class 7: Machine learning 1

Bayah Essayem (A17303992)

2025-01-28

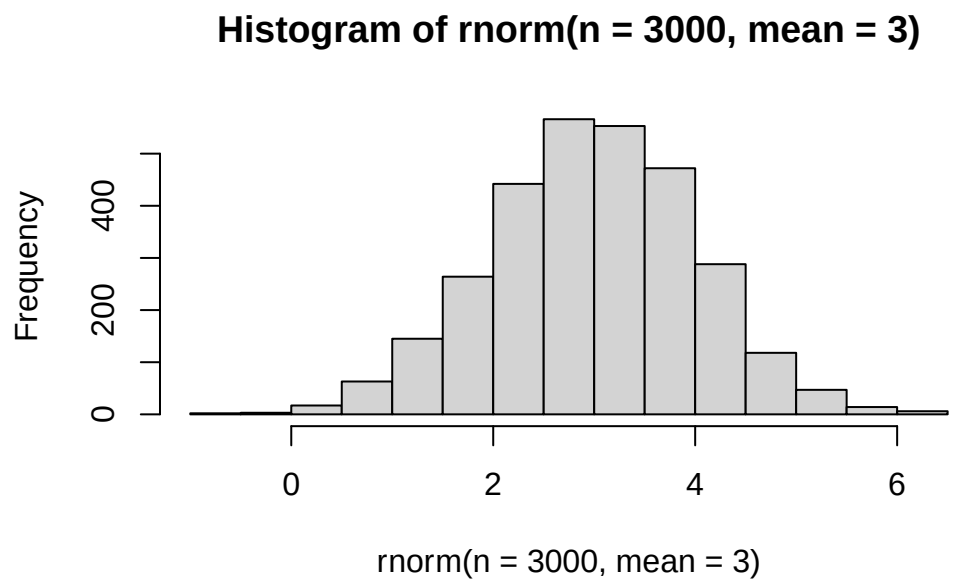
Table of contents

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) thay we can use to test out different clustering methods.

We can use the 'rnorm()' function to help us here:

```
hist(rnorm(n=3000, mean=3))
```

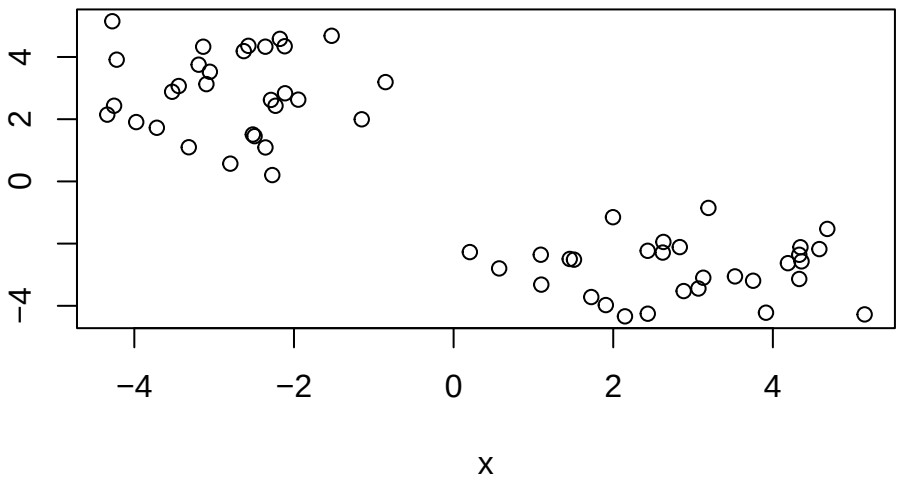


Make data with two "clusters"

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

       rnorm (30, mean = +3))

z <- cbind(x, rev(x))

head(z)
```

```
             x
[1,] -3.718059 1.724816
[2,] -2.628548 4.188661
[3,] -2.358557 4.331065
[4,] -3.097194 3.127760
[5,] -2.571010 4.358998
[6,] -4.338894 2.147087
```

```
plot(z)
```



K-means clustering

The main function in "base" R for K-means clustering is called 'kmeans()'

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

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

Cluster means:
          x
1  2.869271 -2.798360
2 -2.798360  2.869271

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] 75.1339 75.1339
 (between_SS / total_SS =  86.5 %)
```

```
Available components:

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

How big is z?

```
nrow(z) #60
```

```
[1] 60
```

```
ncol(z) #2
```

```
[1] 2
```

```
attributes(k)
```

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

$class
[1] "kmeans"
```

Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What component of our results tell us about the cluster membership (i.e. which point likes in which cluster) ?

```
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
```

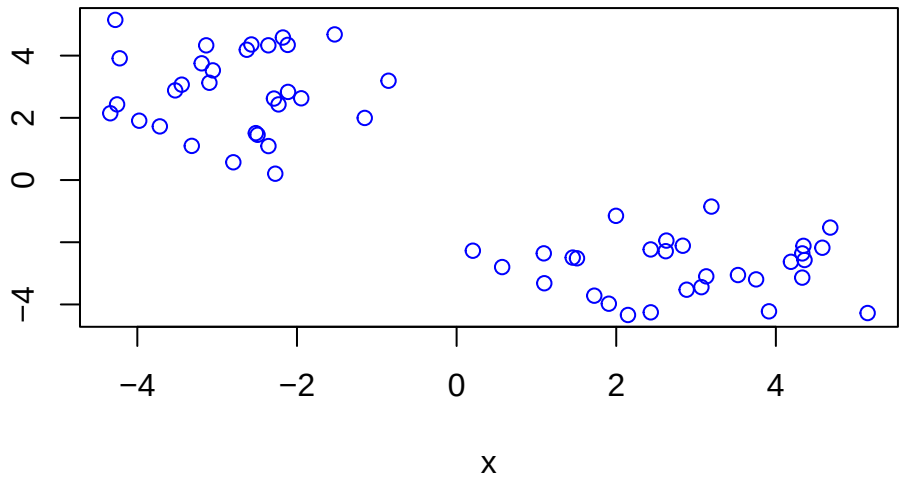
Q. Center of each cluster?

```
k$centers
```

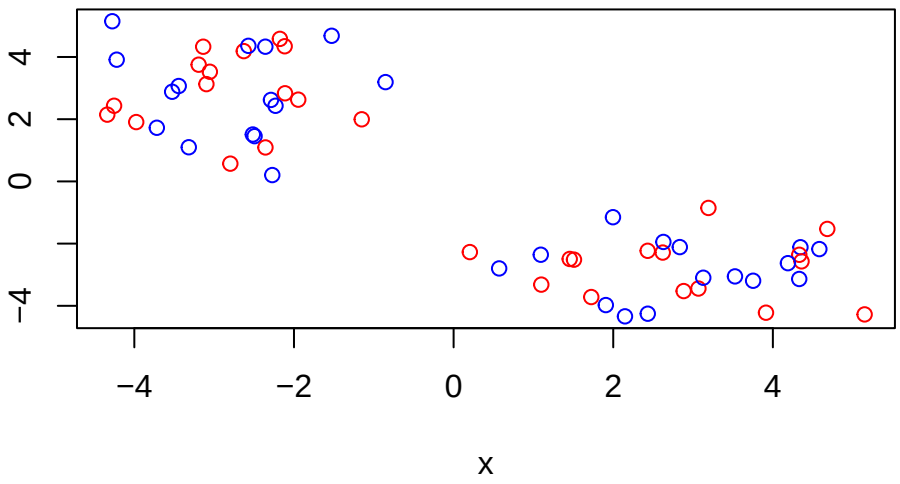
```
          x
1  2.869271 -2.798360
2 -2.798360  2.869271
```

Q. Put this result info together and make a little "base R" plot of our clustering result. Also add the cluster center points to this plot.

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

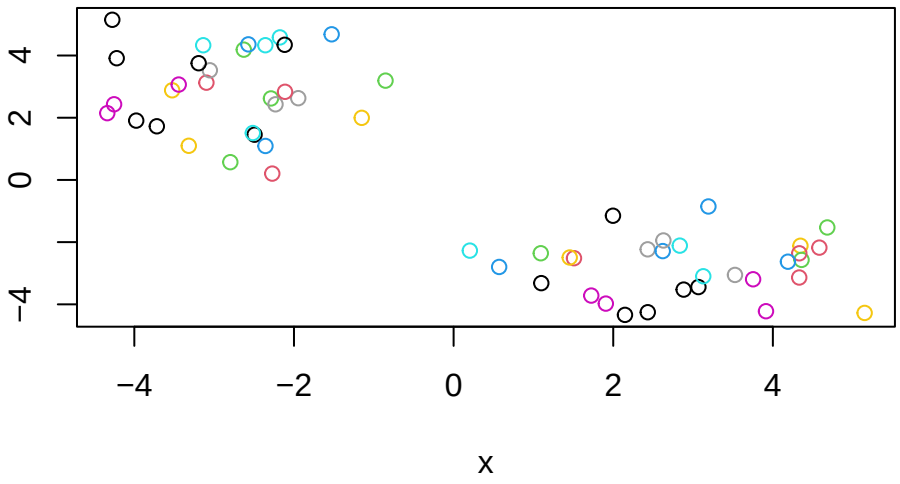



```
plot(z, col = c("blue", "red") )
```



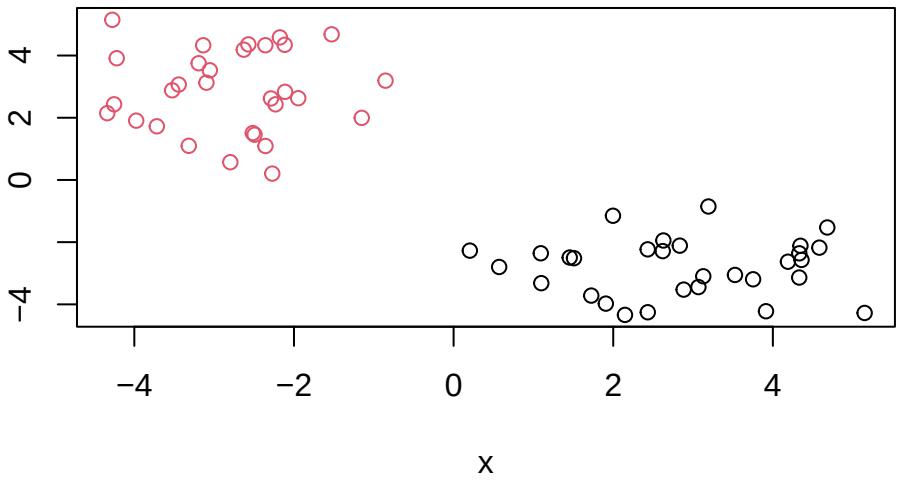
You can color by number, each color corresponds with a number, like 7 = yellow.

```
plot(z, col = c(1,3,5,2,4,6,7,8,9))
```



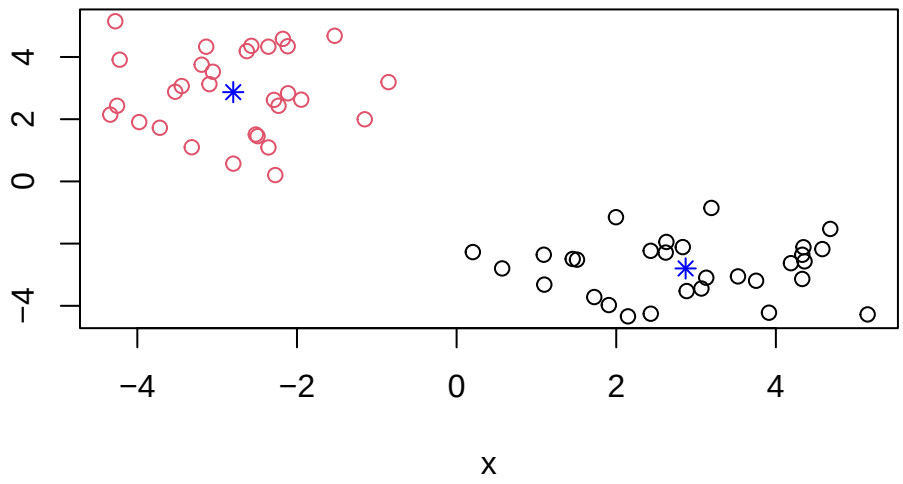
Plot colored by cluster membership:

```
plot(z, col = k$cluster)
```



Now, we must add points for the cluster centers

```
plot(z, col=k$cluster)
points(k$centers, col="blue", pch = 8)
```



Q. Run kmeans on our input (z) and define 4 clusters making the same result visualization plot as above (plot of z colored by cluster membership).

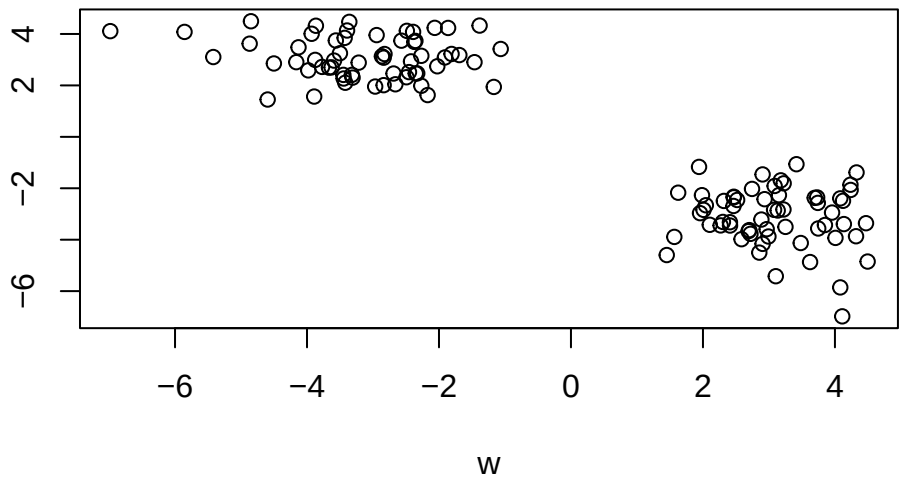
```
w <- c( rnorm (30, mean = -3), rnorm (30, mean = +3), rnorm (30, mean = -3), rnorm (30, mean
```

```
r <- cbind(w, rev(w))
```

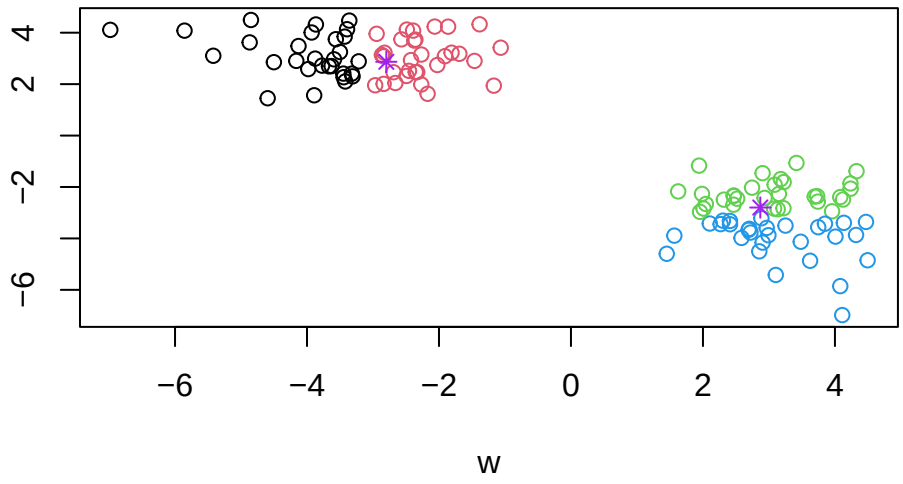
```
head(r)
```

```
             w
[1,] -2.352427 2.466369
[2,] -2.967250 1.955713
[3,] -3.771594 2.720052
[4,] -5.420164 3.103842
[5,] -2.357473 3.730654
[6,] -4.596165 1.451304
```

```
plot(r)
```




```
new_plot <- kmeans(r, centers = 4)
plot(r, col = new_plot$cluster)
points(k$centers, col = "purple", pch = 8)
```



Hierachical Clustering

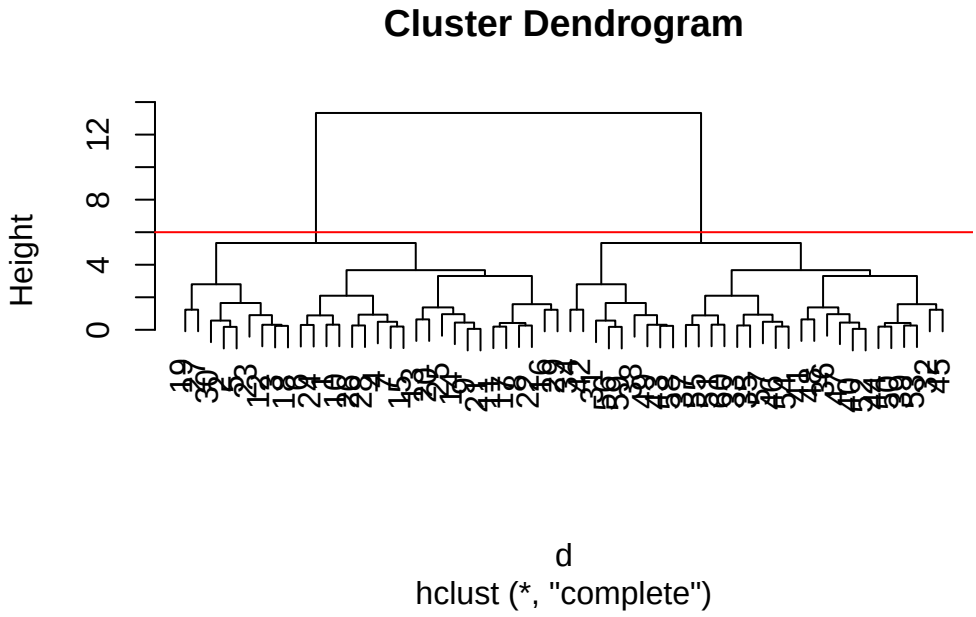
Main function in base R for this is called 'hclust()' it will take as input a distance matrix (key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data).

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

```
Call:
hclust(d = d)

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

This plot separates 1-30 and 31-60 in two clusters

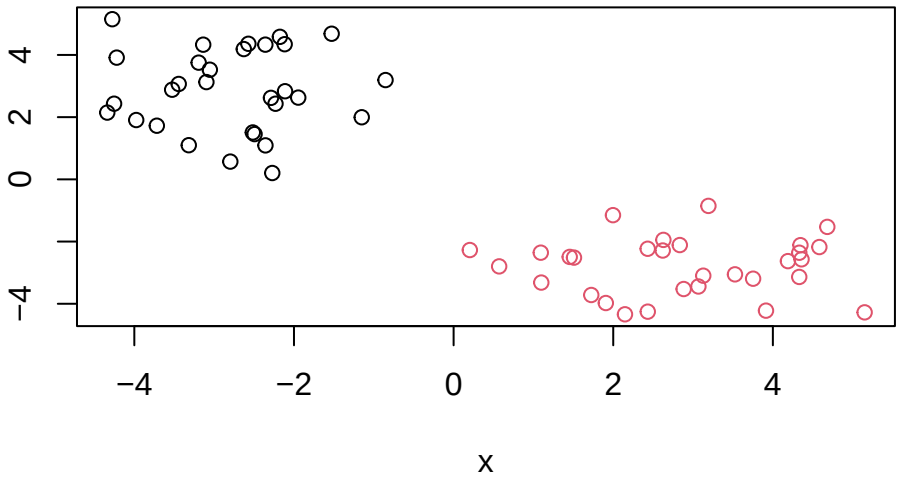
```
plot(hc)
abline(h=6, col="red")
```



Once I inspect the dendrogram (clustering tree), I can "cut" the tree to yield my groupings or clusters. The function to do this is called 'cutree()'

```
cut <- cutree(hc, h = 6)
```

```
plot(z, col=cut)
```



Hands on with Principal Component Analysis (PCA)

Let's examine some silly 17-dimensional data detailing food consumption in the UK (England, Scotland, Wales, and N. Ireland). Are these countries eating habits different or similar and if so, how?

Data import

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
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
Fresh_potatoes          720   874      566      1033
Fresh_Veg               253   265      171       143
Other_Veg               488   570      418       355
Processed_potatoes      198   203      220       187
Processed_Veg           360   365      337       334
Fresh_fruit            1102  1137      957       674
Cereals                1472  1582     1462      1494
Beverages                57    73       53        47
Soft_drinks            1374  1256     1572      1506
Alcoholic_drinks        375   475      458       135
Confectionery            54    64       62        41
```

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

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 4
```

```
dim(x)
```

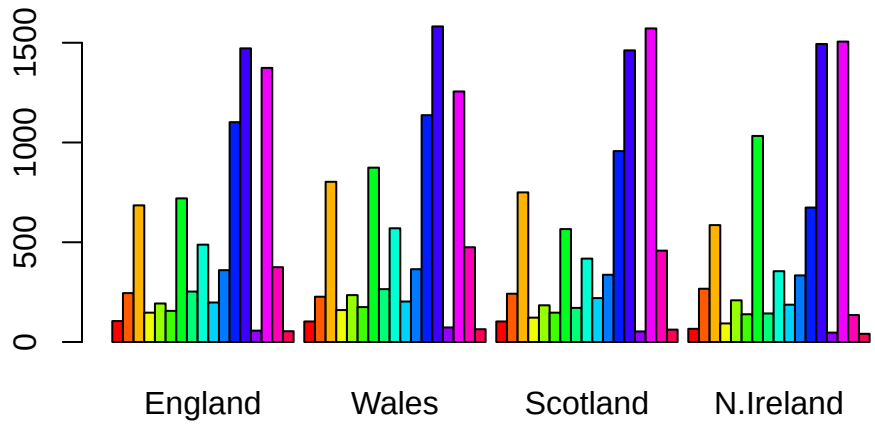
```
[1] 17  4
```

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? A2. I prefer the approach where the file is read and the 'row.names()' argument is used to set the first column to read as rows. This is because of all the methods, it seems as the most straightforward and creates the most minimal code that is able to accomplish the same goals as other approaches. This approach is more robust since the approach of manually setting the row names can cause

pitfalls related to missing values, duplicates, and unintended type conversions. It requires extra validation to ensure row names are unique and properly formatted before assignment.

The food category is defined by colors, as a bar plot

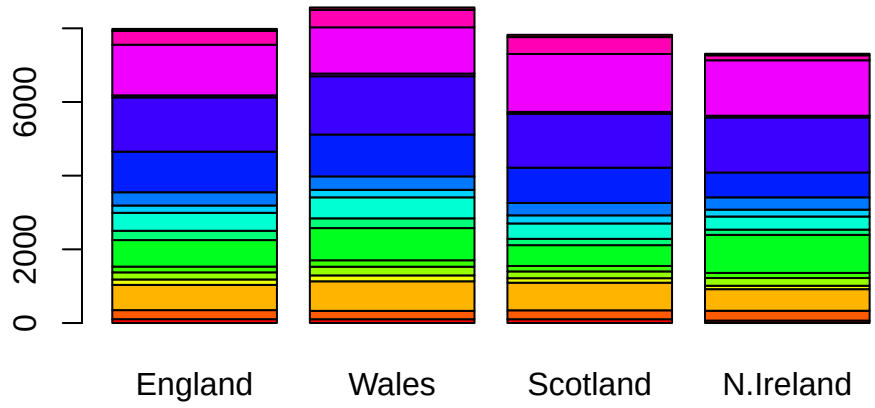
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot? A3. By changing 'beside=' into False rather than True, the bars will stack rather than be placed side by side.

Now, if you made it stacked it makes an even worse graph (done by changing beside to F):

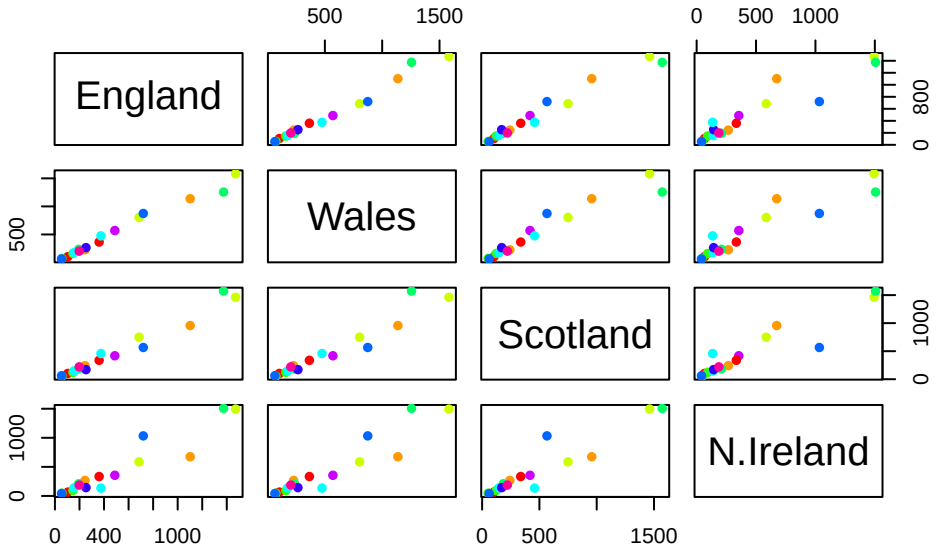
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



Q. 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?

A5. This graph shows a graph comparing all UK countries, so for the first plot on the top left, it shows England on y axis and wales on x axis, then for the graph right next to it, it's England (y) and Scotland (x). When there's points plotted outside of the diagonal it'll be because one nation consumes more of a certain item than the other, on the graph that shows England and N. Ireland there is a blue point deviating showing Ireland consumes more of whatever the blue point represents.

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



Looking at these types of "pairwise plots" can be helpful but it does not scale well and kind of sucks! Basically, there's probably a better way to plot this info

PCA to the rescue!

The main function for PCA in base R is called 'prcomp()'. This function wants the transpose of our input data - i.e. the important food categories as columns and countries as rows.

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? A6. The main difference between N. Ireland and the other countries of the UK lies in PC1, which captures 67.4% of the total variance. N. Ireland's significantly higher PC1 value indicates that it differs substantially in the primary features driving the data, compared to England, Wales, and Scotland. In contrast, the other UK countries share more similar values on PC1, with Scotland showing some distinct behavior along PC2.

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

```
Importance of components:
                            PC1      PC2      PC3          PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
```

```
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

For the cumulative portion, pc1 cumulative proportion was added with pc2 proportion of variance to result in pc2 cumulative proportion then that was added with pc3 proportion of variance to give pc3 cumulative proportion.

Let's see what is in our PCA result object 'pca'

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

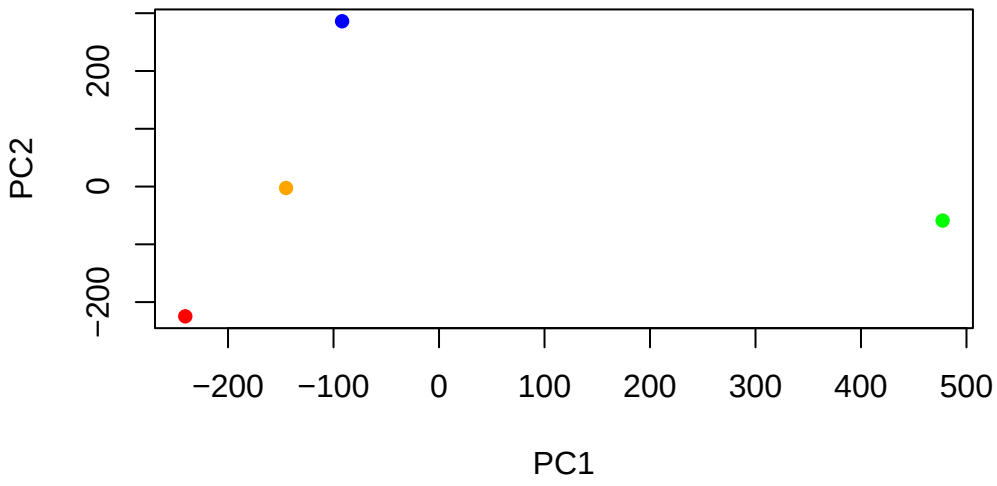
The 'pca$x' result object is where we will focus first as this details how the countries are related to each other in terms of our new "axis" (aka "PCs", "eigenvectors", etc.)

```
head(pca$x)
```

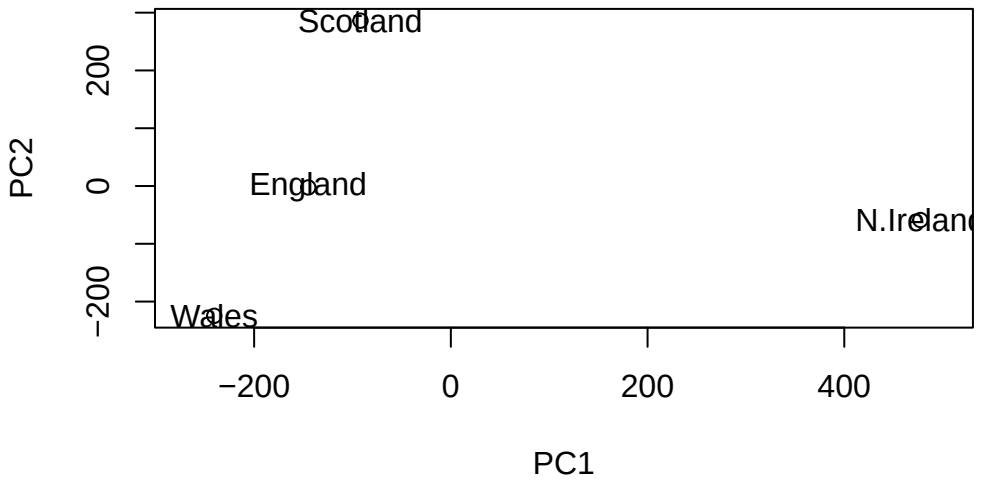
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

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], pch = 16, col = c("orange", "red", "blue", "green"), xlab = "PC1"
```

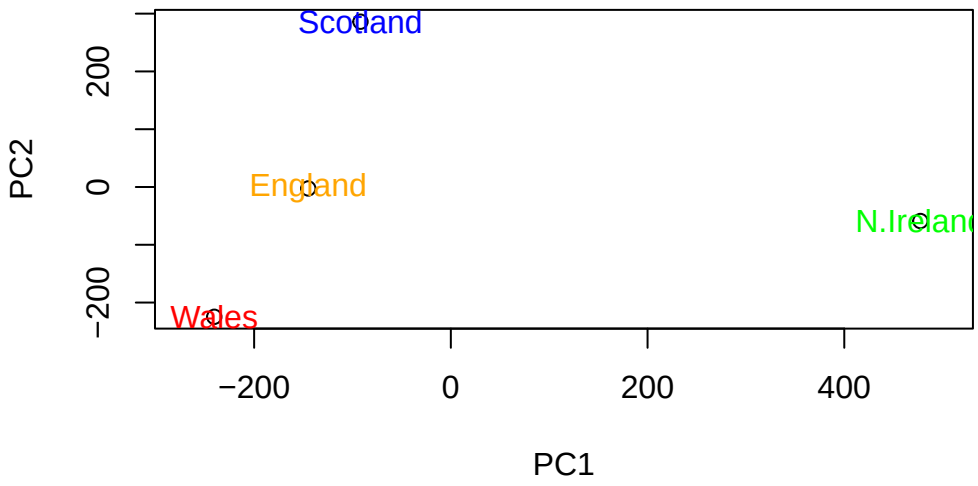



```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



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", "green"))
```



We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PCs (i.e. how the original variables contribute to our new better variable)

Writing [,1] makes it so it only displays the values that contribute to PC 1 instead of all the data

```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat               Fish
    -0.056955380         0.047927628        -0.258916658        -0.084414983
Fats_and_oils              Sugars     Fresh_potatoes           Fresh_Veg
    -0.005193623        -0.037620983         0.401402060        -0.151849942
   Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
    -0.243593729        -0.026886233        -0.036488269        -0.632640898
     Cereals            Beverages        Soft_drinks    Alcoholic_drinks
    -0.047702858        -0.026187756         0.232244140        -0.463968168
Confectionery
    -0.029650201
```

Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? A9. The two food groups that are featured prominantely are fresh potatoes and soft drinks. PC2 mainly tells us about the second-largest variance of the dataset which displays a different variance than that of PC1

Q10: How many genes and samples are in this data set? A10.