Class 07 - Machine Learning pt. 1

Gabriella Tanoto (A18024184)

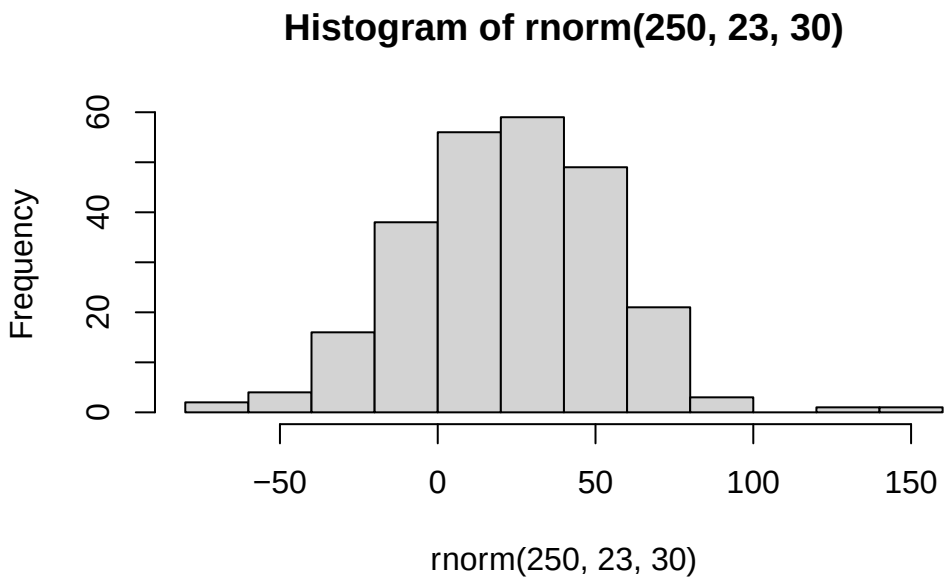
Table of contents

Today, we are exploring unsupervised machine learning starting with *clustering* and *dimentionality reduction*!

Clustering

Let's make a data where we know what the answer should be, just to get used to the function and see if it works! The `rnorm()` function will help us.

```
hist(rnorm(250, 23, 30))
```



Return 30 numbers centered at -3

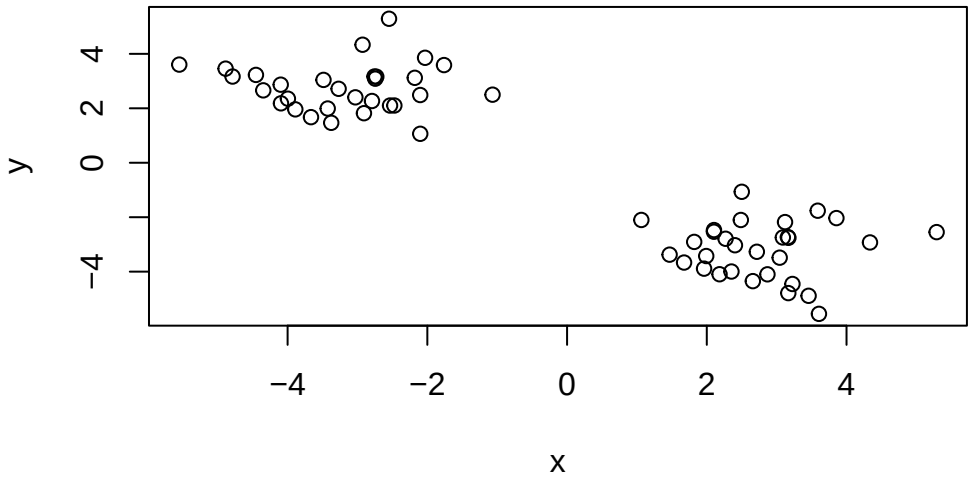
```
c(rnorm(30, -3), rnorm(30, 3))
```

```
 [1] -4.0797131 -3.5476526 -3.1922873 -4.1022650 -4.2324042 -3.6388201
 [7] -2.4319615 -3.8168936 -2.7561134 -2.0412102 -1.9956784 -2.3478639
[13] -2.8236184 -2.5283182 -1.3910000 -4.1933143 -2.4627504 -1.5830252
[19] -3.2765972 -1.6849563 -4.7363680 -4.2450875 -1.8810337 -3.3309069
[25] -3.5508715 -2.8031507 -3.6959825 -2.5737699 -1.9705357 -3.5507087
[31]  3.2403781  2.9805515  2.2193983  3.5606173  3.0731060  0.7973581
[37]  1.9818087  2.1090681  3.0134149  2.6475173  2.3470285  2.5749775
[43]  4.1863120  4.6307863  3.7558047  1.3426547  2.8307374  3.5322329
[49]  1.4553030  3.8268165  3.2864105  2.7038179  2.0309688  2.5900813
[55]  3.4647259  2.0870814  1.7962168  1.5778262  2.1993344  2.4358626
```

```
#same as:
tmp <- tmp <-  c(rnorm(30,-3), rnorm(30,3))
x <-  cbind(x=tmp, y=rev(tmp))
```

Now, make this into a Plot:

```r
plot(x)
```



K-Means

Main functuion in "base R" for K-mean clustering is called `kmeans()`

```r
km <- kmeans(x, centers=2) #centers refers to how many groups we want it to give us.
  #clustering vector is which cluster each of the points are at (i.e., cluster 1 or 2).
km
```

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

Cluster means:
          x         y
1  2.757616 -3.200589
2 -3.200589  2.757616

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] 53.43509 53.43509
 (between_SS / total_SS =  90.9 %)

Available components:

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

The `kmeans()` function returns a list of 9 components. We can see the attributes from `attributes()` function!

```
#Look at the attributes of the Km:
attributes(km)
```

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

$class
[1] "kmeans"
```

Q1. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

Q2. Cluster assignment/mamber 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
```

```
#cluster: It's telling us which pints belong to which clusters.
```

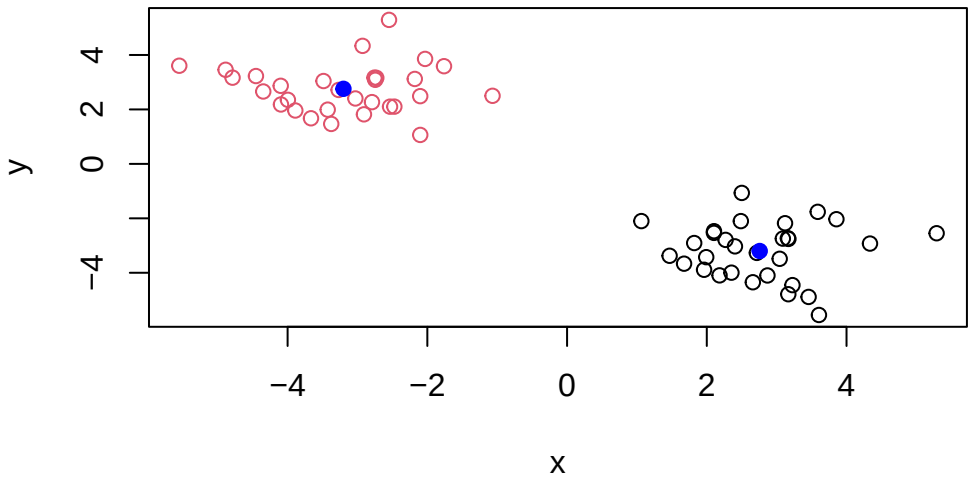
Q3. Cluster Centers?

```
km$centers
```

```
          x         y
1  2.757616 -3.200589
2 -3.200589  2.757616
```

Q4. Make a plot of our `kmeans()` results, with cluster assignment different colors, and centers blue.

```
plot(x, col=km$cluster) + points(km$centers, col="blue", pch=19)
```



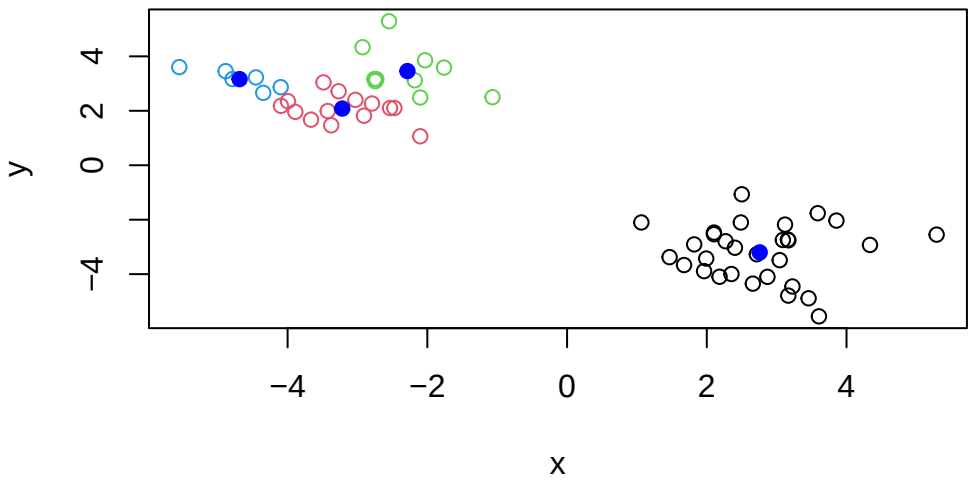

```
integer(0)
```

Q5. Run `k-means` again on `x`, but with 4 groups cluster, and plot the same result fig as above.

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

```
K-means clustering with 4 clusters of sizes 30, 14, 10, 6

Cluster means:
          x         y
1  2.757616 -3.200589
2 -3.217223  2.082186
3 -2.285045  3.459460
4 -4.687682  3.163877

Clustering vector:
 [1] 2 3 3 2 2 3 2 3 4 2 2 3 2 2 3 3 2 3 4 4 2 4 3 2 2 3 4 2 2 4 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] 53.435090  7.974641  9.564389  1.938290
 (between_SS / total_SS =  93.8 %)

Available components:

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

```r
#Plotting km4
plot(x, col= km4$cluster) + points(km4$centers, col="blue", pch=19)
```




```
integer(0)
```

Key point - **BE WARY**:
Kmeans is super popular because it's easy to understand, but it can be **self-fulfilling and MISUSED**. One big limitation is: it can impose a clustering pattern even if natural grouping doesn't exist.
We can just cluster anything into what we think it is, when we determine the `centers`. Say, even though it's only 2 clusters, we put in 4 and it still gives out a result.

Hierarchical Clustering

Main function in base R is `hclust()`.

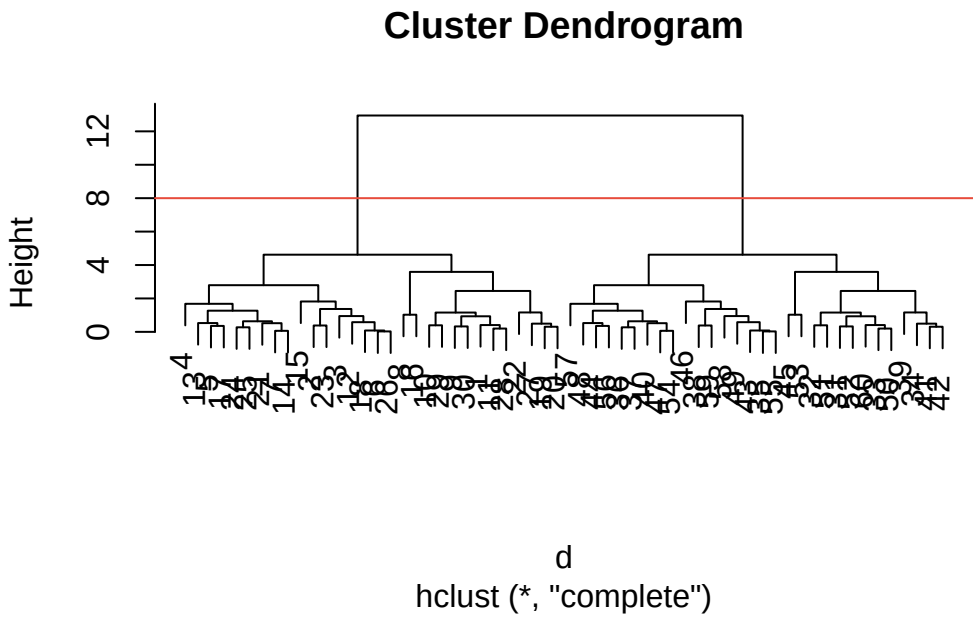
You can't just pass the dataset as is into `hclust()`. We have to make a *distance matrix* (dissimilarity distance) first. But this makes it more flexible (doesn't have to be Euclidean distances only like the `kmeans()`). Flexible as in we can do sequence alignments too!

```
d <- dist(x)
hc <- hclust(d)
hc #not very useful without plotting it.
```

```
Call:
hclust(d = d)

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

The results of `hclust()` doesn't have very useful `print` method, but it has special `plot()` method. Will give out a "dendrogram" or a "tree diagram".

```
plot(hc) +   #each labels here #each labels here is just the data label.
  abline(h=8, col = "#E74C3C")
```




```
integer(0)
```

`hclust()` is a bottom-up clustering method.

To get our main cluster assignment (membership vector), we need to `cut` our tree.

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

```
 [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
```

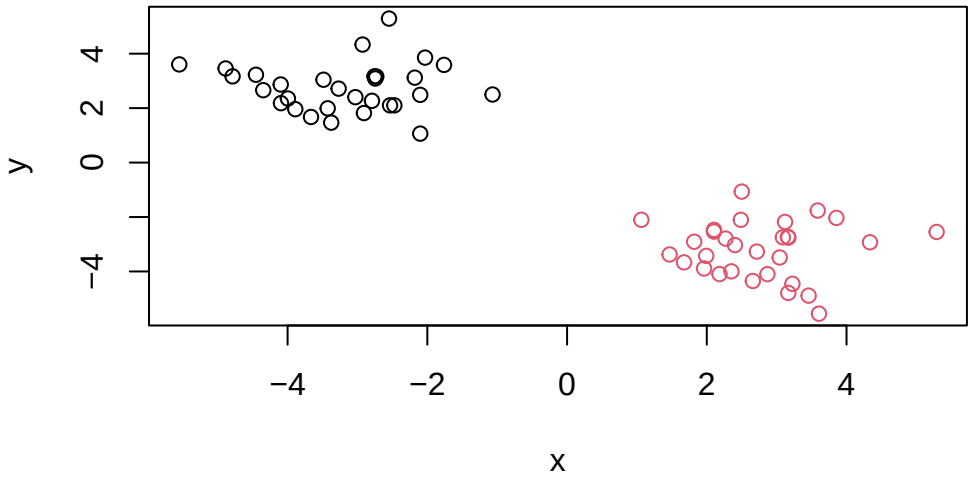
We can see the attributes of `groups` by using `table()`

```
table(groups)
```

```
groups
 1  2
30 30
```

Plotting the same one (Q4) where we determine the colors of the clusters:

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



Hierarchical clusterng is distinct, in that the dendogram (tree figure) can reveal the *potential groupings* in our data, unlike K-means.

Dimentional Reduction

PCA (Principal Component Analysis)

PC is a common and useful dimentionality reduction technique used in many fields, particularly Bioinformatics. It basically lines that are of *best fit* for our data. So these PC lines are better at representind the data points compared to any of the original axes.

PC's capture the "spread" of the data. The PC1 axis will capture the most variation, followed by the PC2, etc.

Objectives of PC:

- Reduce dimentionality
- Choose most useful characters

Basically like a filter!

Analyzing the UK food data:

Importing the File:

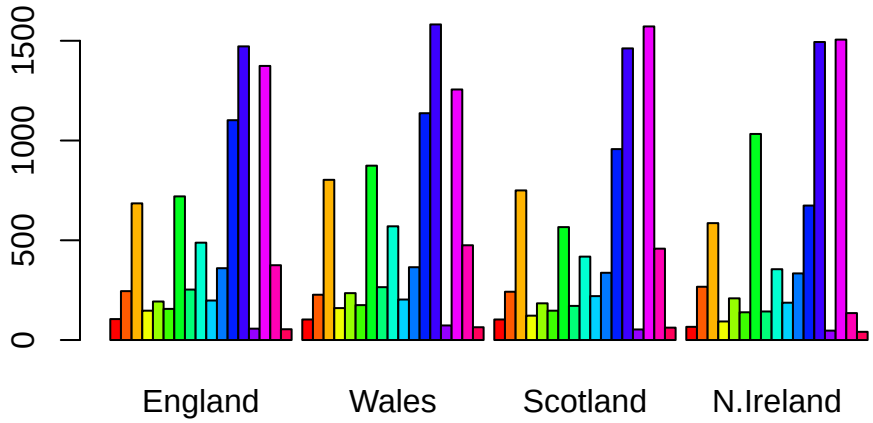
```
url <- "https://tinyurl.com/UK-foods"
uk <- read.csv(url, row.names = 1) #this row.names is a function that sets the first column i
head(uk, 6)
```

```
               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
```

Now, let's try plotting them.

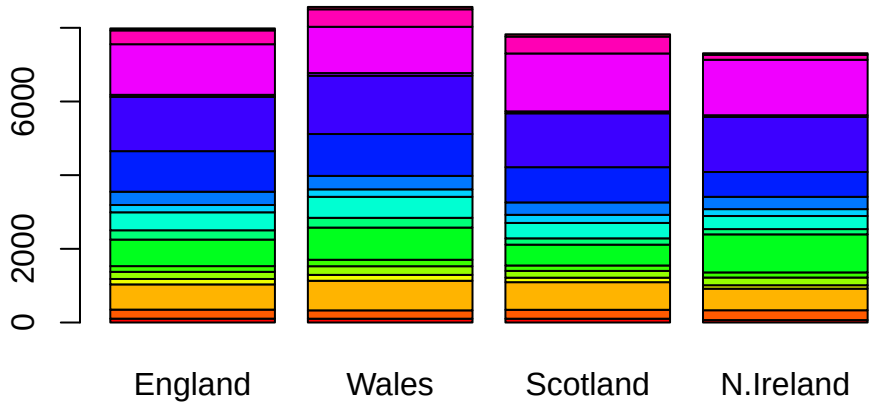
Barplot

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



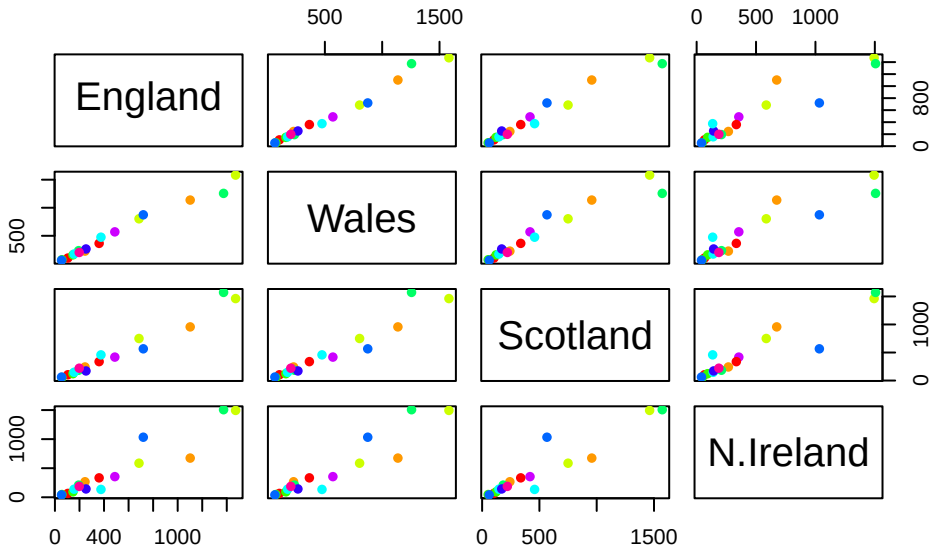
Stacked barplot

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



Pairs Plot
Now a more useful plot!

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



PCA to the Rescue!

The main function in R for PCA: `prcomp()`

```
t(uk) #transposing the data, so we have the columns as the food type.
```

```
          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 <- prcomp(t(uk))
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
```

PC1 is the one that will catch most variation. This makes sense, since **67%** of Proportion of Variance is captured by PC1, and *29%* is captured by the PC2.

The `prcomp()` function returns a list of object of our results with 5 attributes/components.

```
attributes (pca)
```

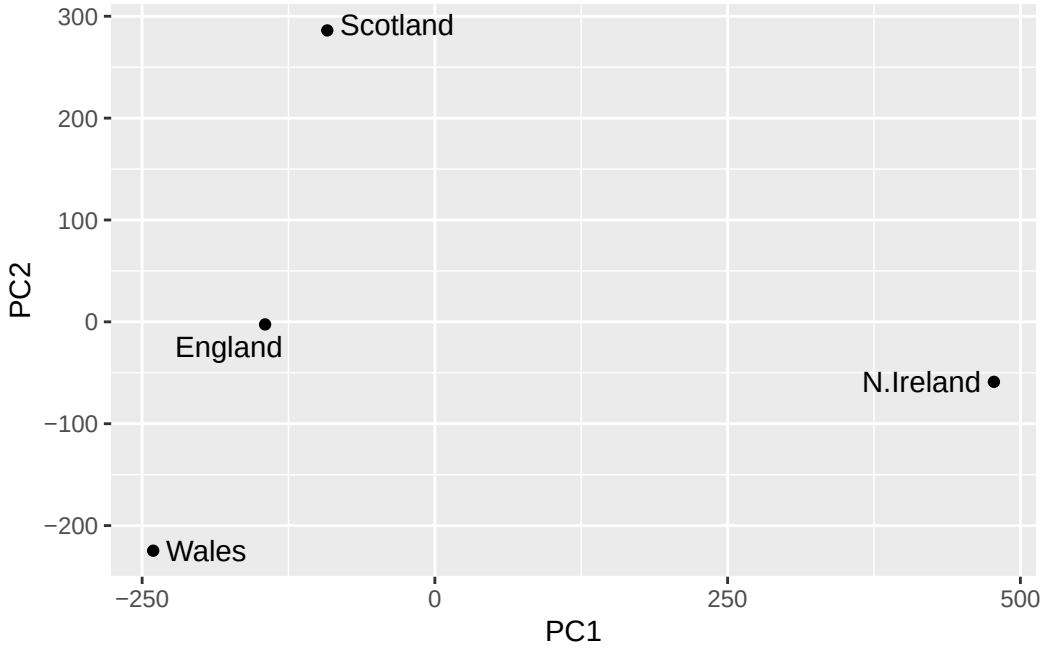
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

The two main results in here are: `pca$x` and `pca$rotation`. The `pca$x` contains scores of fata on the new PC axis – we use these to make the "PCA plot"

```
library(ggplot2)
library(ggrepel)

#make a plot of pca$x of PC1 v PC2
ggplot(pca$x)+
  aes (PC1, PC2, label= rownames(pca$x))+
  geom_point()+
  geom_text_repel()
```



This plot mainly shows that Ireland consumes quite different foods than England, Wales, and Scotland. The PC1 shows highest variation, while the PC2 shows the second highest.

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

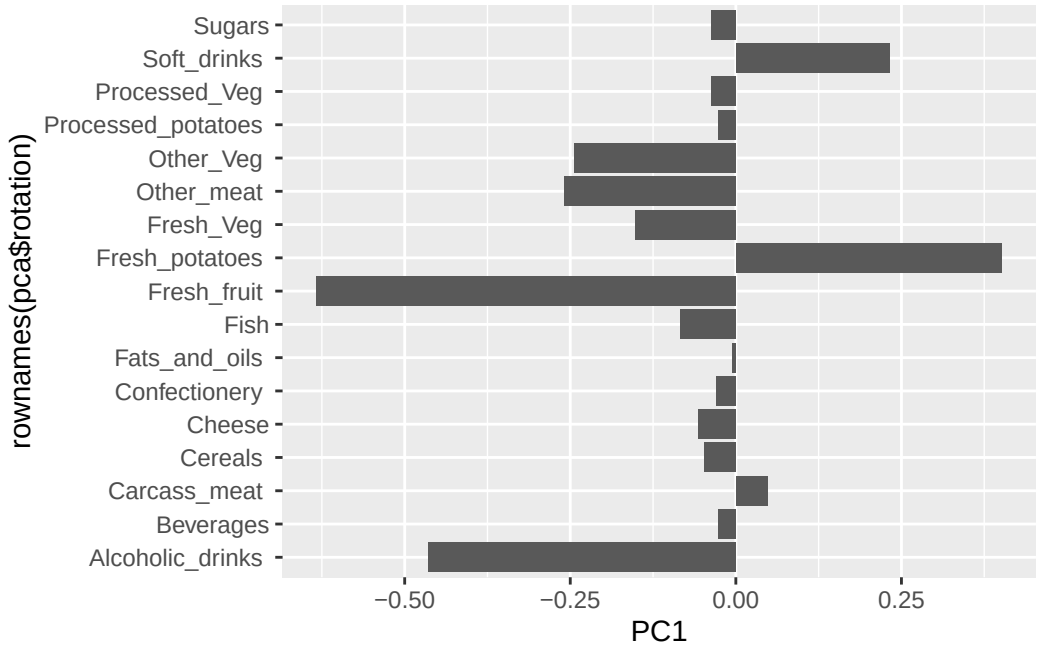
```
head(pca$rotation, 4)
```

```
                    PC1         PC2         PC3          PC4
Cheese      -0.05695538  0.01601285  0.02394295 -0.694538519
```

```
Carcass_meat   0.04792763  0.01391582  0.06367111  0.489884628
Other_meat    -0.25891666 -0.01533114 -0.55384854  0.279023718
Fish          -0.08441498 -0.05075495  0.03906481 -0.008483145
```

Each factor (the food) contribution to the PC1 (the new axis!):

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



This second plot shows how much each of the food types contribute to the PC; how much they affect the variance!

UNDERSTANDING THE TWO PLOTS:

Combined with the previous plot (L: Ireland, R: England, Wales, and Scots), the two shows that Ireland eats more fresh potatoes and soft drinks, but the Fresh fruit adn alcoholic drinks are less consumed there.