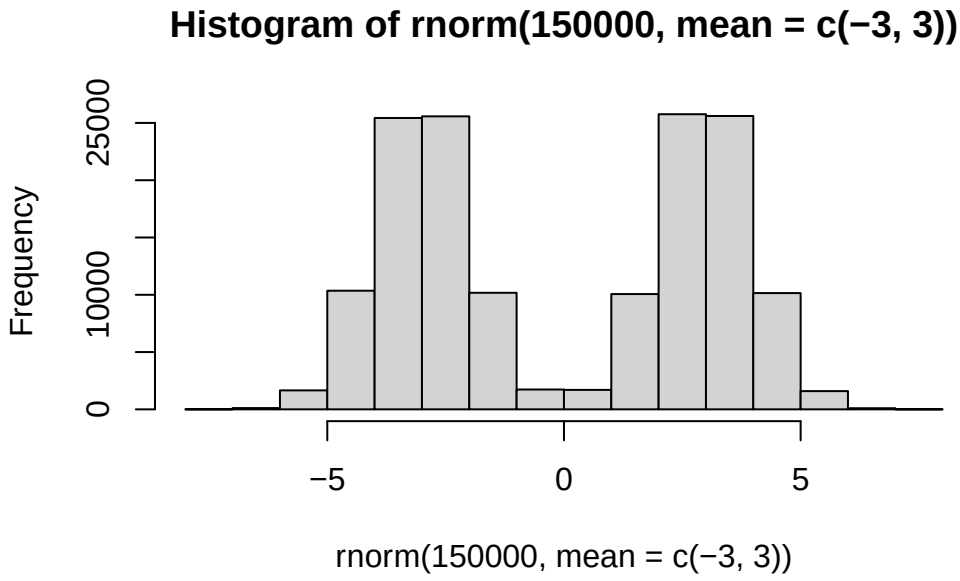
Angelica Rock (PID:15781397)

Before we get into clustering methods let's make some sample data to cluster where we know what the answer should be

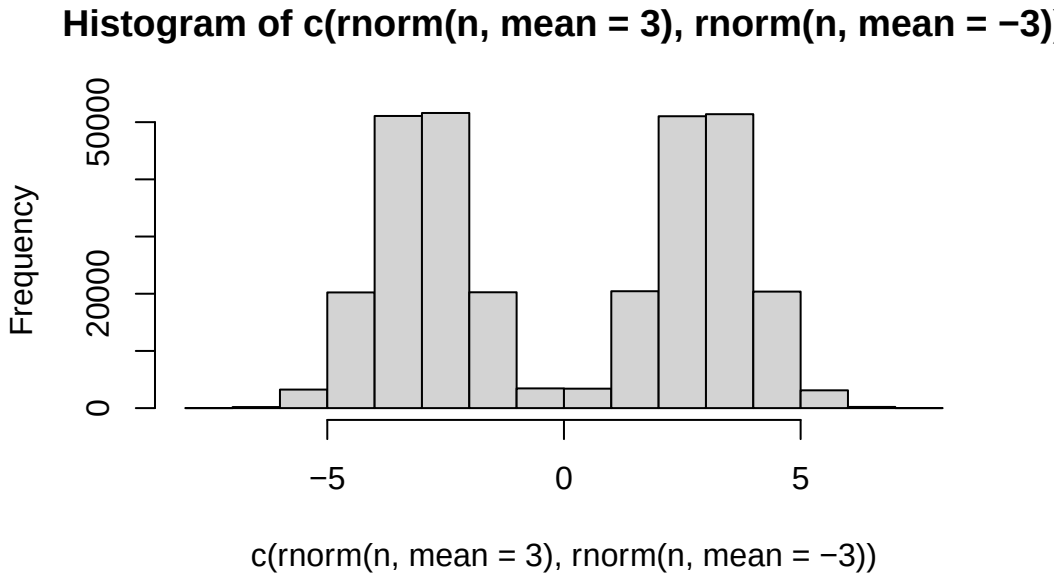
```
hist(rnorm(150000, mean=3))
```




```
hist(rnorm(150000, mean=c(-3,3)))
```




```
#this is the same as
n=150000
hist(c(rnorm(n, mean=3), rnorm(n, mean=-3)))
```



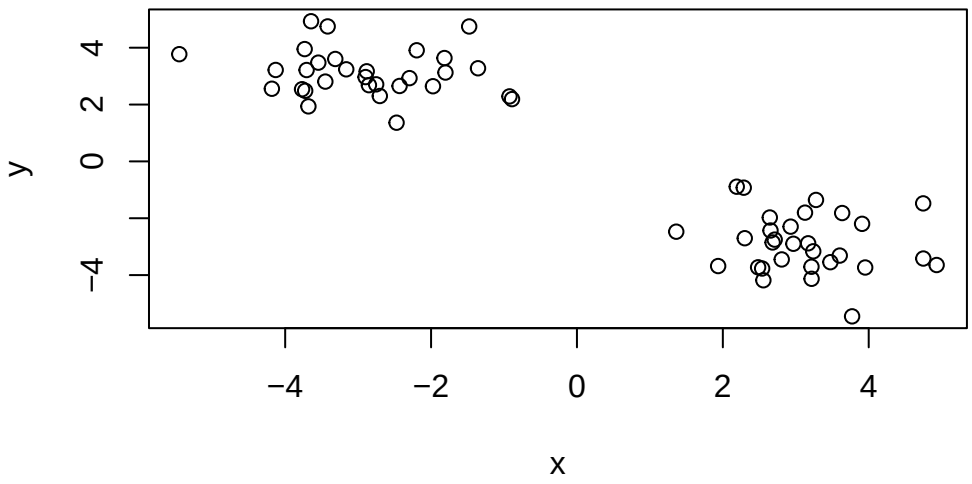
```r
n=30
x <- c(rnorm(n, mean = 3), rnorm(n, mean=-3))
y <- rev(x)

z <- cbind(x,y)
z
```

```
              x          y
 [1,]  3.1697368 -2.8819998
 [2,]  3.6360024 -1.8165045
 [3,]  2.8072143 -3.4494796
 [4,]  2.9294654 -2.2958481
 [5,]  2.6439203 -1.9740658
 [6,]  3.2150437 -3.7066034
 [7,]  3.7721545 -5.4518853
 [8,]  2.1908688 -0.8903190
 [9,]  4.7467848 -1.4774035
[10,]  3.1264297 -1.8028274
[11,]  3.6027403 -3.3132618
[12,]  2.5556648 -4.1825009
[13,]  2.9669174 -2.8980017
[14,]  2.3004609 -2.7026560
[15,]  3.2770414 -1.3564144
[16,]  3.4743426 -3.5456792
[17,]  2.2868375 -0.9250961
[18,]  4.9310601 -3.6454606
[19,]  3.2168794 -4.1303874
[20,]  2.5389624 -3.7688583
[21,]  2.7108792 -2.7527014
[22,]  3.9516972 -3.7329904
[23,]  2.4836406 -3.7259200
[24,]  2.6534309 -2.4317775
[25,]  3.9093810 -2.1969386
[26,]  3.2394534 -3.1630513
[27,]  1.3614104 -2.4730276
[28,]  1.9358524 -3.6816248
[29,]  2.6822756 -2.8519939
[30,]  4.7485767 -3.4175996
[31,] -3.4175996  4.7485767
[32,] -2.8519939  2.6822756
[33,] -3.6816248  1.9358524
[34,] -2.4730276  1.3614104
```

```
[35,] -3.1630513  3.2394534
[36,] -2.1969386  3.9093810
[37,] -2.4317775  2.6534309
[38,] -3.7259200  2.4836406
[39,] -3.7329904  3.9516972
[40,] -2.7527014  2.7108792
[41,] -3.7688583  2.5389624
[42,] -4.1303874  3.2168794
[43,] -3.6454606  4.9310601
[44,] -0.9250961  2.2868375
[45,] -3.5456792  3.4743426
[46,] -1.3564144  3.2770414
[47,] -2.7026560  2.3004609
[48,] -2.8980017  2.9669174
[49,] -4.1825009  2.5556648
[50,] -3.3132618  3.6027403
[51,] -1.8028274  3.1264297
[52,] -1.4774035  4.7467848
[53,] -0.8903190  2.1908688
[54,] -5.4518853  3.7721545
[55,] -3.7066034  3.2150437
[56,] -1.9740658  2.6439203
[57,] -2.2958481  2.9294654
[58,] -3.4494796  2.8072143
[59,] -1.8165045  3.6360024
[60,] -2.8819998  3.1697368
```

```
plot(z)
```



K-means clustering

The function in base R for k-means clustering is called `kmeans()`.

```
km <- kmeans(z, centers = 2)
```

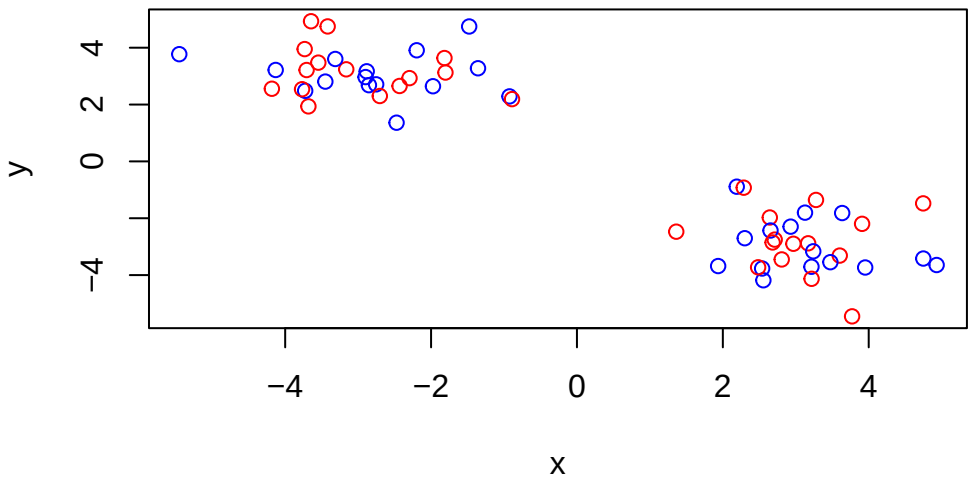
Q. Print out the cluster membership vector (i.e. our main answer)

```
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
```

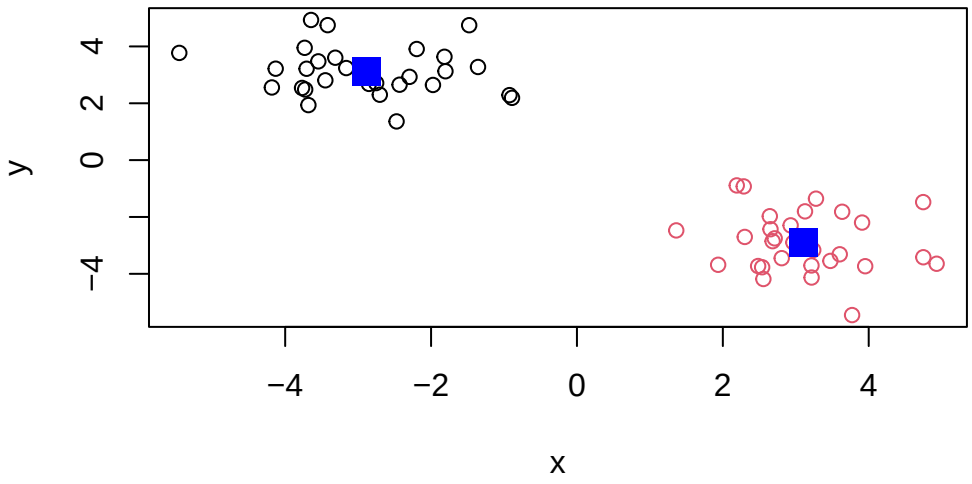
Plot z with the cluster

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



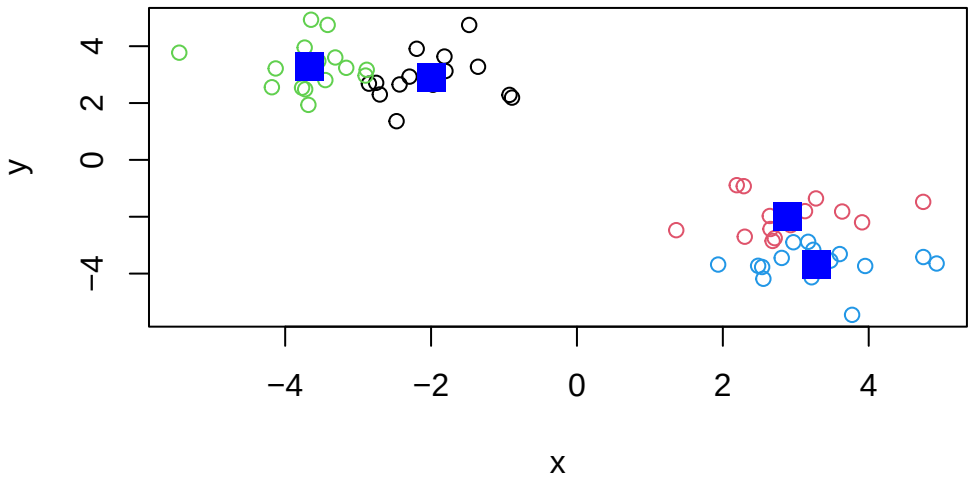
Plot with clustering results

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



Q. Can you cluster our data in `z` into four clusters?

```
km4 <- kmeans(z, centers = 4)
plot(z, col=km4$cluster)
points(km4$centers, col="blue", pch = 15, cex =2)
```



Hierarchical Clustering

The main function for hierarchical clustering in base R is called `hclust()`

Unlike `kmeans()` I cannot just pass in my data as input. I first need a distance matrix from my data

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

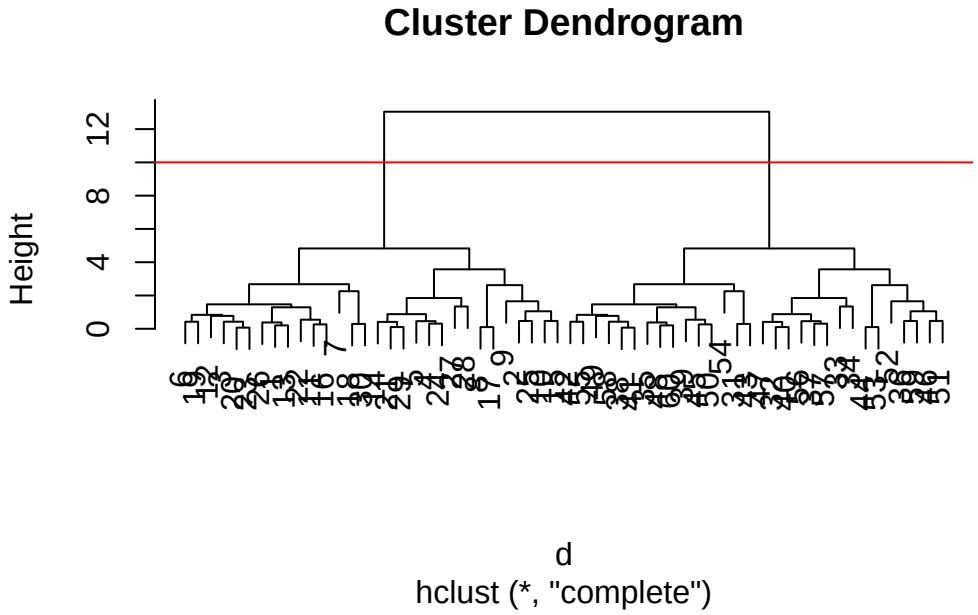
```
Call:
hclust(d = d)

Cluster method   : complete
```

```
Distance         : euclidean
Number of objects: 60
```

There is a specific hclust plot() method...

```
plot(hc)
abline(h=10, col="red")
```

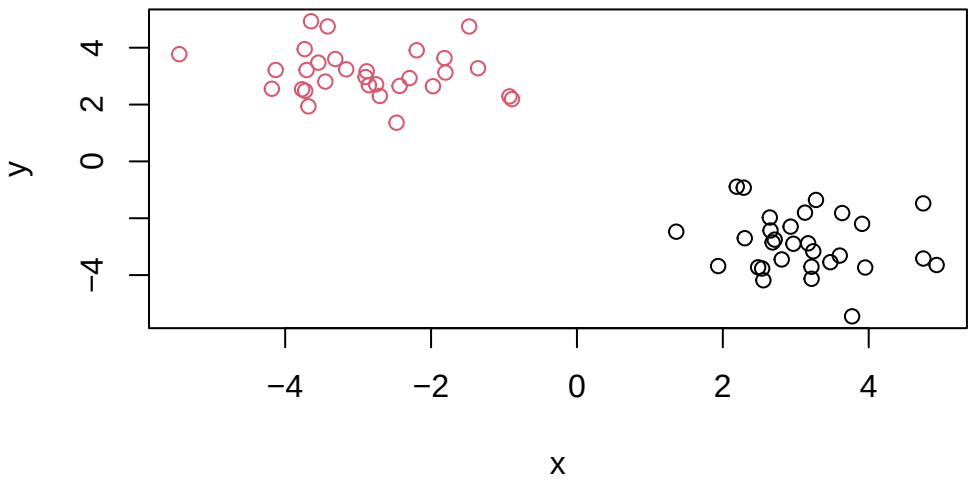


To get my main clustering result (i.e. the membership vector) I can "cut" my tree at a given height. To do this I will use the `cutree()`

```
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(z, col=grps)
```



Principal Component Analysis

Principal component analysis (PCA) is a well established "multivariate statistical technique" used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong paterns and relationships in large datasets (i.e. revealing major similarities and diferences) that are otherwise hard to visualize.

PCA of UK food data

Data import

```
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?

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

```
# use you can use
#dim(x)
```

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
```

First approach: We don't want to have row names set as the first column of the x data frame, only the 4 countries, so we can change this using the function below: # Note how the minus indexing works and check the dimensions

```
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)
```

Second approach: Instead of using the above code, you can instead use this to change the row names from being set as the first column of the x data frame to only having the four countries as row names

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

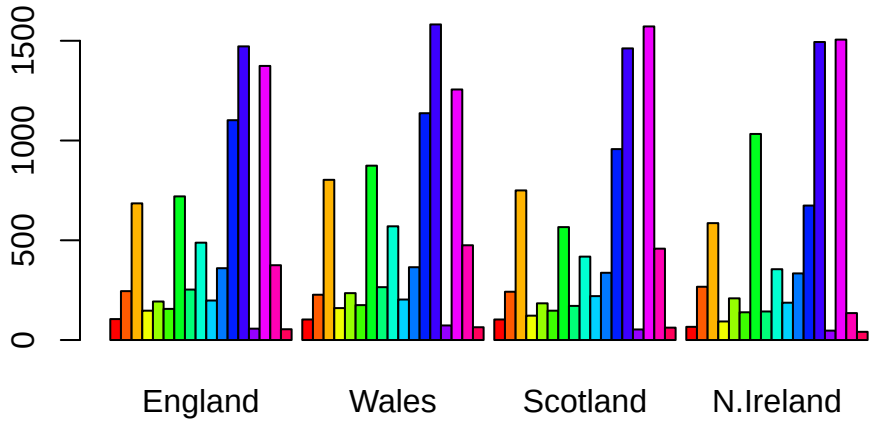
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             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 to have the second approach because if you were to rerun the first approach block multiple times, it will keep cutting down the row name of the one in the first place and the number of countries will shrink from 3 to 2 and so on

Generating a bar plot from the data

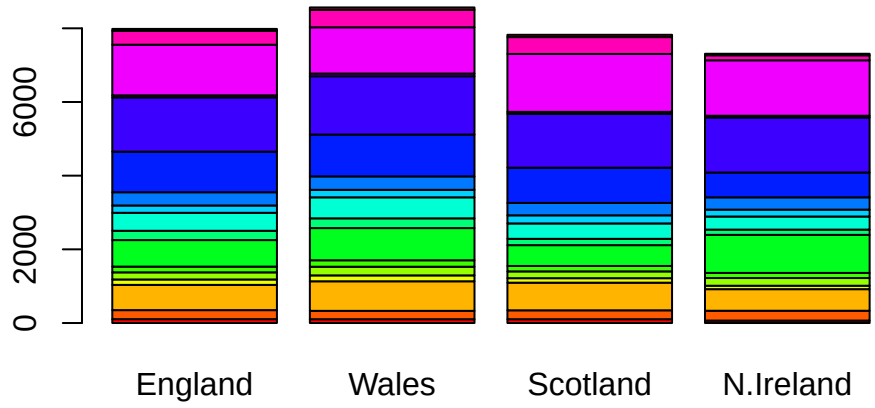
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



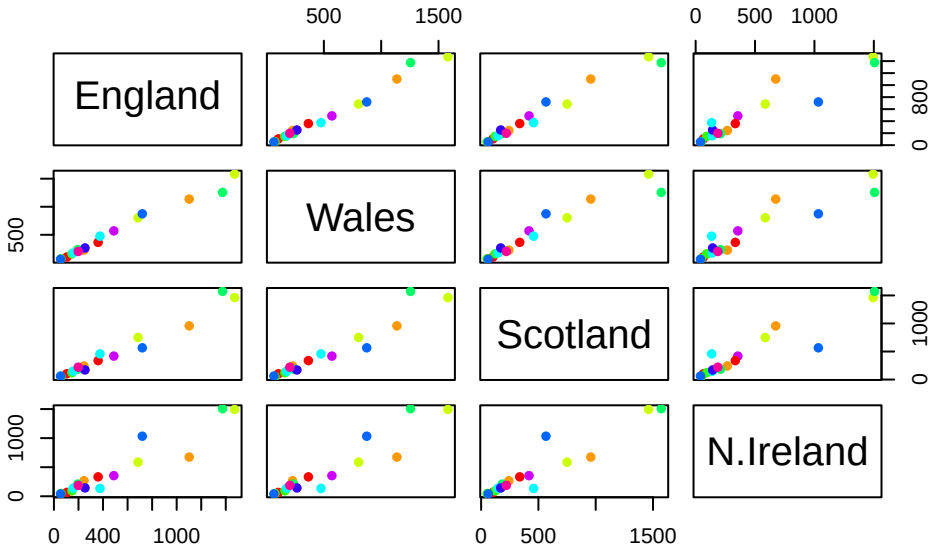
Q3: Changing what optional argument in the above barplot() function results in the following plot?

beside=FALSE leads to the plot below because changing beside to false makes the columns of height portrayed as stacked bars, whereas with true, the columns are portrayed as juxtaposed bars

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




```
pairs(x, col=rainbow(10), pch=16)
```



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?

This code and resulting figure is showing a matrix of scatterplots where you can compare the individual food types with another country. Each point represents a different food type. I think this means that the dots on the diagonal are more strongly correlated with the other country whereas the dots away from the diagonal are less strongly correlated with the other country.

PCA to the rescue

The main function to do PCA in base R is called `prcomp()`

Note that I need to take the transpose of this particular data as that is what the `prcomp()` help page was asking for

```
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
```

Let's see what is inside our result object "pca" that we just calculated:

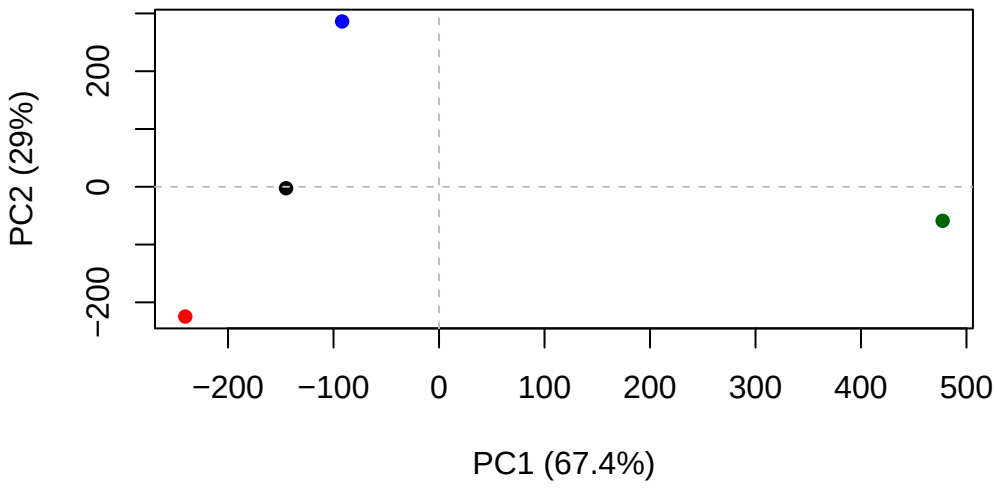
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

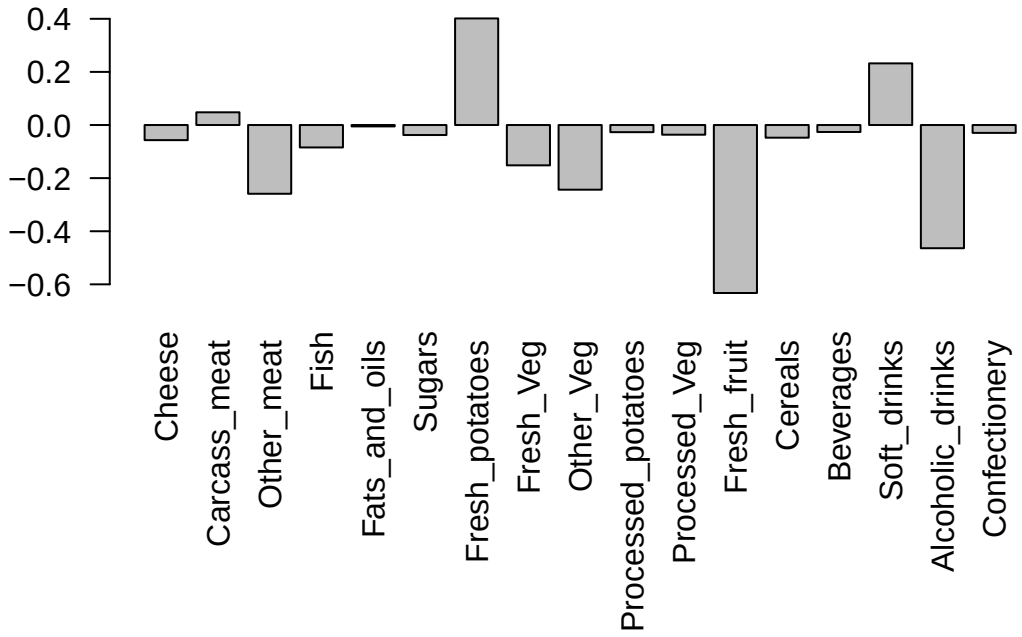
To make our main result figure, called a "PC plot" (or "score plot", "ordination plot", or "PC1 vs PC2 plot")

```r
plot(pca$x[,1], pca$x[,2],
  col=c("black", "red", "blue", "darkgreen"),
  pch=16, xlab="PC1 (67.4%)", ylab="PC2 (29%)")
abline(h=0, col="grey", lty="dashed")
abline(v=0, col="grey", lty="dashed")
```



Variable Loadings Plot

```r
## 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 )
```


0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery