Class 7: Machine Learning 1

Andres Vasquez (A16278181)

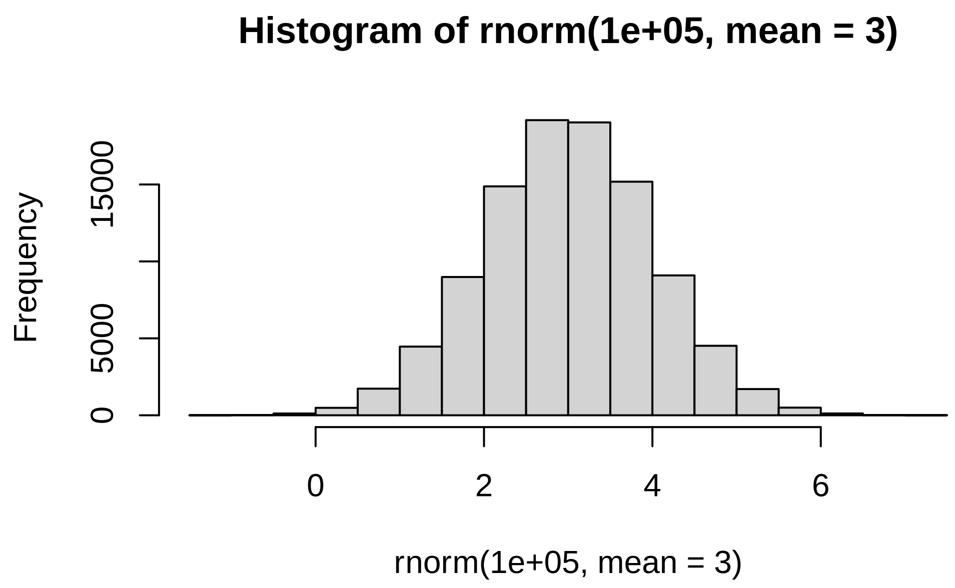
Today we will start our multi-part exploration of some key machine learning methods. We will begin with clustering - finding groupings in data, and then dimensionallity reduction.

Clustering

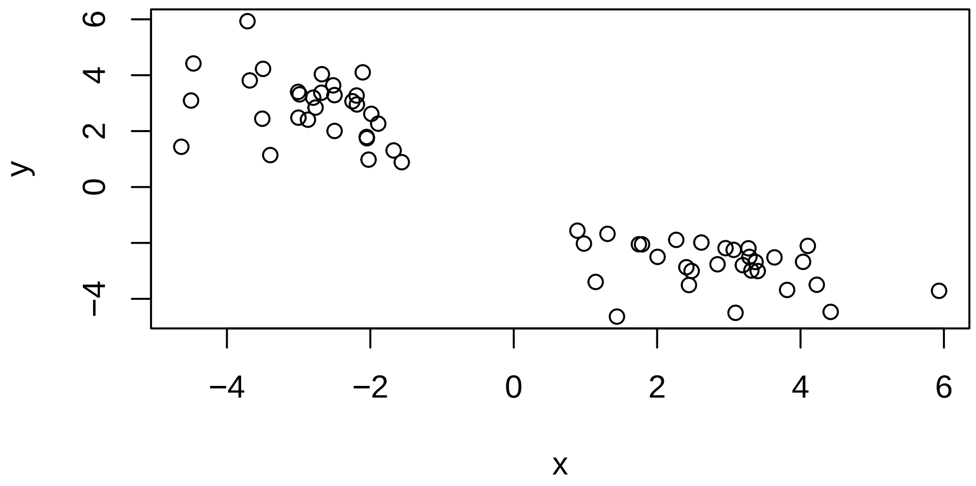
Lets start with "k-means" clustering.

The main function in base R for this `kmeans()`.

```r
# Make up some data
hist(rnorm(100000, mean =3 ))
```



```r
tmp <- c(rnorm(30, -3),
rnorm(30, +3))
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



Now let's tru out `kmeans()`

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

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

Cluster means:
          x         y
1  2.849577 -2.788943
2 -2.788943  2.849577

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 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 1 1
```

```
Within cluster sum of squares by cluster:
[1] 58.31159 58.31159
 (between_SS / total_SS =  89.1 %)

Available components:

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

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. How many points in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. What component of your result object defaults 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 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
```

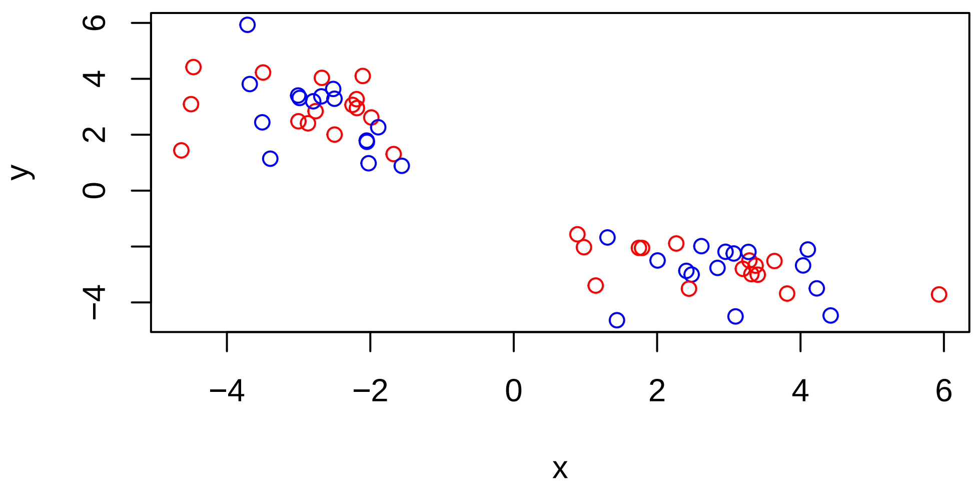
Q. What are centers/mean values of each cluster?

```
km$centers
```

```
          x         y
1  2.849577 -2.788943
2 -2.788943  2.849577
```

Q. Make a plot of your data showing your clustering results (groupings/clusters and cluster centers).

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

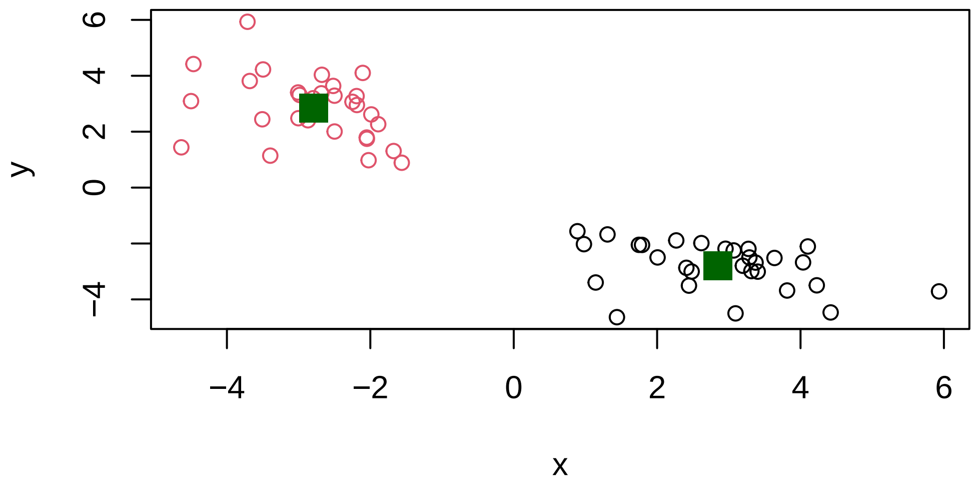


```
c(1:5) +c(100,1)
```

```
Warning in c(1:5) + c(100, 1): longer object length is not a multiple of
shorter object length
```

```
[1] 101   3 103   5 105
```

```
v <- plot(x, col = km$cluster) +
points(km$centers, col = "darkgreen", pch = 15, cex = 2)
```

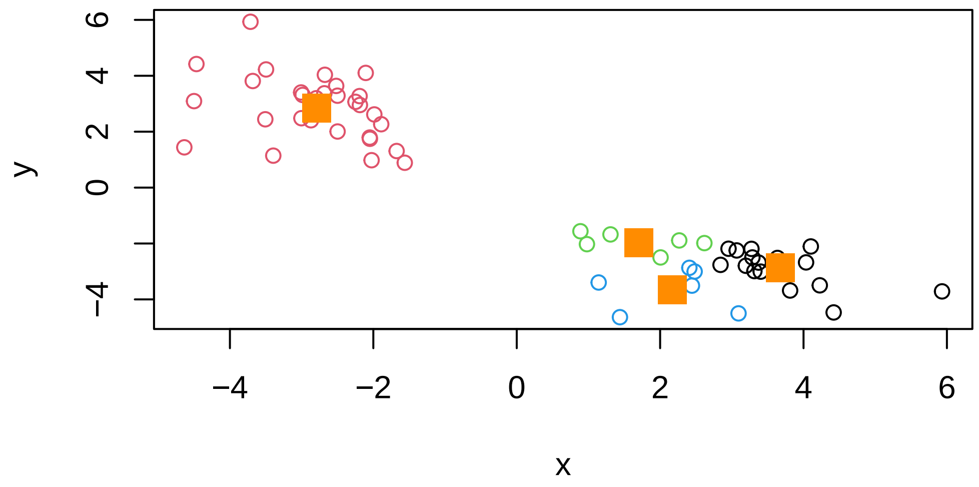


```
v
```

```
integer(0)
```

Q. Run `kmeans()` again and cluster in 4 groups and plot the results.

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



Hierarchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into a even smaller number of clusters.

The main function in base R is `hclust()`. This function does not take our input data directly but wants a "distance matrix" that details how (dis)similar all our input points are to each other.

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

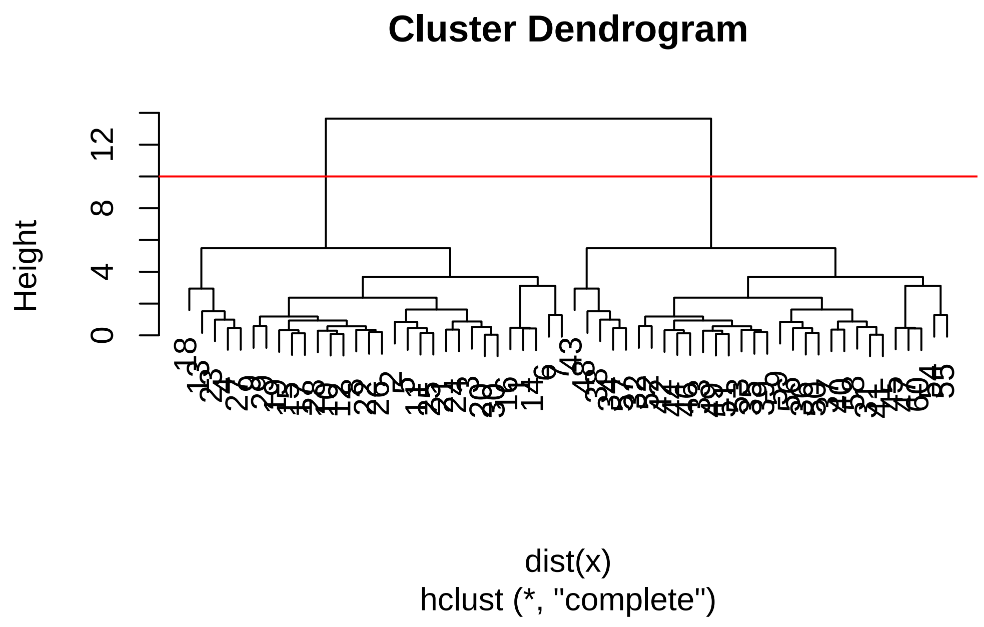
```

Call:
hclust(d = dist(x))

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

The print out above is not very useful (unlike that from kmeans) but there is a useful `plot()` method.

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

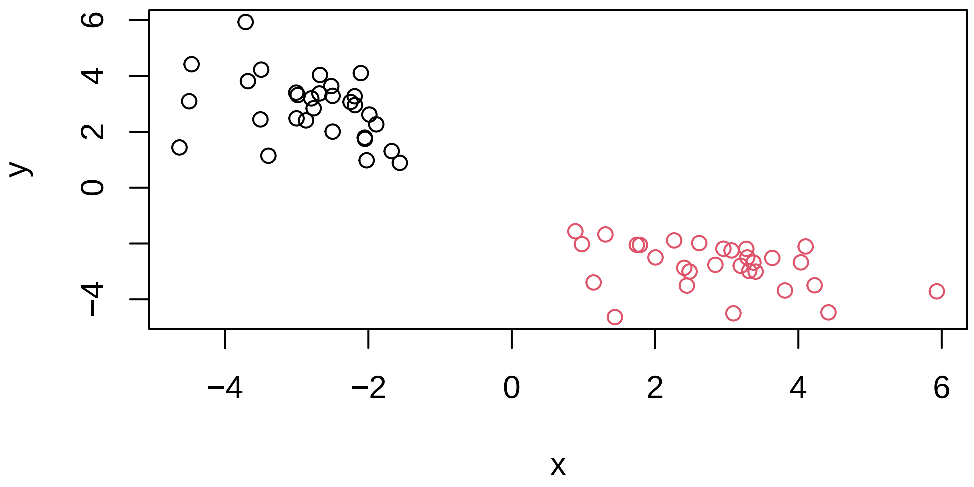


To get my main result (my cluster membership vector) I need to "cut" my tree using the function `cutree()`

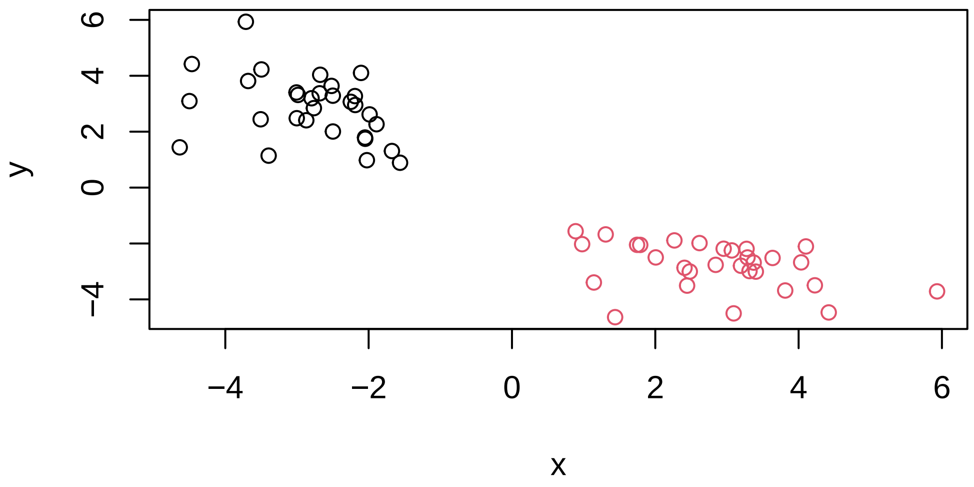
```r
grps <- cutree(hc, h = 10)
grps
```

```
 [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 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
```

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



```
plot(x, col = cutree(hc, h = 6))
```



Hands on with Principal Component Analysis (PCA)

The goal of PCA is to reduce the dimensionallity of a dataset down to some smaller subset of new variables (called PCs) that are useful bases for further analysis, like visualization, clustering, etc.

Data import

Read data about crazy eating trends in the UK and N. Ireland

```
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?

```
dim(x)
```

```
[1] 17  5
```

```
# 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
```

```
# Note how the minus indexing works
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
```

"Alternative approach"

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

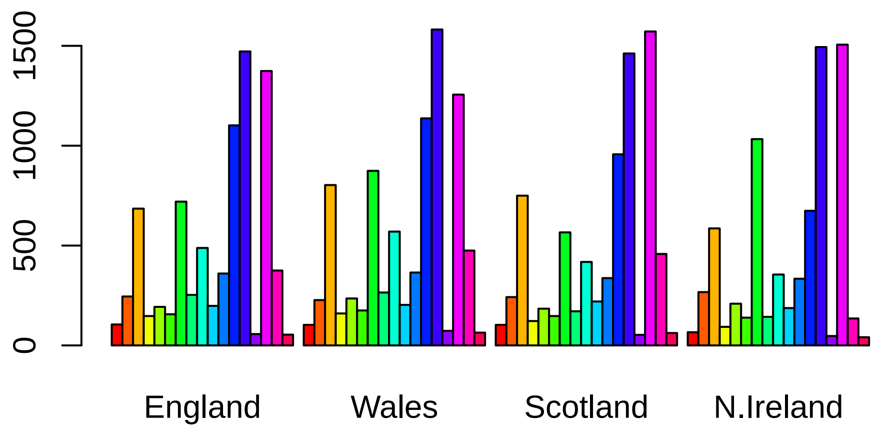
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             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 the second approach because it requires less steps, however the first approach would be more robust.

Spotting major differences and trends

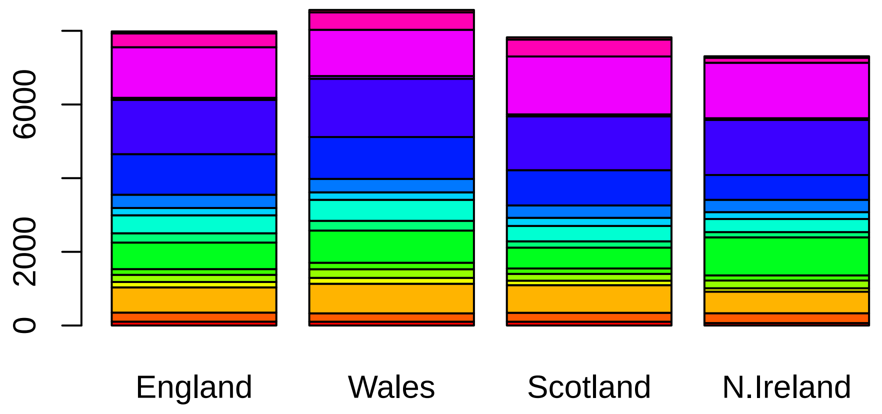
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

Adding the `beside =` affects the result of the plot.

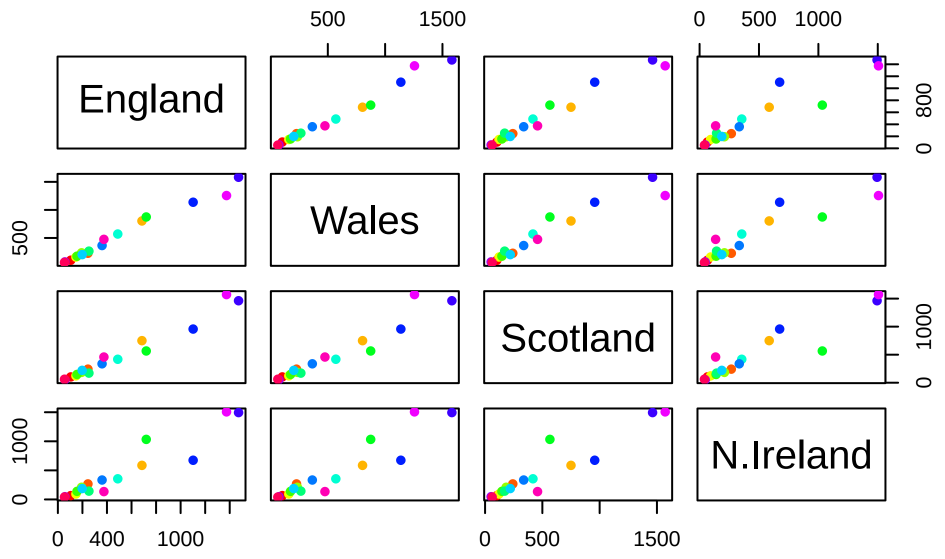
```
barplot(as.matrix(x), col=rainbow(nrow(x)))
```



The so-called "pairs" plot can be useful for small datasets:

> 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?

```r
pairs(x, col=rainbow(nrow(x)), pch = 16)
```



The location of the point indicates with country has a higher or lower trend compared to the other country. If the point is in the middle, then both countries have a similar trend.

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

PCA to the rescue

So the pairs plot is useful for small datasets but it can be lots of work to interpret and gets intractable for larger datasets.

So PCA to the rescue…

The main function in base R is called `prcomp()`. This function wants the transpose of our data in thise case, `t()`

```r
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
```

```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

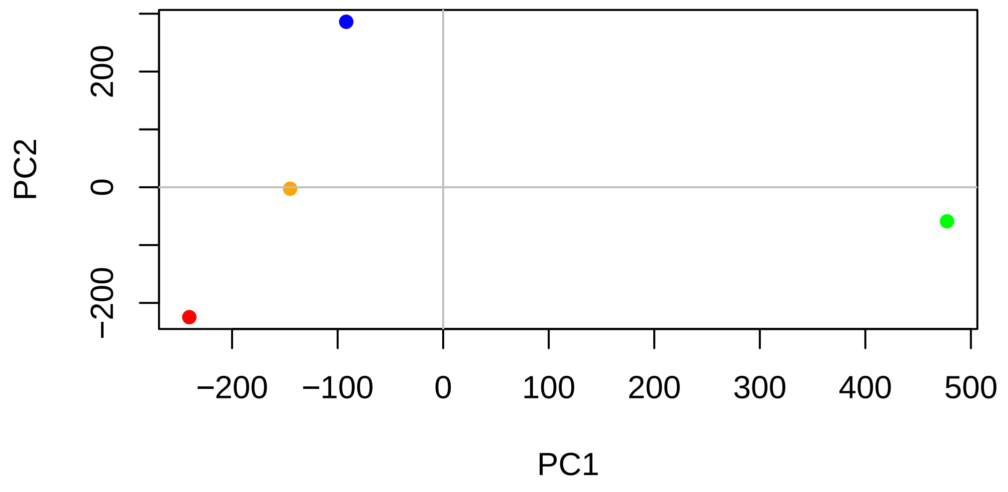
```r
pca$x
```

```
                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
```

A major PCA result viz is called a "PCA plot" (a.k.a: a score plot, biplot, PC1 vs. PC2 plot, ordination plot)

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

```r
# Plot PC1 vs PC2
mycols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], col = mycols, xlab="PC1", ylab="PC2", pch = 16)
abline(h=0, col = "gray")
abline(v=0, col = "gray")
```



Another important output from PCA is called the "loadings" vector or the "rotation" component - this tells us how much the original vairables (the foods in this case) contribute to the new PCs.

```r
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese              -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat         0.047927628  0.013915823  0.06367111  0.489884628
Other_meat          -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish                -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils       -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars              -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes       0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg           -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg           -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes  -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg       -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals             -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages           -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.103508492
```

```
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

PCA looks to be a super useful method for gaining some insight into high dimensional data that is difficult to examine in other ways.