Lab 7: Machine Learning 1

Rocio Silenciario

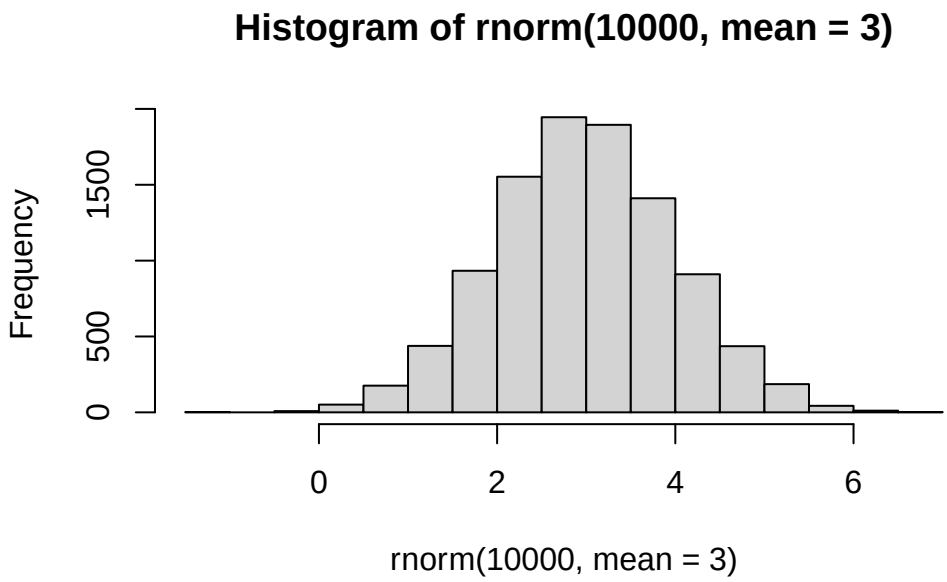
Table of contents

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

Clustering

To start let's make up some data to cluster where we know what the answer should be. The `rnorm()` function will help us here.

```
hist(rnorm(10000, mean=3))
```



Return 30 numbers centered on -3

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

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

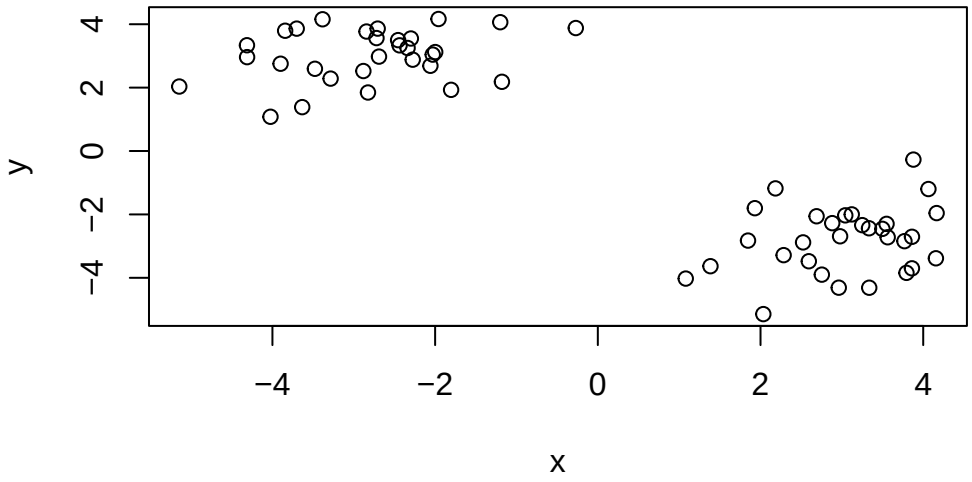
```
                x          y
 [1,] -3.6338093  1.3844848
 [2,] -3.4777442  2.5934648
 [3,] -1.8036192  1.9300521
 [4,] -2.6901183  2.9779347
 [5,] -3.2846615  2.2839820
 [6,] -4.3131171  3.3364983
 [7,] -1.1995826  4.0638823
 [8,] -1.9990160  3.1212024
 [9,] -2.3385863  3.2500842
[10,] -2.2755508  2.8813144
[11,] -2.0580808  2.6891074
[12,] -2.7062489  3.8616004
[13,] -2.4552351  3.4979252
[14,] -3.8998683  2.7533634
[15,] -4.3114643  2.9618247
```

```
[16,] -5.1450938  2.0344201
[17,] -3.3840761  4.1560378
[18,] -2.8823236  2.5229340
[19,] -2.8254236  1.8462017
[20,] -1.9585141  4.1647554
[21,] -4.0240896  1.0807356
[22,] -1.1788362  2.1833659
[23,] -2.7208708  3.5632795
[24,] -0.2709014  3.8791512
[25,] -3.8436981  3.7940160
[26,] -2.2982329  3.5492586
[27,] -2.4370945  3.3334399
[28,] -2.0297087  3.0417275
[29,] -2.8436511  3.7691317
[30,] -3.7015453  3.8610612
[31,]  3.8610612 -3.7015453
[32,]  3.7691317 -2.8436511
[33,]  3.0417275 -2.0297087
[34,]  3.3334399 -2.4370945
[35,]  3.5492586 -2.2982329
[36,]  3.7940160 -3.8436981
[37,]  3.8791512 -0.2709014
[38,]  3.5632795 -2.7208708
[39,]  2.1833659 -1.1788362
[40,]  1.0807356 -4.0240896
[41,]  4.1647554 -1.9585141
[42,]  1.8462017 -2.8254236
[43,]  2.5229340 -2.8823236
[44,]  4.1560378 -3.3840761
[45,]  2.0344201 -5.1450938
[46,]  2.9618247 -4.3114643
[47,]  2.7533634 -3.8998683
[48,]  3.4979252 -2.4552351
[49,]  3.8616004 -2.7062489
[50,]  2.6891074 -2.0580808
[51,]  2.8813144 -2.2755508
[52,]  3.2500842 -2.3385863
[53,]  3.1212024 -1.9990160
[54,]  4.0638823 -1.1995826
[55,]  3.3364983 -4.3131171
[56,]  2.2839820 -3.2846615
[57,]  2.9779347 -2.6901183
[58,]  1.9300521 -1.8036192
```

```
[59,]  2.5934648 -3.4777442
[60,]  1.3844848 -3.6338093
```

Make a plot of `x`

```
plot(x)
```



K-means

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

```
km <- kmeans(x, 2)
km
```

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

Cluster means:
          x         y
1  3.012208 -2.799692
2 -2.799692  3.012208
```

```
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] 52.4304 52.4304
 (between_SS / total_SS =  90.6 %)

Available components:

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

The `kmeans()` function returns a "list" with 9 components. You can see the named components of any list with the `attributes()` function.

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. Cluster assignment/membership vector?

```
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
```

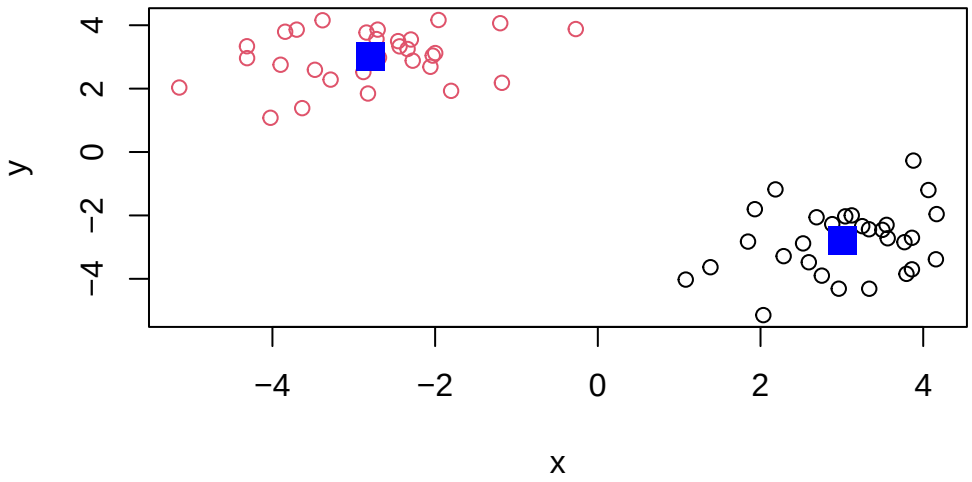
Q. Cluster centers?

```
km$centers
```

```
          x         y
1  3.012208 -2.799692
2 -2.799692  3.012208
```

Q. Make a plot of our `kmeans()` results showing cluster assignment using different colors for each cluster/group of points and cluster centers in blue?

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



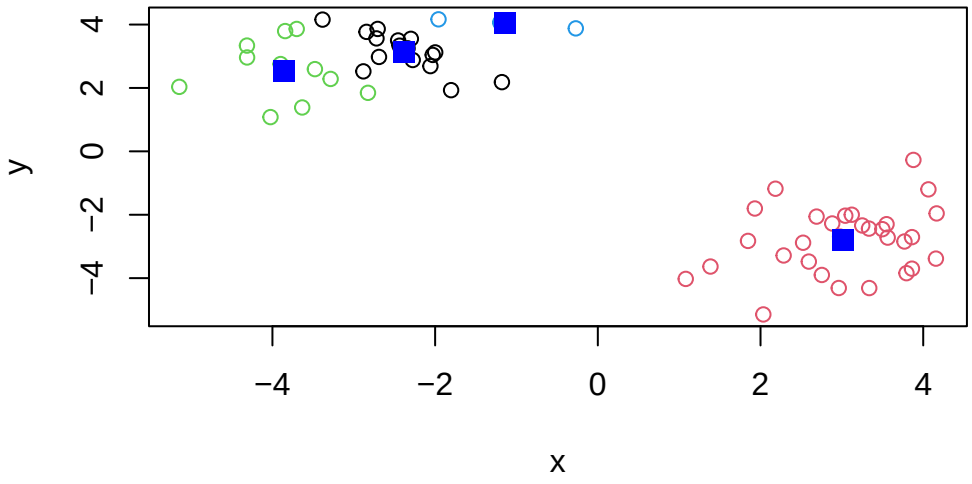
Q. Run `kmeans()` again on `x` and this cluster into 4 groups/clusters, and plot the same result figure as above.

```
fm <- kmeans(x, 4)
fm
```

```
K-means clustering with 4 clusters of sizes 16, 30, 11, 3

Cluster means:
```

```
          x         y
1 -2.381328  3.145525
2  3.012208 -2.799692
3 -3.860047  2.539096
4 -1.142999  4.035930

Clustering vector:
 [1] 3 3 1 1 3 3 4 1 1 1 1 1 1 3 3 3 1 1 3 4 3 1 1 4 3 1 1 1 1 3 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

Within cluster sum of squares by cluster:
[1]  9.508028 52.430399 12.158793  1.470778
 (between_SS / total_SS =  93.2 %)

Available components:

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

```r
plot(x, col=fm$cluster)
points(fm$centers, col="blue", pch=15, cex=1.5)
```



key-point: K-means clustering is super popular but can be miss-used. One big

limitation is that it can impose a clustering patern on your data even if clear natural grouping don't exist - i.e. it does what you tell it to do in terms of `centers`.

Hierarchical Clustering

The main function in "base" R for Hierarchical Clustering is called `hclust()`

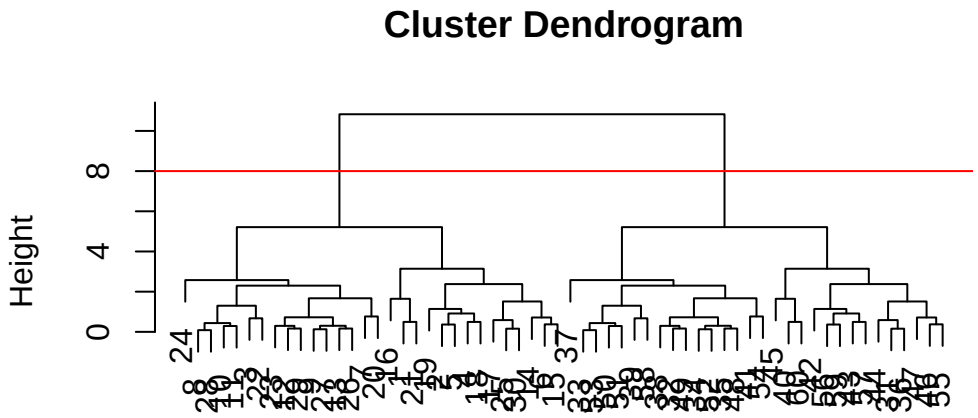
You can't just pass our dataset as is into `hclust()`, you must give "distance matrix" as input. We can get this from the `dist()` function in R.

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

```
Call:
hclust(d = d)

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

The results of `hclust()` don't have a useful `print()` method but do have a special `plot()` method.

```
plot(hc)
abline(h=8, col="red")
```



To get our main cluster assignment (membership vector) we need to "cut" the tree at the big goal posts…

```
grps <- cutree(hc, h=8)
grps
```

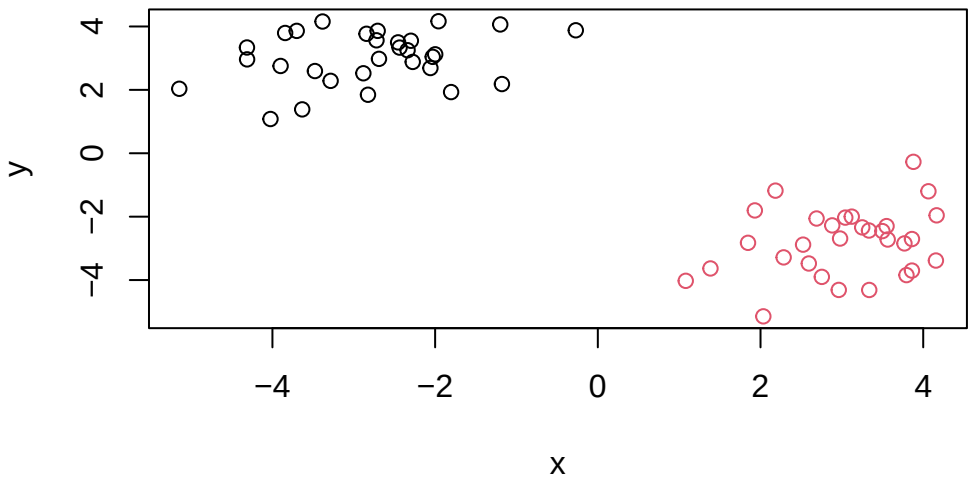
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

To determine how many vectors are in each assignment when using hc:

```
table(grps)
```

```
grps
 1  2
30 30
```

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



Hierarchical Clustering is distinct in that the dendrogram (tree figure) can reveal the potential grouping in your data (unlike K-means).

#Principal Component Analysis (PCA)

PCA is a common and highly useful dimensionality reduction technique used in many fields - particularly bioinformatics.

Here we will analyze some data from the UK on food consumption.

Data import

```
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
```

```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

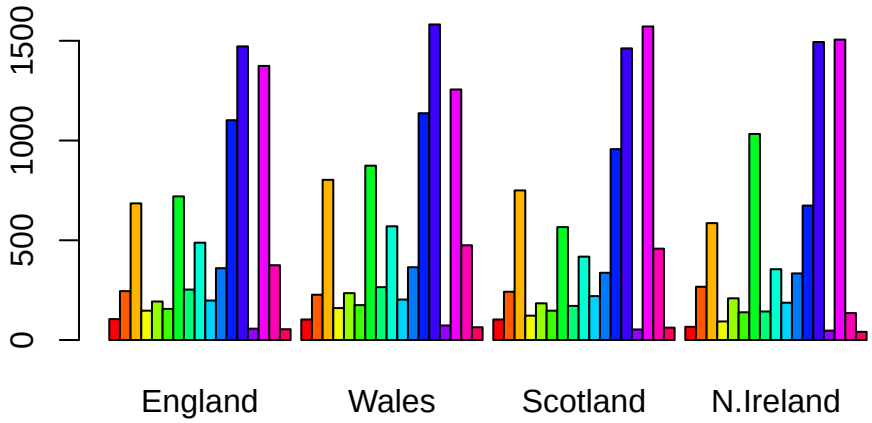
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

Setting the first column to be the names instead of numbers:

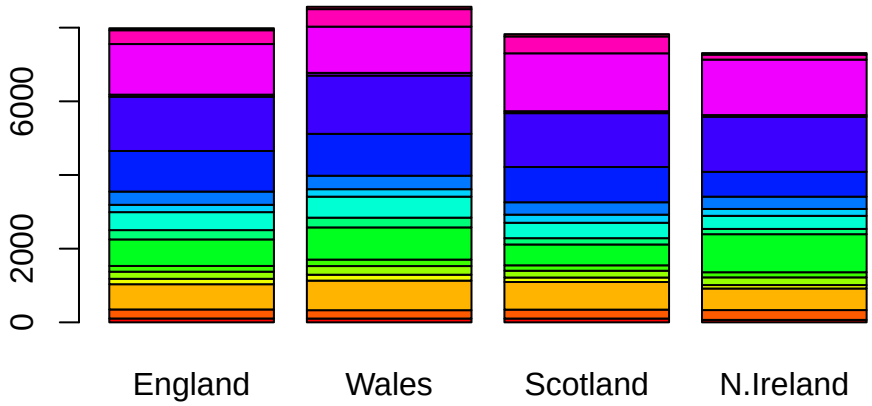
```
x <- read.csv(url, row.names =1)
head(x)
```

```
               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
```

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

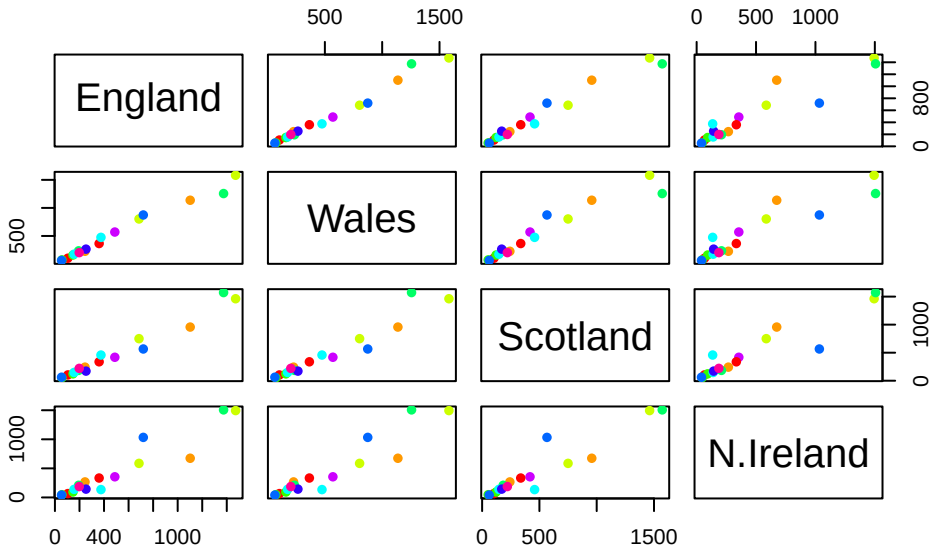



```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



One conventional plot that can be useful is called a "pairs" plot.

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



This figure shows the similarities between the four countries as represented by the linear pattern present when comparing England, Wales and Scotland to each other. When comparing N. Ireland to the former three, we can see there is more variability in the plots, as they do not follow the same linear pattern as the others.

PCA to the rescue

The main function in base R for PCA is called `prcomp()`.

```
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
```

The `prcomp` function returns a list object of our result with five attributes/components.

```
attributes(pca)
```

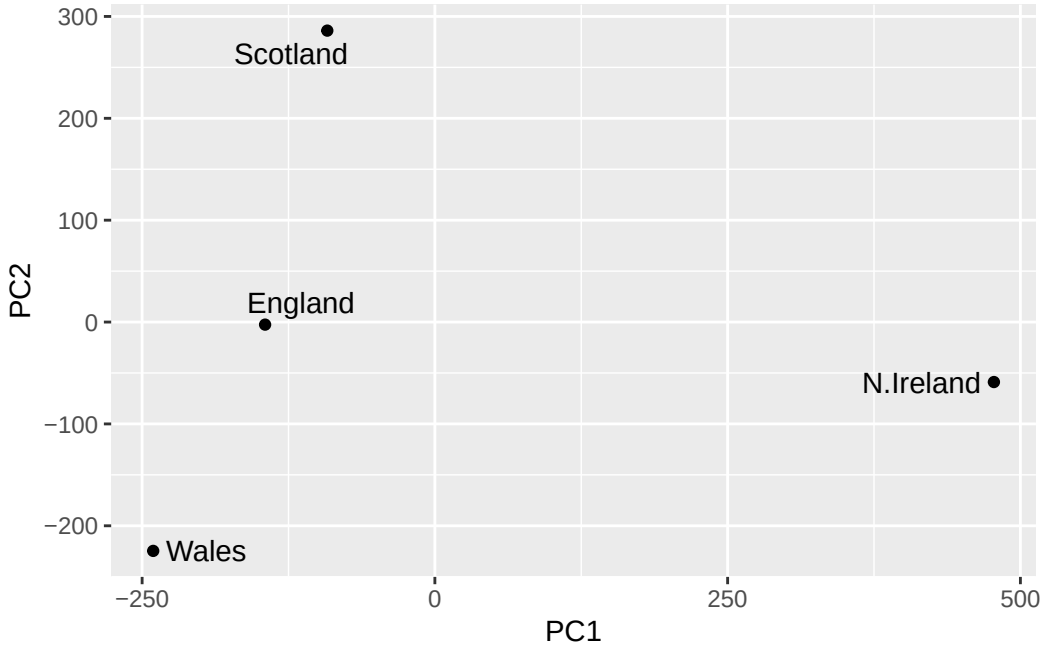
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The two main "results" in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains the score of the data on the new PC axis - we use these to make our "PCA plot".

```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

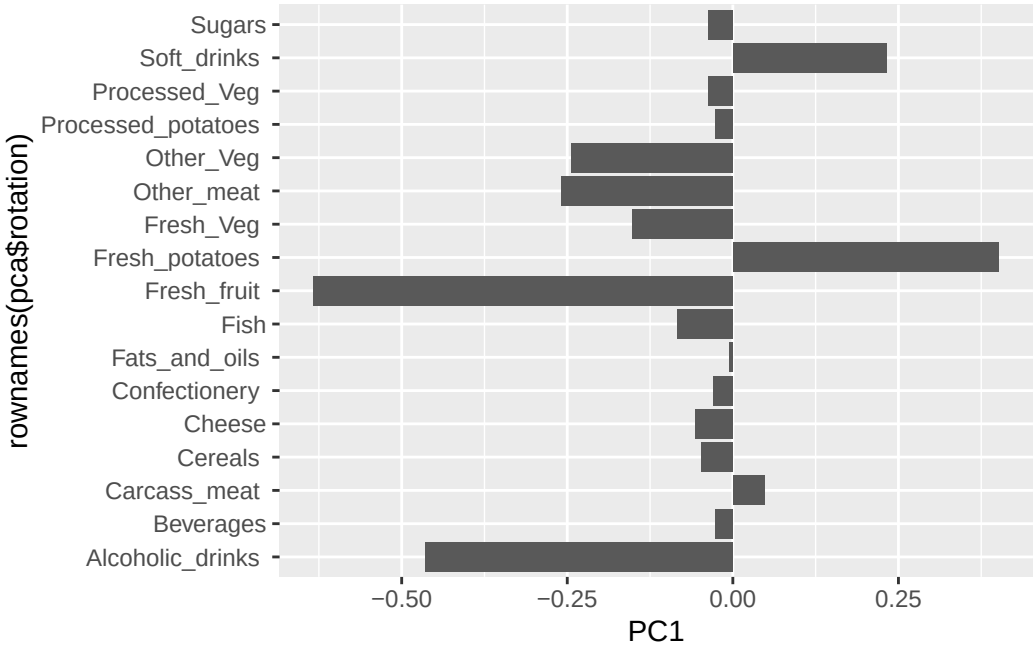
```
library(ggplot2)
library(ggrepel)

# Make a plot of pca$x with PC1 vs PC2

ggplot(pca$x) +
  aes(PC1,PC2, label=rownames(pca$x))+
  geom_point()+
  geom_text_repel()
```



This figure shows the variability between the four countries plotted in a PC1 vs PC2 plot. We can see that Scotland, England and Wales are very similar to each other according to the PC1 axis, while N. Ireland is not. According to the PC2 axis we can see that there is some similarities between N. Ireland and England, but there is still a significant amount of variability between N. Ireland and Wales & Scotland.

The second major result is contained in the `pca$rotation` object or component. Let's plot this to see what PCA is picking up…

```
ggplot(pca$rotation)+
  aes(PC1, rownames(pca$rotation))+
  geom_col()
```



This figure shows the ways in which N. Ireland is different than England, Wales and Scotland. N. Ireland differences are highlighted by the bars pointing to the positive numbers, with soft drinks, fresh potatoes, and carcass meat being the main foods that N. Ireland consumes. The bars pointing to the negative. numbers represent the food consumed mostly by England, Wales and Scotland.