Intro to Machine Learning 1

Karina Cardenas, A16742606

2025-04-22

Table of contents

Intro to Machine learning

There are different types of machine learning, a few notable mentions:

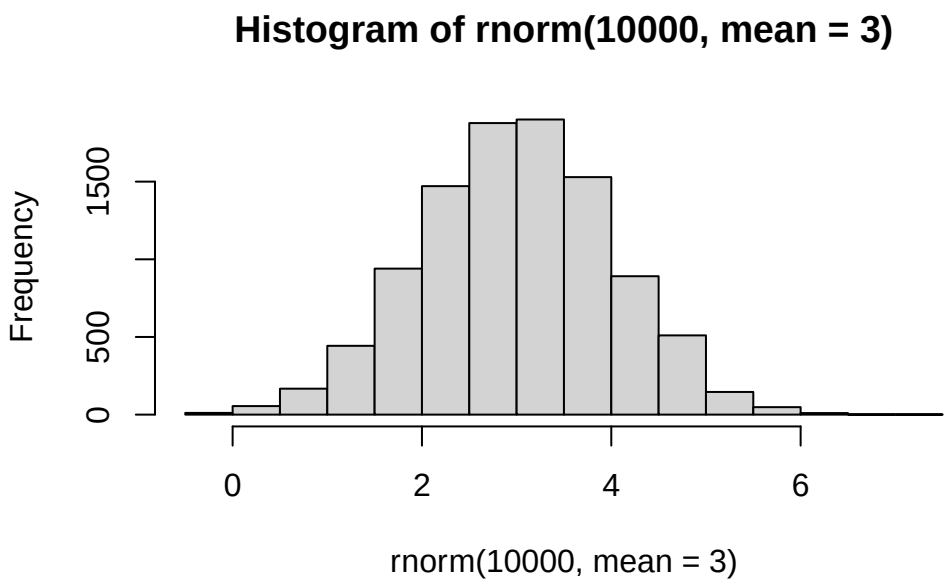
- **Unsupervised learning**: Finding structure in unlabeled data
- **Supervised learning**: Making predictions based on labeled data i.e regression/classification
- **Reinforcement learning**: Making decisions based on past experiences

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
```

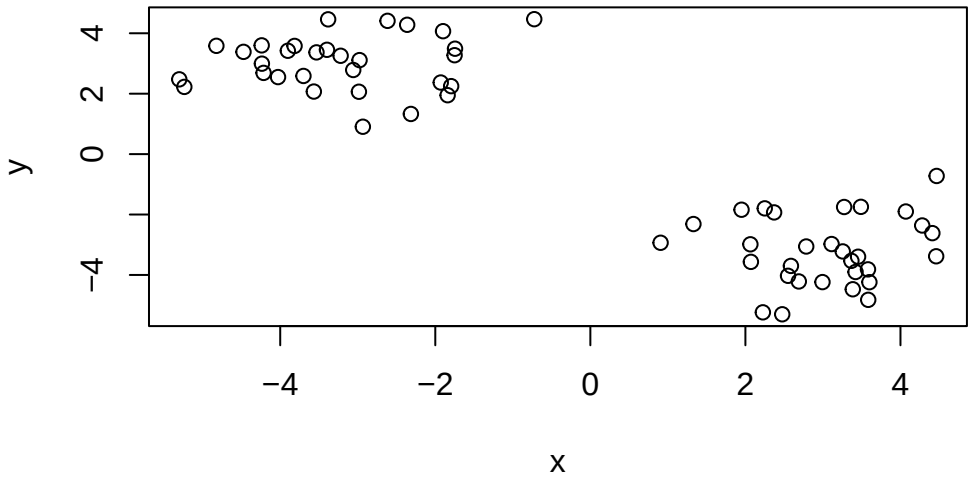
```
               x          y
 [1,] -2.9767105  3.1114617
 [2,] -4.8235034  3.5841082
 [3,] -5.3020147  2.4749565
 [4,] -4.2390199  3.5990991
 [5,] -2.9852269  2.0641428
 [6,] -2.3146612  1.3293529
```

```
 [7,] -1.8999455  4.0688101
 [8,] -1.7510140  3.2731864
 [9,] -3.7004737  2.5870029
[10,] -1.9296286  2.3701207
[11,] -3.2208503  3.2554209
[12,] -2.3625383  4.2804249
[13,] -1.7456592  3.4893401
[14,] -4.2365031  2.9937973
[15,] -3.9017569  3.4206261
[16,] -3.0583619  2.7844152
[17,] -3.3821010  4.4604195
[18,] -1.7952746  2.2495290
[19,] -4.2148528  2.6886318
[20,] -3.8151237  3.5808544
[21,] -1.8406185  1.9498786
[22,] -3.5321962  3.3670303
[23,] -5.2371504  2.2250322
[24,] -2.6159083  4.4114858
[25,] -3.3973083  3.4531332
[26,] -0.7234705  4.4663621
[27,] -4.0283540  2.5499375
[28,] -3.5665264  2.0707579
[29,] -4.4738736  3.3848688
[30,] -2.9342472  0.9060031
[31,]  0.9060031 -2.9342472
[32,]  3.3848688 -4.4738736
[33,]  2.0707579 -3.5665264
[34,]  2.5499375 -4.0283540
[35,]  4.4663621 -0.7234705
[36,]  3.4531332 -3.3973083
[37,]  4.4114858 -2.6159083
[38,]  2.2250322 -5.2371504
[39,]  3.3670303 -3.5321962
[40,]  1.9498786 -1.8406185
[41,]  3.5808544 -3.8151237
[42,]  2.6886318 -4.2148528
[43,]  2.2495290 -1.7952746
[44,]  4.4604195 -3.3821010
[45,]  2.7844152 -3.0583619
[46,]  3.4206261 -3.9017569
[47,]  2.9937973 -4.2365031
[48,]  3.4893401 -1.7456592
[49,]  4.2804249 -2.3625383
```

```
[50,]  3.2554209 -3.2208503
[51,]  2.3701207 -1.9296286
[52,]  2.5870029 -3.7004737
[53,]  3.2731864 -1.7510140
[54,]  4.0688101 -1.8999455
[55,]  1.3293529 -2.3146612
[56,]  2.0641428 -2.9852269
[57,]  3.5990991 -4.2390199
[58,]  2.4749565 -5.3020147
[59,]  3.5841082 -4.8235034
[60,]  3.1114617 -2.9767105
```

Make a plot of X

```
plot(x)
```



K-means

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

```r
#x = x
#centers = 2, # of groups

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

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

Cluster means:
          x         y
1 -3.200162  3.015006
2  3.015006 -3.200162

Clustering vector:
 [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

Within cluster sum of squares by cluster:
[1] 61.13703 61.13703
 (between_SS / total_SS =  90.5 %)

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

```r
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. How do we get the cluster membership assignment?

```
km$cluster
```

```
 [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
```

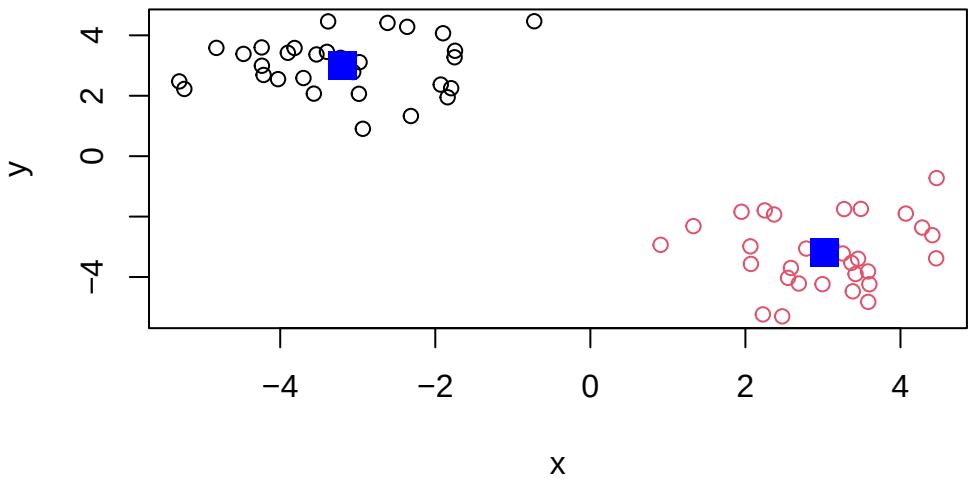
Q. Cluster centers?

```
km$centers
```

```
          x         y
1 -3.200162  3.015006
2  3.015006 -3.200162
```

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

```
#different colors for each cluster/group
plot(x, col = km$cluster)

#cluster centers: col = color, pch = shape, cex = character size
points(km$centers, col = "blue", pch = 15, cex = 2)
```



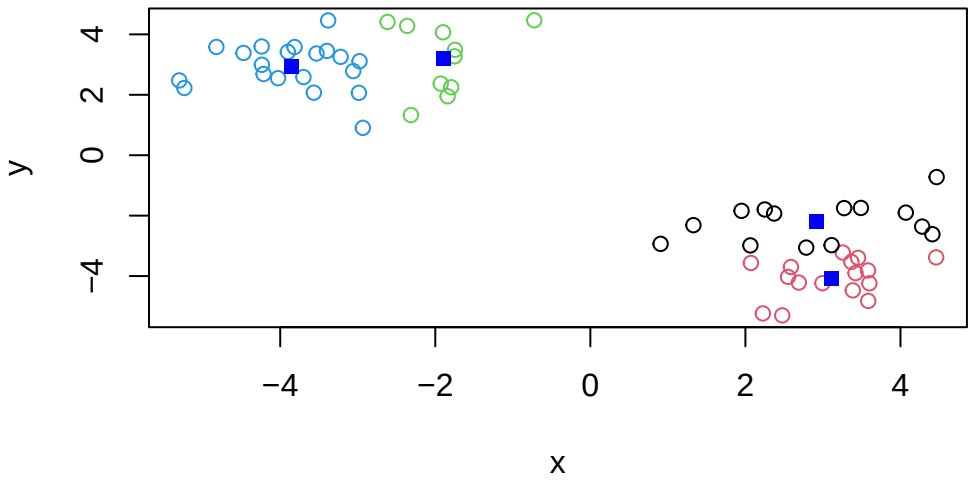
Q. Run `kmeans()` again on `x` and this time cluster it into 4 groups/clusters and plot the same result figure as above.

```
km4 <- kmeans(x, centers = 4 )
km4
```

```
K-means clustering with 4 clusters of sizes 14, 16, 10, 20

Cluster means:
          x         y
1  2.911037 -2.209519
2  3.105980 -4.066976
3 -1.897872  3.188849
4 -3.851308  2.928085

Clustering vector:
 [1] 4 4 4 4 4 3 3 3 4 3 4 3 3 4 4 4 4 3 4 4 3 4 4 3 4 3 4 4 4 4 1 2 2 2 1 2 1 2
[39] 2 1 2 2 1 2 1 2 2 1 1 2 1 2 1 1 1 1 2 2 2 1

Within cluster sum of squares by cluster:
[1] 23.05224 12.03994 14.07955 21.16475
 (between_SS / total_SS =  94.5 %)
```

```
Available components:

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

```
plot(x, col = km4$cluster, )
points(km4$centers, col = "blue", pch = 15, cex = 1)
```



keypoint: K -means clustering is super popular but can be misused. one big limitarion is that it can impose a clustering pattern on your data even if clear natural groupiing doesn't exist - i.e it does what you tell it to do in terms of `centers`

Heirarchical Clustering

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

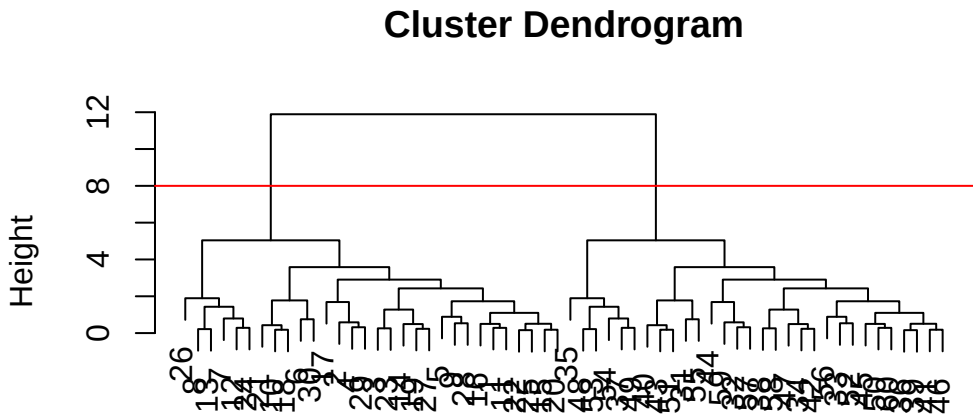
You can't just pass our input dataset as is into `hclust()` as we did with `kmeans()`. You must give "distance matrix" as input. We can get this from the `dist()` function in R.

```r
#calculating distance matrix
d <- dist(x)

#clustering d/x
hc <- hclust(d)

#printing hc
hc
```

```
Call:
hclust(d = d)

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

The results of `hclust()` dont have a useful `print()` method but do have a special `plot()` method.

```r
#x = hc
plot(hc)

#adds a horizontal line to cut the tree
abline(h = 8, col = "red")
```



To get out main cluster assignment (membership vector), we need to "cut" the tree at the big line.

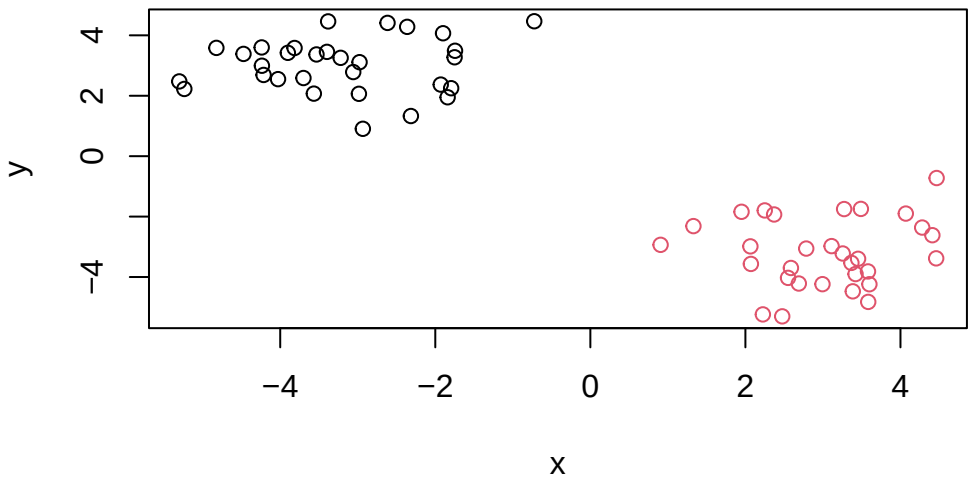
```
#cutree = function, hc = plot/data, h = height at cutting
grps <- cutree(hc, h = 8)
grps
```

```
 [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 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
```

```
#table function
table(grps)
```

```
grps
 1  2
30 30
```

```
#plotting x, with hc grps
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.

```
#Reading csv file
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
```

we need to change the first column to be the names of the foods and not numbered. There are several ways to do so, but this way is inefficient and destructive.

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

#overwriting x by removing a column everytime it is ran
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
```

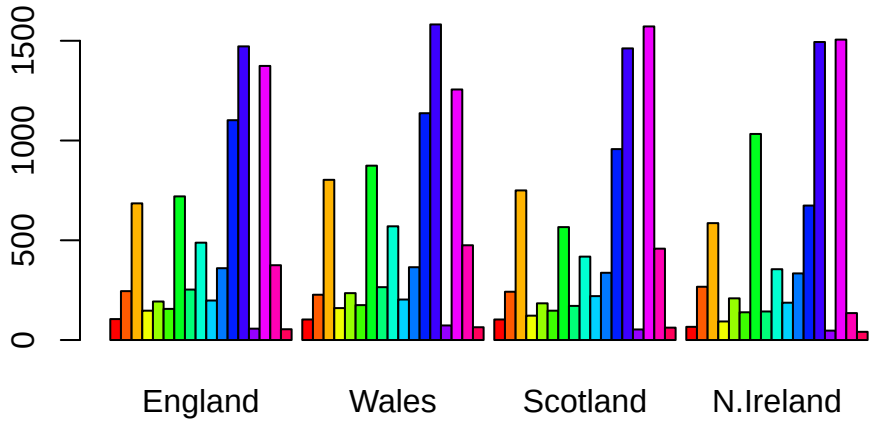
However, this way changes the row names of the first column without removing the country columns.

```
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
```

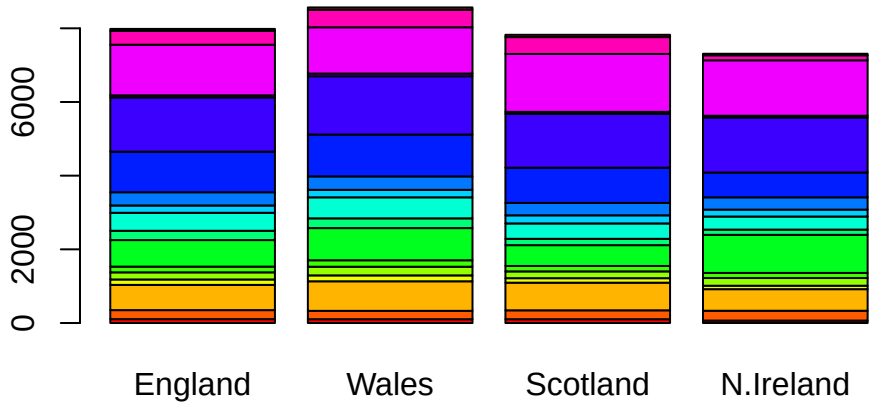
barplot 1

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



barplot 2

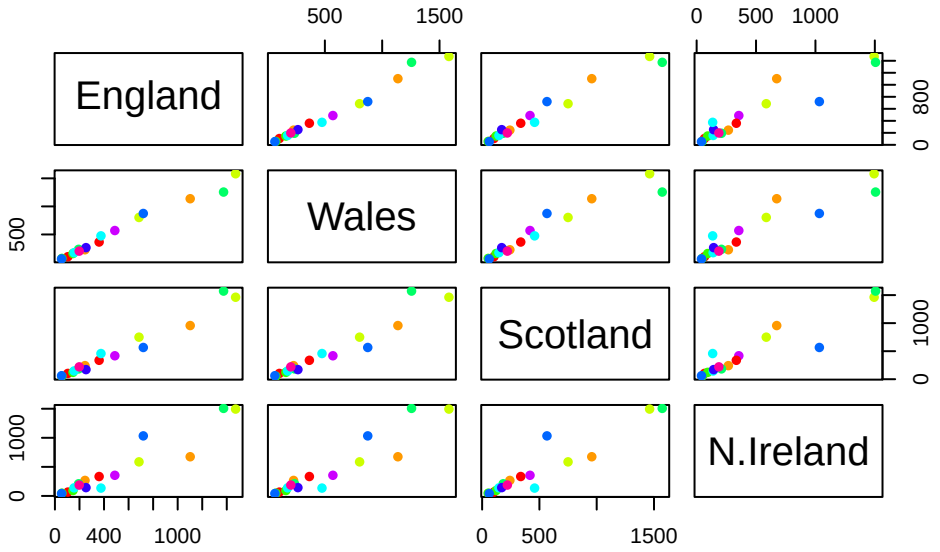
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



Paris plot

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

```
#paris = type of plot, x = data, col = color, pch = style of marker
pairs(x, col=rainbow(10), pch=16)
```



PCA to the Rescue

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

```
#t = transpose, make the countries be the rows and cheese be the columns
t(x)
```

```
           Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
England       105           245         685  147            193     156
Wales         103           227         803  160            235     175
Scotland      103           242         750  122            184     147
N.Ireland      66           267         586   93            209     139
           Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
England               720        253        488                 198
Wales                 874        265        570                 203
Scotland              566        171        418                 220
N.Ireland            1033        143        355                 187
           Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England              360         1102     1472         57         1374
Wales                365         1137     1582         73         1256
Scotland             337          957     1462         53         1572
N.Ireland            334          674     1494         47         1506
```

```
          Alcoholic_drinks  Confectionery
England                375             54
Wales                  475             64
Scotland               458             62
N.Ireland              135             41
```

```
#pca = anlaysis
pca <- prcomp(t(x))

#overview of pca results
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 results with fivee attributes/components

```
attributes(pca)
```

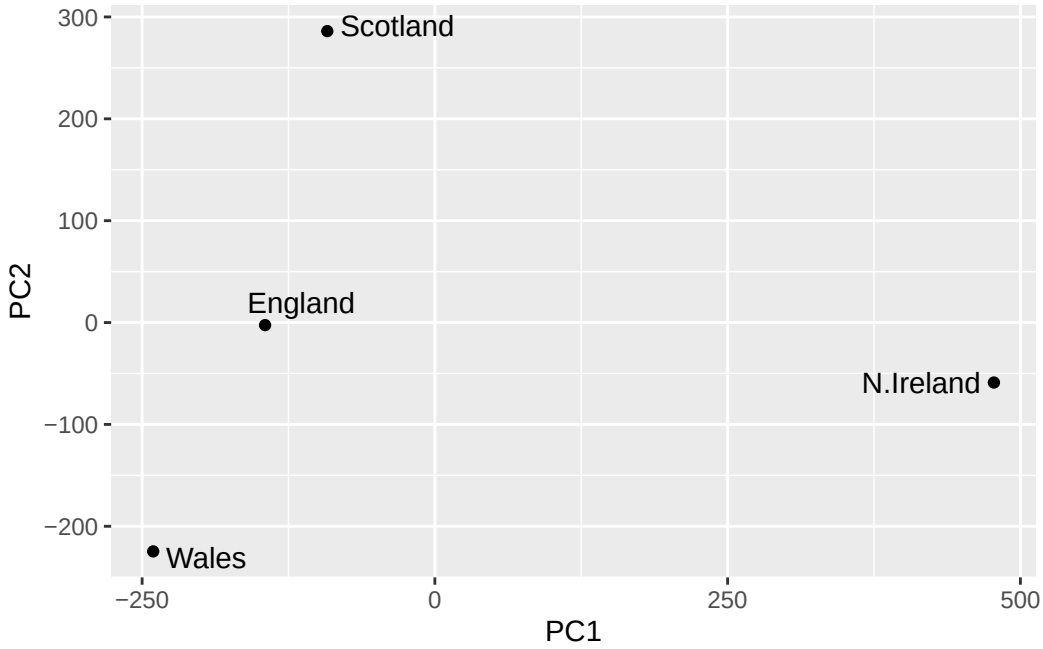
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The two main "results" in here are `pca$x` and `pca$rotation`. The first set of (`pcs$x`) contains the scores of the data on the new PC acis - 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
```

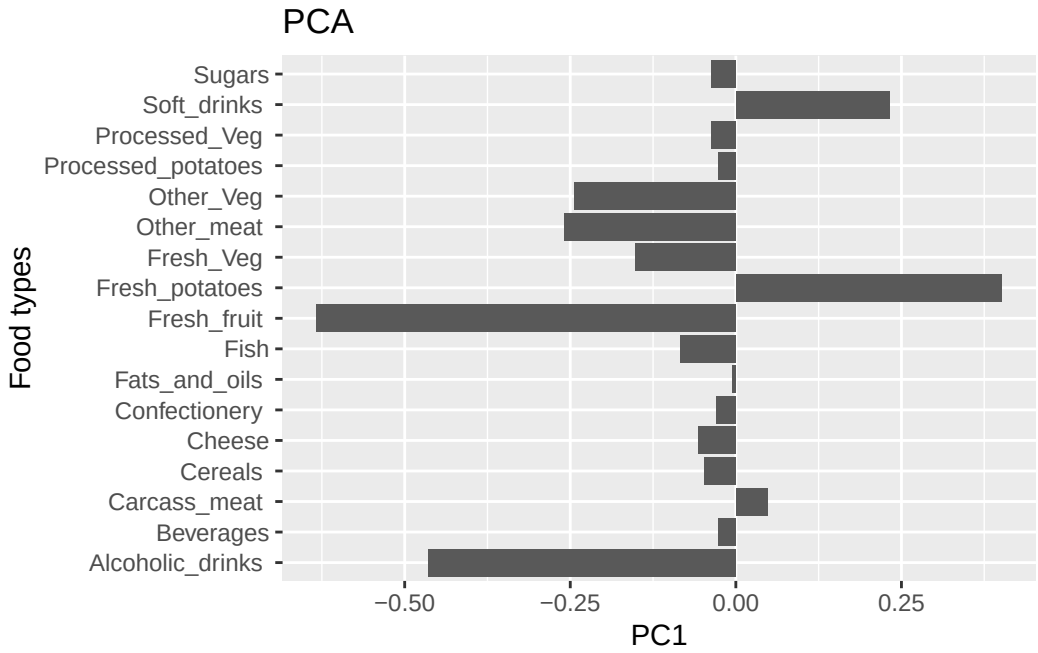
```
library(ggplot2)
library(ggrepel)

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



The plot utilizes **PCA** to display the similarities that are observed within scotland, England and wales using summarized components. Within this plot, N.Ireland is observed as an outleir, but fails to specify what food category creates this disparity.

The second major result is contained in the `pca$rotation` object/component

```
ggplot(pca$rotation) +
  aes(PC1, rownames(pca$rotation)) +
  geom_col() +
  labs(title = "PCA", x = "PC1", y = "Food types")
```



Tells us how the original variables contribute to PCA. Anything to the right side of the plot (positive values) is what abundantly consumed in Ireland. It visually displays the differences of Ireland previously not visible with just the data.