Class 7: Machine Learning 1

Allen (A16897142)

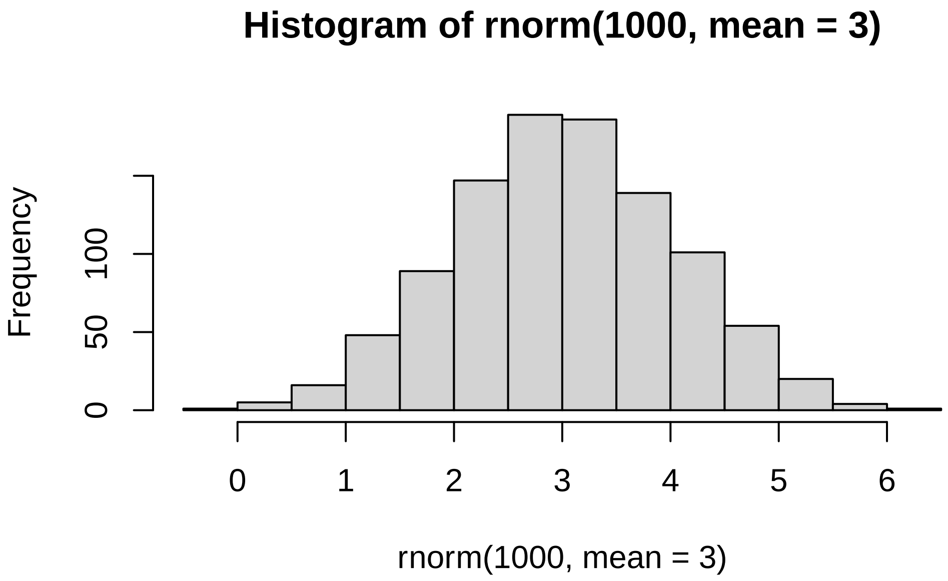
Table of contents

We will be exploring unsupervised machine learning methods. The first ones are clustering and dimensionality reduction.

Clustering

Let's make up some data to cluster where we know what the answer will be. The `rnorm()` function will be able to help us.

```r
hist(rnorm(1000,mean=3))
```



Now we want to return 30 numbers centered on -3

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

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

x
```

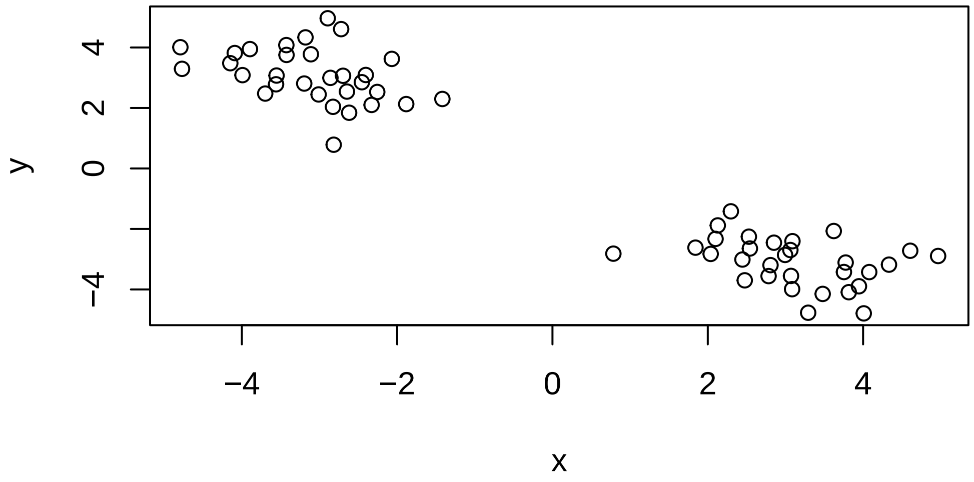
```
               x         y
 [1,] -2.7211154 4.6073881
 [2,] -2.2561009 2.5291685
 [3,] -4.1488060 3.4799248
 [4,] -2.6466059 2.5423427
 [5,] -2.8172904 0.7845324
 [6,] -4.0912523 3.8160377
 [7,] -2.8593696 2.9955223
 [8,] -2.3291621 2.1004532
 [9,] -2.6968424 3.0637487
[10,] -3.5589850 2.7832871
[11,] -3.1108965 3.7758852
[12,] -3.1803577 4.3338815
[13,] -2.4546093 2.8520330
```

```
[14,] -3.4248771  3.7533475
[15,] -2.6182381  1.8417334
[16,] -2.4041042  3.0908535
[17,] -3.9916901  3.0862243
[18,] -4.7702208  3.2931729
[19,] -1.8830801  2.1288483
[20,] -3.8954701  3.9470548
[21,] -3.0113889  2.4465219
[22,] -3.6988116  2.4763069
[23,] -2.8942964  4.9663322
[24,] -2.8261994  2.0377633
[25,] -2.0688641  3.6225027
[26,] -3.1965448  2.8077999
[27,] -3.4266845  4.0793622
[28,] -4.7915374  4.0087171
[29,] -1.4176593  2.2974603
[30,] -3.5531841  3.0718452
[31,]  3.0718452 -3.5531841
[32,]  2.2974603 -1.4176593
[33,]  4.0087171 -4.7915374
[34,]  4.0793622 -3.4266845
[35,]  2.8077999 -3.1965448
[36,]  3.6225027 -2.0688641
[37,]  2.0377633 -2.8261994
[38,]  4.9663322 -2.8942964
[39,]  2.4763069 -3.6988116
[40,]  2.4465219 -3.0113889
[41,]  3.9470548 -3.8954701
[42,]  2.1288483 -1.8830801
[43,]  3.2931729 -4.7702208
[44,]  3.0862243 -3.9916901
[45,]  3.0908535 -2.4041042
[46,]  1.8417334 -2.6182381
[47,]  3.7533475 -3.4248771
[48,]  2.8520330 -2.4546093
[49,]  4.3338815 -3.1803577
[50,]  3.7758852 -3.1108965
[51,]  2.7832871 -3.5589850
[52,]  3.0637487 -2.6968424
[53,]  2.1004532 -2.3291621
[54,]  2.9955223 -2.8593696
[55,]  3.8160377 -4.0912523
[56,]  0.7845324 -2.8172904
```

```
[57,]  2.5423427 -2.6466059
[58,]  3.4799248 -4.1488060
[59,]  2.5291685 -2.2561009
[60,]  4.6073881 -2.7211154
```

Now plot `x`

```
plot(x)
```



K-means

Base R's main function for K-means clustering is called `kmeans()`:

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

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

Cluster means:
          x         y
1 -3.091475  3.087335
```

```
2  3.087335 -3.091475

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 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

Within cluster sum of squares by cluster:
[1] 42.74942 42.74942
 (between_SS / total_SS =  93.1 %)

Available components:

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

The`kmeans()`function is now able to return a list with 9 components and you can see the named componenets of any list with `attribuets()` function.

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

How many points are in each cluster?

```r
km$size
```

```
[1] 30 30
```

Cluster Membership/Assignment:

```r
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 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
```

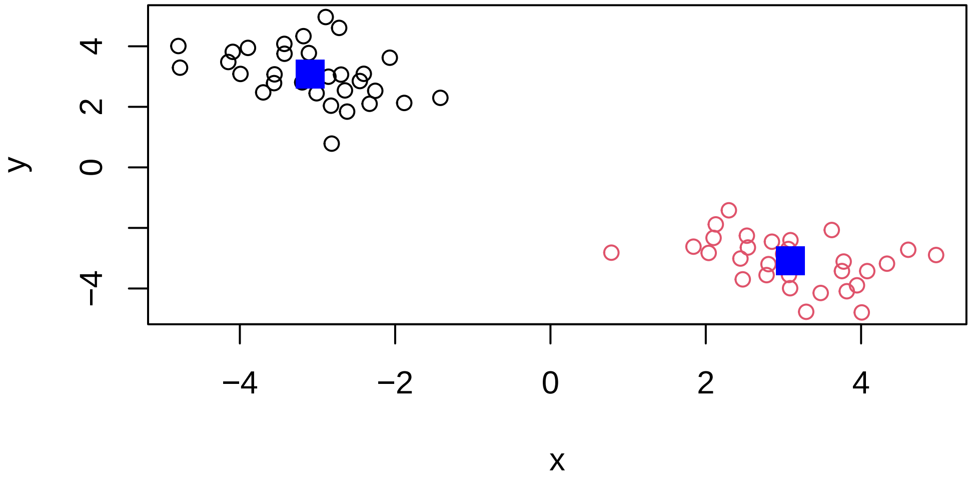
Cluster center:

```
km$centers
```

```
          x         y
1 -3.091475  3.087335
2  3.087335 -3.091475
```

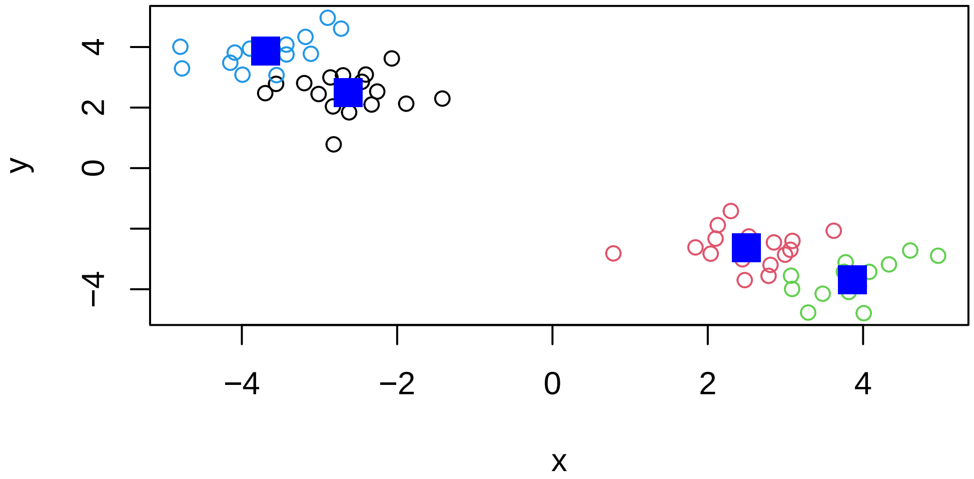
Make a plot of `kmeans()` results showing cluster assignment using different colors for each group or points and cluster centers in blue.

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



Run `kmeans()` again on `x` and this cluster into 4 groups/clusters and plot the same resulting figure as above:

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



key-point: K-means clustering is super popular but can easily be misused. A limitation is that it can force a clustering pattern even if data shows an otherwise natural grouping that does not exist in terms of `centers`.

Hierarchal Clustering

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

Note: You can not just pass a data set as is into `hclust()`. You need to give a distance matrix.

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

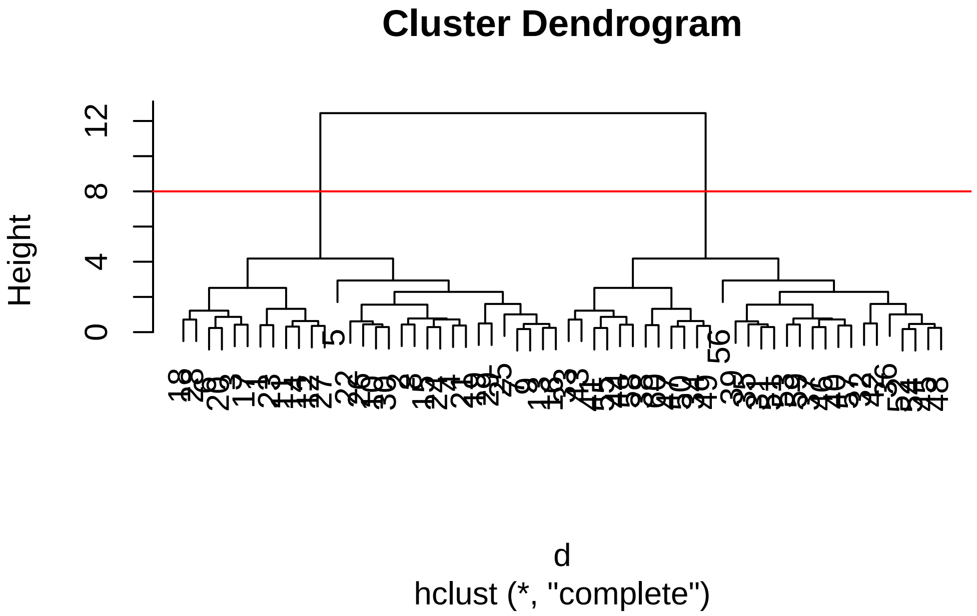
```

Call:
hclust(d = d)

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

The results of `hclust()` are not very useful typically. And there is no useful `print()` method. However, there is a special `plot()`.

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



To get our cluster assignment aka membership vector, you will need to cut the tree at the goal posts in different areas.

```r
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 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
```

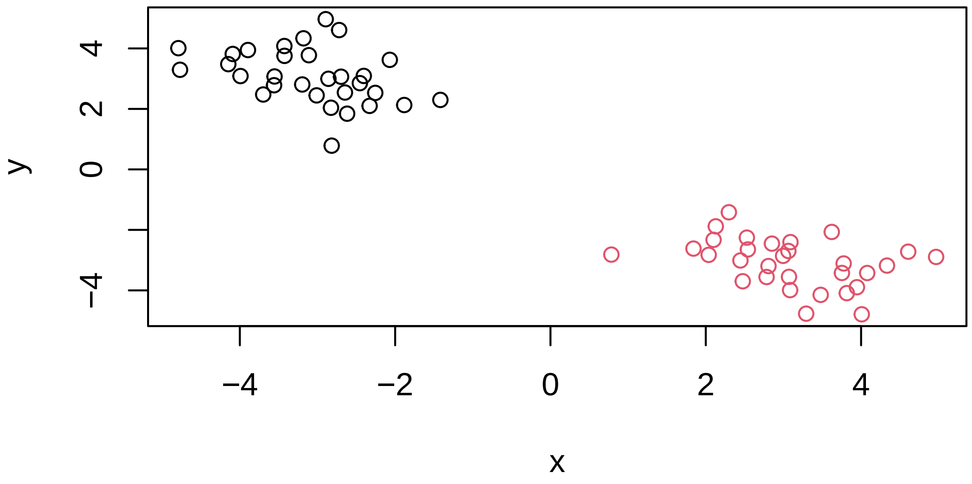
```r
table(grps)
```

```
grps
 1  2 
30 30 
```

```r
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
```

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



Hierarchical Clustering is distinct as the dendrogram can reveal groups in your data that K-means clustering can not accomplish.

Principal Component Analysis (PCA)

PCA is used as a dimensional reduction technique and to find which dimension is the primary dimenstion in the data.

Data from the UK on food consumption will be used.

```r
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
```

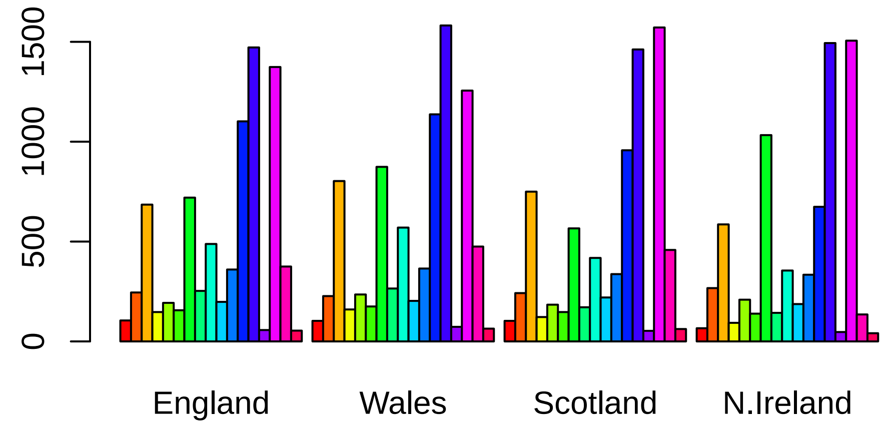
```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
               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
```

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
head(x)
```

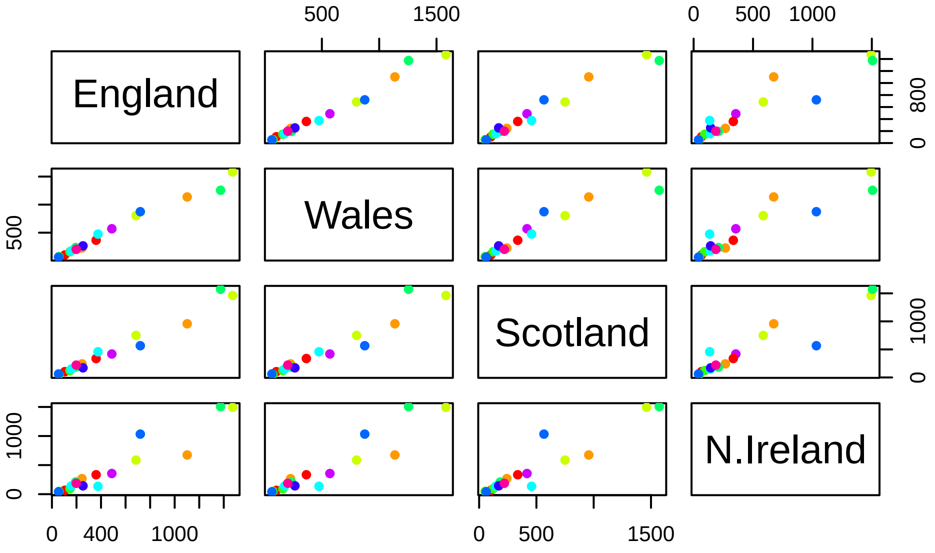
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

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



A "paris" plot can be useful as it compares two countries. Wherever the country is on axis wise is where it is on the pairs plot.

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



PCA to the rescue!

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

```
## the PCA code
pca <- prcomp(t(x))
##overview of 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 with our results.

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

The main results that we are looking for are `pca$x` and `pca$rotation`. `pcz$x` contains the scores of data on the PC axis we use the make our PCA plot with.

```
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()
```

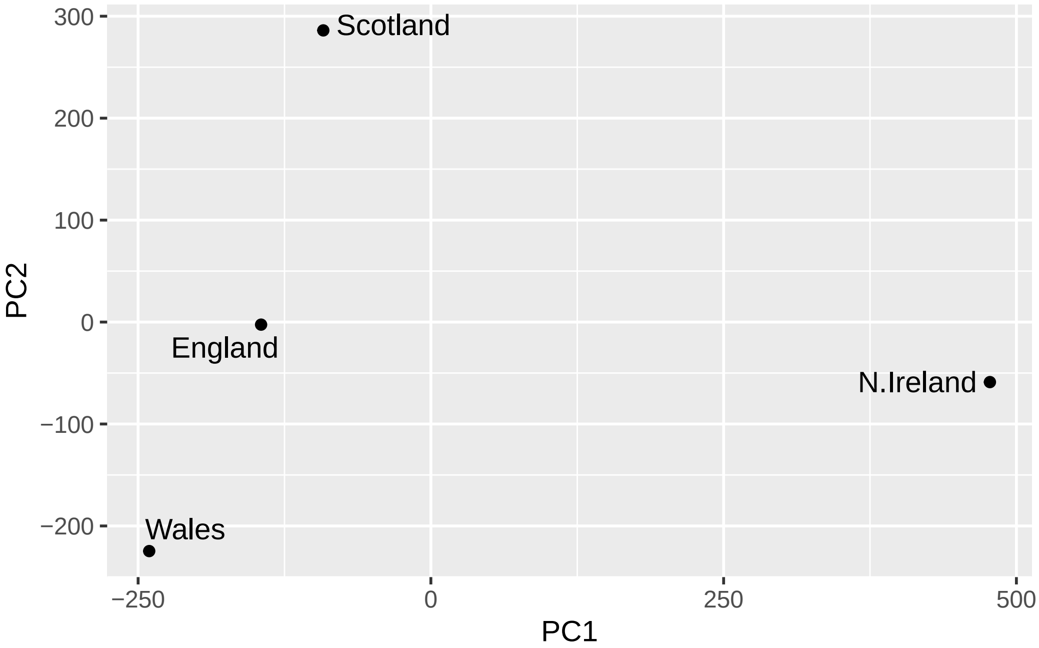


Figure 1: Plot demonstrating different countries on their average food group consumption aligned on PC1 axis vs PC2 axis using Principal Component Analysis.

`pca$rotation` contains our second major result. To see what PCA is picking up:

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

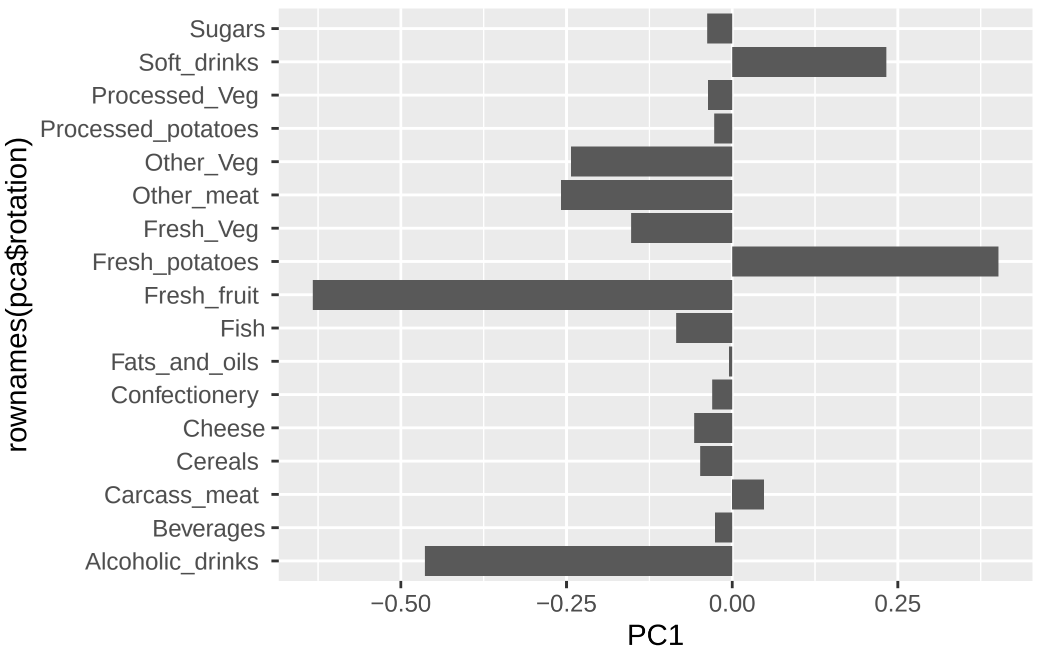


Figure 2: Barplot indicating which foods explains the trend on the PC plot. If the bar is negative, that means it is more likely explained by a negative country in the PC plot. If the bar is positive, that means it is more likely explaied by a positive country in the PC plot.